



CSE319

Machine Learning Module 1

CONTENTS

- Introduction to Machine learning
- Types of Machine Learning
- Models selection and generalization
- Machine Learning concept work flow
- Applications

Introduction to Machine learning

What is Learning?

- Herbert Simon: “Learning is any process by which a system improves performance from experience.”
- What is the task?
 - Classification
 - Categorization/clustering
 - Problem solving / planning / control
 - Prediction

Why “Learn” ?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- Learning is used when:
 - Human expertise does not exist (navigating on Mars),
 - Humans are unable to explain their expertise (speech recognition)
 - Solution changes in time (routing on a computer network)
 - Solution needs to be adapted to particular cases (user biometrics)

What We Talk About When We Talk About “Learning”

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:
People who bought “Blink” also bought “Outliers” (www.amazon.com)
- Build a model that is *a good and useful approximation* to the data.

Data Mining

- Retail: Market basket analysis, Customer relationship management (CRM)
- Finance: Credit scoring, fraud detection
- Manufacturing: Control, robotics, troubleshooting
- Medicine: Medical diagnosis
- Telecommunications: Spam filters, intrusion detection
- Bioinformatics: Motifs, alignment
- Web mining: Search engines
- ...

What is Machine Learning?

- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
 - Solve the optimization problem
 - Representing and evaluating the model for inference

It provides us statistical tools to explore and analyses the data

Applications

- Association
- Supervised Learning
 - Classification
 - Regression
- Unsupervised Learning
- Reinforcement Learning

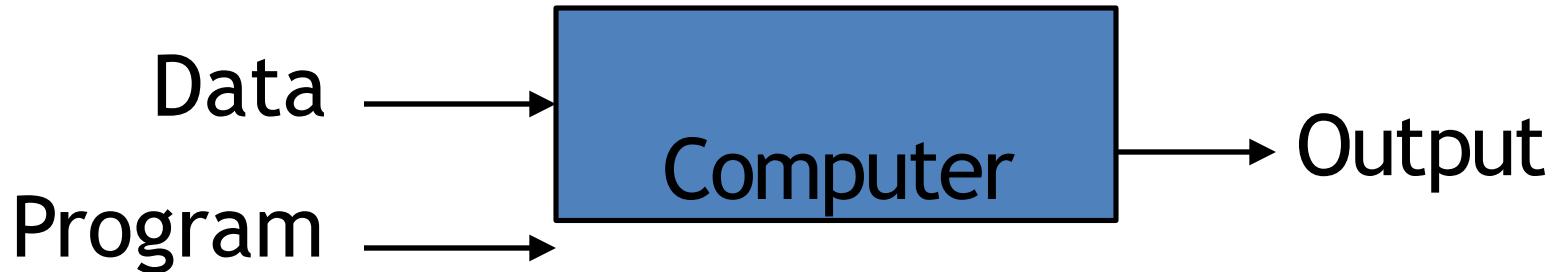
Why Study Machine Learning? Developing Better Computing Systems

- Develop systems that are too difficult/expensive to construct manually because they require specific detailed skills or knowledge tuned to a specific task (***knowledge engineering bottleneck***).
- Develop systems that can automatically adapt and customize themselves to individual users.
 - Personalized news or mail filter
 - Personalized tutoring
- Discover new knowledge from large databases (***data mining***).
 - Market basket analysis (e.g. diapers and beer)
 - Medical text mining (e.g. migraines to calcium channel blockers to magnesium)

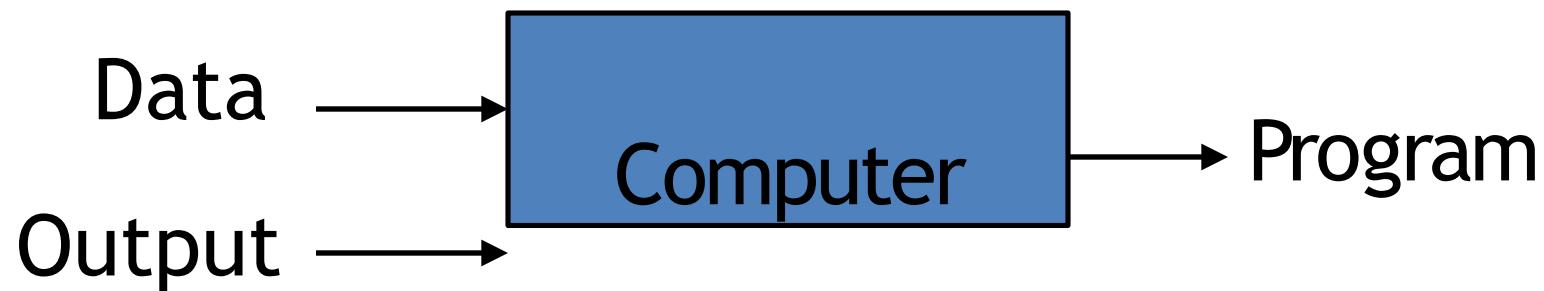
Related Disciplines

- Artificial Intelligence
- Data Mining
- Probability and Statistics
- Information theory
- Numerical optimization
- Computational complexity theory
- Control theory (adaptive)
- Psychology (developmental, cognitive)
- Neurobiology and many more

Traditional Programming



Machine Learning



Human Learning

- It is a process of gaining information through observation, to solve with real world scenarios.
- Human learning happens in one of the three ways:
 - An expert in the subject directly teaches us
 - We build our own notion indirectly based on what we have learnt in the past
 - We learn ourselves, maybe after multiple attempts.

Types of Human Learning

- Learning under expert guidance
- Learning guided by knowledge gained from experts
- Learning by self

MACHINE LEARNING

A computer program is said to learn from experience ‘E’ with respect to some class of tasks ‘T’ and performance measure ‘P’, if its performance at tasks in ‘T’, as measured by ‘P’, improves with experience ‘E’.

- *Tom M. Mitchell*

History of Machine Learning

- 1950s
 - Samuel's checker player
- 1960s:
 - Neural networks: Perceptron
 - Pattern recognition
 - Learning in the limit theory
 - Minsky and Papert prove limitations of Perceptron
- 1970s:
 - Symbolic concept induction
 - Winston's arch learner
 - Expert systems and the knowledge acquisition bottleneck
 - Quinlan's ID3
 - Michalski's AQ and soybean diagnosis

History of Machine Learning (cont.)

- 1980s:
 - Advanced decision tree and rule learning
 - Explanation-based Learning (EBL)
 - Learning and planning and problem solving
 - Utility problem, Analogy
 - Cognitive architectures
 - Resurgence of neural networks (connectionism, backpropagation)
- 1990s
 - Data mining
 - Adaptive software agents and web applications
 - Text learning
 - Reinforcement learning (RL)
 - Inductive Logic Programming (ILP)
 - Ensembles: Bagging, Boosting, and Stacking
 - Bayes Net learning

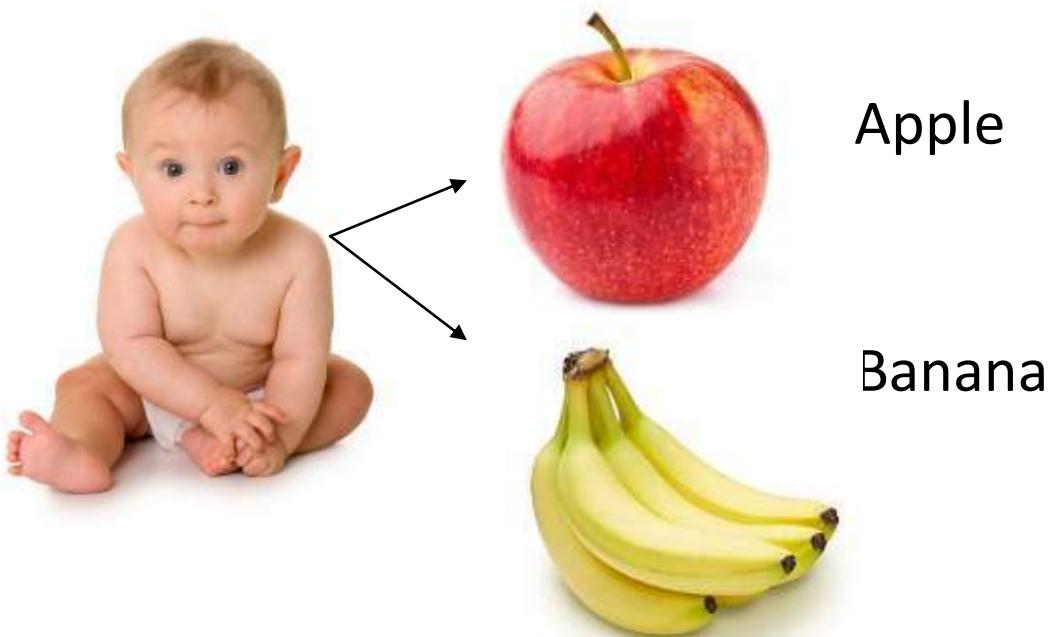
TYPES of MACHINE LEARNING

Types of Machine Learning

- **Supervised learning:** A machine predicts the class of unknown objects based on prior class-related information of similar objects. Also called predictive learning.
- **Unsupervised/clustering learning:** A machine finds patterns in unknown objects by grouping similar objects together. Also called descriptive learning.
- **Reinforcement learning:** A machine learns to act on its own to achieve the given goals.

SUPERVISED LEARNING

- The major motivation of supervised learning is to learn from past information.
- But how do machine learns?
 - By TRAINING DATA using labels.



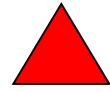
Apple

Banana

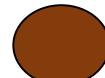
What's this?



Training Data



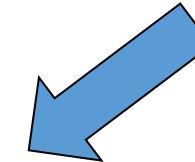
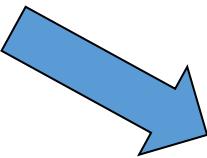
Triangle



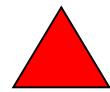
Circle



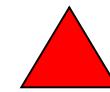
Rectangle



Testing Data



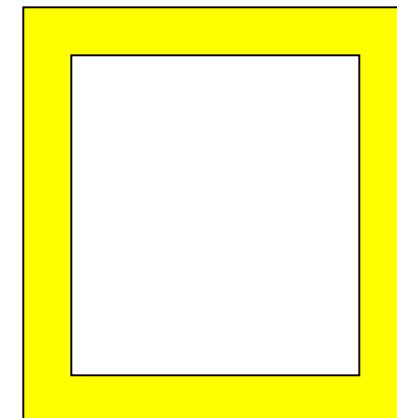
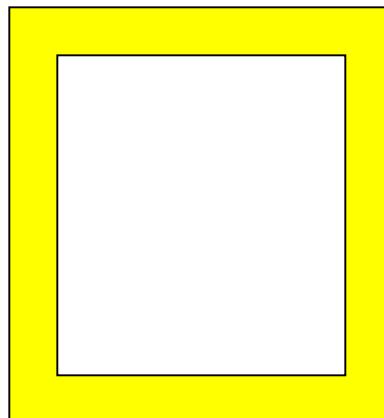
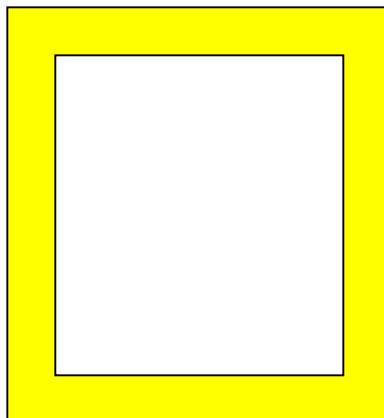
Triangle



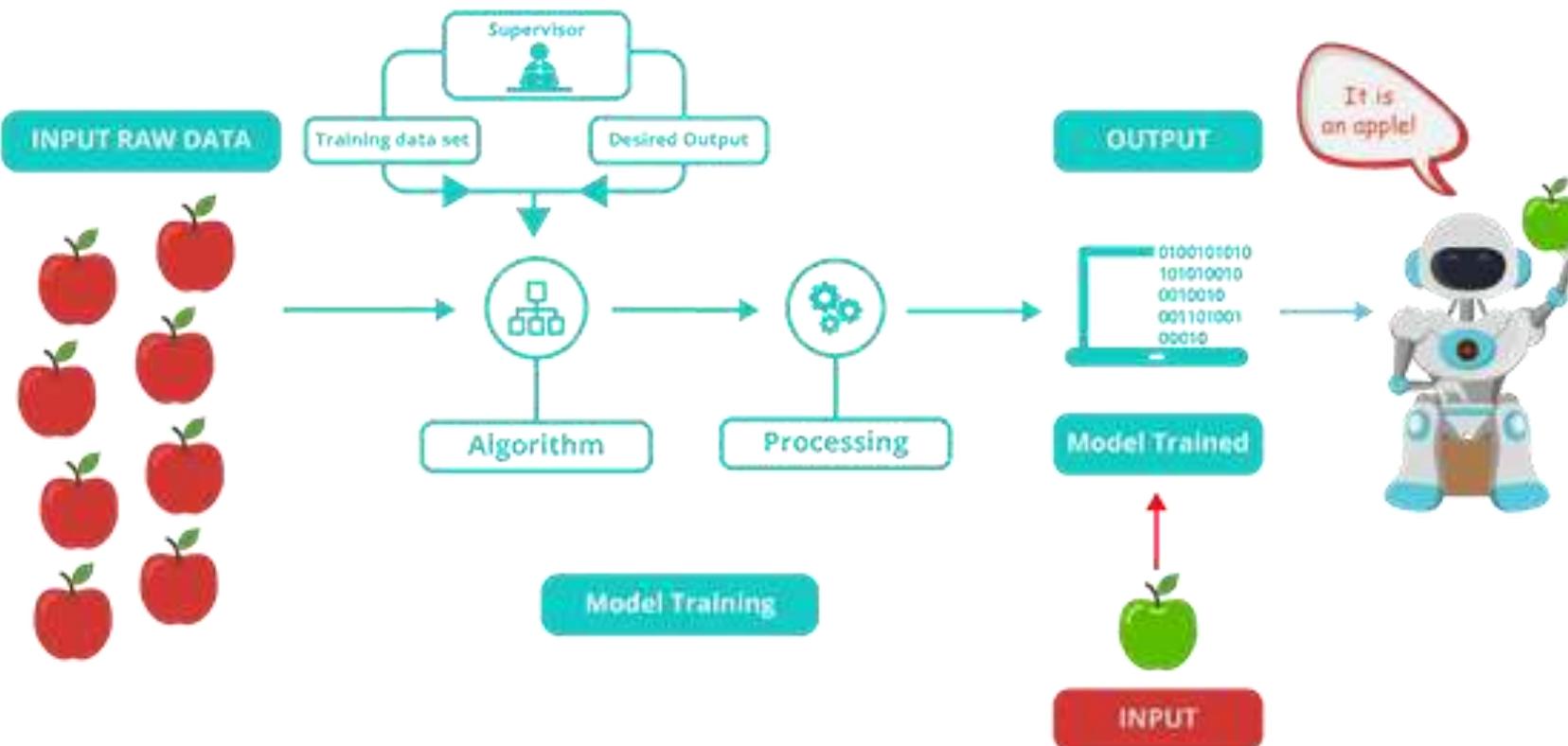
Circle



Rectangle



Supervised Learning



Examples of Supervised Learning

- Predicting the results of a game.
- Predicting a tumour is malignant or benign.
- Predicting the piece of domains like real estate, stocks, etc.
- Classify texts such as classifying a set of emails as spam or non-spam.

Classification & Regression

- When we are trying to predict a categorical or nominal variable, the problem is known as a **classification** problem.
 - Ex: Identify Cat or Dog
- When we are trying to predict a real-valued variable, the problem falls under the category of **regression**.
 - Ex: Predict the value of a property.

Unsupervised Learning

- Unsupervised learning(USL) is a type of self-organized learning that helps find previously unknown patterns in data set without pre-existing labels.
- The objective is to take a dataset as input and try to find natural grouping or patterns within the data elements or records.
- Hence, USL is termed as Descriptive Model and the process of USL is called Pattern or Knowledge Discovery.
- In unsupervised learning, the system is presented with **unlabeled, uncategorized** data and the system's algorithms act on the data without prior training. The output is dependent upon the coded algorithms.

Unsupervised Learning

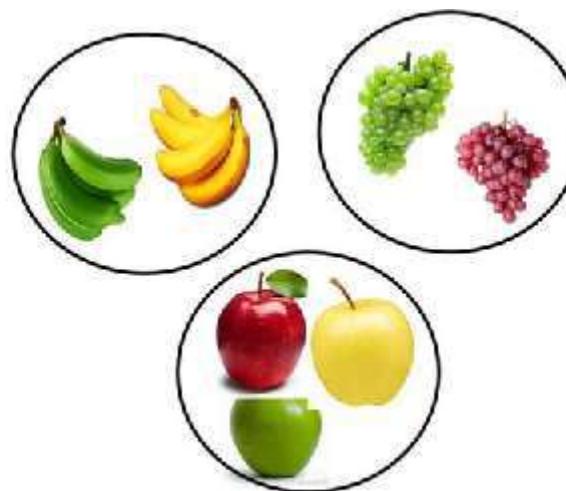
- Clustering is the main type of Unsupervised Learning.
- Clustering groups similar objects together.

Input Data



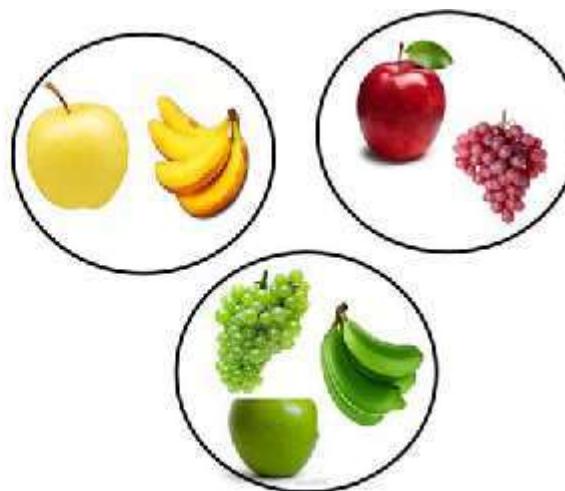
(a)

Cluster by type



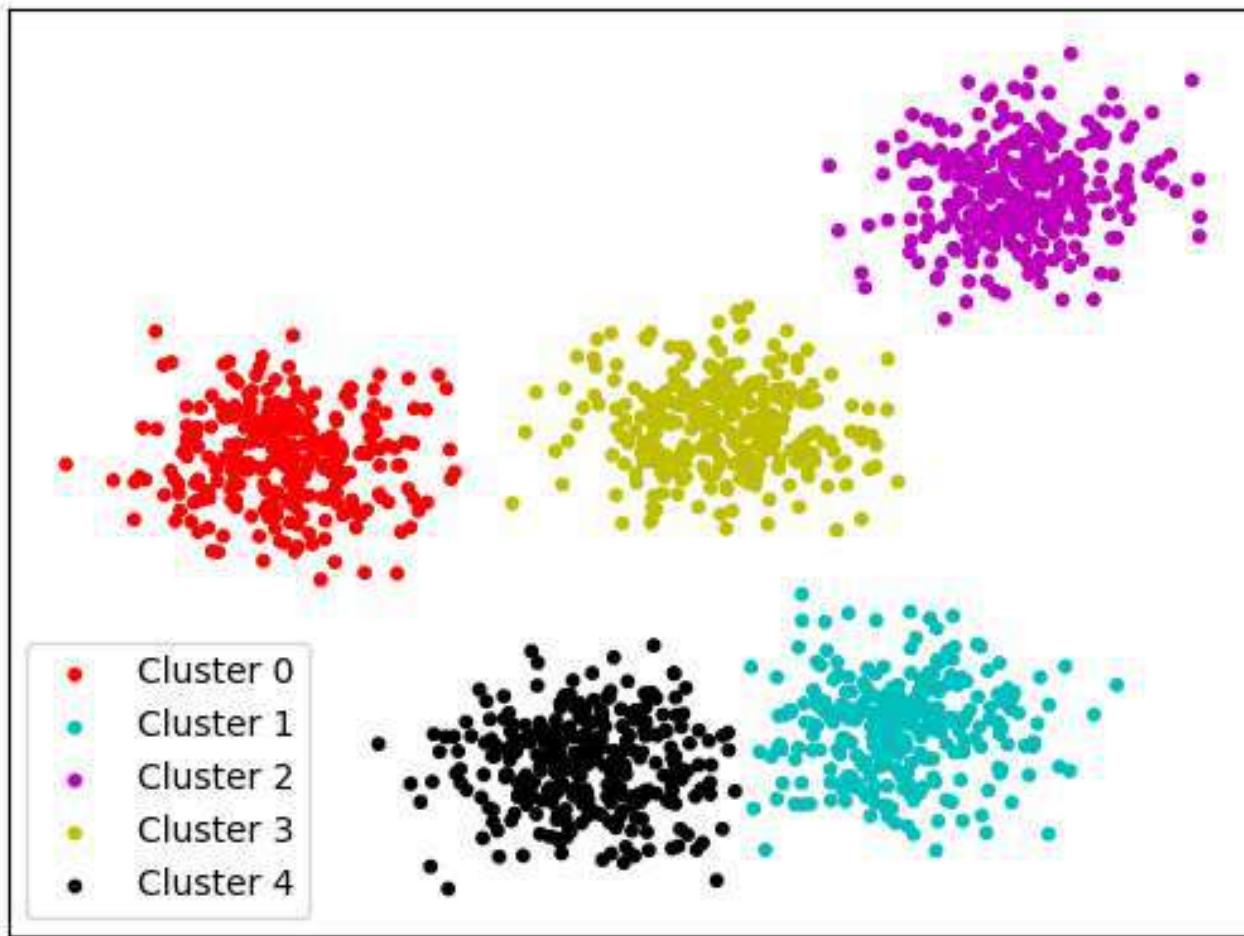
(b)

Cluster by color

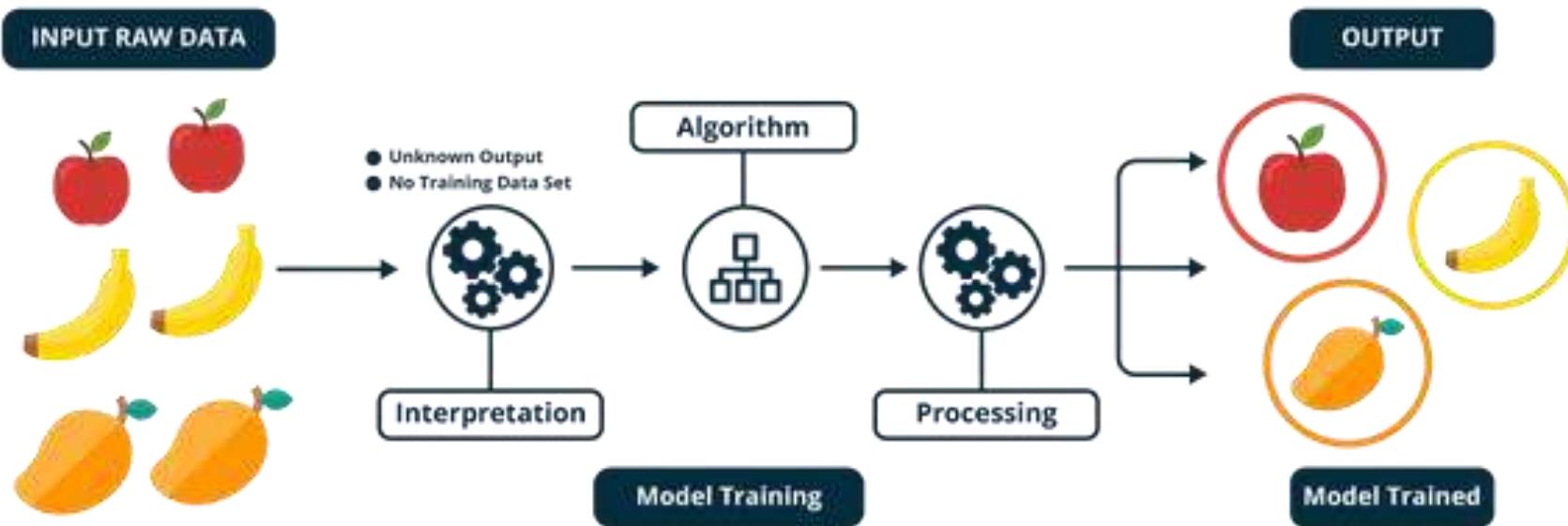


(c)

Sample Clustering



Unsupervised Learning



Unsupervised Learning Example: Categorize Cats and Dogs

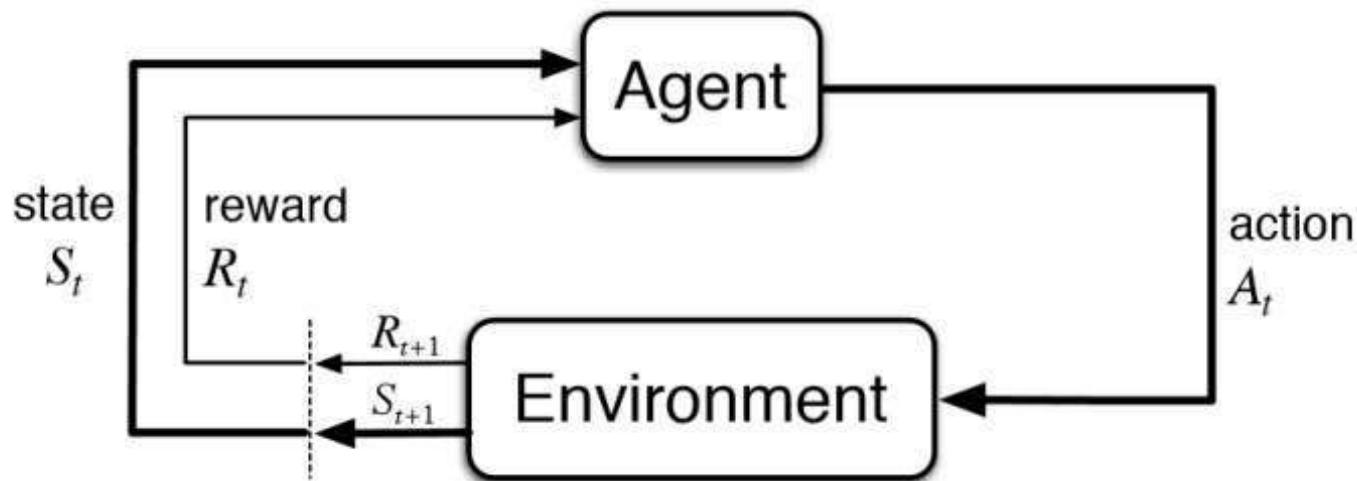


Reinforcement Learning

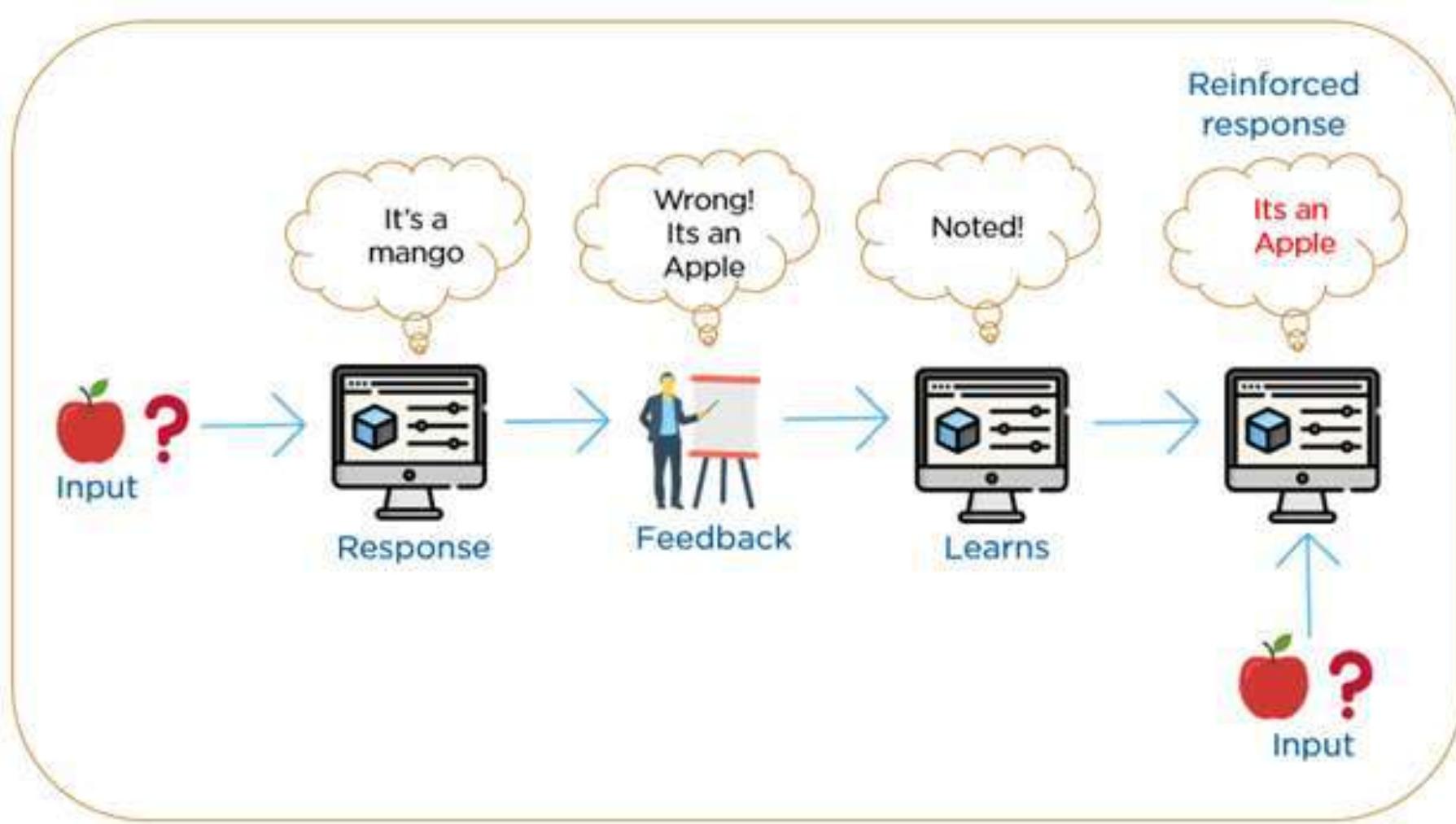
- It is about taking suitable action to maximize reward in a particular situation.
- It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation.
- Close to human learning.

Cont'd

- Algorithm learns a policy of how to act in a given environment.
- Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm.



Reinforcement Learning



Different Varieties of Machine Learning

- Concept Learning
- Clustering Algorithms
- Connectionist Algorithms
- Genetic Algorithms
- Explanation-based and Transformation-based Learning
- Reinforcement and Case-based Learning
- Macro Learning
- Evaluation Functions
- Cognitive Learning Architectures
- Constructive Induction
- Discovery Systems

Languages or Tools for Machine Learning

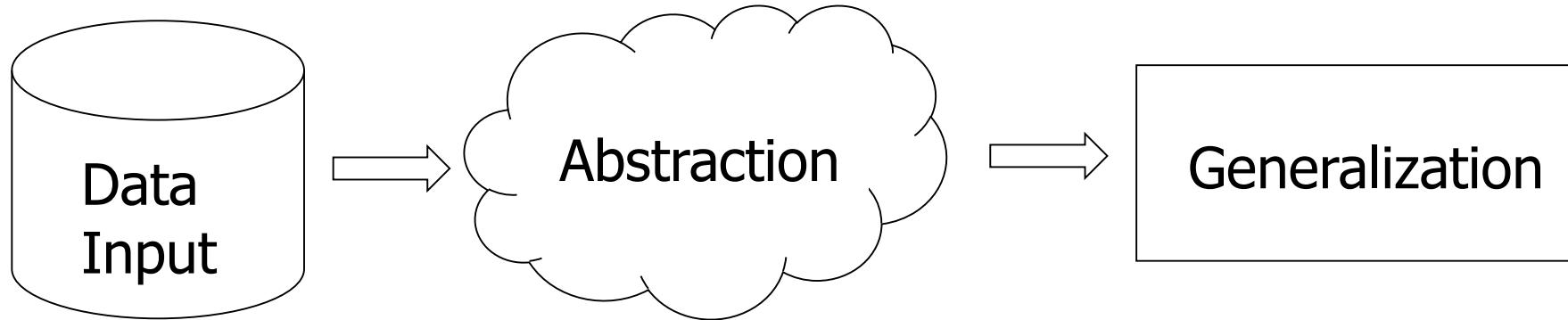
- Python – Open source programming language adopted for machine learning.
- R – Open source software. Used for statistical computing and data analysis
- Matlab - Developed by MathWorks. Licensed version. Used for variety of applications.
- SAS – Statistical Analysis System, was developed and licensed by SAS Institute provides strong support for ML.
- Others-
 - SPSS(Statistical Package for the Social Sciences) – IBM
 - Julia – MIT(Massachusetts Institute of Technology)

History of Machine Learning (cont.)

- 2000s
 - Support vector machines
 - Kernel & Graphical models
 - Statistical relational and Transfer learning
 - Sequence labeling
 - Collective classification and structured outputs
 - Computer Systems Applications
 - Compilers
 - Debugging
 - Graphics
 - Security (intrusion, virus, and worm detection)
 - E-mail management
 - Personalized assistants that learn
 - Learning in robotics and vision

MODEL SELECTION and GENERALIZATION

How do Machines Learn?



1. Data Input: Past data or information is utilized as a basis for future decision-making
2. Abstraction: The input data is represented in a broader way through the underlying algorithm.
3. Generalization: The abstracted representation is generalized to form a framework for making decisions.

Abstraction

- The data, given as input, cannot be used in the original shape and form.
- Abstraction helps in deriving a conceptual map based on the input data.
- The model may be in any one of forms:
 - Computational blocks like if/else rules
 - Mathematical equations
 - Specific data structures like trees or graphs
 - Logical groupings of similar observations

Abstraction Cont'd

- The choice of the model is human specific.
- Selection of model is based on:
 - The type of problem to be solved.
 - Nature of the input data.
 - Domain of the problem.

Generalization

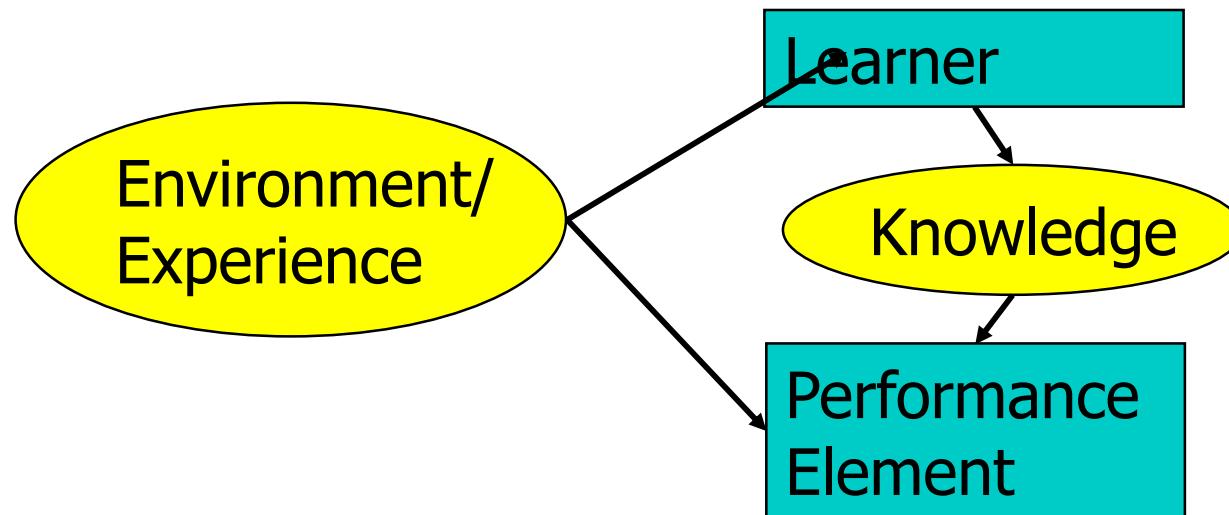
- Generalization is used for taking the decisions after training the model.
- The model is trained for a limited set of data. If we want to apply the model to take decision on a set of unknown data, we may encounter following problems:
 - The trained model is aligned with training data too much, hence may not represent the actual trend.
 - The test data possess certain characteristics apparently unknown to the training data.

Well-posed learning problem

- A framework can be designed for deciding whether a problem can be solved using ML. The framework should answer:
 - What is the problem?
 - Why does the problem need to be solved?
 - How to solve the problem?

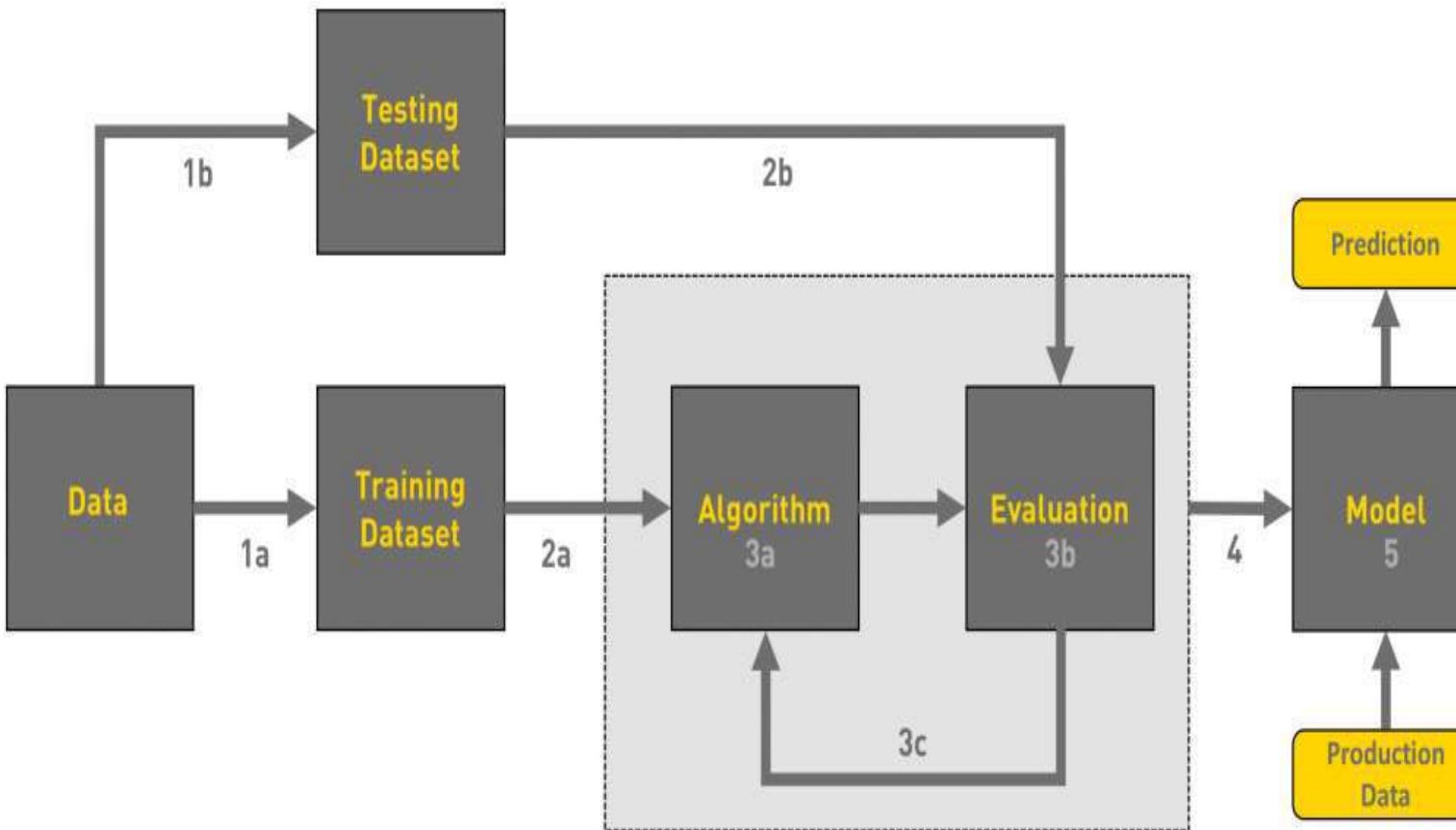
Designing a Learning System

- Choose the training experience
- Choose exactly what is to be learned, i.e. the ***target function***.
- Choose how to represent the target function.
- Choose a learning algorithm to infer the target function from the experience.



The Machine Learning Workflow

ML WorkFl ow



ML

WorkFl

OW

We can define the machine learning workflow in 3 stages.

1. Gathering data
2. Data pre-processing
3. Researching the model that will be best for the type of data
4. Training and testing the model
5. Evaluation

Gathering Data

- The process of gathering data depends on the type of project we desire to make, if we want to make an ML project that uses real-time data, then we can build an IoT system that uses different sensors data. The data set can be collected from various sources such as a file, database, sensor and many other such sources but the collected data cannot be used directly for performing the analysis process as there might be a lot of missing data, extremely large values, unorganized text data or noisy data. Therefore, to solve this problem Data Preparation is done.
- We can also use some free data sets which are present on the internet. [Kaggle](#) and [UCI Machine learning Repository](#) are the repositories that are used the most for making Machine learning models. Kaggle is one of the most visited websites that is used for practicing machine learning algorithms, they also host competitions in which people can participate and get to test their knowledge of machine learning.

Data pre-processing

- Data pre-processing is one of the most important steps in machine learning. It is the most important step that helps in building machine learning models more accurately. **In machine learning, there is an 80/20 rule. Every data scientist should spend 80% time for data pre-processing and 20% time to actually perform the analysis.**
- What is data pre-processing?
- Data pre-processing is a process of cleaning the raw data i.e. the data is collected in the real world and is converted to a clean data set. In other words, whenever the data is gathered from different sources it is collected in a raw format and this data isn't feasible for the analysis. Therefore, certain steps are executed to convert the data into a small clean data set, this part of the process is called as data pre-processing.
- Why do we need it?
- As we know that data pre-processing is a process of cleaning the raw data into clean data, so that can be used to train the model. So, we definitely need data pre-processing to achieve good results from the applied model in machine learning and deep learning projects.

Data pre-processing

- Most of the real-world data is messy, some of these types of data are:
- 1. Missing data: Missing data can be found when it is not continuously created or due to technical issues in the application (IOT system).
- 2. Noisy data: This type of data is also called outliers, this can occur due to human errors (human manually gathering the data) or some technical problem of the device at the time of collection of data.
- 3. Inconsistent data: This type of data might be collected due to human errors (mistakes with the name or values) or duplication of data.

Three Types of Data

- 1. Numeric e.g. income, age
- 2. Categorical e.g. gender, nationality
- 3. Ordinal e.g. low/medium/high

Data pre-processing

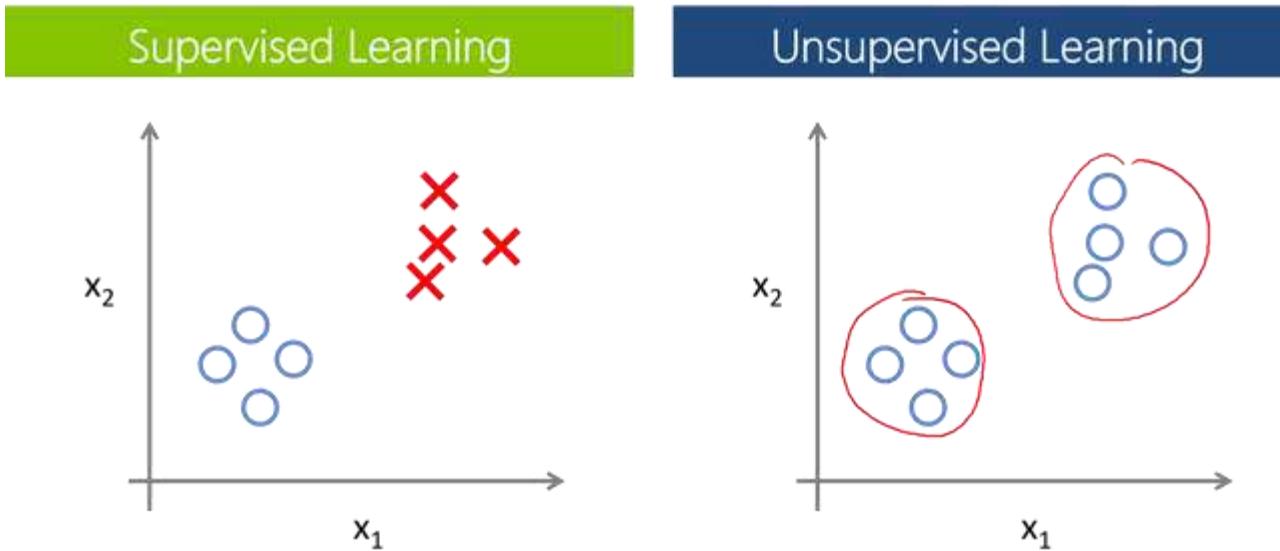
- How can data pre-processing be performed?

These are some of the basic pre — processing techniques that can be used to convert raw data.

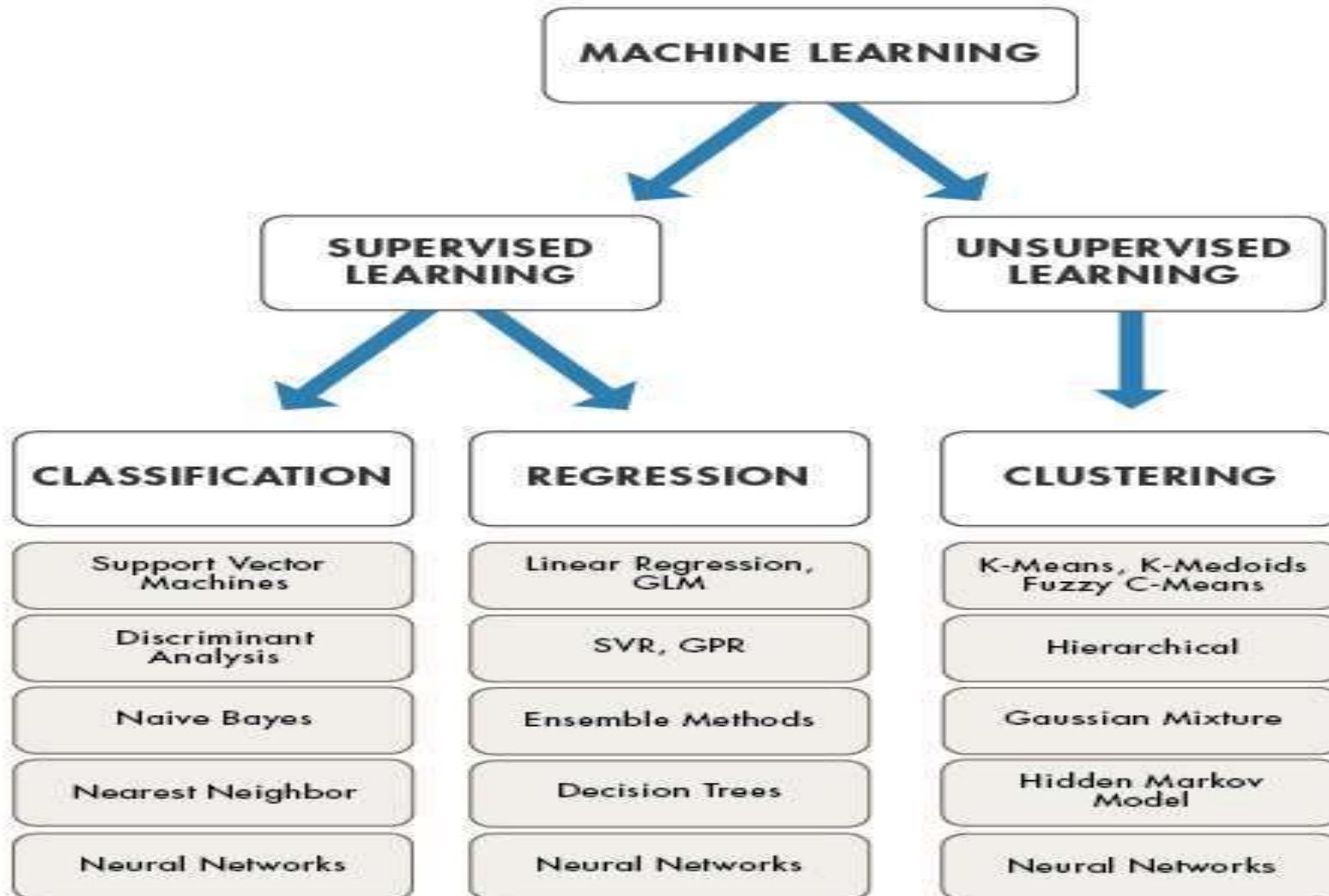
1. Conversion of data: As we know that Machine Learning models can only handle numeric features, hence categorical and ordinal data must be somehow converted into numeric features.
2. Ignoring the missing values: Whenever we encounter missing data in the data set then we can remove the row or column of data depending on our need. This method is known to be efficient but it shouldn't be performed if there are a lot of missing values in the dataset.
3. Filling the missing values: Whenever we encounter missing data in the data set then we can fill the missing data manually, most commonly the mean, median or highest frequency value is used.
4. Machine learning: If we have some missing data then we can predict what data shall be present at the empty position by using the existing data.
5. Outliers detection: There are some error data that might be present in our data set that deviates drastically from other observations in a data set. [Example: human weight = 800 Kg; due to mistyping of extra 0]

Researching the model that will be best for the type of data

- Our main goal is to train the best performing model possible, using the pre-processed data.

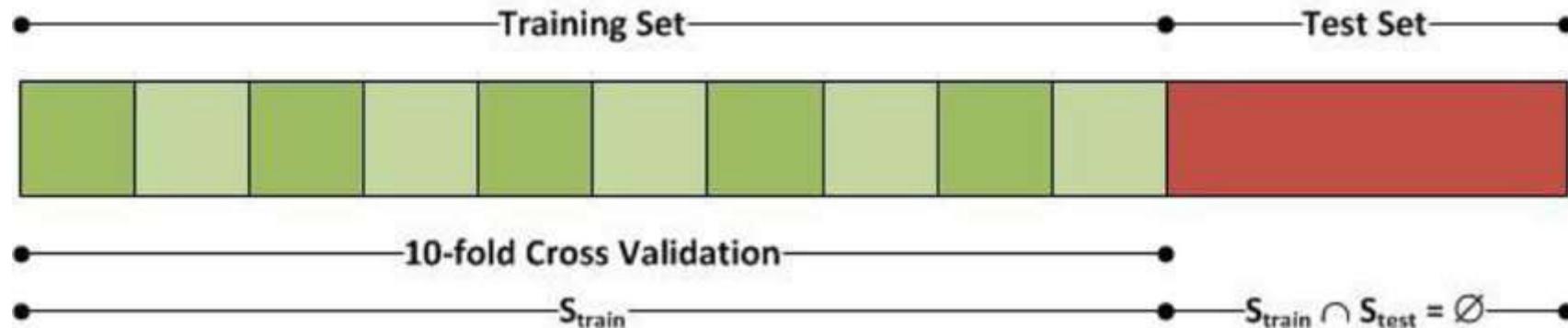


Researching the model that will be best for the type of data



testing the model

- For training a model we initially split the model into 3 three sections which are ‘Training data’ ,‘Validation data’ and ‘Testing data’.
- You train the classifier using ‘training data set’, tune the parameters using ‘validation set’ and then test the performance of your classifier on unseen ‘test data set’. An important point to note is that during training the classifier only the training and/or validation set is available. The test data set must not be used during training the classifier. The test set will only be available during testing the classifier



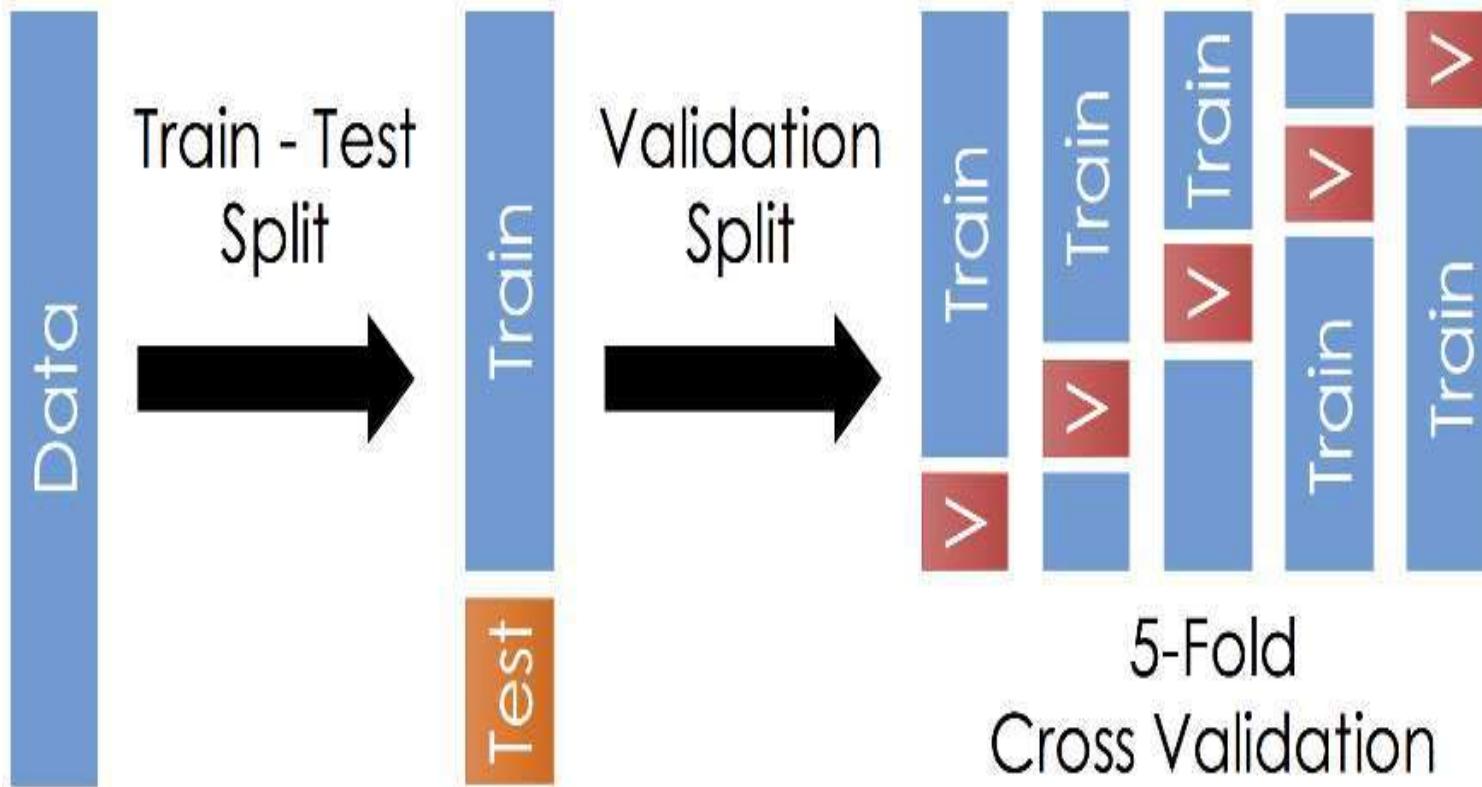
testing the model

- Training set: The training set is the material through which the computer learns how to process information. Machine learning uses algorithms to perform the training part. A set of data used for learning, that is to fit the parameters of the classifier.
- Validation set: Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. A set of unseen data is used from the training data to tune the parameters of a classifier.
- Test set: A set of unseen data used only to assess the performance of a fully-specified classifier

Once the data is divided into the 3 given segments we can start the training process.

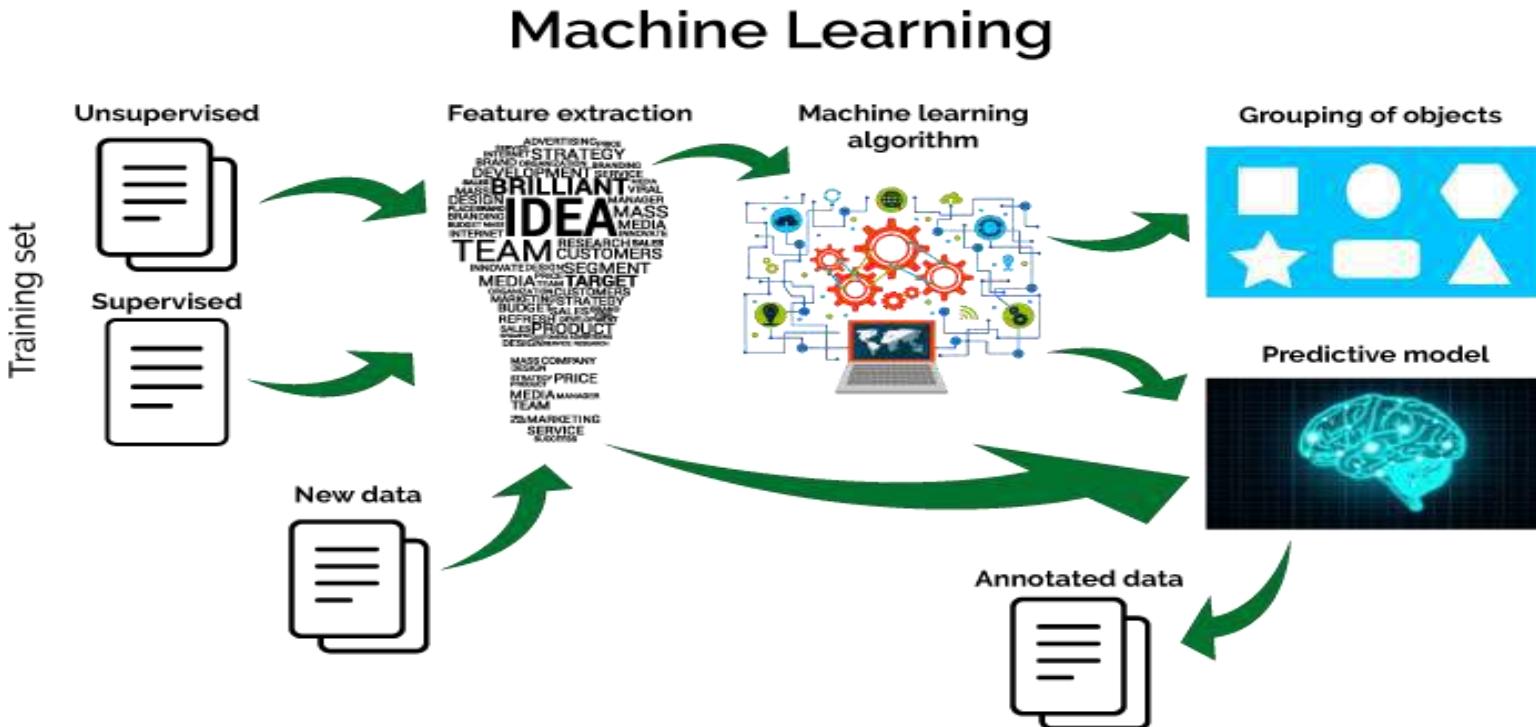
In a data set, a training set is implemented to build up a model, while a test (or validation) set is to validate the model built. Data points in the training set are excluded from the test (validation) set. Usually, a data set is divided into a training set, a validation set (some people use ‘test set’ instead) in each iteration, or divided into a training set, a validation set and a test set in each iteration.

Training and testing the model



EVALUATION

- Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. To improve the model we might tune the hyper-parameters of the model and try to improve the accuracy and also looking at the confusion matrix to try to increase the number of true positives and true negatives.



Sample Applications

- Web search
- Computational biology
- Finance
- E-commerce
- Space exploration
- Robotics
- Information extraction
- Social networks
- Debugging software
- [Your favorite area]

Supervised Learning Classification

- Example: Cancer diagnosis

Patient ID	# of Tumors	Avg Area	Avg Density	Diagnosis
1	5	20	118	Malignant
2	3	15	130	Benign
3	7	10	52	Benign
4	2	30	100	Malignant

Training
Set

- Use this **training set** to learn how to classify patients where diagnosis is not known:

Patient ID	# of Tumors	Avg Area	Avg Density	Diagnosis
101	4	16	95	?
102	9	22	125	?
103	1	14	80	?

Test Set

Input Data Classification

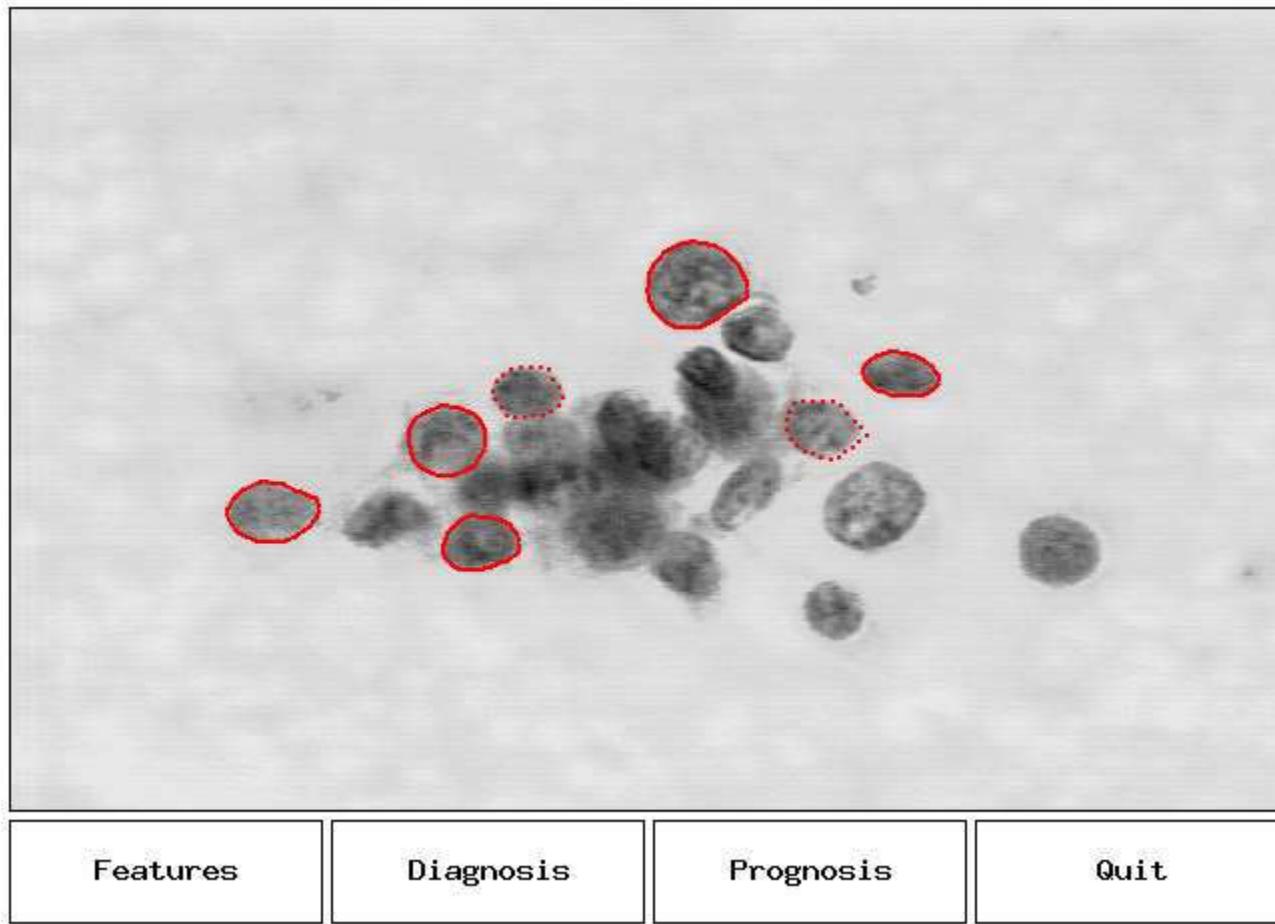
- The **input data** is often easily obtained, whereas the **classification** is not.

Classification Problem

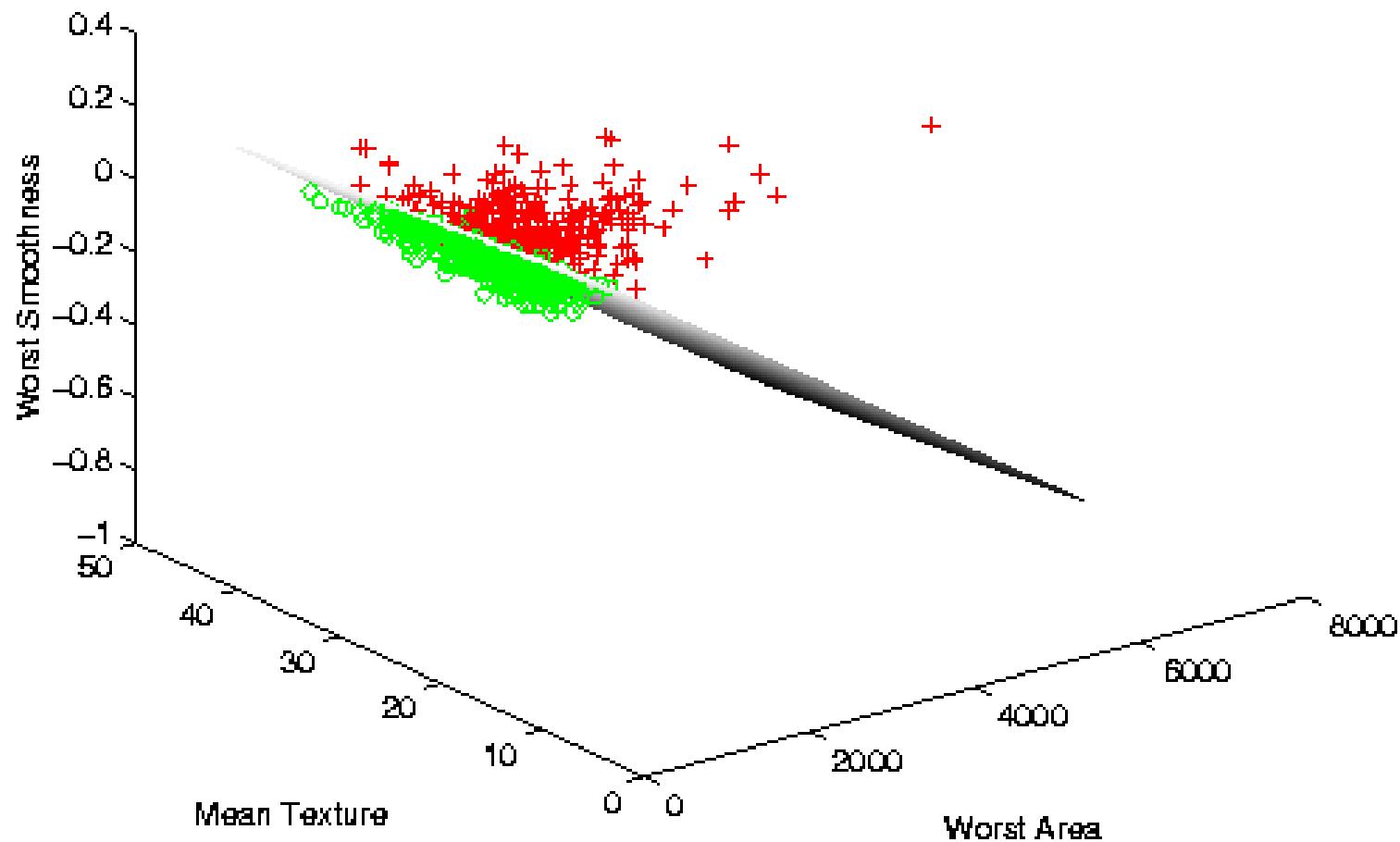
- Goal: Use training set + some learning method to produce a **predictive model**.
- Use this predictive model to classify new data.
- Sample applications:

Application	Input Data	Classification
Medical Diagnosis	Noninvasive tests	Results from invasive measurements
Optical Character Recognition	Scanned bitmaps	Letter A-Z
Protein Folding	Amino acid construction	Protein shape (helices, loops, sheets)
Research Paper Acceptance	Words in paper title	Paper accepted or rejected

Application: Cancer Diagnosis

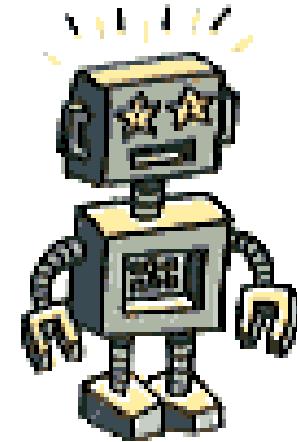


Cancer Diagnosis Separation



Robotics and ML

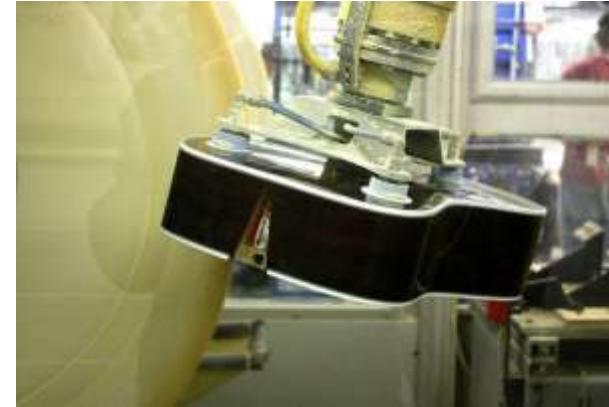
- Areas that robots are used:
 - Industrial robots
 - Military, government and space robots
 - Service robots for home, healthcare, laboratory
- Why are robots used?
 - Dangerous tasks or in hazardous environments
 - Repetitive tasks
 - High precision tasks or those requiring high quality
 - Labor savings
- Control technologies:
 - Autonomous (self-controlled), tele-operated (remote control)



Industrial Robots

- Uses for robots in manufacturing:

- Welding
- Painting
- Cutting
- Dispensing
- Assembly
- Polishing/Finishing
- Material Handling
 - Packaging, Palletizing
 - Machine loading



Space Robots

- Mars Rovers – Spirit and Opportunity
 - Autonomous navigation features with human remote control and oversight



Service Robots

- Many uses...
 - Cleaning & Housekeeping
 - Humanitarian Demining
 - Rehabilitation
 - Inspection
 - Agriculture & Harvesting
 - Lawn Mowers
 - Surveillance
 - Mining Applications
 - Construction
 - Automatic Refilling
 - Fire Fighters
 - Search & Rescue



iRobot Roomba vacuum
cleaner robot

Issues in Machine Learning

- What algorithms can approximate functions well and when?
 - How does the number of training examples influence accuracy
- Problem representation / feature extraction
- Intention/independent learning
- Integrating learning with systems
- What are the theoretical limits of learnability
- Transfer learning
- Continuous learning

Scaling issues in ML

- Number of
 - Inputs
 - Outputs
 - Batch vs realtime
 - Training vs testing

Machine Learning VS Human Learning

- Some ML behavior can challenge the performance of human experts (e.g., playing chess)
- Although ML sometimes matches human learning capabilities, it is not able to learn as well as humans or in the same way that humans do
- There is no claim that machine learning can be applied in a truly creative way
- Formal theories of ML systems exist but are often lacking (why a method succeeds or fails is not clear)
- ML success is often attributed to manipulation of symbols (rather than mere numeric information)

END OF MODULE-1

MODULE-2

Supervised Learning

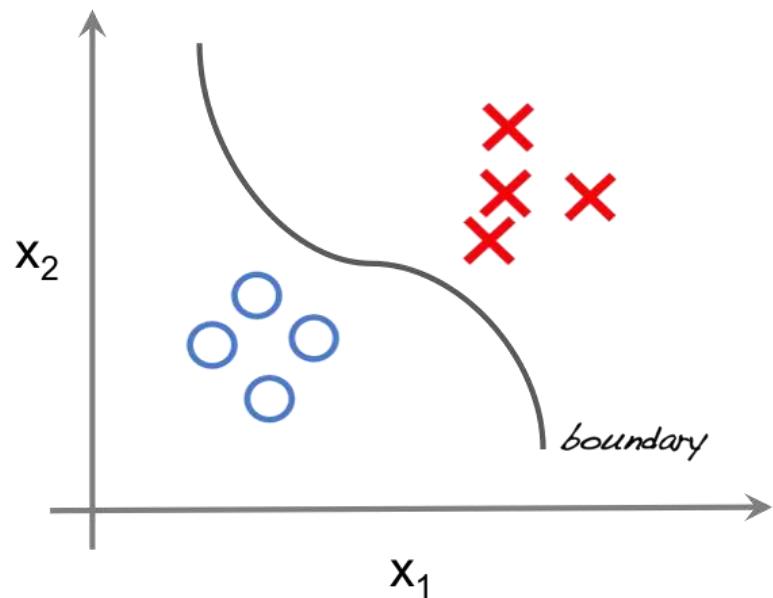


SUPERVISED LEARNING

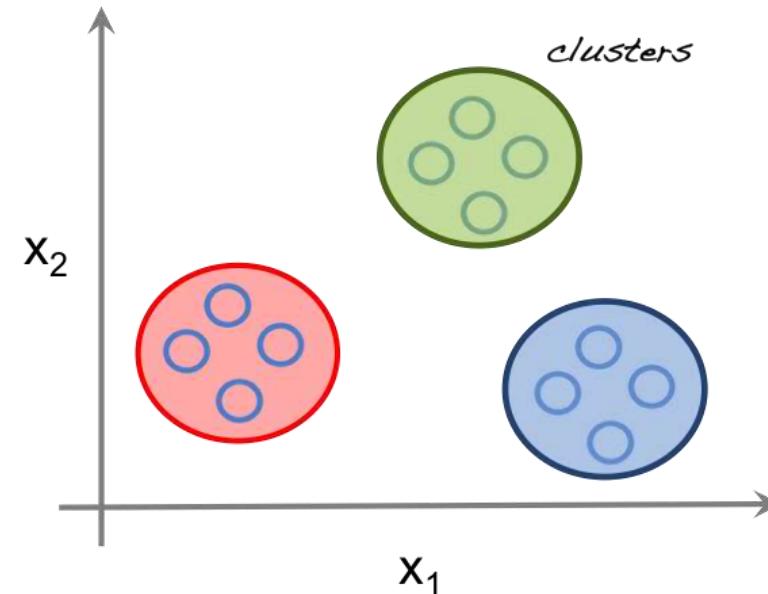
- In supervised learning, the labelled training data provide the basis for learning.
- The process of learning from the training data by a machine can be related to an expert supervising the learning process of a student.
- Here the expert is the training data.
- Training data is the past information with known value of class field or 'label'.
- Unsupervised learning uses no labelled data.
- Semi-supervised learning uses a small amount of labelled data.

Supervised vs Unsupervised

Supervised learning



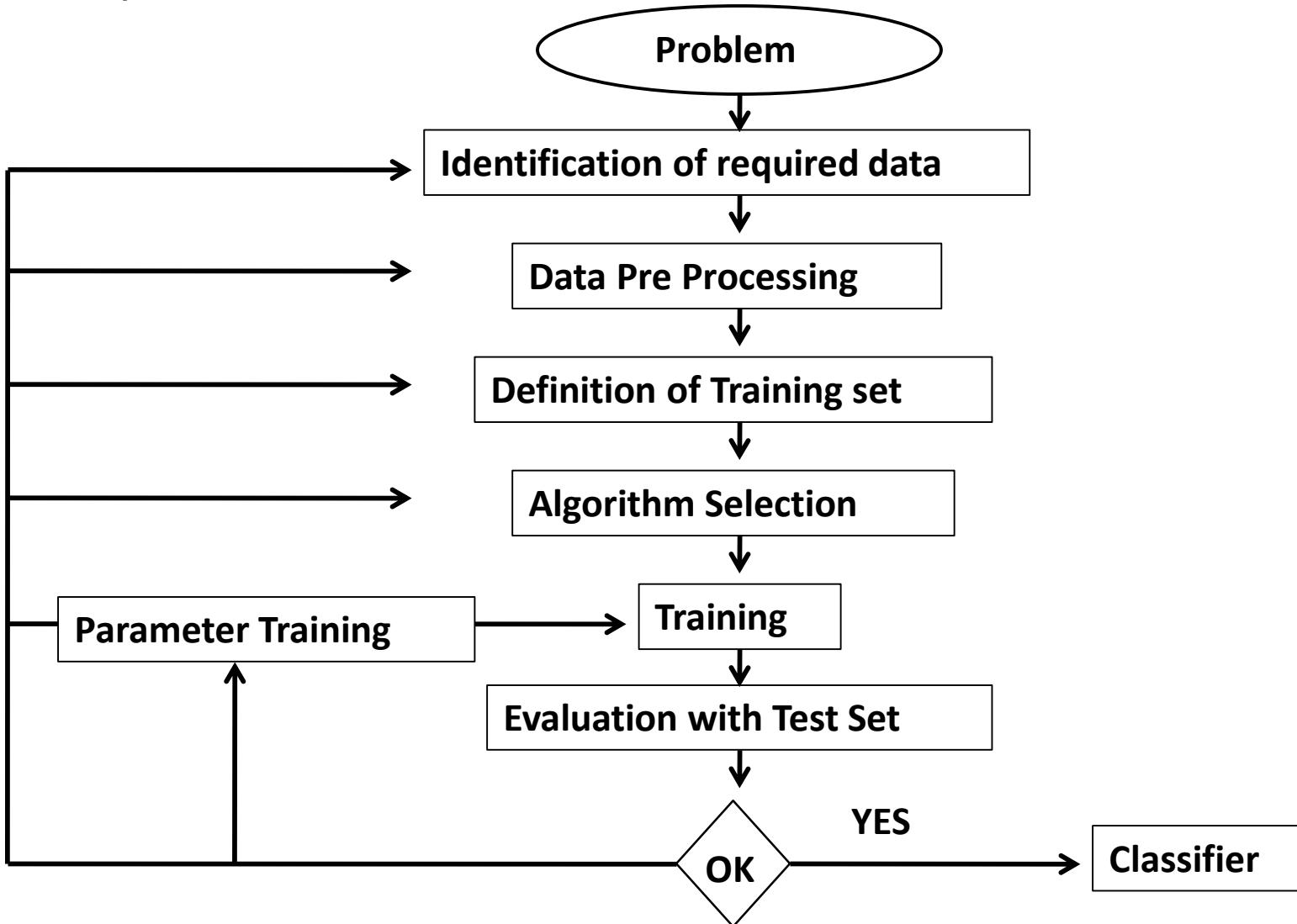
Unsupervised learning



Classification Model

- When we try to predict a categorical or nominal variable, the problem is known as a classification problem.
- Here, the problem centres around assigning a label or category or class to the test data on the basis of the label or category or class information imparted by training data.
- Classification is a type of supervised learning where a target feature, i.e. A categorical type, is predicted for test data on the basis of information obtained from training data.
- This categorical feature is known as class.

Classification Learning Steps



Common Classification Algorithms

1. k-Nearest Neighbour (kNN)
2. Decision tree
3. Random forest
4. Support Vector Machine (SVM)
5. Naive Bayes classifier

ORIGINS OF K-NN

- Nearest Neighbors have been used in statistical estimation and pattern recognition already in the beginning of 1970's (non-parametric techniques).
- The method prevailed in several disciplines and still it is one of the top 10 Data Mining algorithm.

IN A SENTENCE K-NN IS.....

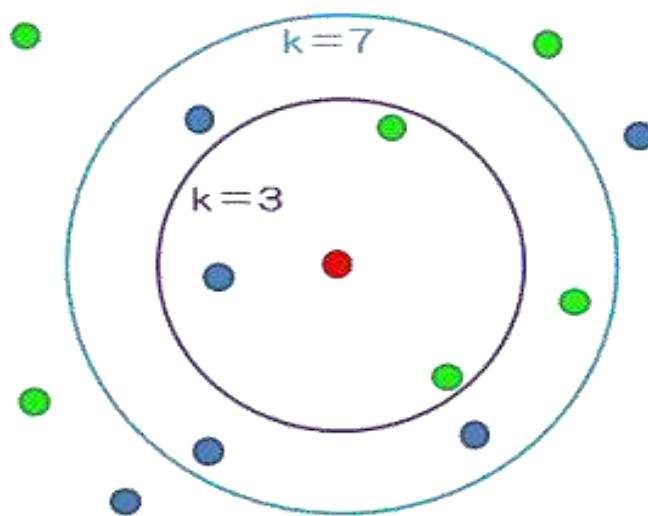
- It's how people judge by observing our peers.
- We tend to move with people of similar attributes so does data.



DEFINITION

- K-Nearest Neighbor is considered a lazy learning algorithm that classifies data sets based on their similarity with neighbors.
- “K” stands for number of data set items that are considered for the classification.

Ex: Image shows classification for different k-values.



TECHNICALLY..

- For the given attributes $A=\{X_1, X_2, \dots, X_D\}$ Where D is the dimension of the data, we need to predict the corresponding classification group $G=\{Y_1, Y_2, \dots, Y_n\}$ using the proximity metric over K items in D dimension that defines the closeness of association such that $X \in R^D$ and $Y_p \in G$.

THAT IS..

- Attribute A={Color, Outline, Dot}
- Classification Group,
G={triangle, square}
- D=3, we are free to choose K value.

Attribute

#	S	A	Attribute	Shape
	Color	Outline	Dot	
1	green	dashed	no	triangle
2	green	dashed	yes	triangle
3	yellow	dashed	no	square
4	red	dashed	no	square
5	red	solid	no	square
6	red	solid	yes	triangle
7	green	solid	no	square
8	green	dashed	no	triangle
9	yellow	solid	yes	square
10	red	solid	no	square
11	green	solid	yes	square
12	yellow	dashed	yes	square
13	yellow	solid	no	square
14	red	dashed	yes	triangle

C
l
a
s
s
i
f
i
c
a
t
i
o
n

G
r
o
u
p

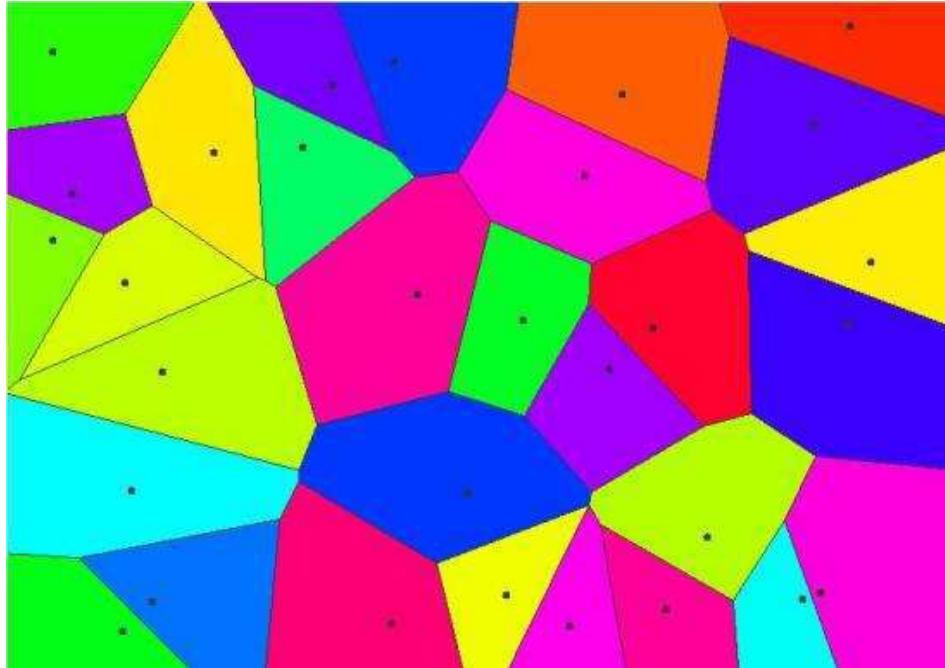
PROXIMITY METRIC

- Definition: Also termed as “Similarity Measure” quantifies the association among different items.
- Following is a table of measures for different data items:

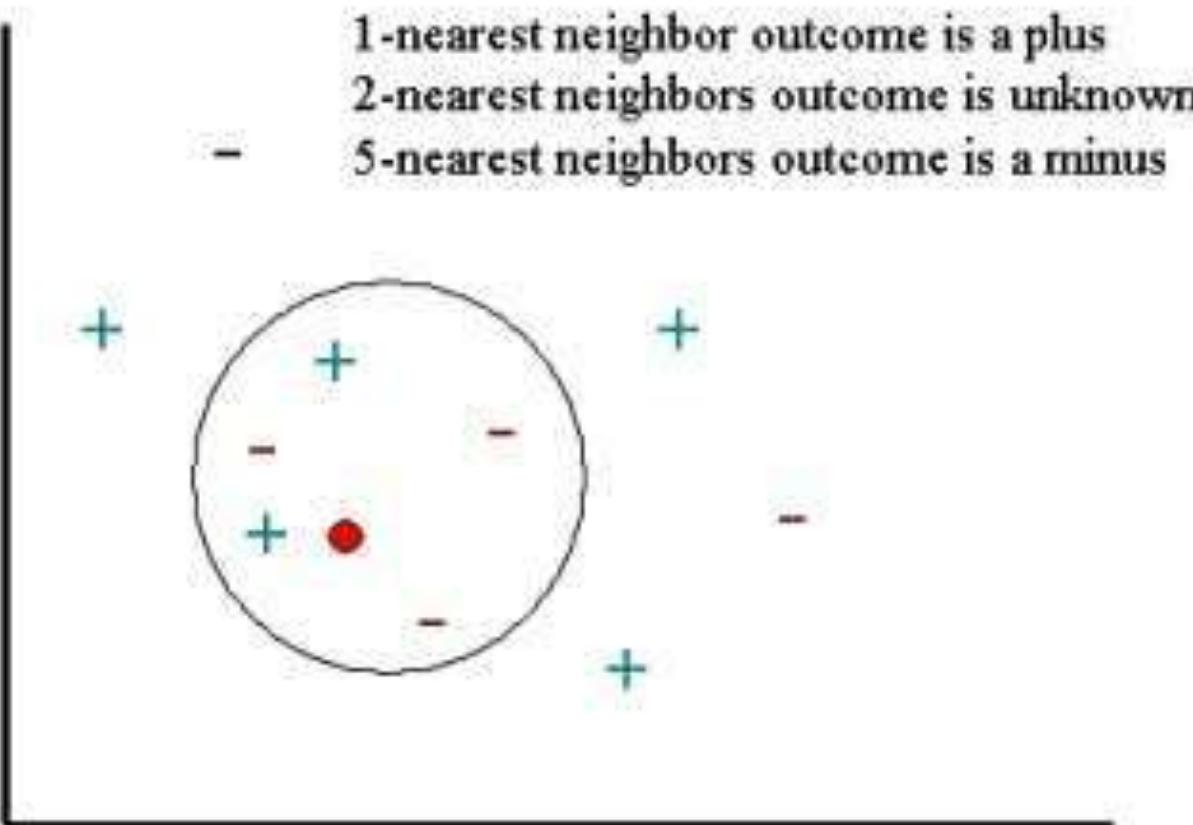
Similarity Measure	Data Format
Contingency Table, Jaccard coefficient, Distance Measure	Binary
Z-Score, Min-Max Normalization, Distance Measures	Numeric
Cosine Similarity, Dot Product	Vectors

Voronoi diagram

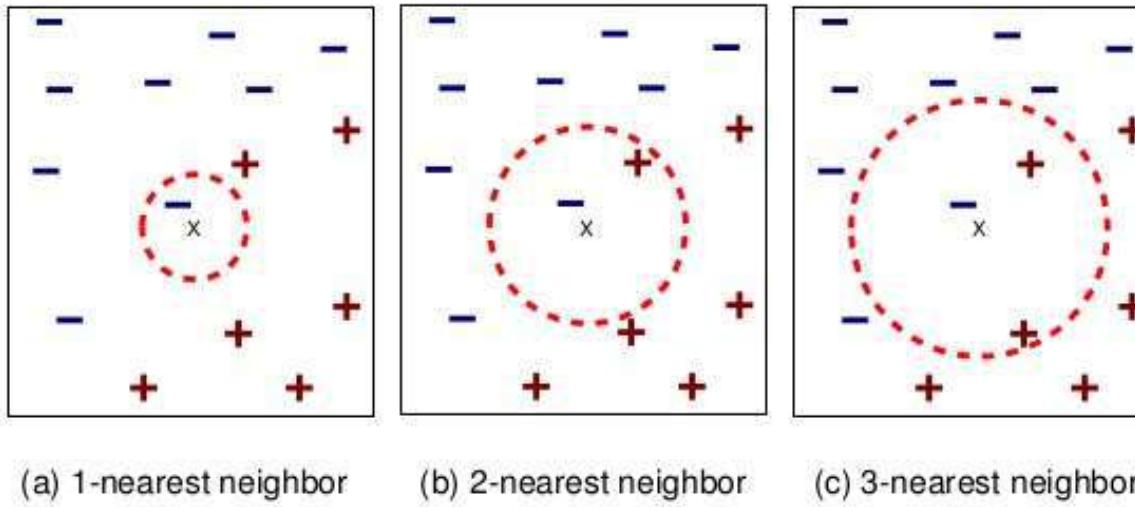
- A Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane.
- Here, $k=1$.



K-NN Example



K-NN Example



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points
that have the k smallest distance to x

PROXIMITY METRIC

- For the numeric data let us consider some distance measures:

- Manhattan Distance:

$$X = \langle x_1, x_2, \dots, x_n \rangle \quad Y = \langle y_1, y_2, \dots, y_n \rangle$$

$$\text{dist}(X, Y) = |x_1 - y_1| + |x_2 - y_2| + \dots + |x_n - y_n|$$

- Ex: Given $X = \{1, 2\}$ & $Y = \{2, 5\}$

$$\begin{aligned}\text{Manhattan Distance} &= \text{dist}(X, Y) = |1-2| + |2-5| \\ &= 1+3 \\ &= 4\end{aligned}$$

PROXIMITY METRIC

- Euclidean Distance:

$$X = \langle x_1, x_2, \dots, x_n \rangle \quad Y = \langle y_1, y_2, \dots, y_n \rangle$$

$$dist(X, Y) = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

- Ex: Given $X = \{-2, 2\}$ & $Y = \{2, 5\}$

Euclidean Distance =

K-NN IN ACTION

- Consider the following data:
 $A = \{\text{weight}, \text{color}\}$
 $G = \{\text{Apple(A)}, \text{Banana(B)}\}$
- We need to predict the type of a fruit with:
weight = 378
color = red

weight (g)	color	Type of fruit
303	3	Banana
370	1	Apple
298	3	Banana
277	3	Banana
377	4	Apple
299	3	Banana
382	1	Apple
374	4	Apple
303	4	Banana
309	3	Banana
359	1	Apple
366	1	Apple
311	3	Banana
302	3	Banana
373	4	Apple
305	3	Banana
371	3	Apple

SOME PROCESSING..

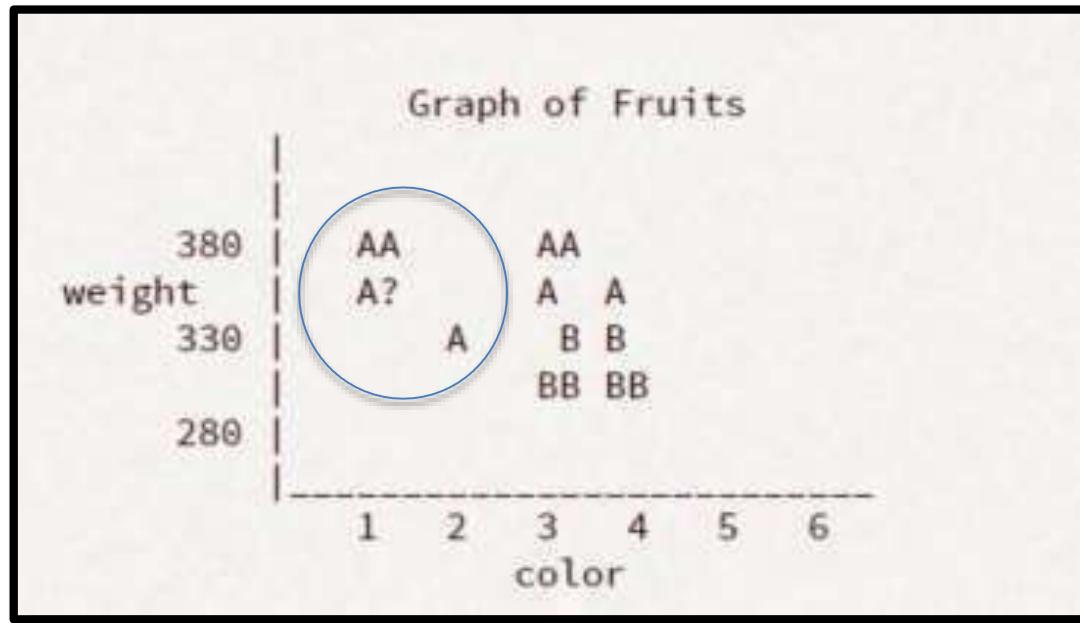
- Assign color codes to convert into numerical data:

red	1
orange	2
yellow	3
green	4
blue	5
purple	6

- Let's label Apple as “A” and Banana as “B”

PLOTTING

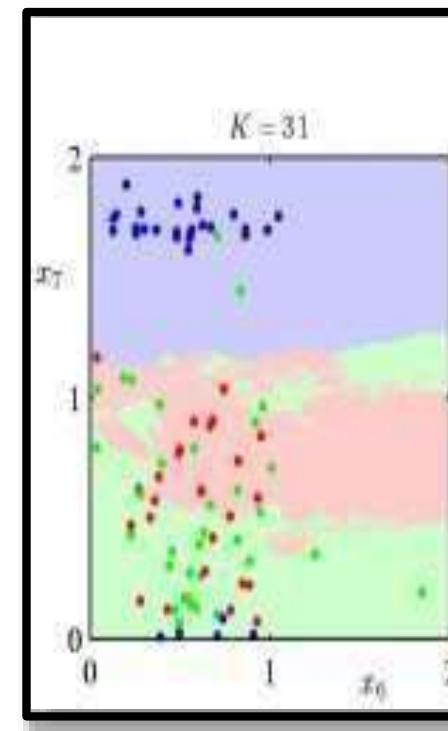
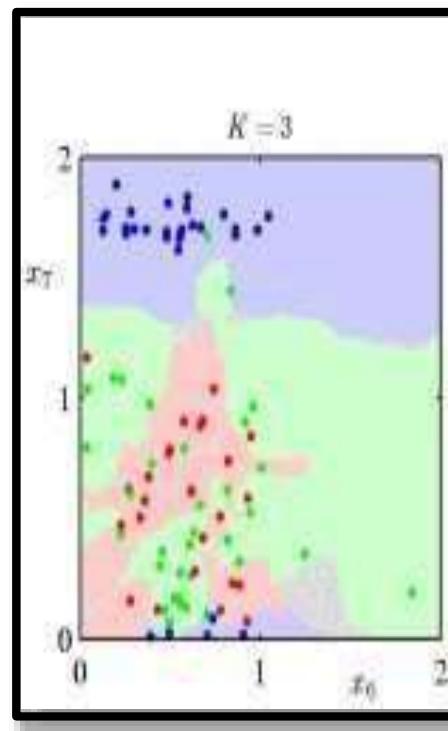
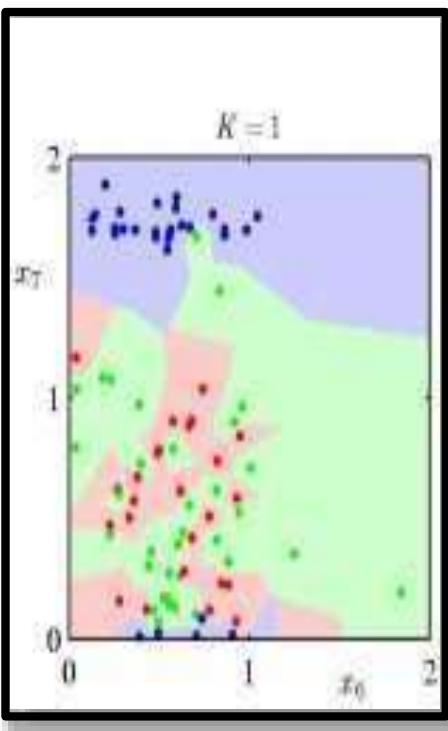
- Using K=3,
Our result will be,



AS 'K' VARIES....

- Clearly, K has an impact on the classification.

Can you guess?



K-NN PROPERTIES

- K-NN is a lazy algorithm
- The processing defers with respect to K value.
- Result is generated after analysis of stored data.
- It neglects any intermediate values.

REMARKS: FIRST THE GOOD

Advantages

- Can be applied to the data from any distribution,
for example, data does not have to be separable
with a linear boundary
- Very simple and intuitive
- Good classification if the number of samples is large
enough

NOW THE BAD....

Disadvantages

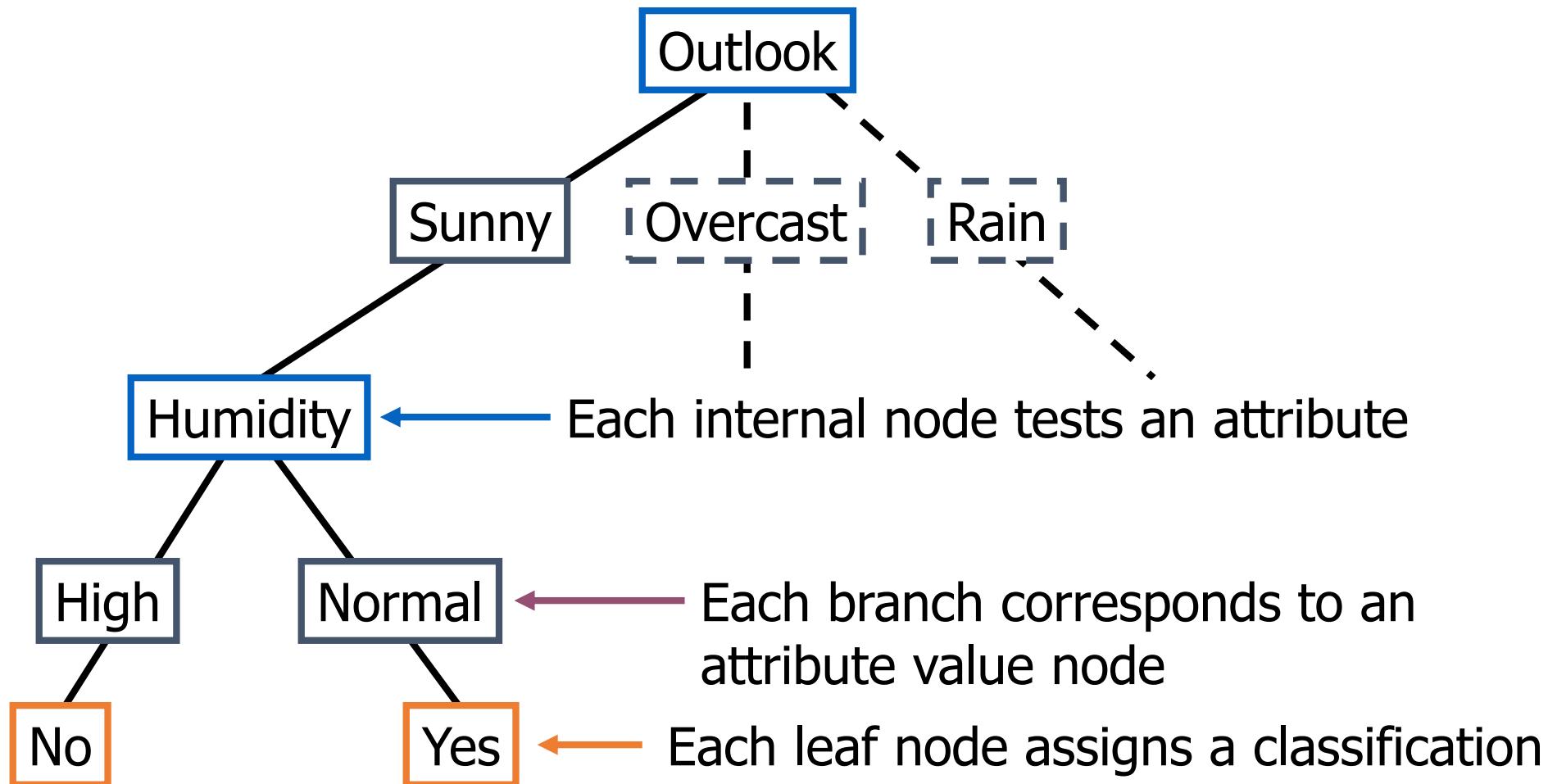
- Dependent on K Value
- Test stage is computationally expensive
- No training stage, all the work is done during the test stage
- This is actually the opposite of what we want. Usually we can afford training step to take a long time, but we want fast test step.
- Need large number of samples for accuracy

DECISION TREE

- This is one of the most adopted algorithms for classification.
- It builds a model in the form of a tree structure.
- A decision tree is used for multi-dimensional analysis with multiple classes and is characterized by ease of interpretation of rules and fast execution.
- The goal of decision tree learning is to create a model that predicts the value of the output variable based on the input variables in the feature vector.
- It contains a decision node and a leaf node.
- Each decision node corresponds to one of the feature vector.

- From every node, there are edges to children, wherein there is an edge for each of the possible values of the feature associated with the node.
- The output variable is determined by following a path that starts at the root and is guided by the values of the input variables.
- Decision trees can be used for both classification and regression.

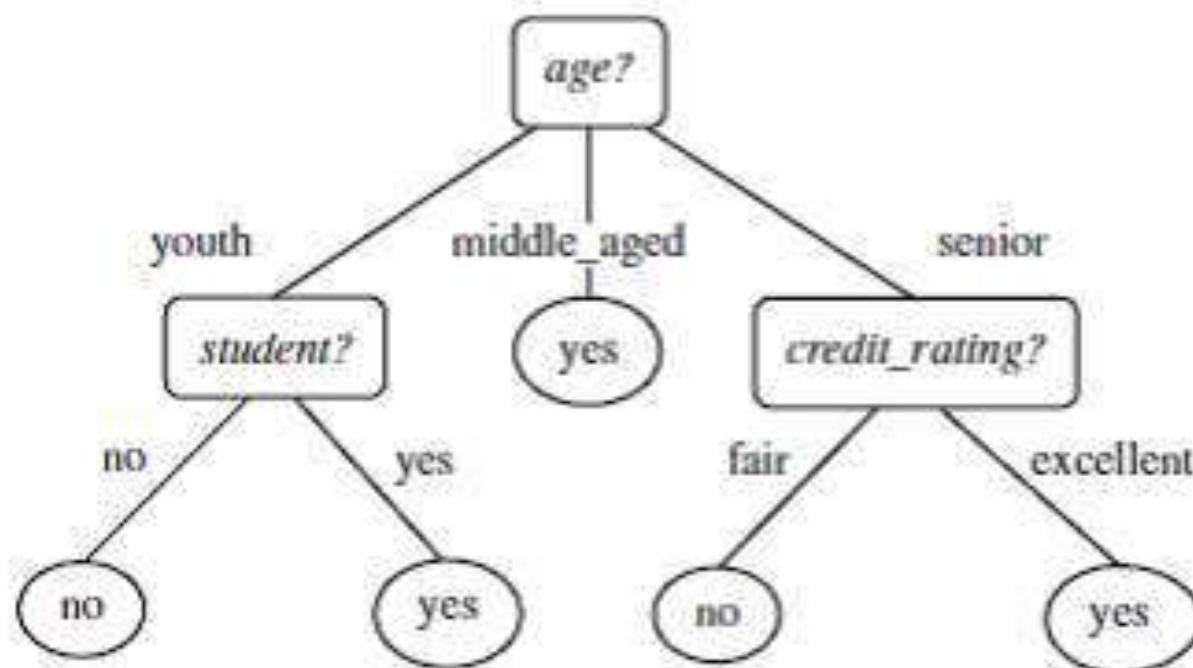
Decision Tree for PlayTennis



Example

-1

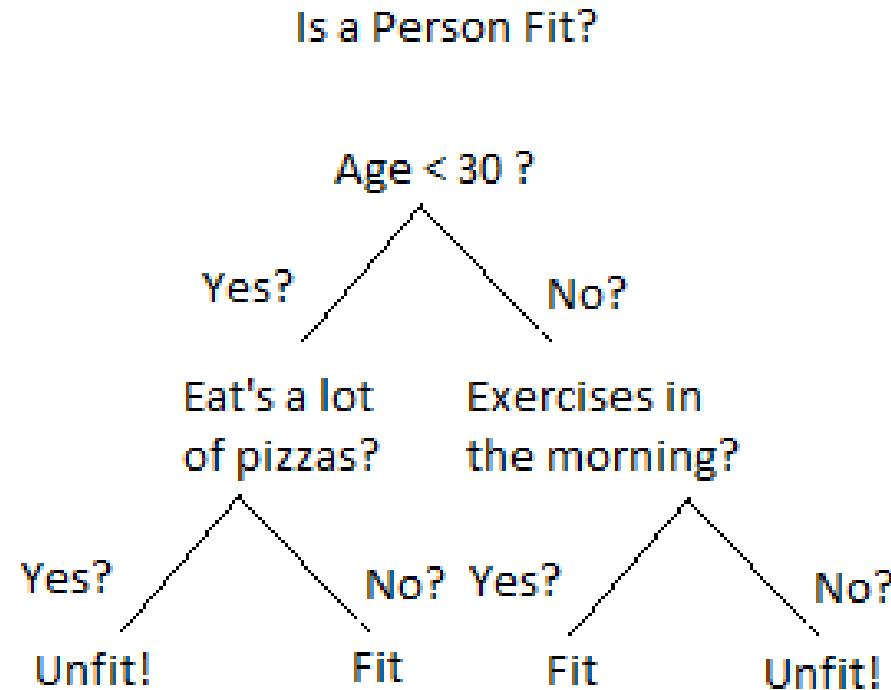
- Will a person buy a computer?



Example

-2

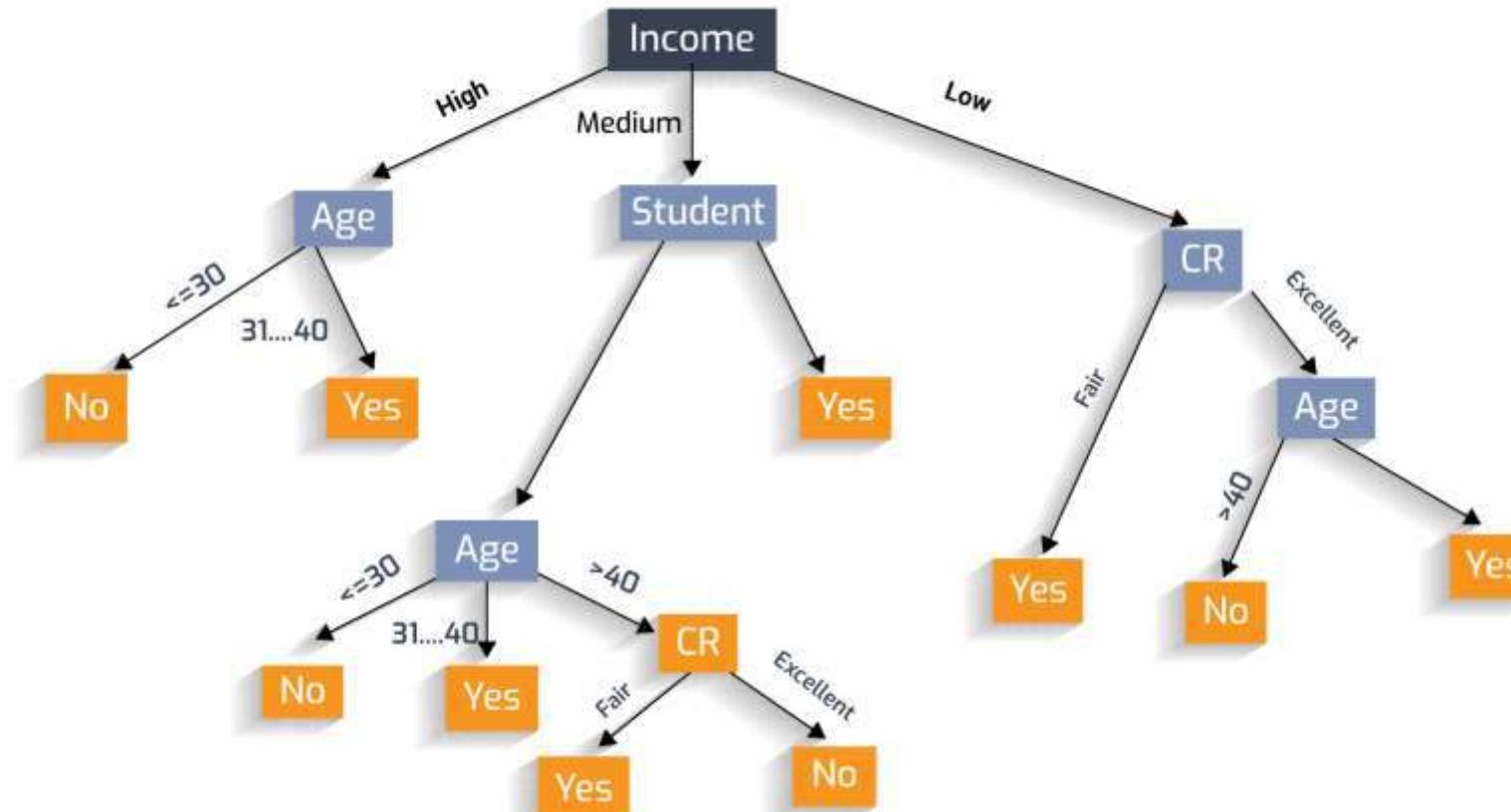
- Is a person fit?



Example

-3

- Should the LOAN be sanctioned?



Training Data for GTS recruitment

CGPA	Communication	Aptitude	Programming Skills	Job Offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

Entropy of a decision tree

- Entropy, as it relates to machine learning, is a measure of the randomness in the information being processed.
- The higher the entropy, the harder it is to draw any conclusion

$$\text{Entropy} = \sum_{i=1}^C -p_i * \log_2(p_i)$$

- Ex: For class ‘Job Offered?’ we have two values: Yes and No.
- Pi values for Yes= $8/18 = 0.44$ & No= $10/18 = 0.56$
- Entropy(S) = $-0.44 \log_2(0.44) - 0.56 \log_2(0.56)$
 $= 0.99$

Information gain of a decision tree

- The information gain is created on the basis of the decrease in entropy(S) after a data set is split according to a particular attribute(A).
- Constructing a decision tree is all about finding an attribute that returns the highest information gain.
- If information gain is 0, it means that there is no reduction in entropy due to split of the data set according to that particular feature.
- The maximum amount of information gain which may happen is the entropy of the data set before the split.

- Information gain for a particular feature A is calculated by the difference in entropy before a split(S_{bs}) with the entropy after the split(S_{as}).
- Information gain(S, A) = Entropy(S_{bs}) – Entropy(S_{as})
- For weighted summation, the proportion of examples falling into each partition is used as weight.
- $\text{Entropy}(S_{as}) = \sum (i=1 \text{ to } n) w_i \text{Entropy}(p_i)$

a) Original data set

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*LOG(pi)	0.52	0.47	0.99

Total Entropy = 0.99

b) Splitted data set(based on the CGPA)

CGPA = High				CGPA = Medium				CGPA = Low			
	Yes	No	Total		Yes	No	Total		Yes	No	Total
Count	4	2	6	Count	4	3	7	Count	0	5	5
pi	0.67	0.33		pi	0.57	0.43		pi	0	1	
-pi*LOG(pi)	0.39	0.53	0.92	-pi*LOG(pi)	0.46	0.52	0.99	-pi*LOG(pi)	0	0	0

$$\begin{aligned} \text{Total Entropy} &= (6/18 * 0.92 + 7/18 * 0.99 + 5/18 * 0) \\ &= 0.69 \end{aligned}$$

$$\begin{aligned} \text{Information Gain} &= 0.99 - 0.69 = 0.30 \end{aligned}$$

c) Splitted data set(based on ‘Communication’)

Communication = ‘Good’ Communication = ‘Bad’

Total Entropy = 0.63

Information Gain = 0.36

d) Splitted data set(based on ‘Aptitude’)

Aptitude = ‘High’

Aptitude = ‘Low’

Total Entropy = 0.52

Information Gain = 0.47(Entropy=0)

e) Splitted data set(based on ‘Programming Skills’)

Programming Skills = ‘Good’

Programming Skills = ‘Bad’

Total Entropy = 0.95

Information Gain = 0.04

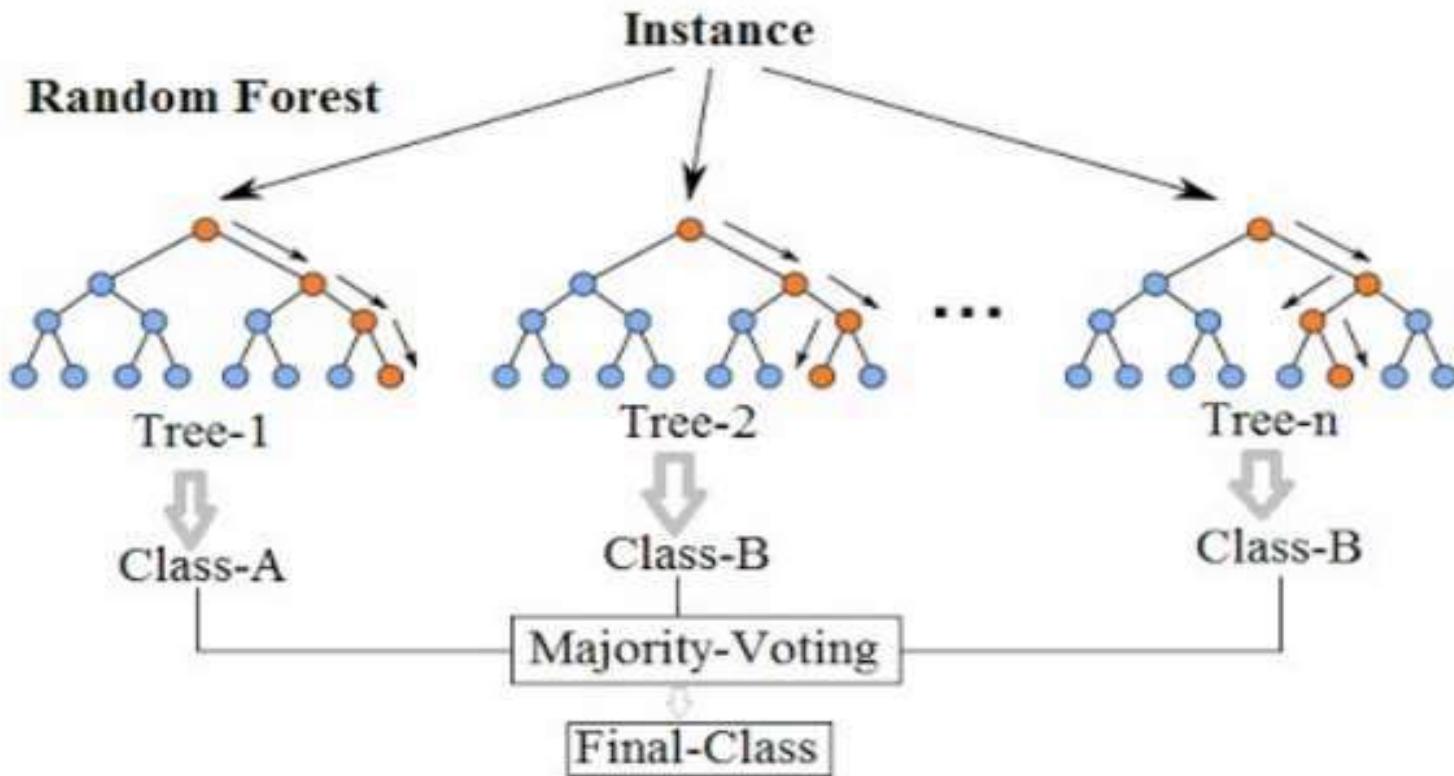
Avoiding overfitting in decision tree- pruning

- The decision tree algorithm, unless a stopping criterion is applied, may keep growing indefinitely.
- To prevent a decision tree getting overfitted to the training data, pruning of the decision tree is essential.
- Pruning a decision tree reduces the size of the tree such that the model is more generalized and can classify unknown and unlabeled data in a better way.
- Pre-pruning: Stop growing the tree before it reaches perfection.
- Post-pruning: Allow the tree to grow entirely and then post-prune some of the branches from it.

Random Forest Model

- It is an ensemble classifier, i.e., a combining classifier that uses and combines many decision tree classifiers.
- Ensembling is usually done using the concept of bagging with different feature sets.
- The reason for using large number of trees in random forest is to train the trees enough such that contribution from each feature comes in a number of models.
- After the random forest is generated by combining the trees, majority vote is applied to combine the output of the different trees.
- Ensembled model yields better result than decision trees.

Random Forest Simplified



Random forest algorithm

- The algorithm works as follows:
 1. If there are N variables or features in the input data set, select a subset of ' m ' ($m < N$) features at random out of the N features.
 2. Use the best split principle on these ' m ' features to calculate the number of nodes ' d '.
 3. Keep splitting the nodes to child nodes till the tree is grown to maximum possible extent.
 4. Select a different subset of the training data 'with replacement' to train another DT with steps (1) to (3). Repeat this to build and train ' n ' decision trees.
 5. Final class assignment is done on the basis of the majority votes from the ' n ' trees.

Strengths of RF

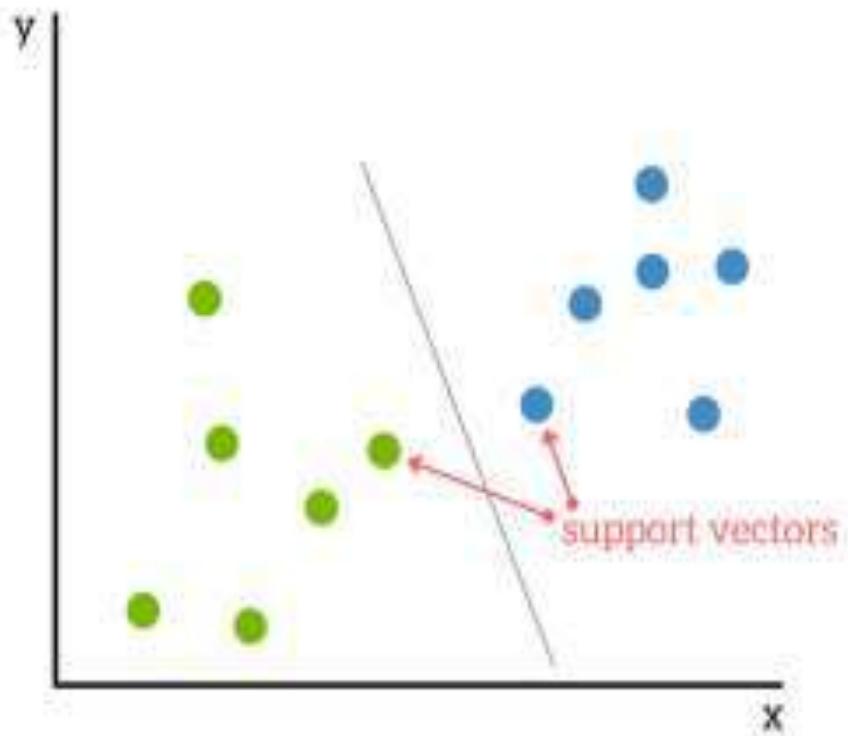
- It runs efficiently on large and expensive data sets.
- It has a robust method for estimating missing data and maintains precision when a large proportion of data is absent.
- It has powerful techniques for balancing errors in a class population of unbalanced data sets.
- It gives estimates about which features are the most important ones in the overall classification.

Drawback of RF

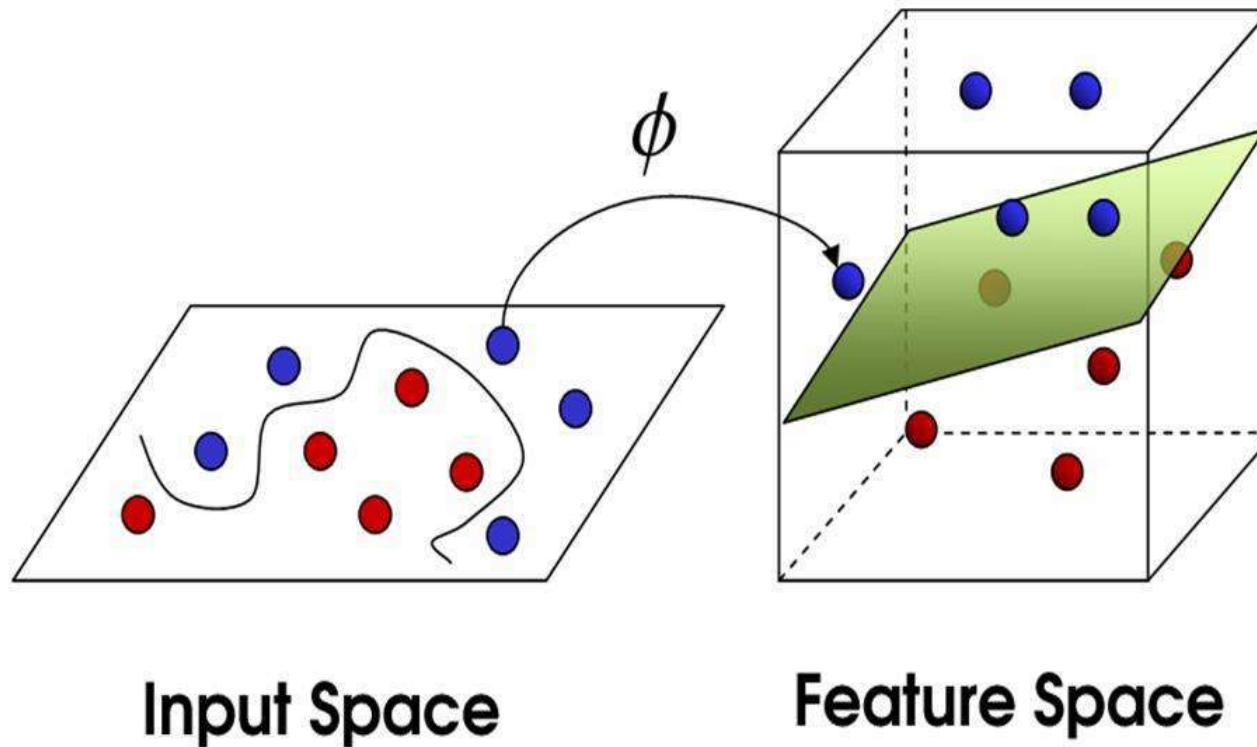
- As it combines many decision trees, it is not easy to understand as a decision tree model.
- Computationally, it is much more expensive than a simple decision tree.

Support Vector Machine

- SVM is a model which can perform linear classification as well as regression.
- It is based on the concept of a surface called hyperplane, which draws a boundary between data instances plotted on a multi-dimensional feature space.
- The output prediction is one of the two classes defined in the training data.

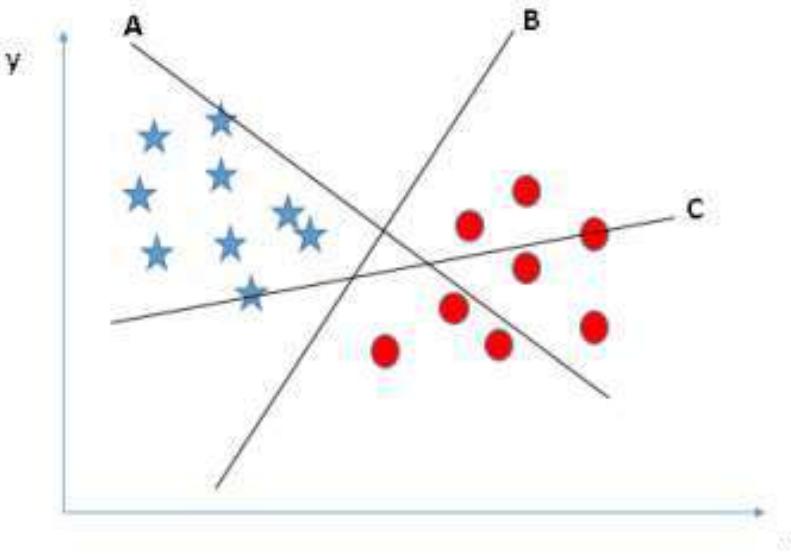


Principle of Support Vector Machines (SVM)



Identify the right hyper-plane (Scenario-1)

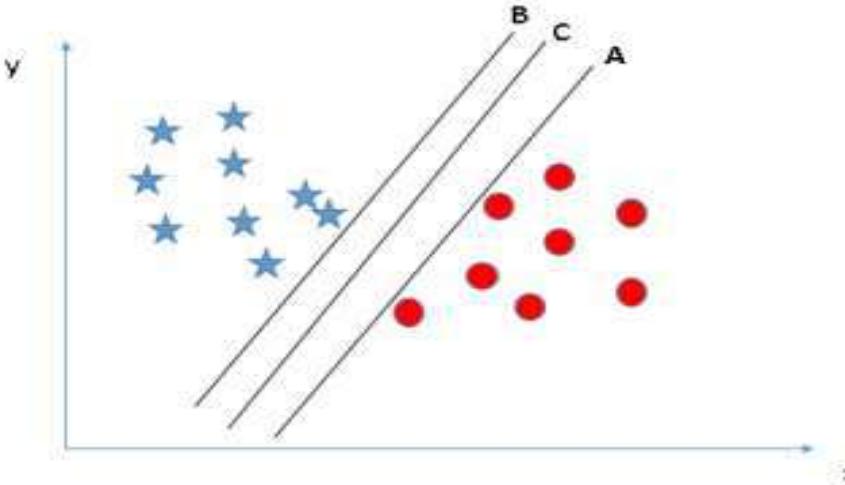
- Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.
- Select the hyper-plane which segregates the two classes better?



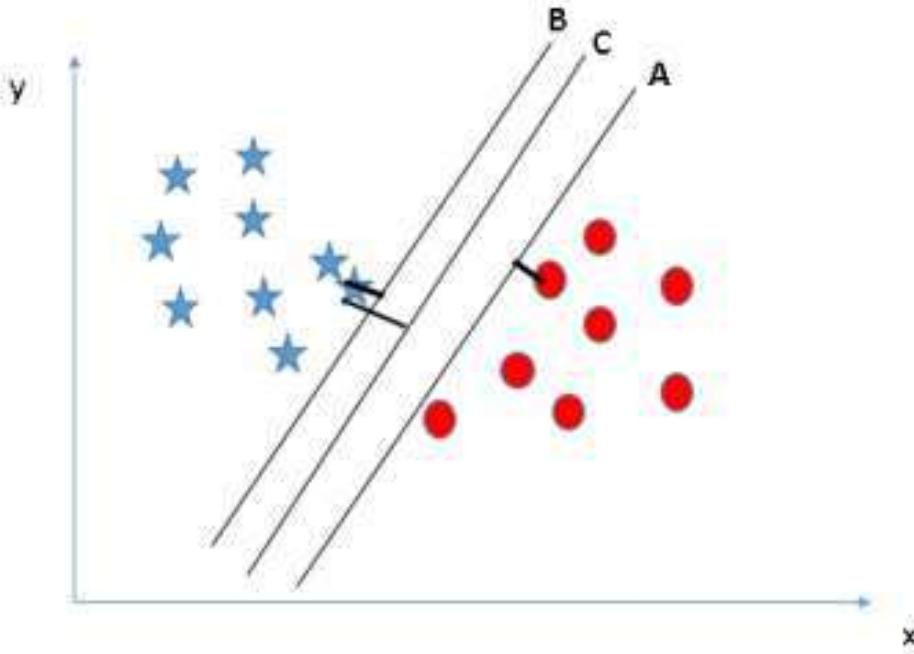
- In this scenario, we performed this, ... s excellently

Identify the right hyper-plane (Scenario-2)

- Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

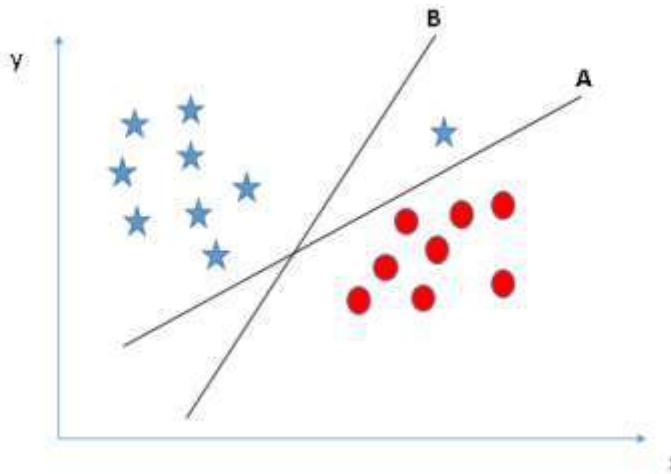


- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.



- We can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C.

Scenario -3

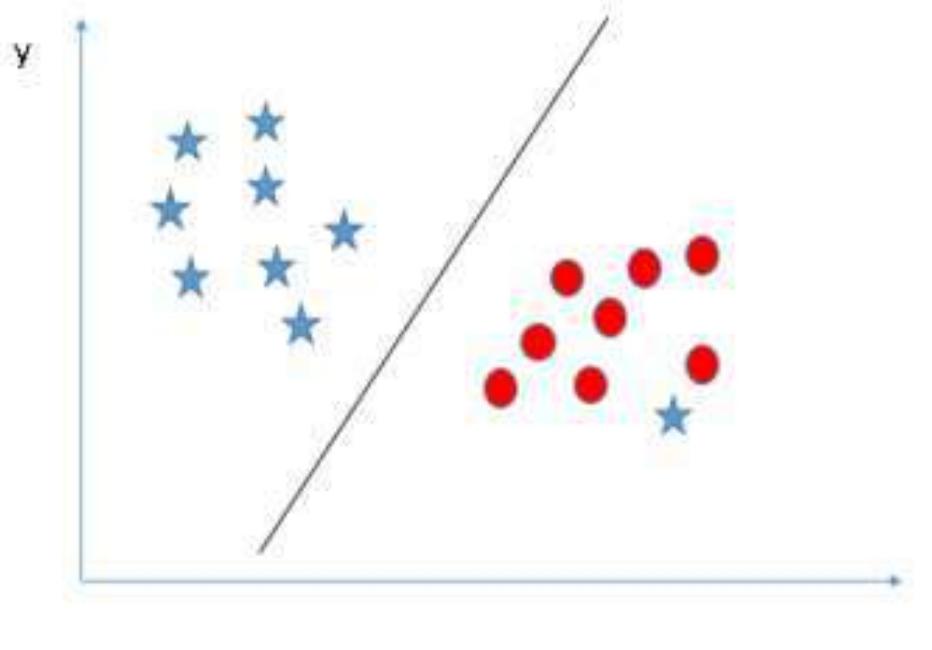


- Here hyper-plane B as it has higher margin compared to A.
- SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.
- Hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

Scenario

-4

- Here, we are unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.
- SVM has a feature to ignore outliers and find the hyperplane that has maximum margin. SVM is robust to outliers.



Strengths of SVM

- SVM can be used for both classification & regression.
- It is robust, i.e. not much impacted by data with noise or outliers.
- The prediction results using this model are very strong.

Weakness of SVM

- SVM is applicable only for binary classification, i.e. when there are only two classes in the problem.
- While dealing with high dimensional data, it becomes very complex.
- It is slow for large dataset, i.e. a data set with more features or instances.
- It is memory-intensive(throughput is bounded by the device memory bandwidth).

Introduction to Regression

- Regression is a technique used to model and analyze the relationships between variables and often times how they contribute and are related to producing a particular outcome together.
- Here, dependent variable (Y) is the one whose value is to be predicted, ex.- the price quote of the real estate property.
- This variable is presumed to be functionally related to one (X) or more independent variables called predictors.
- $Y = f(X)$

Common Regression Algorithms

- The common regression algorithms are:
 1. Simple linear regression
 2. Multiple linear regression
 3. Polynomial regression
 4. Multivariate adaptive regression splines
 5. Logistic regression
 6. Maximum likelihood estimation(least square)

Regression examples

Stock market



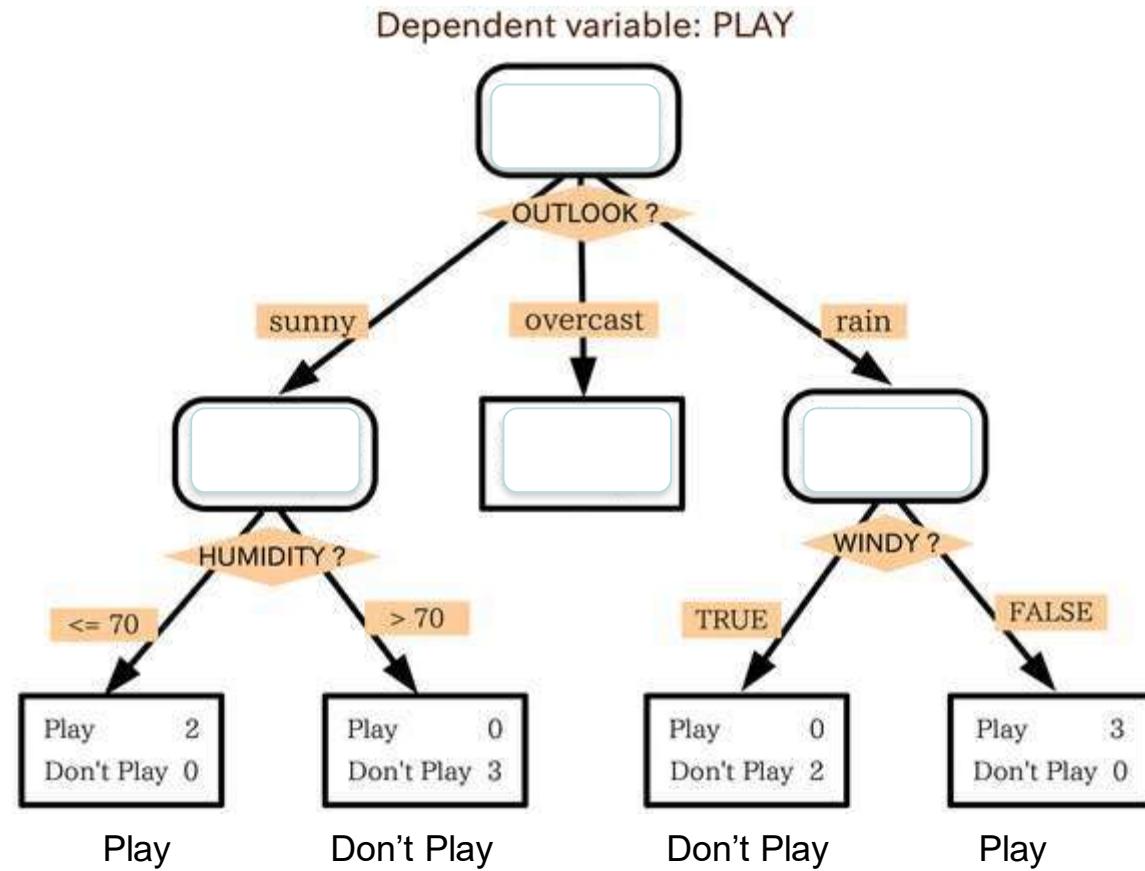
Weather prediction



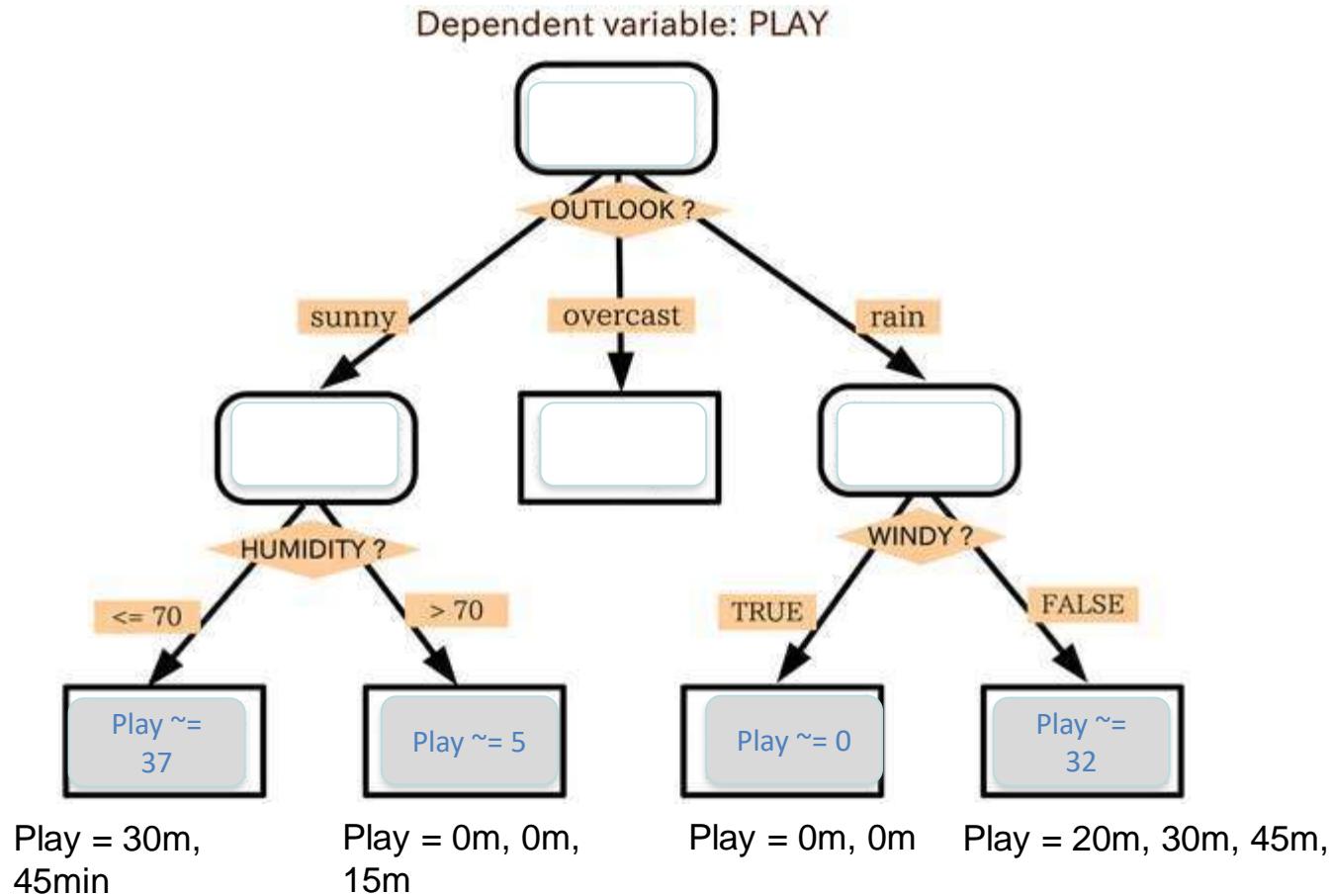
Temperature
72° F

Predict the temperature at any given location

A decision tree: classification

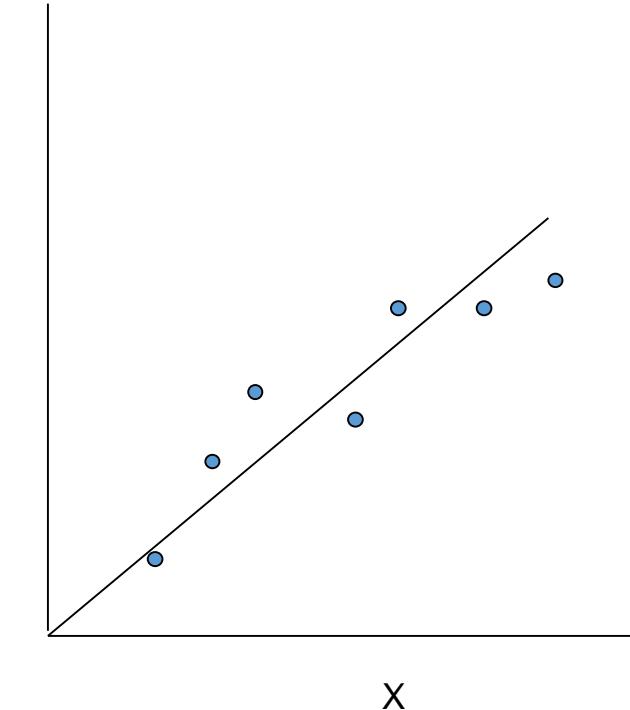


A regression tree



Linear regression

- Given an input x we would like to compute an output y
- For example:
 - Predict height from age
 - Predict Google's price from Yahoo's price
 - Predict distance from wall from sensors

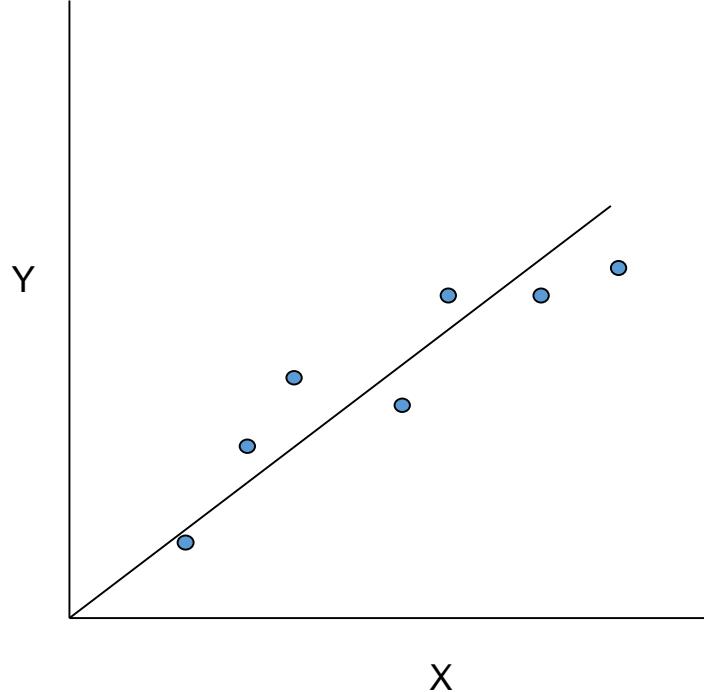


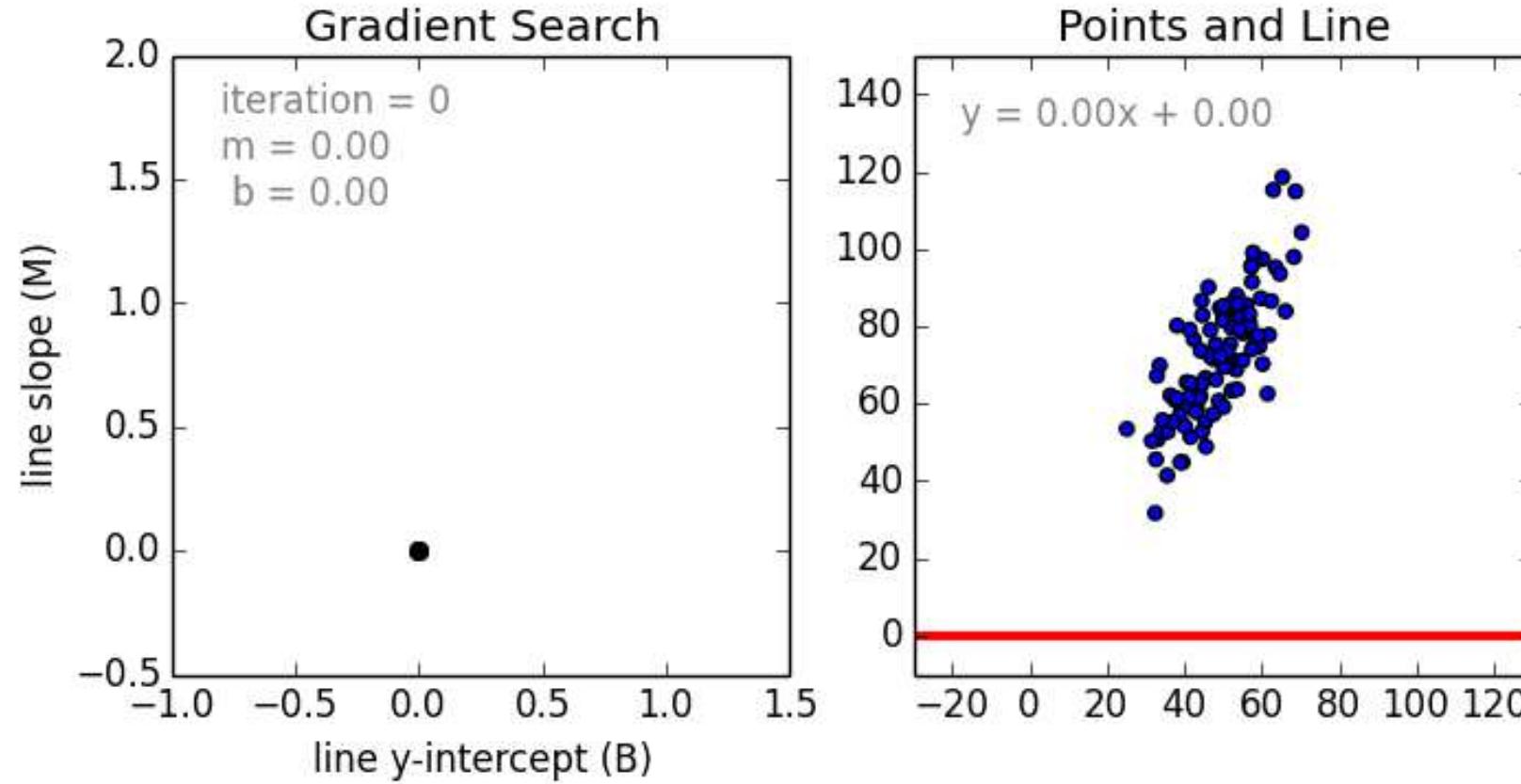
Linear regression

- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

What we are trying to predict $\xrightarrow{y = wx + \varepsilon}$ Observed values

where w is a parameter and ε represents measurement noise, model noise or other (data) noise





Linear regression

- Our goal is to estimate w from a training data of $\langle x_i, y_i \rangle$ pairs

- Optimization goal: minimize squared error (least squares):

$$\arg \min_w \sum_i (y_i - w x_i)^2$$

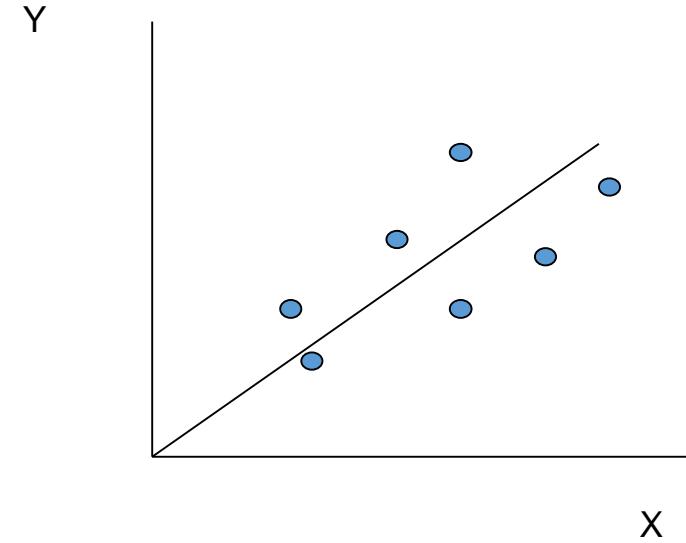
- Why least squares?

- minimizes squared distance between measurements and predicted line

- has a nice probabilistic interpretation (Gaussian Likelihood same as Mean Sq.)

- first degree polynomial model

$$y = w x + \epsilon$$



Multivariate regression

- What if we have several inputs?
 - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate regression problem
- Again, its easy to model:

$$y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$$

The diagram illustrates the multivariate regression equation $y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$. Each term in the equation is annotated with a cyan box containing the variable name followed by 'stock price'. Arrows point from these boxes to the corresponding terms in the equation:

- An arrow points from the box 'Google's stock price' to the term w_0 .
- An arrow points from the box 'Yahoo's stock price' to the term w_1x_1 .
- An arrow points from the box 'Microsoft's stock price' to the term w_kx_k .

- Price of property = f (Area, location, floor, ageing, amenities)
- $Y = a + b_1 X_1 + b_2 X_2$

Where Y is the three-dimensional space, X_1 & X_2 are the predictor variables, b_1 & b_2 are referred as partial regression coefficients.

Assumptions in Regression Analysis

1. The dependent variable(Y) can be calculated as a linear function of a specific set of independent variables(X) and an error term(ϵ).
2. The number of observations(n) is greater than the number of parameters(k) to be estimated, ie $n>k$.
3. Regression line can be valid only over a limited range of data.
4. Variance is the same for all values of X.
5. The error term(ϵ) is normally distributed.
6. The values of the error term(ϵ) are independent and are not related to any values of X.

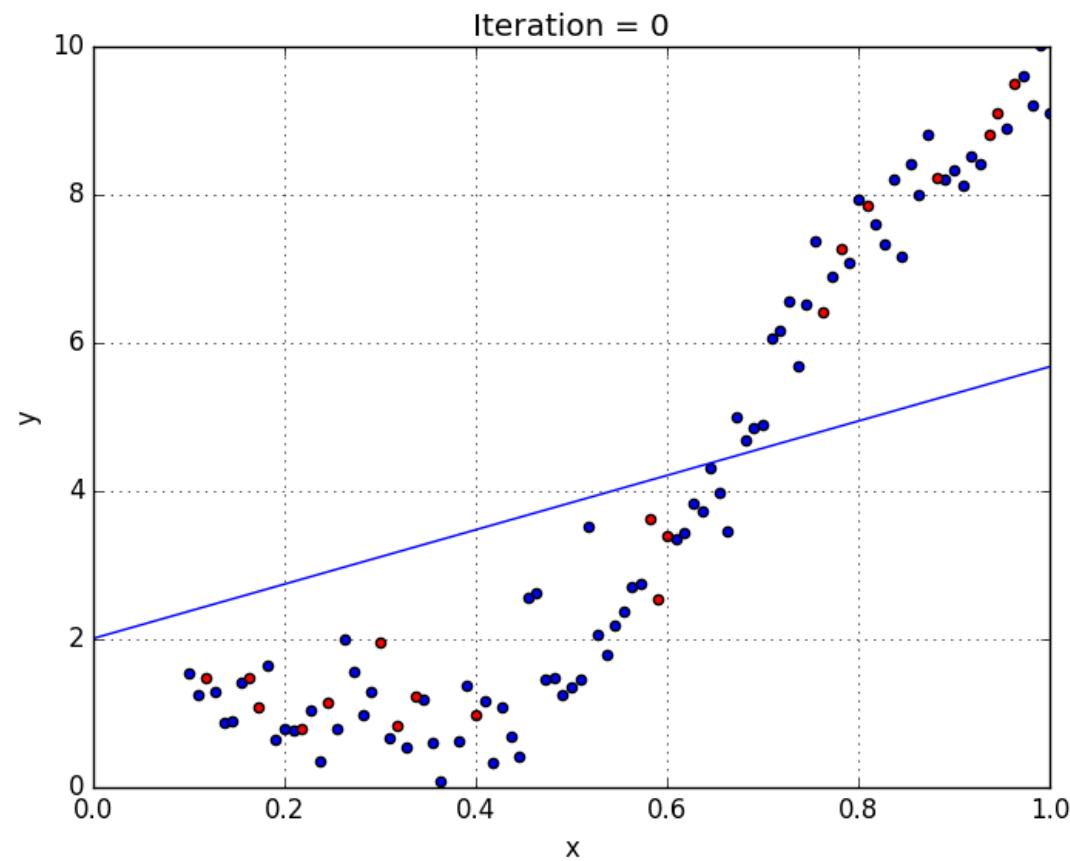
The OLS(Ordinary least square) estimator is the Best Linear Unbiased Estimator(BLUE) and this is called as Gauss-Markov Theorem.

Polynomial regression

Polynomial regression model is the extension of the simple linear model by adding extra predictors obtained by raising(squaring) each of the original predictors to a power.

If there are three variable, X, X^2, X^3 are used as predictors.

$$F(x) = C_0 + C_1 X^1 + C_2 X^2 + C_3 X^3$$

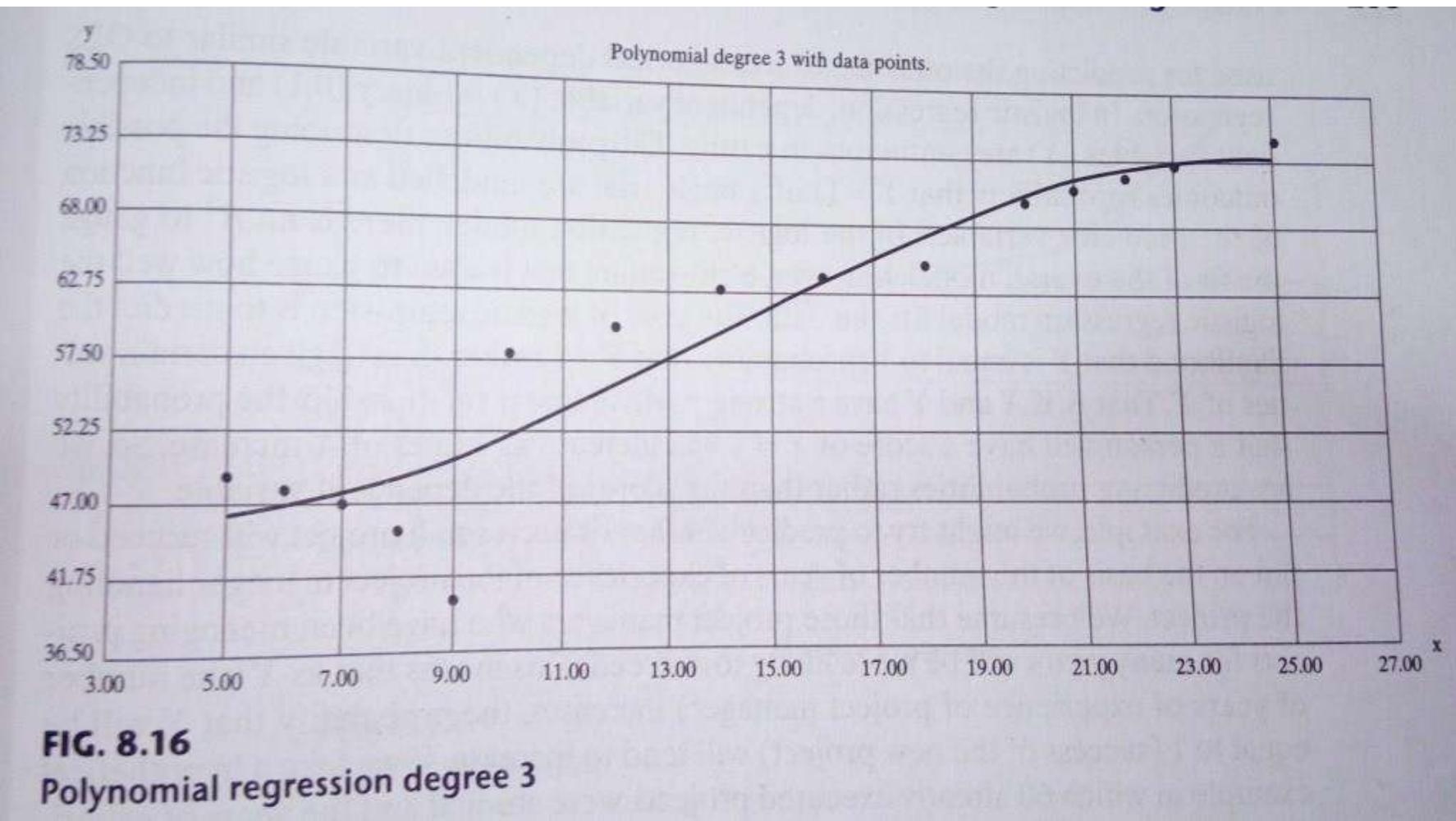


Let us use the below data set of (X, Y) for degree 3 polynomial

Internal Exam(X)	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam(X)	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

The regression line is slightly curved for degree = 3.

The regression line will curve further if we increase the polynomial degree.



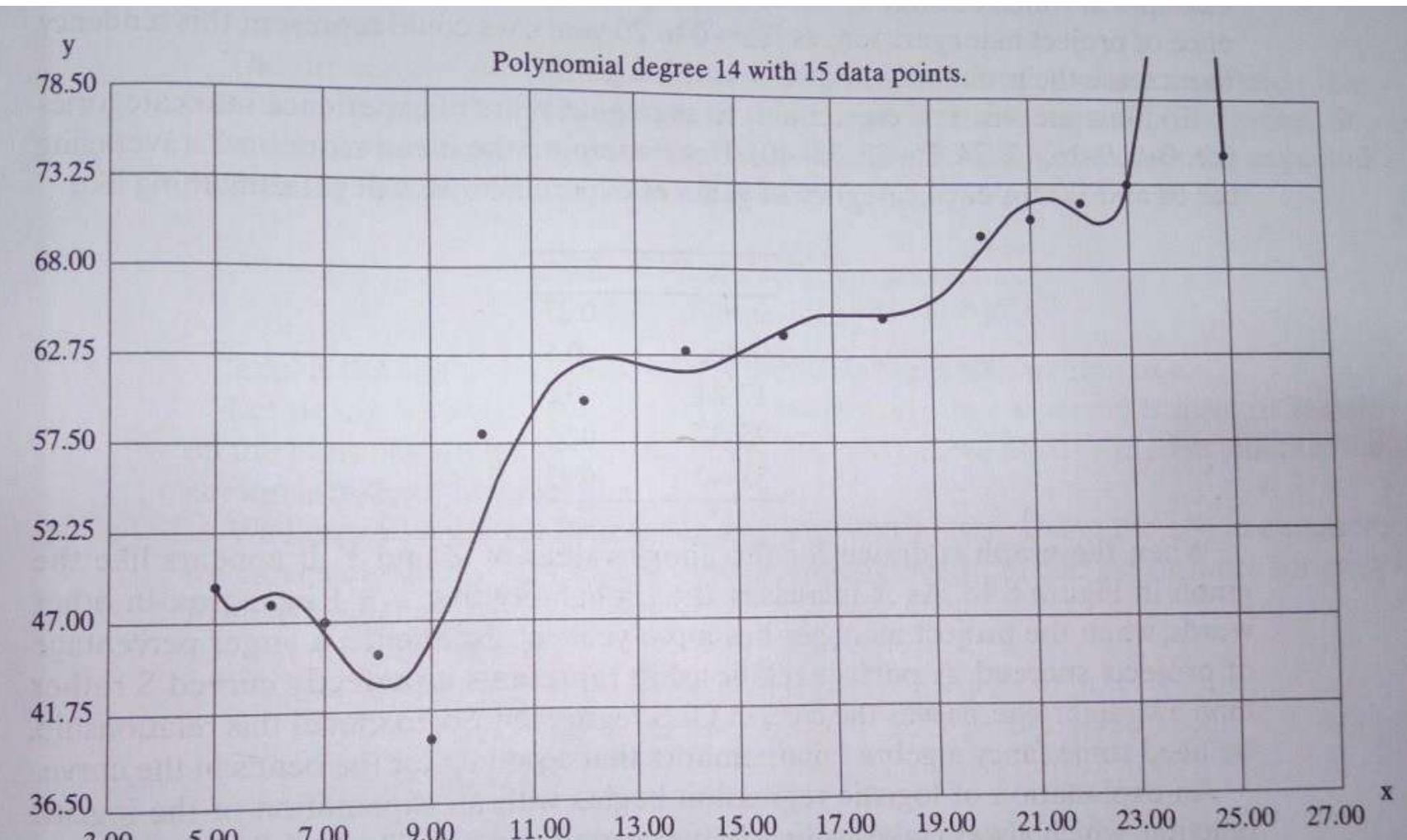


FIG. 8.17

Polynomial regression degree 14

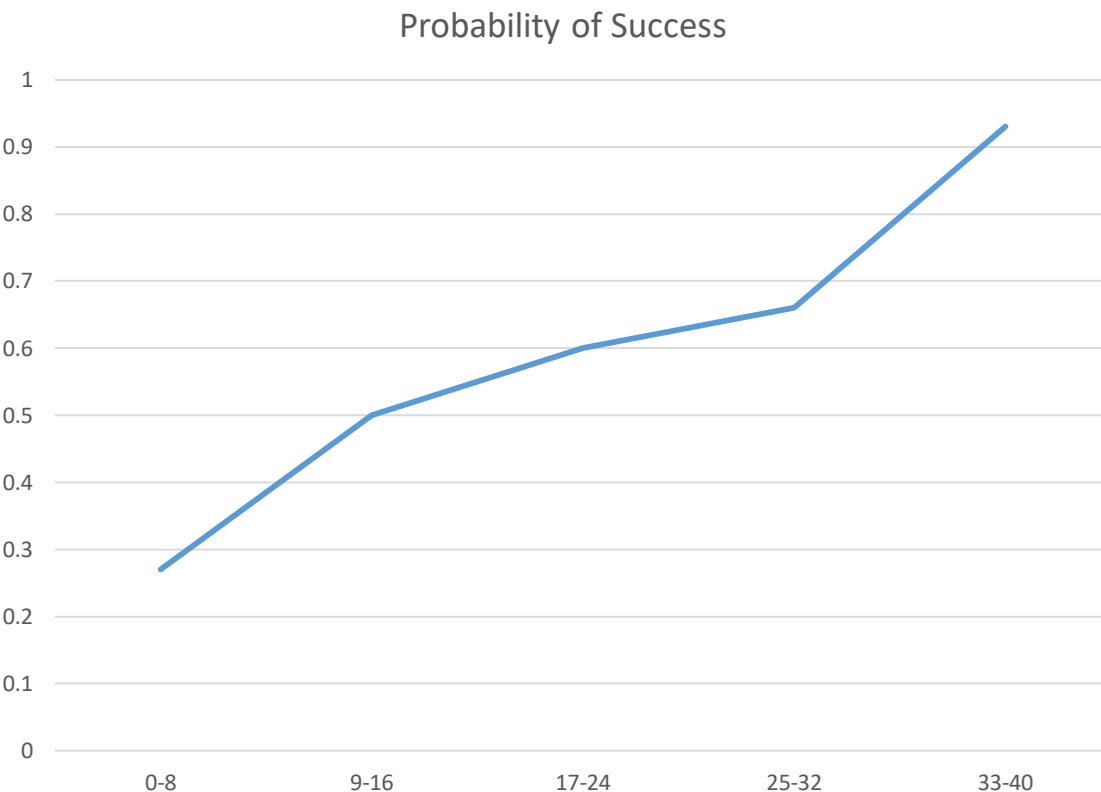
Logistic Regression

- The **logistic model** is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.
- Logistic regression(LR) is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.
- It can be used for both Classification and Regression based on the given problem.
- The goal of LR is to predict the likelihood that Y is equal to 1(probability that Y = 1 rather than 0) given certain values of X.

Example

- We may predict the success or failure of a small project on the basis of the number of years of experience of the project manager handling the project.
- This means that as X (the number of years of experience of manager) increases, the probability that Y will be equal to 1(success of project) will tend to increase.

Years of Experience	Probability of Success
0-8	0.27
9-16	0.5
17-24	0.6
25-32	0.66
33-40	0.93



END OF MODULE-2

Difference between Artificial intelligence and Machine learning

Artificial Intelligence

Artificial intelligence is a technology using which we can create intelligent systems that can simulate human intelligence.

The Artificial intelligence system does not require to be pre-programmed, instead of that, they use such algorithms which can work with their own intelligence.

It involves machine learning algorithms such as Reinforcement learning algorithm and deep learning neural networks.

Machine learning

Machine learning is a subfield of artificial intelligence, which enables machines to learn from past data or experiences without being explicitly programmed.

It provides us statistical tools to explore or understanding the data.

ML is subfield of AI that focuses on the design system that can learn from and make decisions and predictions based on experience which is data in the case of machines.

ML enables computers to act and make data driven decisions rather than being explicitly programmed to carry out a certain task.

These programs are designed to learn and improve over time when exposed to new data.

Machine learning enables a computer system to make predictions or take some decisions using historical data without being explicitly programmed.

It can be divided into three types:

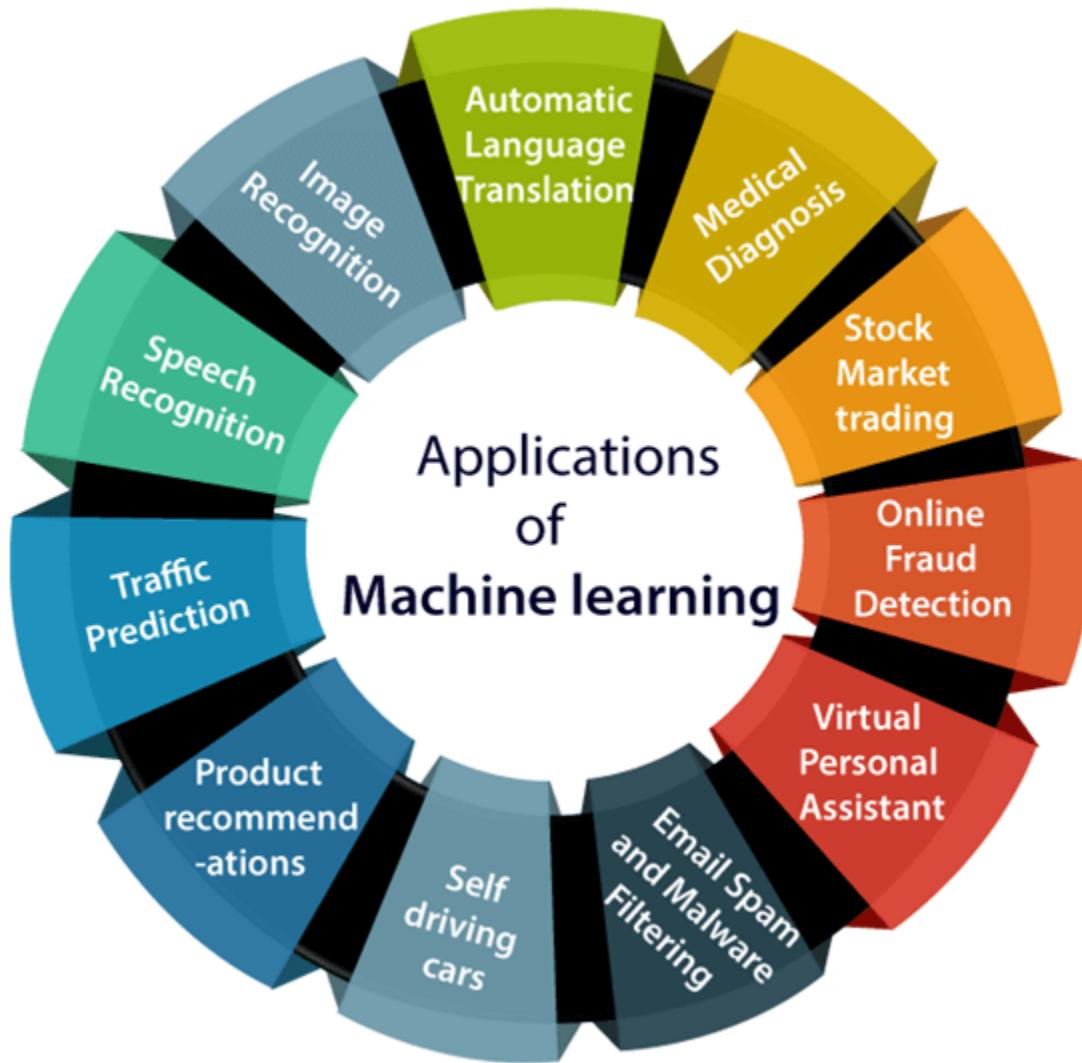
- **Supervised learning**
- **Reinforcement learning**
- **Unsupervised learning**

Key differences between Artificial Intelligence (AI) and Machine learning (ML):

Artificial Intelligence	Machine learning
Artificial intelligence is a technology which enables a machine to simulate human behavior.	Machine learning is a subset of AI which allows a machine to automatically learn from past data without programming explicitly.
The goal of AI is to make a smart computer system like humans to solve complex problems.	The goal of ML is to allow machines to learn from data so that they can give accurate output.
In AI, we make intelligent systems to perform any task like a human.	In ML, we teach machines with data to perform a particular task and give an accurate result.
Machine learning and deep learning are the two main subsets of AI.	Deep learning is a main subset of machine learning.
AI has a very wide range of scope.	Machine learning has a limited scope.
AI is working to create an intelligent system which can perform various complex tasks.	Machine learning is working to create machines that can perform only those specific tasks for which they are trained.
AI system is concerned about maximizing the chances of success.	Machine learning is mainly concerned about accuracy and patterns.
The main applications of AI are Siri, customer support using catboats, Expert System, Online	The main applications of machine learning are Online recommender system, Google search

game playing, intelligent humanoid robot, etc.	algorithms, Facebook auto friend tagging suggestions , etc.
On the basis of capabilities, AI can be divided into three types, which are, Weak AI , General AI , and Strong AI .	Machine learning can also be divided into mainly three types that are Supervised learning , Unsupervised learning , and Reinforcement learning .
It includes learning, reasoning, and self-correction.	It includes learning and self-correction when introduced with new data.
AI completely deals with Structured, semi-structured, and unstructured data.	Machine learning deals with Structured and semi-structured data.

Applications of Machine learning

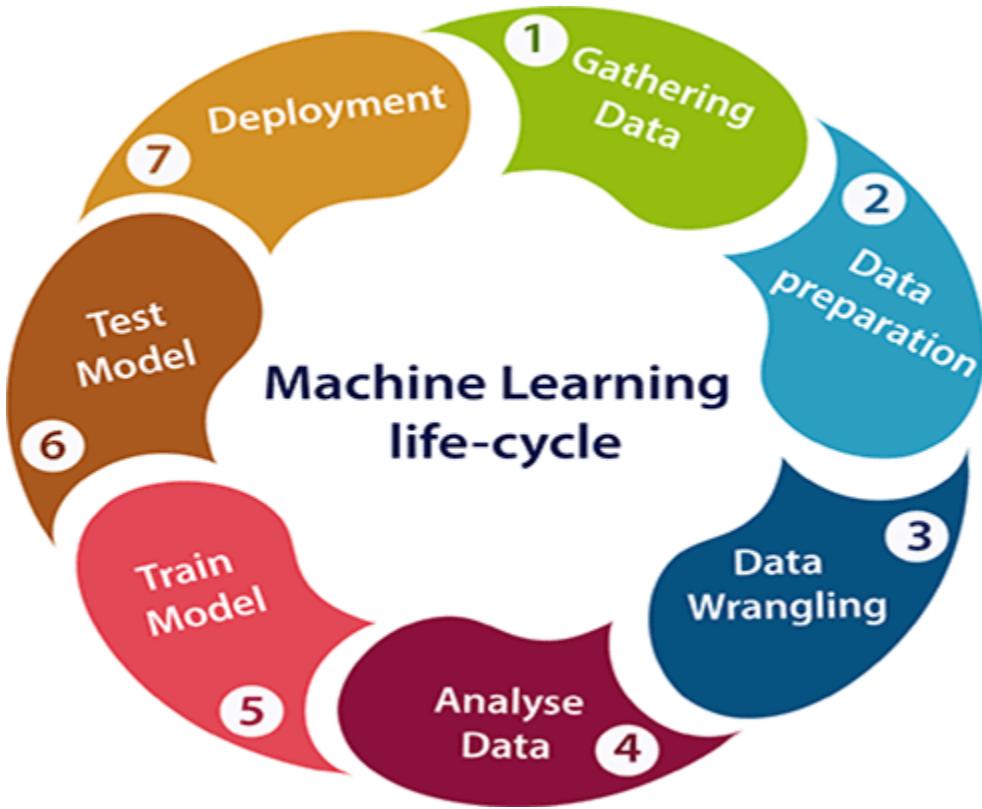


Machine learning Life cycle

Machine learning life cycle involves seven major steps, which are given below:

- **Gathering Data**
- **Data preparation**
- **Data Wrangling**
- **Analyse Data**
- **Train the model**

- **Test the model**
- **Deployment**



1. Gathering Data:

This step includes the below tasks:

- **Identify various data sources**
- **Collect data**
- **Integrate the data obtained from different sources**

2. Data preparation

This step can be further divided into two processes:

- **Data exploration:**

It is used to understand the nature of data that we have to work with. We need to understand the characteristics, format, and quality of data.

A better understanding of data leads to an effective outcome. In this, we find Correlations, general trends, and outliers.

- **Data pre-processing:**

Now the next step is preprocessing of data for its analysis.

3. Data Wrangling

Data wrangling is the process of cleaning and converting raw data into a useable format

- **Missing Values**
- **Duplicate data**
- **Invalid data**
- **Noise**

So, we use various filtering techniques to clean the data.

4. Data Analysis

Now the cleaned and prepared data is passed on to the analysis step. This step involves:

- **Selection of analytical techniques**
- **Building models**
- **Review the result**

5. Train Model

Now the next step is to train the model, in this step we train our model to improve its performance for better outcome of the problem.

We use datasets to train the model using various machine learning algorithms. Training a model is required so that it can understand the various patterns, rules, and, features.

6. Test Model

Once our machine learning model has been trained on a given dataset, then we test the model. In this step, we check for the accuracy of our model by providing a test dataset to it.

Testing the model determines the percentage accuracy of the model as per the requirement of project or problem.

7. Deployment

The last step of machine learning life cycle is deployment, where we deploy the model in the real-world system.

Difference between Data mining and Machine learning

Data mining:

The process of extracting useful information from a huge amount of data is called Data mining. Data mining is a tool that is used by humans to discover new, accurate, and useful patterns in data or meaningful relevant information for the ones who need it.

Machine learning:

The process of discovering algorithms that have improved courtesy of experience derived data is known as machine learning.

It is the algorithm that permits the machine to learn without human intervention.

It's a tool to make machines smarter, eliminating the human element.

S.No.	Data Mining	Machine Learning
1.	Extracting useful information from large amount of data	Introduce algorithm from data as well as from past experience
2.	Used to understand the data flow	Teaches the computer to learn and understand from the data flow
3.	Huge databases with unstructured data	Existing data as well as algorithms
4.	Models can be developed for using data mining technique	machine learning algorithm can be used in the decision tree, neural networks and some other area of artificial intelligence
5.	Human interference is more in it.	No human effort required after design

S.No.	Data Mining	Machine Learning
6.	It is used in cluster analysis	It is used in web Search, spam filter, fraud detection and computer design
7.	Data mining abstract from the data warehouse	Machine learning reads machine
8.	Data mining is more of a research using methods like machine learning	Self-learned and trains system to do the intelligent task
9.	Applied in limited area	Can be used in vast area

Example numerical on KNN classification

Name	Acid durability	Strength	Class
Type - 1	7	7	Bad
Type - 2	7	4	Bad
Type - 3	3	4	Good
Type - 4	1	4	Good

Test data \rightarrow Acid durability = 3

$$\text{Strength} = 7$$

$$\text{class} = ?$$

Name	Acid durability	Strength	Class	Distance
Type - 1	7	7	Bad	$\sqrt{(7-3)^2 + (7-7)^2} = 4$

Type - 2	7	4	Bad	$\sqrt{(7-3)^2 + (7-4)^2} = 5$
----------	---	---	-----	--------------------------------

Type - 3	3	4	Good	$\sqrt{(3-3)^2 + (7-4)^2} = 3$
----------	---	---	------	--------------------------------

Type - 4	1	4	Good	$\sqrt{(3-1)^2 + (7-4)^2} = 3.6$
----------	---	---	------	----------------------------------

Write the Rank for the records based on the distance between them.

Type Acid Strength class distance Rank

dura
bility

Type-1 7 7 Bad 4 3

-2 7 4 Bad 5 4

-3 3 4 Good 3 1

-4 1 4 Good 3.6 2

For.

K = 1

Test data belongs to class "Good".

K = 2

Test data belongs to class "Good".

$\mu = \frac{1}{4}(1+7+2+4) = 4$. $\sigma^2 = \frac{1}{4}[(1-4)^2 + (7-4)^2 + (2-4)^2 + (4-4)^2] = 3.5$

$\mu = \frac{1}{4}(1+7+2+4) = 4$. $\sigma^2 = \frac{1}{4}[(1-4)^2 + (7-4)^2 + (2-4)^2 + (4-4)^2] = 3.5$

$\mu = \frac{1}{4}(1+7+2+4) = 4$. $\sigma^2 = \frac{1}{4}[(1-4)^2 + (7-4)^2 + (2-4)^2 + (4-4)^2] = 3.5$

$\mu = \frac{1}{4}(1+7+2+4) = 4$. $\sigma^2 = \frac{1}{4}[(1-4)^2 + (7-4)^2 + (2-4)^2 + (4-4)^2] = 3.5$

Based above all the data with their results consider 3 methods all are

Linear Regression

- **Straight-line linear regression:**

- involves a response variable y and a single predictor variable x

$$y = w_0 + w_1 x$$

- w_0 : y -intercept
 - w_1 : slope
 - w_0 & w_1 are **regression coefficients**

Linear regression

- **Method of least squares**: estimates the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- D : a training set
- x : values of predictor variable
- y : values of response variable
- $|D|$: data points of the form $(x_1, y_1), (x_2, y_2), \dots, (x|D|, y|D|)$.
- \bar{x} : the mean value of $x_1, x_2, \dots, x|D|$
- \bar{y} : the mean value of $y_1, y_2, \dots, y|D|$

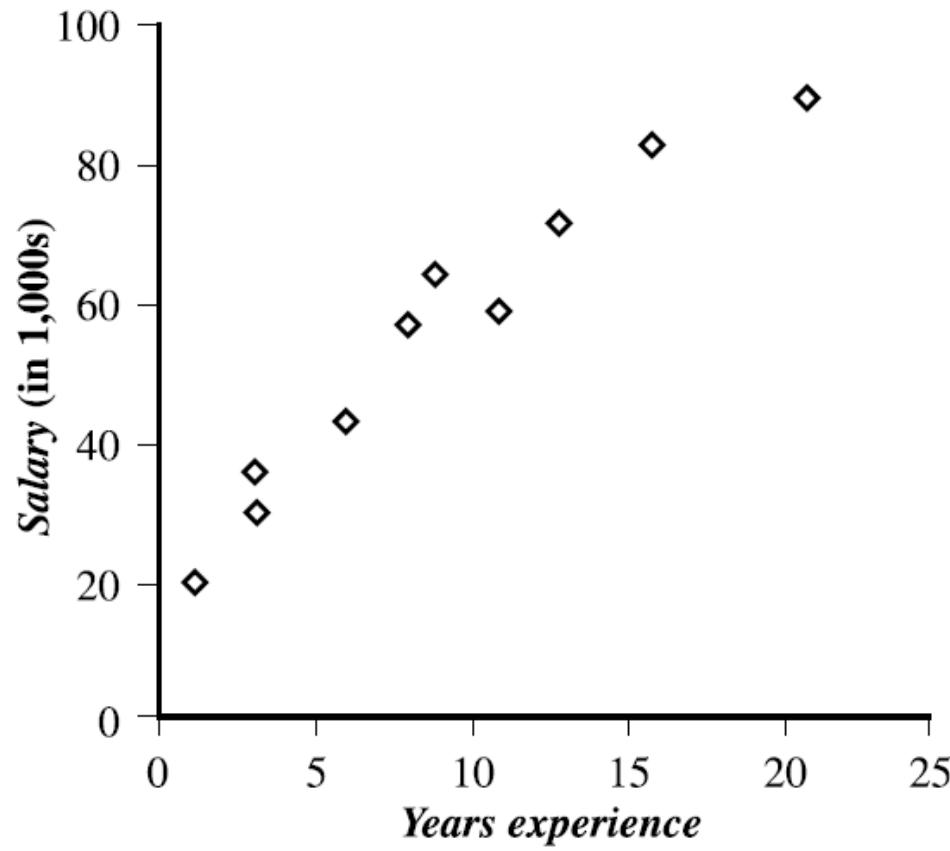
Example: Salary problem

- The table shows a set of paired data where x is the number of years of work experience of a college graduate and y is the corresponding salary of the graduate.

x years experience	y salary (in \$1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Linear Regression

- The 2-D data can be graphed on a **scatter plot**.
- The plot suggests a linear relationship between the two variables, x and y .



Example: Salary data

- Given the above data, we compute

$$\bar{x} = 9.1 \text{ and } \bar{y} = 55.4$$

- we get

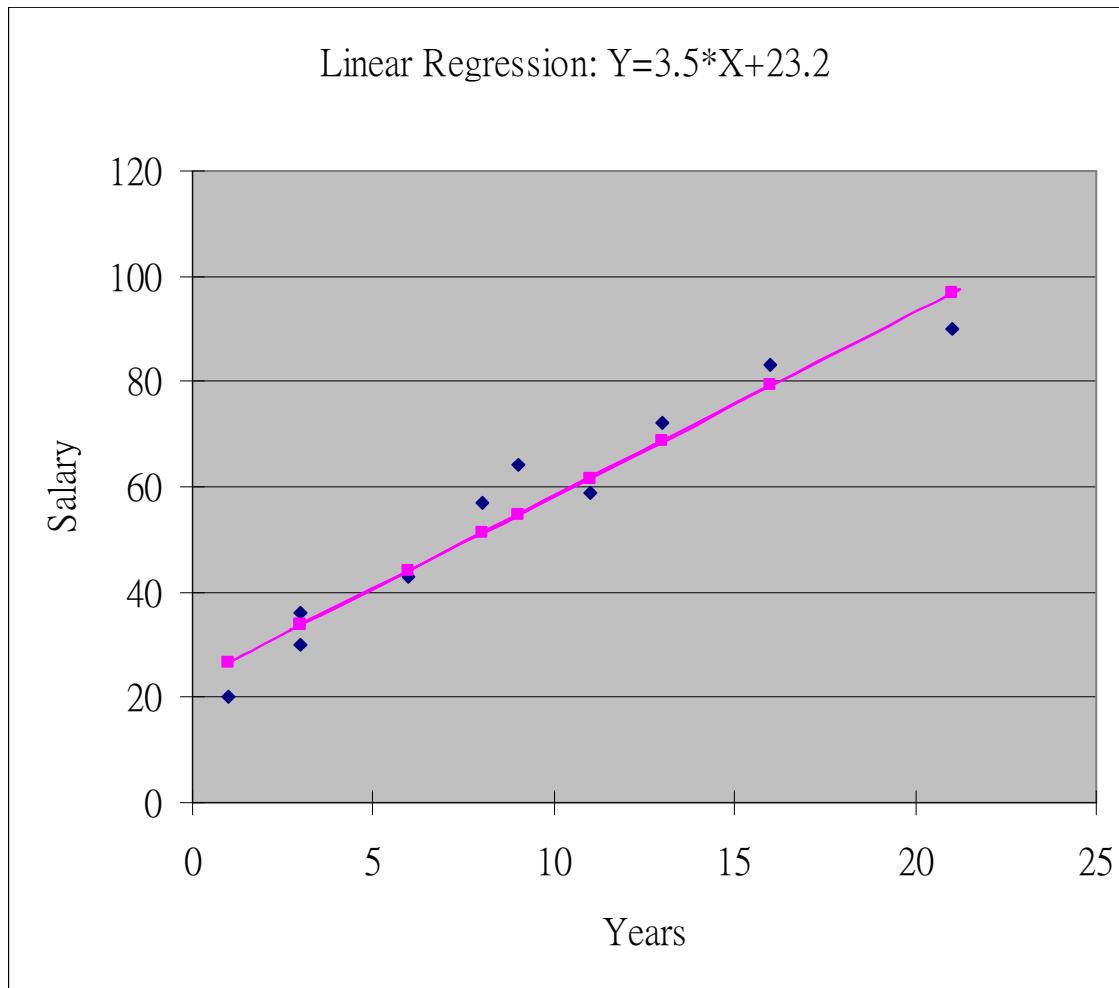
$$w_1 = \frac{(3 - 9.1)(30 - 55.4) + (8 - 9.1)(57 - 55.4) + \cdots + (16 - 9.1)(83 - 55.4)}{(3 - 9.1)^2 + (8 - 9.1)^2 + \cdots + (16 - 9.1)^2} = 3.5$$

$$w_0 = 55.4 - (3.5)(9.1) = 23.6$$

- The equation of the least squares line is estimated by

$$y = 23.6 + 3.5x$$

Example: Salary data





MODULE-2

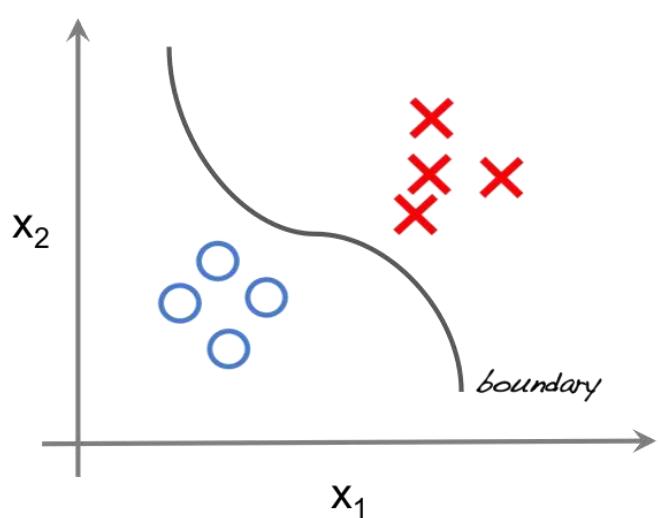
Supervised Learning

SUPERVISED LEARNING

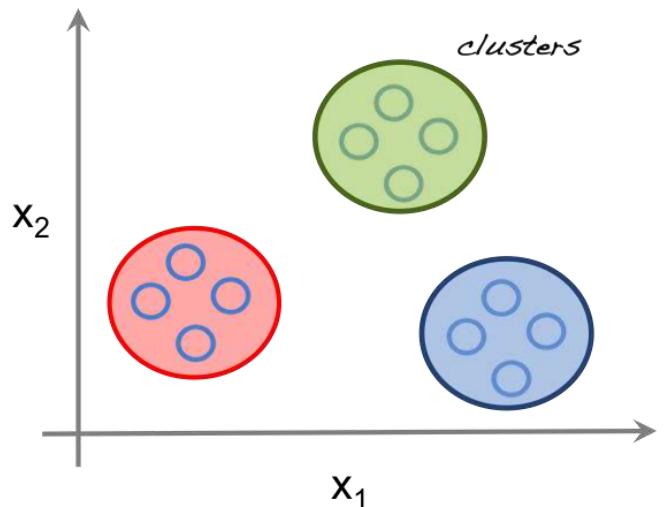
- In supervised learning, the labelled training data provide the basis for learning.
- The process of learning from the training data by a machine can be related to an expert supervising the learning process of a student.
- Here the expert is the training data.
- Training data is the past information with known value of class field or ‘label’.
- Unsupervised learning uses no labelled data.
- Semi-supervised learning uses a small amount of labelled data.

Supervised vs Unsupervised

Supervised learning



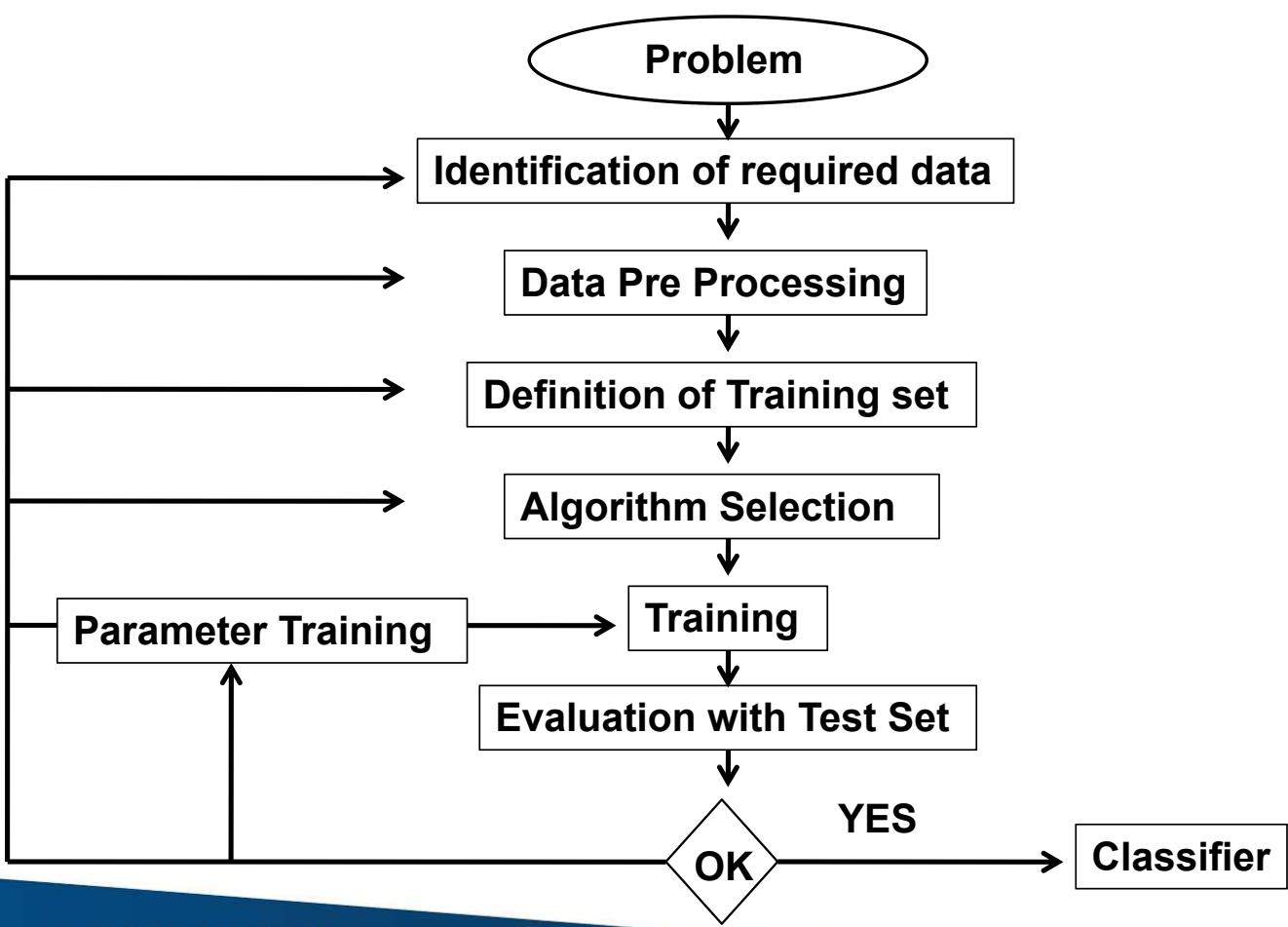
Unsupervised learning



Classification Model

- When we try to predict a categorical or nominal variable, the problem is known as a classification problem.
- Here, the problem centres around assigning a label or category or class to the test data on the basis of the label or category or class information imparted by training data.
- Classification is a type of supervised learning where a target feature, i.e. A categorical type, is predicted for test data on the basis of information obtained from training data.
- This categorical feature is known as class.

Classification Learning Steps



Common Classification Algorithms

1. k-Nearest Neighbour (kNN)
2. Decision tree
3. Random forest
4. Support Vector Machine (SVM)
5. Naive Bayes classifier

ORIGINS OF K-NN

- Nearest Neighbors have been used in statistical estimation and pattern recognition already in the beginning of 1970's (non-parametric techniques).
- The method prevailed in several disciplines and still it is one of the top 10 Data Mining algorithm.

IN A SENTENCE K-NN IS.....

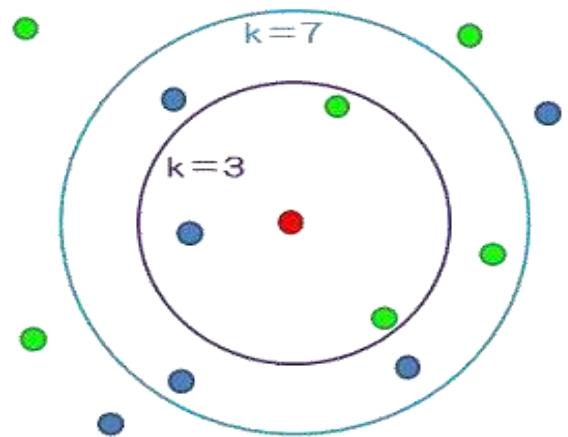
- It's how people judge by observing our peers.
- We tend to move with people of similar attributes so does data.



DEFINITION

- K-Nearest Neighbor is considered a lazy learning algorithm that classifies data sets based on their similarity with neighbors.
- “K” stands for number of data set items that are considered for the classification.

Ex: Image shows classification for different k-values.



TECHNICALLY..

- For the given attributes $A = \{X_1, X_2, \dots, X_D\}$ Where D is the dimension of the data, we need to predict the corresponding classification group $G = \{Y_1, Y_2, \dots, Y_n\}$ using the proximity metric over K items in D dimension that defines the closeness of association such that $X \in R^D$ and $Y_p \in G$.

THAT IS..

Attributes A

- Attribute A={Color, Outline, Dot}
- Classification Group,
G={triangle, square}
- D=3, we are free to choose K value.

#	Attribute			Shape
	Color	Outline	Dot	
1	green	dashed	no	triangle
2	green	dashed	yes	triangle
3	yellow	dashed	no	square
4	red	dashed	no	square
5	red	solid	no	square
6	red	solid	yes	triangle
7	green	solid	no	square
8	green	dashed	no	triangle
9	yellow	solid	yes	square
10	red	solid	no	square
11	green	solid	yes	square
12	yellow	dashed	yes	square
13	yellow	solid	no	square
14	red	dashed	yes	triangle

C
l
a
s
s
i
f
i
c
a
t
i
o
n

G
r
o
u
p

PROXIMITY METRIC

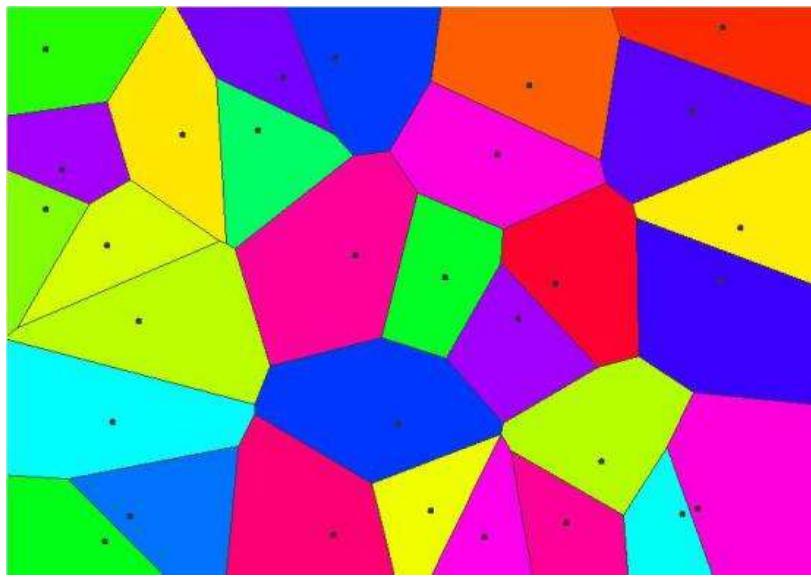
- Definition: Also termed as “Similarity Measure” quantifies the association among different items.
- Following is a table of measures for different data items:

Similarity Measure	Data Format
Contingency Table, Jaccard coefficient, Distance Measure	Binary
Z-Score, Min-Max Normalization, Distance Measures	Numeric
Cosine Similarity, Dot Product	Vectors

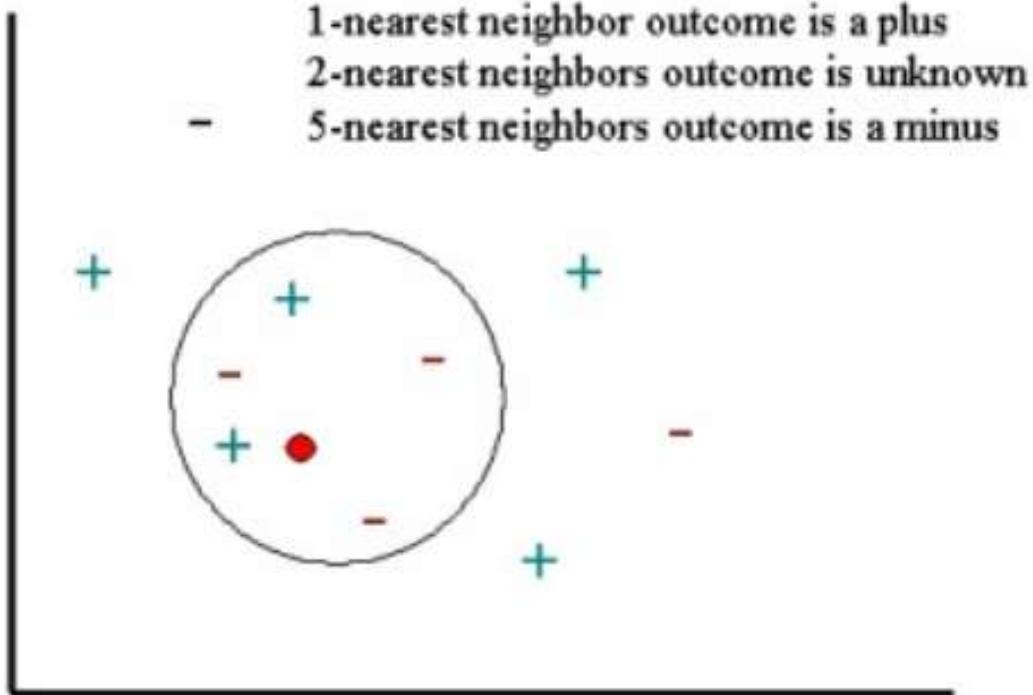


Voronoi diagram

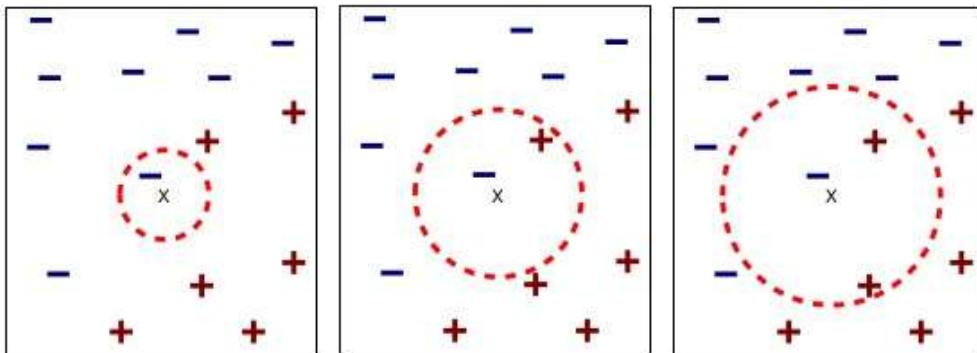
- A Voronoi diagram is a partitioning of a plane into regions based on distance to points in a specific subset of the plane.
- Here, $k=1$.



K-NN Example



K-NN Example



(a) 1-nearest neighbor

(b) 2-nearest neighbor

(c) 3-nearest neighbor

K-nearest neighbors of a record x are data points
that have the k smallest distance to x

PROXIMITY METRIC

- For the numeric data let us consider some distance measures:

- Manhattan Distance:

$$X = \langle x_1, x_2, \dots, x_n \rangle \quad Y = \langle y_1, y_2, \dots, y_n \rangle$$

$$dist(X, Y) = |x_1 - y_1| + |x_2 - y_2| + \dots + |x_n - y_n|$$

- Ex: Given X = {1,2} & Y = {2,5}

$$\begin{aligned}\text{Manhattan Distance} &= dist(X, Y) = |1-2| + |2-5| \\ &= 1 + 3 \\ &= 4\end{aligned}$$

PROXIMITY METRIC

- Euclidean Distance:

$$X = \langle x_1, x_2, \dots, x_n \rangle \quad Y = \langle y_1, y_2, \dots, y_n \rangle$$

$$dist(X, Y) = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

- Ex: Given $X = \{-2, 2\}$ & $Y = \{2, 5\}$
Euclidean Distance

K-NN IN ACTION

- Consider the following data:
 $A = \{\text{weight}, \text{color}\}$
 $G = \{\text{Apple}(A), \text{Banana}(B)\}$
- We need to predict the type of a fruit with:
weight = 378
color = red

weight (g)	color	Type of fruit
303	3	Banana
370	1	Apple
298	3	Banana
277	3	Banana
377	4	Apple
299	3	Banana
382	1	Apple
374	4	Apple
303	4	Banana
309	3	Banana
359	1	Apple
366	1	Apple
311	3	Banana
302	3	Banana
373	4	Apple
305	3	Banana
371	3	Apple

SOME PROCESSING..

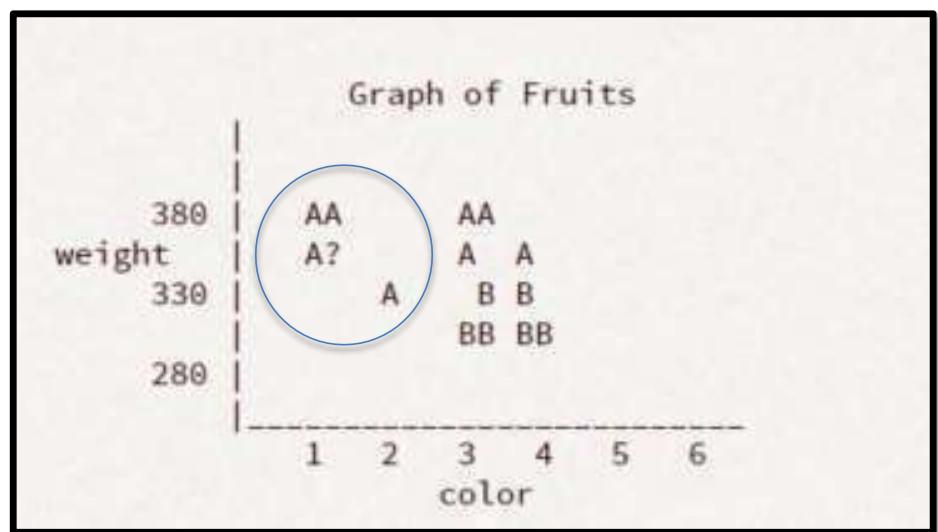
- Assign color codes to convert into numerical data:

red	1
orange	2
yellow	3
green	4
blue	5
purple	6

- Let's label Apple as "A" and Banana as "B"

PLOTTING

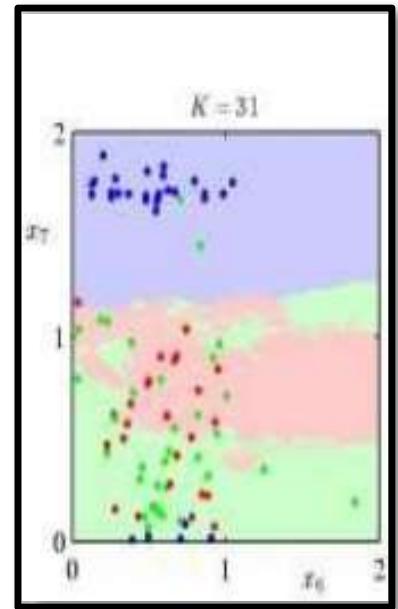
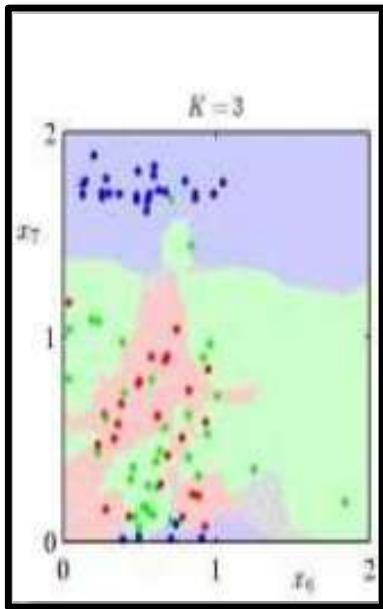
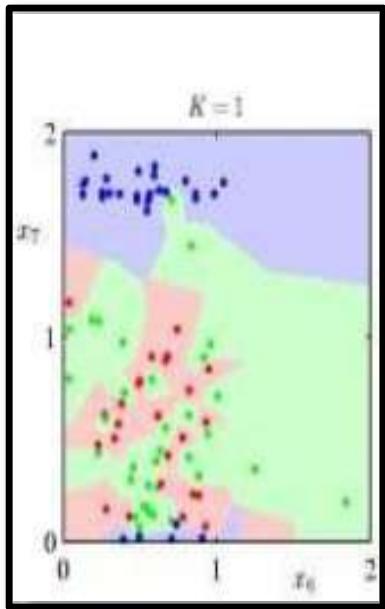
- Using K=3,
Our result will be,



AS 'K' VARIES....

- Clearly, K has an impact on the classification.

Can you guess?



K-NN PROPERTIES

- K-NN is a lazy algorithm
- The processing defers with respect to K value.
- Result is generated after analysis of stored data.
- It neglects any intermediate values.

REMARKS: FIRST THE GOOD

Advantages

- Can be applied to the data from any distribution, for example, data does not have to be separable with a linear boundary
- Very simple and intuitive
- Good classification if the number of samples is large enough

NOW THE BAD....

Disadvantages

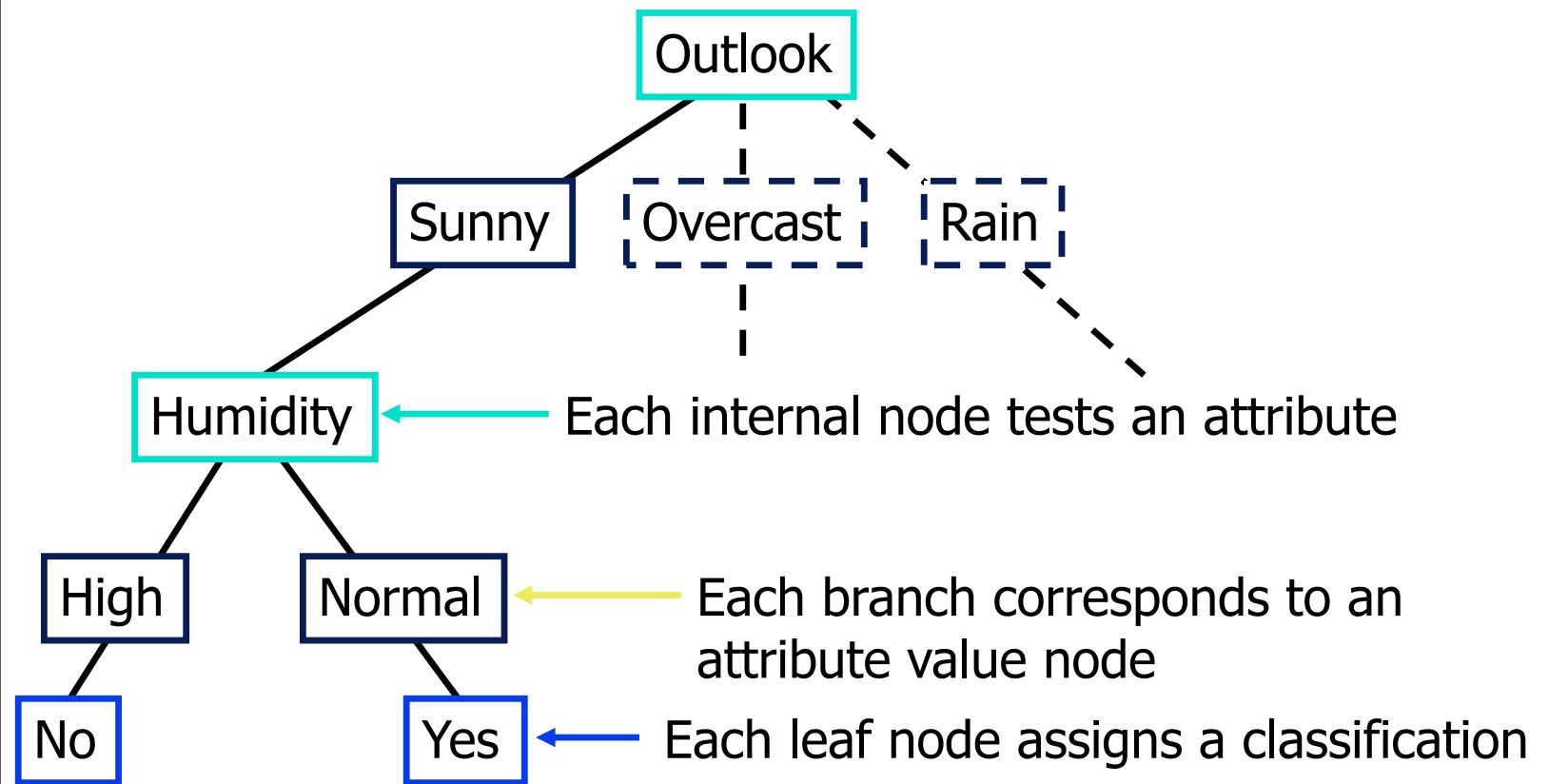
- Dependent on K Value
- Test stage is computationally expensive
- No training stage, all the work is done during the test stage
- This is actually the opposite of what we want. Usually we can afford training step to take a long time, but we want fast test step.
- Need large number of samples for accuracy

DECISION TREE

-
- This is one of the most adopted algorithms for classification.
 - It builds a model in the form of a tree structure.
 - A decision tree is used for multi-dimensional analysis with multiple classes and is characterized by ease of interpretation of rules and fast execution.
 - The goal of decision tree learning is to create a model that predicts the value of the output variable based on the input variables in the feature vector.
 - It contains a decision node and a leaf node.
 - Each decision node corresponds to one of the feature vector.

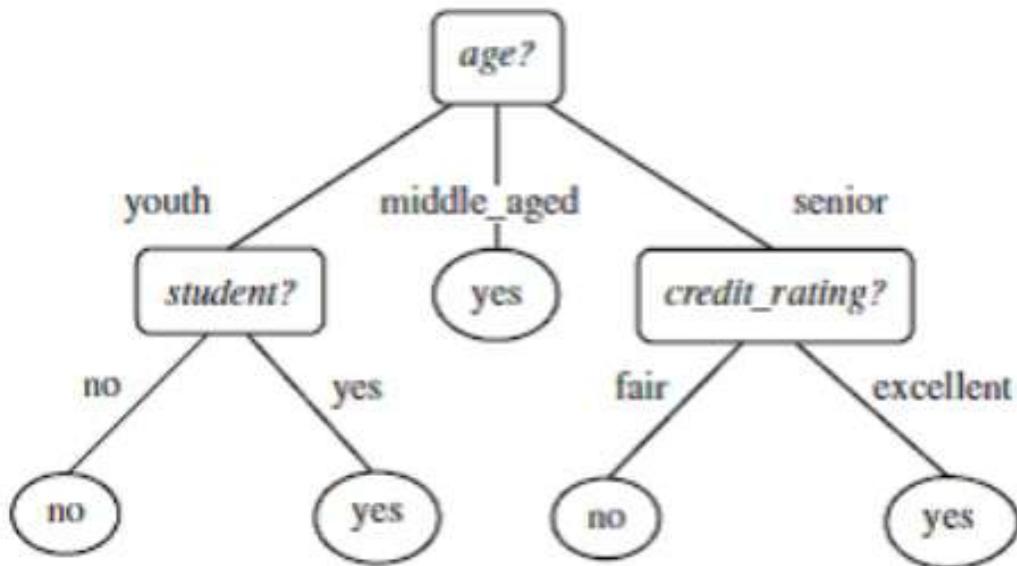
-
- From every node, there are edges to children, wherein there is an edge for each of the possible values of the feature associated with the node.
 - The output variable is determined by following a path that starts at the root and is guided by the values of the input variables.
 - Decision trees can be used for both classification and regression.

Decision Tree for PlayTennis



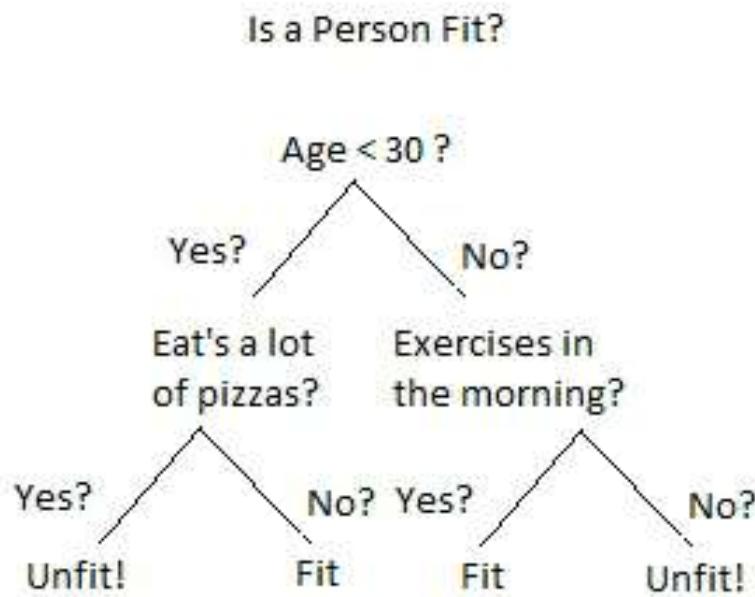
Example-1

- Will a person buy a computer?



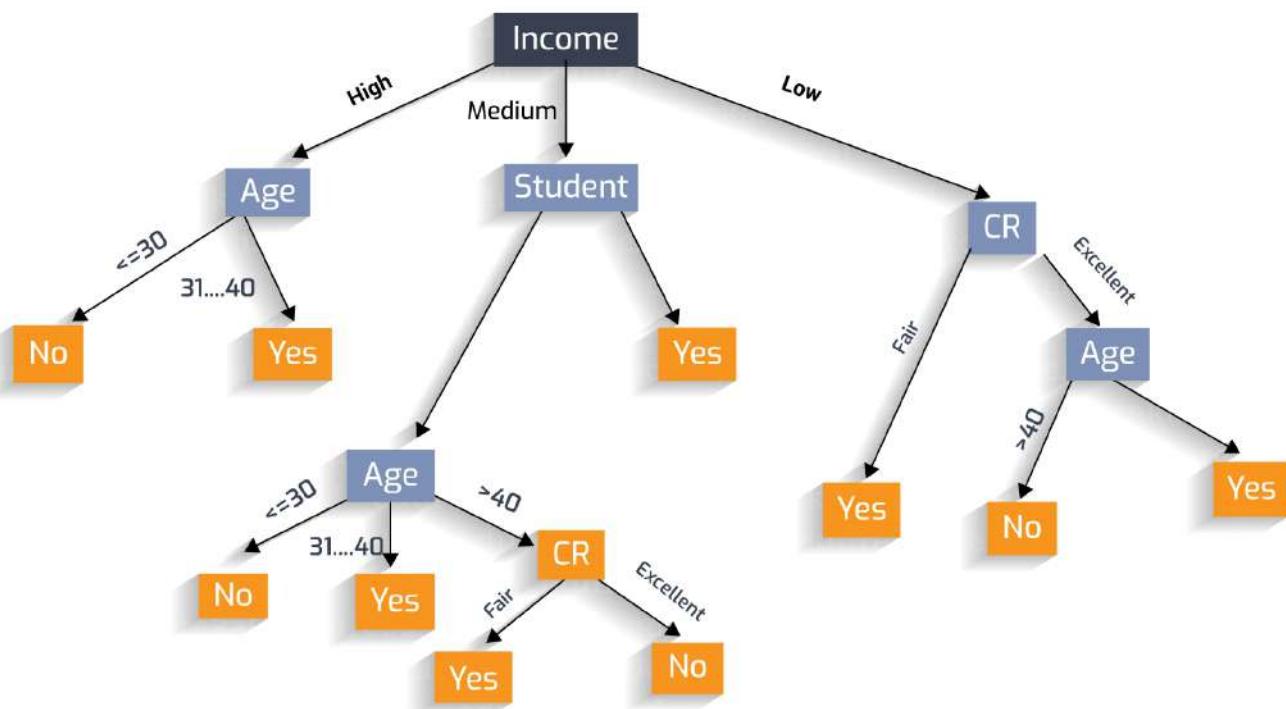
Example-2

- Is a person fit?



Example-3

- Should the LOAN be sanctioned?



Training Data for GTS recruitment

CGPA	Communication	Aptitude	Programming Skills	Job Offered?
High	Good	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	Low	Good	No
Low	Good	Low	Bad	No
High	Good	High	Bad	Yes
High	Good	High	Good	Yes
Medium	Bad	Low	Bad	No
Medium	Bad	Low	Good	No
High	Bad	High	Good	Yes
Medium	Good	High	Good	Yes
Low	Bad	High	Bad	No
Low	Bad	High	Bad	No
Medium	Good	High	Bad	Yes
Low	Good	Low	Good	No
High	Bad	Low	Bad	No
Medium	Bad	High	Good	No
High	Bad	Low	Bad	No
Medium	Good	High	Bad	Yes

Entropy of a decision tree

- Entropy, as it relates to machine learning, is a measure of the randomness in the information being processed.
- The higher the entropy, the harder it is to draw any conclusions from that information.

$$Entropy = \sum_{i=1}^C -p_i * \log_2(p_i)$$

- Ex: For class ‘Job Offered?’ we have two values: Yes and No.
- Pi values for Yes= 8/18 = 0.44 & No= 10/18= 0.56
- Entropy(S) = $-0.44 \log_2(0.44) - 0.56 \log_2(0.56)$
= 0.99

Information gain of a decision tree

- The information gain is created on the basis of the decrease in entropy(S) after a data set is split according to a particular attribute(A).
- Constructing a decision tree is all about finding an attribute that returns the highest information gain.
- If information gain is 0, it means that there is no reduction in entropy due to split of the data set according to that particular feature.
- The maximum amount of information gain which may happen is the entropy of the data set before the split.

-
- Information gain for a particular feature A is calculated by the difference in entropy before a split(S_{bs}) with the entropy after the split(S_{as}).
 - Information gain(S, A) = Entropy(S_{bs}) – Entropy(S_{as})
 - For weighted summation, the proportion of examples falling into each partition is used as weight.
 - Entropy(S_{as}) = $\sum (i=1 \text{ to } n) w_i \text{ Entropy}(p_i)$

a) Original data set

	Yes	No	Total
Count	8	10	18
pi	0.44	0.56	
-pi*LOG(pi)	0.52	0.47	0.99

Total Entropy = 0.99

b) Splitted data set(based on the CGPA)

CGPA = High			CGPA = Medium			CGPA = Low					
	Yes	No	Total		Yes	No	Total		Yes	No	Total
Count	4	2	6	Count	4	3	7	Count	0	5	5
pi	0.67	0.33		pi	0.57	0.43		pi	0	1	
-pi*LOG(pi)	0.39	0.53	0.92	-pi*LOG(pi)	0.46	0.52	0.99	-pi*LOG(pi)	0	0	0

$$\text{Total Entropy} = (6/18 * 0.92 + 7/18 * 0.99 + 5/18 * 0) \\ = 0.69$$

$$\text{Information Gain} \\ = 0.99 - 0.69 = 0.30$$

c) Splitted data set(based on ‘Communication’)

Communication = ‘Good’

Communication = ‘Bad’

Total Entropy = 0.63

Information Gain = 0.36

d) Splitted data set(based on ‘Aptitude’)

Aptitude = ‘High’

Aptitude = ‘Low’

Total Entropy = 0.52

Information Gain = 0.47(Entropy=0)

e) Splitted data set(based on ‘Programming Skills’)

Programming Skills = ‘Good’

Programming Skills = ‘Bad’

Total Entropy = 0.95

Information Gain = 0.04

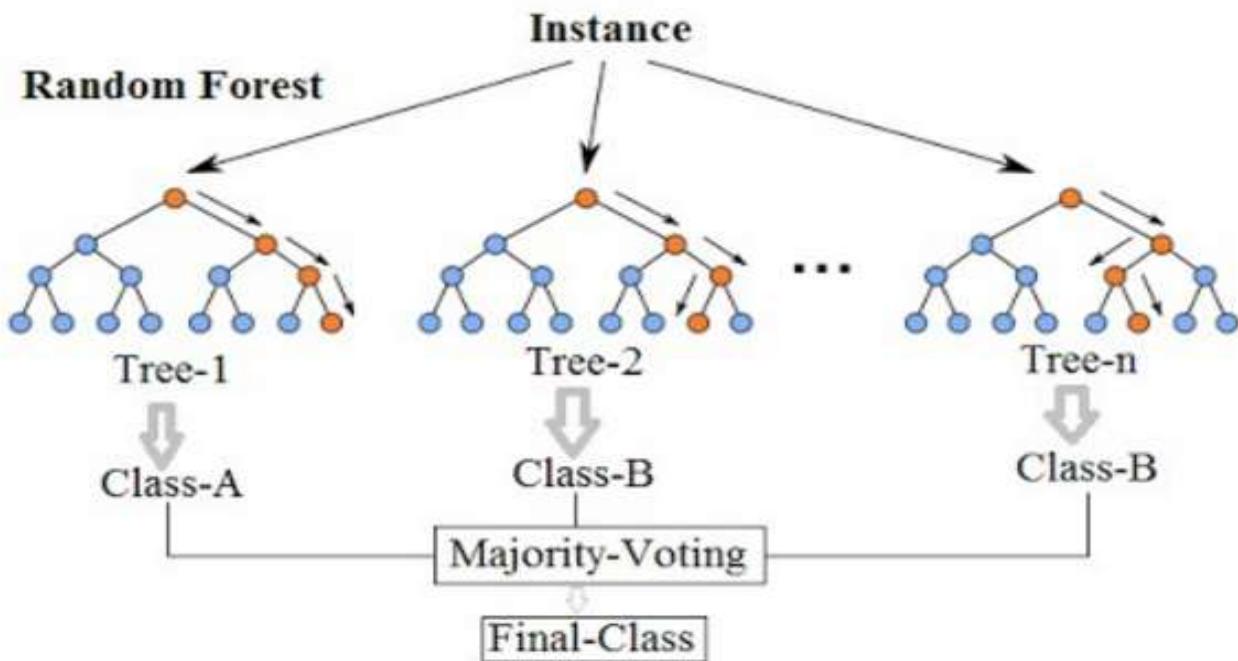
Avoiding overfitting in decision tree- pruning

- The decision tree algorithm, unless a stopping criterion is applied, may keep growing indefinitely.
- To prevent a decision tree getting overfitted to the training data, pruning of the decision tree is essential.
- Pruning a decision tree reduces the size of the tree such that the model is more generalized and can classify unknown and unlabeled data in a better way.
- Pre-pruning: Stop growing the tree before it reaches perfection.
- Post-pruning: Allow the tree to grow entirely and then post-prune some of the branches from it.

Random Forest Model

- It is an ensemble classifier, i.e., a combining classifier that uses and combines many decision tree classifiers.
- Ensembling is usually done using the concept of bagging with different feature sets.
- The reason for using large number of trees in random forest is to train the trees enough such that contribution from each feature comes in a number of models.
- After the random forest is generated by combining the trees, majority vote is applied to combine the output of the different trees.
- Ensembled model yields better result than decision trees.

Random Forest Simplified



Random forest algorithm

- The algorithm works as follows:
 1. If there are N variables or features in the input data set, select a subset of 'm' ($m < N$) features at random out of the N features.
 2. Use the best split principle on these 'm' features to calculate the number of nodes 'd'.
 3. Keep splitting the nodes to child nodes till the tree is grown to maximum possible extent.
 4. Select a different subset of the training data 'with replacement' to train another DT with steps (1) to (3). Repeat this to build and train 'n' decision trees.
 5. Final class assignment is done on the basis of the majority votes from the 'n' trees.

Strengths of RF

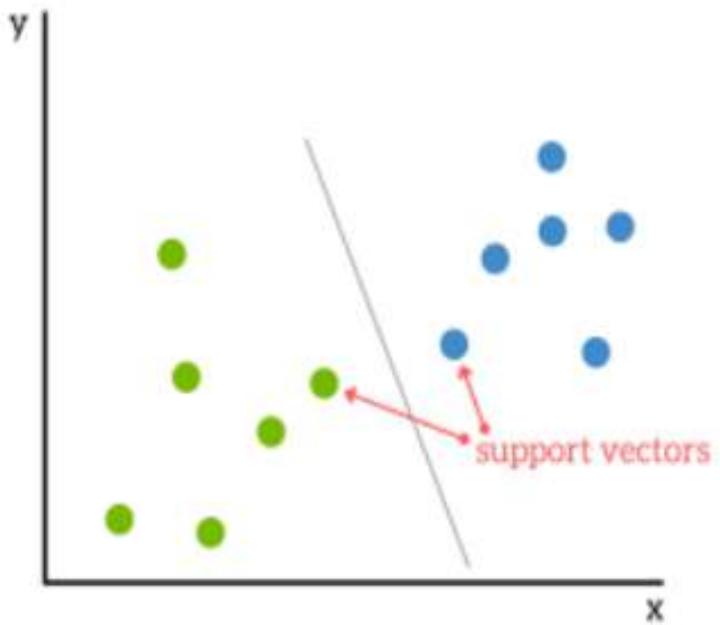
- It runs efficiently on large and expensive data sets.
- It has a robust method for estimating missing data and maintains precision when a large proportion of data is absent.
- It has powerful techniques for balancing errors in a class population of unbalanced data sets.
- It gives estimates about which features are the most important ones in the overall classification.

Drawback of RF

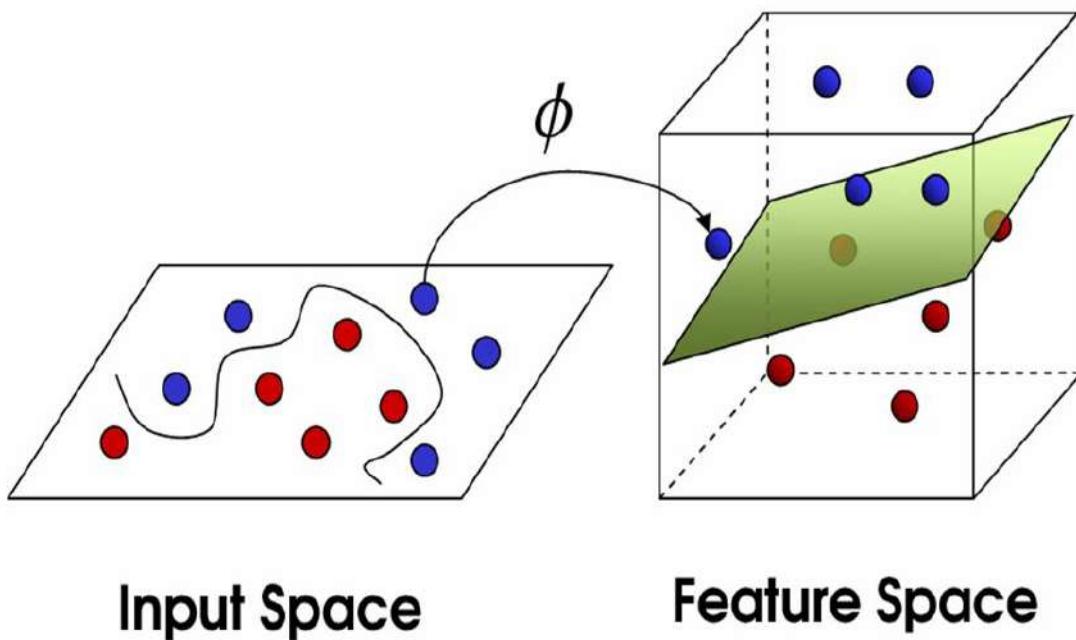
- As it combines many decision trees, it is not easy to understand as a decision tree model.
- Computationally, it is much more expensive than a simple decision tree.

Support Vector Machine

- SVM is a model which can perform linear classification as well as regression.
- It is based on the concept of a surface called hyperplane, which draws a boundary between data instances plotted on a multi-dimensional feature space.
- The output prediction is one of the two classes defined in the training data.

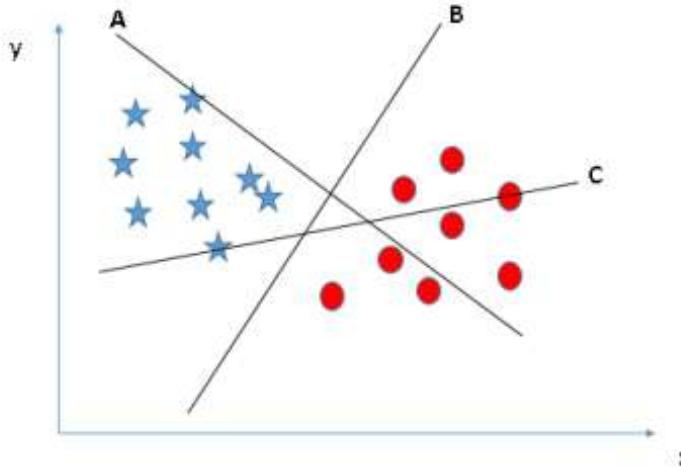


Principle of Support Vector Machines (SVM)



Identify the right hyper-plane (Scenario-1)

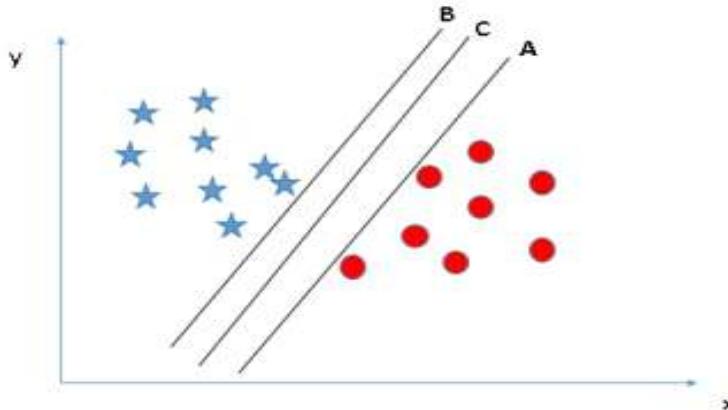
- Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.
- Select the hyper-plane which segregates the two classes better?



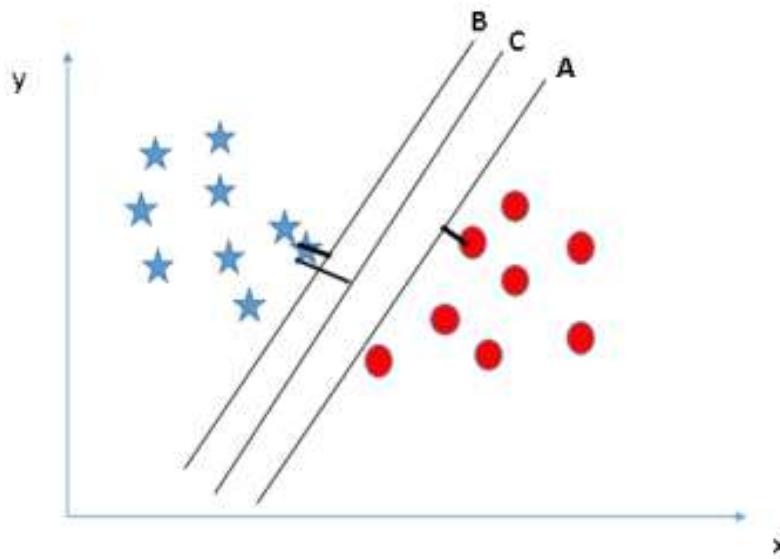
- In this scenario, hyper-plane “B” has excellently performed this job

Identify the right hyper-plane (Scenario-2)

- Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane?

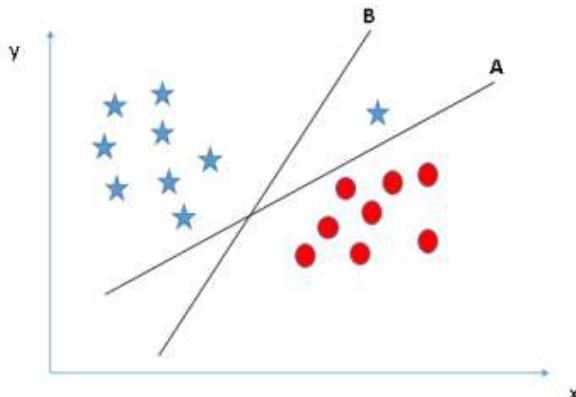


- Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin.



- We can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C.

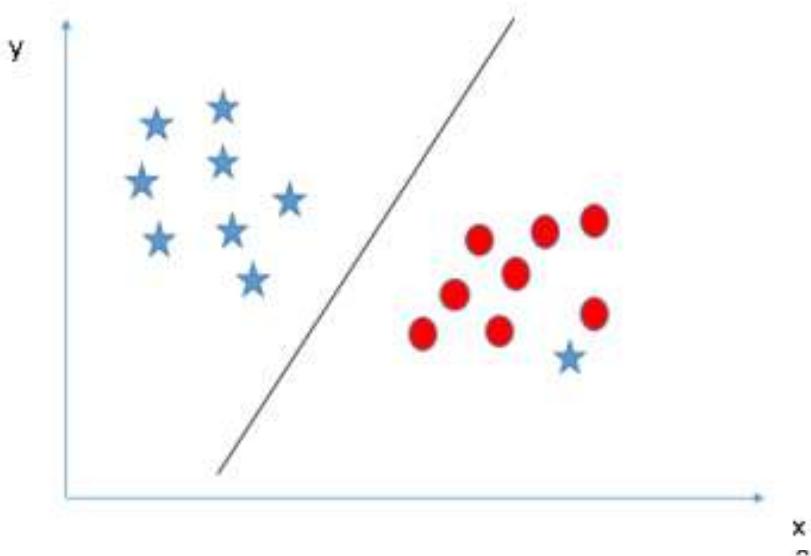
Scenario-3



- Here hyper-plane B as it has higher margin compared to A.
- SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin.
- Hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

Scenario-4

- Here, we are unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.
- SVM has a feature to ignore outliers and find the hyperplane that has maximum margin. SVM is robust to outliers.



Strengths of SVM

- SVM can be used for both classification & regression.
- It is robust, i.e. not much impacted by data with noise or outliers.
- The prediction results using this model are very strong.

Weakness of SVM

- SVM is applicable only for binary classification, i.e. when there are only two classes in the problem.
- While dealing with high dimensional data, it becomes very complex.
- It is slow for large dataset, i.e. a data set with more features or instances.
- It is memory-intensive(throughput is bounded by the device memory bandwidth).

Introduction to Regression

- Regression is a technique used to model and analyze the relationships between variables and often times how they contribute and are related to producing a particular outcome together.
- Here, dependent variable (Y) is the one whose value is to be predicted, ex.- the price quote of the real estate property.
- This variable is presumed to be functionally related to one (X) or more independent variables called predictors.
- $Y = f(X)$

Common Regression Algorithms

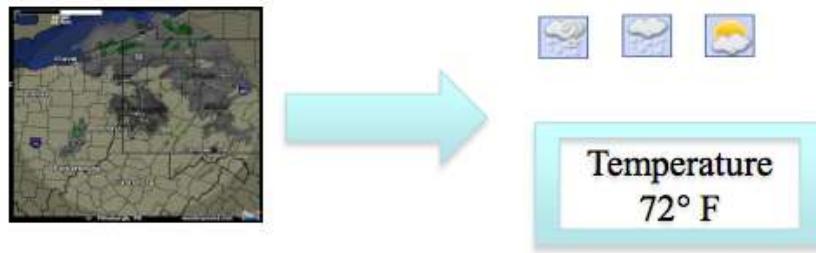
- The common regression algorithms are:
 1. Simple linear regression
 2. Multiple linear regression
 3. Polynomial regression
 4. Multivariate adaptive regression splines
 5. Logistic regression
 6. Maximum likelihood estimation(least square)

Regression examples

Stock market

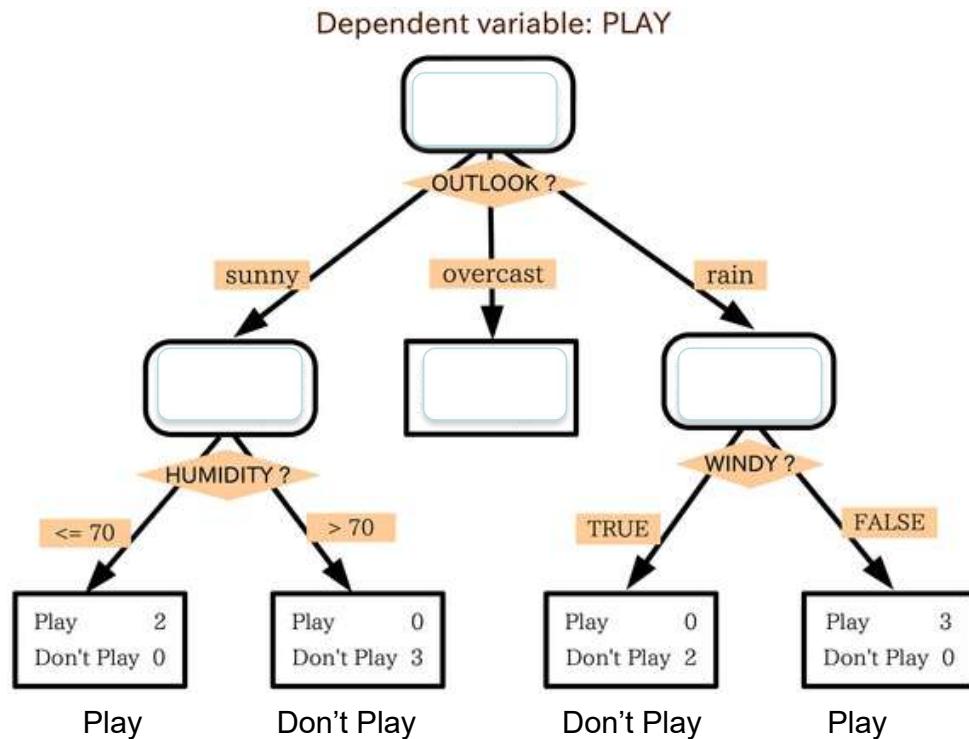


Weather prediction

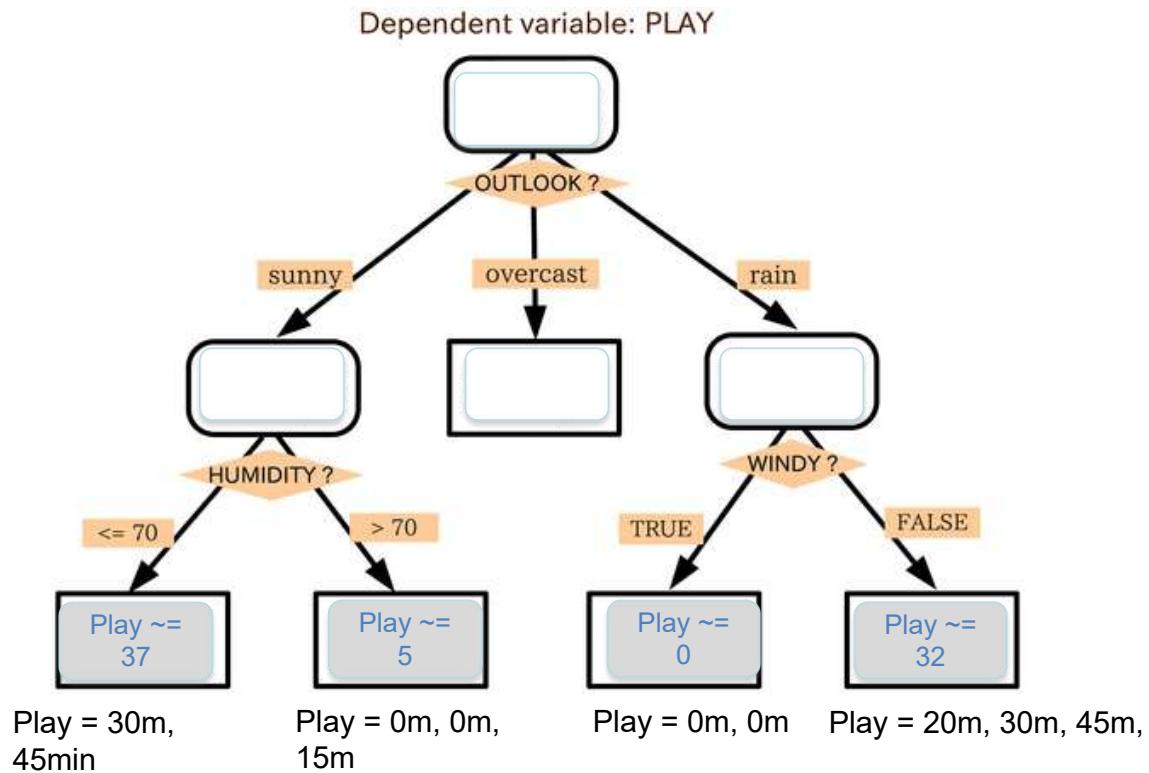


Predict the temperature at any given location

A decision tree: classification

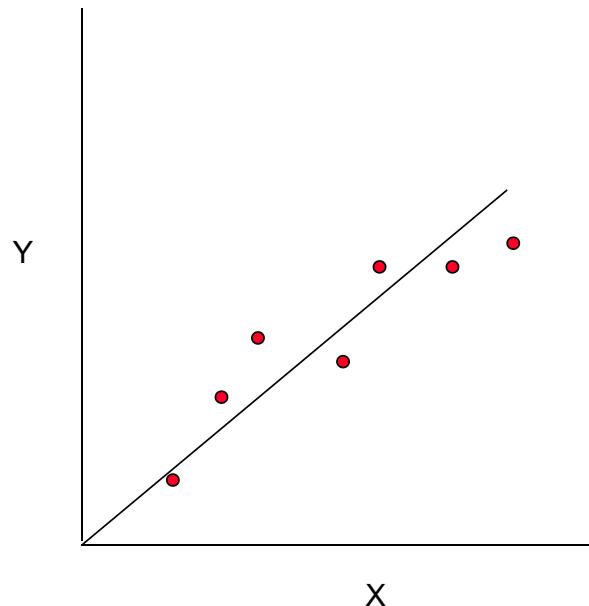


A regression tree



Linear regression

- Given an input x we would like to compute an output y
- For example:
 - Predict height from age
 - Predict Google's price from Yahoo's price
 - Predict distance from wall from sensors



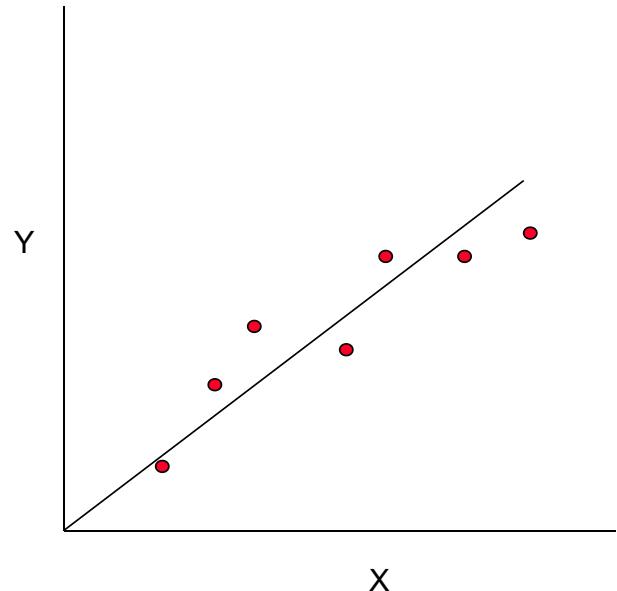
Linear regression

- Given an input x we would like to compute an output y
- In linear regression we assume that y and x are related with the following equation:

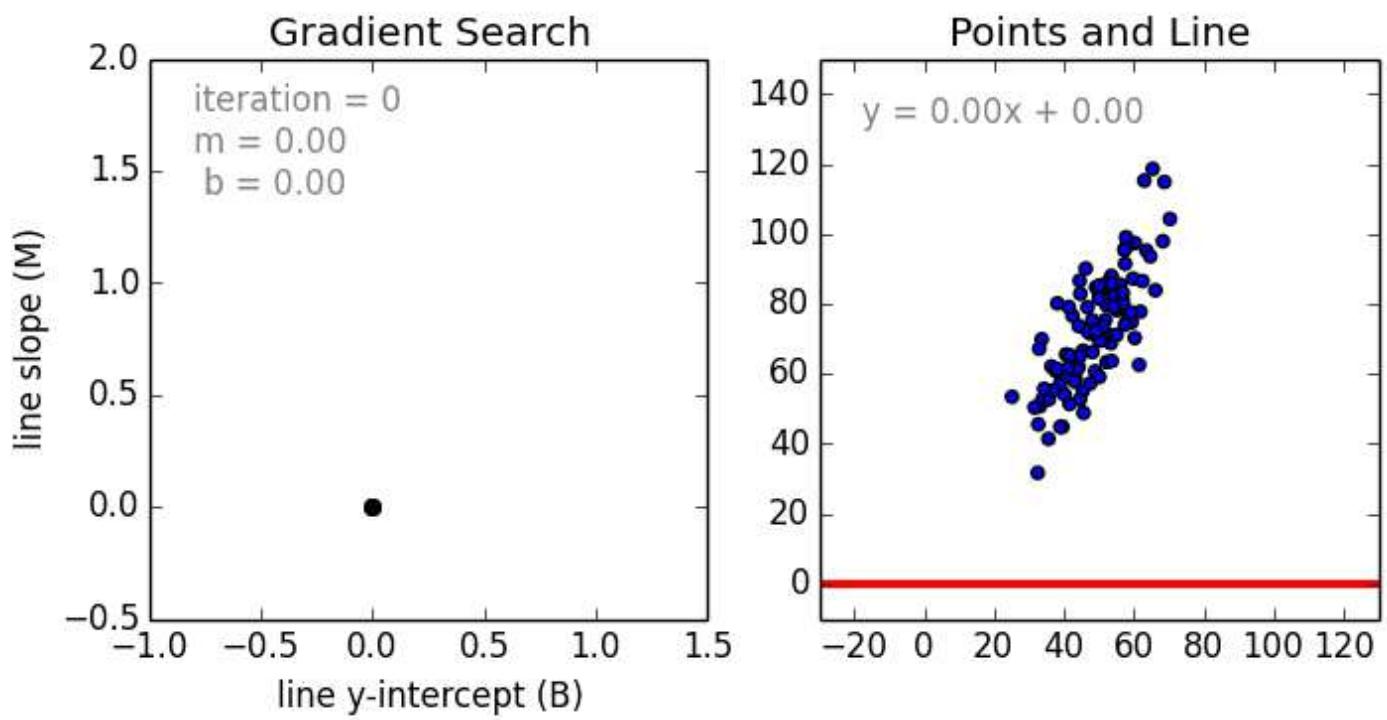
$$y = wx + \varepsilon$$

What we are trying to predict

Observed values



where w is a parameter and ε represents measurement noise, model noise or other (data) noise



Linear regression

- Our goal is to estimate w from a training data of $\langle x_i, y_i \rangle$ pairs

$$y = WX + \varepsilon$$

- Optimization goal: minimize squared error (least squares):

$$\arg \min_w \sum_i (y_i - wx_i)^2$$

- Why least squares?

- minimizes squared distance between measurements and predicted line

- has a nice probabilistic interpretation (Gaussian Likelihood same as Mean Sq.)

- first degree polynomial model



Multivariate regression

- What if we have several inputs?
 - Stock prices for Yahoo, Microsoft and Ebay for the Google prediction task
- This becomes a multivariate regression problem
- Again, it's easy to model:

$$y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$$

The diagram illustrates the components of a multivariate regression equation. At the top is the equation $y = w_0 + w_1x_1 + \dots + w_kx_k + \varepsilon$. Below the equation, three arrows point from boxes containing stock price data to specific terms: one arrow points from a box labeled "Google's stock price" to the term w_1x_1 ; another arrow points from a box labeled "Microsoft's stock price" to the term w_kx_k ; and a third arrow points from a box labeled "Yahoo's stock price" to the error term ε .

-
- Price of property = f (Area, location, floor, ageing, amenities)
 - $Y = a + b_1 X_1 + b_2 X_2$

Where Y is the three-dimensional space, X_1 & X_2 are the predictor variables, b_1 & b_2 are referred as partial regression coefficients.

Assumptions in Regression Analysis

1. The dependent variable(Y) can be calculated as a linear function of a specific set of independent variables(X) and an error term(ϵ).
2. The number of observations(n) is greater than the number of parameters(k) to be estimated, ie $n>k$.
3. Regression line can be valid only over a limited range of data.
4. Variance is the same for all values of X.
5. The error term(ϵ) is normally distributed.
6. The values of the error term(ϵ) are independent and are not related to any values of X.

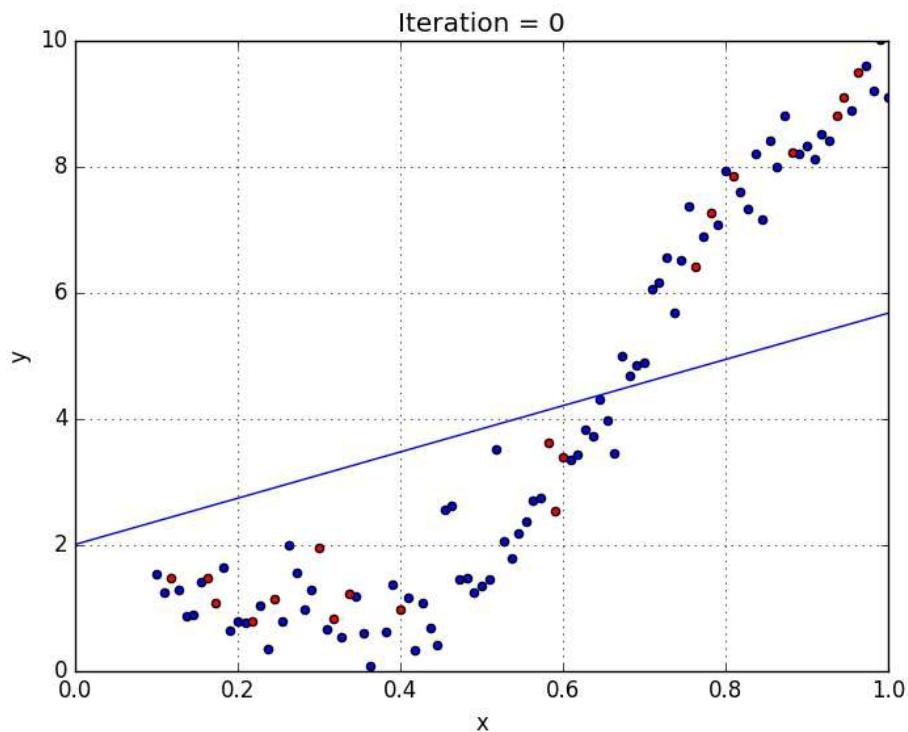
The OLS(Ordinary least square) estimator is the Best Linear Unbiased Estimator(BLUE) and this is called as Gauss-Markov Theorem.

Polynomial regression

Polynomial regression model is the extension of the simple linear model by adding extra predictors obtained by raising(squaring) each of the original predictors to a power.

If there are three variable, X , X^2 , X^3 are used as predictors.

$$F(x) = C_0 + C_1 X^1 + C_2 X^2 + C_3 X^3$$



Let us use the below data set of (X, Y) for degree 3 polynomial

Internal Exam(X)	15	23	18	23	24	22	22	19	19	16	24	11	24	16	23
External Exam(X)	49	63	58	60	58	61	60	63	60	52	62	30	59	49	68

The regression line is slightly curved for degree = 3.

The regression line will curve further if we increase the polynomial degree.

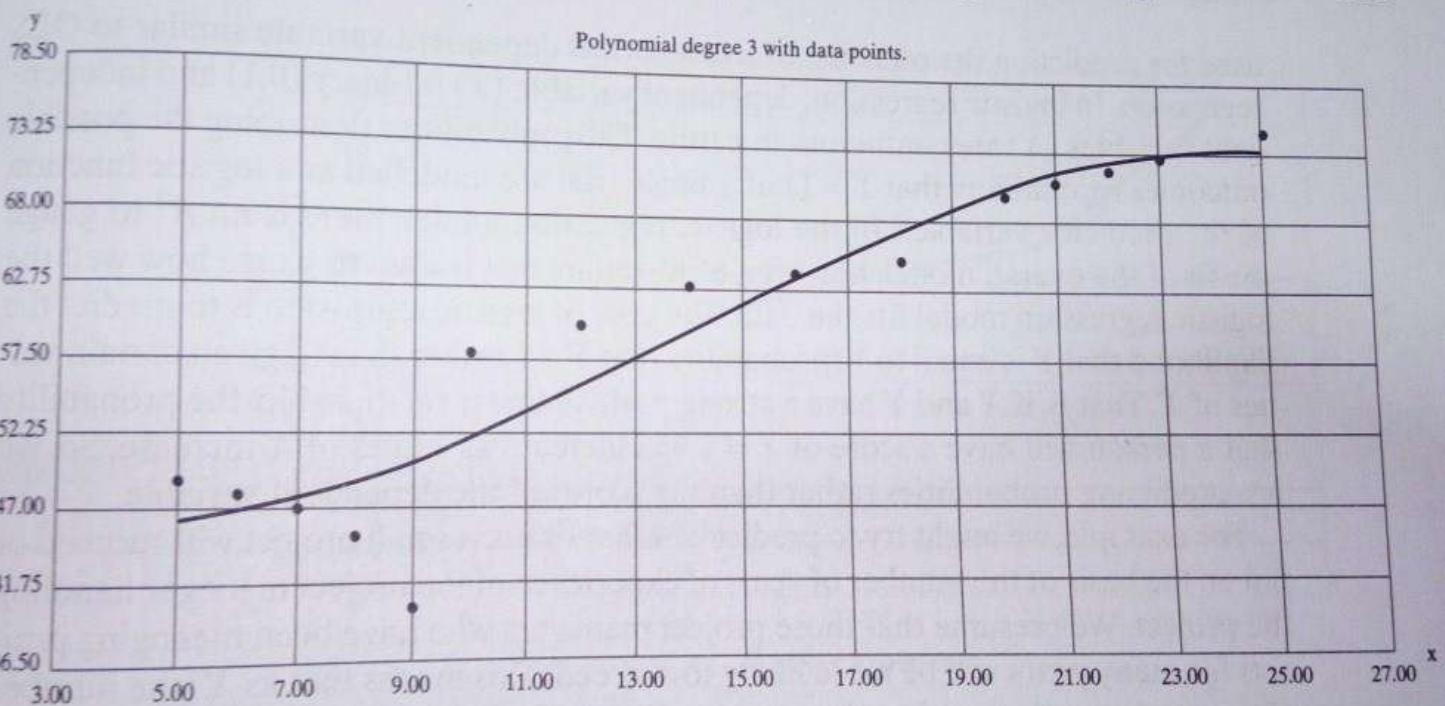


FIG. 8.16
Polynomial regression degree 3

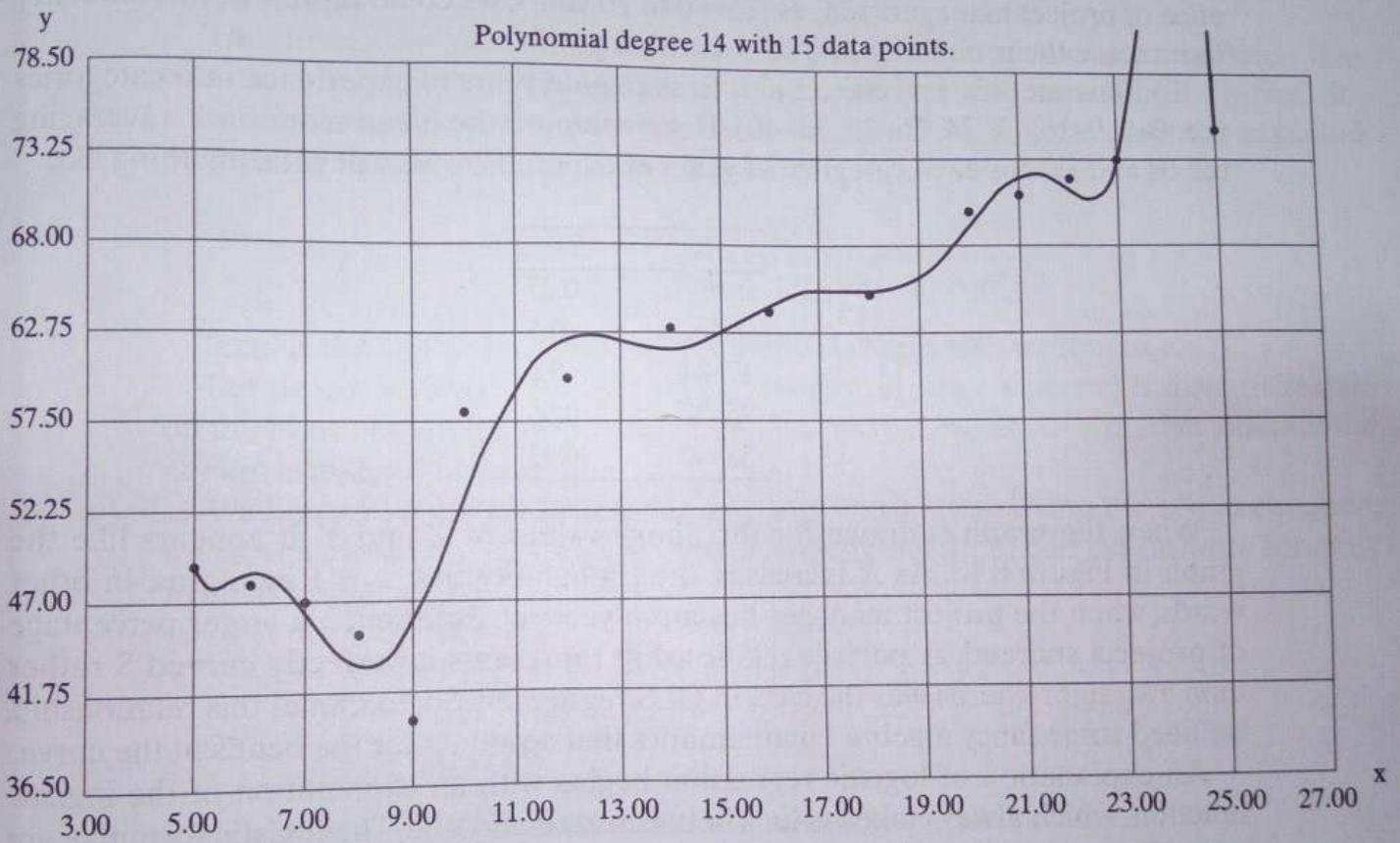


FIG. 8.17
Polynomial regression degree 14

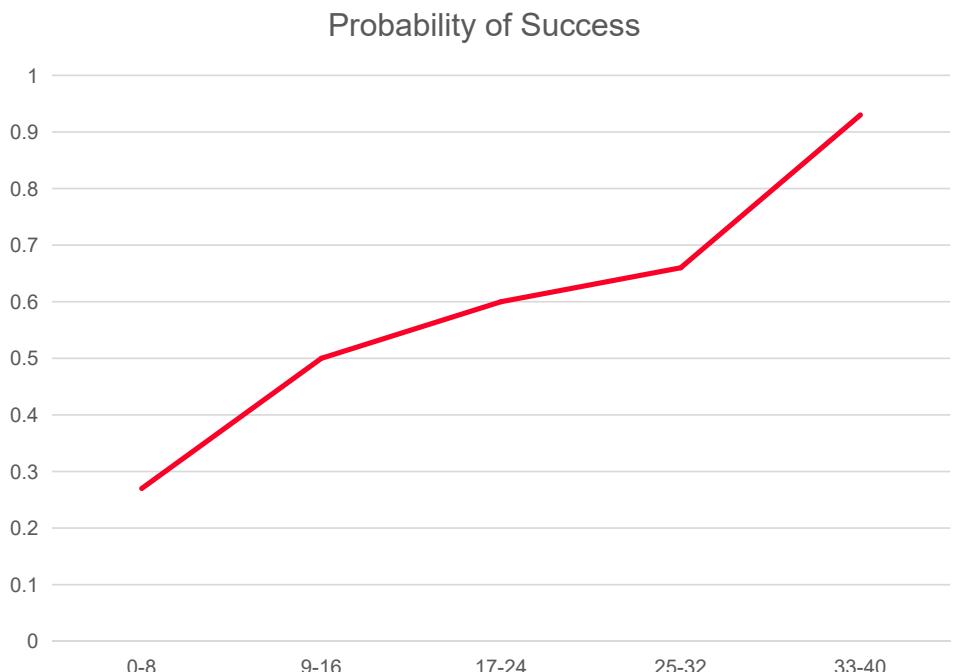
Logistic Regression

- The **logistic model** is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.
- Logistic regression(LR) is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.
- It can be used for both Classification and Regression based on the given problem.
- The goal of LR is to predict the likelihood that Y is equal to 1(probability that $Y = 1$ rather than 0) given certain values of X.
- **A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.**

Example

- We may predict the success or failure of a small project on the basis of the number of years of experience of the project manager handling the project.
- This means that as X (the number of years of experience of manager) increases, the probability that Y will be equal to 1(success of project) will tend to increase.

Years of Experience	Probability of Success
0-8	0.27
9-16	0.5
17-24	0.6
25-32	0.66
33-40	0.93



END OF MODULE-2

Classification Metrics in ML

1- Confusion Matrix

- One of the key concept in classification performance is **confusion matrix** which is a tabular visualization of the model predictions versus the ground-truth labels.
- Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.

	Class 1 Actual	Class2 Actual
Class 1 Predicted	TP	FP
Class 2 predicted	FN	TN

- Here,
- Class 1 : Positive
- Class 2 : Negative
- **Definition of the Terms:**
 - Positive (P) : Observation is positive (for example: is an apple).
 - Negative (N) : Observation is not positive (for example: is not an apple).
 - True Positive (TP) : Observation is positive, and is predicted to be positive.
 - False Negative (FN) : Observation is positive, but is predicted negative.
 - True Negative (TN) : Observation is negative, and is predicted to be negative.
 - False Positive (FP) : Observation is negative, but is predicted positive.

Let's go through this with an example. Let's assume we are building a binary classification to classify cat images from non-cat images. And let's assume our test set has 1100 images (1000 non-cat images, and 100 cat images), with the below confusion matrix.

		Actual Class	
		Cat	Non-Cat
Predicted Class	Cat	90	60
	Non-Cat	10	940

- **Out of 100 cat images** the model has predicted 90 of them correctly and has mis-classified 10 of them. If we refer to the “cat” class as positive and the non-cat class as negative class, then 90 samples predicted as cat are considered as **true-positive**, and the 10 samples predicted as non-cat are **false negative**.
- **Out of 1000 non-cat images**, the model has classified 940 of them correctly, and mis-classified 60 of them. The 940 correctly classified samples are referred as **true-negative**, and those 60 are referred as **false-positive**
- As we can see diagonal elements of this matrix denote the correct prediction for different classes, while the off-diagonal elements denote the samples which are mis-classified.

Classification Accuracy

- Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the **number of correct predictions divided by the total number of predictions**, multiplied by 100. So in the above example, out of 1100 samples 1030 are predicted correctly, resulting in a classification accuracy of:
- **Classification accuracy**= $(90+940)/(1000+100)= 1030/1100= 93.6\%$

3- Precision

- There are many cases in which classification accuracy is not a good indicator of your model performance. One of these scenarios is when your class distribution is imbalanced (one class is more frequent than others). In this case, even if you predict all samples as the most frequent class you would get a high accuracy rate, which does not make sense at all (because your model is not learning anything, and is just predicting everything as the top class). For example in our cat vs non-cat classification above, if the model predicts all samples as non-cat, it would result in a $1000/1100 = 90.9\%$.

- Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:
- **Precision= True_Positive/ (True_Positive+ False_Positive)**
- The precision of Cat and Non-Cat class in above example can be calculated as:
- **Precision_cat= #samples correctly predicted cat/#samples predicted as cat = $90/(90+60) = 60\%$**
- **Precision_NonCat= 940/950= 98.9%**
- As we can see the model has much higher precision in predicting non-cat samples, versus cats. This is not surprising, as model has seen more examples of non-cat images during training, making it better in classifying that class.

Recall

- Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:
- **Recall= True_Positive/ (True_Positive+ False_Negative)**
- Therefore, for our example above, the recall rate of cat and non-cat classes can be found as:
- **Recall_cat= 90/100= 90%**
- **Recall_NonCat= 940/1000= 94%**

Sensitivity and Specificity

- Sensitivity and specificity are two other popular metrics mostly used in medical and biology related fields, and are defined as:
- **True Positive Rate (Sensitivity)** : True Positive Rate is defined as $TP / (FN+TP)$. True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.
- **False Positive Rate (Specificity)** : False Positive Rate is defined as $FP / (FP+TN)$. False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

F-Measure

One measure of performance that takes into account both recall and precision

Harmonic mean of recall and precision:

$$F_1 = 2 \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Compared to arithmetic mean, both need to be high for harmonic mean to be high

AUC, F1 Score, ROC???

- Home Work.

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayes Classifier...

Notations:-

- $P(A)$ = prob. that event A will occur
- $P(A / B)$ = prob. that event A will occur, given that event B has already occurred.
- ∵ it is the conditional prob. of A
- ∵ it is based on the condition that B has already occurred.

Ex:-

- A..... Prob of a student passing course A
- B..... Prob of a student passing course B
- Then, $P(A / B)$?
- It is the prob. of the student passing A, when we know that he has passed B.



Bayes Classifier

- A probabilistic framework for solving classification problems

- Conditional Probability:

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

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

- Bayes theorem:

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



Bayesian Classifiers

- From where?
 - $P(A/B) = \frac{P(A \text{ & } B)}{P(B)}$ 1
 - $P(B/A) = \frac{P(A \text{ & } B)}{P(A)}$ 2

$\therefore \frac{1}{2}$ gives Bayes theorem

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

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

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A_1, A_2, \dots, A_n)
 - Goal is to predict class C
 - Specifically, we want to find the value of C that maximizes $P(C | A_1, A_2, \dots, A_n)$
- Can we estimate $P(C | A_1, A_2, \dots, A_n)$ directly from data?



Bayesian Classifiers

- Approach:
 - compute the posterior probability $P(C | A_1, A_2, \dots, A_n)$ for all values of C using the Bayes theorem
- Choose value of C that maximizes
 $P(C | A_1, A_2, \dots, A_n)$
- Equivalent to choosing value of C that maximizes
 $P(A_1, A_2, \dots, A_n | C) P(C)$
- How to estimate $P(A_1, A_2, \dots, A_n | C)$?



Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
 - $P(A_1, A_2, \dots, A_n | C) = P(A_1 | C_j) P(A_2 | C_j) \dots P(A_n | C_j)$
 - Can estimate $P(A_i | C_j)$ for all A_i and C_j .
 - New point is classified to C_j if $P(C_j) \prod P(A_i | C_j)$ is maximal.



Naive Bayes Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayes Classifier: An Example

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

- $P(C_i)$: $P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$
 $P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$

- Compute $P(X|C_i)$ for each class

$$P(\text{age} = \text{"}<=30\text{"} | \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{"}<= 30\text{"} | \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} | \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit_rating} = \text{"fair"} | \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$$

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i)*P(C_i) : P(X|\text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) = 0.007$$

Therefore, X belongs to class ("buys_computer = yes")

Naïve Bayes (Summary)

- Robust to isolated noise points. They are averaged when estimating conditional prob.
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
 - Use other techniques such as Bayesian Belief Networks (BBN)

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t, n_c is total no of classes).

- Maximum ($1 - 1/n_c$) when records are equally distributed among all classes, implying least interesting information
- Minimum (0.0) when all records belong to one class, implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j | t)]^2$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Gini} = 1 - P(C1)^2 - P(C2)^2 = 1 - 0 - 1 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Gini} = 1 - (1/6)^2 - (5/6)^2 = 0.278$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Gini} = 1 - (2/6)^2 - (4/6)^2 = 0.444$$



Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^k \frac{n_i}{n} GINI(i)$$

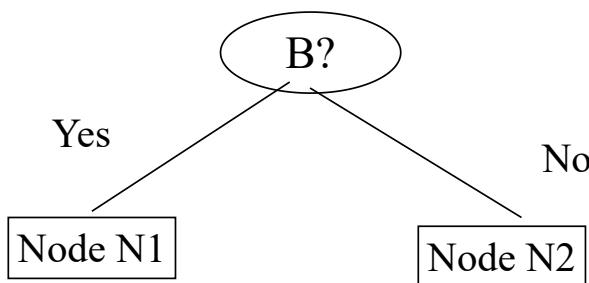
where, n_i = number of records at child i,
 n = number of records at node p.



Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
 - Larger and Purer Partitions are sought for.

$$\begin{aligned} \text{Gini}(N_1) &= 1 - (5/7)^2 - (2/7)^2 \\ &= 0.194 \\ \text{Gini}(N_2) &= 1 - (1/5)^2 - (4/5)^2 \\ &= 0.528 \end{aligned}$$



	N1	N2
C1	5	1
C2	2	4
Gini=0.333		

	Parent
C1	6
C2	6
Gini = 0.500	

$$\begin{aligned} \text{Gini(Children)} &= 7/12 * 0.194 + \\ &\quad 5/12 * 0.528 \\ &= 0.333 \end{aligned}$$

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

CarType		
{Sports, Luxury}	{Family}	
C1	3	
C2	2	
Gini	0.400	

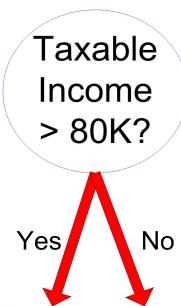
CarType		
{Sports}	{Family, Luxury}	
C1	2	
C2	1	
Gini	0.419	



Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
 - Number of possible splitting values = Number of distinct values
- Each splitting value has a count matrix associated with it
 - Class counts in each of the partitions, $A < v$ and $A \geq v$
- Simple method to choose best v
 - For each v , scan the database to gather count matrix and compute its Gini index
 - Computationally Inefficient!
Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
 - Sort the attribute on values
 - Linearly scan these values, each time updating the count matrix and computing gini index
 - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No										
Taxable Income																				
Sorted Values	60	70	75	85	90	95	100	120	125	220										
Split Positions	55	65	72	80	87	92	97	110	122	172	230									
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>								
Yes	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0		
No	0	7	1	6	2	5	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420									

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j | t) \log p(j | t)$$

(NOTE: $p(j | t)$ is the relative frequency of class j at node t).

- Measures homogeneity of a node.
 - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
 - Minimum (0.0) when all records belong to one class, implying most information



Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j | t) \log_2 p(j | t)$$

C1	0
C2	6

$$P(C1) = 0/6 = 0 \quad P(C2) = 6/6 = 1$$

$$\text{Entropy} = -0 \log 0 - 1 \log 1 = -0 - 0 = 0$$

C1	1
C2	5

$$P(C1) = 1/6 \quad P(C2) = 5/6$$

$$\text{Entropy} = -(1/6) \log_2 (1/6) - (5/6) \log_2 (5/6) = 0.65$$

C1	2
C2	4

$$P(C1) = 2/6 \quad P(C2) = 4/6$$

$$\text{Entropy} = -(2/6) \log_2 (2/6) - (4/6) \log_2 (4/6) = 0.92$$



Splitting Based on INFO...

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^k \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3 and C4.5
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.



ID3 – Iterative Dichotomizer

- Invented by J. Ross Quinlan
- Employs a top-down greedy search through the space of possible decision trees.
Greedy because there is no backtracking. It picks highest values first.
- Select attribute that is most useful for classifying examples (attribute that has the highest Information Gain).



Entropy

- Entropy measures the impurity of an arbitrary collection of examples.
- For a collection S, entropy is given as:

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

- For a collection S having positive and negative examples

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

where p_+ is the proportion of positive examples

and p_- is the proportion of negative examples

In general, $\text{Entropy}(S) = 0$ if all members of S belong to the same class.

$\text{Entropy}(S) = 1$ (maximum) when all members are split equally.

Information Gain

- Measures the expected reduction in entropy. The higher the IG, more is the expected reduction in entropy.

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

where $Values(A)$ is the set of all possible values for attribute A,

S_v is the subset of S for which attribute A has value v.



Example : Determine whether an animal lays eggs using entropy

Independent/Condition attributes					Dependent/ Decision attributes
Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

$$Entropy(S) = \sum_{i=1}^c - p_i \log_2 p_i$$

$$\begin{aligned}\text{Entropy(4Y,2N): } & -(4/6)\log_2(4/6) - (2/6)\log_2(2/6) \\ & = 0.91829\end{aligned}$$

Now, we have to find the IG for all four attributes
Warm-blooded, Feathers, Fur, Swims

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

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

For attribute ‘Warm-blooded’:

Values(Warm-blooded) : [Yes, No]

S = [4Y, 2N]

S_{Yes} = [3Y, 2N] E(S_{Yes}) = 0.97095

S_{No} = [1Y, 0N] E(S_{No}) = 0 (all members belong to same class)

$$\begin{aligned} Gain(S, \text{Warm-blooded}) &= 0.91829 - [(5/6)*0.97095 + (1/6)*0] \\ &= 0.10916 \end{aligned}$$

For attribute ‘Feathers’:

Values(Feathers) : [Yes, No]

S = [4Y, 2N]

S_{Yes} = [3Y, 0N] E(S_{Yes}) = 0

S_{No} = [1Y, 2N] E(S_{No}) = 0.91829

$$\begin{aligned} Gain(S, \text{Feathers}) &= 0.91829 - [(3/6)*0 + (3/6)*0.91829] \\ &= 0.45914 \end{aligned}$$



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

For attribute ‘Fur’:

Values(Fur) : [Yes, No]

S = [4Y, 2N]

S_{Yes} = [0Y, 1N] E(S_{Yes}) = 0

S_{No} = [4Y, 1N] E(S_{No}) = 0.7219

$$\begin{aligned} Gain(S, Fur) &= 0.91829 - [(1/6)*0 + (5/6)*0.7219] \\ &= 0.3167 \end{aligned}$$

For attribute ‘Swims’:

Values(Swims) : [Yes, No]

S = [4Y, 2N]

S_{Yes} = [1Y, 1N] E(S_{Yes}) = 1 (equal members in both classes)

S_{No} = [3Y, 1N] E(S_{No}) = 0.81127

$$\begin{aligned} Gain(S, Swims) &= 0.91829 - [(2/6)*1 + (4/6)*0.81127] \\ &= 0.04411 \end{aligned}$$

$\text{Gain}(S, \text{Warm-blooded}) = 0.10916$

$\text{Gain}(S, \text{Feathers}) = 0.45914$

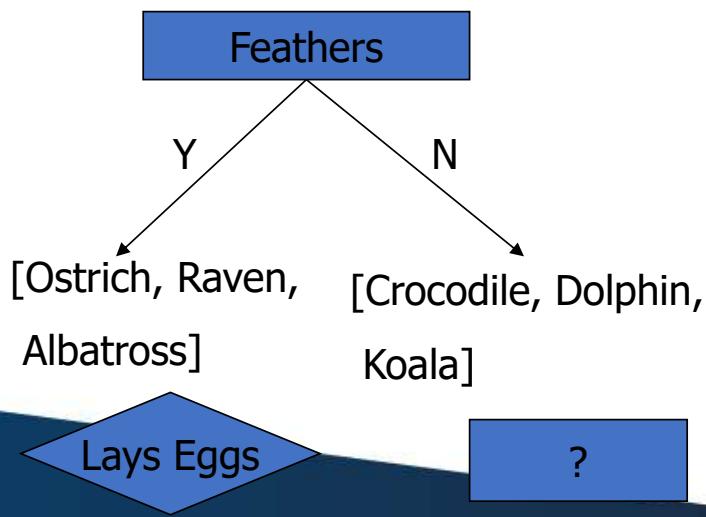
$\text{Gain}(S, \text{Fur}) = 0.31670$

$\text{Gain}(S, \text{Swims}) = 0.04411$

$\text{Gain}(S, \text{Feathers})$ is maximum, so it is considered as the root node

Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Ostrich	Yes	Yes	No	No	Yes
Crocodile	No	No	No	Yes	Yes
Raven	Yes	Yes	No	No	Yes
Albatross	Yes	Yes	No	No	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

The 'Y' descendant has only positive examples and becomes the leaf node with classification 'Lays Eggs'



Animal	Warm-blooded	Feathers	Fur	Swims	Lays Eggs
Crocodile	No	No	No	Yes	Yes
Dolphin	Yes	No	No	Yes	No
Koala	Yes	No	Yes	No	No

We now repeat the procedure,

S: [Crocodile, Dolphin, Koala]

S: [1+,2-]

$$Entropy(S) = \sum_{i=1}^c - p_i \log_2 p_i$$

$$\begin{aligned} \text{Entropy}(S) &= -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) \\ &= 0.91829 \end{aligned}$$

- For attribute ‘Warm-blooded’:

Values(Warm-blooded) : [Yes,No]

$$S = [1Y, 2N]$$

$$S_{\text{Yes}} = [0Y, 2N] \quad E(S_{\text{Yes}}) = 0$$

$$S_{\text{No}} = [1Y, 0N] \quad E(S_{\text{No}}) = 0$$

$$\text{Gain}(S, \text{Warm-blooded}) = 0.91829 - [(2/3)*0 + (1/3)*0] = \mathbf{0.91829}$$

- For attribute ‘Fur’:

Values(Fur) : [Yes,No]

$$S = [1Y, 2N]$$

$$S_{\text{Yes}} = [0Y, 1N] \quad E(S_{\text{Yes}}) = 0$$

$$S_{\text{No}} = [1Y, 1N] \quad E(S_{\text{No}}) = 1$$

$$\text{Gain}(S, \text{Fur}) = 0.91829 - [(1/3)*0 + (2/3)*1] = \mathbf{0.25162}$$

For attribute ‘Swims’:

Values(Swims) : [Yes,No]

$$S = [1Y, 2N]$$

$$S_{\text{Yes}} = [1Y, 1N] \quad E(S_{\text{Yes}}) = 1$$

$$S_{\text{No}} = [0Y, 1N] \quad E(S_{\text{No}}) = 0$$

$$\text{Gain}(S, \text{Swims}) = 0.91829 - [(2/3)*1 + (1/3)*0] = \\ \mathbf{0.25162}$$

Gain(S,Warm-blooded) is maximum

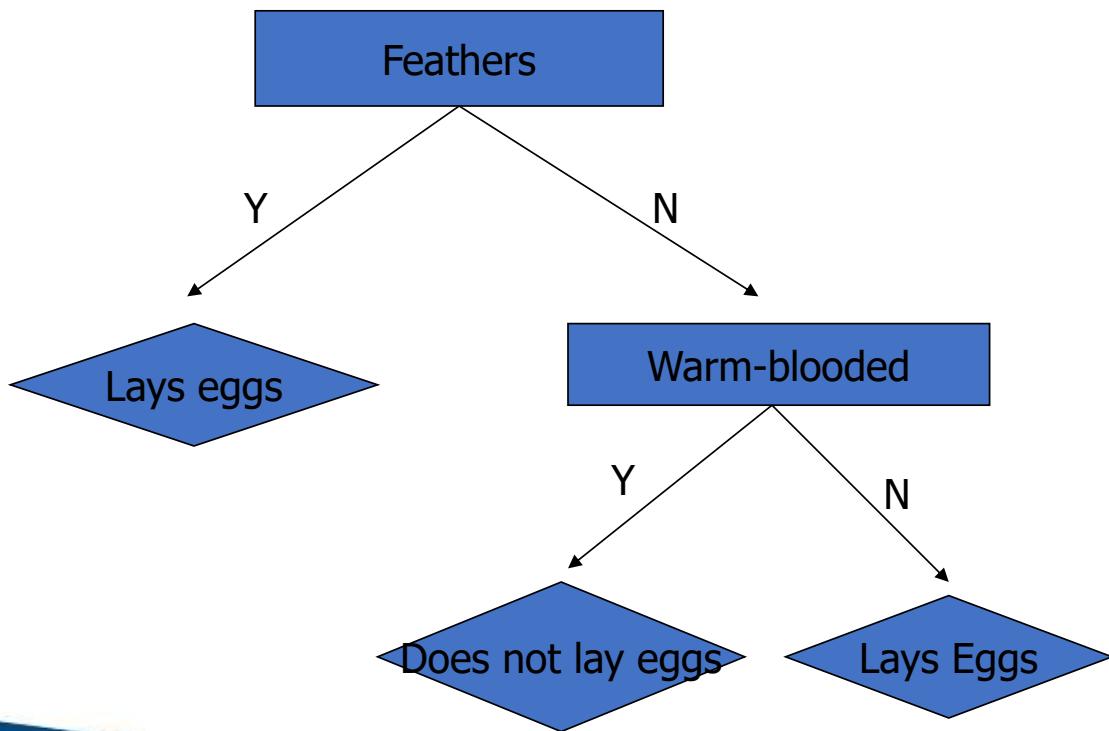


**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



The final decision tree will be:



Object Analysis: Classification

Credits: Material for the slides is drawn from a variety of sources including Object Oriented Analysis and Design using UML by Ali Bahrami.

Outline

- The concept of classification.
- How to identify classes
 - Noun phrase approach.
 - Common class patterns approach.
 - Use case driven approach

... Intelligent classification is intellectually hard work, and it best comes about through an incremental and iterative process

Booch

..There is no such thing as the perfect class structure, nor the right set of objects. As in any engineering discipline, our design choice is compromisingly shaped by many competing factors.

Booch

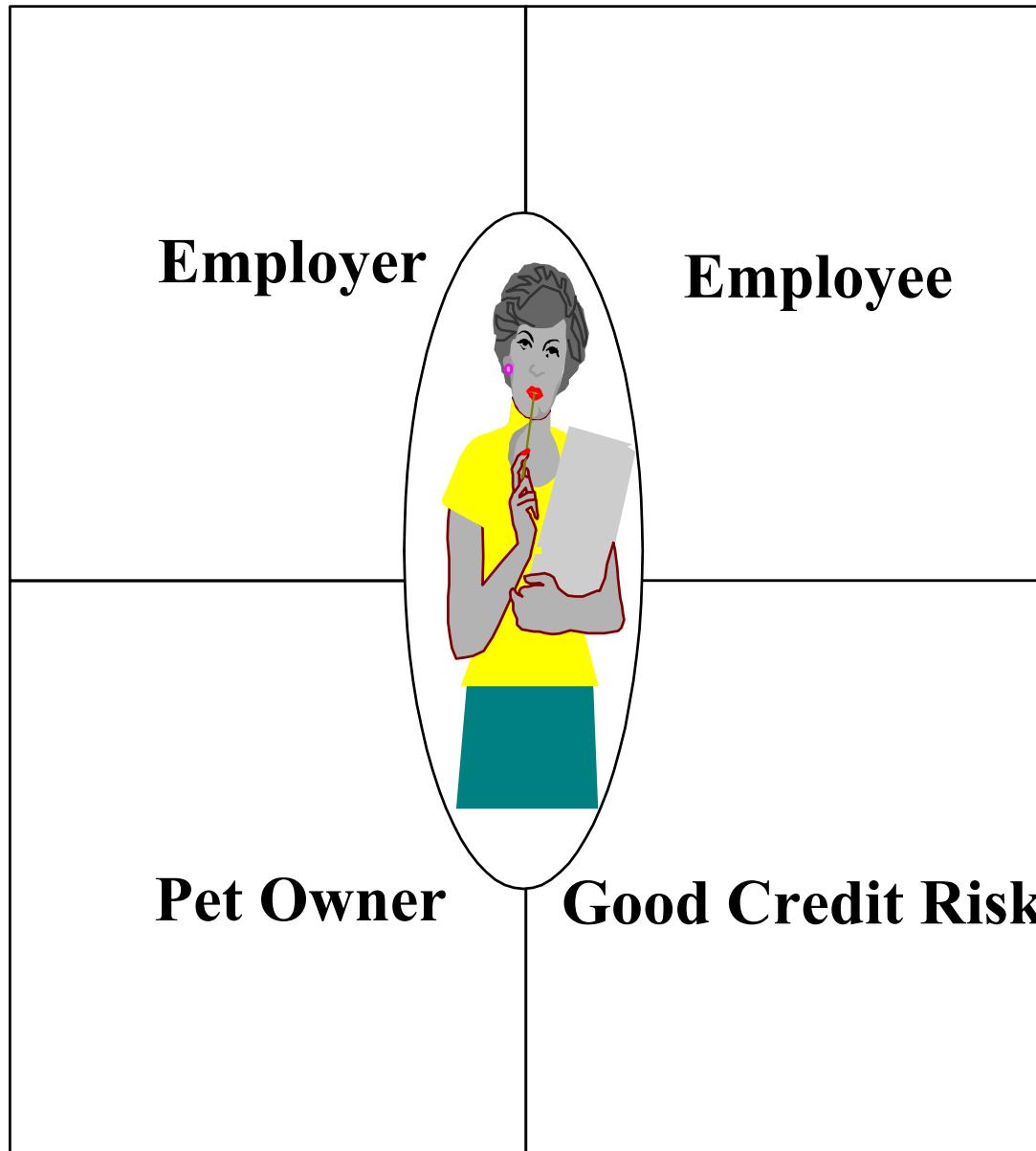
The Concept of Classification

- Classification is the process of checking to see if an object belongs to a category or a class and it is regarded as a basic attribute of human nature.
- A class is a specification of structure, behavior, and the description of an object.

The Challenge of Classification

- Intelligent classification is intellectually hard work and may seem rather arbitrary.
- Martin and Odell have observed in object-oriented analysis and design, that

“In fact, an object can be categorized in more than one way.”



Linear Regression

- **Straight-line linear regression:**

- involves a response variable y and a single predictor variable x

$$y = w_0 + w_1 x$$

- w_0 : y -intercept
 - w_1 : slope
 - w_0 & w_1 are **regression coefficients**

Linear regression

- **Method of least squares**: estimates the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- D : a training set
- x : values of predictor variable
- y : values of response variable
- $|D|$: data points of the form $(x_1, y_1), (x_2, y_2), \dots, (x|D|, y|D|)$.
- \bar{x} : the mean value of $x_1, x_2, \dots, x|D|$
- \bar{y} : the mean value of $y_1, y_2, \dots, y|D|$

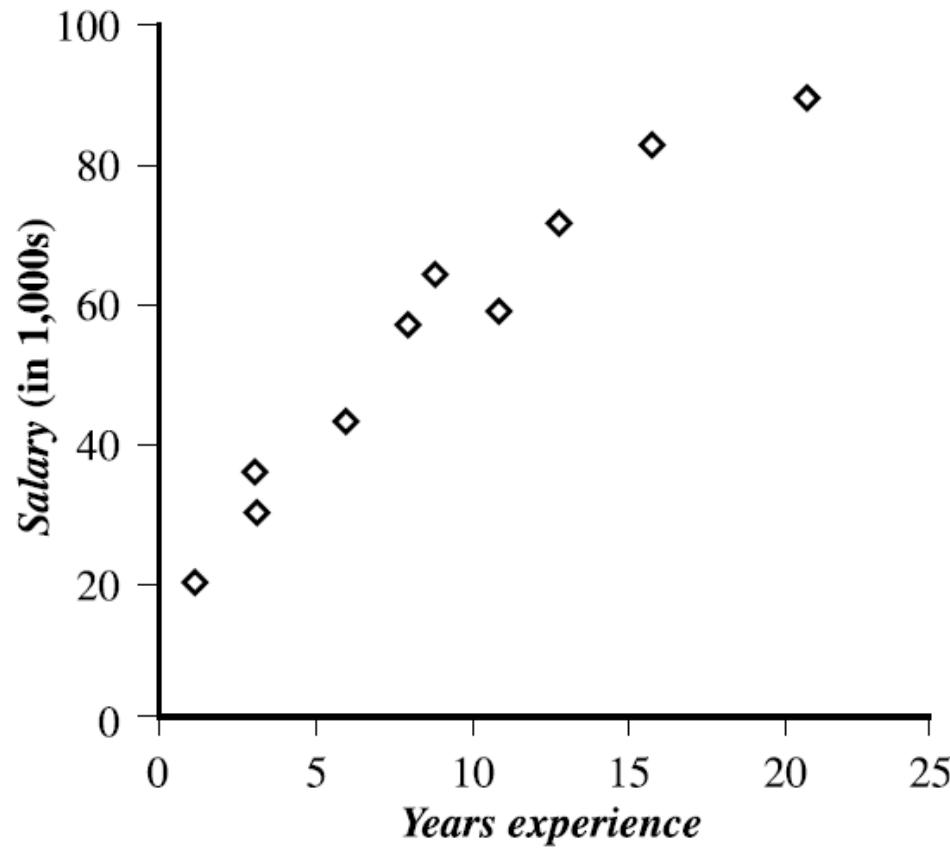
Example: Salary problem

- The table shows a set of paired data where x is the number of years of work experience of a college graduate and y is the corresponding salary of the graduate.

x years experience	y salary (in \$1000s)
3	30
8	57
9	64
13	72
3	36
6	43
11	59
21	90
1	20
16	83

Linear Regression

- The 2-D data can be graphed on a **scatter plot**.
- The plot suggests a linear relationship between the two variables, x and y .



Example: Salary data

- Given the above data, we compute

$$\bar{x} = 9.1 \text{ and } \bar{y} = 55.4$$

- we get

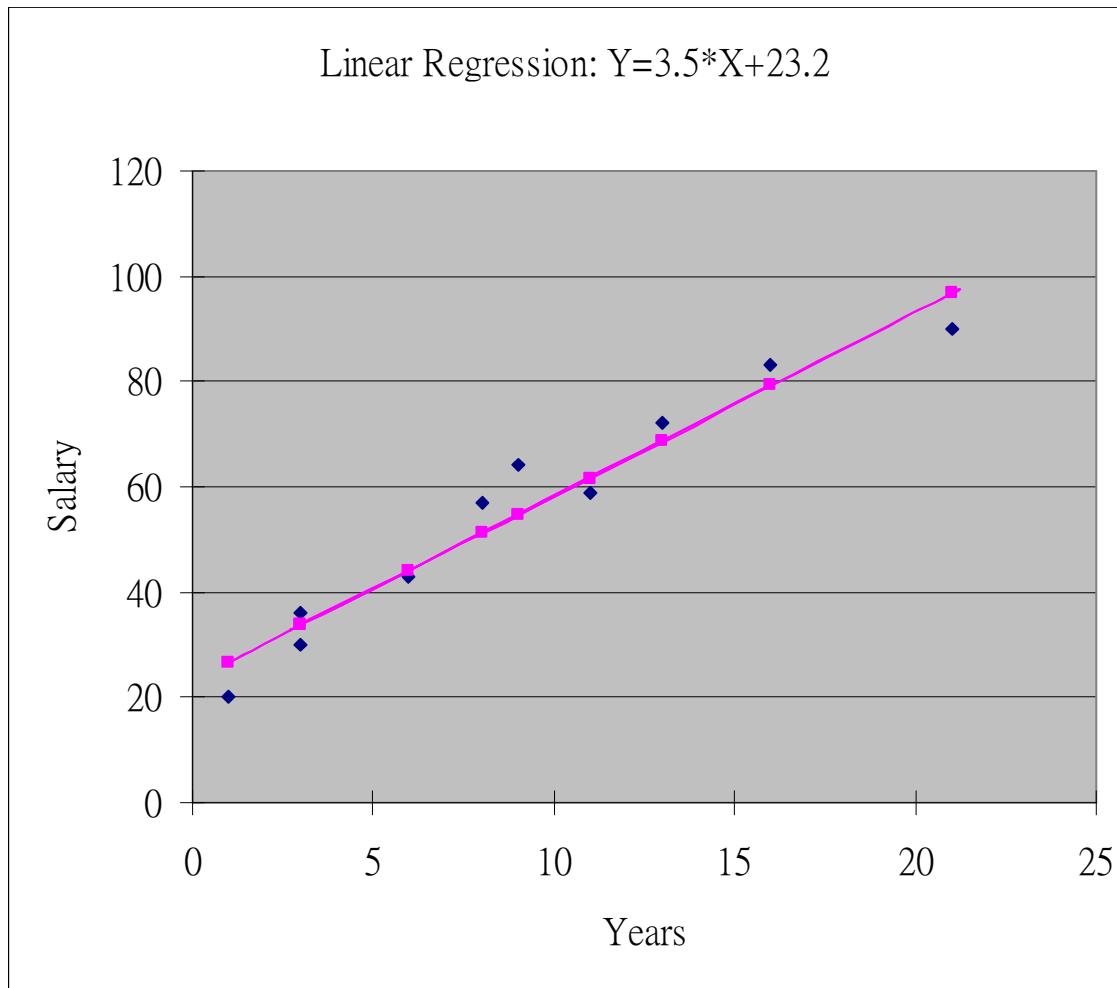
$$w_1 = \frac{(3 - 9.1)(30 - 55.4) + (8 - 9.1)(57 - 55.4) + \cdots + (16 - 9.1)(83 - 55.4)}{(3 - 9.1)^2 + (8 - 9.1)^2 + \cdots + (16 - 9.1)^2} = 3.5$$

$$w_0 = 55.4 - (3.5)(9.1) = 23.6$$

- The equation of the least squares line is estimated by

$$y = 23.6 + 3.5x$$

Example: Salary data





R^2 , Adjusted
 R^2 , SVM(Basics)

R-squared (R^2)

- ❖ It measures the proportion of the variation in your dependent variable explained by all of your independent variables in the model.
- ❖ It assumes that every independent variable in the model helps to explain variation in the dependent variable.
- ❖ In reality, some independent variables (predictors) don't help to explain dependent (target) variable.
In other words, some variables do not contribute in predicting target

- Mathematically, R-squared is calculated by dividing sum of squares of residuals (SS_{res}) by total sum of squares (SS_{tot}) and then subtract it from 1.
- In this case, SS_{tot} measures total variation. SS_{reg} measures explained variation. SS_{res} measures unexplained variation.

$$R^2 \equiv 1 - \frac{SS_{res}}{SS_{tot}} \equiv 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

\downarrow

$$R^2 = \frac{SS_{reg}}{SS_{tot}}$$

R - Squared is also called **coefficient of determination**. It lies between 0% and 100%. A r-squared value of 100% means the model explains all the variation of the target variable. And a value of 0% measures zero predictive power of the model. **Higher R-squared value, better the model.**

Adjusted R-Squared

- ❖ It measures the proportion of variation explained by only those independent variables that really help in explaining the dependent variable.
- ❖ It penalizes you for adding independent variable that do not help in predicting the dependent variable.

Adjusted R-squared value can be calculated based on value of r-squared, number of independent variables (predictors), total sample size.

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square

p = Number of predictors

N = Total sample size.

❖ Problem Statement:

❖ A fund has a sample R-squared value close to 0.5 and it is doubtlessly offering higher risk adjusted returns with the sample size of 50 for 5 predictors. Find Adjusted R square value.

❖ **Solution:**

❖ Sample size = 50 Number of predictor = 5 Sample R - square = 0.5. Substitute the qualities in the equation,

$$\begin{aligned}R_{adj}^2 &= 1 - \left[\frac{(1-0.5^2)(50-1)}{50-5-1} \right] \\&= 1 - (0.75) \times \frac{49}{44}, \\&= 1 - 0.8352, \\&= 0.1648\end{aligned}$$

Difference between R-square and Adjusted R-square

- Every time you add a independent variable to a model, the R-squared **increases**, even if the independent variable is insignificant. It never declines.
- Whereas Adjusted R-squared increases only when independent variable is significant and affects dependent variable.

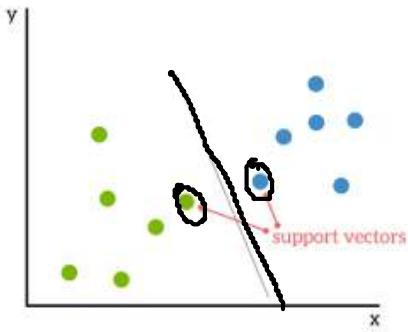
☞ In the table below, adjusted r-squared is maximum when we included two variables. It declines when third variable is added. Whereas r-squared increases when we included third variable. It means third variable is insignificant.

Variables	R-Squared	Adjusted R- Squared
1	67.5	67.1
2	85.9	84.2
3	88.9	81.7

- ❖ Adjusted r-squared can be negative when r-squared is close to zero.
- ❖ Adjusted r-squared value always be less than or equal to r-squared value.
- ❖ Adjusted R-square should be used to compare models with different numbers of independent variables. Adjusted R-square should be used while selecting important predictors (independent variables)

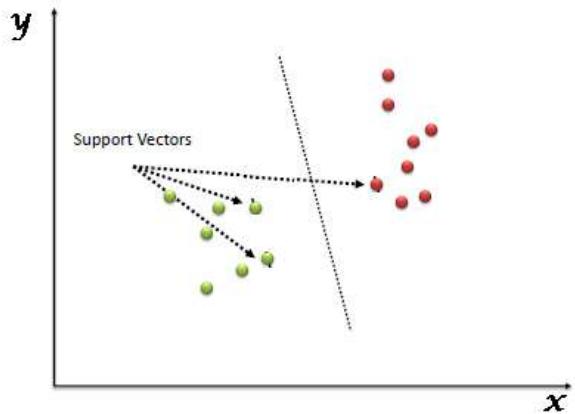
BASICS OF SVM

- ❖ A Support Vector Machine (SVM) is a supervised machine learning algorithm that can be employed for both classification and regression purposes. SVMs are more commonly used in classification problems and as such, this is what we will focus on in this post.
- ❖ SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes, as shown in the image below.



What is Support Vector Machine?

- “Support Vector Machine” (SVM) is a supervised [machine learning algorithm](#) which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).



❖ Support Vectors

❖ Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

❖ What is a hyperplane?

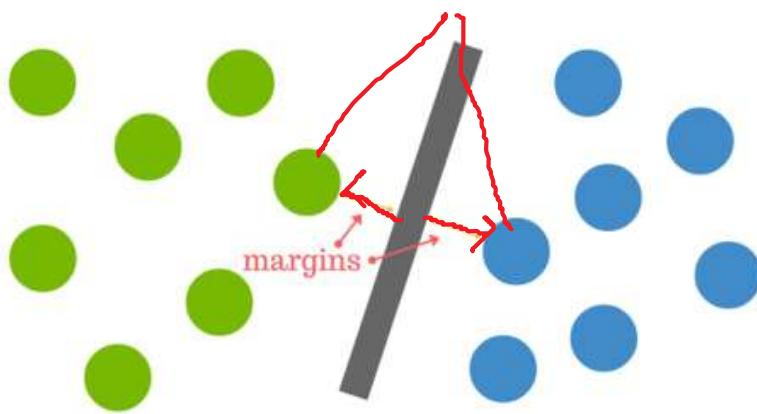
❖ As a simple example, for a classification task with only two features (like the image above), you can think of a hyperplane as a line that linearly separates and classifies a set of data.

❖ Intuitively, the further from the hyperplane our data points lie, the more confident we are that they have been correctly classified. We therefore want our data points to be as far away from the hyperplane as possible, while still being on the correct side of it.

❖ So when new testing data is added, whatever side of the hyperplane it lands will decide the class that we assign to it.

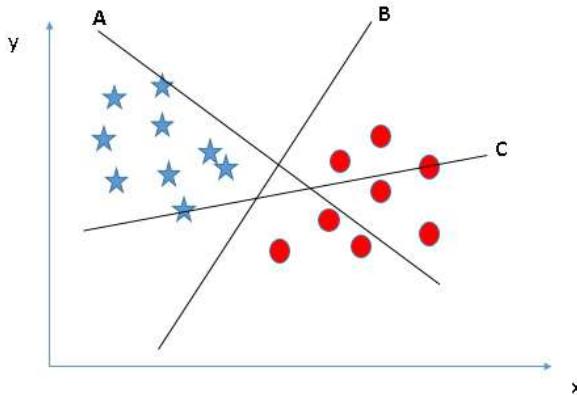
How do we find the right hyperplane?

- ❖ The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.

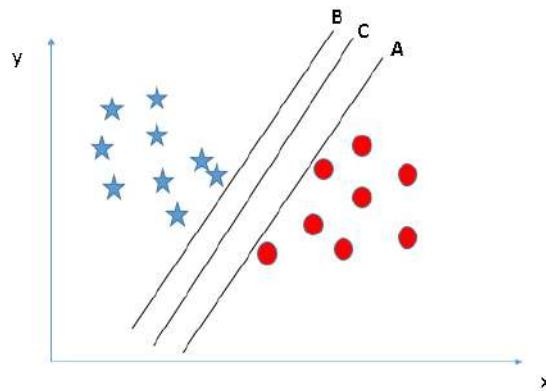


How does it work?

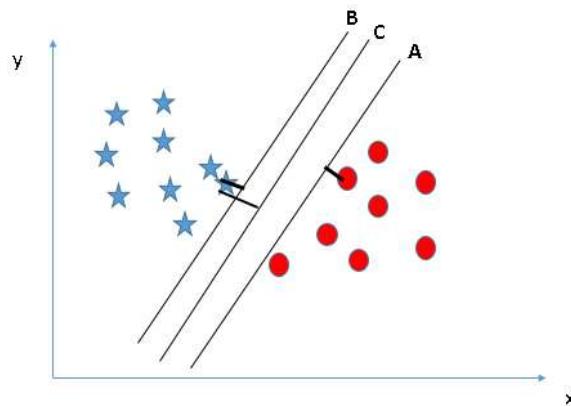
- Above, we got accustomed to the process of segregating the two classes with a hyper-plane. Now the burning question is “How can we identify the right hyper-plane?”
- Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle. You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job



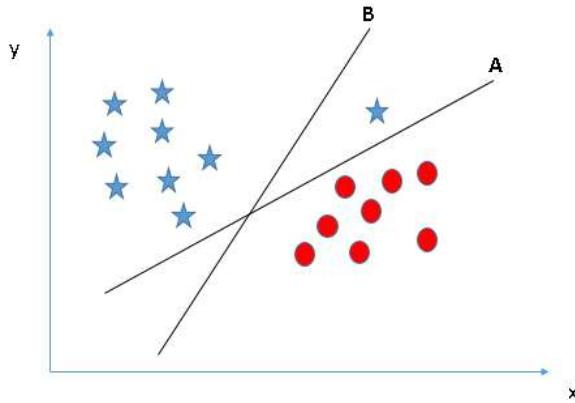
- Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, How can we identify the right hyper-plane? Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**.



- Let's look at the below snapshot: Below you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is high chance of miss-classification.

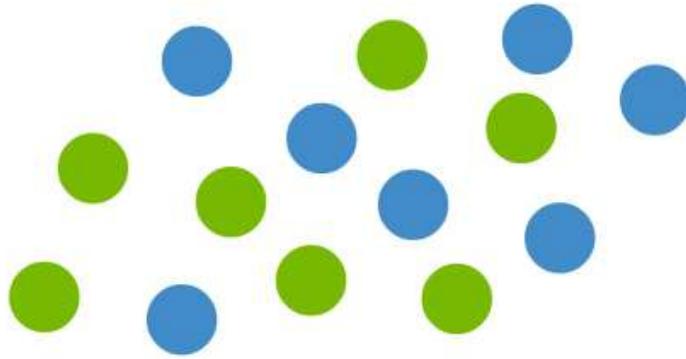


- Identify the right hyper-plane (Scenario-3): Hint: Use the rules as discussed in previous section to identify the right hyper-plane. Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A**. But, here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is **A**.

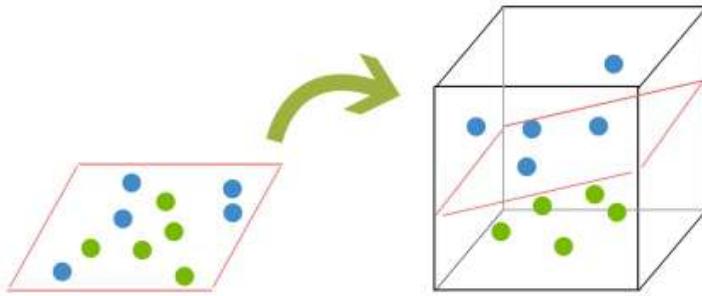


But what happens when there is no clear hyperplane?

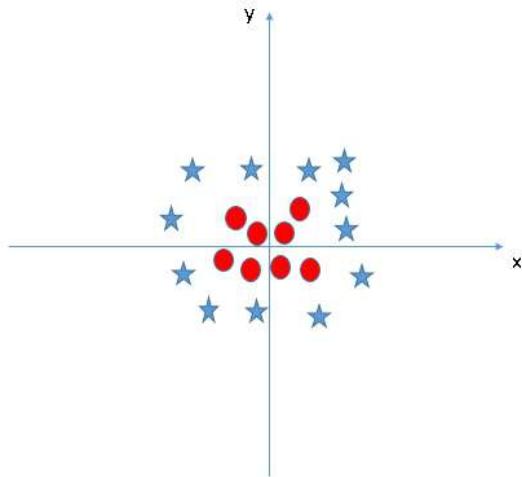
☞ This is where it can get tricky. Data is rarely ever as clean as our simple example above. A dataset will often look more like the jumbled balls below which represent a linearly non separable dataset.



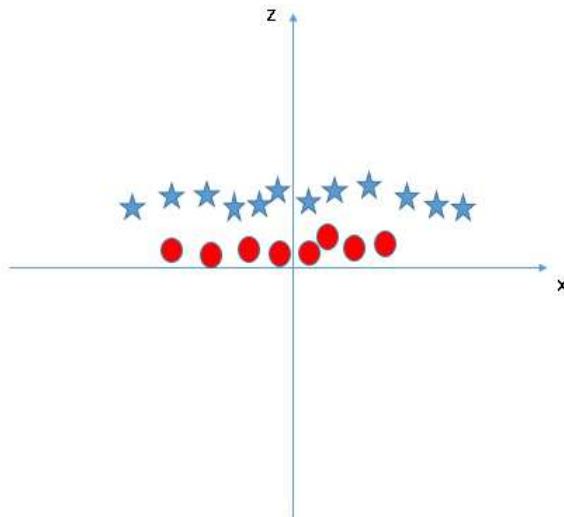
- ❖ In order to classify a dataset like the one above it's necessary to move away from a 2d view of the data to a 3d view. Explaining this is easiest with another simplified example. Imagine that our two sets of colored balls above are sitting on a sheet and this sheet is lifted suddenly, launching the balls into the air. While the balls are up in the air, you use the sheet to separate them. This 'lifting' of the balls represents the mapping of data into a higher dimension. Because we are now in three dimensions, our hyperplane can no longer be a line. It must now be a plane as shown in the example above. The idea is that the data will continue to be mapped into higher and higher dimensions until a hyperplane can be formed to segregate it.



- ☞ **Find the hyper-plane to segregate to classes (Scenario-5):** In the scenario below, we can't have linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.



- SVM can solve this problem. Easily! It solves this problem by introducing additional feature. Here, we will add a new feature $z=x^2+y^2$. Now, let's plot the data points on axis



- ❖ In above plot, points to consider are:
 - All values for z would be positive always because z is the squared sum of both x and y
 - In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.
- ❖ In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the **kernel trick**. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

Pros & Cons of Support Vector Machines

Pros

- ❖ Accuracy
- ❖ Works well on smaller cleaner datasets
- ❖ It can be more efficient because it uses a subset of training points

Cons

- ❖ Isn't suited to larger datasets as the training time with SVMs can be high
- ❖ Less effective on noisier datasets with overlapping classes

SVM Uses

- ❖ SVM is used for text classification tasks such as category assignment, detecting spam and sentiment analysis. It is also commonly used for image recognition challenges, performing particularly well in aspect-based recognition and color-based classification. SVM also plays a vital role in many areas of handwritten digit recognition, such as postal automation services

x_1	x_2	x_3	Class
R	W	R	Yes
R	W	W	Yes
N	W	W	No
W	R	W	No

$$\text{Entropy (class)} = \sum_{i=1}^C -p_i \log_2 p_i$$

in this case only 2 classes \rightarrow Yes and No

$$= -(3/4) \log_2(3/4) - 1/4 \log_2(1/4)$$

$$= 0.98$$

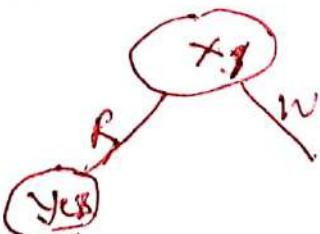
Given class entropy Ex. we calculate gain
with x_1, x_2 and x_3

$$\text{suppose! } \text{Gain}, x_1 = 0.98$$

$$\text{Gain}, x_2 = 0.95$$

$$\text{Gain}, x_3 = 0.96$$

so, x_3 gives highest gain so this will be at top. Tree

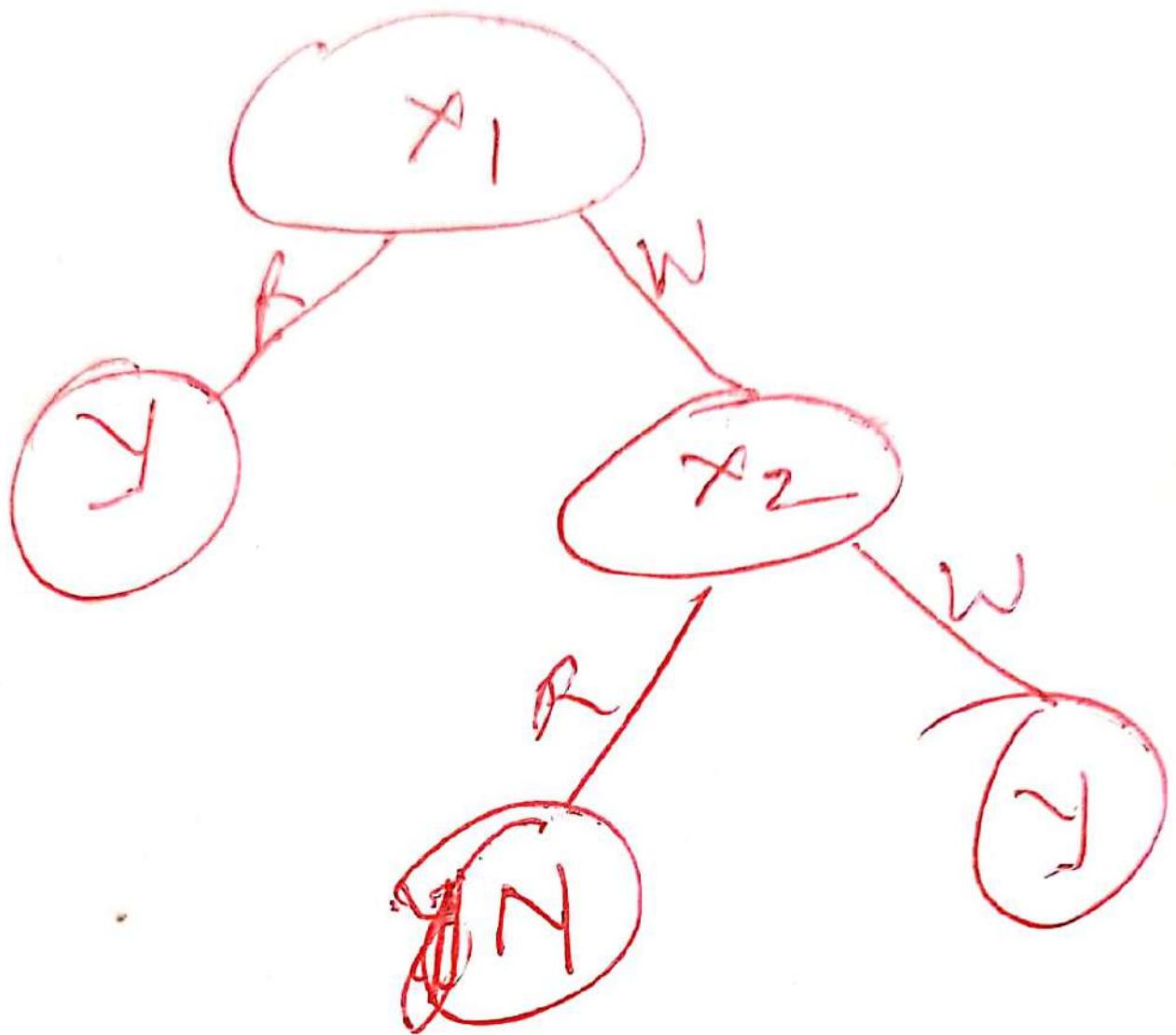


x_1	x_2	x_3	Class
W	W	W	Yes
W	R	W	No

$$\text{Now } Ex = 1$$

$$\text{Gain}, x_2 = 0.91$$

$$\text{Gain}, x_3 = 0.8$$



Gradient Descent

Gradient Descent can be used in different machine learning algorithms, including neural networks.

Gradient Descent can be used in linear regression to find best fit line for given training data set.

Linear Regression

$$\hat{y} = mx + b$$

In order to fit the regression line, we tune two parameters: slope (m) and intercept (b). Once optimal parameters are found, we usually evaluate results with a mean squared error (MSE). We remember that smaller MSE — better. In other words, we are trying to minimize it.

Minimization of the function is the exact task of the Gradient Descent algorithm. It takes parameters and tunes them till the local minimum is reached.

1. First, we take a function we would like to minimize, and very frequently it will be Mean Squared Errors function.
2. We identify parameters, such as m and b in the regression function and we take partial derivatives of MSE with respect to these parameters. This is the most crucial and hardest part. Each derived function can tell which way we should tune parameters and by how much.

3. We update parameters by iterating through our derived functions and gradually minimizing MSE. In this process, we use an additional parameter `learning rate` which helps us define the step we take towards updating parameters with each iteration. By setting a smaller learning rate we make sure our model wouldn't jump over a minimum point of MSE and converge nicely.

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

Derivatives

$$f(m, b) = \frac{1}{n} \sum_{i=1}^n (y_i - (mx_i + b))^2$$

$$\frac{\partial f}{\partial m} = \frac{1}{n} \sum_{i=1}^n -2x_i(y_i - (mx_i + b))$$

$$\frac{\partial f}{\partial b} = \frac{1}{n} \sum_{i=1}^n -2(y_i - (mx_i + b))$$

Update parameters

$$m = m - \alpha * \partial f / \partial m$$

$$b = b - \alpha * \partial f / \partial b$$

Supervised Learning

Linear Models and Gradient Decent Algorithm

EXAMPLE

- We will now go through each of the steps in detail. But before that, we have to standardize the data as it makes the optimization process faster.

HOUSING DATA	
House Size (X)	House Price (Y)
1,100	1,99,000
1,400	2,45,000
1,425	3,19,000
1,550	2,40,000
1,600	3,12,000
1,700	2,79,000
1,700	3,10,000
1,875	3,08,000
2,350	4,05,000
2,450	3,24,000

Min-Max Standardization	
X (X-Min/Max-min)	Y (Y-Min/Max-Min)
0.00	0.00
0.22	0.22
0.24	0.58
0.33	0.20
0.37	0.55
0.44	0.39
0.44	0.54
0.57	0.53
0.93	1.00
1.00	0.61

Step 1: To fit a line $Y_{\text{pred}} = a + b X$, start off with random values of a and b and calculate prediction error (SSE)

a	b	X	Y	YP=a+bX	SSE=1/2(Y-YP)^2
0.45	0.75	0.00	0.00	0.45	0.101
		0.22	0.22	0.62	0.077
		0.24	0.58	0.63	0.001
		0.33	0.20	0.70	0.125
		0.37	0.55	0.73	0.016
		0.44	0.39	0.78	0.078
		0.44	0.54	0.78	0.030
		0.57	0.53	0.88	0.062
		0.93	1.00	1.14	0.010
		1.00	0.61	1.20	0.176
				Total SSE	0.677

Step 2: Calculate the error gradient w.r.t the weights

$$\partial SSE/\partial a = -(Y - YP)$$

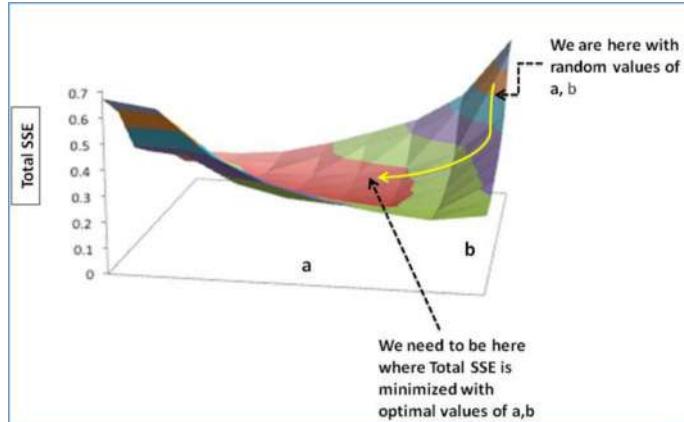
$$\partial SSE/\partial b = -(Y - YP)X$$

$$\text{Here, } SSE = \frac{1}{2} (Y - YP)^2 = \frac{1}{2} (Y - (a + bX))^2$$

$\partial SSE/\partial a$ and $\partial SSE/\partial b$ are the **gradients** and they give the direction of the movement of a , b w.r.t to SSE.

a	b	X	Y	YP=a+bX	SSE	$\partial SSE/\partial a$ $= -(Y - YP)$	$\partial SSE/\partial b$ $= -(Y - YP)X$	
0.45	0.75	0.00	0.00	0.45	0.101	0.45	0.00	
		0.22	0.22	0.62	0.077	0.39	0.09	
		0.24	0.58	0.63	0.001	0.05	0.01	
		0.33	0.20	0.70	0.125	0.50	0.17	
		0.37	0.55	0.73	0.016	0.18	0.07	
		0.44	0.39	0.78	0.078	0.39	0.18	
		0.44	0.54	0.78	0.030	0.24	0.11	
		0.57	0.53	0.88	0.062	0.35	0.20	
		0.93	1.00	1.14	0.010	0.14	0.13	
		1.00	0.61	1.20	0.176	0.59	0.59	
					Total SSE	0.677	Sum	
						3.300	1.545	

Step 3: Adjust the weights with the gradients to reach the optimal values where SSE is minimized



We need to update the random values of a, b so that we move in the direction of optimal a, b.
Update rules:

$$\begin{aligned} a &- \frac{\partial \text{SSE}}{\partial a} \\ b &- \frac{\partial \text{SSE}}{\partial b} \end{aligned}$$

So, update rules:

$$\text{New } a = a - \alpha * \frac{\partial \text{SSE}}{\partial a} = 0.45 - 0.01 * 3.300 = 0.42$$

$$\text{New } b = b - \alpha * \frac{\partial \text{SSE}}{\partial b} = 0.75 - 0.01 * 1.545 = 0.73$$

Here, α is the learning rate = 0.01, which is the pace of adjustment to the weights.

Step 4: Use new a and b for prediction and to calculate new Total SSE

a	b	X	Y	Y $\hat{=}$ a+bX	SSE	$\partial \text{SSE} / \partial a$	$\partial \text{SSE} / \partial b$
0.42	0.73	0.00	0.00	0.42	0.087	0.42	0.00
		0.22	0.22	0.58	0.064	0.36	0.08
		0.24	0.58	0.59	0.000	0.01	0.00
		0.33	0.20	0.66	0.107	0.46	0.15
		0.37	0.55	0.69	0.010	0.14	0.05
		0.44	0.39	0.74	0.063	0.36	0.16
		0.44	0.54	0.74	0.021	0.20	0.09
		0.57	0.53	0.84	0.048	0.31	0.18
		0.93	1.00	1.10	0.005	0.10	0.09
		1.00	0.61	1.15	0.148	0.54	0.54
		Total SSE		0.553	Sum	2.900	1.350

You can see with the new prediction, the total SSE has gone down (0.677 to 0.553). That means prediction accuracy has improved.

Step 5: Repeat step 3 and 4 till the time further adjustments to a and b doesn't significantly reduces the error. At that time, we have arrived at the optimal a and b with the highest prediction accuracy.

- This is the Gradient Descent Algorithm. This optimization algorithm and its variants form the core of many machine learning algorithms like Neural Networks and even Deep Learning.

- To see different types of gradient descent algorithms, you can visit <http://ruder.io/optimizing-gradient-descent/index.html#stochasticgradientdescent>

Example numerical on KNN classification

Name	Acid durability	Strength	Class
Type - 1	7	7	Bad
Type - 2	7	4	Bad
Type - 3	3	4	Good
Type - 4	1	4	Good

Test data \rightarrow Acid durability = 3
 Strength = 7
 Class = ?

Name Acid durability Strength Class Distance

Type - 1	7	7	Bad	$\sqrt{(7-3)^2 + (7-7)^2} = 4$
Type - 2	7	4	Bad	$\sqrt{(7-3)^2 + (7-4)^2} = 5$
Type - 3	3	4	Good	$\sqrt{(3-3)^2 + (7-4)^2} = 3$
Type - 4	1	4	Good	$\sqrt{(3-1)^2 + (7-4)^2} = 3.6$

Write the Rank for the records based on the distance between them.

Type Durability Strength class distance Rank

Type-1 7 7 Bad 4 3

-2 7 4 Bad 5 4

-3 3 4 Good 3 1

-4 1 4 Good 3.6 2

For.

$k = 1$

Test data belongs to class "Good".

$k = 2$

Test data belongs to class "Good".

$p = \frac{1}{2}(\alpha + \beta(1-\alpha))$. Bad F F

$\alpha = \frac{1}{2}(\alpha + \beta(1-\alpha))$. Bad F F

$\beta = \frac{1}{2}(\alpha + \beta(1-\alpha))$. Good F F

$\alpha \cdot \beta = \frac{1}{2}(\alpha + \beta(1-\alpha))$. Good F F

Based on the test result, it can be concluded that the test data is correctly classified.

1

A breathalyzer registers someone's blood alcohol content to tell if they are "over the limit" or "under the influence" of alcohol. It is typically used at roadside police stops to determine if someone is legally able to drive.

1. What is the positive class?
 2. What would a recall of 70% mean?
 3. What would a precision of 90% mean?
-
1. The positive case would be detecting someone with blood alcohol content (BAC) over the limit.
 2. A recall of 70% means there is a 70% chance of someone who has a BAC over the limit of setting off the detector. A low recall would mean that people with high BAC are not getting caught when stopped.
 3. A precision of 90% means that 90% of the people who blow a positive result actually have a BAC that is too high. Low precision would mean that a significant fraction of the people who blow a positive result would have a follow-up test (such as a blood test) or a ticket when they were within the legal limit.

2

We are looking to develop a machine learning algorithm to predict whether someone will pay a loan back or not.

1. What is the positive class?
2. What would a recall of 75% mean?
3. What would a precision of 85% mean?

1. The positive class are the borrowers that pay back the loans.
2. 75% recall means that 75% of the borrowers that would pay back the loan are approved by our system. We miss 25% of people that would have paid us back by rejecting them. In general, the problem with a low recall is that we are rejecting customers who we would have paid us back (and for whom we would have made interest).
3. 85% precision means that of all the loans we approve, 85% pay us back. The remaining 15% of approved loans go into default. The problem with a low precision is that we are approving loans that are defaulting.

Should we unlock a phone?

We are building a facial recognition algorithm to allow people to unlock their phone. If the phone recognizes the person as the authorized user, it will unlock the phone. If it doesn't recognize the user, it will prompt them to try again or try an alternative method (such as a passphrase).

1. What is the positive class?
2. What would a recall of 80% mean?
3. What would a precision of 70% mean

1. The positive class is recognizing the user as authorized.
2. 80% recall means 80% of the times the authorized user tries to use this feature, the phone unlocks. The remaining 20% of the time, the authorized user was asked to try again or use a passphrase.
3. 70% precision means that out of all the times the phone was unlocked using this feature, it was unlocked by an authorized user. The remaining 30% of the times it was unlocked, it allowed in someone that was not authorized.

4

Detect malicious programs

When running a program for the first time, we are running some information about the program (such as where it was downloaded, size of the executable, etc) through a classifier. If the program is deemed safe, it will run. If it is deemed unsafe, the user will be prompted to confirm that the program is safe before running.

1. What is the positive class?
2. What does 85% recall mean?
3. What does 75% precision mean?

1. The positive class is "this program is not safe to run"
2. 85% recall means that 85% of unsafe programs are detected, and showed the prompt. The other 15% of malicious programs did not require a prompt.
3. A precision of 75% means that out of all the programs that set off the prompt, 75% of them were malicious. The other 25% of the times the dialog popped up, it was for programs that were fine.

6

You are writing a deduplication algorithm. Its goal is to flag entries in your database that are duplicates of existing records. It is more complicated than checking if two records are identical (Which is an easy problem), but instead tries to assess if differences are meaningful. For example:

- The names *Jane Smith*, *Ms Jane Smith*, and *Jane J. Smith*, may refer to the same person
- The addresses *101 Main St Apt 101*, *101 Main St, Apt 101*, *101 Main Street #101* may all refer to the same street address

Records that our algorithm detects as duplicates will be reviewed, and if we are sure they are duplicates, they are removed.

1. What is the positive class?
2. What does 85% recall mean?
3. What does 75% precision mean?

1. The positive class is the record is a duplicate
2. 85% recall means that 85% of the duplicate records are detected by our algorithm. The remaining 15% of duplicate records are not detected. The problem with a low recall is that we are not ending up with a clean database.
3. 75% precision means that 75% of the flagged records are actually duplicates, while the remaining 25% of the flagged records are not duplicates. The problem with a low precision is that we have a lot of irrelevant records to filter through.

NEUMERICALS SOLVED EXAMPLES

<u>S.No.</u>	<u>Topic</u>	<u>Page No.</u>
1	KNN	2
2	Decision tree	5
3	METRICS	11
4	Naïve Bayes	22
5	K-means clustering	30
6	Hierarchical clustering	40
7	Dimensionality reduction techniques (PCA)	51

KNN Numerical Example (hand computation)

Numerical Example of K Nearest Neighbor Algorithm

Here is step by step on how to compute K-nearest neighbors KNN algorithm:

1. Determine parameter $K = \text{number of nearest neighbors}$
2. Calculate the distance between the query-instance and all the training samples
3. Sort the distance and determine nearest neighbors based on the K -th minimum distance
4. Gather the category y of the nearest neighbors
5. Use simple majority of the category of nearest neighbors as the prediction value of the query instance

Example

We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples

$X_2 = \text{Strength}$

$X_1 = \text{Acid Durability}$	$X_2 = \text{Strength}$	$Y = \text{Classification}$
(seconds)	(kg/square meter)	
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that pass laboratory test with $X_1 = 3$ and $X_2 = 7$. Without another expensive survey, can we guess what the classification of this new tissue is?

1. *Determine parameter $K = \text{number of nearest neighbors}$*

Suppose use $K = 3$

2. *Calculate the distance between the query-instance and all the training samples*

Coordinate of query instance is $(3, 7)$, instead of calculating the distance we compute square distance which is faster to calculate (without square root)

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance (3,7)	
7	7	$(7-3)^2+(7-7)^2=16$	
7	4	$(7-3)^2+(4-7)^2=25$	
3	4	$(3-3)^2+(4-7)^2=9$	
1	4	$(1-3)^2+(4-7)^2=13$	

3. Sort the distance and determine nearest neighbors based on the K-th minimum distance				
X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Square Distance to query instance	Rank minimum (3, 7) distance	Is it included in 3- Nearest neighbors?
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes
7	4	$(7-3)^2+(4-7)^2=25$	4	No
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes

4. Gather the category Y of the nearest neighbors. Notice in the second row last column that the category of nearest neighbor (Y) is not included because the rank of this data is more than 3 (=K).

X1 = Acid Durability	X2 =Strength	Square Distance to query instance (3, 7)	Rank minimum	Is it included in 3-Nearest neighbors?	Y = Category of nearest Neighbor
(seconds)	(kg/square meter)		distance		
7	7	$(7-3)^2+(7-7)^2=16$	3	yes	Bad
7	4	$(7-3)^2+(4-7)^2=25$	4	no	-
3	4	$(3-3)^2+(4-7)^2=9$	1	yes	good
1	4	$(1-3)^2+(4-7)^2=13$	2	yes	good

5. Use simple majority of the category of nearest neighbors as the prediction value of the query instance

We have 2 good and 1 bad, since $2 > 1$ then we conclude that a new paper tissue that pass laboratory test with $X1 = 3$ and $X2 = 7$ is included in **Good** category.

Decision Tree

Data set

For instance, the following table informs about decision making factors to play tennis at outside for previous 14 days.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We can summarize the ID3 algorithm as illustrated below

$$\text{Entropy}(S) = \sum - p(I) \cdot \log_2 p(I)$$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum [p(S|A) \cdot \text{Entropy}(S|A)]$$

.Entropy

We need to calculate the entropy first. Decision column consists of 14 instances and includes two labels: yes and no. There are 9 decisions labeled yes, and 5 decisions labeled no.

$$\text{Entropy}(\text{Decision}) = - p(\text{Yes}) \cdot \log_2 p(\text{Yes}) - p(\text{No}) \cdot \log_2 p(\text{No})$$

$$\text{Entropy}(\text{Decision}) = -(9/14) \cdot \log_2(9/14) - (5/14) \cdot \log_2(5/14) = 0.940$$

Now, we need to find the most dominant factor for decisioning.

Wind factor on decision

$$\text{Gain}(\text{Decision}, \text{Wind}) = \text{Entropy}(\text{Decision}) - \sum [p(\text{Decision}|\text{Wind}) \cdot \text{Entropy}(\text{Decision}|\text{Wind})]$$

Wind attribute has two labels: weak and strong. We would reflect it to the formula.

$$\begin{aligned} \text{Gain}(\text{Decision}, \text{Wind}) &= \text{Entropy}(\text{Decision}) - [p(\text{Decision}|\text{Wind}=\text{Weak}) \cdot \\ &\quad \text{Entropy}(\text{Decision}|\text{Wind}=\text{Weak})] - [p(\text{Decision}|\text{Wind}=\text{Strong}) \cdot \\ &\quad \text{Entropy}(\text{Decision}|\text{Wind}=\text{Strong})] \end{aligned}$$

Now, we need to calculate $(\text{Decision}|\text{Wind}=\text{Weak})$ and $(\text{Decision}|\text{Wind}=\text{Strong})$ respectively.

Weak wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
13	Overcast	Hot	Normal	Weak	Yes

There are 8 instances for weak wind. Decision of 2 items are no and 6 items are yes as illustrated below.

$$1- \text{Entropy}(\text{Decision}|\text{Wind}=\text{Weak}) = - p(\text{No}) \cdot \log_2 p(\text{No}) - p(\text{Yes}) \cdot \log_2 p(\text{Yes})$$

$$2- \text{Entropy}(\text{Decision}|\text{Wind}=\text{Weak}) = -(2/8) \cdot \log_2(2/8) - (6/8) \cdot \log_2(6/8) = 0.811$$

Strong wind factor on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
2	Sunny	Hot	High	Strong	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
14	Rain	Mild	High	Strong	No

Here, there are 6 instances for strong wind. Decision is divided into two equal parts.

$$1- \text{Entropy}(\text{Decision}|\text{Wind}=\text{Strong}) = - p(\text{No}) \cdot \log_2 p(\text{No}) - p(\text{Yes}) \cdot \log_2 p(\text{Yes})$$

$$2- \text{Entropy}(\text{Decision}|\text{Wind}=\text{Strong}) = -(3/6) \cdot \log_2(3/6) - (3/6) \cdot \log_2(3/6) = 1$$

Now, we can turn back to Gain(Decision, Wind) equation.

$$\begin{aligned} \text{Gain}(\text{Decision}, \text{Wind}) &= \text{Entropy}(\text{Decision}) - [\text{p}(\text{Decision}|\text{Wind}=\text{Weak}) \cdot \\ &\quad \text{Entropy}(\text{Decision}|\text{Wind}=\text{Weak})] - [\text{p}(\text{Decision}|\text{Wind}=\text{Strong}) \cdot \\ &\quad \text{Entropy}(\text{Decision}|\text{Wind}=\text{Strong})] = 0.940 - [(8/14) \cdot 0.811] - [(6/14) \cdot 1] = 0.048 \end{aligned}$$

Calculations for wind column is over. Now, we need to apply same calculations for other columns to find the most dominant factor on decision.

Other factors on decision

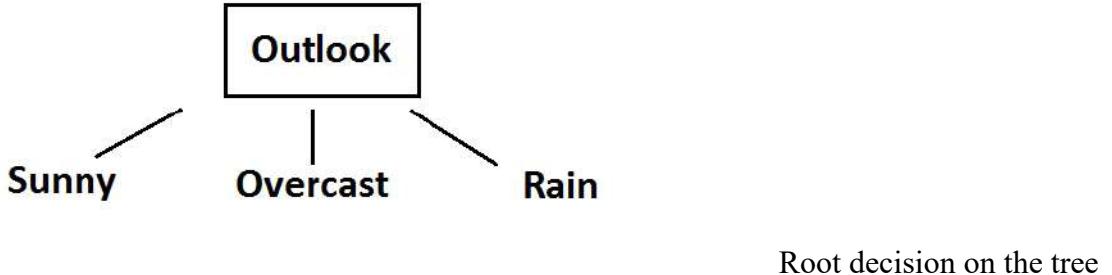
We have applied similar calculation on the other columns.

$$1- \text{Gain}(\text{Decision}, \text{Outlook}) = 0.246$$

$$2- \text{Gain}(\text{Decision}, \text{Temperature}) = 0.029$$

$$3- \text{Gain}(\text{Decision}, \text{Humidity}) = 0.151$$

As seen, outlook factor on decision produces the highest score. That's why, outlook decision will appear in the root node of the tree.



Now, we need to test dataset for custom subsets of outlook attribute.

Overcast outlook on decision

Basically, decision will always be yes if outlook were overcast.

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Sunny outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Here, there are 5 instances for sunny outlook. Decision would be probably 3/5 percent no, 2/5 percent yes.

$$1- \text{Gain}(\text{Outlook}=\text{Sunny}|\text{Temperature}) = 0.570$$

$$2- \text{Gain}(\text{Outlook}=\text{Sunny}|\text{Humidity}) = 0.970$$

$$3- \text{Gain}(\text{Outlook}=\text{Sunny}|\text{Wind}) = 0.019$$

Now, humidity is the decision because it produces the highest score if outlook were sunny.

At this point, decision will always be no if humidity were high.

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

On the other hand, decision will always be yes if humidity were normal

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Finally, it means that we need to check the humidity and decide if outlook were sunny.

Rain outlook on decision

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

1- Gain(Outlook=Rain | Temperature) = 0.01997309402197489

2- Gain(Outlook=Rain | Humidity) = 0.01997309402197489

3- Gain(Outlook=Rain | Wind) = 0.9709505944546686

Here, wind produces the highest score if outlook were rain. That's why, we need to check wind attribute in 2nd level if outlook were rain.

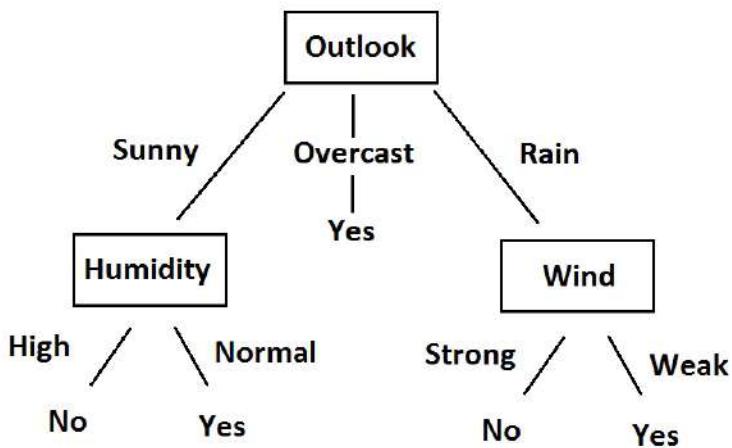
So, it is revealed that decision will always be yes if wind were weak and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

What's more, decision will be always no if wind were strong and outlook were rain.

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

So, decision tree construction is over. We can use the following rules for decisioning.



Final version of decision tree

Accuracy

- Ratio between the number of correct predictions and total number of predictions

$$\text{accuracy} = \frac{\# \text{ correct predictions}}{\# \text{ total data points}}$$

- Example: Suppose we have 100 examples in the positive class and 200 examples in the negative class. Our model declares 80 out of 100 positives as positive correctly and 195 out of 200 negatives as negative correctly.

- So, accuracy is $= (80 + 195)/(100 + 200) = 91.7\%$

1- Confusion Matrix

- One of the key concept in classification performance is **confusion matrix** which is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.

Class 1 Predicted

Class 2 predicted

- Here,
- Class 1 : Pos
- Class 2 : Neg
- **Definition**
- Positive (P)
- Negative (N)
- True Positive
- False Negative
- True Negative
- False Positive

Let's go through this with an example. Let's assume we are building a binary classification to classify cat images from non-cat images. And let's assume our test set has 1100 images (1000 non-cat images, and 100 cat images), with the below confusion matrix.

		Actual Class	
		Cat	Non-Cat
Predicted Class	Cat	90	60
	Non-Cat	10	940

- **Out of 100 cat images** the model has predicted 90 of them correctly and has mis-classified 10 of them. If we refer to the “cat” class as positive and the non-cat class as negative class, then 90 samples predicted as cat are considered as as **true-positive**, and the 10 samples predicted as non-cat are **false negative**.
- **Out of 1000 non-cat images**, the model has classified 940 of them correctly, and mis-classified 60 of them. The 940 correctly classified samples are referred as **true-negative**, and those 60 are referred as **false-positive**
- As we can see diagonal elements of this matrix denote the correct prediction for different classes, while the off-diagonal elements denote the samples which are mis-classified.

Classification Accuracy

- Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the **number of correct predictions divided by the total number of predictions**, multiplied by 100. So in the above example, out of 1100 samples 1030 are predicted correctly, resulting in a classification accuracy of:
- **Classification accuracy**= $(90+940)/(1000+100) = 1030/1100 = 93.6\%$

- Therefore we need to look at class specific performance metrics too. Precision is one of such metrics, which is defined as:
- **Precision= True_Positive / (True_Positive+ False_Positive)**
- The precision of Cat and Non-Cat class in above example can be calculated as:
- **Precision_cat= #samples correctly predicted cat/#samples predicted as cat = $90/(90+60) = 60\%$**
- **Precision_NonCat= $940/950 = 98.9\%$**
- As we can see the model has much higher precision in predicting non-cat samples, versus cats. This is not surprising, as model has seen more examples of non-cat images during training, making it better in classifying that class.

Recall

- Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:
- **Recall= True_Positive / (True_Positive+ False_Negative)**
- Therefore, for our example above, the recall rate of cat and non-cat classes can be found as:
- **Recall_cat= $90/100 = 90\%$**
- **Recall_NonCat= $940/1000 = 94\%$**

3- Precision

- There are many cases in which classification accuracy is not a good indicator of your model performance (for example, if your class distribution is imbalanced, or if you care more about one class than others). In this case, even if you predict the most frequent class you would get a high accuracy, which does not make sense at all (because your model is not learning anything, just predicting everything as the top class). For example, in the non-cat classification above, if the model always predicted non-cat, it would result in a $1000/1100 = 90.9\%$ accuracy.

Sensitivity and Specificity

- Sensitivity and specificity are two other popular metrics mostly used in medical and biology related fields, and are defined as:
- **True Positive Rate (Sensitivity)** : True Positive Rate is defined as $TP / (FN+TP)$. True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.
- **False Positive Rate (Specificity)** : False Positive Rate is defined as $FP / (FP+TN)$. False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

F-Measure

One measure of performance that takes into account both recall and precision

Harmonic mean of recall and precision:

$$F_1 = 2 \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Compared to arithmetic mean, both need to be high for harmonic mean to be high

R-squared (R^2)

- It measures the proportion of the variation in your dependent variable explained by all of your independent variables in the model.
- It assumes that every independent variable in the model helps to explain variation in the dependent variable.
- In reality, some independent variables (predictors) don't help to explain dependent (target) variable. In other words, some variables do not contribute in predicting target variable.

- Mathematically, R-squared is calculated by dividing sum of squares of residuals (**SSres**) by total sum of squares (**SStot**) and then subtract it from 1.
- In this case, SStot measures total variation. **SSreg** measures explained variation and SSres measures unexplained variation.

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \equiv 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

\downarrow

$$R^2 = \frac{SS_{reg}}{SS_{tot}}$$

R-Squared is also called **coefficient of determination**. It lies between **0%** and **100%**. A r-squared value of 100% means the model explains all the variation of the target variable. And a value of 0% measures zero predictive power of the model. **Higher R-squared value, better the model.**

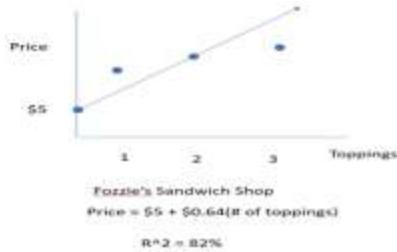
Example

- To help explain what exactly R-squared means, I'm going to tell you about two sandwich shops in my town, Jimmy's Sandwich Shop and Fozzie's Sandwich Emporium. At Jimmy John's they charge \$5 for a sandwich and \$1.00 for each additional topping (i.e. double meat \$1.00, double cheese \$1.00, or double lettuce for \$1.00). At Fozzie's they also charge \$5.00 for a sandwich, but different topping prices (i.e. double meat \$1.50, double cheese \$0.75, or double lettuce \$0.50).

- Now, I sat outside Jimmy's for 7 days and took a survey from all the customers leaving and asked how many toppings they ordered and what was the total price. The first customer, Joe, ordered 3 toppings and the price was \$8.00. The second customer, Suzie, ordered 4 toppings and the price was \$9.00. The third customer, Pat, ordered 5 toppings and the price was \$10.00. I ended up collecting 100 samples. At the end of the week, I plotted these points on a chart and found out that I could explain the price using the following equation Price = \$5 + \$1.00(# of toppings) and figured out that
- r-squared = 100%.



- The next day I did the same thing and sat outside Fozzie's for 7 days and collected another 100 customer's orders. And at the end of the week I went back home, plotted the points on a chart and found that I could explain the price using the following equation. Price = \$5 + \$0.64(# of toppings) and R² = 82%.



- To understand what r-square tells us you must understand the word variability. When I say variability, you should think of the word "differs." We know that prices of sandwiches vary, or they differ based on the number of toppings. What R² tells us for Jimmy's Sandwich shop is that 100% of the differences in price can be explained by the number toppings. Or in other words, the sole reason that prices differ at Jimmy's, can be explained by the number of toppings. Again, 100% of the variability in sandwich price is explained by the variability of toppings. Prices differ because toppings differ.
- At Fozzie's, it's a different story. While the model does explain 82% of how the price differed, it doesn't explain all the price differences. There are other reasons besides the number of toppings why two sandwiches might cost differently. Again, 82% of the price differences can be explained by the differences in the number of toppings. The other 28% are in the residuals. They are unexplained. The model doesn't explain that part. Again, what R² tells you is that the percent in the variability in Y that is explained by the model.

Practical Application

- Investors use the r-squared measurement to compare a portfolio's performance with the broader market and predict trends that might occur in the future. For instance, let's assume that an investor wants to purchase an investment fund that is strongly correlated with the S&P 500. The investor would look for a fund that has an r-squared value close to 1. The closer the value gets to 1, the more correlated it is.
- Let's assume the investor can choose between three funds with R² values of .5, .7, and .9. The investor should pick the .9 fund because its performance is most correlated to the S&P 500.

Adjusted R-Squared

- It measures the proportion of variation explained by only those independent variables that really help in explaining the dependent variable.
- It penalizes you for adding independent variable that do not help in predicting the dependent variable.

- Adjusted R-squared value can be calculated based on value of r-squared, number of independent variables (predictors), total sample size.

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}$$

where

R^2 = sample R-square
 p = Number of predictors
 N = Total sample size.

- Problem Statement:**

- A fund has a sample R-squared value close to 0.5 and it is doubtlessly offering higher risk adjusted returns with the sample size of 50 for 5 predictors. Find Adjusted R square value.
- Solution:**
- Sample size = 50 Number of predictor = 5 Sample R - square = 0.5. Substitute the qualities in the equation,

$$\begin{aligned} R^2_{\text{adj}} &= 1 - \left[\frac{(1 - 0.5^2)(50 - 1)}{50 - 5 - 1} \right] \\ &= 1 - (0.75) \times \frac{49}{44}, \\ &= 1 - 0.8352, \\ &= 0.1648 \end{aligned}$$

Difference between R-square and Adjusted R-square

- Every time you add a independent variable to a model, the **R-squared increases**, even if the independent variable is insignificant. It never declines.
- Whereas **Adjusted R-squared** increases only when independent variable is significant and affects dependent variable.

- In the table below, adjusted r-squared is maximum when we included two variables. It declines when third variable is added. Whereas r-squared increases when we included third variable. It means third variable is insignificant to the model.

Variables	R-Squared	Adjusted R- Squared
1	67.5	67.1
2	85.9	84.2
3	88.9	81.7

- Adjusted r-squared can be negative when r-squared is close to zero.
- Adjusted r-squared value always be less than or equal to r-squared value.
- Adjusted R-square should be used to compare models with different numbers of independent variables. Adjusted R-square should be used while selecting important predictors (independent variables) for the regression model.

Naive Bayes Example

Say you have 1000 fruits which could be either ‘banana’, ‘orange’ or ‘other’. These are the 3 possible classes of the Y variable.

We have data for the following X variables, all of which are binary (1 or 0).

- Long
- Sweet
- Yellow

The first few rows of the training dataset look like this:

Fruit	Long (x1)	Sweet (x2)	Yellow (x3)
Orange	0	1	0
Banana	1	0	1
Banana	1	1	1
Other	1	1	0

For the sake of computing the probabilities, let's aggregate the training data to form a counts table like this.

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

So the objective of the classifier is to predict if a given fruit is a 'Banana' or 'Orange' or 'Other' when only the 3 features (long, sweet and yellow) are known.

Let's say you are given a fruit that is: Long, Sweet and Yellow, can you predict what fruit it is?

This is the same of predicting the Y when only the X variables in testing data are known. Let's solve it by hand using Naive Bayes.

The idea is to compute the 3 probabilities, that is the probability of the fruit being a banana, orange or other. Whichever fruit type gets the highest probability wins

All the information to calculate these probabilities is present in the above tabulation.

Step 1: Compute the 'Prior' probabilities for each of the class of fruits.

That is, the proportion of each fruit class out of all the fruits from the population. You can provide the 'Priors' from prior information about the population. Otherwise, it can be computed from the training data.

For this case, let's compute from the training data. Out of 1000 records in training data, you have 500 Bananas, 300 Oranges and 200 Others. So the respective priors are 0.5, 0.3 and 0.2.

$$P(Y=\text{Banana}) = 500 / 1000 = 0.50$$

$$P(Y=Orange) = 300 / 1000 = 0.30$$

$$P(Y=Other) = 200 / 1000 = 0.20$$

Step 2: Compute the probability of evidence that goes in the denominator.

This is nothing but the product of P of Xs for all X. This is an optional step because the denominator is the same for all the classes and so will not affect the probabilities.

$$P(x_1=Long) = 500 / 1000 = 0.50$$

$$P(x_2=Sweet) = 650 / 1000 = 0.65$$

$$P(x_3=Yellow) = 800 / 1000 = 0.80$$

Step 3: Compute the probability of likelihood of evidences that goes in the numerator.

It is the product of conditional probabilities of the 3 features. If you refer back to the formula, it says $P(X_1 | Y=k)$. Here X_1 is 'Long' and k is 'Banana'. That means the probability the fruit is 'Long' given that it is a Banana. In the above table, you have 500 Bananas. Out of that 400 is long. So, $P(Long | Banana) = 400/500 = 0.8$.

Here, I have done it for Banana alone.

Probability of Likelihood for Banana

$$P(x_1=Long | Y=Banana) = 400 / 500 = 0.80$$

$$P(x_2=Sweet | Y=Banana) = 350 / 500 = 0.70$$

$$P(x_3=Yellow | Y=Banana) = 450 / 500 = 0.90$$

So, the overall probability of Likelihood of evidence for Banana = $0.8 * 0.7 * 0.9 = 0.504$

Step 4: Substitute all the 3 equations into the Naive Bayes formula, to get the probability that it is a banana.

Step 4: If a fruit is ‘Long’, ‘Sweet’ and ‘Yellow’, what fruit is it?

$$P(\text{Banana} | \text{Long, Sweet and Yellow}) = \frac{P(\text{Long} | \text{Banana}) * P(\text{Sweet} | \text{Banana}) * P(\text{Yellow} | \text{Banana}) * P(\text{banana})}{P(\text{Long}) * P(\text{Sweet}) * P(\text{Yellow})}$$

$$= \frac{0.8 * 0.7 * 0.9 * 0.5}{P(\text{Evidence})} = 0.252 / P(\text{Evidence})$$

$$P(\text{Orange} | \text{Long, Sweet and Yellow}) = 0, \text{ because } P(\text{Long} | \text{Orange}) = 0$$

$$P(\text{Other Fruit} | \text{Long, Sweet and Yellow}) = 0.01875 / P(\text{Evidence})$$

Answer: Banana - Since it has highest probability amongst the 3 classes

Similarly, you can compute the probabilities for ‘Orange’ and ‘Other fruit’. The denominator is the same for all 3 cases, so it’s optional to compute.

Clearly, Banana gets the highest probability, so that will be our predicted class.

What is Laplace Correction?

The value of $P(\text{Orange} | \text{Long, Sweet and Yellow})$ was zero in the above example, because, $P(\text{Long} | \text{Orange})$ was zero. That is, there were no ‘Long’ oranges in the training data.

It makes sense, but when you have a model with many features, the entire probability will become zero because one of the feature’s value was zero. To avoid this, we increase the count of the variable with zero to a small value (usually 1) in the numerator, so that the overall probability doesn’t become zero.

This correction is called ‘Laplace Correction’. Most Naive Bayes model implementations accept this or an equivalent form of correction as a parameter.

ANOTHER TEXT EXAMPLE (NAÏVE BAYES)

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

which tag does the sentence *A very close game* belong to?

- Since Naive Bayes is a probabilistic classifier, we want to calculate the probability that the sentence "A very close game" is Sports and the probability that it's Not Sports. Then, we take the largest one.
- Written mathematically, what we want higher of the two
 1. $P(\text{Sports} \mid \text{a very close, game})$ — the probability that the tag of a sentence is Sports given that the sentence is "A very close game".
 2. $P(\text{not Sports} \mid \text{a very close, game})$ — the probability that the tag of a sentence is Not Sports given that the sentence is "A very close game".

Feature Engineering

- The first thing we need to do when creating a machine learning model is to decide what to use as features. We call **features** the pieces of information that we take from the text and give to the algorithm so it can work its magic. For example, if we were doing classification on health, some features could be a person's height, weight, gender, and so on. We would exclude things that maybe are known but aren't useful to the model, like a person's name or favorite color.
- In this case though, we don't even have numeric features. We just have text. We need to somehow convert this text into numbers that we can do calculations on. We use **word frequencies**.

Bayes' Theorem is useful when working with conditional probabilities (like we are doing here), because it provides us with a way to reverse them:

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

In our case, we have $P(\text{sports}|\text{a very close game})$, so using this theorem we can reverse the conditional probability:

$$P(\text{sports}|\text{a very close game}) = \frac{P(\text{a very close game}|\text{sports}) \times P(\text{sports})}{P(\text{a very close game})}$$

Since for our classifier we're just trying to find out which tag has a bigger probability, we can discard the divisor –which is the same for both tags– and just compare

$$P(\text{a very close game}|\text{Sports}) \times P(\text{Sports})$$

with

$$P(\text{a very close game}|\text{Not Sports}) \times P(\text{Not Sports})$$

Being Naive

- So here comes the *Naive* part: we assume that every word in a sentence is **independent** of the other ones. This means that we're no longer looking at entire sentences, but rather at individual words. So for our purposes, “this was a fun party” is the same as “this party was fun” and “party fun was this”.

We write this as:

$$P(\text{a very close game}) = P(a) \times P(\text{very}) \times P(\text{close}) \times P(\text{game})$$

- Above assumption is very strong but super useful. It's what makes this model work well with little data or data that may be mislabeled. The next step is just applying this to what we had before:

$$P(a \text{ very close game} | \text{Sports}) = P(a | \text{Sports}) \times P(\text{very} | \text{Sports}) \times P(\text{close} | \text{Sports}) \times \\ P(\text{game} | \text{Sports})$$

Calculating Probabilities

- First, we calculate the *a priori* probability of each tag: for a given sentence in our training data, the probability that it is *Sports* $P(\text{Sports})$ is %. Then, $P(\text{Not Sports})$ is %. That's easy enough.
- Then, calculating $P(\text{game} | \text{Sports})$ means counting how many times the word "game" appears in Sports texts (2) divided by the total number of words in sports (11). Therefore, $P(\text{game} | \text{Sports}) = 2/11$.

Laplace Smoothing

- However, we run into a problem here: "close" doesn't appear in any Sports text! That means that $P(\text{close} | \text{Sports}) = 0$. This is rather inconvenient since we are going to be multiplying it with the other probabilities, so we'll end up with $P(a | \text{Sports}) * P(\text{very} | \text{Sports}) * 0 * P(\text{game} | \text{Sports})$. This equals 0, since in a multiplication if one of the terms is zero, the whole calculation is nullified. Doing things this way simply doesn't give us any information at all, so we have to find a way around.

By using something called **Laplace Smoothing** we add 1 to every count so it's never zero. To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1.

- In our case, the possible words are ['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match'].
- Since the number of possible words is 14

$P(game|sports) = \frac{2+1}{11+14}$. The full results are:

Word	P(word Sports)	P(word Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

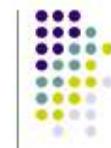
Final Result

Now we just multiply all the probabilities, and see who is bigger:

$$\begin{aligned} & P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times \\ & P(Sports) \\ & = 2.76 \times 10^{-5} \\ & = 0.0000276 \end{aligned}$$

$$\begin{aligned} & P(a|Not\ Sports) \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times P(game|Not\ Sports) \times \\ & P(Not\ Sports) \\ & = 0.572 \times 10^{-5} \\ & = 0.00000572 \end{aligned}$$

Our classifier gives "A very close game" the Sports tag. Great!!!

K-
Means
Clustering

- **Step 1:** Begin with a decision on the value of k = number of clusters .
- **Step 2:** Put any initial partition that classifies the data into k clusters. You may assign the training samples randomly, or systematically as the following:
 1. Take the first k training sample as single-element clusters
 2. Assign each of the remaining $(N-k)$ training sample to the cluster with the nearest centroid. After each assignment, recompute the centroid of the gaining cluster.



- **Step 3:** Take each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the cluster losing the sample.
- **Step 4.** Repeat step 3 until convergence is achieved, that is until a pass through the training sample causes no new assignments.

A Simple example showing the implementation of k-means algorithm (using K=2)



Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Step 1:

Initialization: Randomly we choose following two centroids ($k=2$) for two clusters.



In this case the 2 centroid are: $m_1=(1.0, 1.0)$ and $m_2=(5.0, 7.0)$. (Individuals who are farthest apart using Euclidean distance)

Individual	Variable 1	Variable 2
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Individual	Mean Vector
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)



The remaining individuals are now examined in sequence and allocated to the cluster to which they are closest, in terms of Euclidean distance to the cluster mean. The mean vector is recalculated each time a new member is added. This leads to the following series of steps:

Step	Cluster 1		Cluster 2	
	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)



Now the initial partition has changed, and the two clusters at this stage having the following characteristics:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)



Step 3:

- Now using these centroids we compute the Euclidean distance of each object, as shown in table.
- Therefore, the new clusters are:
{1,2} and {3,4,5,6,7}
- Next centroids are:
 $m_1=(1.3, 1.5)$ and $m_2 = (3.9, 5.1)$

Individual	Centroid 1	Centroid 2
1	1.57	5.38
2	0.47	4.28
3	2.04	1.78
4	5.64	1.84
5	3.15	0.73
6	3.78	0.54
7	2.74	1.08



$$= (4.12, 5.38)$$

$$d(m_1, 2) = \sqrt{(1.0 - 1.5)^2 + (1.0 - 2.0)^2} = 1.12$$

$$d(m_2, 2) = \sqrt{(5.0 - 1.5)^2 + (7.0 - 2.0)^2} = 6.10$$



Only individual 3 is nearer to the mean of the opposite cluster (Cluster 2) than its own (Cluster 1). In other words, each individual's distance to its own cluster mean should be smaller than the distance to the other cluster's mean (which is not the case with individual 3). Thus, individual 3 is relocated to Cluster 2 resulting in the new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

- **Step 4 :**
The clusters obtained are:
 $\{1,2\}$ and $\{3,4,5,6,7\}$



- Therefore, there is no change in the cluster.
- Thus, the algorithm comes to a halt here and final result consist of 2 clusters $\{1,2\}$ and $\{3,4,5,6,7\}$.

Individual	Centroid 1	Centroid 2
1	0.56	5.02
2	0.56	3.92
3	3.05	1.42
4	6.66	2.20
5	4.16	0.41
6	4.78	0.81
7	3.75	0.72

Real-Life Numerical Example of K-Means Clustering



We have 4 medicines as our training data points object and each medicine has 2 attributes. Each attribute represents coordinate of the object. We have to determine which medicines belong to cluster 1 and which medicines belong to the other cluster.

Object	Attribute 1 weight index	(X): Attribute 2 (Y): pH
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

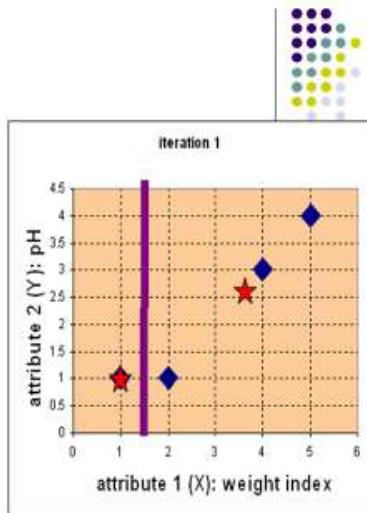
ANOTHER EXAMPLE FOR K-Means Clustering

Step 2:

- **Objects clustering :** We assign each object based on the minimum distance.
- Medicine A is assigned to group 1, medicine B to group 2, medicine C to group 2 and medicine D to group 2.
- The elements of Group matrix below is 1 if and only if the object is assigned to that group.

$$\mathbf{G}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{array}{l} \text{group -1} \\ \text{group -2} \end{array}$$

A B C D



- **Iteration-1. Objects-Centroids distances :** The next step is to compute the distance of all objects to the new centroids.
- Similar to step 2, we have distance matrix at iteration 1 is

$$\mathbf{D}^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1, 1) \text{ group -1} \\ \mathbf{c}_2 = (\frac{11}{3}, \frac{8}{3}) \text{ group -2} \end{array}$$

A B C D

$$\begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix} \quad X$$

$$\begin{bmatrix} 1 & 1 & 3 & 4 \end{bmatrix} \quad Y$$

- **Objects-Centroids distance :** we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 is

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{array}{l} \mathbf{c}_1 = (1, 1) \text{ group -1} \\ \mathbf{c}_2 = (2, 1) \text{ group -2} \end{array}$$

A B C D

$$\begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix} \quad X$$

$$\begin{bmatrix} 1 & 1 & 3 & 4 \end{bmatrix} \quad Y$$

- Each column in the distance matrix symbolizes the object.
- The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid.
- For example, distance from medicine C = (4, 3) to the first centroid is , and its distance to the second centroid is , etc.

$\mathbf{c}_1 = (2, 1)$



□ **Iteration-2. Objects-Centroids distances :**

Repeat step 2 again, we have new distance matrix at iteration 2 as

$$\mathbf{D}^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \\ A & B & C & D \\ [1 & 2 & 4 & 5] & X \\ [1 & 1 & 3 & 4] & Y \end{bmatrix}$$

□ **Iteration-2. Objects clustering:** Again, we assign each object based on the minimum distance.

$$\mathbf{G}^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ A & B & C & D \end{bmatrix}$$

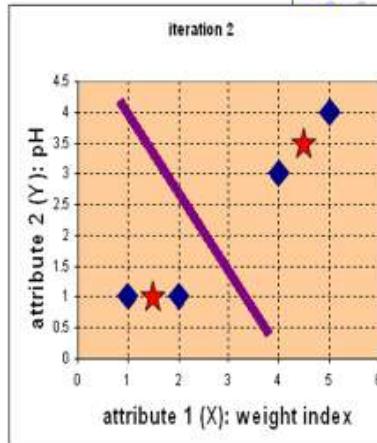
□ We obtain result that $\mathbf{G}^2 = \mathbf{G}^1$. Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore.

□ Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed..

□ **Iteration-1. Objects clustering:**

clustering: Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain. The Group matrix is shown below

$$\mathbf{G}^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$



□ **Iteration 2. determine centroids:**

Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are

and

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2} \right) = (1.5, 1.0)$$

$$c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2} \right) = (4.5, 3.5)$$

ANOTHER EXAMPLE OF K-MEANS (One Dimension)

Suppose we want to group the visitors to a website using just their age (one-dimensional space) as follows:

$$n = 19$$

15,15,16,19,19,20,20,21,22,28,35,40,41,42,43,44,60,61,65

Initial clusters (random centroid or average):

$$k = 2$$

$$c_1 = 16$$

$$c_2 = 22$$



We get the final grouping as the results as:

Object	Feature1(X): weight index	Feature2 (Y): pH	Group (result)
Medicine A	1	1	1
Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2

$$\text{Distance 1} = |x_i - c_1|$$

$$\text{Distance 2} = |x_i - c_2|$$

Iteration 1:

$$c_1 = 15.33$$

$$c_2 = 36.25$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	15.33
15	16	22	1	7	1	
16	16	22	0	6	1	
19	16	22	9	3	2	
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

Iteration 2:

$$c_1 = 18.56$$

$$c_2 = 45.90$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	18.56
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	

19	15.33	36.25	3.67	17.25	1	
19	15.33	36.25	3.67	17.25	1	
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

45.9

Iteration 3:

$$c_1 = 19.50$$

$$c_2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	
20	18.56	45.9	1.44	25.9	1	
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	
43	18.56	45.9	24.44	2.9	2	
44	18.56	45.9	25.44	1.9	2	

19.50

47.89

60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

Iteration 4:

$$c_1 = 19.50$$

$$c_2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	19.50
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	
19	19.5	47.89	0.50	28.89	1	
19	19.5	47.89	0.50	28.89	1	
20	19.5	47.89	0.50	27.89	1	
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	47.89
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

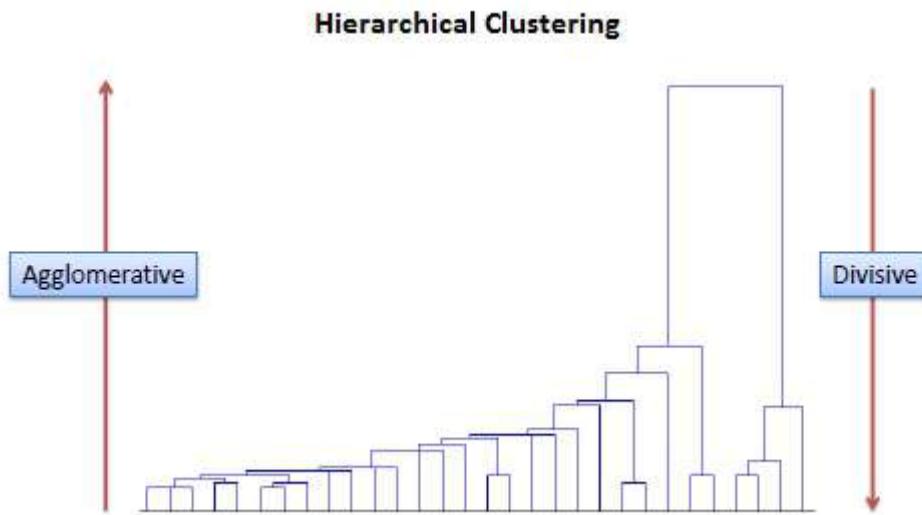
No change between iterations 3 and 4 has been noted. By using clustering, 2 groups have been identified 15-28 and 35-65. The initial choice of centroids can affect the output clusters, so the algorithm is often run multiple times with different starting conditions in order to get a fair view of what the clusters should be.

Videos Tutorials

1. K Means Clustering Algorithm: URL:
https://www.youtube.com/watch?v=1XqG0kaJVHY&feature=emb_logo
2. K Means Clustering Algorithm: URL: <https://www.youtube.com/watch?v=ElIUEPCIzM>

Hierarchical Clustering

Hierarchical clustering is another unsupervised learning algorithm that is used to group together the unlabeled data points having similar characteristics. Hierarchical clustering algorithms falls into following two categories.



Agglomerative hierarchical algorithms – In agglomerative hierarchical algorithms, each data point is treated as a single cluster and then successively merge or agglomerate (bottom-up approach) the pairs of clusters. The hierarchy of the clusters is represented as a dendrogram or tree structure.

Divisive hierarchical algorithms – On the other hand, in divisive hierarchical algorithms, all the data points are treated as one big cluster and the process of clustering involves dividing (Top-down approach) the one big cluster into various small clusters.

Agglomerative clustering

In agglomerative or bottom-up clustering method we assign each observation to its own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters. Finally, repeat steps 2 and 3 until there is only a single cluster left. The related algorithm is shown below.

Given:

A set X of objects $\{x_1, \dots, x_n\}$
A distance function $dist(c_1, c_2)$

for $i = 1$ to n

$c_i = \{x_i\}$

end for

$C = \{c_1, \dots, c_n\}$

$l = n+1$

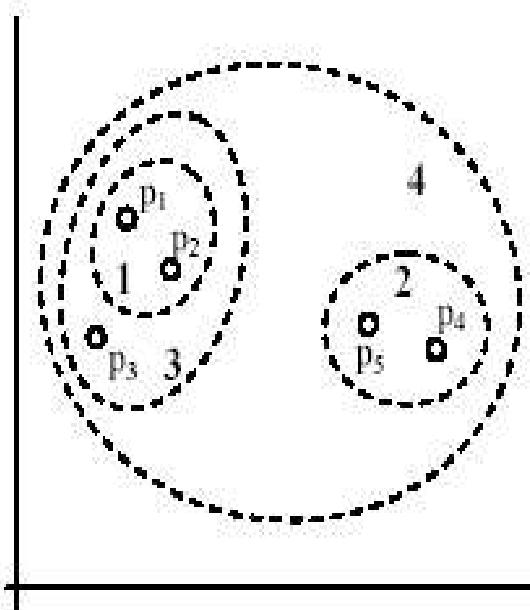
while $C.size > 1$ **do**

- $(c_{min1}, c_{min2}) = \text{minimum } dist(c_p, c_j) \text{ for all } c_p, c_j \text{ in } C$
- remove c_{min1} and c_{min2} from C
- add $\{c_{min1}, c_{min2}\}$ to C
- $l = l + 1$

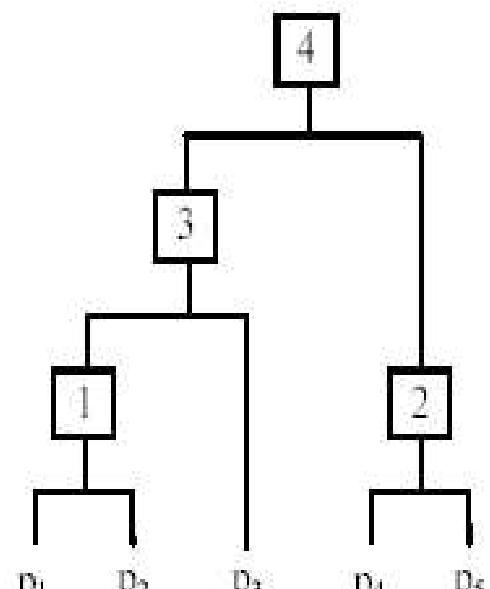
end while

Before any clustering is performed, it is required to determine the proximity matrix containing the distance between each object. Then, the matrix is updated to display the distance between each cluster. The following diagram illustrates how the distance between each cluster is measured.

An example: working of the algorithm



(A). Nested clusters



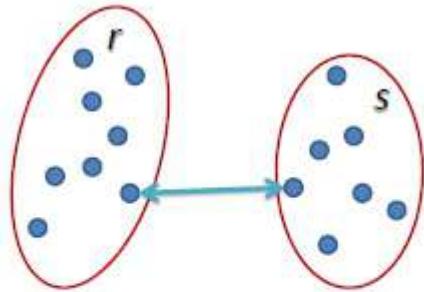
(B) Dendrogram

Measuring the distance of two clusters

- A few ways to measure distances of two clusters.
- Results in different variations of the algorithm.
 - Single link
 - Complete link
 - Average link
 - Centroids
 - ...

Single link method

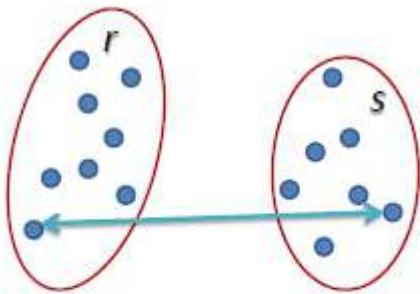
In single linkage hierarchical clustering, the distance between two clusters is defined as the shortest distance between two points in each cluster. For example, the distance between clusters “r” and “s” to the left is equal to the length of the arrow between their two closest points.



$$L(r, s) = \min(D(x_{ri}, x_{sj}))$$

Complete link method

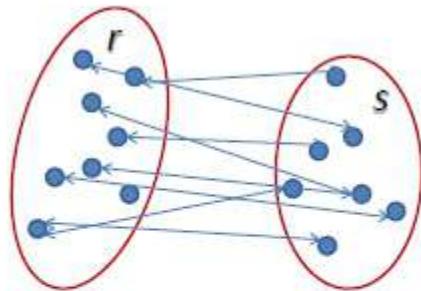
In complete linkage hierarchical clustering, the distance between two clusters is defined as the longest distance between two points in each cluster. For example, the distance between clusters “r” and “s” to the left is equal to the length of the arrow between their two furthest points.



$$L(r, s) = \max(D(x_{ri}, x_{sj}))$$

Average link method:

In average linkage hierarchical clustering, the distance between two clusters is defined as the average distance between each point in one cluster to every point in the other cluster. For example, the distance between clusters “r” and “s” to the left is equal to the average length each arrow between connecting the points of one cluster to the other.



$$L(r, s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Centroid method:

- In this method, the distance between two clusters is the distance between their centroids

The complexity

- All the algorithms are at least $O(n^2)$. n is the number of data points.
- Single link can be done in $O(n^2)$.
- Complete and average links can be done in $O(n^2 \log n)$.
- Due to the complexity, hard to use for large data sets.
 - Sampling

- Scale-up methods (e.g., BIRCH).

An Example

Let's now see a simple example: a hierarchical clustering of distances in kilometers between some Italian cities. The method used is single-linkage.

Input distance matrix (L = 0 for all the clusters):

	BA	FI	MI	NA	RM	TO
BA	0	662	877	255	412	996
FI	662	0	295	468	268	400
MI	877	295	0	754	564	138
NA	255	468	754	0	219	869
RM	412	268	564	219	0	669
TO	996	400	138	869	669	0



The nearest pair of cities is MI and TO, at distance 138. These are merged into a single cluster called "MI/TO". The level of the new cluster is $L(MI/TO) = 138$ and the new sequence number is $m = 1$.

Then we compute the distance from this new compound object to all other objects. In single link clustering the rule is that the distance from the compound object to another object is equal to the shortest distance from any member of the cluster to the outside object. So the distance from "MI/TO" to RM is chosen to be 564, which is the distance from MI to RM, and so on.

After merging MI with TO we obtain the following matrix:

	BA	FI	MI/TO	NA	RM
--	----	----	-------	----	----

BA	0	662	877	255	412
FI	662	0	295	468	268
MI/TO	877	295	0	754	564
NA	255	468	754	0	219
RM	412	268	564	219	0



$\min d(i,j) = d(NA, RM) = 219 \Rightarrow$ merge NA and RM into a new cluster called NA/RM

$L(NA/RM) = 219$

$m = 2$

	BA	FI	MI/TO	NA/RM
BA	0	662	877	255
FI	662	0	295	268
MI/TO	877	295	0	564
NA/RM	255	268	564	0



$\min d(i,j) = d(BA, NA/RM) = 255 \Rightarrow$ merge BA and NA/RM into a new cluster called BA/NA/RM
 $L(BA/NA/RM) = 255$

$m = 3$

	BA/NA/RM	FI	MI/TO
BA/NA/RM	0	268	564
FI	268	0	295
MI/TO	564	295	0



$\min d(i,j) = d(BA/NA/RM, FI) = 268 \Rightarrow$ merge BA/NA/RM and FI into a new cluster called BA/FI/NA/RM
 $L(BA/FI/NA/RM) = 268$

$m = 4$

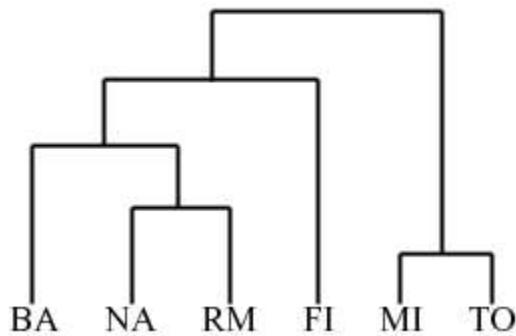
	BA/FI/NA/RM	MI/TO
--	--------------------	--------------

BA/FI/NA/RM	0	295
MI/TO	295	0



Finally, we merge the last two clusters at level 295.

The process is summarized by the following hierarchical tree

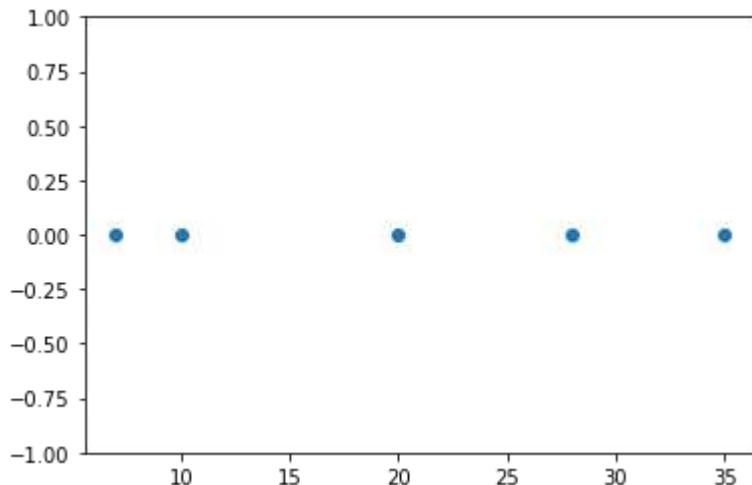


ANOTHER HIERARCHICAL EXAMPLE ?

What is hierarchical clustering (agglomerative) ?

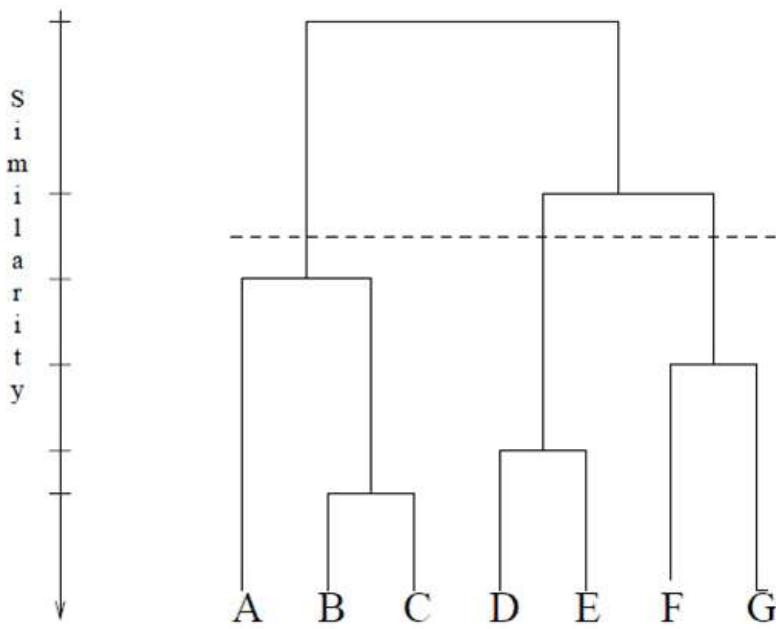
Clustering is a data mining technique to group a set of objects in a way such that objects in the same cluster are more similar to each other than to those in other clusters.

In hierarchical clustering, we assign each object (data point) to a separate cluster. Then compute the distance (similarity) between each of the clusters and join the two most similar clusters. Let's understand further by solving an example.



Objective : For the one dimensional data set {7,10,20,28,35}, perform hierarchical clustering and plot the dendrogram to visualize it.

Solution : First, let's visualize the data.



Observing the plot above, we can intuitively conclude that:

The first two points (7 and 10) are close to each other

and should be in the same cluster

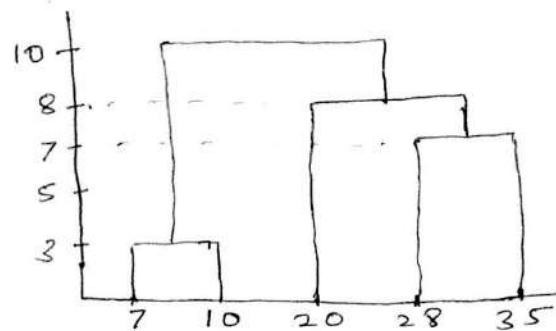
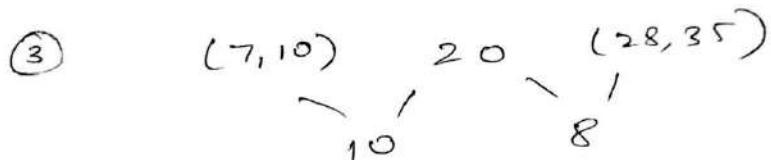
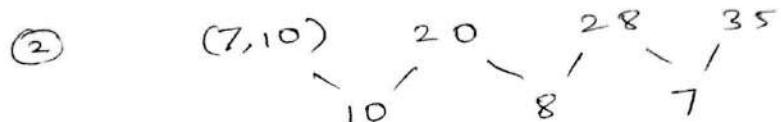
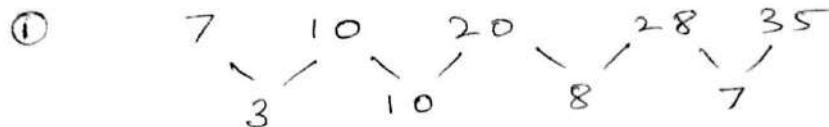
Also, the last two points (28 and 35) are close to each other and should be in the same cluster

Cluster of the center point (20) is not easy to conclude

Let's solve the problem by hand using both the types of agglomerative hierarchical clustering:

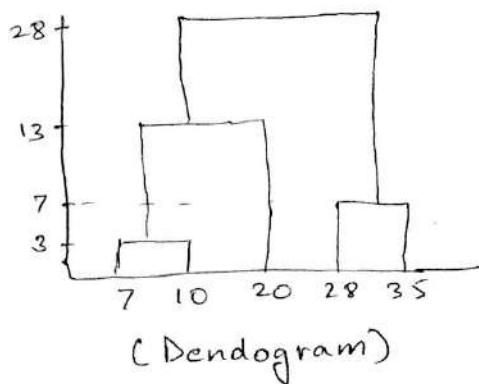
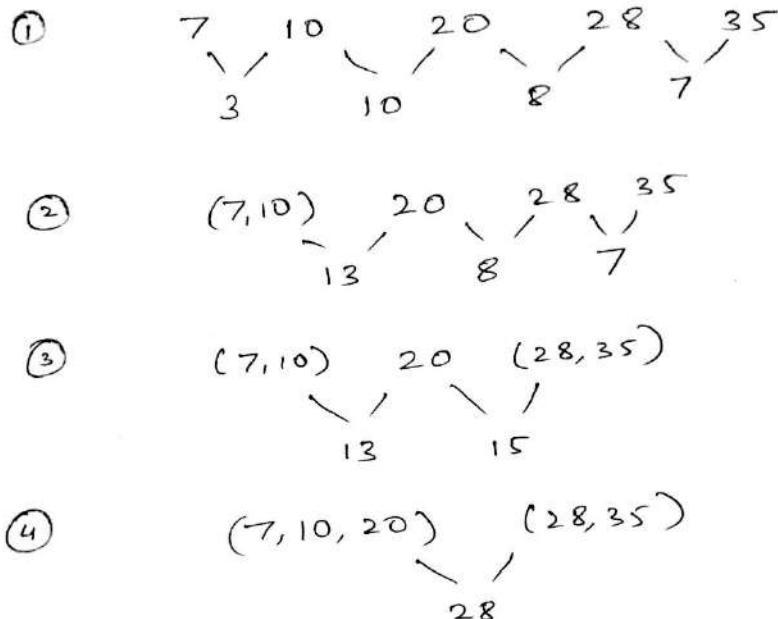
Single Linkage : In single link hierarchical clustering, we merge in each step the two clusters, whose two closest members have the smallest distance.

Single Linkage



(Dendogram)

Complete Linkage



Using single linkage two clusters are formed :

Cluster 1 : (7, 10)
 Cluster 2 : (20, 28, 35)

2. Complete Linkage : In complete link hierarchical clustering, we merge in the members of the clusters in each step, which

provide the smallest maximum pairwise distance

Using complete linkage two clusters are formed :

Cluster 1 : (7,10,20)

Cluster 2 : (28,35)

Conclusion : Hierarchical clustering is mostly used when the application requires a hierarchy, e.g creation of a taxonomy. However, they are expensive in terms of their computational and storage requirements.

Video:

1. Hierarchical Clustering: URL:
https://www.youtube.com/watch?v=tIIV3IT_hHk&feature=emb_logo
2. Hierarchical Clustering :URL: <https://www.youtube.com/watch?v=9U4h6pZw6f8>

PRINCIPAL COMPONENT ANALYSIS

Q.) Let our data matrix be score of 5 students: ① - Aditya

<u>Student</u>	<u>Math</u>	<u>English</u>	<u>Art</u>
1	90	60	90
2	90	90	30
3	60	60	60
4	60	60	90
5	30	30	30

Step 1: Convert the data into Matrix form:

$$A = \begin{bmatrix} 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix}$$

Note: While converting remove the ordinal or class labels:

Here we have 3 dimension and 5 samples
so matrix A (5,3)

Step 2: find the mean value for each dimension:
 In above matrix:

$$A = \begin{bmatrix} x & y & z \\ 90 & 60 & 90 \\ 90 & 90 & 30 \\ 60 & 60 & 60 \\ 60 & 60 & 90 \\ 30 & 30 & 30 \end{bmatrix} \Rightarrow \bar{A} = \begin{bmatrix} \bar{x} & \bar{y} & \bar{z} \\ 66 & 60 & 60 \end{bmatrix}$$

Step 3: Subtract the mean from each dimension.

$$X = (A - \bar{A}) = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} & z_1 - \bar{z} \\ x_2 - \bar{x} & y_2 - \bar{y} & z_2 - \bar{z} \\ x_3 - \bar{x} & y_3 - \bar{y} & z_3 - \bar{z} \end{bmatrix} = \begin{bmatrix} 24 & 0 & 30 \\ 24 & 30 & -30 \\ -6 & 0 & 0 \\ -6 & 0 & 30 \\ -36 & -30 & -30 \end{bmatrix}$$

Step:4 Calculate covariance matrix:

(3)

$$\text{Cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) \Rightarrow \text{for given sample}$$

"we have 3 dimensions so covariance matrix will be (3×3)

$$\begin{bmatrix} \text{Cov}(x, x) & \text{Cov}(x, y) & \text{Cov}(x, z) \\ \text{Cov}(y, x) & \text{Cov}(y, y) & \text{Cov}(y, z) \\ \text{Cov}(z, x) & \text{Cov}(z, y) & \text{Cov}(z, z) \end{bmatrix}$$

↓

$$\begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix}$$

Note: 1. Covariance $(x, y) = \text{covariance}(y, x)$

2. Covariance (x, x) = Variance of (x)

3. Calculation will be reduced.

Step:5 Calculate Eigen values and its corresponding eigen vectors: ④ ARS

Note: If A is our square matrix, v a vector and λ a scalar that satisfies $Av = \lambda v$, then λ is called eigen value associated with eigenvector v of A .

In our case A will always be a covariance matrix.

$$\begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \quad (q)$$

↓ ↓ → Eigen value:
 covariance eigen vector
 matrix to be calculate

Remember: ∵ we have 3-dimensions, we should get 3-eigen values associated with 3 eigen vectors:

Now we can write:

(5)
Ans

$$Ar - \lambda r = 0$$

$$\Rightarrow A - \lambda I \quad [A - \lambda I]r = 0$$

$\hookrightarrow I$ is identity matrix

Now Eigen values of A can be found by finding determinant of matrix $[A - \lambda I]$ which is written as:

$$|A - \lambda I| = 0$$

$$\Rightarrow \left| \begin{pmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right| = 0$$

$$\Rightarrow \left| \begin{matrix} 504 - \lambda & 360 & 180 \\ 360 & 360 - \lambda & 0 \\ 180 & 0 & 720 - \lambda \end{matrix} \right| = 0$$

\Rightarrow which gives equation:

⑥ Ans

$$-\lambda^3 + 158\lambda^2 - 641520\lambda + 25660800 = 0$$

After solving above equation we get 3 λ values
or 3 eigen values:

$$\lambda_1 \approx 44.819 \quad \lambda_2 \approx 629.110 \quad \lambda_3 \approx 910.069$$

Step 6: Now for each λ value we need to find
Eigen vectors by putting ~~eq~~ each λ
values in equation (a) in page 4.

for $\lambda_1 \approx 44.819$

$$\begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = 44.819 \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

$\underbrace{\hspace{1cm}}$ \downarrow
 Eigen vector
for λ_1

(7) Ans

for $\lambda_2 \approx 629.110$

$$\begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{\text{Eigenvector}} = 629.110 \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Eigenvector

for λ_2 for $\lambda_3 \approx 910.069$

$$\begin{bmatrix} 504 & 360 & 180 \\ 360 & 360 & 0 \\ 180 & 0 & 720 \end{bmatrix} \underbrace{\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}}_{\text{Eigen vector}} = 910.069 \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Eigen vector

 910.069

After calculation we get 3 eigen values and corresponding eigen vectors:

$$\lambda_1 = 44.819 \quad \lambda_2 = 629.110 \quad \lambda_3 = 910.069 \quad (8) \text{ AKS}$$

$$EV_1 = \begin{bmatrix} -3.75 \\ 4.28 \\ 1 \end{bmatrix} \quad EV_2 = \begin{bmatrix} -0.504 \\ -0.675 \\ 1 \end{bmatrix} \quad EV_3 = \begin{bmatrix} 1.055 \\ 0.691 \\ 1 \end{bmatrix}$$

Step:7: Sort Eigen Vectors in decreasing Eigen values and chose K eigen vector starting from largest eigen value to form $(d \times k)$ dimensional matrix W.

So in our case we have 3 Eigen values so we choose 2 eigen vectors ($\because K=2$) corresponding to top 2 eigen values. Thus our matrix W will be

$$W = \begin{bmatrix} 1.055 & -0.504 \\ 0.691 & -0.675 \\ 1 & 1 \end{bmatrix}_{3 \times 2}$$

$(d \times k)$
 $d \rightarrow$ no of dimension
 $K \Rightarrow$ no of eigen vector chosen

Note: If we take all the eigen vectors then there will be no dimension reduction.

(8) Step 8: Transform the samples into new subspace.

(9) ABS

final data = Row projection vector \times Row zero mean data

$$Y = W^T \cdot X^T$$

$$Y = \begin{bmatrix} 1.055 & 0.691 & 1 \\ -0.504 & 0.675 & 1 \end{bmatrix}_{2 \times 3} \quad \begin{bmatrix} 24 & 24 & -6 & -6 & -36 \\ 0 & 30 & 0 & 0 & -30 \\ 30 & -30 & 0 & 30 & -30 \end{bmatrix}_{3 \times 5}$$

$\underbrace{\hspace{10em}}$
 $W\text{-transpse}$

\times Transpose

(calculated in page no
2 Step 3)

$$Y = \begin{bmatrix} 55.32 & 16.05 & -6.33 & 23.67 & -88.71 \\ 17.904 & -21.846 & 3.024 & 33.024 & -32.106 \end{bmatrix}_{2 \times 5}$$

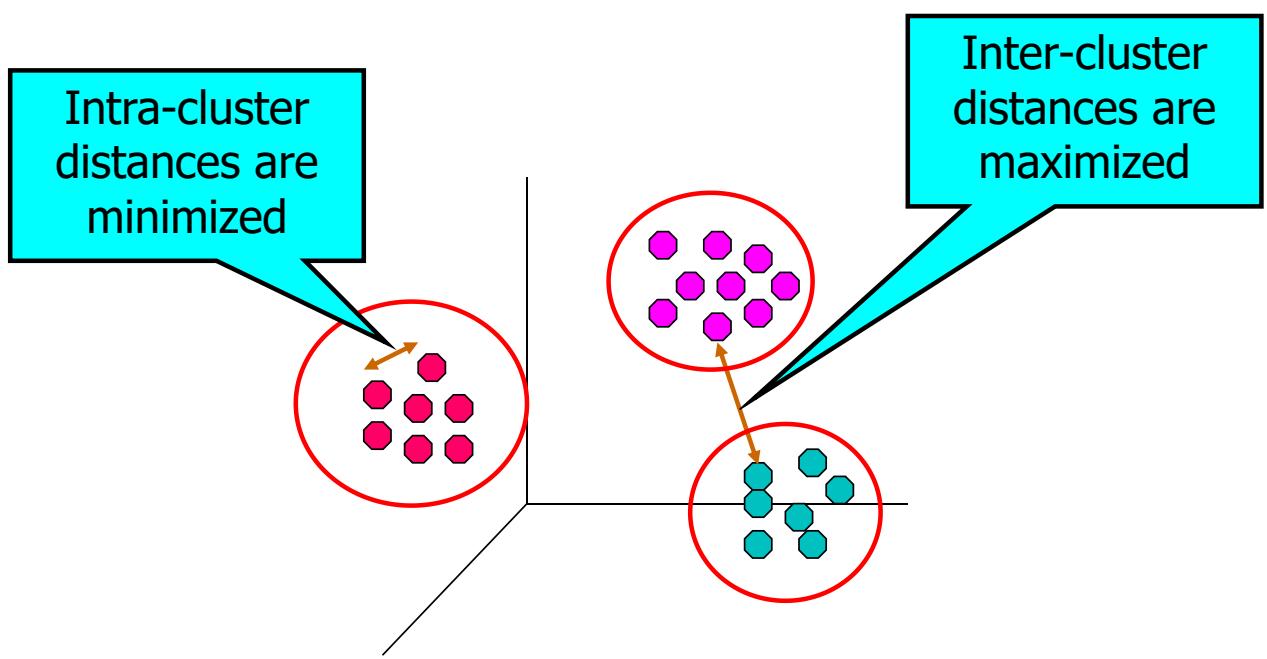
We see our data set is reduced from 3×5 matrix to 2×5 matrix.
Done:

EXTERNAL RESOURCE:

1. <https://youtu.be/FgakZw6K1QQ>
2. <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>
3. <https://towardsdatascience.com/the-mathematics-behind-principal-component-analysis-fff2d7f4b643>

Cluster Analysis:

What is Cluster Analysis?



Clustering

- Clustering of data is a method by which large sets of data is grouped into clusters of smaller sets of similar data
- Objects in one cluster have high similarity to each other and are dissimilar to objects in other clusters
- An example of unsupervised learning
- Group objects that share common characteristics

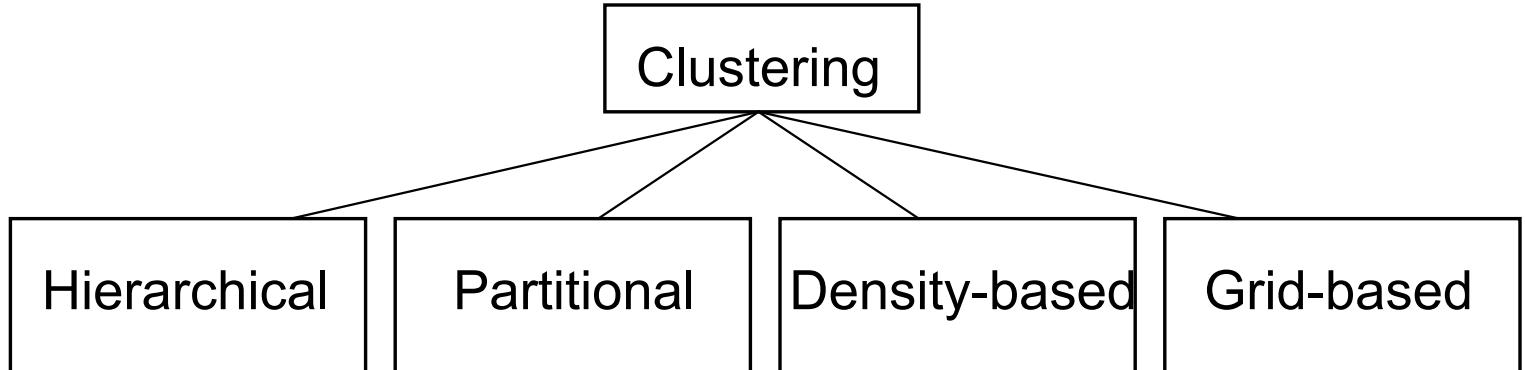
Examples of Clustering Applications

- Marketing:
- Land use:
- Insurance:
- City-planning:
- Earth-quake studies:

What Is Good Clustering?

- A good clustering method will produce high quality clusters with
 - high intra-class similarity
 - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.

Clustering Approaches

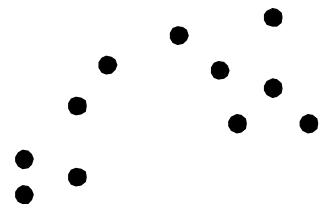


Partitioning Methods

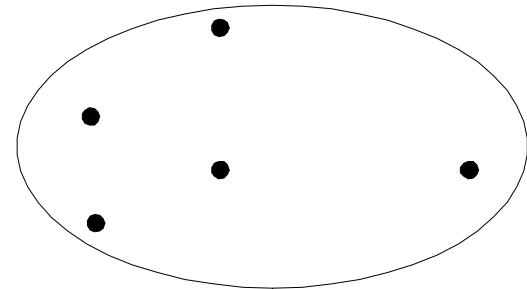
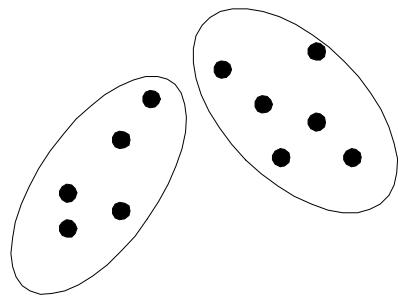
Given a DB of n objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$ such that

1. Each group must contain atleast one object, and
2. Each object must belong to exactly one group

Partitional Clustering



Original Points



A Partitional Clustering

Density-based Methods

- Most partitioning-based methods cluster objects based on distances between them
- Can find only spherical-shaped clusters
- Density-based clustering
- Continue growing a given cluster as long as the density in the ‘neighborhood’ exceeds some threshold.

Hierarchical Clustering

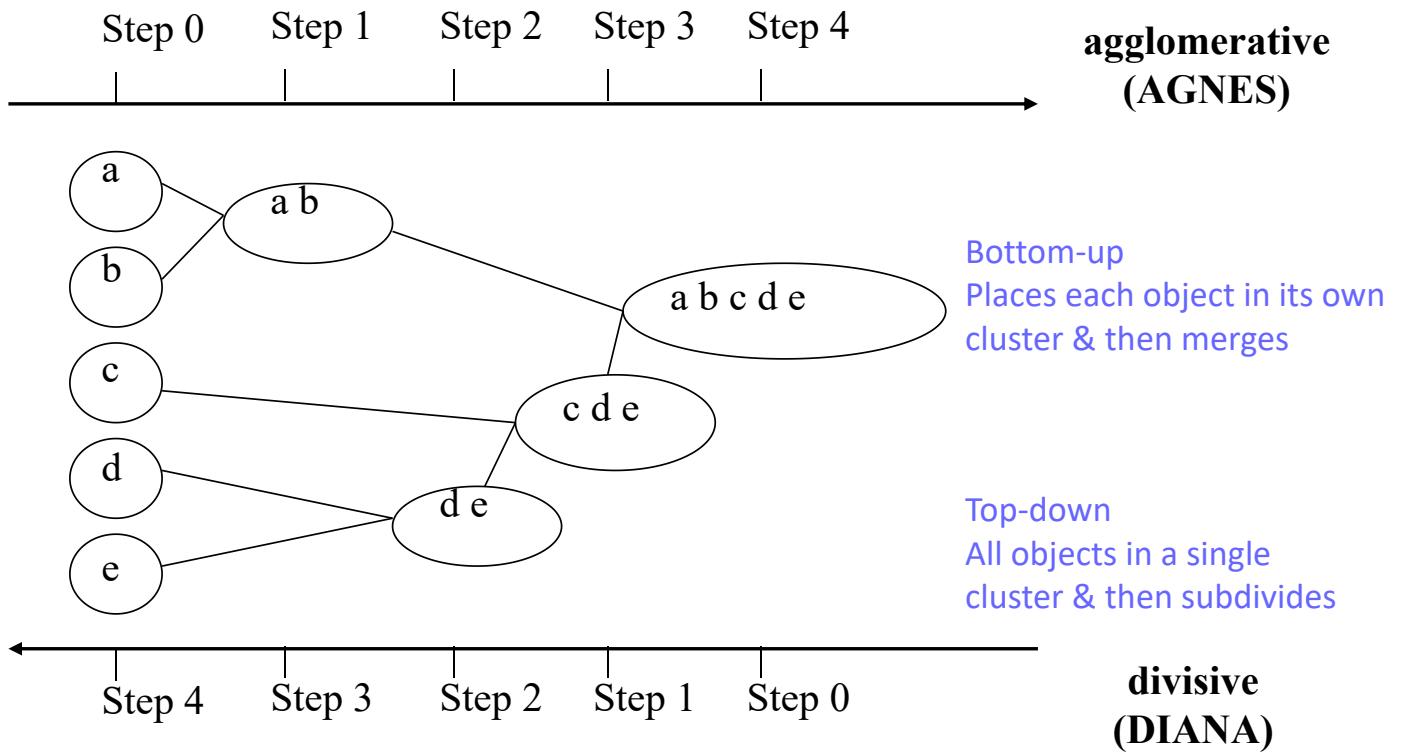
a) Agglomerative : (Bottom Up)

- Start with each object in a cluster.
- Combine similar clusters into larger and larger clusters.

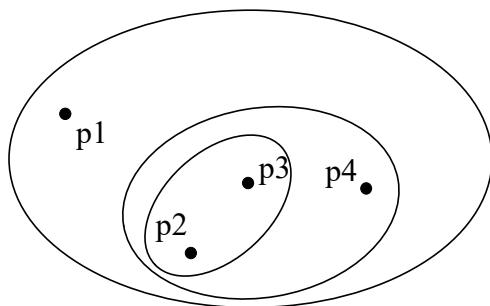
b) Divisive: (Top Down)

- Start with one large cluster and split into smaller and smaller clusters

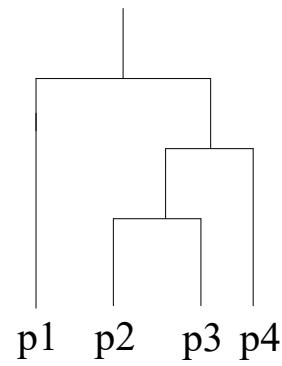
Hierarchical Clustering



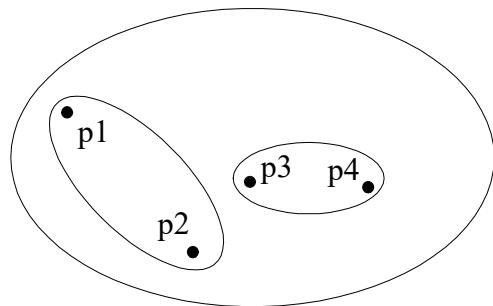
Hierarchical Clustering



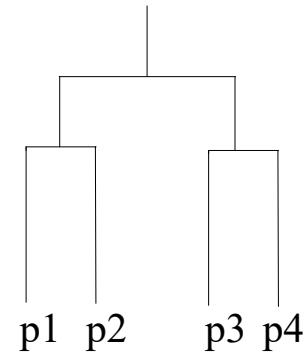
Traditional Hierarchical Clustering



Traditional Dendrogram

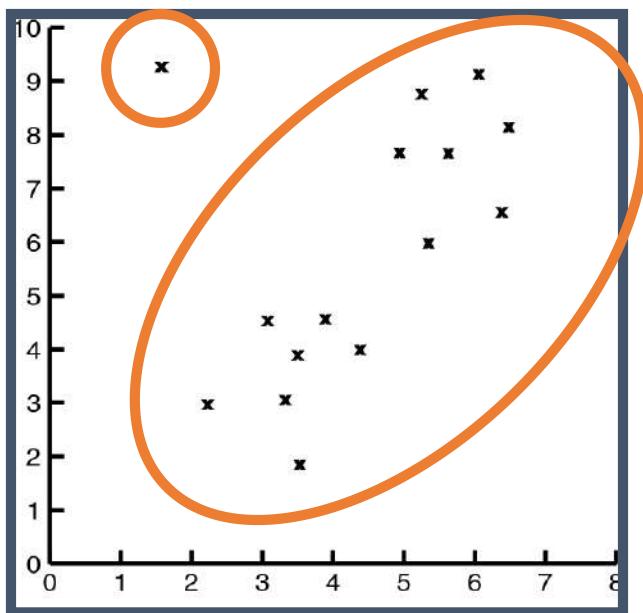


Non-traditional Hierarchical Clustering



Non-traditional Dendrogram

Impact of Outliers on Clustering



Partitioning Algorithms: Basic Concept

- Partitioning method: Construct a partition of a database D of n objects into a set of k clusters
- Given a k , find a partition of k *clusters* that optimizes the chosen partitioning criterion
 - Global optimal: exhaustively enumerate all partitions
 - Heuristic methods: *k-means* and *k-medoids* algorithms
 - *k-means* (MacQueen'67): Each cluster is represented by the center of the cluster
 - *k-medoids* or PAM (Partition around medoids) (Kaufman & Rousseeuw'87): Each cluster is represented by one of the objects in the cluster

K-means Clustering

1. Select the number of clusters, **k**.
2. Pick **k** seeds randomly as centroids of the **k** clusters.
3. Compute the Euclidean distance of each object from each of the centroids.
4. Allocate each object to its nearest cluster.
5. Compute the new centroids of the clusters
 - Mean of the attribute values of the objects in each cluster is the cluster centroid.
6. Check if the stopping condition has been met. If yes, stop else go to step 3.
 - No change in the cluster membership.

K-means Example 1

- For simplicity, 1-dimension objects and k=2.
 - Numerical difference is used as the distance
- Objects: 1, 2, 5, 6, 7
- K-means:
 - Randomly select 5 and 6 as centroids;
 - End of Iteration 1 : Two clusters {1,2,5} and {6,7}; meanC1=8/3, meanC2=6.5
 - End of Iteration 2 : {1,2}, {5,6,7}; meanC1=1.5, meanC2=6
 - End of Iteration 3 : no change.

K-means Clustering – Details

- Initial centroids are often chosen randomly.
 - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- ‘Closeness’ is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
 - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- Complexity is $O(n * K * I * d)$
 - $n = \text{number of points}$, $K = \text{number of clusters}$,
 $I = \text{number of iterations}$, $d = \text{number of attributes}$

Evaluating K-means Clusters

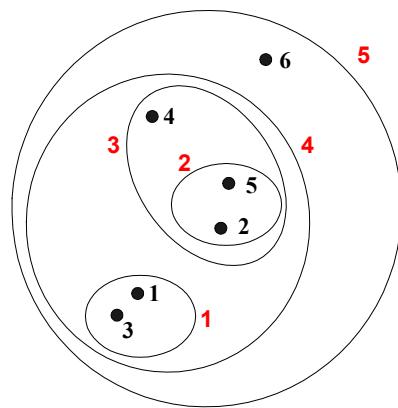
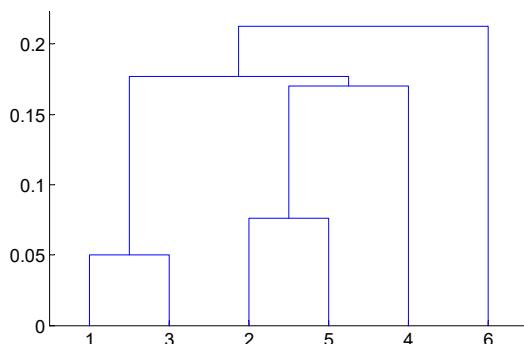
- Most common measure is Sum of Squared Error (SSE)
 - For each point, the error is the distance to the nearest cluster
 - To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} dist^2(m_i, x)$$

- x is a data point in cluster C_i and m_i is the representative point for cluster C_i
 - can show that m_i corresponds to the center (mean) of the cluster
- Given two clustering results, we can choose the one with the smallest error
- One easy way to reduce SSE is to increase K, the number of clusters
 - A good clustering with smaller K can have a lower SSE than a poor clustering with higher K

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendograms
 - A tree like diagram that records the sequences of merges or splits



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
 - Any desired number of clusters can be obtained by ‘cutting’ the dendrogram at the proper level

Hierarchical Clustering

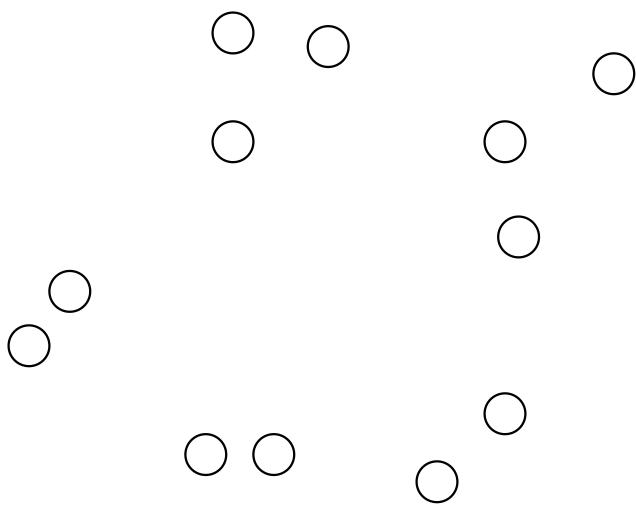
- Two main types of hierarchical clustering
 - Agglomerative:
 - Start with the points as individual clusters
 - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
 - Divisive:
 - Start with one, all-inclusive cluster
 - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
 - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
 1. Compute the proximity matrix
 2. Let each data point be a cluster
 3. **Repeat**
 4. Merge the two closest clusters
 5. Update the proximity matrix
 6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
 - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

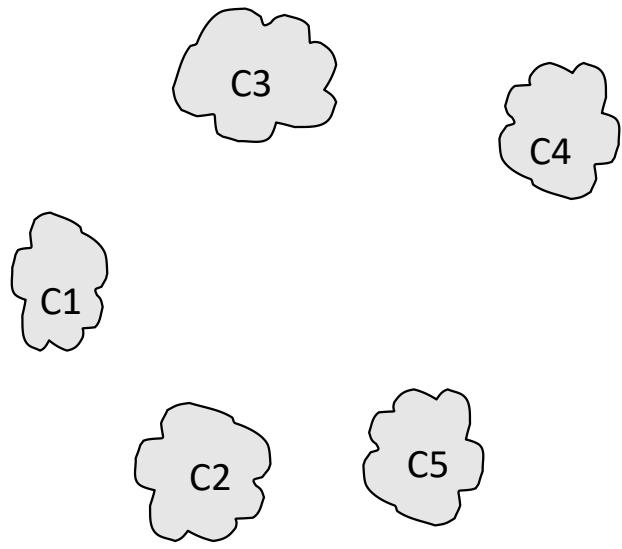


	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
Proximity Matrix						

p1 p2 p3 p4 ... p9 p10 p11 p12

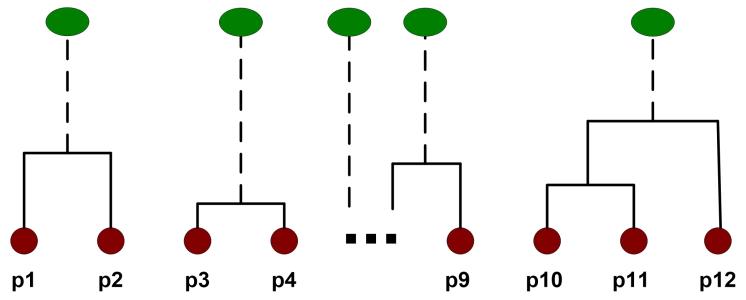
Intermediate Situation

- After some merging steps, we have some clusters



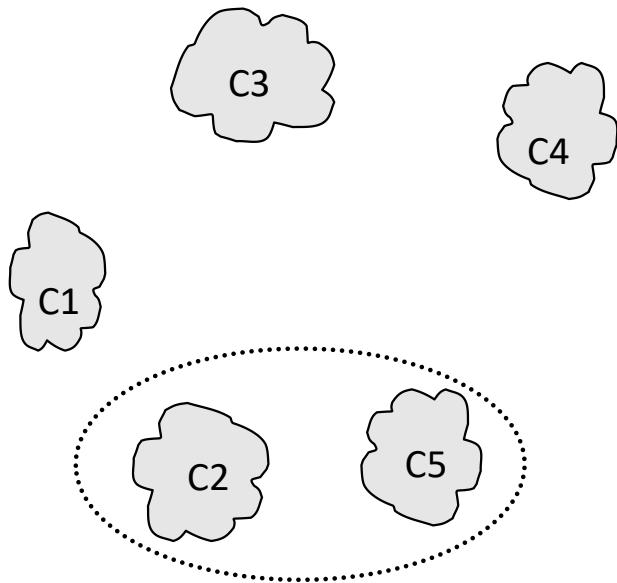
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



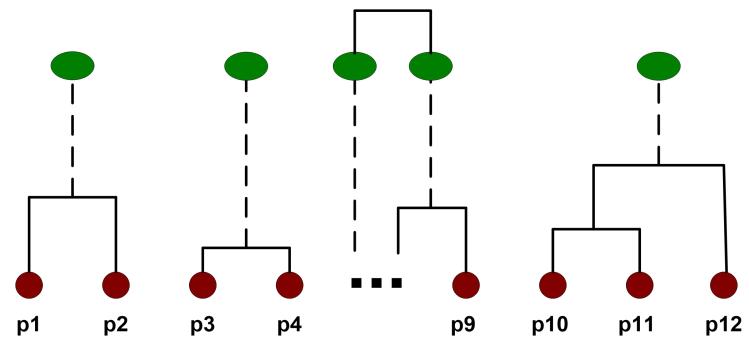
Intermediate Situation

- We want to merge the two closest clusters (C_2 and C_5) and update the proximity matrix.



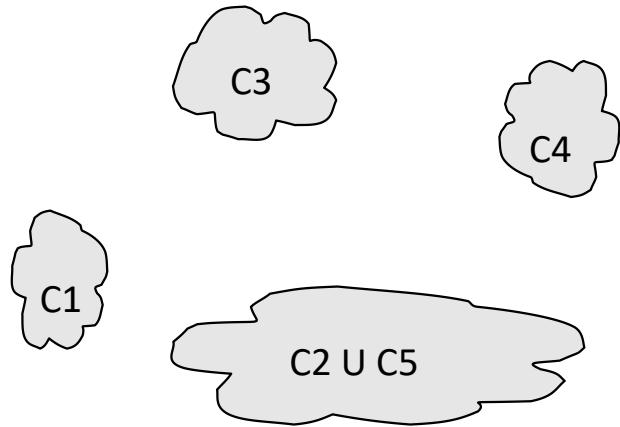
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix

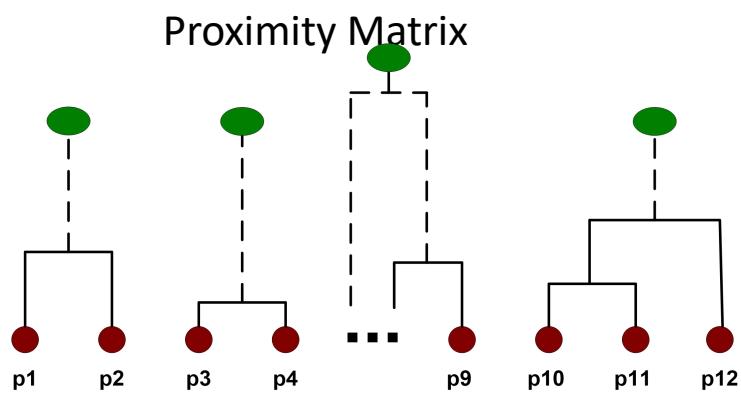


After Merging

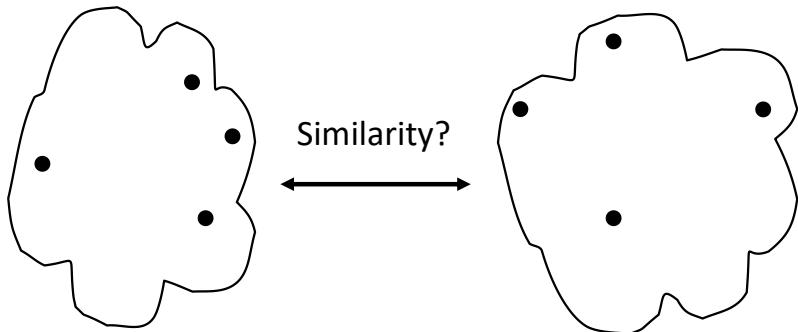
- The question is “How do we update the proximity matrix?”



		C2		
		U		
	C1	C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		



How to Define Inter-Cluster Similarity

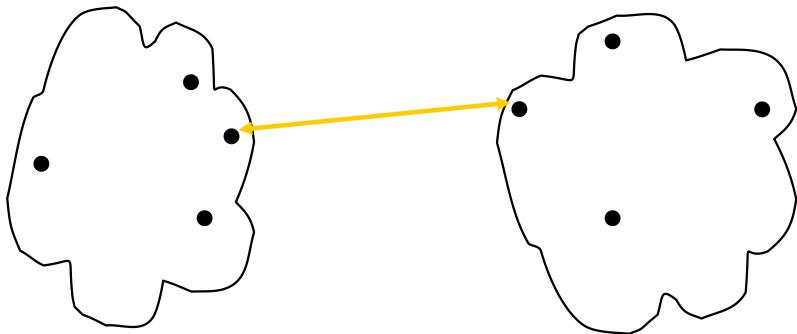


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

. Proximity Matrix

How to Define Inter-Cluster Similarity

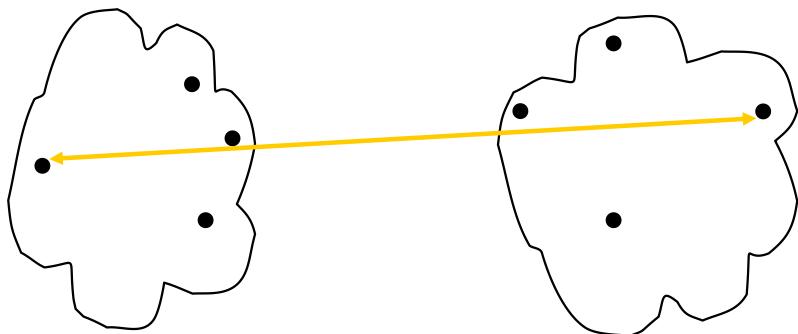


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

Proximity Matrix

How to Define Inter-Cluster Similarity

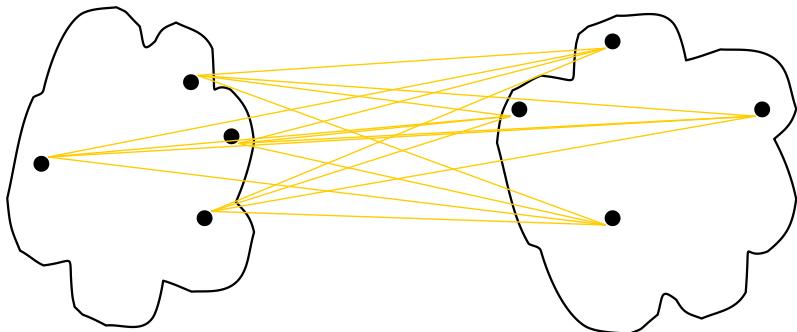


- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

Proximity Matrix

How to Define Inter-Cluster Similarity

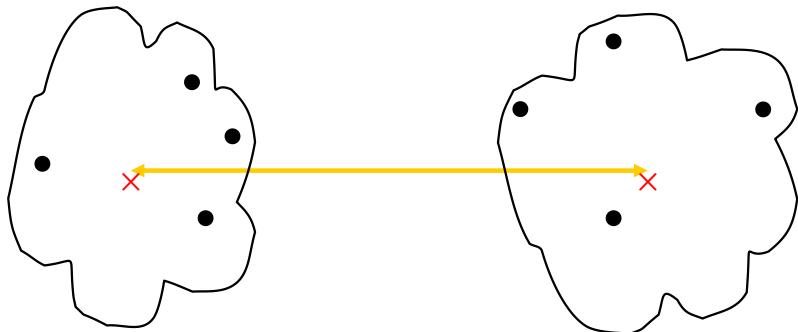


- MIN
- MAX
- **Group Average**
- Distance Between Centroids
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

Proximity Matrix

How to Define Inter-Cluster Similarity



- MIN
- MAX
- Group Average
- **Distance Between Centroids**
- Other methods driven by an objective function
 - Ward's Method uses squared error

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.

Proximity Matrix

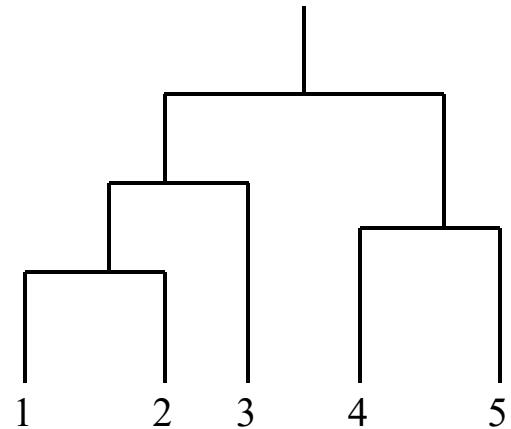
Cluster Similarity: Group Average

- Proximity of two clusters is the average of pairwise proximity between points in the two clusters.

$$\text{proximity}(\text{Cluster}_i, \text{Cluster}_j) = \frac{\sum_{\substack{p_i \in \text{Cluster}_i \\ p_j \in \text{Cluster}_j}} \text{proximity}(p_i, p_j)}{|\text{Cluster}_i| * |\text{Cluster}_j|}$$

- Need to use average connectivity for scalability since total proximity favors large clusters

I1	I2	I3	I4	I5	
I1	1.00	0.90	0.10	0.65	0.20
I2	0.90	1.00	0.70	0.60	0.50
I3	0.10	0.70	1.00	0.40	0.30
I4	0.65	0.60	0.40	1.00	0.80
I5	0.20	0.50	0.30	0.80	1.00





Unsupervised Learning and Curse of Dimensionality

CONTENT

1 INTRODUCTION

2 CLUSTERING

3 BLIND SIGNAL SEPARATION SELF-ORGANIZING
MAPS(SOM)

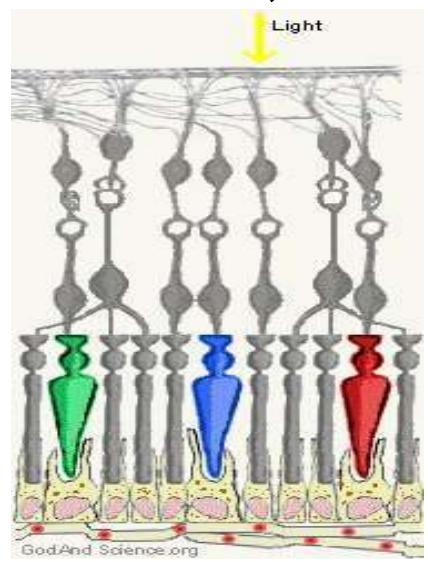
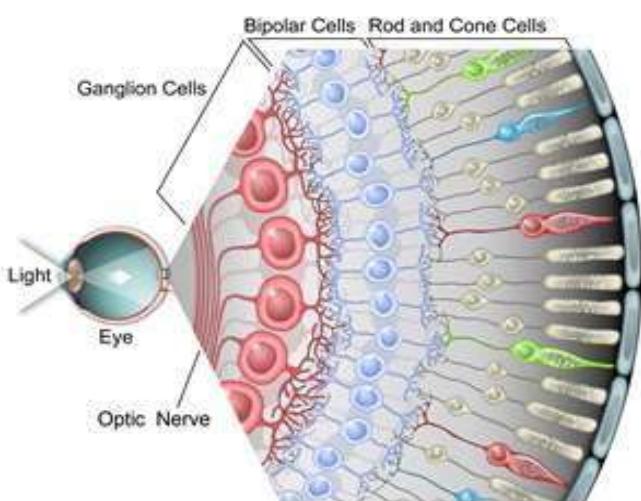
4 Curse of Dimensionality



Introduction

- Unsupervised learning studies how systems can learn to represent particular input patterns in a way that reflects the statistical structure of the overall collection of input patterns.
- There are no explicit target outputs rather the unsupervised learner brings to bear prior biases as to what aspects of the structure of the input should be captured in the output.
- indeed structural and physiological properties of synapses in the neocortex are known to be substantially influenced by the patterns of activity in sensory neurons that occur.
- essentially none of the information about the contents of scenes is available during learning.

- Unsupervised learning is important since it is likely to be much more common in the brain than supervised learning.
- For instance there are around 106 photoreceptors in each eye whose activities are constantly changing with the visual world and which provide all the information that is available to indicate what objects there are in the world, how they are presented, what the lighting conditions are, etc.



➤ History

- Horace Barlow (see Barlow, 1992), who sought ways of characterizing neural codes,
- David Marr (1970), who made an early unsupervised learning postulate about the goal of learning in his model of the neocortex.
- The Hebb rule (Hebb, 1949), which links statistical methods to neurophysiological experiments on plasticity
- Geoffrey Hinton and Terrence Sejnowski in inventing a model of learning called the Boltzmann machine (1986)

➤ Examples of unsupervised learning approaches

- ✓ Clustering.
- ✓ Blind signal separation.
- ✓ Self-organising maps

Clustering

is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups (clusters).

It is a main task of exploratory data mining , and a common technique for statistical data analysis .

used in many fields, including machine learning , pattern recognition image analysis.

➤ Clustering algorithms

✓ Hierarchical clustering

creating clusters that have a predetermined ordering from top to bottom. For example, all files and folders on the hard disk are organized in a hierarchy.

- Agglomerative algorithm.
- Divisive algorithm.

✓ Partitional clustering

Decompose the data set into a set of disjoint clusters

- K-mean algorithm.

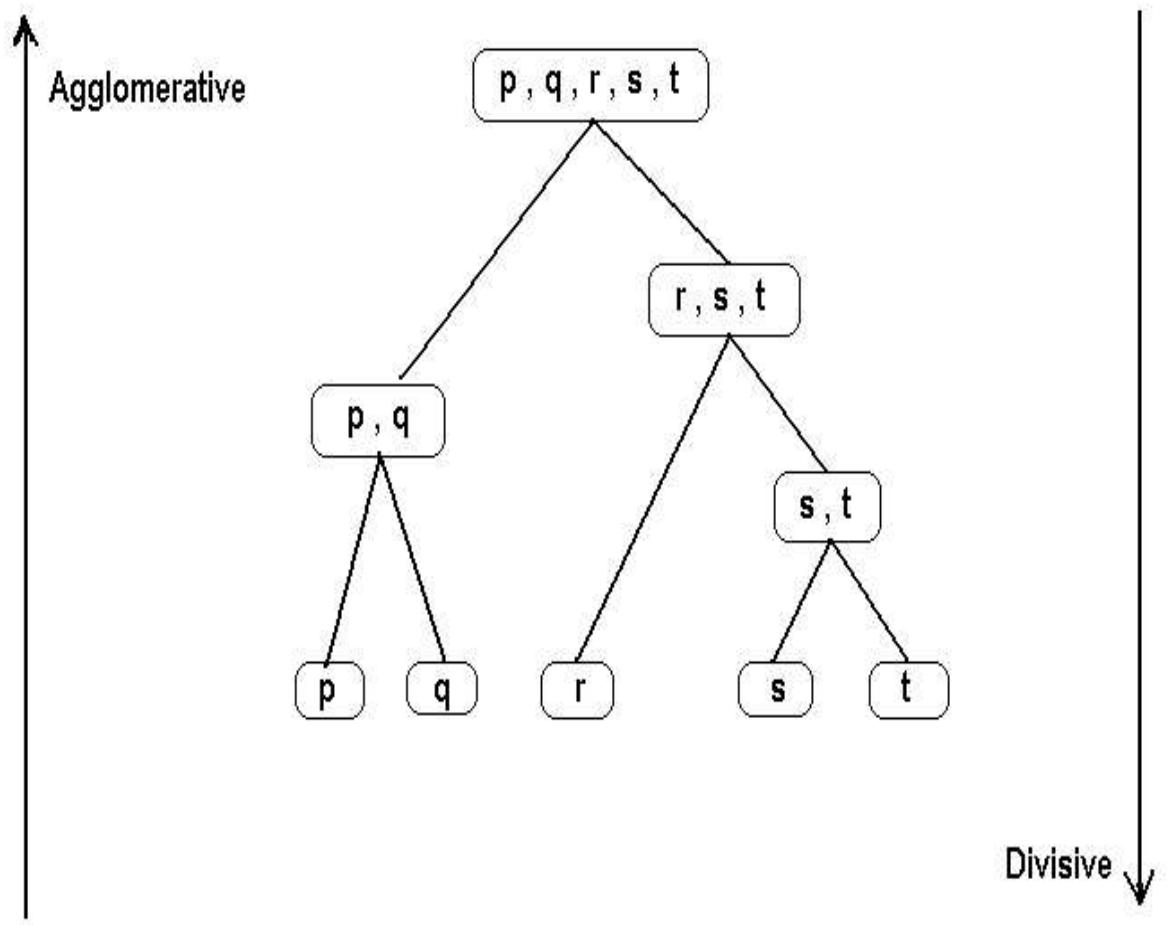
➤ Hierarchical clustering

✓ Agglomerative clustering

- In this method we assign each observation to its own cluster.
- Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters.
- Finally, repeat steps 2 and 3 until there is only a single cluster left. The related algorithm is shown below.

✓ Divisive clustering

- In this method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters. Finally, we proceed recursively on each cluster until there is one cluster for each observation.

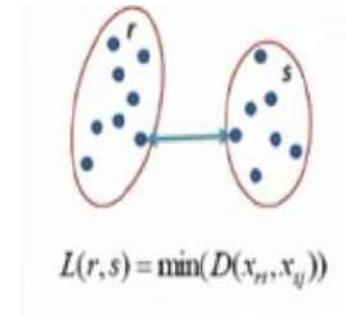


Agglomerative methods

➤ proximity matrix

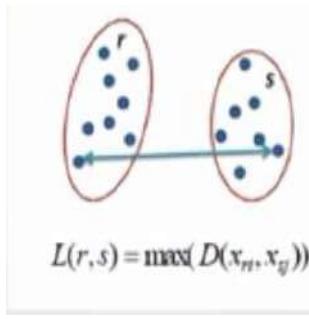
- Before any clustering is performed, it is required to determine the proximity matrix containing the distance between each point using a distance function. Then, the matrix is updated to display the distance between each cluster

Single Linkage



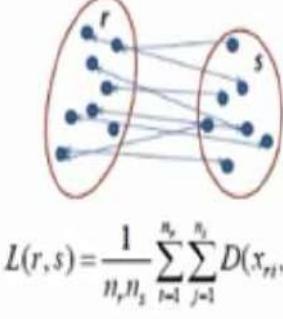
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete Linkage



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

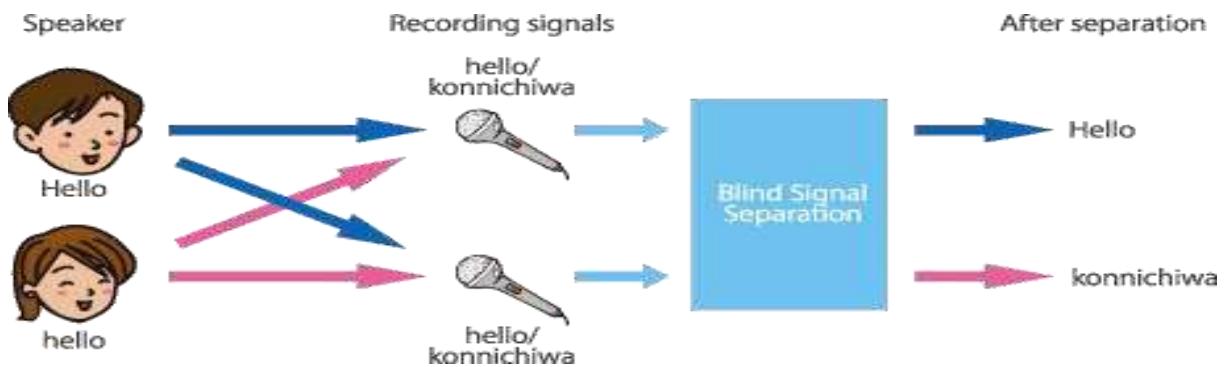
Average Linkage



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Blind signal separation

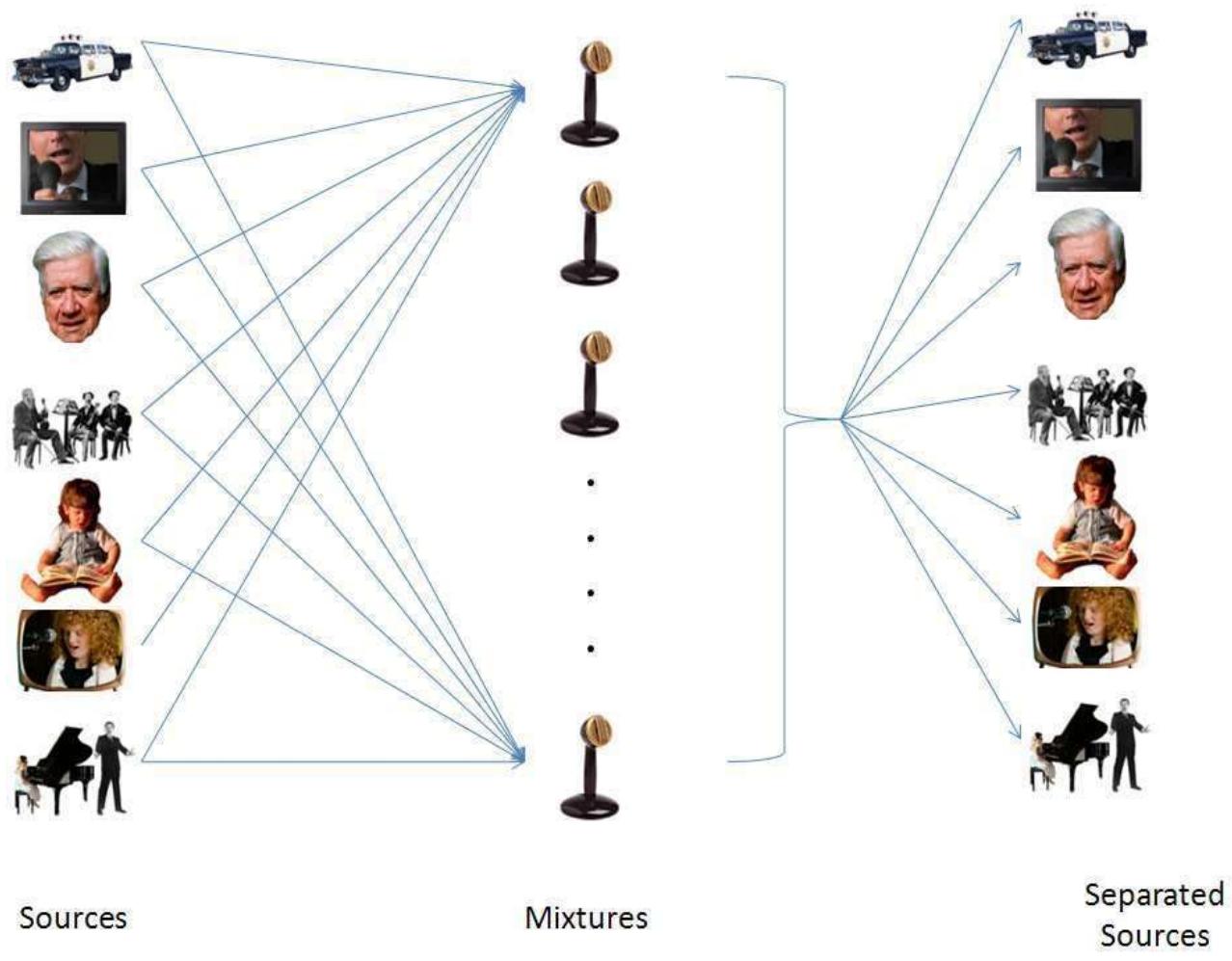
- Blind signal separation, also known as blind source separation.
- is the separation of a set of source signals from a set of mixed signals, without the aid of information (or with very little information) about the source signals or the mixing process.
- methods for blind source separation generally seek to narrow the set of possible solutions in a way that is unlikely to exclude the desired solution.



- Recently, blind source separation by Independent Component Analysis (ICA) has received attention because of its potential applications in signal processing such as in speech recognition systems, telecommunications and medical signal processing.
- The goal of ICA is to recover independent sources given only sensor observations that are unknown linear mixtures of the unobserved independent source signals.
- ICA not only de correlates the signals (2nd-order statistics) but also reduces higher-order statistical dependencies, attempting to make the signals as independent as possible.

➤ Partitionning techniques

16



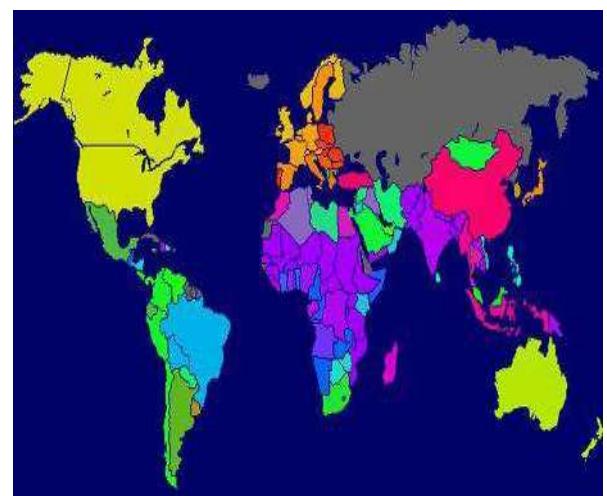
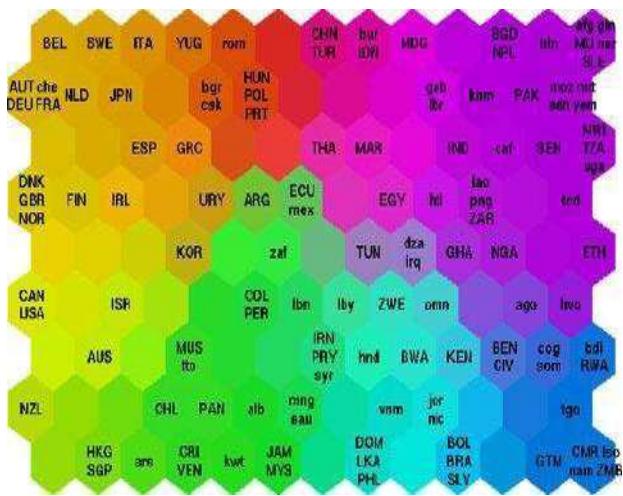
Self-organizing Maps(som)

- Self-Organizing Map (SOM) is an unsupervised learning algorithm.
- SOM is a visualization method to represent higher dimensional data in an usually 1-D, 2-D or 3-D manner.
- SOMs have two phases:
 - Learning phase: map is built, network organizes using a competitive process

➤ Applications of SOMs

- SOMs are commonly used as visualization aids. They can make it easy for us humans to see relationships between vast amounts of data

✓ World Poverty Map



- Example: Data sets for poverty levels in different countries.
 - Data sets have many different statistics for each country.
 - SOM does not show poverty levels, rather it shows how similar the poverty sets for different countries are to each other.



Curse of Dimensionality

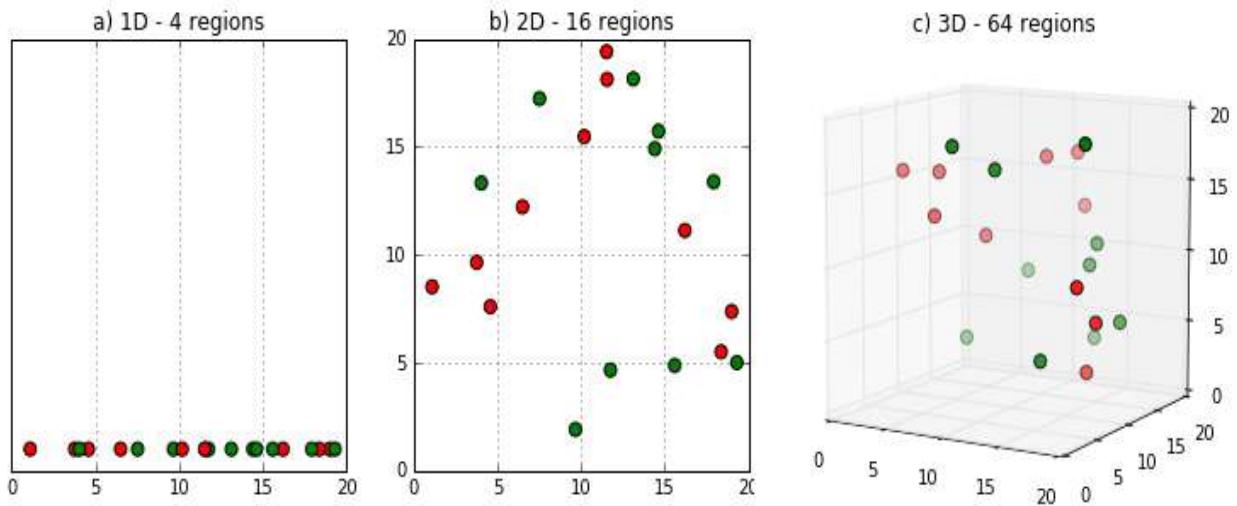
Introduction

- ❖ The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional (hundreds or thousands) spaces.
- ❖ The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data

- ❖ This sparsity is problematic for any method that requires statistical significance.
- ❖ “As the number of features or dimensions grows, the amount of data we need to generalize accurately grows exponentially.” – Charles Isbell

Example

Let's take an example. Fig. 1 (a) shows 10 data points in one dimension i.e. there is only one feature in the data set. It can be easily represented on a line with only 10 values, $x=1, 2,$



But if we add one more feature, same data will be represented in 2 dimensions (Fig. 1 (b)) causing increase in dimension space to $10 \times 10 = 100$. And again if we add 3rd feature, dimension space will increase to $10 \times 10 \times 10 = 1000$. As dimensions grows, dimensions space increases exponentially.

- ☞ This exponential growth in data causes high sparsity in the data set and unnecessarily increases storage space and processing time for the particular modeling algorithm and that's why it's called **Curse of Dimensionality**.
- ☞ Value added by additional dimension is much smaller compared to overhead it adds to the algorithm.

Techniques to Avoid Curse of Dimensionality

- ❖ Manual Features Selection
- ❖ Pearson Correlation Coefficient
- ❖ Principle Component Analysis (PCA)
- ❖ Singular Value Decomposition
- ❖ Multidimensional Scaling
- ❖ Locally Linear Embedding

Association analysis

- Association analysis is the discovery of association rules showing attribute-value conditions that occur frequently together in a given set of data.
- Produce dependency rules which will predict occurrence of an item based on occurrences of other items
- **Example-** frequent patterns refers to a set of items that often appear together in a transactional data set.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:

$\{Milk\} \rightarrow \{Coke\}$

$\{Diaper, Milk\} \rightarrow \{Beer\}$



Private University Estd. in Karnataka State by Act No. 41 of 2013

Association Analysis: Applications

- Market-basket analysis
 - Rules are used for sales promotion, shelf management, and inventory management
- Telecommunication alarm diagnosis
 - Rules are used to find combination of alarms that occur together frequently in the same time period
- Medical Informatics
 - Rules are used to find combination of patient symptoms and test results associated with certain diseases

Association Rule Discovery: Application 1

- Suppose a marketing manager at AllElectronics, wants to know which items are frequently purchased together (i.e., within the same transaction).
 - An example of such a rule, mined from the AllElectronics transactional database, is
- buys(X, "computer") → buys(X, "software") [support = 1%,confidence = 50%],**
- where X is a variable representing a customer.
 - A confidence, or certainty, of 50% means that if a customer buys a computer, there is a 50% chance that she will buy software as well.
 - A 1% support means that 1% of all the transactions under analysis show that computer and software are purchased together.



Private University Estd. in Karnataka State by Act No. 41 of 2013

Association Rule

- Association rules like

$\text{age}(X, "20..29") \text{ and } \text{income}(X, "40K..49K") \rightarrow \text{buys}(X, "laptop")$ [support = 2%, confidence = 60%].

Frequent subsequences (also known as sequential patterns)-

➤ A frequently occurring subsequence, such as the pattern that customers, tend to purchase first a laptop, followed by a digital camera, and then a memory card, is a (frequent) sequential pattern.



Association Analysis

- Basic Concepts 
- Frequent Itemset Mining Methods
- Which Patterns Are Interesting?—Pattern Evaluation Methods
- Summary

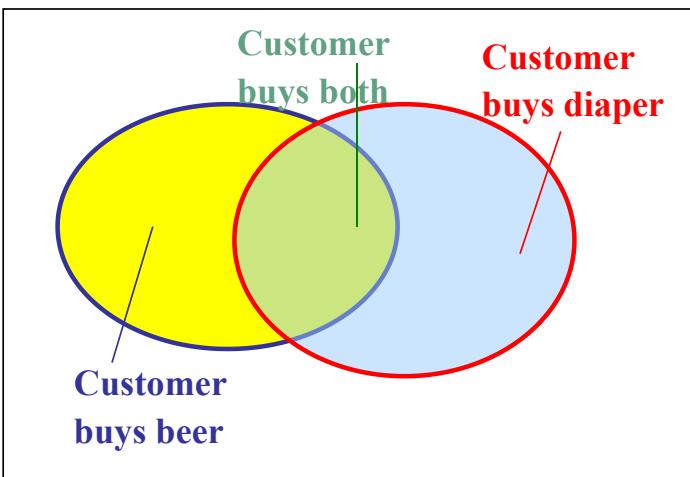
What is Frequent Pattern Analysis?

- **Frequent pattern:** a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
 - What products were often purchased together?— Beer and diapers?!
 - What are the subsequent purchases after buying a PC?
 - What kinds of DNA are sensitive to this new drug?
 - Can we automatically classify web documents?
- Applications
 - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Basic Concepts: Frequent Patterns

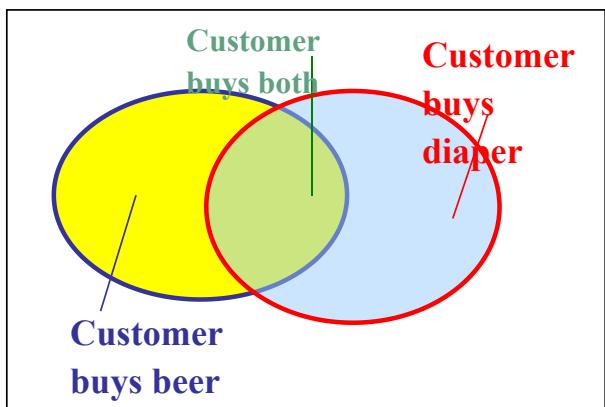
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

- **itemset**: A set of one or more items
- **k-itemset** $X = \{x_1, \dots, x_k\}$
- **(absolute) support**, or, **support count** of X : Frequency or occurrence of an itemset X
- **(relative) support**, s , is the fraction of transactions that contains X (i.e., the **probability** that a transaction contains X)
- An itemset X is **frequent** if X 's support is no less than a *minsup* threshold



Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
 - **support**, s , **probability** that a transaction contains $X \cup Y$
 - **confidence**, c , **conditional probability** that a transaction having X also contains Y

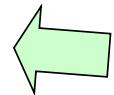
Let $\text{minsup} = 50\%$, $\text{minconf} = 50\%$

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
 - $\text{Beer} \rightarrow \text{Diaper}$ (60%, 100%)
 - $\text{Diaper} \rightarrow \text{Beer}$ (60%, 75%)

Frequent Itemset Mining Method

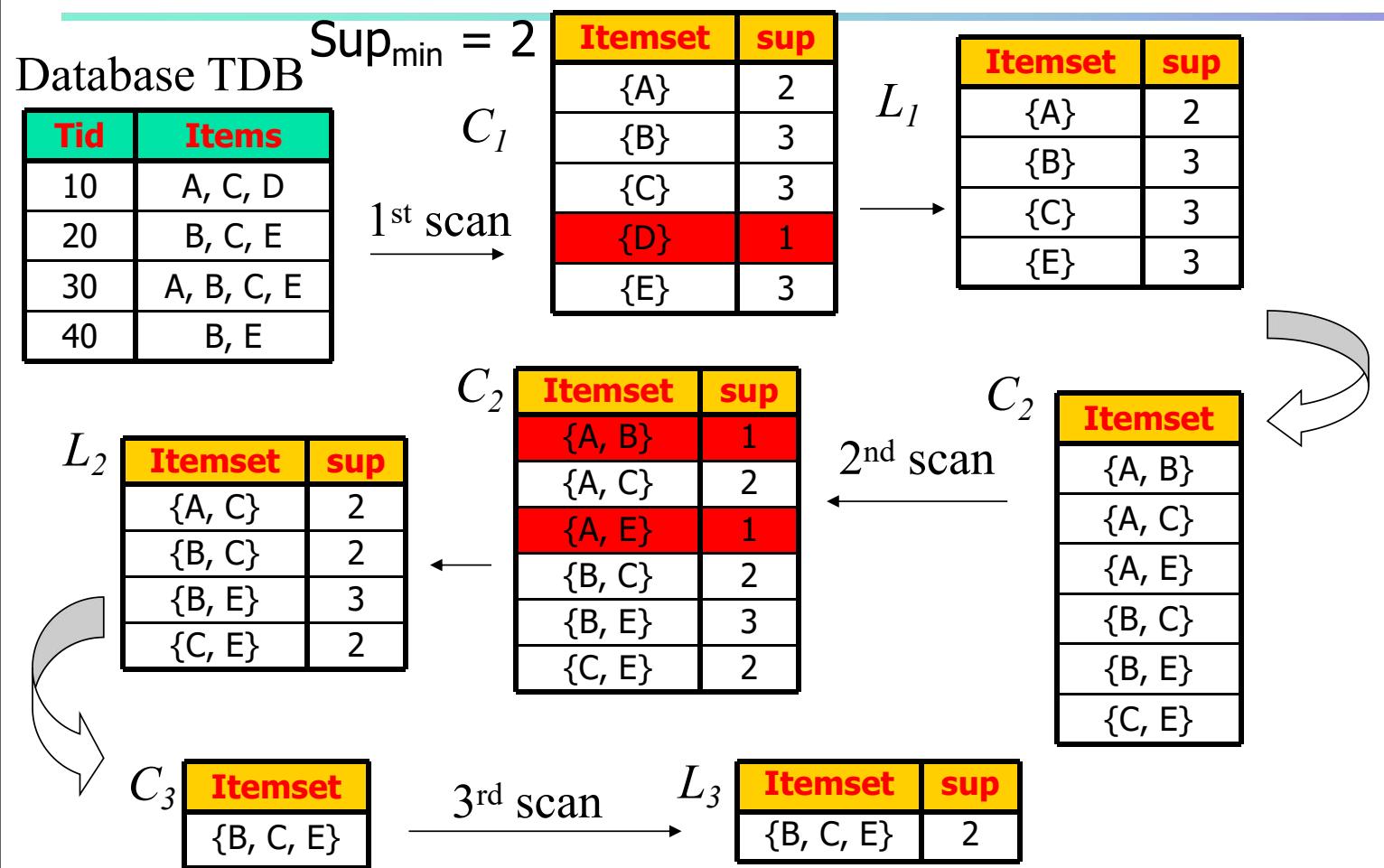
- Apriori: A Candidate Generation-and-Test Approach



Apriori: A Candidate Generation & Test Approach

- **Apriori pruning principle:** If there is **any itemset** which is infrequent, its superset should not be generated/tested! (Agrawal & Srikant @VLDB'94, Mannila, et al. @ KDD' 94)
- Method:
 - Initially, scan DB once to get frequent 1-itemset
 - Generate length $(k+1)$ **candidate itemsets** from length k **frequent itemsets**
 - **Test** the candidates against DB
 - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example



The Apriori Algorithm (Pseudo-Code)

C_k : Candidate itemset of size k

L_k : frequent itemset of size k

$L_1 = \{\text{frequent items}\};$

for ($k = 1; L_k \neq \emptyset; k++$) **do begin**

C_{k+1} = candidates generated from L_k ;

for each transaction t in database do

 increment the count of all candidates in C_{k+1} that
 are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

Implementation of Apriori

- How to generate candidates?
 - Step 1: self-joining L_k
 - Step 2: pruning
- Example of Candidate-generation
 - $L_3 = \{abc, abd, acd, ace, bcd\}$
 - Self-joining: $L_3 * L_3$
 - $abcd$ from abc and abd
 - $acde$ from acd and ace
 - Pruning:
 - $acde$ is removed because ade is not in L_3
 - $C_4 = \{abcd\}$

Why Mining Association Rules?

► Objective:

- Finding interesting co-occurring items (or objects, events) in a given data set.

► Examples:

- Given a database of transactions, each transaction is a list of items (purchased by a customer in a visit), you may find:

computer → financial_management_software
[support=2%, confidence=60%]

- From a student database, you may find

► $\text{major}(x, \text{"CS"}) \wedge \text{gpa}(x, \text{"A"}) \rightarrow \text{has_taken}(x, \text{"DB"})$ [1%, 75%]

What Kind of Databases?

Transaction database TDB

TID	Items
100	f, a, c, d, g, i, m, p
200	a, b, c, f, l, m, o
300	b, f, h, j, o
400	b, c, k, s, p
500	a, f, c, e, l, p, m, n

- ▶ Itemset: a set of items
- ▶ A *transaction* is a tuple (tid, X)
 - ▶ Transaction ID tid
 - ▶ Itemset X
- ▶ A *transaction database* is a set of transactions
 - ▶ In many cases, a transaction database can be treated as a set of itemsets (ignore TIDs)
- ▶ Association rule from TDB (relates two itemsets):
 - ▶ $\{a, c, m\} \rightarrow \{l\}$ [support=40%, confidence=66.7%]

(2)

Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\}$,
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\}$,
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\}$,

Implication means co-occurrence,
not causality!

Definition: Association Rule

- **Association Rule**

- An implication expression of the form $X \rightarrow Y$, where X and Y are itemsets
- Example:
 $\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$

- **Rule Evaluation Metrics**

- Support (s)
 - ◆ Fraction of transactions that contain both X and Y
- Confidence (c)
 - ◆ Measures how often items in Y appear in transactions that contain X

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk, Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|T|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

Mining Association Rules

► Problem statement

Given a *minimum support* (min_sup), also called *support threshold*, and a *minimum confidence* (min_conf), also called *confidence threshold*, find all association rules that satisfy both min_sup and min_conf from a data set D .

Definition: Frequent Itemset

- **Itemset**

- A collection of one or more items
 - ◆ Example: {Milk, Bread, Diaper}
- k-itemset
 - ◆ An itemset that contains k items

- **Support count (σ)**

- Frequency of occurrence of an itemset
- E.g. $\sigma(\{\text{Milk, Bread, Diaper}\}) = 2$

- **Support**

- Fraction of transactions that contain an itemset
- E.g. $s(\{\text{Milk, Bread, Diaper}\}) = 2/5$

- **Frequent Itemset**

- An itemset whose support is greater than or equal to a *minsup* threshold

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Mining Association Rules

- Two-step approach:
 1. Frequent Itemset Generation
 - Generate all itemsets whose support $\geq \text{minsup}$
 2. Rule Generation
 - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

Support and Confidence: Example

$$\text{support}(X \rightarrow Y) = P(X \cup Y)$$

$$\text{confidence}(X \rightarrow Y) = P(Y|X)$$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Relative frequency is used to estimate the Probability.

- ▶ $A \rightarrow C$
- ▶ $C \rightarrow A$
- ▶ $A \& C \rightarrow B$
- ▶ $A \& B \rightarrow E$

Support and Confidence: Example

$$\text{support}(A \rightarrow B) = P(A \cup B)$$

$$\text{confidence}(A \rightarrow B) = P(B|A)$$

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- ▶ $A \rightarrow C$ (50%, 66.7%)
- ▶ $C \rightarrow A$ (50%, 100%)
- ▶ $A \& C \rightarrow B$ (25%, 50%)
- ▶ $A \& B \rightarrow E$ (0%, 0%)

(11)

How to Mine Association Rules

- A two-step process:
 - Find all frequent itemsets ---- the key step
 - Generate strong association rules from frequent itemsets.
- Example: given min_sup=50% and min_conf=50%

Transaction ID	Items Bought	Frequent Itemset	Support
2000	A,B,C	{A}	75%
1000	A,C	{B}	50%
4000	A,D	{C}	50%
5000	B,E,F	{A, C}	50%

- Generate strong rules:
 - $\{A\} \rightarrow \{C\}$ [support=50%, confidence=66.6%]
 - $\{C\} \rightarrow \{A\}$ [support=50%, confidence=100%]

(r2)

Finding Frequent Itemsets

- ▶ Objective: find the itemsets that satisfy the minimum support count.
- ▶ Algorithms
 - ▶ Apriori
 - ▶ FP-Growth
 - ▶ H-Mine
 - ▶ Partition
 - ▶ CLOSET
 - ▶ CHARM
 - ▶ etc.

(13)

The Apriori Algorithm — Example

Database D	
TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C_1

Scan D

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Scan D

itemset	sup.
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

itemsetsct	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

itemset
{2 3 5}

Scan D

itemset	sup
{2 3 5}	2

Assume:
min_sup_count = 2

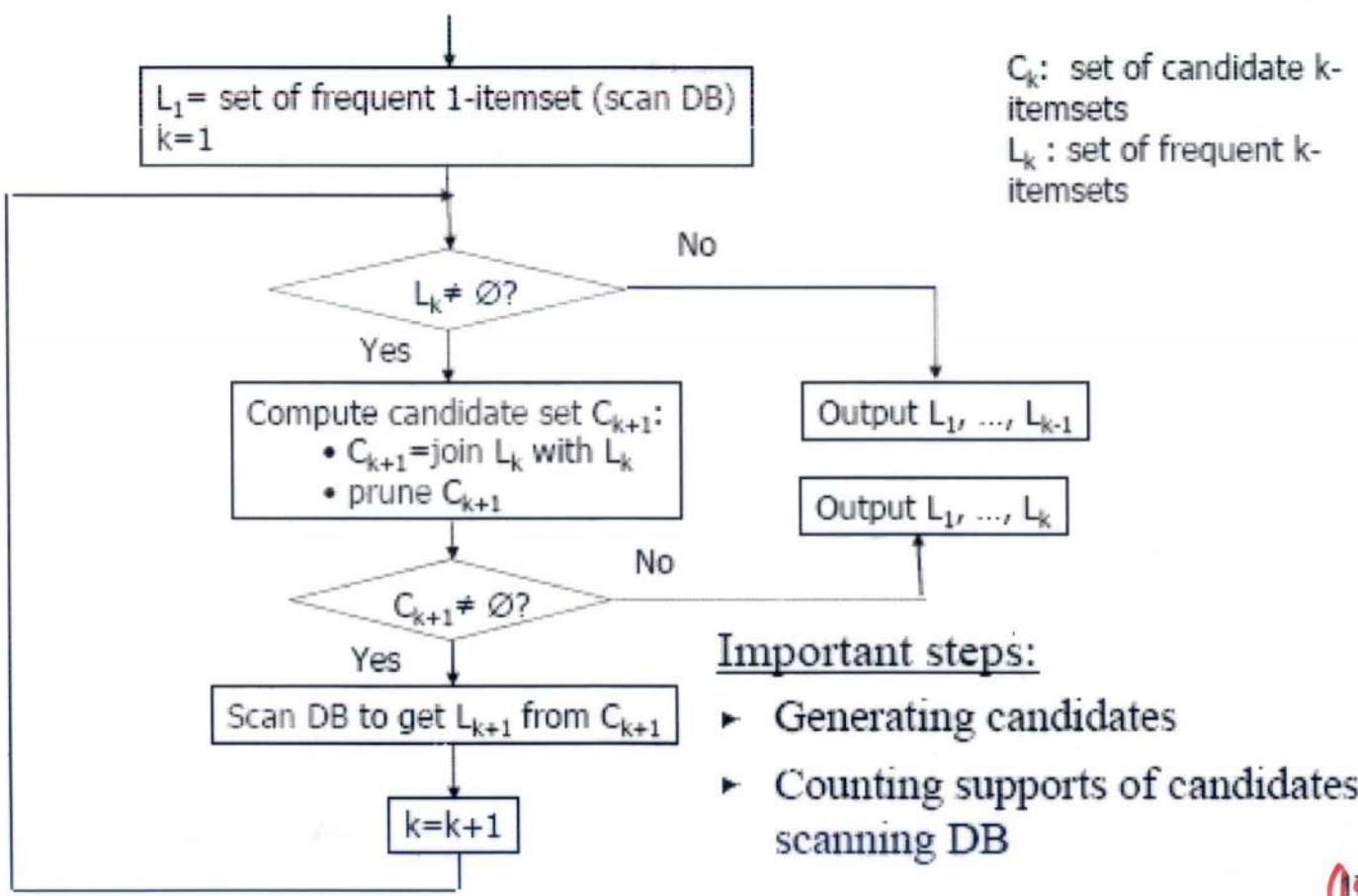
(17)

Apriori

- ▶ The Apriori property (an anti-monotone property), also called *downward closure* property:
Any nonempty subset of a frequent itemset must be frequent
 - ▶ i.e., if $\{A,B,C\}$ is a frequent itemset, $\{A,B\}$, $\{A,C\}$, $\{B,C\}$, $\{A\}$, $\{B\}$ and $\{C\}$ must also be frequent.
 - ▶ This is because a transaction containing $\{A, B, C\}$ also contains $\{A,B\}$, $\{B,C\}$, ...
- ▶ No superset of any infrequent itemset should be checked.
 - ▶ Many item combinations can be pruned from the search space.
- ▶ Apriori-based mining (level-wise iterations)
 - ▶ Generate length $(k+1)$ candidate itemsets from frequent k -itemsets
 - ▶ Test the candidates against DB



Apriori Algorithm (Flow Chart)



The Apriori Algorithm

- Based on the Apriori property, use *iterative level-wise approach* and *candidate generation-and-test*
- Pseudo-code:

C_k : a set of candidate itemsets of size k
 L_k : the set of frequent itemsets of size k

$L_1 = \{\text{frequent items}\};$ (*Scan database to find all frequent 1-itemsets*)
for ($k = 1; L_k \neq \emptyset; k++$) do begin

$C_{k+1} = \text{candidates generated from } L_k;$

for each transaction t in database do

increment the count of all candidates in C_{k+1}
that are contained in t

$L_{k+1} = \text{candidates in } C_{k+1} \text{ satisfying min_support}$

end

return $\cup_k L_{k-1};$

Level-wise Generation process: $L_k \rightarrow C_{k+1} \rightarrow L_{k+1}$

(20)

Scan database to calculate support for each itemset in C_{k+1}

How to Generate Candidates?

(i.e., How to Generate C_{k+1} from L_k)

- Given L_k = the set of frequent k -itemsets

- List the items in each itemset of L_k in an order

$$L_3 =$$

{1 2 3}
{1 2 4}
{1 3 4}
{1 3 5}
{2 3 4}

- Given L_k , generate C_{k+1} in two steps:

- Join Step: Join L_k with L_k by joining two k -itemsets in L_k . Two k -itemsets are joinable if their first ($k-1$) items are the same and the last item in the first itemset is smaller than the last item in the second itemset (the condition for joining two members of L_k).

Now, $C_4 = \{\{1 2 3 4\}, \{1 3 4 5\}\}$

- Prune Step: Delete all candidates in C_{k+1} that have a non-frequent subset by checking all length- k subsets of a candidate

- Now, $C_4 = \{\{1 2 3 4\}\}$

(2D)

Example of Candidate-generation

- ▶ $L_4 = \{abcd, abcg, abdg, abef, abeh, acdg, bcdg\}$
- ▶ Self-joining: $L_4 * L_4$
 - ▶ $abcdg$ from $abcd$ and $abcg$
 - ▶ $abefh$ from $abef$ and $aceh$
- ▶ Pruning:
 - ▶ $abefh$ is removed because $abfh$ or $aefh$ or $befh$ is not in L_4
 - ▶ $C_5 = \{abcdg\}$

Generate Association Rules from Frequent Itemsets

- ▶ Naïve algorithm:

for each frequent itemset l do

 for each nonempty proper subset s of l do

 if ($\text{support}(l)/\text{support}(s) \geq \text{min_conf}$)

 output the rule $s \rightarrow l-s$, with

 support = $\text{support}(l)$ and

 confidence = $\text{support}(l)/\text{support}(s)$



Generate Association Rules from Frequent Itemsets

► Example:

- Given a frequent itemset: $I = \{I_1, I_2, I_5\}$
- nonempty subsets of I are $\{I_1, I_2\}$, $\{I_1, I_5\}$, $\{I_2, I_5\}$, $\{I_1\}$, $\{I_2\}$, $\{I_5\}$
- resulting association rules:

$I_1 \wedge I_2 \rightarrow I_5$,	<i>confidence</i>	= 50%
$I_1 \wedge I_5 \rightarrow I_2$,	<i>confidence</i>	= 100% ✓
$I_2 \wedge I_5 \rightarrow I_1$,	<i>confidence</i>	= 100% ✓
$I_1 \rightarrow I_2 \wedge I_5$,	<i>confidence</i>	= 33%
$I_2 \rightarrow I_1 \wedge I_5$,	<i>confidence</i>	= 29%
$I_5 \rightarrow I_1 \wedge I_2$,	<i>confidence</i>	= 100% ✓

If minimum confidence threshold is 70%, only 3 rules are output.

(P4)

Example for Association Rule Mining.

For the following TDB, find all association rules with 50% support and 75% confidence.

TID	Items
100	Bread, Cheese, Eggs, Juice
200	Bread, Cheese, Juice
300	Bread, Milk, Yogurt
400	Bread, Juice, Milk
500	Cheese, Juice, Milk

Assume Minimum Support Count = 3

Solution:

C1: Stem	Sup	L1: Stem	Sup
Bread	4	Bread	4
Cheese	3	Cheese	3
Juice	4	Juice	4
Milk	3	Milk	3
* Eggs	1		
* Yogurt	1		

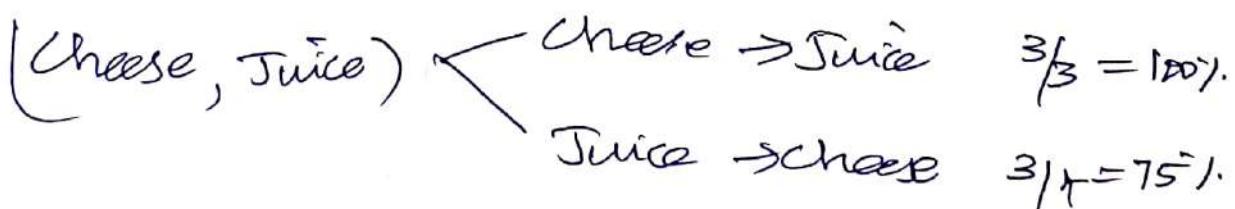
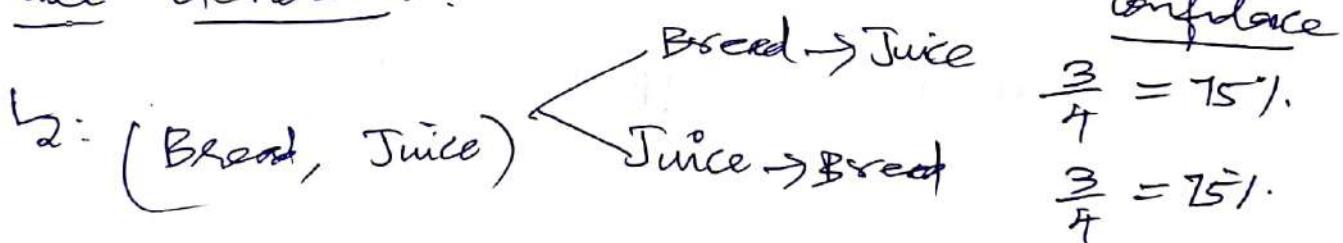
Σ_2 : Itemset	Sup. Count	
(Bread, Cheese)	2	x
(Bread, Juice)	3	
(Bread, Milk)	2	x
(Cheese, Juice)	2	
(Cheese, Milk)	1	x
(Juice, Milk)	2	x

⇒

Σ_2 : Itemset	Sup
(Bread, Juice)	3
(Cheese, Juice)	3

No C₃

Rule Generation:



∴ All have minimum 75% confidence, all qualify.

Principal Component Analysis

Dimension Reduction

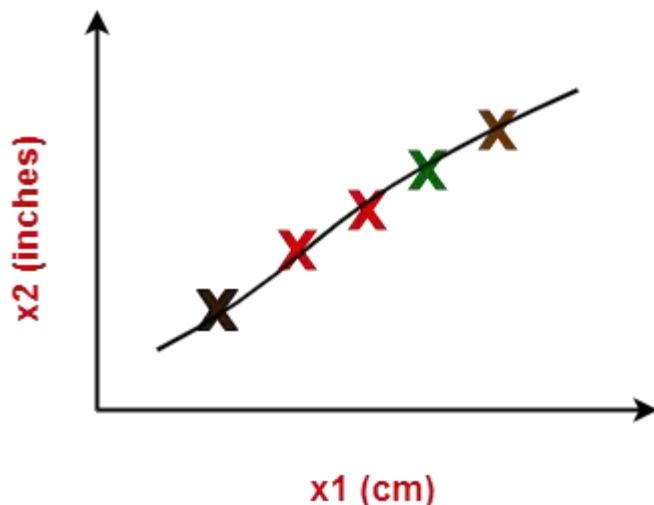
In pattern recognition, Dimension Reduction is defined as-

- It is a process of converting a data set having vast dimensions into a data set with lesser dimensions.
- It ensures that the converted data set conveys similar information concisely.

Example-

Consider the following example-

- The following graph shows two dimensions x_1 and x_2 .
- X_1 represents the measurement of several objects in cm.
- X_2 represents the measurement of several objects in inches.

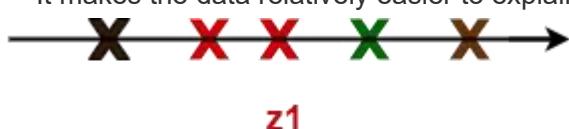


In machine learning,

- Using both these dimensions convey similar information.
- Also, they introduce a lot of noise in the system.
- So, it is better to use just one dimension.

Using dimension reduction techniques-

- We convert the dimensions of data from 2 dimensions (x_1 and x_2) to 1 dimension (z_1).
- It makes the data relatively easier to explain.



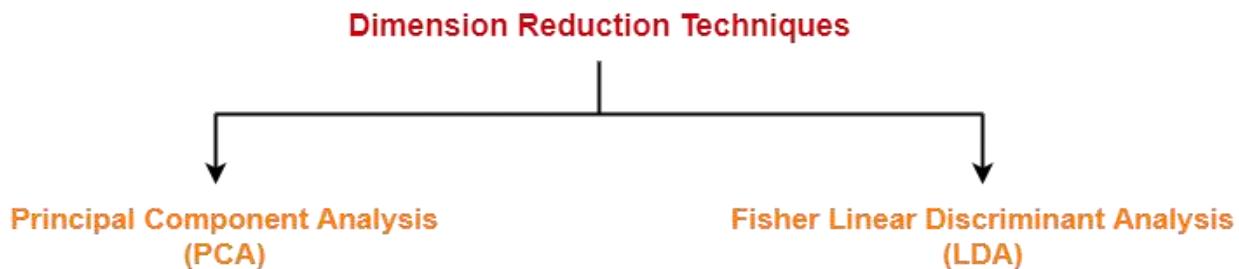
Benefits-

Dimension reduction offers several benefits such as-

- It compresses the data and thus reduces the storage space requirements.
- It reduces the time required for computation since less dimensions require less computation.
- It eliminates the redundant features.
- It improves the model performance.

Dimension Reduction Techniques-

The two popular and well-known dimension reduction techniques are-



Principal Component Analysis-

- Principal Component Analysis is a well-known dimension reduction technique.
- It transforms the variables into a new set of variables called as principal components.
- These principal components are linear combination of original variables and are orthogonal.
- There can be only two principal components for a two-dimensional data set.

PCA Algorithm-

The steps involved in PCA Algorithm are as follows-

Step-01: Get data.

Step-02: Compute the mean vector (μ).

Step-03: Subtract mean from the given data.

Step-04: Calculate the covariance matrix.

Step-05: Calculate the Eigen vectors and Eigen values of the covariance matrix.

Step-06: Choosing components and forming a feature vector.

Step-07: Deriving the new data set.

PRACTICE PROBLEMS BASED ON PRINCIPAL COMPONENT ANALYSIS-

Problem-01:

Given data = {2, 3, 4, 5, 6, 7; 1, 5, 3, 6, 7, 8}.

Compute the principal component using PCA Algorithm.

OR

Consider the two dimensional patterns (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).

Compute the principal component using PCA Algorithm.

OR

Compute the principal component of following data-

CLASS 1

$$X = 2, 3, 4$$

$$Y = 1, 5, 3$$

CLASS 2

$$X = 5, 6, 7$$

$$Y = 6, 7, 8$$

Solution-

We use the above discussed PCA Algorithm-

Step-01:

Get data.

The given feature vectors are-

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$

- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

Step-02:

Calculate the mean vector (μ).

Mean vector (μ)

$$= ((2 + 3 + 4 + 5 + 6 + 7) / 6, (1 + 5 + 3 + 6 + 7 + 8) / 6) \\ = (4.5, 5)$$

Mean vector (μ) = $\begin{bmatrix} 4.5 \\ 5 \end{bmatrix}$

Step-03:

Subtract mean vector (μ) from the given feature vectors.

- $x_1 - \mu = (2 - 4.5, 1 - 5) = (-2.5, -4)$
- $x_2 - \mu = (3 - 4.5, 5 - 5) = (-1.5, 0)$
- $x_3 - \mu = (4 - 4.5, 3 - 5) = (-0.5, -2)$
- $x_4 - \mu = (5 - 4.5, 6 - 5) = (0.5, 1)$
- $x_5 - \mu = (6 - 4.5, 7 - 5) = (1.5, 2)$
- $x_6 - \mu = (7 - 4.5, 8 - 5) = (2.5, 3)$

Feature vectors (x_i) after subtracting mean vector (μ) are-

$$\begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 2.5 \\ 3 \end{bmatrix}$$

Step-04:

Calculate the covariance matrix.

Covariance matrix is given by-

$$\text{Covariance Matrix} = \frac{\sum (x_i - \mu)(x_i - \mu)^t}{n}$$

$$m_1 = (x_1 - \mu)(x_1 - \mu)^t = \begin{bmatrix} -2.5 \\ -4 \end{bmatrix} \begin{bmatrix} -2.5 & -4 \end{bmatrix}^t = \begin{bmatrix} 6.25 & 10 \\ 10 & 16 \end{bmatrix}$$

$$m_2 = (x_2 - \mu)(x_2 - \mu)^t = \begin{bmatrix} -1.5 \\ 0 \end{bmatrix} \begin{bmatrix} -1.5 & 0 \end{bmatrix}^t = \begin{bmatrix} 2.25 & 0 \\ 0 & 0 \end{bmatrix}$$

$$m_3 = (x_3 - \mu)(x_3 - \mu)^t = \begin{bmatrix} -0.5 \\ -2 \end{bmatrix} \begin{bmatrix} -0.5 & -2 \end{bmatrix}^t = \begin{bmatrix} 0.25 & 1 \\ 1 & 4 \end{bmatrix}$$

$$m_4 = (x_4 - \mu)(x_4 - \mu)^t = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \begin{bmatrix} 0.5 & 1 \end{bmatrix}^t = \begin{bmatrix} 0.25 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

$$m_5 = (x_5 - \mu)(x_5 - \mu)^t = \begin{bmatrix} 1.5 \\ 2 \end{bmatrix} \begin{bmatrix} 1.5 & 2 \end{bmatrix}^t = \begin{bmatrix} 2.25 & 3 \\ 3 & 4 \end{bmatrix}$$

$$m_6 = (x_6 - \mu)(x_6 - \mu)^t = \begin{bmatrix} 2.5 \\ 3 \end{bmatrix} \begin{bmatrix} 2.5 & 3 \end{bmatrix}^t = \begin{bmatrix} 6.25 & 7.5 \\ 7.5 & 9 \end{bmatrix}$$

Now,

Covariance matrix

$$= (m_1 + m_2 + m_3 + m_4 + m_5 + m_6) / 6$$

On adding the above matrices and dividing by 6, we get-

$$\text{Covariance Matrix} = \frac{1}{6} \begin{bmatrix} 17.5 & 22 \\ 22 & 34 \end{bmatrix}$$

$$\text{Covariance Matrix} = \begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix}$$

Step-05:

Calculate the Eigen values and Eigen vectors of the covariance matrix.

λ is an eigen value for a matrix M if it is a solution of the characteristic equation $|M - \lambda I| = 0$.

So, we have-

$$\begin{vmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{vmatrix} - \begin{vmatrix} \lambda & 0 \\ 0 & \lambda \end{vmatrix} = 0$$

$$\begin{vmatrix} 2.92 - \lambda & 3.67 \\ 3.67 & 5.67 - \lambda \end{vmatrix} = 0$$

From here,

$$(2.92 - \lambda)(5.67 - \lambda) - (3.67 \times 3.67) = 0$$

$$16.56 - 2.92\lambda - 5.67\lambda + \lambda^2 - 13.47 = 0$$

$$\lambda^2 - 8.59\lambda + 3.09 = 0$$

Solving this quadratic equation, we get $\lambda = 8.22, 0.38$

Thus, two eigen values are $\lambda_1 = 8.22$ and $\lambda_2 = 0.38$.

Clearly, the second eigen value is very small compared to the first eigen value.

So, the second eigen vector can be left out.

Eigen vector corresponding to the greatest eigen value is the principal component for the given data set.

So. we find the eigen vector corresponding to eigen value λ_1 .

We use the following equation to find the Eigen vector-

$$MX = \lambda X$$

where-

- M = Covariance Matrix
- X = Eigen vector
- λ = Eigen value

Substituting the values in the above equation, we get-

$$\begin{bmatrix} 2.92 & 3.67 \\ 3.67 & 5.67 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = 8.22 \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

Solving these, we get-

$$2.92X_1 + 3.67X_2 = 8.22X_1$$

$$3.67X_1 + 5.67X_2 = 8.22X_2$$

On simplification, we get-

$$5.3X_1 = 3.67X_2 \dots\dots\dots (1)$$

$$3.67X_1 = 2.55X_2 \dots\dots\dots (2)$$

From (1) and (2), $X_1 = 0.69X_2$

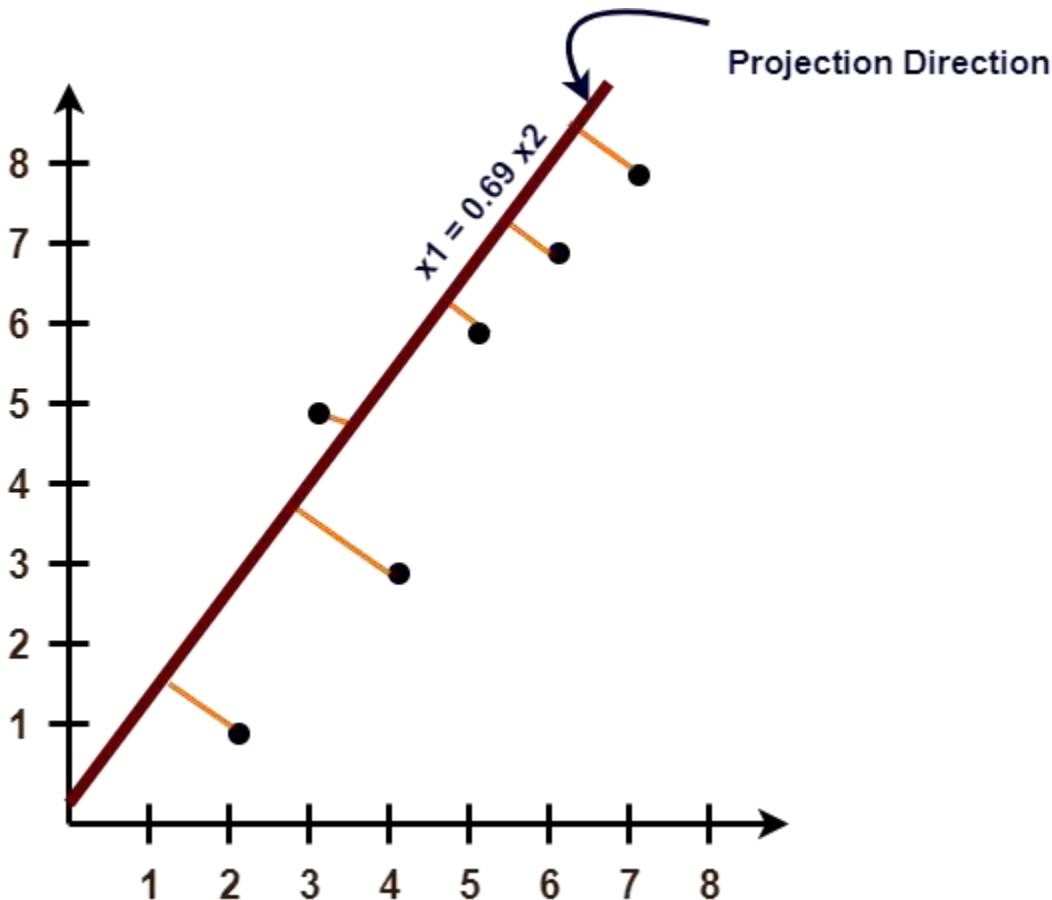
From (2), the eigen vector is-

Eigen Vector :
$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

Thus, principal component for the given data set is-

Principal Component :
$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

we project the data points onto the new subspace as-



Problem-02:

Use PCA Algorithm to transform the pattern (2, 1) onto the Eigen vector in the previous question.

Solution-

The given feature vectors are-

- $x_1 = (2, 1)$
- $x_2 = (3, 5)$
- $x_3 = (4, 3)$
- $x_4 = (5, 6)$
- $x_5 = (6, 7)$
- $x_6 = (7, 8)$

Principal Component :

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}$$

The given feature vector is (2, 1).

Given Feature Vector :

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

The feature vector gets transformed to

= Transpose of Eigen vector x (Feature Vector – Mean Vector)

$$= \begin{bmatrix} 2.55 \\ 3.67 \end{bmatrix}^T \times \left(\begin{bmatrix} 2 \\ 1 \end{bmatrix} - \begin{bmatrix} 4.5 \\ 5 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 2.55 & 3.67 \end{bmatrix} \times \begin{bmatrix} -2.5 \\ -4 \end{bmatrix}$$

$$= -21.055$$

Clustering

K-means Ex:-1

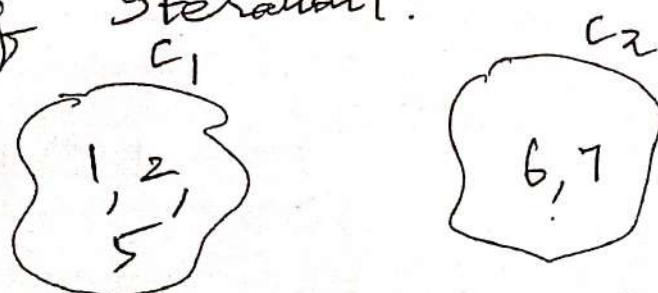
For simplicity, consider 1-7 objects and $k=2$
objects: 1, 2, 5, 6, 7

Randomly choose 5 and 6 as centroids

Identical objects	Dist from C1	Dist from C2	Allocated to cluster
1	4	5	C1
2	3	3	C1 or C2
5	0	1	C1
6	1	0	C2
7	2	1	C2

Assume absolute value of difference as the distance measure.

End of iteration:



$$\text{Mean of } C_1 = 8/3 = \underline{\underline{2.67}} \quad \text{Mean of } C_2 = \underline{\underline{6.5}}$$

Step 2: Centroid of cluster 1 = 2.67 Centroid of cluster 2 = 6.5

Object	Dist from c_1	Dist from c_2	Allocated to cluster
1	<u>1.67</u>	5.5	c_1
2	<u>0.67</u>	4.5	c_1
5	2.33	<u>1.5</u>	c_2
6	4.67	<u>0.5</u>	c_2
7	4.33	<u>0.5</u>	c_2



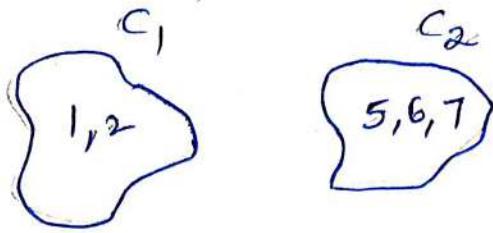
$$\text{Mean of } c_1 = 1.5 \quad \text{Mean of } c_2 = 6$$

Step 3: Centroid of $c_1 = 1.5$ Centroid of $c_2 \} = 6$

Obj	Dist from c_1	Dist from c_2	Allocated to cluster
1	<u>0.5</u>	5	c_1
2	<u>0.5</u>	4	c_1
5	3.5	<u>1</u>	c_2
6	4.5	<u>0</u>	c_2
7	5.5	<u>1</u>	c_2

Pg : ②

End of Iter 3



\therefore No change in cluster membership,
solution converges.

Example - 2:

- Consider the data about students given in the table below. Group them into three clusters, assuming s_1, s_2 and s_3 as initial seeds/centroids.

Student	Age	Mark 1	Mark 2	Mark 3
s_1	18	73	75	51
s_2	18	79	85	75
s_3	23	70	70	52
s_4	20	55	55	55
s_5	22	85	86	87
s_6	19	91	96	89
s_7	20	70	65	60
s_8	21	53	56	59
s_9	19	82	82	60
s_{10}	47	75	96	77

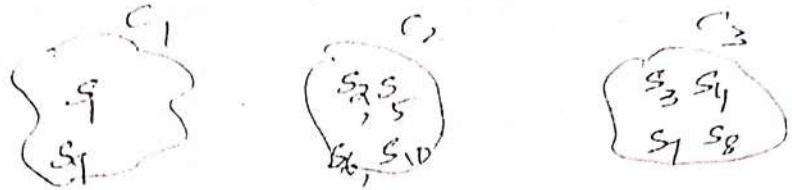
Pg : ③

The three initial seeds

Student	Age	Mark1	Mark2	Mark3
S ₁	18	73	75	57
S ₂	18	79	85	75
S ₃	23	70	70	52

Iteration:

List from					Allocate to cluster	
	C ₁	C ₂	C ₃			
c ₁	18	73	75	57		
c ₂	18	79	85	75		
c ₃	23	70	70	52		
S ₁	18	73	75	57	0	C ₁
S ₂	18	79	85	75	34	C ₂
S ₃	23	70	70	52	18	C ₃
S ₄	20	55	55	55	42	C ₃
S ₅	22	85	86	87	57	C ₂
S ₆	19	91	90	89	66	C ₂
S ₇	20	70	65	60	18	C ₃
S ₈	21	53	56	59	44	C ₃
S ₉	17	82	82	60	20	C ₁
S ₁₀	47	75	76	71	52	C ₂
						F5 : C ₄



∴ New Centroids of:

C_1 = Average of S_1 and S_9

C_2 = " " $S_2 S_5 S_6 S_{10}$

C_3 = " " $S_3 S_4 S_7 S_8$

x Cluster Centroid	Age	Mark1	Mark2	Mark3
C_1	18.5	77.5	78.5	58.5
C_2	26.5	82.5	84.3	82.0
C_3	21	61.5	61.5	56.5

Iteration:- - Find the distance of each object from the new cluster centroids:-

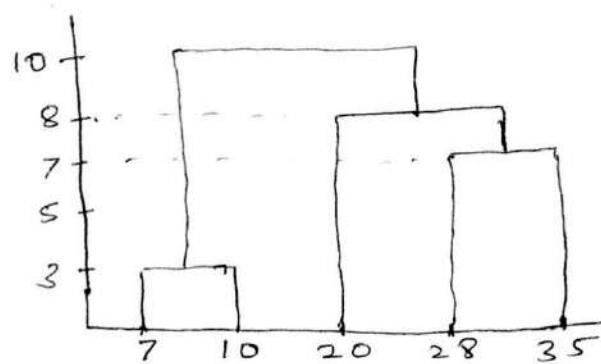
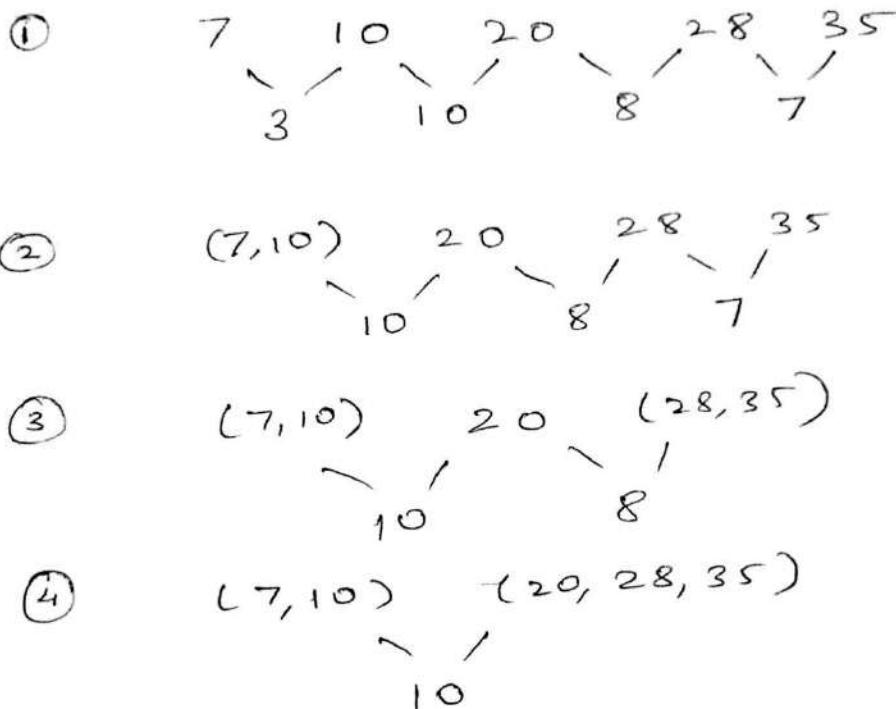
	Data from							
	C_1	C_2	C_3		Alloted to cluster			
C_1	18.5	77.5	78.5	58.5				
C_2	26.5	62.5	84.5	82				
C_3	21	61.5	61.5	56.5				
S_1	18	73	75	51	10	52.3	28	C_1
S_2	18	71	85	75	25	19.8	62	C_2
S_3	23	70	70	52	24	60.3	23	C_3
S_4	20	55	55	55	51	90.3	16	C_3
S_5	22	85	86	81	49	13.8	74	C_2
S_6	19	91	90	89	56	28.8	92	C_2
S_7	20	70	65	60	24	60.3	16	C_3
S_8	21	53	56	59	50	86.3	17	C_3
S_9	19	82	82	60	10	32.3	46	C_1
S_{10}	17	75	76	77	52	44.3	74	C_2
	C_1	C_2	C_3					
	S_1 S_7	S_2 S_5 S_6 S_{10}	S_3 S_4 S_7 S_8					

∴ No change in clusters. Stop.

Objective: For the one dimensional data set {7, 10, 20, 28, and 35}, perform hierarchical clustering and plot the dendrogram to visualize it.

1. **Single Linkage:** In single link hierarchical clustering, we merge in each step the two clusters, whose two closest members have the smallest distance.

Single Linkage



(Dendrogram)

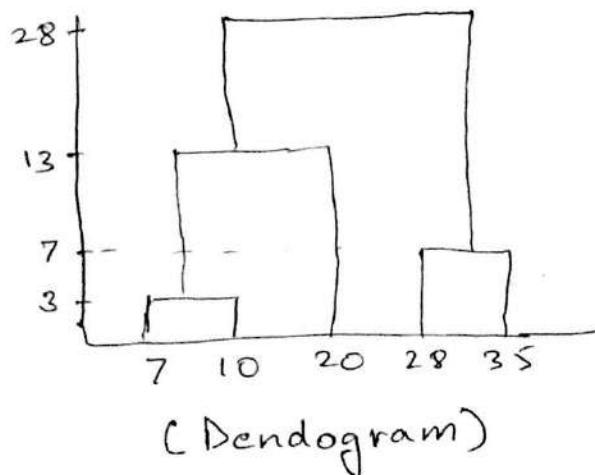
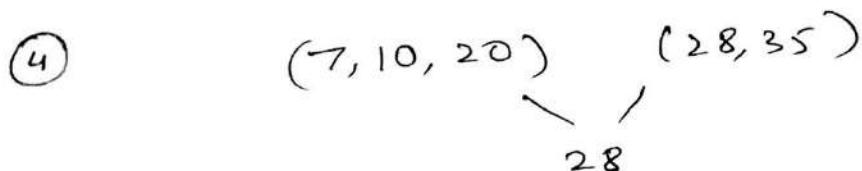
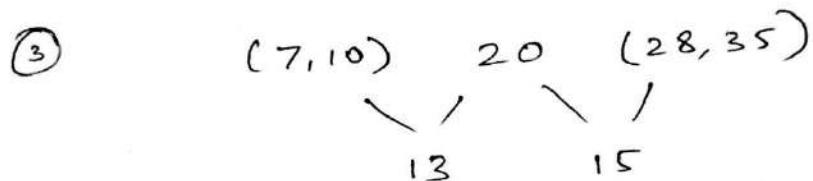
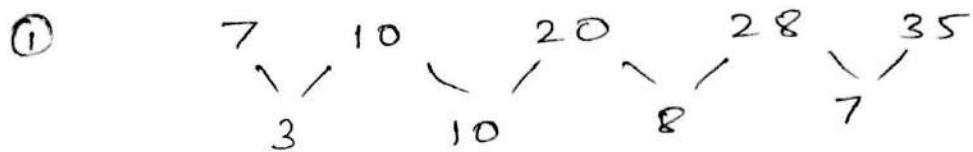
Using single linkage two clusters are formed:

Cluster 1 : (7,10)

Cluster 2 : (20,28,35)

2. Complete Linkage: In complete link hierarchical clustering, we merge in the members of the clusters in each step, which provide the smallest maximum pairwise distance.

Complete Linkage



Using complete linkage two clusters are formed:

Cluster 1 : $(7, 10, 20)$

Cluster 2 : $(28, 35)$

Before we start understanding the algorithm, go through some definitions which are explained in my previous post.

Consider the following dataset and we will find frequent itemsets and generate association rules for them.

TID	items
1	I1, I2, I5
2	I2, I4
3	I2, I3
4	I1, I2, I4
5	I1, I3
6	I2, I3
7	I1, I3
8	I1, I2, I3, I5
9	I1, I2, I3

minimum support count is 2
minimum confidence is 60%

Step-1: K=1

(I) Create a table containing support count of each item present in dataset – Called **C1(candidate set)**

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

C₁ candidate set

(II) compare candidate set item's support count with minimum support count(here min_support=2 if support_count of candidate set items is less than min_support then remove those items). This gives us itemset L1.

Itemset	sup_count
I1	6
I2	7
I3	6
I4	2
I5	2

L₁ dataset

Step-2: K=2

K-2

2-2=0

(K-2), 2-2=0

- Generate candidate set C2 using L1 (this is called join step). Condition of joining L_{k-1} and L_{k-1} is that it should have $(K-2)$ elements in common.
- Check all subsets of an itemset are frequent or not and if not frequent remove that itemset.(Example subset of {I1, I2} are {I1}, {I2} they are frequent.Check for each itemset)
- Now find support count of these itemsets by searching in dataset.

Candidate set C2

Itemset	sup_count
I1, I2	4
I1, I3	4
I1, I4	1
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2
I3, I4	0
I3, I5	1
I4, I5	0

(II) compare candidate (C2) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L2.

$$K=3 \quad (K-2) \\ 3-2=1$$

$(l_2, l_3, l_4) \times$
 \downarrow
 $(l_2, l_3) (l_3, l_4) (l_2, l_4)$
 $4 \quad 0 \quad 2$

Step-3:

- Generate candidate set C3 using L2 (join step). Condition of joining L_{k-1} and L_{k-1} is that it should have $(K-2)$ elements in common. So here, for L2, first element should match.

So itemset generated by joining L2 is {I1, I2, I3}{I1, I2, I5}{I1, I3, I5}{I2, I3, I4}{I2, I4, I5}{I2, I3, I5}

- Check if all subsets of these itemsets are frequent or not and if not, then remove that itemset.(Here subset of {I1, I2, I3} are {I1, I2},{I2, I3},{I1, I3} which are frequent. For {I2, I3, I4}, subset {I3, I4} is not frequent so remove it. Similarly check for every itemset)

Itemset	sup_count
I1, I2	4
I1, I3	4
I1, I5	2
I2, I3	4
I2, I4	2
I2, I5	2
I2, I5	2

L2 dataset

$(l_1, l_2, l_3) \leftarrow$

\downarrow
 $(l_1, l_2) (l_2, l_3) (l_1, l_3)$
 $\downarrow \quad \downarrow \quad \downarrow$

~~(l₁, l₂, l₃, l₅)~~

- find support count of these remaining itemset by searching in dataset.

Itemset	sup_count
I1, I2, I3	2
I1, I2, I5	2

$$K-2 \\ 4-2 = 2$$

(II) Compare candidate (C3) support count with minimum support count(here min_support=2 if support_count of candidate set item is less than min_support then remove those items) this gives us itemset L3.

Step-4:

K=4

- Generate candidate set C4 using L3 (join step). Condition of joining L_{k-1} and L_{k-1} (K=4) is that, they should have (K-2) elements in common. So here, for L3, first 2 elements (items) should match.
- Check all subsets of these itemsets are frequent or not (Here itemset formed by joining L3 is {I1, I2, I3, I5} so its subset contains {I1, I3, I5}, which is not frequent). So no itemset in C4 .
- We stop here because no frequent itemsets are found further

Itemset	sup_count
I1, I2, I3	2
I1, I2, I5	2

$l_1 \rightarrow l_2 \quad l_3$

Thus, we have discovered all the frequent item-sets. Now generation of strong association rule comes into picture. For that we need to calculate confidence of each rule.

~~(l₁, l₂, l₃)~~

Confidence –

A confidence of 60% means that 60% of the customers, who purchased milk and bread also bought butter.

$$\text{Confidence } (A \rightarrow B) = \text{Support_count}(A \cup B) / \text{Support_count}(A)$$

So here, by taking an example of any frequent itemset, we will show the rule generation.

Itemset {I1, I2, I3} //from L3

$(l_1, l_2) \rightarrow l_3, (l_1, l_3) \rightarrow l_2$

SO rules can be

$$\{ [I1 \wedge I2] \Rightarrow [I3] \text{ //confidence} = \text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I1 \wedge I2) = 2/4 * 100 = 50\% \}$$

$$\{ [I1 \wedge I3] \Rightarrow [I2] \text{ //confidence} = \text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I1 \wedge I3) = 2/4 * 100 = 50\% \}$$

$$\{ [I2 \wedge I3] \Rightarrow [I1] \text{ //confidence} = \text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I2 \wedge I3) = 2/4 * 100 = 50\% \}$$

$$\{ [I1] \Rightarrow [I2 \wedge I3] \text{ //confidence} = \text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I1) = 2/6 * 100 = 33\% \}$$

$l_1 \rightarrow \{ l_2, l_3 \}$

$[I2] \Rightarrow [I1 \wedge I3]$ //confidence = $\text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I2) = 2/7 * 100 = 28\%$

$[I3] \Rightarrow [I1 \wedge I2]$ //confidence = $\text{sup}(I1 \wedge I2 \wedge I3) / \text{sup}(I3) = 2/6 * 100 = 33\%$

So if minimum confidence is 50%, then first 3 rules can be considered as strong association rules.

Limitations of Apriori Algorithm

Apriori Algorithm can be slow. The main limitation is time required to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets i.e. it is not an efficient approach for large number of datasets.

MODULE -4

ARTIFICIAL NEURAL NETWORKS

CONTENT

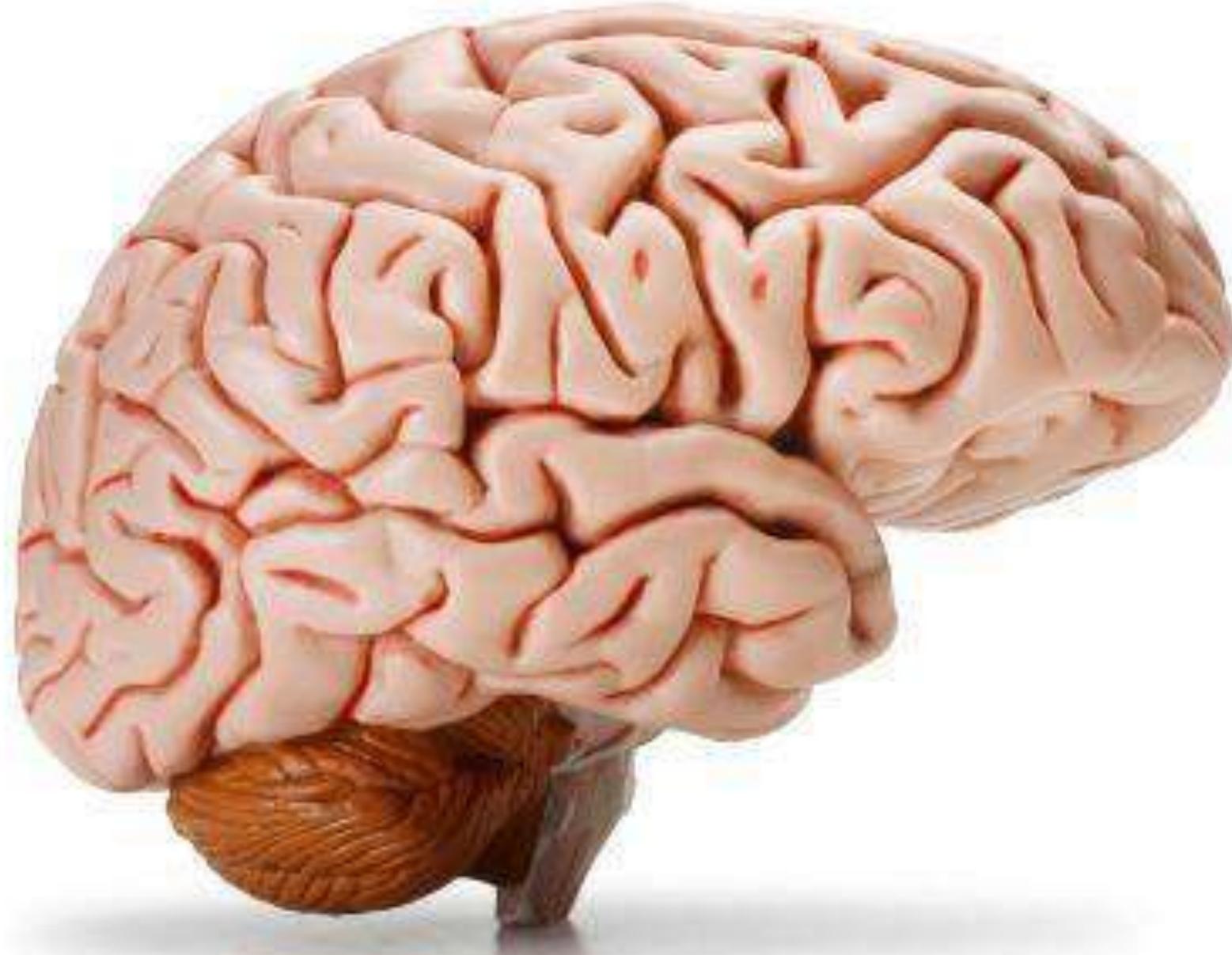
- Introduction
- Neural Network Representation
- Appropriate Problems for Neural Network Learning
- Perceptrons
- Multilayer Networks and BACKPROPAGATION Algorithms
- Remarks on the BACKPROPAGATION Algorithms

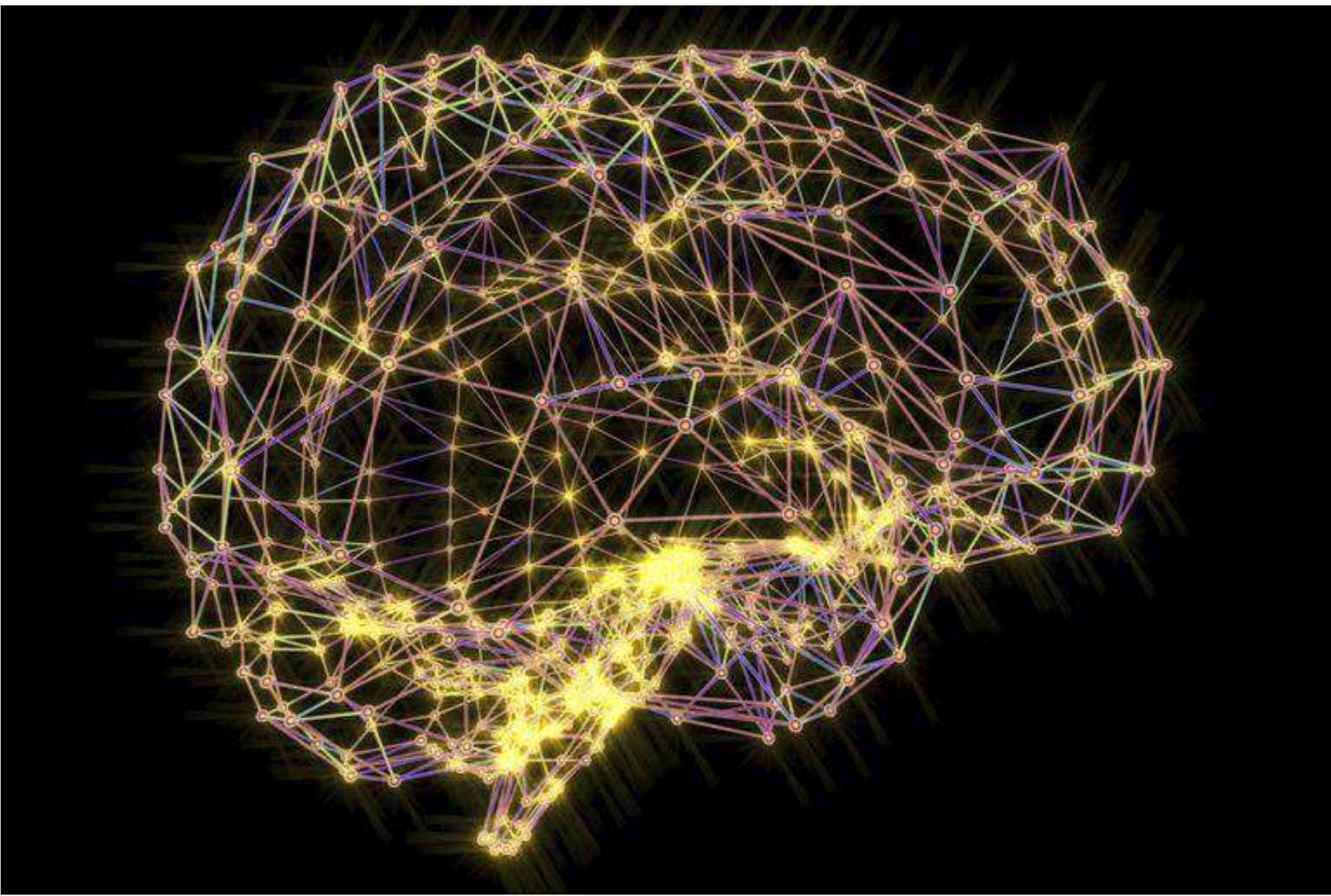
INTRODUCTION

Artificial neural networks (ANNs) provide a general, practical method for learning real-valued, discrete-valued, and vector-valued target functions from examples.

Biological Motivation

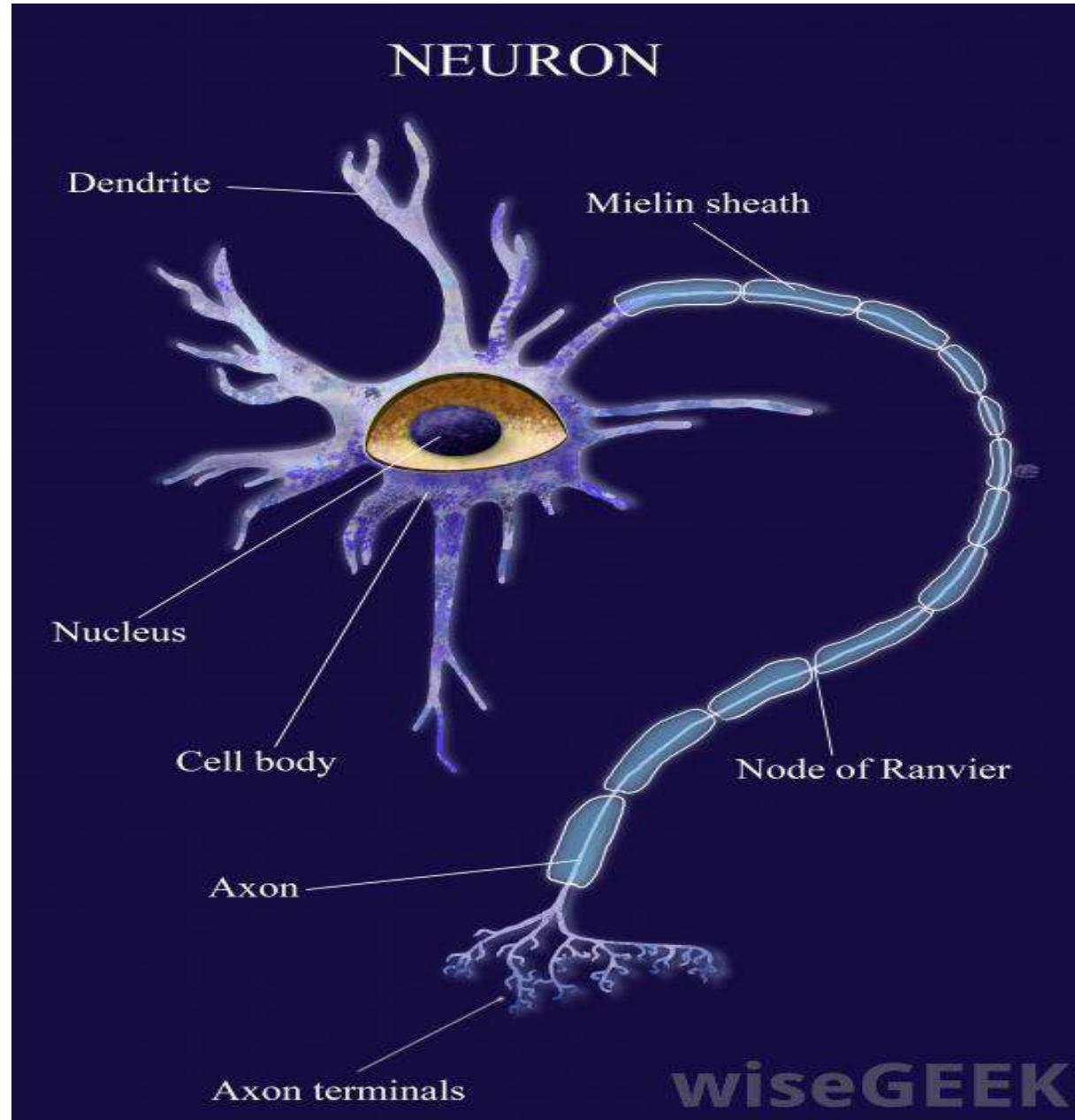
- The study of artificial neural networks (ANNs) has been inspired by the observation that biological learning systems are built of very complex webs of interconnected ***Neurons***
- Human information processing system consists of brain neuron: basic building block cell that communicates information to and from various parts of body
- Simplest model of a neuron: considered as a threshold unit –a processing element (PE)
- Collects inputs & produces output if the sum of the input exceeds an internal threshold value



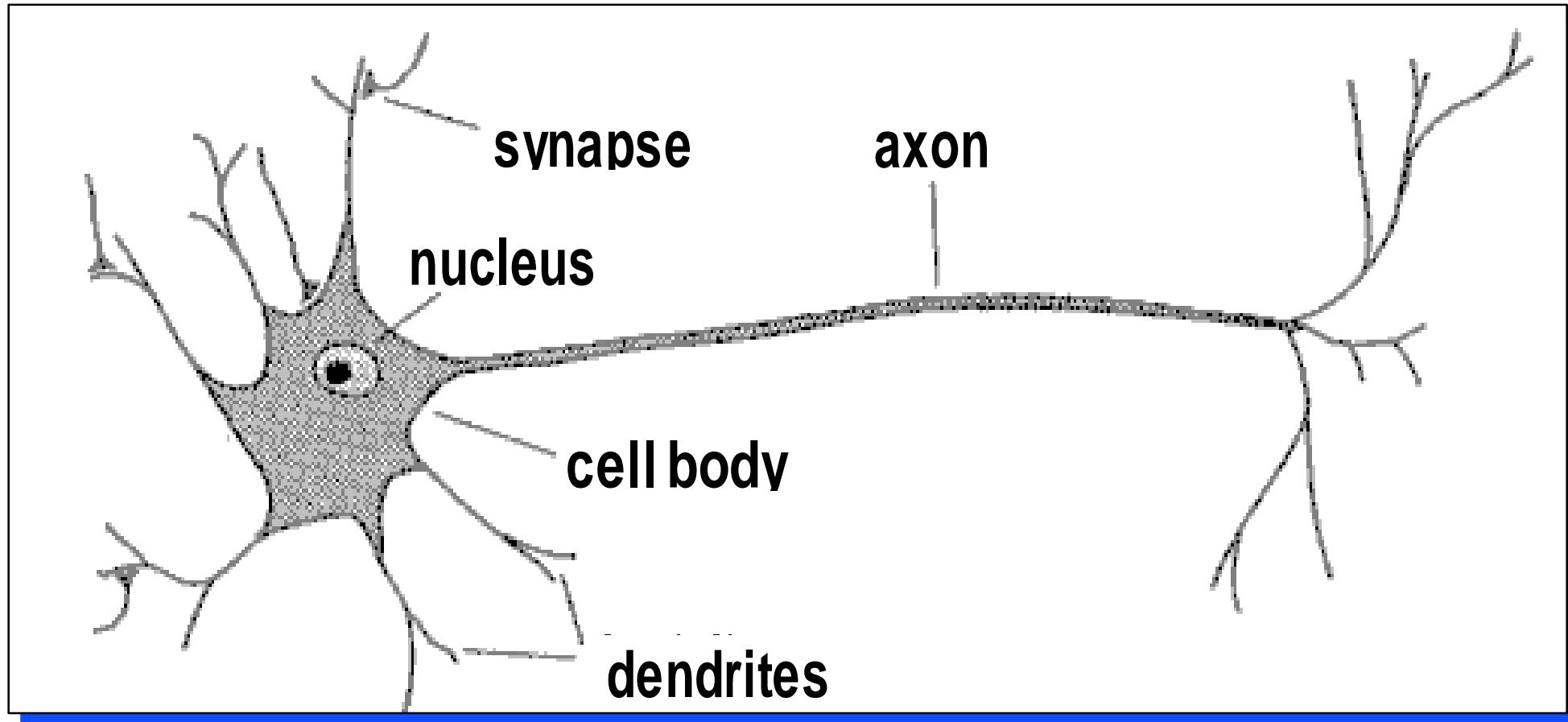








wiseGEEK



Facts of Human Neurobiology

- Number of neurons $\sim 10^{11}$
- Connection per neuron $\sim 10^{4-5}$
- Neuron switching time ~ 0.001 second or 10^{-3}
- Scene recognition time ~ 0.1 second
- 100 inference steps doesn't seem like enough
- Highly parallel computation based on distributed representation

Properties of Neural Networks

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically
- Input is a high-dimensional discrete or real-valued (e.g, sensor input)

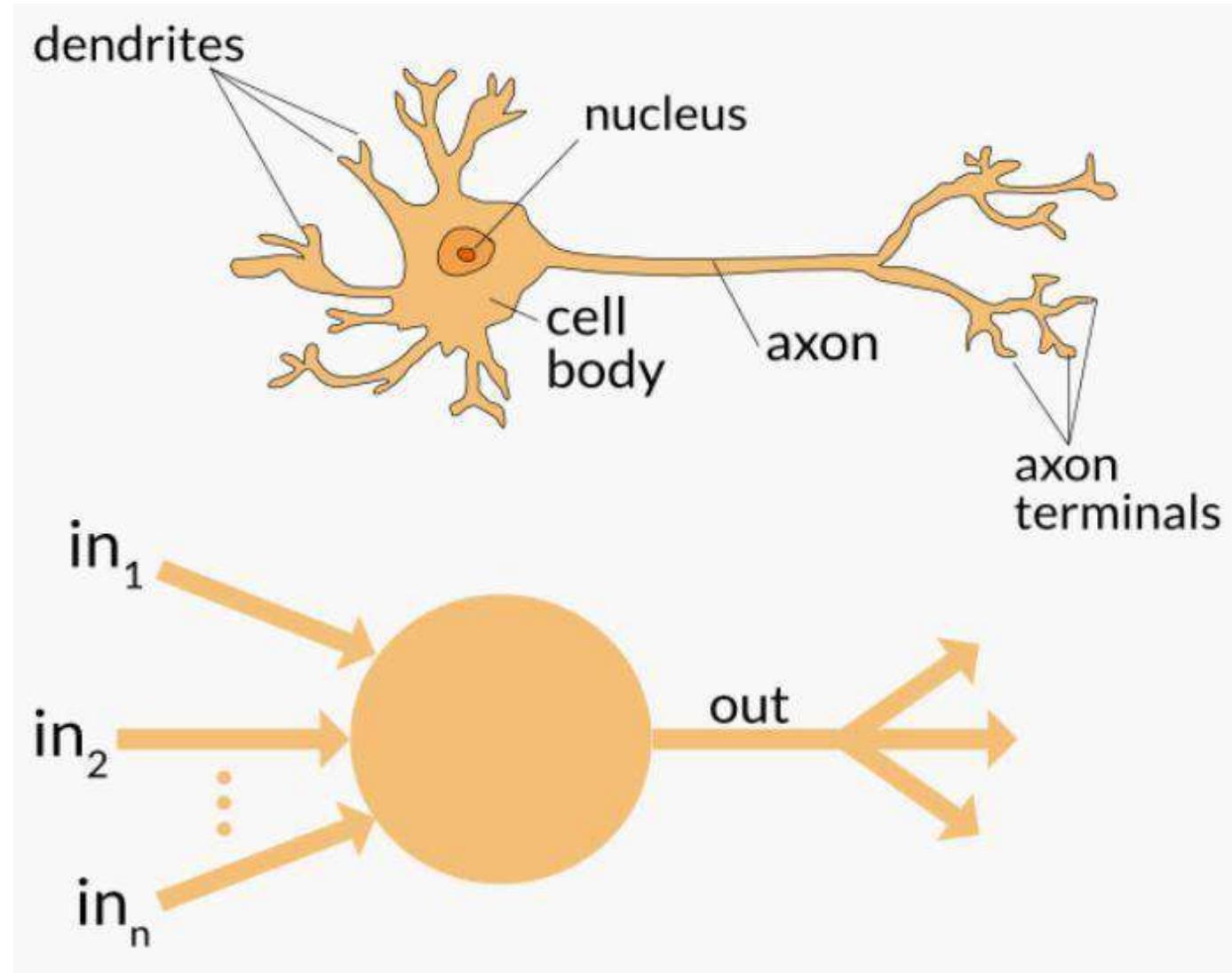
When to consider Neural Networks ?

- Input is a high-dimensional discrete or real-valued (e.g., sensor input)
- Output is discrete or real-valued
- Output is a vector of values
- Possibly noisy data
- Form of target function is unknown
- Human readability of result is unimportant

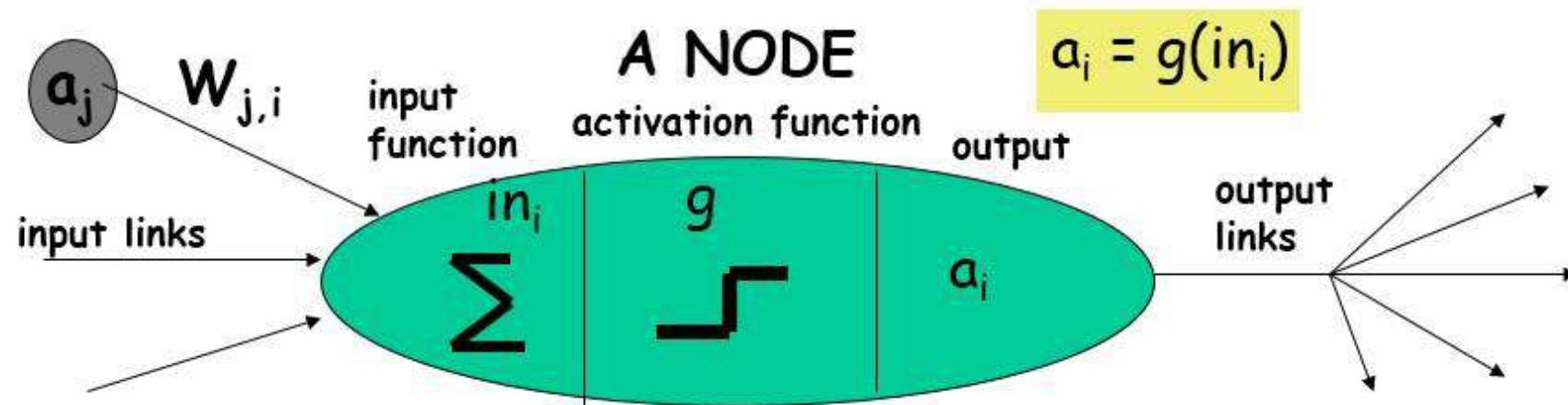
Examples:

1. Speech phoneme recognition
2. Image classification
3. Financial prediction

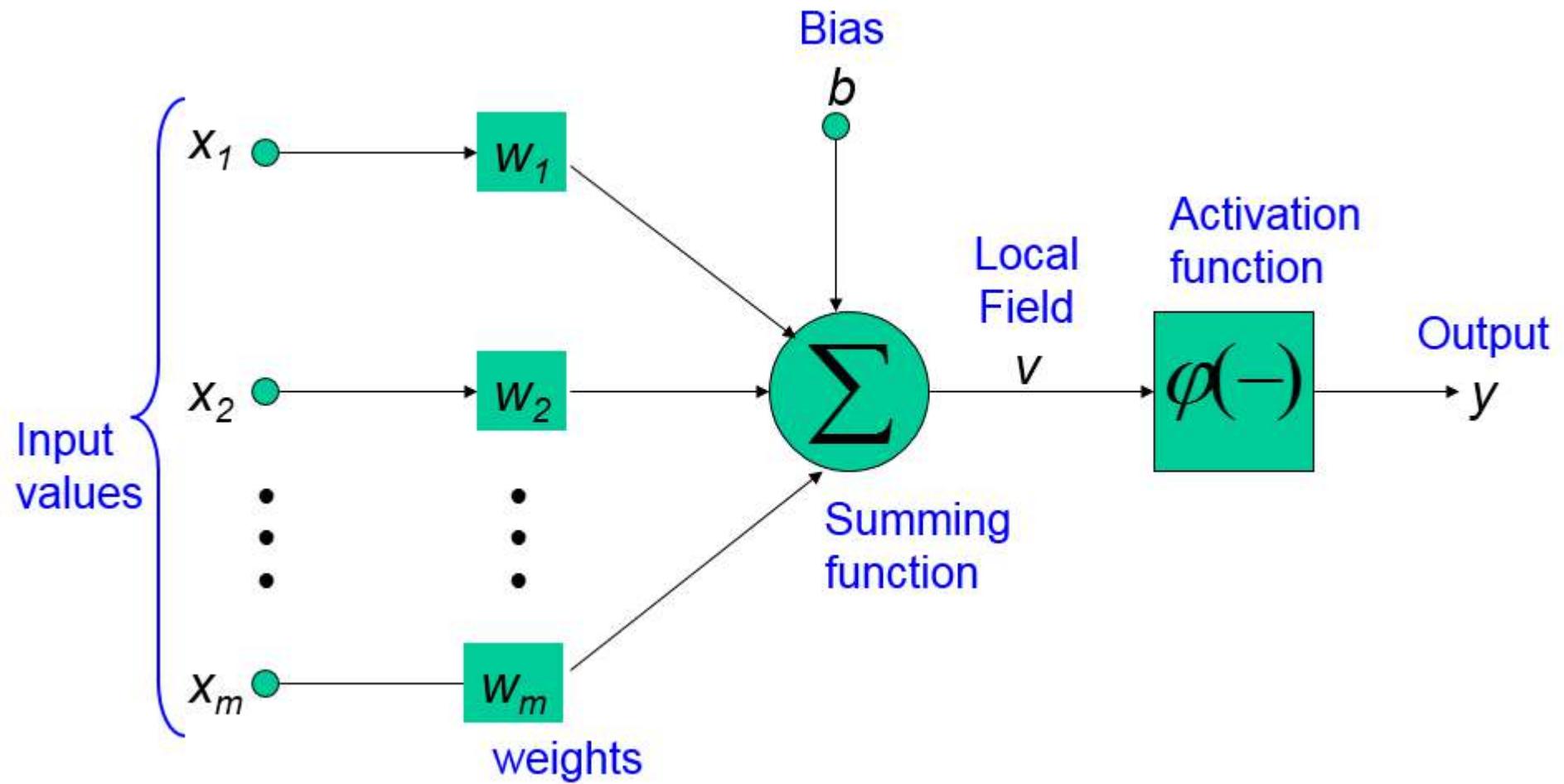
Neuron



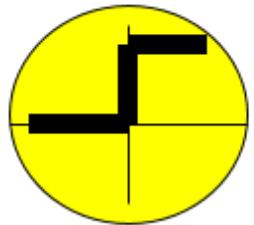
Neuron



Neuron



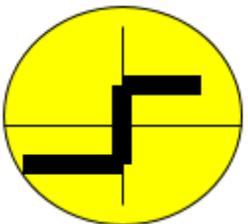
Neuron



Step function

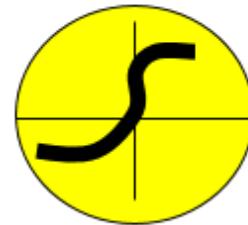
(Linear Threshold Unit)

step(x) = 1, if x >= threshold
0, if x < threshold



Sign function

$\text{sign}(x) = +1, \text{ if } x \geq 0$
 $-1, \text{ if } x < 0$

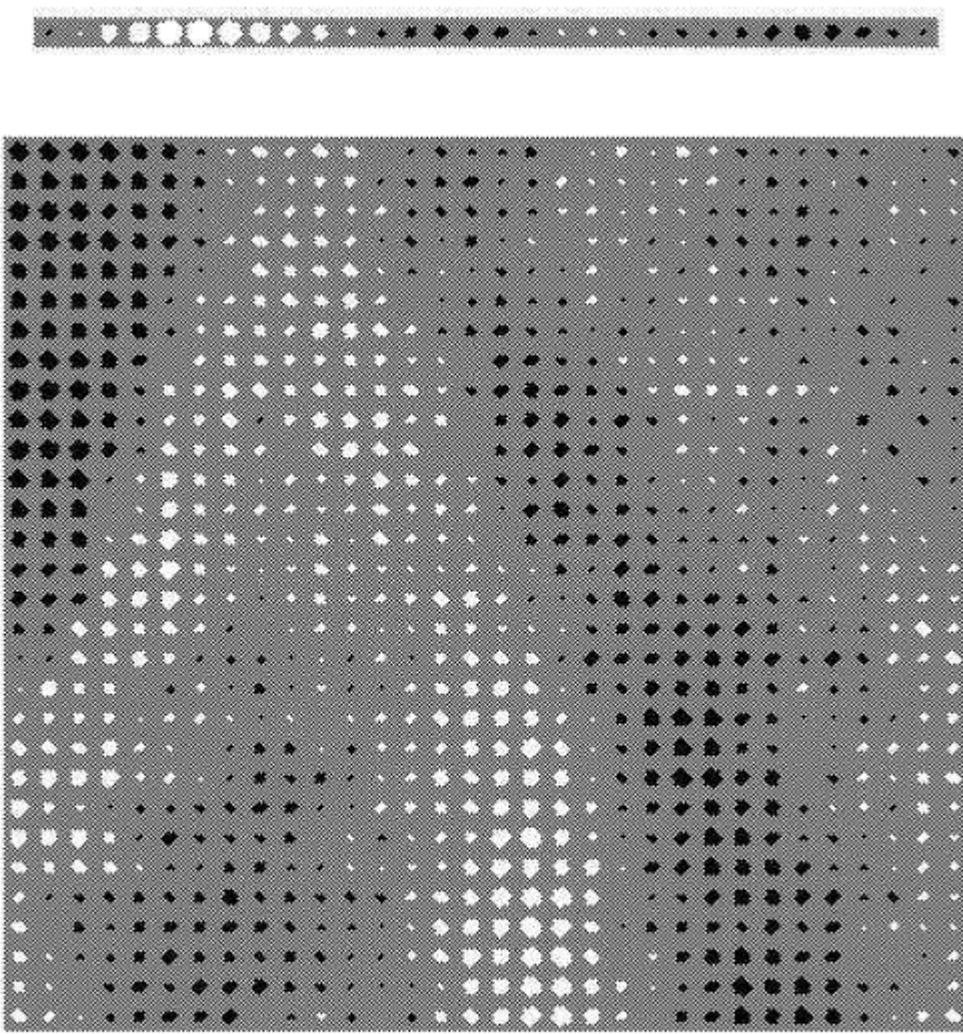
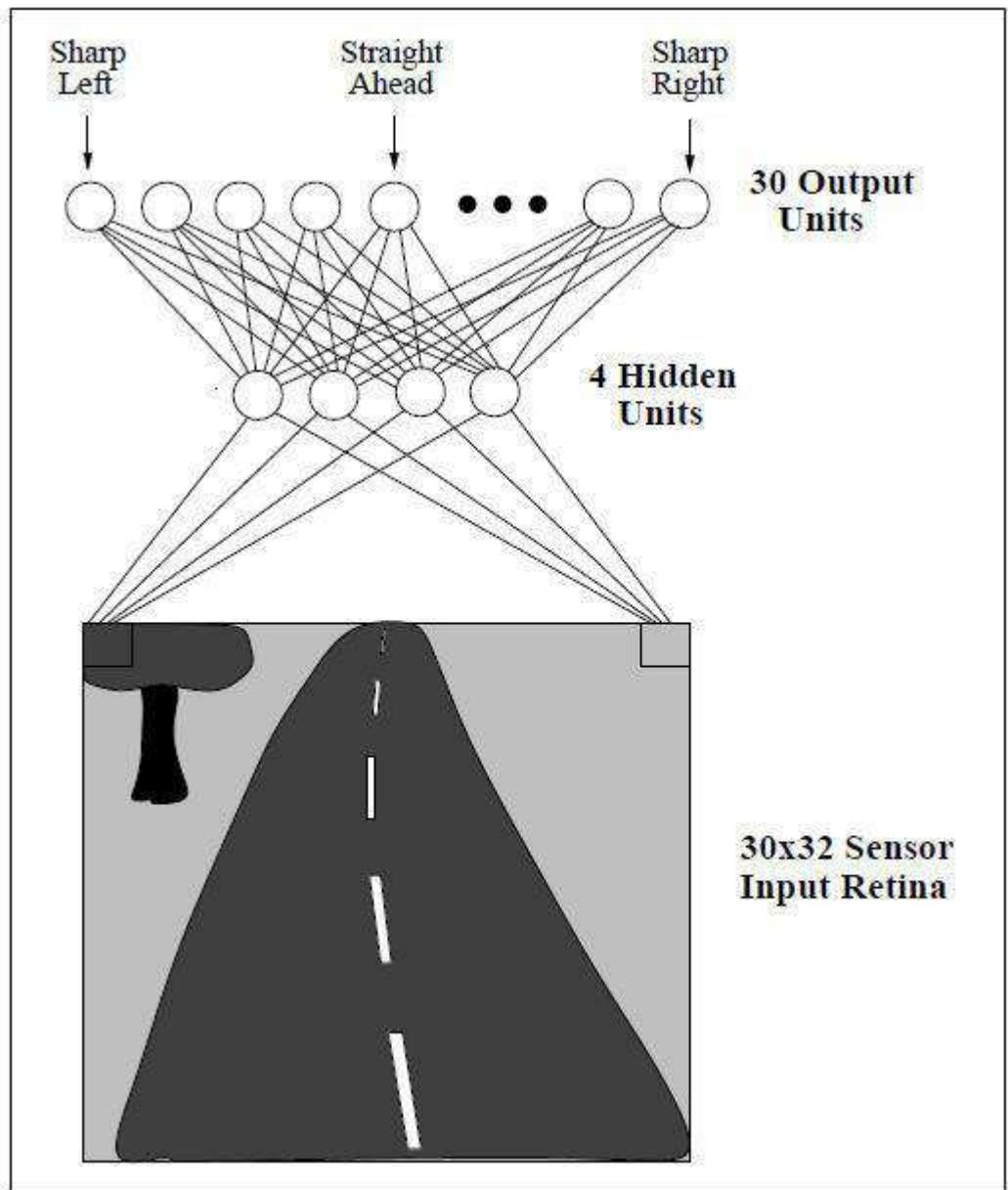


Sigmoid function

$\text{sigmoid}(x) = 1/(1+e^{-x})$

NEURAL NETWORK REPRESENTATIONS





- A prototypical example of ANN learning is provided by Pomerleau's (1993) system ALVINN, which uses a learned ANN to steer an autonomous vehicle driving at normal speeds on public highways.
- The input to the neural network is a 30x32 grid of pixel intensities obtained from a forward-pointed camera mounted on the vehicle.
- The network output is the direction in which the vehicle is steered.

- Figure illustrates the neural network representation.
- The network is shown on the left side of the figure, with the input camera image depicted below it.
- Each node (i.e., circle) in the network diagram corresponds to the output of a single network *unit*, and the lines entering the node from below are its *inputs*.
- There are four units that receive inputs directly from all of the 30×32 pixels in the image. These are called "*hidden*" units because their output is available only within the network and is not available as part of the global network output. Each of these four hidden units computes a single real-valued output based on a weighted combination of its 960 inputs
- These hidden unit outputs are then used as inputs to a second layer of 30 "output" units.
- Each output unit corresponds to a particular steering direction, and the output values of these units determine which steering direction is recommended most strongly.

- The diagrams on the right side of the figure depict the learned weight values associated with one of the four hidden units in this ANN.
- The large matrix of black and white boxes on the lower right depicts the weights from the 30×32 pixel inputs into the hidden unit. Here, a white box indicates a positive weight, a black box a negative weight, and the size of the box indicates the weight magnitude.
- The smaller rectangular diagram directly above the large matrix shows the weights from this hidden unit to each of the 30 output units.

APPROPRIATE PROBLEMS FOR NEURAL NETWORK LEARNING

ANN is appropriate for problems with the following characteristics :

- Instances are represented by many attribute-value pairs.
- The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes.
- The training examples may contain errors.
- Long training times are acceptable.
- Fast evaluation of the learned target function may be required
- The ability of humans to understand the learned target function is not important

Architectures of Artificial Neural Networks

An artificial neural network can be divided into three parts (layers), which are known as:

- ***Input layer***: This layer is responsible for receiving information (data), signals, features, or measurements from the external environment. These inputs are usually normalized within the limit values produced by activation functions
- ***Hidden, intermediate, or invisible layers***: These layers are composed of neurons which are responsible for extracting patterns associated with the process or system being analysed. These layers perform most of the internal processing from a network.
- ***Output layer*** : This layer is also composed of neurons, and thus is responsible for producing and presenting the final network outputs, which result from the processing performed by the neurons in the previous layers.

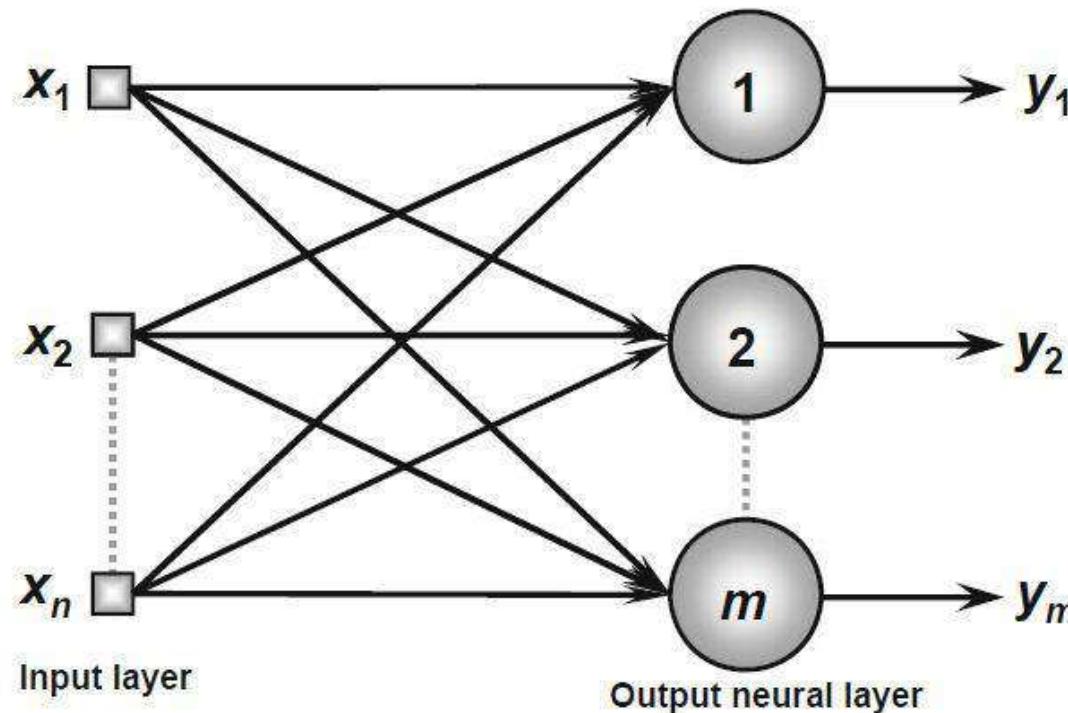
Architectures of Artificial Neural Networks

The main architectures of artificial neural networks, considering the neuron disposition, how they are interconnected and how its layers are composed, can be divided as follows:

1. Single-layer feedforward network
2. Multi-layer feedforward networks
3. Recurrent or Feedback networks
4. Mesh networks

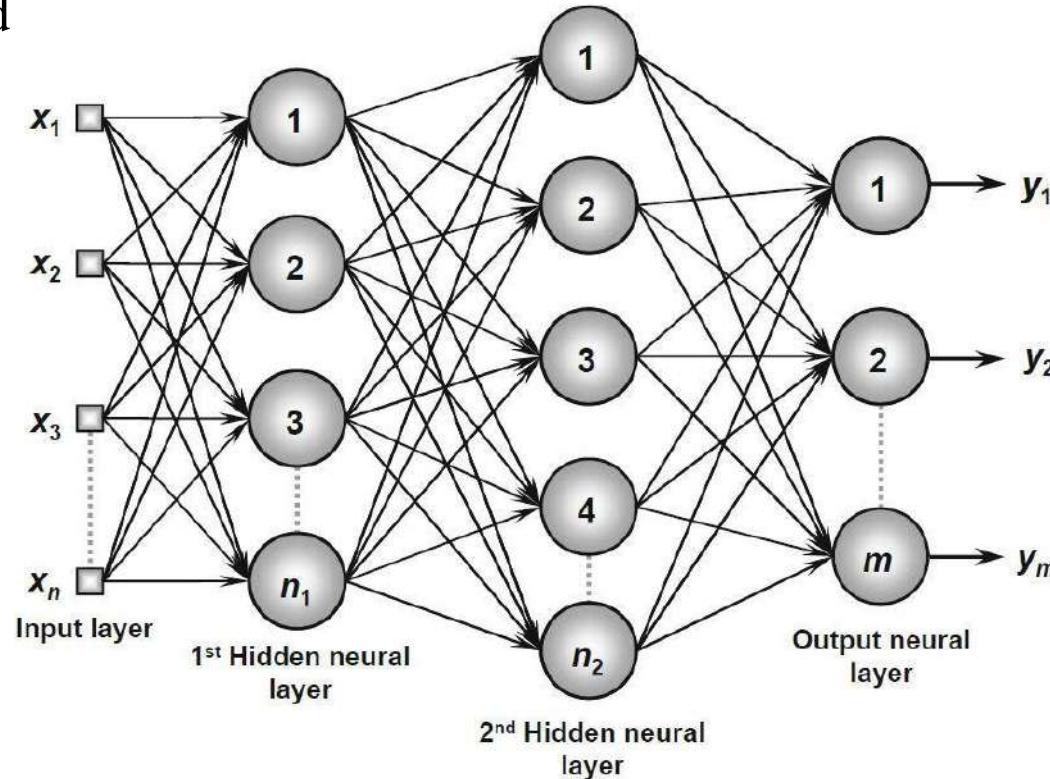
Single-Layer Feedforward Architecture

- This artificial neural network has just one input layer and a single neural layer, which is also the output layer.
- Figure illustrates a simple-layer feedforward network composed of n inputs and m outputs.
- The information always flows in a single direction (thus, unidirectional), which is from the input layer to the output layer



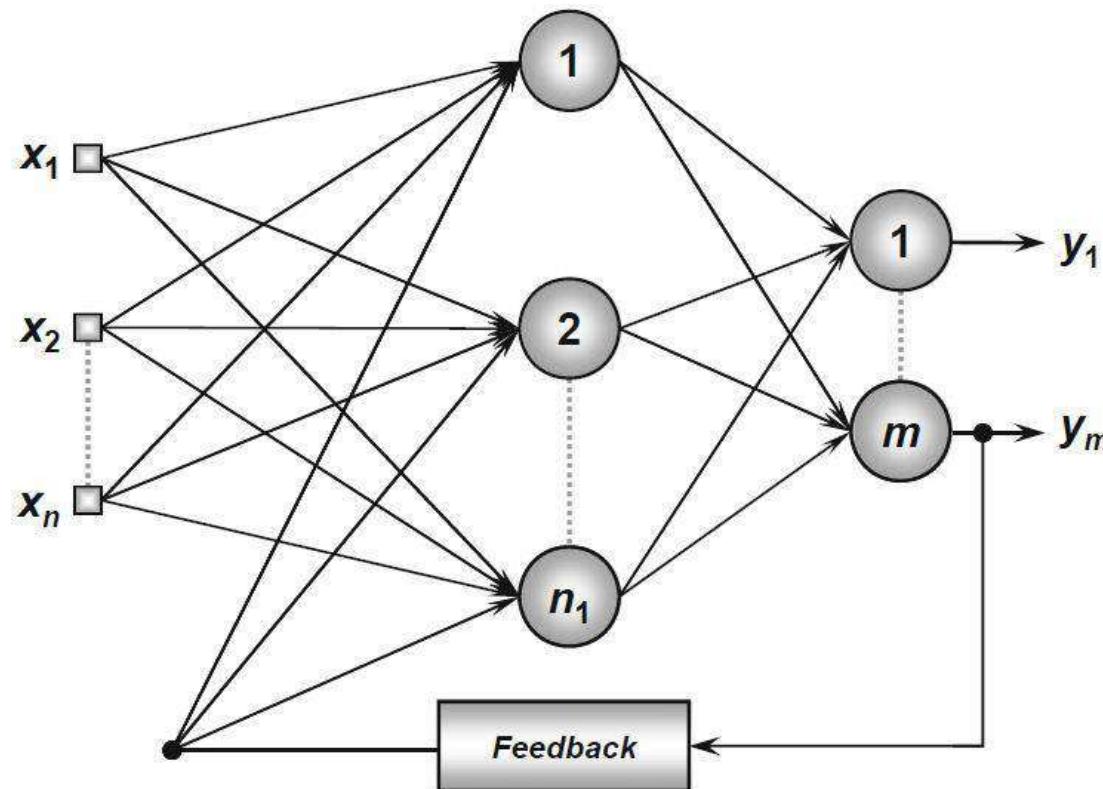
Multi-Layer Feedforward Architecture

- This artificial neural feedforward networks with multiple layers are composed of one or more hidden neural layers.
- Figure shows a feedforward network with multiple layers composed of one input layer with n sample signals, two hidden neural layers consisting of n_1 and n_2 neurons respectively, and, finally, one output neural layer composed of m neurons representing the respective output values of the problem being analyzed



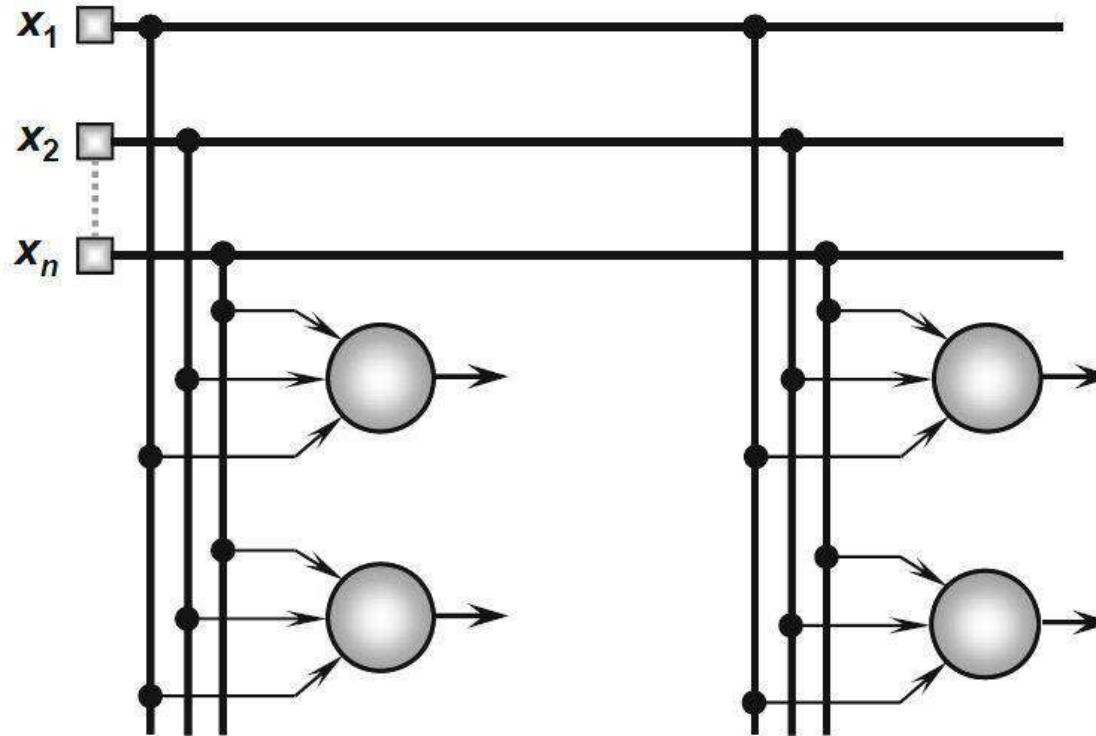
Recurrent or Feedback Architecture

- In these networks, the outputs of the neurons are used as feedback inputs for other neurons.
- Figure illustrates an example of a Perceptron network with feedback, where one of its output signals is fed back to the middle layer.



Mesh Architectures

- The main features of networks with mesh structures reside in considering the spatial arrangement of neurons for pattern extraction purposes, that is, the spatial localization of the neurons is directly related to the process of adjusting their synaptic weights and thresholds.
- Figure illustrates an example of the Kohonen network where its neurons are arranged within a two-dimensional space



PERCEPTRONS

- Perceptron is a single layer neural network.
- A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise
- Given inputs x_1 through x_n , the output $\mathbf{O}(x_1, \dots, x_n)$ computed by the perceptron is

$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \dots + w_nx_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

- where each w_i is a real-valued constant, or weight, that determines the contribution of input x_i to the perceptron output.
- $-w_0$ is a threshold that the weighted combination of inputs $w_1x_1 + \dots + w_nx_n$ must surpass in order for the perceptron to output a 1.

Sometimes, the perceptron function is written as,

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

Where,

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

Learning a perceptron involves choosing values for the weights w_0, \dots, w_n . Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors

$$H = \{\vec{w} \mid \vec{w} \in \Re^{(n+1)}\}$$

Why do we need Weights and Bias?

Weights shows the strength of the particular node.

A *bias* value allows you to shift the activation function curve up or down

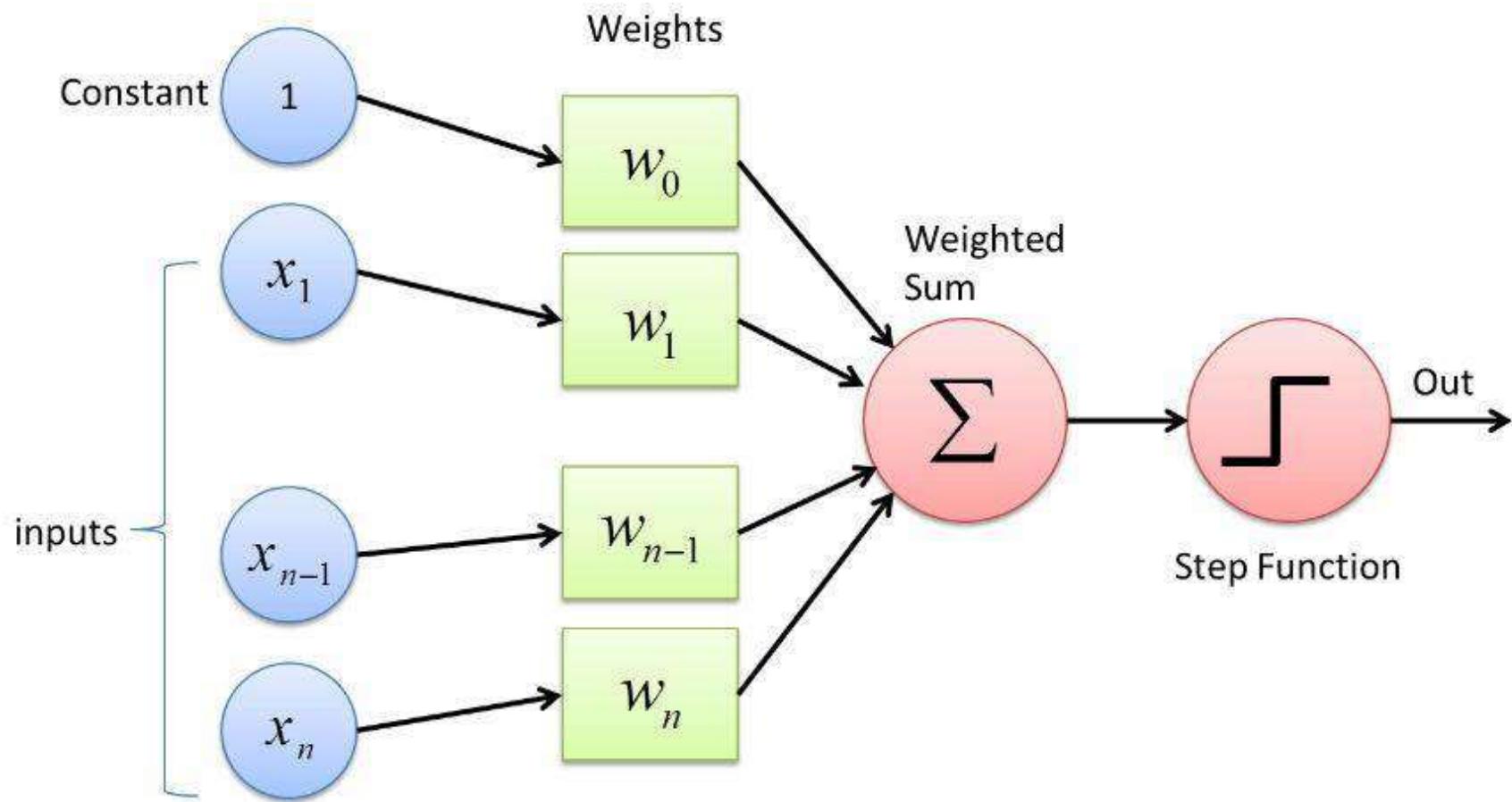
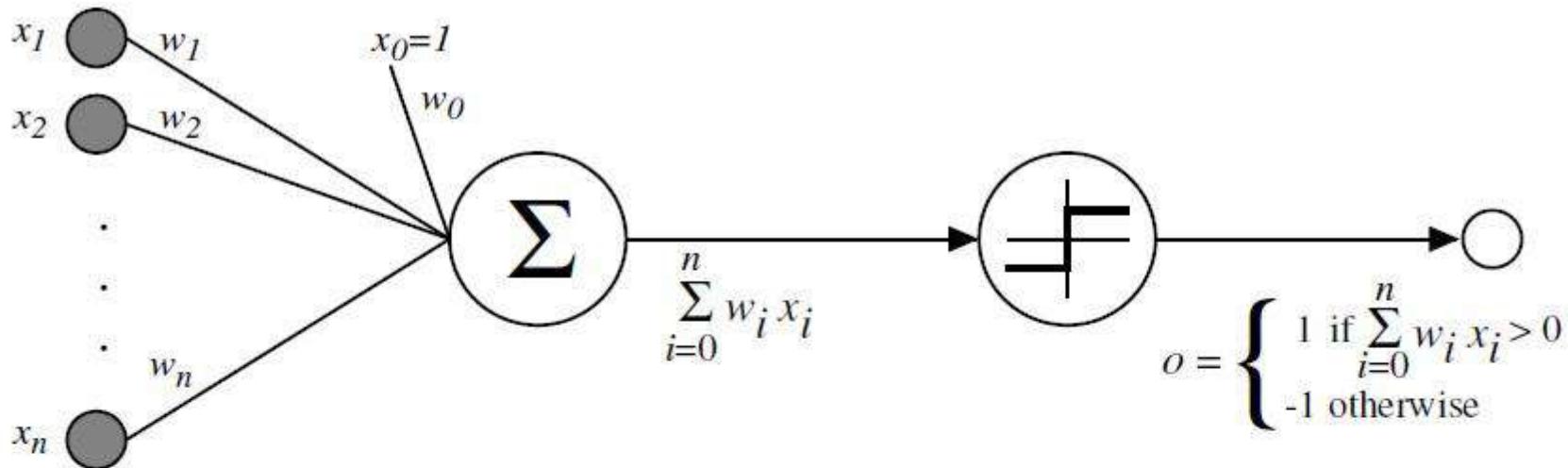


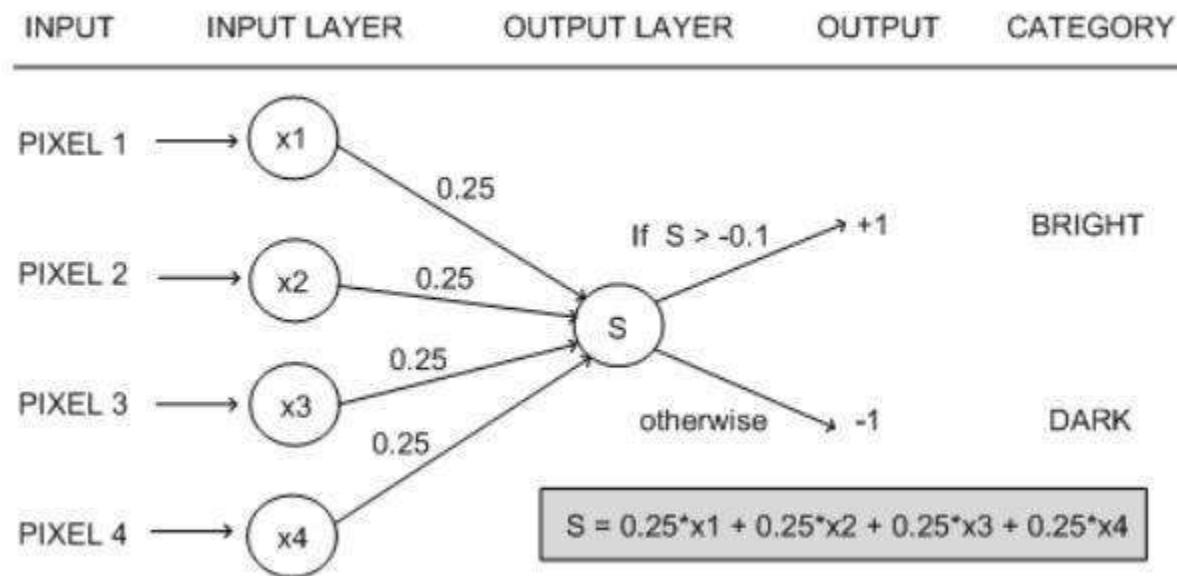
Fig : Perceptron



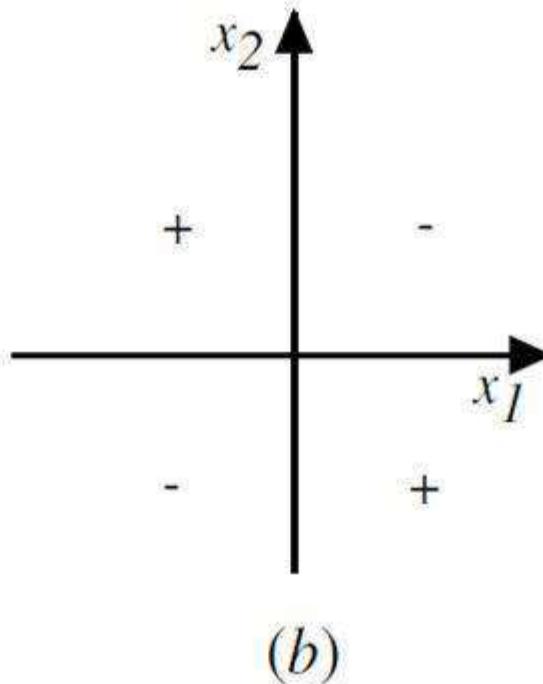
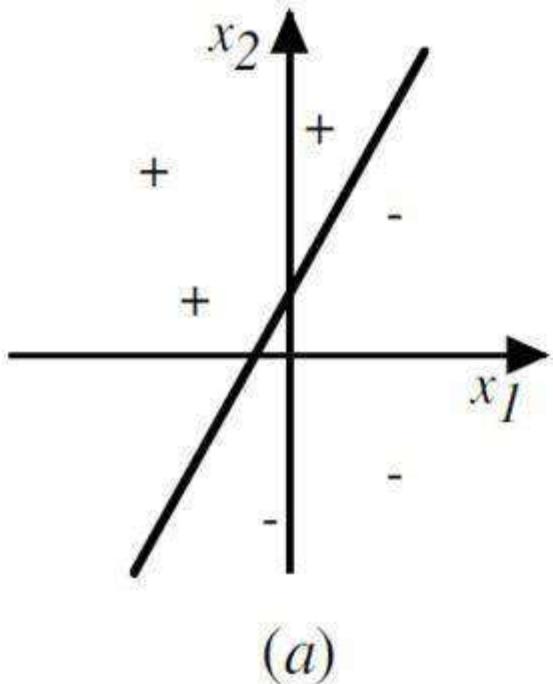
$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \dots + w_nx_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$



Representational Power of Perceptrons



- * The perceptron can be viewed as representing a hyperplane decision surface in the n-dimensional space of instances.
- * The perceptron outputs a 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side

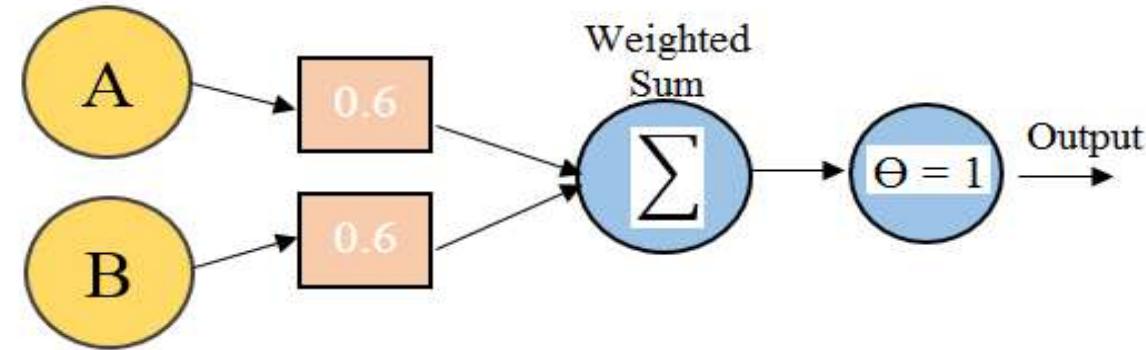
Figure : The decision surface represented by a two-input perceptron.

(a) A set of training examples and the decision surface of a perceptron that classifies them correctly. (b) A set of training examples that is not linearly separable.

x_1 and x_2 are the Perceptron inputs. Positive examples are indicated by "+", negative by "-".

A single perceptron can be used to represent many Boolean functions
AND function

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1



- If $A=0 \ \& \ B=0 \rightarrow 0*0.6 + 0*0.6 = 0$.
This is not greater than the threshold of 1, so the output = 0.
- If $A=0 \ \& \ B=1 \rightarrow 0*0.6 + 1*0.6 = 0.6$.
This is not greater than the threshold, so the output = 0.
- If $A=1 \ \& \ B=0 \rightarrow 1*0.6 + 0*0.6 = 0.6$.
This is not greater than the threshold, so the output = 0.
- If $A=1 \ \& \ B=1 \rightarrow 1*0.6 + 1*0.6 = 1.2$.
This exceeds the threshold, so the output = 1.

The Perceptron Training Rule

The learning problem is to determine a weight vector that causes the perceptron to produce the correct + 1 or - 1 output for each of the given training examples.

To learn an acceptable weight vector

- Begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight w_i associated with input x_i according to the rule.

$$w_i \leftarrow w_i + \Delta w_i$$

Where,

$$\Delta w_i = \eta(t - o)x_i$$

Here,

t is the target output for the current training example

o is the output generated by the perceptron

η is a positive constant called the ***learning rate***

- The role of the ***learning rate*** is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases

Drawback: The perceptron rule finds a successful weight vector when the training examples are linearly separable, it can fail to converge if the examples are not linearly separable.

MULTILAYER NETWORKS AND THE BACKPROPAGATION ALGORITHM

Multilayer networks learned by the **BACKPROPACATION** algorithm are capable of expressing a rich variety of nonlinear decision surfaces.

A Differentiable Threshold Unit

- Sigmoid unit-a unit very much like a perceptron, but based on a smoothed, differentiable threshold function.
- The sigmoid unit first computes a linear combination of its inputs, then applies a threshold to the result. In the case of the sigmoid unit, however, the threshold output is a continuous function of its input.
- More precisely, the sigmoid unit computes its output O as

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

Where,

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

o is the sigmoid function

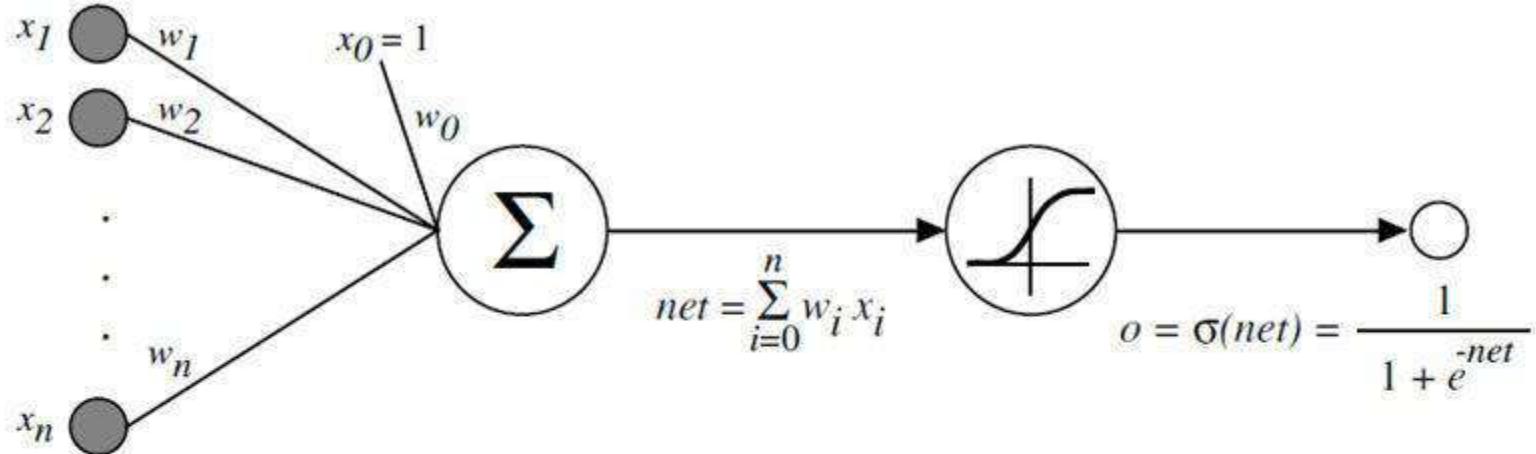


Figure: A Sigmoid Threshold Unit

$\sigma(y)$ is the sigmoid function

$$\frac{1}{1 + e^{-y}}$$

Nice property: $\frac{d\sigma(y)}{dy} = \sigma(y)(1 - \sigma(y))$

The BACKPROPAGATION Algorithm

- The BACKPROPAGATION Algorithm learns the weights for a multilayer network, given a network with a fixed set of units and interconnections. It employs gradient descent to attempt to minimize the squared error between the network output values and the target values for these outputs.
- In BACKPROPAGATION algorithm, we consider networks with multiple output units rather than single units as before, so we redefine E to sum the errors over all of the network output units.

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} \sum_{k \in \text{outputs}} (t_{kd} - o_{kd})^2 \quad \dots\dots \text{equ. (1)}$$

where,

- **outputs** - is the set of output units in the network
- t_{kd} and O_{kd} - the target and output values associated with the k^{th} output unit
- d - training example

|BACKPROPAGATION (*training_example*, η , n_{in} , n_{out} , n_{hidden})

Each training example is a pair of the form (\vec{x}, \vec{t}) , where (\vec{x}) is the vector of network input values, (\vec{t}) and is the vector of target network output values.

η is the learning rate (e.g., .05). n_{in} is the number of network inputs, n_{hidden} the number of units in the hidden layer, and n_{out} the number of output units.

The input from unit i into unit j is denoted x_{ji} and the weight from unit i to unit j is denoted w_{ji}

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
 - For each (\vec{x}, \vec{t}) , in training examples, Do

Propagate the input forward through the network:

1. Input the instance \vec{x} , to the network and compute the output o_u of every unit u in the network.

Propagate the errors backward through the network:

2. For each network output unit k , calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h , calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k} \delta_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

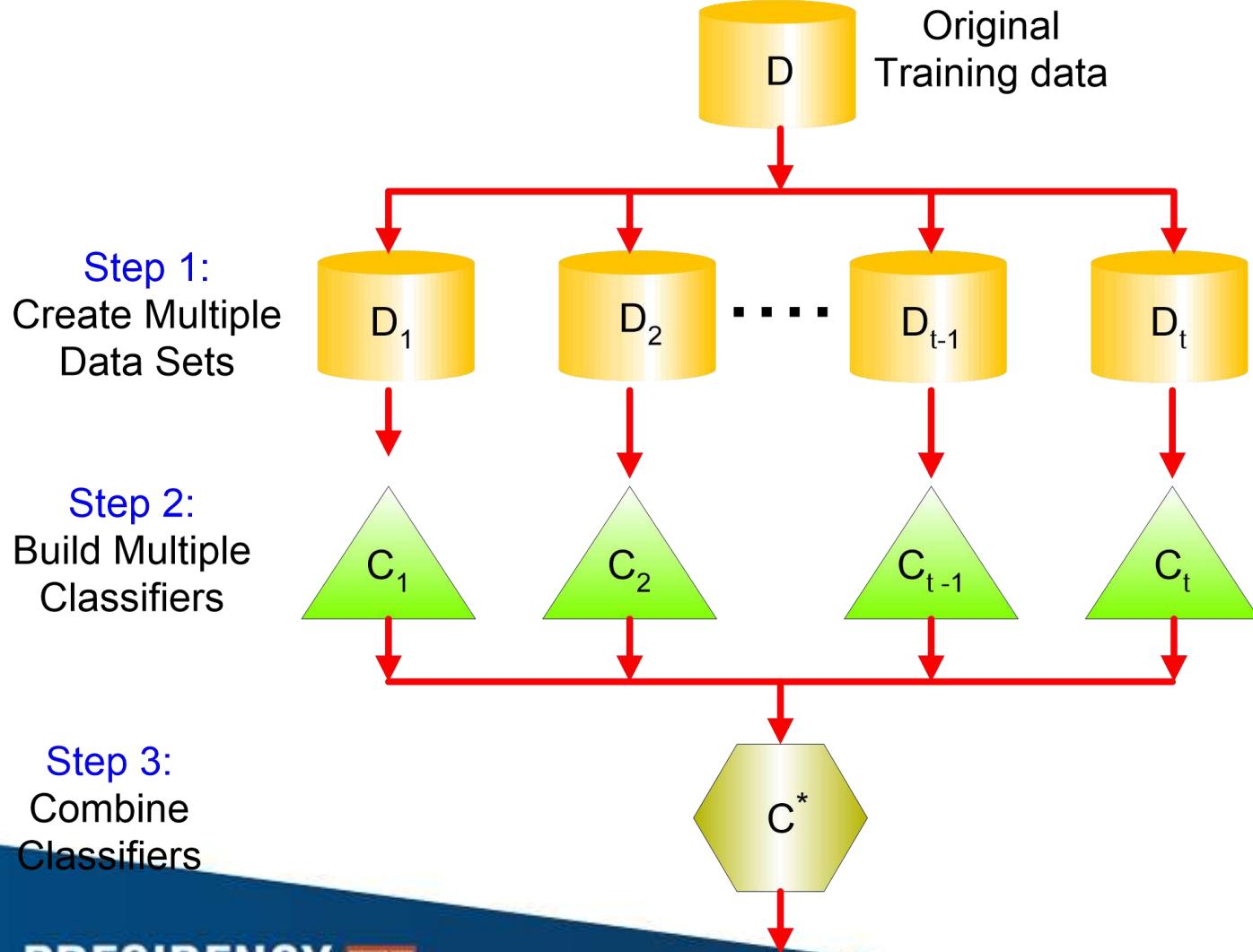
Where

$$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

Ensemble Methods

- Construct a set of classifiers from the training data
- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers
- Improves the classification accuracy
- Predicted output of the base classifiers is combined by majority voting
- Build different experts and let them vote.

General Idea



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
 - Bagging }
 - Boosting }
 - **By choosing a subset of features**
 - ◆ Each training set has a subset of features chosen randomly. Base classifiers
 - ◆ Used when data set has more redundant features
 - ◆ Ex: Random forest....decision tree is base classifier
 - **By manipulating the class labels**
 - Class labels are partitioned randomly into two disjoint subsets A_0 and A_1

Examples of Ensemble Methods by manipulating the class labels

- Instances $\in A_0$ labeled as class 0
- Instances $\in A_1$ labeled as class 1
- Relabeled examples train base classifier
- Class relabeling, model building are repeated for several iterations.
- Each iteration builds a base classifier.
- ∴ an ensemble of base classifiers is obtained.
- When a test record is presented, each base classifier C_i gives its predicted output.
- if $C_i = 0$... All classes $\in A_0$ receives a vote
- if $C_i = 1$... All classes $\in A_1$ receives a vote



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



40

YEARS
OF ACADEMIC
FREEDOM

Examples of Ensemble Methods

- repeated for each C_i
- Votes are tallied
- Test record is assigned to the class with highest vote.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



General Procedure for ensemble methods

1. Let D The original training data, k ... no. of base classifiers, T Test data
2. For $i = 1$ to k do
 1. Create training set D_i from D
 2. Build a base classifier C_i from D
3. End for
4. For each test record $x \in T$ do
5. $C^*(x) = \text{vote}(C_1(x), C_2(x), \dots, C_k(x))$
6. Each C_i returns its class prediction.
7. End for



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Bagging

- Sampling with replacement
- Bootstrap samples D_i , each with 63% of original data

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Each sample has probability $(1 - 1/n)^n$ of being selected in each D_i

Example of Bagging

- Refer notes for a numerical example.
- Data Set used to construct an ensemble of bagging classifiers

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights, $1/N$
 - Unlike bagging, weights may change at the end of boosting round
 - With each boosting sample, a classifier is induced(iteratively) and is used to classify all training examples.
 - Misclassified examples are assigned more weights for the next round.

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds
- Final ensemble is an aggregate of the base classifiers got from each boosting round.

How Boosting Works?

- Weights are assigned to each training tuple.
- A series of k classifiers is iteratively learnt
- After a classifier M_i is learnt, the weights are updated to allow the subsequent classifier M_{i+1} , to pay more attention to the training tuples that were misclassified by M_i .
- The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.

Basic Idea

- Suppose there are just 5 training examples {1, 2, 3, 4, 5}
- Initially each example has 0.2 (1/5) probability, of being sampled.
- If the boosting samples for the first round are {2,4,4,3,2}, a base classifier is built from this.
- Suppose 2,3,5 are correctly predicted by this classifier and 1,4 are wrongly predicted:
 - Weight of 1,4 is increased
 - Weight of 2,3,5 is decreased.
- Second round of boosting , again 5 samples, but now 1,4 are more likely to be sampled.



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Boosting

- Is an iterative procedure.
- The distribution of training examples are adaptively changed, so that the base classifiers in the next iteration, focus more on examples that are wrongly predicted in the previous iteration.
- Boosting assigns a weight to each example.
- Weights are adaptively changed at the end of each boosting round.
- Weights assigned to the training examples are used in the following ways:-

Boosting

1. To draw a set of bootstrap samples from the original data
2. Can be used by the base classifier to learn a model that is biased towards higher weight examples.

Steps:-

1. Initially wt of all examples are same $1/N$.
2. A sample is drawn as per the sampling distbn of the training examples to get a new training set.
3. A classifier is induced from this training set.

Boosting

4. All examples of the original data are classified using this classifier.
5. Wrongly classified examples ... increase in weight

Correctly classified examples decrease in weight.

So, wrongly classified examples will be focussed more in subsequent iterations.

6. Repeat steps 2 to 5 for k times($k = \text{no. of base classifiers}$)



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Boosting

7. As boosting round proceeds, wrongly classified examples become more prevalent.
8. Final ensemble is got by aggregating base classifiers got from each boosting round.

Several implementations of the boosting algorithm have been developed. They all differ in terms of

- 1) How the weights of the examples are updated.
- 2) How the predictions of the base classifiers are combined.

Example: AdaBoost

1. Let $\{(x_j, y_j) \mid j = 1, 2, 3, 4, \dots, N\}$ is a set of N training examples.

- Base classifiers: C_1, C_2, \dots, C_T

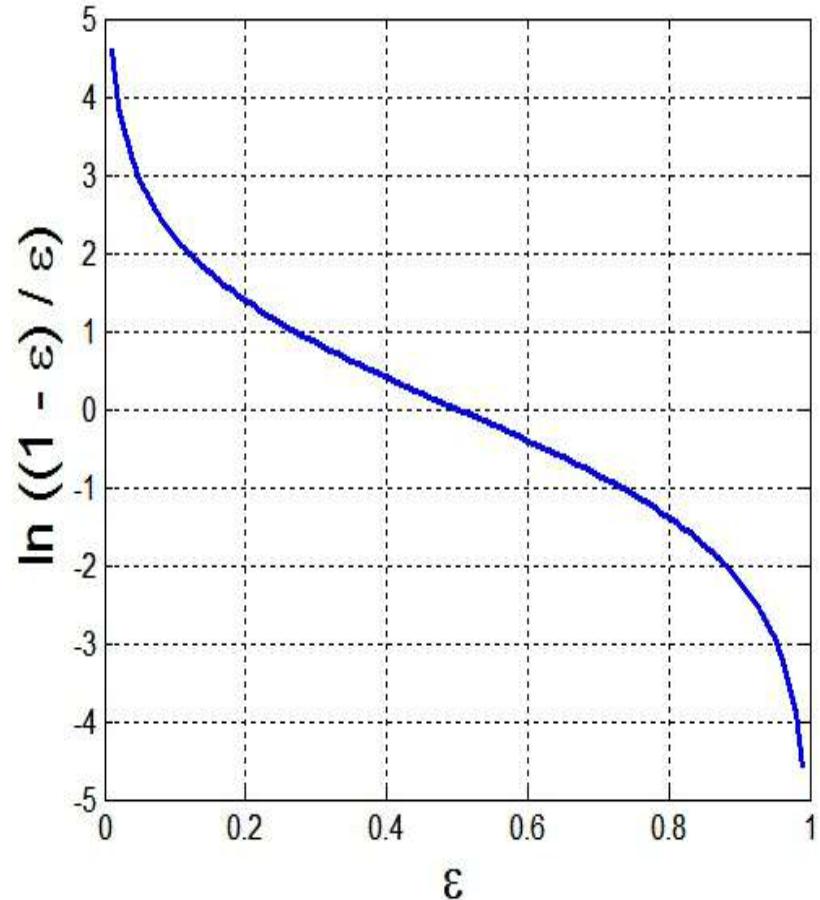
2. Error rate of a base classifier:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

where $\delta(P) = 1$ if P is true
= 0 otherwise

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$



**PRESIDENCY
UNIVERSITY**

Private University Estd. in Karnataka State by Act No. 41 of 2013



Example: AdaBoost

3. α_i is also used to update the weight of training examples.

Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

Z_j is used to ensure that $\sum w_i^{j+1} = 1$

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated

Ada Boost

4. Instead of majority voting, the prediction made by each C_j is weighted according to α_j . The weighted average of this is the final ensemble.

Classification

$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

[Refer book for the algorithm
Refer transparencies for a numerical example]

Class Imbalance Problem

- In many real time data sets the class distbn is imbalanced.
- Correct classfn of the rare class is of great value, than the correct classfn of the majority class.
- ∵ accuracy can't be used to evaluate the classifiers.
- ∵ other evaluation metrics based on ROC are used.

Alternative metrics:-

Rare class +ve class

Majority class..... -ve class.

TPR, TNR, FPR, FNR, Recall, Precision and F_1 measure

Multi-class problem

- Binary classifiers can be extended to handle multiclass problems.
- Ex:- $Y = \{y_1, y_2, \dots, y_k\}$ where $y_i \in Y$ is the set of classes of the input data.
- **Approach 1:- (one-against-rest) (1-r)**
- Multiclass problem – K binary problems.
- K iterations K binary classifiers are built.
- In iteration i,
 - $y_i \in Y$ are taken as +ve class examples .
 - all other classes are taken as -ve examples.

Multi-class problem

- Approach 2:- (one-against-one) (1 – 1)
- $\frac{k(k-1)}{2}$ binary classifiers are built.
- Each classifier distinguishes a pair of classes (y_i, y_j)
- Instances $\notin (y_i, y_j)$ are ignored.

Classifying a test instance:-

- In both (1-r) and (1-1), the predictions made by all base classifiers are combined by majority voting.

Multi-class problem

- Ex:- $Y = \{y_1, y_2, y_3, y_4\}$. Suppose a test instance is classified as (+, -, -, -) by (1-r)approach. What is its predicted class?

Base Classifier	+ve example	-ve examples	vote
B_1	y_1	$y_2 y_3 y_4$	y_1
B_2	y_2	$y_1 y_3 y_4$	$y_1 y_3 y_4$
B_3	y_3	$y_1 y_2 y_4$	$y_1 y_2 y_4$
B_4	y_4	$y_1 y_2 y_3$	$y_1 y_2 y_3$



PRESIDENCY
UNIVERSITY



Private University Estd. in Karnataka State by Act No. 41 of 2013

• In majority voting, class of test instance is y_1 .

Using 1-1 approach:-

- If the test instance is classified as shown:

Binary pair of classes	+ : y_1	+ : y_1	+ : y_1	+ : y_2	+ : y_2	+ : y_3
	- : y_2	- : y_3	- : y_4	- : y_3	- : y_4	- : y_4
Classfn	+	+	-	+	-	+

- Votes:- $y_1 = 2$, $y_2 = 1$, $y_3 = 1$, $y_4 = 2$
- \therefore test record is classified as y_1 or y_4 depending on the tie breaking procedure.

Error-correcting output coding:-

- Using Hamming distance – distance of two bit strings is the no. of bits that differ.

Class	Code word
y_1	1 1 1 1 1 1 1
y_2	0 0 0 0 1 1 1
y_3	0 0 1 1 0 0 1
y_4	0 1 0 1 0 1 0

- Each class is encoded by a 7-bit code-word.
- Each bit b_i of the code word is used to train a binary classifier.
- If a test instance is classified as {0,1,1,1,1,1,1} by all binary classifiers.

Error-correcting output coding:-

- Hamming distance between test code word and y_1 is 1, with remaining classes is 3.
- \therefore test instance is classified as y_1 .

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1

$$x \leq 0.65 \Rightarrow y = 1$$

$$x > 0.65 \Rightarrow y = -1$$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$$x \leq 0.3 \Rightarrow y = 1$$

$$x > 0.3 \Rightarrow y = -1$$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$$x \leq 0.35 \Rightarrow y = 1$$

$$x > 0.35 \Rightarrow y = -1$$

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$$x \leq 0.75 \Rightarrow y = -1$$

$$x > 0.75 \Rightarrow y = 1$$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$$x \leq 0.05 \Rightarrow y = -1$$

$$x > 0.05 \Rightarrow y = 1$$

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True Class	1	1	1	-1	-1	-1	-1	1	1	1

Ex: Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

Weights of the training records:

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

Decision Stump is the base classifier.

Round	Split point	left class	Right class	α
1	0.75	-1	1	1.738
2	0.05	-1	1	2.7784
3	0.3	1	-1	4.1195

Details of Round 1:-

Original training set:-

x	y	wt	P(x)
0.1	1	0.1	-1 x
0.2	1	0.1	-1 x
0.3	1	0.1	-1 x
0.4	-1	0.1	-1
0.5	-1	0.1	-1
0.6	-1	0.1	-1
0.7	-1	0.1	-1
0.8	+1	0.1	1
0.9	1	0.1	1
1	1	0.1	1

$\therefore 3$ wrong predictions

$$\epsilon_i = \frac{1}{10} [3 \times 0.1]$$

$$= 0.03$$

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

$$= \frac{1}{2} \ln (32.33)$$

$$= \frac{1}{2} \times 3.48$$

$$= 1.738$$

wt updation:

$$\lambda = 1.738$$

<u>x</u>	<u>WP/CP</u>	<u>old wt</u>	<u>Newwt:</u>	$\exp(-\lambda x) =$ <u>Normal wt</u>
0.1	x	0.1	0.1×5.69	
0.2	x	0.1	0.1×5.69	0.569
0.3	x	0.1	0.1×5.69	
0.4	✓	0.1	0.1×0.176	
0.5	✓	0.1	0.1×0.176	
0.6	✓	0.1	0.1×0.176	0.0176
0.7	✓	0.1	"	0.02
0.8	✓	0.1	"	
0.9	✓	0.1	"	
1	✓	0.1	"	

<u>λ</u>	<u>$\exp(\lambda)$</u>	<u>$\exp(-\lambda)$</u>
1.738	5.69	0.17587

Normalizing the wts:

Sum of the wts: $3 \times 0.569 + 7 \times 0.0176$

$$= 1.707 + 0.1232$$

$$= 1.8302$$

Divide each wt by the sum of wts.

Combining classifiers

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	0.97	0.97	0.97	0.97
Sign	1	1	1	-1	-1	-1	-1	1	1	1