

# MACHINE LEARNING (UE22CS352A)



## Introduction to Machine Learning

---

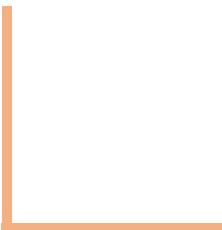
**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

# MACHINE LEARNING

---

## Introduction to Machine Learning

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.



### **Unit1: Introduction, Performance Metrics, Classification with Decision trees and KNN:**

Introduction to ML: Machine Learning and its Models, Concept Learning: Concepts of hypotheses, Version space, inductive bias, Performance metrics-accuracy, precision, recall, sensitivity, specificity, AUC, ROC, Bias Variance decomposition. Decision Trees-Basic algorithm(ID3) for classification, Decision boundary for decision trees(x-y axis), Hypothesis search and Inductive bias, Issues in Decision Tree Learning – Over fitting, Solutions to over fitting, dealing with continuous values, Logistic Regression, Instance-based learning: k-nearest neighbor learning (Classification & Regression), Decision boundary for KNN.

**14 Hours**

### **Unit2: Ensemble Models, Supervised Learning with ANN and SVM:**

Combining weak learners, Improving performance with Gradient Boost, Random Forest, Artificial Neural networks: Introduction, Perceptions, Multi-layer networks and back-propagation, Activation functions (Step, Sigmoid, Tanh, ReLU), Various Optimizers(GD, SGD, Momentum-based, Adagrad, Adam,), Support Vector Machines – margin and maximization, SVM - The primal problem, the Lagrangian dual, SVM –Solution to the Lagrangian dual Hard Margin( Classification ONLY)

**Self Learning :**Soft Margin( Classification ONLY), Kernel functions :Linear, polynomial (Derivation only for linear function).

**14 Hours**

### **Unit3: Stochastic Models, unsupervised learning, Dimensionality Reduction :**

Bayesian Learning –Bayes theorem, Maximum likelihood, Bayes optimal classifier, Naïve Bayes classifier, Expectation maximization and Gaussian Mixture Models, Hidden Markov Models, Hierarchical vs. non-hierarchical clustering, Agglomerative and divisive clustering, K-means clustering, Bisecting K-means, K-Means as special case of Expectation Maximization.

**14 Hours**

### **Unit4 : Introduction to Deep Learning techniques and Transformers**

Introduction to Deep Learning, Introduction to Convolution operation and Convolution Neural Network. (Parameter calculation, Max pooling, Avg pooling). Introduction to Recurrent Neural Network, Vanishing and exploding gradient, Basic GAN and its architecture, Basic architecture of Transformers- (Encoder-Decoder perspective with basics of attention mechanism

**Self learning :** Variants of RNN: LSTM, GRU (mention of gates).

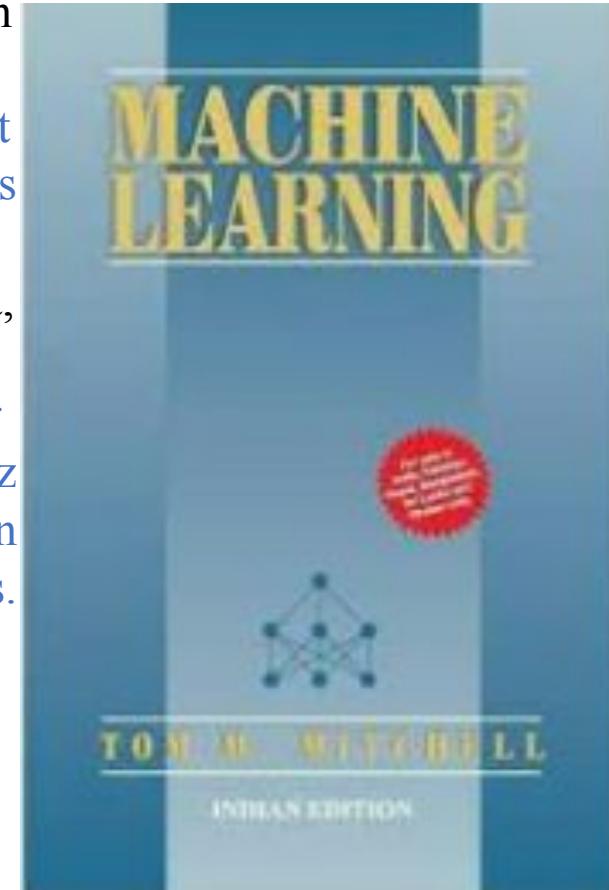
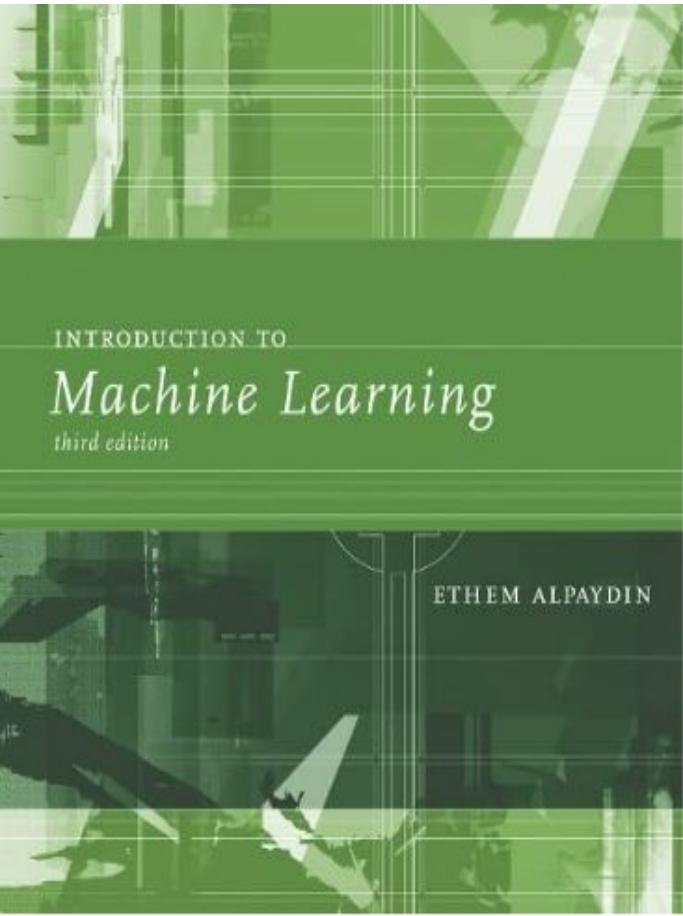
**14 Hours**

### Text Books(s):

- 1: "Introduction to Machine Learning", Ethem Alpaydin, Third edition, MIT Press Ltd.
- 2: "Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville, The MIT Press, 2017.

### Reference Books:-

- 1: "Machine Learning", Tom Mitchell, McGraw Hill Education (India),2013.
- 2: "Machine Learning: The Art and Science of Algorithms that Make Sense of Data", Peter Flach, Cambridge University Press (2012).
- 3: "Generative Deep Learning, David Foster, O'reilly media, 2nd Edition, 2023, ISBN : 9781098134181.
- 4: "**Attention is all you need**", Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Łukasz, and Polosukhin Illia. 2017, In Advances in Neural Information Processing Systems. 5998–6008



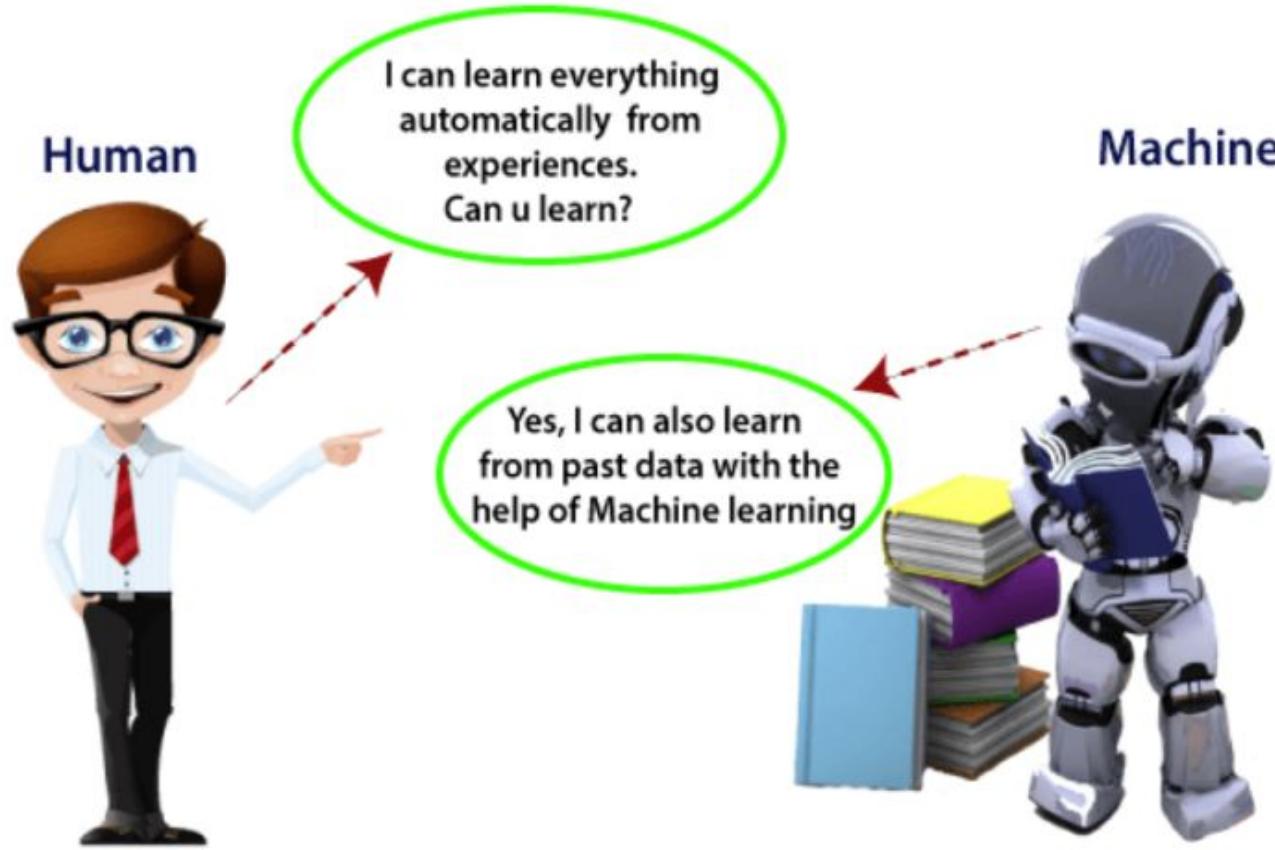
### **Lab/ Hands-on:**

Applications using

- Decision Trees.
- ANN
- SVM
- Naïve Bayes
- HMM
- Clustering
- CNN model.

# MACHINE LEARNING

## INTRODUCTION



# MACHINE LEARNING

## INTRODUCTION

### What is a Machine ?

- In the context of Machine Learning, a machine refers to a **computing system** that is capable of learning and improving its performance on a **specific task** without being explicitly programmed.
- These machines are designed to **process large amounts of data**, extract patterns, and make predictions or decisions based on the learned information.
- The term "machine" in this context typically refers to a **computer or computational device equipped with algorithms** and models that enable it to perform learning tasks autonomously.



Digital Computer Processing System

# MACHINE LEARNING

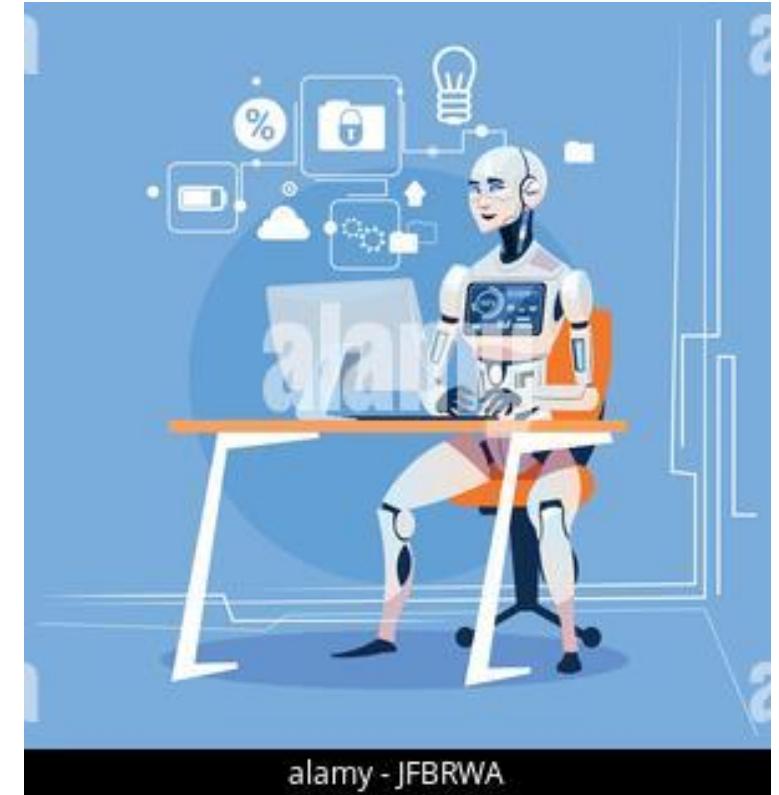
## INTRODUCTION

### Why Learn?

Machine learning is programming computers to optimize a performance criterion using example data or past experience.

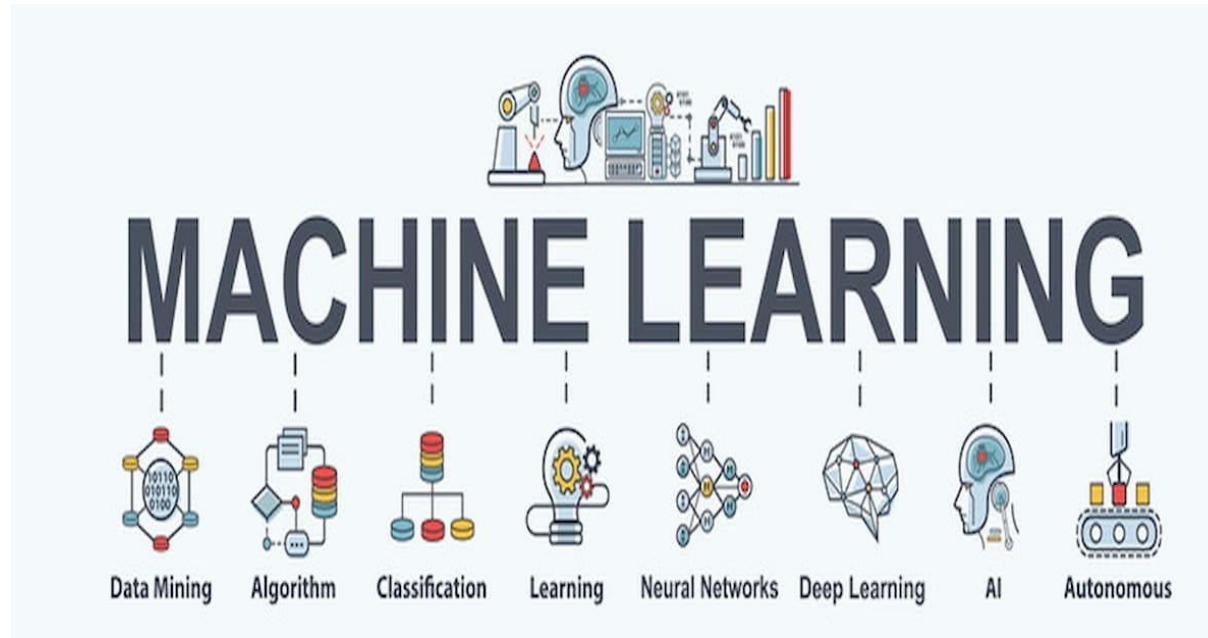
Learning is used when:

- 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)
- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Build a model that is a good and useful approximation to the data.



## What is Machine Learning

Machine learning (ML) is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data.



# MACHINE LEARNING

## Applications of Machine Learning

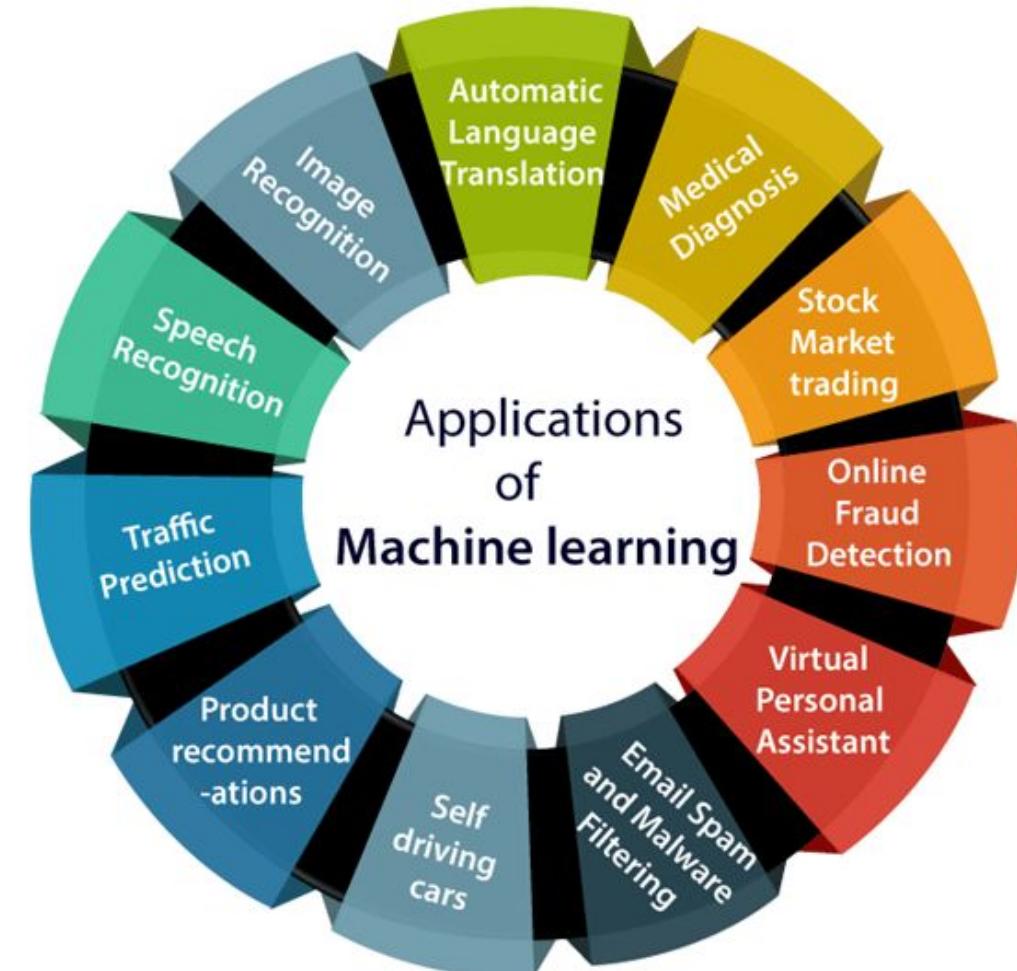
**Healthcare:** Diagnosing diseases, personalized medicine, predicting patient outcomes.

**Finance:** Fraud detection, algorithmic trading, credit scoring.

**Retail:** Personalized recommendations, inventory management, demand forecasting.

**Transportation:** Autonomous vehicles, traffic prediction, route optimization.

**Natural Language Processing (NLP):** Language translation, sentiment analysis, chatbots.



### Machine Intelligence

Machine learning is defined as systems that enable a computer system to learn from inputs.



Artificial intelligence is composed of systems that allow computers to imitate human cognitive processes

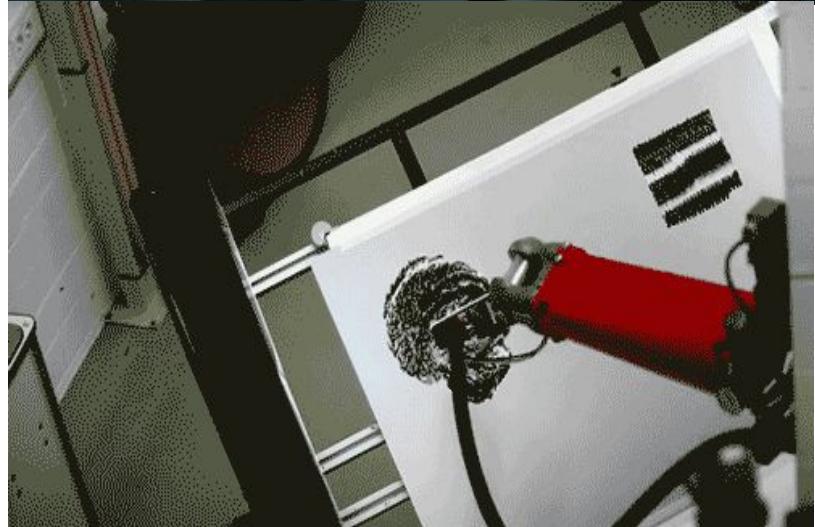
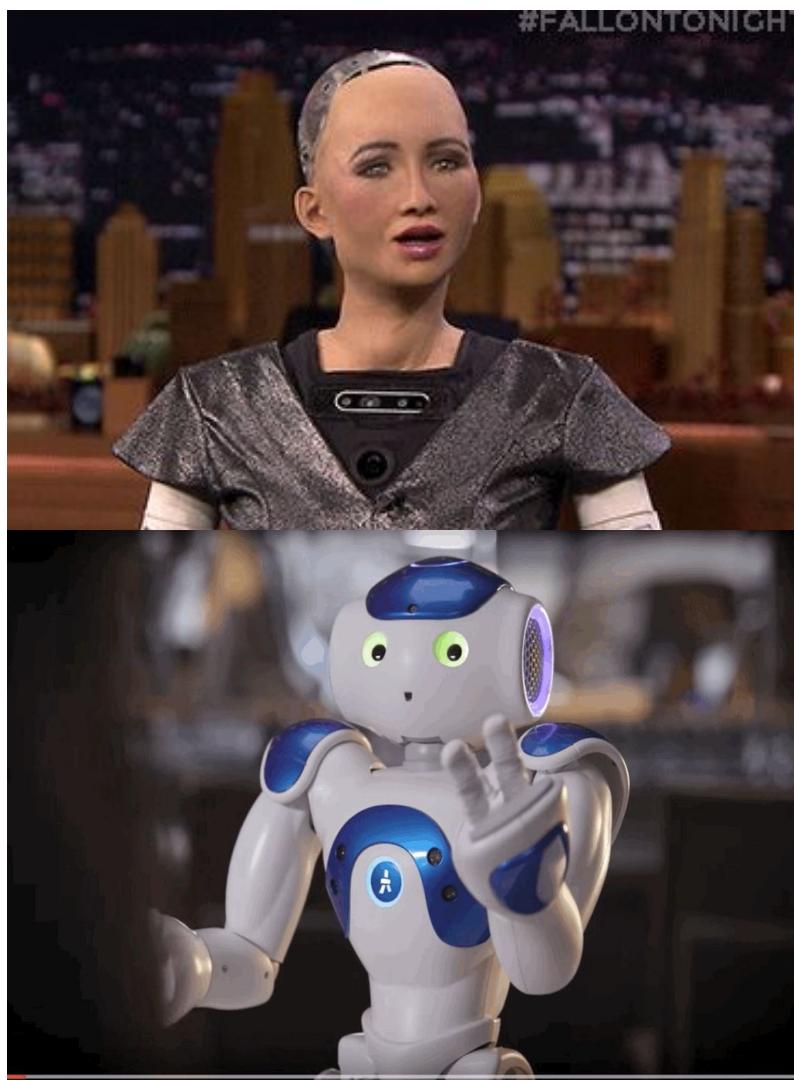
### Artificial Intelligence

### Machine Learning

The mental action or process of acquiring knowledge and understanding through thought, experience, and the senses.

# MACHINE LEARNING

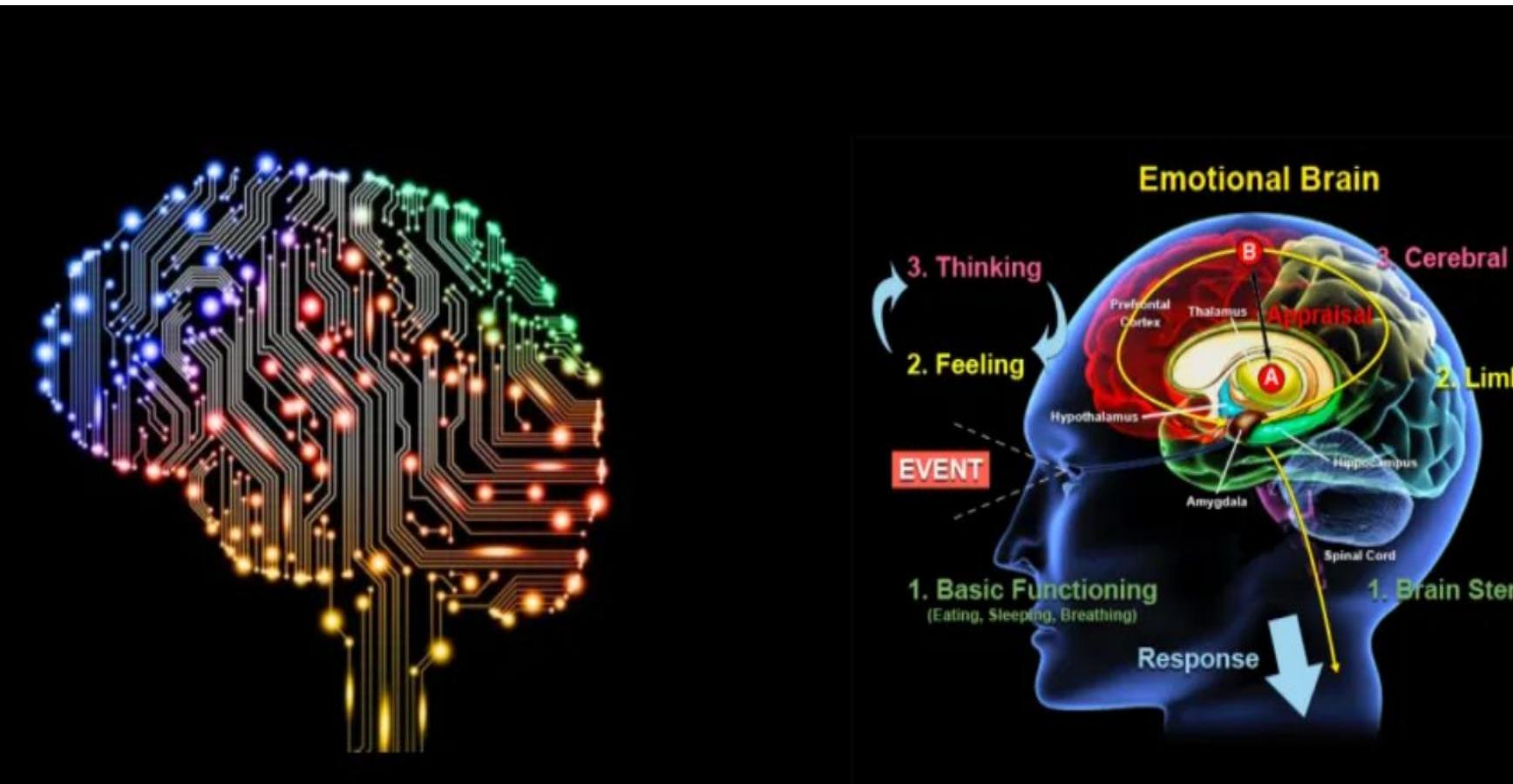
## Examples of Machine Intelligence



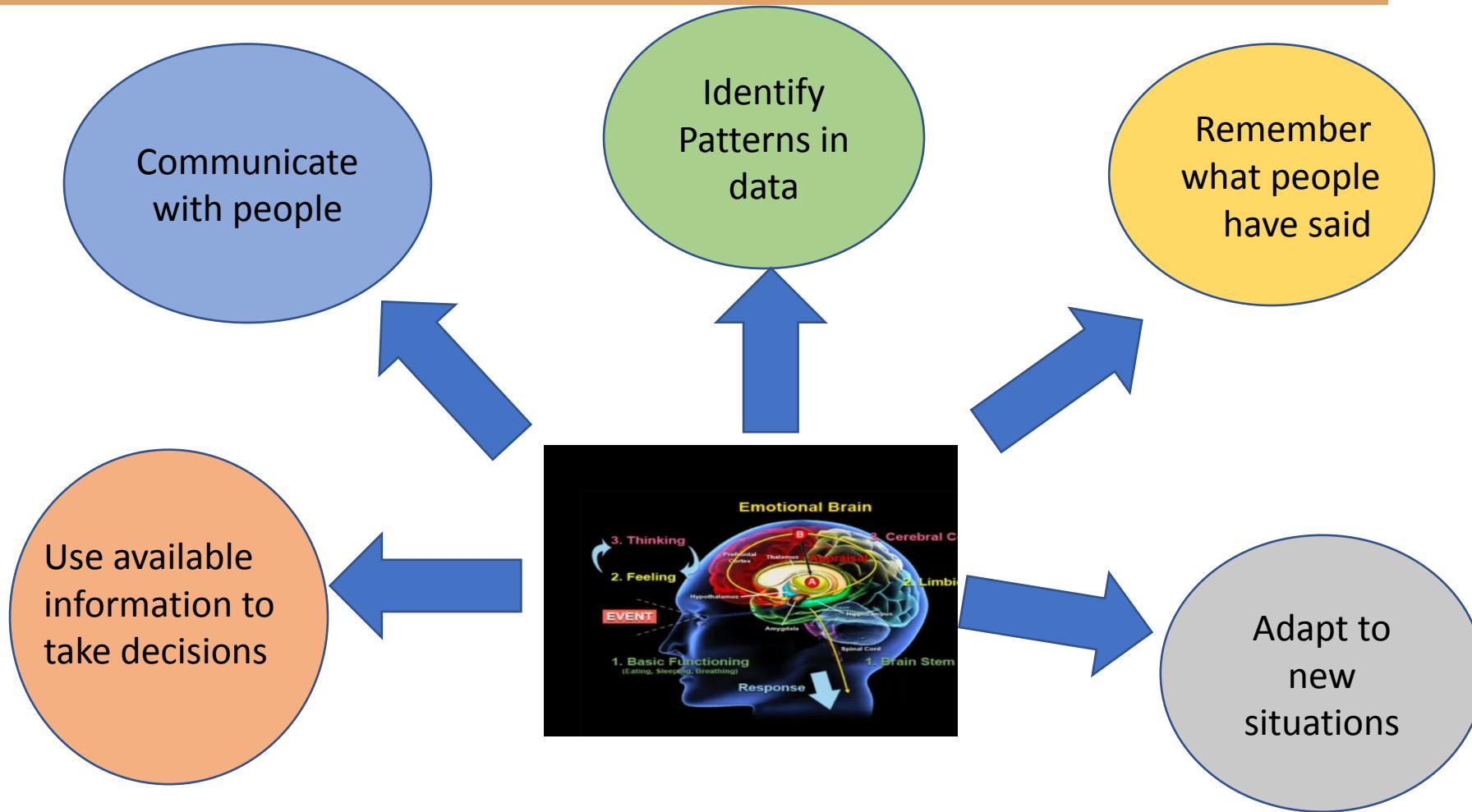
Source: <https://giphy.com/search/artificial-intelligence>

# MACHINE LEARNING

## Artificial Intelligence Vs. Human Intelligence

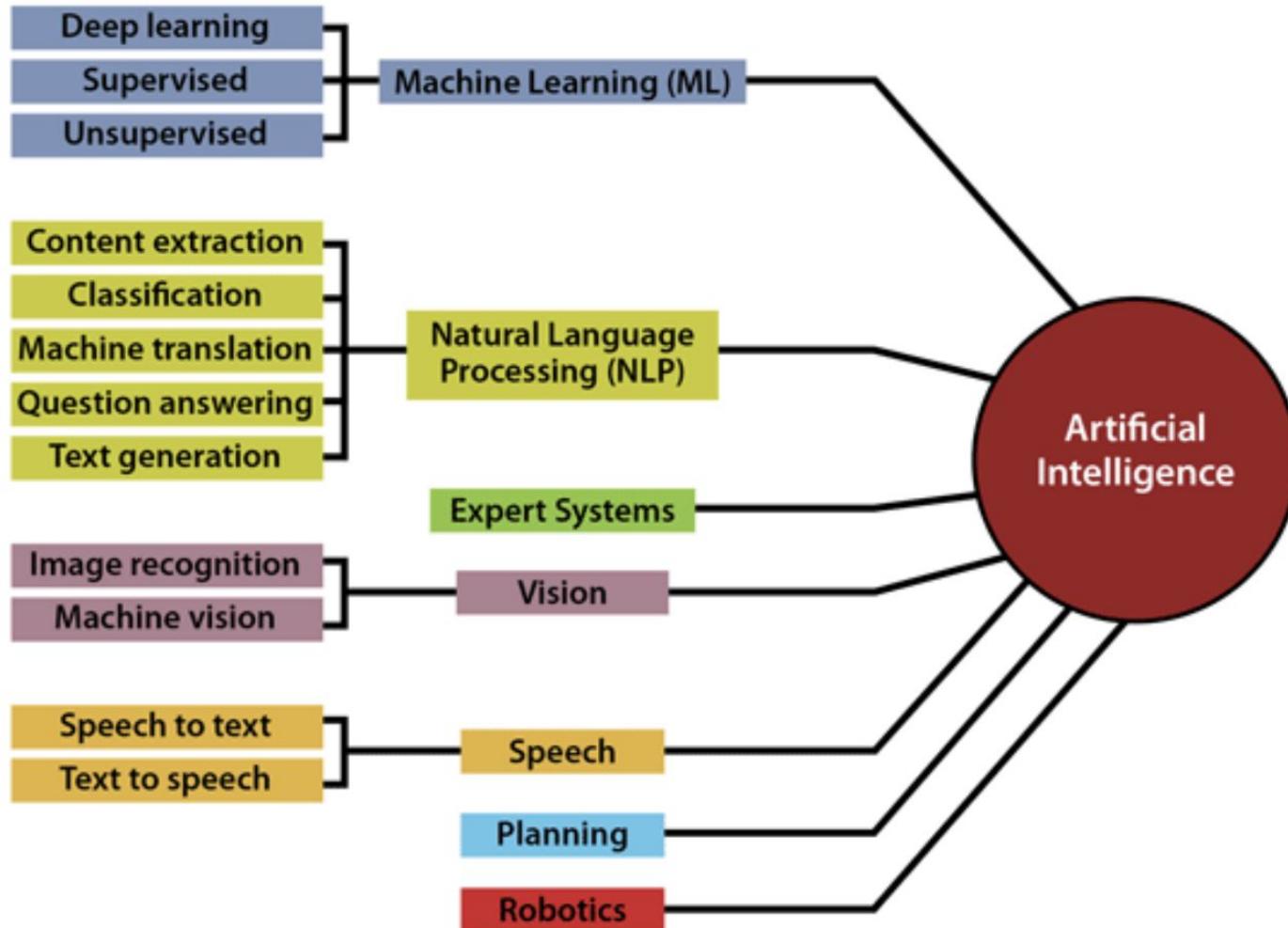


Source:<https://techswizard.com/gadget/artificial-intelligence-vs-human-intelligence/>



# MACHINE LEARNING

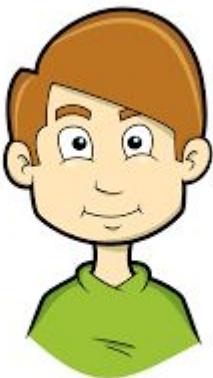
## Artificial Intelligence



# MACHINE LEARNING

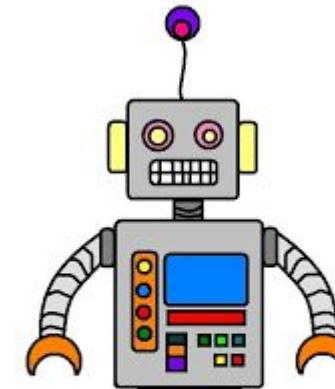
## An Intuitive Definition

- Consider the world ,we have humans and we have computers
- Can we get computers to learn from experience too???
- YES -and that is precisely what machine learning means
- but for computers we have a different term for experience that is data



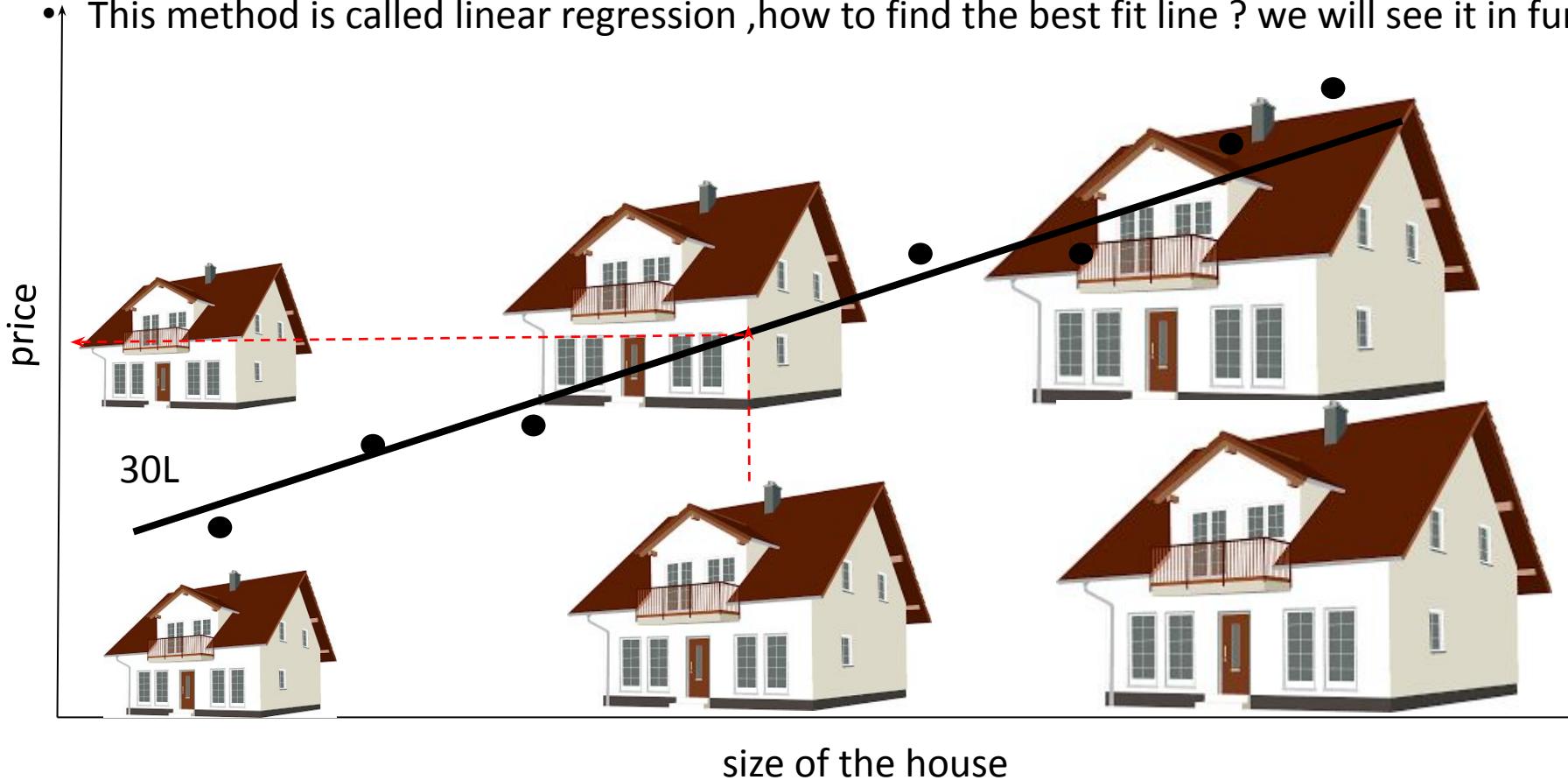
Learn from experience

data  
Learn from experience



Follow instructions

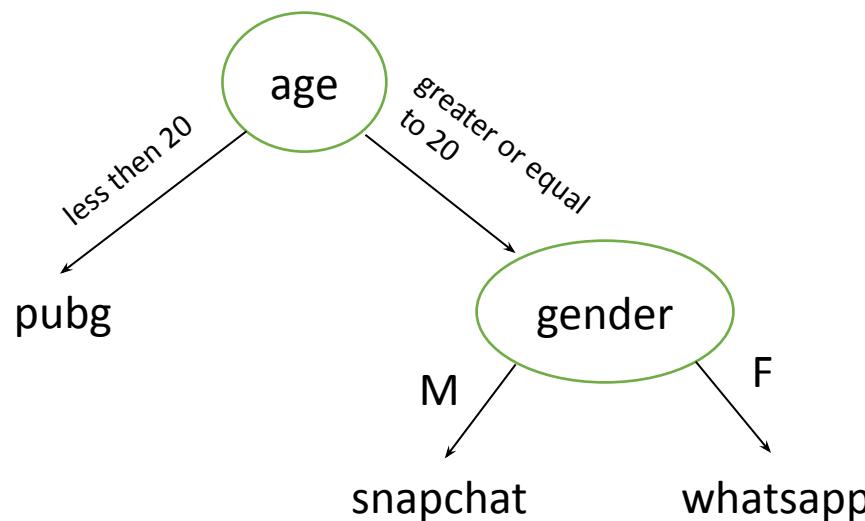
- Let us see one example to understand how a machine learns from experience(data)
- Consider we have two house with following price and we need to predict the price of the medium sized house
- We will plot them on a graph with some other data ,find a best fit line to predict its price
- This method is called linear regression ,how to find the best fit line ? we will see it in further session.



## Learning from Data

- we are on a task to built a app recommendation system with some previous data
- what do think can be criteria that influences the recommendation more , gender or age
- There is not much split in gender
- If we use the age split we see people below age 20 downloaded pubg and other downloaded whatsapp and snapchat
- we can decide the following algorithm
- This is known as decision tree learning and we will study this in detail in upcoming sections

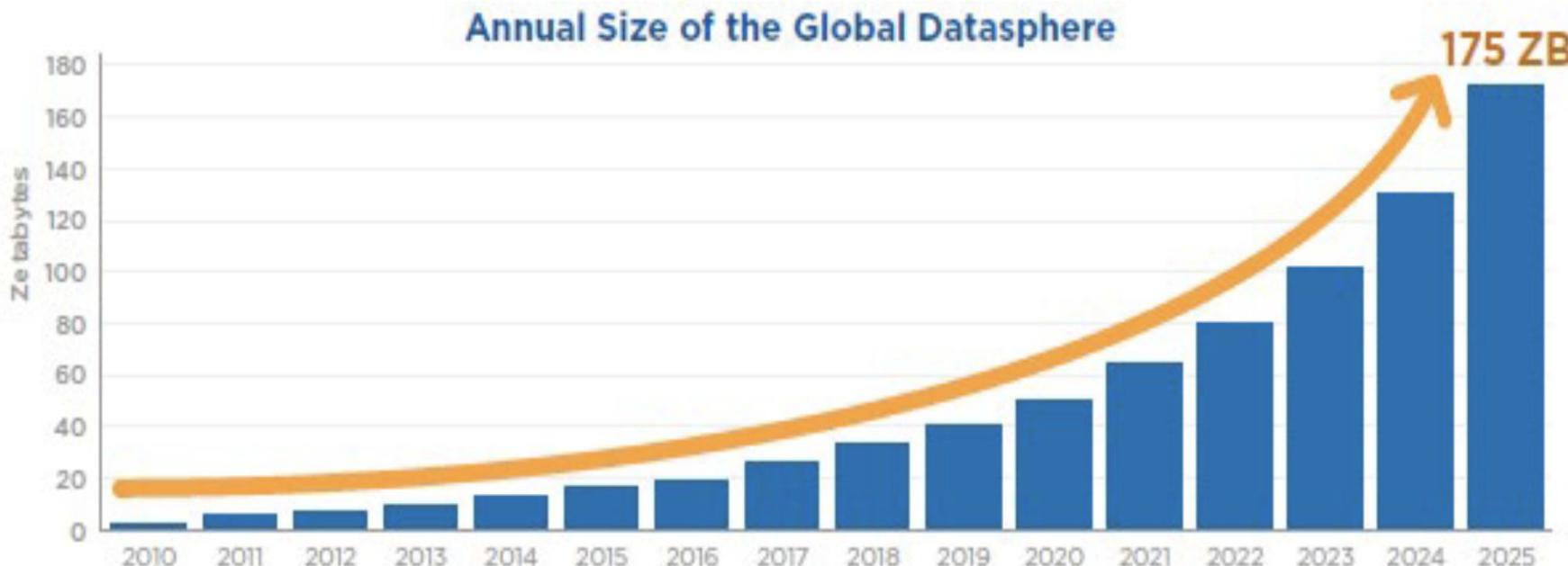
Gender	Age	App
F	15	pubg
F	25	whatsapp
M	32	snapchat
F	40	whatsapp
M	12	pubg
M	14	pubg



# MACHINE LEARNING

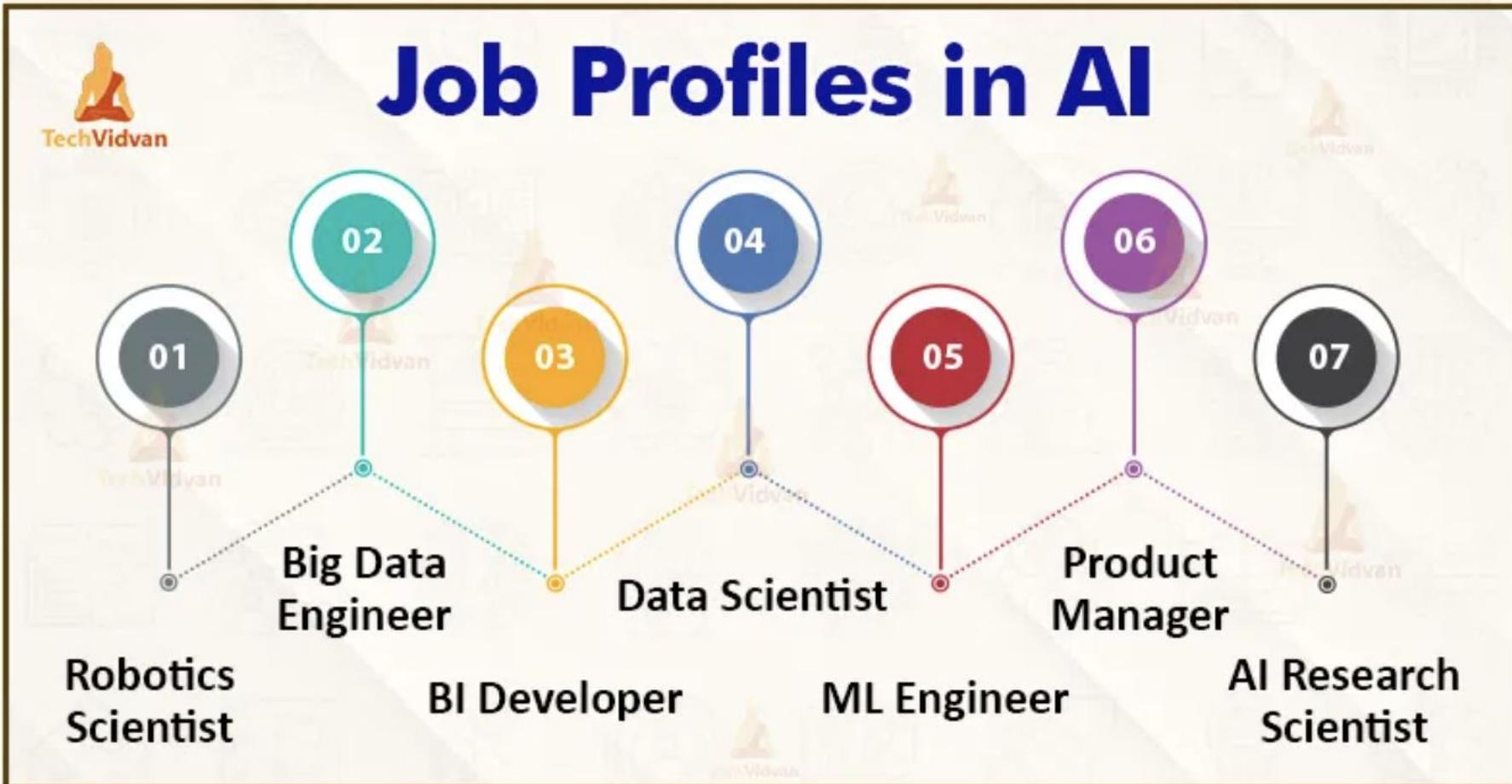
## The new dawn of Machine Learning and Artificial Intelligence

Chart 1: Exponential growth of data



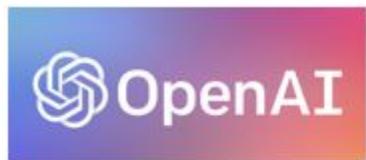
1 kilobyte	1,000
1 megabyte	1,000,000
1 gigabyte	1,000,000,000
1 terabyte	1,000,000,000,000
1 petabyte	1,000,000,000,000,000
1 exabyte	1,000,000,000,000,000,000
1 zettabyte	1,000,000,000,000,000,000,000

Source: Data Age 2025 report, sponsored by Seagate with data from IDC Global Datasphere, November 2018



# MACHINE LEARNING

Some facts about the career in this field



Source: <https://techvidvan.com/tutorials/career-in-ai/>

# MACHINE LEARNING

## Some facts about the career in this field



**The United States:** They are keen on senior roles, which offer an **average annual salary** of US\$314,000, due in part to a global talent shortage.

**The United Kingdom:** According to **IT Jobs Watch**, advertised roles in AI offer an **average of £60,000 a year** with 10 percent top positions offering an **average of £105,500**.

**Canada:** Specialists with a bit of experience can expect between **US\$70,000** and **US\$90,000 a year**.

**India:** The AI Engineer salary in India increases with years of expertise, a 2 to 4-year experienced person can get up to **₹15–2,000,000/year**.

Source: <https://intellipaat.com/blog/artificial-intelligence-salary/>

**AI DOESN'T REPLACE  
OUR CREATIVITY,  
IT EMPOWERS IT.**

# MACHINE LEARNING (UE22CS352A)



## Machine Learning Models

---

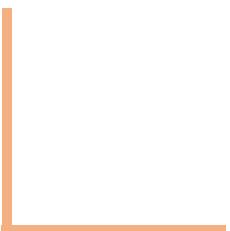
**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

# MACHINE LEARNING

---

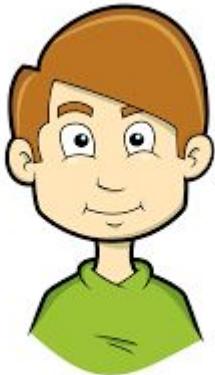
## Machine Learning Models

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.



## An intuitive definition

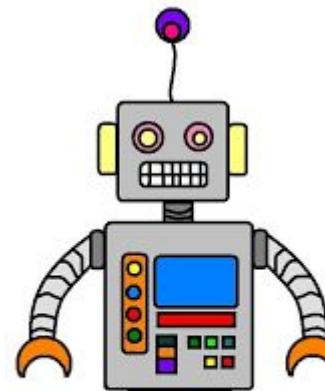
- Consider the world ,we have humans and we have computers
- Can we get computers to learn from experience too???
- YES -and that is precisely what machine learning means
- but for computers we have a different term for experience that is data



Learn from data



Learn from experience



Follow instructions

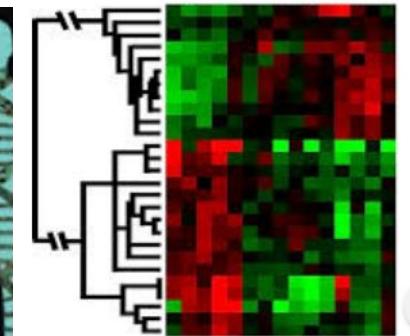
## Traditional Modelling vs Machine Learning

---

Traditional Modelling	Machine Learning
<p><b>Traditional programming</b> languages typically take data and rules as input and apply the rules to the data, to come up with answers as output.</p>	<p>On the other hand, in <b>machine learning</b> paradigm the data and answers (or labels) go in as input and the learned rules (models) come out as output</p>
<p>For example, a traditionally programmed sales analysis application will read in sales data along with rules, and then apply those rules to the data and output the key trends, analytics or insight</p>	<p>Machine learning paradigm is uniquely valuable because it lets computer to learn new rules in complex and high dimensional space, a space harder to comprehend by humans.</p>

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



## Example of a Machine Learning Task

A classic example of a task that requires machine learning:

It is very hard to say what makes a 2:

0 0 0 1 1 1 1 1 2

2 2 2 2 2 2 3 3 3

3 4 4 4 4 4 5 5 5

6 6 7 7 7 7 8 8 8

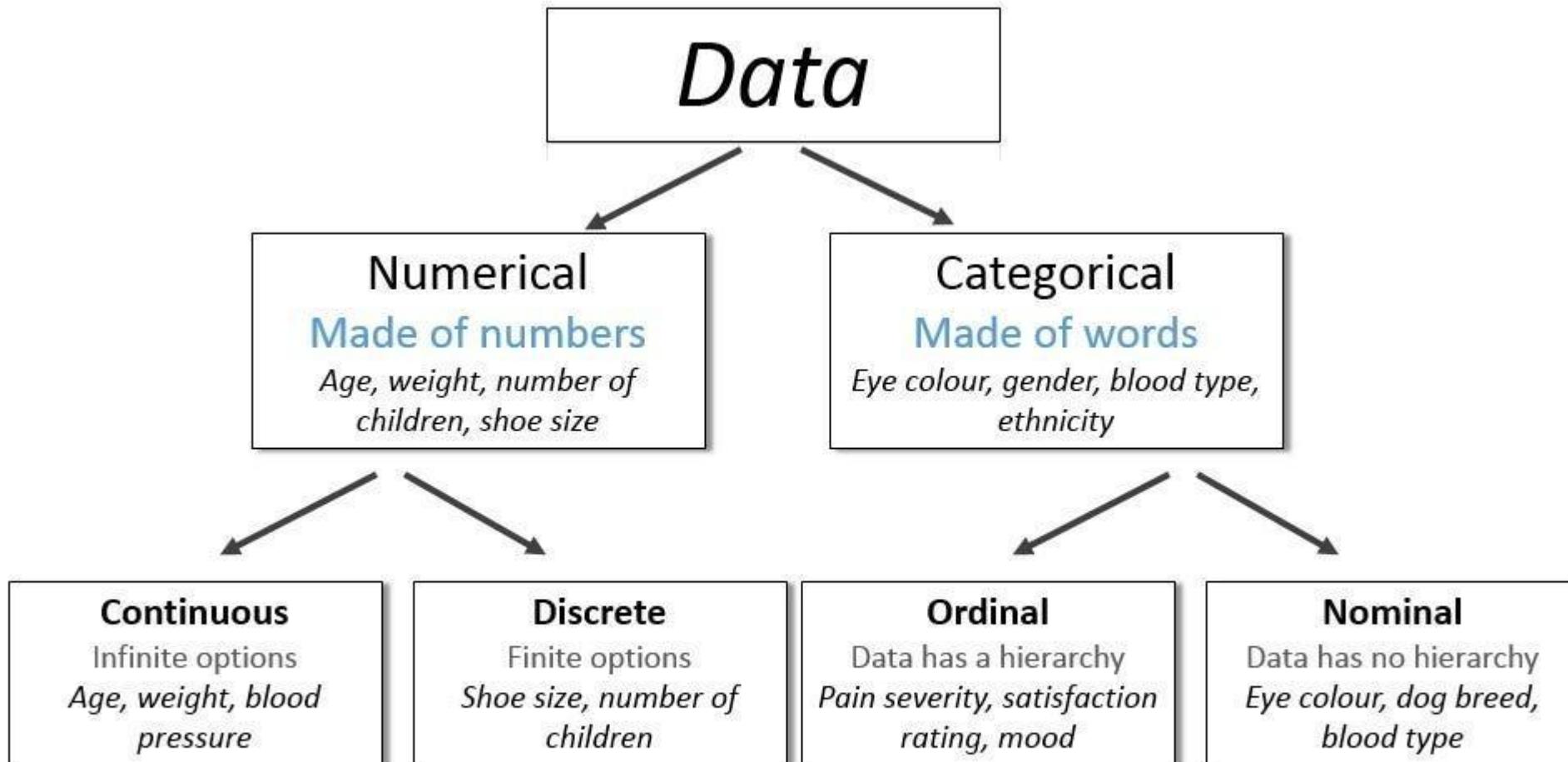
8 8 8 8 9 9 9 9 9

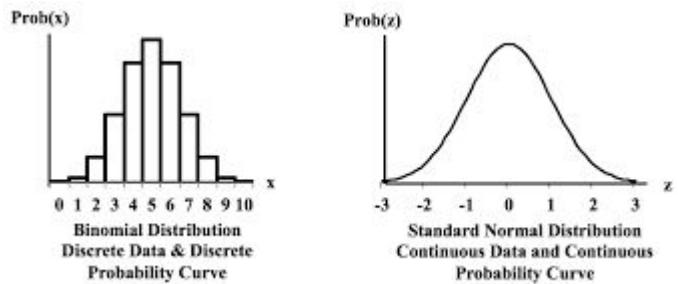
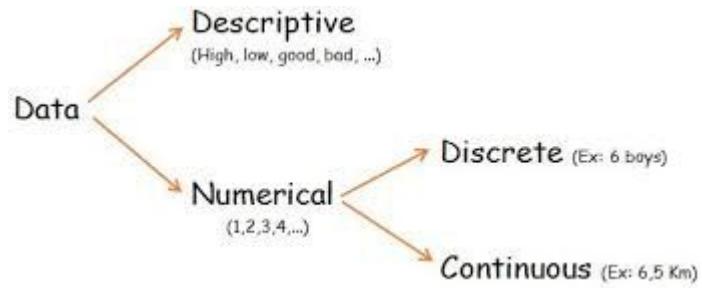


“ Machine learning is the study of computer algorithms that allow computer programs to automatically improve through experience.

~ Tom Mitchell,  
Machine Learning, McGraw Hill, 1997

Carnegie Mellon University  
Machine Learning





### Training

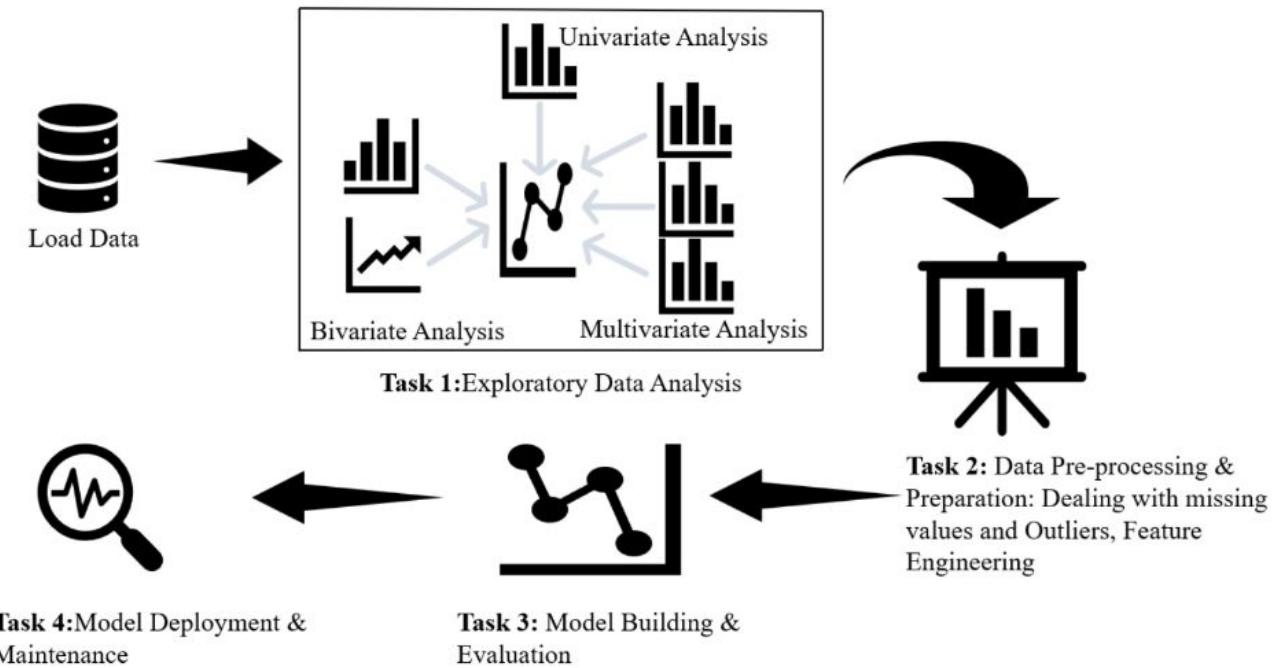
using  
*data*

### Prediction

*answer  
questions*

- **Training** refers to using the data to create and fine tune a predictive model.
- The predictive model is used to serve up **predictions** on previously **unseen data** and answer those queries.

- Gathering data
- Data preparation
- Choosing a model
- Training
- Evaluation
- Parameter tuning
- Prediction



Source: <https://faculty.iitr.ac.in/cs/bala/ML/csn382.html>

“Learning is any process by which a system improves performance from experience.” -Herbert Simon

Definition by Tom Mitchell (1998):

Machine Learning is the study of algorithms that

- improve their performance P
- • at some task T
- • with experience E.

A well-defined learning task is given by  $\langle P, T, E \rangle$ .

## Machine Learning UE22CS352A

### PTE - A Well-posed/defined ML Problem

---

**Identify Task(T), Training Experience (E) and Performance Measure (P)  
for the following problems :**

Example 1 :	Learning to play Checkers Game
Example 2 :	Handwriting Recognition Learning Problem
Example 3 :	Self-driven cars (Robot driving problem)
Example 4 :	Text Categorization Problem
Example 5 :	Learning to Check for a spam
Example 6 :	Learning to detect Credit Card Fraud

### Well-posed Learning Problem

#### Examples:

Example 1 :	Learning Checkers Game
Task	Playing Checkers
Training Experience	Playing practice games against itself
Performance Measure	Fraction of games won

### Well-posed Learning Problem

#### Examples:

Example 2 :	Handwriting Recognition Learning Problem
Task	Recognizing & Classifying handwritten words with images
Training Experience	Database of handwritten words with given classification
Performance Measure	Fraction of correct words identified.

### Well-posed Learning Problem

#### Examples:

Example 3 :	Self-driven cars
Task	To drive on the road using vision sensor
Training Experience	Sequence of images & Steering commands recorded while observing human driver
Performance Measure	Average distance travelled before an error is made

### Well-posed Learning Problem

#### Examples:

Example 4 :	Text Categorization Problem
Task	Assign a document to its content category
Training Experience	Database of pre-classified documents
Performance Measure	Fraction of documents correctly tagged

### Well-posed Learning Problem

#### Examples:

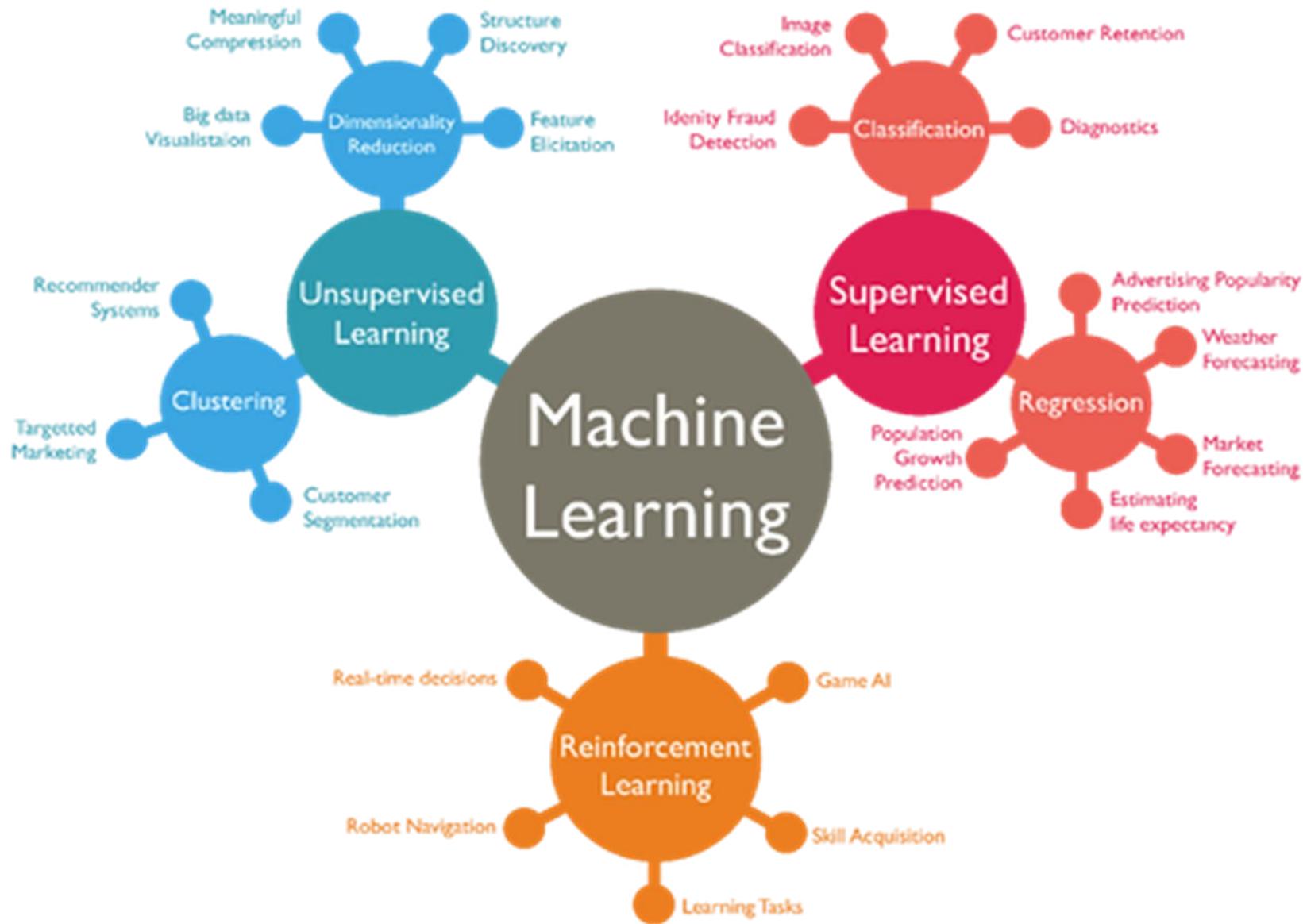
Example 5 :	Learning to Check for a spam
Task	Classify the email as spam or non-spam
Training Experience	Watching the user classify emails as spam or non-spam / Database of emails labelled as spam/ non-spam
Performance Measure	Fraction of emails correctly classified.

### Well-posed Learning Problem

#### Examples:

Example 6 :	Learning to detect Credit Card Fraud
Task	Assign label of fraud or not fraud to credit card transaction
Training Experience	Historical credit card transactions labeled as fraud or not
Performance Measure	Accuracy of fraud classifier With higher penalty when fraud is labeled as not fraud

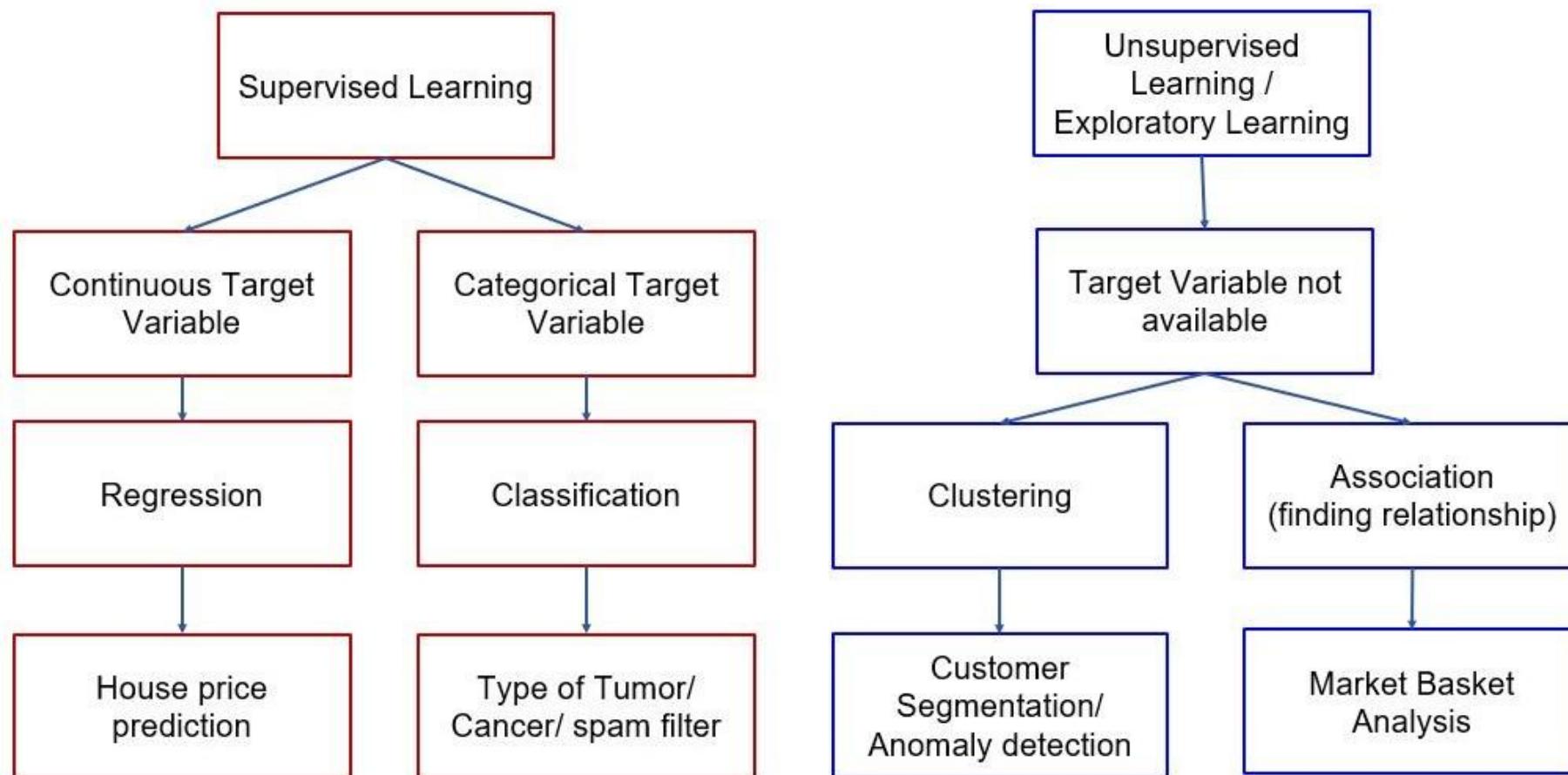
## Three Broad Categories of Machine Learning



## Types of Machine Learning

---

- Supervised (inductive) learning-
  - Given: training data +desired outputs (labels)
- Unsupervised learning-
  - Given:training data (without desired outputs)
- Reinforcement learning
  - Rewards from sequence of actions

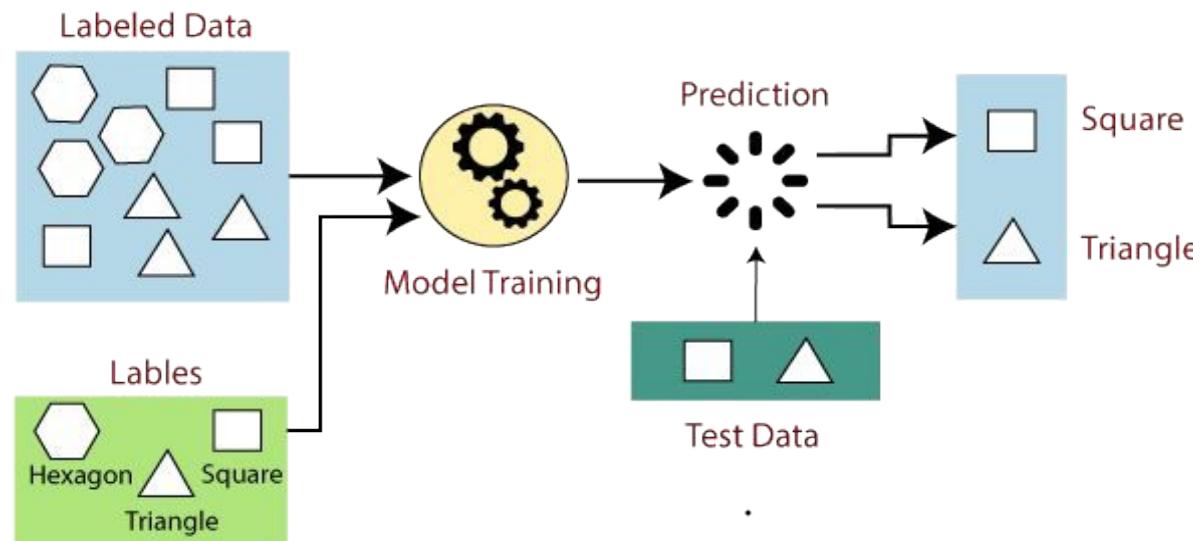


- Supervised learning is where you have input variables (X) and an output variable (Y) and you use an algorithm **to learn the mapping function from the input to the output.**

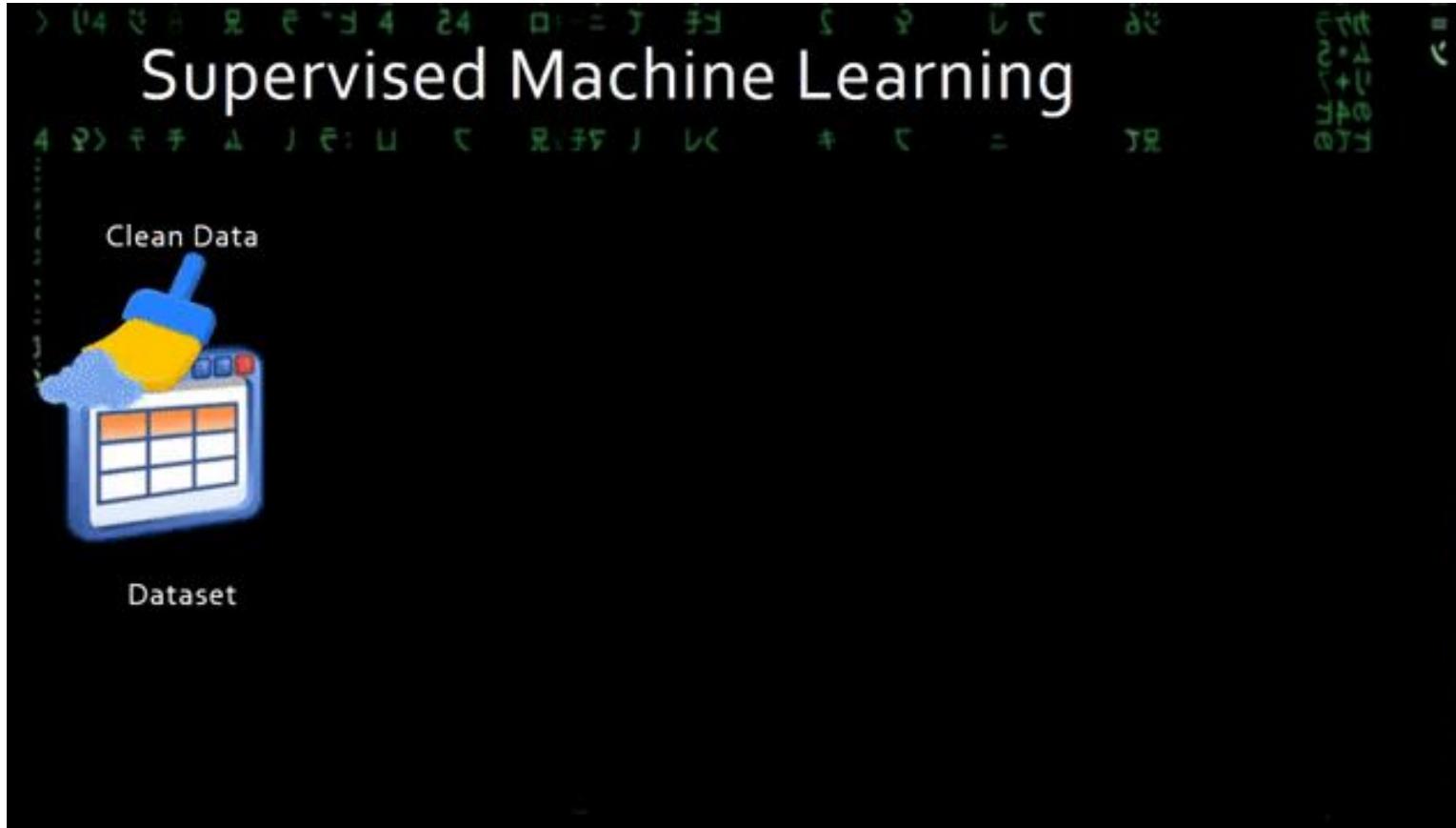
$$Y = f(X)$$

- The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (Y) for that data.
- The function approximation is also called a hypothesis function. (which means an approximation/claim)

- Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.
- Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

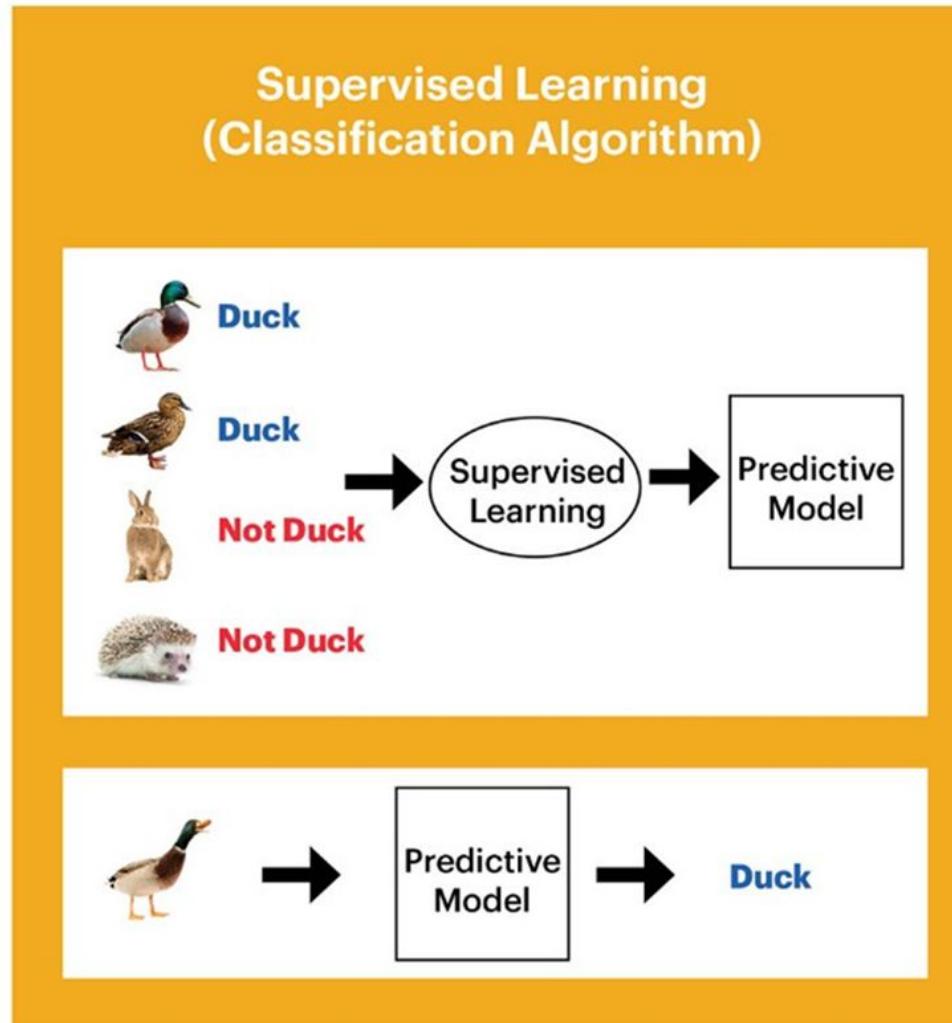


## Supervised Learning



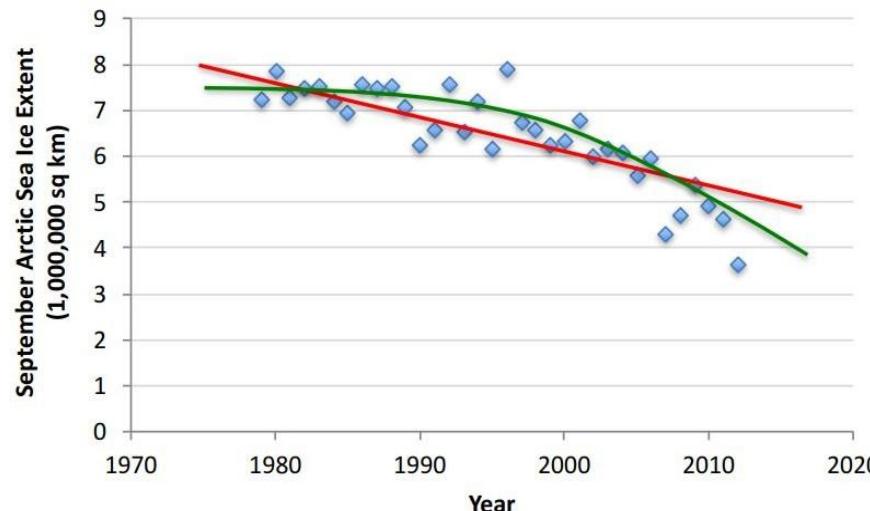
# Machine Learning UE22CS352A

## Supervised Learning



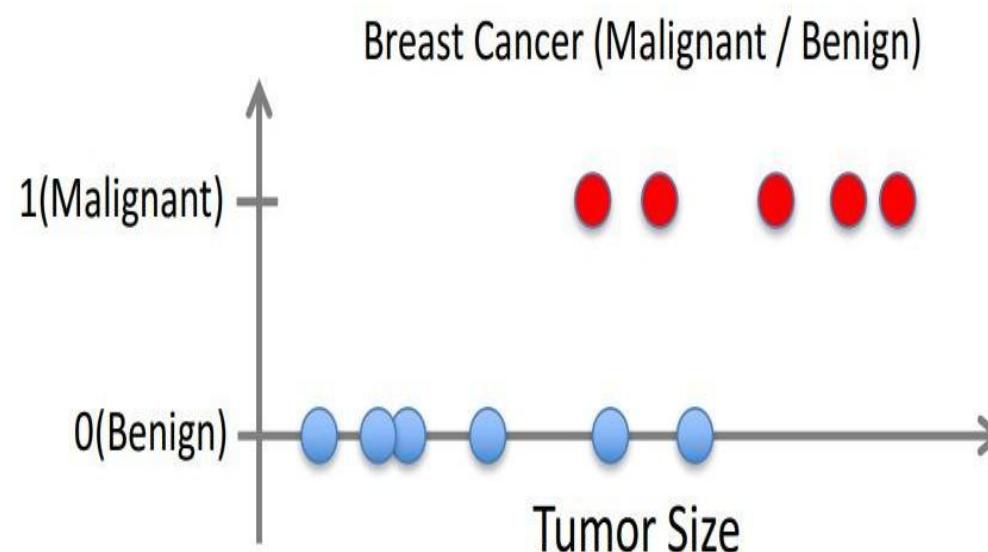
### 1. REGRESSION

- Regression algorithms are used if there is a relationship between the input variable and the output variable.
- Given  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , learn a function  $f(x)$  to predict  $y$  given  $x$  where  $y$  is real-valued, is called as regression.



### 2. CLASSIFICATION

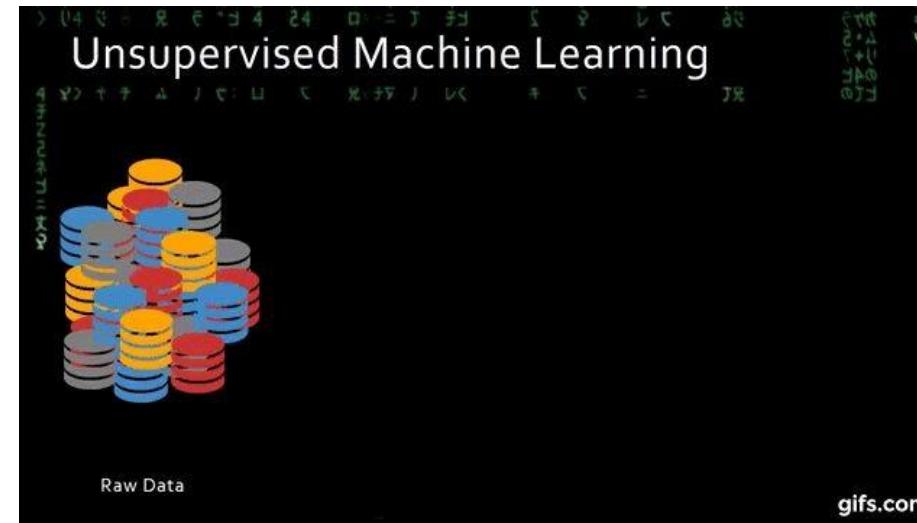
- Classification algorithms are used when the output variable is categorical
- Given  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Learn a function  $f(x)$  to predict  $y$  given  $x$  where  $y$  is categorical, is called as classification.



## Unsupervised Learning

---

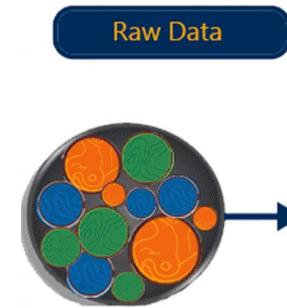
- Given - training data (without desired outputs)
- Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision
- The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.



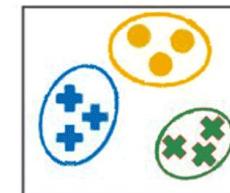
# Machine Learning UE22CS352A

## Unsupervised Learning

- Given - training data (without desired outputs)
- Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision
- The goal of unsupervised learning is to find the underlying structure of dataset, group that data according to similarities, and represent that dataset in a compressed format.



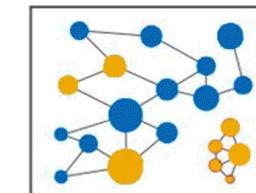
### EXAMPLES



#### CLUSTERING

Identifies similarities in groups:

Are there patterns in the data that indicate which groups to target?

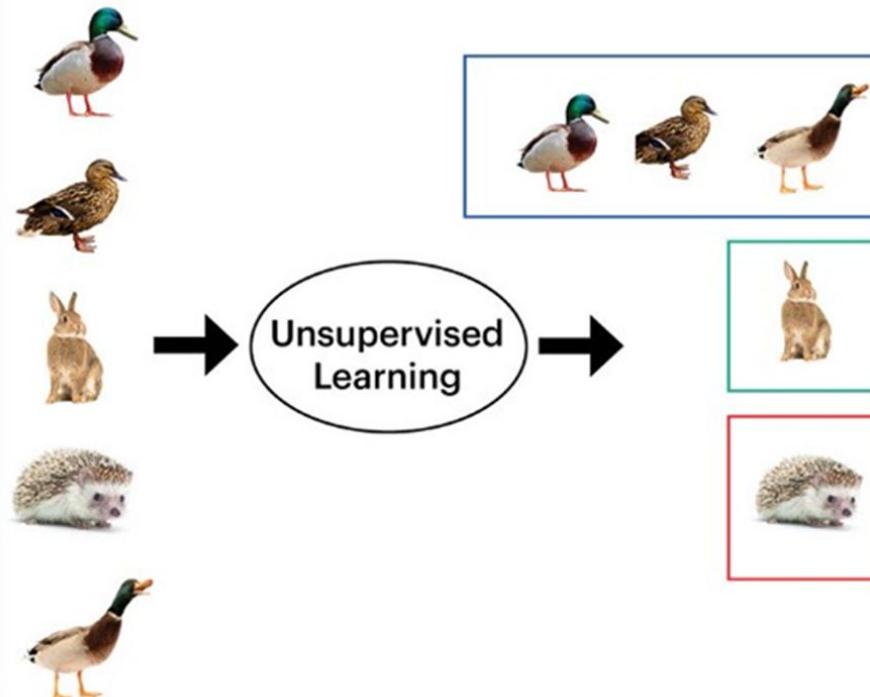


#### ANOMALY DETECTION

Identifies abnormalities in dataset:

Is the user behaving as it should? Is a hacker intruding the network?

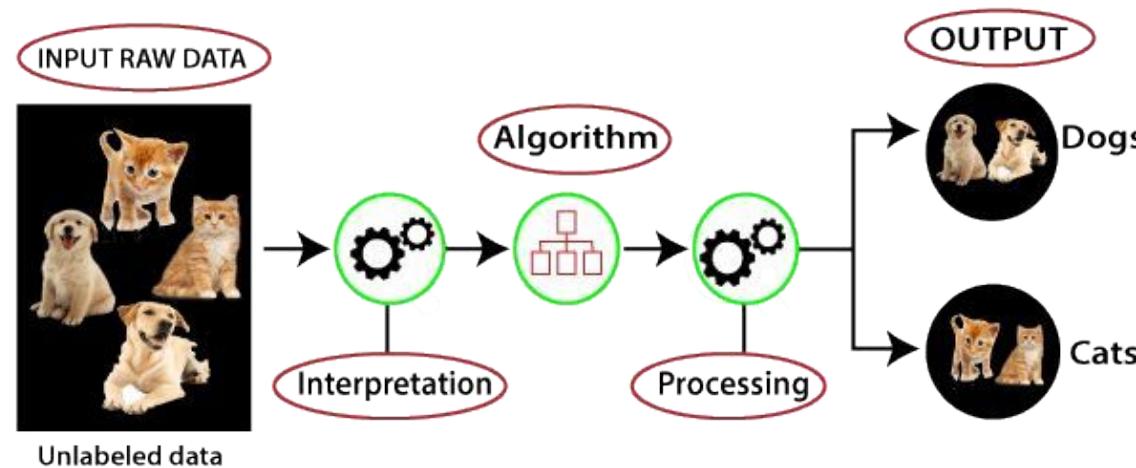
### Unsupervised Learning (Clustering Algorithm)

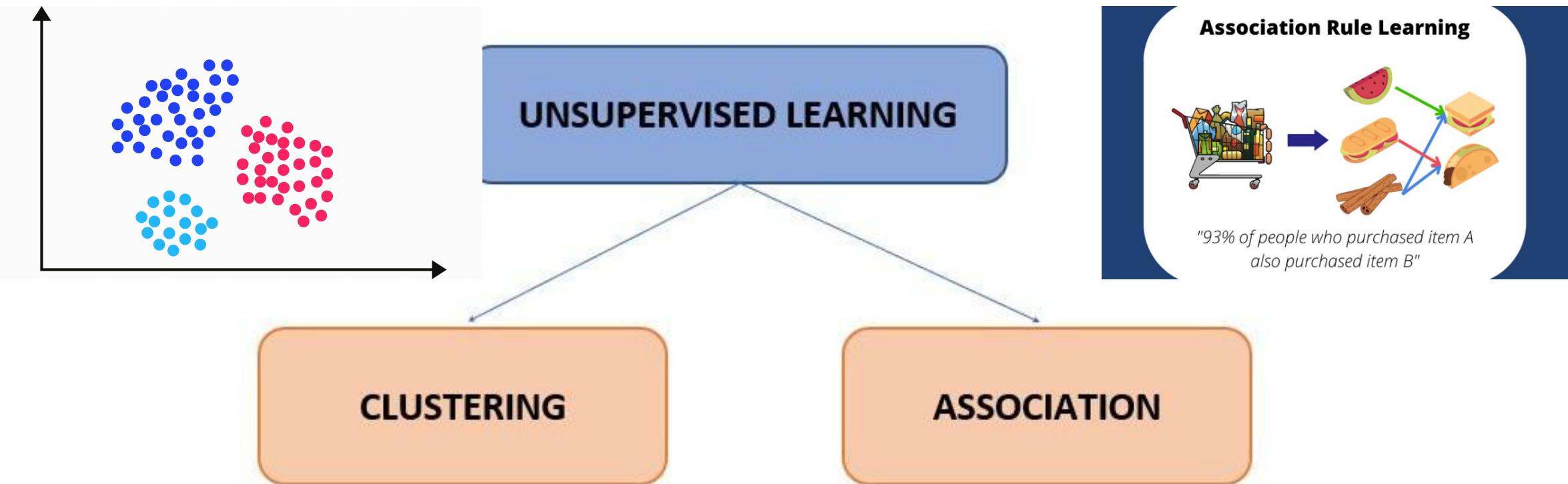


## Unsupervised Learning

---

- Unlabeled input data is fed to the machine learning model in order to train it.
- Firstly, it will interpret the raw data to find the hidden patterns from the data and then will apply suitable algorithms such as k-means clustering, Decision tree, etc.
- Once it applies the suitable algorithm, the algorithm divides the data objects into groups according to the similarities and difference between the objects.





### CLUSTERING

- Clustering is a method of grouping the objects into clusters such that objects with most similarities remains into a group and has less or no similarities with the objects of another group.



sample

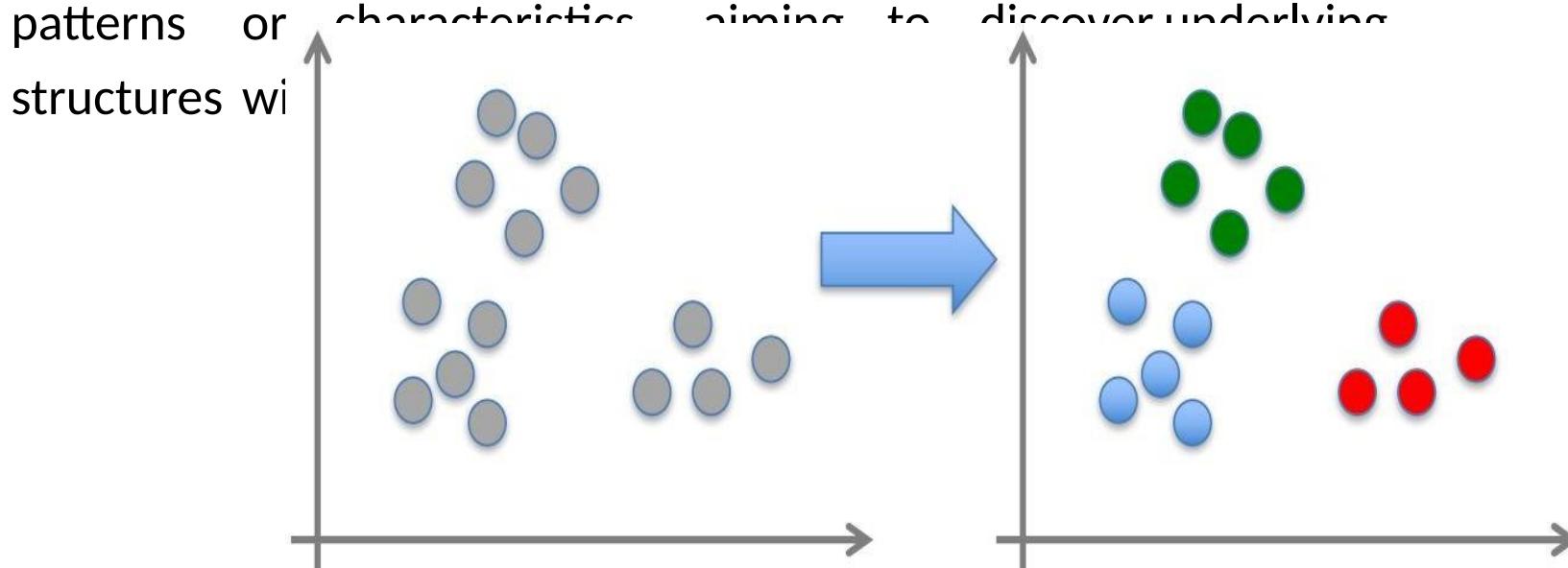


Cluster/group

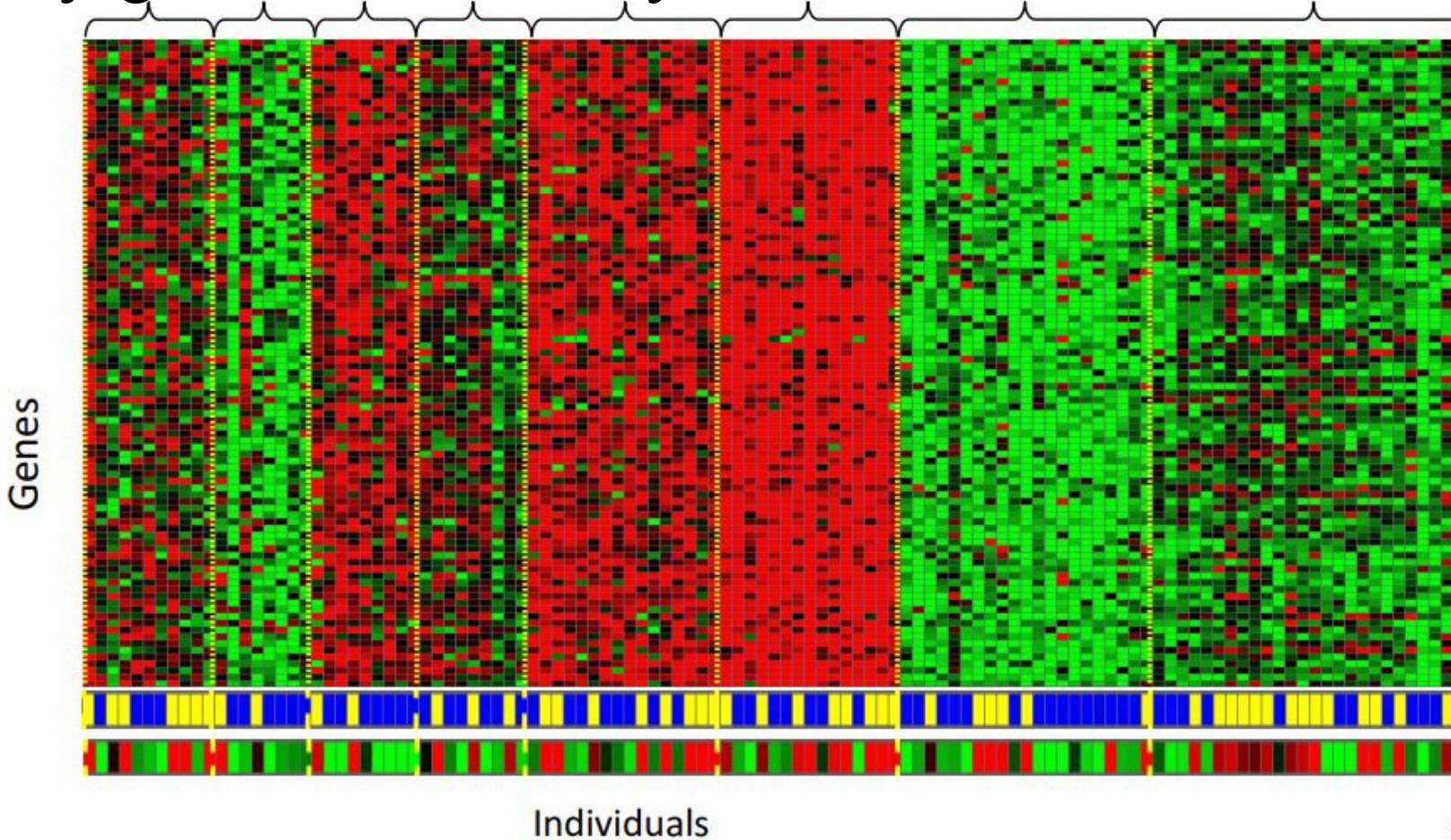
# Machine Learning UE22CS352A

## Unsupervised Learning - Clustering

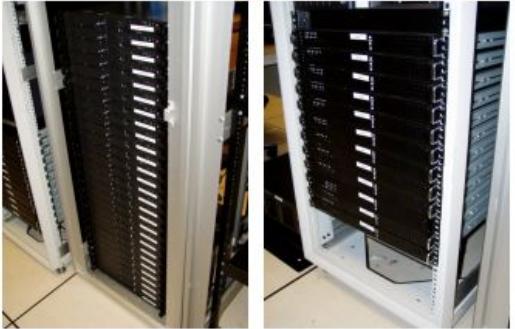
- Given  $x_1, x_2, \dots, x_n$  (without labels), clustering in machine learning is a technique that groups similar data points together based on their intrinsic patterns or characteristics aiming to discover underlying structures without explicit labeling.



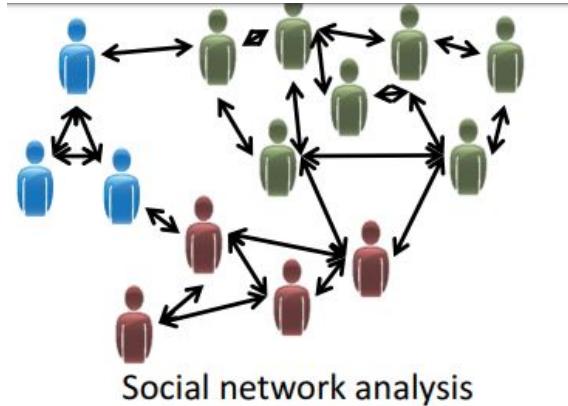
## Genomics application: group individuals by genetic similarity



## Example of Unsupervised Learning Applications



Organize computing clusters



Market segmentation



Astronomical data analysis

#### ASSOCIATION:

- An association rule is an unsupervised learning method which is used for finding the relationships between variables in the large database.
- It determines the set of items that occurs together in the dataset.

#### Examples:

- People that buy a new home most likely to buy new furniture.
- A subgroup of cancer patients grouped by their gene expression measurements
- Groups of shopper based on their browsing and purchasing histories
- Movie group by the rating given by movies viewers

- K-means clustering
- Hierarchical clustering
- Anomaly detection
- Principal Component Analysis (PCA)
- Independent Component Analysis
- Apriori algorithm
- Singular value decomposition

## Reinforcement Learning

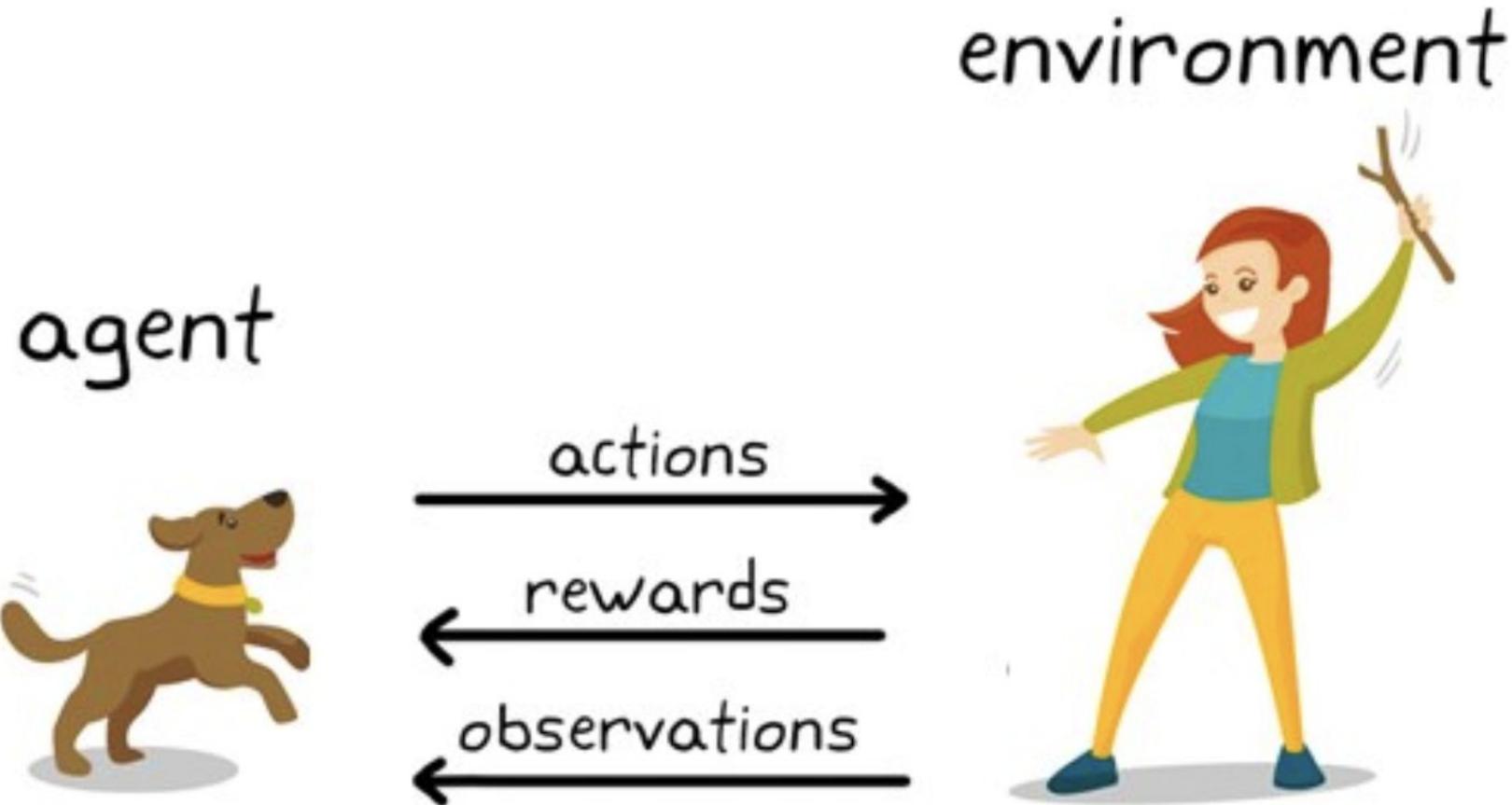
---

- Reinforcement Learning is learning **what to do and how to map situations to actions.**
- It is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."
- The learner is not told which action to take, but **instead must discover** which action will yield the maximum reward
- The algorithm (agent) evaluates a current situation (state), takes an action, and receives feedback (reward) from the environment after each act.
- Positive feedback is a reward, and negative feedback is punishment for making a mistake.



# Machine Learning UE22CS352A

## Reinforcement Learning



<https://www.kdnuggets.com/2019/10/mathworks-reinforcement-learning.html>

## Training a logistics robot



Source: <https://www.scribbr.com/ai-tools/reinforcement-learning/>

- 1) **Agent:** The learner or decision-maker that interacts with the environment.
- 2) **Environment:** The external system with which the agent interacts.
- 3) **State (s):** A representation of the current situation of the environment. It's the input to the agent's decision-making process.
- 4) **Action (a):** The choices available to the agent at each state.
- 5) **Policy ( $\pi$ ):** A strategy or rule that the agent follows to select actions.
- 6) **Reward (r):** A scalar feedback signal from the environment indicating how good or bad the action taken by the agent was.
- 7) **Value function (V or Q):** It predicts the expected cumulative reward the agent would receive starting from a certain state (or state-action pair) and following a particular policy.

**The goal of the agent is to learn the optimal policy that maximizes cumulative rewards.**

- RL involves a lot of computing.
- No training dataset is there for training any model.
- Q-table in RL
  - Is a **look up table** where **maximum expected future rewards** for an **action** are computed at each state.
  - This table tells us which action is best at each state.
  - For learning/ computing each value of **Q-table**, **Q-learning** algorithm is used.
  - **Q-function** takes two inputs: **state and action** and uses Bellman's equation:

$$Q'(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | (s_t, a_t)]$$

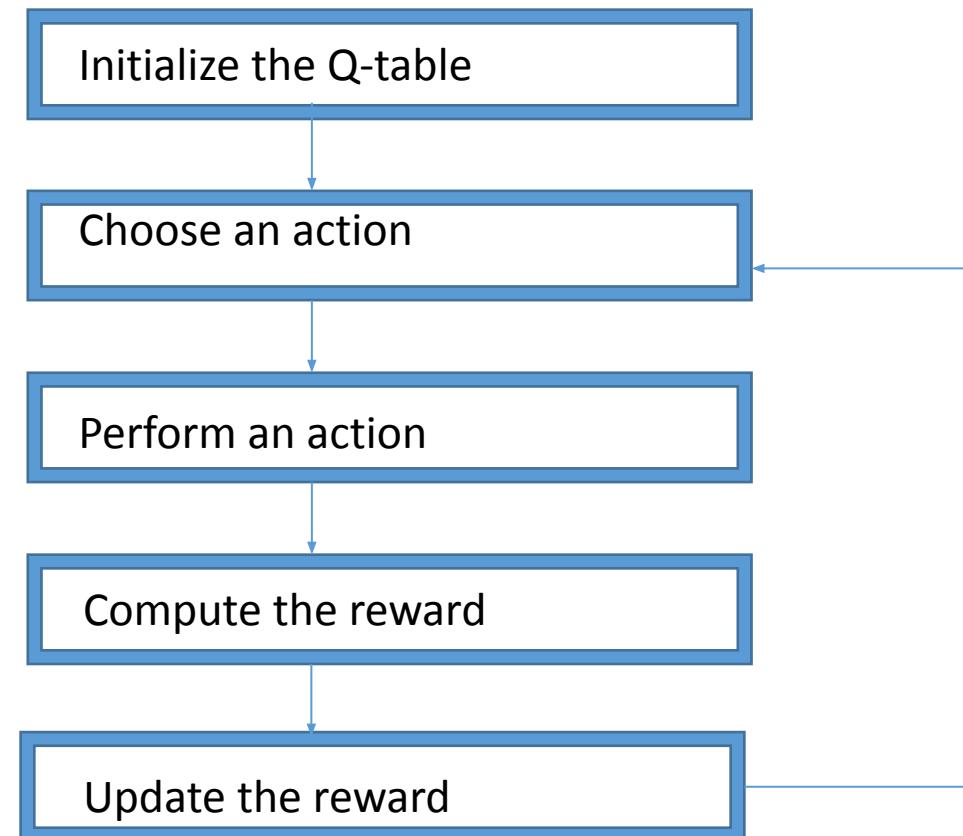
Q-value for the state at time t

Reward

Discount rate

Initially the values in the Q table are zeros

Q-learning algorithm:



Many iterations to get good Q-table

What is a Policy?  
For a given state, which is the optimal action?  
For (s, a) pair, what is the optimal action?

An intuitive way to understand the relation between the agent and its environment is with the following example dialogue.

**Environment:** You are in state 65. You have 4 possible actions.

**Agent:** I'll take action 2.

**Environment:** You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.

**Agent:** I'll take action 1.

**Environment:** You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.

**Agent:** I'll take action 2.

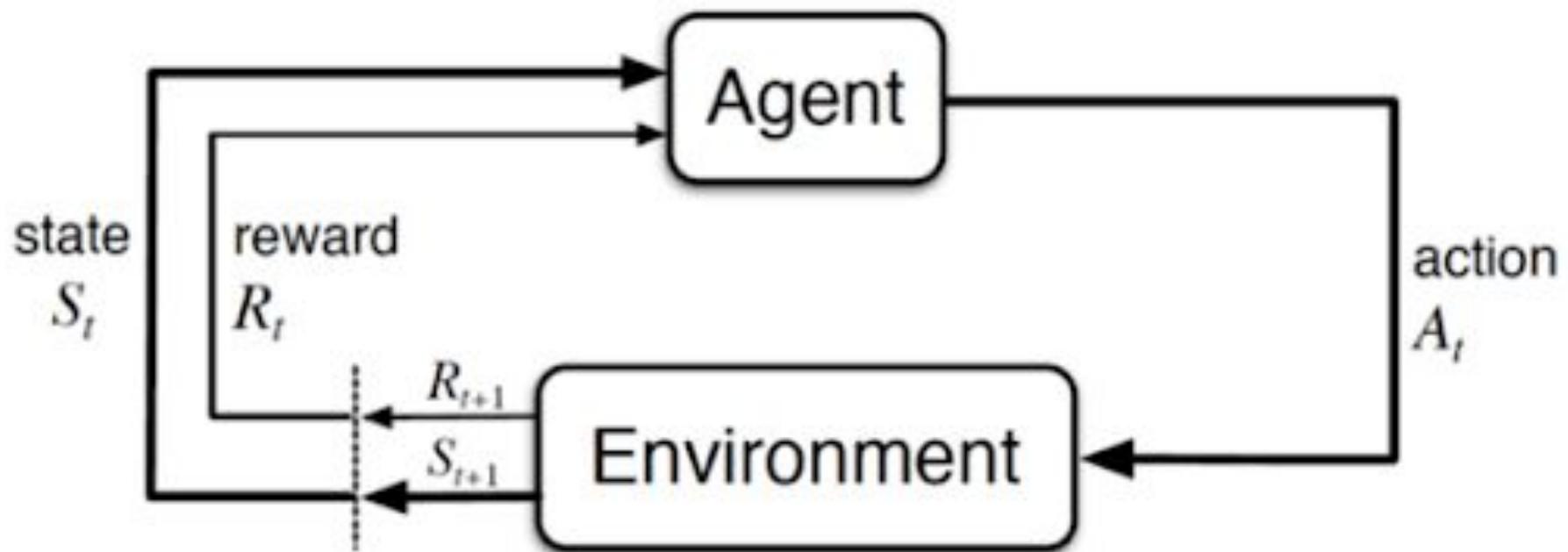
**Environment:** You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.

:

:

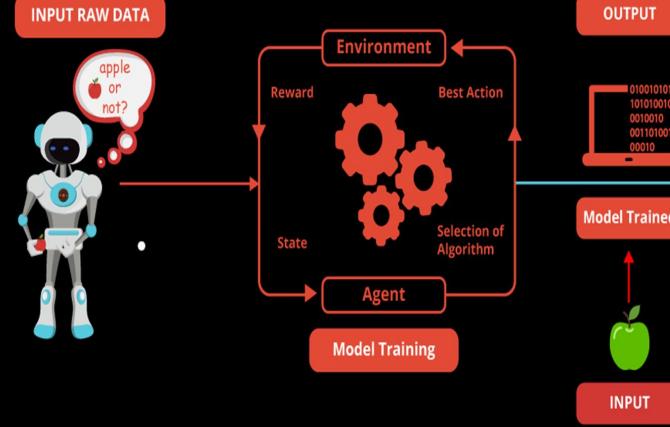
# Machine Learning UE22CS352A

## Reinforcement Learning



# Machine Learning UE22CS352A

## Reinforcement learning



## **Game Board:**



Current state ( $s$ ):      **0 0 0**  
                                **0 1 0**

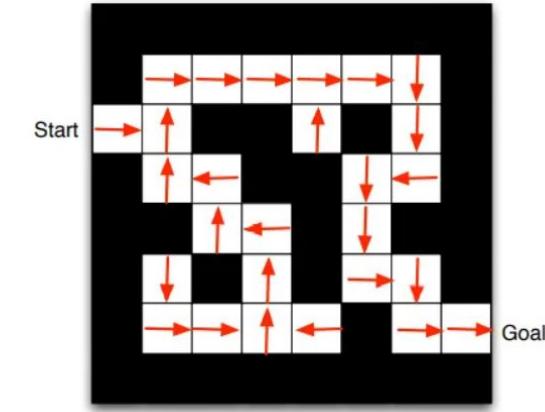
### **Q Table:**

	<b>0 0 0</b> <b>1 0 0</b>	<b>0 0 0</b> <b>0 1 0</b>	<b>0 0 0</b> <b>0 0 1</b>	<b>1 0 0</b> <b>0 0 0</b>	<b>0 1 0</b> <b>0 0 0</b>	<b>0 0 1</b> <b>0 0 0</b>
	0.2	0.3	1.0	-0.22	-0.3	0.0
	-0.5	-0.4	-0.2	-0.04	-0.02	0.0
	0.21	0.4	-0.3	0.5	1.0	0.0
	-0.6	-0.1	-0.1	-0.31	-0.01	0.0

Y = 0.95

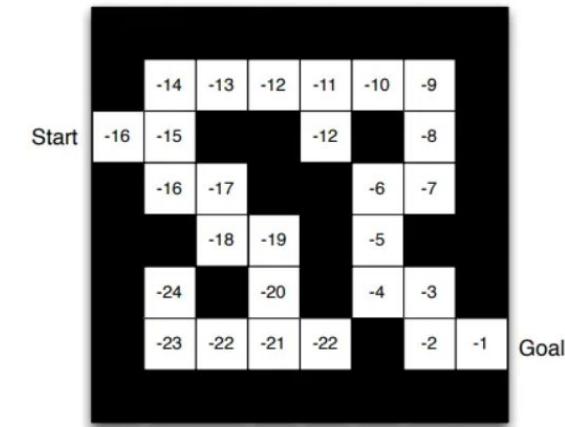


Source: <https://medium.com/analytics-vidhya/basic-terminology-reinforcement-learning-2357fd5f0e51>



- Arrows represent policy  $\pi(s)$  for each state  $s$

Policy for an Autonomous Car to reach its end goal

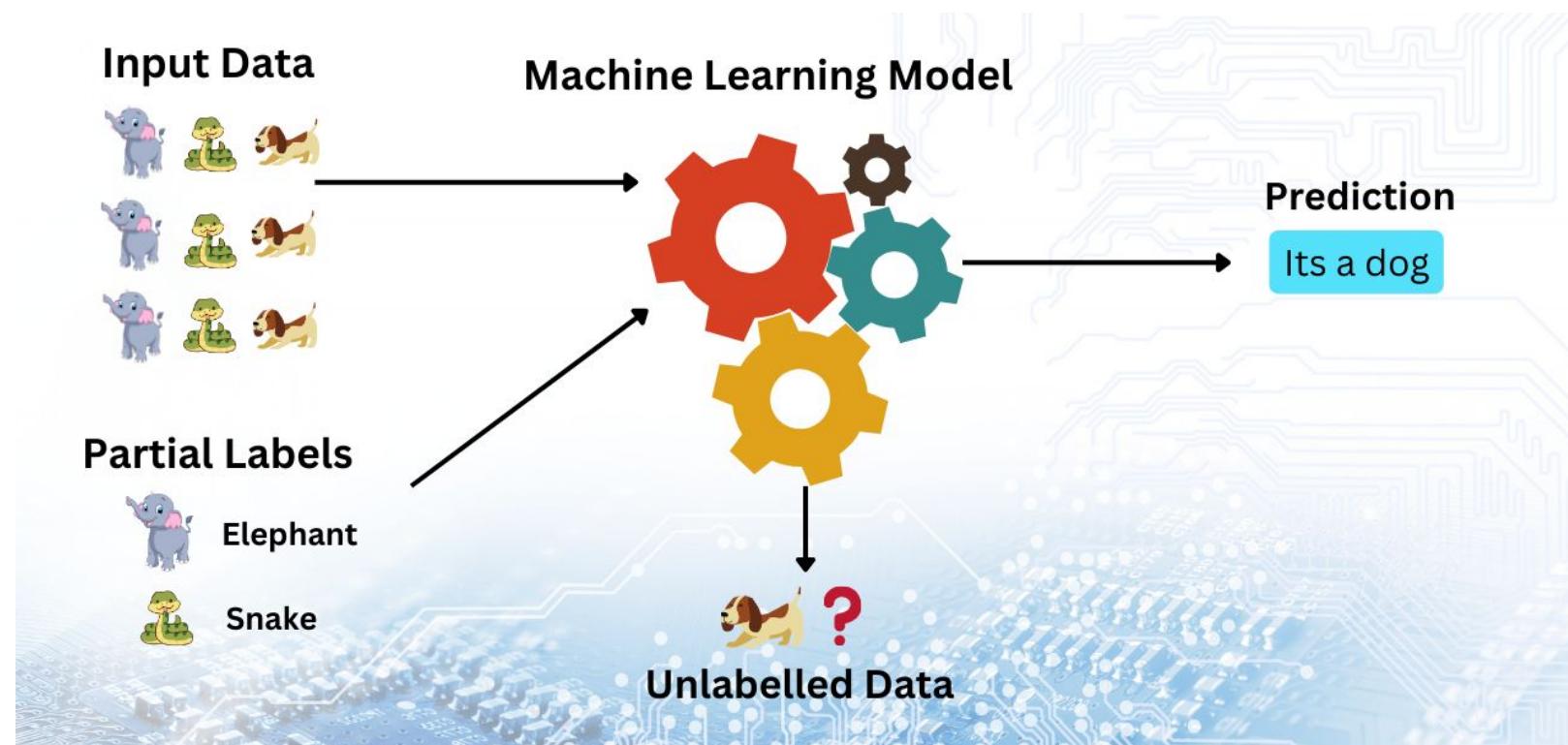


- Numbers represent value  $v_\pi(s)$  of each state  $s$

Value Function for an Autonomous Car to reach its goal.

## Semi-Supervised Learning

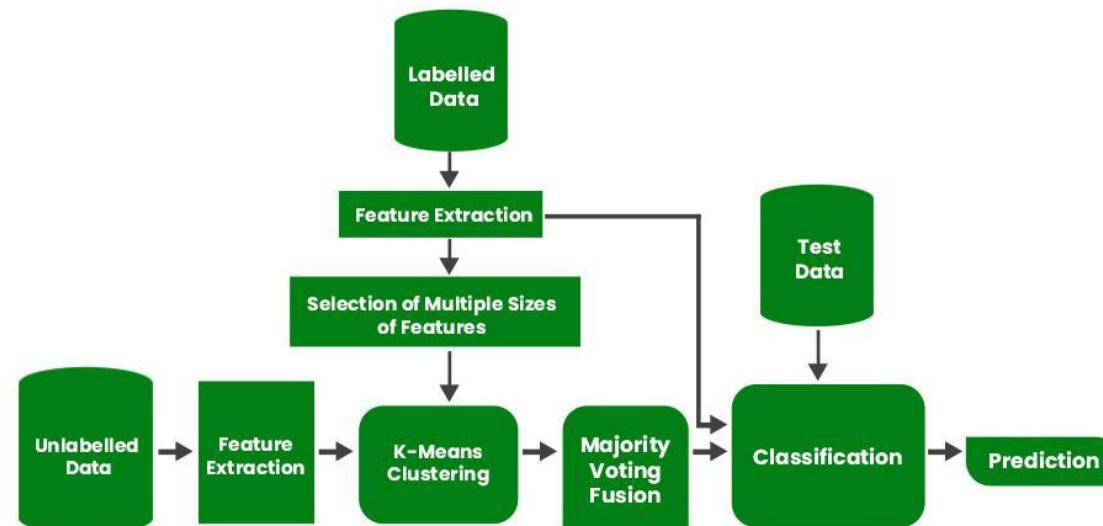
- Semi-supervised learning is a type of machine learning that falls in between supervised and unsupervised learning.
- It is a method that uses a small amount of labeled data and a large amount of unlabeled data to train a model.



## Semi-Supervised Learning

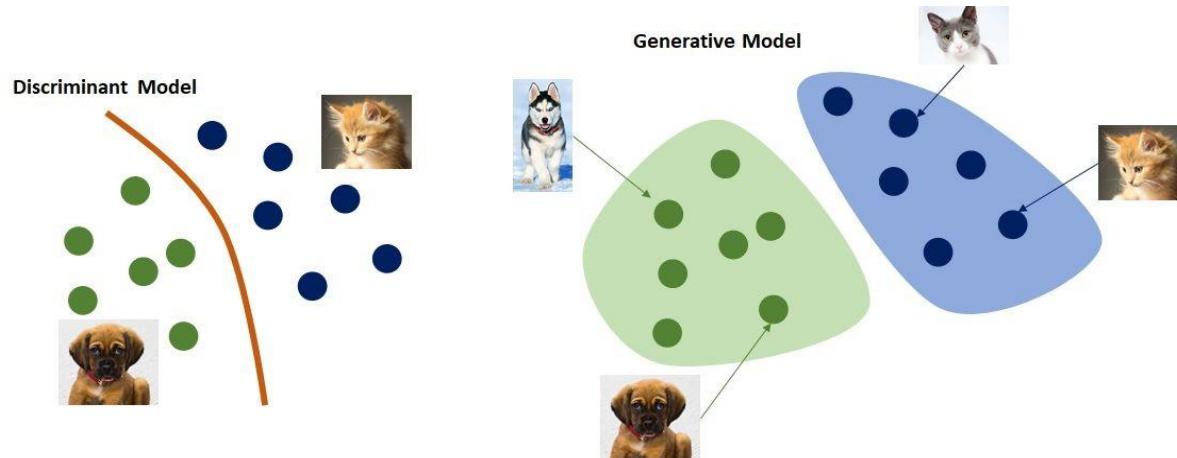
---

- The goal of semi-supervised learning is to learn a function that can accurately predict the output variable based on the input variables, similar to supervised learning.
- However, unlike supervised learning, the algorithm is trained on a dataset that contains both labeled and unlabeled data.
- Semi-supervised learning is particularly useful when there is a large amount of unlabeled data available, but it's too expensive or difficult to label all of it.



- Machine learning models can be classified into **discriminative and generative models**.
- In simple words, **a discriminative model** makes predictions based on conditional probability and is either used for classification or regression.
- On the other hand, **a generative model** revolves around the distribution of a dataset to return a probability for a given example.

Discriminative models draw boundaries in the data space, while generative ones model how data is placed throughout the



## Generative and Discriminative Models: An Analogy

Let's say you have input data  $x$  and you want to classify the data into labels  $y$ . A generative model learns the **joint** probability distribution  $p(x,y)$  and a discriminative model learns the **conditional** probability distribution  $p(y|x)$  - which you should read as "*the probability of  $y$  given  $x$* ".

$p(x,y)$  is

	$y=0$	$y=1$
$x=1$	1/2	0
$x=2$	1/4	1/4

$p(y|x)$  is

	$y=0$	$y=1$
$x=1$	1	0
$x=2$	1/2	1/2

x(input)	y(label)
1	0
1	0
2	0
2	1

The distribution  $p(y|x)$  is the natural distribution for classifying a given example  $x$  into a class  $y$ , which is why algorithms that model this directly are called discriminative algorithms. Generative algorithms model  $p(x,y)$ , which can be transformed into  $p(y|x)$  by applying Bayes rule and then used for classification. However, the distribution  $p(x,y)$  can also be used for other purposes. For example, you could use  $p(x,y)$  to generate likely  $(x,y)$  pairs.

Types of discriminative models in machine learning include:

- Logistic regression
- Support Vector Machine
- Decision Tree
- Random Forest

## Generative Model

---

Types of generative models in machine learning include:

- Bayesian Network
- Hidden Markov model
- Autoregressive model
- Generative adversarial network (GANs)

# MACHINE LEARNING (UE22CS352A)



## Concept Learning

---

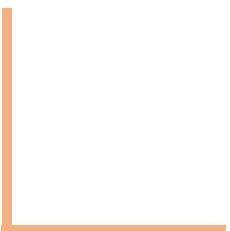
**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

# MACHINE LEARNING

---

## Concept Learning

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.

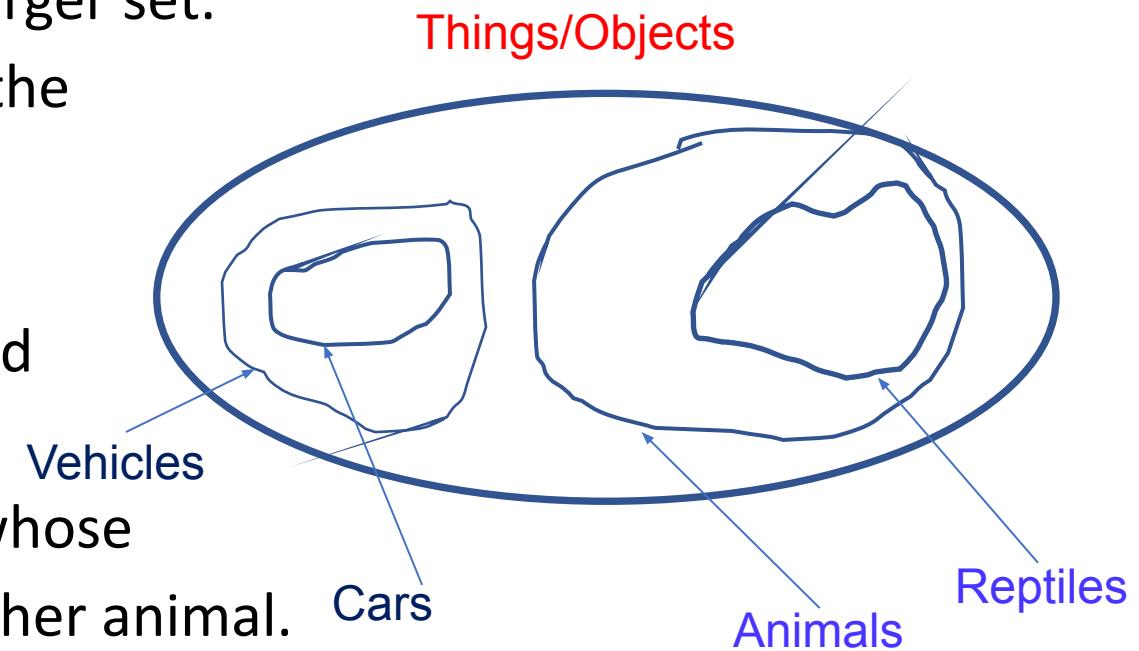


- Concept
- Instance Space
- Hypothesis
- Hypothesis Space
- Inductive Bias
- Satisfied Hypothesis
- Consistent Hypothesis
- Most general Hypothesis ?(+ve)
- Most Specific Hypothesis  $\varphi$  (-ve)
- Find-S

- What is a Concept



- Is a subset of objects/events defined over a larger set.
- Eg. The concept of **reptile** is the subset of all the **animals** that belong to the category of **reptile**.
- A concept is a boolean valued function defined over a larger set.
- Eg. A function **f** defined over all **Animals A**, whose value is true for **reptiles r** and false for every other animal.  
 $f(A,r) = \text{true or false}$



# MACHINE LEARNING

## Concept Learning

So, Basically what a concept does is it labels objects in one of two types (**binary Concept**)



- What is a **concept**?

In machine learning, models need to acquire **an ability** to take **an object** and say if **it belongs** to a particular **concept or not**.

Concept could be *some kind of notion* or something that defines an object

Something that has **attribute(s) of chair** so it will belong to the **concept Chair**.

Something not having **attribute(s) of chair** so it **does not belong** to the concept **Chair**.

Consider a data object x and concept C

Lets have **class label** for each case say, **class label C1 = 1 if  $x \in C$**  else **0 if  $x \notin C$** .

Data object x	Concept C	<u>label</u>
Either $x \in$ concept C		1
or		
$x \notin$ concept C		0



# MACHINE LEARNING

## Concept Learning

So, Basically what a concept does is it labels objects in one of two types (**binary Concept**)



- What is a **concept**?

In machine learning, models need to acquire **an ability** to take **an object** and say if **it belongs** to a particular **concept or not**.

Consider a data object x and concept C

Lets have **class label** for each case  
say, class label  $C_1 = 1$  if  $x \in C$   
else 0 if  $x \notin C$ .

	<u>label</u>
Either $x \in$ concept C	1
or $x \notin$ concept C	0

Concept could be *some kind of notion* or something that defines an object

Something that has **attribute(s)** of Aaloo Paratha so it will belong to the **concept Aaloo Paratha**.

Something not having **attribute(s)** of Aaloo Paratha so it **does not belong to** the concept **Aaloo Paratha**.

**CONCEPT?  
YES**



- A ML model must *learn the concept* so as to *automatically infer* by looking into *the feature space* and learn the concept in a *general way*.
- By looking into *the feature space* and using some *language model* eg. A conjunctive model or disjunctive model or linear separator model, model will learn.
- Given a set of examples *labeled as belonging or not-belonging* to a concept, concept learning consists of *automatically inferring the general definition of this concept*.
- In other words, concept learning consists of *approximating a Boolean-valued function* from training examples of input (values of attributes) & output( target class).

# MACHINE LEARNING

## Concept Learning

Consider an object x that is defined by 4 features namely **Shape, Texture, Size, Color**

X={ (oval, soft, large, dark), .....  
(circular, hard, small, light)}

Let's say each attribute takes only two values as below

Shape	Texture	Size	Color
Oval/Circular	Soft/Hard	Large/Small	Dark/Light

Each **object x** is defined by **specific features** through which we can decide if it belongs to the **Concept**



Now machine learns this unknown concept and once model learns **then it can label any new data** accordingly....this the **Concept learning task**

**Concept** takes each object and assigns it a label

Object_Id	Shape	Texture	Size	Color	Label
Obj_1	Oval	soft	Large	Dark	Aaloo
Obj_2	Circular	hard	Large	Light	~Aaloo
Obj_3	Oval	soft	Small	Dark	Aaloo
Obj_4	Circular	hard	Small	Light	~Aaloo

Data object x	Concept C	<u>label</u>
Either $x \in$ concept C	1	
or		
$x \notin$ concept C	0	

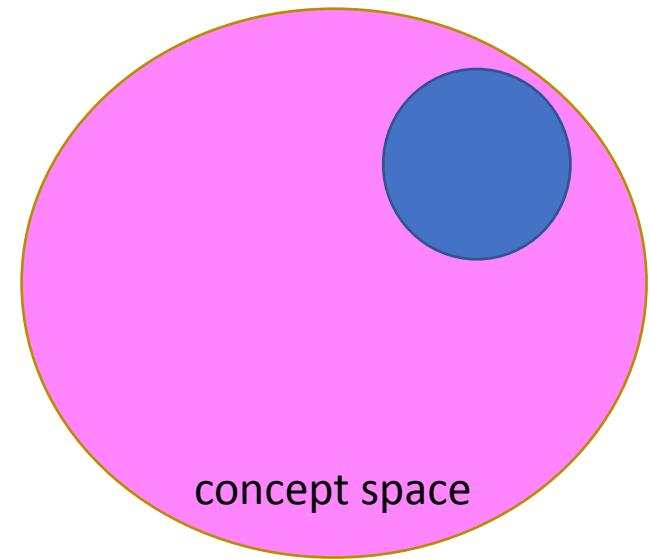
Machine learns this unknown concept and once model learns then it can label any new data accordingly....this the Concept learning task

$x$  is a dataset, that can be defined by value of the attributes for multiple instances.

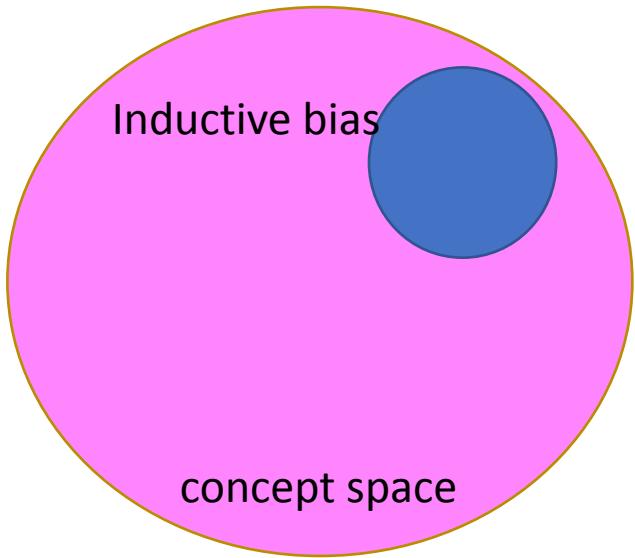
## Concept Learning: Instance Space, Hypothesis space, Hypothesis

- The set of all instances, not just instances that are available as training data forms the **instance space**.
- Given the instances in the previous slide, we can make any claim about a concept.
- Consider the following claims:
- H1: “ If shape is oval and Texture is soft” then “it is a Aaloo Prantha”
- H2: “ If shape is oval and Color is Dark” then “ it is a Aaloo Prantha”
- H3: “ If shape is circular and size is small” then “ it is a Aaloo Prantha”
- A **claim made about the concept is called a hypothesis**. Claims could be valid or invalid.
- The set of all hypothesis is called a **hypothesis space**.

Object_Id	Shape	Texture	Size	Color	Label
Obj_1	Oval	soft	Large	Dark	Aaloo
Obj_2	Circular	hard	Large	Light	~Aaloo
Obj_3	Oval	soft	Small	Dark	Aaloo
Obj_4	Circular	hard	Small	Light	~Aaloo



- If 'd' binary attributes then there are  $2^d$  instances.
  - If these instances are to split in binary way then then total no. of concepts is  $2^{2^d}$ .
  - Like in previous example there are 4 attributes namely shape, texture, color and size, so there are  $2^4=16$  subsets/instances of this dataset and  $2^{16}=65,536$  concepts.( to divide 16 instances into 2 sets)
  - The ML task is to look into this space and choosing the right concept, which is a challenging task since for a 4 attribute object we have space with 65,536 concepts
- In order to divide 16 instances into 2 datasets, each instance has 2 choices, i.e.  
 $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$   
 $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$   
 $2 \times 2 = 2^{16}$

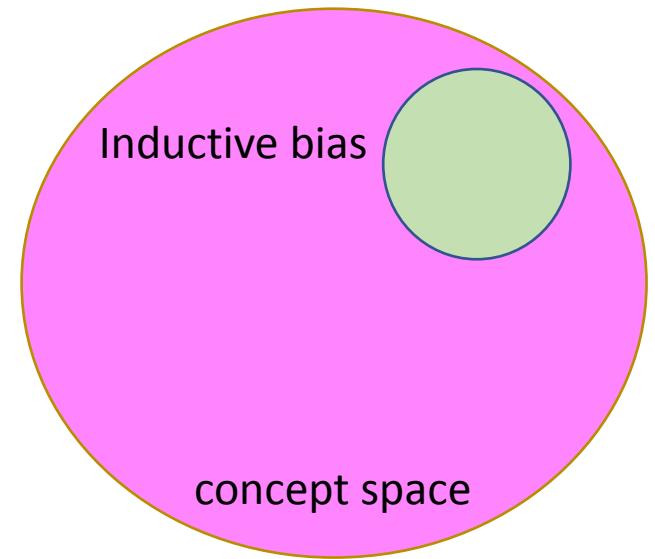


Now the task is how to make the machine learn this concept

- To reduce the concept space, introduce the **bias**.
- Bias can be introduced by restricting hypothesis that is by **making some assumption about the nature of the concept**, like hypothesis will be formed only by
  1. Using Conjunctions of the features
  2. No disjunction is allowed
- This reduction of the concept space is called as **inductive bias**.

Eg. Consider the following 3 binary attributes, so the no. of instances or dataset will be  $2^3=8$  and no. of concepts = $2^8$

Corresponding to each instance there are two possibilities for target label either 1 or 0. So for each instance there are 2 class possibilities. Hence, there are  $2^8=256$  possible *concepts.*



circular ^ dark

$$\begin{aligned}X_1 &: (1, 0, 0, 1) \\X_2 &: (0, 0, 0, 1)\end{aligned}$$

If both of the features are true for a data object then the object belongs to the *Concept*.

Consider the conjunctive concept as a vector  
Consider this concept

So we have 16 instances.

Conjunctive concept means a mathematical function which take conjunction (logical AND) of few of the features( basically a subset of all possible feature)

	0	1
Shape	oval	circular
Texture	Soft	Hard
Size	small	large
Color	light	dark

\_  $\wedge$  \_  $\wedge$  \_  $\wedge$  \_

circular  $\wedge$  dark can be represented like this

circular  $\wedge$  \_ ? \_ ? \_  $\wedge$  dark

? means any acceptable value there is true

?  $\wedge$  ?  $\wedge$  ?  $\wedge$  ? is a concept that accepts everything ,we call it accept all

$\emptyset$  is a concept that rejects everything ,we call it reject all

## Most General and Most Specific hypothesis

---

? $\wedge$ ? $\wedge$ ?  $\wedge$ ?

This is called **Most General hypothesis**, that accepts any test case: All positive class

$\phi \wedge \phi \wedge \phi \wedge \phi$

This is called **Most Specific hypothesis**, that will not accept any test case: All negative class

—  $\wedge$  —  $\wedge$  —  $\wedge$  —

For every position we have **3 choices**  
that is **2 binary values and ?**

Also we have **one concept** of  $\emptyset$

So there are  $(3 \times 3 \times 3 \times 3) + 1 = 82$   
**conjunctive concepts.**

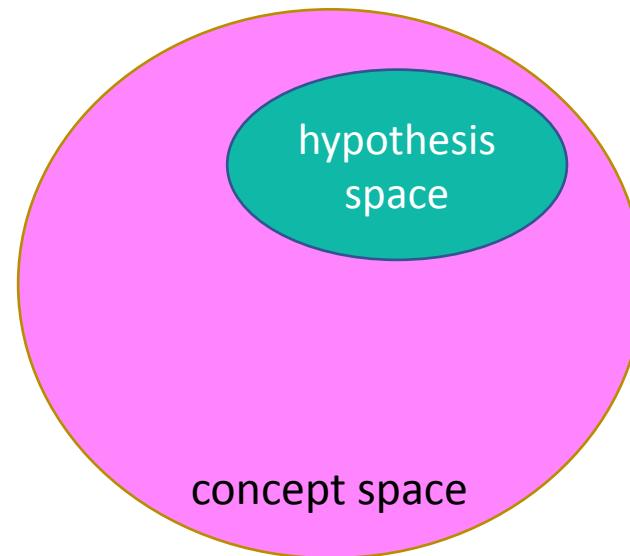
A total of **65,536 concepts** in the  
concept space is reduced to **82**  
**concepts** in the conjunctive hypothesis  
space

i.e This way the **search space** is shrunk.

How many conjunctive concepts can be there for  
an object with 4 binary attributes?

So, In general if we have  $d$  binary  
features then we will have  $3^d + 1$   
conjunctive concepts

This shrunk space is what we  
call as ***hypothesis space***



- Consider the following dataset:

Is foodie	Has free time	Has money	Has work to be finished	Has someone's company	<u>Will go to Restaurant?</u>
Y/N	Y/N	Y/N	Y/N	Y/N	T/F

- In how many ways can 32 instances having all binary features be split into 2 concepts?

- Consider the following dataset:

Is foodie	Has free time	Has money	Has work to be finished	Has someone's company	<u>Will go to Restaurant?</u>
Y/N	Y/N	Y/N	Y/N	Y/N	T/F

- In how many ways can 32 instances having all binary features be split into 2 concepts?
- $2^{32}$  ( This is the no. of ways in which one can divide the dataset into 2 subsets)

It can be reduced to **244** concepts in conjunctive hypothesis space

### Example of Play Tennis Concept

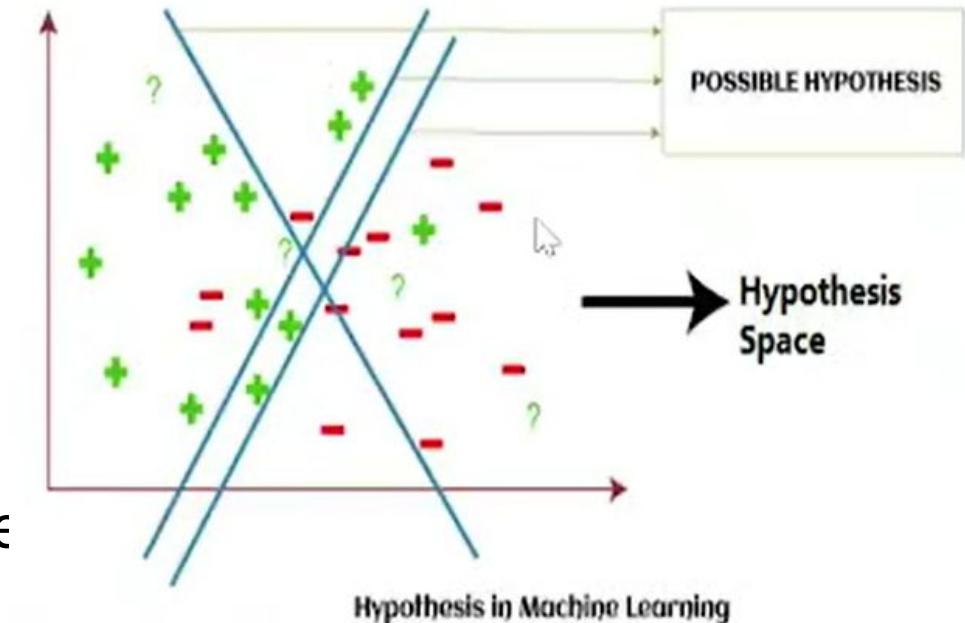
Day	<i>Outlook</i>	<i>Temperature</i>	<i>Humidity</i>	<i>Wind</i>	<i>PlayTennis</i>
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- A **supervised learning m/c** can be thought of as a m/c that explores **hypothesis space**.
- Each setting of the parameters in the m/c is a different hypothesis about the function that maps i/p vectors to o/p vectors.
- So, Hypothesis space is a space of all **valid/invalid hypothesis** that is described using chosen features and chosen language  
$$H: h \in H$$
- Given the training set, learning algorithm comes up with a hypothesis, its choice of **hypothesis** depends on
  - i) the data
  - ii) the restrictions being put on it.and also depends on

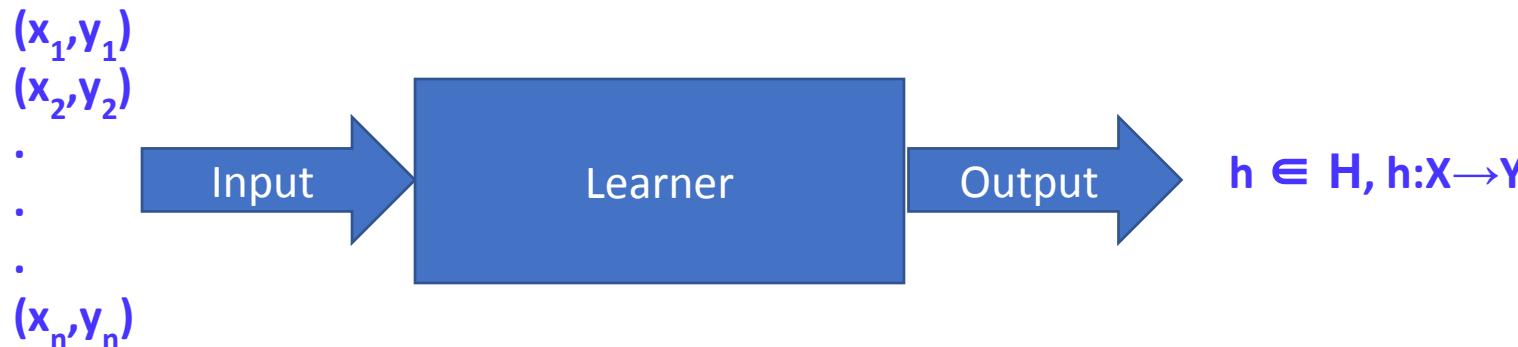
- Consider
  - ❖ *Example(x,y)*: An example with features  $x$  and label  $y=f(x)$
  - ❖ *Training Data D*: Collection of all examples observed by the learning algorithm
  - ❖ *Instance Space X*: set of all possible instances/objects with all possible values of features.
  - ❖ *Concept C*: Subset of objects from  $X$ (  $C$  is unknown)  $C \subseteq X$ .( Concept  $C$  defines the positive concept)
  - ❖ *Training function f*: It maps each instance  $x \in X$  to target class  $y \in Y$ .( $f$  is a function used to describe  $C$ ).
  - ❖ We would try to come up with  $h \in H$  that approximates  $f$

## Hypothesis Space: An example

- Consider a classification problem,  $f(x)$  ( discrete)
- Say examples are  $(x,y)$ .
- Say we have to find the class of test instance ‘?’
- Say a line  $l_1$  is the function that approximates the prediction. This  $l_1$  is the hypothesis for the prediction.
- Other valid hypothesis could be  $l_2$  or  $l_3$  that has the capacity to predict the test data ‘?’
- **Set of all such hypotheses is hypothesis space.**
- Based on features and language, the hypothesis space can be defined.



- $x$ : descriptions
- $Y$ : predictions
- $H$ : hypothesis space, the set of all possible hypothesis
- $h$ : target hypothesis



- All hypothesis whether valid or invalid forms the **hypothesis space**.
- A hypothesis  $h(x)$  is said to be **satisfied**, if there is *atleast one set of features which are present there in the training data set.*
- A hypothesis is said to be **consistent** when
$$\forall i c(x_i)=h(x_i)$$
 $c(x_i)$  is the true concept and  $h(x_i)$  is the predicted concept that learner has learnt.

( *For each of the training set, hypothesis learner is able to correctly find out all the concepts that exist in training set.*)



- A **satisfied hypothesis** accurately captures *the underlying pattern in the training data but might still make some errors* (i.e., it's a good approximation of the true concept).
- A **consistent hypothesis** is a subset of a satisfied hypothesis; *it is one that fits the training data perfectly without any errors* (i.e., it matches the true concept for the training data).



**Concept-** Days on which person enjoys sports

**Attributes-**

**Sky**-sunny,rainy

**Temp**-warm,cold

**Humidity**-Normal,High

**Wind**- strong,weak

**Water**-warm,cool

**Forecast**-same,change

With all this set, we will move forward to the first algorithm that is Find S algorithm

We will learn this algorithm with a concept search problem

Let us see the problem description

## Find S algorithm

1. start with  $h = \emptyset$
2. use next instance  $\{x, c(x)\}$
3. if  $c(x) = 0$ , go to step 2
4.  $h \leftarrow h \wedge x$  (pairwise-and)
5. if more examples .Go to step 2
6. stop

Let us look at the pseudo code of this algorithm

### Pairwise -and rules

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_x \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

# MACHINE LEARNING

## Find S algorithm-Problem

**step1:** start with  $h=\emptyset$

$h=\{\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset\}$  REJECT ALL

**step2:** use next input  $\{x, c(x)\}$

$x=\{\text{sunny, warm, normal, strong, warm, same}\}$

$c(x)=\text{yes}==1$

**step3:** if  $c(x)=0$ , go to step 2

**step4:**  $h <--- h \wedge x$  (pairwise-and)

$h_0 = h_0 \wedge x$

$h_0 = \{\text{sunny, warm, normal, strong, warm, same}\}$

**step5:** if more examples . Go to step 2

$h(1)=\text{warm}$   
 $h(2)=\text{normal}$   
 $h(3)=\text{strong}$   
 $h(4)=\text{warm}$   
 $h(5)=\text{same}$

Let us take the concept of step by step by step rule we need to learn this concept with the training data below

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_x \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

# MACHINE LEARNING

## Find S algorithm-Problem

**step2:** next input  $\{x, c(x)\}$

$x = \{\text{sunny, warm, high, strong, warm, same}\}$   
 $c(x) = \text{yes} == 1$

**step3:** if  $c(x)=0$ , go to step 2

**step4:**  $h \leftarrow h \wedge x$  (pairwise-and)

$$h_0 = h_0 \wedge x$$

$h_0 = \{\text{sunny, warm, ?, strong, warm, same}\}$

**step5:** if more examples

Go to step 2

$h(0) = \text{sunny}$

$h(1) = \text{warm}$

$h(2) = ?$

$h(3) = \text{strong}$

$h(4) = \text{warm}$

$h(5) = \text{same}$

$h(0) = \text{sunny}$   
 $h(1) = \text{warm}$   
 $h(2) = \text{normal}$   
 $h(3) = \text{strong}$   
 $h(4) = \text{warm}$   
 $h(5) = \text{same}$

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_x \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

# MACHINE LEARNING

## Find S algorithm-Problem

step2: next input  $\{x, c(x)\}$

$x=\{\text{rainy}, \text{cold}, \text{high}, \text{strong}, \text{warm}, \text{change}\}$   
 $c(x)=\text{yes}==0$

step3: if  $c(x)=0$  ,go to step 2

$h(0)=\text{sunny}$   
 $h(1)=\text{warm}$   
 $h(2)=?$   
 $h(3)=\text{strong}$   
 $h(4)=\text{warm}$   
 $h(5)=\text{same}$

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

# MACHINE LEARNING

## Find S algorithm-Problem

**step2:** next input  $\{x, c(x)\}$

$x = \{\text{sunny, warm, high, strong, cool, same}\}$   
 $c(x) = \text{yes} == 1$

**step3:** if  $c(x) = 0$ , go to step 2

**step4:**  $h \leftarrow h \wedge x$  (pairwise-and)

$$h_0 = h_0 \wedge x$$

$h_0 = \{\text{sunny, warm, ?, strong, ?, same}\}$

**step5:** if more examples Go to step 2

**step6:** stop

$h(0) = \text{sunny}$   
 $h(1) = \text{warm}$   
 $h(2) = ?$   
 $h(3) = \text{strong}$   
 $h(4) = ?$   
 $h(5) = \text{same}$

$h(0) = \text{sunny}$   
 $h(1) = \text{warm}$   
 $h(2) = ?$   
 $h(3) = \text{strong}$   
 $h(4) = \text{warm}$   
 $h(5) = \text{same}$

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

$$a_h \wedge a_x = \begin{cases} a_x & : \text{if } a_h = \emptyset \\ a_x & : \text{if } a_h = a_h \\ ? & : \text{if } a_h \neq a_x \\ ? & : \text{if } a_h = ? \end{cases}$$

$C = \{\text{sunny, warm, ?, strong, ?, same}\}$

- Suppose now on a given day we get the following data

$x = \{\text{sunny, warm, high, strong, warm, same}\}$   
then  $c(x) = 1$

- We use the concept learnt to predict the label that is if the person will play a sport or no.

- We can pass the data through our concept and predict the label

We are done with all the training data set and we give our final hypothesis as our concept that our machine has learned to label other data sets

sky	temp	humidity	wind	water	forecast	enjoy
sunny	warm	normal	strong	warm	same	yes
sunny	warm	high	strong	warm	same	yes
rainy	cold	high	strong	warm	change	no
sunny	warm	high	strong	cool	same	yes

# MACHINE LEARNING

## Concept Learning: Instance Space, Hypothesis space, Hypothesis

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Distinct Hypothesis?

Syntactically distinct hypothesis?

Semantically distinct hypothesis?

# MACHINE LEARNING

## Concept Learning: Instance Space, Hypothesis space, Hypothesis

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

TABLE 2.1

Positive and negative training examples for the target concept *EnjoySport*.

Sky/Outlook	Air Temp	Humidity	Wind	Water	Forecast
3 choices	2 choices	2 choices	2 choices	2 choices	2 choices

The instance space will have

$$= 3 \times 2 \times 2 \times 2 \times 2 \times 2 = 96 \text{ distinct instances}$$

There will be

$$= 5 \times 4 \times 4 \times 4 \times 4 \times 4 = 5120 \text{ Syntactically distinct hypothesis in } H$$

And there will be

$$= (4 \times 3 \times 3 \times 3 \times 3 \times 3) + 1 = 973 \text{ Semantically distinct hypothesis in } H$$

- A **hypothesis** is said to be **consistent** wrt to training data set **if it classifies all the objects of training data set to their corresponding classes**

X	C(X)
$x_1$	$c(x_1)$
$x_2$	$c(x_2)$
so on.....	

- A hypothesis is said to be **consistent** if  $h(x_i) = C(x_i)$
- At any given point let  $H$  be our hypothesis space so a **version space is a subset of  $H$**  but all concepts in version space VS are consistent wrt to training set

i.e  $VS = \{h : h \in H\}$  and  $h$  is consistent with  $D_{\text{training}}$

This ensures our algorithm learns only the best hypothesis.

- A version space VS is **a set of all consistent hypothesis.**
- Sometimes having *a less training data for an object with many attributes the hypothesis space may never find the right concept* that is to be learned and may predict wrong.
- A better version of find S algorithm is candidate elimination algorithm which can be learnt by yourself for deeper understanding

## Concept Learning: Instance Space, Hypothesis space, Hypothesis

• • •

Represent this problem as a ML problem in terms of <P,T,E>. Apply Find-s algorithm to it to learn the concept.

Learning the concept of "Japanese Economy Car"

**Features:** ( Country of Origin, Manufacturer, Color, Decade, Type )

<i>Origin</i>	<i>Manufacturer</i>	<i>Color</i>	<i>Decade</i>	<i>Type</i>	<i>Example Type</i>
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive

## Inductive Bias

---

**inductive bias** refers to the assumptions or preconceptions that a **model or algorithm** makes about the **underlying distribution of data**. These biases can influence the model's ability to learn from a given dataset and can affect the performance of the model on new, unseen data.

It reflects the prior knowledge or beliefs that guide the learning process. Inductive bias helps the model generalize from the training data to unseen data by favoring certain hypotheses or solutions over others.

An example :-

Consider a spam email classifier. The inductive bias could be that shorter emails with a lot of capitalized words and certain keywords are more likely to be spam. This bias guides the classifier to make predictions based on these characteristics.

## Restrictive Bias

---

**Restrictive bias** refers to the limitations or constraints imposed on a **model's learning capacity**. These constraints can be **explicit**, such as limiting the complexity of the model architecture, or **implicit**, such as using a specific optimization algorithm that might restrict the types of functions the model can learn.

Restrictive bias can help prevent overfitting and enhance model interpretability, but it can also limit the model's ability to capture intricate patterns in the data.

### An example:-

In the context of image recognition, using a simple linear regression imposes a restrictive bias. This bias assumes that the relationship between pixel values and the image's class is linear, which might not capture the intricate features present in images. choosing a linear regression model imposes a restrictive bias by assuming that the relationship between input features and the target variable is linear. This could be a limitation if the true relationship is more complex.

## A Quick Question: Concept Vs Hypothesis?? Same or different??

---

### Concept

- Refers to the *true underlying pattern, relationship, or mapping that exists in the data.*
- Its something that the machine learning algorithm aims to learn or approximate. The concept is what we ultimately want the algorithm to discover or capture.
- The ML algorithm aims to approximate it as closely as possible through the hypothesis it generates.
- The decision boundary of a concept is true, ideal boundary that separates different classes in a dataset.



### Hypothesis

- *Is a candidate explanation or model that the machine learning algorithm proposes based on the observed data.*
- ***It is a simplified representation of the concept*** that the algorithm believes might explain the underlying pattern in the data.
- The learned boundary that the algorithm proposes based on the training data to approximate the true concept's boundary

## MACHINE LEARNING

Spam email filtering: This example to be referred when you know about decision boundaries

---

- Concept here refers to **the true underlying pattern** that distinguishes between spam emails and legitimate emails.
- It represents the **ideal decision boundary** that perfectly separates spam emails from non-spam (legitimate) emails based on their features. The concept is what we want our machine learning algorithm to learn or approximate.
- The hypothesis is the **decision boundary** that the machine learning algorithm generates based on the training data.
- ***It is a simplified representation of the true concept's boundary*** that the algorithm believes will effectively separate spam from legitimate emails.



- Concept
- Instance Space
- Hypothesis
- Hypothesis Space
- Inductive Bias
- Satisfied Hypothesis
- Consistent Hypothesis
- Most general Hypothesis ?(+ve)
- Most Specific Hypothesis  $\varphi$  (-ve)
- Find-S
- Version space



THANK YOU

---

Dr. Pooja Agarwal  
[poojaagarwal@pes.edu](mailto:poojaagarwal@pes.edu)

# MACHINE LEARNING (UE22CS352A)



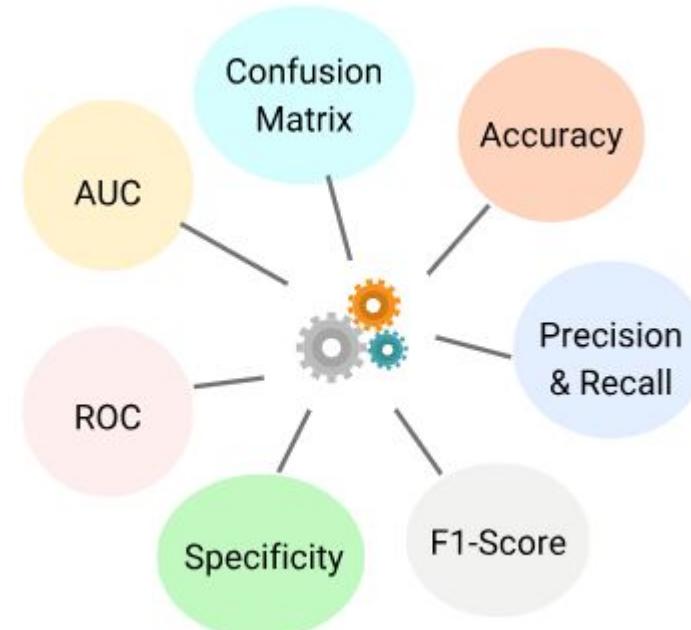
## Performance Metrics

---

**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

1. Accuracy
2. Precision
3. Recall
4. Specificity
5. Receiver Operating Characteristics ( ROC)
6. Area Under Curve (AUC)

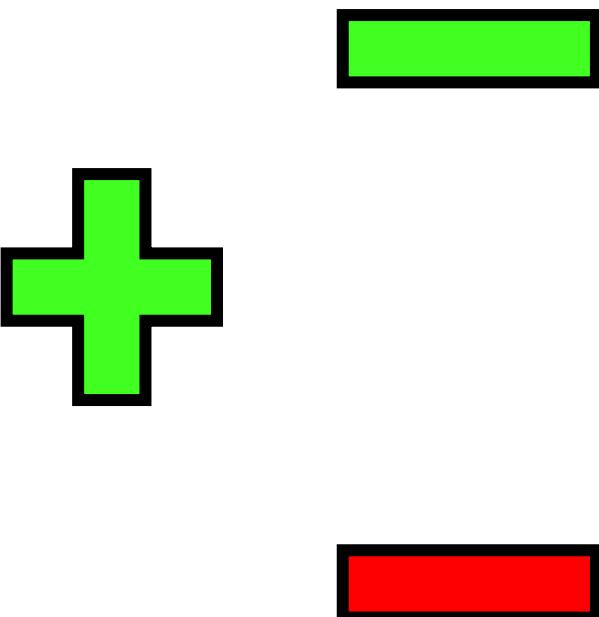
After the initial feature engineering and processing, implementing a model on test dataset will get some output in terms of a class or probability.



1. True Positive – truly Predicted +ve
2. True Negatives – truly Predicted -ve
3. False Positive - Falsely predicted as +ve
4. False Negatives – Falsely predicted as -ve

Before moving on to study the following, have a look at the following, see what is a confusion matrix.

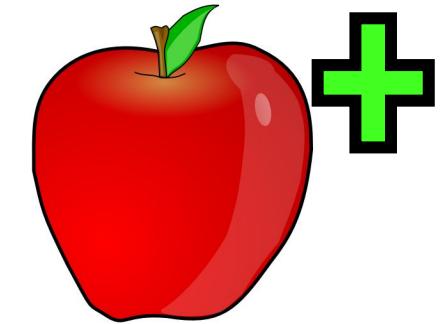
It is a simple way to lay out ,how many predicted categories or classes were correctly predicted and how many were not.



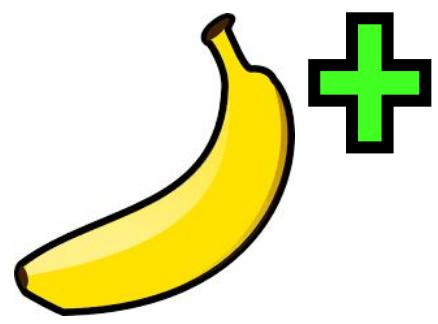
		Predicted Class	
		Spam	Non-Spam
Actual Class	Spam	TP=45	FN=20
	Non-Spam	FP=5	TN=30

# MACHINE LEARNING

## Confusion Matrix



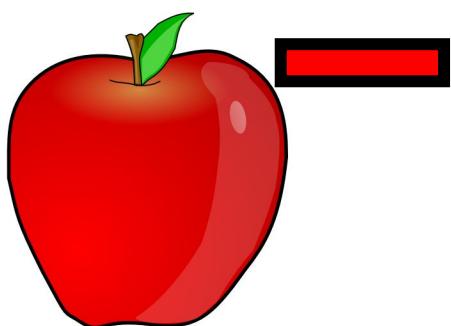
class A correctly predicted as class A



class B correctly predicted as class B

suppose our classification model has two class class A(**apple**) and class B (**all other fruits**)

essentially the confusion matrix is keeping track of

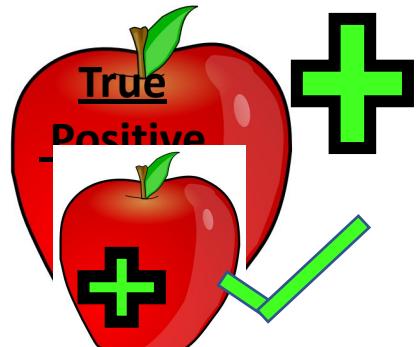


class A incorrectly predicted as class B

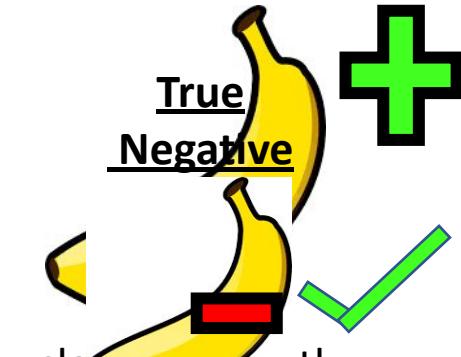


class B incorrectly predicted as class A

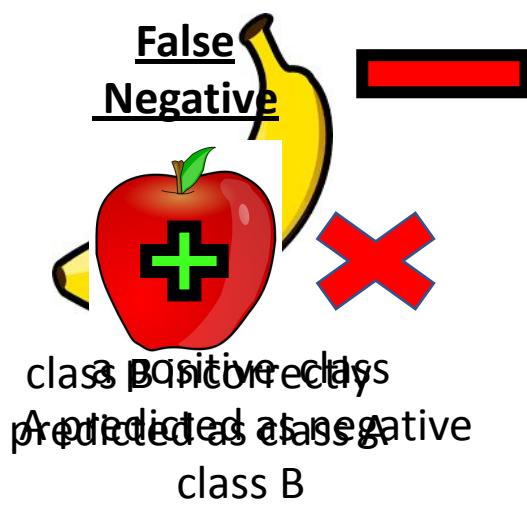
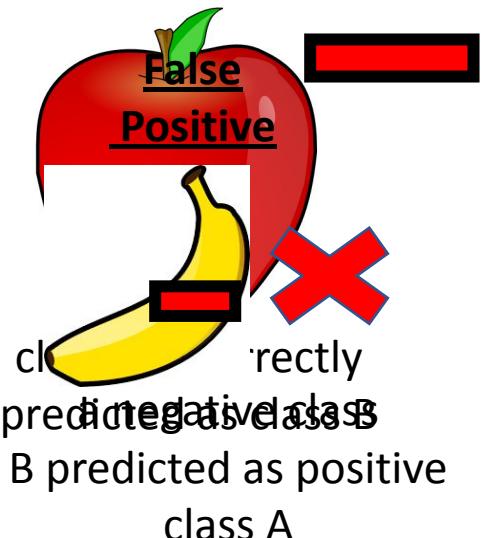
1. True Positive – truly Predicted +ve
2. True Negatives – truly Predicted -ve
3. False Positive - Falsely predicted as +ve
4. False Negatives – Falsely predicted as -ve



class A correctly predicted as class A  
a positive class  
A predicted as class A



class B correctly predicted as class B  
a negative class  
B predicted as class B



the proper classification track negative as follows A (apple) and class B (all other fruits)

essentially the confusion matrix is keeping track of our false positive and false negative are as follows

## GOAL

As many predictions as possible  
More true than false

- Consider a situation,
  - **Concept:** Cancer? If yes, then +ve class otherwise –ve class
  - **FP:** A blood test/ MRI report indicates, patient is diagnosed with cancer wherein he actually doesn't have.
  - **FN:** if patient is not diagnosed with Cancer, wherein he actually has.

# MACHINE LEARNING

## Confusion Matrix

along x axis we represent predicted values and along y axis we represent actual values

N: 8000	Predicted: CLASS A	Predicted: CLASS B
Actual: CLASS A	TP 6088	FN 12
Actual: CLASS B	FP 12	TN 1888

### NOTE:

FP and FN are **type 1** and **type 2** error respectively (same as what you learn't in SDS course in 3rd sem)

Confusion matrix data set of 8000 objects of the set of data has 6100 pairs from class A and 1900 from class B that we have made  $1888+6088=7976$  +ve and -ve class

with predictions from our model as below  
and total wrong predictions  $8+16=24$

N	class A	class B
8000	6100	1900

- 6088 of the data object were correctly predicted positive class A as in the data set
- 1888 of the data object were correctly predicted as negative class B as in the data set

## Accuracy

Accuracy is given by,

$$\text{Accuracy} = \frac{TP + TN}{(TP + TN + FP + FN)}$$

**Accuracy** in classification problems is the number of correct predictions made by the model over all kinds predictions made. **Accuracy** is generally a good measure when the target variable classes are nearly balanced.

- Consider the following scenario:

		Predicted		
		1.	0	
Actual	1	TP(13)	FN(17)	30
	0	FP(25)	TN(80)	105
		38.	97	<b>135</b>

$$\text{Accuracy} = (13+80)/135 = 68.88\%$$

If TP<FP and we change the classification rule to always classify -ve class, the accuracy will increase and this is called Accuracy Paradox

TP(0)	FN(30)
FP(0)	TN(105)

$$\text{Accuracy} = 105/135 = 77.77\%$$

This classifier has no predictive power but still has approx. 78% accuracy !!!

Accuracy Paradox.

- Accuracy measure should be used when the training dataset is balanced ie the no. of instances belonging to 2 classes are reasonably representing the classes.
- If the data is skewed ie instances wrt on class are much more than the other then Accuracy is not an appropriate measure.

- Consider the following scenario:

		Predicted		30	105	$\text{Accuracy} = (13+80)/135=68.88\%$
		1.	0			
Actual	1	TP(13)	FN(17)			
	0	FP(25)	TN(80)			
		38.	97	<b>135</b>		

TP(0)	FN(30)
FP(0)	TN(105)

$$\text{Accuracy} = 105/135=77.77\%$$

This classifier has no predictive power  
but still has approx. 78% accuracy !!!

- Accuracy measure should be used when the training dataset is balanced ie the no. of instances belonging to 2 classes are reasonably representing the classes.
- If the data is skewed ie instances wrt on class are much more than the other then Accuracy is not an appropriate measure.



		Diagnosis	
		Diagnosed sick	Diagnosed Healthy
Patients	Sick	1000	200
	Healthy	800	8000

Accuracy: Out of all the patients, how many did we classify correctly?

$$\text{Accuracy} = \frac{1,000 + 8,000}{1,000 + 200 + 800 + 8,000} = 90\%$$

## Accuracy



		Folder
		Spam Folder
E-mail	Spam	100
	Not spam	30
		Inbox
		170
		700

Accuracy: Out of all the e-mails, how many did we classify correctly?

$$\begin{aligned} \text{Accuracy} &= \frac{100 + 700}{100 + 170 + 30 + 700} \\ &= 80\% \end{aligned}$$

Precision( Positive Predictive value) is given by,

$$\text{Precision} = \frac{TP}{(TP + FP)}$$

Recall ( True positive rate)is given by,

$$\text{Recall} = \frac{TP}{(TP + FN)}$$

Precision: How many +ve cases did we catch?

Recall: How many did we miss? (sensitivity)

**Sensitivity** tells us what proportion of the positive class got correctly classified.

False Negative Rate (FNR) tells us what proportion of the positive class got incorrectly classified by the classifier.

$$FNR = \frac{FN}{TP + FN}$$

A higher TPR and a lower FNR is desirable since we want to correctly classify the positive class.

- ✓ Used widely in cases where **cost of false positives are high.**

### Example:

- ~ In a medical diagnosis scenario, high precision is desirable.
- ~ Avoid misdiagnosing healthy patient as having disease. (false positive)

- ✓ Focusses on reducing **Type – 1 Error (FP).**

### What are Type – 1 errors?

- ~ Error that occurs when a model wrongly detects a condition or event that is not present, a false alarm in simple words.
- ~ Eg. Labeling a non-cancerous patient as cancerous



		Diagnosis	
		Diagnosed sick	Diagnosed Healthy
Patients	Sick	1000	200
	Healthy	800	8000

Precision: Out of the patients we diagnosed with an illness, how many did we classify correctly?

$$\text{Precision} = \frac{1,000}{1,000 + 800} = 55.7\%$$



E-mail	Folder	Spam Folder	Inbox
Spam	100	170	
Not spam	30		700

Precision: Out of all the e-mails, sent to the spam inbox, how many were actually spam?

$$\text{Precision} = \frac{100}{100 + 30} = 76.9\%$$

- ✓ Used widely in cases where **cost of false negatives are high**.

### Example:

- ~ In medical testing, high recall is desirable, to ensure as few cases of a disease are missed as possible.

- ✓ Focuses on reducing **Type – 2 Error (FN)**.

### What are Type – 2 errors?

- ~ Error that occurs when a model fails to detect a condition or event that is actually present.
- ~ Eg. test fails to diagnose a sick person correctly and mistakenly indicates that a person is healthy.



		Diagnosis	
		Diagnosed sick	Diagnosed Healthy
Patients	Sick	1000	200
	Healthy	800	8000

Recall: Out of the sick patients, how many did we correctly diagnose as sick?

$$\text{Recall} = \frac{1,000}{1,000 + 200} = 83.3\%$$

# MACHINE LEARNING

## Metrics calculations

N: 200	Actual: CLASS A	Actual: CLASS B	CLASSIFI- CATION OVERALL	
Predicted: CLASS A	TP 60	FP 40	100	precision
Predicted: CLASS B	FN 70	TN 30	100	
TRUTH OVERALL	130	70	200	
	recall	specificity		accuracy

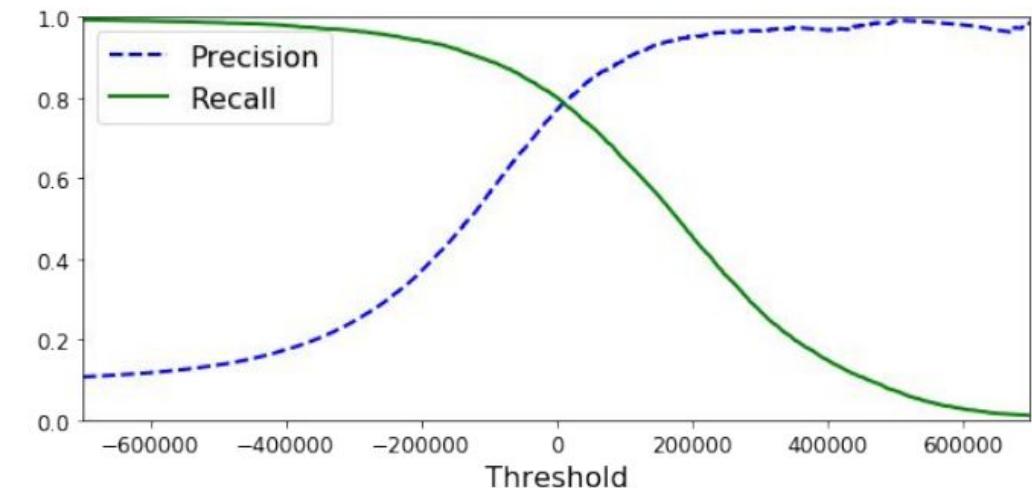
consider the following confusion matrix  
Let us calculate all the metrics we discussed till now

$$\begin{array}{r} 0.42 \\ + \\ \hline & = 0.46 \\ & & 0.6 \end{array}$$

## Precision Recall Trade off

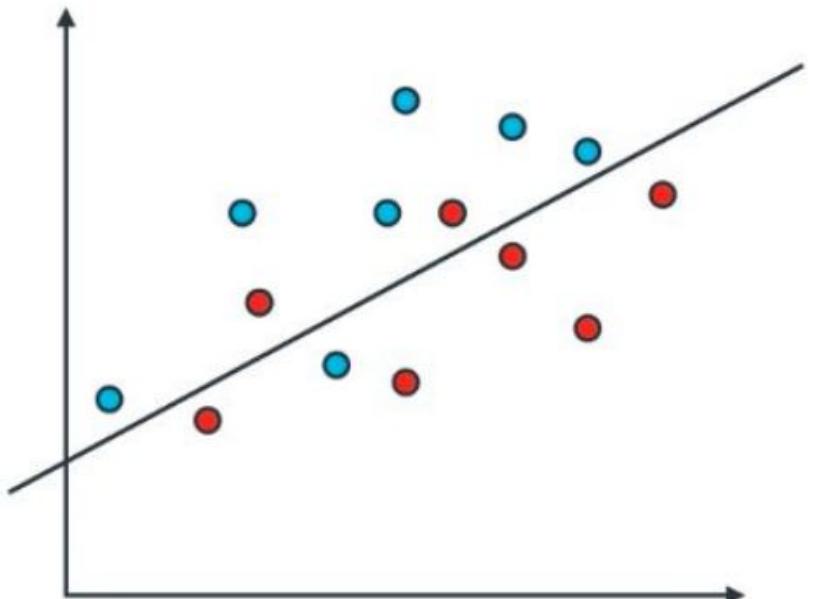
- Improving precision typically reduces recall and vice versa.
- To fully evaluate effectiveness of a model, examine **both** precision and recall, need for a combined metric.

Precision	Recall
Measures the reliability of the model in classifying the model as positive.	Measures model's ability to detect positive samples.



## Question

Find the number of TP, FP, TN, FN cases for the given graph, where blue is positive and red is negative.

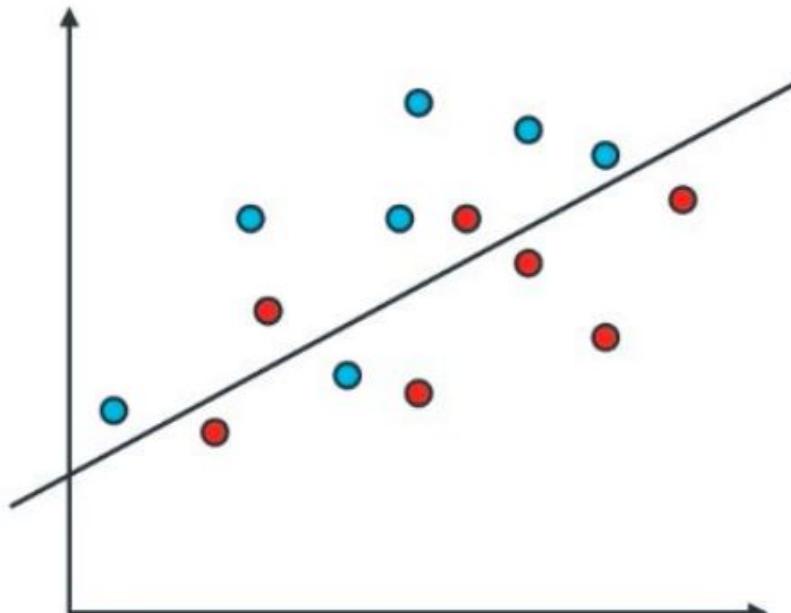


		Prediction	
		Guessed Positive	Guessed Negative
Data	Positive		
	Negative		

# MACHINE LEARNING

## Question - SOLUTION

Find the number of TP, FP, TN, FN cases for the given graph, where blue is positive and red is negative.



		Prediction	
		Guessed Positive	Guessed Negative
Data	Positive	6	1
	True positives	False Negatives	
	Negative	2	5
	False Positives	True Negatives	

## Specificity (Fallout) and F1 score

Specificity ( True Negative rate) is given by,

$$\text{Specificity} = \frac{TN}{(TN + FP)} = 1 - FPR$$

F1 score is given by,

$$\begin{aligned}\text{F1 score} &= \frac{2 * (\text{recall} * \text{precision})}{(\text{recall} + \text{precision})} \\ &= 2TP / (2TP + FP + FN)\end{aligned}$$

Specificity : How many -ve cases did we catch?

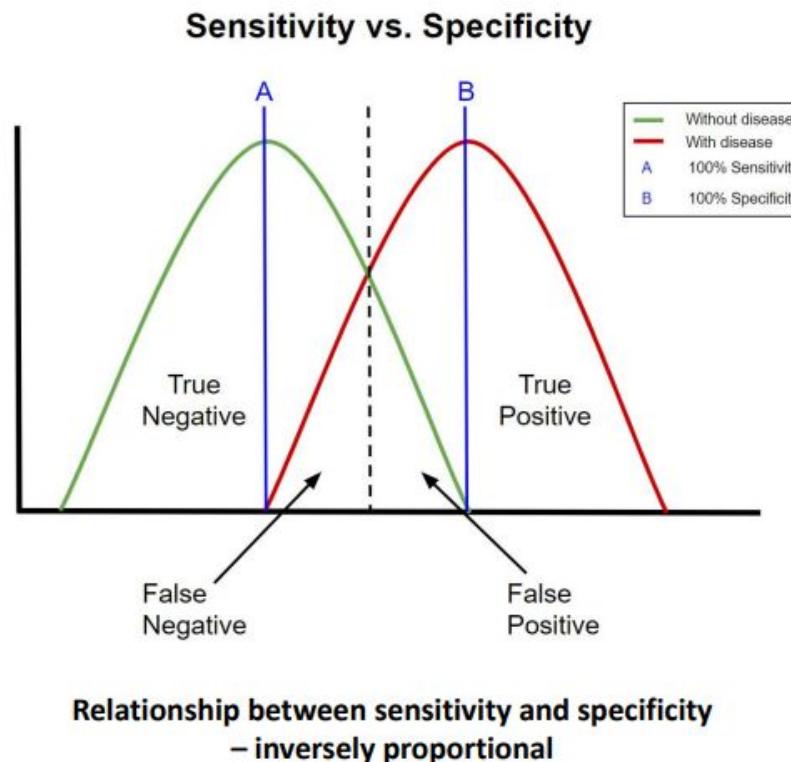
F1-score: The harmonic mean of precision and recall.

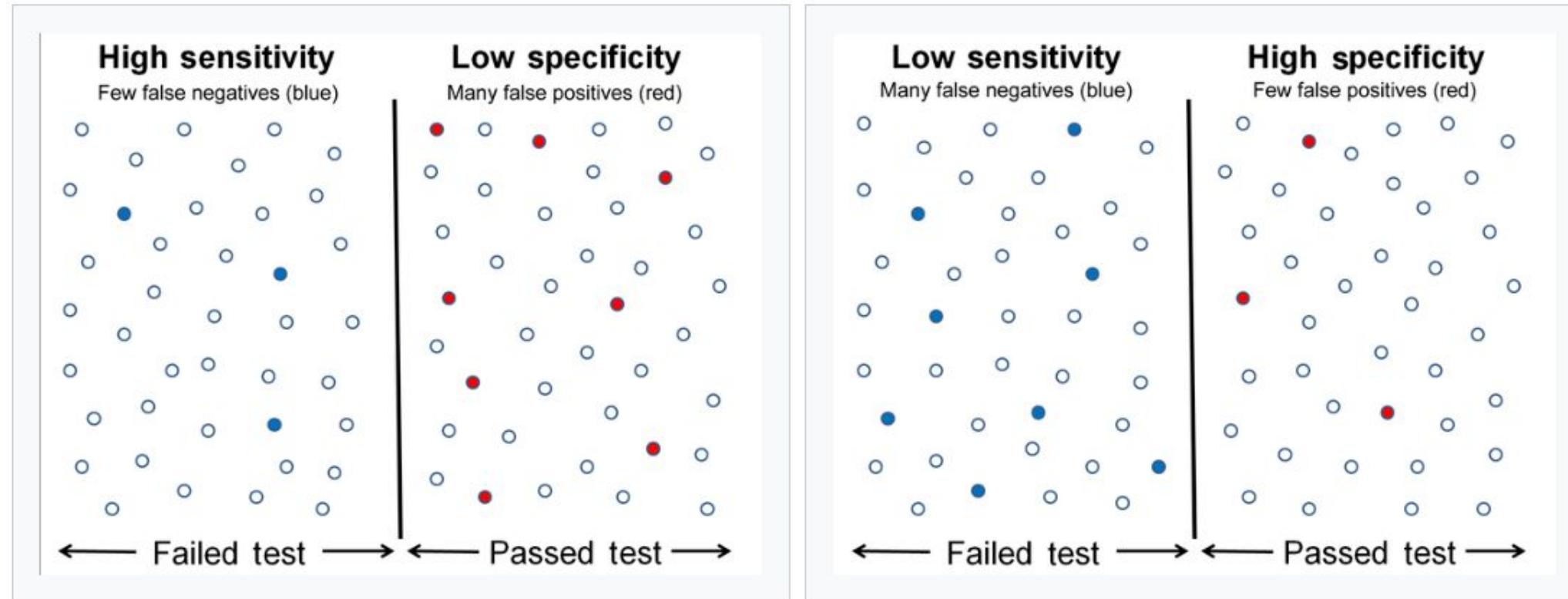
$$=FP/(TN+FP)$$

FPR tells us what proportion of the negative class got incorrectly classified by the classifier.

A higher TNR and a lower FPR is desirable since we want to correctly classify the negative class.

- **Black Dotted Line** – sensitivity and specificity are the same.
- **Left of Dotted Line** – Sensitivity increases & reaches 100% at Line A. Specificity decreases.
- **Right of Dotted Line** – Specificity increases & reaches 100% at Line B. Sensitivity decreases.
- The graph illustrates the **trade-off between sensitivity and specificity** in binary classifiers.

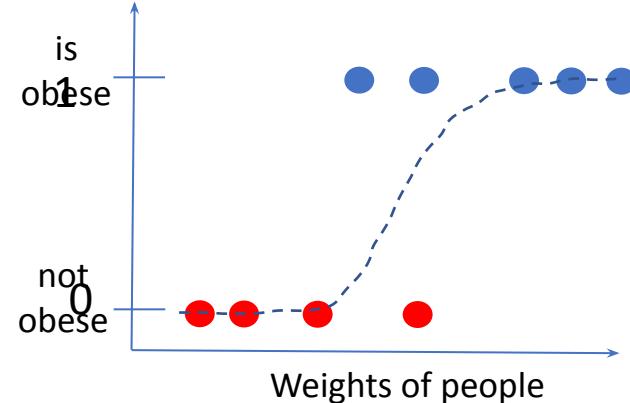




High sensitivity and low specificity

Low sensitivity and high specificity

## ROC -Receiver Operating Characteristics



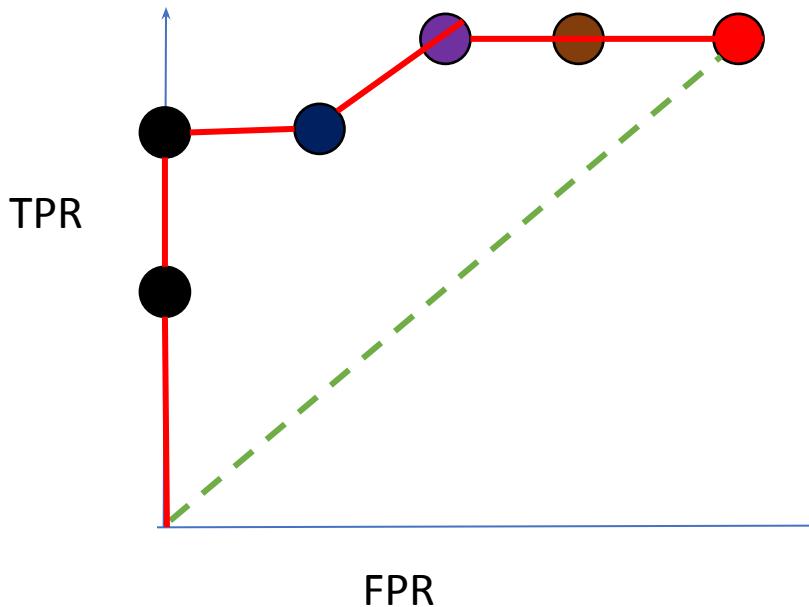
This fitted logistic regression curve we get the data from setting threshold. We need to convert probability to classification of confusion matrix for n chosen threshold. One method is setting up a threshold say at 0.5 now the threshold is too high so that the sample is classified as obese if the probability more than 0.5 is obese and vice versa

- the y axis has two categories- obese and not obese
- the blue dots represent sample who are obese
- the red dots represent sample that are not obese
- along x axis we have weights

The **logistic regression model** (or **logit 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.

## ROC -Receiver Operating Characteristics

Threshold	FPR	TPR
0 (all sample classified obese)	1	1
0.3	0.75	1
0.4	0.5	1
0.6	0.25	0.75
0.7	0	0.75
0.9	0	0.5



This slide will help you in understanding the ROC curve. We are discussing the receiver operating characteristic curve which helps us to understand the performance of a classification model. We are willing to accept the optimal one will be the samples that were classified correctly. We can summarize the output of samples that were incorrectly classified as obese

this means the new threshold is better than the first one

- y axis represents true positive rate (TPR) that is sensitivity
- x axis represents False positive rate (FPR) that is specificity

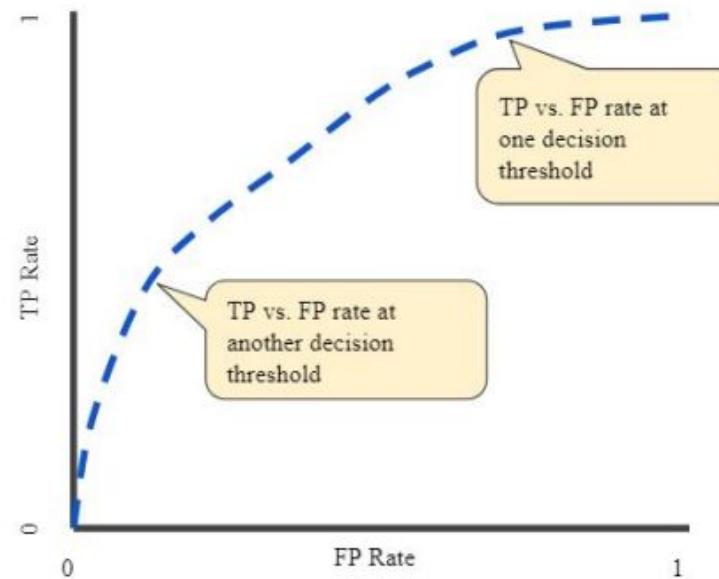
- As seen, an ROC curve plots TPR vs FPR for different classification threshold.
- *If we lower the threshold value then model will classify more objects as +ve class and hence increases both FP and TP.* (This is again food for the brain. Justify)

## ROC - Receiver Operating Characteristics

- An **ROC** curve plots True Positive Rate (TPR) vs. False Positive Rate (FPR) at different classification thresholds for a given model.
- Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

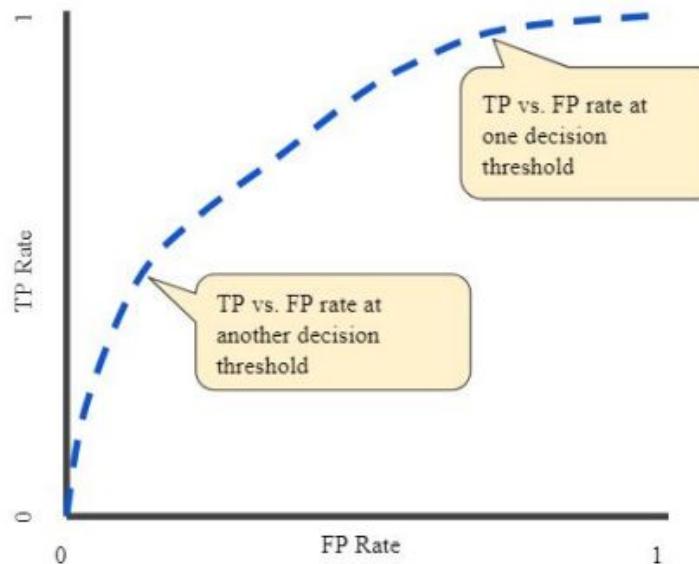


## ROC - Receiver Operating Characteristics

- To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient.
- Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called **Area Under Curve (AUC)**.

$$TPR = \frac{TP}{TP + FN}$$

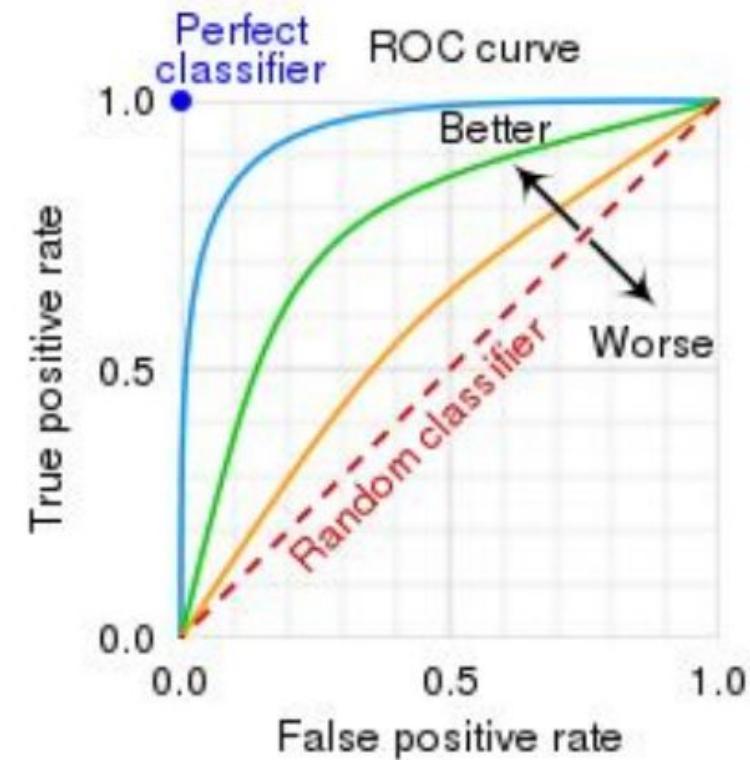
$$FPR = \frac{FP}{FP + TN}$$



# MACHINE LEARNING

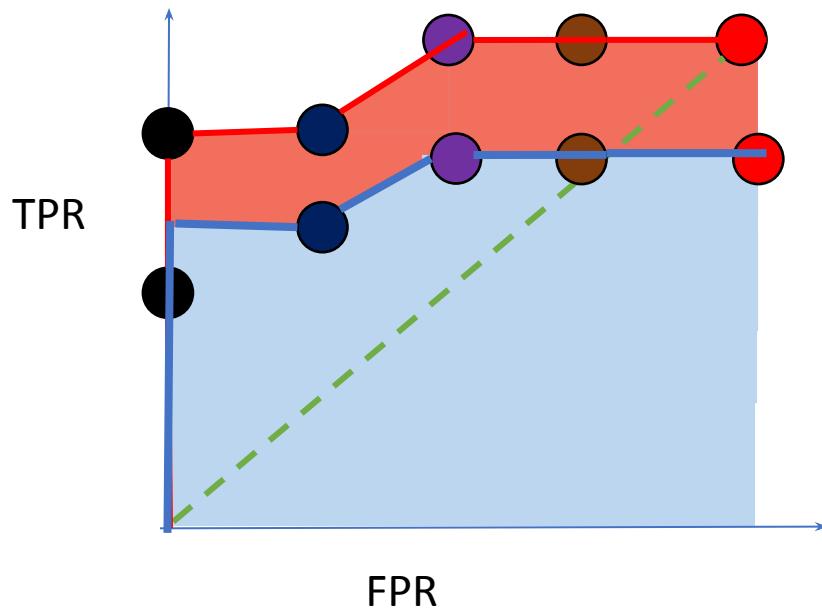
## ROC Interpretation

- The ROC curve shows the trade-off between sensitivity (or TPR) and specificity ( $1 - FPR$ ).
- Classifiers that give curves closer to the top-left corner indicate a better performance.
- As a baseline, a random classifier is expected to give points lying along the diagonal ( $FPR = TPR$ ).
- The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.



## AUC -Area Under The Curve

- AUC - ROC curve is a performance measurement for classification problem at various thresholds settings.
- It tells how much model is capable of distinguishing between classes



The AUC of the above model is approximately 0.65. The AUC of a good model is higher than that of a bad model. The closer the AUC is to 1.0, the better the model is at distinguishing between classes.

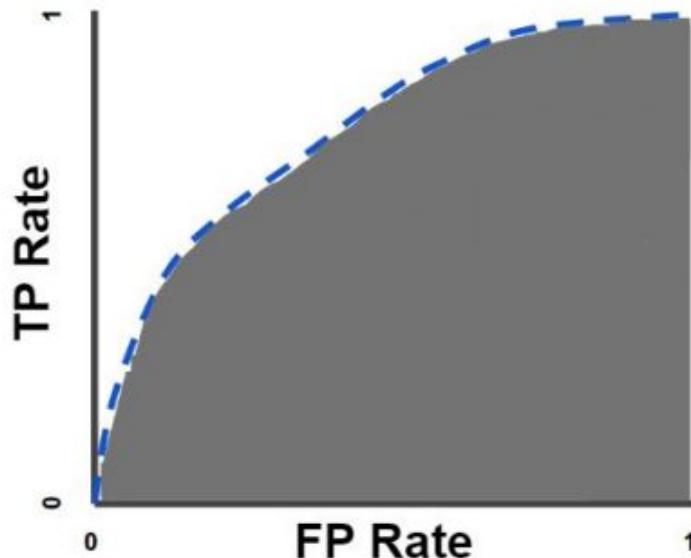
( OPTIONAL )

In order to get detailed idea about systematic performance measures for ML models refer the following paper:

<http://atour.iro.umontreal.ca/rali/sites/default/files/publis/SokolovaLapalme-JIPM09.pdf>

## AUC - Area Under the ROC Curve

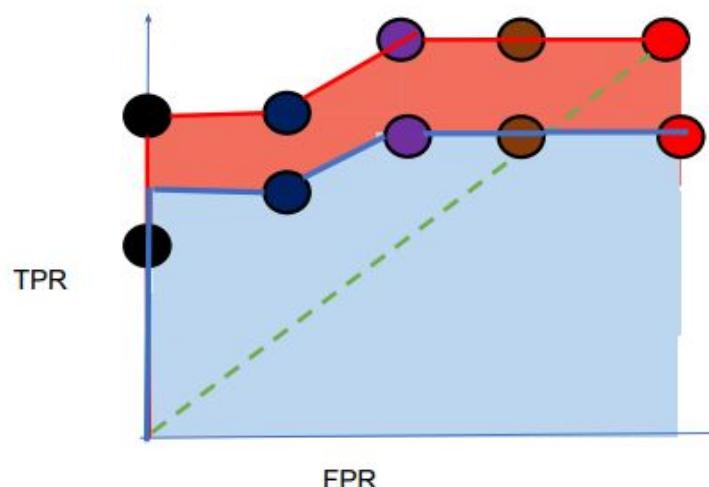
- AUC measures **the entire two-dimensional area underneath the entire ROC curve from (0,0) to (1,1)**.
- AUC provides an aggregate measure of performance across all possible classification thresholds.



The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

## AUC - Area Under the ROC Curve

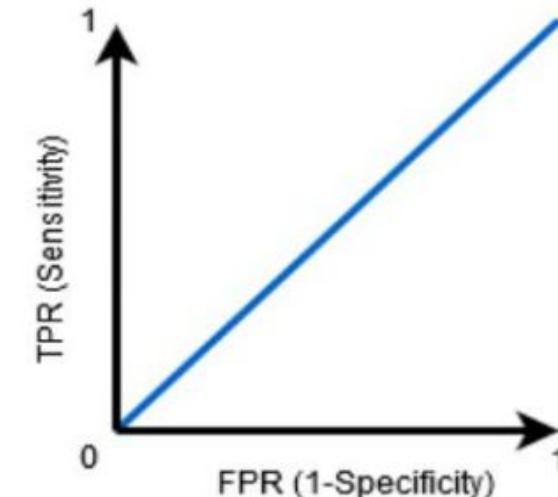
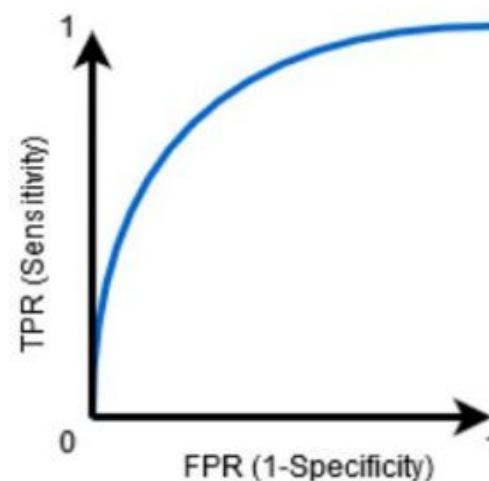
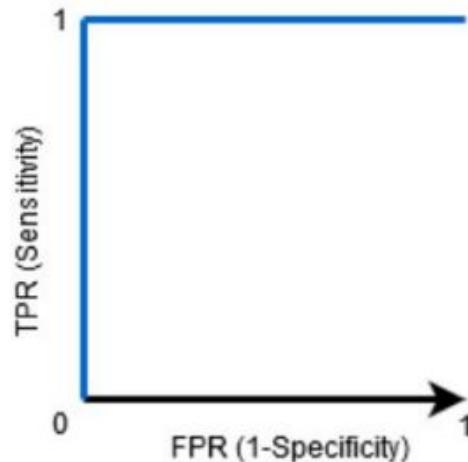
- The AUC defines how good is our model, **more the AUC better is the model.**
- This helps us in comparing two models and selecting best among them for our problem
- The AUC for the red ROC curve is greater than the AUC for the blue which suggests method 1 is better than method 2.



# MACHINE LEARNING

## AUC Interpretation

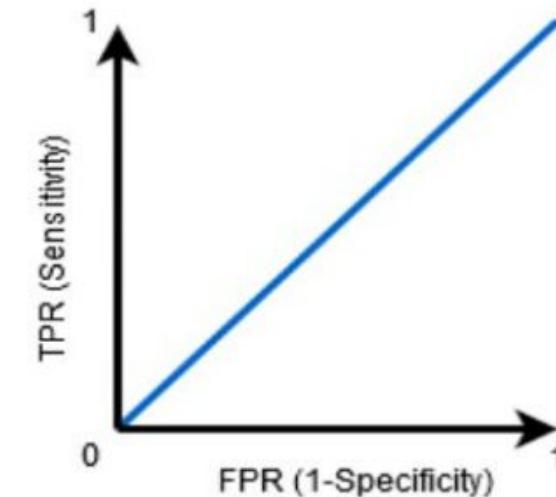
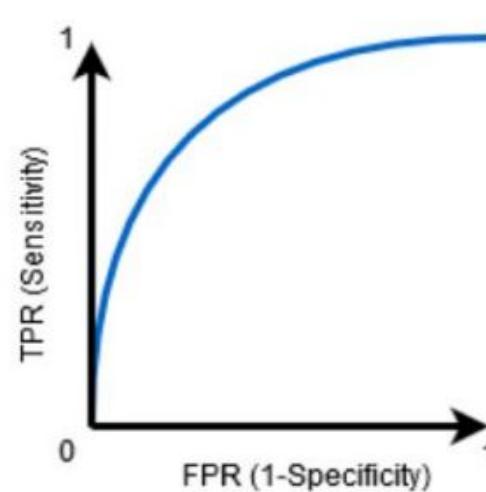
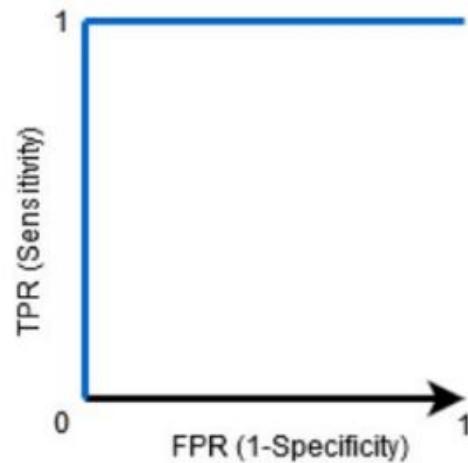
- When **AUC = 1**, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly.
- If, however, the **AUC had been 0**, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.



# MACHINE LEARNING

## AUC Interpretation

- When  $0.5 < \text{AUC} < 1$ , there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values.
- When  $\text{AUC} = 0.5$ , the classifier is not able to distinguish between Positive and Negative class points, predicting random class or constant class for all the data points.



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) <b>Type II Error</b>	<b>Sensitivity</b> $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) <b>Type I Error</b>	True Negative (TN)	<b>Specificity</b> $\frac{TN}{(TN + FP)}$
		<b>Precision</b> $\frac{TP}{(TP + FP)}$	<b>Negative Predictive Value</b> $\frac{TN}{(TN + FN)}$	<b>Accuracy</b> $\frac{TP + TN}{(TP + TN + FP + FN)}$

# MACHINE LEARNING

## Practice Question



For the given table below predict the TPR, TNR, FPR, FNR for the thresholds of 0.6, 0.7 and 0.8 for all the values and draw the AUC-ROC curve.

ID	Actual	Prediction Probability :
1	0	0.98
2	1	0.67
3	1	0.58
4	0	0.78
5	1	0.85
6	0	0.86
7	0	0.79
8	0	0.89
9	1	0.82
10	0	0.86

# MACHINE LEARNING

## ROC -Receiver Operating Characteristics



Solution:

ID	Actual	Prediction Probability	>0.6	>0.7	>0.8	Metric
1	0	0.98	1	1	1	
2	1	0.67	1	0	0	
3	1	0.58	0	0	0	
4	0	0.78	1	1	0	
5	1	0.85	1	1	1	
6	0	0.86	1	1	1	
7	0	0.79	1	1	0	
8	0	0.89	1	1	1	
9	1	0.82	1	1	1	
10	0	0.86	1	1	1	
			0.75	0.5	0.5	TPR
			1	1	0.66	FPR
			0	0	0.33	TNR
			0.25	0.5	0.5	FNR

# MACHINE LEARNING



**PES**  
UNIVERSITY

CELEBRATING 50 YEARS

## Performance metrics – Multi Class Classification

---

## Example - Multi-class confusion matrix

Let's assume that our multi-class classification model is one that classifies images of dogs into the following breeds:

Husky   Labrador   Bulldog



		Predicted		
		Husky	Labrador	Bulldog
Actual	Husky	P(H,H)	P(L,H)	P(B,H)
	Labrador	P(H,L)	P(L,L)	P(B,L)
	Bulldog	P(H,B)	P(L,B)	P(B,B)

Notation Used :  
 $P(\text{Predicted, Actual})$

## Metrics calculations - Multi-class confusion matrix

For Husky class

True Positive	$P(H,H)$
True Negative	$P(L,L) + P(B,L) + P(L,B) + P(B,B)$
False Positive	$P(H,L) + P(H, B)$
False Negative	$P(L,H) + P(B,H)$



Notation Used :  
 $P(\text{Predicted, Actual})$

		Predicted		
		Husky	Labrador	Bulldog
Actual	Husky	$P(H,H)$	$P(L,H)$	$P(B,H)$
	Labrador	$P(H,L)$	$P(L,L)$	$P(B,L)$
	Bulldog	$P(H,B)$	$P(L,B)$	$P(B,B)$

## Metrics calculations - Multi-class confusion matrix

For Labrador class

True Positive	$P(L,L)$
True Negative	$P(H,H) + P(B,H) + P(H,B) + P(B,B)$
False Positive	$P(L,H) + P(L,B)$
False Negative	$P(H,L) + P(B,L)$



Notation Used :  
 $P(\text{Predicted, Actual})$

		Predicted		
		Husky	Labrador	Bulldog
Actual	Husky	$P(H,H)$	$P(L,H)$	$P(B,H)$
	Labrador	$P(H,L)$	$P(L,L)$	$P(B,L)$
	Bulldog	$P(H,B)$	$P(L,B)$	$P(B,B)$

## Metrics calculations - Multi-class confusion matrix

For Bulldog class

True Positive	$P(B,B)$
True Negative	$P(H,H) + P(L,H) + P(H,L) + P(B,B)$
False Positive	$P(B,H) + P(B,L)$
False Negative	$P(H,B) + P(L,B)$



Notation Used :  
 $P(\text{Predicted, Actual})$

		Predicted		
		Husky	Labrador	Bulldog
Actual	Husky	$P(H,H)$	$P(L,H)$	$P(B,H)$
	Labrador	$P(H,L)$	$P(L,L)$	$P(B,L)$
	Bulldog	$P(H,B)$	$P(L,B)$	$P(B,B)$

## Accuracy calculation - multi-class confusion matrix

$$P(H,H) + P(L,L) + P(B,B)$$

Accuracy =

$$(P(H,H) + P(L,H) + P(B,H) + P(H,L) + P(L,L) + P(B,L) + P(H,B) + P(L,B) + P(B,B))$$

Let us calculate  
Accuracy of the  
Classifier

Notation Used :  
 $P(\text{Predicted, Actual})$

		Predicted		
		Husky	Labrador	Bulldog
Actual	Husky	$P(H,H)$	$P(L,H)$	$P(B,H)$
	Labrador	$P(H,L)$	$P(L,L)$	$P(B,L)$
	Bulldog	$P(H,B)$	$P(L,B)$	$P(B,B)$

## Precision Calculation - multi-class confusion matrix

For Husky class

True Positive	$P(H,H)$
True Negative	$P(L,L) + P(B,L) + P(L,B) + P(B,B)$
False Positive	$P(H,L) + P(H,B)$
False Negative	$P(L,H) + P(B,H)$

For Labrador class

True Positive	$P(L,L)$
True Negative	$P(H,H) + P(B,H) + P(H,B) + P(B,B)$
False Positive	$P(L,H) + P(L,B)$
False Negative	$P(H,L) + P(B,L)$

For Bulldog class

True Positive	$P(B,B)$
True Negative	$P(H,H) + P(L,H) + P(H,L) + P(B,B)$
False Positive	$P(B,H) + P(B,L)$
False Negative	$P(H,B) + P(L,B)$

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False Positives}}$$

$$\text{Precision (Husky)} = \frac{P(H,H)}{P(H,H)+P(H,L)+P(H,B)}$$

$$-----$$

$$\text{Precision (Labrador)} = \frac{P(L,L)}{P(L,L)+P(L,H)+P(L,B)}$$

$$-----$$

$$\text{Precision (Bulldog)} = \frac{P(B,B)}{P(B,B)+P(B,H)+P(B,L)}$$

$$-----$$

## Recall/TPR Calculation - multi-class confusion matrix

For Husky class

True Positive	$P(H,H)$
True Negative	$P(L,L) + P(B,L) + P(L,B) + P(B,B)$
False Positive	$P(H,L) + P(H,B)$
False Negative	$P(L,H) + P(B,H)$

For Labrador class

True Positive	$P(L,L)$
True Negative	$P(H,H) + P(B,H) + P(H,B) + P(B,B)$
False Positive	$P(L,H) + P(L,B)$
False Negative	$P(H,L) + P(B,L)$

For Bulldog class

True Positive	$P(B,B)$
True Negative	$P(H,H) + P(L,H) + P(H,L) + P(B,B)$
False Positive	$P(B,H) + P(B,L)$
False Negative	$P(H,B) + P(L,B)$

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False Negatives}}$$

$$\text{Recall (Husky)} = \frac{P(H,H)}{P(H,H)+P(L,H)+P(B,H)}$$

$$-----$$

$$\text{Recall (Labrador)} = \frac{P(L,L)}{P(L,L)+P(H,L)+P(B,L)}$$

$$-----$$

$$\text{Recall (Bulldog)} = \frac{P(B,B)}{P(B,B)+P(H,B)+P(L,B)}$$

$$-----$$

## Multi-class confusion matrix - Practice Problem

Now you know how to calculate the metric for n classes.

Evaluate the given confusion matrix...

		Actual values →			
		A	B	C	D
Predicted values ↓	A	100	0	0	0
	B	80	9	1	1
	C	10	0	8	0
	D	10	1	1	9



**PES**  
UNIVERSITY

# **MACHINE LEARNING(UE22CS352A)**

## **Bias Variance Trade off**

---

**Dr. Arti Arya**  
Department of Computer Science  
and Engineering

# Machine Learning

---

## Bias Variance Trade off

**Dr. Arti Arya**

Department of Computer Science

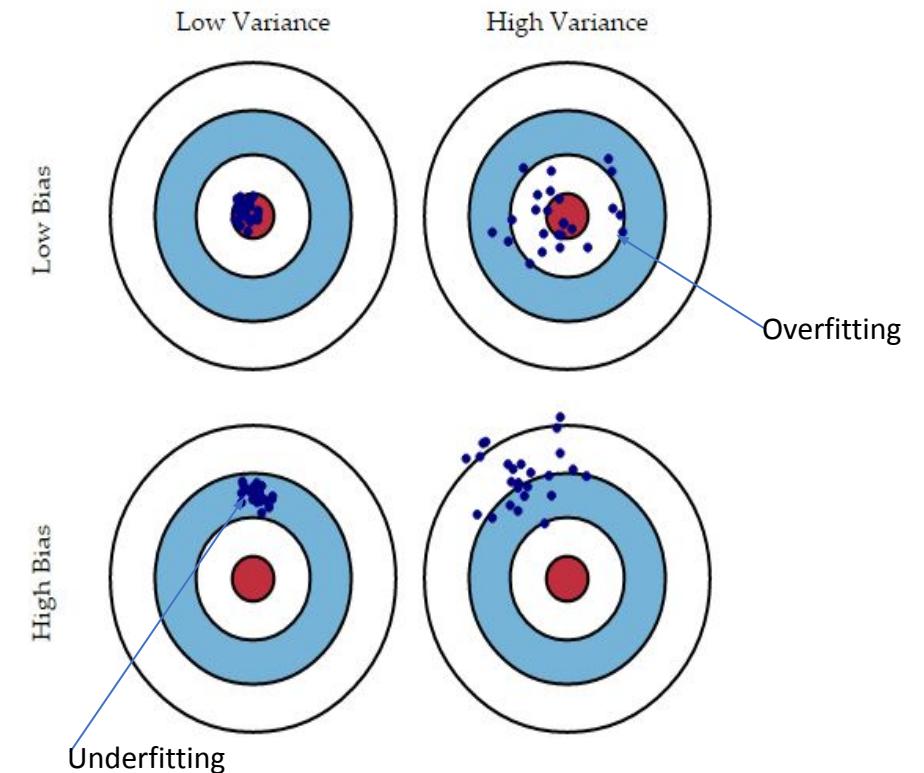
## Tradeoff between Bias and Variance

In a learning model, prediction error is bcoz of **Bias**, **Variance** and **Noise** in the data.

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E\left[ \hat{f}(x) - E[\hat{f}(x)] \right]^2 + \sigma_e^2$$

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

- The **bias** is an error from **erroneous assumptions** in the learning algorithm.
- Bias error** tells how much on an average are **the predicted values different from the actual value**.
- A **high bias error** means **model will be under-performing** which keeps on **missing important trends**.
- Bias is the difference between the average prediction and the correct value.



Assume that red spot is the true value and blue dots are predictions

## Tradeoff between Bias and Variance

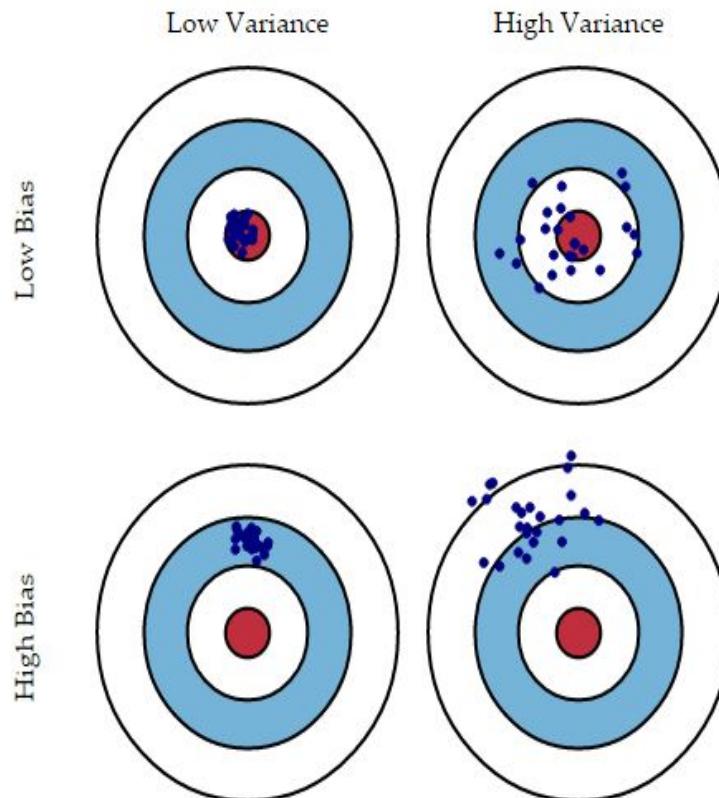
**Irreducible error** is a measure of noise inherent in the data.

In a learning model, prediction error is bcoz of **Bias**, **Variance** and **Noise** in the data.

$$Err(x) = \left( E[\hat{f}(x)] - f(x) \right)^2 + E\left[ \hat{f}(x) - E[\hat{f}(x)] \right]^2 + \sigma_e^2$$

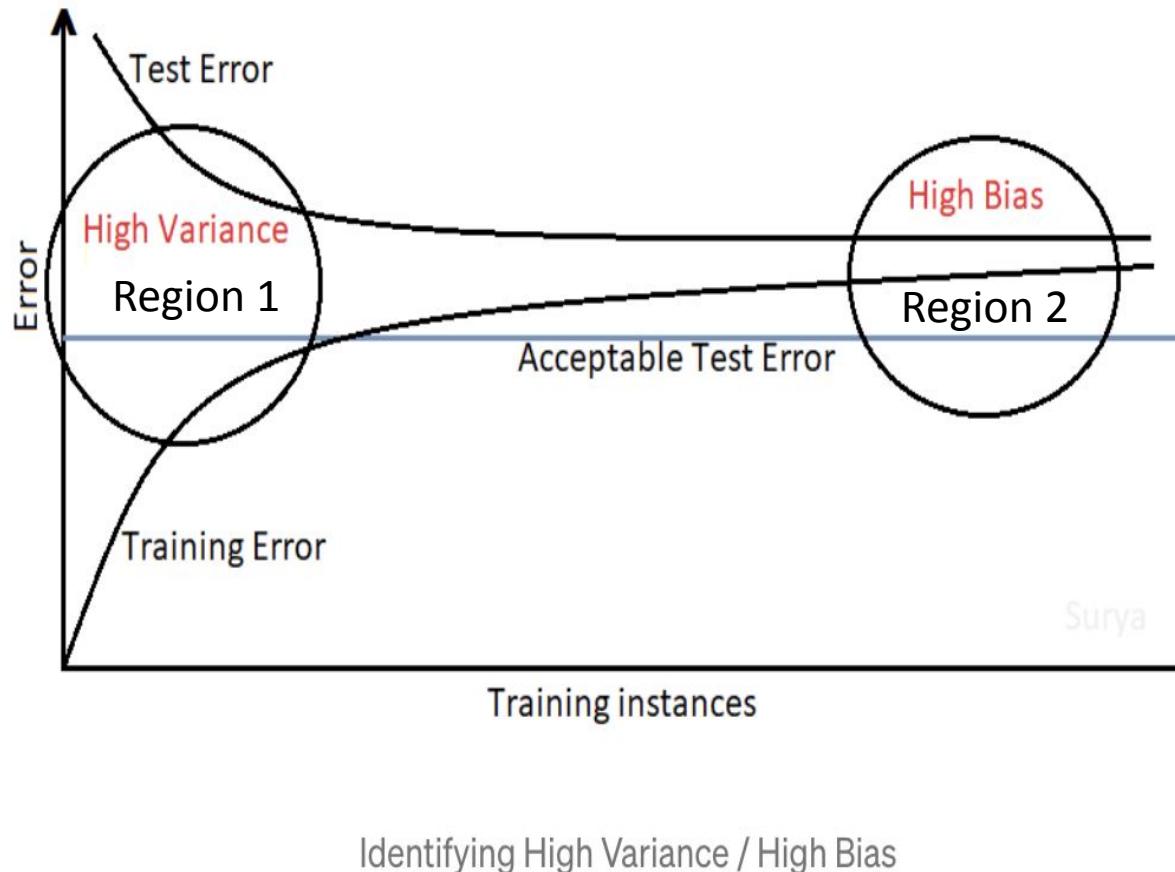
$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- The variance is an error from sensitivity to small fluctuations in the training set.
- Variance** quantifies how are the predictions made on same observation different from each other.
- A **high variance model** will **overfit** the training data and performs badly on any observation beyond training



Assume that red spot is the true value and blue dots are predictions

# How to identify High Variance or High Bias?



## Regime 1 (High Variance)

In the Region 1, the cause of the poor performance is high variance.

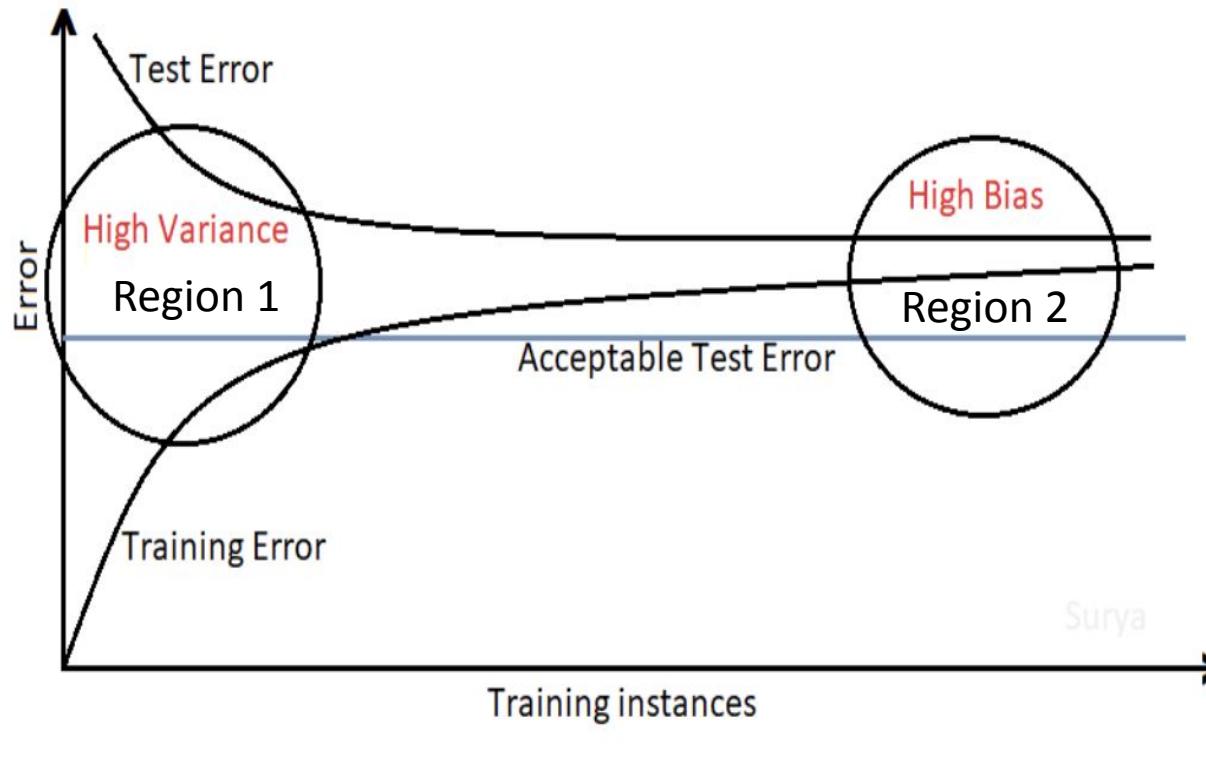
### Symptoms:

- Training error is much lower than test error
- Training error is lower than  $\epsilon$
- Test error is above  $\epsilon$

### Remedies:

- Add more training data
- Reduce model complexity -- complex models are prone to high variance
- Bagging

# How to identify High Variance or High Bias?



Identifying High Variance / High Bias

## Regime 2 (High Bias)

The Region 2 indicates high bias: the model being used is not robust enough to produce an accurate prediction.

### Symptoms:

- Training error is higher than  $\epsilon$

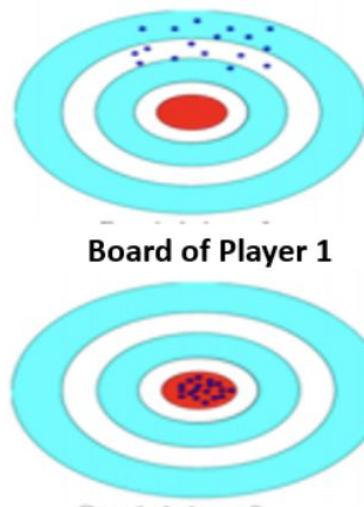
### Remedies:

- Use more complex model (e.g. kernelize, use non-linear models)
- Add features
- Boosting (will be covered later in the course)

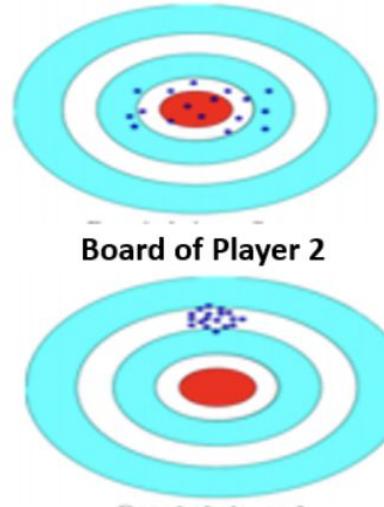
# Look into this problem

Bias and Variance can be visualized using a classic example of a dart game. We can think of the true value of the parameters as the bull's-eye on a target, and the arrow's value as the estimated value from each sample. Consider the following situations of four players, and select the correct option(s)

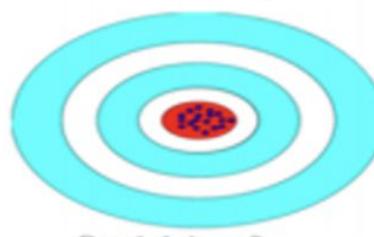
- (a) Player 1 has low variance compared to player 4
- (b) Player 2 has low bias and high variance as compared to 4



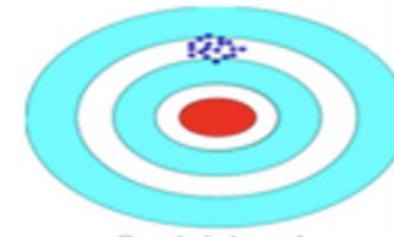
Board of Player 1



Board of Player 2



Board of Player 3



Board of Player 4

*Solution: We can think of the true value of the population parameter as the bulls-eye on a target, and of the sample statistic as an arrow fired at the bulls-eye.*

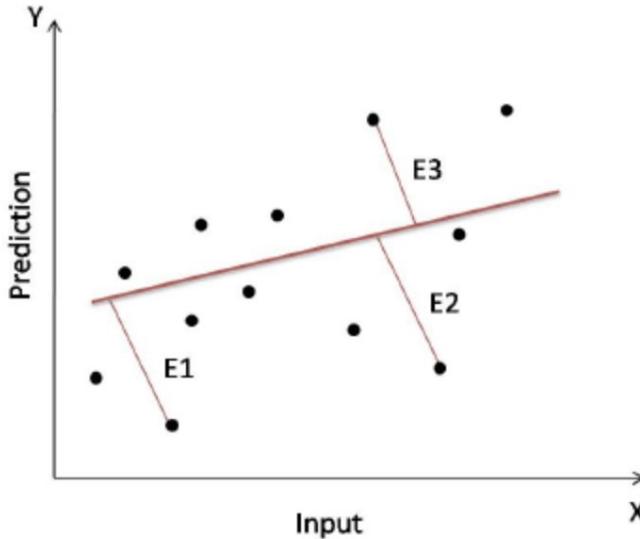
*Bias and variability describe what happens when a player fires many arrows at the target. Bias means that the aim is off, and the arrows land consistently off the bulls-eye in the same direction.*

*The sample values do not center about the population value. Large variability means that repeated shots are widely scattered on the target. Repeated samples do not give similar results but differ widely among themselves.*

- (a) Player 1 - Shows high bias and high variance compared to 4
- (b) Player 2 - Shows low bias and high variance

Look into

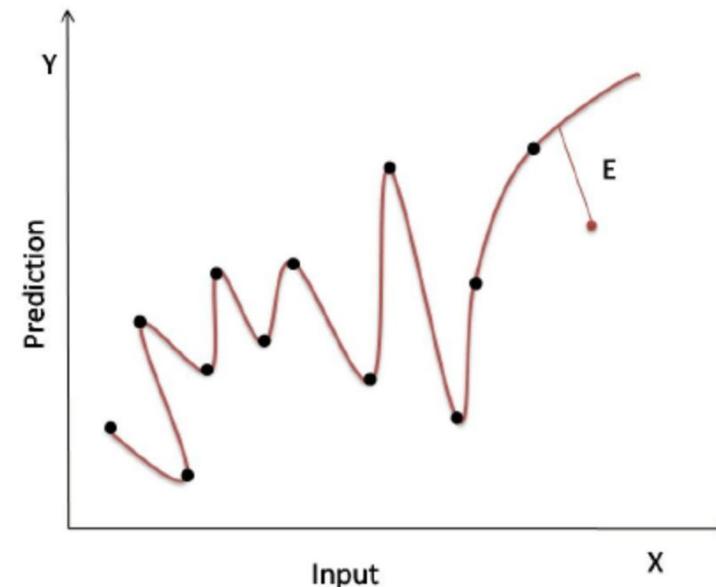
this...



Q: What comment can you make about bias from this graph?

Ans:

***High bias, showing how poorly a function fits datapoints, depicting underfitting.***



Q: What comment can you make about variance from this graph?

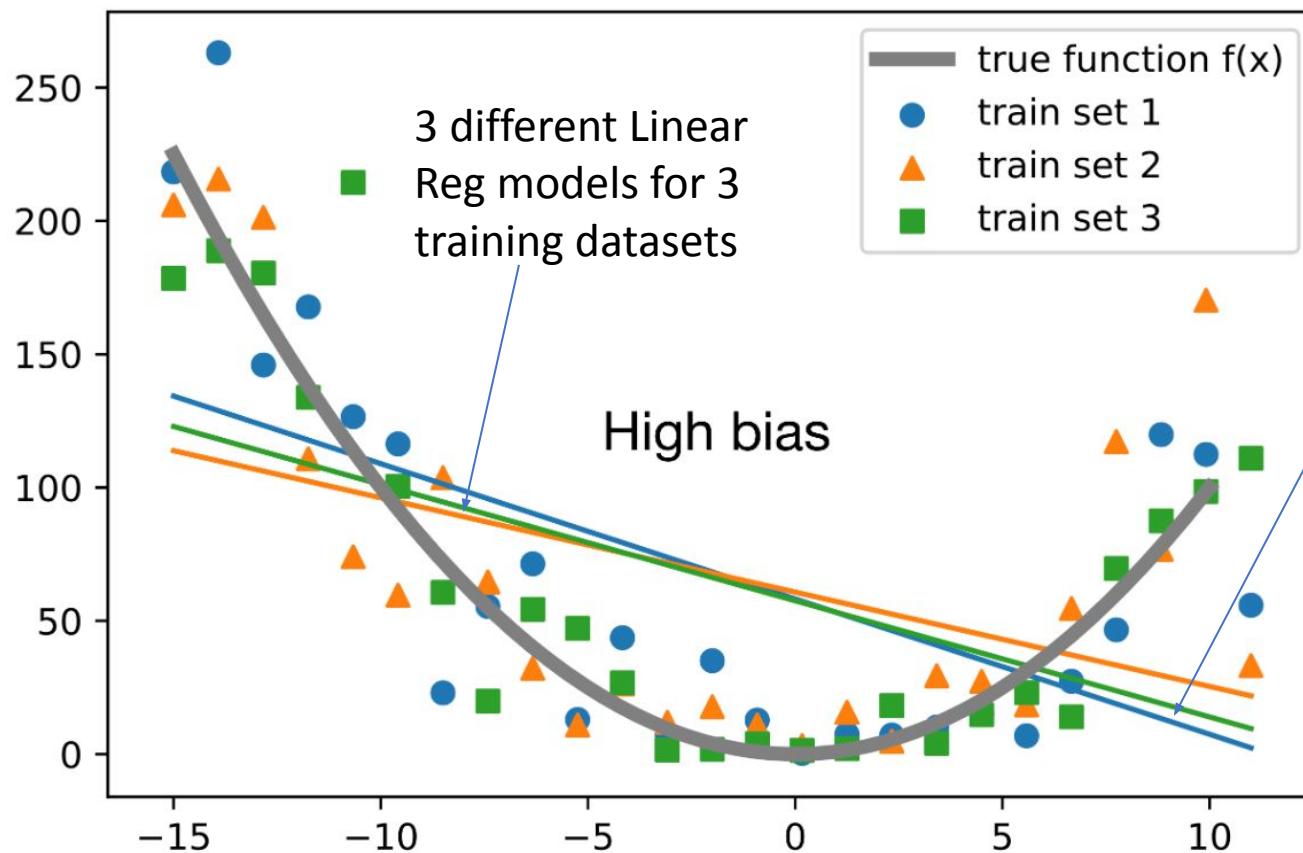
Ans:

***High variance. The function fits the points too closely. Overfitting is present.***

Look into

this...

Suppose there is an unknown target function to which we do want to approximate.



Consider, 3 different training sets drawn from an unknown distribution defined as "true function + noise."

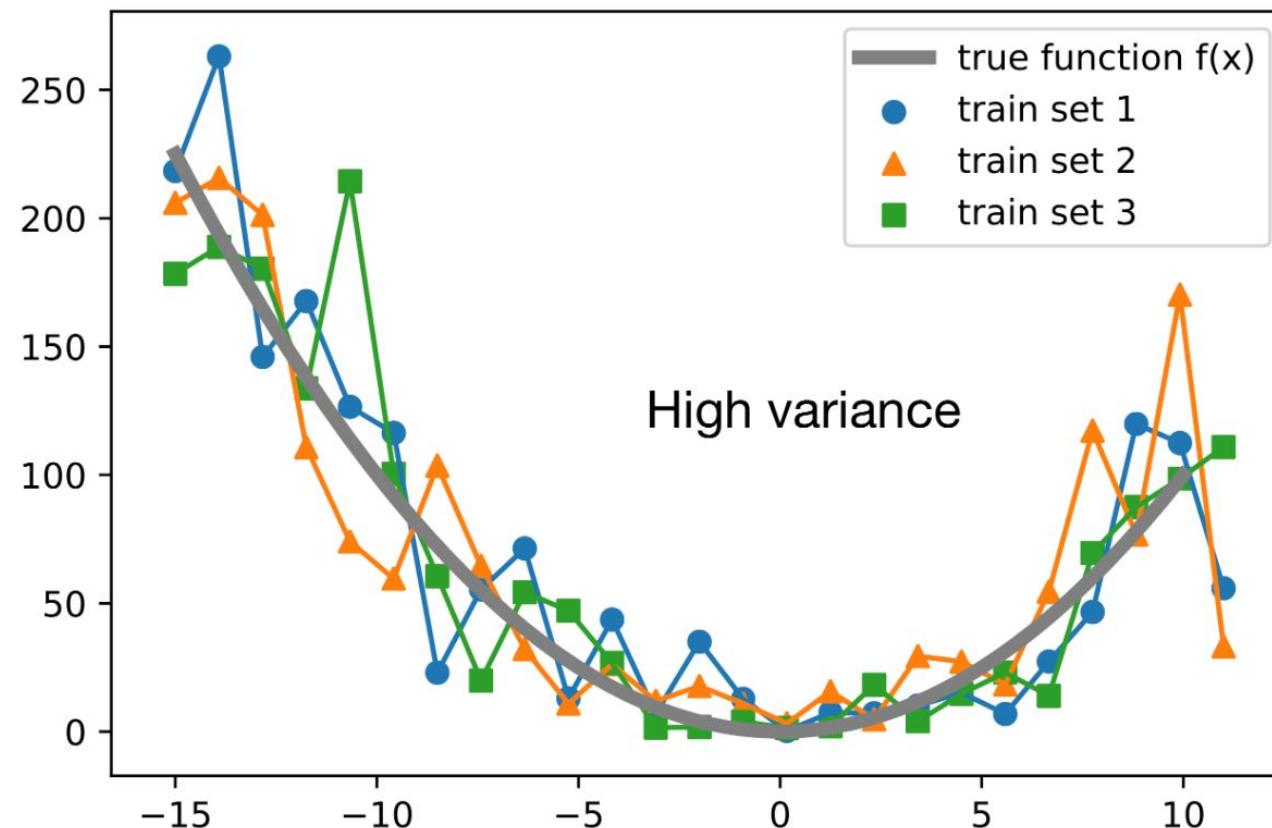
None of these hypotheses approximate the true function well, except at two points (around  $x=-10$  and  $x=6$ ).

Here, **the bias is large** because the difference between the true value and the predicted value, on average (here, average means "expectation of the training sets" not "expectation over examples in the training set"), is large

Look into

this...

Suppose there is an unknown target function to which we do want to approximate.



Consider, 3 different training sets drawn from an unknown distribution defined as "true function + noise."

- Say, different unpruned decision tree models, each fit to a different training set.
- These hypotheses fit the training data very closely.
- But **the variance is very large**, since on average, a prediction differs a lot from the expectation value of the prediction:

## Tradeoff between Bias and Variance

---

- Eg. **Low-bias and High variance** ML algorithms include: Decision Trees, k-Nearest Neighbors and SVMs. (Analyse yourself)
- Eg. **High-bias and Low variance** ML algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression. (Analyse yourself)
- **The goal of any supervised machine learning algorithm is to achieve low bias and low variance . In turn the algorithm should achieve good prediction performance.**
- Ensemble of classifiers may or may not be more accurate than any of its individual model.

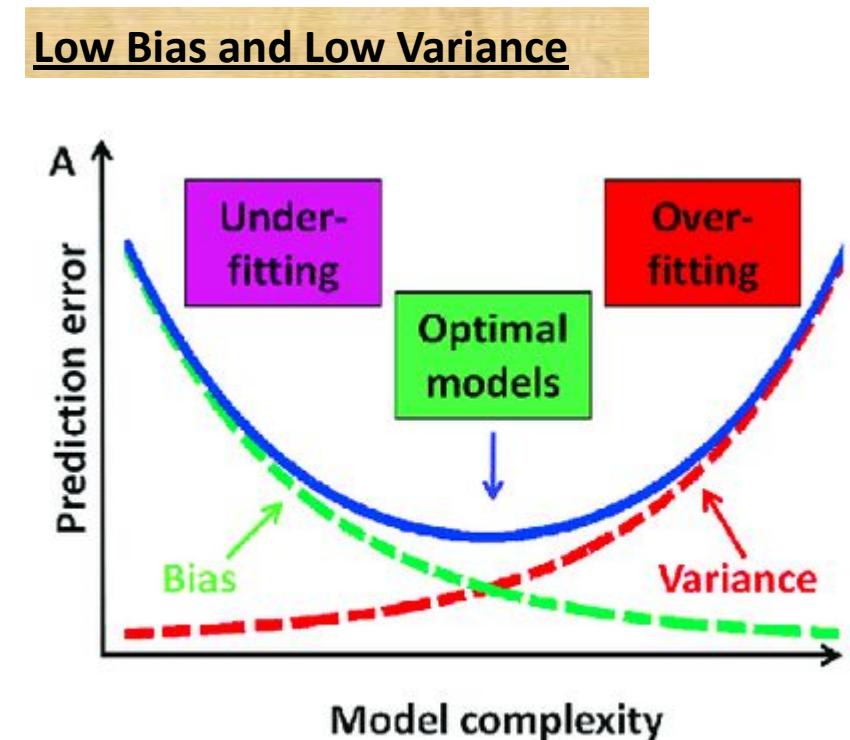
## Tradeoff between Bias and Variance

---

- As the **complexity of the model increases**, a reduction in error happens due to **lower bias** in the model.( but only till a particular point)
- As one continues to make the model more complex, it ends up **over-fitting the model** and hence the model will start suffering from **high variance**.
- **A great ML model must maintain a balance between these two types of errors.**
- This is known as the **trade-off management of bias-variance errors**.
- [Ensemble learning](#) is one way to execute this trade off analysis.

# Bias Variance - Recap

- A model with **high bias** is too simple and has **low number of predictors**
- Due to which it **is unable to capture the underlying pattern** of data.
- It pays very little attention to the **training data** and **oversimplifies** the model. This leads to **high error** on **training and test data**.
- Any model which has **very large number of predictors** will end up being a **very complex model**
  - which **will deliver very accurate predictions** for the **training data** that it has seen already but this complexity makes **the generalization of this model to unseen data** very difficult i.e a **high variance** model.



Source: <https://towardsdatascience.com/holy-grail-for-bias-variance-tradeoff-overfitting-underfitting-7fad64ab5d76>

# Bias and Variance

- We have seen in Neural networks that if we have very large number of epochs model may overfit, ie high variance.
- Low Bias – High flexibility (DT/ANN)
- High Variance – we give different subsets of training data we get different models
- More flexible representations(Low Bias) have High variance
- More powerful representations have high variance
- We want to have a low bias and low variance model

.

Basic models perform not so well by themselves either because they have a high bias (low degree of freedom models, for example) or because they have too much variance to be robust (high degree of freedom models, for example)

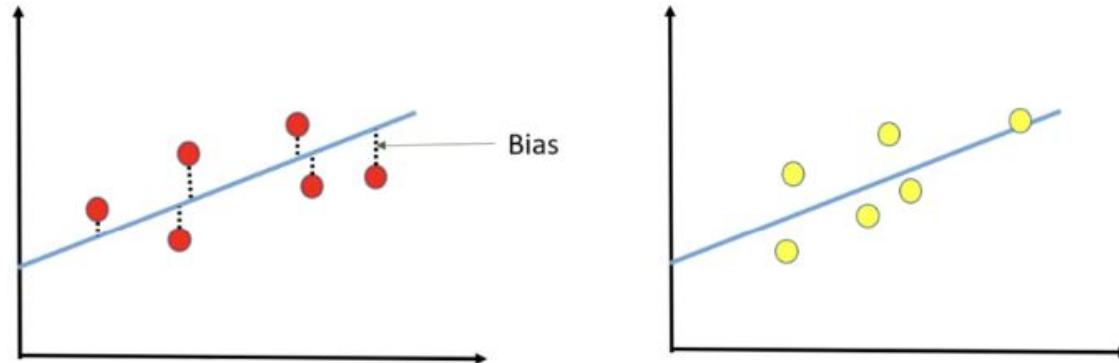
The ensemble itself will produce a model with low bias and low variance as illustrated earlier

In fact even if the individual learners have high bias the new combined learner will have a low bias

The hypothesis of the new learner may not even be in the HS of the individual learners

# Bias and Variance Summary

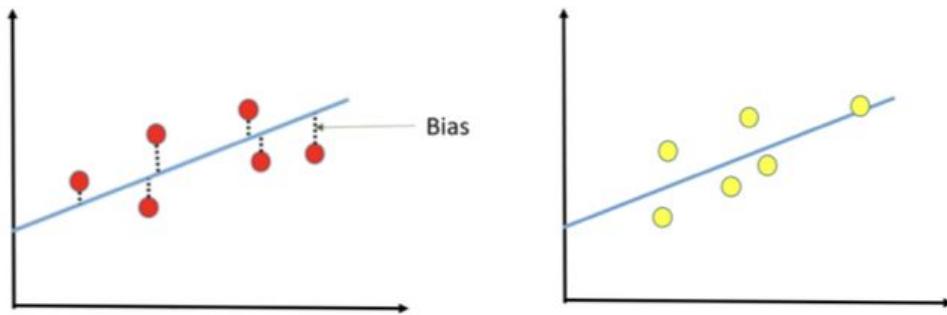
Bias: The difference between the average prediction of our model and the correct value which we're trying to predict.



For example, we have a model that predicts how much a person X spends per month given his monthly spending data for the past 5 years. Lets say X spends 40,000 rupees this month but based on the data our model has predicted that he spends 38,000 rupees this month. The difference between the two values, 2000 rupees is the bias in this scenario.

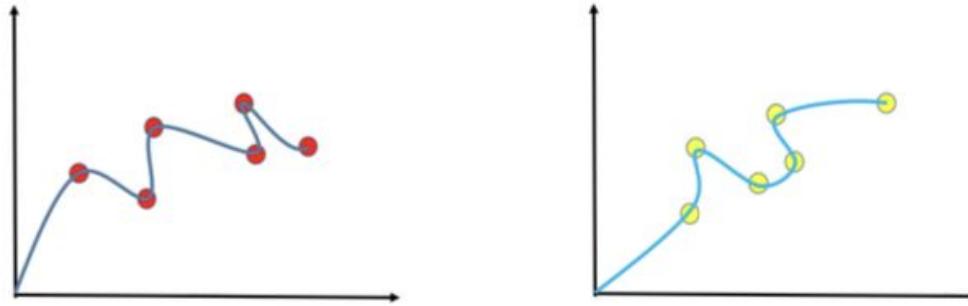
# Bias and Variance Summary

Variance: The amount that the estimate of the target function will change if different training data was used.



Here although the training data changes, the target function is mostly the same, there is very less variation, i.e, low variance. But there's a distance between the line and the points, so, high bias.

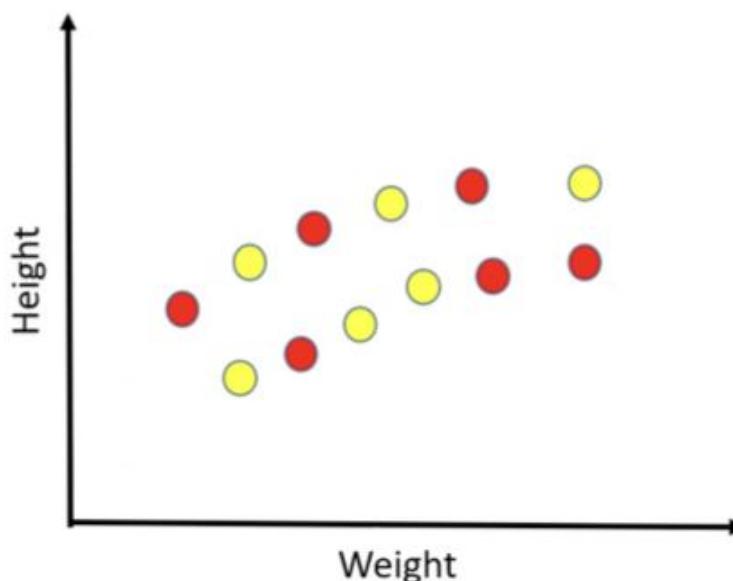
# Bias and Variance Summary



- In the above diagram, when the training data changes the target function changes significantly, i.e, high variance. But the model fits all the points, so low bias (in this case bias=0 since it touches all the points)

# Bias and Variance Example

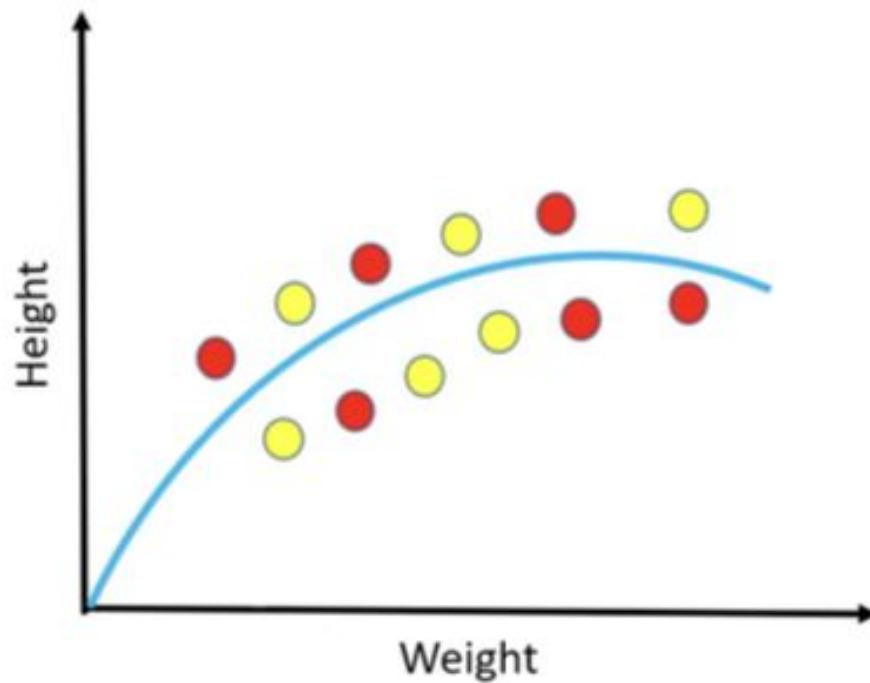
- Summing up everything we just learnt.....
- We need to identify an appropriate model to predict the height of a person when their weight is given.



- Do we need a simple linear model or a complex model? Let's analyze the graph first.
- Up to a certain point when height increases weight also increases, but after that even though the weight increases, the height doesn't.

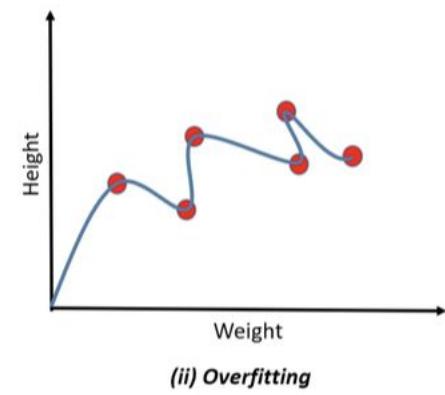
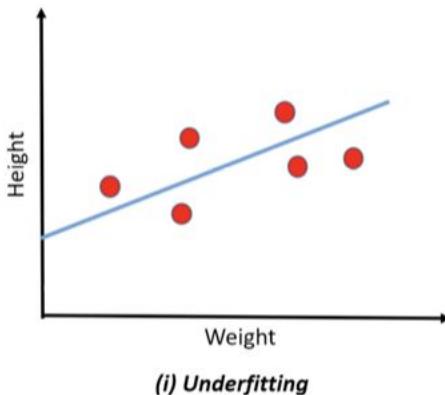
# Bias and Variance Example

The optimum curve for these points would be :



# Bias and Variance Example

- But how does this lead to overfitting and underfitting?
- Let's use a linear and a complex model on the testing and training data for the example to understand better:



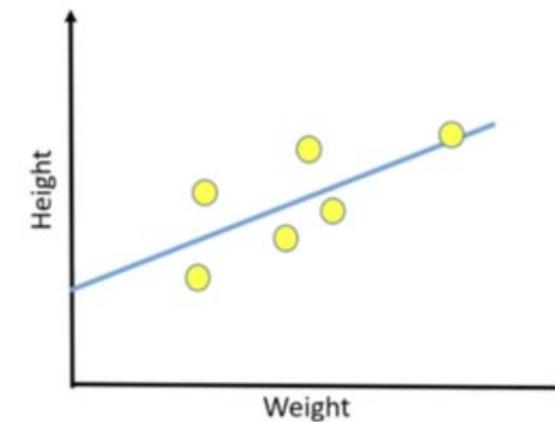
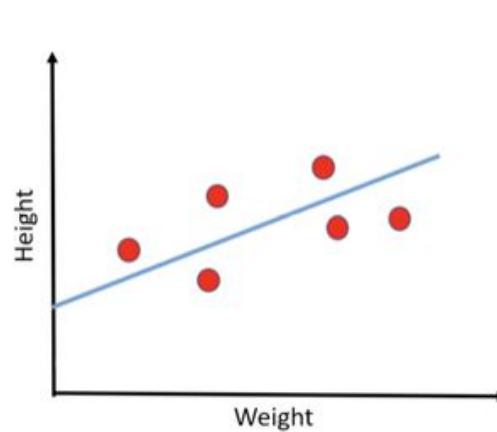
*When a linear model is used on the training data:* It doesn't find the pattern in the data. Its inference is that if the height of a person increases, then their weight also increases, which is wrong, because we know that after a certain point, even when the weight increases the height doesn't increase.

*When a complex model is used on the training data:* The model tries to fit to all the points and we won't get any trend.

# Bias and Variance Example

But what is the bias and variance for these two models?

*For the simple linear model:*



Red- training data, yellow- testing data

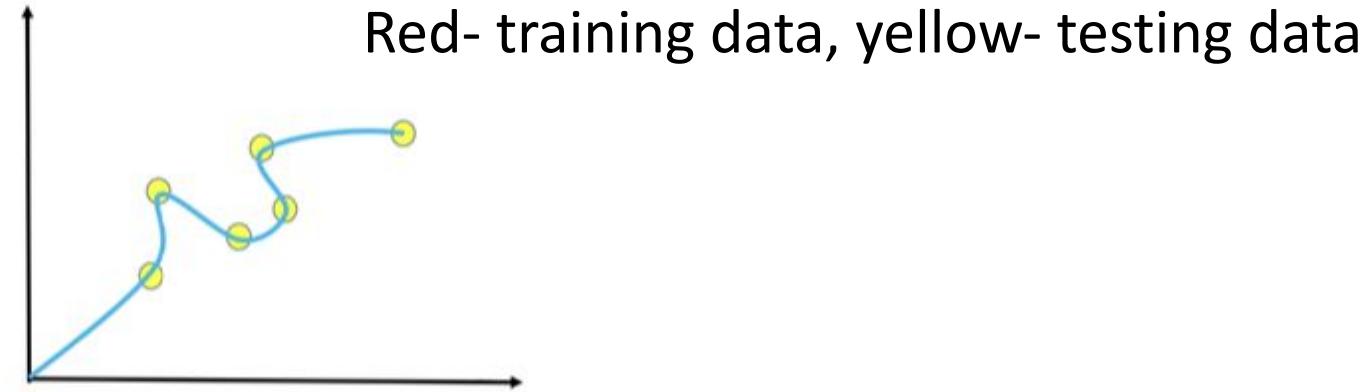
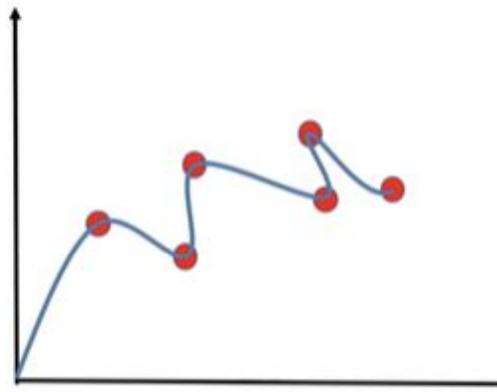
Here, the bias is high, but the variation (variance) between the models used on the testing and training data is very less (comparing the intercept and slope of the lines).

Inference:

- High bias
- Low variance

# Bias and Variance Example

*For the complex model:*

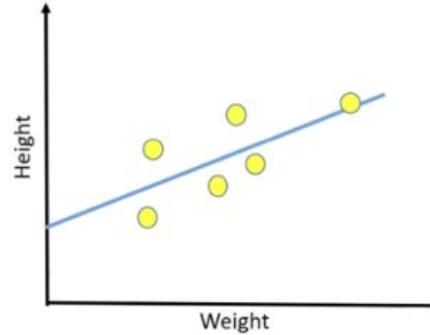


Here, the bias is zero, but the variation (variance) between the models used on the testing and training data is very high

Inference:

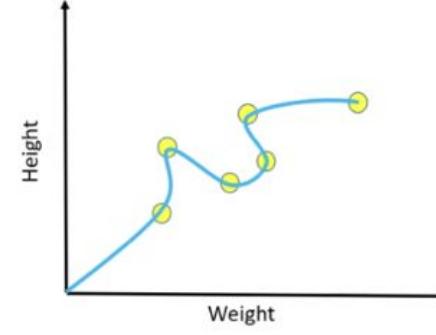
- Low bias
- High variance

# Bias and Variance Example



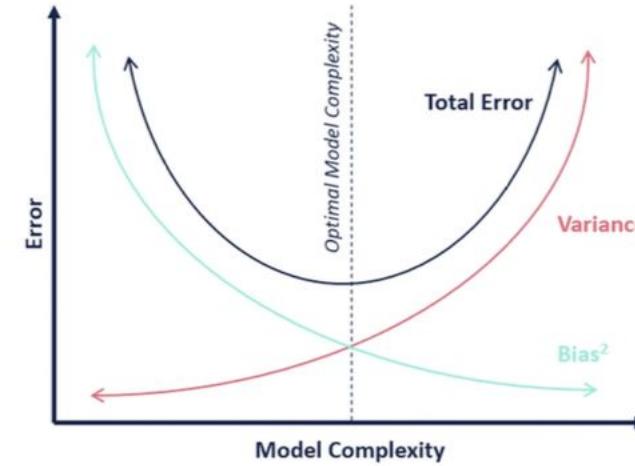
(i) Underfitting

*Inference:* a. High Bias  
b. Low Variance



(ii) Overfitting

*Inference:* a. Low Bias  
b. High Variance



Q: What is a good model?

Ans: A model that neither underfits nor overfits is a good model.

Q: How do we get a good model?

Ans: By adjusting the values of bias and variance. A good balance of bias and variance gives an optimised model.

# Some Interview questions on Bias and Variance

---

1. What impact do Bias and Variance in data have on Machine Learning models?
2. Can Machine models overcome underfitting on biased data and overfitting on data with variance? Does this guarantee correct results?
3. How can you identify a High Bias (Low Variance) model?
4. How can you fix a High Bias model?
5. How can you identify a High Variance (Low Bias) model?
6. How can you fix a High Variance model?
7. What is the Bias and Variance Tradeoff?
8. Would it be better if an ML algorithm exhibits a greater amount of Bias or a greater amount of Variance?

Refer for answers:

<https://analyticsarora.com/8-unique-machine-learning-interview-questions-about-bias-and-variance/>



THANK YOU

---

Dr. Arti Arya  
[artarya@pes.edu](mailto:artarya@pes.edu)





**PES**  
UNIVERSITY

# MACHINE LEARNING

## DECISION TREE -ID3 ALGORITHM

---

**Course Instructor: Dr. Arti Arya**  
Department of Computer Science and Engineering

# MACHINE LEARNING

---

## DECISION TREE -ID3 ALGORITHM

**Dr. Arti Arya**

Professor, Department of Computer Science Engg

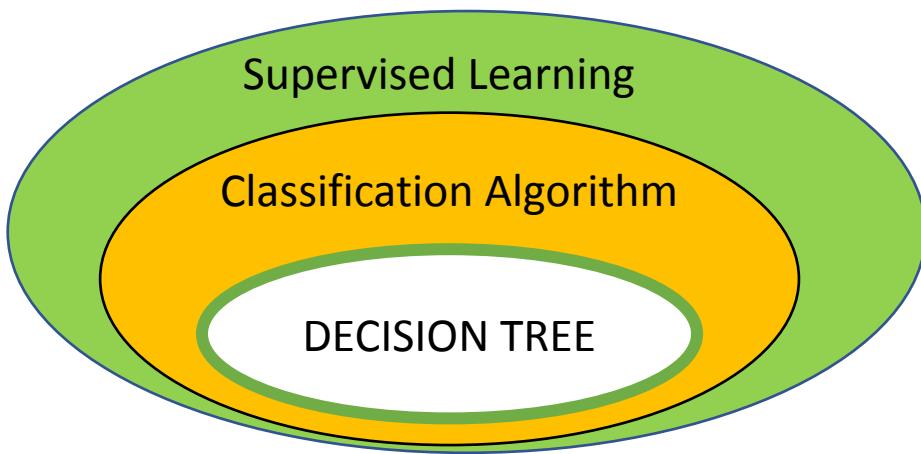
## Acknowledgement

---

The slides are prepared from various from various Universities from abroad and India also. Also, some material is taken from reliable resources from internet throughout this course and sources are mentioned. The inputs to the slides are by Prof. K S Srinivas and Dr. Arti Arya.

## What is a Decision Tree algorithm?

- A type of classification algorithm comes under supervised learning technique.



# MACHINE LEARNING

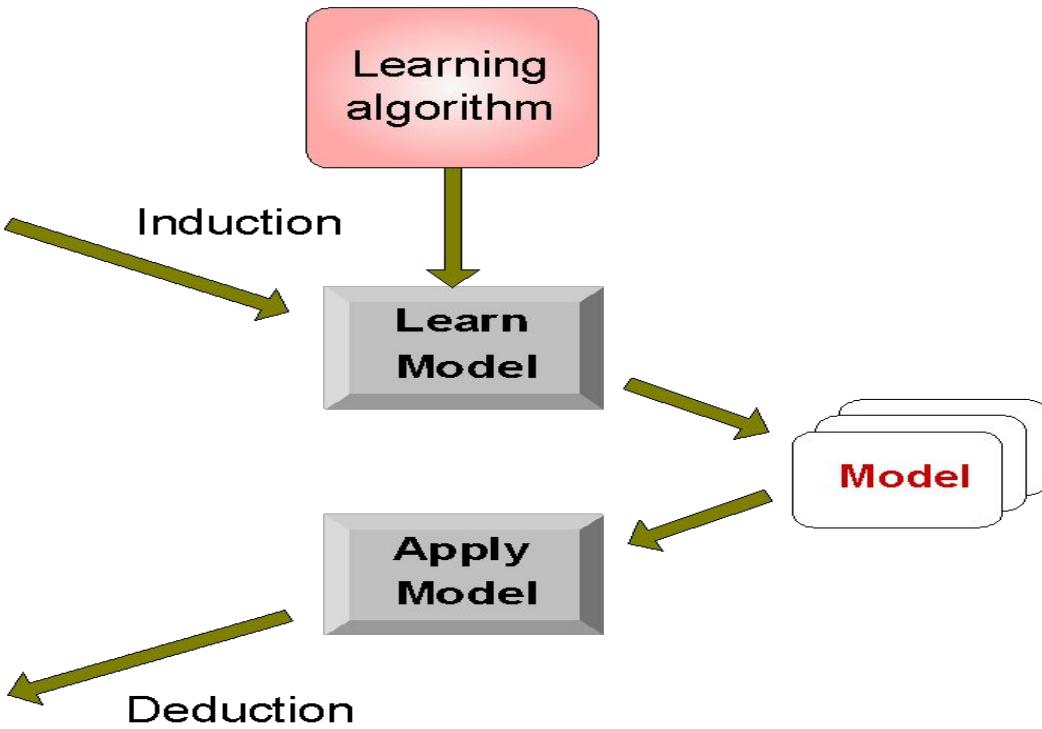
## Classification Recap

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Classification can be defined as learning a function  $f$  over set of attributes  $x$  to get the target variable  $y$ ,  $f(x)=y$

$$f(\text{attrib1}, \text{attrib2}, \text{attrib3}) = \text{Class}$$

### Base Classifiers

Decision Tree based Methods

Rule-based Methods

Nearest-neighbor

Neural Networks

Naïve Bayes and Bayesian Belief Networks

Support Vector Machines

### Ensemble Classifiers

Boosting,

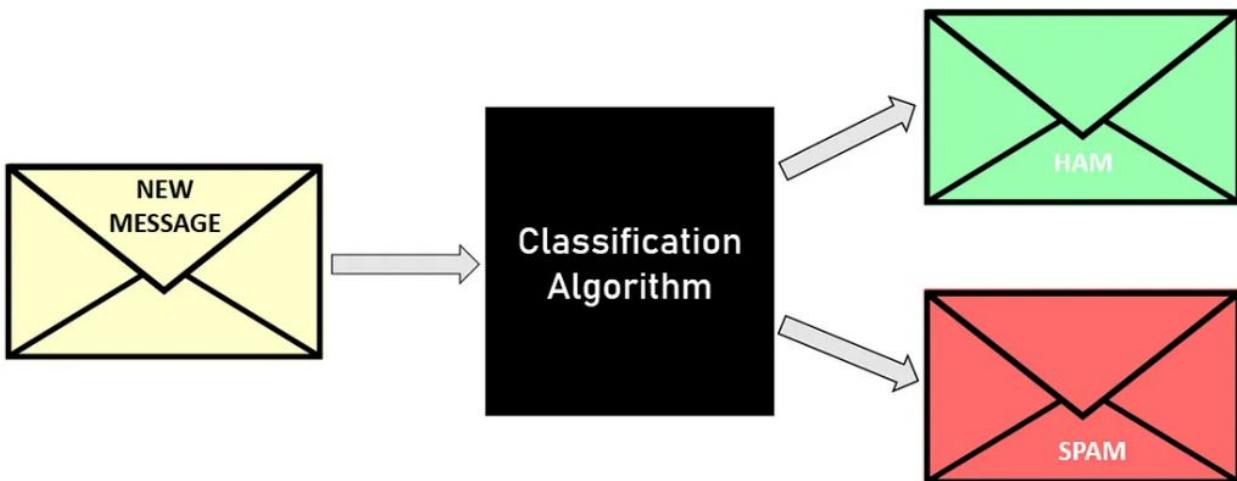
Bagging,

Random Forests

## What is classification algorithm?

***“Classification is the process of dividing the data sets into different categories or groups by adding label”***

Example: Classification of emails as spam or not spam based on certain conditions



[Source: Spam and Ham Text Classification using Naive Bayes and Support Vector Machines | by Sudeshna dutta | Medium](#)

- Graphical representation of all the possible **solutions** to a **decision**
- Decisions are based on some **conditions**
- Decision made can be easily explained



Given an occurrence of a word, decide which sense, or meaning, was intended.

Example: "run"

run1: move swiftly (I ran to the store.)

run2: operate (I run a store.)

run3: flow (Water runs from the spring.)

run4: length of torn stitches (Her stockings had a run.)

etc.

### Classes

Use **word sense labels** (run1, run2, etc.) to name the possible Classes.

### Features

Features describe the *context* of the word we want to disambiguate.

Possible features include:

**near(w)**: is the given word near an occurrence of word w?

**pos**: the word's part of speech

**left(w)**: is the word immediately preceded by the word w?

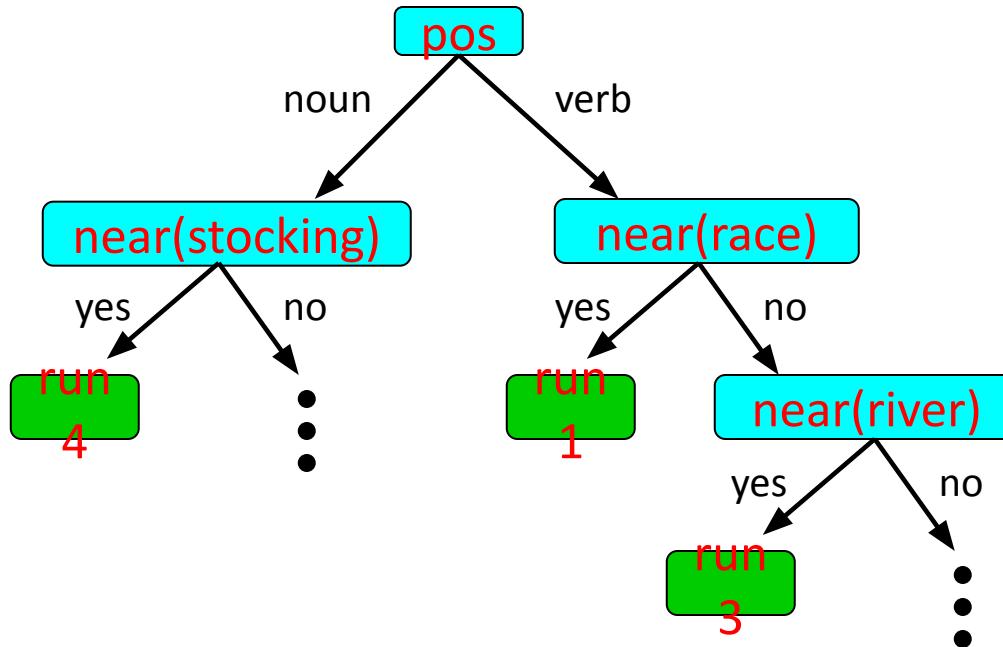
etc.

## What is a Decision Tree algorithm?

---

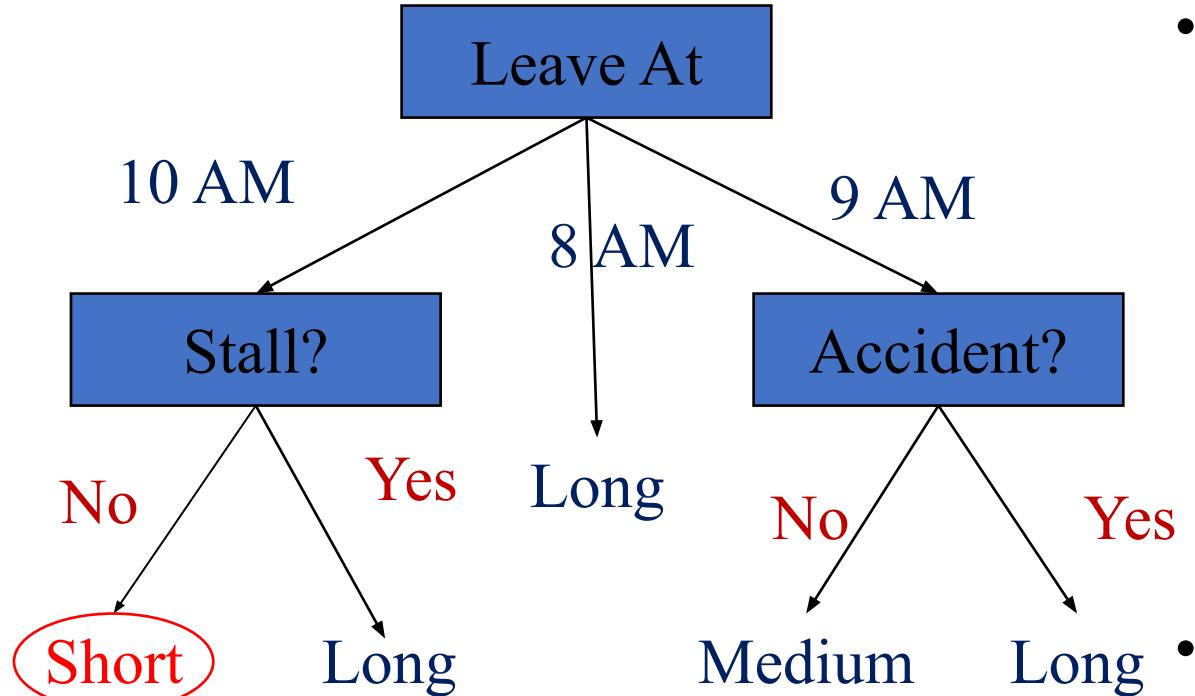
Features				Word Sense
pos	near(race)	near(river)	near(stockings)	
noun	yes	no	no	run1
verb	no	no	no	run1
verb	no	yes	no	run3
noun	yes	yes	yes	run4
verb	no	no	yes	run1
verb	yes	yes	no	run2
verb	no	yes	yes	run3

Example decision tree:



(Note: Decision trees for WSD tend to be quite large)

## Another Example



- In this decision tree, we made a series of Boolean decisions and followed the corresponding branch

Did we leave at 10 AM?  
Did a car stall on the road?  
Is there an accident on the road?

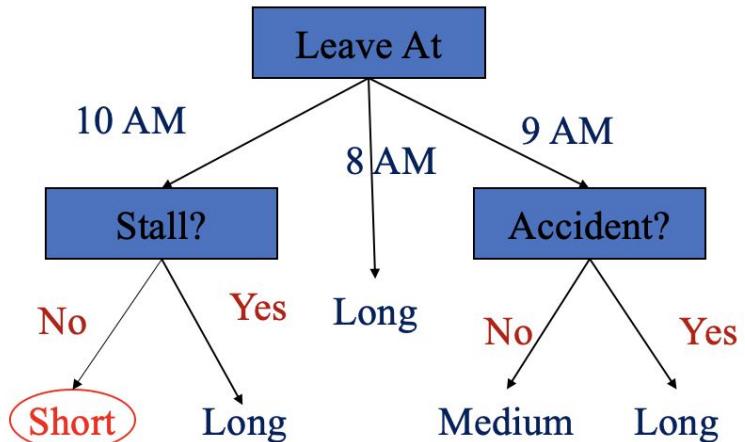
- By answering each of these yes/no questions, we then came to a *decision* on how long our commute might take

If we leave at 10 AM and there are no cars stalled on the road, what will our commute time be?

## Decision Tree as a Rule or function

We can represent DT as a set of rules.

$$f(\text{Leave at}, \text{Accident}, \text{Stall}) =$$

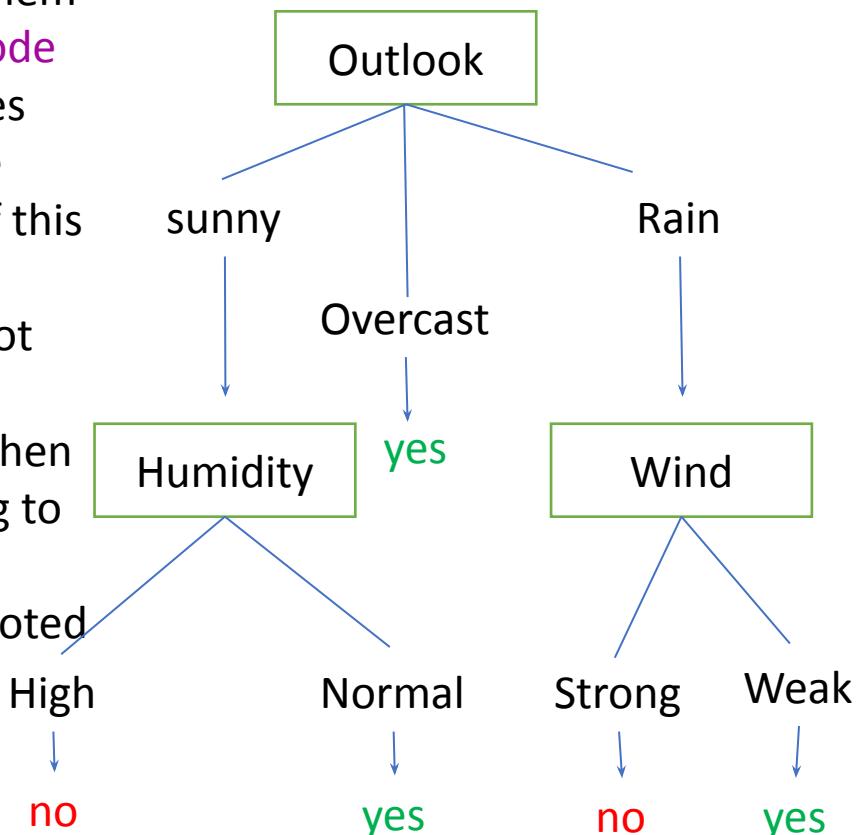


- 1 if (Leave at == 8am)  
OR  
(Leave at==9AM AND  
Accident == yes)  
OR  
( Leave at == 10am AND  
Stall= yes)
- 2 if(Leave at==9 AM AND  
Accident=NO)
- 3 if(Leave at==10 AM AND  
Stall= NO)

***“Decision tree learning is a method for approximating discrete-valued target function, in which the learned function is represented by a decision tree.”***

- Decision trees classify instances by sorting them down the tree from the root to **some leaf node**, which provides classification of the instances and each branch descending from that node corresponds to one of the possible values of this attribute.
- An instance is classified by starting at the root node of the tree
- **Testing** the attribute specified by this node then moving down the tree branch corresponding to value of the attribute.
- This process is then repeated for sub tree rooted at the new node

Decision tree for concept to “PlayTennis” or “ EnjoySports”

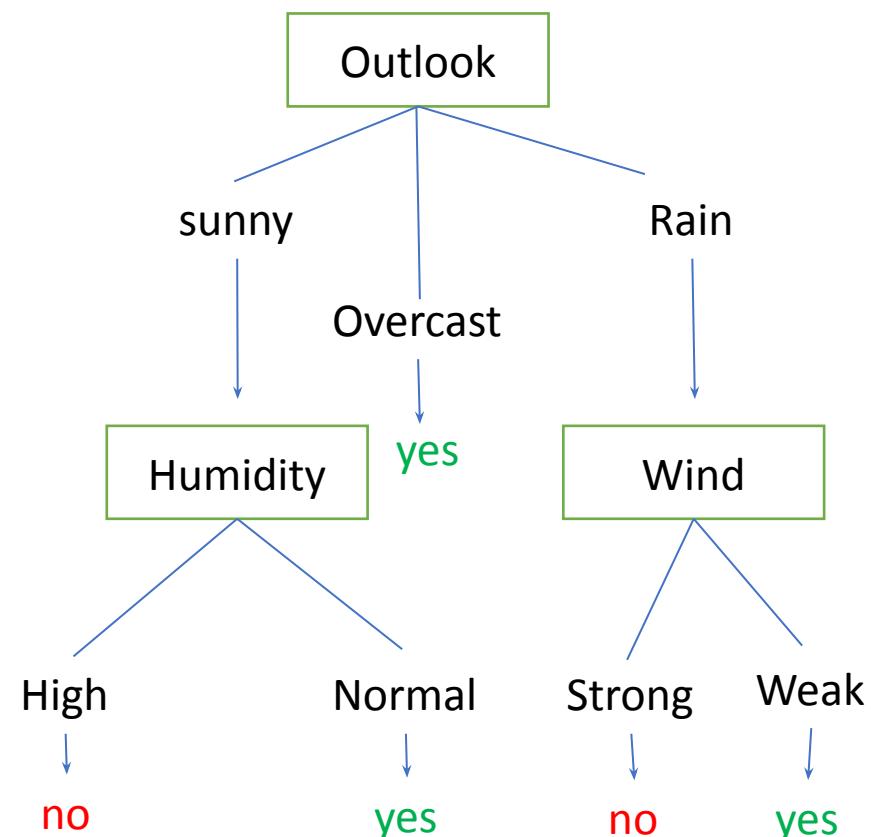


## Decision Function

- Based on this decision tree **lets try to think of the function that our machine may have learnt**

$f(\text{outlook}, \text{humidity}, \text{wind}) =$

```
{
  1 if (outlook=sunny and humidity=Normal)
     or
     (outlook=overcast)
     or
     (outlook=Rain and Wind=weak)
  0 for any other inputs
}
```



- The **leaf nodes** are the final classification categories (classification) or real values( regression).
- Decision trees are easy to understand and are explainable.

## Decision Tree

---

- A **decision tree** is a supervised machine learning model used to predict a target by learning decision rules from features.
- The preceding examples illustrates the concept of a decision tree based on categorical targets (**classification**), the same concept applies if our targets are real numbers (**regression**).
- It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- The final result is a tree with **decision nodes** and **leaf nodes**.

## Decision Tree for Regression

- A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested.
- Leaf node (e.g., Hours Played) represents a decision on the numerical target.
- The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.



## Decision Tree for Regression: CART

---

- **CART( Classification And Regression Tree)** is a variation of the decision tree algorithm. It can handle both classification and regression tasks.
- Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees(also called “growing” trees).
- CART algorithm uses **Gini Impurity** to split the dataset into a decision tree

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

Where  $p(j|t)$  is the fraction of examples belonging to class j at node t ([a solved example for constructing DT using Gini Index at the end](#))

## Appropriate Problem for Decision Tree Learning

---

- **Instances are Represented by attribute-value pairs**  
example: Temperature and their values
- **The target function has discrete output values**  
example the previous example of concept of playing tennis
- **Disjunctive descriptions may be required**  
As noted above, decision trees naturally represent disjunctive expressions.
- **The training data may contain errors**  
Decision tree learning methods are **robust to errors**, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
- **The training data may contain missing attribute values**  
Decision tree methods can be used even when some training examples have unknown values

## Splitting the node in a Decision Tree Learning

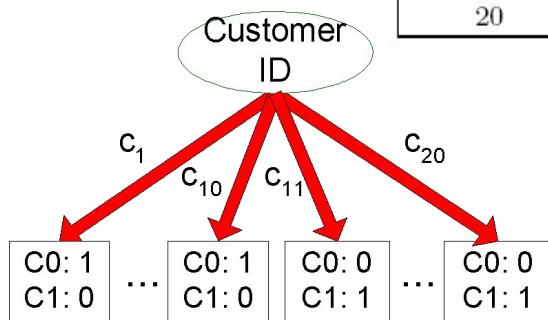
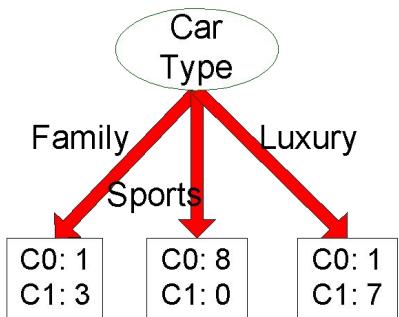
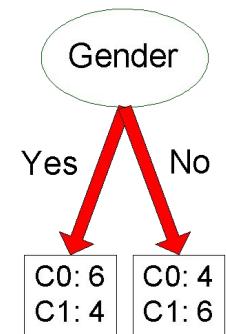
---

- **How should training examples be split?**
  - ❖ Method for specifying test condition depends on attribute types.
  - ❖ Measure for evaluating the goodness of a test condition.
  
- **How should the splitting procedure stop?**
  - ❖ Stop splitting if all the records belong to the same class or have identical attribute values
  - ❖ Early termination

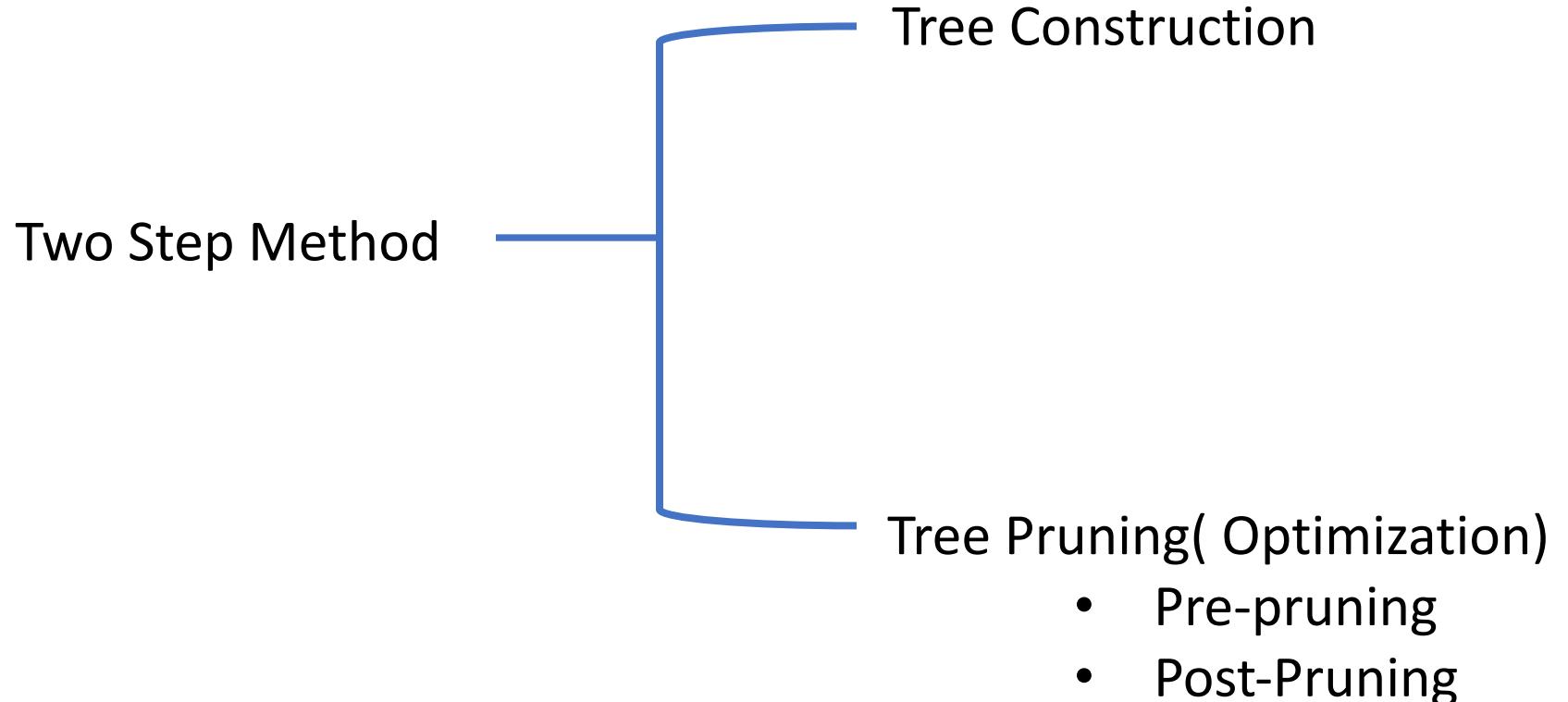
## Splitting in Decision Tree Learning

A few things about DTs:

1. It is Disjunctions of conjunctions( Inductive Bias)
2. It tends to look for short fat trees.
3. Makes use of a heuristics for best split



Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1



## How to determine the best split for Decision Tree Learning?

How to reduce the impurity??

Greedy approach:

Nodes with **purer** class distribution are preferred

Need a **measure of node impurity**:

C0: 5
C1: 5

Highly Impure Node

C0: 9
C1: 1

Almost Pure Node

C0 : 0
C1 : 10

Pure Node

## 1. Entropy

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

Fraction of examples  
belonging to class j at node t

## 2. Gini Index

$$\text{Gini}(t) = 1 - \sum_j [p(j|t)]^2$$

Already seen this formula before

## 3. Misclassification error

$$\text{Error}(t) = 1 - \max p(j|t)$$

## Entropy

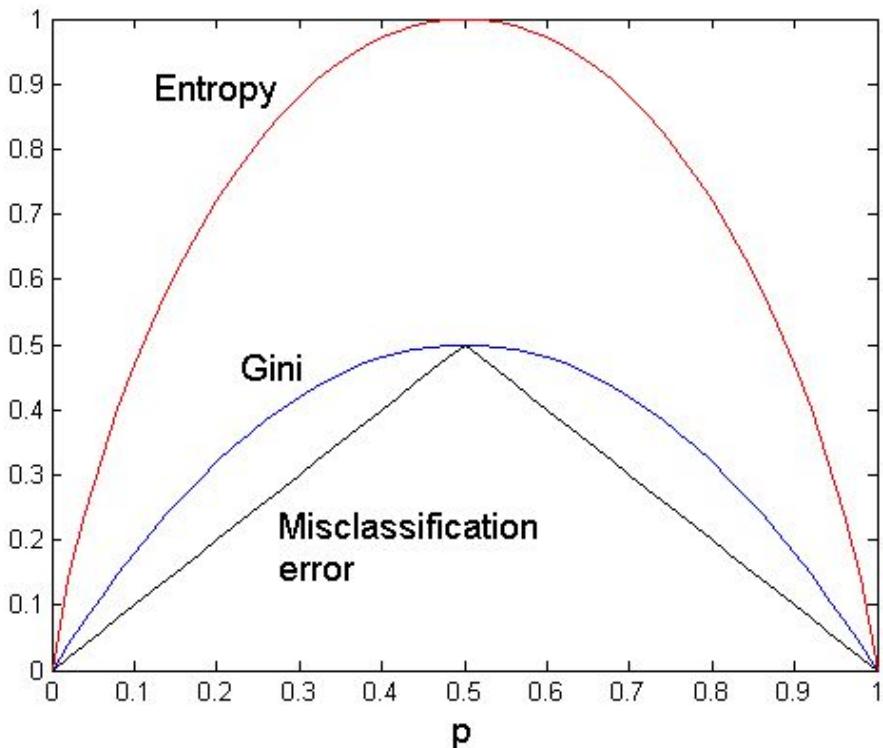
- In order to define *information gain*, a measure (commonly used in information theory) called **entropy** is used.
- Characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection  $S$ , containing positive (**p**) and negative (**n**) examples of some target concept, the entropy of  $S$  relative to this Boolean classification is

$$\text{Entropy}(S) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

uncertainty due to positive  
examples in data set

uncertainty due to  
negative examples in  
data set

## Comparison amongst Impurity Measures



C1	<b>0</b>
C2	<b>6</b>

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

$$\text{Entropy} = -0 \log_2 0 - 1 \log_2 1 = -0 - 0 = 0$$

C1	<b>1</b>
C2	<b>5</b>

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

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

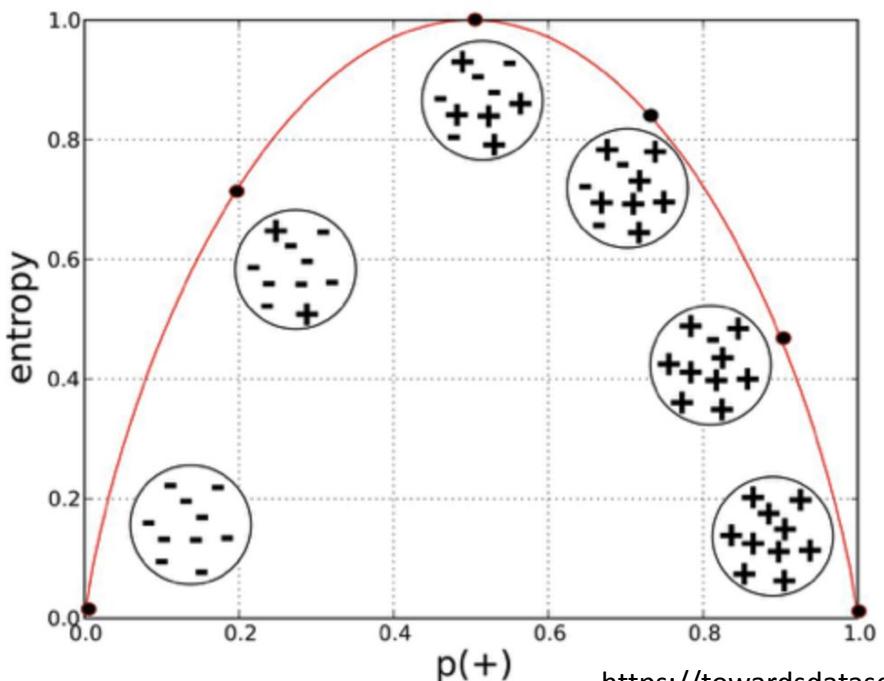
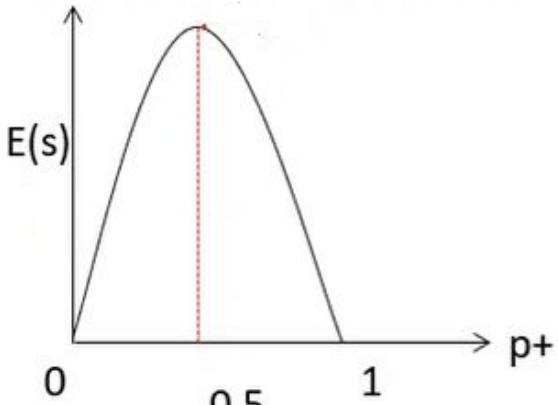
C1	<b>2</b>
C2	<b>4</b>

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

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

In a 2-class problem, class dist can be written as  $(p_0, p_1)$  where  $p_1=1-p_0$

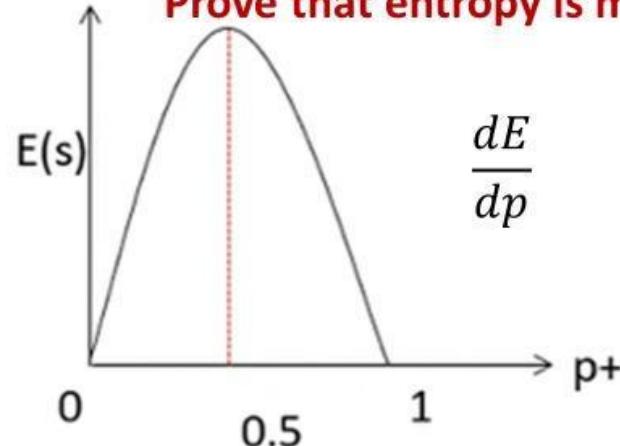
## Can we use Entropy directly as our heuristic?



- consider the plot of entropy vs probability of positive sample
- when  $p=0$  entropy is 0
- and even when  $p=1$  entropy is 0
- the maximum is at when  $p=0.5$
- this means that when  $p+=0.5$ , probability of correctly classifying positive samples is one half
- that also means probability of correctly classifying negative sample is also one half
- this is not useful as when  $E$  is maximum we should have either one of them as maximum  $p+$  or  $p-(n)$
- but here maximum entropy is achieved when all outcome is equally likely and minimum is achieved when one of them is certain
- hence we need some improvements in this and hence we go for another heuristic that is information gain

## Can we use Entropy directly as our heuristic?

Prove that entropy is maximum when  $p = 0.5$



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

Let us consider only 2 classes, i.e. binary classification.

$$\therefore \text{Entropy}(S) = - p \log_2 p - (1 - p) \log_2(1 - p)$$

Rewriting  $\text{Entropy}(S)$ ,

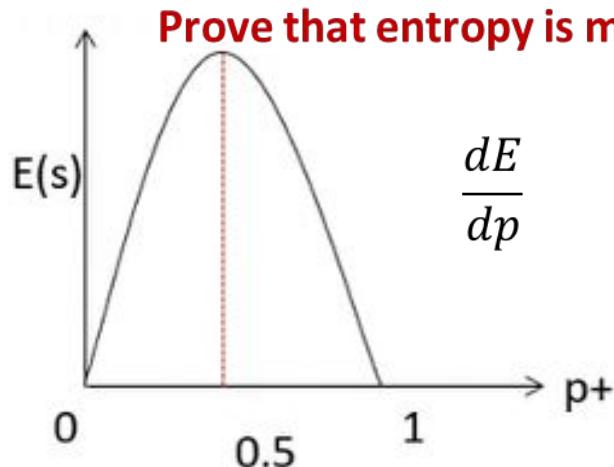
$$\text{Entropy}(S) = - \frac{1}{\log_e 2} [ p \log_e p + (1 - p) \log_e(1 - p) ]$$

Where  $p$  is the fraction of samples belonging to one class and remaining  $(1 - p)$  belongs to the other class.

In order to find the critical point of this entropy,  
lets take its derivative

$$\begin{aligned} \frac{dE(s)}{dp} &= - \frac{1}{\log_e 2} [ \log_e p + 1 + (1 - p) \times \frac{(-1)}{(1-p)} + (-1) \log_e(1 - p) ] \\ &= - \frac{1}{\log_e 2} [ \log_e p - \log_e(1 - p) ] = - \frac{1}{\log_e 2} \left[ \log_e \frac{p}{1-p} \right] \end{aligned}$$

## Can we use Entropy directly as our heuristic?



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

Let us consider only 2 classes, i.e. binary classification.

$$\therefore \text{Entropy}(S) = - p \log_2 p - (1 - p) \log_2 (1 - p)$$

Rewriting  $\text{Entropy}(S)$ ,

$$\text{Entropy}(S) = - \frac{1}{\log_e 2} [ p \log_e p - (1 - p) \log_e (1 - p) ]$$

Where  $p$  is the fraction of samples belonging to one class and remaining  $(1 - p)$  belongs to the other class.

In order to find the critical point of this entropy,  
lets take its derivative

$$\begin{aligned} \frac{dE(s)}{dp} &= - \frac{1}{\log_e 2} [ \log_e p + 1 + (1 - p) \times \frac{(-1)}{(1-p)} + (-1) \log_e (1 - p) ] \\ &= - \frac{1}{\log_e 2} [ \log_e p - \log_e (1 - p) ] = - \frac{1}{\log_e 2} \left[ \log_e \frac{p}{1-p} \right] \end{aligned}$$

## Can we use Entropy directly as our heuristic?

For Critical Points,  $\frac{dE}{dp} = 0$

$$\text{i.e. } -\frac{1}{\log_e 2} \left[ \log_e \frac{p}{1-p} \right] = 0$$

$$= \log_e \frac{p}{1-p} = \log_e 1$$

$$\text{or } \frac{p}{1-p} = 1$$

$$\text{or } p = 1 - p$$

$$\text{or } p = \frac{1}{2}$$

To find whether it is a maxima or minima, take the second derivative of  $E(S)$

$$\frac{d^2E}{dp^2} = -\frac{1}{\log_e 2} \left[ \frac{1}{p} + \frac{1}{1-p} \right]$$

Here  $p > 0$  &  $1 - p > 0$

$$\therefore \frac{d^2E}{dp^2} < 0$$

i.e.  $p = 0.5$  is a point of maxima. Thus,  $E(S)$  is max at  $p = 0.5$

## Can we use Entropy directly as our heuristic?

---

For Critical Points,  $\frac{dE}{dp} = 0$

$$\text{i.e. } -\frac{1}{\log_e 2} \left[ \log_e \frac{p}{1-p} \right] = 0$$

$$= \log_e \frac{p}{1-p} = \log_e 1$$

$$\text{or } \frac{p}{1-p} = 1$$

$$\text{or } p = 1 - p$$

$$\text{or } p = \frac{1}{2}$$

To find whether it is a maxima or minima, take the second derivative of  $E(S)$

$$\frac{d^2E}{dp^2} = -\frac{1}{\log_e 2} \left[ \frac{1}{p} + \frac{1}{1-p} \right]$$

Here  $p > 0$  &  $1 - p > 0$

$$\therefore \frac{d^2E}{dp^2} > 0$$

i.e.  $p = 0.5$  is a point of maxima. Thus,  $E(S)$  is max at  $p = 0.5$

# Information Gain

- Given entropy as a measure of the impurity in a collection of training examples, we can now define *a measure of the effectiveness of an attribute* in classifying the training data.
  - Is simply the expected reduction in entropy caused by partitioning the examples according to this attribute
  - Information Gain is used to test the goodness of a split**

$$\Delta = I(\text{parent node}) - \sum_{j=1}^k \frac{N(v_j)}{N} I(v_j)$$


 Impurity of parent node      Average weighted entropy  
OR Average Information

N is the total no. of instances at the parent node

k is total possible attribute values

$N(v_i)$  is the total no. of instances associated with child node  $v_i$

- Why Info Gain when we already have Entropy?
- In order to determine how well a test condition performs, compare the degree of impurity of parent node( before splitting) with the degree of impurity of child nodes( after splitting).
- The *larger the difference, better the condition.*

## ID3 Approach ( Iterative Dichotomizer 3)

---

- Our basic algorithm ,ID3 learns decision trees by constructing them top-down, beginning with question .  
**"which attribute should be tested at the root of the tree?"**
- The simple answer **statistical test which is information gain**
- We evaluate each instance attribute using statistical test to determine how well it **alone** classifies the training example.
- The best attribute is selected and used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node.
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree. This forms a greedy search for an acceptable decision tree, in which ***the algorithm never backtracks to reconsider earlier choices.***

Q: Consider the following training dataset for binary classification:

i) Calculate the entropy of this training sample

Solution:

As there are 4 + instances and  
5 – instances

Therefore, Entropy I of the training dataset,

$$I = -\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9}$$

$$= 0.9911$$

Instance	$a_1$	$a_2$	$a_3$	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
8	T	F	7.0	+
9	F	T	5.0	-

## Entropy-Calculation

What are the information gains of  $a_1$  and  $a_2$  relative to these training examples?

$$\text{Entropy at Node } N_1 \text{ for } a_1, = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} = 0.8113$$

$$\text{Entropy at Node } N_2 \text{ for } a_1, = 0.7219$$

Avg. wtd. Entropy -

$$= \frac{4}{9} (0.8113) + \frac{5}{9} (0.7129) = 0.7617$$

$$Info\ gain(a_1) = 0.9911 - 0.7617 = 0.2293$$

Similarly, Info gain corresponding to  $a_2$

$$\text{Entropy}(N_1) = 0.971, \quad \text{Entropy}(N_2) = 1$$

$$\text{Avg. wtd. Entropy} = 0.9838$$

$$\text{Info gain} = 0.9911 - 0.9838$$

$$= 0.0072$$

ID3 Can't handle continuous values of the attributes.  
So, CART & C4.5 came into existence.

## Information Gain- Calculation

ii) What are the information gains of  $a_1$  and  $a_2$  relative to these training examples?

For attribute  $a_1$ , the corresponding counts are:

$a_1$	+	-
T	3	1
F	1	4

The average wtd.entropy for  $a_1$  is

$$+ \frac{4}{9} \left[ - (3/4) \log_2(3/4) - (1/4) \log_2(1/4) \right] + \frac{5}{9} \left[ - (1/5) \log_2(1/5) - (4/5) \log_2(4/5) \right] = 0.7616.$$

Instance	$a_1$	$a_2$	$a_3$	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
8	T	F	7.0	+

Instance	$a_1$	$a_2$	$a_3$	Target Class
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
9	F	T	5.0	-

Therefore, the information gain for  $a_1$  is  $0.9911 - 0.7616 = 0.2294$ .

Information Gain- Calculation

---

For attribute a<sub>2</sub>, the corresponding counts are:

a <sub>2</sub>	+	-
T	2	3
F	2	2

The avg. wtd.  
entropy for a<sub>2</sub> is

$$\begin{aligned} & \frac{5}{9} \left[ -\left(\frac{2}{5}\right) \log_2\left(\frac{2}{5}\right) - \left(\frac{3}{5}\right) \log_2\left(\frac{3}{5}\right) \right] \\ & + \frac{4}{9} \left[ -\left(\frac{2}{4}\right) \log_2\left(\frac{2}{4}\right) - \left(\frac{2}{4}\right) \log_2\left(\frac{2}{4}\right) \right] = 0.9839. \end{aligned}$$

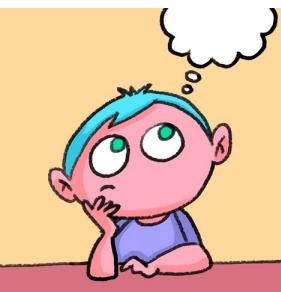
Therefore, the information gain for a<sub>2</sub> is 0.9911  
– 0.9839 = 0.0072.

## Information Gain- Calculation

iii) For  $a_3$ , which is a continuous attribute, compute the information gain for every possible split

Instance	$a_1$	$a_2$	$a_3$	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
8	T	F	7.0	+
9	F	T	5.0	-

For solving this part, we have to look into how to deal with continuous variables!!!



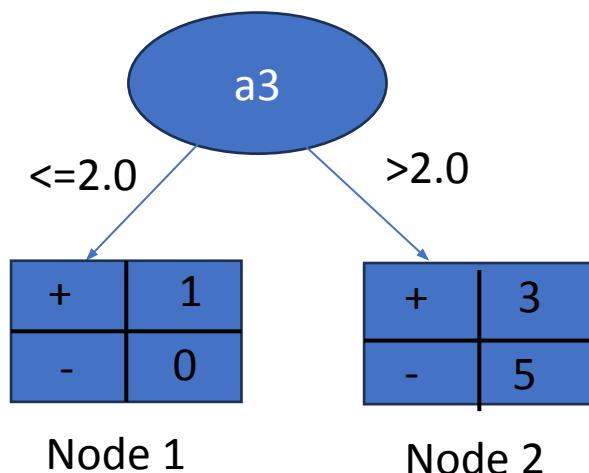
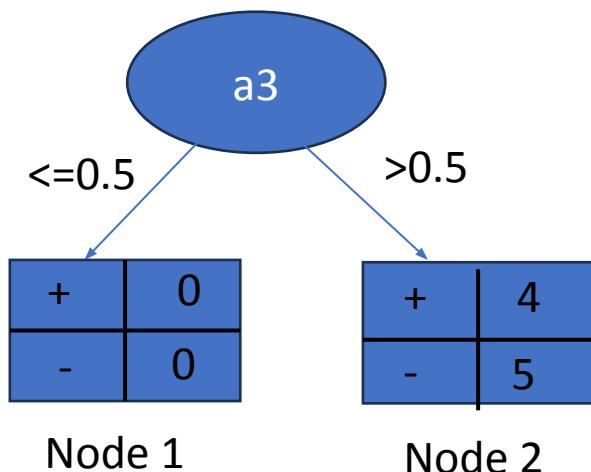
## Dealing with continuous variables

- 1. Sort the Variable:** Sort the continuous variable values in ascending order.
- 2. Identify Split Points:** Identify potential split points between each pair of consecutive values.
- 3. Evaluate Splits:** For each split point, divide the data and calculate impurity measures (e.g., Gini, entropy, variance).
- 4. Select Best Split:** Choose the split point with the greatest reduction in impurity.
- 5. Create the Split:** Create a decision node with the selected split point, splitting the data into two branches.
- 6. Repeat Process:** Repeat these steps for each branch until a stopping criterion is met.

## Dealing with continuous variables

For the given problem a3 is continuous variable:

1. Sort the values of a3: 1.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0
  2. Split points would be :  $\frac{1}{2} = 0.5$ ,  $\frac{1+3}{2} = 2.0$ ,  $\frac{3+4}{2} = 3.5$ ,  $\frac{4+5}{2} = 4.5$  and so on.
  3. For each split point, divide the data and compute the impurity measure:
- For split point 0.5, the dataset is divided as examples where  $a3 \leq 0.5$  and  $> 0.5$



And so on..... All the split points are shown on next slide along with the nodes

(iii) For a3, which is a continuous attribute, compute the information gain for every possible split.

## Entropy-Calculation for Continuous attributes

Entropy(S1) or I(S1)=0 therefore, Info Gain(S1)=0

1.0.      3.0.      4.0.      5.0.      6.0.      7.0

	0.5	2.0	3.5	4.5	5.5	6.5	7.5	
<=	>	<=	>	<=	>	<=	>	<=
+	0	4	1	3	1	3	2	0
-	0	5	0	5	1	4	1	1
	S1	S2	S3	S4	S5	S6	S7	

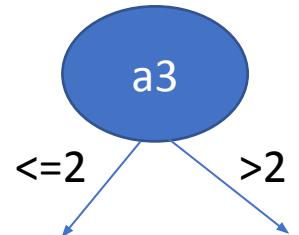
Instance	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	Target Class
1	T	T	1.0	+
2	T	T	6.0	+
3	T	F	5.0	-
4	F	F	4.0	+
5	F	T	7.0	-
6	F	T	3.0	-
7	F	F	8.0	-
8	T	F	7.0	+
9	F	T	5.0	-

Av. Wtd. Entropy(S2)=  $(1/9)[-(1/1)\log_2 1] + (8/9)[(-3/8)\log_2(3/8) - (5/8)\log_2(5/8)] = 0.8475$

**Info Gain(S1)**=I(parent)- Avg. wtd. Entropy(S2)=0.9911-0.8475=0.1435

Similarly, calculate the Info gain for all splits: Info gain( S3)=0.0026 and so on.

Since Info gain is maximum at the split point 2. All the values corresponding to different split points are given in the table on next slide.



## Information Gain- Calculation

iii) For  $a_3$ , which is a continuous attribute, compute the information gain for every possible split

$a_3$	Class label	Split point	Entropy	Info Gain
1.0	+	2.0	0.8484	0.1427
3.0	-	3.5	0.9885	0.0026
4.0	+	4.5	0.9183	0.0728
5.0	-			
5.0	-	5.5	0.9839	0.0072
6.0	+	6.5	0.9728	0.0183
7.0	+			
7.0	-	7.5	0.8889	0.1022

The best split for  $a_3$  occurs at split point equals to

iv) What is the best split (among  $a_1$ ,  $a_2$ , and  $a_3$ ) according to the information gain?

**Answer:** According to information gain,  $a_1$  produces the best split.



Is it necessary that all the leaf nodes are pure nodes in a Decision Tree?

- No, it is not necessary for all leaf nodes to be pure nodes in a decision tree.
- In fact, it is quite common for decision trees to have impure (non-pure) leaf nodes, especially when dealing with real-world datasets that may have noise, uncertainty, or overlapping features.
- Achieving pure leaf nodes can sometimes lead to overfitting, where the tree captures noise and fluctuations in the training data that don't generalize well to new, unseen data.

Step 1: Data Preprocessing: Clean and preprocess the data.

Step 2: Selecting the Root Node using Entropy for the target variable.

Step 3: Calculating Information Gain of each attribute.

Step 4: Selecting the Best Attribute with highest Info Gain.

Step 5: Splitting the Dataset.

Step 6: Repeat the Process step 2 to 5 till the desired tree is obtained.

- ID3 can handle training data with discrete and symbolic attribute values only.
- In order to deal with continuous-valued attribute , ID3 must treat them as discrete attribute with many possible values,

## Decision Tree Using Gini Index : Practice Time

Create a decision tree for the following data using Gini Index.

Income	Student	Credit Rating	Buys computer
High	No	Fair	No
High	No	Excellent	No
High	Yes	Fair	Yes
High	Yes	Excellent	Yes
Medium	Yes	Fair	Yes
Low	Yes	Fair	Yes
Low	No	Excellent	No
Low	Yes	Excellent	Yes
Medium	No	Fair	No
Low	No	Fair	No
Medium	Yes	Excellent	Yes
Medium	No	Excellent	Yes

Gini Index of Dataset (7 Yes, 5 No) -

$$Gini(S) = 1 - \left[ \left( \frac{7}{12} \right)^2 + \left( \frac{5}{12} \right)^2 \right] = 0.4861$$

Gini Index of Income = High (2 Yes, 2 No) -

$$Gini(\text{Income} = \text{High}) = 1 - \left[ \left( \frac{2}{4} \right)^2 + \left( \frac{2}{4} \right)^2 \right] = 0.5$$

Gini Index of Income = Medium (3 Yes, 1 No) -

$$Gini(\text{Income} = \text{Medium}) = 1 - \left[ \left( \frac{3}{4} \right)^2 + \left( \frac{1}{4} \right)^2 \right] = 0.3750$$

Gini Index of Income = Low (2 Yes, 2 No) -

$$Gini(\text{Income} = \text{Low}) = 1 - \left[ \left( \frac{2}{4} \right)^2 + \left( \frac{2}{4} \right)^2 \right] = 0.5$$

Gini Index of Income -

$$Gini(\text{Income}) = \frac{4}{12} \times 0.5 + \frac{4}{12} \times 0.375 + \frac{4}{12} \times 0.5 = 0.4583$$

Info Gain of Income -

$$\text{InfoGain}(\text{Income}) = 0.4861 - 0.4583 = 0.0278$$

## Solution

---

Gini Index of Student = Yes (6 Yes, 0 No) -

$$Gini(Student = Yes) = 1 - \left[ \left( \frac{6}{6} \right)^2 + \left( \frac{0}{6} \right)^2 \right] = 0$$

Gini Index of Student = No (1 Yes, 5 No) -

$$Gini(Student = No) = 1 - \left[ \left( \frac{1}{6} \right)^2 + \left( \frac{5}{6} \right)^2 \right] = 0.2778$$

Gini Index of Student -

$$Gini(Student) = \frac{6}{12} \times 0 + \frac{6}{12} \times 0.2778 = 0.1389$$

Info Gain of Student -

$$InfoGain(Student) = 0.4861 - 0.1389 = 0.3472$$

Gini Index of Credit Rating = Excellent (4 Yes, 2 No) -

$$Gini(Credit\ Rating = Excellent) = 1 - \left[ \left( \frac{4}{6} \right)^2 + \left( \frac{2}{6} \right)^2 \right] = 0.4444$$

Gini Index of Credit Rating = Fair (3 Yes, 3 No) -

$$Gini(Credit\ Rating = Fair) = 1 - \left[ \left( \frac{3}{6} \right)^2 + \left( \frac{3}{6} \right)^2 \right] = 0.5$$

Gini Index of Credit Rating -

$$Gini(Credit\ Rating) = \frac{6}{12} \times 0.4444 + \frac{6}{12} \times 0.5 = 0.4722$$

Info Gain of Credit Rating -

$$InfoGain(Credit\ Rating) = 0.4861 - 0.4722 = 0.0139$$

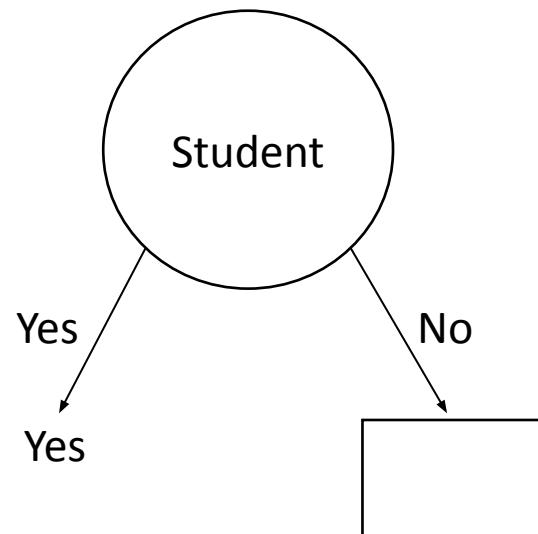
Comparing Info Gains -

$$\text{InfoGain}(Income) = 0.0278$$

$$\text{InfoGain}(Student) = 0.3472$$

$$\text{InfoGain}(Credit\ Rating) = 0.0139$$

Hence the first attribute chosen will be Student -



Dataset with Student = No -

Income	Credit Rating	Buys Computer
High	Fair	No
High	Excellent	No
Low	Excellent	No
Low	Fair	No
Medium	Fair	No
Medium	Excellent	Yes

## Solution

---

Gini Index of Dataset Student = No (1 Yes, 5 No) -

$$Gini(S) = 1 - \left[ \left( \frac{1}{6} \right)^2 + \left( \frac{5}{6} \right)^2 \right] = 0.2778$$

Gini Index of Income = High (0 Yes, 2 No) -

$$Gini(\text{Income} = \text{High}) = 1 - \left[ \left( \frac{0}{2} \right)^2 + \left( \frac{2}{2} \right)^2 \right] = 0$$

Gini Index of Income = Medium (1 Yes, 1 No) -

$$Gini(\text{Income} = \text{Medium}) = 1 - \left[ \left( \frac{1}{2} \right)^2 + \left( \frac{1}{2} \right)^2 \right] = 0.5$$

Gini Index of Income = Low (0 Yes, 2 No) -

$$Gini(\text{Income} = \text{Low}) = 1 - \left[ \left( \frac{0}{2} \right)^2 + \left( \frac{2}{2} \right)^2 \right] = 0$$

Gini Index of Income -

$$Gini(\text{Income}) = \frac{2}{6} \times 0 + \frac{2}{6} \times 0.5 + \frac{2}{6} \times 0 = 0.1667$$

Info Gain of Income -

$$\text{InfoGain}(\text{Income}) = 0.2778 - 0.1667 = 0.1111$$

## Solution

---

Gini Index of Credit Rating = Excellent (1 Yes, 2 No) -

$$Gini(Credit\ Rating = Excellent) = 1 - \left[ \left( \frac{1}{3} \right)^2 + \left( \frac{2}{3} \right)^2 \right] = 0.4444$$

Gini Index of Credit Rating = Fair (0 Yes, 3 No) -

$$Gini(Credit\ Rating = Fair) = 1 - \left[ \left( \frac{0}{3} \right)^2 + \left( \frac{3}{3} \right)^2 \right] = 0$$

Gini Index of Credit Rating -

$$Gini(Credit\ Rating) = \frac{3}{6} \times 0 + \frac{3}{6} \times 0.4444 = 0.2222$$

Info Gain of Credit Rating -

$$InfoGain(Credit\ Rating) = 0.2778 - 0.2222 = 0.0556$$

## Solution

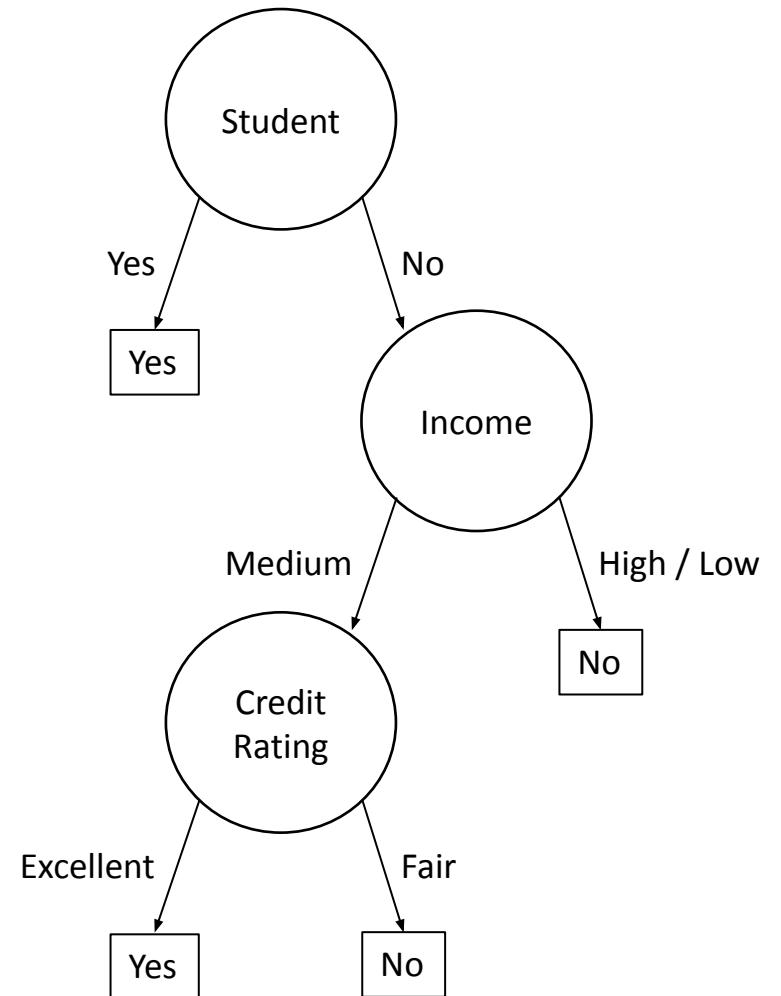
---

Comparing Info Gains -

$$\text{InfoGain}(Income) = 0.1111$$

$$\text{InfoGain}(Credit\ Rating) = 0.0556$$

Hence the second attribute chosen will be Income  
and the third attribute chosen will be Credit Rating -



**Another Problem( Practice Problem)**

Another Example:

Consider the following dataset consisting of 3 attributes and one target variable shape. Construct a DT using the given dataset to classify an object as Triangle or square.

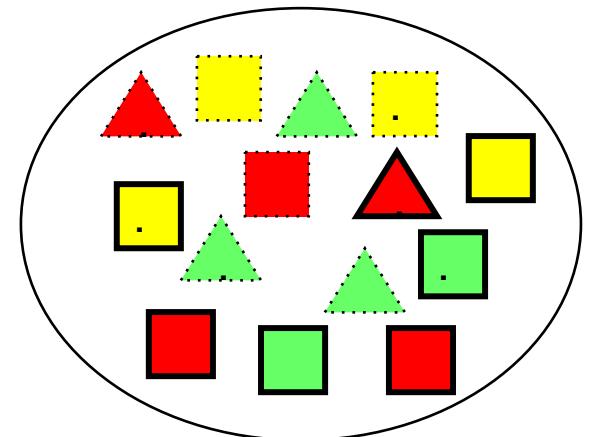
(Calculate Entropy as well as Gini Index to form the DT)

#	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

## Information Gain- Calculation

#	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

Data Set:  
A set of classified objects



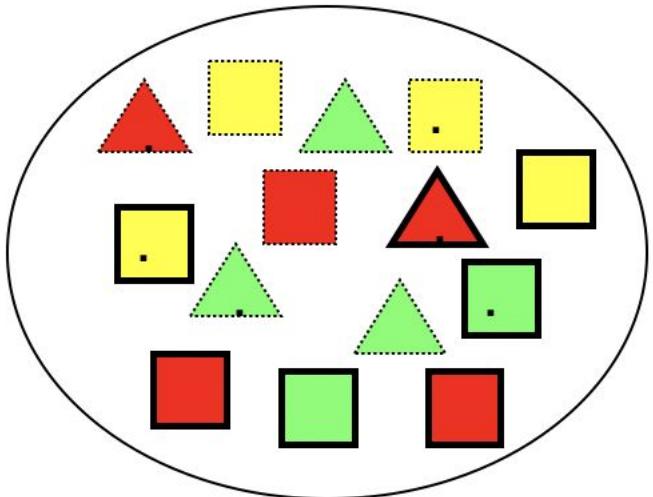
Remember

$$\log_a b = \frac{\ln b}{\ln a}$$

$$0 \log_2 0 = 0$$

1. Calculate the entropy of the training dataset.
2. Calculate info gain wrt color, outline and dot attr.

## Information Gain- Calculation



- 5 triangles
- 9 squares
- class probabilities

$$p(\square) = \frac{9}{14}$$

$$p(\Delta) = \frac{5}{14}$$

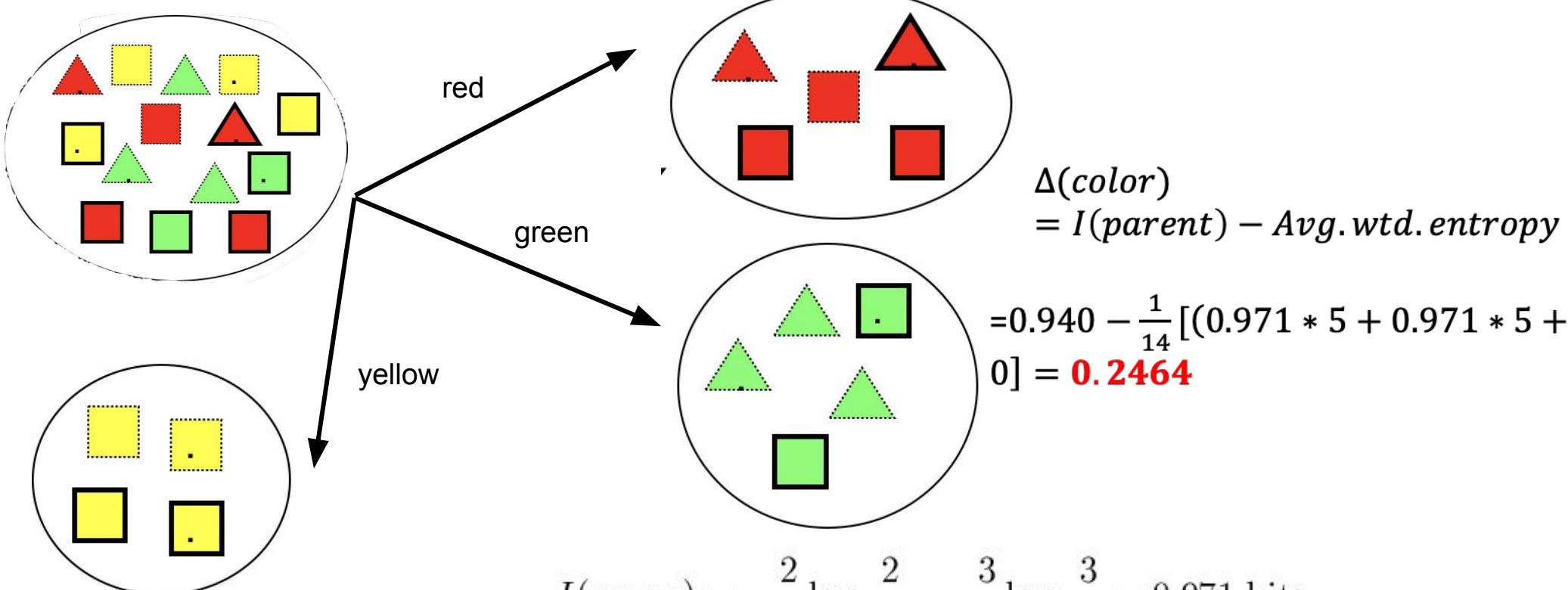
Entropy of the training sample,  $I$  is given by,

$$I = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940 \text{ bits}$$

## Entropy-Calculation

Consider the attribute *Color*

$$I(red) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971 \text{ bits}$$



$$I(yellow) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0.0 \text{ bits}$$

## Entropy-Calculation

For Outline,

	Δ	□	Entropy, I
Dashed	4	3	0.9878
Solid	1	6	0.5913

Since Info Gain wrt Color is highest so Color will be the root node and Training Dataset will get split into 2 sub datasets having color – red and green as color= yellow has already become the pure node.

$$\text{Info.gain(outline)} = I(\text{parent}) - \frac{1}{14} [7 * 0.9878 + 7 * 0.5913] = 0.940 - 0.78955 = 0.151$$

For Dot,

	Δ	□	Entropy,I
Yes	3	3	1
No	2	6	0.8113

$$\text{Info.gain(Dot)} = I(\text{parent}) - \frac{1}{14} [1 * 6 + 0.8113 * 8] = 0.940 - 0.8926 = 0.0474$$

For Color Red, the subset of Training Dataset is

4	red	dashed	no	square
5	red	solid	no	square
6	red	solid	yes	triangle
10	red	solid	no	square
14	red	dashed	yes	triangle

	$\Delta$	$\square$	Entropy, I
Dashed	1	1	1
Solid	1	2	0

$$I(\text{Red color, when outline} = \text{dashed}) = ???$$

$$I(\text{Red color, when outline} = \text{solid}) = ???$$

For Green,

1	green	dashed	no	triangle
2	green	dashed	yes	triangle
7	green	solid	no	square
8	green	dashed	no	triangle
11	green	solid	yes	square

	$\Delta$	$\square$	Entropy, I
Dashed	3	0	0
Solid	0	2	0

$$I(\text{Green color, outline} = \text{dashed}) = ???$$

Computation of Entropy and Info gain for **Outline** ( previous slide)

$$\text{Entropy } I(\text{for Outline=Dashed}) = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) \\ = 1$$

$$\text{Entropy } I(\text{Outline=Solid}) = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) \\ = 0.9177$$

$$\text{Entropy ( parent node)} = -(2/5)\log_2(2/5) - (3/5)\log_2(3/5) = 0.971$$

$$\Delta(\text{Outline}) = 0.971 - (1/5)[2*1 + 0.9177*3] = \textcolor{red}{0.02038}$$

Avg wtd Entropy

Computation of Entropy and Info gain for **Dot**

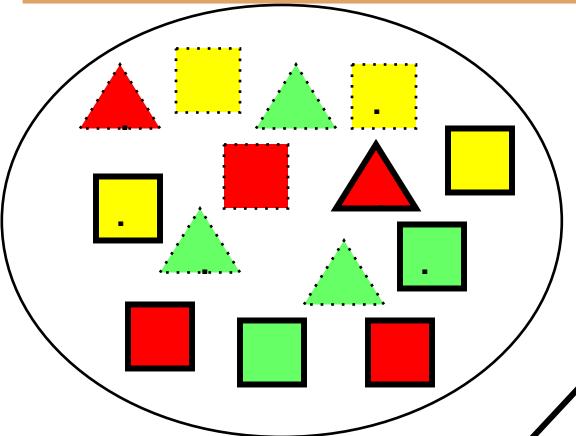
$$\Delta(\text{Dot}) = \textcolor{purple}{0.971}$$

Outline	$\Delta$	$\square$	Entropy, I
Dashed	1	1	1
Solid	1	2	0.9177

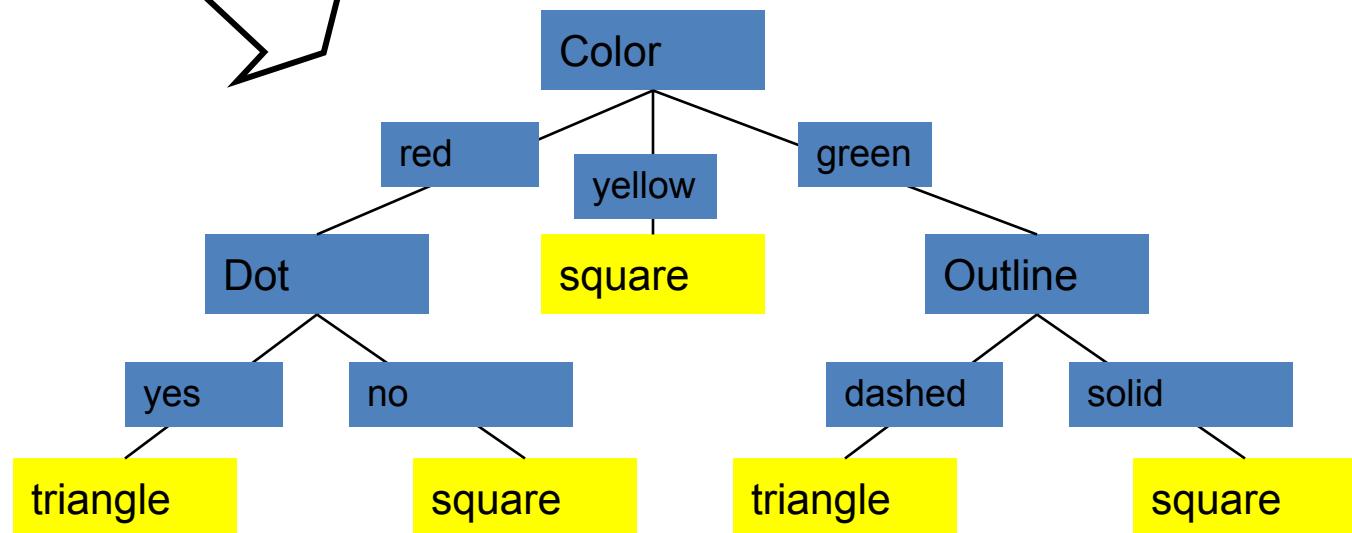
Dot	$\Delta$	$\square$	Entropy, I
Yes	2	0	0
No	0	3	0

As  $\Delta(\text{Dot}) = 0.971$  is more than Outline, therefore Dot will be considered as next node for split.

## Information Gain- Calculation



- Heuristics: attribute with the highest gain is chosen
- Amongst all the Info Gains, **Color has the highest** so Color is the **root node**. Then Dot and Outline are considered at the next level.



Note: ENTROPY =0 when all samples are of one class

ENTROPY=1 when all class have equal samples

## Another Example( For Practice)

Consider the following data set of concept of accepting a job

First we will calculate the entropy of entire data set S

Let us first recall the formula  $\text{Entropy}(t) = - \sum p(j|t) \log_2 p(j|t)$

Now let us calculate total number of positive and negative points

number of positive samples p=3

number of negative samples n = 4

salary	Location	job acceptance
Tier1	MUM	YES
Tier 2	BLR	YES
Tier 1	BLR	NO
Tier 1	HYD	NO
Tier 2	MUM	YES
Tier 1	HYD	NO
Tier 1	HYD	NO

$$\begin{aligned}
 \text{Entropy}(S) &= \frac{-3}{3+4} \log_2 \left( \frac{3}{3+4} \right) - \frac{4}{4+3} \log_2 \left( \frac{4}{4+3} \right) \\
 &= -0.428 \times (-1.222) - 0.5714 \times -0.8 \\
 &= 0.98
 \end{aligned}$$

## Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{n+p} \log_2 \left( \frac{n}{n+p} \right)$$

Now let us calculate the entropy for the attribute salary  
 Salary takes two values tier 1 and tier 2 so we need to split our data  
 we will first calculate entropy of salary=Tier1  
 number of positive points = 1      number of negative points = 6

$$\begin{aligned}\text{Entropy}(\text{Salary}=\text{Tier 1}) &= \frac{-1}{1+6} \log_2 \left( \frac{1}{1+6} \right) - \frac{6}{1+6} \log_2 \left( \frac{6}{1+6} \right) \\ &= 0.40107 - 0.2224. \\ &= 0.62347\end{aligned}$$

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543

# MACHINE LEARNING

## Entropy Calculation

$$\text{Entropy}(S) = \frac{-p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{n+p} \log_2 \left( \frac{n}{n+p} \right)$$

we will calculate entropy of salary=Tier2

number of positive points = 2

number of negative points = 0

$$\text{Entropy}(\text{Salary} = \text{Tier2}) = \frac{-2}{2+0} \log_2 \left( \frac{2}{2+0} \right) - \frac{0}{2+0} \log_2 \left( \frac{0}{2+0} \right)$$
$$= 0 - 0$$

$$= 0$$

Note: ENTROPY =0 when all samples are of one class  
ENTROPY=1 when all class have equal samples

salary	Location	job acceptance
Tier1	MUM	YES
Tier 1	BLR	NO
Tier 1	HYD	NO

salary	Location	job acceptance
Tier 2	BLR	YES
Tier 2	MUM	YES

salary	entropy
Tier1	0.543
Tier2	0

## Average Information or Avg. weighted Entropy

---

The statistical term Average Information of a attribute is given by

$$I(\text{Attribute}) = \sum \frac{p_i + n_i}{p + n} \text{Entropy}(A)$$

Lets us understand this by using the previous calculations we did

now

,

$$I(\text{SALARY}) = \frac{p_{\text{tier1}} + n_{\text{tier1}}}{p + n} \text{Entropy}(\text{salary} = \text{tier1}) +$$

$$\frac{p_{\text{tier2}} + n_{\text{tier2}}}{p + n} \text{Entropy}(\text{salary} = \text{tier2})$$

salary	entropy
Tier1	0.543
Tier2	0

$$I(\text{SALARY}) = \frac{1+7}{3+7} \times 0.543 + \frac{2+0}{3+7} \times 0$$

$$=0.4344$$

## Information Gain

---

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data
- The measure we will use, called **information gain**
- Is simply the expected reduction in entropy caused by partitioning the examples according to this attribute
- More precisely Information gain  $G(S,A)$  of an attribute A relative to collection of example S is defined by

$$G(S, A) = \text{ENTROPY}(S) - I(A)$$

that is differences of entropy of the collection of example S and information gain of the attribute A

## Information Gain- Calculation

---

from our previous calculation we have the following data

Entropy(S)	0.98
I(salary)	0.4344

let us calculate the Information gain G(S, Salary)

$$G(S, A) = \text{ENTROPY}(S) - I(A)$$

$$G(S, \text{SALARY}) = \text{ENTROPY}(S) - I(\text{SALARY})$$

$$\begin{aligned}G(S, \text{SALARY}) &= 0.98 - 0.4344 \\&= 0.5456\end{aligned}$$

## The ID3 Algorithm- problem

Let us now use all the previous knowledge and create a decision tree of the following data set

Lets create decision tree for this by following the steps

step 1: COMPUTE THE ENTROPY FOR DATA-SET

**ENTROPY(S)**

$$\text{Entropy}(S) = \frac{p}{p+n} \log_2 \left( \frac{p}{p+n} \right) - \frac{n}{n+p} \log_2 \left( \frac{n}{n+p} \right)$$

number of positive points = 9

number of negative points = 5

$$\text{Entropy}(S) = \frac{-9}{9+5} \log_2 \left( \frac{9}{9+5} \right) - \frac{5}{5+9} \log_2 \left( \frac{5}{5+9} \right)$$

$$=0.94$$

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Overcast	High	High	Weak	Yes
Rainy	Medium	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rainy	Medium	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes
Overcast	Medium	High	Strong	Yes
Overcast	High	Normal	Weak	Yes
Rainy	Medium	High	Strong	No

## The ID3 Algorithm- problem

For outlook=sunny

number of positive points = 2      number of negative points = 3

$$\text{Entropy}(\text{outlook} = \text{sunny}) = \frac{-2}{2+3} \log_2\left(\frac{2}{2+3}\right) - \frac{3}{2+3} \log_2\left(\frac{3}{2+3}\right)$$

$$= 0.971$$

Similarly doing it for rainy and overcast we have the following results

$$\text{Entropy}(\text{outlook} = \text{overcast}) = -1 \log_2(1) - 0 \log_2(0) = 0$$

$$\text{Entropy}(\text{outlook} = \text{rainy}) = \frac{-3}{2+3} \log_2\left(\frac{3}{2+3}\right) - \frac{2}{2+3} \log_2\left(\frac{2}{2+3}\right) = 0.971$$

Now we calculate Average information of the attribute outlook

$$I(\text{outlook}) = \frac{3+2}{9+5} * 0.971 + \frac{2+3}{9+5} * 0.971 + \frac{4+0}{9+5} * 0 = 0.693$$

Finally we calculate Information gain for the attribute outlook

$$G(S, \text{outlook}) = 0.94 - 0.693 = 0.247$$

Outlook	Play tennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

property	value
Entropy(s)	0.94
G(outlook)	0.247

## The ID3 Algorithm- problem

Repeat the same procedures for other table and the obtain the following result

property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

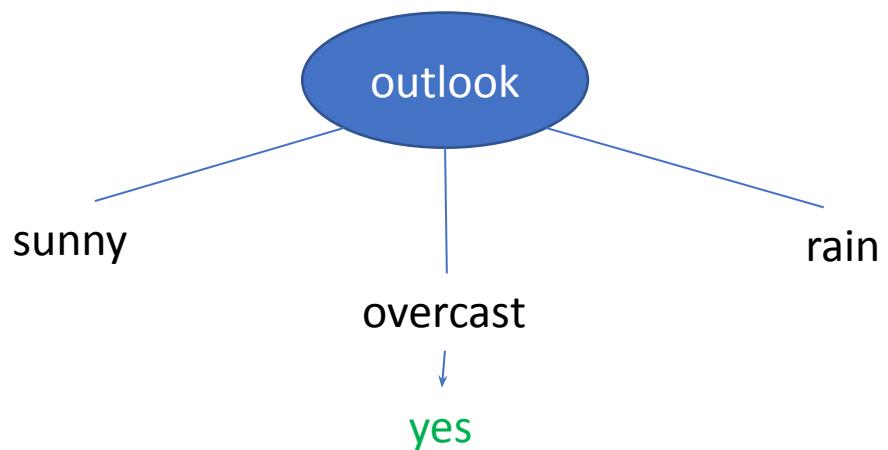
## The ID3 Algorithm- problem

3. PICK THE HIGHEST GAIN ATTRIBUTE

4. REPEAT UNTIL WE GET THE TREE WE DESIRED

we will now create our tree using the 3rd and 4th step

we choose our root node as outlook and split according to its three values



property	value
Entropy(s)	0.94
G(outlook)	0.247
G(temp)	0.029
G(humidity)	0.152
G(windy)	0.048

sunny and rainy still requires splitting so we repeat our procedures for  
outlook=sunny and outlook=rainy

## The ID3 Algorithm- problem

with outlook=sunny our data would look something like this

first we calculate entropy of the data set

$$p=2 \ n=3$$

$$\text{Entropy}(S_{\text{sunny}}) = \frac{-2}{2+3} \log_2\left(\frac{2}{2+3}\right) - \frac{3}{3+2} \log_2\left(\frac{3}{3+2}\right) = 0.97$$

next we need to consider each attribute and calculate its gain

Outlook	Temp	Humidity	Windy	Play tennis
Sunny	High	High	Weak	No
Sunny	High	High	Strong	No
Sunny	Medium	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Medium	Normal	Strong	Yes

## The ID3 Algorithm- problem

considering temperature attribute

Temperature	p	n	Entropy
cool	1	0	0
high	0	2	0
medium	1	1	1

Outlook	Temp	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	Medium	No
Sunny	Cool	Yes
Sunny	Medium	Yes

Average Information Entropy:  $I(\text{Temp})=0.4$

Gain :  $G(\text{Temp})=0.571$

## The ID3 Algorithm- problem

considering humidity attribute

Temperature	p	n	Entropy
normal	2	0	0
high	0	3	0

Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

Average Information Entropy:  $I(\text{Temp})=0$

Gain :  $G(\text{Temp})=0.971$

## The ID3 Algorithm- problem

considering windy attribute

Temperature	p	n	Entropy
strong	1	1	1
weak	1	2	0.918

Outlook	Windy	Play tennis
Sunny	Weak	No
Sunny	Strong	No
Sunny	Weak	No
Sunny	Weak	Yes
Sunny	Strong	Yes

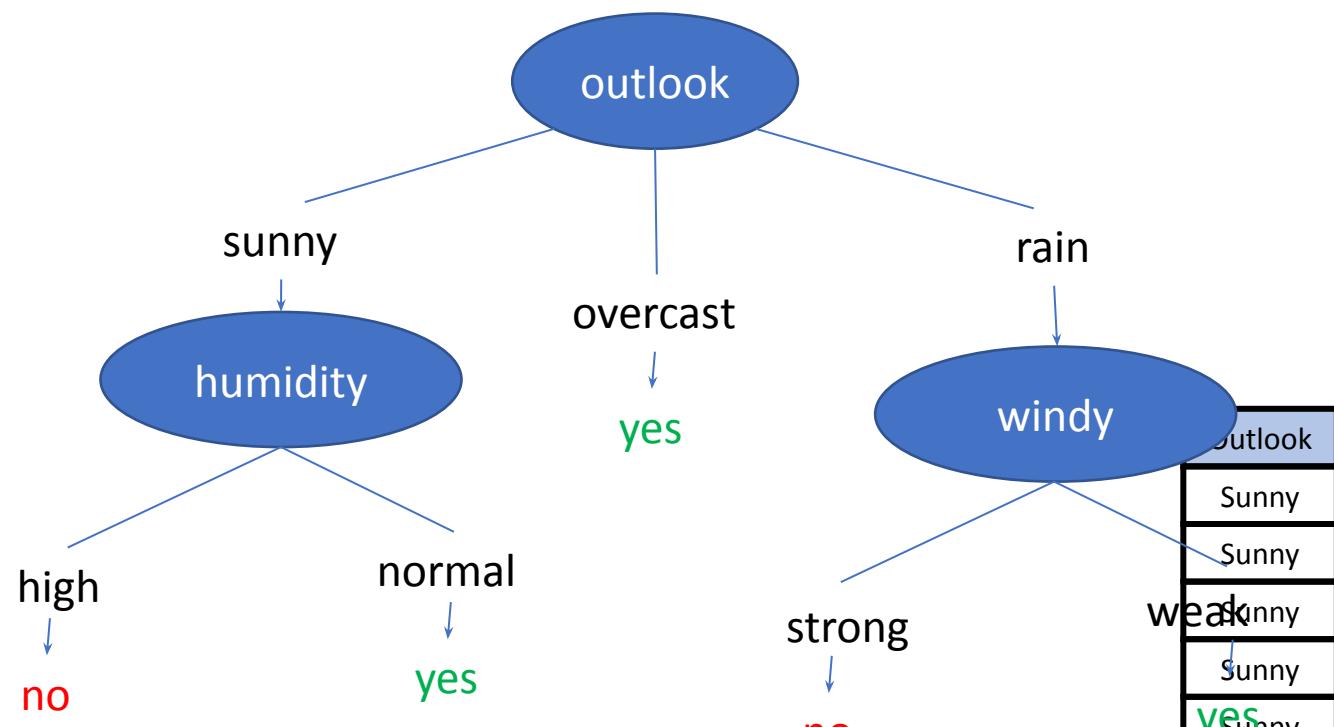
Average Information Entropy:  $I(\text{Temp})=0.951$

Gain :  $G(\text{Temp})=0.020$

## The ID3 Algorithm- problem

after the previous calculations we have the following data for the data with outlook=sunny

humidity has the highest gain hence we choose humidity at second level after outlook as sunny



property	value
Entropy( $S_{\text{sunny}}$ )	0.97
G(temp)	0.571
G(humidity)	0.971
G(windy)	0.02

Outlook	Humidity	Play tennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

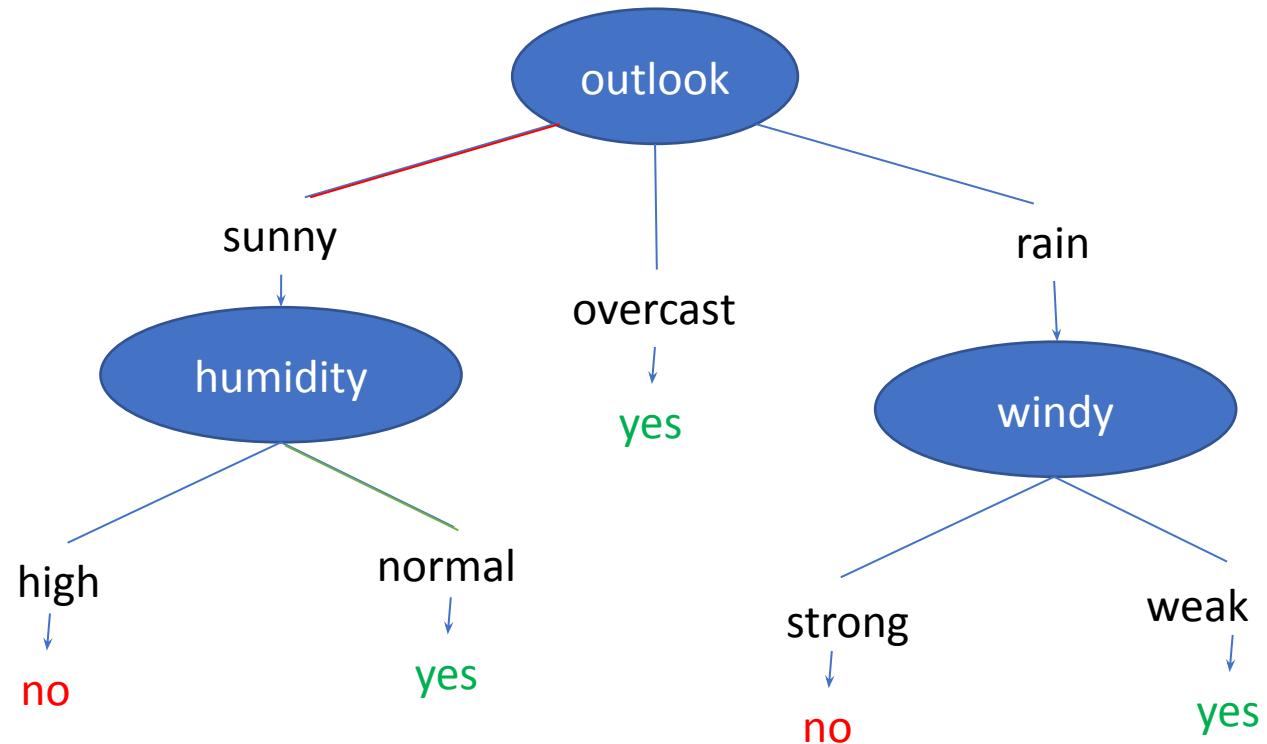
doing the same procedure for the table with outlook=rain we expand the rain leading node to get the following decision tree

## Decision Tree analysis

Let us see how decision tree helps us predicting if a player will play tennis or not

on a given day let this be the report of the weather forecast

***"the day would be sunny with normal humidity and weak wind"***



## Few facts

---

### About Entropy

- ✓ At leaf node, Entropy =0
- ✓ A formal *definition of Entropy from coding theory* is “ Entropy of a sample is **expected number of bits needed to encode a particular class(+,-)** of randomly drawn class from S ( to be done under optimal shortest length code)”
- ✓ Entropy is not a very good measure of impurity bcoz it tends to **favour attributes that have large number of distinct values.**
- ✓ Entropy of a dataset or instance space can be  $>1$  (bcoz its not probability).
- ✓ Say entropy of a sample= 1.35, 1.35 would translate into average depth of the tree.
- ✓ Avg wtd Entropy is also called as **Average Information.**

### About ID3

- ❖ Its greedy approach bcoz it picks up the attribute for split with highest information gain.
- ❖ Language of DT is Disjunction of Conjunction.
- ❖ It performs better on shorter trees than on longer trees.
- ❖ It does a complete search over the hypothesis space given a training sample.

## Practice Time

---

Q1: Draw the full decision tree for the parity function of four Boolean attributes, A, B, C, and D intuitively.

Q2: Consider the following training dataset:

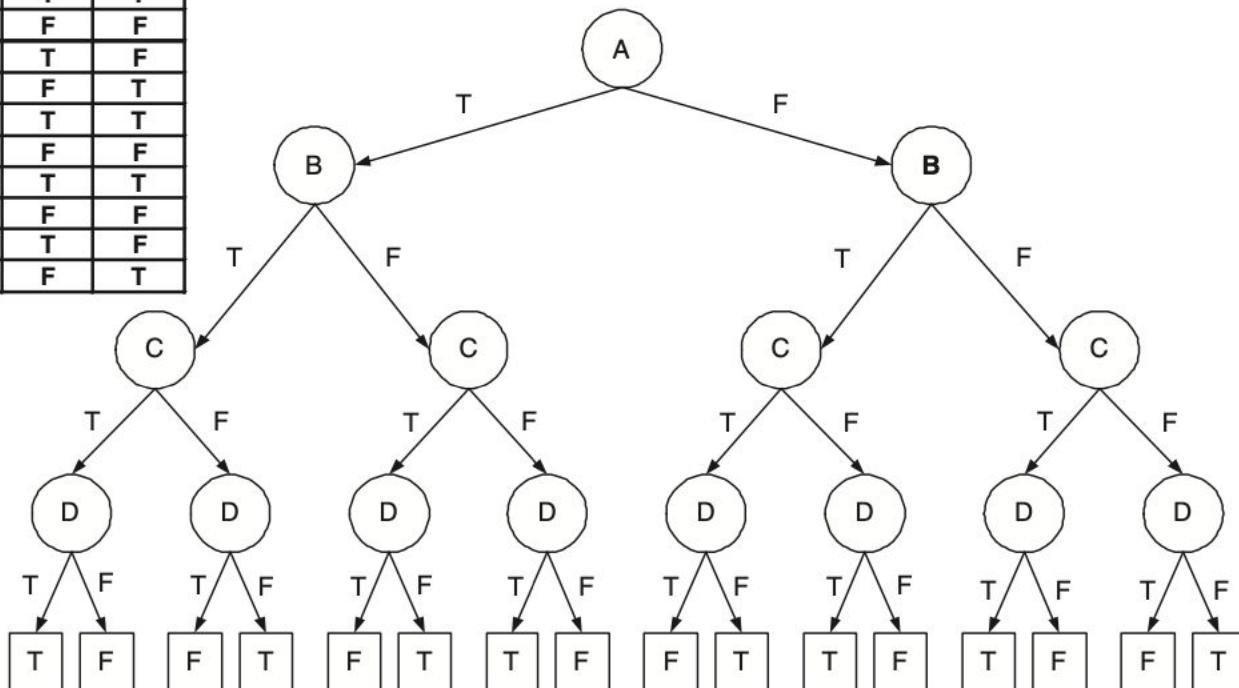
A	B	Class Label
T	F	+
T	T	+
T	T	+
T	F	-
T	T	+
F	F	-
F	F	-
F	F	-
T	T	-
T	F	-

a)Calculate the information gain when splitting on A and B. Which attribute would the ID3 algorithm choose?

# MACHINE LEARNING

Ans to Q1.

A	B	C	D	Class
T	T	T	T	T
T	T	T	F	F
T	T	F	T	F
T	T	F	F	T
T	F	T	T	F
T	F	T	F	T
T	F	F	T	T
T	F	F	F	F
F	T	T	T	F
F	T	T	F	T
F	T	F	T	T
F	T	F	F	F
F	F	T	T	T
F	F	T	F	F
F	F	F	T	F
F	F	F	F	T



## Information Gain- Calculation

Sol 2: The contingency tables after splitting on attributes  $A$  and  $B$  are:

	$A = T$	$A = F$		$B = T$	$B = F$
+	4	0	+	3	1
-	3	3	-	1	5

The overall entropy before splitting is:

$$E_{orig} = -0.4 \log 0.4 - 0.6 \log 0.6 = 0.9710$$

The information gain after splitting on A is:

$$E_{A=T} = -\frac{4}{7} \log \frac{4}{7} - \frac{3}{7} \log \frac{3}{7} = 0.9852$$

$$E_{A=F} = -\frac{3}{3} \log \frac{3}{3} - \frac{0}{3} \log \frac{0}{3} = 0$$

$$\Delta = E_{orig} - 7/10E_{A=T} - 3/10E_{A=F} = 0.2813$$

The information gain after splitting on B is:

$$E_{B=T} = -\frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4} = 0.8113$$

$$E_{B=F} = -\frac{1}{6} \log \frac{1}{6} - \frac{5}{6} \log \frac{5}{6} = 0.6500$$

$$\Delta = E_{orig} - 4/10E_{B=T} - 6/10E_{B=F} = 0.2565$$

Therefore, attribute  $A$  will be chosen to split the node.

## Practice Time

---

Create a decision tree for the following data using the ID3 algorithm.

Skills	Interactive	Knowledge	Job Offer
Poor	No	Average	No
Poor	No	Average	No
Good	No	Average	Yes
Good	Yes	Very Good	Yes
Good	Yes	Very Good	Yes
Poor	Yes	Average	No
Poor	No	Average	No
Poor	No	Very Good	Yes
Good	Yes	Very Good	Yes
Good	Yes	Average	Yes

## Solution

---

Entropy of Dataset (6 Yes, 4 No) -

$$\text{Entropy}(S) = -\frac{6}{10} \log_2 \frac{6}{10} - \frac{4}{10} \log_2 \frac{4}{10} = 0.9710$$

Entropy of Skills = Good (5 Yes, 0 No) -

$$\text{Entropy}(\text{Skills} = \text{Good}) = -\frac{5}{5} \log_2 \frac{5}{5} - \frac{0}{5} \log_2 \frac{0}{5} = 0$$

Entropy of Skills = Poor (1 Yes, 4 No) -

$$\text{Entropy}(\text{Skills} = \text{Poor}) = -\frac{1}{5} \log_2 \frac{1}{5} - \frac{4}{5} \log_2 \frac{4}{5} = 0.7219$$

Entropy of Skills -

$$\text{Entropy}(\text{Skills}) = \frac{5}{10} \times 0.7219 + \frac{5}{10} \times 0.0 = 0.3610$$

Info Gain of Skills -

$$\text{InfoGain}(\text{Skills}) = 0.9710 - 0.3610 = 0.6100$$

## Solution

---

Entropy of Interactive = Yes (4 Yes, 1 No) -

$$\text{Entropy}(\text{Interactive} = \text{Yes}) = -\frac{4}{5} \log_2 \frac{4}{5} - \frac{1}{5} \log_2 \frac{1}{5} = 0.7219$$

Entropy of Interactive = No (2 Yes, 3 No) -

$$\text{Entropy}(\text{Interactive} = \text{No}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.9710$$

Entropy of Interactive -

$$\text{Entropy}(\text{Interactive}) = \frac{5}{10} \times 0.7219 + \frac{5}{10} \times 0.9710 = 0.8465$$

Info Gain of Interactive -

$$\text{InfoGain}(\text{Interactive}) = 0.9710 - 0.8465 = 0.1245$$

## Solution

---

Entropy of Knowledge = Very Good (4 Yes, 0 No) -

$$\text{Entropy}(\text{Knowledge} = \text{Very Good}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$$

Entropy of Knowledge = Average (2 Yes, 4 No) -

$$\text{Entropy}(\text{Knowledge} = \text{Average}) = -\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} = 0.9183$$

Entropy of Knowledge -

$$\text{Entropy}(\text{Knowledge}) = \frac{4}{10} \times 0 + \frac{6}{10} \times 0.9182 = 0.5510$$

Info Gain of Knowledge -

$$\text{InfoGain}(\text{Knowledge}) = 0.9709 - 0.5510 = 0.4199$$

## Solution

---

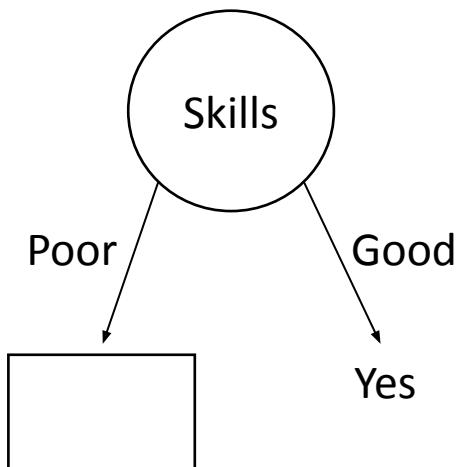
Comparing Info Gains -

$$\text{InfoGain}(\text{Skills}) = 0.6100$$

$$\text{InfoGain}(\text{Interactive}) = 0.1245$$

$$\text{InfoGain}(\text{Knowledge}) = 0.4199$$

Hence the first attribute chosen will be Skills -



Dataset with Skills = Poor -

Interactive	Knowledge	Job Offer
No	Average	No
No	Average	No
Yes	Average	No
No	Average	No
No	Very Good	Yes

## Solution

---

Entropy of Dataset (1 Yes, 4 No) -

$$\text{Entropy}(S) = -\frac{1}{5} \log_2 \frac{1}{5} - \frac{4}{5} \log_2 \frac{4}{5} = 0.7219$$

Entropy of Interactive = Yes (0 Yes, 1 No) -

$$\text{Entropy}(\text{Interactive} = \text{Yes}) = -\frac{0}{1} \log_2 \frac{0}{1} - \frac{1}{1} \log_2 \frac{1}{1} = 0$$

Entropy of Interactive = No (1 Yes, 3 No) -

$$\text{Entropy}(\text{Interactive} = \text{No}) = -\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{4}{4} = 0.8113$$

Entropy of Interactive -

$$\text{Entropy}(\text{Interactive}) = \frac{1}{5} \times 0 + \frac{4}{5} \times 0.8113 = 0.6490$$

Info Gain of Interactive -

$$\text{InfoGain}(\text{Interactive}) = 0.7219 - 0.6490 = 0.0729$$

## Solution

---

Entropy of Knowledge = Very Good (1 Yes, 0 No) -

$$\text{Entropy}(\text{Knowledge} = \text{Very Good}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

Entropy of Knowledge = Average (0 Yes, 4 No) -

$$\text{Entropy}(\text{Knowledge} = \text{Average}) = -\frac{0}{4} \log_2 \frac{0}{4} - \frac{4}{4} \log_2 \frac{4}{4} = 0$$

Entropy of Knowledge -

$$\text{Entropy}(\text{Knowledge}) = \frac{1}{5} \times 0 + \frac{4}{5} \times 0 = 0$$

Info Gain of Knowledge -

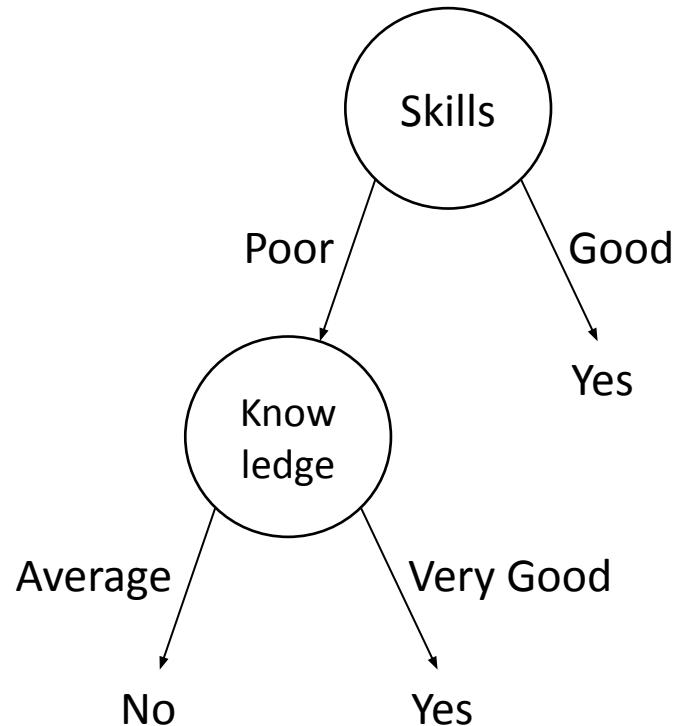
$$\text{InfoGain}(\text{Knowledge}) = 0.7219 - 0 = 0.7219$$

Comparing Info Gains -

$$\text{InfoGain(Interactive)} = 0.0729$$

$$\text{InfoGain(Knowledge)} = 0.7219$$

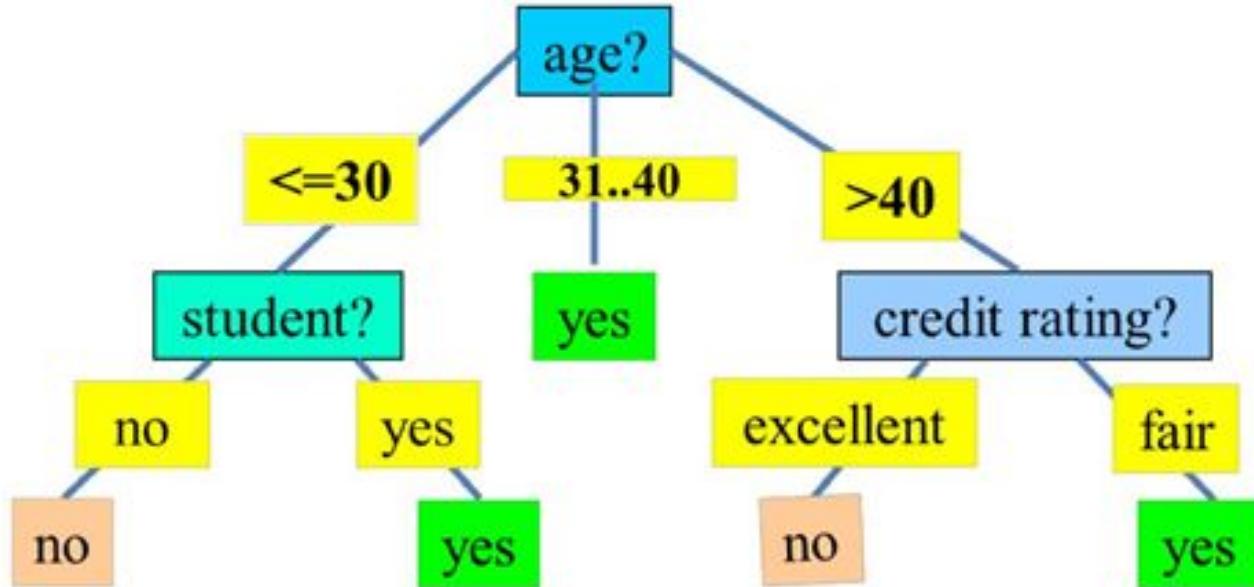
Hence the second attribute chosen will be Knowledge -



## Additional Question 1

---

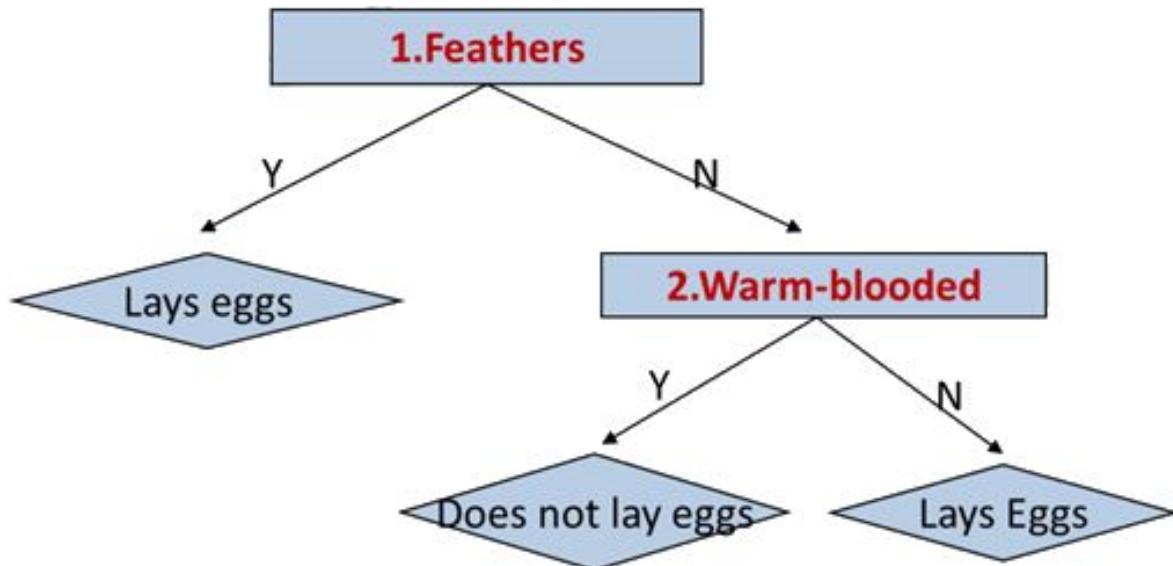
age	income	student	credit rating	buys computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no



## Additional Question 2

---

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



## Additional Question 3

Consider the following dataset and construct a DT using Info gain as heuristic.

Data

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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	80K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

## Additional Question 4

---

Name	Hair	Height	Weight	Lotion	Sunburned
Sarah	Blonde	Average	Light	No	Yes
Dana	Blonde	Tall	Average	Yes	No
Alex	Brown	Short	Average	Yes	No
Annie	Blonde	Short	Average	No	Yes
Emily	Red	Average	Heavy	No	Yes
Pete	Brown	Tall	Heavy	No	No
John	Brown	Average	Heavy	No	No
Katie	Blonde	Short	Light	Yes	No

## Limitations of ID3 Algorithm

---

### 1. Handling Numeric Data/ Inability to Handle Regression :

ID3 is designed primarily for categorical data for classification, meaning it doesn't handle continuous or numeric attributes very well. It requires discretization of numeric attributes, which can be problematic and can result in loss of information.

### 2. Biased Towards Attributes with Many Values or more categories :

ID3 tends to favor attributes with a large number of values or categories. This can lead to biased splits that may not necessarily be the most informative for classification.

### 3. Prone to Overfitting:

ID3 is prone to overfitting, especially when the decision tree becomes deep and captures noise in the training data. The algorithm doesn't incorporate strong pruning mechanisms to prevent overfitting.

## Limitations of ID3 Algorithm

---

### 4. No Handling of Missing Values:

ID3 does not handle missing attribute values well. It typically excludes instances with missing values, which can lead to biased or inaccurate results.

### 5. Unbalanced Class Distributions:

ID3 doesn't handle unbalanced class distributions effectively. It can lead to biased trees that reflect the majority class more strongly.

### 6. Greedy Approach:

ID3 uses a greedy approach to select attributes for splitting at each node. It doesn't consider future nodes and how earlier choices might affect later splits, which can result in suboptimal trees.

Due to these limitations, subsequent decision tree algorithms *like C4.5 (which overcomes many of ID3's limitations) and CART (Classification and Regression Trees) have been developed*. These algorithms address issues such as handling numeric attributes, handling missing values, handling unbalanced data, and incorporating pruning techniques to prevent overfitting. Despite its limitations, ID3 played a significant role in laying the foundation for the development of more sophisticated decision tree algorithms.



THANK YOU

---

Dr. Arti Arya  
[artarya@pes.edu](mailto:artarya@pes.edu)



**PES**  
UNIVERSITY

# MACHINE LEARNING

## Hypothesis Search and Inductive bias-ID3

---

**Dr. Arti Arya**

Department of Computer Science and Engineering

# MACHINE LEARNING

---

## Hypothesis Search and Inductive bias-ID3

**Dr. Arti Arya**

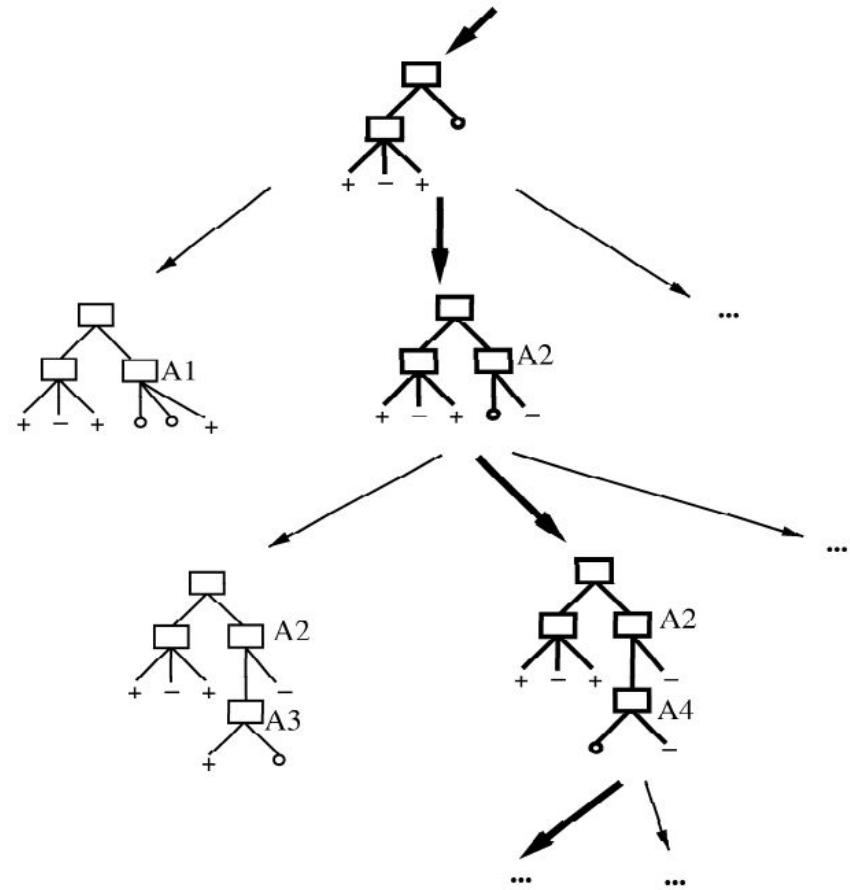
Professor, Department of Computer Science

## Acknowledgement

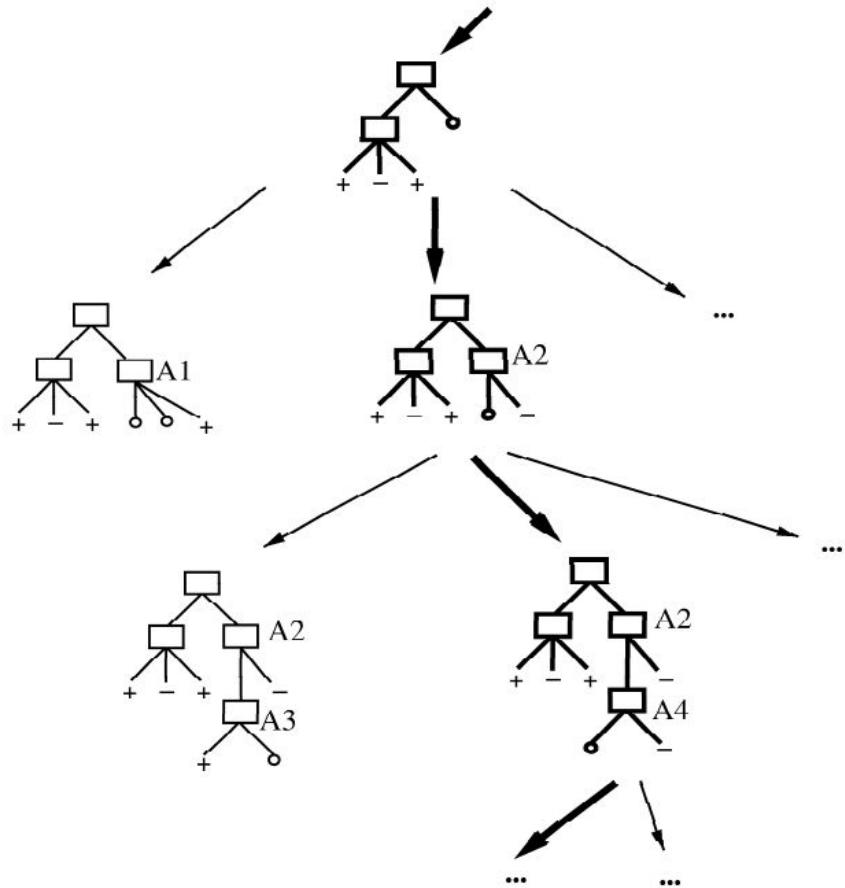
---

The slides are prepared from various from various Universities from abroad and India also. Also, some material is taken from reliable resources from internet throughout this course and sources are mentioned. The inputs to the slides are by Prof. K S Srinivas and Dr. Arti Arya.

- As with other inductive learning methods, *ID3 can be characterized as searching a hypotheses space for one that fits the training examples.*
- The hypothesis space (searched by ID3) *is the set of possible decision trees.*
- *ID3 performs a simple-to-complex, hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.*

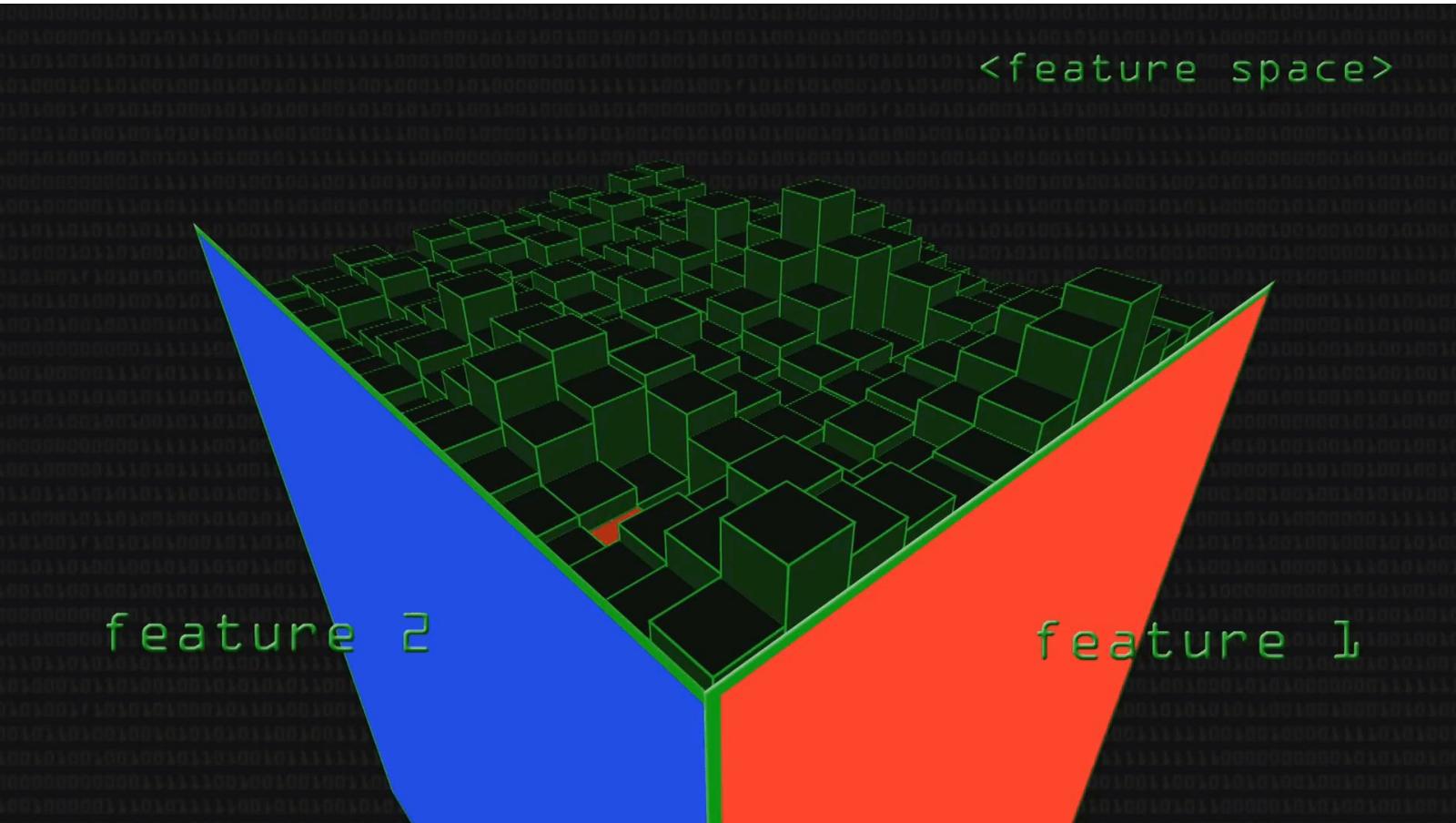


- ✓ Hill climbing is a **simple optimization algorithm** to find the **best possible** solution for a given problem.
- ✓ It belongs to the family of **local search algorithms** and is often used in optimization problems where the goal is to find the best solution from a set of possible solutions.
- ✓ In Hill Climbing, the algorithm starts *with an initial solution and then iteratively makes small changes* to it in order to improve the solution.
- ✓ These changes are based on **a heuristic function** that evaluates the quality of the solution.
- ✓ The algorithm *continues to make these small changes until it reaches a local maximum*, meaning that no further improvement can be made with the current set of moves.



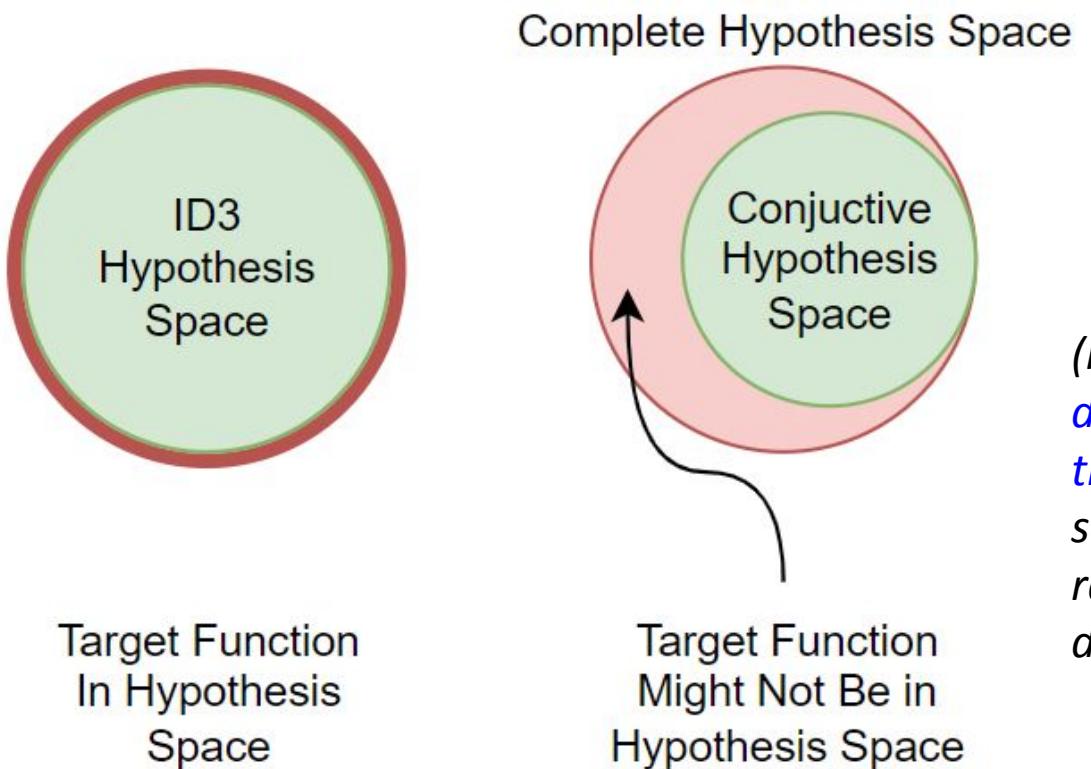
# MACHINE LEARNING

## Hill Climbing Algorithm



- The evaluation function that guides this hill-climbing search is the **information gain measure**.
- ID3's hypothesis space of all decision trees is a **complete space of finite discrete-valued functions**, relative to the available attribute.

- Because every finite discrete-valued function can be represented by some decision tree, *ID3 avoids one of the major risks of methods that search incomplete hypothesis spaces* (such as methods that consider only conjunctive hypotheses): that the hypothesis space might not contain the target function.



(means *ID3 systematically explores different attribute splits and partitions the data based on those splits*. By doing so, *ID3 ensures that it considers a broad range of possible hypotheses for dividing the data into classes*.)

- ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
- By determining only a single hypothesis, *ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses.*
- For example, it does not have the ability to determine how many alternative decision trees are consistent with the available training data, or to pose new instance queries that optimally resolve among these competing hypotheses

- ID3 in its pure form *performs no backtracking in its search*. Once it selects an attribute to test at a particular level in the tree, **it never backtracks to reconsider this choice**.
- ID3 uses training examples at each step in the search to make statistically based decisions regarding **how to refine its current hypothesis**.
- This contrasts with methods that make decisions incrementally, based on individual training examples (e.g., FIND-S or CANDIDATE-ELIMINATION).
- One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples. ID3 can be easily extended to handle noisy training data.

# MACHINE LEARNING

## INDUCTIVE BIAS

---

- *Given a collection of training examples, there are typically many decision trees consistent with these examples.*
- Describing the inductive bias of ID3 therefore consists of describing the basis by which *it chooses one of these consistent hypotheses over the others.*

- Which of these decision trees does ID3 choose?
- It chooses the first acceptable tree it encounters in its simple-to-complex, hill-climbing search through the space of possible trees.
- ID3 search strategy
  1. selects in favor of shorter trees over longer ones
  2. selects trees that place the attributes with highest information gain closest to the root.

**Approximate inductive bias of ID3:** Shorter trees are preferred over larger trees

**A closer approximation to the inductive bias of ID3:** Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

Is Inductive bias of ID3 favoring shorter trees a sound basis for generalization beyond training data?

**Read Chapter 3 of Tom Mitchell( Decision Tree Learning)**

- The inductive bias of ID3 favoring shorter trees is indeed a reasonable basis for generalization beyond the training data in some contexts, but it's important to understand both its benefits and limitations.
- Inductive bias refers to the set of assumptions or preferences that a learning algorithm uses to generalize from training data to unseen data.
- In the case of ID3's bias towards shorter trees, it stems from the Occam's razor principle, which suggests that simpler explanations (in this case, shorter trees) are generally preferred over more complex ones when both explain the data equally well.
- This bias helps to prevent overfitting, where a model becomes too tailored to the training data and fails to generalize to new data.

### Advantages of Favoring Shorter Trees:

**Simplicity and Interpretability:** Shorter decision trees are simpler and easier to understand. They tend to capture the most important and general patterns in the data without overemphasizing noise or outliers.

**Generalization:** By favoring shorter trees, ID3 is less likely to memorize the training data and more likely to identify higher-level patterns that generalize well to new, unseen data.

**Reduced Overfitting:** Shorter trees are less likely to overfit the noise in the training data, resulting in better performance on unseen data.

### Limitations of Favoring Shorter Trees:

**Underfitting:** While simpler trees can help prevent overfitting, they might also lead to underfitting, where the model is too simplistic to capture important relationships in the data. This is especially true when the underlying patterns are complex.

**Loss of Information:** Overly simplified trees might not capture nuances and interactions present in the data, potentially leading to decreased predictive accuracy.

**Biased against Complex Patterns:** The bias towards shorter trees might cause ID3 to miss some complex patterns that could be important for accurate generalization.

- So, the inductive bias of favoring shorter trees in ID3 is a sound approach for preventing overfitting and promoting generalization beyond the training data.
- However, it's not a universal solution, and the right balance between model complexity and generalization capability should be considered for each specific problem.

This is why more advanced decision tree algorithms like C4.5 and CART, which aim to strike a better balance between simplicity and complexity, have been developed to improve upon ID3's limitations.



THANK YOU

---

Dr. Arti Arya  
[artarya@pes.edu](mailto:artarya@pes.edu)



## MACHINE INTELLIGENCE

### Issues in Decision Tree Learning and solutions to it

---

**Course Instructor: Dr. Arti Arya**

Department of Computer Science and Engineering

# MACHINE LEARNING

---

## Issues in Decision Tree Learning and solutions to it

Dr. Arti arya

# MACHINE LEARNING

## Acknowledgement

---

The slides are prepared from various from various Universities from abroad and India also. Also, some material is taken from reliable resources from internet throughout this course and sources are mentioned. The inputs to the slides are by Prof. K S Srinivas and Dr. Arti Arya.

- The most important issues in decision tree comes at following places:
  - Over fitting of data
  - Determining how deeply to grow the decision tree
  - Handling continuous attributes
  - Choosing an appropriate attribute selection measure
  - Handling missing data
  - Handling attributes with differing costs, and improving computational efficiency.

The ID3 algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples

It can lead to difficulties when

- there is **noise in the data**, or
- when the **number of training examples** is **too small** to produce a representative sample of the true target function.

In either of these cases, this **simple algorithm** can produce trees that **overfits** the training examples.

We will say that a *hypothesis overfits the training examples if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances.*

Always classify the training dataset perfectly but doesn't work well on new data.

- **Definition:**

Given a hypothesis space  $H$ ,

a hypothesis  $h \in H$  is said to overfit the training data if  $\exists h' \in H$ .

(an alternative hypothesis), such that

$h$  has smaller error than  $h'$  over the training examples,

but  $h'$  has a smaller error than  $h$  over the entire distribution of instances.

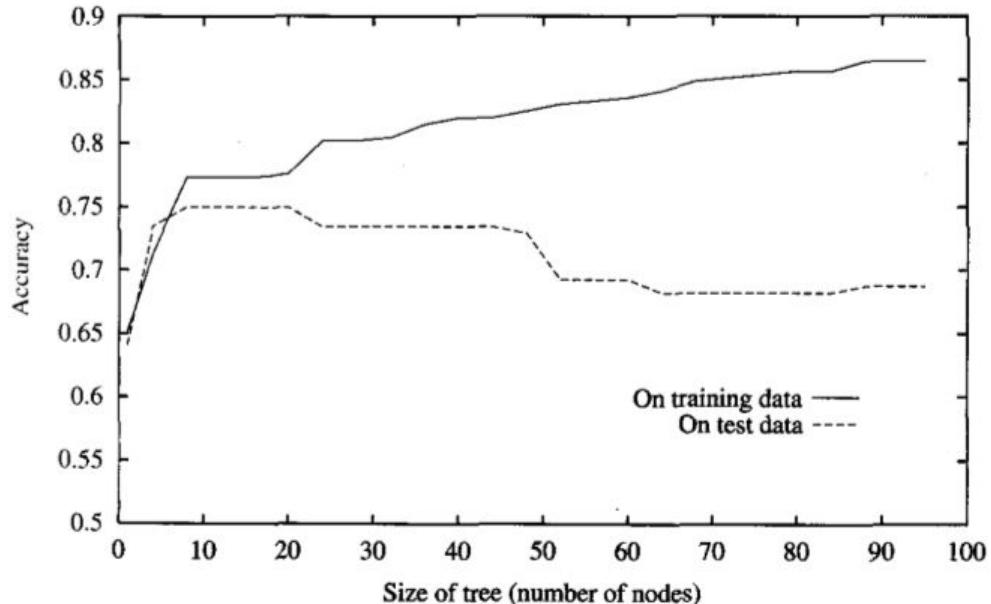
$$\text{error}(h) \underset{\text{on training examples}}{<} \text{error}(h')$$

$$\text{error}(h') \underset{\text{on training examples}}{<} \text{error}(h)$$

(over the entire dist of instances( that includes unseen records also))

## Overfitting of data

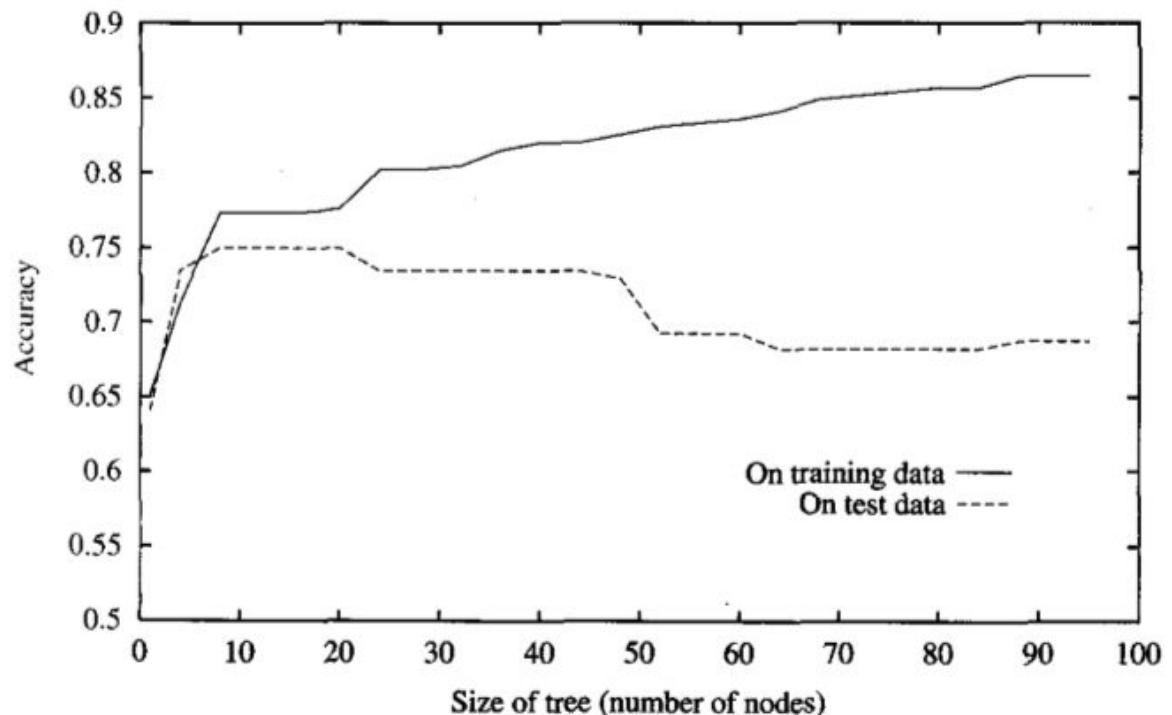
- The graph illustrates the **impact of overfitting** in a typical application of decision tree learning.
- In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes
- The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree.



- The accuracy of the model improves to some extent and then starts declining.
- We have to find the accurate number of nodes for which accuracy has improved and we can get the optimal DT.
- As we have discussed that **Inductive bias of DT** is to have short fat trees than long thin trees.

## Over fitting of data

- The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples (not included in the training set).
- Predictably, the accuracy of the tree over the training examples increases monotonically as the tree is grown.
- The accuracy measured over the independent test examples first increases, then decreases. As can be seen, once the tree size exceeds approximately 25 nodes.
- Further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.



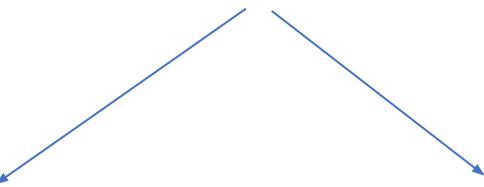
## Over fitting of data

- Generally, while using decision trees, there is a high chance of **overfitting the model**. Because it gets very complex with greater *depth* and greater *number of splits*.
- Decision trees are notoriously **famous for overfitting**. Pruning is a regularization method which penalizes the length of tree, i.e. increases the value of cost function.
- Pruning is of two types:
  - **Pre Pruning(Forward Pruning):** . .
  - **Post Pruning(Backward Pruning):**

## Avoiding Overfitting of data

Overfitting=  $f(\# \text{of nodes})$

- There are several approaches to avoid overfitting in decision tree learning. These can be grouped into two classes:



**Pre pruning:** Approaches that **stop growing the tree earlier**, before it reaches the point where it perfectly classifies the training data,

**Post pruning:** Approaches that **allow the tree to overfit the data**, and then post-prune the tree.

More useful is post pruning.

**Pre – Pruning** Halt the growth of the tree when the goodness of the split falls below a threshold ex. Should have a certain minimum information gain or setting up a depth before the algorithm begins

**Post Pruning** Fully grow the tree and then selective chop leaves and aggregate them into a parent with the predictor being the most common value

- Although the first of these approaches might seem more direct, the **second approach of post-pruning overfit trees has been found to be more successful in practice.**
- This is due to the difficulty in the **first approach of estimating precisely when to stop growing the tree.**
- **Decision Tree pruning** ensures trimming down a full tree to **reduce the complexity and variance of the model**. It makes the decision tree versatile enough to adapt any kind of new data fed to it, thereby fixing the problem of overfitting. It reduces the size of the decision tree which **might slightly increase the training error but drastically decrease the testing error.**

## Issues in Decision Tree Learning: Pre pruning(Corrective measures)

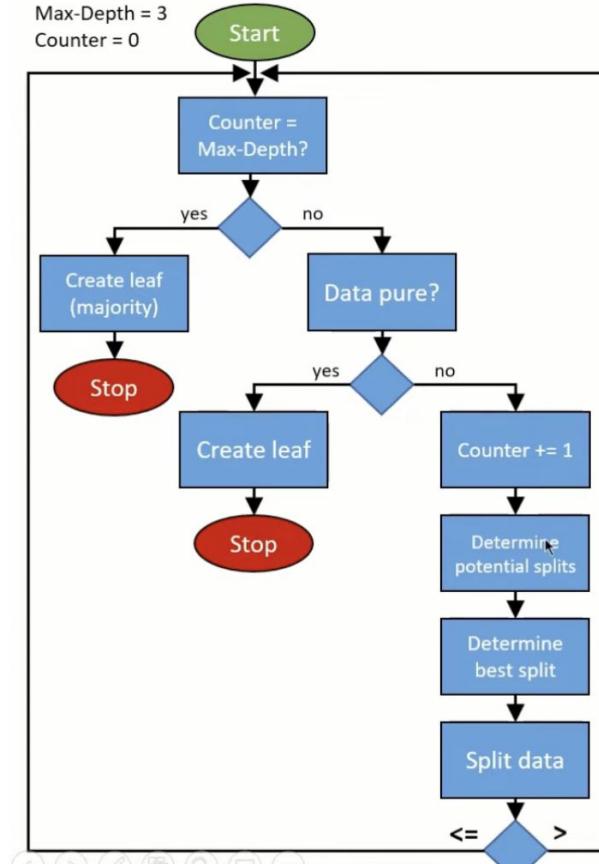
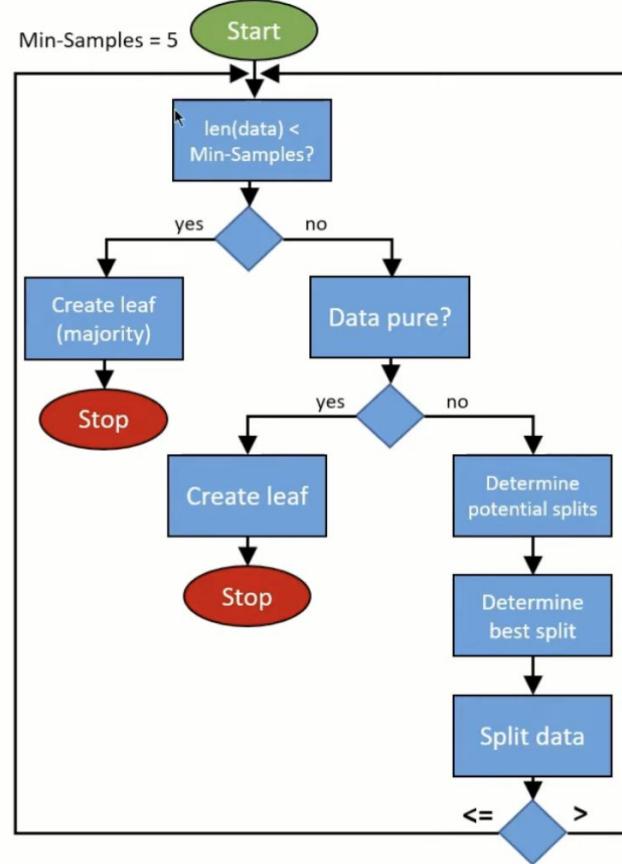
---

- For pre-pruning that we can limit the growth of trees by setting constraints.
- We can limit parameters like *max\_depth* , *min\_samples* etc.

As of now we can control these parameters:

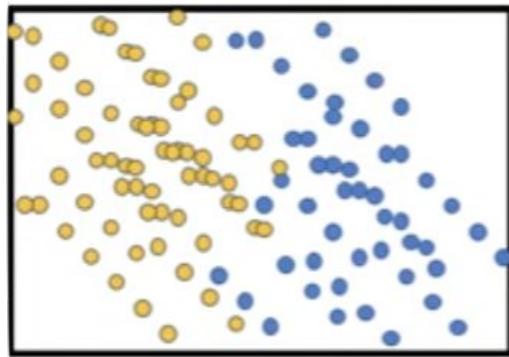
- *max\_depth*: maximum depth of decision tree
- *min\_sample\_split*: The minimum number of samples required to split an internal node
- *min\_samples\_leaf*: The minimum number of samples required to be at a leaf node.

## Issues in Decision Tree Learning: Pre pruning(Corrective measures)

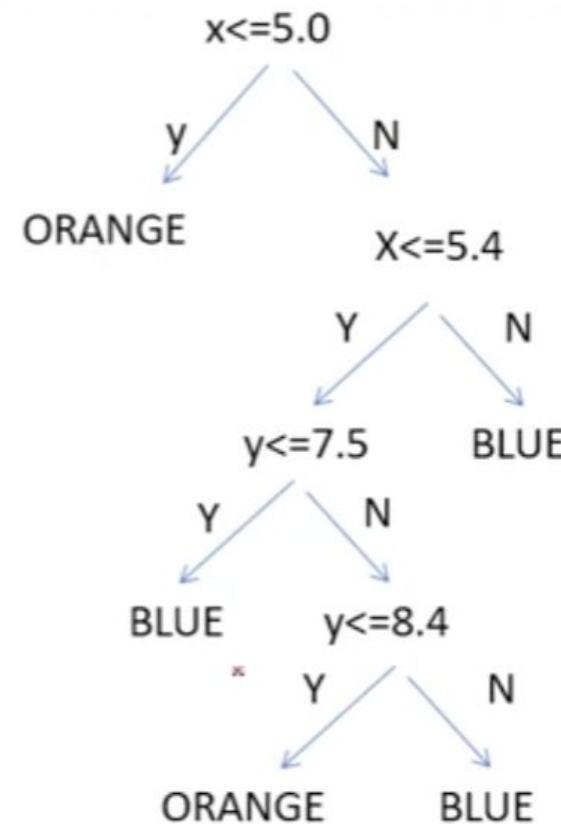


## Issues in Decision Tree Learning: Pre pruning(Corrective measures)

## Max depth pre pruning



Prune this part of the tree and assign the class label to the above node by majority voting.



- consider the following data set and the task is to build a decision tree to classify the object with attribute x and y into blue and orange
- suppose we get the following decision tree
- this has depth of 4 (root being at depth 0)
- we will set the depth as 3 for our pre pruning
- let us see how it works

## Avoiding Overfitting of data: Finding the right size of the tree

---

- Regardless of whether the correct tree size is found by **stopping early or by post-pruning**, a key question is what criterion is to be used to determine the correct final tree size.
- Approaches include:
  1. Use a **separate set of examples**, distinct from the training examples, to evaluate the **utility of post-pruning nodes** from the tree.
  2. Use all the available data for training, but **apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set**. For example, Quinlan (1986) used a chi-square test to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
  3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This approach is based on a heuristic called as **Minimum description Length principle**.
- **Finding the rt. depth of the tree** is thru **Info gain**(heuristic) that is often learnt thru NNs.
- Also **GAs** are applied for finding the right size of the tree.
- Occam's razor

## Issues in Decision Tree Learning: Post Pruning: Reduced error pruning

---

- Consider each of the decision nodes in the tree to be candidates for pruning.
- Pruning a decision node consists of
  - Removing the sub tree rooted at that node
  - Making it a leaf node
  - Finally, assigning it the most common classification of the training examples affiliated with that node.
- Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set

## Issues in Decision Tree Learning: Reduced error pruning

- Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set.
- Pruning of nodes continues until further pruning is harmful (i.e., decreases accuracy of the tree over the validation set).
- It is possible that *with first pruning the error rate decreases, so we'll go for next pruning, say error rate further reduces and continuing like this we can prune the tree till error rate increases at certain prune*. Then we'll include the pruned portion in the tree to have reduced error.
- So after having the fully grown tree, start pruning from the bottom and keep pruning till error decreases.

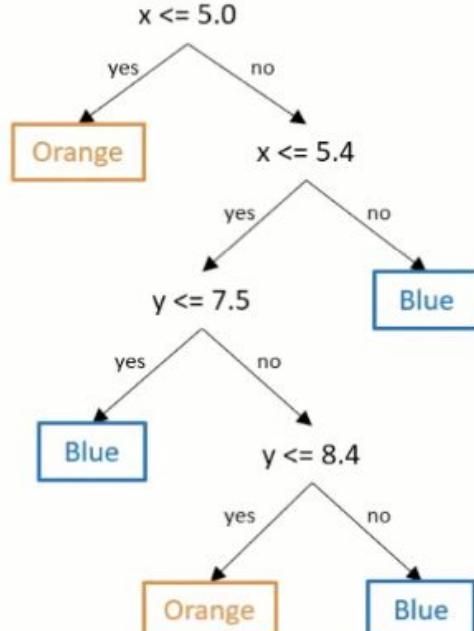
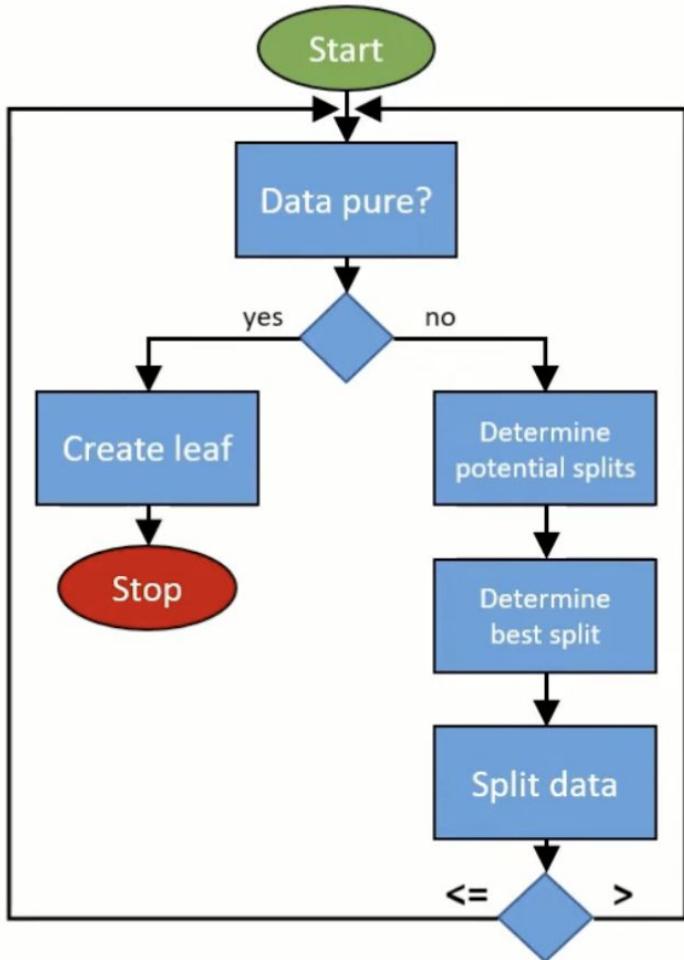
*Training set:* A set of examples used for learning, that is to fit the parameters of the classifier.

*Validation set:* A set of examples used to tune the parameters of a classifier, for example to choose the number of hidden units in a neural network.

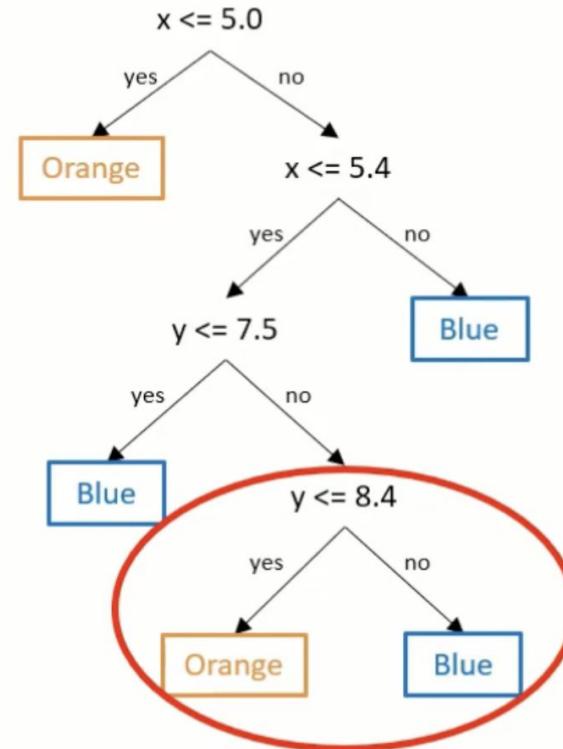
*Test set:* A set of examples used only to assess the performance of a fully-specified classifier.

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing over fitting to occur.
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
- Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

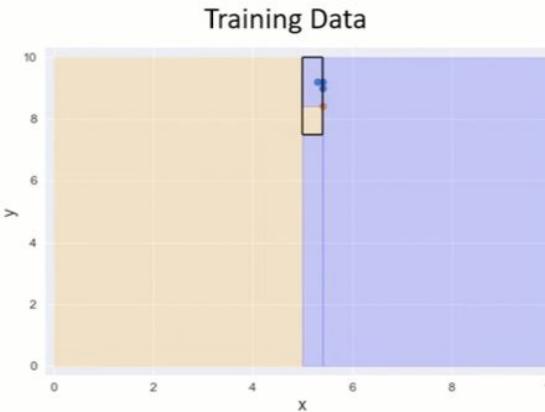
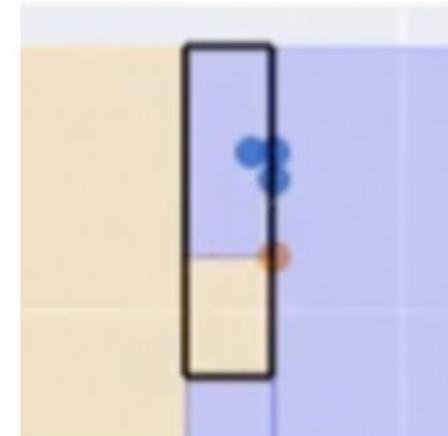
## Issues in Decision Tree Learning: Rule based Post pruning



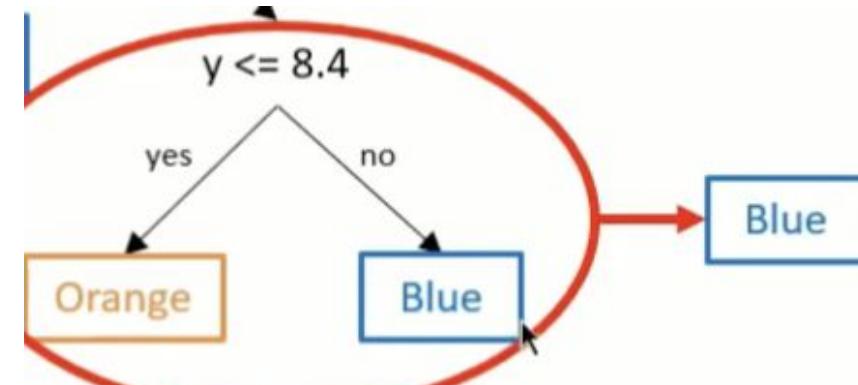
To post-prune, we first start at the deepest layer as given in the diagram below.



Based on the Training data,

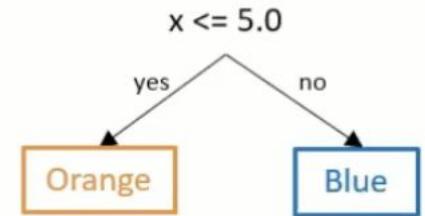
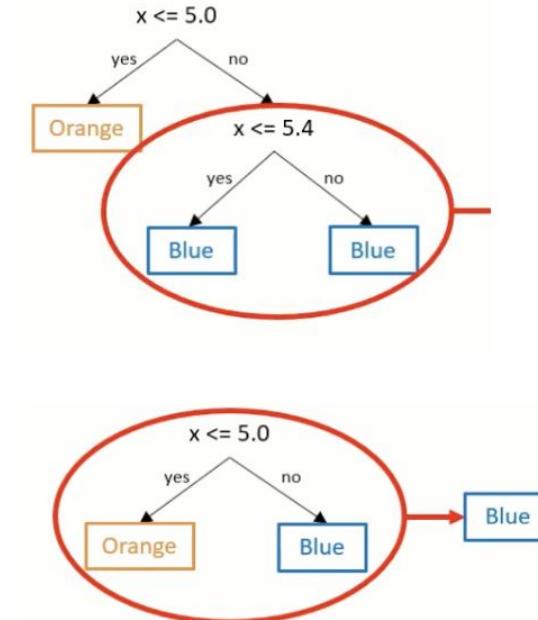
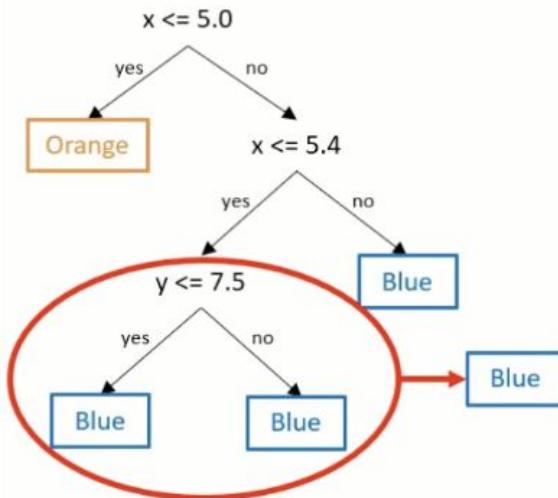
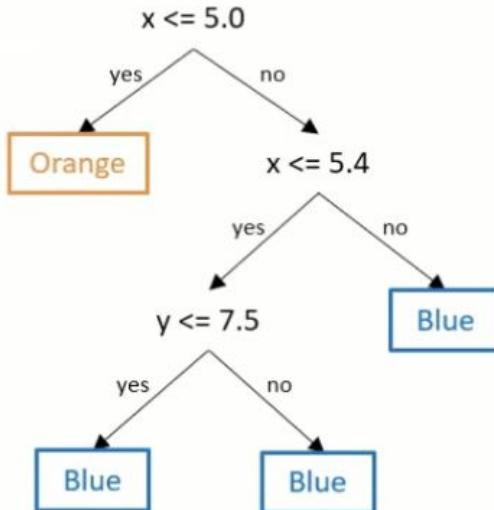


Given the conditions, we have 3 blue dots and one orange. Since **blue** has clear majority, we change the decision tree accordingly:



## Issues in Decision Tree Learning: Rule based Post pruning

Similarly, prune the rest of the tree:



- Our initial definition of ID3 is restricted to attributes that take on a **discrete set of values**.
- First, the target attribute whose value is predicted by the learned tree must be discrete valued.
- Second, the attributes tested in the decision nodes of the tree must also be discrete valued
- This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.
- This can be accomplished by dynamically defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals.
- In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new Boolean attribute  $A_c$ , that is true if  $A < c$  and false otherwise. The only question is how to select the best value for the threshold  $c$ .

## Dealing with continuous value( already done, once more)

- As an example, suppose we wish to include the continuous-valued attribute Temperature in describing the training example days in the learning task in the given table

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- What threshold-based Boolean attribute should be defined based on Temperature?
- Clearly, we would like to pick a threshold,  $c$ , that produces the greatest information gain.
- By sorting the examples according to the continuous attribute A, then identifying adjacent examples that differ in their target classification, we can generate a set of candidate thresholds midway between the corresponding values of A.
- It can be shown that the value of  $c$  that maximizes information gain must always lie at such a boundary
- These candidate thresholds can then be evaluated by computing the information gain associated with each.

## Dealing with continuous value: Another Example

TEMP	40	48	60	72	80	90
Play Sport	no	no	yes	yes	yes	no

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of Play Tennis changes:  $(48 + 60)/2$ , and  $(80 + 90)/2$ .
- The information gain can then be computed for each of the candidate attributes
- $\text{Temperature}_{>54}$  and  $\text{Temperature}_{>85}$ , and the best can be selected  $\text{Temperature}_{>54}$ .
- This dynamically created Boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.
- Further extension to this can be splitting data into multiple intervals rather than just two intervals based on a single threshold.

# Characteristics of Decision Tree Induction

- 1. DTI is a **nonparametric** approach for building class. model.
  - Doesn't require any prior assumption about the distribution of the data set
- 2. The algorithm presented so far uses a **top-down, recursive** partitioning strategy to induce a reasonable solution.
- 3. Tech. developed for construction of DT are **computationally inexpensive**, making it possible to construct models very fast even when datasets are very large.



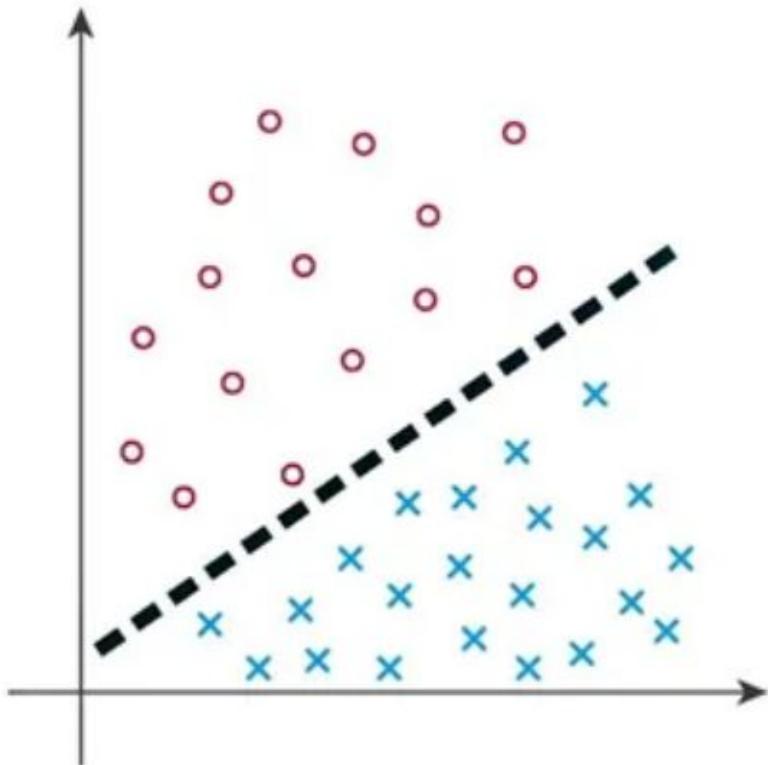
- 4. DT especially small in size are relatively easy to interpret.
- 5. DT are robust to the presence of noise.
- 6. The presence of redundant attr. does not adversely affect the accuracy of the decision tree.
  - An attr. is redundant if it is strongly correlated with another attr. in the data.
  - If data contains irrelevant attr. then feature selection tech. can help to improve the accuracy of the DT by eliminating such attr. during preprocessing.



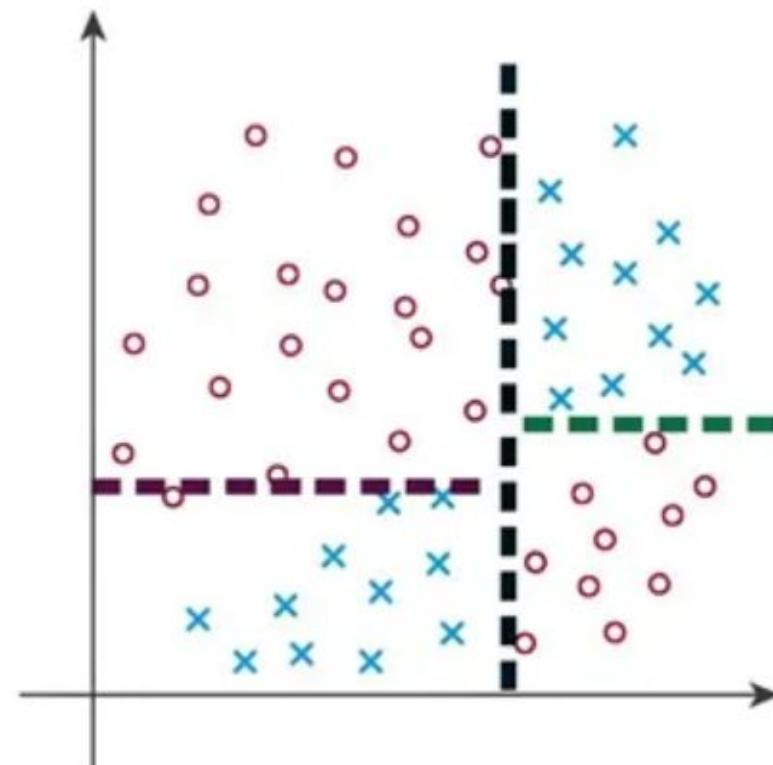
- **7. Data Fragmentation:** Number of instances gets smaller as you traverse down the tree
  - Number of instances at the leaf nodes could be too small to make any statistically significant decision.
  - One possible solution is to disallow further splitting when the no. of records fall below a certain threshold.



- 8. **Expressiveness** : Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree with  $2^d$  nodes, where d is the no. of Boolean attr.
    - (Q: Draw a DT for a parity function with 4 boolean attr. A, B, C, D)
  - Not expressive enough for modeling continuous variables
    - Particularly when test condition involves only a single attribute at-a-time

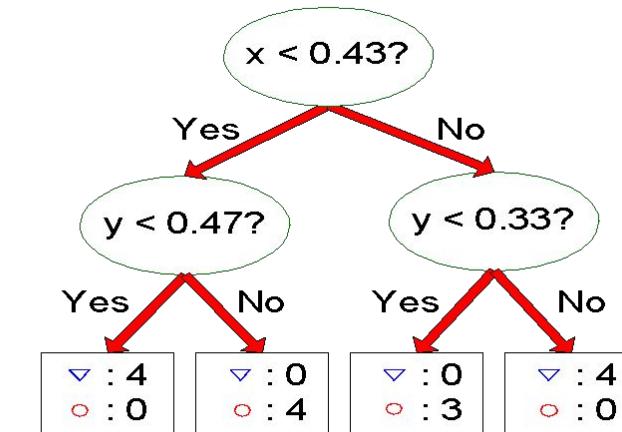
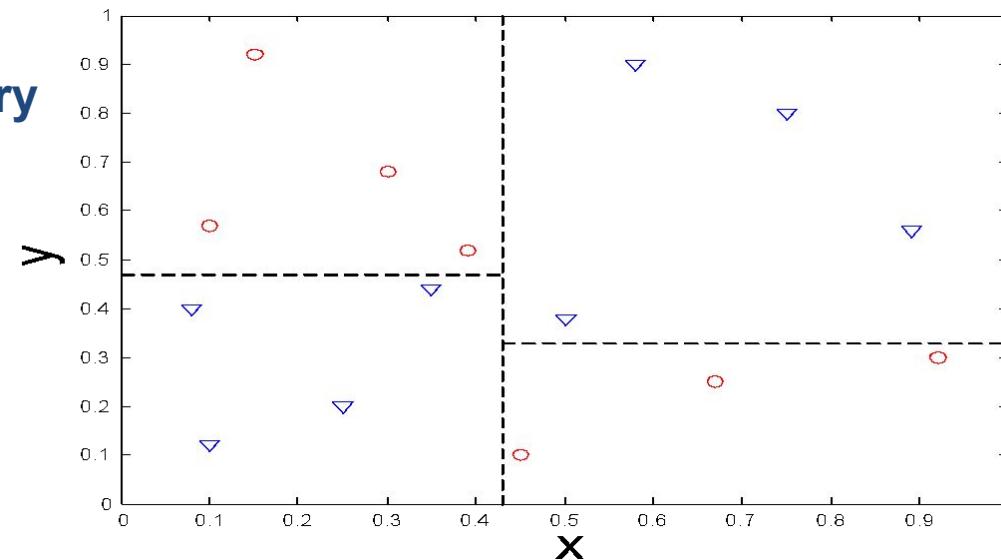


Linearly separable dataset



Linearly inseparable dataset

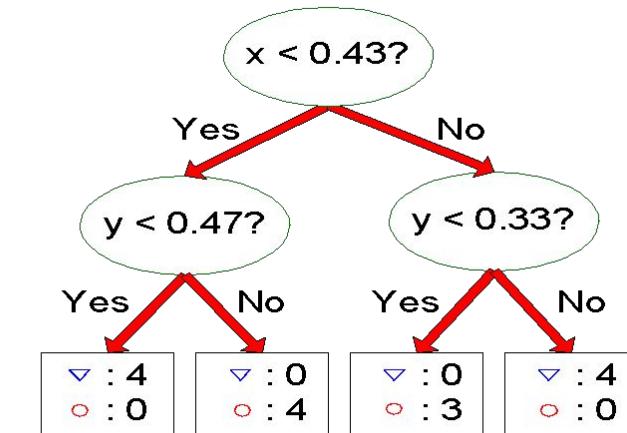
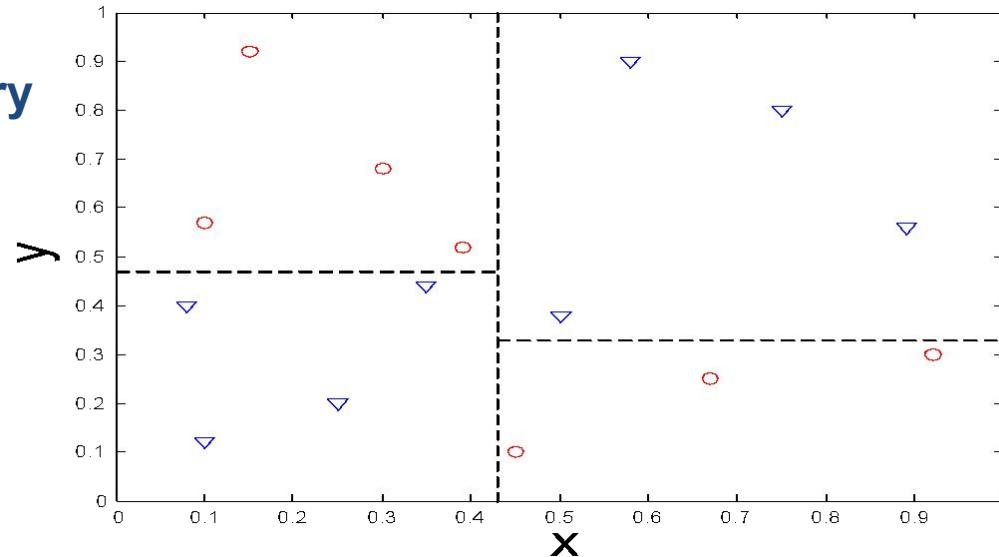
## 9. Decision Boundary



- The test cond. considered so far contains only a single attr. at a time.
- So tree constr. can be viewed as partitioning the attr. space into disjoint regions until each region contains records of the same class label.

Decision Trees divide the input space into axis-parallel rectangles and label each rectangle with one of the K classes

## 9. Decision Boundary

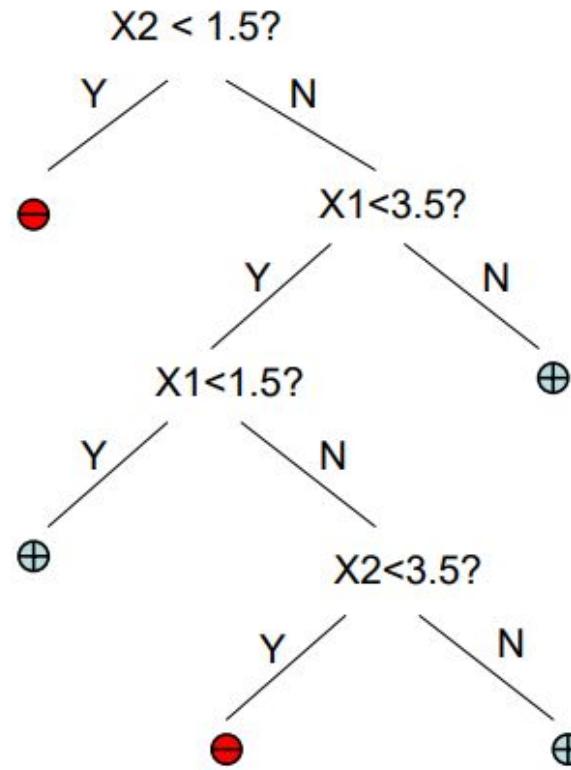
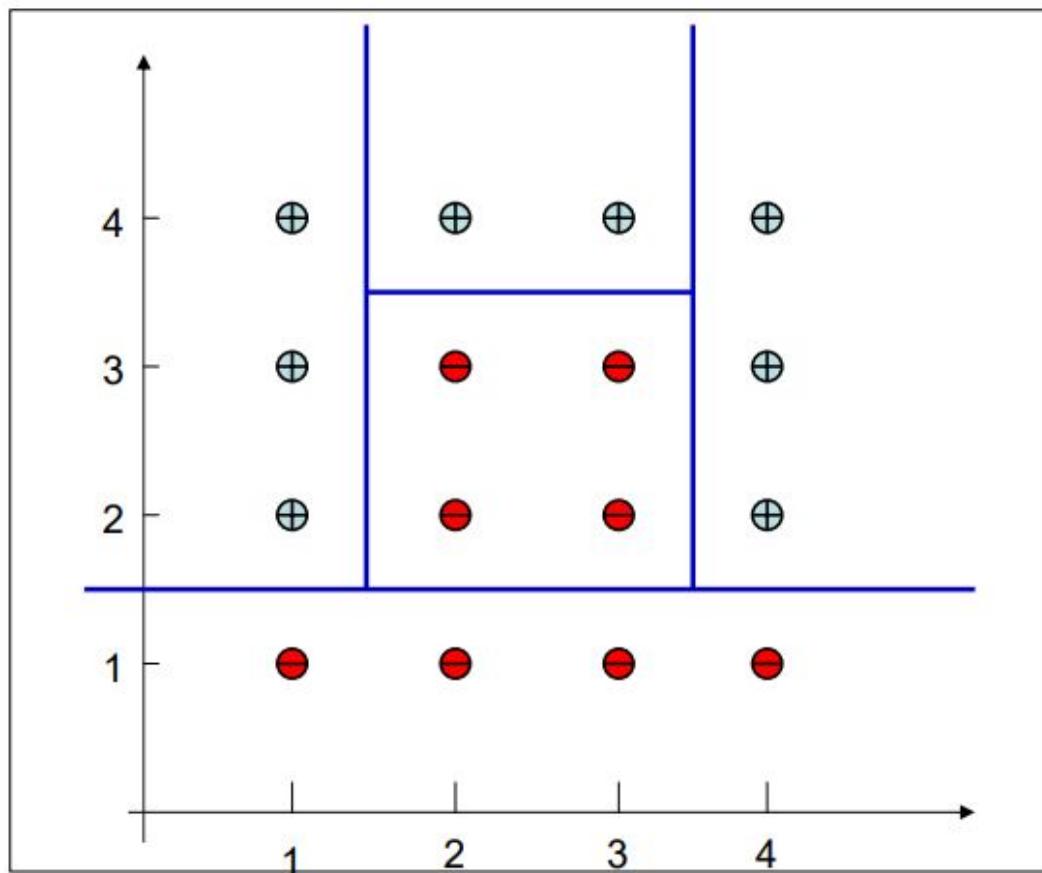


- The test cond. considered so far contains only a single attr. at a time.
- So tree constr. can be viewed as partitioning the attr. space into disjoint regions until each region contains records of the same class label.

In a **classification** problem with two classes, a **decision boundary** or **decision surface** is a **hypersurface** that partitions the underlying feature space into two sets, one for each class.

## 9. Decision Boundary

Decision Trees divide the input space into axis-parallel rectangles and label each rectangle with one of the K classes



## 9. Decision Boundary for categorical attributes

Weather	Temperature	Play Tennis
Sunny	Hot	No
Sunny	Mild	No
Overcast	Mild	Yes
Rainy	Mild	Yes
Rainy	Cold	Yes
Overcast	Hot	Yes
Sunny	Mild	Yes

- ❖ 1<sup>st</sup> decision boundary is based on Weather attribute: Sunny, Overcast and Rainy.
- ❖ Sunny Branch:
  - ❖ Decision Boundary: If **weather =sunny** then further sub divided on **temp**.
  - ❖ Decision Boundary (Sub-Decision): If "Temperature" is "Hot" or "Mild."

## Characteristics of DT: Decision Boundary for DT Classifier and DT Regressor

---

### Decision Tree Classification:

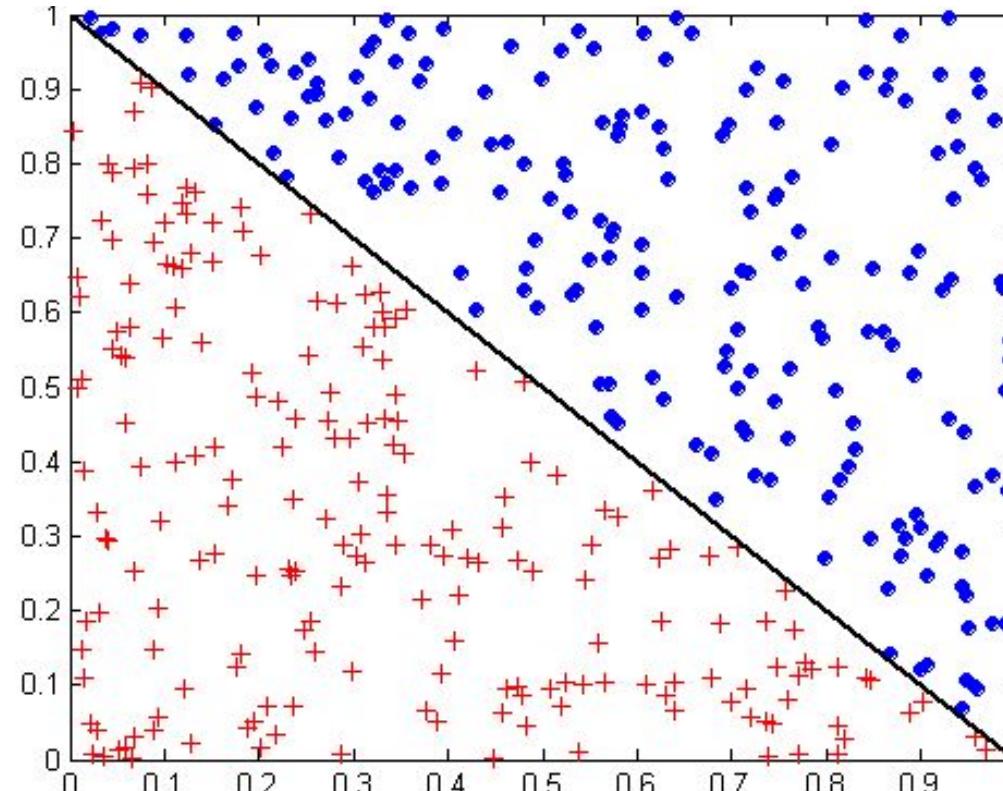
- In decision tree classification, the goal is to classify instances into different classes or categories.
- The decision boundaries are created by determining attribute thresholds that separate different classes.
- Each split node in the tree represents a decision boundary that separates instances of one class from instances of another class.
- Decision tree classifiers are often used for tasks like image classification, spam detection, and medical diagnosis.

Decision boundaries are generally more intuitive and easier to identify in decision tree classifiers because the splits directly correspond to separating different classes

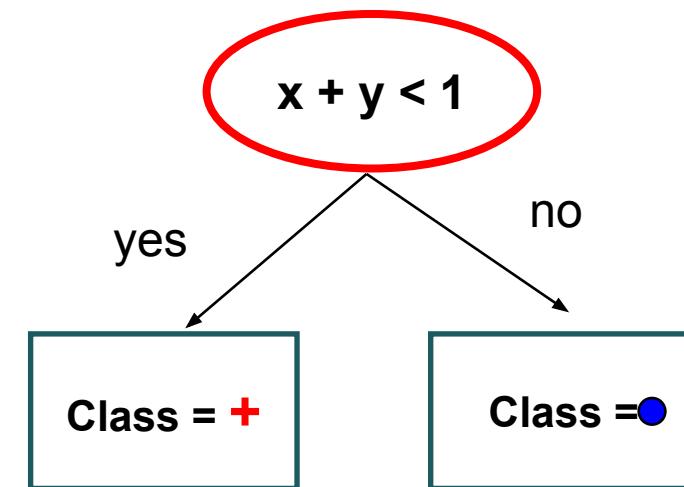
### Decision Tree Regression:

- In decision tree regression, the goal is to predict continuous numerical values instead of discrete classes.
- The decision boundaries in a regression tree are not as straightforward to visualize.
- Each split node in the tree still represents a decision boundary based on an attribute threshold, but the split points may not always correspond to clear class boundaries.
- Regression trees are commonly used for tasks like predicting house prices, stock prices, and temperature forecasts.

## 10. Oblique Decision Trees



The fig shown here can't be classified effectively by a DT that uses single attr. test cond. at a time.



- Test condition may involve multiple attributes in an oblique tree( Multivariate DT).
- More expressive representation
- Finding optimal test condition is computationally expensive

## 10. Oblique Decision Trees(Example)

Consider a dataset with two features ( $x_1$ ) and ( $x_2$ ) and a binary target variable ( $y$ ).

$x_1$	$x_2$	$y$
1	2	0
2	3	0
3	3	1
4	5	1

**Group1:**  $x_1 + 2x_2 \leq 7$

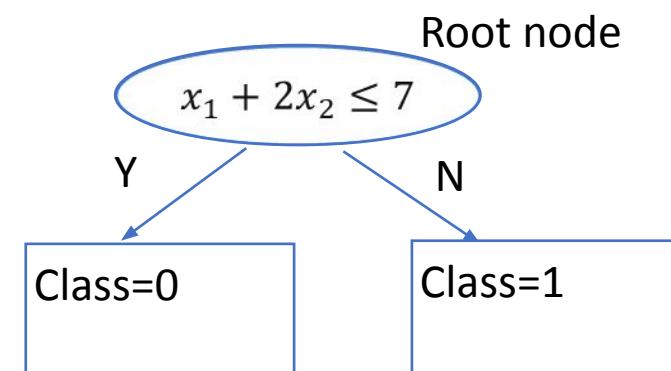
The data points belonging to this group are (1,2)  
Majority class:0

**Group 2:**  $x_1 + 2x_2 > 7$

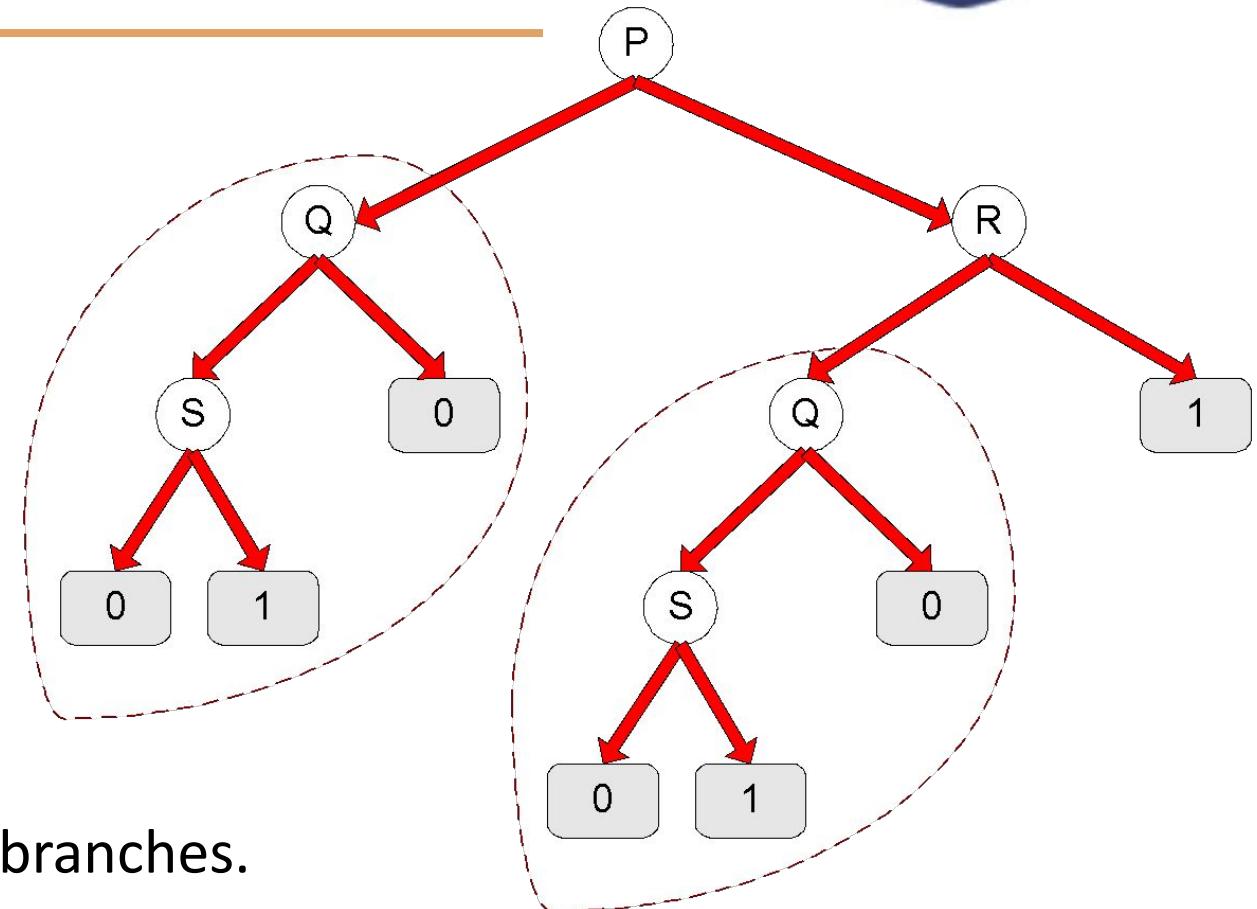
The data points belonging to this group are (2,3), (3,3) and (4,5)  
Majority class: 1

An oblique decision tree splits the data using a hyperplane.  
For simplicity, let's assume the hyperplane equation is  $ax_1 + bx_2 \leq c$ .

We need to find the coefficients a, b, and c that best separate the classes. For this example, let's assume we find the best hyperplane to be  $x_1 + 2x_2 \leq 7$ .  
So, this hyperplane splits the data in two groups:



## 11. Tree Replication



- Same subtree appears in multiple branches.
- DT becomes difficult to interpret.
- Such situation arises when construction of DT depends on a single attr. test cond. at each internal node.



- 12. Studies have shown that **choice of impurity measures** do not impact the performance of the DT induction algorithms.



THANK YOU

---

Arti Arya  
[artaryya@pes.edu](mailto:artaryya@pes.edu)



# MACHINE LEARNING (UE22CS352A)



## Linear Regression

---

**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

## What is Regression in Machine Learning?

---

### Regression:

- Regression is a method/ statistical measure for understanding the relationship between independent variables(X) or features and a dependent variable(Y) or outcome. Outcomes can then be predicted once the relationship between independent and dependent variables has been estimated.
- Forecast value of a dependent variable (Y) from the value of independent variables ( $X_1, X_2, \dots$  ).
- Analyse the specific relationships between two or more variables.
- This is done to gain information about the one through knowing values of the others.

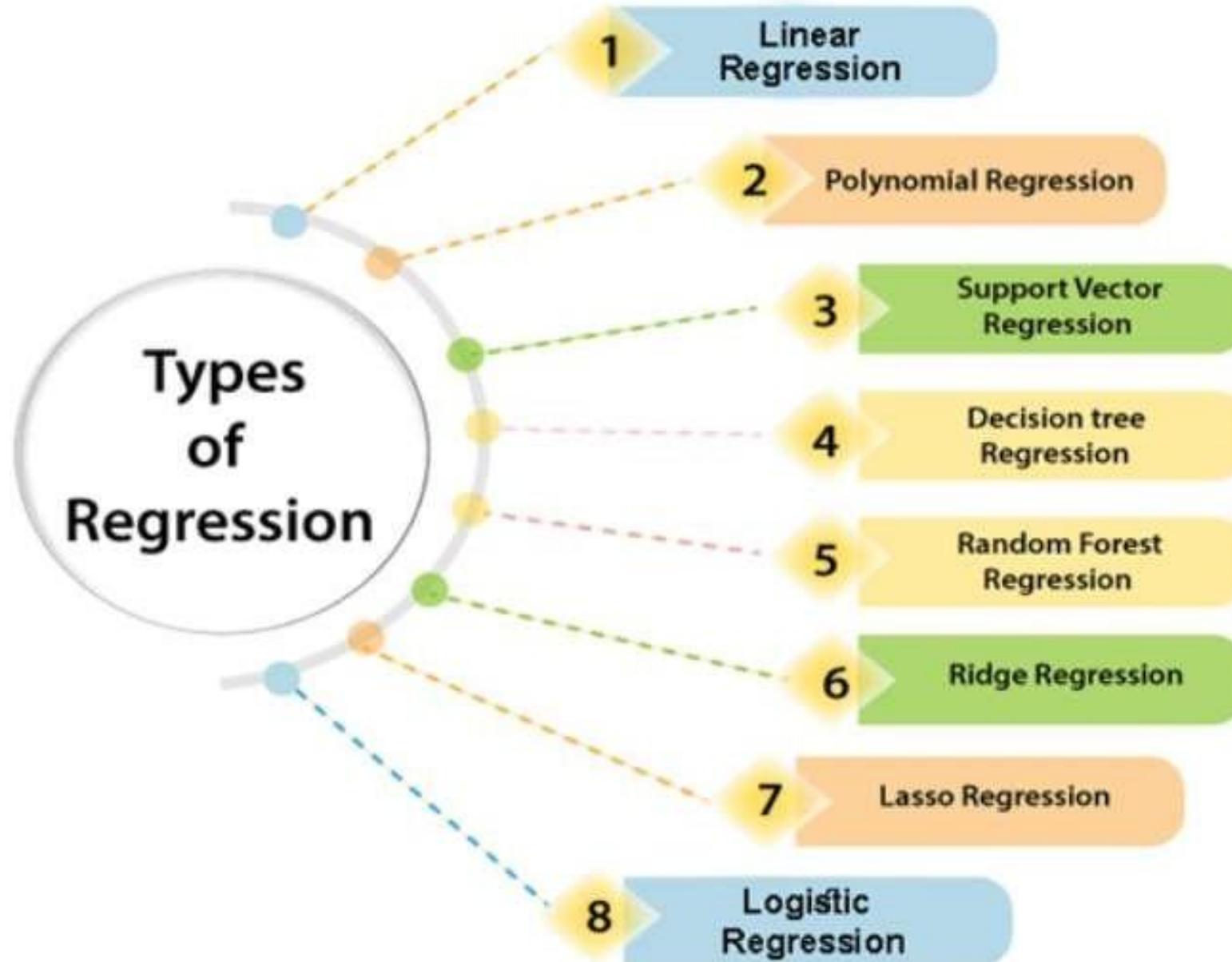
## Examples of Regression

---

**Some examples of regression can be as:**

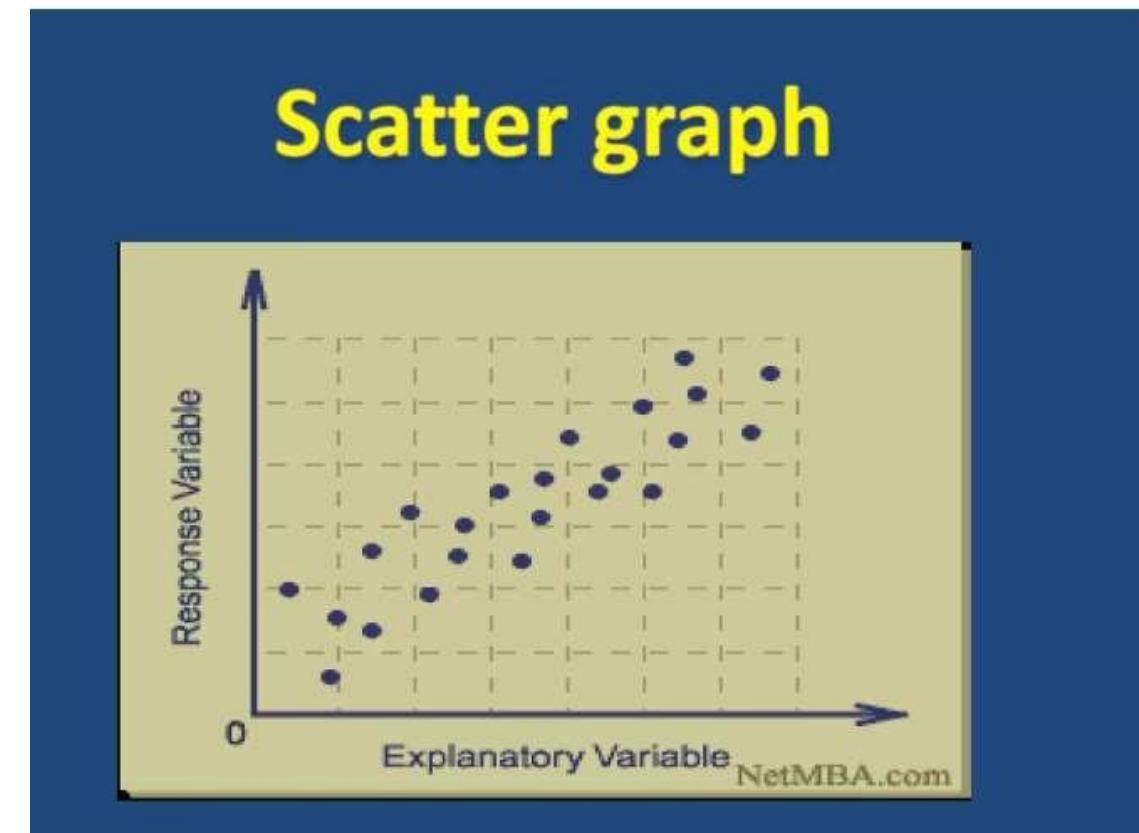
- Prediction of rain using temperature and other factors
- Prediction of road accidents due to rash driving.
- Forecasting continuous outcomes like house prices, stock process or sales.
- Predicting the success of future retail sales or marketing campaigns to ensure resources are used effectively.
- Analyzing datasets to establish the relationships between variables and output.
- Creating time series visualizations

## Types of Regression



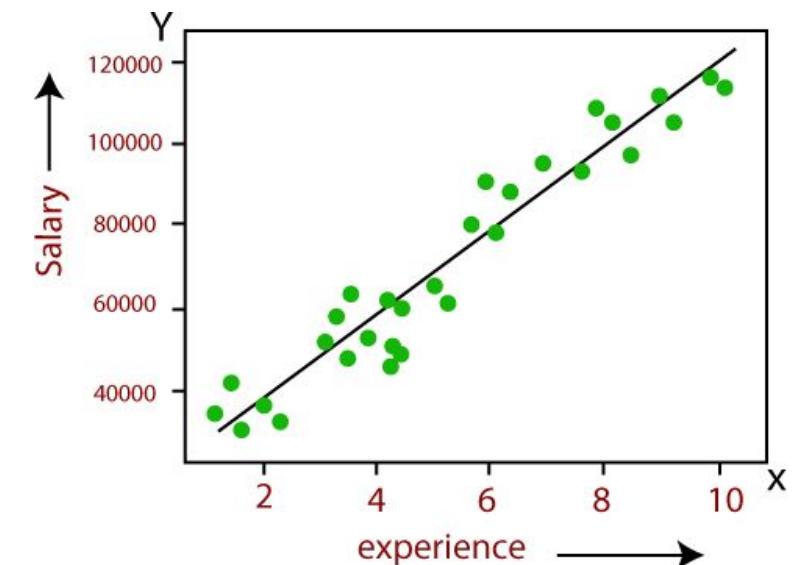
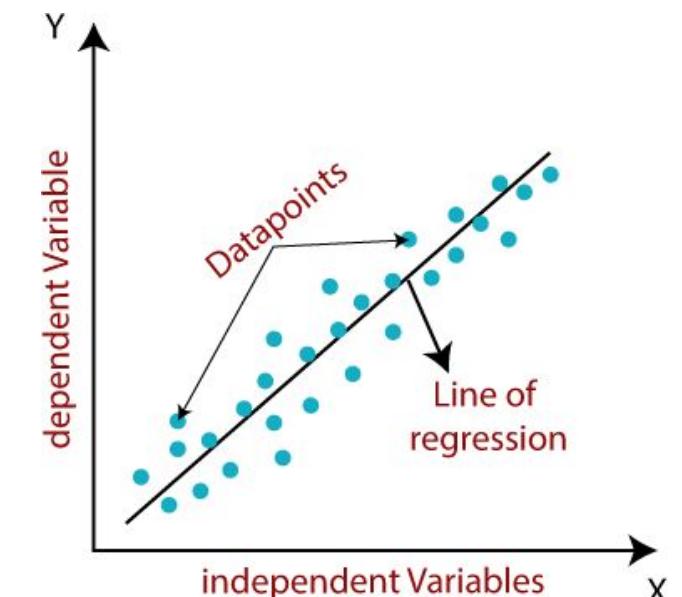
### Choosing the type of regression using scatter plot:

- A scatter plot is a useful visualization tool that can help you decide which **type of regression to use** when analyzing a relationship between two variables.
- Scatter plots show individual data points as dots on a two-dimensional plane, with one variable on the x-axis and the other variable on the y-axis.
- The pattern of the data points in a scatter plot can provide insights into the nature of the relationship between the variables, and this can guide you in choosing the appropriate type of regression analysis.



### Linear Regression:

- Linear regression is a statistical regression method which is used for **predictive analysis**, that shows a **linear relationship between a dependent (y) and one or more independent (x) variables**.
- It is one of the very simple and easy algorithms.
- It is used when you have continuous numerical data.
- Linear Regression is best used when the relationship between the variables can be approximated by a straight line.



### Types of Linear Regression:

#### Simple Linear Regression:

If there is **only one input variable** ( $x$ ), then such linear regression is called **simple linear regression**.

#### Multiple Linear Regression:

If there is **more than one input variable**, then such linear regression is called **multiple linear regression**.

**NOTE: In this course, we will be focusing on Simple Linear Regression only.**

### Simple Linear Regression:

Simple Linear Regression is a type of Regression algorithms that models the relationship between two continuous variables: **a dependent variable and a single independent variable.**

The equation of a simple linear regression model is represented as:

$$y = mx + b$$

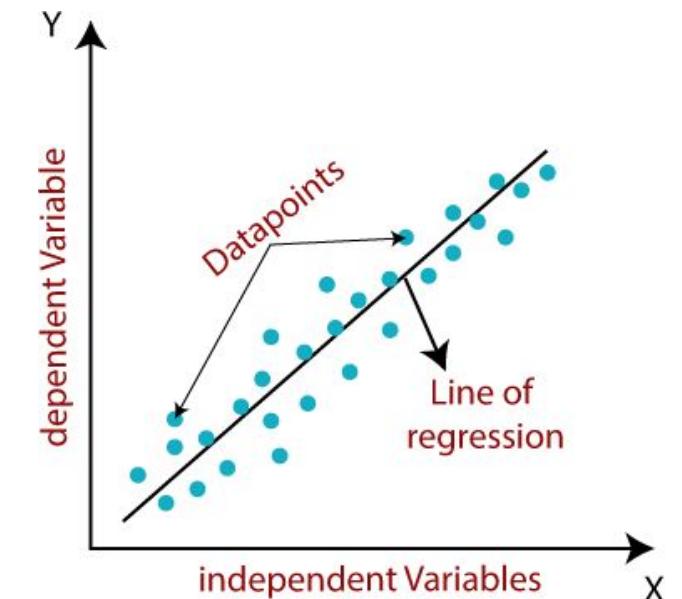
where:

y is the dependent variable.

x is the independent variable.

m is the slope of the line (coefficient), representing how much y changes for a unit change in x.

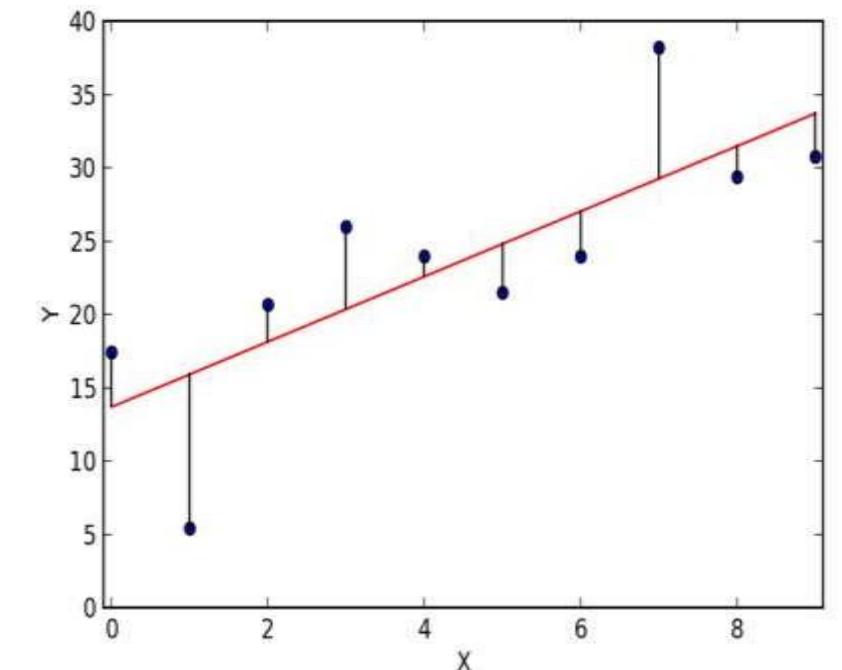
b is the y-intercept, which is the value of y when x is 0.



## Best Fitted Line

### Best Fitted Line:

- Regression shows a **line or curve** that passes through all the datapoints on target-predictor graph.
- A **Line of Best Fit/Best Fitted Line** is a **straight line** that **minimizes the vertical distance** between the datapoints and the regression line.
- The distance between datapoints and line tells whether a model has captured a **strong relationship** or not.
- The **goal is to find the best fitted line** which requires values for  **$m$ (slope) and  $b$ (y-intercept)** (called as parameters) for the given dataset. We will be using **Gradient Descent(GD) Algorithm** to find these parameters.



## Terminologies Used

Error = Actual Value – Predicted Value

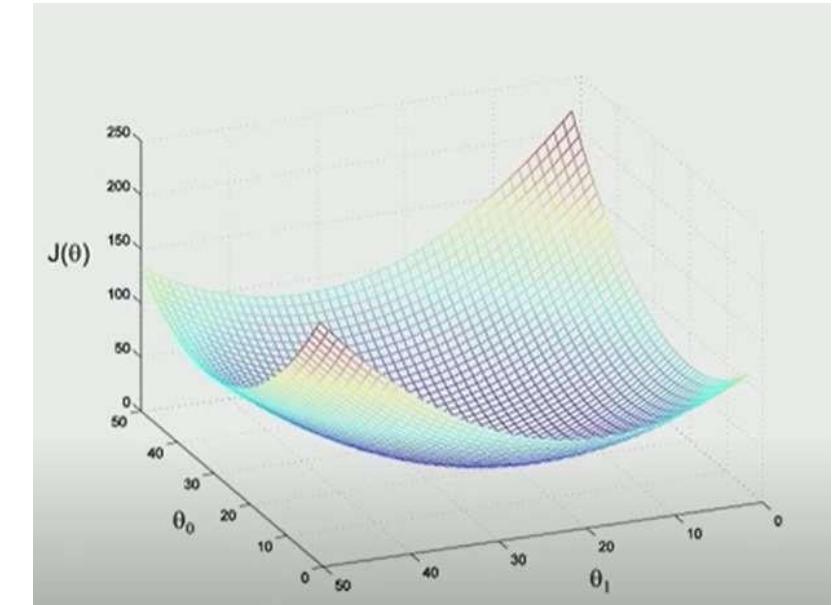
$$= y_i - (mx_i + b)$$

Squared Error =  $(y_i - (mx_i + b))^2$

Sum of Squared Error(SSE) =

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

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



Note: Since MSE is used as the **cost/loss function(J)** in linear regression and J is the sum of the squared terms, it turns out to be a **quadratic function** and the visualization of the same(in 3D) is shown in Figure.

Note: In the figure, theta 0 and theta 1 are corresponding to y-intercept(i.e. b) and slope(i.e. m) respectively in linear regression model.

## Terminologies Used

Error = Actual Value – Predicted Value

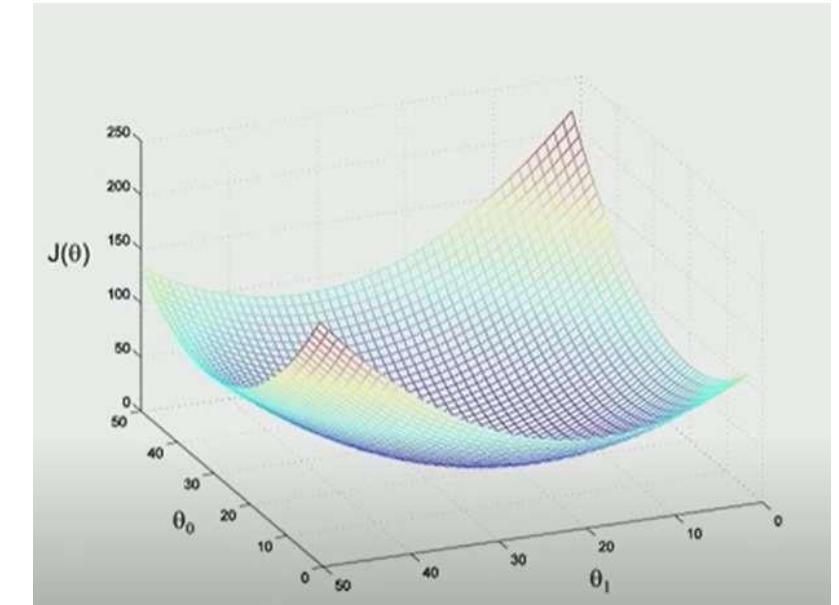
$$= y_i - (mx_i + b)$$

Squared Error =  $(y_i - (mx_i + b))^2$

Sum of Squared Error(SSE) =

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

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



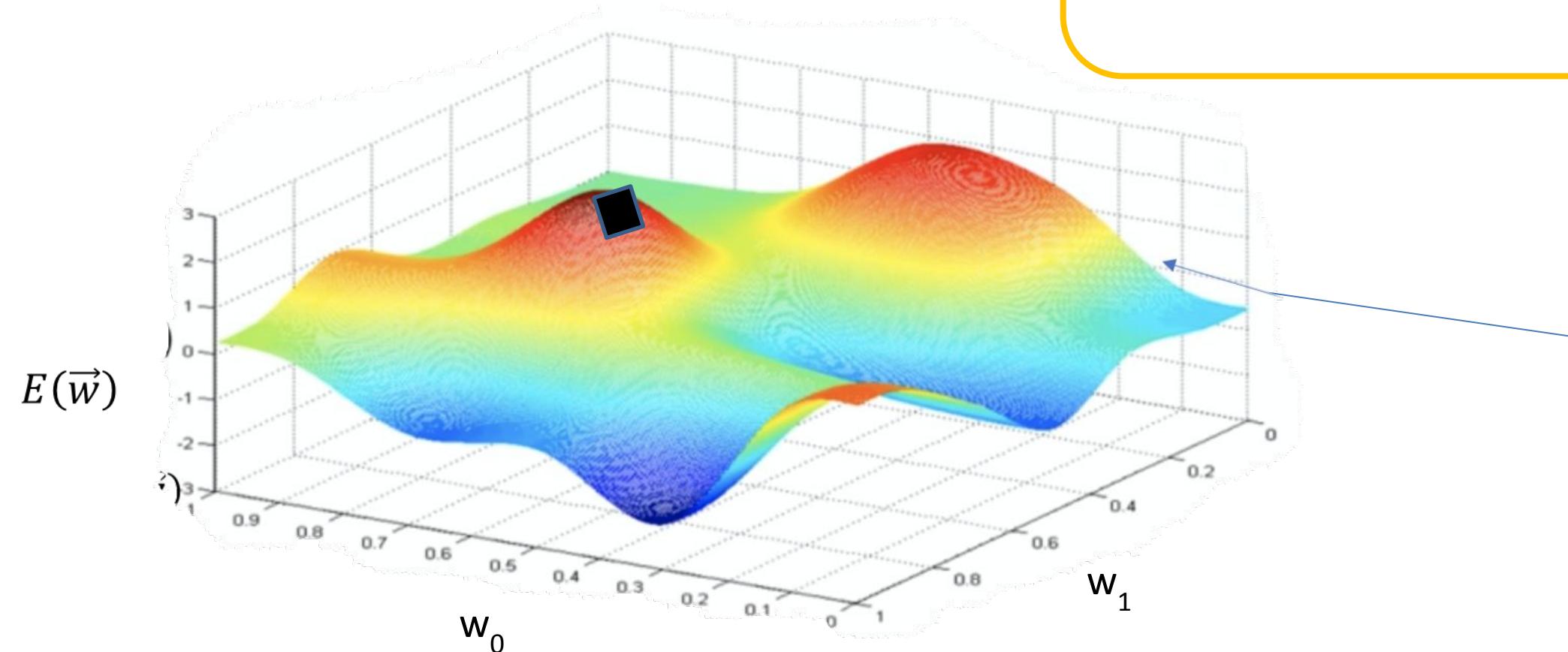
Note: Since MSE is used as the **cost/loss function(J)** in linear regression and J is the sum of the squared terms, it turns out to be a **quadratic function** and the visualization of the same(in 3D) is shown in Figure.

Note: In the figure, theta 0 and theta 1 are corresponding to y-intercept(i.e. b) and slope(i.e. m) respectively in linear regression model.

## Gradient Descent

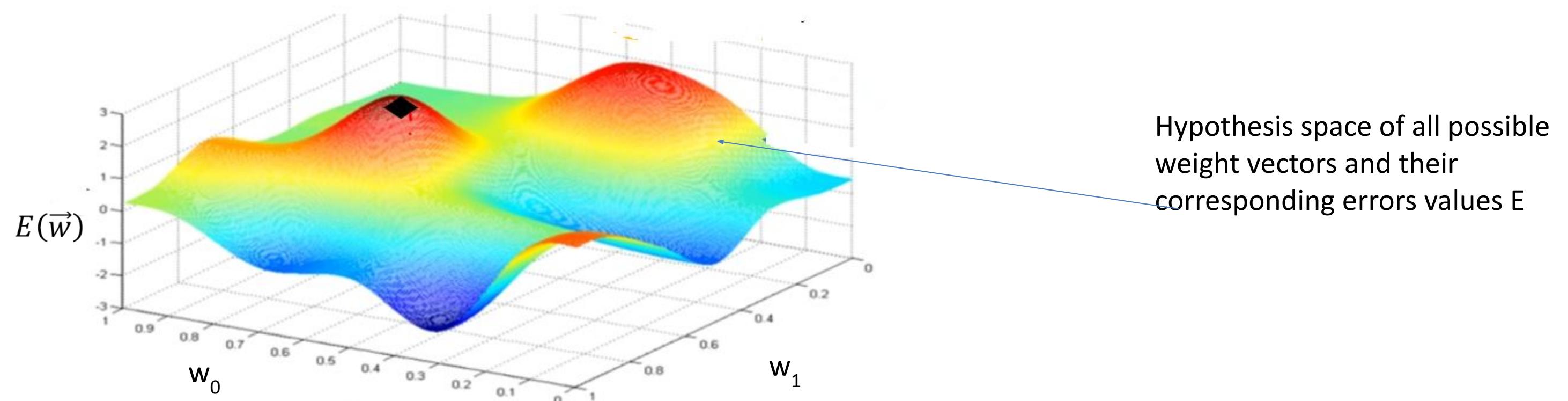
Vertical axis represent error relative to some fixed set of training examples

- let say we are trying to minimize this function
- we first start some random  $E(\mathbf{w})$  or  $J(\theta_0, \theta_1)$

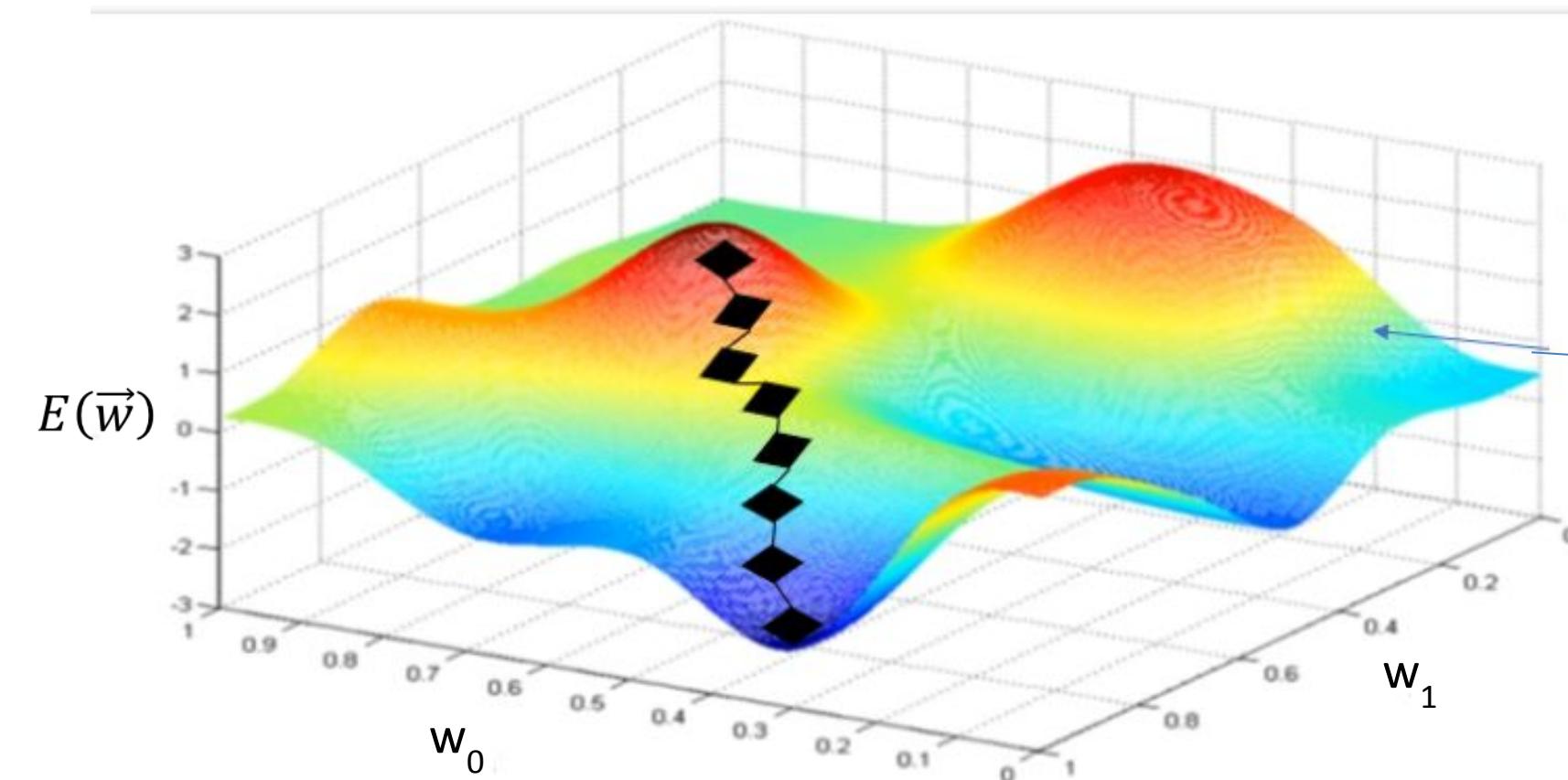


Hypothesis space of all possible weight vectors and their corresponding errors values  $E$

- Imagine the surface as some trekking area and you are standing at that point
- In gradient descent what we do is we take a small step in some direction and go down as quickly as possible
- the question would be what direction to take the step

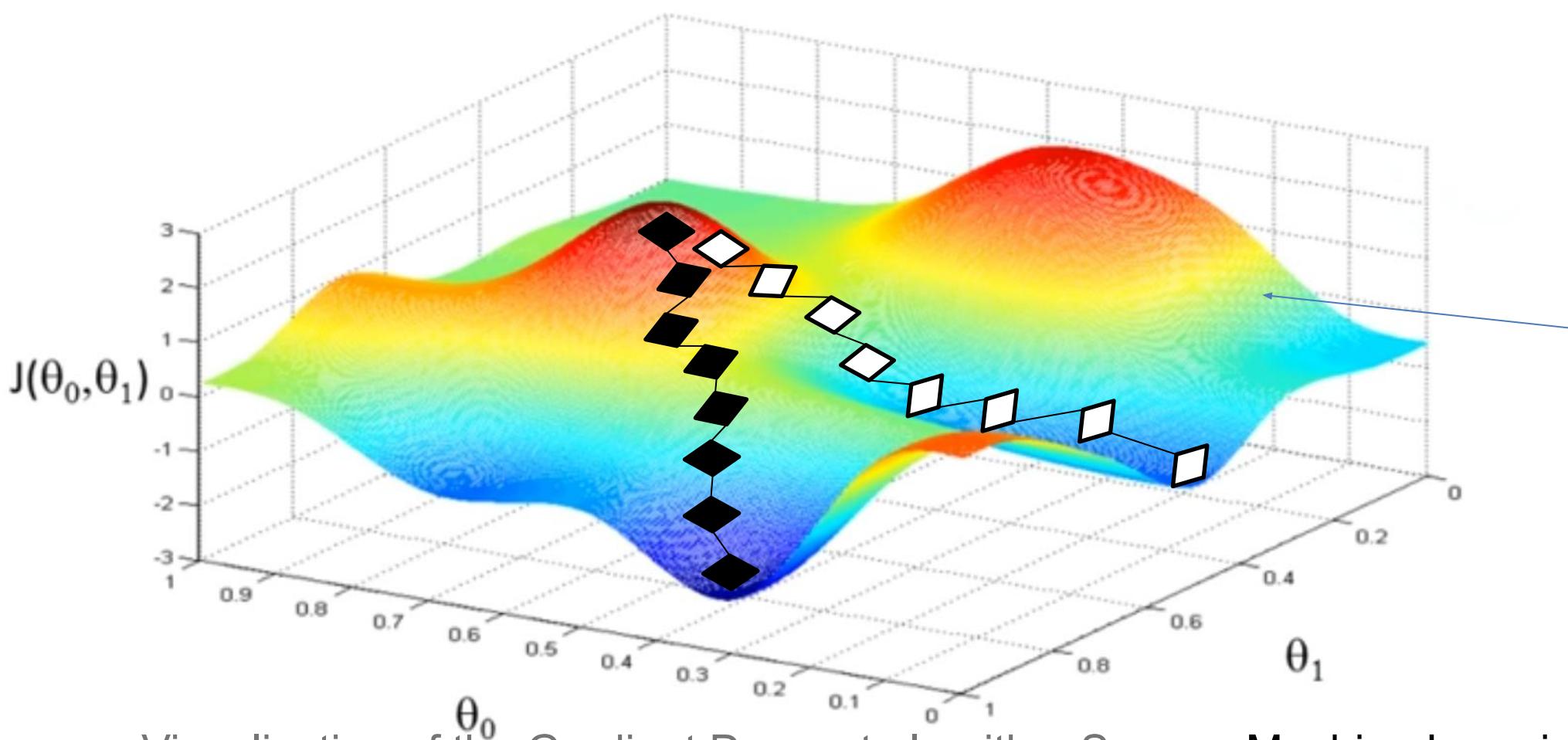


Visualization of the Gradient Descent algorithm Source: [Machine Learning Course](#) by Andrew Ng



Hypothesis space of all possible weight vectors and their corresponding errors values  $E$

Visualization of the Gradient Descent algorithm Source: [Machine Learning Course](#) by Andrew Ng



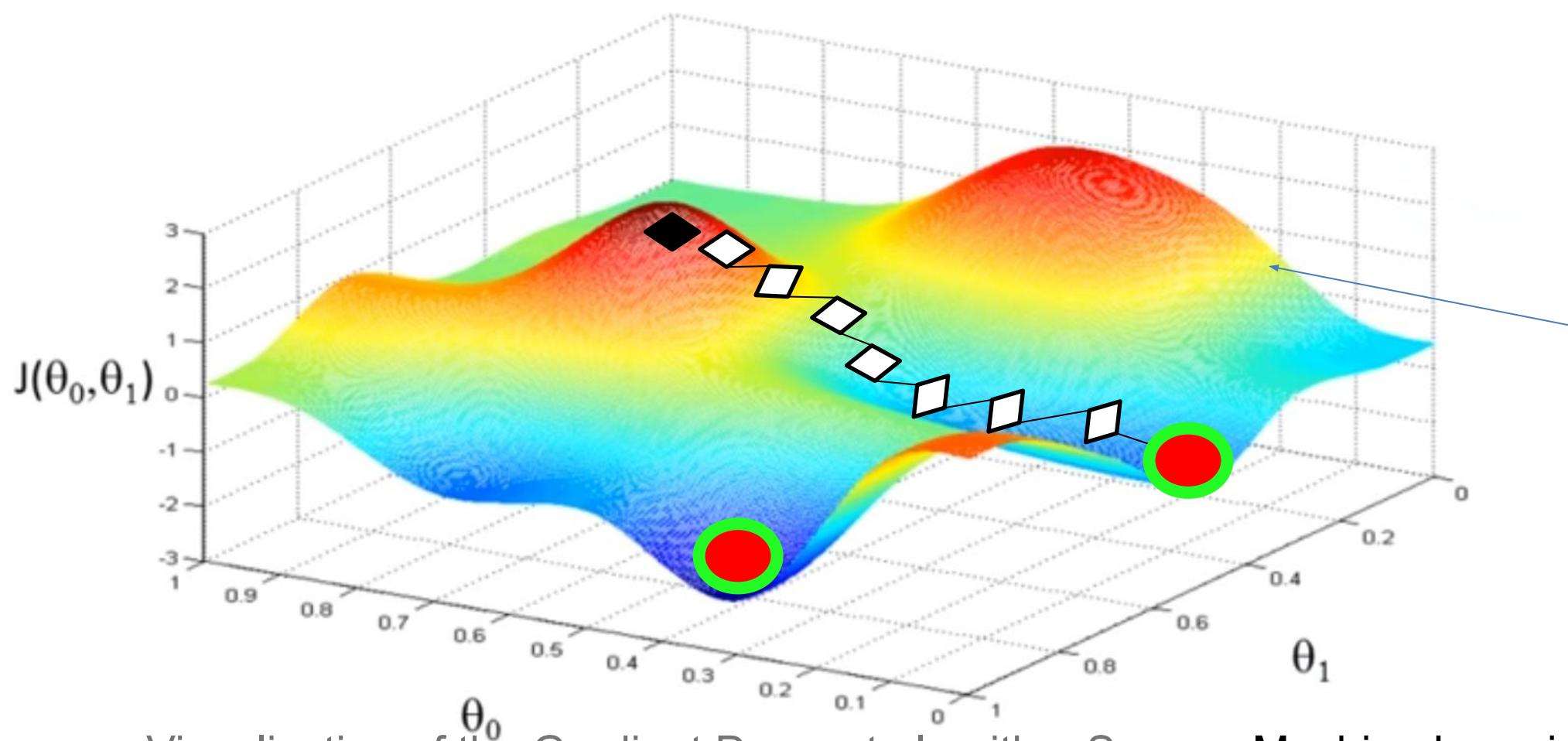
Hypothesis space of all possible weight vectors and their corresponding errors values E

Visualization of the Gradient Descent algorithm Source: [Machine Learning Course](#) by Andrew Ng

## Gradient Descent

GD search algorithm searches a weight vector that minimizes E by starting with an arbitrary initial weight vector and keep on modifying it in small steps.

- Observe that when start with different starting point we have reached different local minimum
- This is one of the property of the **gradient descent search algorithm**



Hypothesis space of all possible weight vectors and their corresponding errors values E

Visualization of the Gradient Descent algorithm Source: [Machine Learning Course](#) by Andrew Ng

- GD search algorithm **searches a weight vector that minimizes E** by starting with an **arbitrary initial weight vector** and keep on modifying it in small steps.
- At each step, the **weight vector is modified in the direction that gives the steepest descent** along the error surface shown in previous figure. This process is repeated until global minimum error is reached.

## Gradient Descent Algorithm

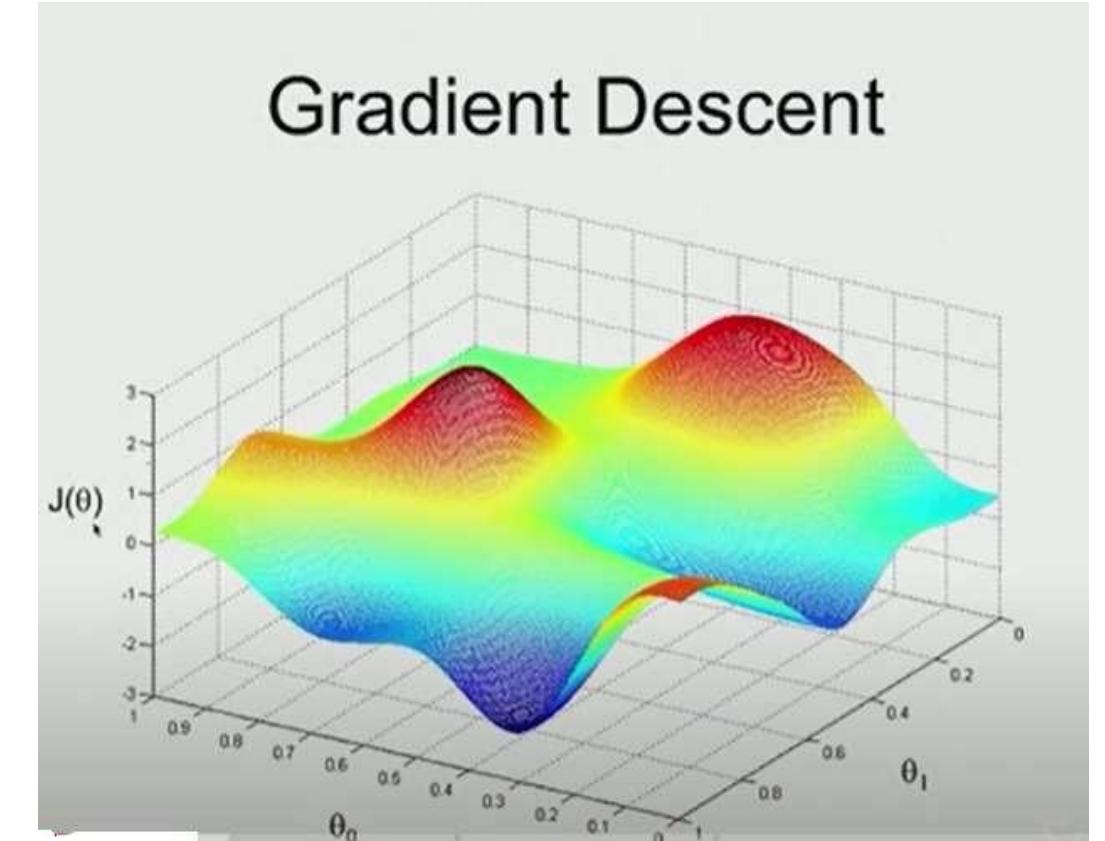
Idea behind GRADIENT DESCENT(GD) ALGORITHM:

Objective : Find optimum values for the parameters so that the value of loss or cost function is minimized.

Gradient Descent is an iterative optimization algorithm used to find the values of parameters( $\theta$ ) to minimize a loss or cost function  $J(\theta)$ .

Depending on number of parameters and model's complexity the cost/loss function might be different and it's corresponding error surface may be complex with so many local minima and a global minimum as shown in the figure.

Note: In the figure, theta 0 and theta 1 are the two parameters of a model and  $J(\theta)$  is the cost function.



## Gradient Descent Algorithm

### Visualization of Gradient Descent:

**Fig.(a) and Fig.(b)represents visualization of Gradient Descent Algorithm in 3-Dimension.**

**Fig.(a)** represents how Gradient Descent algorithm started with **some initial random guess for the parameters** and **Fig.(b)** represents **after several iteration** how the Gradient Descent algorithm approaching **minimum of cost/ loss function**.

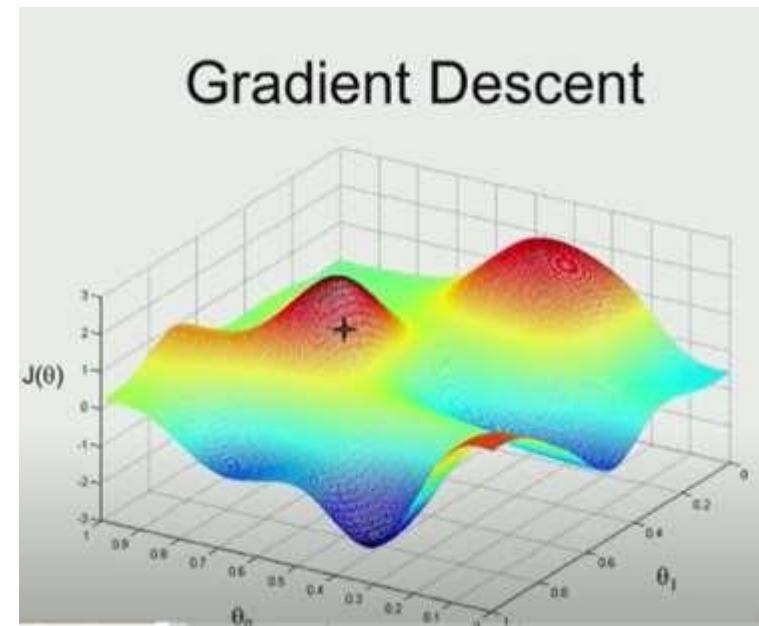


Fig.(a)

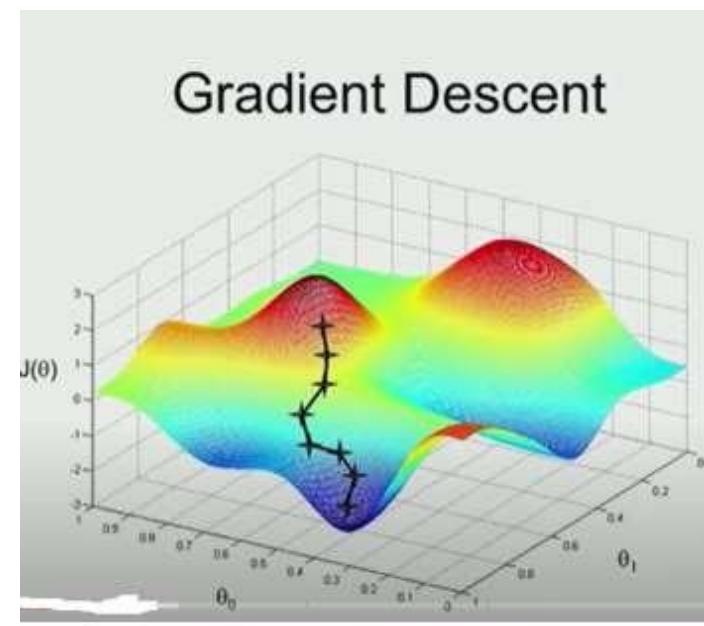


Fig.(b)

## Gradient Descent Algorithm

### Property of Gradient Descent(GD) Algorithm:

One property of GD is that depending on the starting point(where you initialize parameters) and steps taken in different direction leads to different local minimum(shown in Figures (a) and (b)).

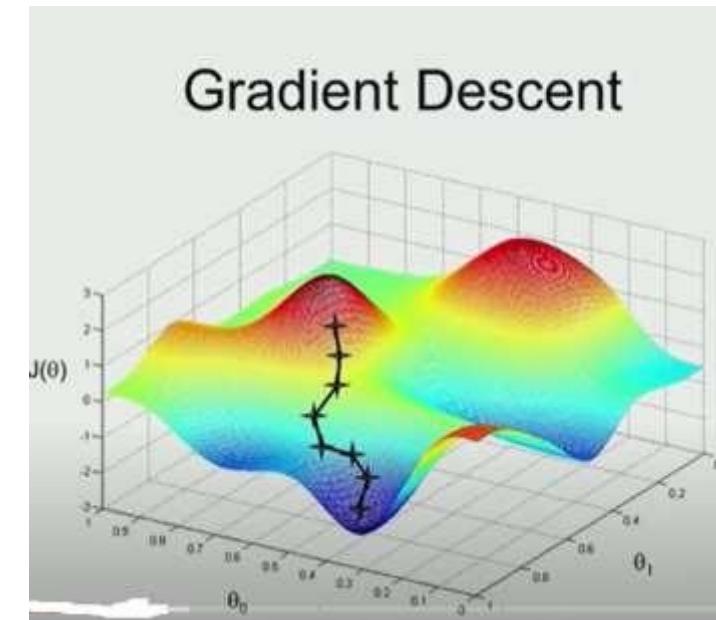


Fig.(a)

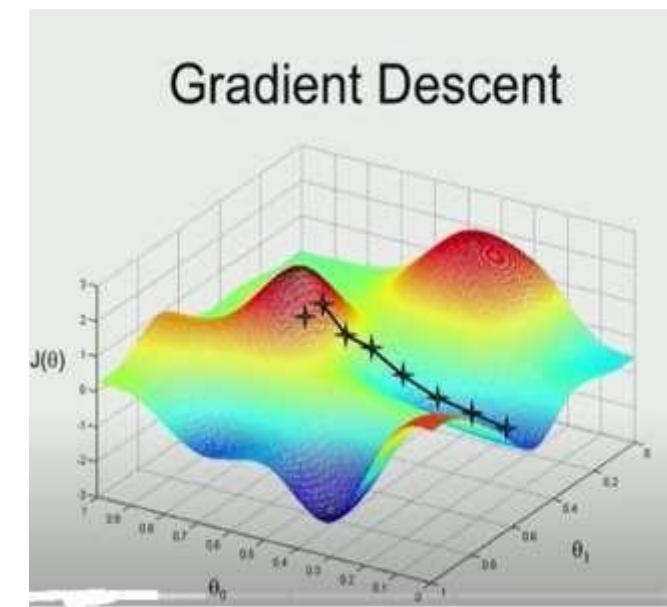


Fig.(b)

## Gradient Descent Algorithm – Working

- Gradient descent works by moving downward toward the pits or valleys in the graph to find the minimum value of the cost/loss function.
- This is achieved by taking the **derivative of the loss function**, as illustrated in the coming slides.
- During each iteration, gradient descent step-downs the cost function in the direction of the steepest descent.
- By adjusting the parameters in this direction, it seeks to reach the **minimum of the loss function and find the best-fit values for the parameters**.
- The size of each step is determined by a parameter  $L$  or  $\alpha$  known as **Learning Rate**.



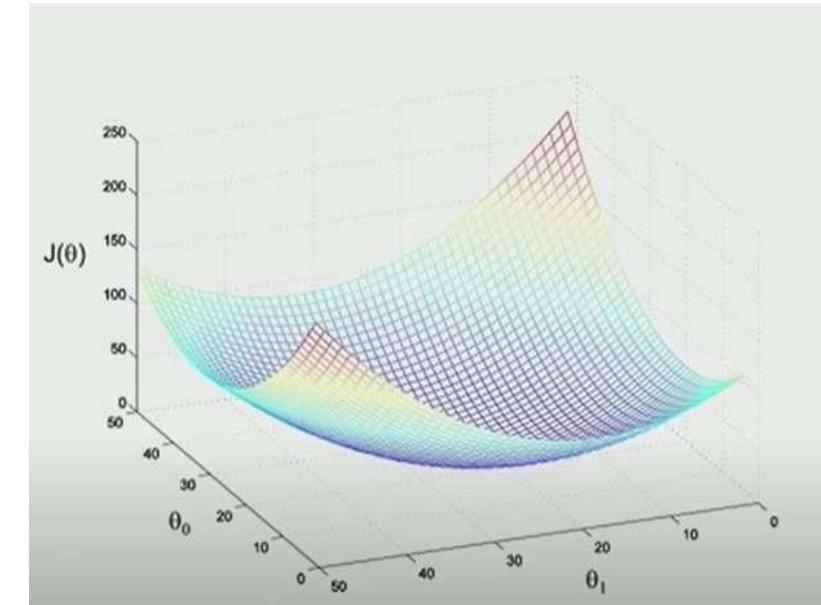
## Gradient Descent Algorithm

### Linear Regression with Gradient Descent:

**Objective :** Find optimum values for slope(m) and y-intercept(b) to find the best fitted line for the given data.

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

**Note:** In the figure, theta 0 and theta 1 are corresponding to y-intercept(i.e. b) and slope(i.e. m) respectively.



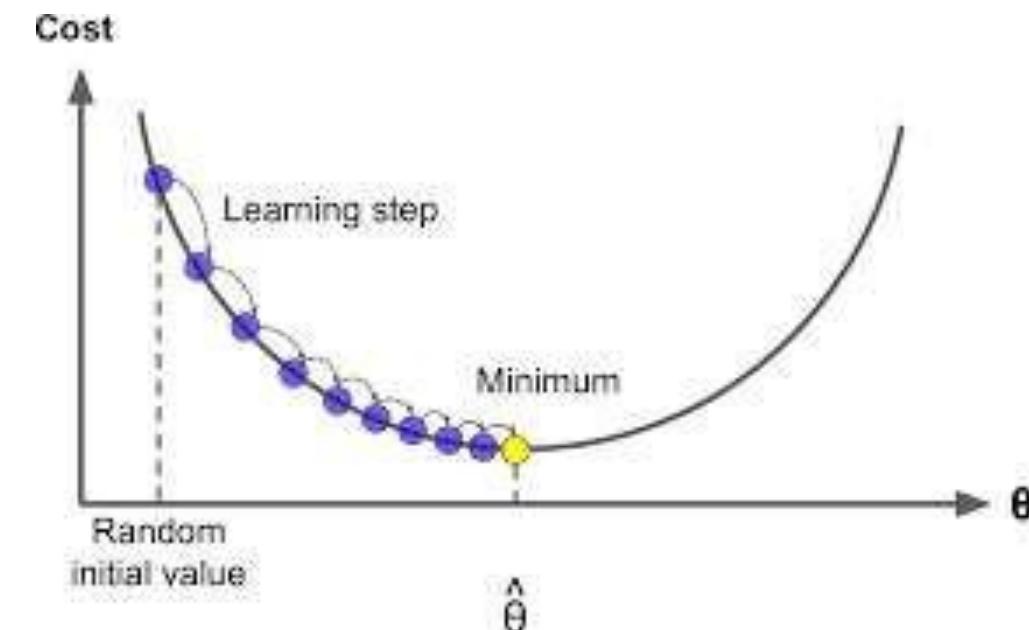
**Note:** In the above MSE formula, the predicted line( $mx_i + b$ ), considers only two parameters. However, the same idea can be extended to any number of parameters in case of multiple linear regression. In such case, the error space dimension will be higher which can't be visualized.

## Gradient Descent Algorithm

### Visualization of Cost Function in 2D:

This figure represents the **visualization of cost function vs. one parameter (2D graph)** and how gradient descent algorithm takes steps to reach the **minimum of the cost function with learning rate(alpha)**.

**Learning Rate Alpha:** is a positive constant determines how much bigger the steps are going to be.



**Note :**

If alpha is extremely large then it is going to overshoot the minimum value of cost function( $J$ ) and might diverge.  
If alpha is smaller value then smaller steps are taken to reach the minimum of cost function.

### Steps to Find Optimum Values of m and b:

1. **Initialize m and b with random values.**
2. **Define the loss function:** The loss function measures how far off the predictions are from the actual values. A common loss function for linear regression is the Mean Squared Error (MSE):

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

Where n is the number of data points,  $y_i$  is the actual y value for the  $i^{\text{th}}$  data point and the corresponding x value.

3. **Calculate the gradient:** Calculate the partial derivatives of the loss function with respect m and b. This tells us how much the loss will change if we make small adjustments to m and b:

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

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

### Steps to Find Optimum Values of m and b:

4. **Update parameters:** Update m and b using the gradients and a learning rate ( $\alpha$ ).  
The learning rate determines the step size in each iteration and should be chosen carefully,

$$m \leftarrow m - \alpha \frac{\partial \text{MSE}}{\partial m}$$
$$b \leftarrow b - \alpha \frac{\partial \text{MSE}}{\partial b}$$

5. **Iterate:** Repeat the steps 3 and 4 for a certain number of iterations or until the parameters converge.

**Note:** To get rid of 2 in the numerator of partial derivatives of MSE( which is our cost function J in Linear Regression) w.r.to m and b, we can also take MSE as

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

Using  $y = mx+b$ , assume  $m=10$ ,  $b=300$ ,  $LR(\alpha)= 0.0001$ . Perform four iterations of gradient descent on this linear regression model to find out the new parameters and observe the reduction in error through each iteration.

First iteration:

$$y = 10^*x + 300$$

$$MSE = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=0}^n (y_i - (mX+b))^2$$

Age	Salary
30	800
37	950
25	600
43	1050
50	1200
29	740
46	1100

Using  $y = mx+b$ , assume  $m=10$ ,  $b=300$ ,  $LR(\alpha)= 0.0001$ . Perform two iterations of gradient descent on this linear regression model to find out the new parameters and observe the reduction in error through each iteration.

First iteration:

$$y = 10^*x + 300$$

$$MSE = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=0}^n (y_i - (m \cdot x + b))^2$$

$$= 1/7 (\text{sum} ((800-600)^2 + (950-670)^2 + (600-550)^2 + (1050-730)^2 + (1200 - 800 )^2 + (740-590 )^2 + (1100 - 760)^2)$$

$$= 74485.7143$$

Age	Salary
30	800
37	950
25	600
43	1050
50	1200
29	740
46	1100

Doing a partial derivative wrt m of the MSE,

$$Dm = \frac{-2}{n} \sum_{i=0}^n (y_i - (mX+b)) \cdot X = -20388.57142$$

$$\begin{aligned} m2 &= m - \text{learning rate} * Dm \\ &= 10 - 0.0001 * (-20388.57142) \\ &= 12.0388 \end{aligned}$$

Doing a partial derivative wrt b of the MSE,

$$Db = \frac{-2}{n} \sum_{i=0}^n (y_i - (mX+b))$$

$$\begin{aligned} b2 &= b - \text{learning rate} * Db \\ &= 300 - 0.0001 * -497.1 \\ &= 300.0497 \end{aligned}$$

Age	Salary
30	800
37	950
25	600
43	1050
50	1200
29	740
46	1100

Now,

$$y = 12.0388 * x + 300.0497$$

$$\text{MSE}_{\text{new}} = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=0}^n (y_i - (mx+b))^2$$

$$\begin{aligned}
 &= 1/7 (\text{sum } ((800-660.94)^2 + (950 - 745.1597)^2 + (600-600.7997)^2 \\
 &+ (1050-817.3397)^2 + (1200-901.5497)^2 + (740-648.9197)^2 \\
 &+ (1100-853.4297)^2)
 \end{aligned}$$

$$= 38957.23$$

Note that in comparison to the start, the error value after one iteration has been reduced.

Let's now predict new y values.

Age	Salary
30	800
37	950
25	600
43	1050
50	1200
29	740
46	1100

Second iteration:

$$y = 12.0388 * x + 300.0497$$

$$\text{MSE} = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2$$

$$\begin{aligned}
 &= 1/7 (\text{sum}((800 - 660.9)^2 \\
 &+ (950 - 745.15)^2 + (600 - \\
 &600.79)^2 + (1050 - 1050.8)^2 + \\
 &(1200 - 901.5)^2 + (740 - 648.9)^2 \\
 &+ (1100 - 853.4)^2)) \\
 &= 38957.23
 \end{aligned}$$

Age	Salary	Ypred	Ypred(1 <sup>st</sup> iter)
30	800	600	660.9
37	950	670	745.15
25	600	550	600.79
43	1050	730	1050.8
50	1200	800	901.5
29	740	590	648.9
46	1100	760	853.4

Doing a partial derivative wrt m of the MSE,

$$D_m = \frac{-2}{n} \sum_{i=0}^n (y_i - mx + b) \cdot x = -14443.2337$$

$$m_2 = m - \text{learning rate} * D_m$$

$$12.0388 - 0.0001 * (-14443.2337) \\ = 13.4831$$

Doing a partial derivative wrt b of the MSE,

$$D_b = \frac{-2}{n} \sum_{i=0}^n (y_i - mx + b) = -345.59$$

$$b_2 = b - \text{learning rate} * D_b$$

$$300.0497 - 0.0001 * -345.59 \\ = 300.0843$$

Age	Salary	Ypred	Ypred(1 <sup>st</sup> iter)
30	800	600	660.9
37	950	670	745.15
25	600	550	600.79
43	1050	730	1050.8
50	1200	800	901.5
29	740	590	648.9
46	1100	760	853.4

Therefore, after two iterations,  
 Value of m is 13.4831 and  
 Value of b is 300.0843.

### Table for iteration 1:

Initial m = 10, b = 300

Iteration 1					
x	y	$y^{\wedge} = mx+b$	$y^{\wedge}-y$	$(y^{\wedge}-y)^2$	$(y^{\wedge}-y)*x$
30	800	600	200	40000	6000
37	950	670	280	78400	10360
25	600	550	50	2500	1250
43	1050	730	320	102400	13760
50	1200	800	400	160000	20000
29	740	590	150	22500	4350
46	1100	760	340	115600	15640
		<b>Sum</b>	<b>1740</b>	<b>521400</b>	<b>71360</b>
MSE = $1/7(\sum((y^{\wedge}-y)^2))$	<b>74485.7143</b>				
Dm = $-2(\sum((y^{\wedge}-y)*x))/7$	<b>-20388.571</b>				
m = m - (alpha * Dm)	<b>12.0388571</b>				
Db = $-2(\sum((y^{\wedge}-y)))/7$	<b>-497.14286</b>				
b = b - (alpha * Db)	<b>300.049714</b>				

### Table for iteration 2:

New m = 12.0388, b = 300.0497

Iteration 2					
x	y	$y^{\wedge} = mx+b$	$y^{\wedge}-y$	$(y^{\wedge}-y)^2$	$(y^{\wedge}-y)*x$
30	800	661.214	138.786	19261.6	4163.59
37	950	745.485	204.515	41826.3	7567.04
25	600	601.02	-1.0197	1.03979	-25.4925
43	1050	817.718	232.282	53954.9	9988.12
50	1200	901.99	298.01	88810.1	14900.5
29	740	649.175	90.8251	8249.2	2633.93
46	1100	853.835	246.166	60597.5	11323.6
		<b>Sum</b>	<b>1209.56</b>	<b>272701</b>	<b>50551.3</b>
MSE = $1/7(\sum((y^{\wedge}-y)^2))$	<b>38957.2</b>				
Dm = $-2(\sum((y^{\wedge}-y)*x))/7$	<b>-14443.2</b>				
m = m - (alpha * Dm)	<b>13.4831</b>				
Db = $-2(\sum((y^{\wedge}-y)))/7$	<b>-345.59</b>				
b = b - (alpha * Db)	<b>300.084</b>				

### Table for iteration 3:

New m = 13.4831, b = 300.0843

Iteration 3					
x	y	$y^{\wedge} = mx+b$	$y^{\wedge}-y$	$(y^{\wedge}-y)^2$	$(y^{\wedge}-y)*x$
30	800	704.577	95.4227	9105.49	2862.68
37	950	798.959	151.041	22813.4	5588.52
25	600	637.162	-37.1618	1381	-929.045
43	1050	879.858	170.142	28948.4	7316.12
50	1200	974.239	225.761	50967.9	11288
29	740	691.094	48.9058	2391.78	1418.27
46	1100	920.307	179.693	32289.6	8265.88
		<b>Sum</b>	<b>833.804</b>	<b>147898</b>	<b>35810.5</b>

$$\text{MSE} = 1/7(\sum((y^{\wedge}-y)^2)) \quad \mathbf{21128.2}$$

$$Dm = -2(\sum((y^{\wedge}-y)*x))/7 \quad -10231.6$$

$$m = m - (\alpha * Dm) \quad \mathbf{14.5063}$$

$$Db = -2(\sum((y^{\wedge}-y)))/7 \quad -238.23$$

$$b = b - (\alpha * Db) \quad \mathbf{300.108}$$

### Table for iteration 4:

New m = 14.5063, b = 300.1081

Iteration 4					
x	y	$y^{\wedge} = mx+b$	$y^{\wedge}-y$	$(y^{\wedge}-y)^2$	$(y^{\wedge}-y)*x$
30	800	735.297	64.7029	4186.47	1941.09
37	950	836.841	113.1588	12804.9	4186.88
25	600	662.766	-62.7656	3939.52	-1569.14
43	1050	923.879	126.121	15906.5	5423.2
50	1200	1025.42	174.5769	30477.1	8728.85
29	740	720.791	19.2092	368.993	557.067
46	1100	967.398	132.6021	17583.3	6099.7
		<b>Sum</b>	<b>567.6053</b>	<b>85266.8</b>	<b>25367.6</b>

$$\text{MSE} = 1/7(\sum((y^{\wedge}-y)^2)) \quad \mathbf{12180.973}$$

$$Dm = -2(\sum((y^{\wedge}-y)*x))/7 \quad -7247.8954$$

$$m = m - (\alpha * Dm) \quad \mathbf{15.2310895}$$

$$Db = -2(\sum((y^{\wedge}-y)))/7 \quad -162.17294$$

$$b = b - (\alpha * Db) \quad \mathbf{300.124317}$$

### Table for iteration 5:

New m = 15.231, b = 300.1243

Iteration 5					
x	y	$y^{\wedge} = mx+b$	$y^{\wedge}-y$	$(y^{\wedge}-y)^2$	$(y^{\wedge}-y)*x$
30	800	757.054	42.9457	1844.33	1288.37
37	950	863.671	86.3287	7452.64	3194.16
25	600	680.899	-80.8993	6544.7	-2022.48
43	1050	955.057	94.9427	9014.12	4082.54
50	1200	1061.67	138.326	19134	6916.29
29	740	741.823	-1.8233	3.32442	-52.8757
46	1100	1000.75	99.2497	9850.5	4565.49
		<b>Sum</b>	<b>379.07</b>	<b>53843.6</b>	<b>17971.5</b>
MSE = 1/7(sum((y^{\wedge}-y)^2))			<b>7691.95</b>		
Dm = -2(sum((y^{\wedge}-y)*x))/7				-5134.71	
m = m - (alpha * Dm)			<b>15.7445</b>		
Db = -2(sum((y^{\wedge}-y)))/7				-108.306	
b = b - (alpha * Db)			<b>300.135</b>		

Note : In every iteration the loss or cost function(MSE) value has been reduced significantly.

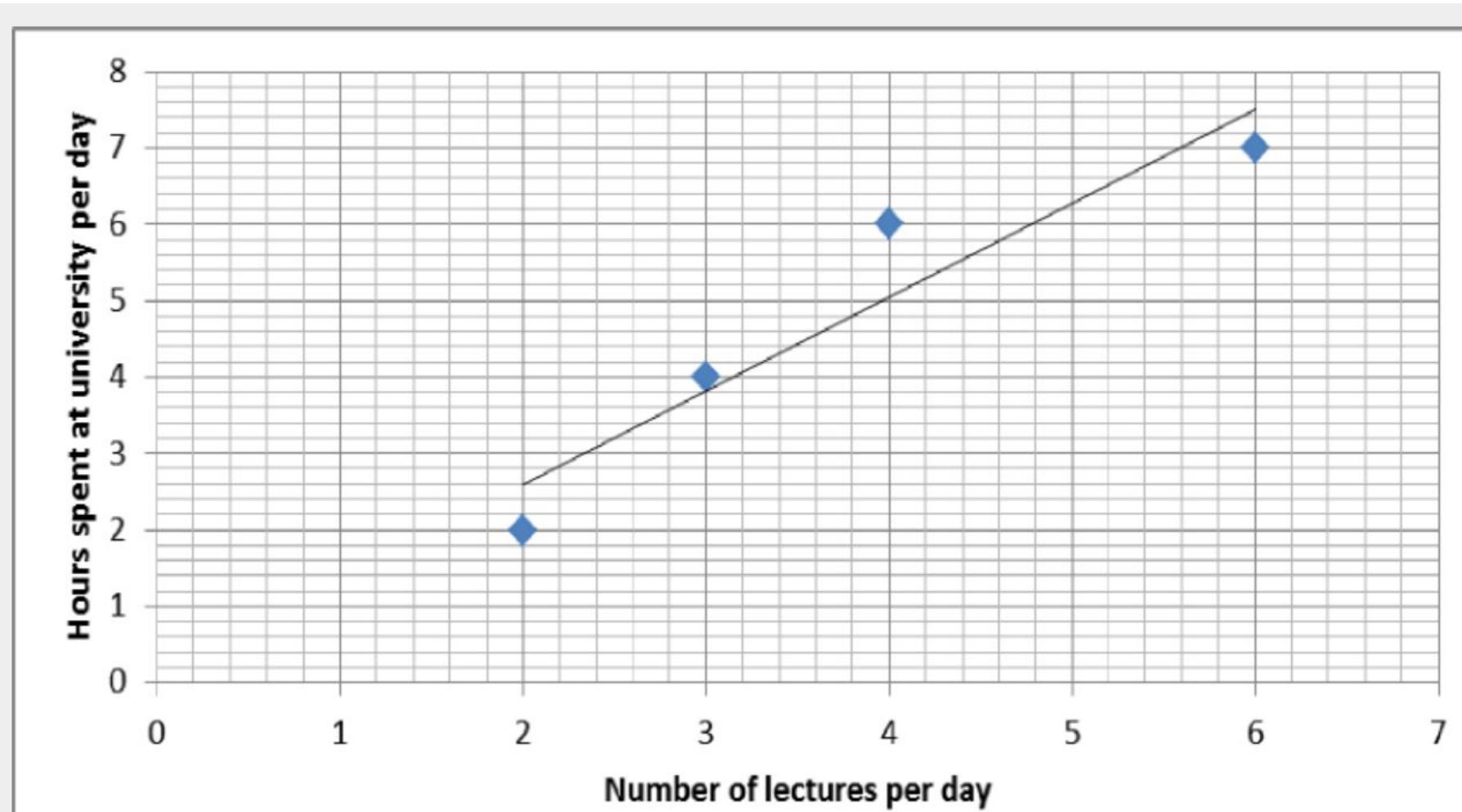
### Measuring Model Performance:

- The **Goodness of fit determines** how the line of regression fits the set of observations. The process of finding the best model out of various models is called **optimization**.

### R-squared method:

- **R-squared** is a statistical method that determines the **goodness of fit**. The value of **R-square lies between 0 to 1**. Where we get **R-square equals 1** when the model perfectly fits the data and there is no difference between the predicted value and actual value.
- However, we get **R-square equals 0** when the model does not predict any variability in the model and it does not learn any relationship between the dependent and independent variables.
- It **measures the strength of the relationship between the dependent and independent variables** on a scale of 0-100%.

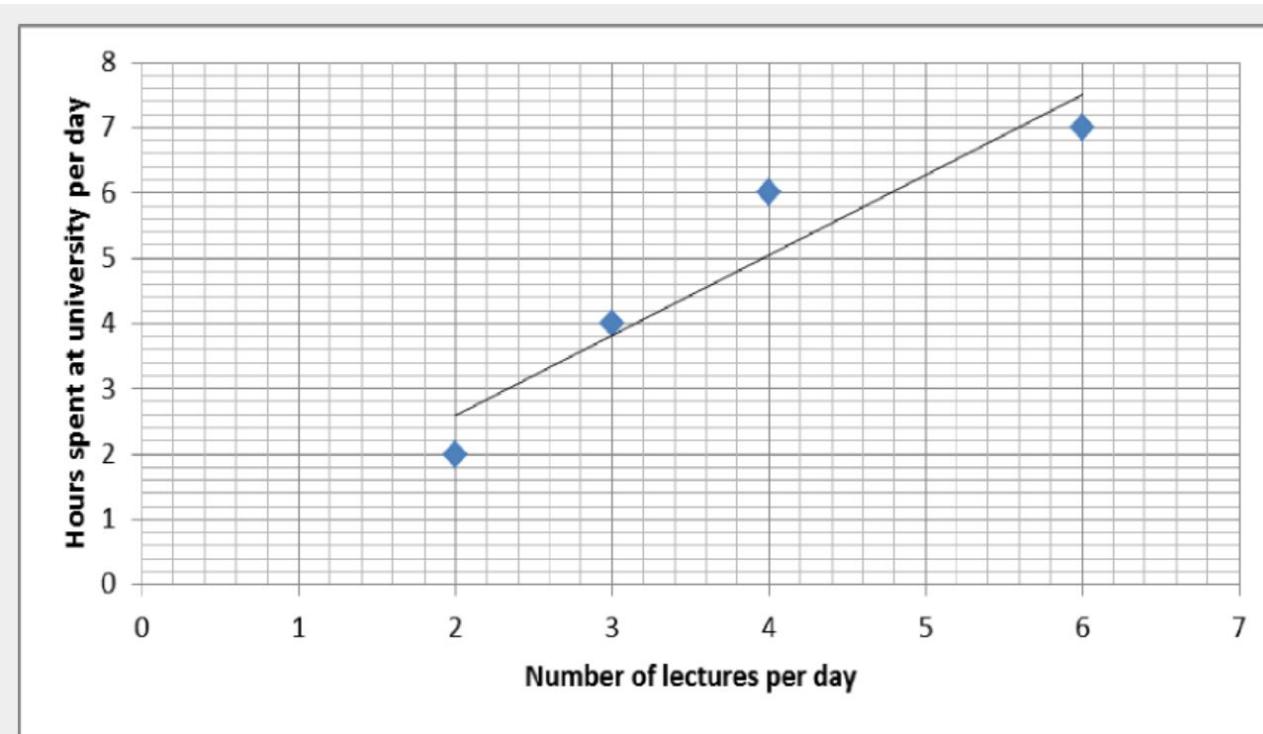
Below is a graph showing how the number of lectures per day affects the number of hours spent at university per day. The regression line is drawn on the graph and it has equation  $\hat{y} = 0.143 + 1.229x$ . Calculate  $R^2$



### R-squared method:

- calculate the mean of the target/dependent variable  $y$  and we denote it by  $\bar{y}$
- Calculate the total sum of squares by subtracting each observation  $y_i$  from  $\bar{y}$ , then squaring it and summing these square differences across all the values. It is denoted by **SST** or **SStot**
- We calculate the Sum of squares due to regression which is denoted by **SSR**(residual sum of squares). This is calculated by subtracting each “predicted value of  $y$ ” denoted by  $\hat{y}_i$  from  $y_i$  and squaring these differences and then summing all the  $n$  terms.

$$\begin{aligned} R^2 &= 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}}, \\ &= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}. \end{aligned}$$



- For the point (2, 2)

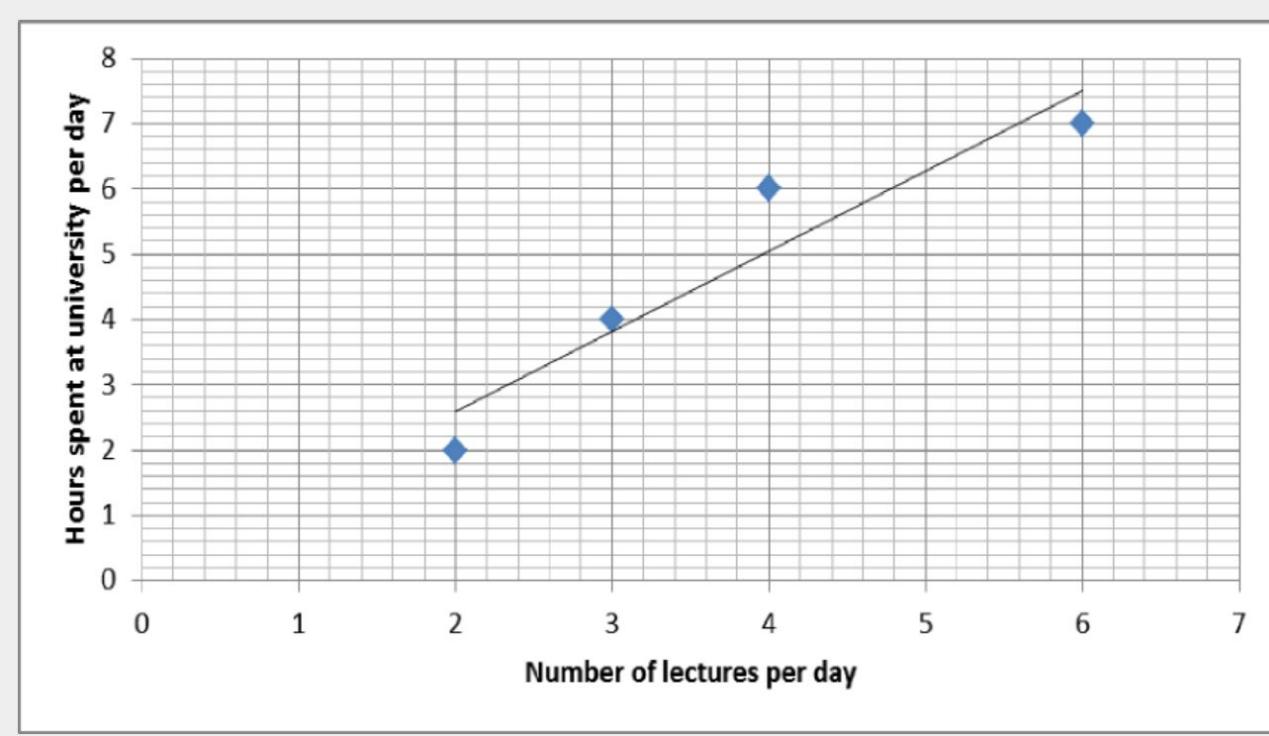
$$\begin{aligned}
 \hat{y} &= 0.143 + 1.229x \\
 &= 0.143 + (1.229 \times 2) \\
 &= 0.143 + 2.458 \\
 &= 2.601
 \end{aligned}$$

The actual value for  $y$  is 2.

Residual = actual  $y$  value – predicted  $y$  value

$$\begin{aligned}
 r_1 &= y_i - \hat{y}_i \\
 &= 2 - 2.601 \\
 &= -0.601
 \end{aligned}$$

As you can see from the graph the actual point is below the regression line, so it makes sense that the residual is negative.



- For the point (3, 4)

$$\begin{aligned}\hat{y} &= 0.143 + 1.229x \\ &= 0.143 + (1.229 \times 3) \\ &= 0.143 + 3.687 \\ &= 3.83\end{aligned}$$

The actual value for  $y$  is 4.

Residual = actual  $y$  value – predicted  $y$  value

$$\begin{aligned}r_2 &= y_i - \hat{y}_i \\ &= 4 - 3.83 \\ &= 0.17\end{aligned}$$

As you can see from the graph the actual point is above the regression line, so it makes sense that the residual is positive.

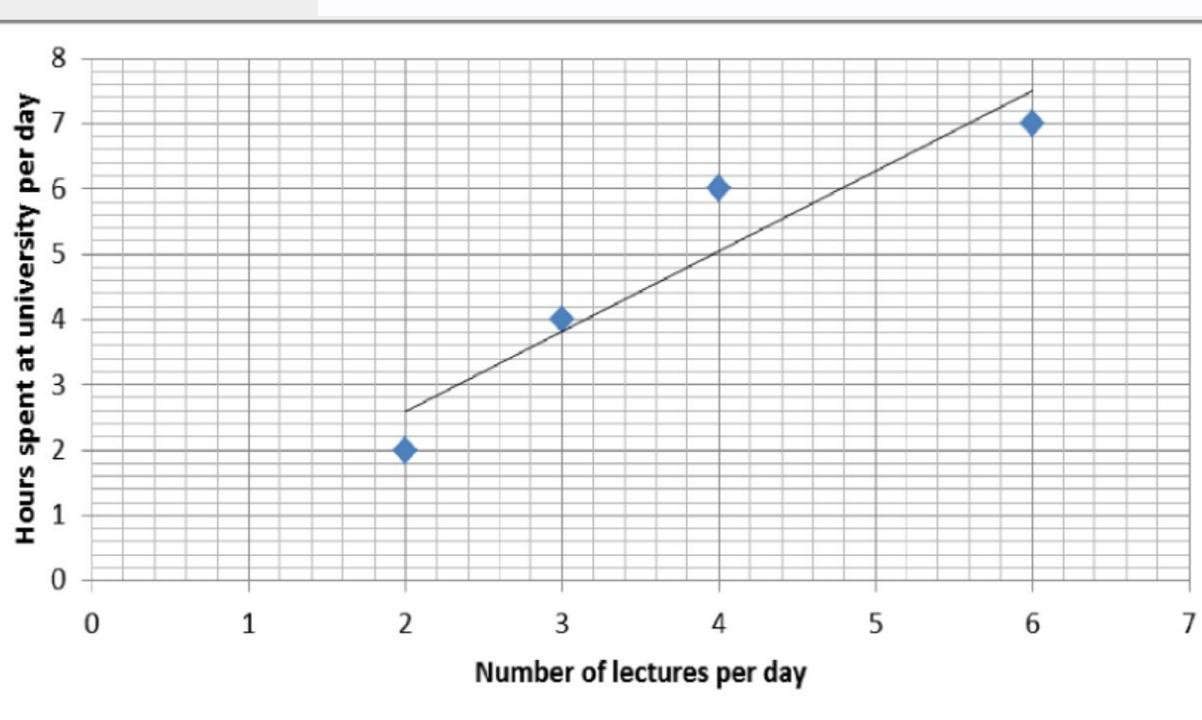
- For the point (4, 6)

$$\begin{aligned}\hat{y} &= 0.143 + 1.229x \\ &= 0.143 + (1.229 \times 4) \\ &= 0.143 + 4.916 \\ &= 5.059\end{aligned}$$

The actual value for  $y$  is 6.

Residual = actual  $y$  value – predicted  $y$  value

$$\begin{aligned}r_3 &= y_i - \hat{y}_i \\ &= 6 - 5.059 \\ &= 0.941\end{aligned}$$



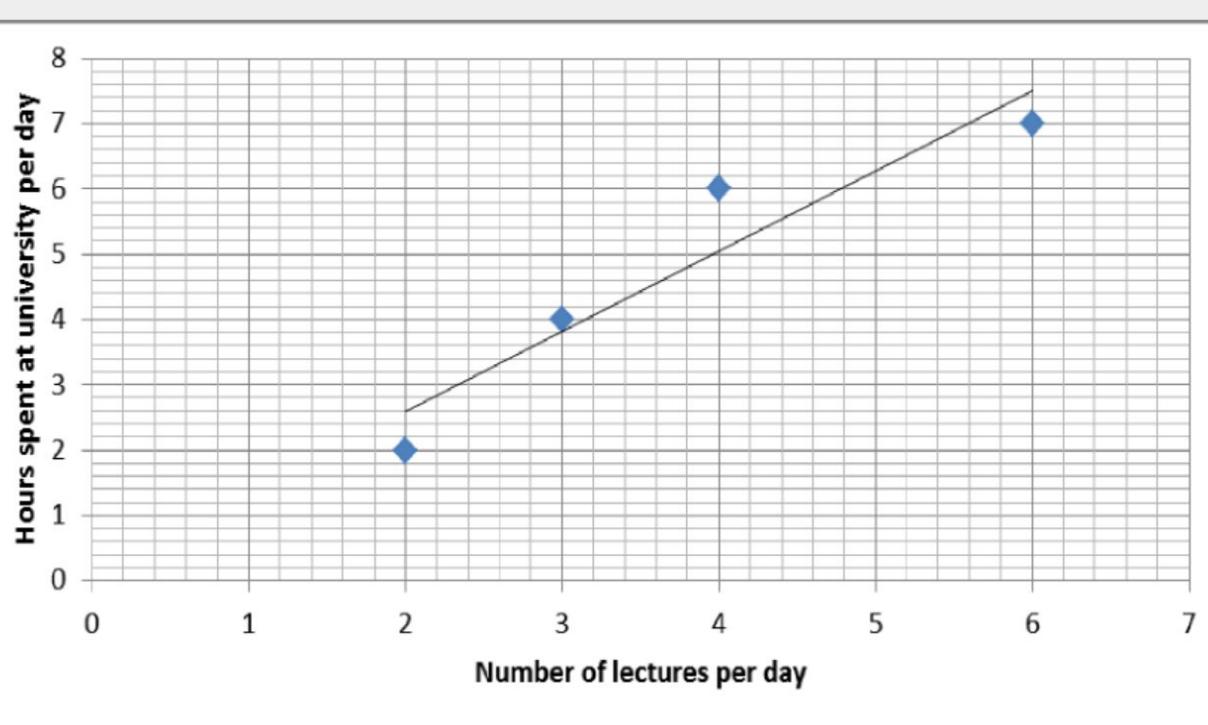
- For the point (6, 7)

$$\begin{aligned}\hat{y} &= 0.143 + 1.229x \\ &= 0.143 + (1.229 \times 6) \\ &= 0.143 + 7.374 \\ &= 7.517\end{aligned}$$

The actual value for  $y$  is 7.

Residual = actual  $y$  value – predicted  $y$  value

$$\begin{aligned}r_4 &= y_i - \hat{y}_i \\ &= 7 - 7.517 \\ &= -0.517\end{aligned}$$



To find the residuals squared we need to square each of  $r_1$  to  $r_4$  and sum them.

$$\begin{aligned}\sum(y_i - \hat{y}_i)^2 &= \sum r_i \\&= {r_1}^2 + {r_2}^2 + {r_3}^2 + {r_4}^2 \\&= (-0.601)^2 + (0.17)^2 + (0.941)^2 - (-0.517)^2 \\&= 1.542871\end{aligned}$$

To find  $\sum(y_i - \bar{y})^2$  you first need to find the mean of the  $y$  values.

$$\begin{aligned}\bar{y} &= \frac{\sum y}{n} \\ &= \frac{2 + 4 + 6 + 7}{4} \\ &= \frac{19}{4} \\ &= 4.75\end{aligned}$$

Now we can calculate  $\sum(y_i - \bar{y})^2$ .

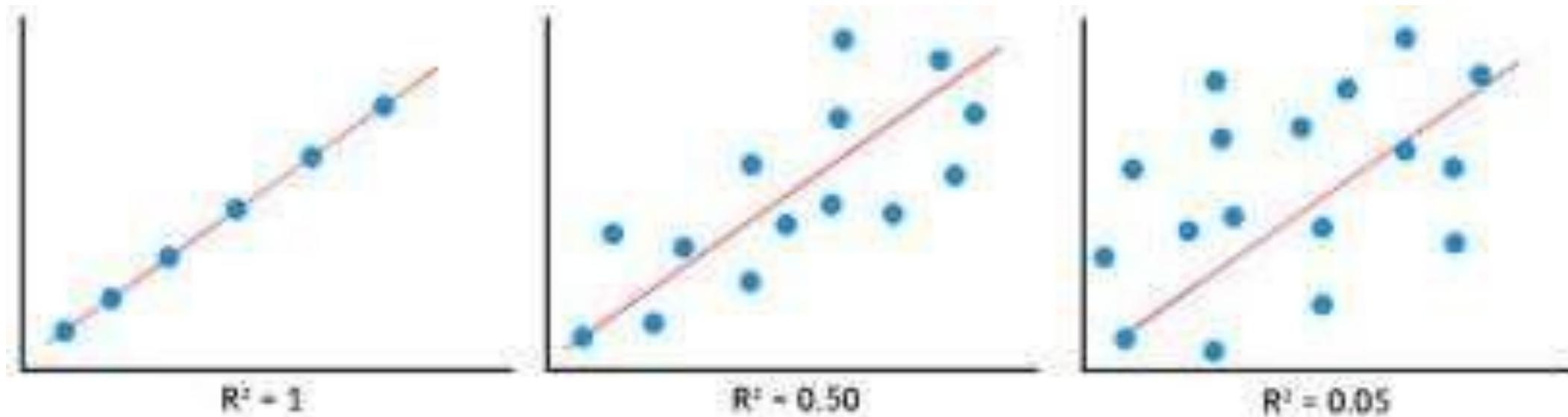
$$\begin{aligned}\sum(y_i - \bar{y})^2 &= (2 - 4.75)^2 + (4 - 4.75)^2 + (6 - 4.75)^2 + (7 - 4.75)^2 \\ &= (-2.75)^2 + (-0.75)^2 + (1.25)^2 + (2.25)^2 \\ &= 14.75\end{aligned}$$

Therefore;

$$\begin{aligned}R^2 &= 1 - \frac{\text{sum squared regression (SSR)}}{\text{total sum of squares (SST)}} \\ &= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \\ &= 1 - \frac{1.542871}{14.75} \\ &= 1 - 0.105 \text{ (3.s.f)} \\ &= 0.895 \text{ (3.s.f)}\end{aligned}$$

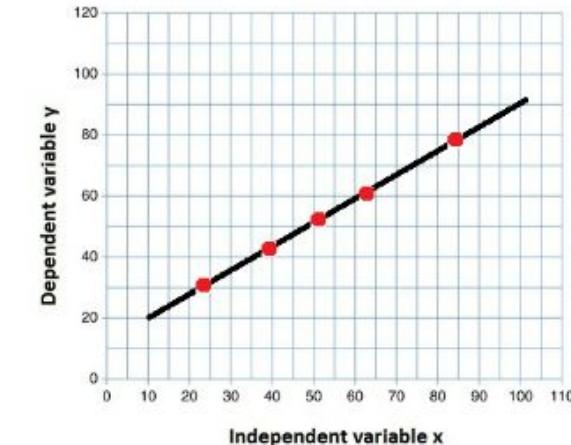
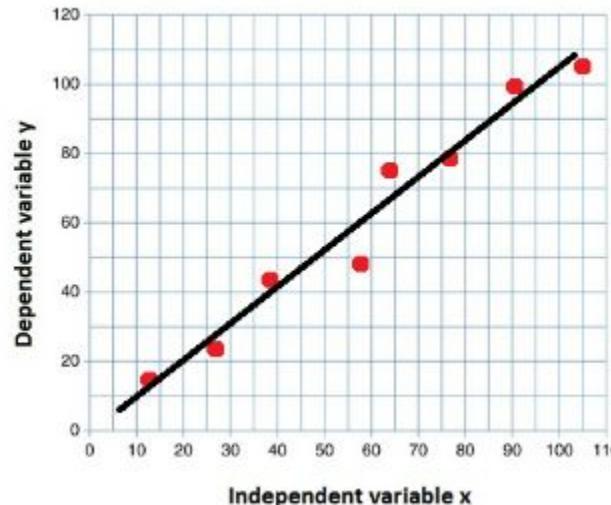
This means that the number of lectures per day account for 89.5% of the variation in the hours people spend at university per day.

- The **high value of R-square** determines the less difference between the predicted values and actual values and hence **represents a good model**.
- It is also called a **coefficient of determination**, or **coefficient of multiple determination** for multiple regression.
- It can be calculated from the formula:  $R\text{-squared} = \frac{\text{Explained variation}}{\text{Total Variation}}$



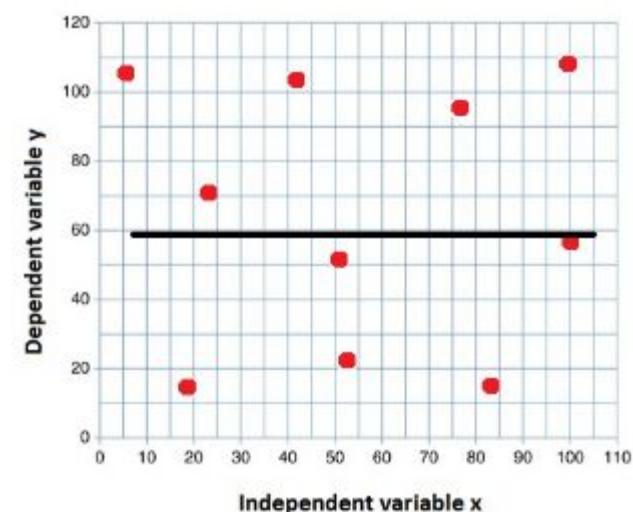
## Measuring Model Performance - Interpretation of the $R^2$ value

$R^2 = 1$  All the variation in the  $y$  values is accounted for by the  $x$  values



$R^2 = 0.83$  83% of the variation in the  $y$  values is accounted for by the  $x$  values

$R^2 = 0$  None of the variation in the  $y$  values is accounted for by the  $x$  values

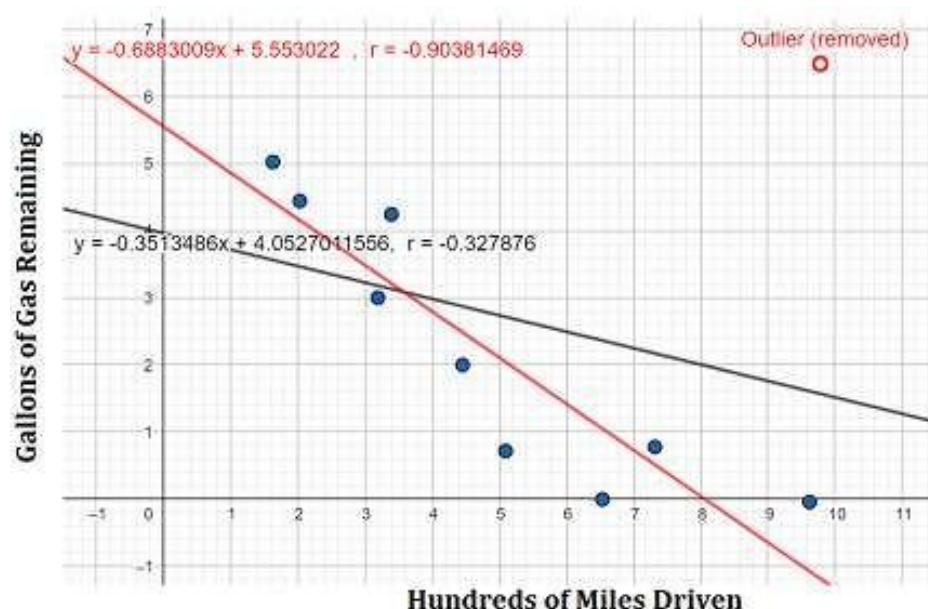


# Machine Learning UE22CS352A

## Outlier – Graphical Representation

### Effect of Outlier in Model Prediction:

An outlier can be higher or lower than expected, or displaced more to the right or left than expected. Outliers can influence regression lines, making the regression less accurate in predicting other data.



## Advantages vs Disadvantages of Linear Regression

Advantages	Disadvantages
Simple to implement and easier to interpret the output coefficients.	Outliers can have huge effects on the regression and boundaries are linear in this technique.
When you know the relationship between the independent and dependent variable have a linear relationship, this algorithm is the best to use because of it's less complexity compared to other algorithms.	Linear regression assumes a linear relationship between dependent and independent variables. That means it assumes that there is a straight-line relationship between them. It assumes independence between attributes.

## Advantages vs Disadvantages of Linear Regression

Advantages	Disadvantages
<p>Linear Regression is susceptible to overfitting but it can be avoided using some dimensionality reduction techniques, regularization (L1 and L2) techniques and cross-validation.</p>	<p>It also looks at a relationship between the mean of the dependent variables and the independent variables. Just as the mean is not a complete description of a single variable, linear regression is not a complete description of relationships among variables.</p>

## References

---

- <https://www.educba.com/linear-regression-analysis/>
- <https://towardsdatascience.com/linear-regression-using-gradient-descent>
- <https://www.geeksforgeeks.org/ml-advantages-and-disadvantages-of-linear-regression/>
- <https://www.javatpoint.com/linear-regression-in-machine-learning>
- <https://www.ibm.com/topics/linear-regression>
- <https://www.javatpoint.com/multiple-linear-regression-in-machine-learning>
- <https://www.scribbr.com/statistics/multiple-linear-regression/>
- <https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931>
- <https://www.vedantu.com/mathematics/linear-regression>
- <https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/>
- <https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/#:~:text=Linear%20regression%20shows%20the%20linear,is%20called%20simple%20linear%20regression.>

# MACHINE LEARNING (UE22CS352A)



**PES**  
UNIVERSITY

## Logistic Regression

---

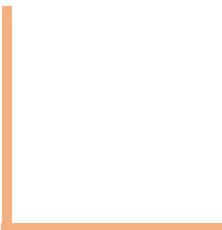
**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

# MACHINE LEARNING

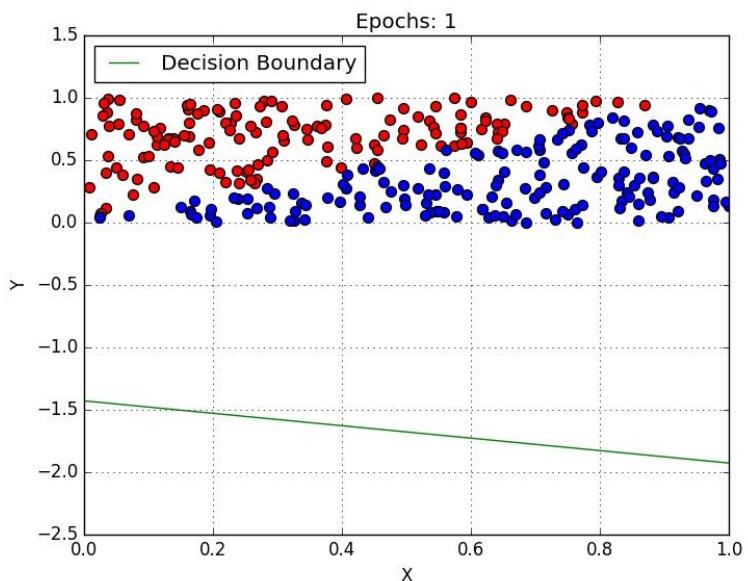
---

## Introduction to Machine Learning

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.

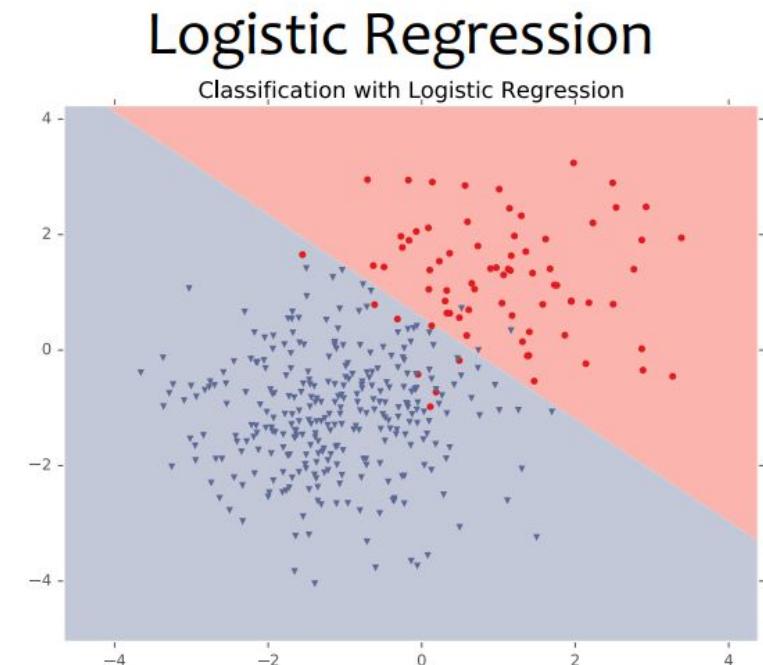


- Logistic regression is a supervised machine learning algorithm mainly used for classification tasks where the goal is to predict the probability that an instance of belonging to a given class.
- It is used for classification algorithms and its name is logistic regression.
- It's referred to as regression because it takes the output of the linear regression function as input and uses a sigmoid function to estimate the probability for the given class.



- It also estimates the **probability that an event occurs** for a randomly selected observation versus the **probability that the event doesn't occur**.
- It predicts the **effect of a series of variables** on a binary response variable.
- It classifies observations by estimating the **probability that an observation is in a particular category**.

Example: The effect of increase in body weight (numerical variable) in causing sleep apnea (categorical variable).



### Approaching the problem using logistic regression

#### 1. Setting up the Problem:

You have two possible outcomes for each mortgage application: either it's approved (label 1) or it's not approved (label 0). This is also a binary classification problem, just like the spam email classification.

#### 2. The Basic Idea:

Logistic regression aims to find a way to express the relationship between the features (like credit score, income, employment status, etc.) of a mortgage applicant and the probability that their application will be approved.

#### 3. Sigmoid Function:

In logistic regression, we still use the sigmoid function. This function takes any input and squashes it between 0 and 1. In the context of predicting mortgage approval, the sigmoid function will give us the probability that an applicant's mortgage will be approved based on their features.

### 4. Linear Combination:

Logistic regression assumes a linear relationship between the features of a mortgage application (such as credit score, income, etc.) and the log-odds of the application being approved. This linear combination of features and their corresponding weights is then passed through the sigmoid function to produce the final probability of mortgage approval.

### 5. Training the Model:

During the training phase, it iteratively adjusts the weights (coefficients) assigned to each feature. The objective is to find the best combination of weights that aligns the predicted probabilities with the actual outcomes (1 for approved, 0 for not approved) in the training dataset. The algorithm uses optimization techniques to minimize the difference between predicted and actual outcomes.

### 6. Interpreting the Model:

You can interpret the coefficients of the features in the context of mortgage approval. A positive coefficient indicates that an increase in that feature's value raises the odds of the mortgage being approved, while a negative coefficient implies the opposite. For instance, a positive coefficient for credit score suggests that higher credit scores are associated with higher odds of mortgage approval.

- Logistic regression is preferred over linear regression when dealing with binary classification problems that involve **modeling probabilities**, **handling imbalanced data**, **capturing non-linear relationships**, and providing interpretable insights into the influence of features on the outcome.
- Logistic regression is generally **more robust to outliers** compared to linear regression due to the **use of sigmoid function**.
- Logistic regression coefficients can be interpreted as **log-odds ratios**.
- It is easy to understand how **each feature contributes to the likelihood of a certain outcome**. Linear regression coefficients, while interpretable, might not provide the same kind of insight for binary classification problems.

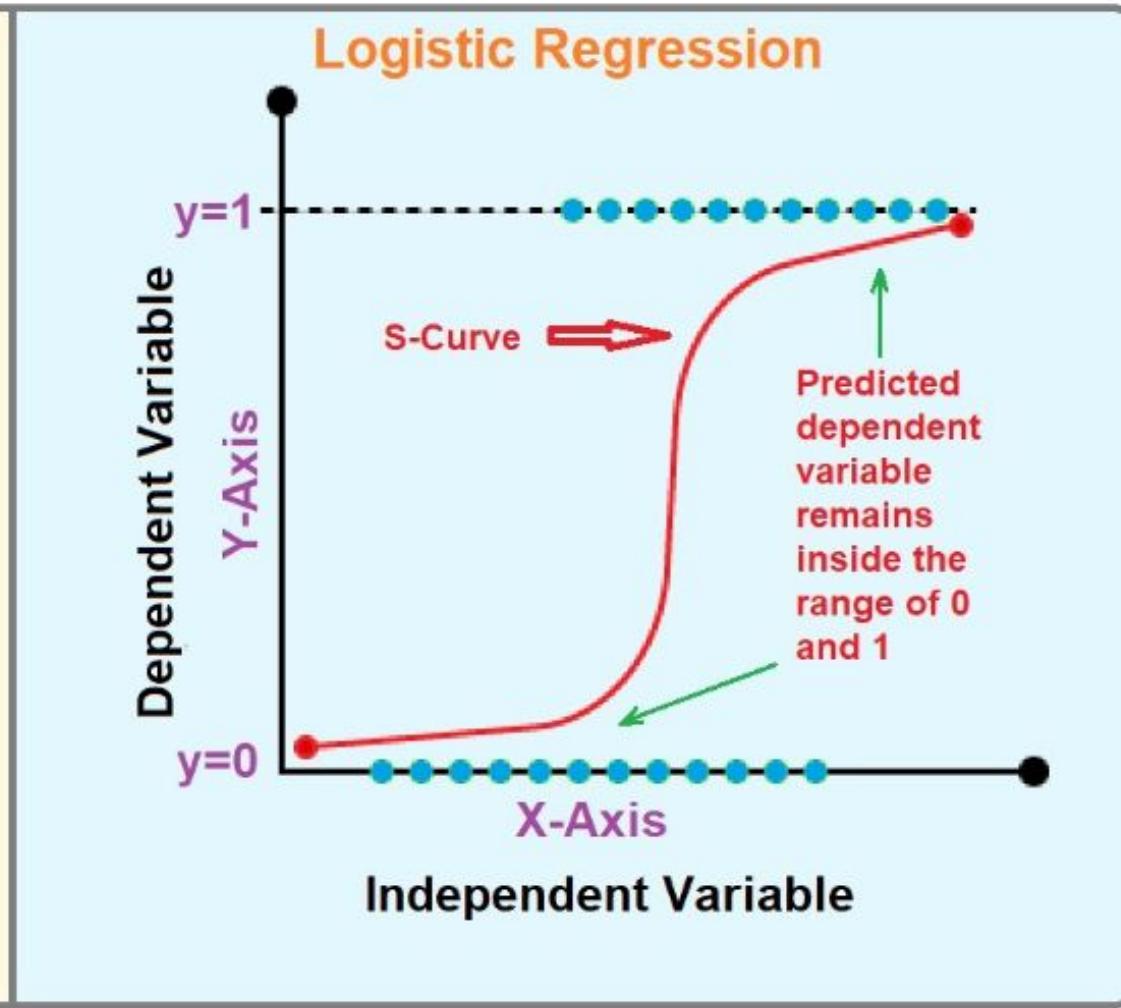
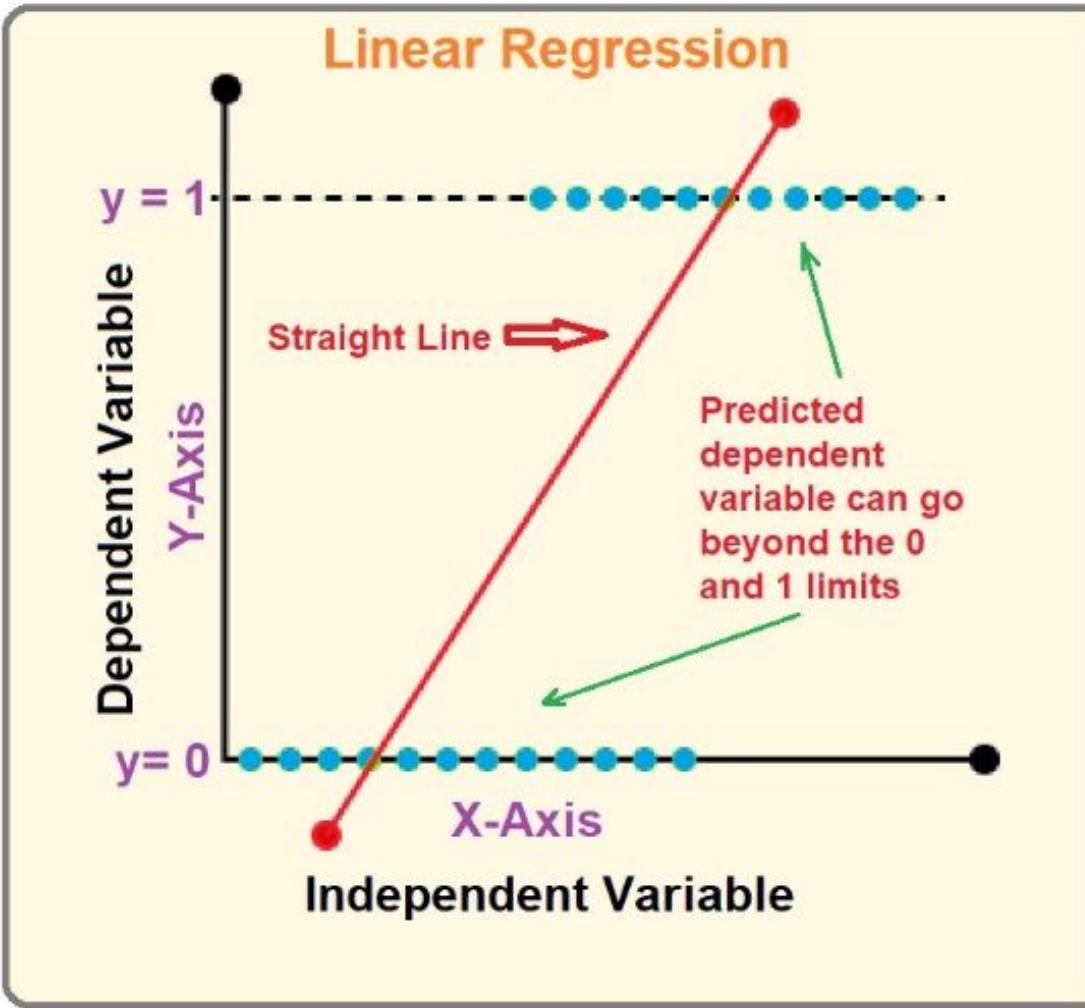
## Difference between Linear Regression & Logistic Regression

Linear Regression	Logistic Regression
Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic regression is used to predict the categorical dependent variable using a given set of independent variables.
Linear regression is used for solving Regression problem.	It is used for solving classification problems.
In this we predict the value of continuous variables	In this we predict values of categorical variables
In this we find best fit line.	In this we find S-Curve .
Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for Estimation of accuracy.

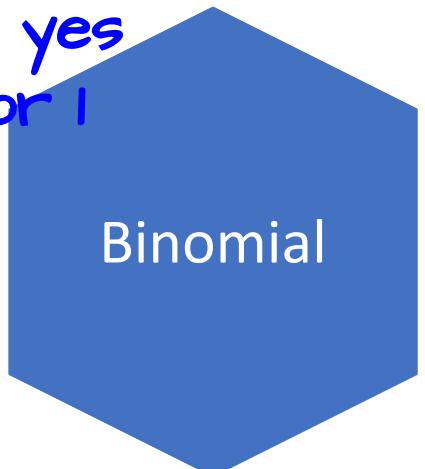
## Difference between Linear Regression & Logistic Regression

Linear Regression	Logistic Regression
The output must be continuous value, such as price, age, etc.	Output is must be categorical value such as 0 or 1, Yes or no, etc.
It required linear relationship between dependent and independent variables.	It not required linear relationship.
There may be collinearity between the independent variables.	There should not be collinearity between independent variable.

## Linear Regression vs Logistic Regression



There can be only two possible types of the dependent variables, like yes or no OR 0 or 1



There can be 3 or more possible unordered types of the dependent variable, such as cat, dogs or sheep.



There can be 3 or more possible ordinal dependent variables, such as "small", "medium", or

- Probability

$$p = \frac{\text{Outcomes of Interest}}{\text{All possible outcomes}}$$

**Example:**  $P(\text{getting heads from a fair coin}) = \frac{1}{2} = 0.5$

$P(\text{getting heads from a biased coin}) = 7/10 = 0.7$  (where heads appear 7 out of 10 trials)

- Odds

$$\text{Odds} = \frac{\text{Probability of an Event Occurring}}{\text{Probability of an Event not occurring}} = \frac{p}{1-p}$$

**Example:** Odds (heads from a fair coin) =  $0.5/0.5 = 1$  (or) 1:1

Odds (heads from a biased coin) =  $0.7/0.3 = 2.333$  (or) 2.333:1

- Odds ratio – it is the ratio of two odds.

$$\text{Example: Odds ratio} = \frac{\text{Odds (heads from a biased coin)}}{\text{Odds (heads from a fair coin)}} = 2.333/1 = 2.333$$

- Interpreting Odds ratio

**Example:** the odds of getting heads on the biased coin are 2.333 times greater than the fair coin.

## Some Terminologies

Let's consider a scenario where you play 10 games of chess against an artificially intelligent (AI) system and 4 times you are able to beat it.

**Odds** are the ratio of **something happening** to **something not happening**. Whereas, **Probability** is the ratio of **something happening** to **everything that could happen**.

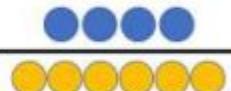
As per the scenario, there are 4 times you are able to beat the system, so the odds of you winning the game are 4 to 6, i.e., out of total 10 games, you win 4 games and lose 6 games.

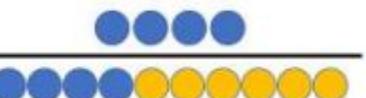
**Odds of winning:**  $4/6 = 0.6666$

**Probability of winning:**  $4/10 = 0.40$

**Probability of losing:**  $6/10 = 0.60$

which is also equal to **1 - Probability of winning:**  $1 - 0.40 = 0.60$

$$\text{Odds} = \frac{\text{Blue Circles}}{\text{Yellow Circles}}$$


$$\text{Probability} = \frac{\text{Blue Circles}}{\text{Total Circles}}$$


### ODDS RATIO IN LOGISTIC REGRESSION:

The odds ratio for a variable in logistic regression represents how the odds change with a one unit increase in that variable holding all other variables constant.

**Example:** Effect of weight (in pounds) with respect to sleep apnea (yes or no).

Let us assume that the weight variable had an odds ratio of 1.07.

This means that one pound increase in body weight increases the odds of having sleep apnea by a factor of 1.07. i.e, 7% increase in the risk of having sleep apnea (low risk).

**NOTE:** Odds ratio holds true for any interval.

**Example:** If a person's weight goes up from 180 to 181 pounds or from 135 to 136 pounds, the odds ratio would still remain 1.07.

### **BERNOULLI DISTRIBUTION & LOGISTIC REGRESSION:**

The dependent variable in logistic regression follows the Bernoulli distribution having an unknown probability  $p$ .

In logistic regression, we are estimating  $p$  for any given linear combinations of the independent variables (input features).

### **LOGIT FUNCTION:**

Logit function is a function that maps or links the linear combination of input variables and the output variable which follows Bernoulli probability distribution.

The goal of logistic regression is to estimate  $p$  for a linear combination of input features (linear combination of independent variables).

**ODDS & LOGIT FUNCTION:**

The logit is nothing but the natural log of the odds. The log-odds of the event ( $Y=1$ ) =  $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$

The sigmoid function transforms the log-odds into a **probability value between 0 and 1**.

This is mathematically represented as:

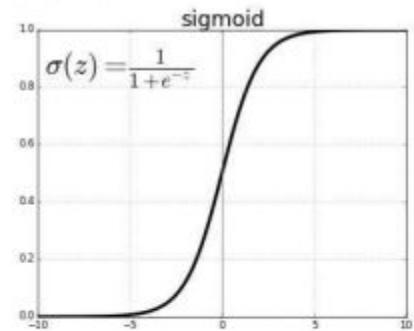
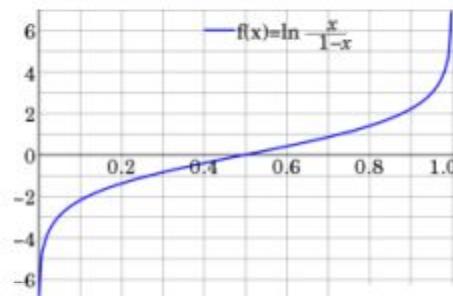
$$\text{Logit}(p) = \ln(\text{odds}) = \ln\left(\frac{p}{1-p}\right) = \ln(p) - \ln(1-p)$$

**RELATIONSHIP BETWEEN LOGIT AND SIGMOID FUNCTION:**

$\text{Logit}^{-1}(z) = 1 / 1+e^{-z}$ , where  $z$  = linear combination of input variables.

$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$  where  $X_1, X_2, \dots$  etc are input features.

The inverse logit function is nothing but, the sigmoid function. In logistic regression, we try to fit/model the data of the form  $y' = 1 / (1+e^{-z})$



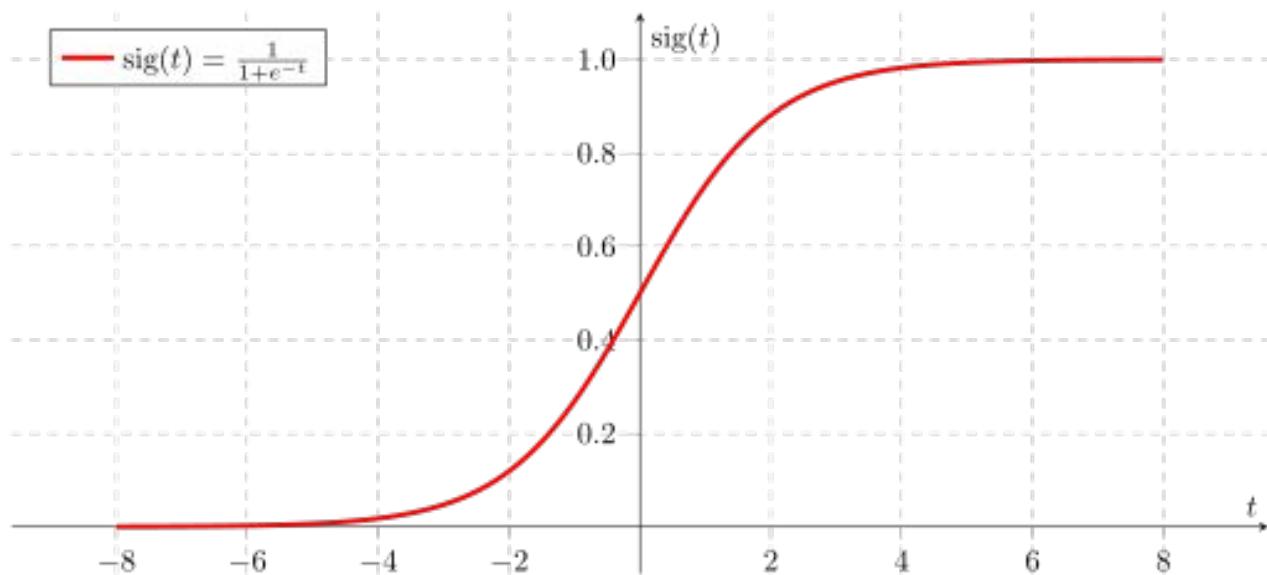
**ODDS & LOGIT FUNCTION:**

Connection between input features and output variables through logit function:

Let

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

The  $\ln$  term on the LHS can be removed by raising the RHS as a power of e.

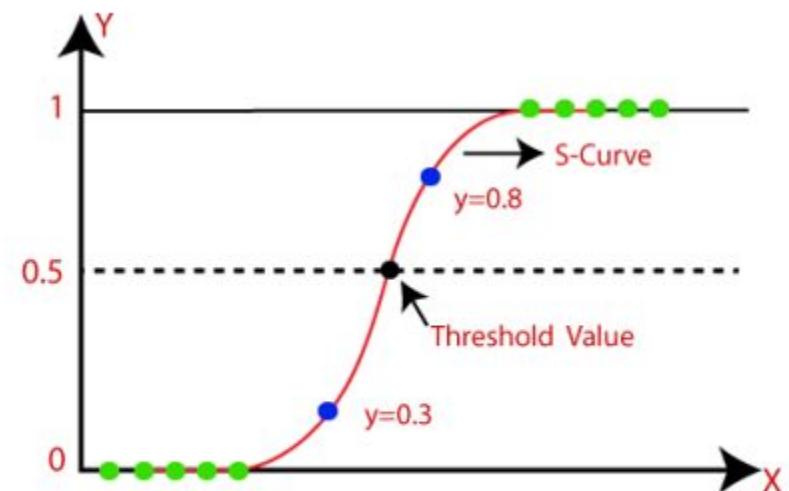


### LOGISTIC/SIGMOID FUNCTION:

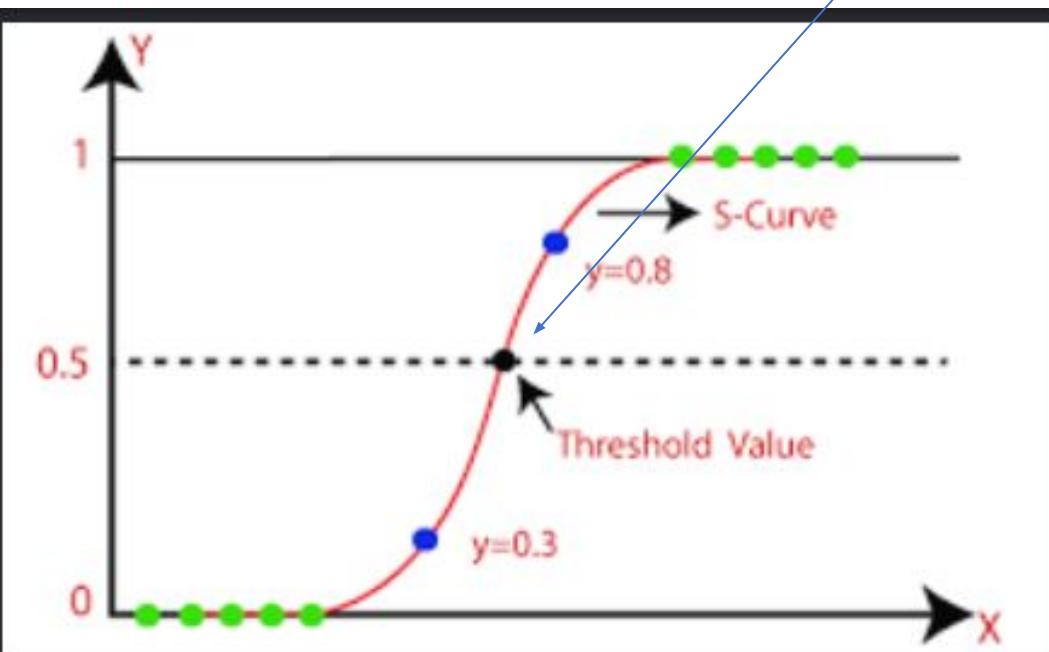
The logistic/sigmoid function is a mathematical function used to **map the predicted values to probabilities**. It maps any real value into another value **within a range of 0 and 1**, forms a curve like the S form.

In logistic regression, we use the concept of the **threshold value**, which defines the **probability of either 0 or 1**.

Such as values **above the threshold value tends to 1**, and a value **below the threshold values tends to 0**.



On the basis of threshold value it is decided whether the probability will be 0 or 1



- For mapping the predicted values to probabilities, the logistic/sigmoid function is used.
- It maps any real value into another value within a range of 0 and 1, forms a curve like the S form.

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

where  $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n)}}$$

$$\sigma(z) \rightarrow 1 \text{ as } z \rightarrow \infty$$

$$\sigma(z) \rightarrow 0 \text{ as } z \rightarrow -\infty$$

$$P(y = 1) = \sigma(z)$$

$$P(y = 0) = 1 - \sigma(z)$$

## Derivative of Sigmoid

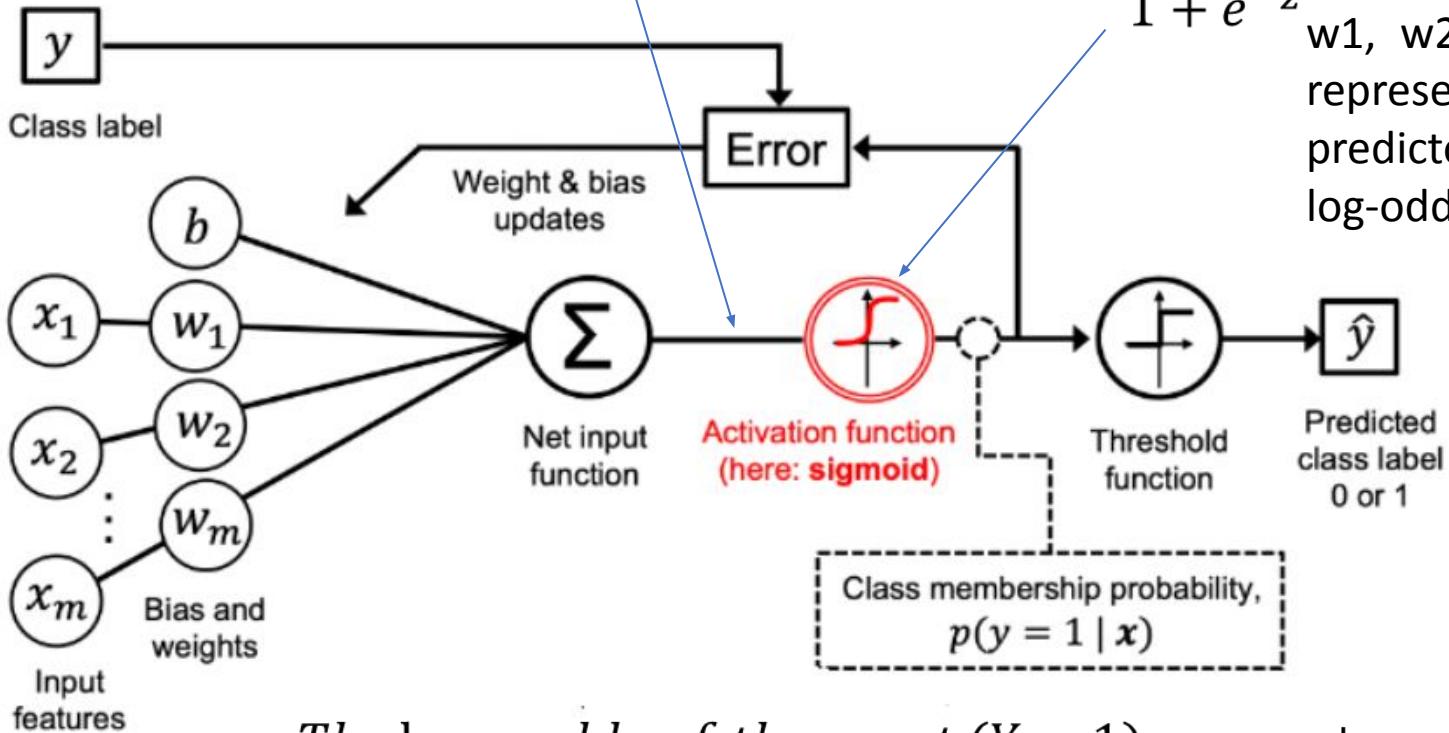
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned}
 \frac{d}{dx} \sigma(x) &= \frac{d}{dx} \frac{1}{1+e^{-x}} \\
 &= \frac{[(1+e^{-x}) * \boxed{\frac{d(1)}{dx}}] - [1 * \boxed{\frac{d(1+e^{-x})}{dx}}]}{(1+e^{-x})^2} \\
 &= \frac{[(1+e^{-x}) * \boxed{0}] - [1 * \boxed{-e^{-x}}]}{(1+e^{-x})^2} \\
 &= \frac{e^{-x}}{(1+e^{-x})^2} = \frac{e^{-x} + 1 - 1}{(1+e^{-x})^2} \\
 &= \frac{e^{-x} + 1}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} = \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} \\
 &= \frac{1}{(1+e^{-x})} \left( 1 - \frac{1}{(1+e^{-x})} \right) = \sigma(x) * (1 - \sigma(x))
 \end{aligned}$$

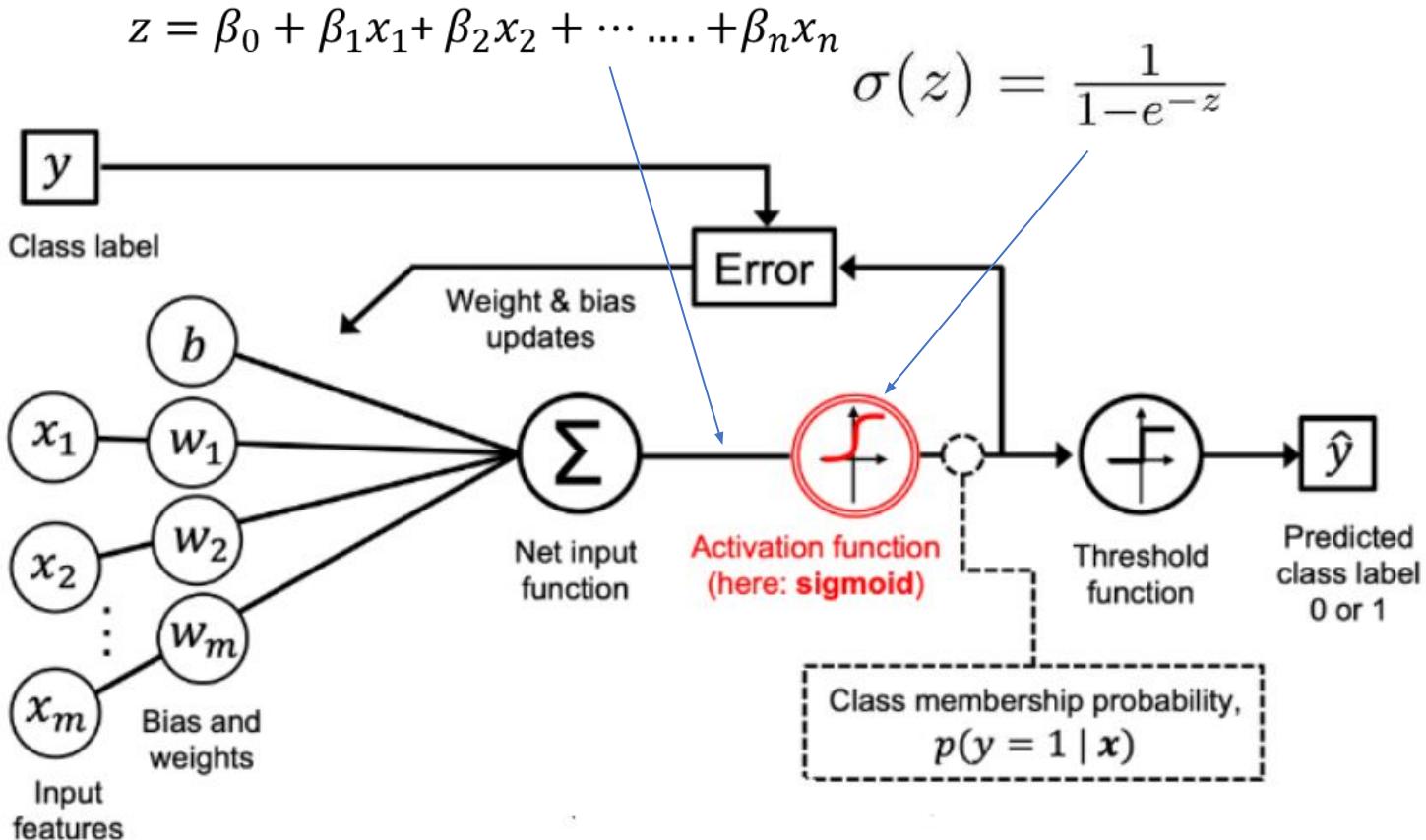
$$z = b + w_1x_1 + w_2x_2 + \dots + w_nx_n$$

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

w1, w2, ... etc. are the coefficients that represent the influence/wt. of each predictor variable( $x_1, x_2\dots$ ) on the log-odds of the event  $Y = 1$ .



*The log – odds of the event ( $Y = 1$ ) =  $w_0 + w_1x_1 + w_2x_2 + \dots + w_n x_n$*   
The sigmoid function transforms the log-odds into a probability value between 0 and 1.



The optimal coefficients  $\beta_0, \beta_1, \beta_2, \dots$  etc. in logistic regression are Maximum Likelihood Estimation (MLE) or its gradient-based variant, Gradient Descent.

1. Load the dataset. Say m training examples and n features.
2. Initialize the parameters/weights/coefficients. Initialize  $\beta_i$  with zeros or very small values.
3. Consider hypothesis function(Sigmoid function) for predicting probabilities. From the training dataset, each datapoint's probability is computed to predict true class.

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where  $z = b + w_1x_1 + w_2x_2 + \dots + w_nx_n$

OR

$$= \beta_0 + \beta_1x_1 + \beta_2x_2 + \dots + \beta_nx_n$$

$\beta_0, \beta_1, \beta_2, \dots$  etc. are the coefficients that represent the influence of each predictor variable on the log-odds of the event  $Y = 1$ .

$x_1, x_2, x_3, \dots$  are the predictor variables.

### Maximum Likelihood Estimation(MLE):

The method used to estimate the coefficients of the logistic regression model which maximizes the likelihood of the observed data given the model.

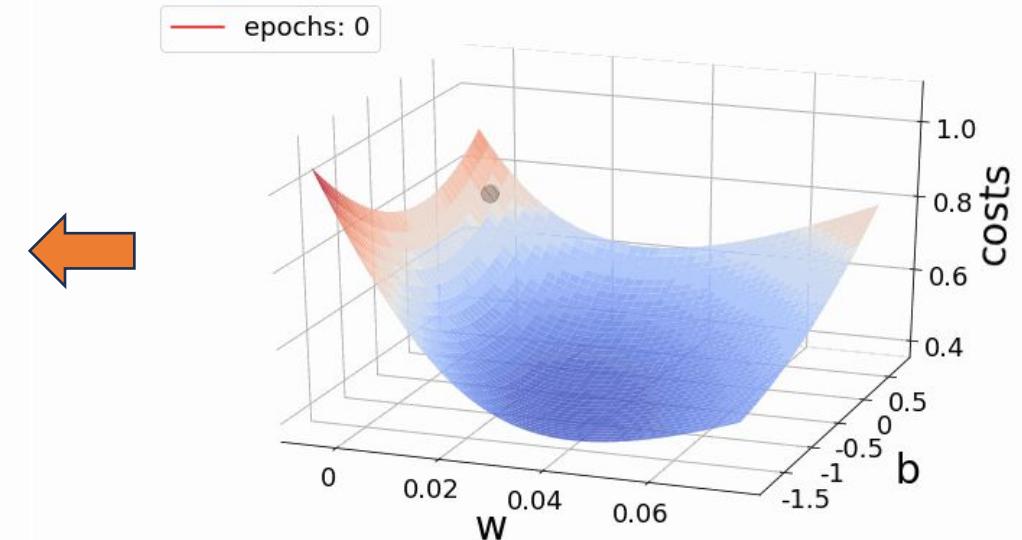
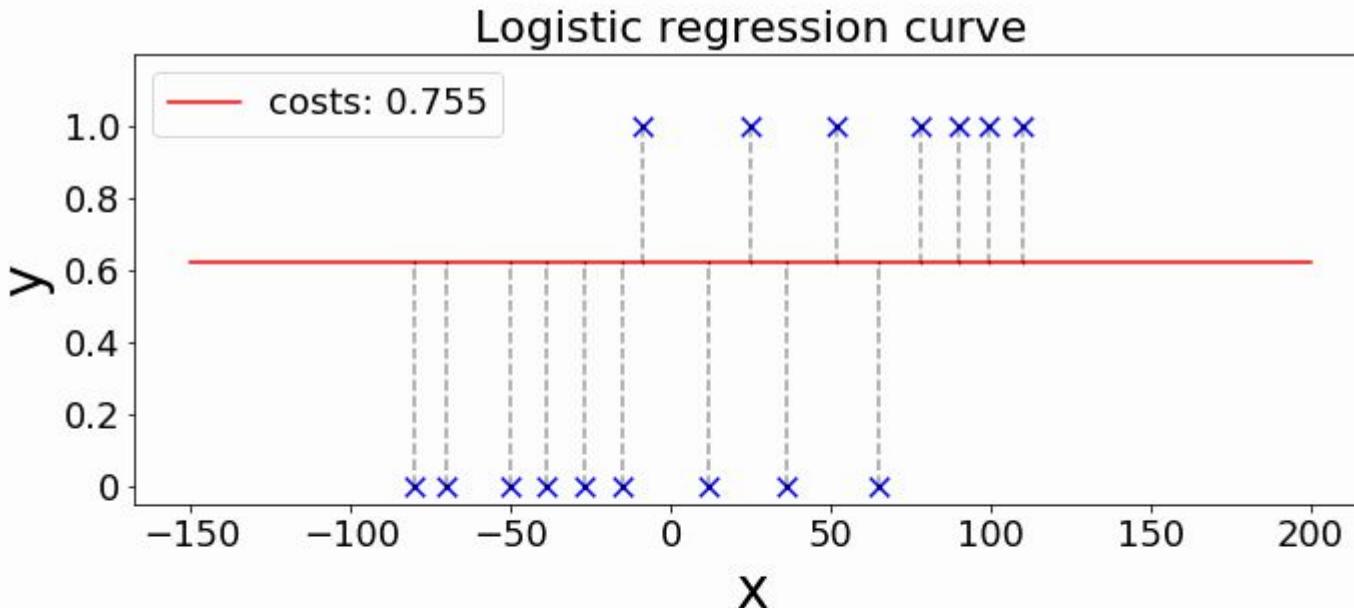
To find the coefficients  $\beta_0, \beta_1, \beta_2, \dots$  etc. in logistic regression, an optimization algorithm is used to minimize the difference between the predicted probabilities and the actual class labels in the training data.

The most commonly used algorithm is the **Maximum Likelihood Estimation (MLE)** or it's gradient-based variant, **Gradient Descent**.

#### 4. Define the cost function/Loss function/Error function

For logistic regression, the cost function is **negative log likelihood**

$$E(\vec{w}) = -\frac{1}{m} \sum_{i=1}^n y_i \log(\sigma(z)) + (1 - y_i) \log(1 - \sigma(z))$$



$$\text{Analysis of } -\sum_{i=1}^n y_i \log(\sigma(z)) + (1 - y_i) \log(1 - \sigma(z))$$

---

- This expression *quantifies the difference between the predicted probabilities* (given by the sigmoid function) and *the actual labels* in the training data.
- $y$  is the actual binary target label (0 or 1) for the training example.
- $\sigma(z)$  is the predicted probability of the positive class (class 1) according to the logistic regression model.
- $1-\sigma(z)$  is the predicted probability of the negative class (class 0).

### Analysis of $-\sum_{i=1}^n y_i \log(\sigma(z)) + (1 - y_i) \log(1 - \sigma(z))$

- The idea behind the expression is to *penalize the model more if it predicts a different probability than the actual label.*
- If  $y$  is 1 (positive class), the first term  $y_i \log(\sigma(z))$  encourages the model to predict a high probability for class 1 and the **log** ensures that the penalty increases as the predicted probability deviates from 1.
- The second term  $(1 - y_i)(1 - \sigma(z))$  encourages the model to predict a low probability for class 0 ( $h\theta(x)$  close to 0) when  $y$  is 0 (negative class).
- The ***negative log-likelihood*** combines these two terms to form a single expression that captures the likelihood of the model's predictions given the actual labels. The overall cost function is the average (or sum, depending on conventions) of these negative log-likelihood terms over all training examples.

## 5. Update the coefficients using Gradient descent and learning rate $\alpha$

The gradient descent update rule for logistic regression is

$$w_i \leftarrow w_i - \alpha \frac{\partial E(\vec{w})}{\partial w_i}$$

The learning rate determines the step size in each iteration and should be chosen carefully if not given.

Assume that there is only one input feature in the given dataset, hence,  $z = mx + b$ .

Now we need to minimize loss w.r.t m and b.

Therefore, find  $\frac{\partial L}{\partial m}$  and  $\frac{\partial L}{\partial b}$ . These terms are required to update the parameters m and b using Gradient Descent.

Repeat Steps 3 to 5 until the coefficients converge.

**UE22CS352A - Machine Learning**

**(Derivative)Loss function in Logistic Regression**

---



**6. Convergence and Termination:** Monitor the changes in the coefficients over iterations. If the change becomes small (i.e the coefficients are not changing significantly), the optimization has converged, and you can terminate the process.

**7. Final coefficient values:** The final converged values of  $\beta_0$  and  $\beta_1$  are the estimated coefficients for your logistic regression model.

You are a data scientist working for a hospital. The hospital wants to predict whether patients are likely to be readmitted within 30 days after discharge. You decide to use logistic regression for this task.

Patient ID	Age	BMI	Blood Pressure	Glucose Level	Days in Hospital	Number of Previous Admissions	Readmitted (1: Yes, 0: No)
1	65	28	140	105	5	2	1
2	50	24	130	110	3	1	0
3	70	30	150	115	7	3	1
4	55	22	125	100	4	0	0
5	60	26	135	108	6	2	1

The logistic regression model has the following weights (coefficients):

$$\begin{aligned}
 \text{logit}(p) = & -3 + 0.04 \cdot \text{Age} + 0.1 \cdot \text{BMI} + 0.03 \cdot \text{Blood Pressure} + 0.02 \cdot \text{Glucose Level} + 0.05 \cdot \text{Days in hospital} \\
 & + 0.2 \cdot \text{Number of Previous Admissions}
 \end{aligned}$$

Consider this new subset of patients for which you need to predict the likelihood of admission:

Patient ID	Age	BMI	Blood Pressure	Glucose Level	Days in Hospital	Number of Previous Admissions
6	45	27	138	102	4	1
7	62	25	132	106	5	2

Using the provided logistic regression model, calculate the probability of readmission for Patient 6 and Patient 7.

### Solution:

Calculate the logit value for each patient:

For Patient 6:

1. Age = 45
2. BMI = 27
3. Blood Pressure = 138
4. Glucose Level = 102
5. Days in Hospital = 4
6. Number of Previous Admissions = 1

$$\text{logit}(6) = -3 + (0.04 \times 45) + (0.1 \times 27) + (0.03 \times 138) + (0.02 \times 102) + (0.05 \times 4) + (0.2 \times 1)$$

$$\text{logit}(6) = -3 + 1.8 + 2.7 + 4.14 + 2.04 + 0.2 + 0.2$$

$$\text{logit}(6) = 8.08$$

# UE22CS352A - Machine Learning

## Logistic Regression

---

Probability of readmission:

$$P_6 = \frac{1}{1 + e^{-8.08}} \approx 0.9996$$

Similarly calculate for Patient 7.

$$P_7 \approx 0.9998$$

Consider a dataset where you want to predict whether a restaurant will receive a high rating based on its location and the quality of its food. We have the following dataset:

Restaurant	Location (1-5)	Food Quality (1-3)	High Rating (Label)
1	3	2	0
2	4	3	1
3	2	1	0
4	5	2	1
5	4	2	1

Using logistic regression, predict whether a restaurant with a location rating of 2 and food quality rating of 2 will receive a high rating or not. Consider the values of parameter  $w_0 = -3$ ,  $w_1 = 0.4$ ,  $w_2 = 2$

Consider a simplified dataset where we want to predict whether a student will pass a math exam based on the number of hours they studied. We have the following dataset. **Using logistic regression, predict whether a student who studies for 4.5 hours will pass the exam or not.**

Hours Studied (X)	Passed (Y)
2	0
3	0
4	0
5	1
6	1
7	1

**SOLUTION:**

We will use the logistic regression equation,

$$P(Y = 1|X) = \frac{1}{1+e^{-(\beta_0 + \beta_1 X)}}$$

For simplicity, let's assume we've already estimated the coefficients as,  
 **$\beta_0 = -6$  and  $\beta_1 = 1.2$** . Substituting the values, we get,

$$\begin{aligned} P(Y = 1|X = 4.5) &= \frac{1}{1+e^{-(-6+1.2\times4.5)}} \\ P(Y = 1|X = 4.5) &= \frac{1}{1+e^{-0.3}} \\ P(Y = 1|X = 4.5) &\approx 0.4256 \end{aligned}$$

So, the predicted probability of passing the exam after studying for 4.5 hours is approximately **0.4256**. Assuming 0.5 as the threshold value, given the probability, the student **will not pass the exam (class is 0 since  $\sigma(4.5) \sim 0.4256 < 0.5$ )**

## Logistic Regression

---

### Loan Default Prediction

You work for a bank and want to predict whether a loan applicant will default on their loan (1) or not (0) based on two attributes: credit score and annual income.

Applicant	Credit Score	Annual Income	Default (Label)
1	650	\$50,000	0
2	720	\$70,000	0
3	580	\$40,000	1
4	800	\$100,000	0
5	500	\$30,000	1
6	670	\$60,000	0
7	600	\$45,000	1

### Step 1: Sigmoid Function

The sigmoid function is used to map the linear combination of attributes and weights to a probability value between 0 and 1.

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

Where  $z$  is the linear combination of the input features and their corresponding weights.

$$z = w_0 + w_1 * \text{credit\_score} + w_2 * \text{annual\_income}$$

### Cost Function

The cost function for logistic regression is the negative log-likelihood (cross-entropy) loss function:

$$E(\vec{w}) = -\frac{1}{n} \sum_{i=1}^n y_i \log(\sigma(z)) + (1 - y_i) \log(1 - \sigma(z))$$

Where n is the number of training examples, y is the actual output (0 or 1), and  $\sigma(z)$  is the predicted probability of the positive class.

### Gradient Descent

Gradient descent is used to find the optimal weights that minimize the cost function.

$$w \leftarrow w - \alpha * \partial J(w) / \partial w$$

### Iterative Updates:

Iterate the gradient descent process to update the weights until convergence or a predefined number of iterations.

Once the weights are learned, you can use them to predict whether a new loan applicant will default or not based on their credit score and annual income

## Advantages and Disadvantages of Logistic Regression

---

### ADVANTAGES OF LOGISTIC REGRESSION:

- Logistic regression performs better when the **data is linearly separable**
- It **does not require too many computational resources** as it's highly interpretable
- There is no problem scaling the input features—**It does not require tuning**
- It is **easy to implement and train** a model using logistic regression
- It gives a measure of how **relevant** a predictor (coefficient size) is, and its **direction of association** (positive or negative)

### DISADVANTAGES OF LOGISTIC REGRESSION:

- Logistic regression **fails to predict a continuous outcome.**
- Logistic regression **assumes linearity** between the predicted (dependent) variable and the predictor (independent) variables.
- Logistic regression **may not be accurate if the sample size is too small.**

## Assumptions of Logistic Regression

---

### ASSUMPTIONS OF LOGISTIC REGRESSION:

- **Independent observations:** Each observation is independent of the other, meaning there is **no correlation** between any input variables.
- **Binary dependent variables:** It takes the assumption that the **dependent variable must be binary or dichotomous**, meaning it can take only two values. For more than two categories softmax functions are used.
- **Linearity relationship between independent variables and log odds:** The relationship between the independent variables and the log odds of the dependent variable should be linear.
- **No outliers:** There should be no outliers in the dataset.
- **Large sample size:** The sample size is sufficiently large.

# MACHINE LEARNING (UE22CS352A)

---



**Instance based learning**

---

**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

## Instance-based Learning

---

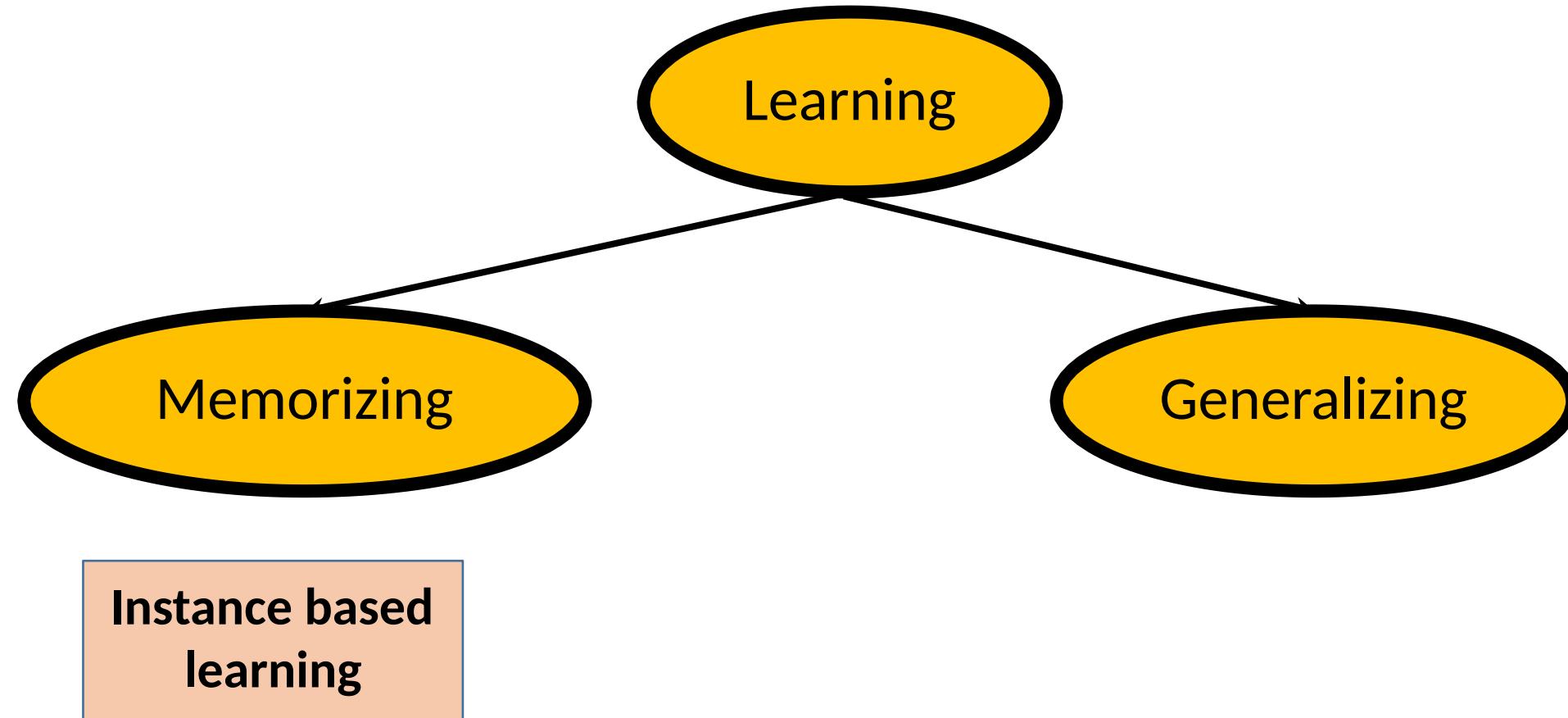
- **Instance-based learning** is a machine learning technique in which the model learns the training examples **by heart and then generalizes** to new instances based on some similarity measure.
- It is called instance-based because it builds the hypotheses from the training instances.
- It is also known as **memory-based learning** or **lazy-learning** (because they delay processing until a new instance must be classified).
- The time complexity of this algorithm depends upon the **size of training data**.
- Worst-case time complexity of this algorithm is **O (n)**, where n is the number of training instances

## Instance-based Learning - Example

- For example, If we were to create a spam filter with an instance-based learning algorithm, instead of just flagging emails that are already marked as spam emails, our spam filter would be programmed to also flag emails that are very similar to them.
- This requires a measure of resemblance between two emails.
- A similarity measure between two emails could be the same sender or the repetitive use of the same keywords or something else.



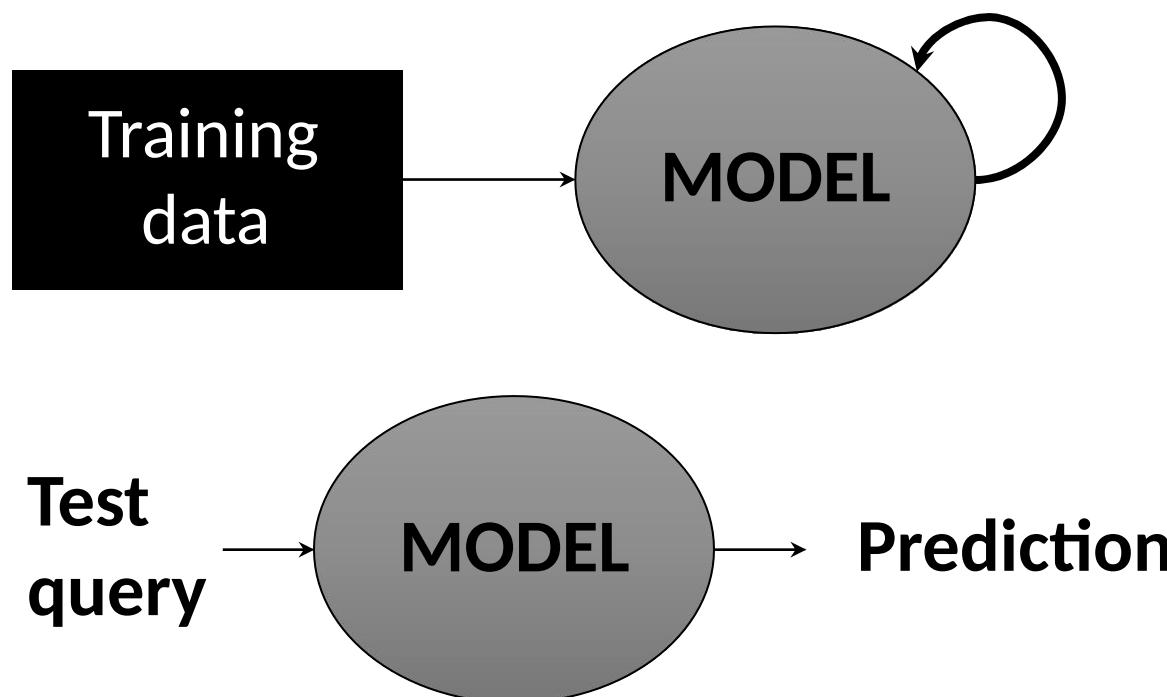
## Instance-based Learning - Example



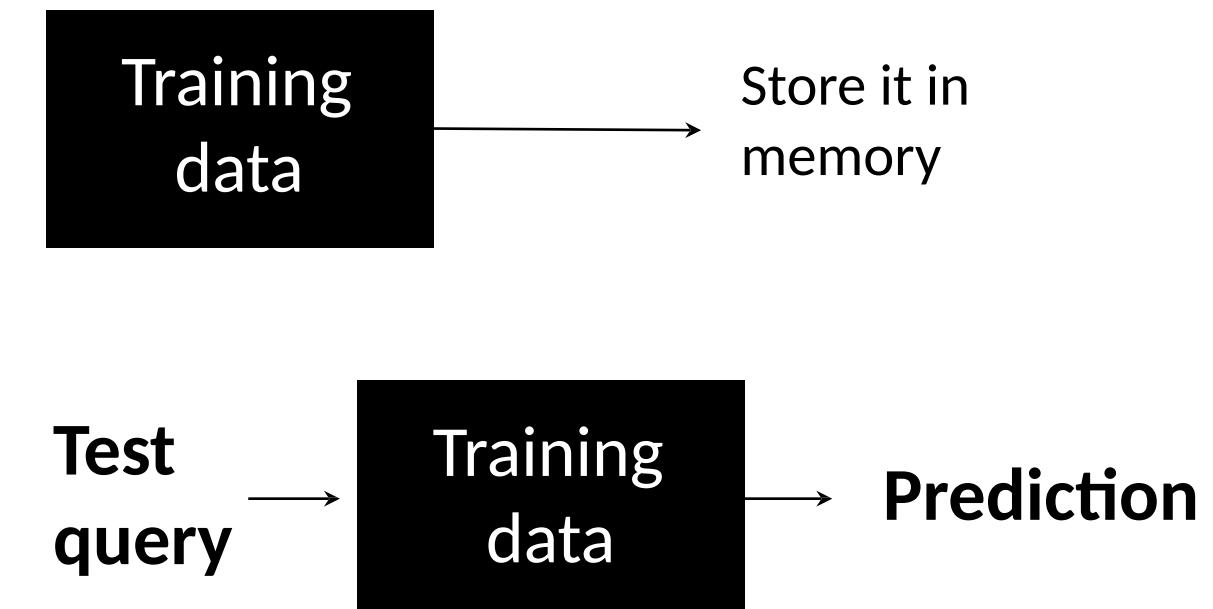
**Instance based learning**  
does Memorizing  
instead of Generalizing

## Eager Learning vs Lazy Learning

### EAGER LEARNING



### LAZY LEARNING



### EAGER LEARNING

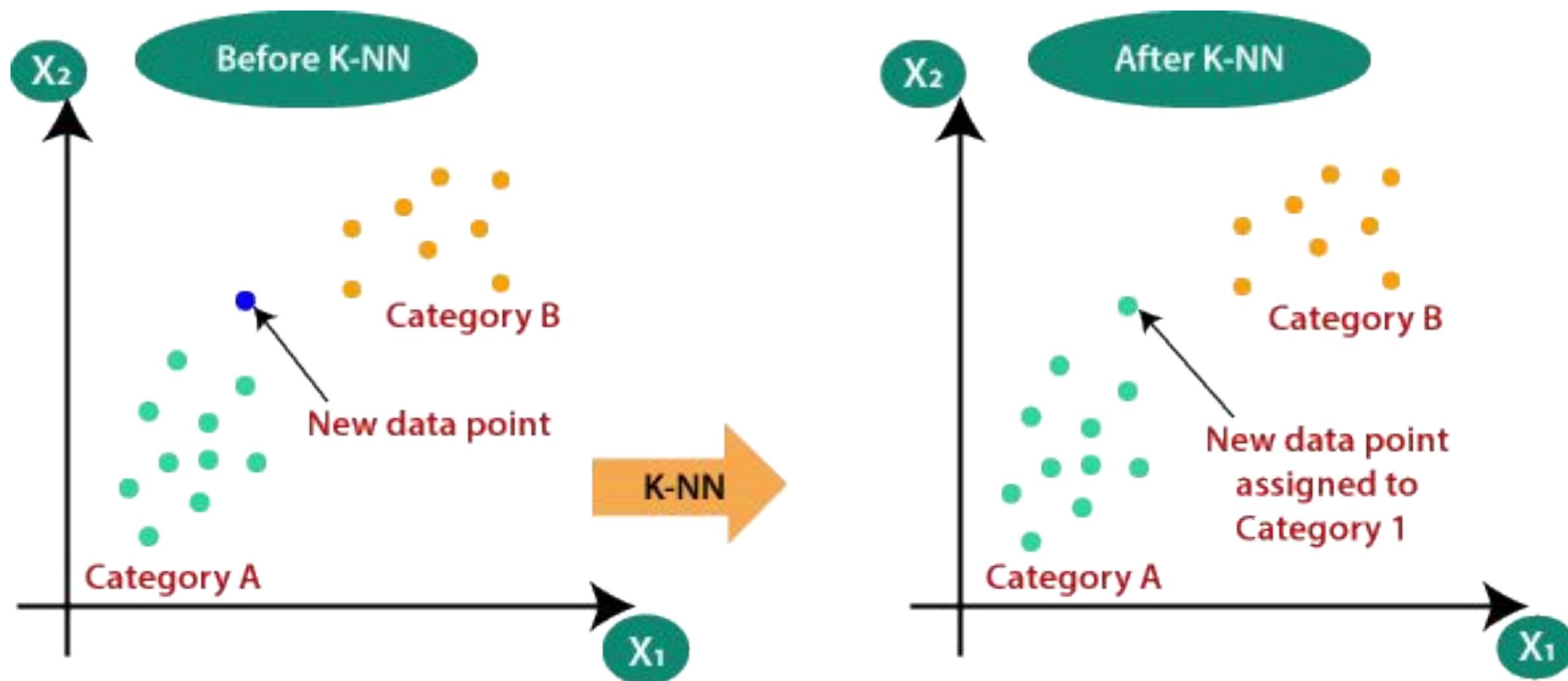
- Training data is used to create global model.
- A generalized model is created based on training set.
- On test query the global model is used to make the prediction.
- Training data is not directly used to make prediction , instead the generated model is used.

### LAZY LEARNING

- No global model is created
- For every test instance a common procedure is run on **training data** to find the prediction value.
- Prediction uses training data directly



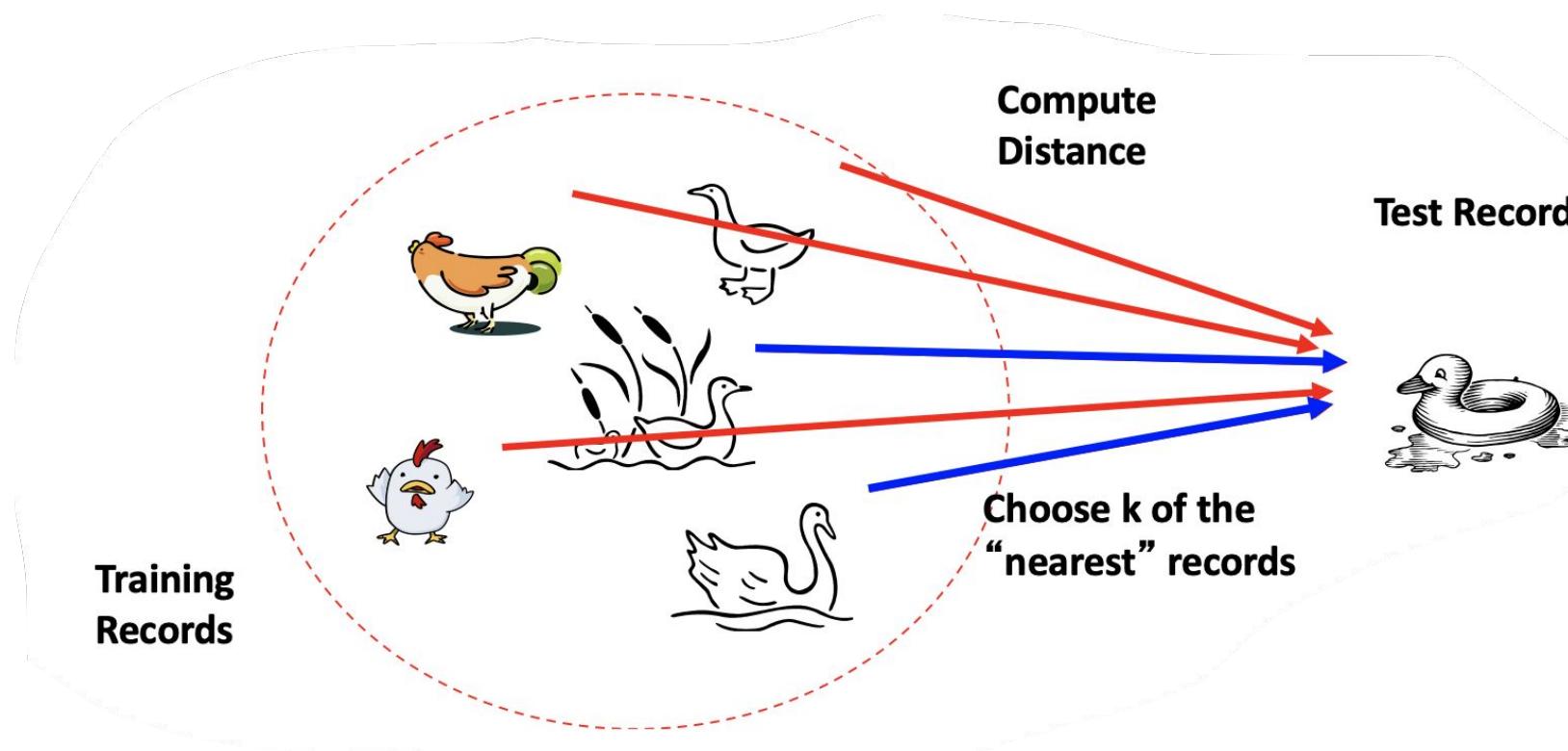
- K Nearest Neighbors (K-NN) is an **instance based, lazy-learner machine learning algorithm, based on supervised learning.**
- K-NN algorithm assumes the **similarity between the new case/data and available cases** and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- A **distance measure between the new data and existing data point is taken as a measure of similarity.**



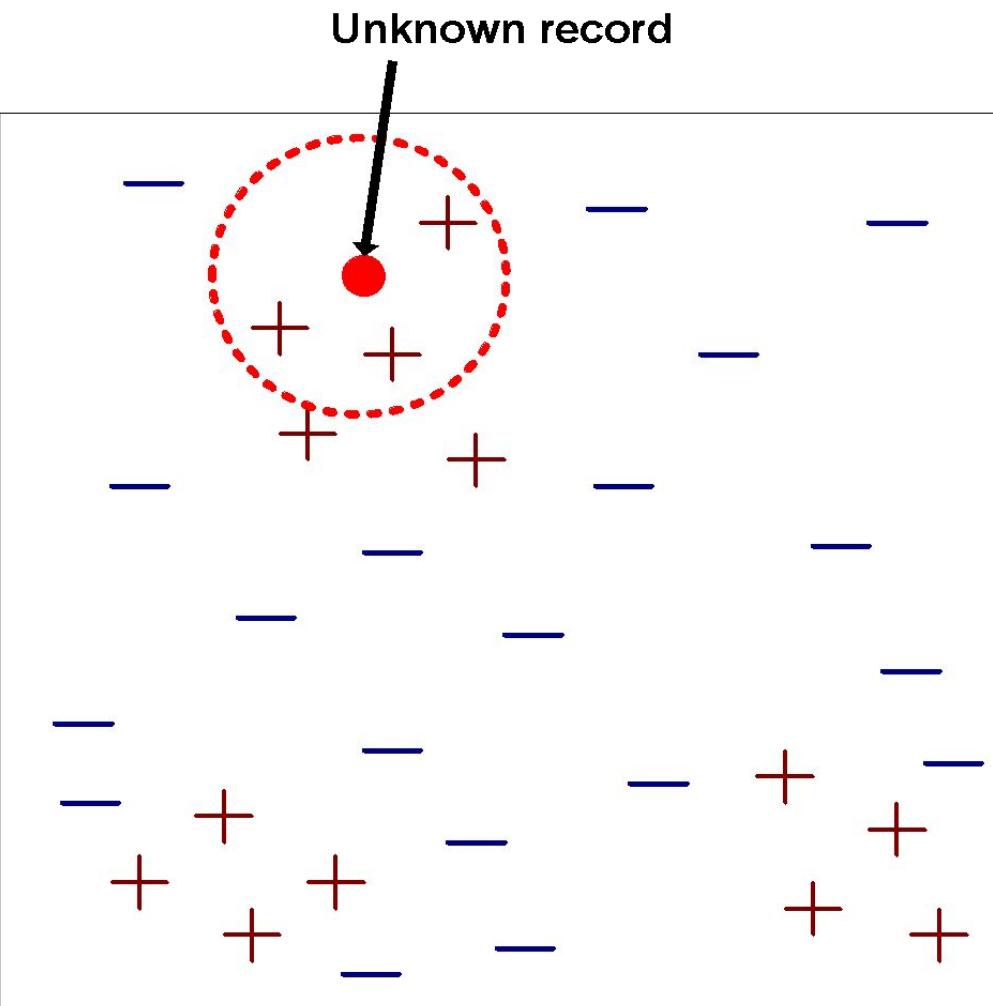
## K-NN Inductive Bias

Basic idea: The main idea or justification of nearest neighbor classifier is emphasized with the following example:

If it walks like a duck, quacks like a duck, looks like a duck, then it's probably a duck



## K-NN Inductive Bias



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of  $k$ , the number of nearest neighbors to retrieve.
- To classify an unknown record:
  - ❖ Compute distance to other training records
  - ❖ Identify  $k$  nearest neighbors
  - ❖ Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

## K-NN working algorithm

---

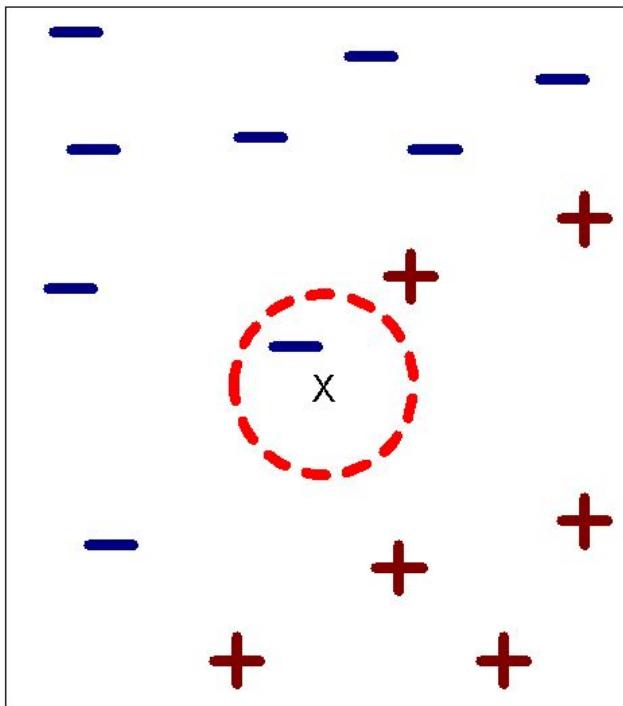
- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

## K-NN working algorithm

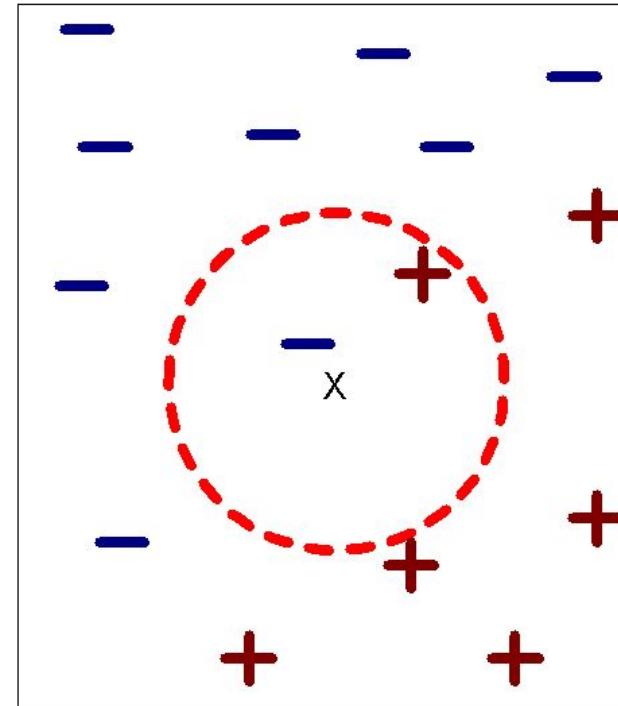
---

- All instances correspond to points in the n-D space.
- The nearest neighbour are defined in terms of distance measure,  $\text{dist}(X_1, X_2)$
- Target function could either be discrete or real valued.
- For **discrete-valued**, k-NN returns the most common value among the  $k$  training examples nearest to  $x_q$
- k-NN for **real-valued prediction** for a given unknown tuple returns the **mean values of the  $k$  nearest neighbors**

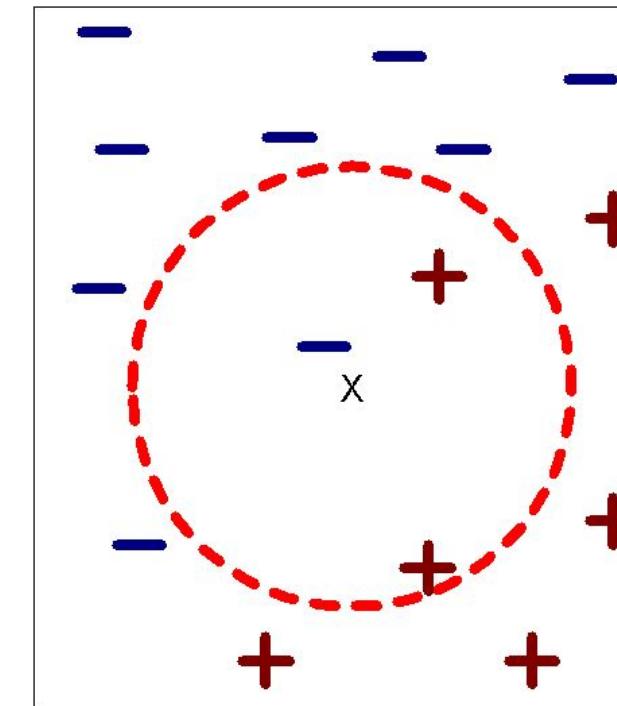
## K-NN working algorithm



(a) 1-nearest neighbor



(b) 2-nearest neighbor



(c) 3-nearest neighbor

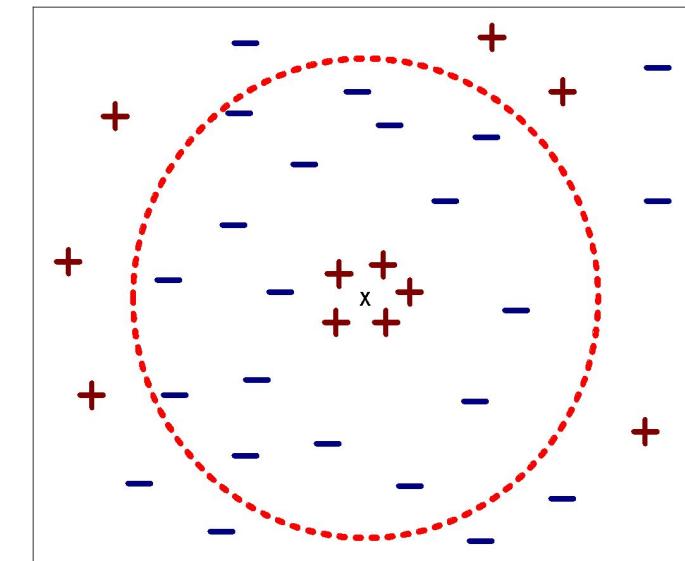
- K-nearest neighbors of a record  $x$  are data points that have the  $k$  smallest distance to  $x$ .

## K-NN working algorithm

### Choosing the value of k:

- ❖ If  $k$  is too small, sensitive to noise points and chances of overfitting. Small  $K$  captures structure of problem space better. May be necessary for small training set.
- ❖ If  $k$  is too large, neighborhood may include points from other classes and hence misclassify the test record. Less sensitive to noise(particularly for class noise). Gives better probability estimates for discrete classes. Large training set allows use of larger  $K$ .

K-nearest  
classification with  
large  $k$



- $K$  preferably be odd(ex.  $K=3,5,7,\dots$ ), Why??

## K-NN working algorithm

---

Consider learning discrete-valued target functions of the form  $f: \mathcal{R}^n \rightarrow V$ , where  $V$  is a finite set  $\{v_1, v_2, \dots, v_n\}$

- **Training algorithm:**

For each training example  $(x, f(x))$ , add the example to the list of training examples.

- **Classification algorithm:**

Given a query instance  $x_q$  to be classified

1. Let  $x_1 \dots x_k$  denote the  $k$  instances from training examples that are nearest to  $x_q$

2. Return

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$$

where  $\delta(a, b) = 1$  if  $a = b$  and where  $\delta(a, b) = 0$  otherwise

Note: *Argmax* is a mathematical function that finds the maximum value of an argument from a target function.

Consider a function approximator/target function as  $f(x)=x^2$ , then  $f(1)=1$ ,  $f(2)=4$ ,  $f(3)=9$ ,  $f(4)=16$

Then  $\operatorname{argmax} f(x)=4$  (value of the argument for which  $f(x)$  is max)

## K-NN working algorithm

---

- Argmax is an operation that finds the argument that gives the maximum value from a target function.
- Argmax is most commonly used in **Machine learning for finding the class with the largest predicted probability**.
- Argmax can be implemented manually, although the argmax() Numpy function is preferred in practice.

## K-NN working algorithm

Consider the given plot of the data set with two classes, this completes the training algorithm

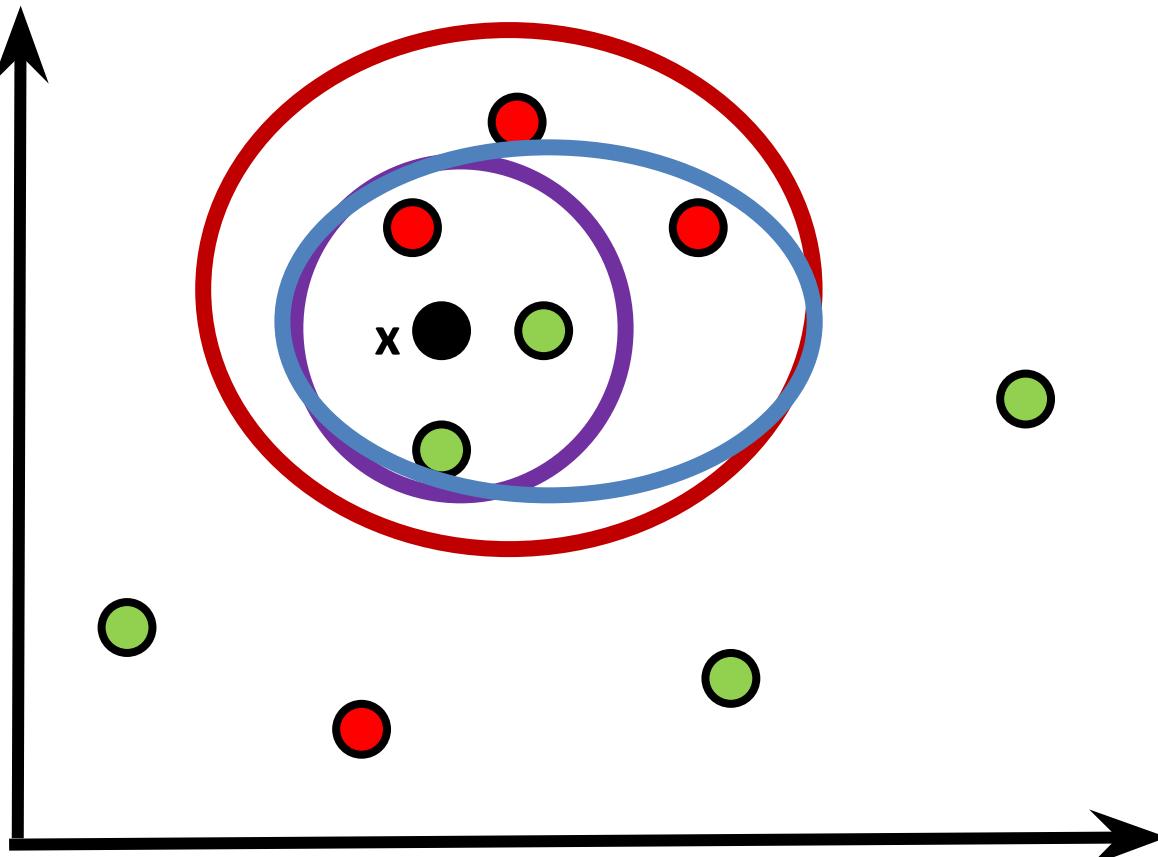
consider  $x$  to be the query point

with  $k=3$  the query is classified as green class

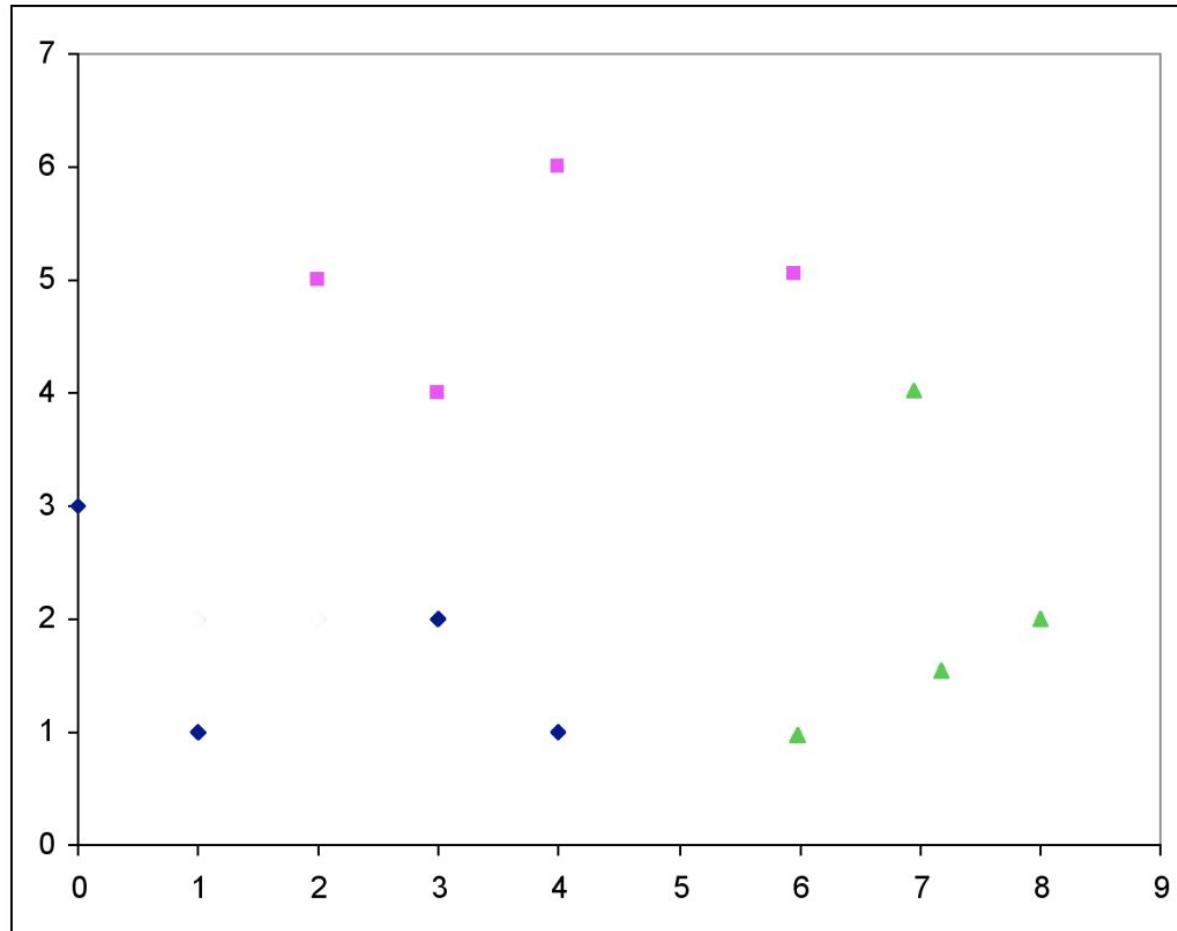
with  $k=5$  the query is classified as red class

we previously said  $K$  should be odd, lets see why

with  $k=4$  the query point can't be classified since it has equal neighbour of both the class



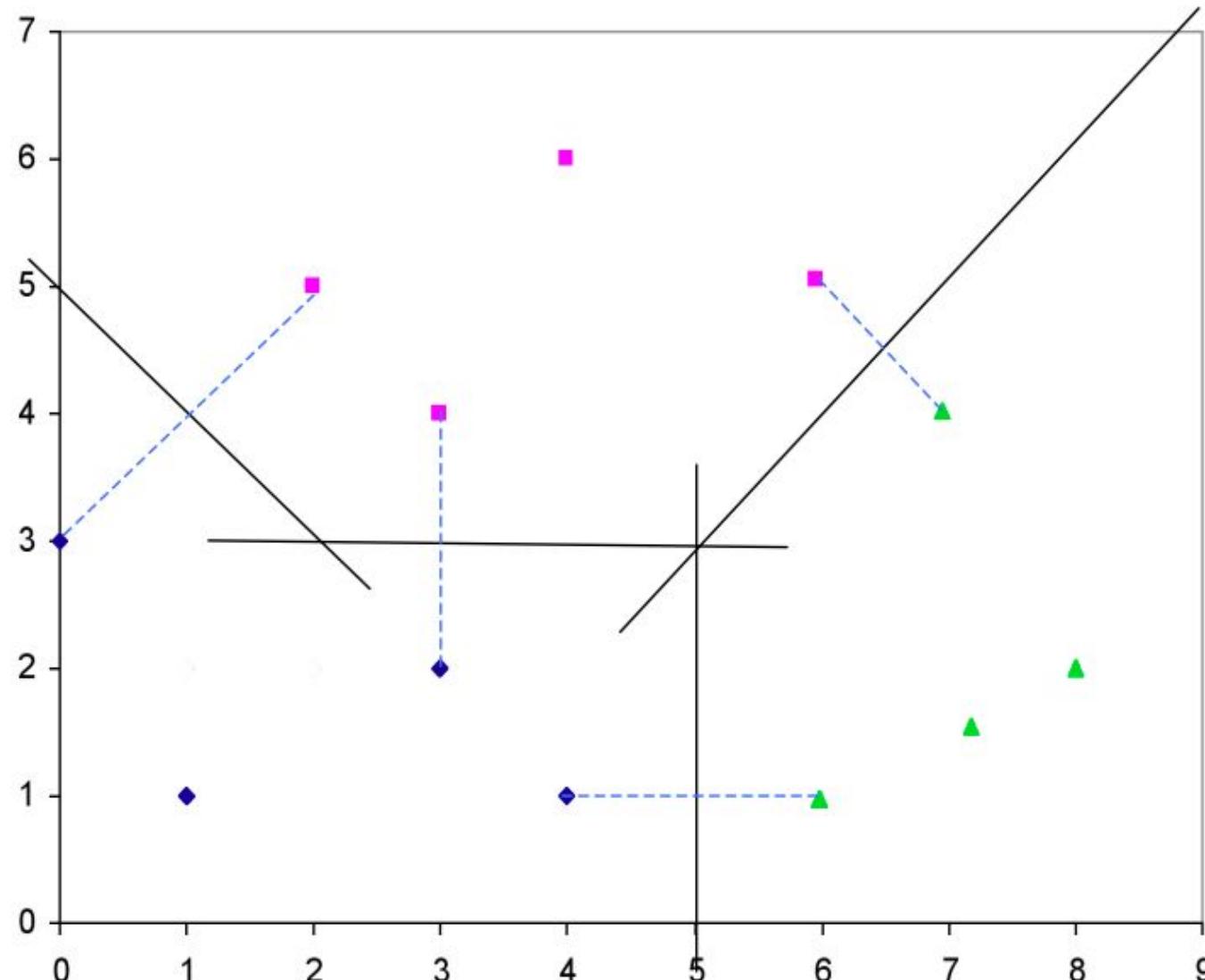
## K-NN working algorithm



1. Boundary lines are formed by the intersection of perpendicular bisectors of every pair of points.
2. Using pairs of closest points in different classes gives a good enough approximation.

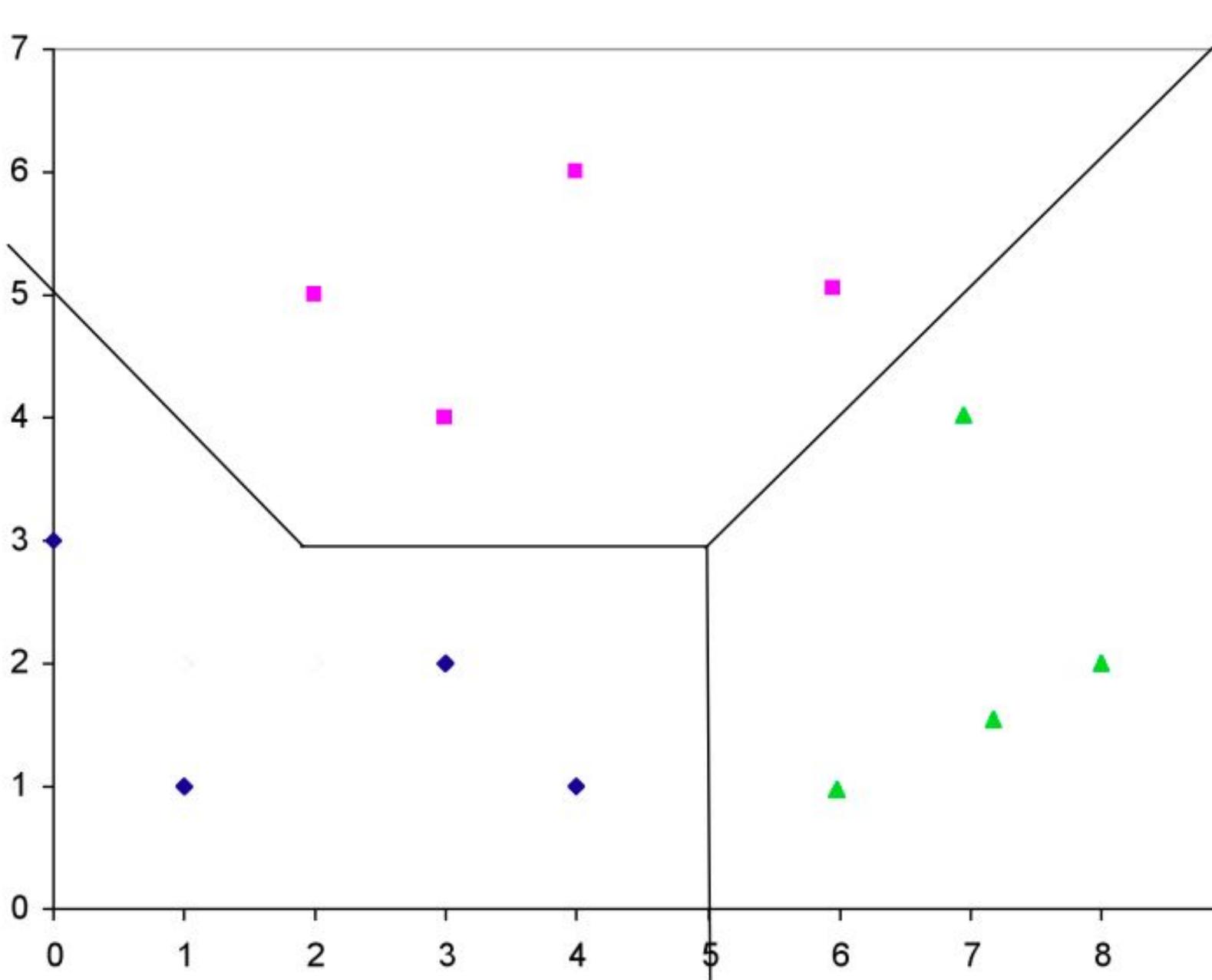
(To be absolutely sure about the boundaries, **we can draw perpendicular bisectors between each pair of neighboring points** to create a region for each point, then consolidate regions belonging to the same class, i.e., remove the boundaries separating points in the same class.)

## K-NN working algorithm



In order to draw Decision boundary for KNN, draw the perpendicular bisector between the closest pair of data objects of different classes.

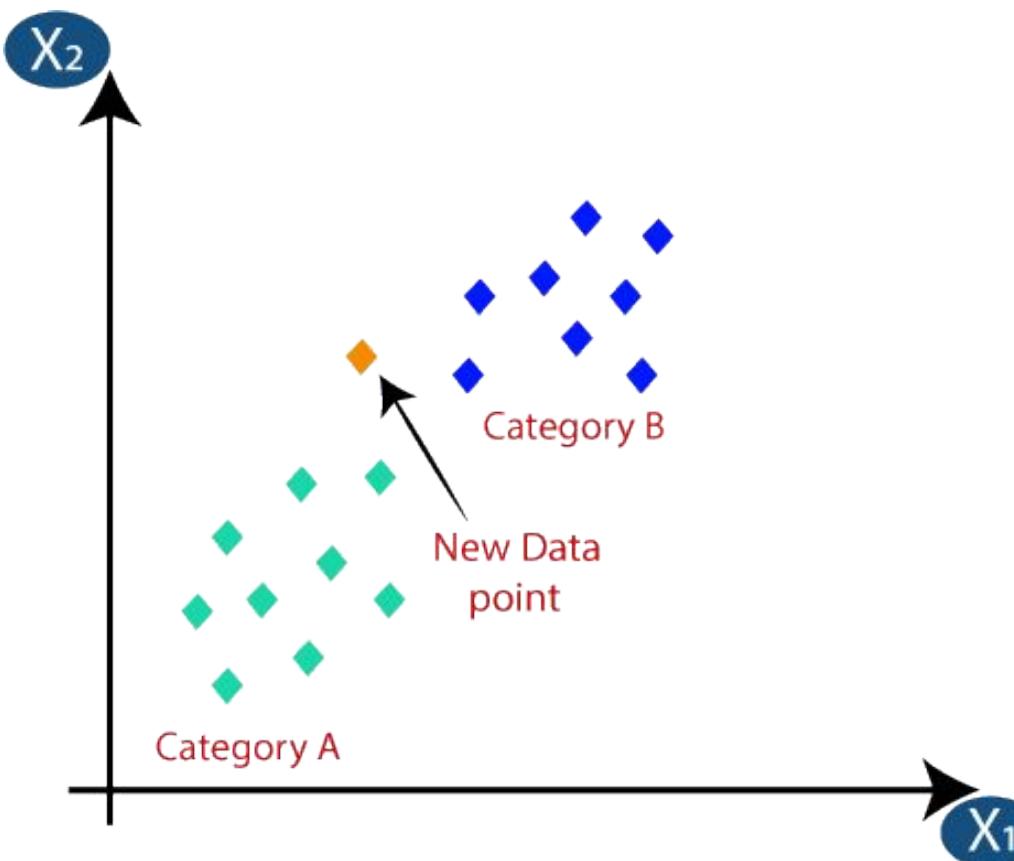
## K-NN working algorithm



The figure on the previous slide can be viewed like this figure and the lines separating the different classes defines the decision boundary for the 3 classes.

## K-NN working algorithm

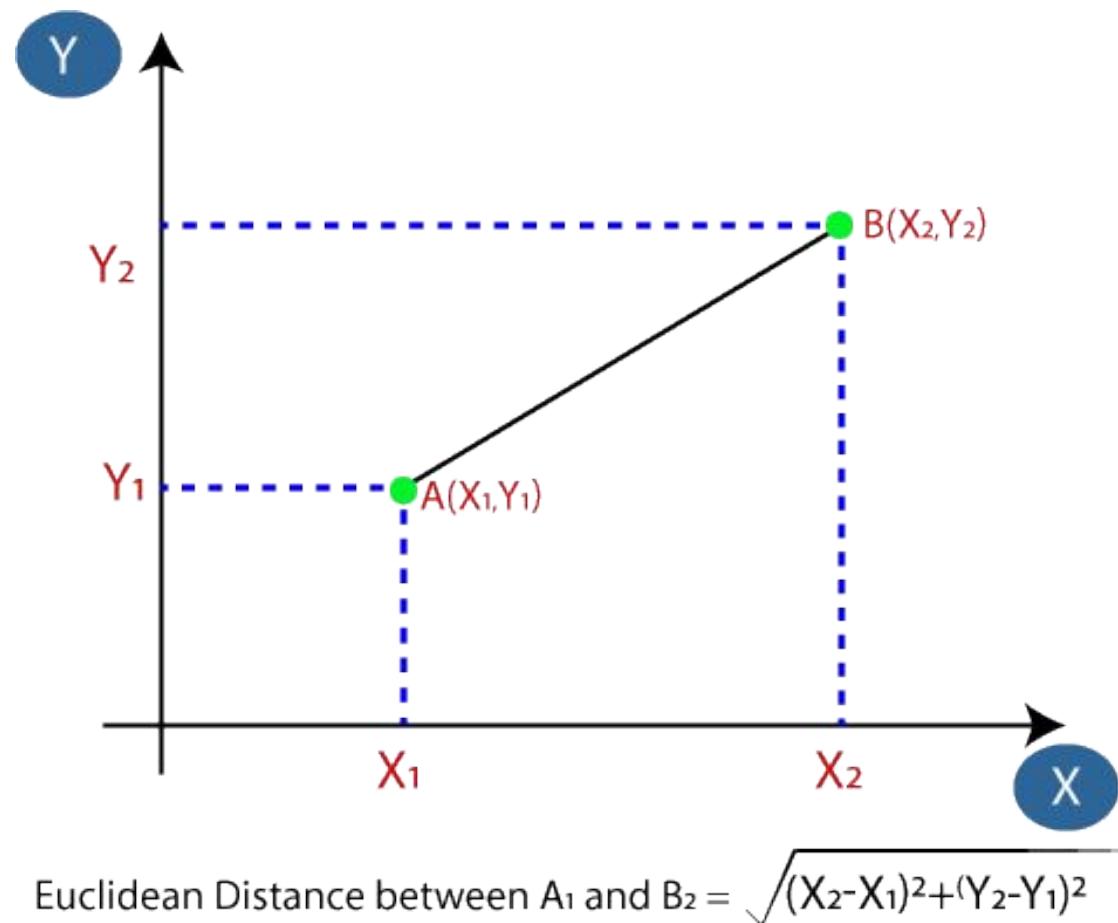
Suppose we have a new data point and we need to put it in the required category.  
Consider the below image:



## K-NN working algorithm

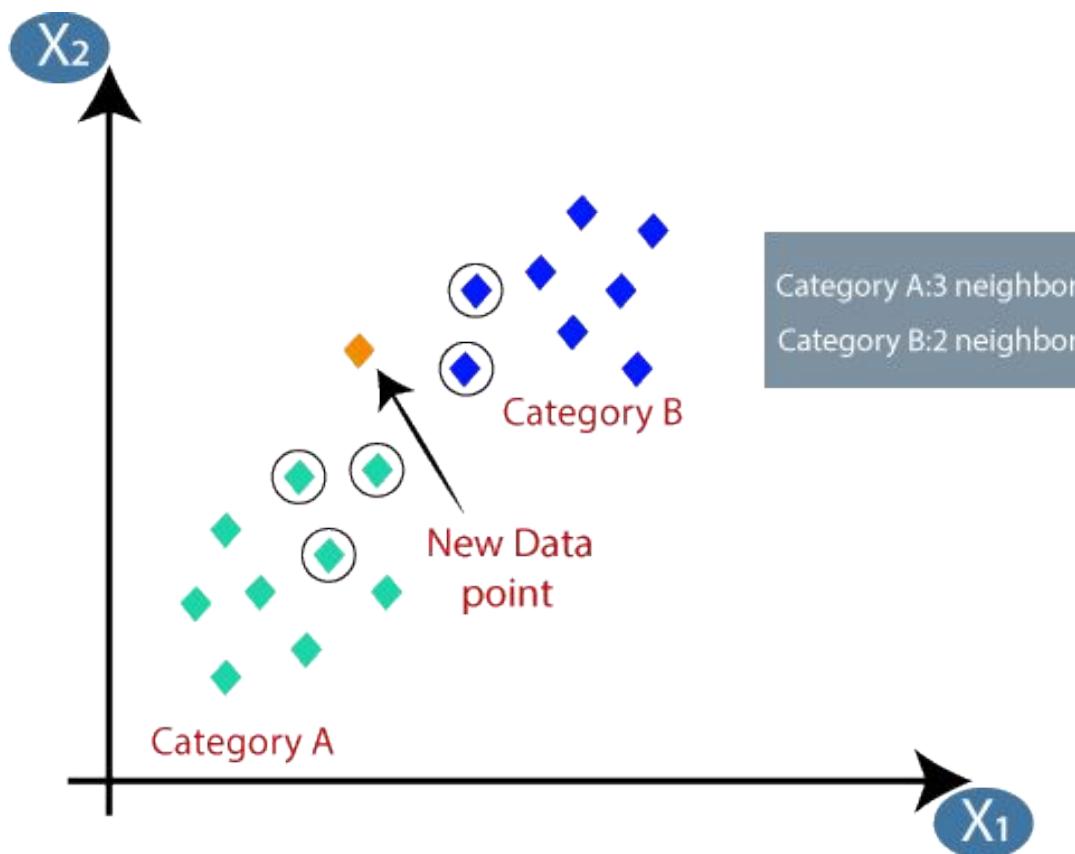
Firstly, we will choose the number of neighbors, so we will choose the k=5.

Next, we will calculate the **Euclidean distance** between the data points.  
It can be calculated as:



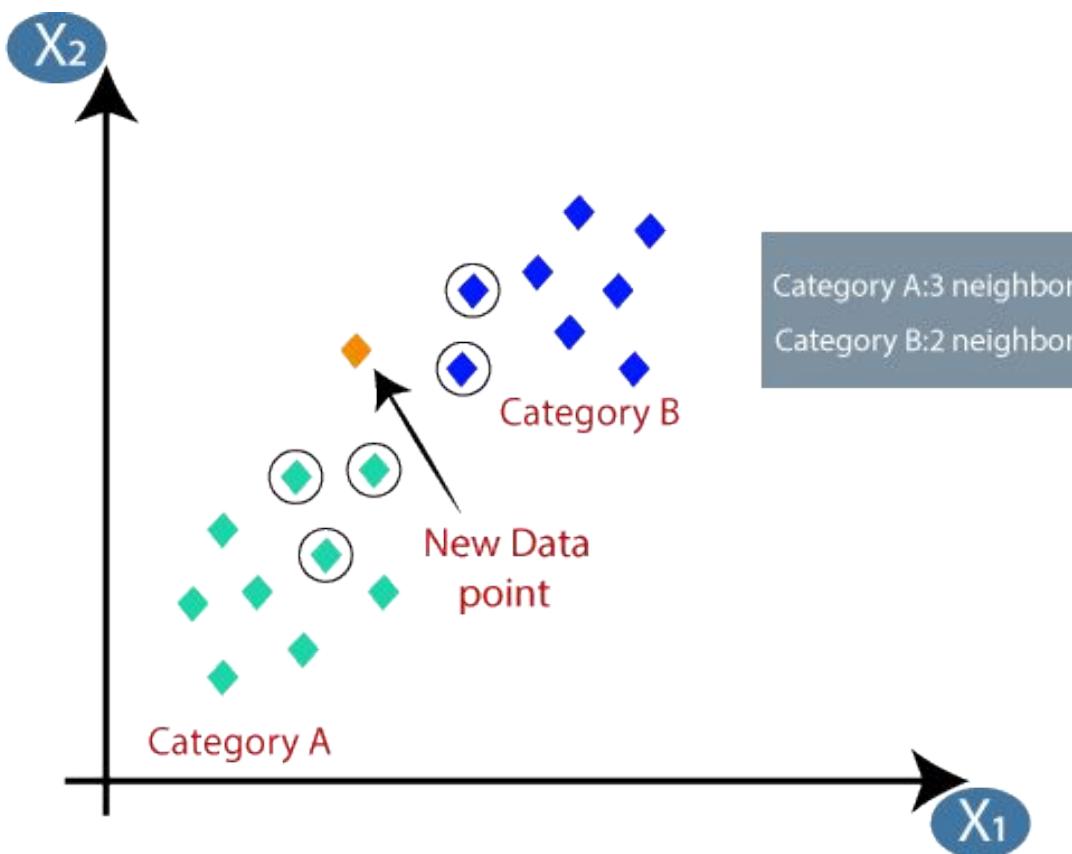
## K-NN working algorithm

By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



## K-NN working algorithm

As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.



In K-NN, every data instance is assumed to be a point in d-dimensional space ( $R^d$ ) where d is the number of features. K-NN can be used for both classification and regression tasks.

### K-NN Classification

- In classification, the target function is discrete valued(m classes).
- Each of the data instances may belong to **one** of the **m classes**.
- We define Set V as the set of all classes,  $V = \{v_1, v_2, v_3, \dots, v_m\}$ .
- We can define the target function as  $f : R^d \rightarrow V$
- Any data instance  $x_i$  will have a label  $f(x_i)$  (- one of the class from set V).

## K-NN Applications

---

In K-NN, every data instance is assumed to be a point in d-dimensional space ( $R^d$ ) where d is the number of features. K-NN can be used for both classification and regression tasks.

### K-NN Regression

- In regression, the target function is real valued(continuous value).
- We can define the target function as  $f : R^d \rightarrow R$

### Query Instance representation

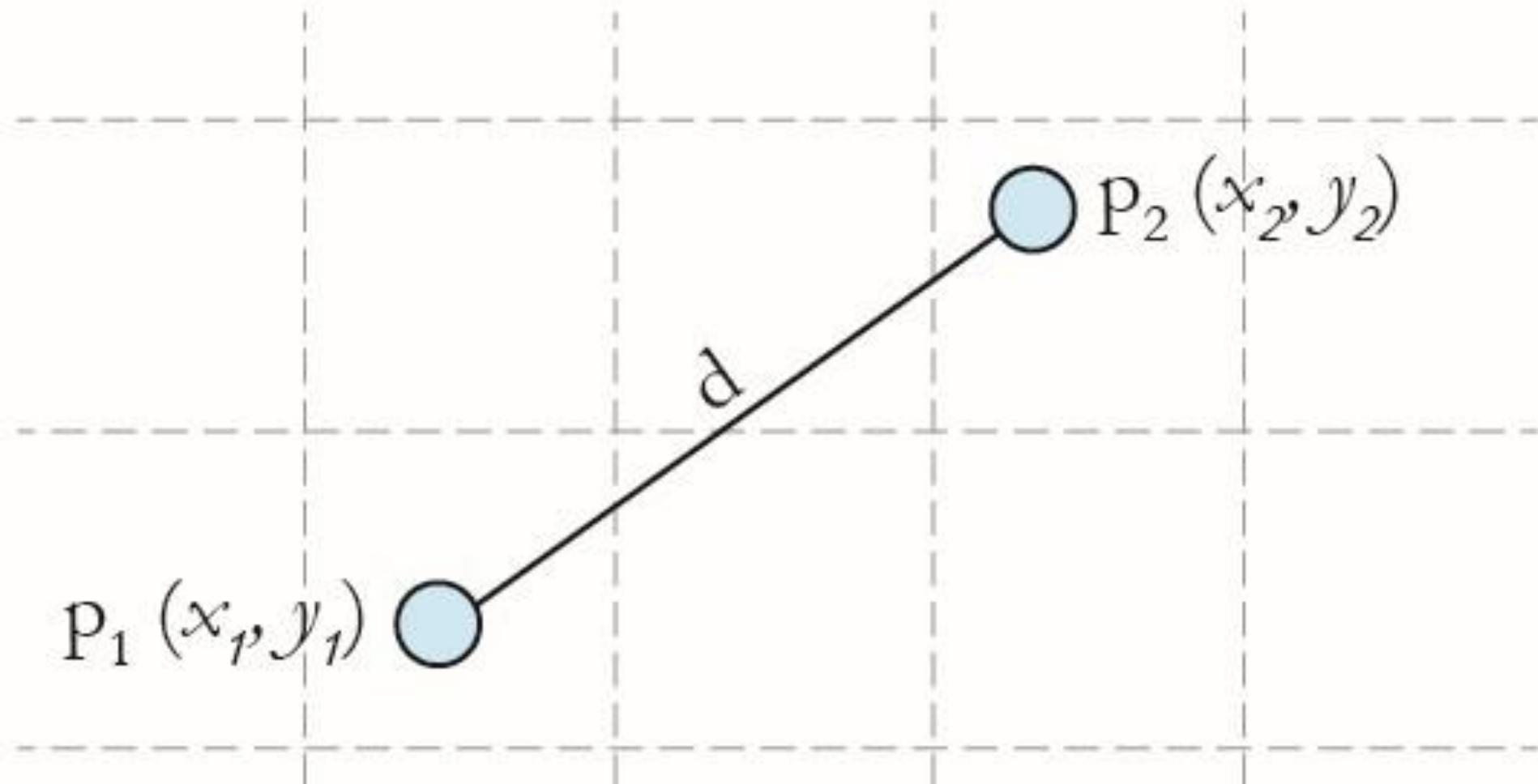
- The query instance is represented as -  $x_q$
- We are interested in estimating the  $f(x_q)$  value, denoted by  $\hat{f}(x_q)$  .

## Choices to consider while implementing K-NN

---

- Decide on the features that are relevant for the problem?
  - Meaning of similarity
- What happens if the range of the features differ a lot?
  - Feature scaling techniques required (use normalization)
- **How to find out how close/far away other points are?**
  - Which distance measure to be used?
- **On how many neighbors should I base my result on?**
  - What is the appropriate value of k to be used for a given problem?

## K-NN Distance measures – Euclidean Distance

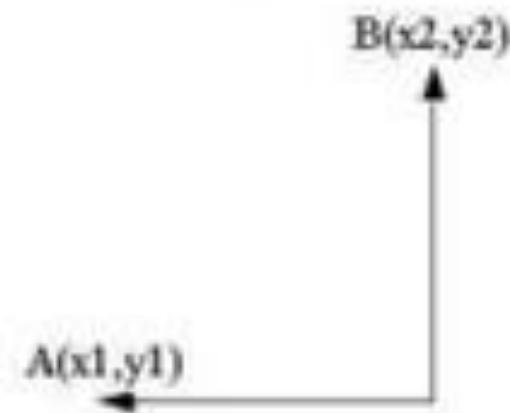


$$\text{Euclidean distance } (d) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

## K-NN Distance measures – Manhattan Distance

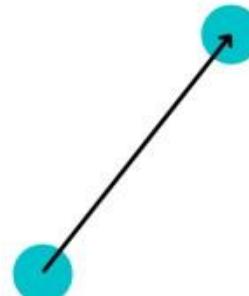
### Manhattan:

$$D_M = |x_2 - x_1| + |y_2 - y_1|$$

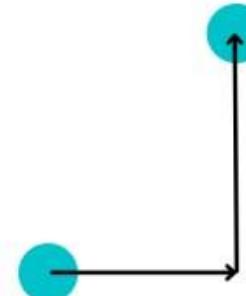


### Differences between Euclidian and Manhattan Distances

datagy.io



Euclidian Distance



Manhattan Distance

## K-NN Distance measures – Minkowski Distance

---

$$\text{Minkowski Distance} = \left( \sum_{i=1}^k (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

k = number of features/attributes

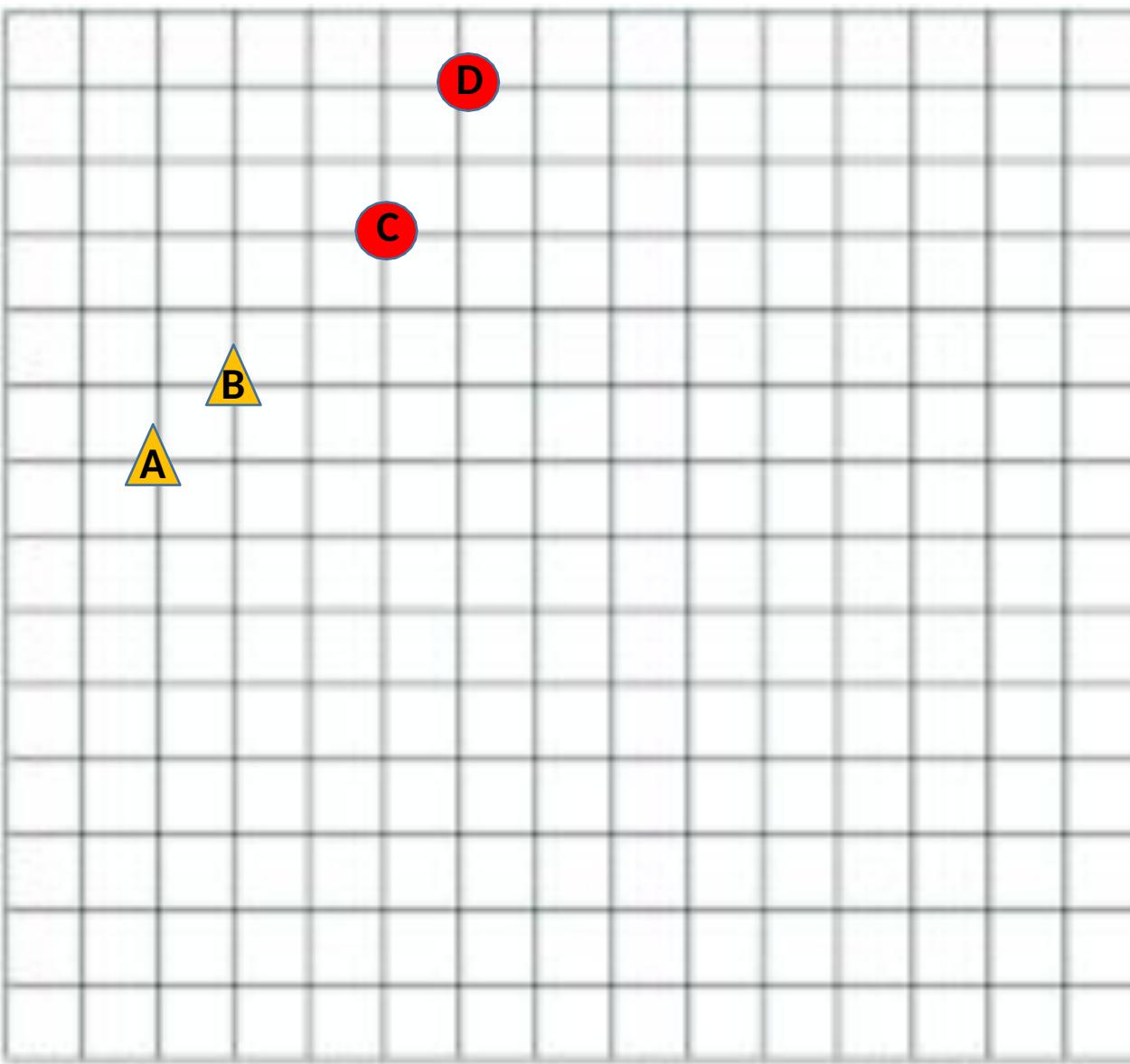
$x_i$  and  $y_i$  -> i<sup>th</sup> attributes

We can manipulate the values of q and calculate different distances:

- q = 1 -> Manhattan distance
- q = 2 -> Euclidean distance

## K-NN How to choose number of neighbors?

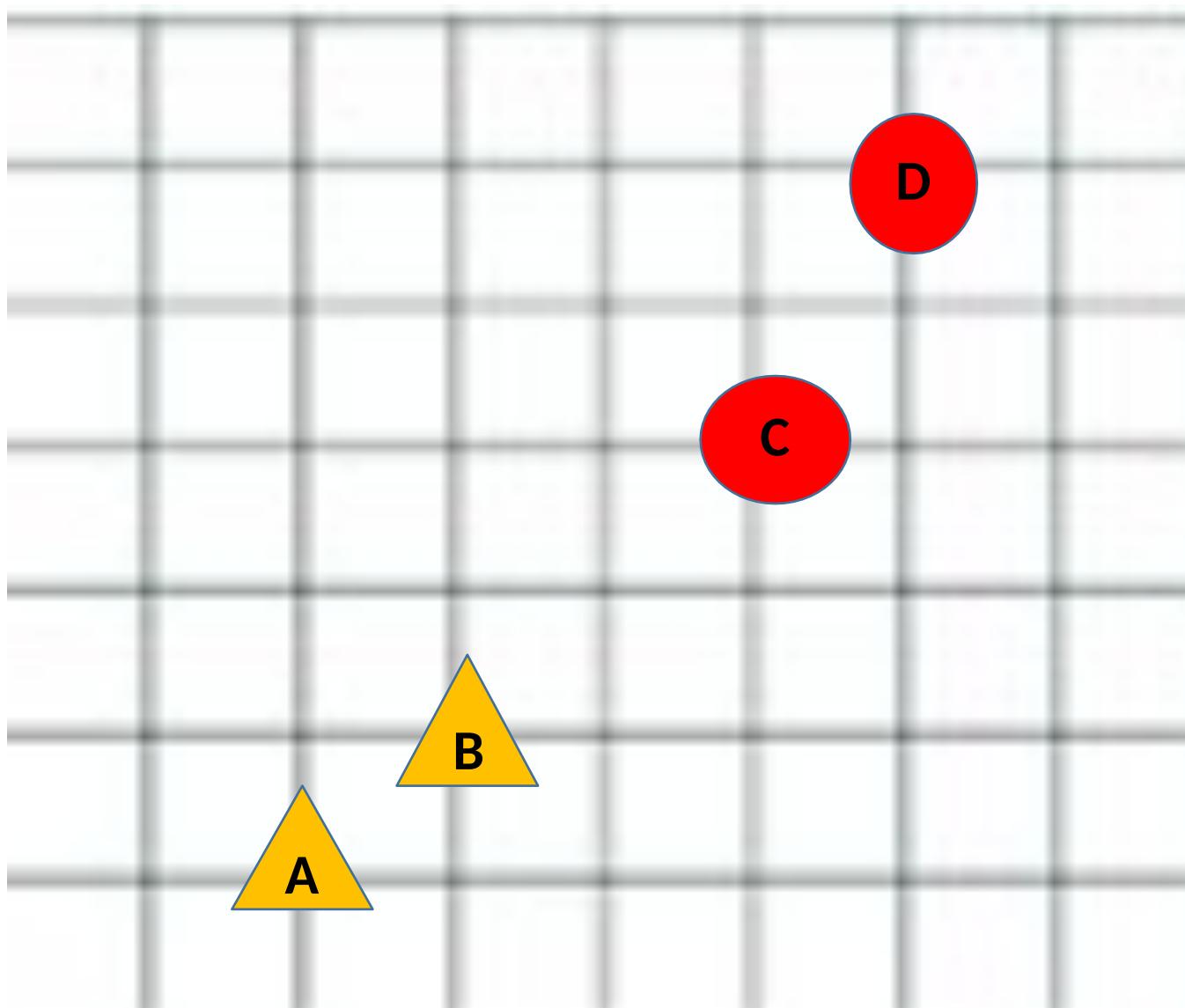
Consider the following points:



Point	X	Y	CLASS
A	2	8	triangle
B	3	9	triangle
C	5	11	circle
D	6	13	circle

## K-NN How to choose number of neighbors?

A closer look:

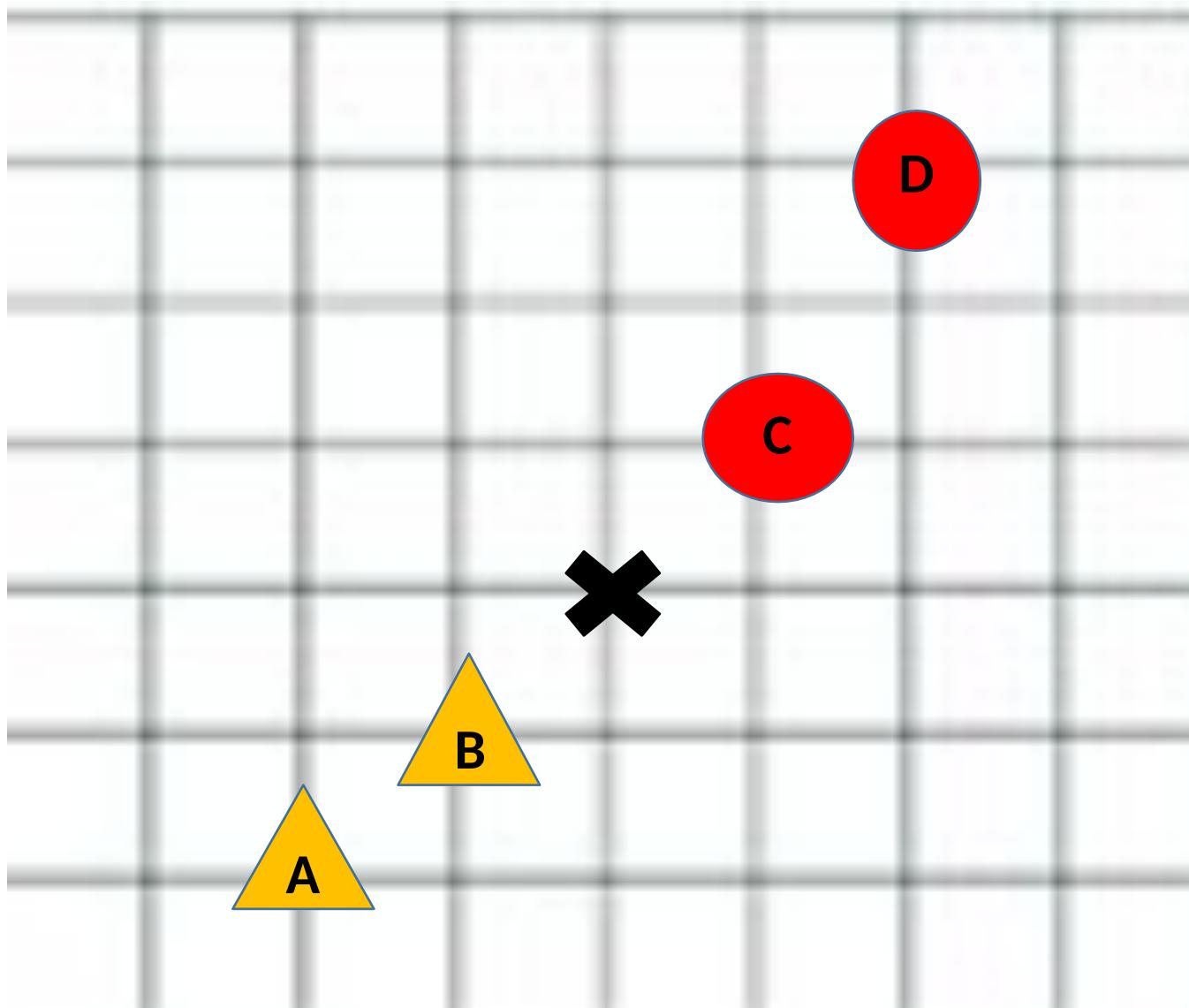


Point	X	Y	CLASS
A	2	8	triangle
B	3	9	triangle
C	5	11	circle
D	6	13	circle

## K-NN How to choose number of neighbors?

Let our query point to be  $x_q = (4, 10)$

Let us calculate Euclidean distance of query point  $x_q$  to all other points.



Point	X	Y	CLASS
A	2	8	triangle
B	3	9	triangle
C	5	11	circle
D	6	13	circle

## K-NN How to choose number of neighbors?

Let our query point to be  $x_q = (4, 10)$

Let us calculate Euclidean distance of query point  $x_q$  to all other points.

Training data

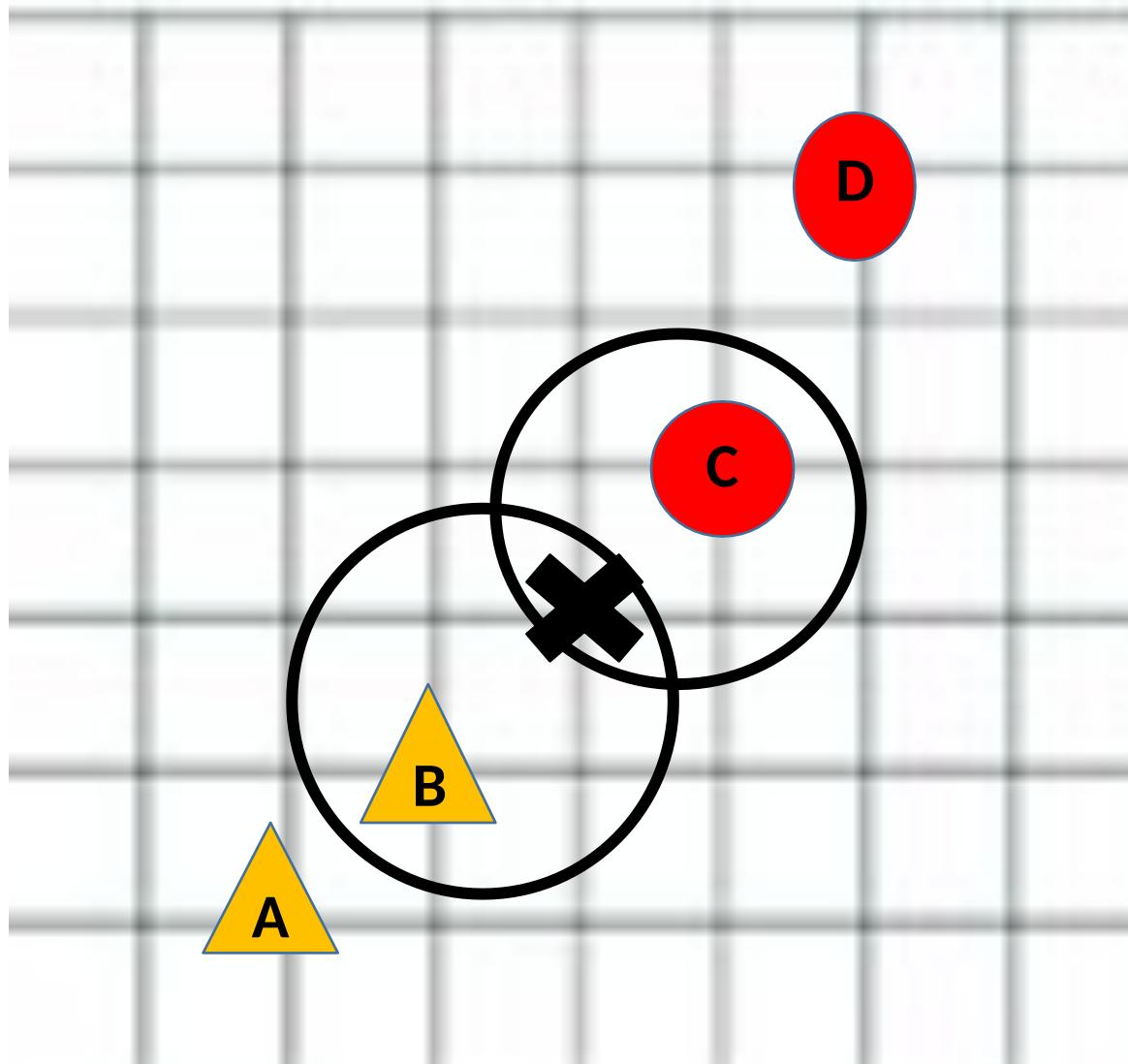
Point	X	Y	CLASS
A	2	8	triangle
B	3	9	triangle
C	5	11	circle
D	6	13	circle

Distance calculation

P1	P2	Distance
$x_q$	A	2.82
$x_q$	B	1.414
$x_q$	C	1.414
$x_q$	D	3.6055

## K-NN How to choose number of neighbors?

Let's start with  $k = 1$ , and experiment with different values of  $k$ . There is a dilemma in choosing neighbors.



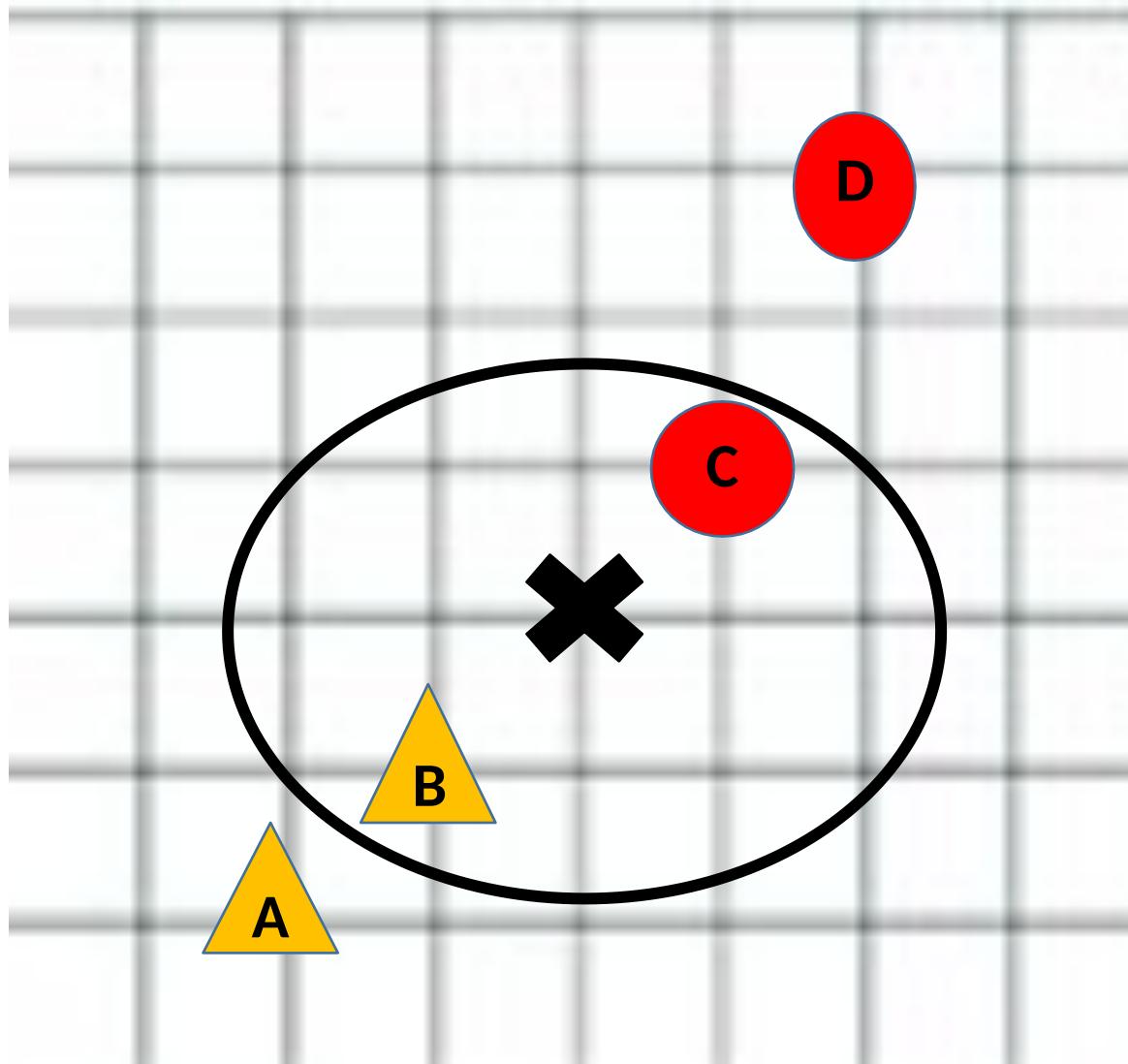
K=1

P1	P2	Distance	Class
$x_q$	A	2.82	Triangle
$x_q$	B	1.414	Triangle
$x_q$	C	1.414	Circle
$x_q$	D	3.6055	Circle

## K-NN How to choose number of neighbors?

Let  $k = 2$ .

There is a dilemma in choosing mode.



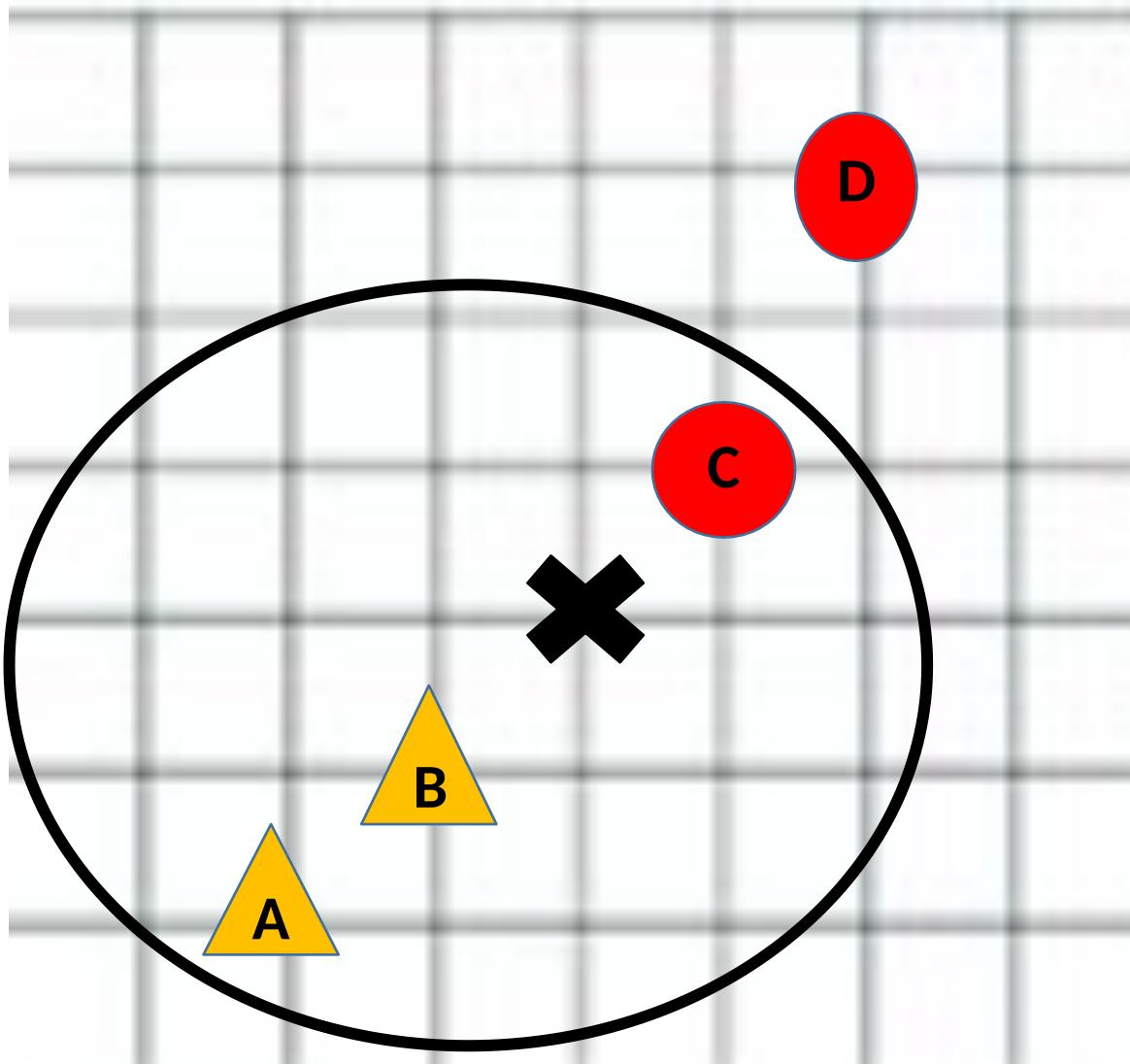
**K=2**

P1	P2	Distance	Class
$x_q$	A	2.82	Triangle
$x_q$	B	1.414	Triangle
$x_q$	C	1.414	Circle
$x_q$	D	3.6055	Circle

## K-NN How to choose number of neighbors?

Let  $k = 3$ .

There is no problem and we can assign the query point as a triangle.



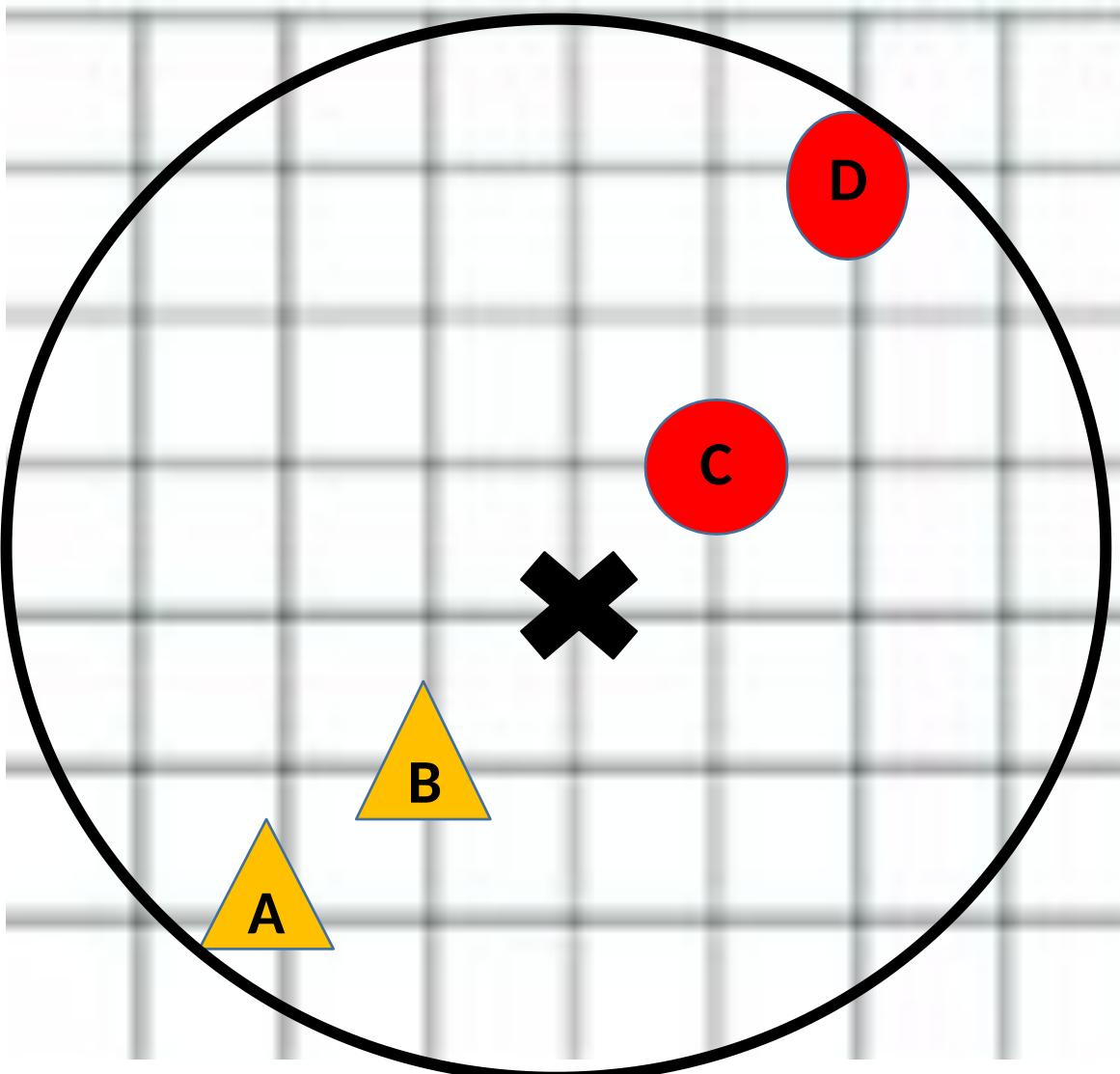
K=3

P1	P2	Distance	Class
$x_q$	A	2.82	Triangle
$x_q$	B	1.414	Triangle
$x_q$	C	1.414	Circle
$x_q$	D	3.6055	Circle

## K-NN How to choose number of neighbors?

Let  $k = 4$ .

There is a problem in choosing the mode again.

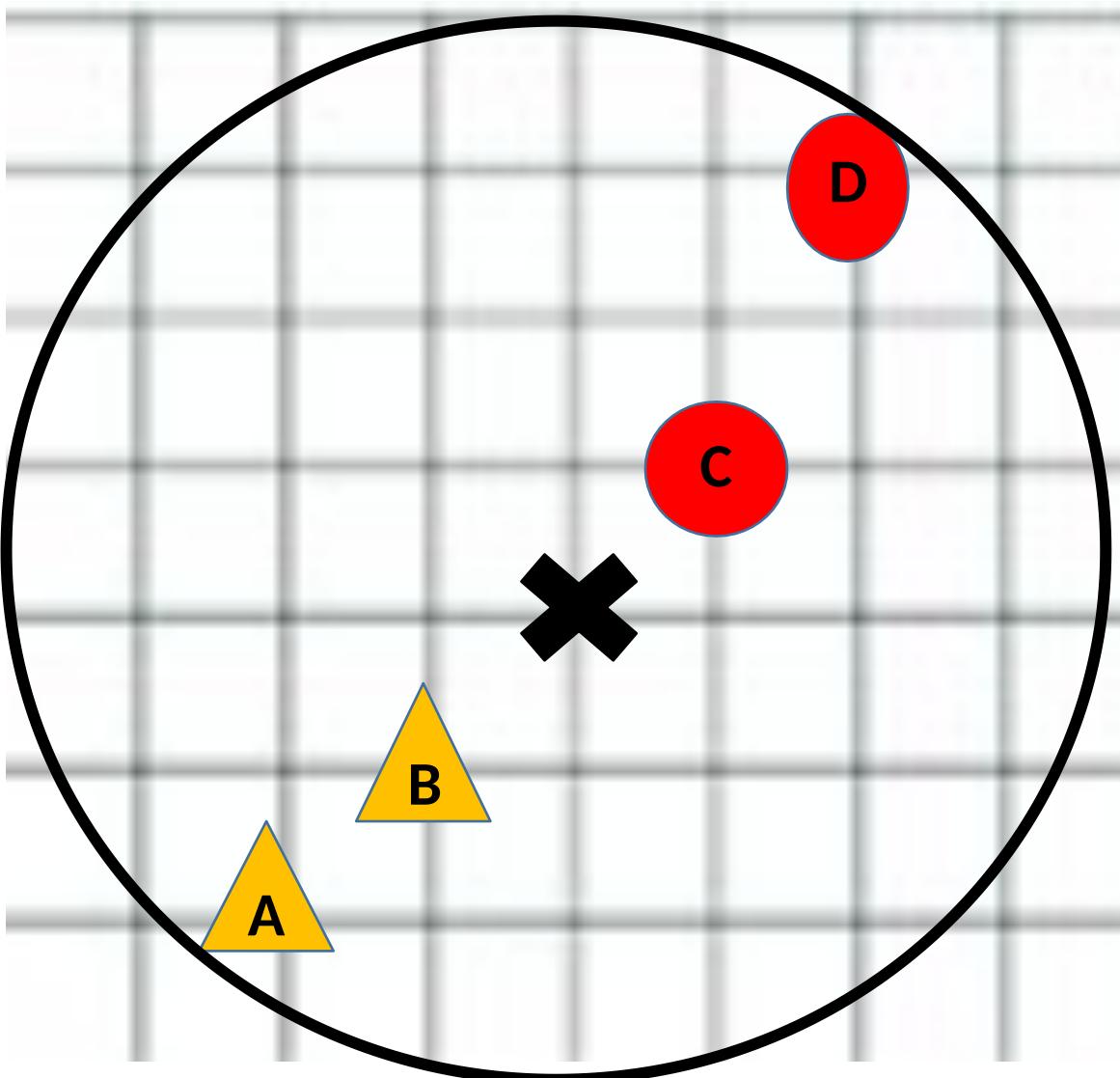


K=4

P1	P2	Distance	Class
$x_q$	A	2.82	Triangle
$x_q$	B	1.414	Triangle
$x_q$	C	1.414	Circle
$x_q$	D	3.6055	Circle

## K-NN How to choose number of neighbors?

From our experiments, K should be odd if number of classes is even and even if number of classes is odd.



K=4

P1	P2	Distance	Class
$x_q$	A	2.82	Triangle
$x_q$	B	1.414	Triangle
$x_q$	C	1.414	Circle
$x_q$	D	3.6055	Circle

## K-NN How to choose number of neighbors?

---

From our experiments, **K should be odd if number of classes is even and even if number of classes is odd.**

However, in practice, there is no right value for k to start with.

### BEST PRACTICE:

- You can start with **k = number of classes + 1**
- In case of tie decrease k by 1.
- Experimental results shows **that  $k = \text{Sqrt}(n)$** , performs well too, where n is the size of dataset.

## K-NN Algorithm for Classification

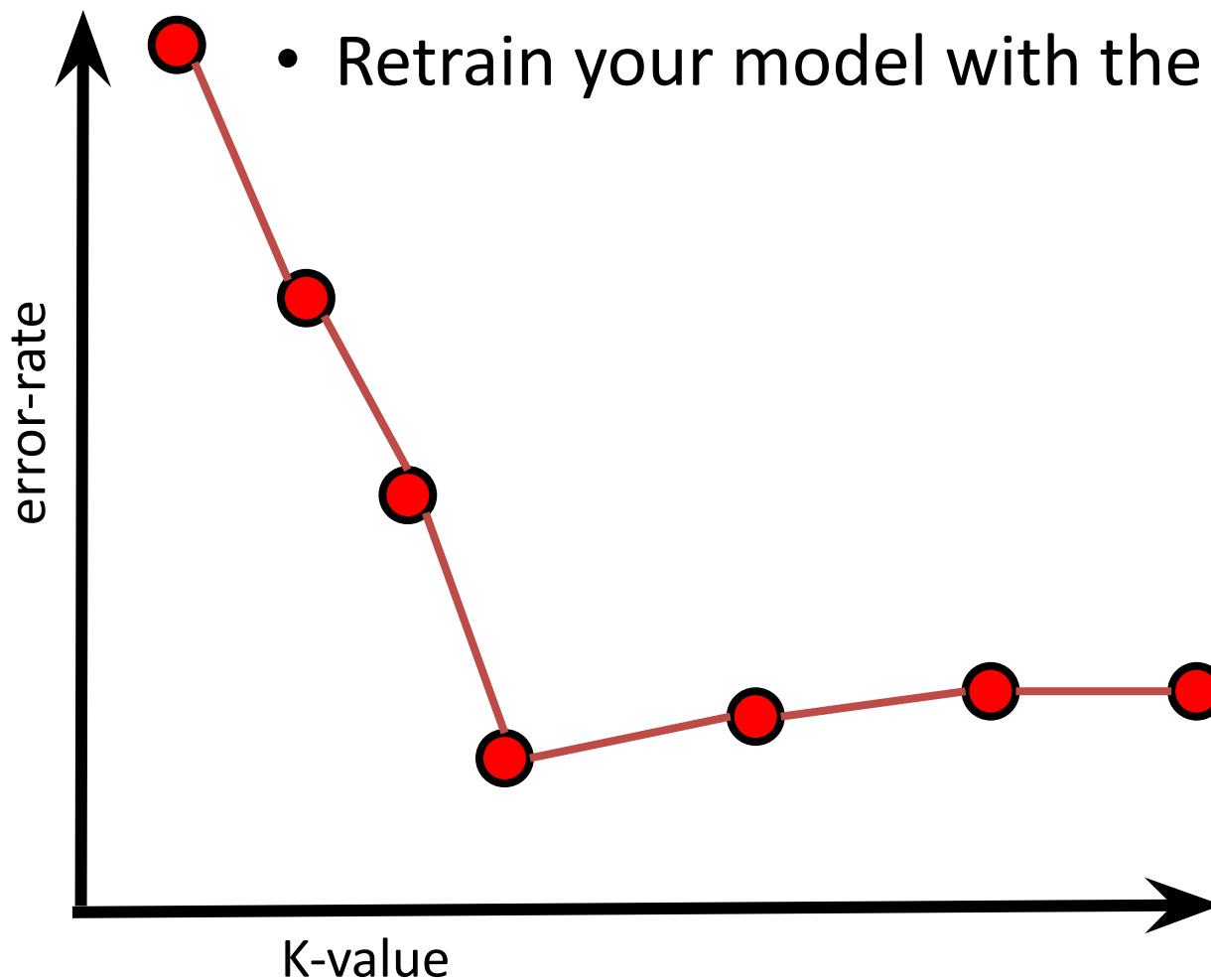
### KNN Algorithm for Discrete-Valued Target Function

- ◆ Training algorithm:
  - Store each training example  $\langle x, f(x) \rangle$
- ◆ Classification algorithm:
  - Given  $x_q$  to be classified
  - Find  $k$  nearest neighbor  $x_1 \dots x_k$  of  $x_q$
  - $\hat{f}(x_q) \leftarrow \arg \max_{v \in V} \sum_{i=1}^k \delta(v, f(x_i))$ 
$$\delta(a, b) = \begin{cases} 1 & a = b \\ 0 & otherwise \end{cases}$$

## K-NN Algorithm for Classification

**Elbow Method** is used to determine the best K value

- Calculate error rate for different K-value
- Choose the optimal k value as elbow value of the curve
- Retrain with new K Value
- Retrain your model with the best K value.



## K-NN Algorithm for Classification

Another is Rule of thumb method

Let  $n$  be the no. of objects.

$$\therefore K = \sqrt{n}/2$$

Say  $n = 250$

$$\therefore K = \sqrt{250}/2 = 11.18 \approx 11$$

### KNN Algorithm for Real-Valued Target Function

- Store each training example  $\langle x, f(x) \rangle$
- Find  $k$  nearest neighbor  $x_1 \dots x_k$  of  $x_q$

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^k f(x_i)}{k}$$

Mean value of  $k$  nearest neighbors is returned as the approximated value of the query instance  $x_q$ .

## K-NN Algorithm for Classification

Consider a one dimensional dataset: Predict the age when wt.= 42

Weight	Age
50	20
54	24
52	27
60	30
35	10

What would be the value of Age if wt.=75 and k=3?

## K-NN Algorithm for Classification

**Find (3,7,?)**

A1	a2	Class
7	7	False
7	4	False
3	4	True
1	4	True

## K-NN Algorithm for Classification

**Let's understand with an example:**

For most of the practical purposes, the dataset is divided into production (training) and test data set (80:20)

--- some kind of pre-processing (learning) is being done

--- Not used as a naive (plain vanilla) version

X	Y	Class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green
5	5	red
4	7	green

## K-NN Algorithm for Classification

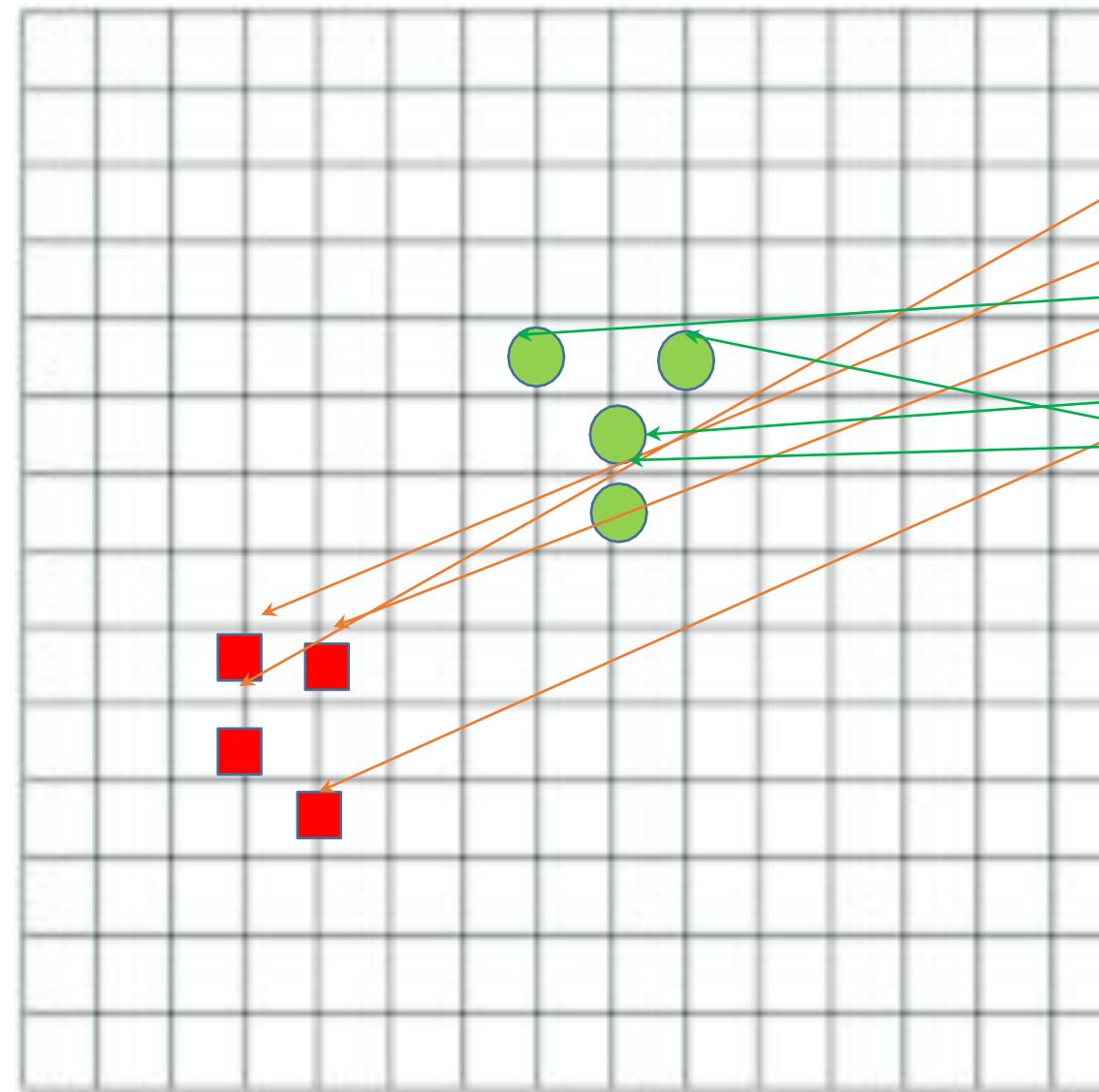
---

- Upload the production data into memory space.
- Decide the distance measure.  
(we will choose Euclidean)
- Decide the k value  
(we will choose k = 4)
- For each data point in test data, find distance with respect to all neighbors and choose the nearest 4 neighbors and assign the prediction as **mode** of the neighbors.
- Calculate the error and accuracy.

X	Y	Class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green
5	5	red
4	7	green

## K-NN Algorithm for Classification

**Result:**

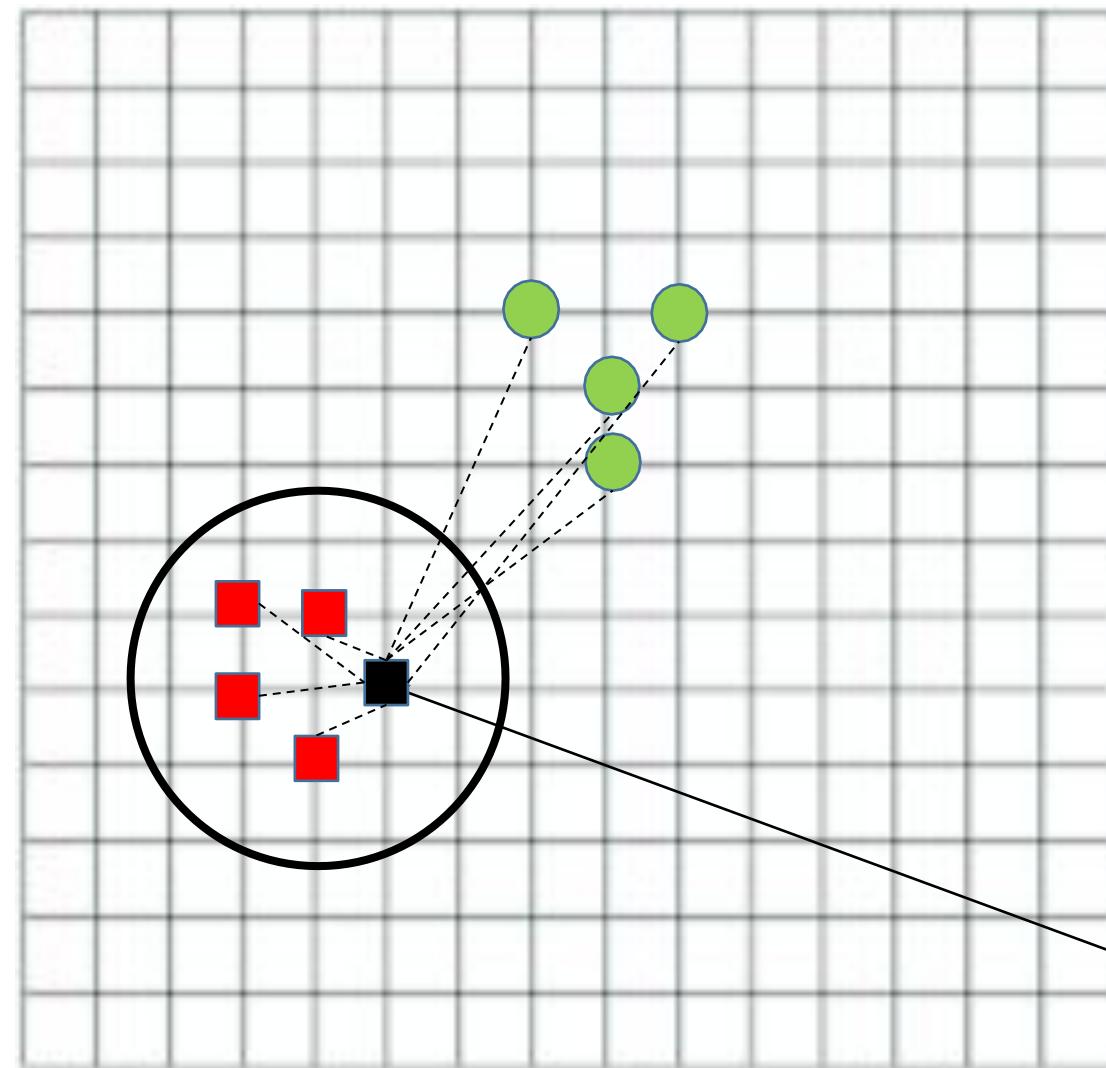


X	Y	Class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green

X	Y	Class
5	5	red
4	7	green

## K-NN Algorithm for Classification

TP	TN	FP	FN	Error = (FP + FN)/ All predictions
1	0	0	0	0

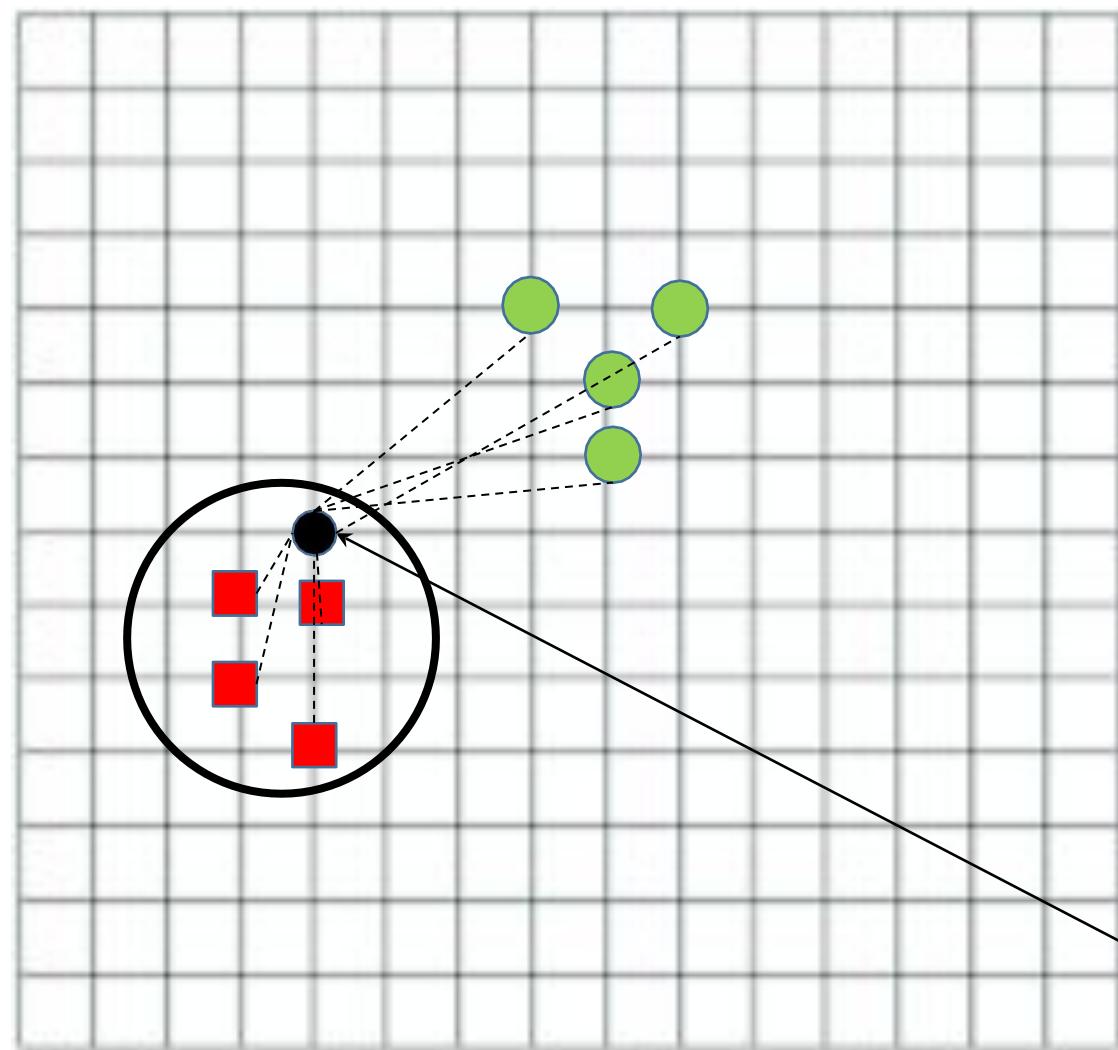


X	Y	Class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green

X	Y	Class
5	5	red
4	7	green

## K-NN Algorithm for Classification

TP	TN	FP	FN	Error = (FP + FN)/ All predictions
1	0	1	0	$\frac{1}{2} = 0.5$



X	Y	Class
3	5	red
3	6	red
4	6	red
4	4	red
7	10	green
8	9	green
8	8	green
9	10	green

X	Y	Class
5	5	red
4	7	green

## Bias-Variance Trade off - How new instances get classified

### K is very low

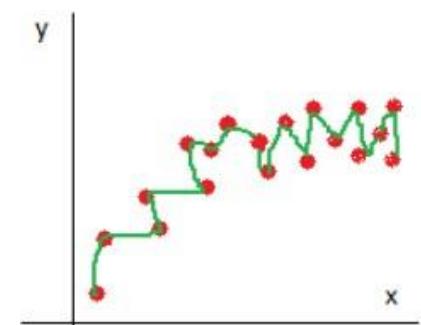
It could label the query points as the point near it as only small points are taken into consideration. Chance for high variance aka over fitting.

### K is very high

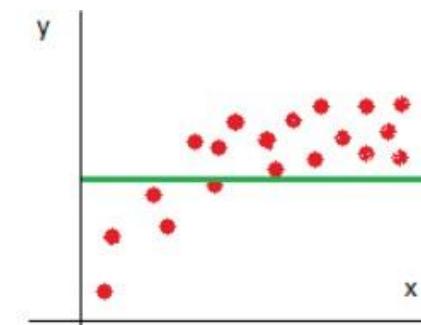
It could mostly label the query points as the class label which has a majority in the given dataset. Chance for high bias aka under fitting.

### K is neither high nor low

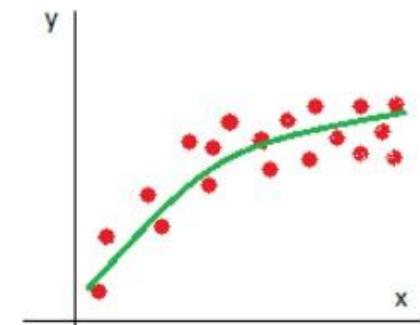
Good trade-off without high bias and variance.



Overfitting (High Variance)



Underfitting (High Bias)



Just Right

## Practice Problem – K-NN for Regression

Find the weight of ID 11. The value of k is 3. Use Euclidean distance

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

## Practice Problem – K-NN for Regression

We find distance between the point id = 11 and every other point.

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

Query pt	Data points	Distance
id 11	id 1	7.017
id 11	id 2	12.006
id 11	id 3	8.006
id 11	id 4	4.019
id 11	id 5	2.118
id 11	id 6	2.022
id 11	id 7	19.001
id 11	id 8	10.00
id 11	id 9	15
id 11	id 10	6.00

## Practice Problem – K-NN for Regression

Sort the distances and choose 3 nearest neighbours.

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

Query pt	Data points	Distance
id 11	id 6	2.022
id 11	id 5	2.118
id 11	id 4	4.019
id 11	id 10	6.00
id 11	id 1	7.017
id 11	id 3	8.006
id 11	id 8	10.00
id 11	id 2	12.006
id 11	id 9	15
id 11	id 7	19.001

## Practice Problem – K-NN for Regression

We choose mean of these 3 nearest neighbors as our answer.

ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

Id	Weight
id6	60
id5	72
id4	59

$$\text{answer} = \frac{60+72+59}{3} = 63.666$$

## Practice Problem – K-NN for Regression

You are a data scientist working for a real estate company. The company wants to predict the price of houses based on various features. You decide to use K-Nearest Neighbors (KNN) regression for this task.

House ID	Size (sq ft)	Bedrooms	Bathrooms	Age (years)	Distance to City Center (miles)	Price (\$)
1	2000	3	2	10	5	300,000
2	1500	2	1	5	7	200,000
3	2500	4	3	15	3	400,000
4	1800	3	2	8	6	280,000
5	2200	4	3	12	4	350,000
6	2100	3	2	9	5	?

Your task is to predict the price of house 6 based on its features using KNN regression with Minkowski distance ( $p=3$ ). Consider the 3 nearest neighbours.

## Practice Problem – K-NN for Regression

**Solution:**

$$\text{Minkowski Distance} = \left( \sum_{i=1}^k (|x_i - y_i|)^q \right)^{\frac{1}{q}}$$

House 6 is where we have to make our target prediction. Considering each house as a point in 5 dimensional space, the co-ordinates of each point would be their attributes (excluding the target variable).

For House 6, the target point would hence be (2100, 3, 2, 9, 5)

Similarly for Houses 1 through 5, their respective points would be:

(2000, 3, 2, 10, 5)

(1500, 2, 1, 5, 7)

(2500, 4, 3, 15, 3)

(1800, 3, 2, 8, 6)

(2200, 4, 3, 12, 4)

## Practice Problem – K-NN for Regression

---

Calculate Minkowski distance between point 6 and the remaining points:

Distance between House 1 and 6:

$$d_{1,6} = (|2000 - 2100|^3 + |3 - 3|^3 + |2 - 2|^3 + |10 - 9|^3 + |5 - 5|^3)^{\frac{1}{3}} \approx 100.00$$

Similarly calculate distances for the other 4 Houses:

$$d_{2,6} \approx 60.44$$

$$d_{3,6} \approx 39.82$$

$$d_{4,6} \approx 30.30$$

$$d_{5,6} \approx 100.01$$

The three nearest neighbours are hence houses 2, 3 and 4. To predict price, calculate the mean price of the three selected houses.

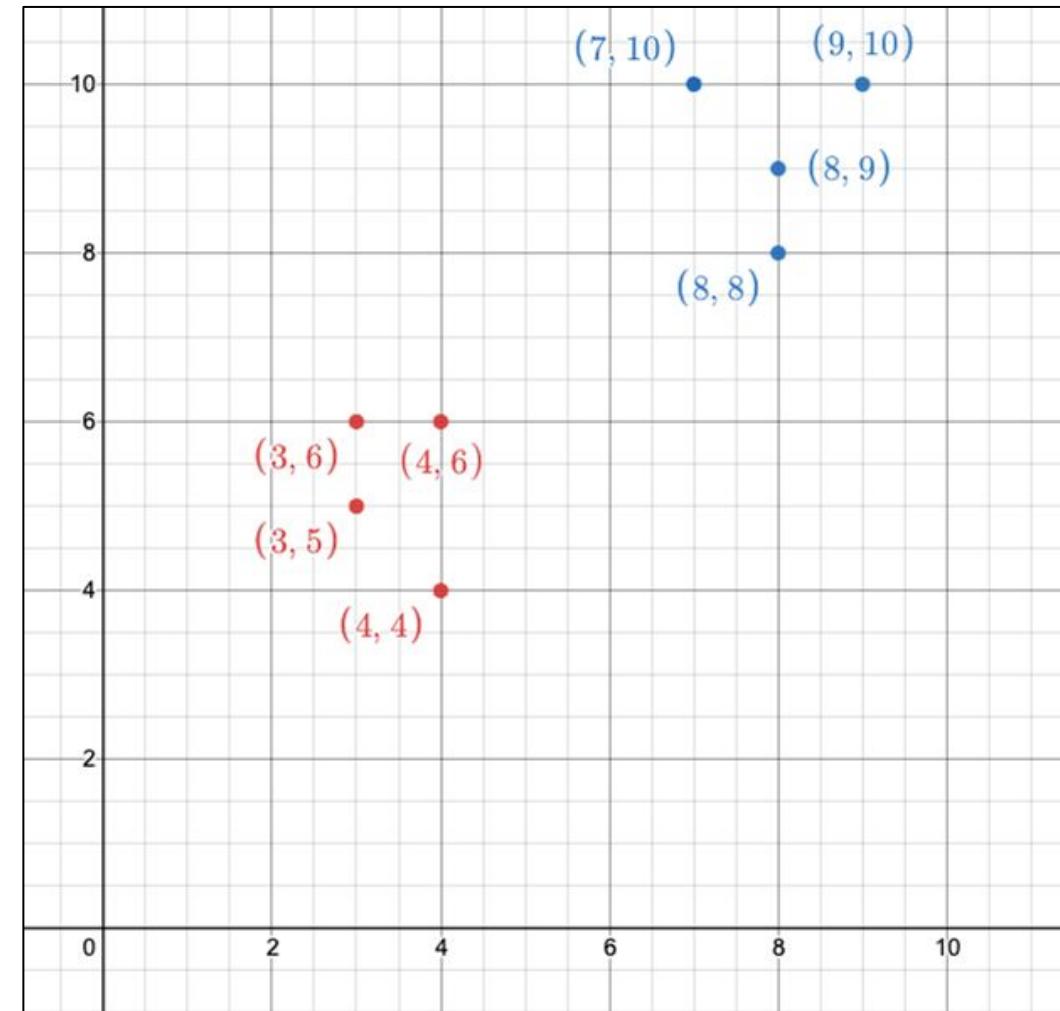
***Answer***  $\approx 293,333$

## K-NN Algorithm for Classification

For each data point in test data, find distance with respect to all neighbors and choose the nearest 4 neighbors and assign the prediction as **mean** of the neighbors.

Calculate the error using MSE.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\underbrace{y_i}_{\text{test set}} - \underbrace{\hat{y}_i}_{\text{predicted value}})^2$$



X	Y	value
3	5	3.5
3	6	2.6
4	6	2.8
4	4	3.1
7	10	8.7
8	9	8.5
8	8	9.3
9	10	9.7

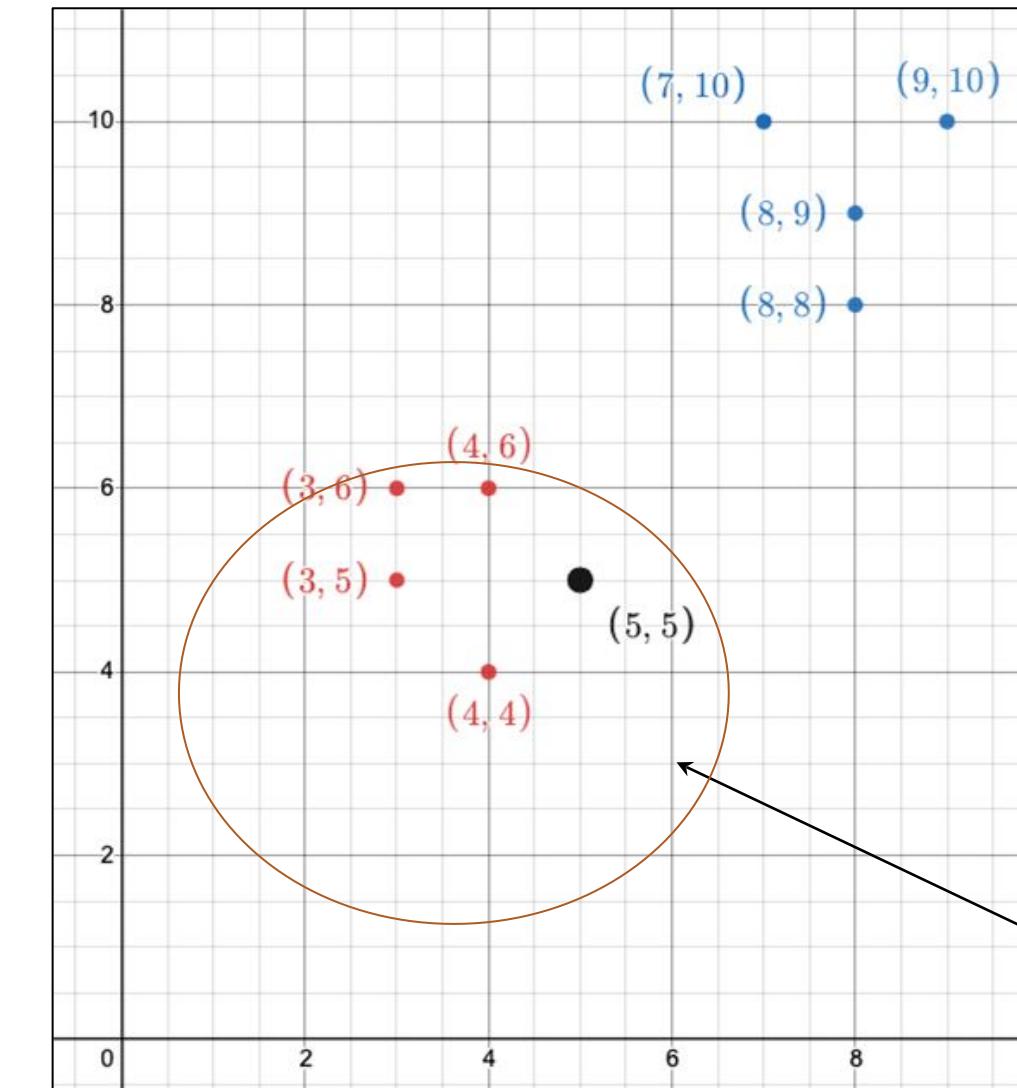
X	Y	value
5	5	3.7
7	8	8.4

## K-NN Algorithm for Classification

$$\hat{y} = \frac{3.5 + 2.6 + 2.8 + 3.1}{4} = 3.0$$

$$e_1 = (3.7 - 3.0)^2 \\ = 0.49$$

e1	0.49



X	Y	value
3	5	3.5
3	6	2.6
4	6	2.8
4	4	3.1
7	10	8.7
8	9	8.5
8	8	9.3
9	10	9.7

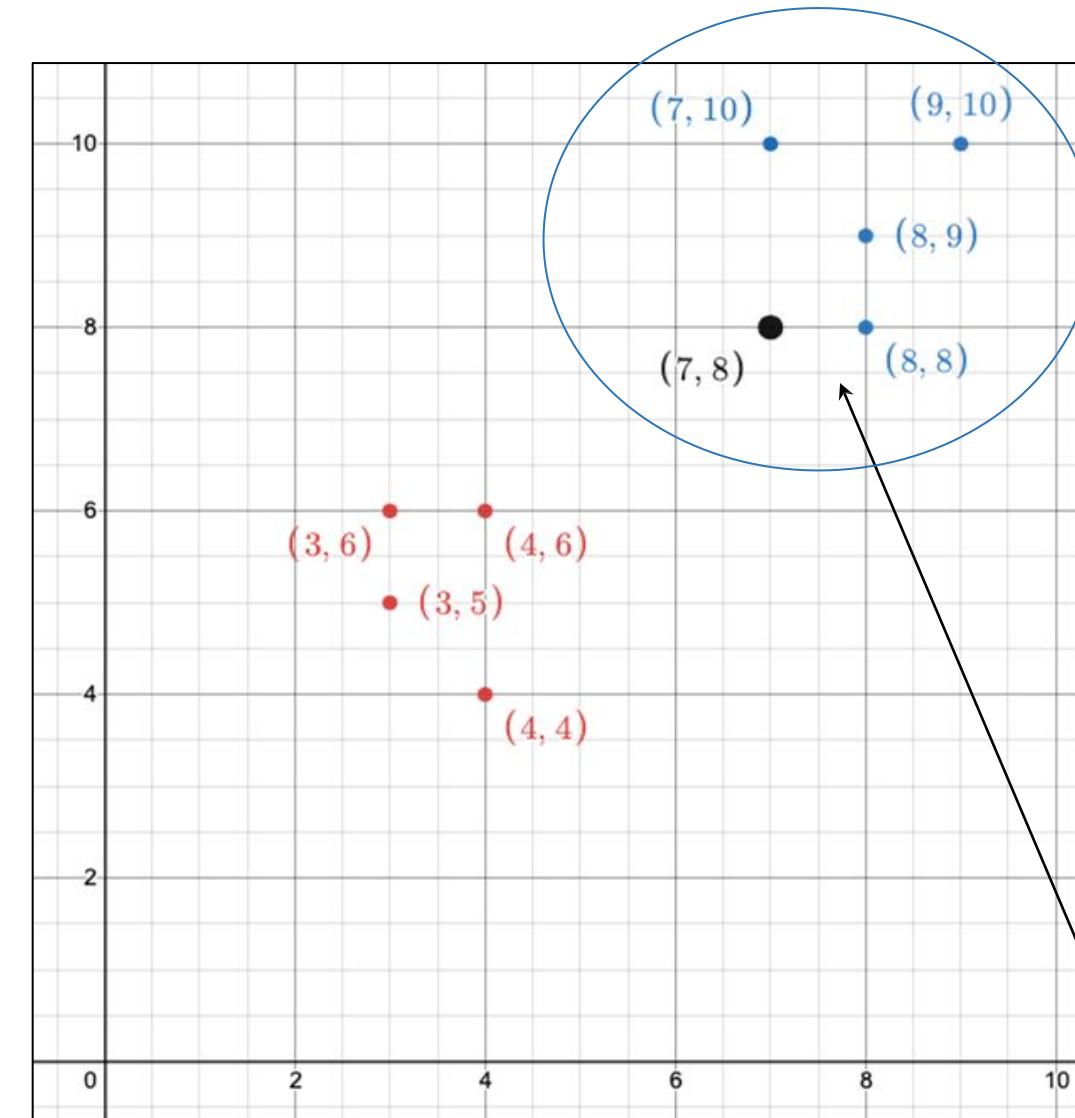
X	Y	value
5	5	3.7
7	8	8.4

## K-NN Algorithm for Classification

$$\hat{y} = \frac{8.7 + 8.5 + 9.3 + 9.7}{4} = 9.05$$

$$e_2 = (8.4 - 9.05)^2 \\ = 0.4225$$

e1	0.49
e2	0.4225



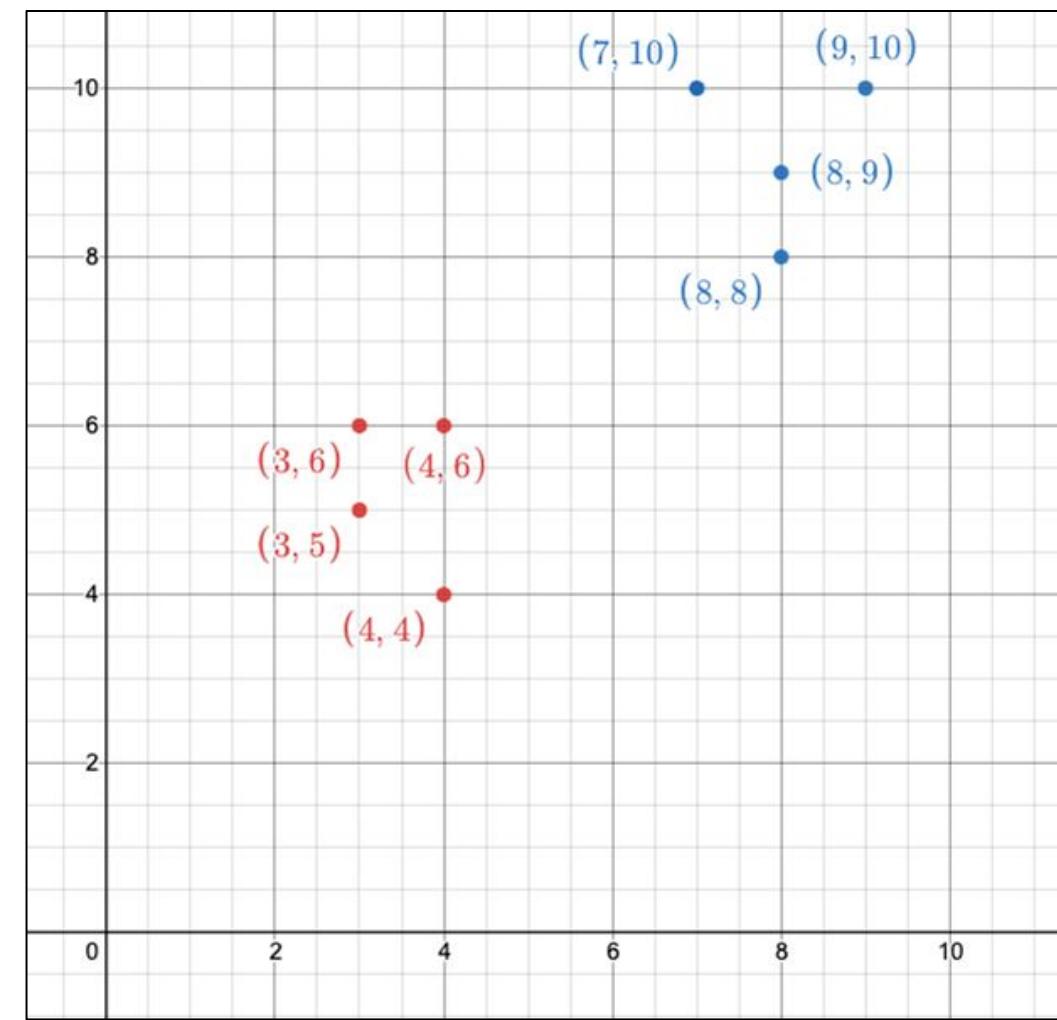
X	Y	value
3	5	3.5
3	6	2.6
4	6	2.8
4	4	3.1
7	10	8.7
8	9	8.5
8	8	9.3
9	10	9.7

X	Y	value
5	5	3.7
7	8	8.4

## K-NN Algorithm for Classification

e1	0.49
e2	0.4225

$$\begin{aligned} \text{total error} &= \frac{1}{2}(e1 + e2) = \frac{0.49 + 0.4225}{2} \\ &= 0.45625 \end{aligned}$$



X	Y	value
3	5	3.5
3	6	2.6
4	6	2.8
4	4	3.1
7	10	8.7
8	9	8.5
8	8	9.3
9	10	9.7

X	Y	value
5	5	3.7
7	8	8.4

## Practice Problem – K-NN for Classification

Consider the following data set. Apply kNN algorithm to assess the risk for a patient whose BP is 100, sugar is 135, Haemoglobin is 12 and WBC count is 8 thousand. Take k =3 and use Euclidean distance measure.

#	B.P.	Sugar	Haemoglobin	WBC(in thousands)	Risk
A	100	120	12	6	No
B	110	130	14	5	Yes
C	120	110	11	7	Yes
D	100	140	13	7	No
E	115	140	11	6	Yes

That means the query point is  $x_q = (100, 135, 12, 8)$

## Problem – K-NN for Classification

We calculate distance between our query point  $x_q = (100, 135, 12, 8)$  and every other point

#	B.P.	Sugar	Haemoglobin	WBC(in thousands)	Risk
A	100	120	12	6	No
B	110	130	14	5	Yes
C	120	110	11	7	Yes
D	100	140	13	7	No
E	115	140	11	6	Yes

Query	other points	Euclidean distance
$x_q$	A	15.13
$x_q$	B	11.74
$x_q$	C	32.04
$x_q$	D	5.19
$x_q$	E	15.96

## Problem – K-NN for Classification

Sort these distances and pick the 3 nearest neighbors. (as k = 3)

#	B.P.	Sugar	Haemoglobin	WBC(in thousands)	Risk
A	100	120	12	6	No
B	110	130	14	5	Yes
C	120	110	11	7	Yes
D	100	140	13	7	No
E	115	140	11	6	Yes

Query	other points	Euclidean distance
$x_q$	D	5.19
$x_q$	B	11.74
$x_q$	A	15.13
$x_q$	E	15.96
$x_q$	C	32.04

## Problem – K-NN for Classification

our choice for the label of  $x_q$  is mode of the risks of D, B, A  
 i.e. mode(NO, YES, NO) = NO

#	B.P.	Sugar	Haemoglobin	WBC(in thousands)	Risk
A	100	120	12	6	No
B	110	130	14	5	Yes
C	120	110	11	7	Yes
D	100	140	13	7	No
E	115	140	11	6	Yes

Query	other points	Euclidean distance
$x_q$	D	5.19
$x_q$	B	11.74
$x_q$	A	15.13
$x_q$	E	15.96
$x_q$	C	32.04

## References

---

- <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- <https://www.geeksforgeeks.org/instance-based-learning/>
- <https://www.cs.ucdavis.edu/~vemuri/classes/ecs271/ch8.pdf>
- <https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python/>
- <https://datagy.io/python-knn/>
- [https://www.bogotobogo.com/python/scikit-learn/scikit machine learning k-NN k-nearest-neighbors-algorithm.php](https://www.bogotobogo.com/python/scikit-learn/scikit_machine_learning_k-NN_k-nearest-neighbors-algorithm.php)
- <https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/>

# MACHINE LEARNING (UE22CS352A)



## Weighted K-Nearest Neighbour

---

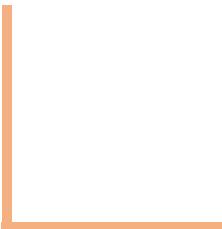
**Course Instructor: Dr Pooja Agarwal**  
Professor, Dept. of CSE

# MACHINE LEARNING

---

## Weighted K-Nearest Neighbour

The slides are generated from various internet resources and books, with valuable contributions from multiple professors and teaching assistants in the university.

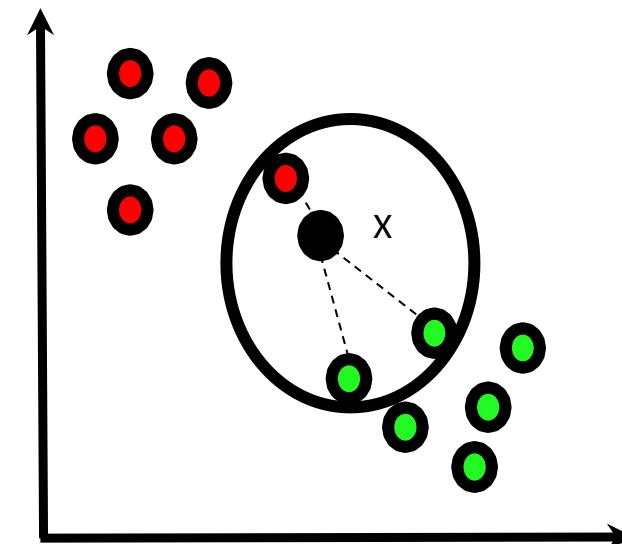


- Consider the given 2-D plot of a data set where the label is binary.
- Consider the query point  $x_q$
- With vanilla K-NN where  $K = 3$ :
  - $x$  will be classified as green, when it's more likely to be red.

### How to handle this?

K-NN follows the basic rule of life!

The closer you are to me the more importance I give you.



- In *majority voting approach*, every **neighbor** has **the same impact** on the classification.
- This factor makes classification algo. sensitive to the choice of k.
- In order to reduce this impact of k, we assign weight to each of the distance for NN say  $x_i$ :

$$w_i = \frac{1}{d(x_q, x_i)^2}$$

- As a result of applying wt. to the distance, the training examples that are located far away from  $x_q$  have a weaker impact on the classification.
- Using the distance-weighted voting scheme, the class label can be determined:

### Distance-wtd. Voting

$$y' = \arg \max_v \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$$

## Few Points about KNN

---

1. NN classification is a part of instance-based learning.
2. Lazy learners like NN classifiers do not need model building(They don't build model).
- 3.NN classifiers make their predictions based on local information whereas DT and rule-based classifiers attempt to find a global model that fits the entire input space (Not good for generalization)
4. Appropriate proximity measures(similarity measure/ distance measures) play a significant role in NN classifiers.

- Distance-weighted nearest neighbor algorithm
- Weight the contribution of each of the  $k$  neighbors according to their distance to the query  $x_q$
- *Greater the weight, closer the neighbors*

$$\hat{f}(x_q) \leftarrow \operatorname{argmax}_{v \in V} \sum_{i=1}^k w_i \delta(v, f(x_i))$$

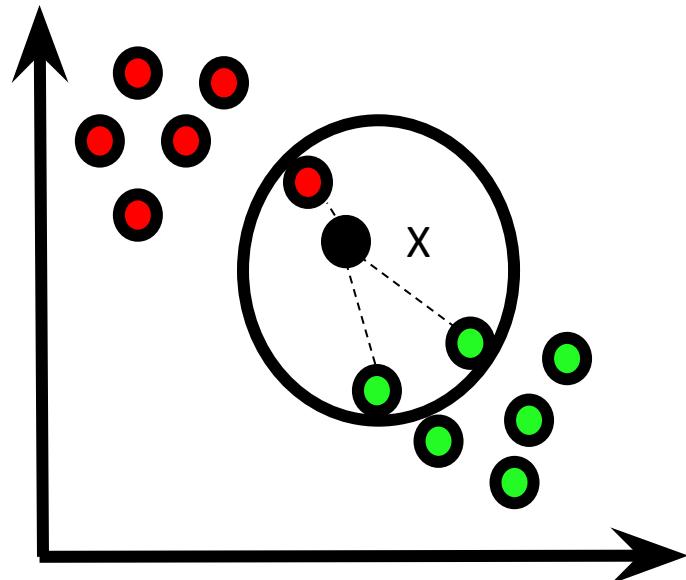
$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Consider the given 2-D plot of a two class data set

Consider the query point  $x$

With normal KNN with  $K=3$ ,  $x$  will be classified as green

With weighted KNN and  $K=3$ ,  $x$  will be classified as red since its more close to the query  $x_q$

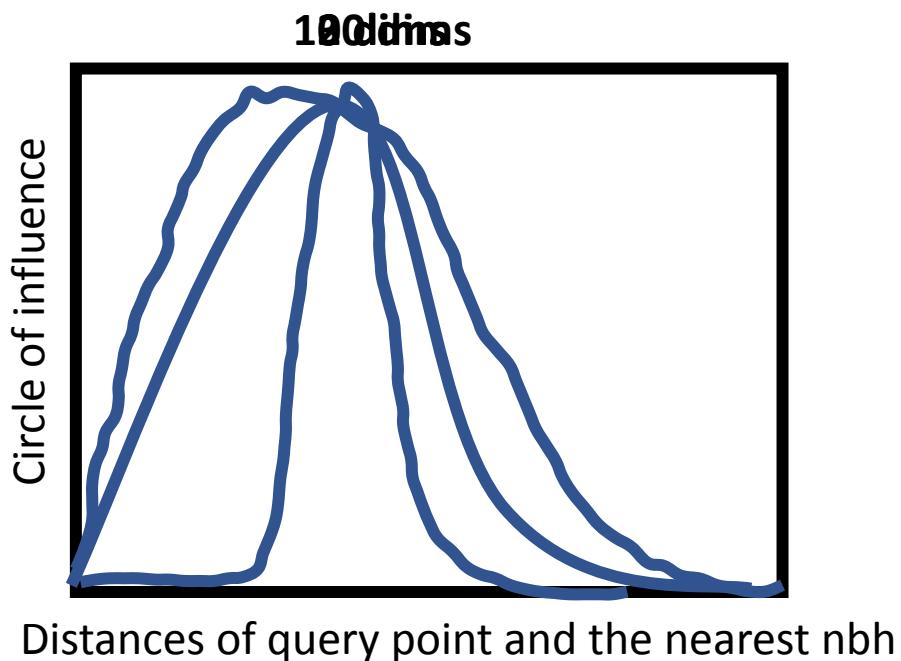


The classification of an instance  $x$ , will be *most similar to the classification of the  $K$  other instances that are in vicinity(neighborhood).*

In simple English **Birds of the same feather flock together**

- **K-NN slow algorithm:** K-NN might be very easy to implement **but as data set grows** efficiency or speed of algorithm declines very fast.
- **Curse of Dimensionality:** KNN works well with small number of input variables but as the **numbers of variables grow** K-NN algorithm struggles to predict the output of new data point.

- The *Radius or circle of influence* of each data point becomes smaller and smaller as we have more attributes.



The curse of Dimensionality can be overcome by:

- **Assigning weights** to the attributes when calculating distances.
- Using the “**Leave-one-out**” approach. Iteratively, we leave out one of the attributes and test the algorithm by the cross-validation method. The exercise can then lead us to the best set of attributes.

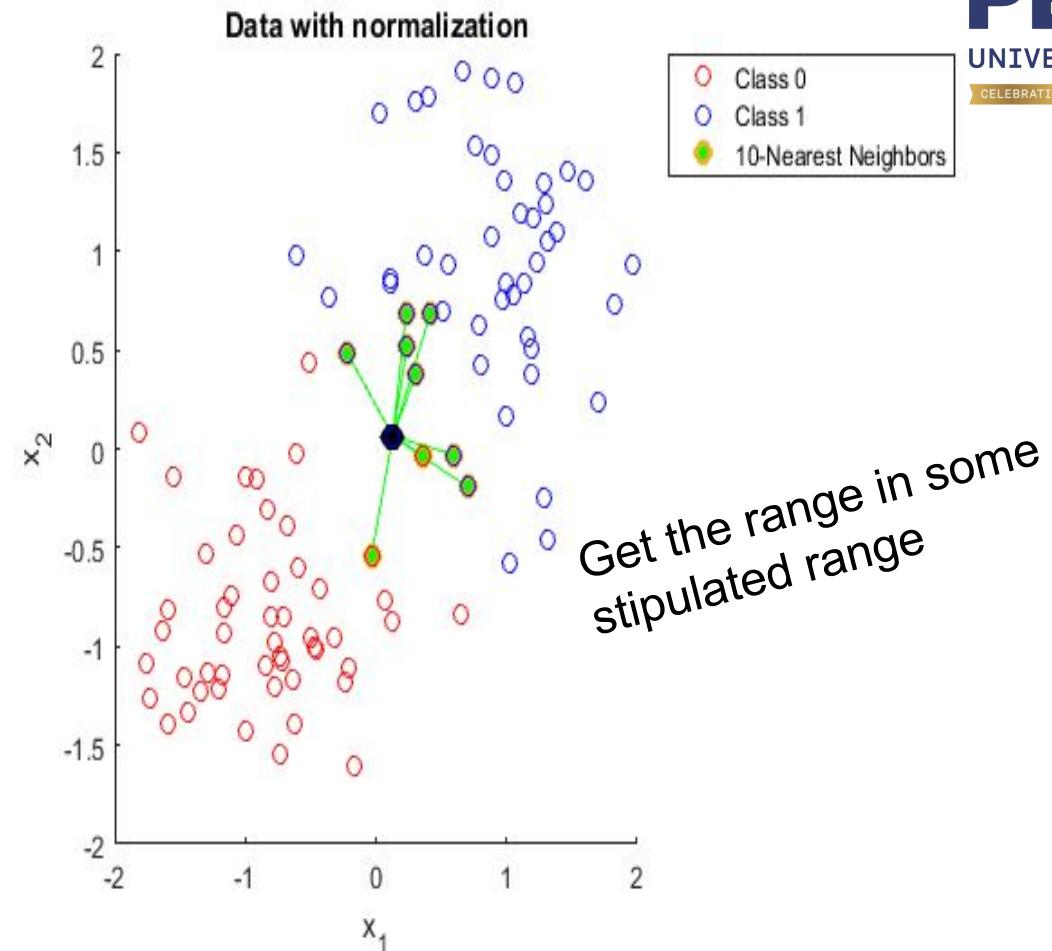
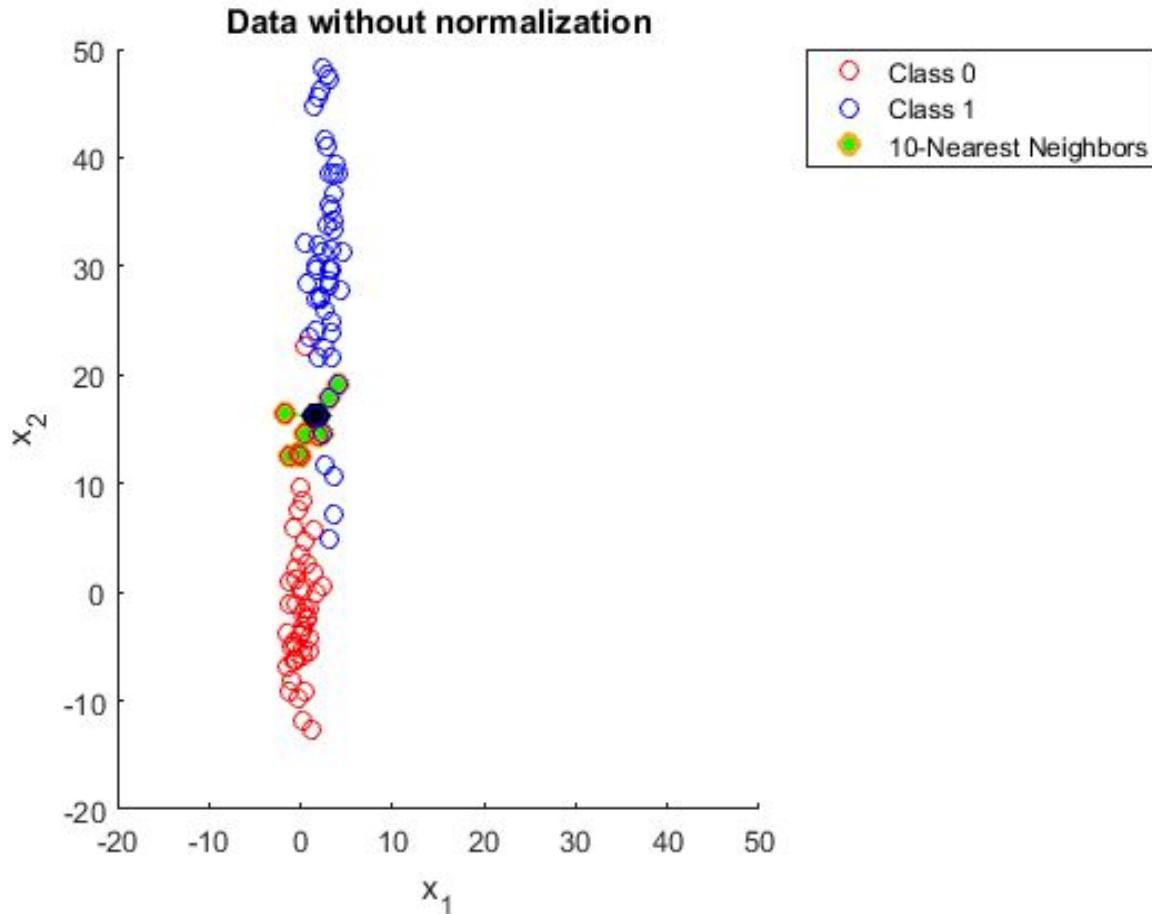
- **Optimal number of neighbors:** One of the biggest issues with K-NN is **to choose the optimal number of neighbors** to be considered while classifying the new data entry
- **Imbalanced data causes problems:** k-NN **doesn't perform well on imbalanced data.** If we consider two classes, A and B, and the majority of the training data is labelled as A, then the model will ultimately give a lot of preference to A. This might result in getting the less common class B wrongly classified.

- **Outlier sensitivity:** K-NN algorithm is **very sensitive to outliers** as it simply chose the neighbors based on distance criteria.
- **Missing Value treatment:** K-NN inherently has no capability of dealing with missing value problem.

- **K-NN needs homogeneous features:** If you decide to build k-NN using a common distance, like **Euclidean or Manhattan distances**, it is completely necessary that **features have the same scale**, since absolute differences in features weight the same, i.e., a given distance in feature 1 must mean the same for feature 2

# MACHINE LEARNING

## Issues with KNN



ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

Source: [Applied Machine Learning Course](#)

Lets calculate the distance between any two data points , say 1 and 2. Clearly the **Euclidean distance between these two points is quite large** bcoz of the **higher magnitude of Income attribute.**

So any distance based models are **highly sensitive to the scale of data** and thus influence the final outcome.

To avoid this biasedness, **scaling or normalization** of attributes is required.

Many techniques, **like computing mean and std deviation** of the attribute and

$$z = (x - \text{mean})/\text{std. deviation}$$

**Or** One can use **Min-Max scaling** also

$$= (x - x_{\min}) / (x_{\max} - x_{\min})$$

# MACHINE LEARNING

## Effect of Normalizing



### PRE NORMALIZING:

Euclidean distance between 1 and 2=

$$[(100000 - 80000)^2 + (30 - 25)^2]^{(1/2)} = \\ 20000$$

ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

High magnitude of income affected the distance between the two points.

This will impact the performance as higher weightage is given to variables with higher magnitude.

ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

# MACHINE LEARNING

## Effect of Normalizing

### NORMALISING:

$$x_i = \frac{x_i - \mu}{\sigma}$$

$\mu$  = Mean

$\sigma$  = Standard Deviation

here,

$$\mu_{age} = 33, \sigma_{age} = 6$$

$$\mu_{income} = 86000, \sigma_{income} = 20591.26$$

ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

### POST NORMALISING:

Euclidean distance between 1 and 2 =  
$$[(0.608 + 0.260)^2 + (-0.447 + 1.192)^2]^{(1/2)}$$
  
**= 1.14**

Distance is not biased towards the income variable anymore.

Similar weightage given to both the variables.

ID	Age	Income(rupees)
1	25	80,000
2	30	100,000
3	40	90,000
4	30	50,000
5	40	110,000

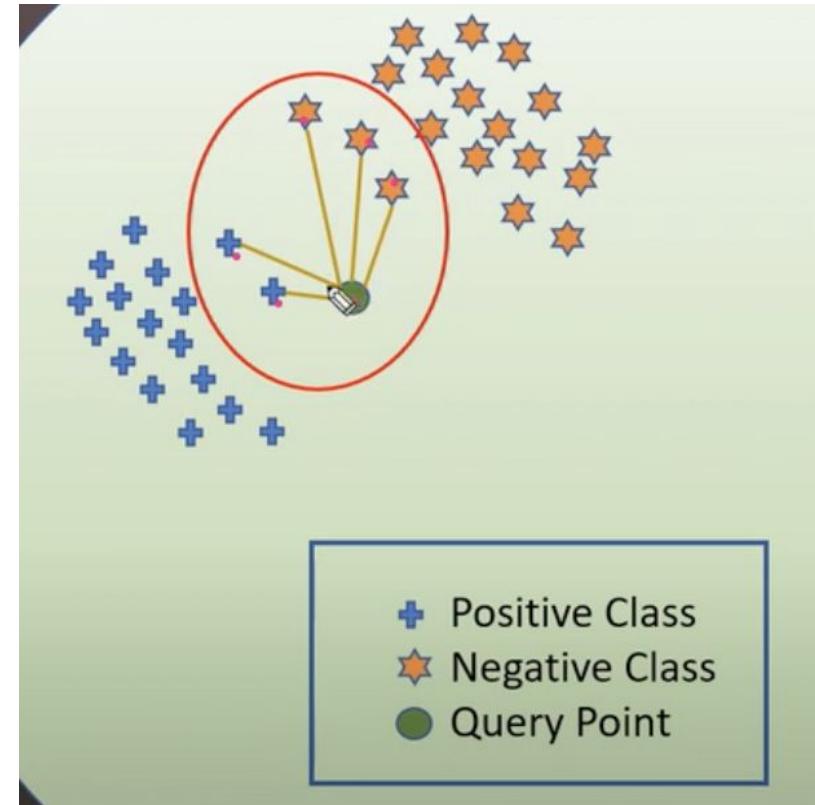
# MACHINE LEARNING

## Weighted KNN

$$weight = F(distance) = \frac{1}{distance}$$

Give Weights based on distance

Neighbour	True Label	Distance	Weight	Sum of Weights
$X_1$	Positive	0.1	10	13.3
$X_2$	Positive	0.3	3.3	
$X_3$	Negative	1	1	1.8
$X_4$	Negative	2	0.5	
$X_5$	Negative	3	0.3	

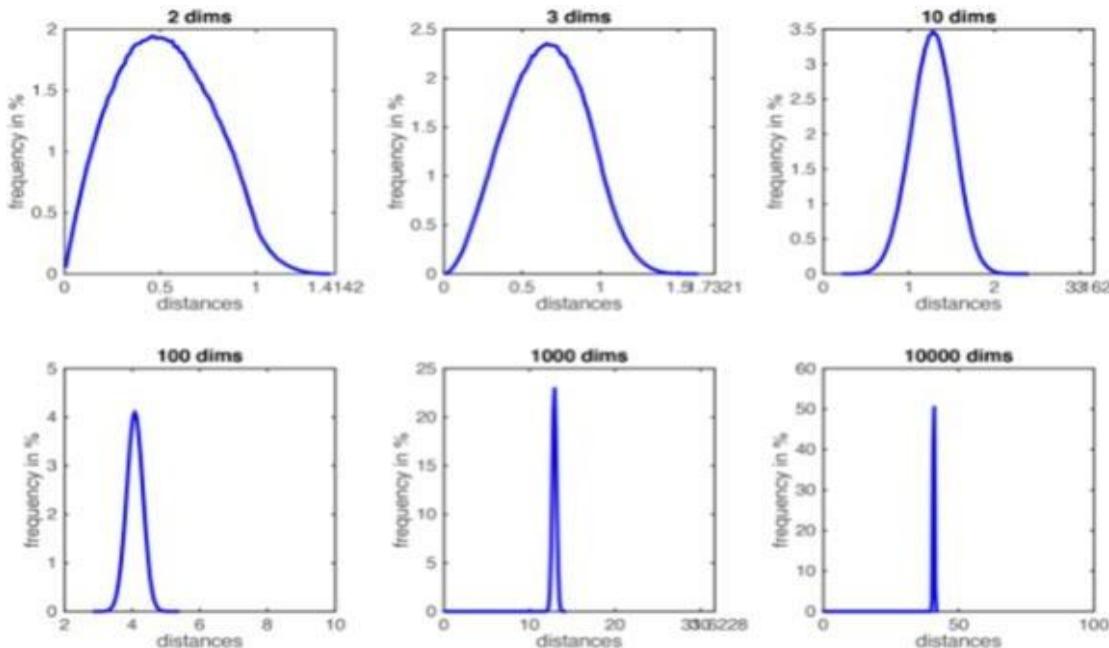


Predict Class labels based on the weighted-sum  
not on majority vote

- Basic kNN algorithm stores all examples
- Suppose we have  $n$  examples each of dimension  $d$
- $O(d)$  to compute distance to one example
- $O(nd)$  to find one nearest neighbor
- $O(knd)$  to find  $k$  closest examples
- Thus complexity is  $O(knd)$
- This is prohibitively expensive for large number of samples
- But we need large number of samples for kNN to work well!

The histogram plots show the distributions of all pairwise distances between randomly distributed points within d-dimensional unit squares.

As the **number of dimensions d grows, all distances concentrate within a very small range.**



One might think that one rescue could be to increase the number of training samples,  $n$ , until the nearest neighbors are truly close to the test point.

**How many data points would we need such that  $\ell$  becomes truly small?**

- Fix  $\ell=1/10=0.1$
- $n=k/\ell^d=k*10^d$ , which grows exponentially!
- For  $d>100$  we would need far more data points than there are electrons in the universe...

### The Curse Of Dimensionality – Dealing With It

- Assigning weights to the attributes when calculating distances. Let's say we're predicting the price of a house, we give higher weights to features like area and locality when compared to color.
- Iteratively, we leave out one of the attributes and test the algorithm. The exercise can then lead us to the best set of attributes.
- Dimensionality reduction using techniques like PCA.

## Handling Types of Attributes

### Handling Types Of Attributes

- In case of Boolean values, then **convert to 0 and 1.**
- **Non-binary characterizations:**
  - Natural order among the data, then convert to numerical values based on order.  
E.g. Educational attainment: HS, College, MS, PhD -> 1, 2, 3, 4.
  - No order and > 1 category, then convert to one hot encoding.  
E.g. Animals: Cat, Dog, Zebra -> (1, 0, 0), (0, 1, 0), (0, 0, 1)

	Cat	Dog	Zebra
	1	0	0
	0	1	0
	0	0	1

## Handling Types Of Attributes – Categorical Features

Let us say we have the following data:

- Education, Place and Gender are attributes
- Eligibility is the target
- Education has natural order that is High school <College <Ph.d
- Place needs to be one-hot encoded
- Gender is binary

Education	Place	Gender	Eligibility
High school	Banglore	M	Yes
College	Udupi	F	No
Phd	Mandya	M	No
Phd	Mandya	M	Yes
College	Udupi	F	No
High School	Mandya	M	No
Phd	Udupi	M	Yes
College	Banglore	F	Yes
Phd	Banglore	F	Yes
High School	Mandya	M	No

## Handling Types Of Attributes – Categorical Features

Converted data:

Education	Place	Gender	Eligibility
High school	Banglore	M	Yes
College	Udupi	F	No
Phd	Mandya	M	No
Phd	Mandya	M	Yes
College	Udupi	F	No
High School	Mandya	M	No
Phd	Udupi	M	Yes
College	Banglore	F	Yes
Phd	Banglore	F	Yes
High School	Mandya	M	No

Education	P1	P2	Gender	Eligibility
1	0	0	1	Yes
2	0	1	0	No
3	1	0	1	No
3	1	0	1	Yes
2	0	1	0	No
1	1	0	1	No
3	0	1	1	Yes
2	0	0	0	Yes
3	0	0	0	Yes
1	1	0	1	No

## Handling Types Of Attributes – Categorical Features

Converted data:

- In Education 1,2,3 represents high school college and Phd respectively.
- Place which had three types of value is broken into two attribute p1 and p2 where  $(p1,p2)=(0,0)$  represents bangalore,  $(p1,p2)=(0,1)$  represents Udupi and  $(p1,p2)=(1,0)$  represents Mandya.
- Since Gender is binary here , we can give 0 to Female and 1 to Male or the other way too.

Education	P1	P2	Gender	Eligibility
1	0	0	1	Yes
2	0	1	0	No
3	1	0	1	No
3	1	0	1	Yes
2	0	1	0	No
1	1	0	1	No
3	0	1	1	Yes
2	0	0	0	Yes
3	0	0	0	Yes
1	1	0	1	No

- <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>
- <https://www.geeksforgeeks.org/instance-based-learning/>
- <https://www.cs.ucdavis.edu/~vemuri/classes/ecs271/ch8.pdf>
- <https://www.analyticsvidhya.com/blog/2021/01/a-quick-introduction-to-k-nearest-neighbor-knn-classification-using-python/>
- <https://datagy.io/python-knn/>
- [https://www.bogotobogo.com/python/scikit\\_learn/scikit\\_machine\\_learning\\_k-NN\\_k-nearest-neighbors-algorithm.php](https://www.bogotobogo.com/python/scikit_learn/scikit_machine_learning_k-NN_k-nearest-neighbors-algorithm.php)
- <https://www.analyticsvidhya.com/blog/2015/08/learning-concept-knn-algorithms-programming/>



**PES**  
UNIVERSITY

# **UE21CS352A Machine Learning**

---

**Surabhi Narayan**

Department of Computer Science & Engineering

TA : Arvin Nooli

# Ensemble Learning

---

**Surabhi Narayan**

Department of Computer Science & Engineering

# Machine Learning

## Acknowledgement

---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

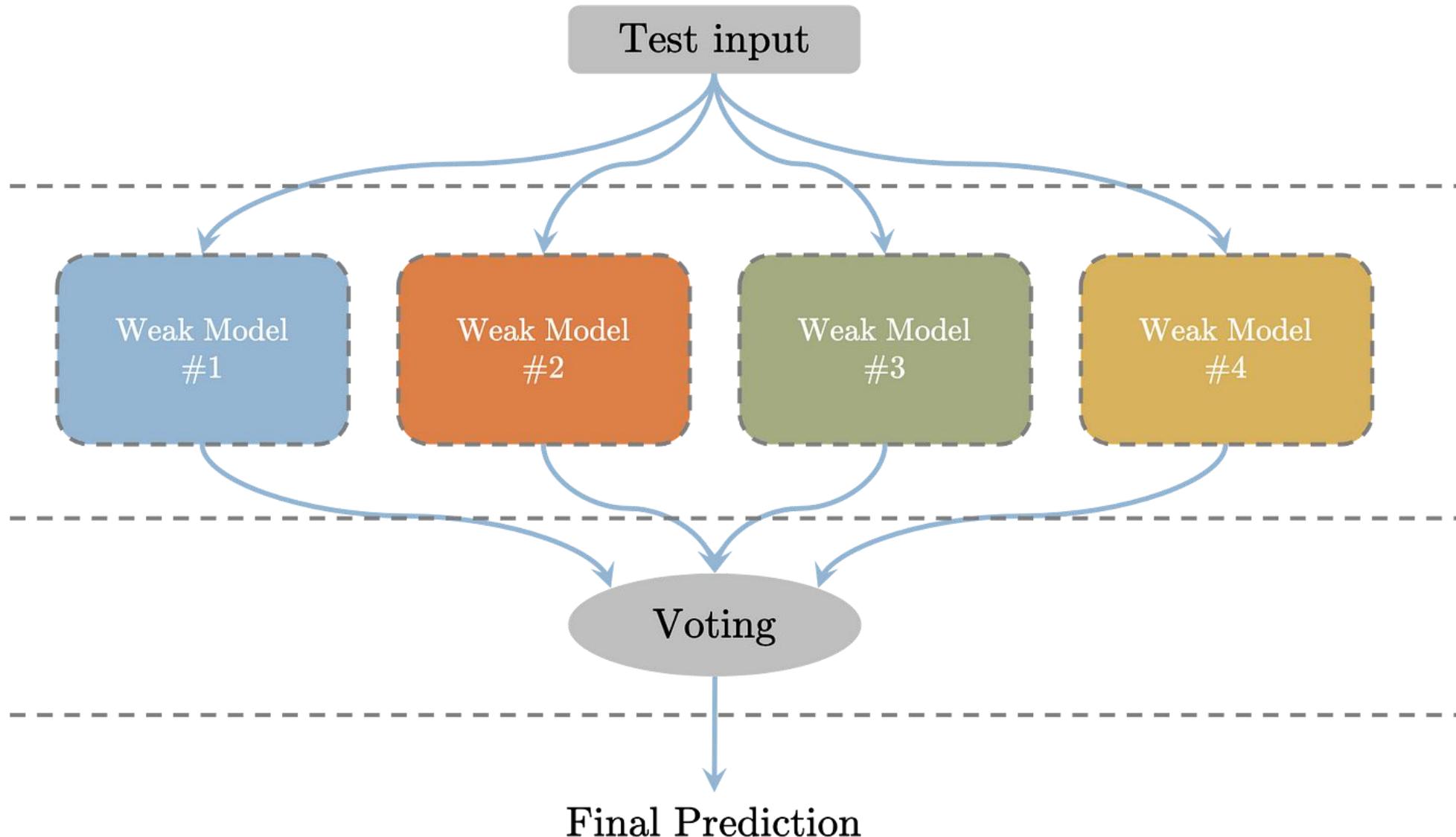
# Ensemble Learning

---

Ensemble learning combines multiple learners to improve predictive performance of models to facilitate accurate and improved decisions



# Ensemble Learning



## Background:

Let's say you moved to a new place and want to dine out. How do you find a good place?

**Solution 1:** Find a food critic who is really good at his/her work and see if he/she has any recommendations for the restaurants in your area

**Solution 2:** Use Google and randomly look at **one user's review** for a couple of restaurants.

**Solution 3:** Ask strangers/local for their opinion and blindly trust it

Let us analyze each of the above-mentioned solutions.



# Ensemble Learning

---

Solution 1:

- Food critics are in general much accurate.
- It is difficult to find a food critic
- Maybe the food critic you found was a strict vegetarian, and you are not. In that case, the recommendations from the food critic will be biased.

Solution 2:

- On the other hand, picking up a random person's star rating for a restaurant on the internet is Much less accurate
- But Easier to find

Solution 3:

- Here we are relying on one random stranger to ask for their opinion which may or may not sit with your taste which again can be biased

Solution:

- Collectively look at multiple reviewers for a couple of restaurants and average out their rating here it can be just the right amount of accuracy you need
- Easier to find over the internet
- Much less biased, since the users who have rated the restaurants come from various backgrounds.

## Why ensemble?

- To improvise on the stability and predictive power of the model.
- To make more accurate predictions than any individual model.
- They are among the most powerful techniques in machine learning, often outperforming other methods.
- High Confidence — When most of the learners/models predict the same class which leads to high confidence.
- It has been adopted in response to issues resulting from limited datasets

## Applications

- KDD cup : Network intrusion detection, molecular bio-activity, customer relationship management, educational data mining
- Netflix competition ([link](#))
- Viola Jones – object detection framework uses adaboost
- Pose-invariant face recognition
- Object tracking – ensemble tracking

# Ensemble Learning

Aspect	Weak Learners	Strong Learners
Definition	Models that perform slightly better than random guessing (accuracy just above 50% for binary classification).	Models that have high predictive accuracy on their own.
Complexity	Simple models, often with high bias and low variance (e.g., decision stumps, small trees).	Complex models, often with low bias and high variance (e.g., large decision trees, neural networks).
Training Process	Typically easy and fast to train due to their simplicity.	Usually require more time and resources to train due to their complexity.
Performance	Individually, they may not perform well but can contribute to a strong ensemble model.	Perform well individually and can often be used as standalone models.
Use in Ensembles	Often used in boosting methods (e.g., AdaBoost), where their errors are iteratively reduced.	May be used in bagging methods (e.g., Random Forests) or as a final model in stacking.
Overfitting	Less prone to overfitting due to their simplicity, but less accurate.	More prone to overfitting, especially on small or noisy datasets.
Bias-Variance Tradeoff	High bias, low variance.	Low bias, high variance.
Example Algorithms	Decision Stumps, Small Trees, Naive Bayes.	Large Decision Trees, SVMs, Neural Networks.
Role in Ensemble Learning	Used to form the basis of strong learners in boosting algorithms.	Used as the final model in stacking or as components in bagging.

How are multiple learners modelled?

By using:

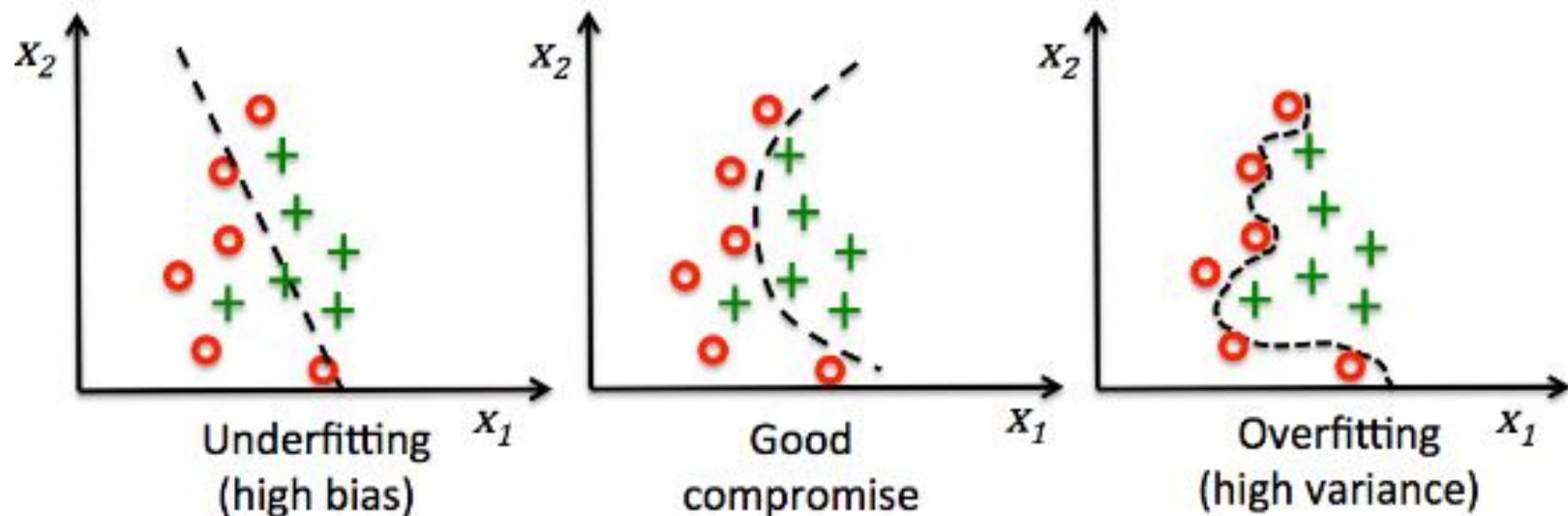
- Different Algorithms
- Different Hyperparameters of the same algorithm
- Different subsets of the training data
- Different Representations

Combine Predictions from multiple learners

# Ensemble Learning

Major Sources of Error in Machine Learning models:

- Noise
- Bias
- Variance



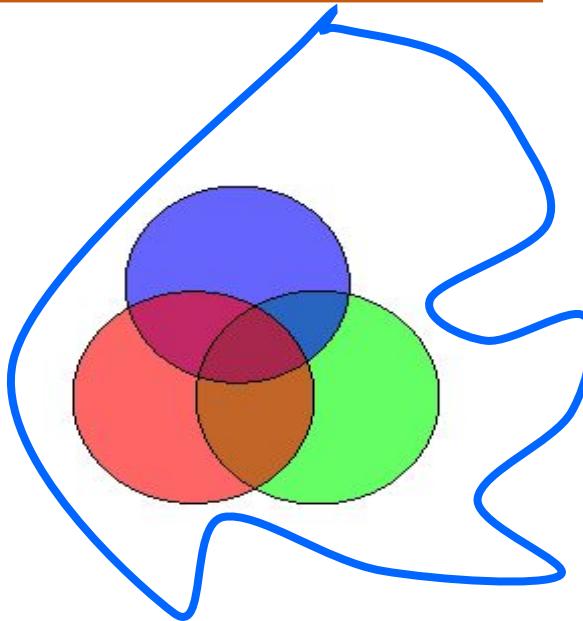
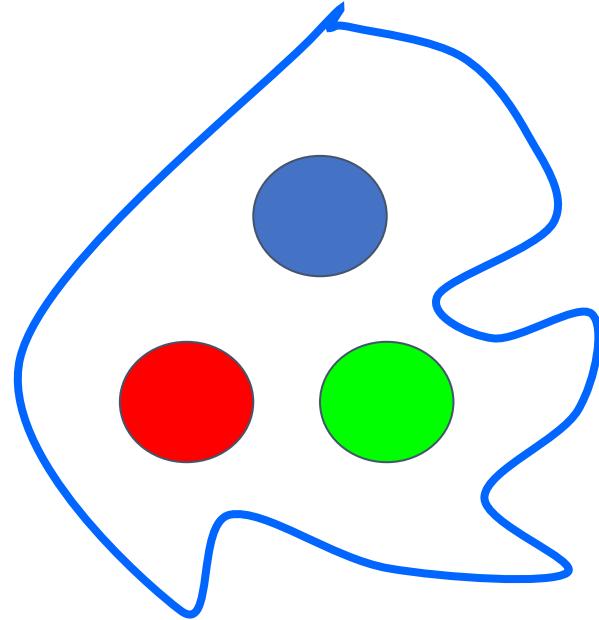
- Goal is to develop a combination of models that makes mostly correct predictions with high confidence.
- In fact even if the individual learners have high bias the new combined learner will have a low bias
- Basic models perform not so well by themselves either because they have a high bias (low degree of freedom) or because they have too much variance to be robust (high degree of freedom)

- Let's assume we have n learners in a binary classification problem
- If all of them have an accuracy of A and predict the same class for a given instance, what would your confident be on the prediction.

$$C = [1 - \{1 - A\}^n]$$

# Ensemble Learning

---

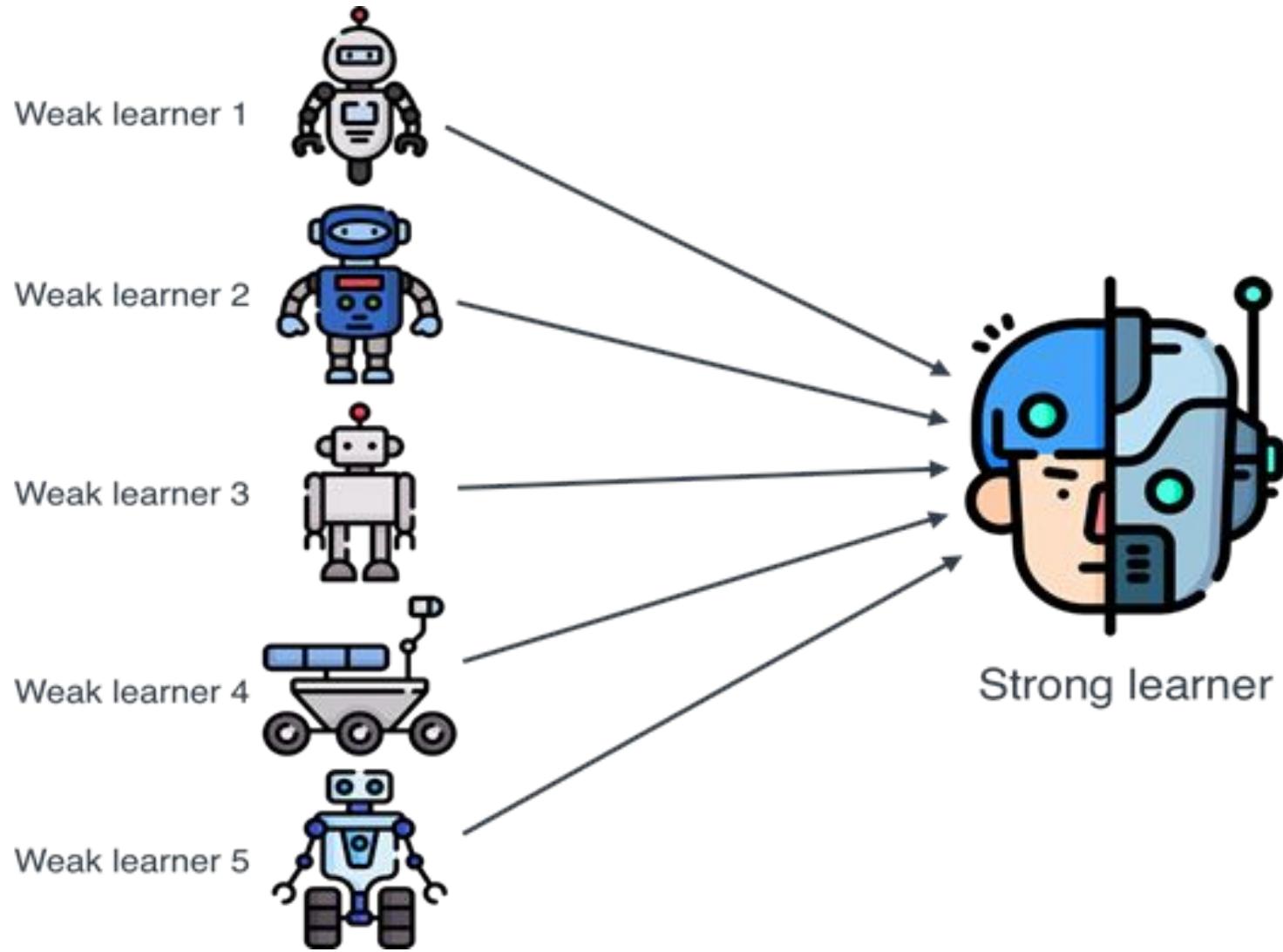


- There is no point in combining learners that always make similar decisions.
- The aim is to be able to find a set of *diverse* learners who differ in their decisions so that they complement each other
- There cannot be a gain in overall success unless the learners are accurate, at least in their domain of expertise

## Diversity vs. Accuracy

- Base learners should be reasonably accurate and hence need not be optimized separately for best accuracy
- Base learners should be diverse, accurate on different instances, specializing in sub domains of the problem.

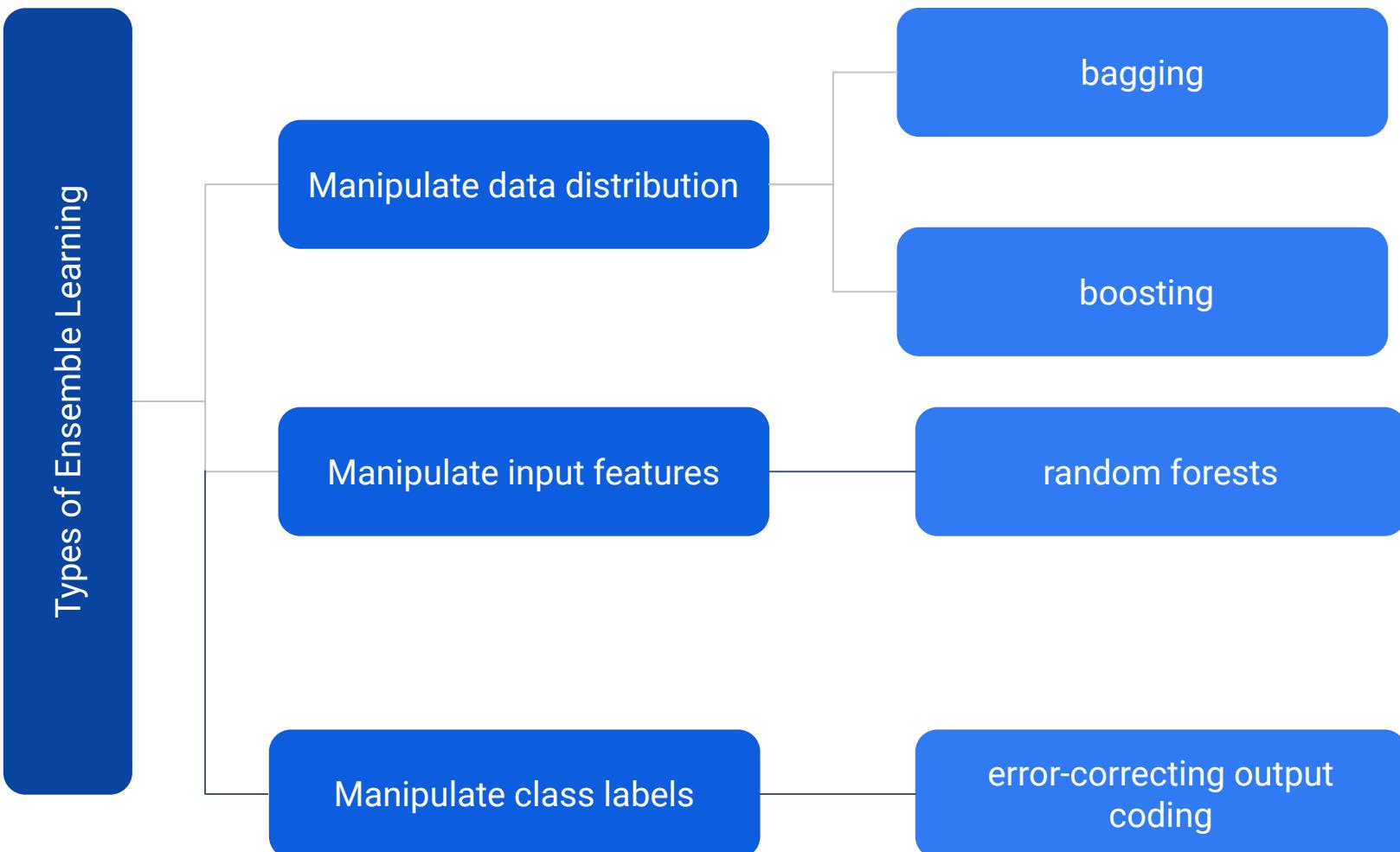
# Ensemble Learning



Combine Predictions:

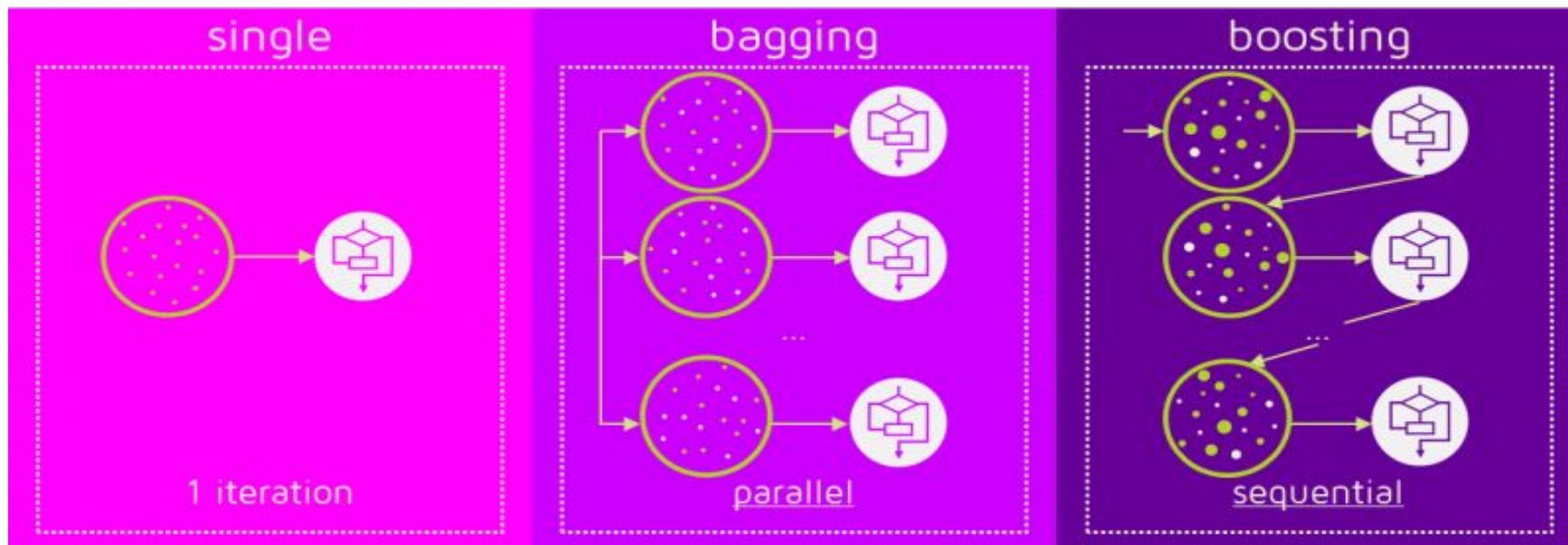
- Non-weighted
- Weighted

# Ensemble Learning



## Types of Ensemble Methods:

- Parallel Ensemble Methods (Bagging)
- Sequential Ensemble Methods (Boosting)



## Parallel Ensemble - Voting

- The simplest way to combine multiple classifiers is by *voting*, which corresponds to taking a linear combination of the learners
- Simple Voting : all learners are given equal weight
- Other possibilities: weighted voting, median, Minimum, Maximum etc

Hansen and Salamon (1990) have shown that given independent two class classifiers with success probability higher than  $\frac{1}{2}$  (better than random guessing), by taking a majority vote, the accuracy increases as the number of voting classifiers increases.

Let us assume that  $d_j$  are iid with expected value  $E[d_j]$  and variance  $\text{Var}(d_j)$ , then when we take a simple average with  $w_j = 1/L$ , the expected value and variance of the output are

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L E[d_j] = E[d_j]$$

$$\text{Var}(y) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2} \text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \text{Var}(d_j) = \frac{1}{L} \text{Var}(d_j)$$

# Ensemble Learning

---

- Given a set of  $n$  independent observations  $X_1, \dots, X_n$ , each with variance  $\sigma^2$ .
- So **variance of the sample mean** to the true mean which gives the std. error:

$$\begin{aligned}\text{Var}(\bar{X}) &= \text{Var}\left(\sum_{i=1}^n \frac{X_i}{n}\right) = \left(\frac{1}{n}\right)^2 \text{Var}\left(\sum_{i=1}^n X_i\right) \\ &= \left(\frac{1}{n}\right)^2 \sum_{i=1}^n \text{Var}(X_i) = \left(\frac{1}{n}\right)^2 \sum_{i=1}^n \sigma^2 = \left(\frac{1}{n}\right)^2 n\sigma^2 = \frac{\sigma^2}{n}\end{aligned}$$

**Properties of Variance:** The variance of a constant times a random variable is that constant squared times the variance of the random variable.

- It clearly shows that averaging a set of observation( ie taking their mean) reduces the variance as  $\sigma^2/n < \sigma^2$ .

The expected value does not change, so the bias does not change. But variance, and therefore mean square error, decreases as the number of independent voters,  $L$ , increases.

In general

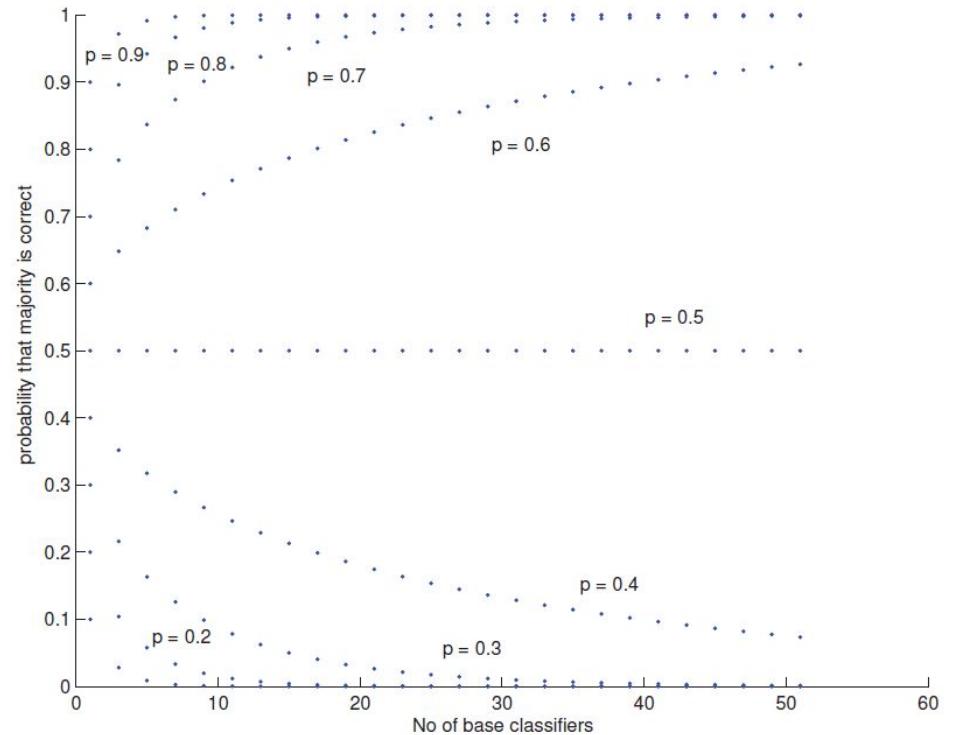
$$\text{Var}(y) = \frac{1}{L^2} \text{Var} \left( \sum_j d_j \right) = \frac{1}{L^2} \left[ \sum_j \text{Var}(d_j) + 2 \sum_j \sum_{i < j} \text{Cov}(d_j, d_i) \right]$$

If learners are positively correlated, variance (and error) increase. We can thus view using different algorithms and input features as efforts to decrease, if not completely eliminate, the positive correlation.

# Ensemble Learning

If each base-learner is iid and correct with probability  $p > 1/2$ , the probability that a majority vote over  $L$  classifiers gives the correct classification is given by

$$P(X \geq \lfloor L/2 \rfloor + 1) = \sum_{i=\lfloor L/2 \rfloor + 1}^L \binom{L}{i} p^i (1-p)^{L-i}$$



**Figure 17.6** Probability that a majority vote is correct as a function of the number of base-learners for different  $p$ . The probability increases only for  $p > 0.5$ .

Assume we have ( $L=10$ ) classifiers, probability of the classifier being correct is 0.6

Case 1: If all classifiers are identical

then the probability of the ensemble being correct is also 0.6

Case 2: If classifiers are independent

then the probability that a majority vote over  $L$  classifiers gives the correct answer is given by

$$P(X \geq 6) = \sum_{k=6}^{10} \binom{10}{k} 0.6^k (0.4)^{10-k}$$

= 0.6916 which is greater than 0.6

$$P(X \geq [L/2] + 1) = \sum_{i=[L/2]+1}^L \binom{L}{i} p^i (1-p)^{L-i}$$

# Ensemble Learning

How can an ensemble method improve a classifier's performance?

“ Assume we have 25 binary classifiers ”

Each has error rate:  $\varepsilon = 0.35$

**1. If all 25 classifiers are identical:**

They will vote the same way on each test instance

Ensemble error rate:  $\varepsilon = 0.35$

**2. If all 25 classifiers are independent** (errors are uncorrelated): Ensemble method only makes a wrong prediction if more than half of the base classifiers predict incorrectly.

$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

6% much less than 35%

- **Base Classifier:** This term is used to indicate the base component of a multiple classifier system. In other words, a multiple classifier system is made up by a set of base classifiers.



**THANK YOU**

---

**Surabhi Narayan**

Department of Computer Science &Engineering

**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
UNIVERSITY

# **UE21CS352A Machine Learning**

---

**Surabhi Narayan**

Department of Computer Science & Engineering

TA : Arvin Nooli

# Machine Learning

## Acknowledgement

---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

## BAGGING

Bootstrap Aggregating

- If we split the data in random different ways , decision trees give different results which basically results in High Variance
- BAGGING: **Bootstrap aggregating** is a method that results in low variance

# Ensemble Learning- Bagging

## How Bagging Works?

(A) bagging

**step 1**

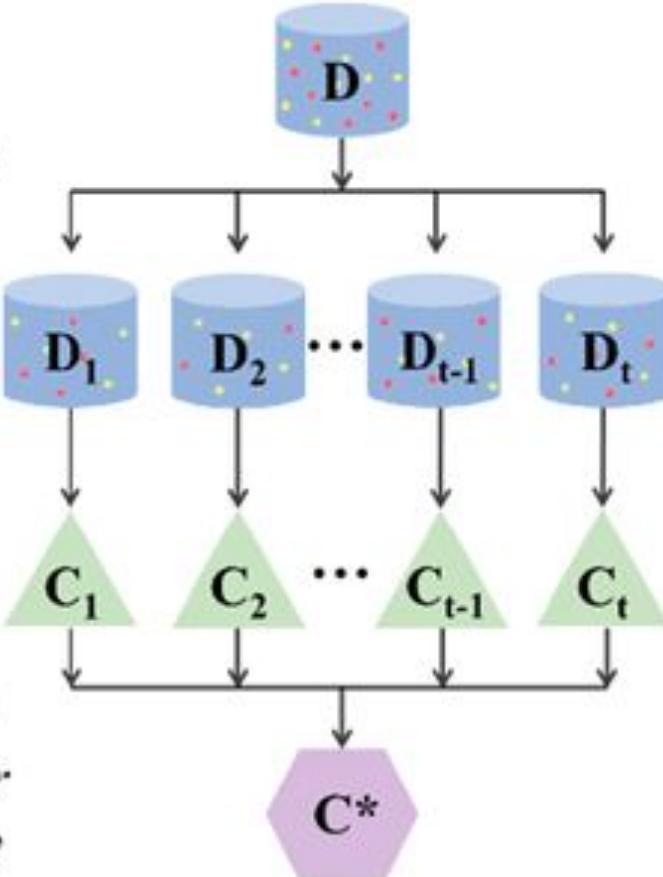
*create multiple data sets through random sampling with replacement*

**step 2**

*build multiple learners in parallel*

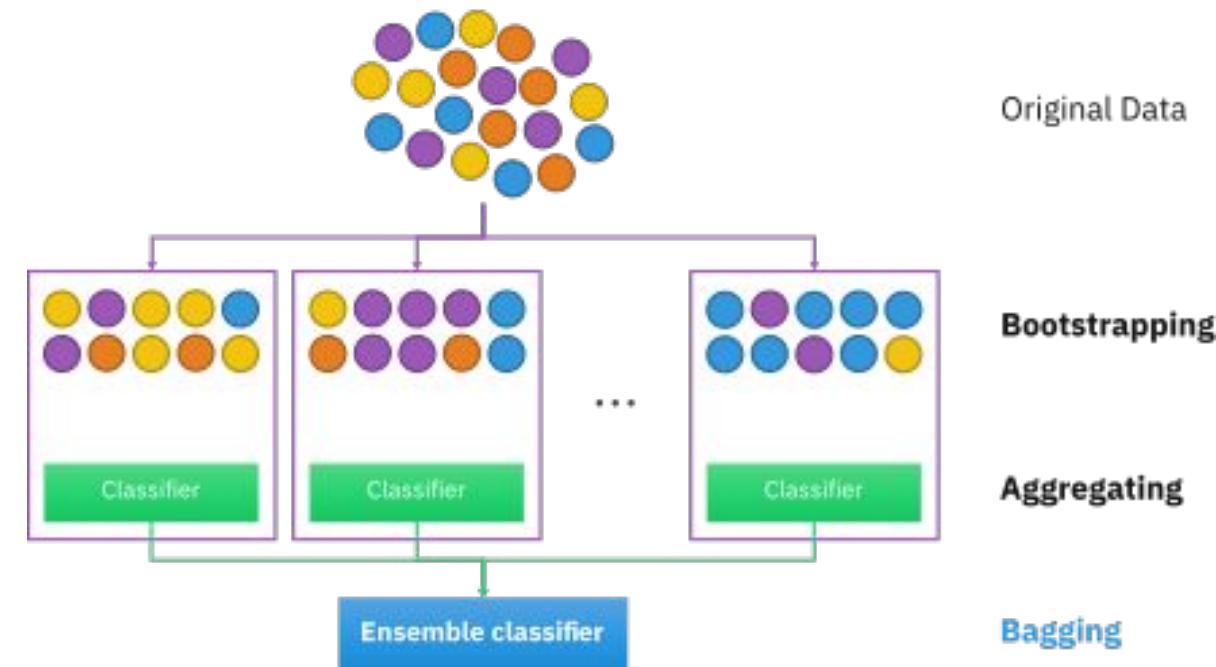
**step 3**

*combine all learners using an averaging or majority-vote strategy*



# Ensemble Learning- Bagging

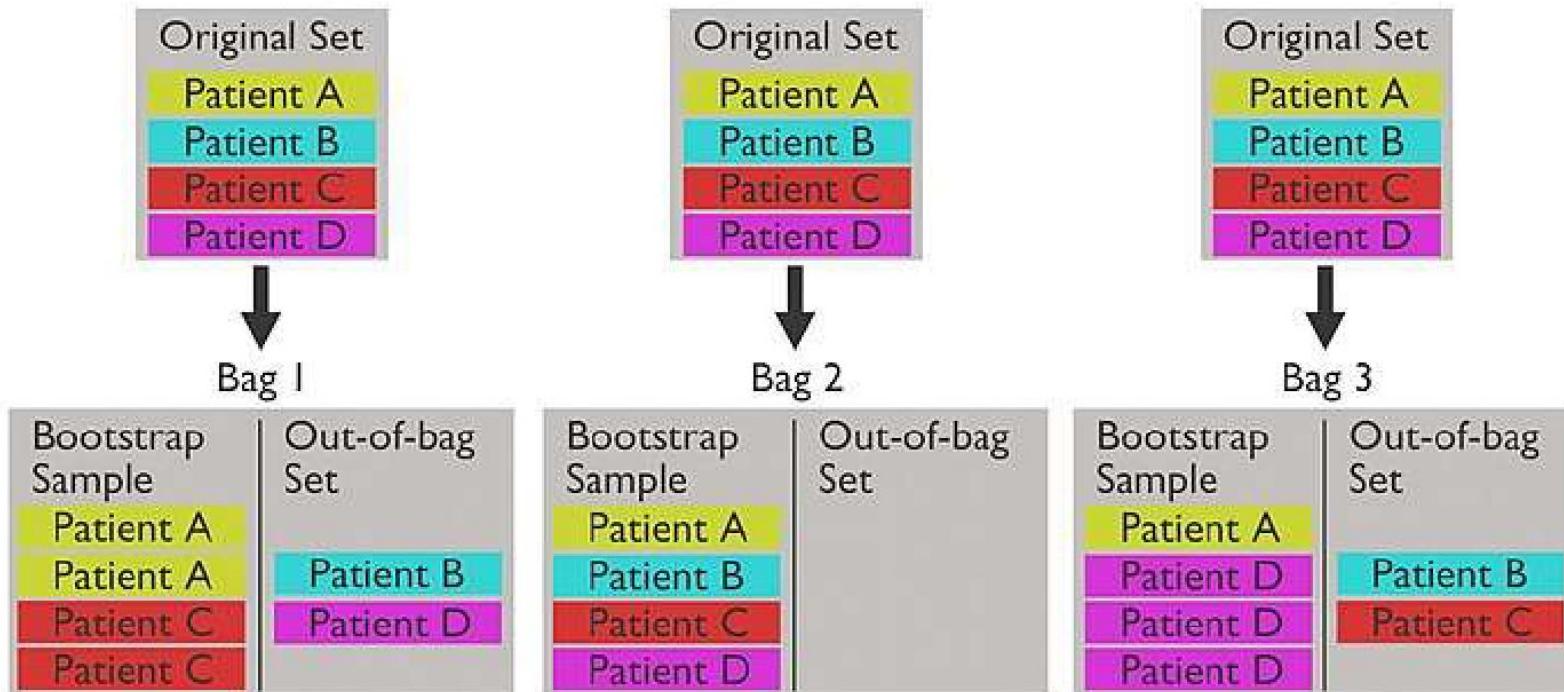
- Multiple subsets are created from the original dataset, selecting observations with replacement.
- A base learner/model (weak learner) is created on each of these subsets.
- The models run in parallel and are independent of each other.
- The final predictions are determined by combining the predictions from all the models. (Voting or averaging)



# Ensemble Learning

## Out of the Bag Data

When bootstrap aggregating is performed, two independent sets are created. One set, the bootstrap sample, is the data chosen to be "in-the-bag" by sampling with replacement. The out-of-bag set is all data not chosen in the sampling process.



## Calculating Out-of-Bag Error

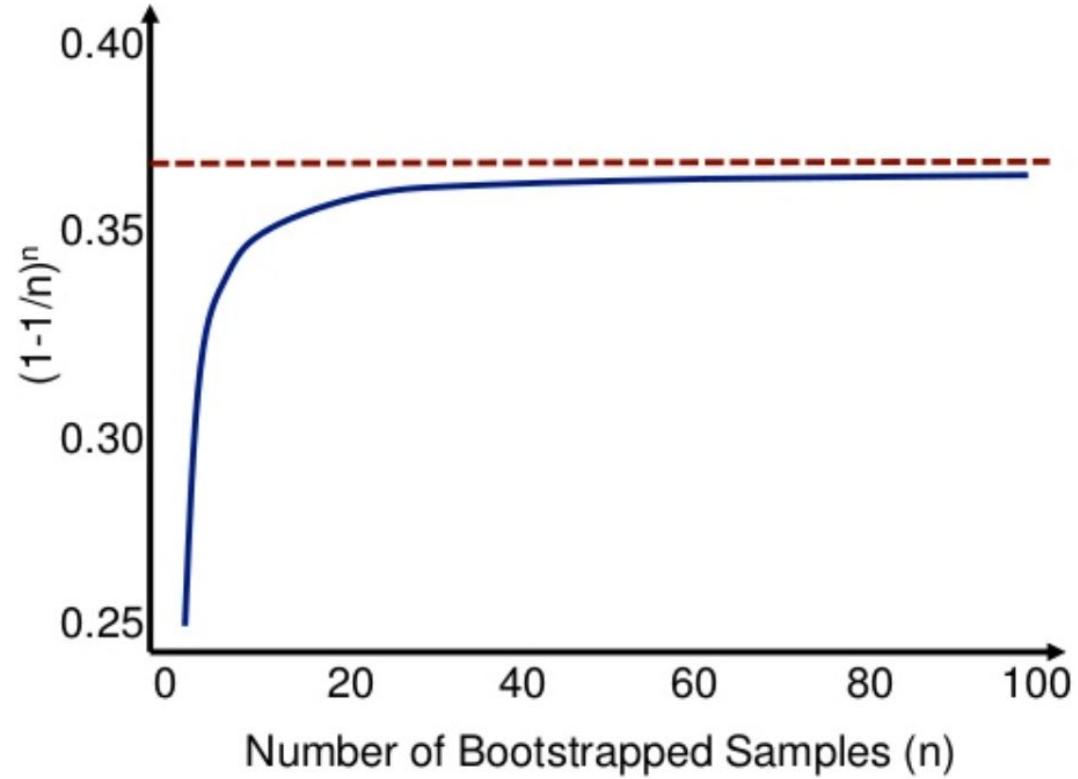
Since each out-of-bag set is not used to train the model, it is a good test for the performance of the model.

The specific calculation of OOB error depends on the implementation of the model, but a general calculation is as follows.

- Find all models that are not trained by the OOB instance.
- Take the majority vote of these models' result for the OOB instance, compared to the true value of the OOB instance.
- Compile the OOB error for all instances in the OOB dataset.

# Ensemble Learning

## Out of the Bag Data



→  $P(\text{Data record } x \text{ not being selected}) = (1 - 1/N)^N$

# Ensemble Learning

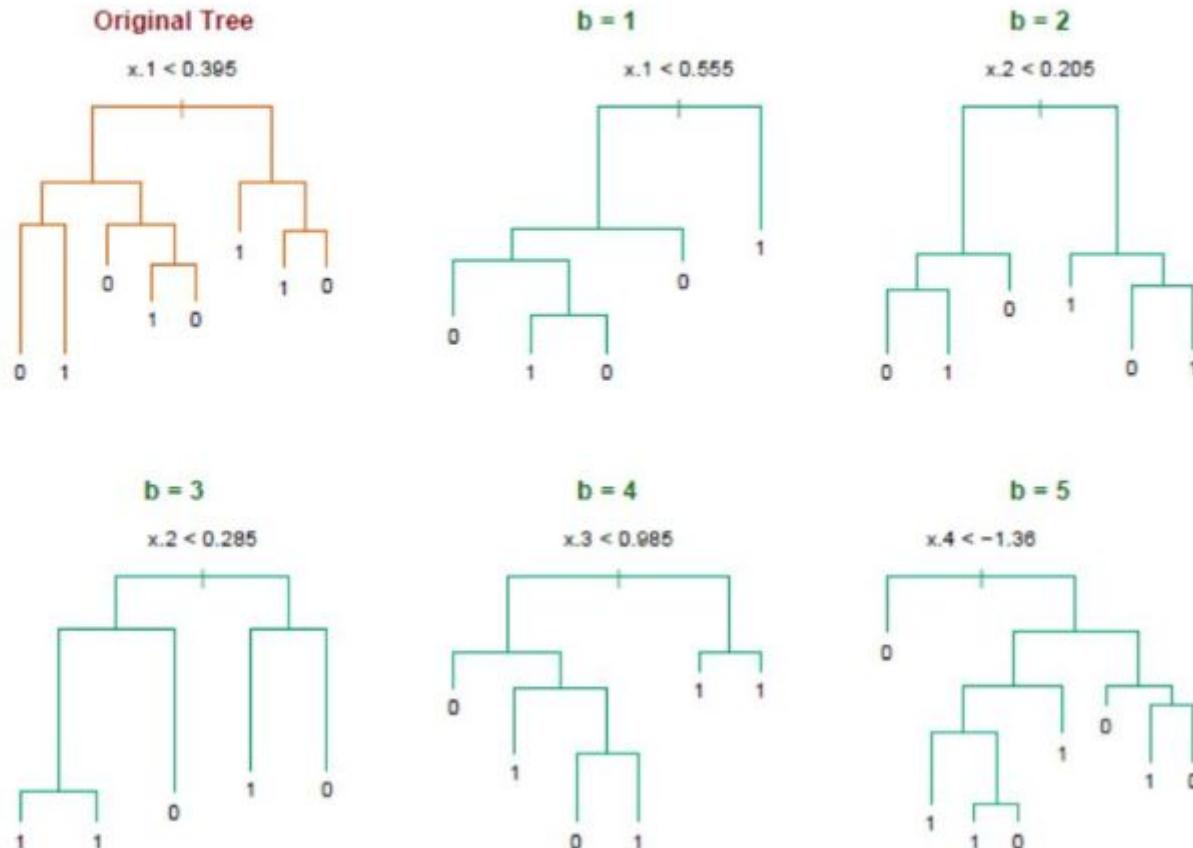
---

Theoretically, with the quite big data set and the number of sampling, it is expected that out-of-bag error will be calculated on 36% of the training set. To prove this, consider that our training set has  $n$  samples. Then, the probability of selecting one particular sample from the training set is  $\frac{1}{n}$ .

Similarly, the probability of not selecting one particular sample is  $1 - \frac{1}{n}$ . Since we select the bootstrap samples with replacement, the probability of one particular sample not being selected  $n$  times is equal to  $(1 - \frac{1}{n})^n$ . Now, if the number  $n$  is pretty big or if it tends to infinity, we'll get a limit below:

$$\lim_{n \rightarrow \infty} \left(1 - \frac{1}{n}\right)^n = \lim_{n \rightarrow \infty} \left(1 + \frac{-1}{n}\right)^n = e^{-1} \approx 0.36 \quad (1)$$

## Bagging decision trees



Hastie et al., "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

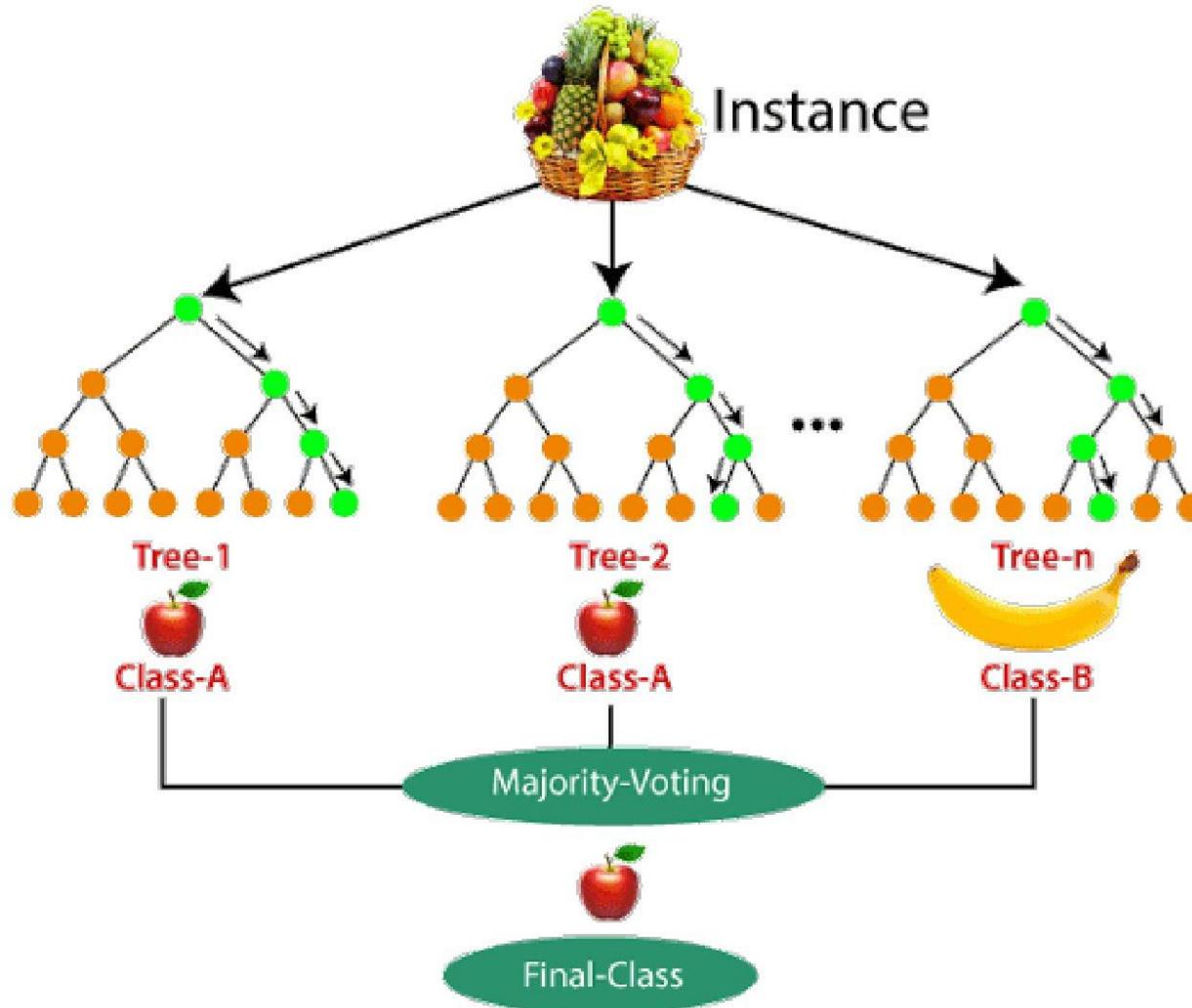
Problems with bagging:

Suppose there is one very strong predictor in the dataset , along with a number of moderately strong predictors

Then all bagged trees will select the strong predictors at top of the tree and the all trees will look similar

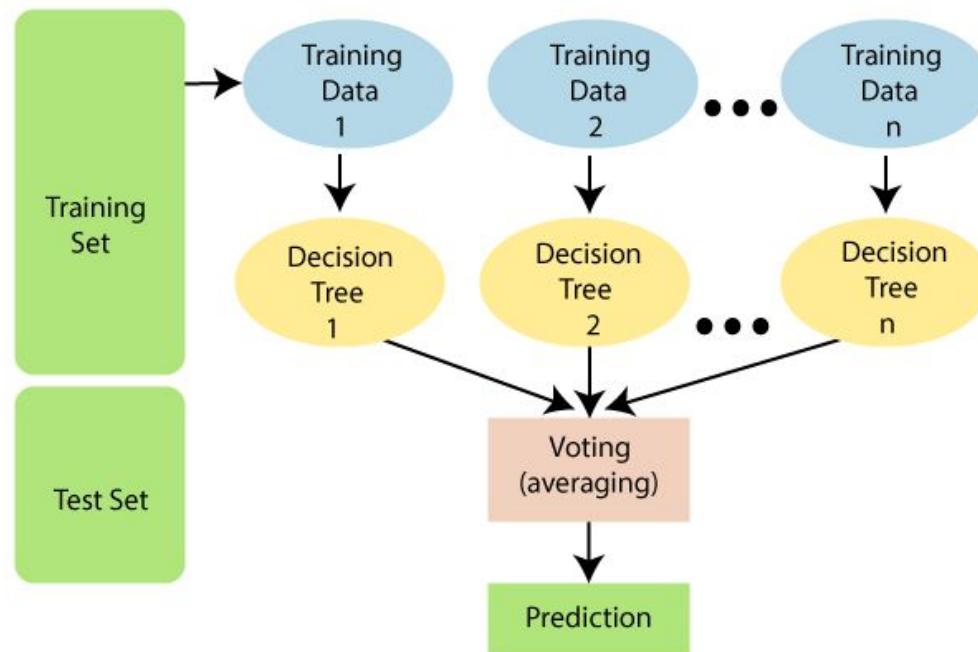
**How do we avoid this?**

# Ensemble Learning – Random Forest



# Ensemble Learning – Random Forest

- **Random forest** is a tree learning technique commonly-used machine learning algorithm that combines the output of multiple decision trees to reach a single result.
- Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



## Assumptions for Random Forest:

Since random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output.

1. There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
  - to ensure the classifier makes predictions based on meaningful data rather than random guesses.
2. The predictions from each tree must have very low correlations.
  - ensure the diversity of the model, reducing the risk of overfitting and improving overall model accuracy.

# Ensemble Learning – Random Forest

## Why use Random Forest?

### 1. Less Training Time Compared to Other Algorithms

- The approach allows it to be relatively fast in training because each decision tree is built using a random subset of the data and features.
- The parallel nature of tree construction and the use of random subsets allow for efficient use of computational resources.
- It can handle large datasets faster than some other complex models like neural networks.



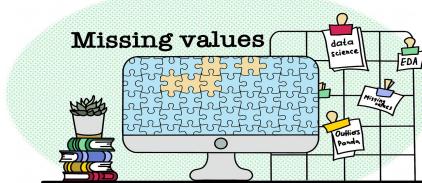
### 2. High Accuracy and Efficiency with Large Datasets

- By combining predictions from multiple decision trees, Random Forest achieves better generalization and higher accuracy on unseen data.
- Its ability to process different parts of a large dataset simultaneously results in faster training and more reliable outcomes, making it ideal for big data applications.



### 3. Maintaining Accuracy with Missing Data

- Random Forests are inherently good at handling missing values in the dataset. As each decision tree is trained on a random subset of the data and features, missing data in one subset does not significantly impact the overall performance of the model.
- The algorithm can ignore missing values during tree construction, ensuring that the prediction accuracy remains relatively stable even when a significant portion of the data is missing.



1. **Create Bootstrapped Dataset:** Sampling with replacement generates diverse datasets, ensuring each decision tree is trained on a unique subset, enhancing robustness and generalization.
2. **Create Decision Trees Using a Random Subset of Features:** Using random features for each tree reduces correlation among trees, promoting model diversity and preventing overfitting.
3. **Compute the Out-of-Bag Error:** Evaluating model performance on unused samples (OOB) provides an unbiased estimate of the Random Forest's predictive accuracy.
4. **Run the Samples Through the Random Forest and Classify Based on Voting:** Aggregating predictions through majority voting increases the overall accuracy and reliability of the model by leveraging the collective decision of all trees.

# Ensemble Learning – Random Forest

## Original Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

# Ensemble Learning – Random Forest

## Original Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

## Bootstrapped Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

# Ensemble Learning – Random Forest

## Original Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

## Bootstrapped Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No

# Ensemble Learning – Random Forest

## Original Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

## Bootstrapped Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes

# Ensemble Learning – Random Forest

## Original Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

## Bootstrapped Dataset

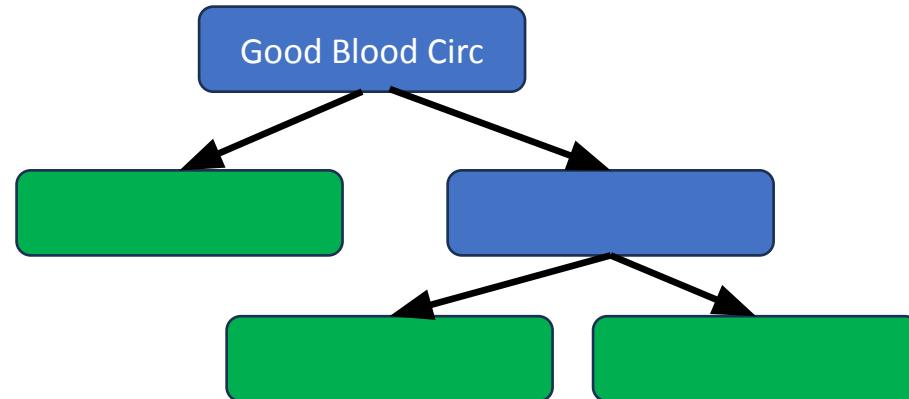
Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

# Ensemble Learning – Random Forest

## Bootstrapped Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random subset of features

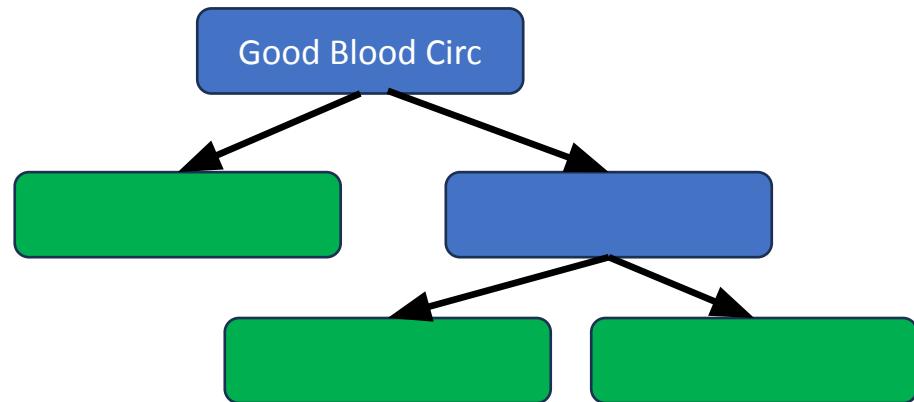


# Ensemble Learning – Random Forest

## Bootstrapped Dataset

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random subset of features



# Ensemble Learning – Random Forest

---

- Remember, in bootstrapping we sample with replacement, and therefore **not all observations are used for each bootstrap sample**. On average 1/3 of them are not used!
- We call them out-of-bag samples (OOB)
- We can predict the response for the  $i$ -th observation using each of the trees in which that observation was OOB and do this for  $n$  observations
- Calculate overall OOB MSE or classification error

## Advantages of Random Forest

- No need for pruning trees
- Accuracy and variable importance generated automatically
- Overfitting is not a problem
- Not very sensitive to outliers in training data
- Easy to set parameters
- Good performance

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

## Missing Data

1. Missing Data in the training sample.
2. Missing data in the test sample

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Did not have heart disease

No is the most common value for Blocked Arteries

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

Did not have heart disease

No is the most common value for Blocked Arteries

Initial Guess

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	167.5	No

Did not have heart disease

Weight is numeric

Initial Guess (Median)

# Ensemble Learning – Random Forest

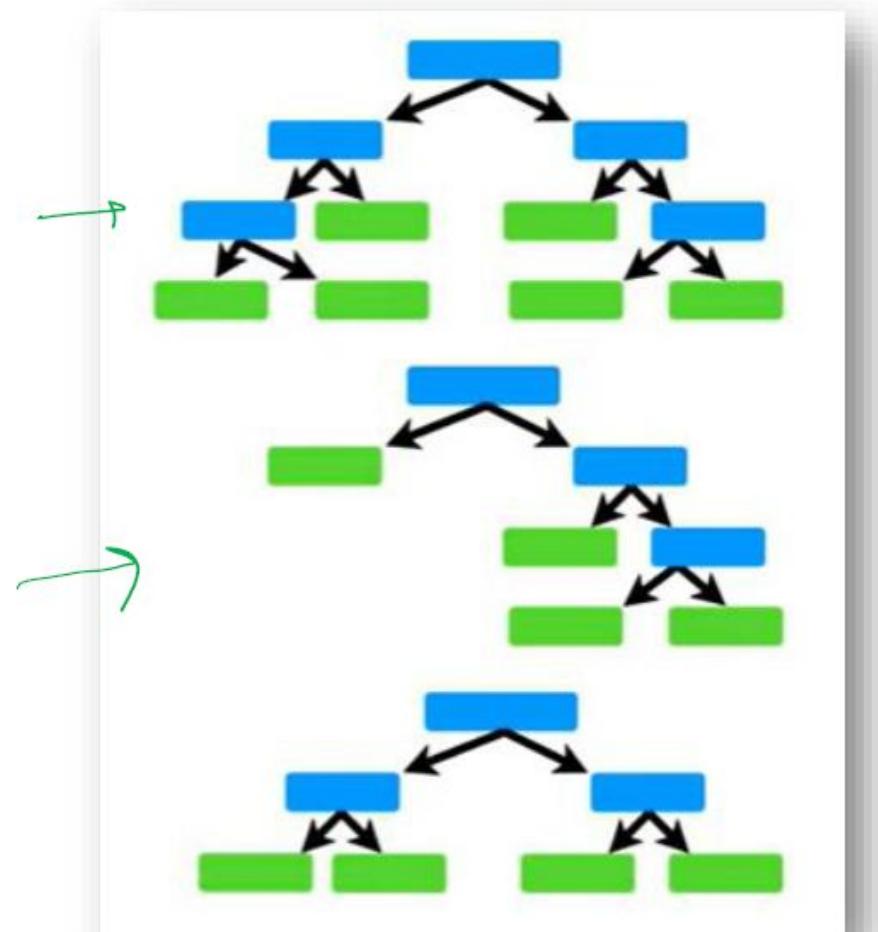
Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

Determine which samples are similar to the one with missing data

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

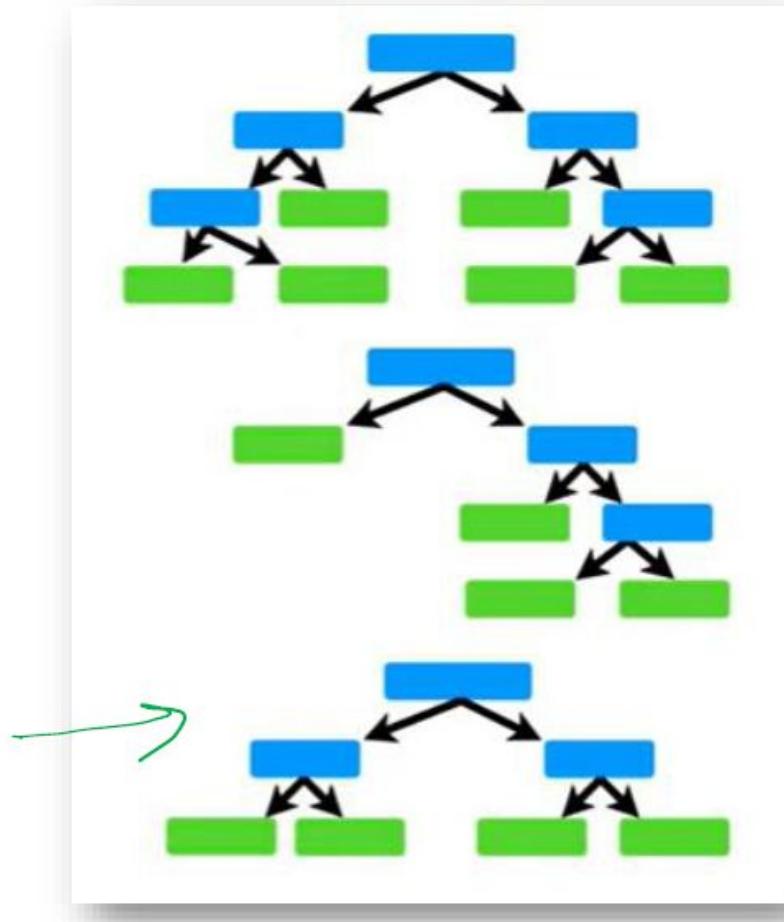
Step 1: Build a random forest



# Ensemble Learning – Random Forest

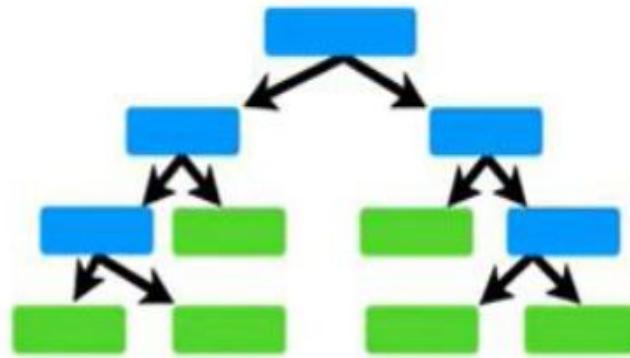
Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

Step 2: Run all of the data down the tree



# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

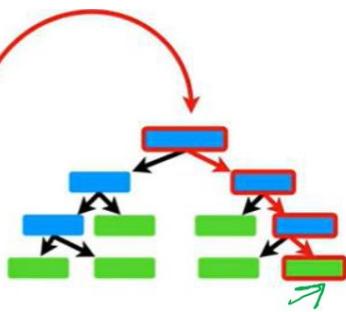


We'll start by running all of the data down the first tree.

# Ensemble Learning – Random Forest

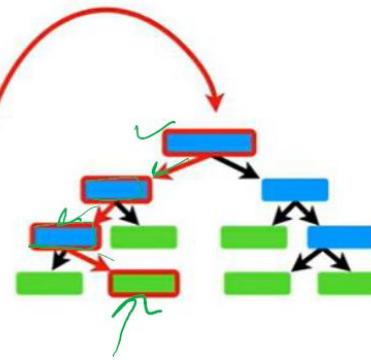
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



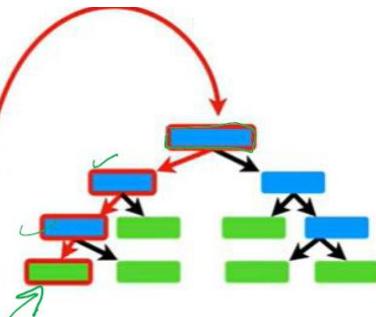
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



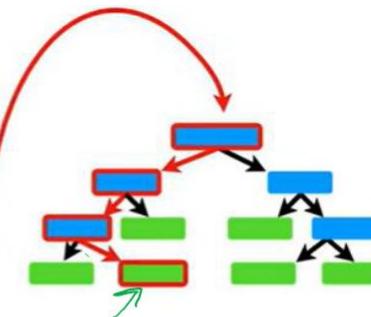
Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



Filled-in Missing Values

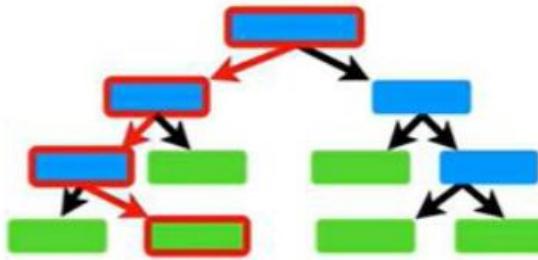
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



# Ensemble Learning – Random Forest

Filled-in Missing Values

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	<b>167.5</b>	No



Notice that Sample 3 and Sample 4 both ended up at the same leaf node.

*That means they are similar*

# Ensemble Learning – Random Forest

We keep track of similar samples using a “Proximity Matrix”

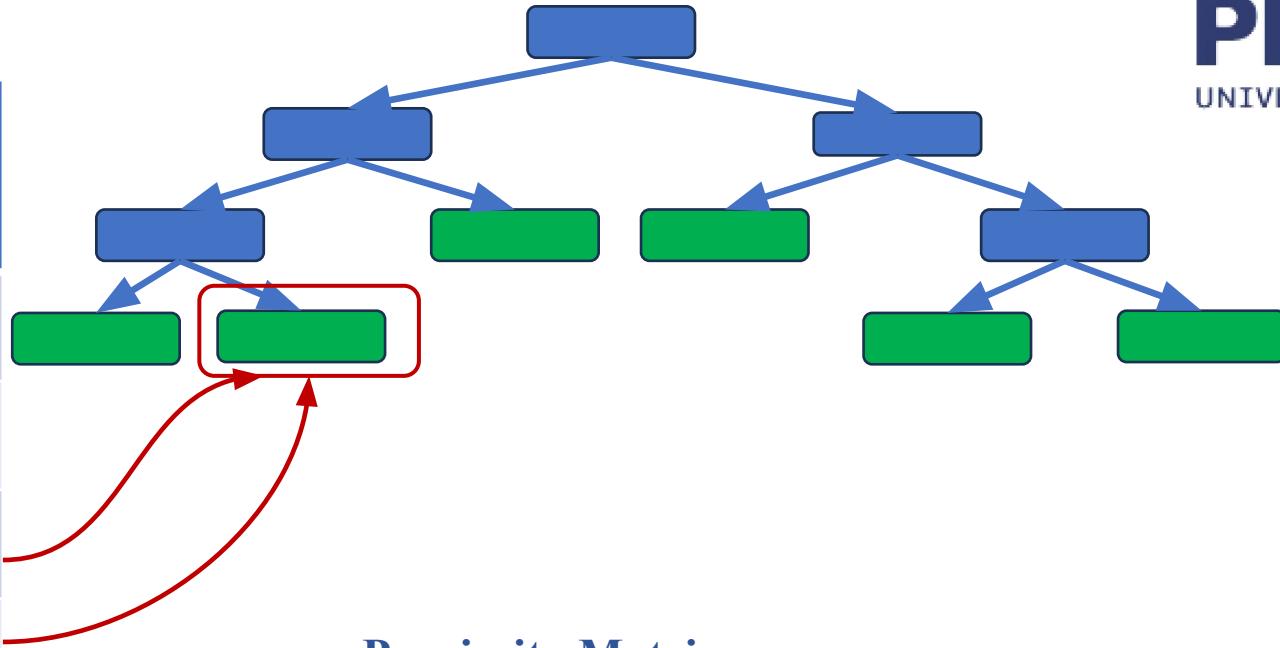
	1	2	3	4
1				
2				
3				
4				

The Proximity matrix has a row for each sample and a column for each sample...

	1	2	3	4
1				
2				
3				
4				

# Ensemble Learning – Random Forest

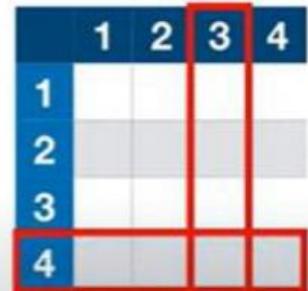
Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



Proximity Matrix

	1	2	3	4
1				

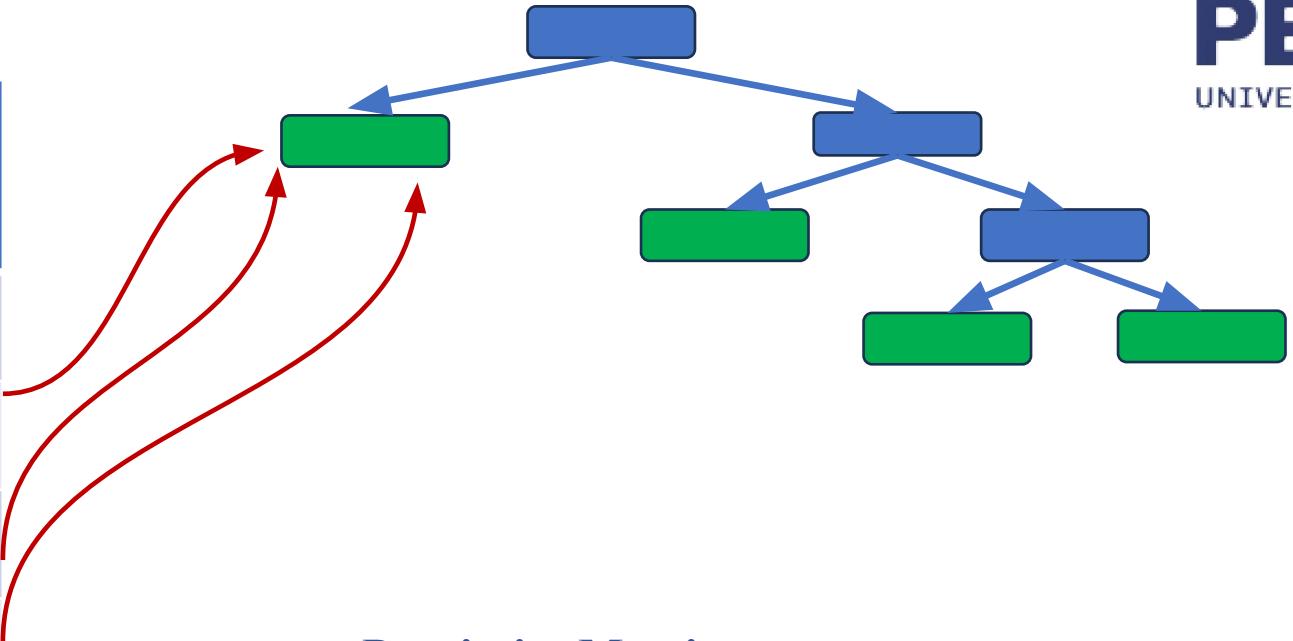
Because sample 3...



...and sample 4 ended up in the same leaf node...

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No



Proximity Matrix

	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

# Ensemble Learning – Random Forest

---

	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	167.5	No

$$\text{Weighted frequency of yes : } 1/3 * \text{weight of yes} \\ = (1/3) * 0.1 = 0.03$$

$$\text{Weighted frequency of no : } 2/3 * \text{weight of no} \\ = (2/3) * 0.9 = 0.6$$

Compute Weighted frequency

$$\begin{aligned}\text{Yes: } & 1/3 \\ \text{No: } & 2/3\end{aligned}$$

$$\begin{aligned}\text{Weight of Yes: } & (\text{proximity of yes}) / \text{all proximities} \\ & = 0.1/1 = 0.1 \\ \text{Weight of No: } & (\text{proximity of no}) / \text{all proximities} \\ & = 0.9/1 = 0.9\end{aligned}$$

# Ensemble Learning – Random Forest

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	198.5	No

Compute Weighted frequency

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

$$\text{Weighted average : } (0.1 * 125) + (0.1 * 180) + (0.8 * 210) \\ = 198.5$$

# Ensemble Learning – Random Forest

New Sample

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	Yes

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	No

# Ensemble Learning – Random Forest

## New Sample

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

Then we use the iterative method we just talked about to make a good guess about the missing values.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	YES	168	YES

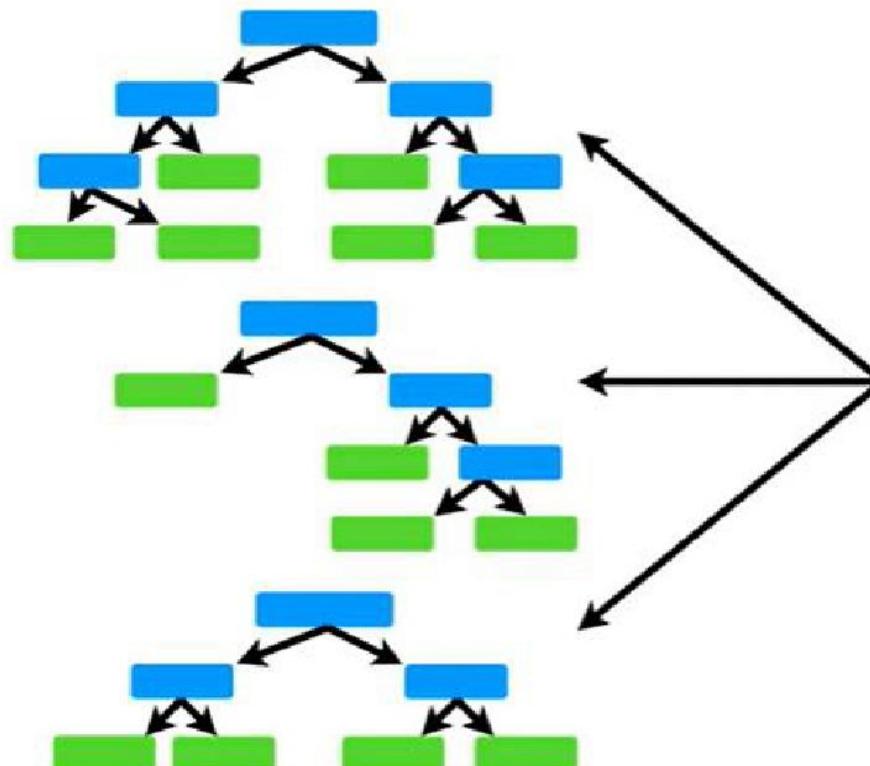
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	NO	168	NO

# Ensemble Learning – Random Forest

New Sample

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	???	168	

...and we see which of the two is correctly labeled by the random forest the most times.

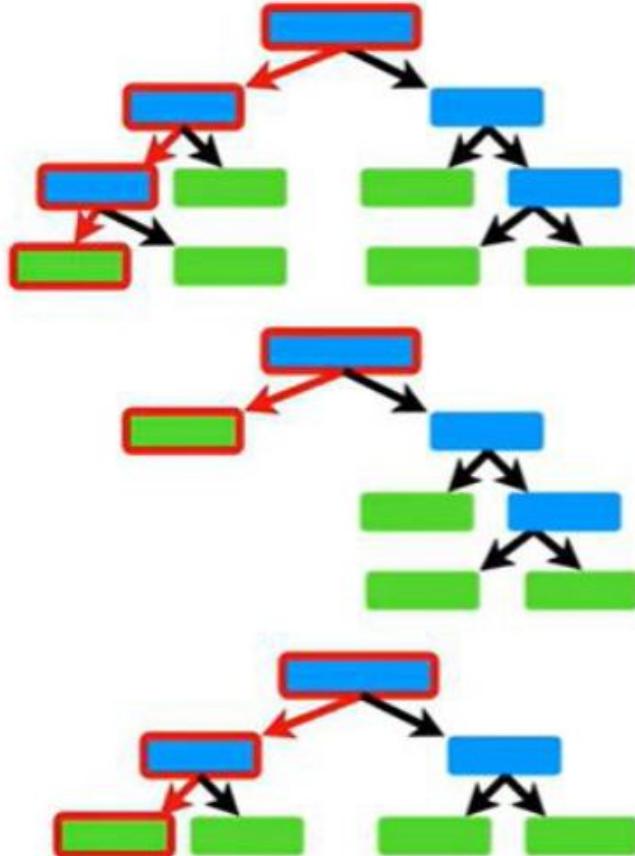


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	YES	168	YES

Then we run the two samples down the trees in the forest...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	NO	168	NO

# Ensemble Learning – Random Forest



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	<b>YES</b>	168	<b>YES</b>

This option wins because it was correctly labeled more than the other option.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	<b>NO</b>	168	<b>NO</b>



**THANK YOU**

---

**Surabhi Narayan**

Department of Computer Science &Engineering

**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
UNIVERSITY

# **UE21CS352A Machine Learning**

---

**Surabhi Narayan**

Department of Computer Science & Engineering

TA : Arvin Nooli

# Ensemble Learning

---

**Surabhi Narayan**

Department of Computer Science & Engineering

# Machine Learning

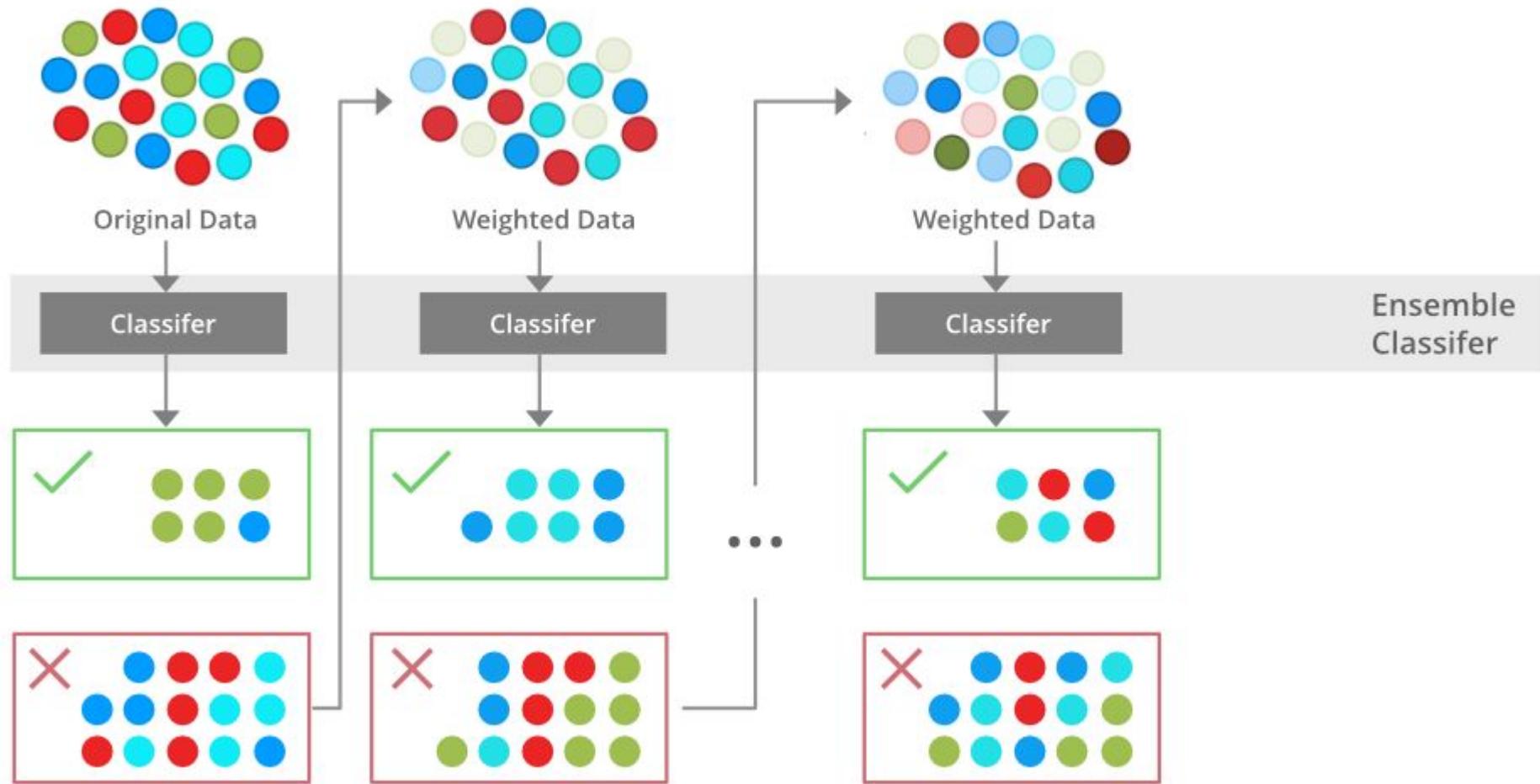
## Acknowledgement

---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Ensemble Learning - Boosting



# Ensemble Learning - Boosting

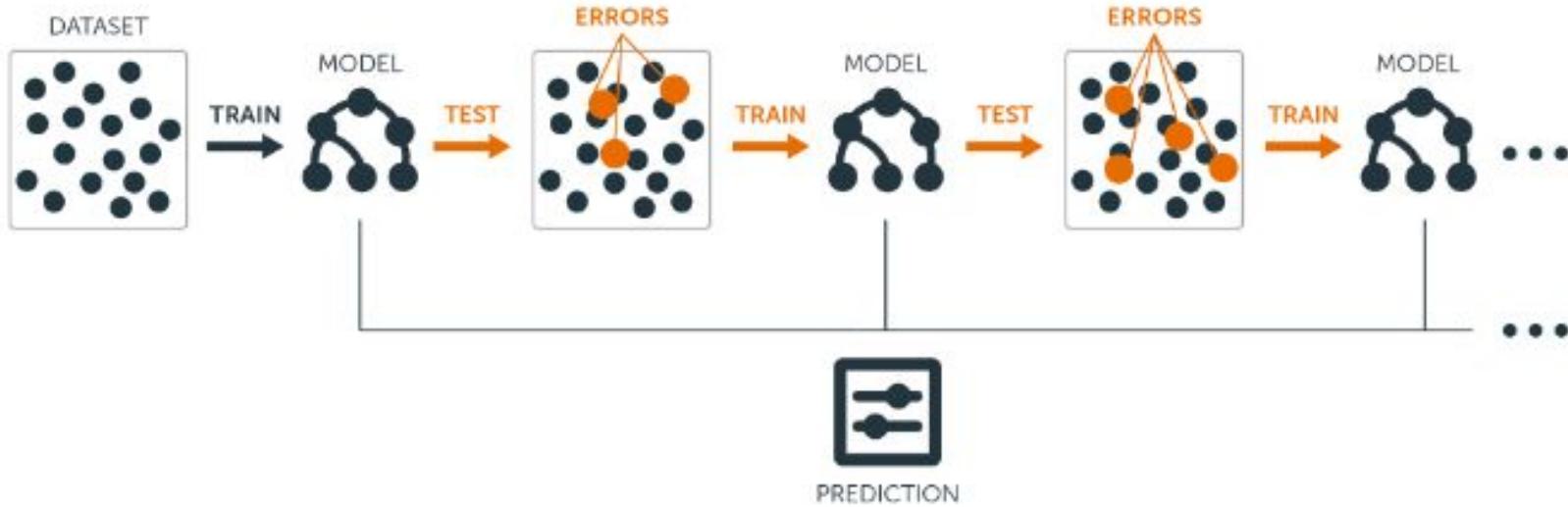
---

Boosting is an ensemble learning technique that aims to create a strong classifier from a series of weak classifiers.

Boosting algorithms can differ in how they create and aggregate weak learners during the sequential process.

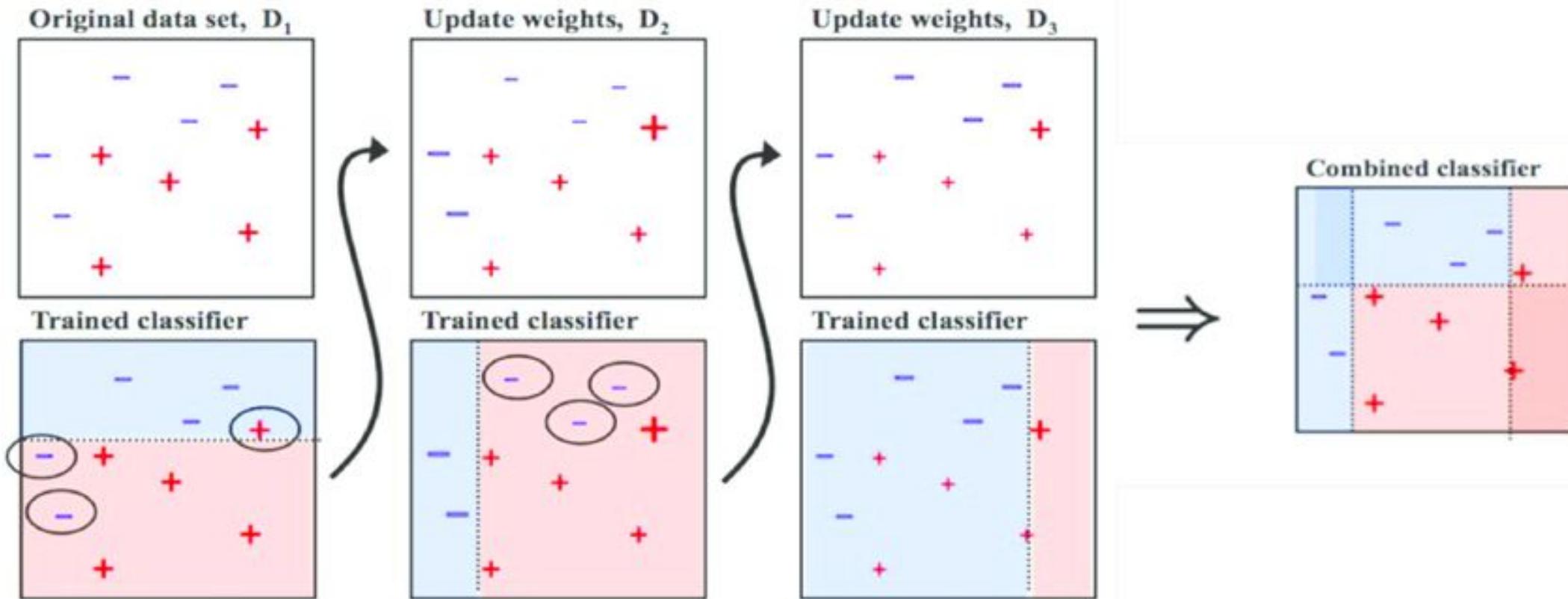
Unlike bagging techniques like Random Forest, where trees are built independently and combined, boosting builds trees **sequentially**, where each subsequent tree focuses on the errors of the previous one like we can see in the previous slide.

# Ensemble Learning - Boosting



Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

# Ensemble Learning - Boosting



# Ensemble Learning - Boosting

---

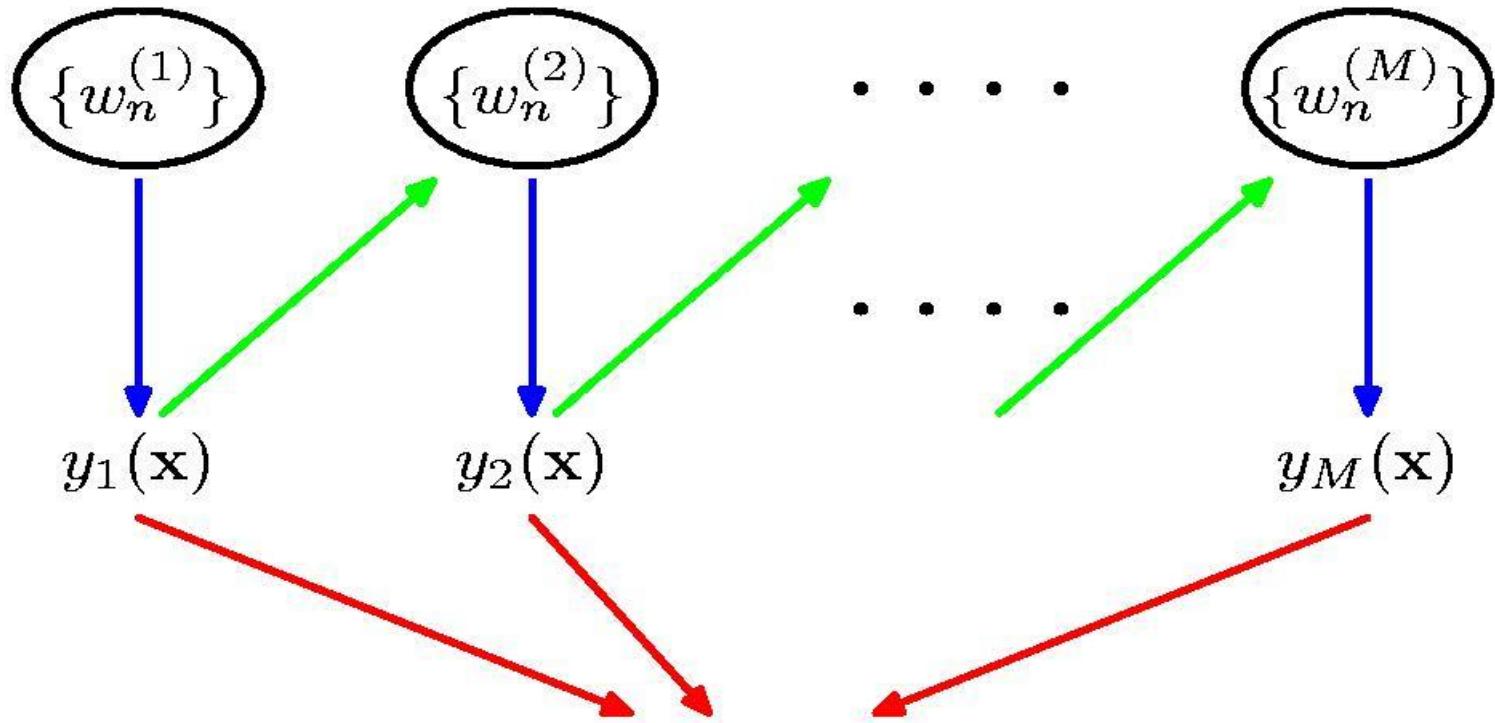
## Advantages:

- Improved Accuracy – Boosting can improve the accuracy of the model by combining several weak models' accuracies and averaging them for regression or voting over them for classification to increase the accuracy of the final model.
- Reduces Bias and Variance: Boosting effectively reduces both bias (by combining multiple weak learners) and variance (by focusing on the difficult-to-predict instances), leading to better generalization on unseen data.
- Robustness to Overfitting – Boosting can reduce the risk of overfitting by reweighting the inputs that are classified wrongly.
- Better handling of imbalanced data – Boosting can handle the imbalance data by focusing more on the data points that are misclassified
- Better Interpretability – Boosting can increase the interpretability of the model by breaking the model decision process into multiple processes.

Three popular types of boosting methods include:

- Adaptive boosting or AdaBoost
- Gradient boosting
- Extreme gradient boosting or XGBoost

# Ensemble Learning – Adaboost



$$Y_M(\mathbf{x}) = \text{sign} \left( \sum_m^M \alpha_m y_m(\mathbf{x}) \right)$$

# Adaptive Boosting or AdaBoost

Three main ideas in Adaboost

1. Adaboost starts by creating a weak learner and estimates the error of the weak learner
2. Weak learners are created sequentially considering the errors of the previous learners
3. Some learners are weighted more than the others (give more weight to the misclassified observations)

The value of the alpha parameter, in this case, will be indirectly proportional to the error of the weak learner

# Ensemble Learning – Adaboost

**Input:** Data set  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ ;  
 Base learning algorithm  $\mathfrak{L}$ ;  
 Number of learning rounds  $T$ .

## **Process:**

- ```

1.  $\mathcal{D}_1(\mathbf{x}) = 1/m.$  % Initialize the weight distribution
2. for  $t = 1, \dots, T:$ 
3.    $h_t = \mathfrak{L}(D, \mathcal{D}_t);$  % Train a classifier  $h_t$  from  $D$  under distribution  $\mathcal{D}_t$ 
4.    $\epsilon_t = P_{\mathbf{x} \sim \mathcal{D}_t}(h_t(\mathbf{x}) \neq f(\mathbf{x}));$  % Evaluate the error of  $h_t$ 
5.   if  $\epsilon_t > 0.5$  then break
6.    $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right);$  % Determine the weight of  $h_t$ 
7.    $\mathcal{D}_{t+1}(\mathbf{x}) = \frac{\mathcal{D}_t(\mathbf{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\mathbf{x}) = f(\mathbf{x}) \\ \exp(\alpha_t) & \text{if } h_t(\mathbf{x}) \neq f(\mathbf{x}) \end{cases}$ 
       $= \frac{\mathcal{D}_t(\mathbf{x}) \exp(-\alpha_t f(\mathbf{x}) h_t(\mathbf{x}))}{Z_t}$  % Update the distribution, where
      %  $Z_t$  is a normalization factor which
      % enables  $\mathcal{D}_{t+1}$  to be a distribution
8. end

```

**Output:**  $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$

## Adaboost with decision trees

1. A Forest of trees, and each tree is a node with two leaves (Decision Stump)
  - Stumps are technically weak learners
2. Stumps are created sequentially considering the errors of the previous Decision stumps
3. Some Stumps are weighted more than the others

# Ensemble Learning - Adaboost



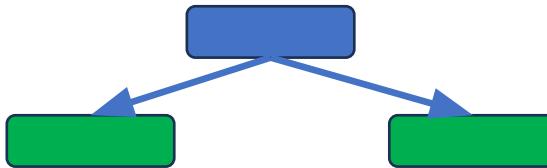
# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |

$$\text{sample weight} = \frac{1}{\text{total number of samples}} = \frac{1}{8}$$

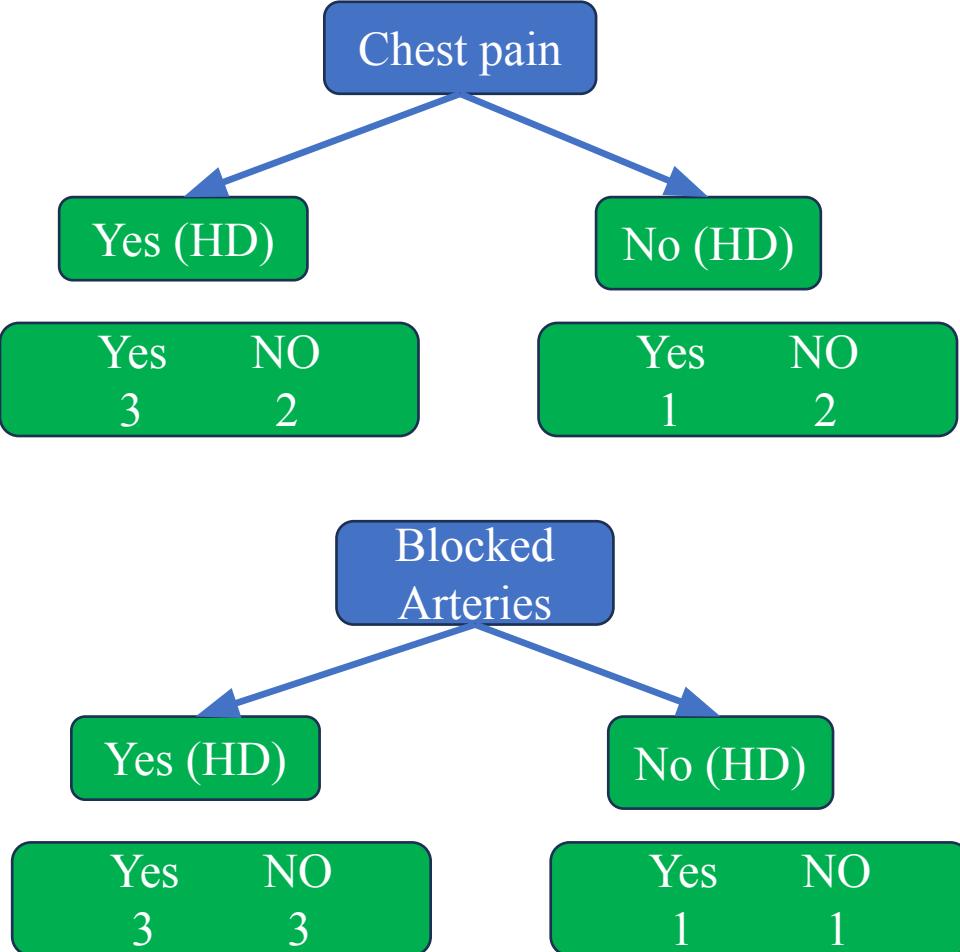
# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



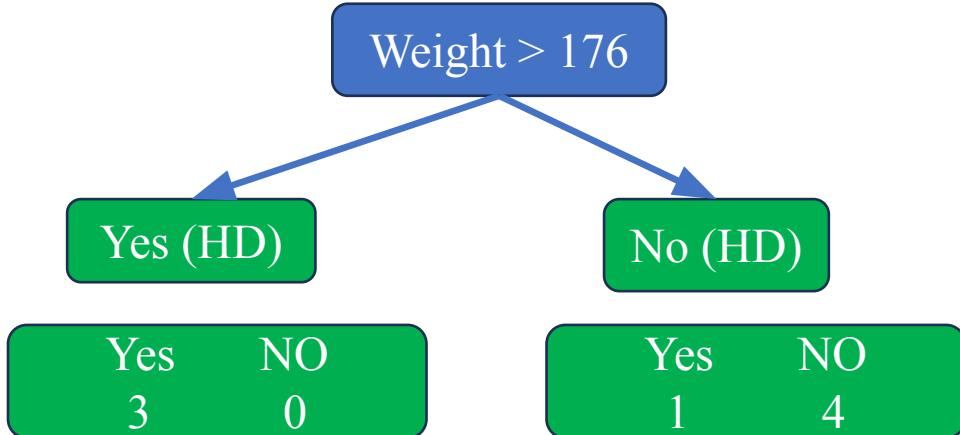
# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



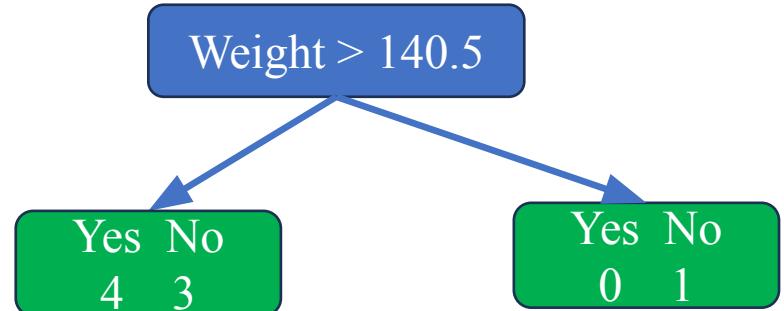
# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



*Gini impurity* =  $1 - (\text{probability of yes})^2 - (\text{probability of no})^2$

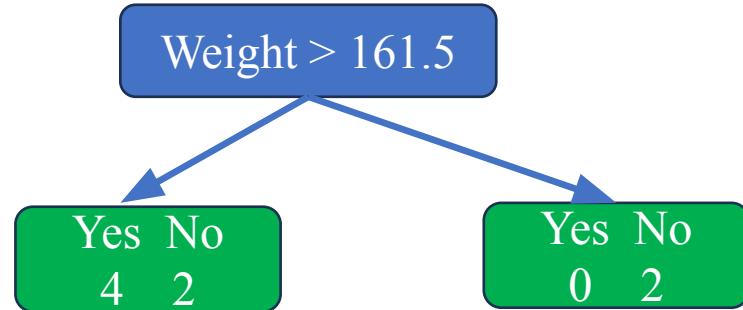
$$\begin{aligned} \text{Gini impurity} &= 1 - \left(\frac{4}{4+3}\right)^2 - \left(\frac{3}{4+3}\right)^2 \\ &= 1 - 0.327 - 0.184 \\ &= 0.489 \end{aligned}$$

$$\text{Gini impurity} = 1 - \left(\frac{0}{0+1}\right)^2 - \left(\frac{1}{0+1}\right)^2 = 0$$

$$\begin{aligned} \text{Total Gini impurity} &= \left(\frac{7}{8}\right)0.489 + \left(\frac{1}{8}\right)0 \\ &= 0.428 \end{aligned}$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



*Gini impurity = 1 - (probability of yes)<sup>2</sup> - (probability of no)<sup>2</sup>*

$$\text{Gini impurity} = 1 - \left( \frac{0}{0+2} \right)^2 - \left( \frac{2}{0+2} \right)^2 = 0$$

$$\text{Gini impurity} = 1 - \left( \frac{4}{4+2} \right)^2 - \left( \frac{2}{4+2} \right)^2$$

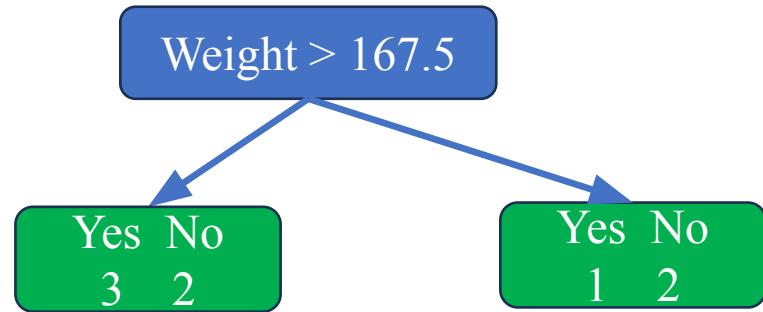
$$= 1 - 0.444 - 0.111 \\ = 0.445$$

$$\text{Total Gini impurity} = \left( \frac{2}{8} \right) 0 + \left( \frac{6}{8} \right) 0.445$$

$$= 0.334$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 140.5          | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 161.5          | Yes           | 1/8           |
| Yes        | No               | 167.5          | No            | 1/8           |
| Yes        | Yes              | 168            | No            | 1/8           |
| Yes        | Yes              | 170            | No            | 1/8           |
| No         | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 176            | Yes           | 1/8           |
| Yes        | Yes              | 192.5          | Yes           | 1/8           |
| Yes        | No               | 205            | Yes           | 1/8           |
| Yes        | No               | 207.5          | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



$$Gini \text{ impurity} = 1 - (\text{probability of yes})^2 - (\text{probability of no})^2$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{1}{1+2}\right)^2 - \left(\frac{2}{1+2}\right)^2 \\ &= 1 - 0.111 - 0.44 \\ &= 0.445 \end{aligned}$$

$$Gini \text{ impurity} = 1 - \left(\frac{3}{3+2}\right)^2 - \left(\frac{2}{3+2}\right)^2$$

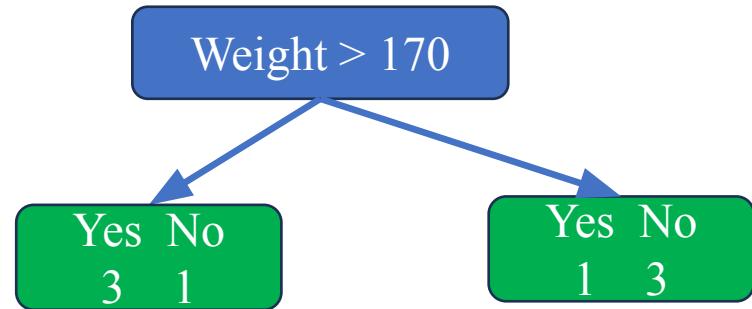
$$\begin{aligned} &= 1 - 0.36 - 0.16 \\ &= 0.48 \end{aligned}$$

$$Total \text{ Gini impurity} = \left(\frac{3}{8}\right)0.445 + \left(\frac{5}{8}\right)0.48$$

$$= 0.167 + 0.3 = 0.467$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



*Gini impurity = 1 - (probability of yes)<sup>2</sup> - (probability of no)<sup>2</sup>*

$$\text{Gini impurity} = 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+2}\right)^2 = 1 - 0.0625 - 0.5625$$

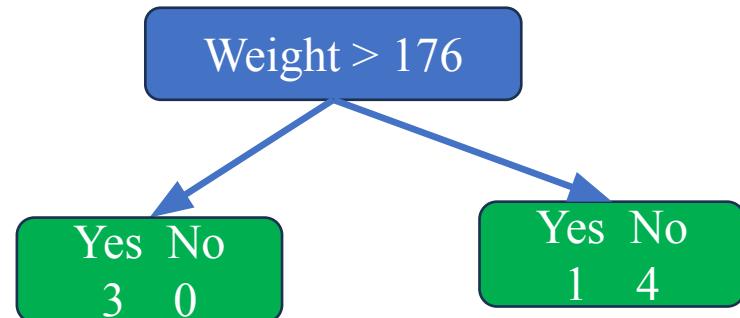
$$\text{Gini impurity} = 1 - \left(\frac{3}{3+1}\right)^2 - \left(\frac{1}{3+1}\right)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

$$= 1 - 0.5625 - 0.0625 = 0.375$$

$$\text{Total Gini impurity} = \left(\frac{4}{8}\right)0.375 + \left(\frac{4}{8}\right)0.375 = 0.375$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 140.5          | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 161.5          | Yes           | 1/8           |
| Yes        | No               | 167.5          | No            | 1/8           |
| Yes        | Yes              | 168            | No            | 1/8           |
| Yes        | Yes              | 170            | No            | 1/8           |
| No         | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 176            | Yes           | 1/8           |
| Yes        | Yes              | 192.5          | Yes           | 1/8           |
| Yes        | No               | 207.5          | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of yes})^2 - (probability \text{ of no})^2$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{1}{1+4}\right)^2 - \left(\frac{4}{1+4}\right)^2 \\ &= 1 - 0.04 - 0.64 \\ &= 0.32 \end{aligned}$$

$$Gini \text{ impurity} = 1 - \left(\frac{3}{3+0}\right)^2 - \left(\frac{0}{3+0}\right)^2$$

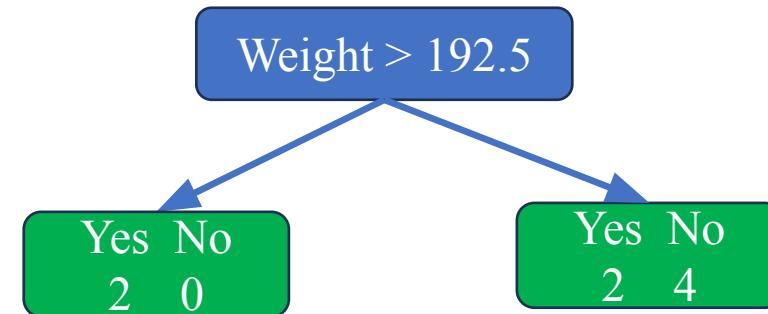
$$= 0$$

$$Total \text{ Gini impurity} = \left(\frac{5}{8}\right)0.32 + \left(\frac{3}{8}\right)0$$

$$= 0.2$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 140.5          | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 161.5          | Yes           | 1/8           |
| Yes        | No               | 167.5          | No            | 1/8           |
| Yes        | Yes              | 168            | No            | 1/8           |
| Yes        | Yes              | 170            | No            | 1/8           |
| No         | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 176            | Yes           | 1/8           |
| Yes        | Yes              | 192.5          | Yes           | 1/8           |
| Yes        | No               | 207.5          | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of } yes)^2 - (probability \text{ of } no)^2$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{2}{2+4}\right)^2 - \left(\frac{4}{2+4}\right)^2 \\ &= 1 - 0.111 - 0.444 \\ &= 0.444 \end{aligned}$$

$$Gini \text{ impurity} = 1 - \left(\frac{2}{2+0}\right)^2 - \left(\frac{0}{2+0}\right)^2$$

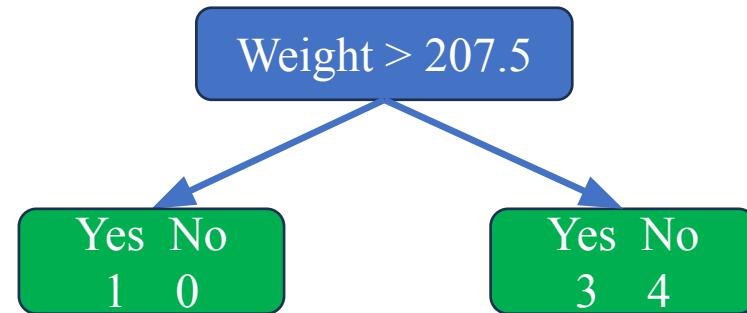
$$= 0$$

$$Total \text{ Gini } \text{ impurity} = \left(\frac{6}{8}\right)0.444 + \left(\frac{2}{8}\right)0$$

$$= 0.333$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 140.5          | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 161.5          | Yes           | 1/8           |
| Yes        | No               | 167.5          | No            | 1/8           |
| Yes        | Yes              | 168            | No            | 1/8           |
| Yes        | Yes              | 170            | No            | 1/8           |
| No         | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 176            | Yes           | 1/8           |
| Yes        | Yes              | 192.5          | Yes           | 1/8           |
| Yes        | No               | 207.5          | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of } yes)^2 - (probability \text{ of } no)^2$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{3}{3+4}\right)^2 - \left(\frac{4}{3+4}\right)^2 \\ &= 1 - 0.184 - 0.327 \\ &= 0.489 \end{aligned}$$

$$Gini \text{ impurity} = 1 - \left(\frac{1}{1+0}\right)^2 - \left(\frac{0}{1+0}\right)^2$$

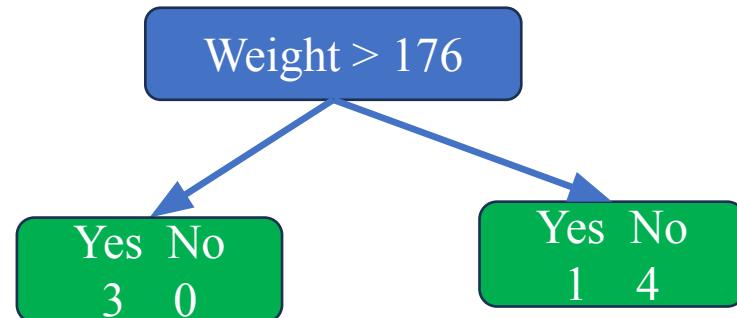
$$= 0$$

$$Total \text{ Gini impurity} = \left(\frac{7}{8}\right)0.489 + \left(\frac{1}{8}\right)0$$

$$= 0.428$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 140.5          | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 161.5          | Yes           | 1/8           |
| Yes        | No               | 167.5          | No            | 1/8           |
| Yes        | Yes              | 168            | No            | 1/8           |
| Yes        | Yes              | 170            | No            | 1/8           |
| No         | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 176            | Yes           | 1/8           |
| Yes        | Yes              | 192.5          | Yes           | 1/8           |
| Yes        | No               | 207.5          | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of } yes)^2 - (probability \text{ of } no)^2$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{1}{1+4}\right)^2 - \left(\frac{4}{1+4}\right)^2 \\ &= 1 - 0.04 - 0.64 \\ &= 0.32 \end{aligned}$$

$$Gini \text{ impurity} = 1 - \left(\frac{3}{3+0}\right)^2 - \left(\frac{0}{3+0}\right)^2$$

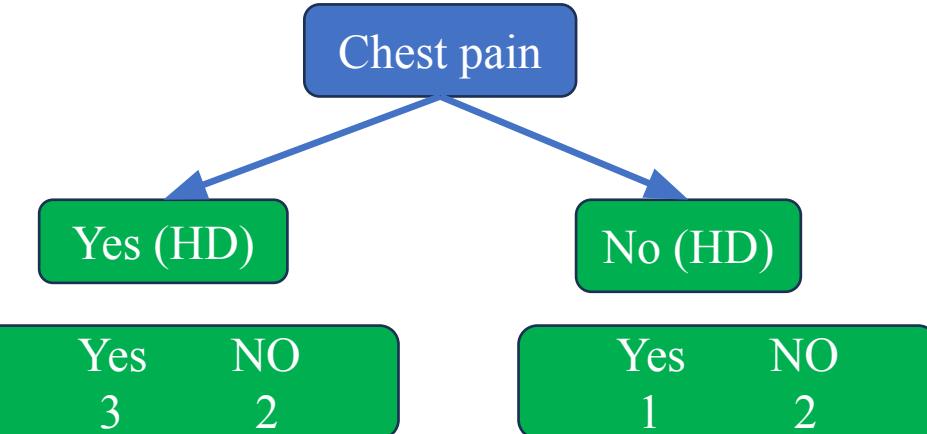
$$= 0$$

$$Total \text{ Gini impurity} = \left(\frac{5}{8}\right)0.32 + \left(\frac{3}{8}\right)0$$

$$= 0.2$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of } yes)^2 - (probability \text{ of } no)^2$$

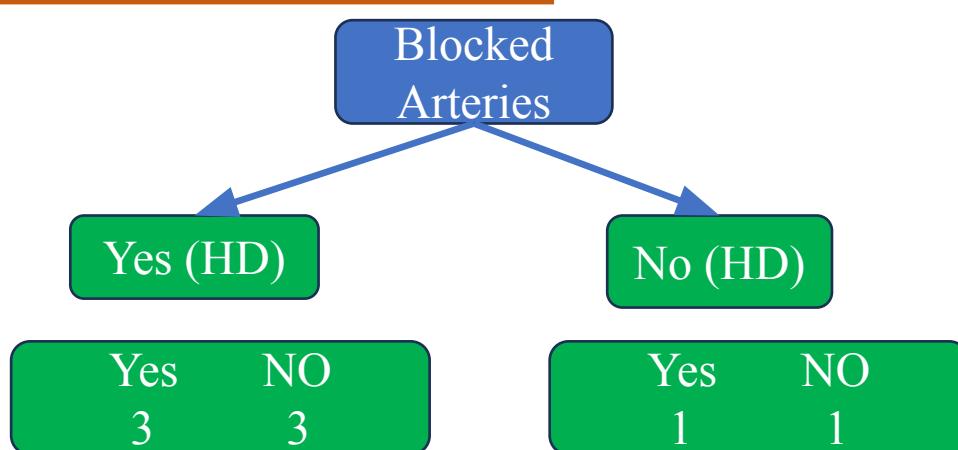
$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{3}{3+2}\right)^2 - \left(\frac{2}{3+2}\right)^2 \\ &= 1 - 0.36 - 0.16 \\ &= 0.48 \end{aligned}$$

$$\begin{aligned} Gini \text{ impurity} &= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 \\ &= 1 - 0.444 - 0.111 \\ &= 0.444 \end{aligned}$$

$$\begin{aligned} Total \text{ Gini impurity} &= \left(\frac{5}{8}\right)0.48 + \left(\frac{3}{8}\right)0.444 \\ &= 0.3 + 0.167 \\ &= 0.467 \end{aligned}$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



$$Gini \text{ impurity} = 1 - (probability \text{ of } yes)^2 - (probability \text{ of } no)^2$$

$$Gini \text{ impurity} = 1 - \left( \frac{3}{3+3} \right)^2 - \left( \frac{3}{3+3} \right)^2 = 1 - 0.25 - 0.25 = 0.5$$

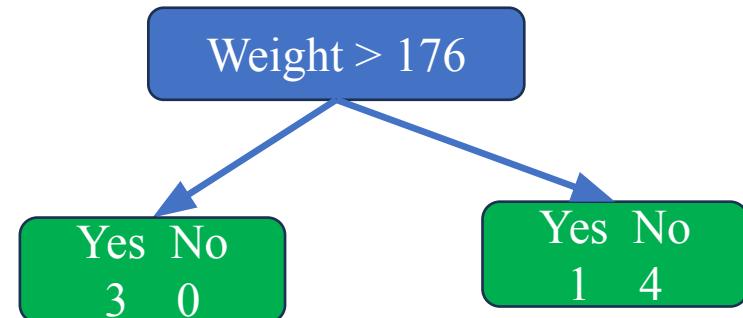
$$Gini \text{ impurity} = 1 - \left( \frac{1}{1+1} \right)^2 - \left( \frac{1}{1+1} \right)^2 = 1 - 0.25 - 0.25 = 0.5$$

$$Total \text{ Gini impurity} = \left( \frac{6}{8} \right) 0.5 + \left( \frac{2}{8} \right) 0.5$$

$$= 0.375 + 0.125 = 0.5$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 125            | No            | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |

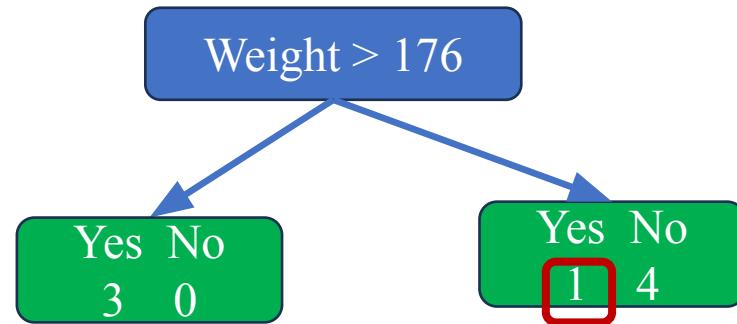


This is the first stump to be created

We have to determine the weight or amount of say of this stump in final classification

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



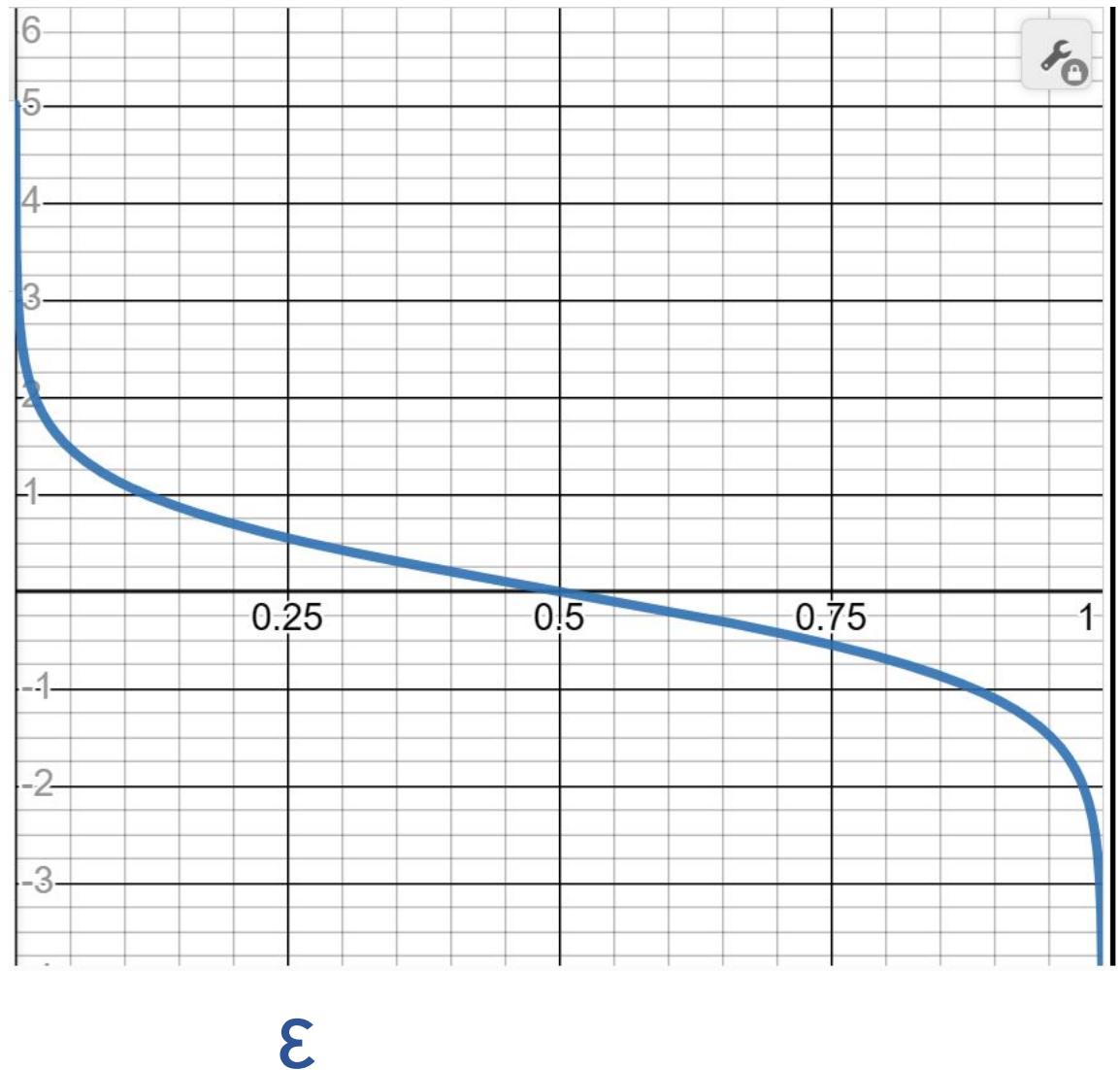
Total Error ( $\epsilon$ ) = 1/8

$$\alpha = \frac{1}{2} \ln \left( \frac{1 - \epsilon}{\epsilon} \right)$$

$$\alpha = \frac{1}{2} \ln \left( \frac{1 - \left( \frac{1}{8} \right)}{\left( \frac{1}{8} \right)} \right)$$

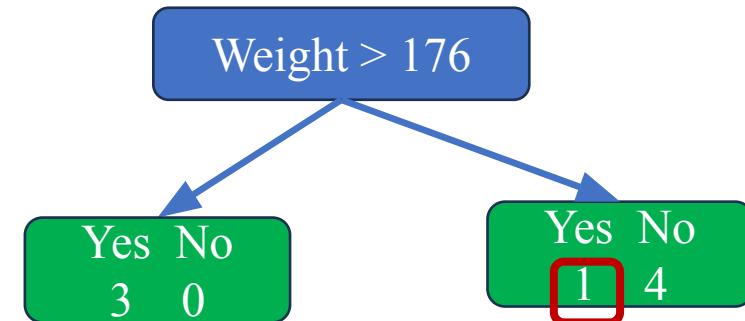
$\alpha = 0.97$

# Ensemble Learning - Adaboost



# Ensemble Learning - Adaboost

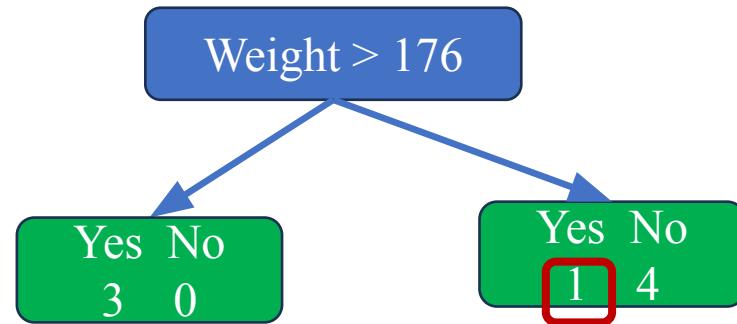
| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



Emphasise this sample by increasing the sample weight and decreasing the sample weight of other samples

# Ensemble Learning - Adaboost

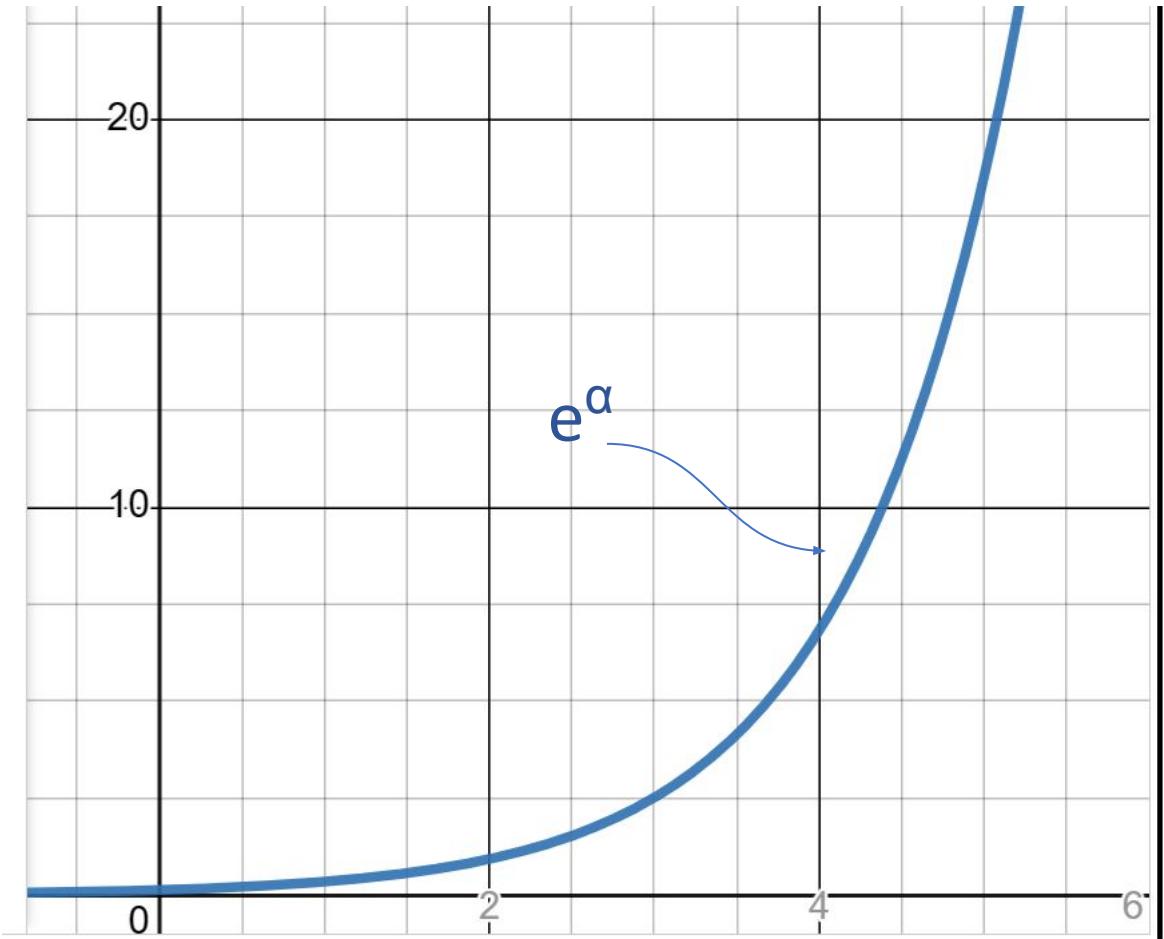
| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



$$\text{new sample weight} = \text{sample weight} * e^{\alpha}$$

$$\text{new sample weight} = (1/8) * e^{0.97}$$

# Ensemble Learning - Adaboost

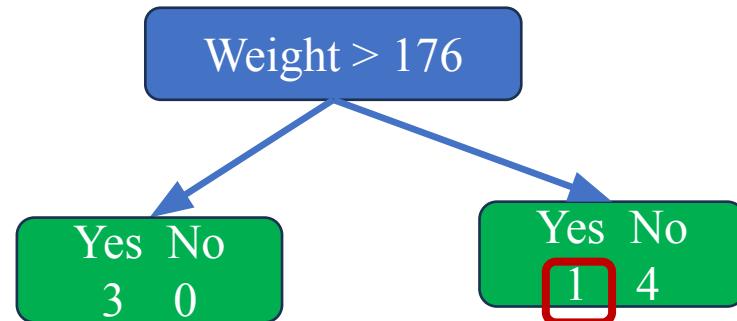


$$\begin{aligned} \text{new sample weight} &= (1/8) * e^{0.97} \\ &= (1/8)*2.64 = 0.33 \end{aligned}$$

If  $\alpha$  is large , new sample weight is large

# Ensemble Learning - Adaboost

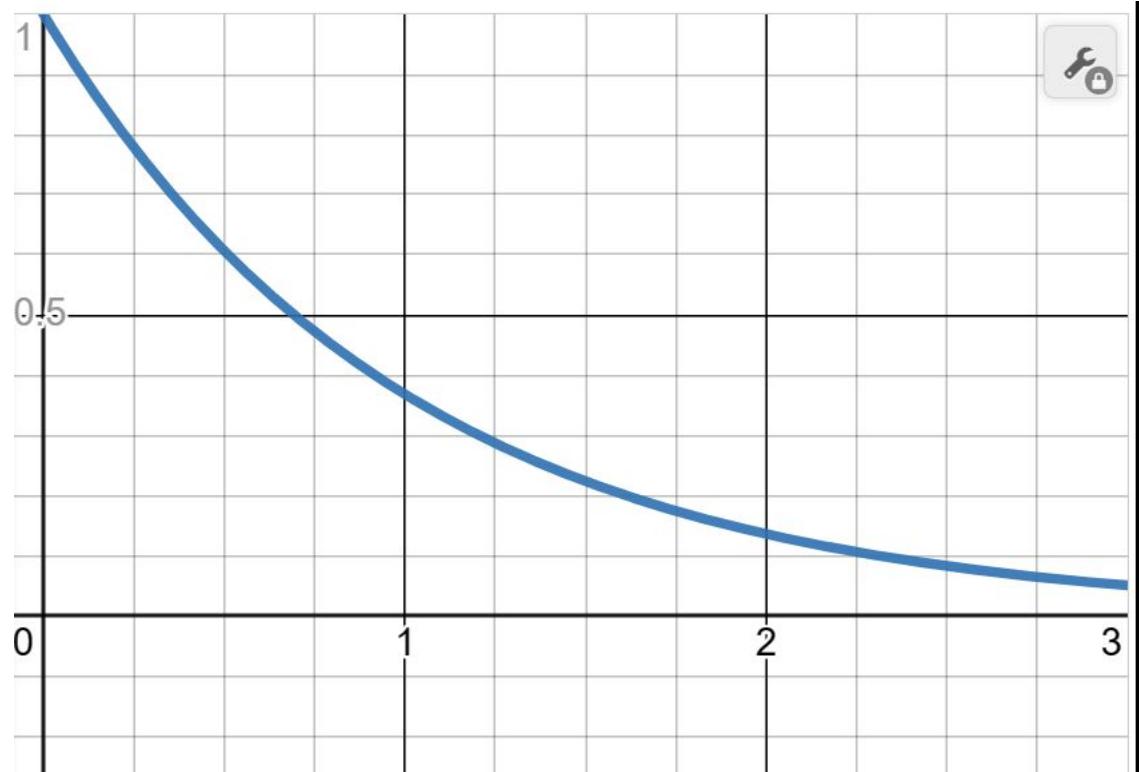
| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 1/8           |
| No         | Yes              | 180            | Yes           | 1/8           |
| Yes        | No               | 210            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 156            | No            | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | No               | 168            | No            | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |



$$\text{new sample weight} = \text{sample weight} * e^{-\alpha}$$

$$\text{new sample weight} = (1/8) * e^{-0.97}$$

# Ensemble Learning - Adaboost



$$= (1/8)*0.38 = 0.05$$

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.05          |
| No         | Yes              | 180            | Yes           | 0.05          |
| Yes        | No               | 210            | Yes           | 0.05          |
| Yes        | Yes              | 167            | Yes           | 0.33          |
| No         | Yes              | 156            | No            | 0.05          |
| No         | Yes              | 125            | No            | 0.05          |
| Yes        | No               | 168            | No            | 0.05          |
| Yes        | Yes              | 172            | No            | 0.05          |

Normalize the new sample weight

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.07          |
| No         | Yes              | 180            | Yes           | 0.07          |
| Yes        | No               | 210            | Yes           | 0.07          |
| Yes        | Yes              | 167            | Yes           | 0.49          |
| No         | Yes              | 156            | No            | 0.07          |
| No         | Yes              | 125            | No            | 0.07          |
| Yes        | No               | 168            | No            | 0.07          |
| Yes        | Yes              | 172            | No            | 0.07          |

Pick a random number between 0 and 1.  
Treat the sample weight as distribution

# Ensemble Learning - Adaboost

0.72, 0.42, 0.83, 0.51, ...

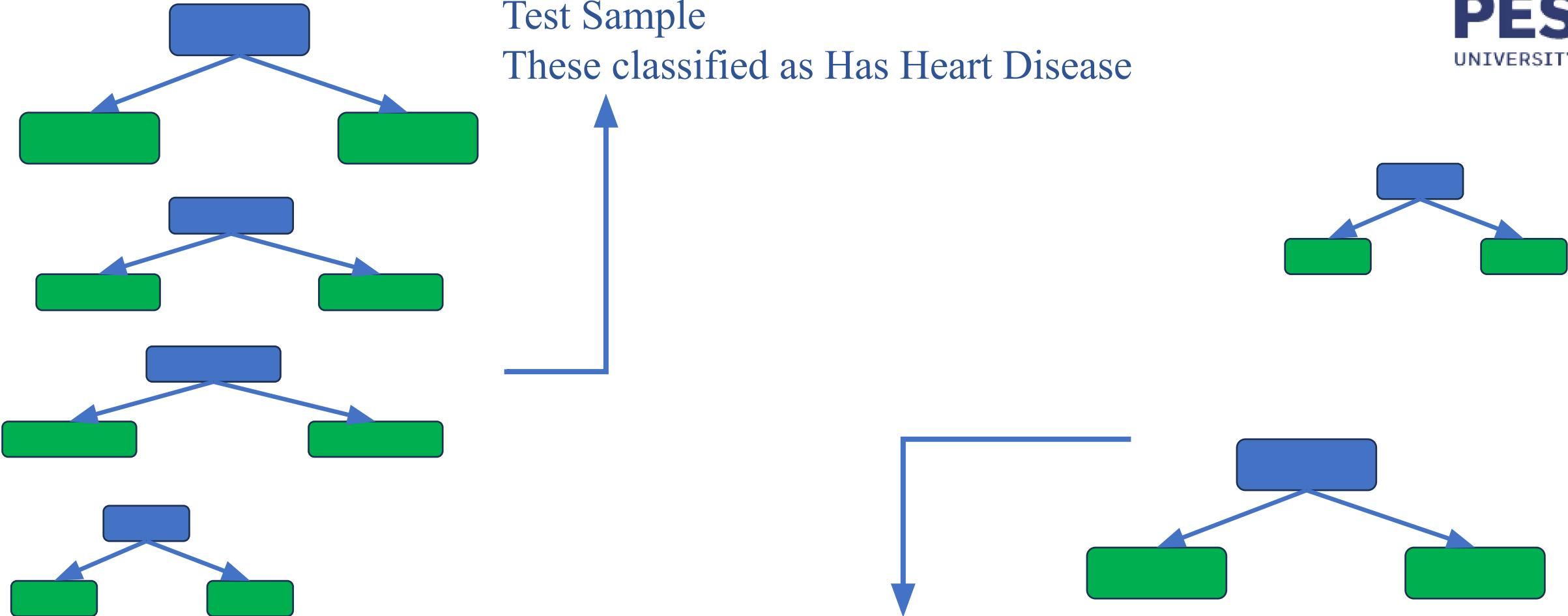
| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| Yes        | Yes              | 205            | Yes           | 0.07          |
| No         | Yes              | 180            | Yes           | 0.07          |
| Yes        | No               | 210            | Yes           | 0.07          |
| Yes        | Yes              | 167            | Yes           | 0.49          |
| No         | Yes              | 156            | No            | 0.07          |
| No         | Yes              | 125            | No            | 0.07          |
| Yes        | No               | 168            | No            | 0.07          |
| Yes        | Yes              | 172            | No            | 0.07          |

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |

# Ensemble Learning - Adaboost

| Chest pain | Blocked Arteries | Patient Weight | Heart Disease | Sample Weight |
|------------|------------------|----------------|---------------|---------------|
| No         | Yes              | 156            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| No         | Yes              | 125            | No            | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |
| Yes        | Yes              | 172            | No            | 1/8           |
| Yes        | Yes              | 205            | Yes           | 1/8           |
| Yes        | Yes              | 167            | Yes           | 1/8           |

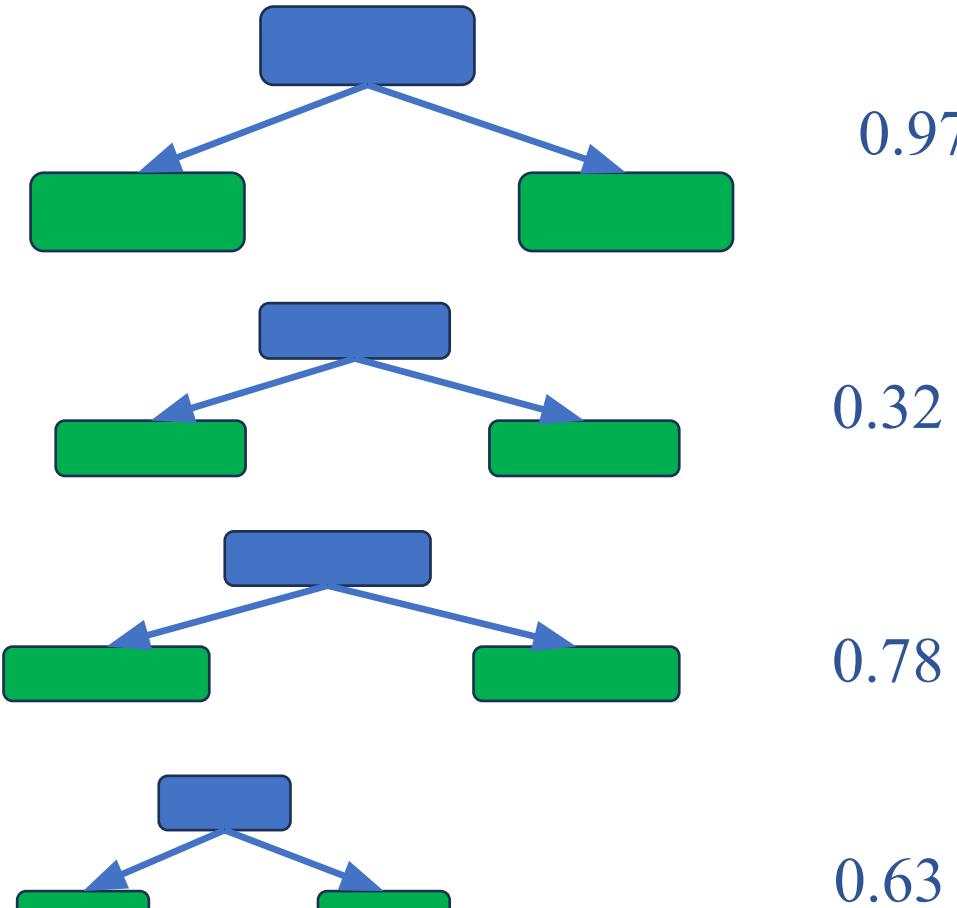
# Ensemble Learning - Adaboost



# Test Sample

## These classified as No Heart Disease

# Ensemble Learning - Adaboost

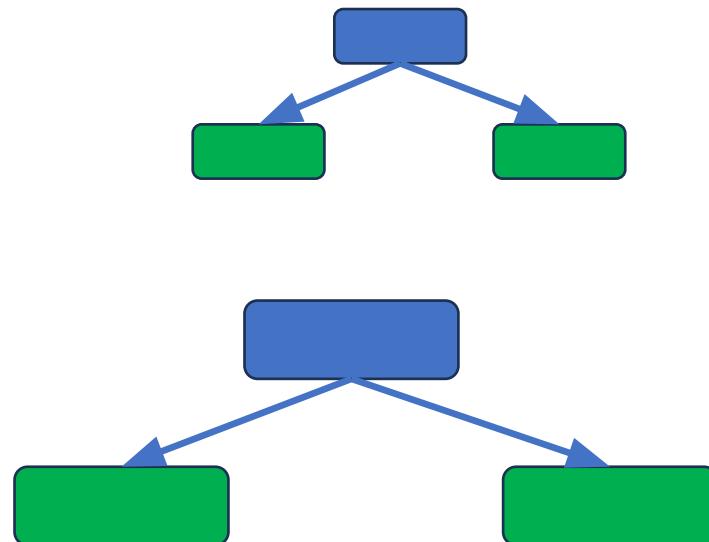


Heart Disease

Total = 2.7

Weight/ Amounts of say

0.41



No Heart Disease

Total = 1.23

Classified as Has Heart Disease

# Ensemble Learning – Gradient Boost

---

Gradient boosting technique that builds a final model from the sum of several weak learning algorithms that were trained on the same dataset.

## How it Operates ?

Step 1) It operates on the idea of stagewise addition. The first weak learner in the gradient boosting algorithm will not be trained on the dataset; instead, it will simply return the mean of the relevant column.

Step 2) The residual for the first weak learner algorithm's output will then be calculated and used as the output column or target column for the next weak learning algorithm that will be trained.

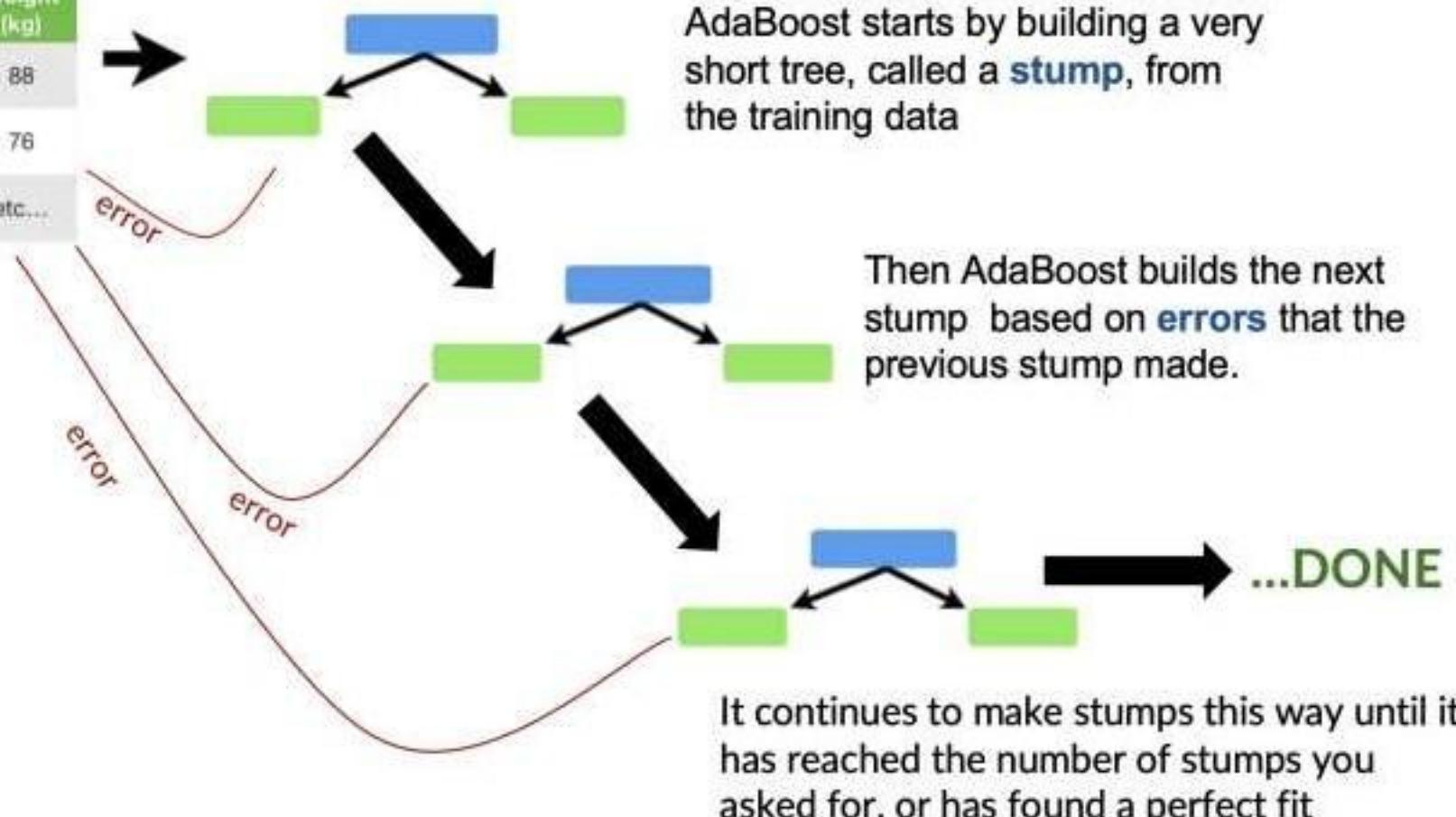
Step 3) The second weak learner will be trained using the same methodology, and the residuals will be computed and utilized as an output column once more for the third weak learner, and so on until we achieve zero residuals.

The dataset for gradient boosting must be in the form of numerical or categorical data, and the loss function used to generate the residuals must be differential at all times.

# Ensemble Learning – Gradient Boost vs Adaboost

## AdaBoost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| etc...     | etc...         | etc... | etc...      |

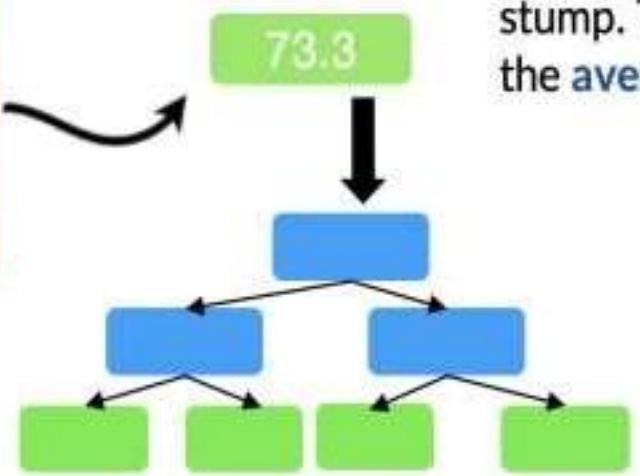


# Ensemble Learning – Gradient Boost vs Adaboost

## Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| etc...     | etc...         | etc... | etc...      |

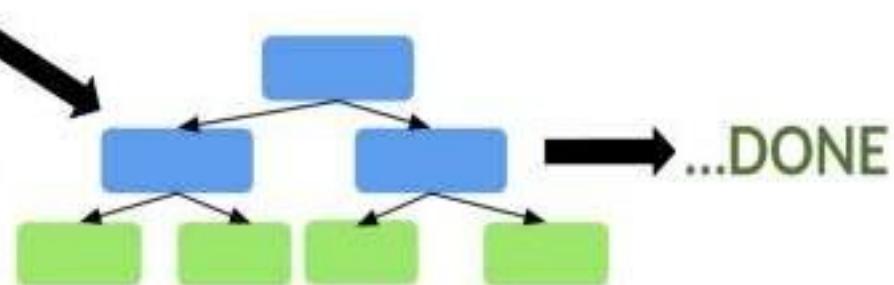
Gradient Boost then builds a tree using the errors of the previous tree



Gradient Boost starts by making a **single leaf**, not a tree or a stump. The first guess is always the **average value**

Thus, like AdaBoost, Gradient Boost creates **fixed size trees**, but unlike AdaBoost, each tree is **more than a stump**

Gradient Boost builds another tree based on the errors of the previous tree



...DONE

# Ensemble Learning – Gradient Boost vs Adaboost

| Aspect                    | AdaBoost                                                                                                  | Gradient Boosting                                                                                                          |
|---------------------------|-----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Basic Principle           | Focuses on correcting the errors of the previous weak learners by adjusting weights on the data points.   | Sequentially minimizes a specified loss function by fitting new models to the residuals (errors) of the previous model.    |
| Weak Learners             | Typically uses decision stumps (single-split trees) as weak learners.                                     | Uses decision trees, typically deeper than stumps, to model residuals.                                                     |
| Weighting Scheme          | Weights of misclassified data points are increased to emphasize difficult cases in subsequent iterations. | No explicit weighting of data points; instead, the model directly optimizes the loss function by fitting on the residuals. |
| Focus on Errors           | Directly focuses on the instances that were misclassified in previous rounds.                             | Focuses on minimizing the overall loss, considering both well-classified and misclassified instances.                      |
| Sensitivity to Outliers   | Highly sensitive to outliers as it gives more weight to misclassified points, which can include outliers. | Less sensitive to outliers as it focuses on reducing overall loss rather than emphasizing specific instances.              |
| Optimization Method       | Uses an iterative reweighting approach to emphasize difficult-to-classify instances.                      | Uses gradient descent to optimize the loss function by fitting the new model to the residuals of the previous models.      |
| Performance               | Generally works well with simpler datasets and when the weak learners are very basic.                     | Often achieves higher accuracy, particularly on complex datasets, due to its ability to model complex relationships.       |
| Flexibility               | Less flexible, primarily used for binary classification, though it can be extended.                       | Highly flexible, used for both classification and regression, and can optimize various loss functions.                     |
| Computational Efficiency  | Computationally efficient due to the use of simple weak learners, but less so with more complex learners. | More computationally intensive due to the iterative fitting of trees to residuals, especially as trees become deeper.      |
| Implementation Complexity | Relatively simple to implement and understand.                                                            | More complex to implement, often requiring careful tuning of hyperparameters like learning rate and tree depth.            |

# Ensemble Learning – Gradient Boost

## Gradient Boosting Algorithm

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^n L(y_i, \gamma)$$

2. for  $m = 1$  to  $M$ :

2-1. Compute residuals  $r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$  for  $i = 1, \dots, n$

2-2. Train regression tree with features  $x$  against  $r$  and create terminal node regions  $R_{jm}$  for  $j = 1, \dots, J_m$

2-3. Compute  $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma)$  for  $j = 1, \dots, J_m$

2-4. Update the model:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}(x \in R_{jm})$$

**Step 3:** Output  $F_M(x)$

# Ensemble Learning – Gradient Boost Example

## Data

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

We shall use this training data  
To predict weight

# Ensemble Learning – Gradient Boost

Lets see how gradient boost can be used to predict the Weight of this training data

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

Step 1: Calculate the average of all weight. This is our first attempt at predicting the weights.

71.2

Step 2: Calculate the errors from the previous stump by calculating the diff between the observed weight and predicted weight  
 $(\text{Observed weight} - \text{Predicted Weight})$

These differences in predictions form our "residuals"

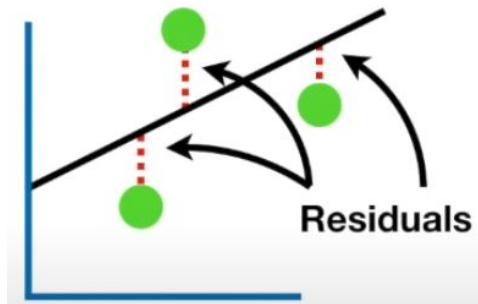
# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 1.6        | Blue           | Male   | 88          | 16.8     |
| 1.6        | Green          | Female | 76          |          |
| 1.5        | Blue           | Female | 56          |          |
| 1.8        | Red            | Male   | 73          |          |
| 1.5        | Green          | Male   | 77          |          |
| 1.4        | Blue           | Female | 57          |          |

...and save the difference, which is called a **Pseudo Residual**, in a new column.

$$(88 - 71.2) = 16.8$$

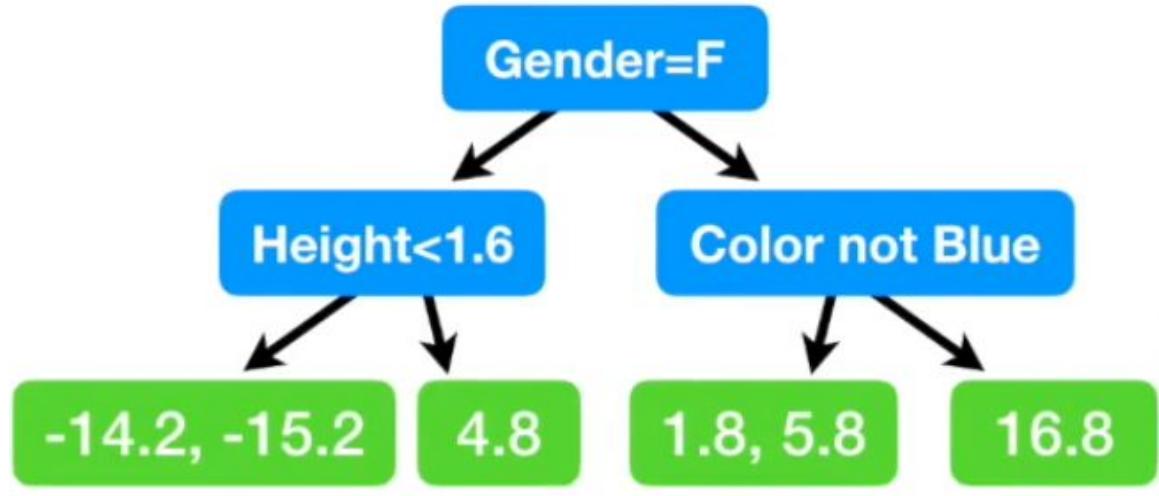
Note: The term pseudo residual is based on linear regression , which is basically the value difference in observed and predicted



# Ensemble Learning – Gradient Boost

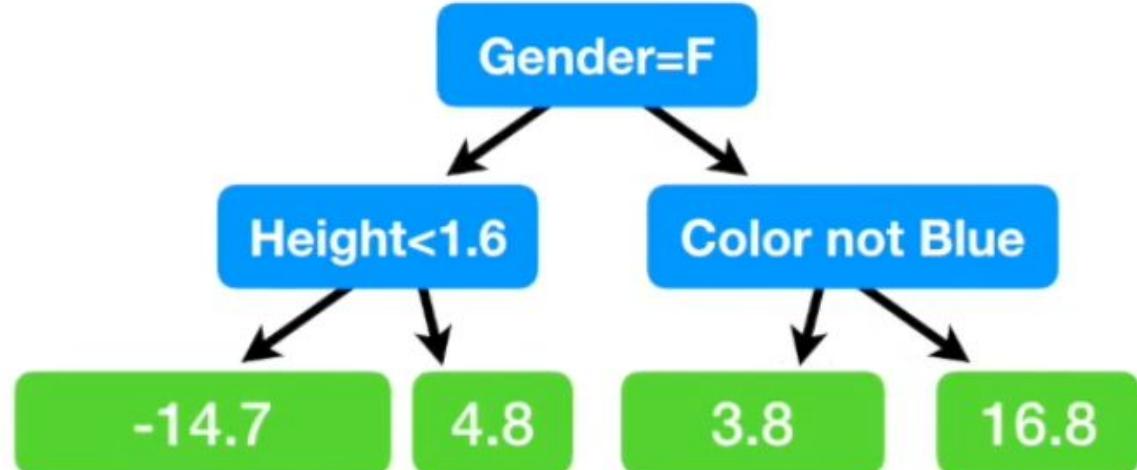
| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 1.6        | Blue           | Male   | 88          | 16.8     |
| 1.6        | Green          | Female | 76          | 4.8      |
| 1.5        | Blue           | Female | 56          | -15.2    |
| 1.8        | Red            | Male   | 73          | 1.8      |
| 1.5        | Green          | Male   | 77          | 5.8      |
| 1.4        | Blue           | Female | 57          | -14.2    |

# Ensemble Learning – Gradient Boost



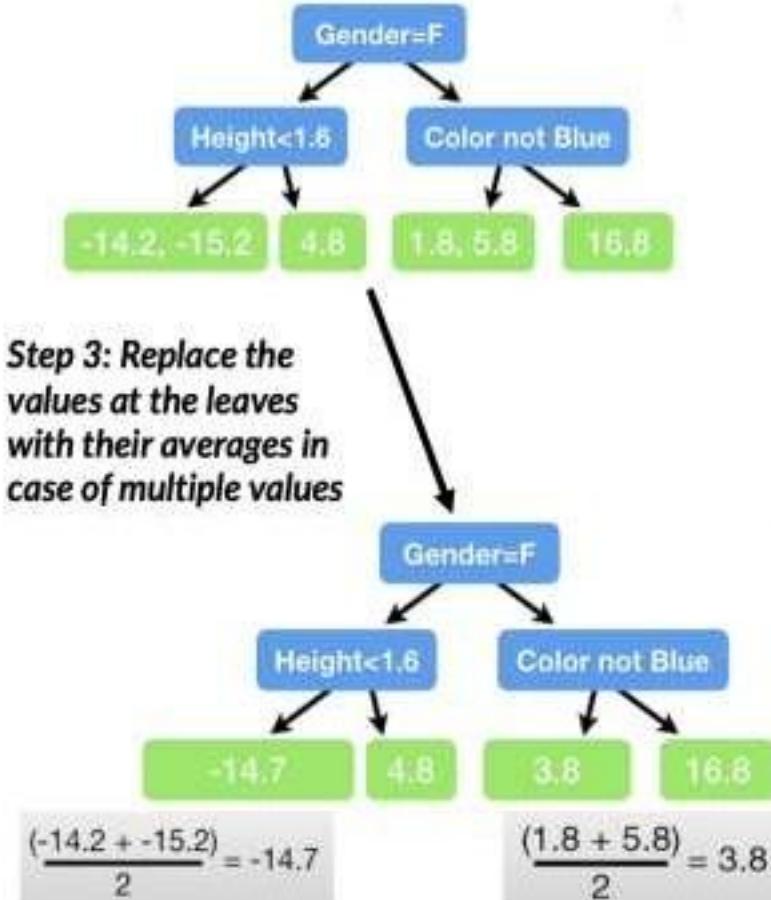
Averaged value so we get only one answer in each leaf node

We are setting only 4 leaf nodes at the end



# Ensemble Learning – Gradient Boost

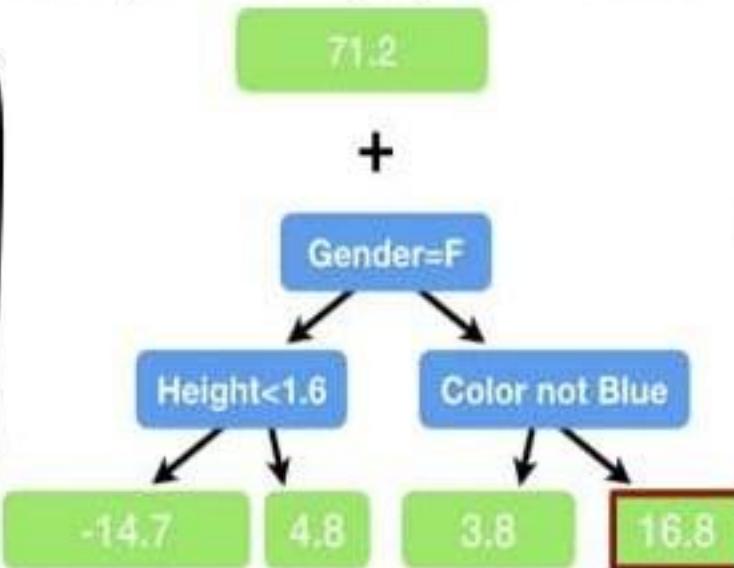
| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 1.6        | Blue           | Male   | 88          | 16.8     |
| 1.6        | Green          | Female | 76          | 4.8      |
| 1.5        | Blue           | Female | 56          | -15.2    |
| 1.8        | Red            | Male   | 73          | 1.8      |
| 1.5        | Green          | Male   | 77          | 5.8      |
| 1.4        | Blue           | Female | 57          | -14.2    |



# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

Step 4: Combine the original tree with the new tree to get a new predicted weight for the same data.



Let's say we get 16.8 after running the data down the tree.

So, the predicted weight is  $71.2 + 16.8 = 88.0$ .  
This is the same as the observed weight.

# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

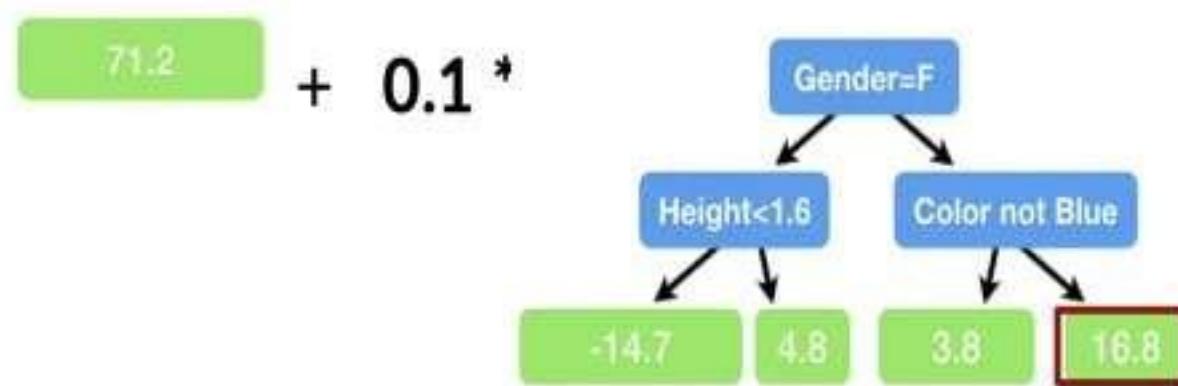
While this may seem too perfect, it is not ideal. The model fits the training data too well. It will have **low bias** and **high variance**.

This is where **learning rate** comes into picture. Gradient boost uses this new parameter to scale the contribution of the new tree. It is normally a value between 0 and 1.

# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

Using a learning rate = 0.1 in the previous case, we get:



So, the predicted weight now is  $71.2 + 0.1 \cdot 16.8 = 72.9$ .

- This is worse than the previous prediction (i.e, 88)
- But it is better than the first prediction (i.e, 71.2)

*Learning rate scales the down a factor, but towards the right direction.*

# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) | Residual |
|------------|----------------|--------|-------------|----------|
| 1.6        | Blue           | Male   | 88          | 15.1     |
| 1.6        | Green          | Female | 76          | 4.3      |
| 1.5        | Blue           | Female | 56          | -13.7    |
| 1.8        | Red            | Male   | 73          | 1.4      |
| 1.5        | Green          | Male   | 77          | 5.4      |
| 1.4        | Blue           | Female | 57          | -12.7    |

**Step 5: Calculate the residuals once again using the predicted values from the previous step, ie.,**

- $88 - [71.2 + 0.1 * (16.8)] = 15.1$
- $76 - [71.2 + 0.1 * (4.8)] = 4.3$
- $56 - [71.2 + 0.1 * (-14.7)] = -13.7$
- $73 - [71.2 + 0.1 * (3.8)] = 1.4$
- $77 - [71.2 + 0.1 * (3.8)] = 5.4$
- $57 - [71.2 + 0.1 * (-14.7)] = -12.7$

**Observe that these residuals are smaller than before, so we're headed in the right direction.**

# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

Step 6: Now build a new tree.....



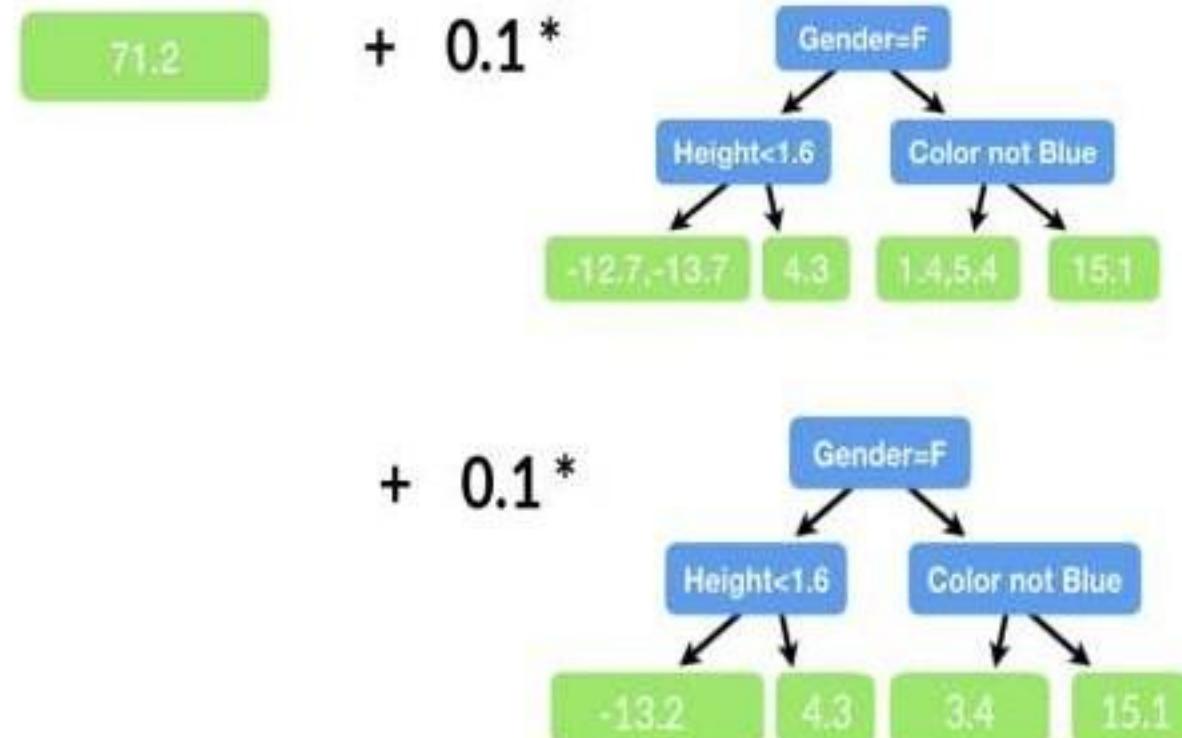
...and replace their  
leaf averages



# Ensemble Learning – Gradient Boost

| Height (m) | Favorite Color | Gender | Weight (kg) |
|------------|----------------|--------|-------------|
| 1.6        | Blue           | Male   | 88          |
| 1.6        | Green          | Female | 76          |
| 1.5        | Blue           | Female | 56          |
| 1.8        | Red            | Male   | 73          |
| 1.5        | Green          | Male   | 77          |
| 1.4        | Blue           | Female | 57          |

Step 7: Now add the new tree to the previous 2 trees (but don't forget to add the scaling factor!) and predict new values



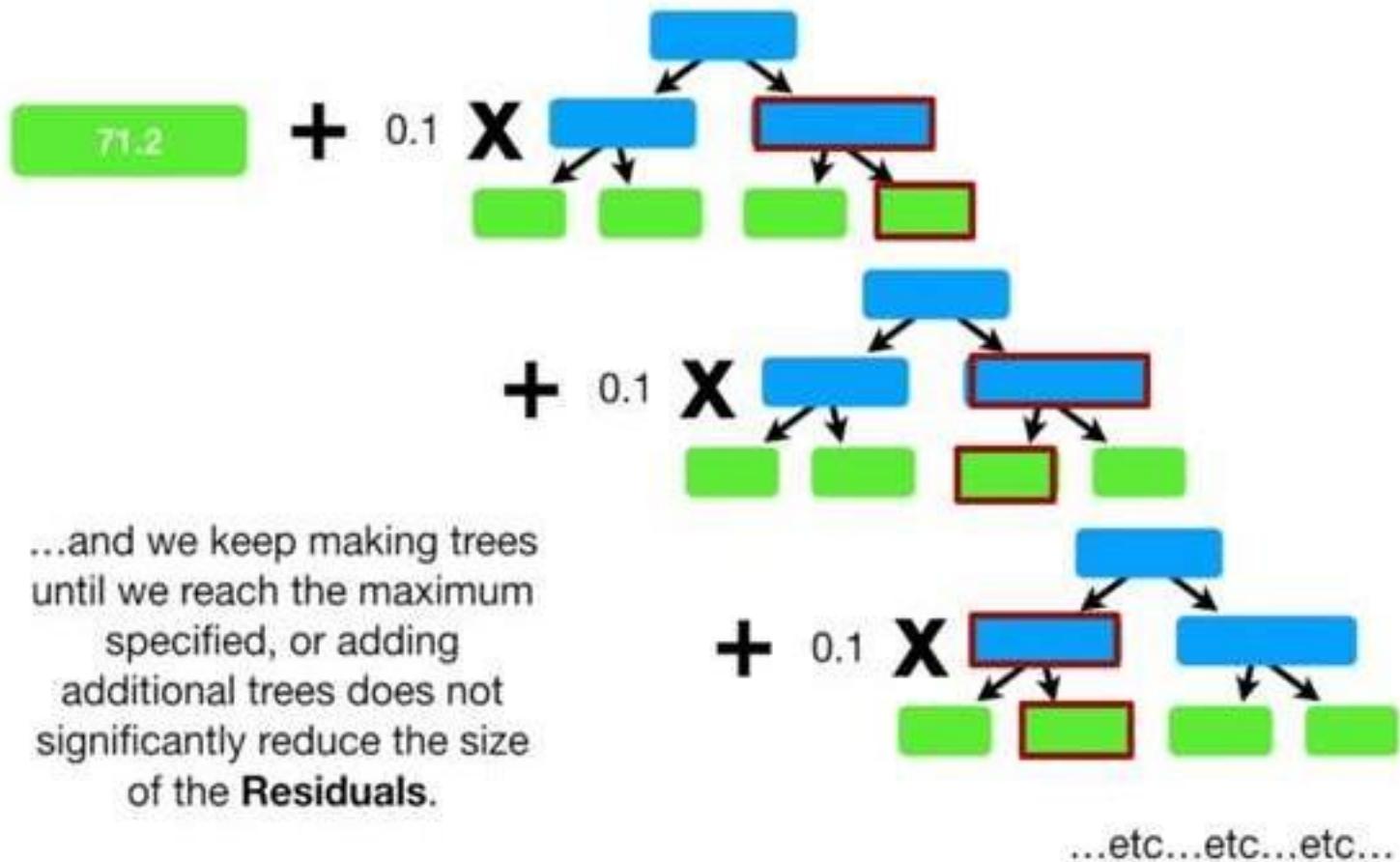
# Ensemble Learning – Gradient Boost

| Residual | Residual | Residual |
|----------|----------|----------|
| 16.8     | 15.1     | 13.6     |
| 4.8      | 4.3      | 3.9      |
| -15.2    | -13.7    | -12.4    |
| 1.8      | 1.4      | 1.1      |
| 5.8      | 5.4      | 5.1      |
| -14.2    | -12.7    | -11.4    |

## Step 8: Calculate the new residual values

Observe that the residual values continuously decrease and head towards the correct values.

# Ensemble Learning – Gradient Boost





**THANK YOU**

---

**Surabhi Narayan**

Department of Computer Science &Engineering

**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



# Machine Learning

## Artificial Neural Network

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**

**Teaching Assistant: Nettem Gayathri (Semester VII)**

# Machine Learning

## Acknowledgement

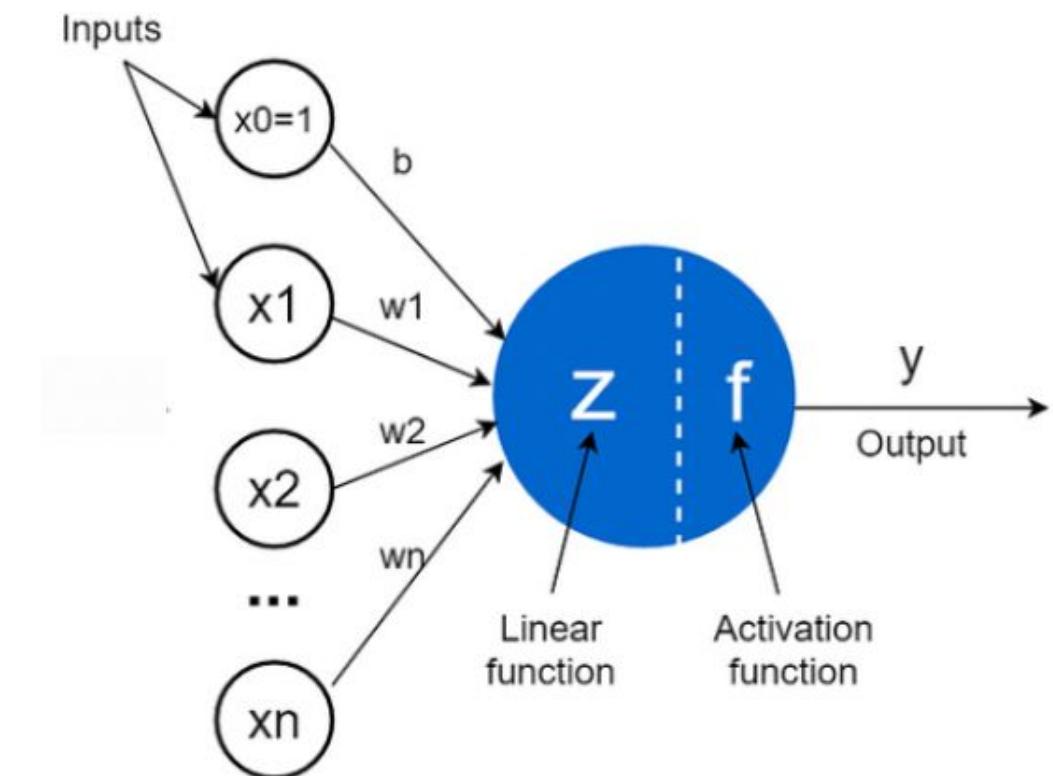
---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Introduction to ANN

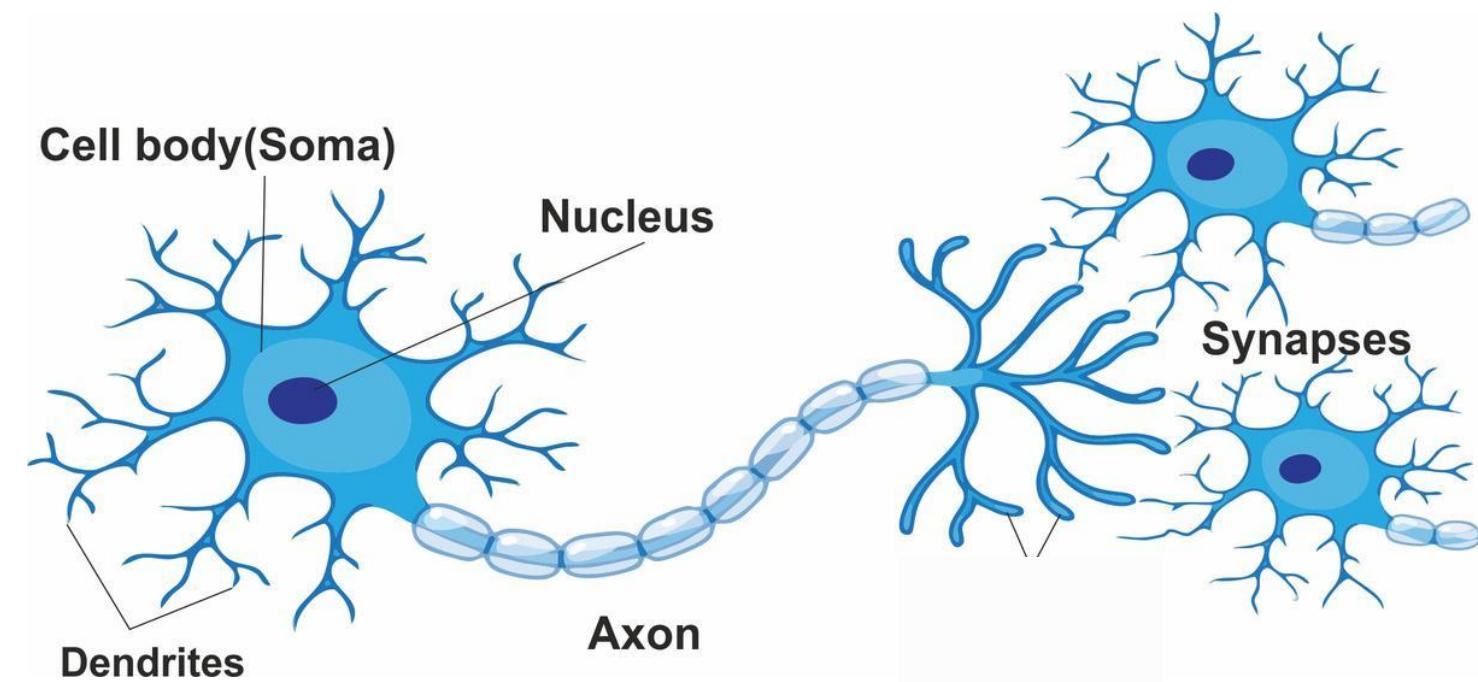
- Artificial neural networks or Deep neural networks are one of the main tools used in machine learning.
- The most fundamental unit of a DNN is called an **artificial neuron**.
- As the “neural” part of their name suggests, they are **brain-inspired** systems that are intended to replicate the way that we humans learn.
- Biological neurons = neural cells = neural processing units.



**Artificial  
Neuron**

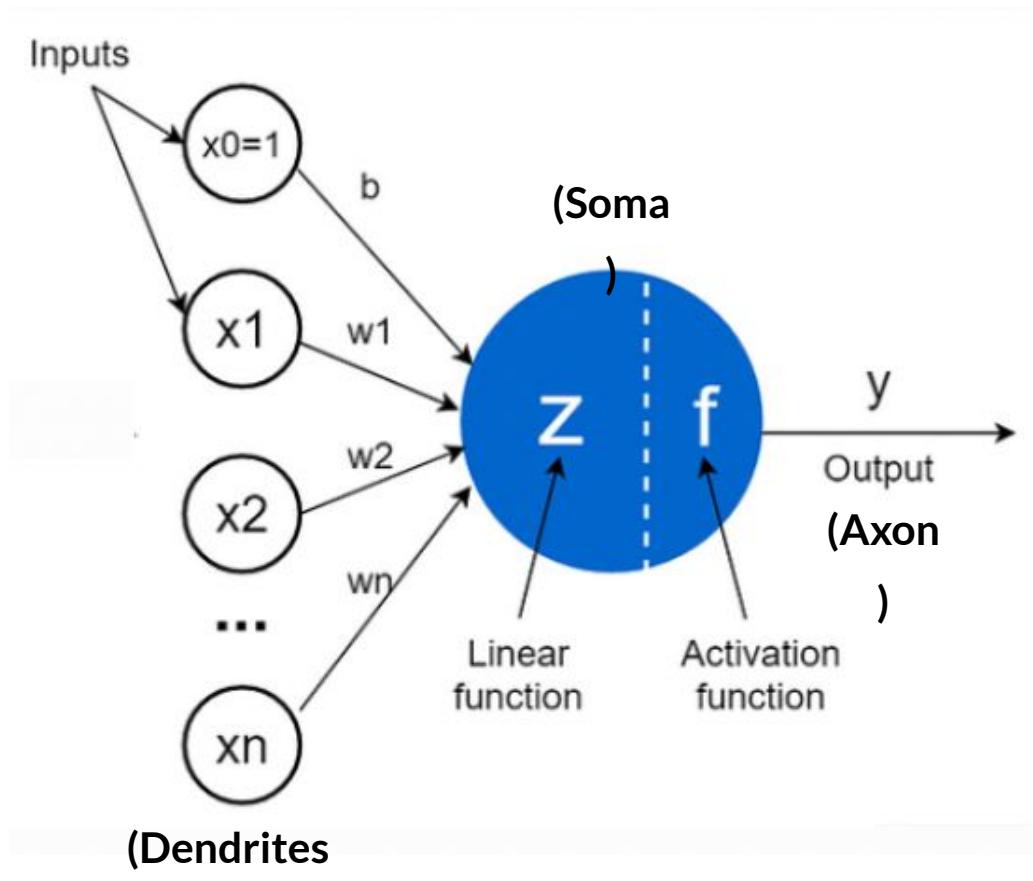
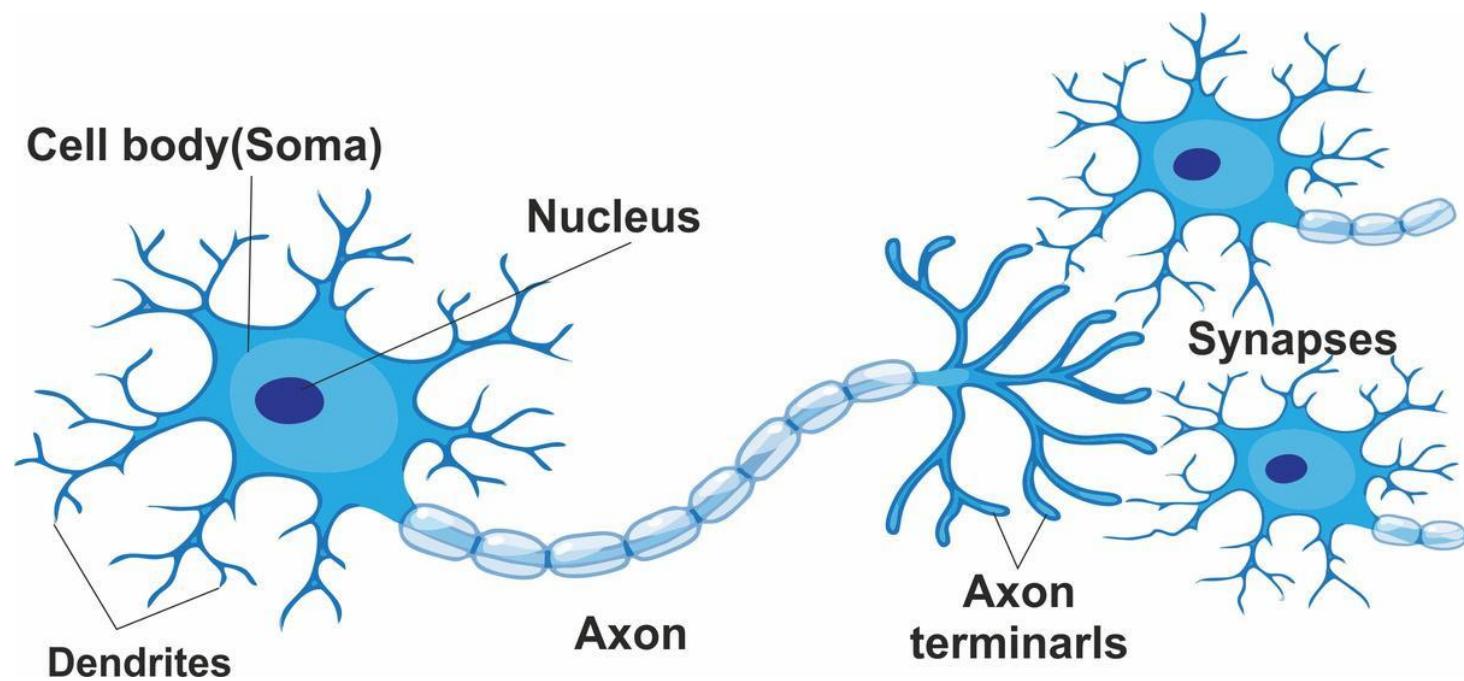
# Biological Neurons

- Dendrite: Obtain information from other cells and carry that information to the cell body. (input)
- Soma: The neuron's control center, processes the information. (processing)
- Axon: It carries messages away from the soma to the other neurons. (output)
- Synapse: The tiny gap between one neuron's axon and another neuron's dendrite where they can pass messages to communicate. It is a point of connection between two



**Biological Neuron**

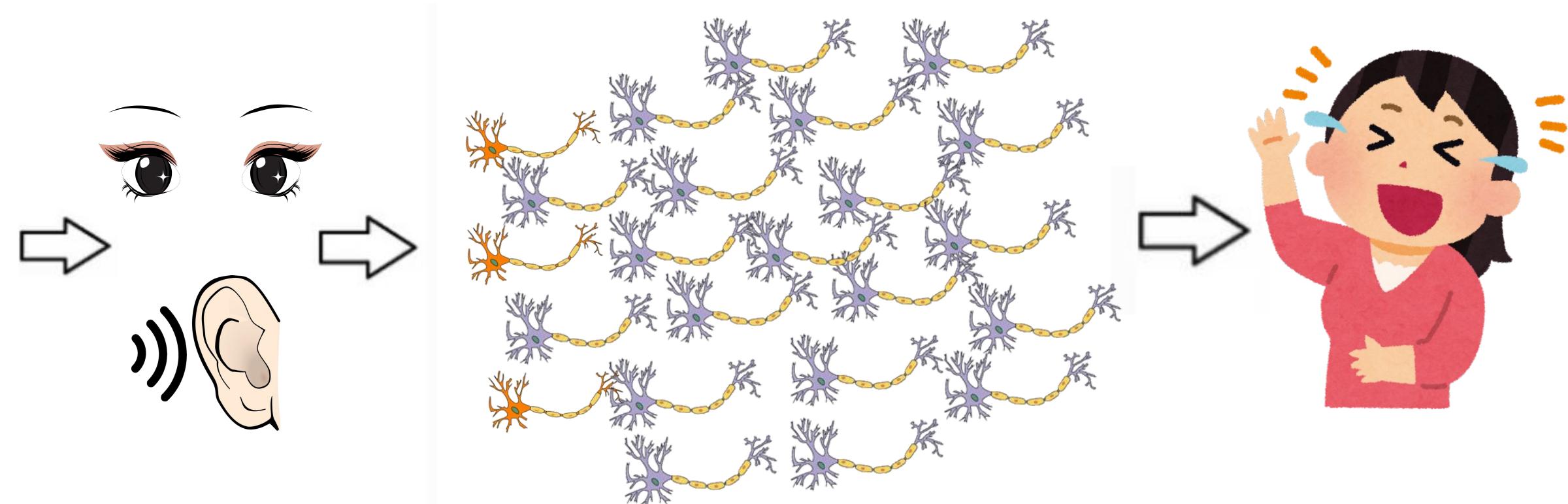
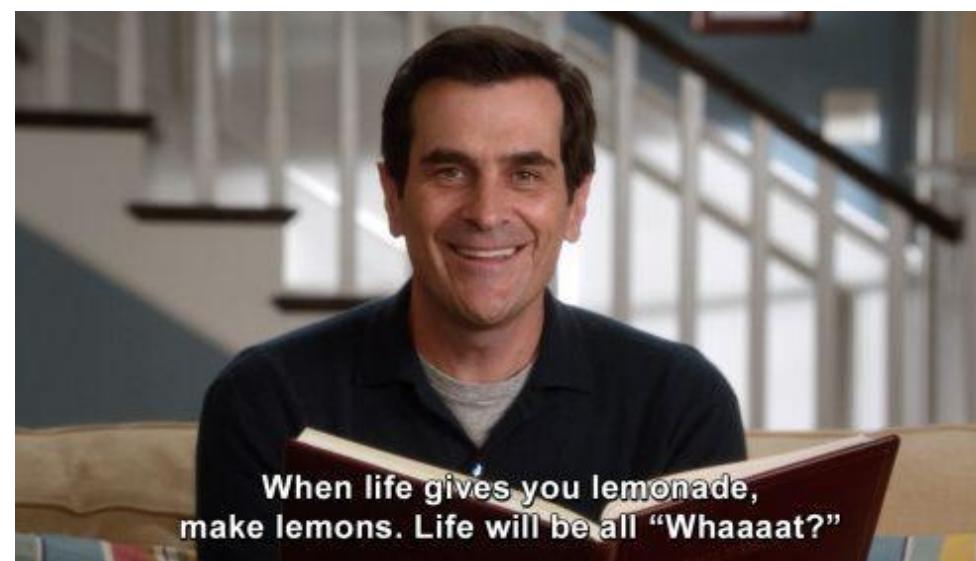
# Biological Neurons



| Biological Neuron | Artificial Neuron |
|-------------------|-------------------|
| Dendrites         | Input             |
| Soma              | Node              |
| Axon              | Output            |
| Synapses          | Interconnections  |

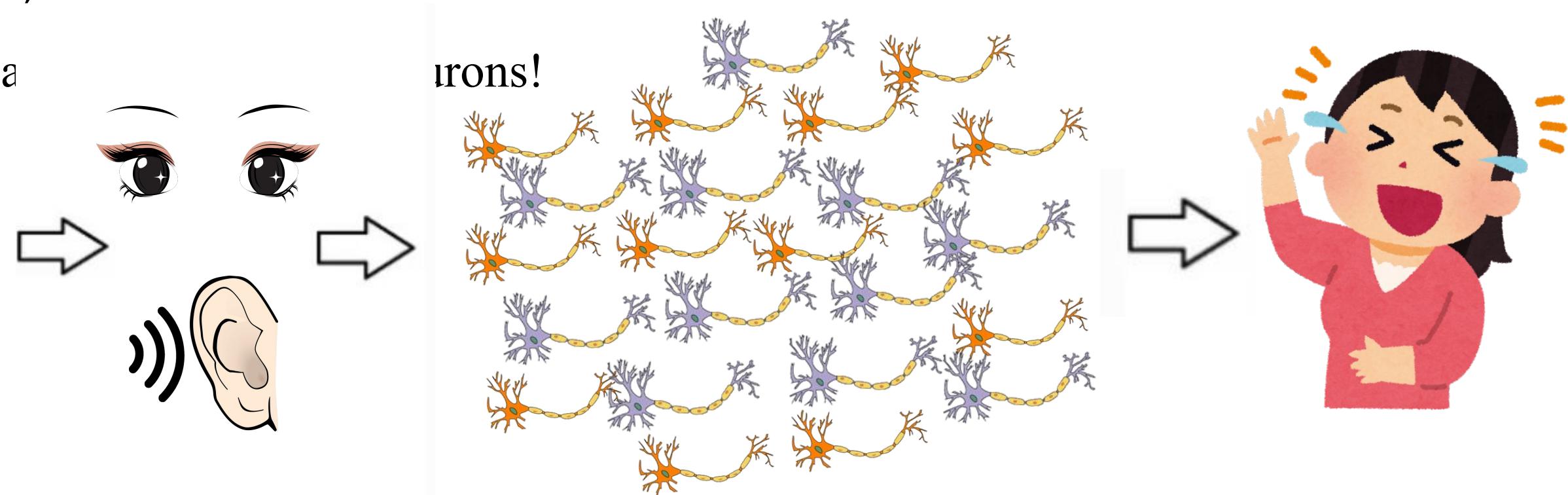
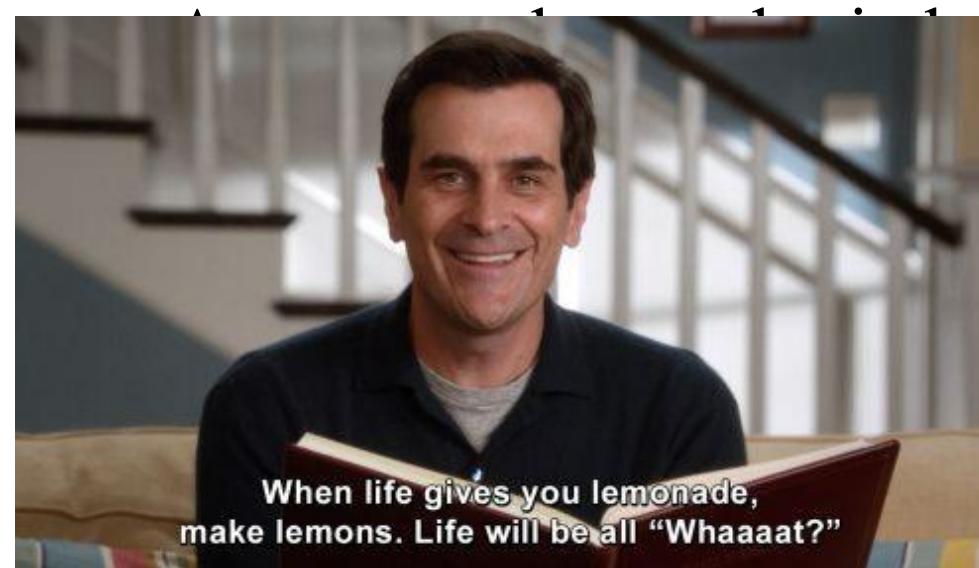
# Simplified Working of a Biological Neuron

- Our sense organs interact with the outer world and relay information to the neurons.
- There is a massively parallel interconnected network of neurons.
- The sense organs relay information to the lowest layer of neurons
- Some of these neurons may fire (in red) in response to this information and in turn relay information to other neurons they are connected to.



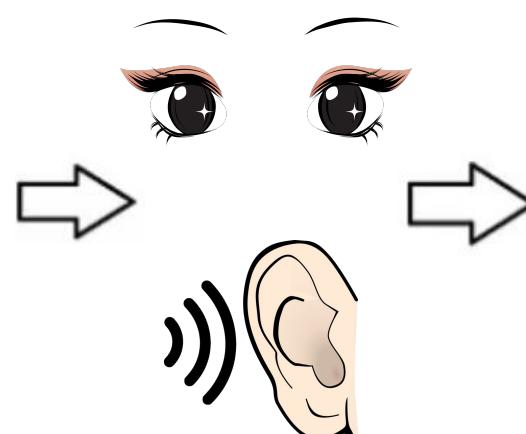
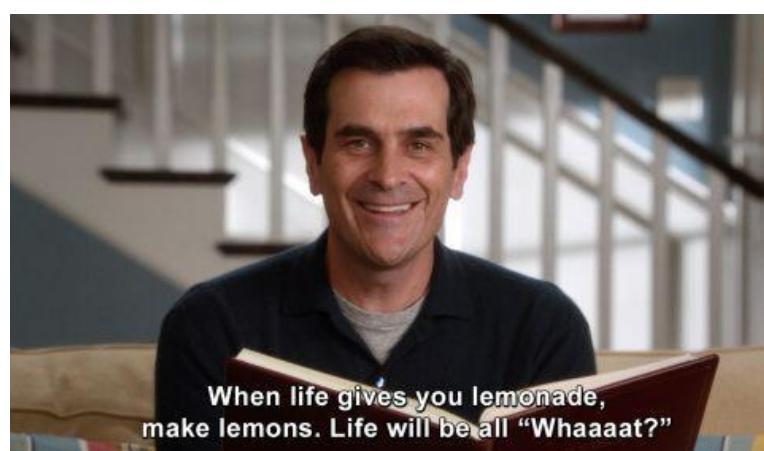
# Simplified Working of a Biological Neuron

- Our sense organs interact with the outer world and relay information to the neurons.
- There is a massively parallel interconnected network of neurons.
- The sense organs relay information to the lowest layer of neurons
- Some of these neurons may fire (in red) in response to this information and in turn relay information to other neurons they are connected to.
- These neurons may also fire (again, in red) and the process continues eventually resulting in a response (laughter in this case)



# Simplified Working of a Biological Neuron

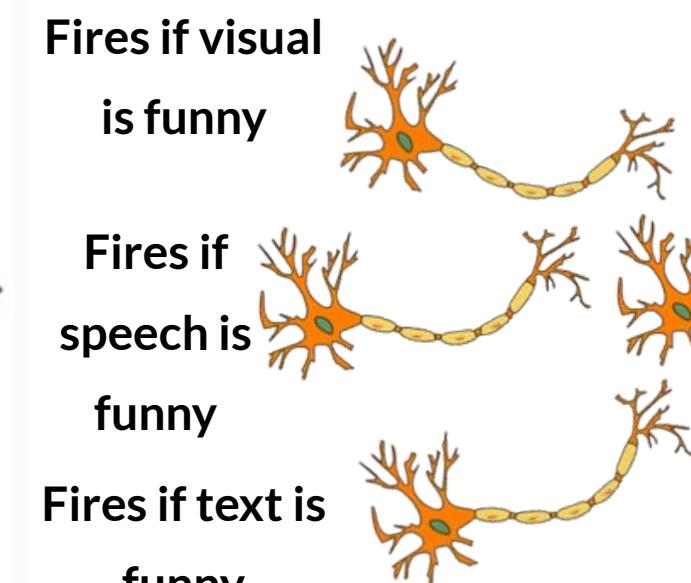
- This massively parallel network also ensures that there is a division of work
- It is believed that neurons are arranged hierarchically and each layer has its own role and responsibility.
- Each neuron may perform a certain role or respond to a certain stimulus
- To detect a face, the brain could be relying on the entire network and not on a single layer.



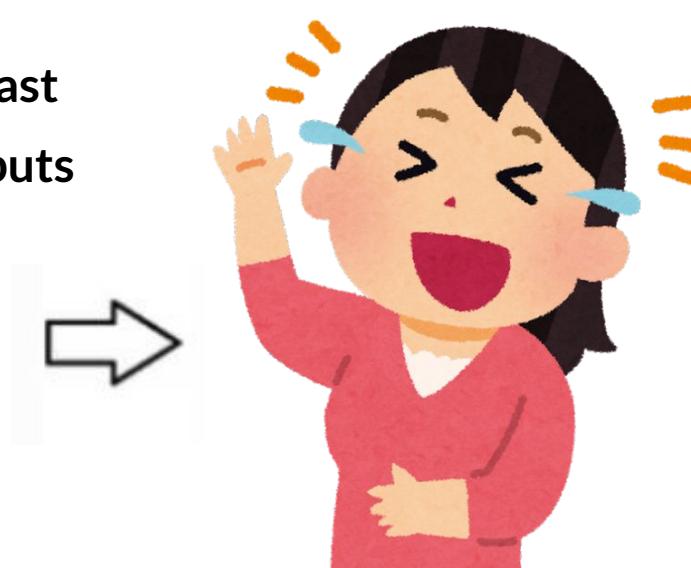
Fires if visual  
is funny

Fires if  
speech is  
funny

Fires if text is  
funny



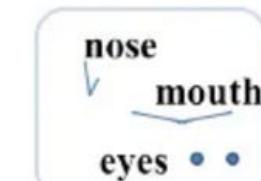
Fires if at least  
2 of the 3 inputs  
fired



## Hierarchical processing



Layer 1: detect edges & corners



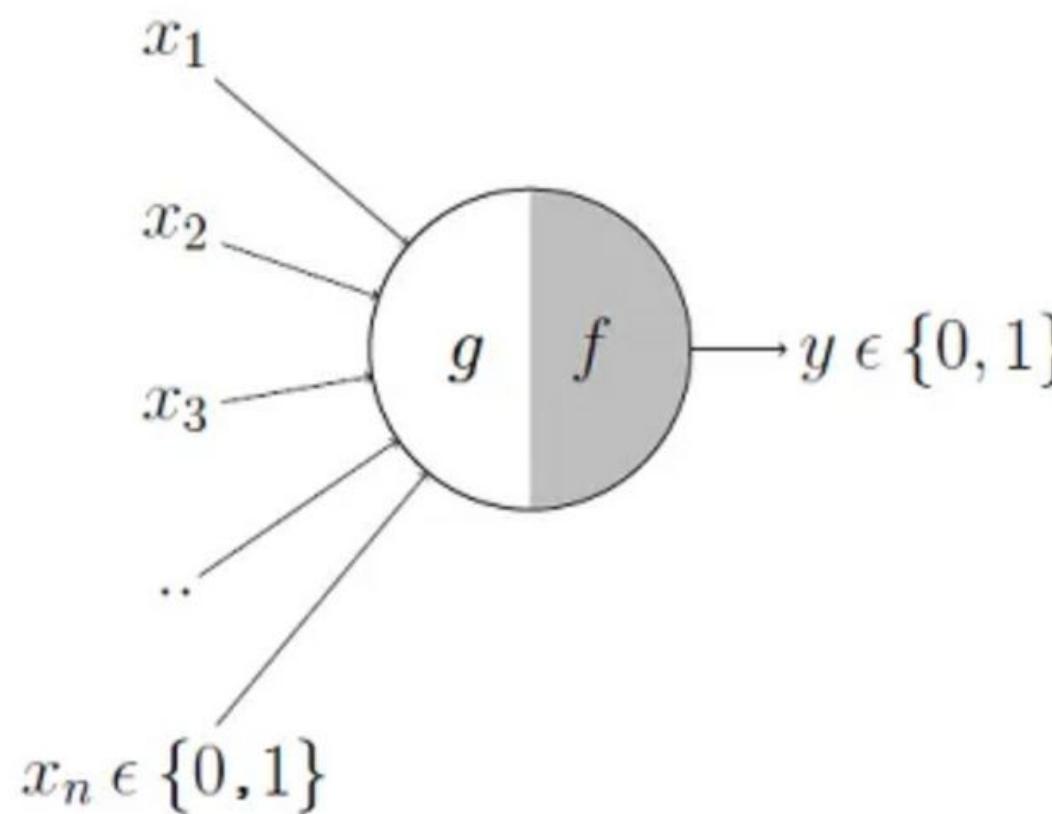
Layer 2: form feature groups



Layer 3: detect high level  
objects, faces, etc.

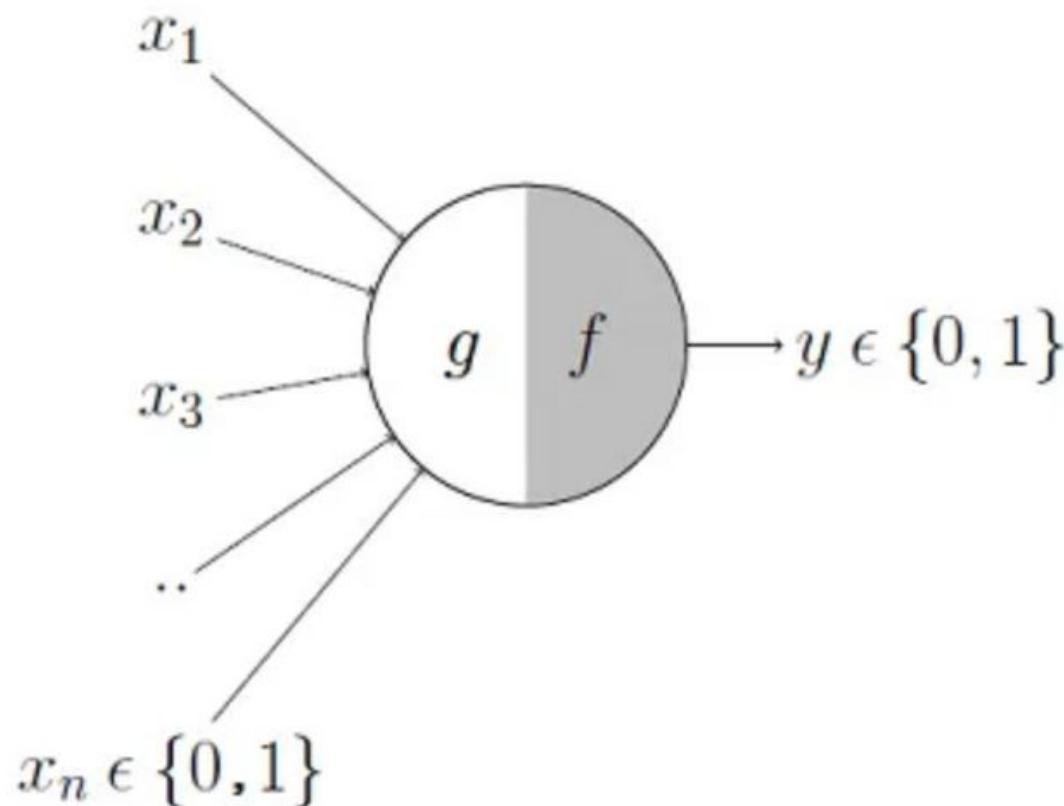
# McCulloch-Pitts Neuron

- The first computational model of a neuron was proposed by Warren McCulloch (neuroscientist) and Walter Pitts (logician) in 1943.
- This neuron takes in **multiple binary inputs** and produces a **binary output**.
- $g$  aggregates the inputs, and based on the aggregated value the second function,  $f$  makes a decision.
- The inputs can be excitatory or inhibitory.
- **Inhibitory inputs** are those that have maximum effect on the decision-making irrespective of other inputs.
- **Excitatory inputs** are NOT the ones that will make the neuron fire on their own but they might fire it when combined.



# McCulloch-Pitts Neuron - Thresholding logic

- $y = 0$  if any  $x_i$  is inhibitory, else



$$g(x_1, x_2, x_3, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 & \text{if } g(\mathbf{x}) \geq \theta \\ &= 0 & \text{if } g(\mathbf{x}) < \theta \end{aligned}$$

- $\theta$  is called the **thresholding parameter**.

# McCulloch-Pitts Neuron - Thresholding logic

Imagine you want to predict whether you'll watch a random F1 race. The inputs are all boolean, i.e.,  $\{0,1\}$ , and the output is also boolean  $\{0: \text{Will watch it}, 1: \text{Won't watch it}\}$ .

For instance:

- $x_1$ : isGrandPrix (vs. a practice or qualifying session).
- $x_2$ : isItAFamousCircuit (Is the race in a country you find interesting?).
- $x_3$ : isBusy (Won't watch if you have other commitments).
- $x_4$ : isFavoriteDriverRacing (Is your favourite driver participating?).

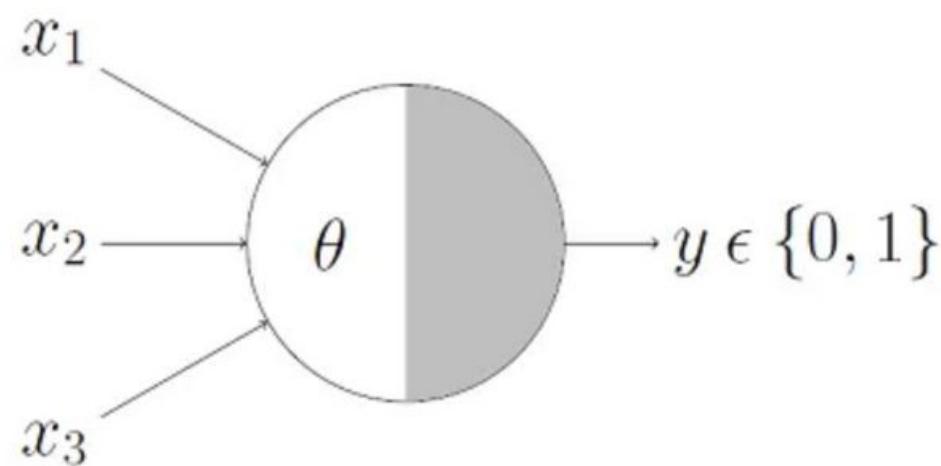
if  $x_3$  (isBusy) is 1 (meaning you are busy), then your output will always be 0 (you won't watch the race), making  $x_3$  an inhibitory input.

Formally, the decision-making process can be expressed as follows:

$$g(x) = x_1 + x_2 + x_4$$

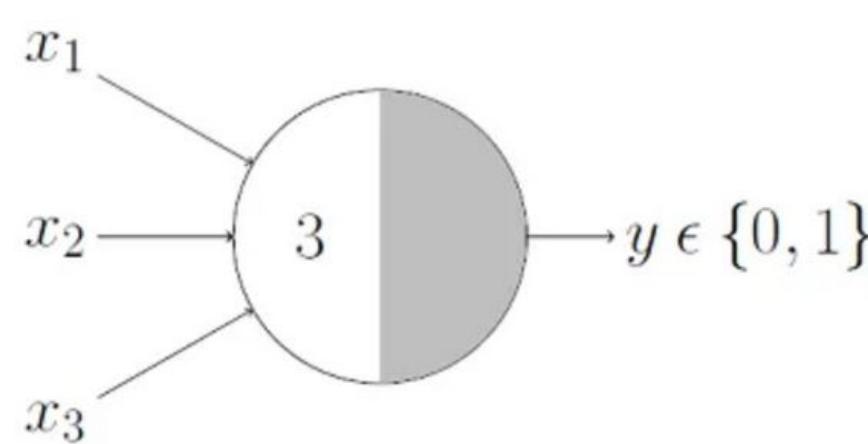
If the sum of the inputs exceeds a certain threshold  $\theta$ , you'll decide to watch the race

# Boolean functions using MP neuron - AND



A McCulloch Pitts unit

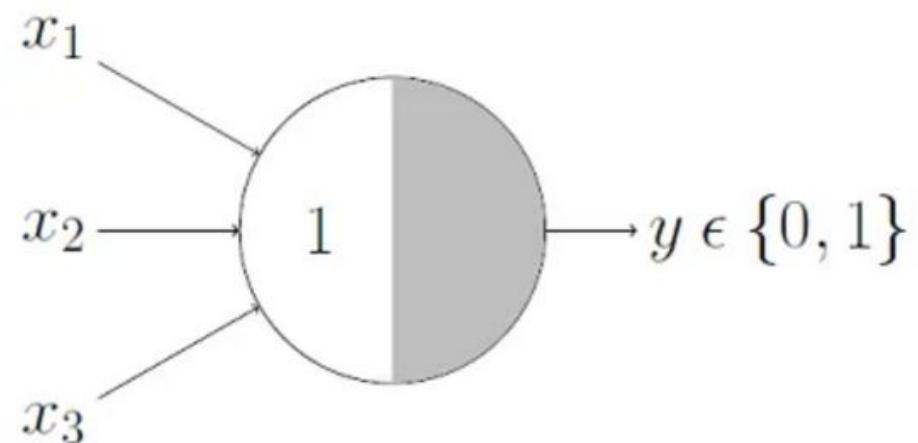
For the boolean inputs  $x_1$ ,  $x_2$  and  $x_3$  if the  $g(x)$  i.e., sum  $\geq \theta$ , the neuron will fire otherwise, it won't.



**AND Function**  
**Fires when ALL the**  
**inputs are ON i.e.,  $g(x) \geq$**   
**3 here.**

| x1 | x2 | x3 | y |
|----|----|----|---|
| 0  | 0  | 0  | 0 |
| 0  | 0  | 1  | 0 |
| 0  | 1  | 0  | 0 |
| 0  | 1  | 1  | 0 |
| 1  | 0  | 0  | 0 |
| 1  | 0  | 1  | 0 |
| 1  | 1  | 0  | 0 |
| 1  | 1  | 1  | 1 |

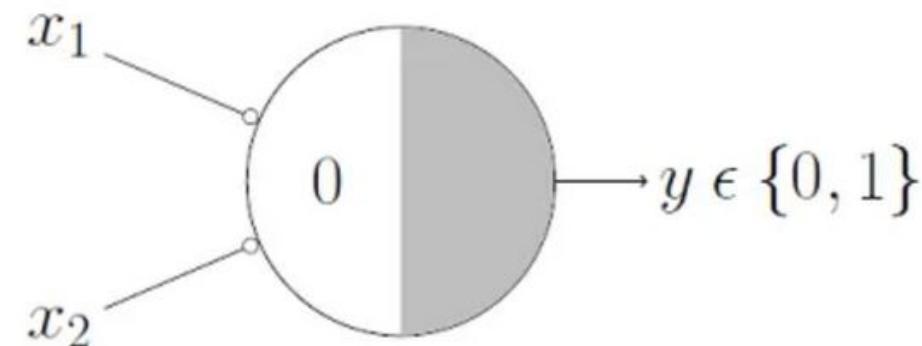
# Boolean functions using MP neuron - OR



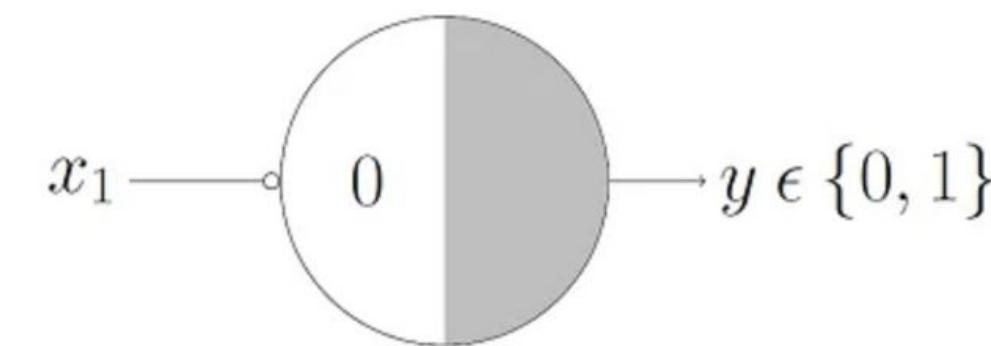
**OR Function**  
Fires if ANY of the inputs is  
ON i.e.,  $g(x) \geq 1$  here.

| $x_1$ | $x_2$ | $x_3$ | $y$ |
|-------|-------|-------|-----|
| 0     | 0     | 0     | 0   |
| 0     | 0     | 1     | 1   |
| 0     | 1     | 0     | 1   |
| 0     | 1     | 1     | 1   |
| 1     | 0     | 0     | 1   |
| 1     | 0     | 1     | 1   |
| 1     | 1     | 0     | 1   |
| 1     | 1     | 1     | 1   |

# Boolean functions using MP neuron - NOR and NOT

**NOR Function**

| x1 | x2 | y |
|----|----|---|
| 0  | 0  | 1 |
| 0  | 1  | 0 |
| 1  | 0  | 0 |
| 1  | 1  | 0 |

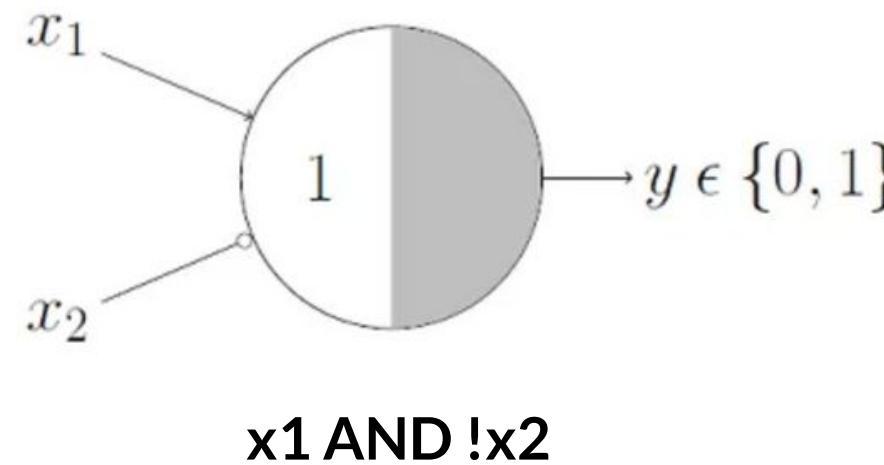
**NOT Function**

| x1 | y |
|----|---|
| 0  | 1 |
| 1  | 0 |

Take the input as an inhibitory input and set the thresholding parameter to 0.

ALL the inputs should be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input.

# A Function With Inhibitory Input



Here, we have an inhibitory input i.e., x<sub>2</sub> so whenever x<sub>2</sub> is 1, the output will be 0.

x<sub>1</sub> AND !x<sub>2</sub> would output 1 only when x<sub>1</sub> is 1 and x<sub>2</sub> is 0, so the threshold parameter should be 1.

$g(x) = x_1 + x_2$  would be  $\geq 1$  in only 3 cases:

Case 1: when x<sub>1</sub> is 1 and x<sub>2</sub> is 0

Case 2: when x<sub>1</sub> is 1 and x<sub>2</sub> is 1

Case 3: when x<sub>1</sub> is 0 and x<sub>2</sub> is 1

But in both Case 2 and Case 3, we know that the output will be 0

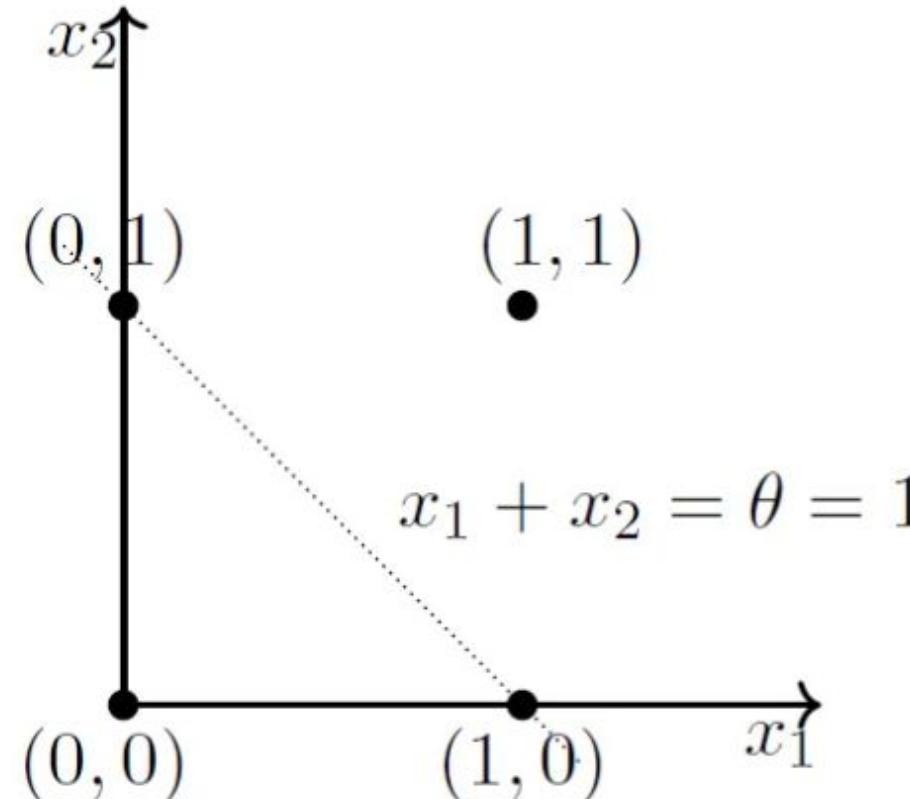
because x<sub>2</sub> is 1 in both, thanks to the inhibition. We also know that x<sub>1</sub>

AND !x<sub>2</sub> would output 1 for Case 1 so our thresholding parameter

| x <sub>1</sub> | x <sub>2</sub> | y |
|----------------|----------------|---|
| 0              | 0              | 0 |
| 0              | 1              | 0 |
| 1              | 0              | 1 |
| 1              | 1              | 0 |

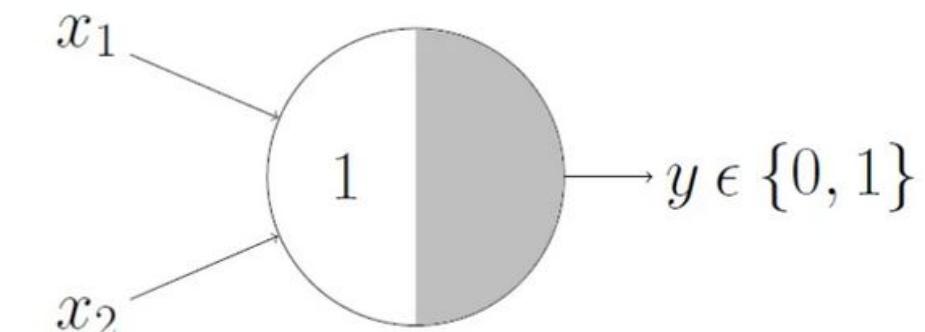
## Geometric Interpretation Of MP Neuron - OR function

- The  $x_1$  and  $x_2$  axes represent only Boolean inputs (0 or 1), not real numbers.
- The line  $x_1+x_2=1$  represents a decision boundary. It passes through points (1, 0) and (0, 1).
  - Inputs with output 0 lie below the line.
  - Inputs with output 1 lie on or above the line.



$$\left( \sum_{i=1}^n x_i < \theta \right)$$

$$\left( \sum_{i=1}^n x_i \geq \theta \right)$$



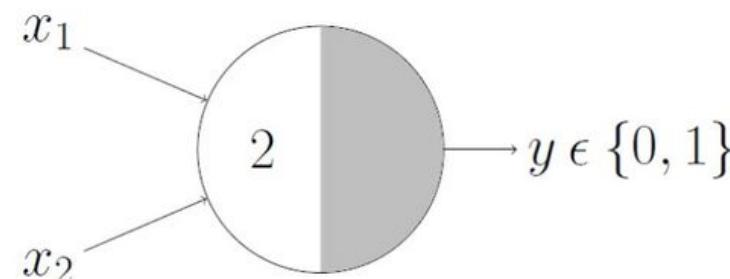
*OR function*

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$

The MP neuron splits the input sets into two classes — positive and negative. Positive ones (which output 1) are those that lie ON or ABOVE the decision boundary and negative ones (which output 0) are those that lie BELOW the decision boundary.

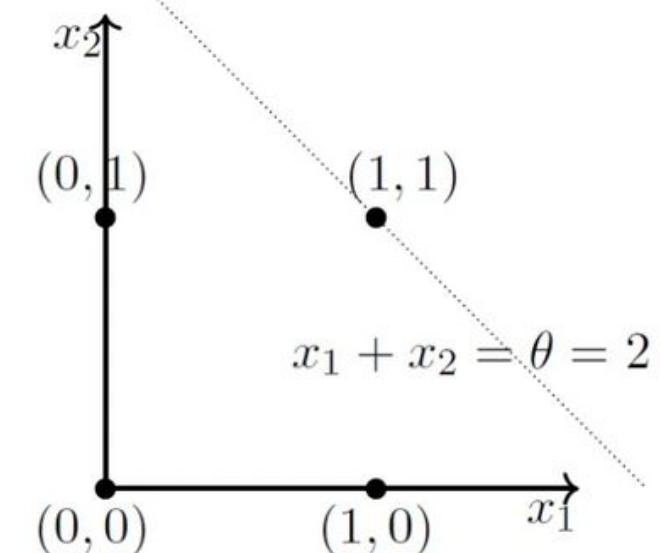
# Geometric Interpretation Of MP Neuron

## AND Function



*AND function*

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$



- The line  $x_1+x_2=2$  represents a decision boundary. It passes through point  $(1, 1)$ .

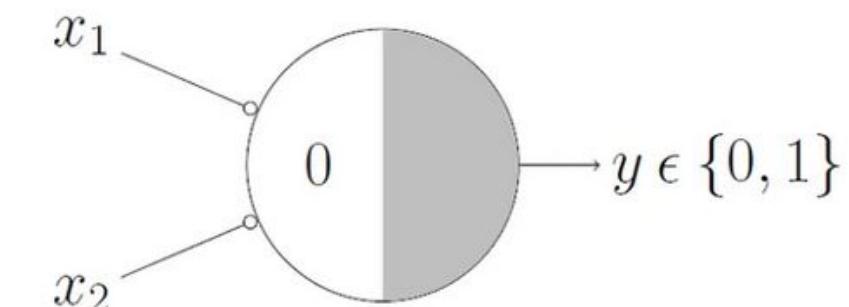
- Inputs with output 0 lie below the line.

$$\left( \sum_{i=1}^n x_i < \theta \right)$$

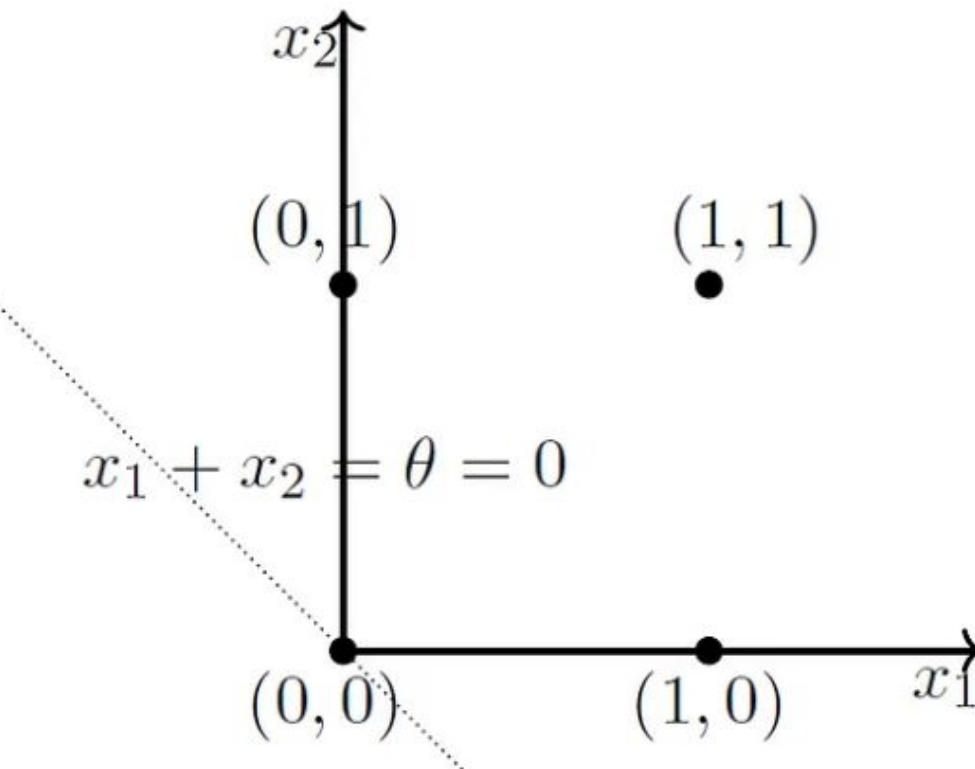
- Inputs with output 1 lie on or above the line.

$$\left( \sum_{i=1}^n x_i \geq \theta \right)$$

## Tautology



*Tautology (always ON)*



- A single MP Neuron can be used to represent Boolean functions which are linearly separable (for Boolean functions)
- There exists a line (plane) such that all inputs that produce a 1 lie on one side of the line (plane) and all inputs that produce a 0 lie on the other side of the line (plane).

## Limits of MP Neuron

---

- Input must be binary (what if the input attribute is real-valued or has more than 2 classes?)
- All inputs are treated equally (what if a few input attributes are more important?)
- The threshold is hand-coded (Can it be learned?)
- Works only for (boolean) functions which are linearly separable.
- The output is binary (categorical)



# THANK YOU

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
**UNIVERSITY**

CELEBRATING 50 YEARS

# Machine Learning

## Perceptron

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**

**Teaching Assistant: Nettem Gayathri (Semester VII)**

# Machine Learning

## Acknowledgement

---



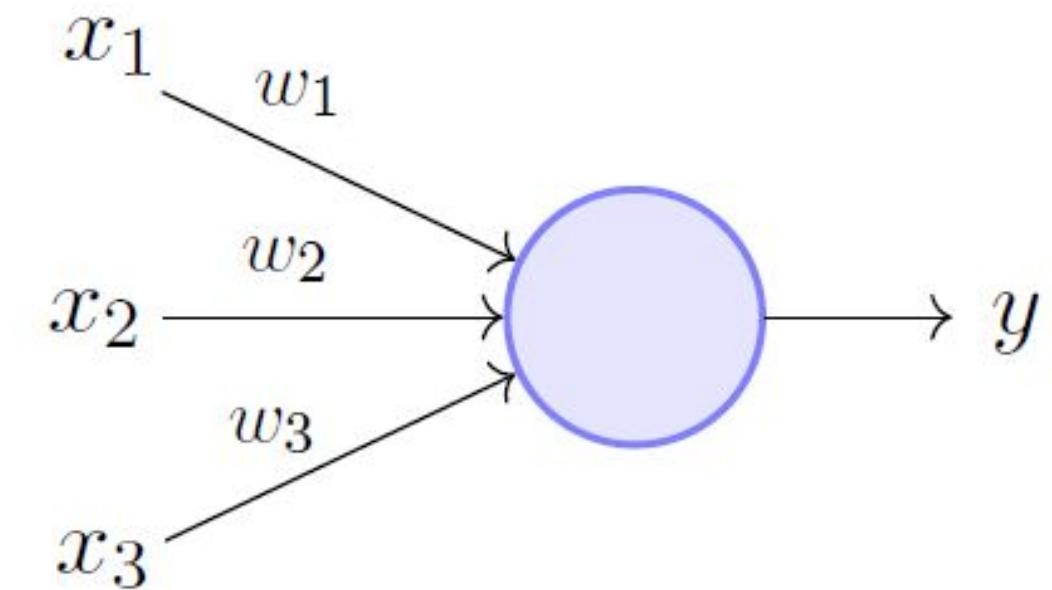
- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Perceptron

- Frank Rosenblatt proposed the classical perceptron model (1958).
- A more general computational model than MP neurons.

Primary differences:

- Introduction of numerical weights for inputs and a mechanism for learning these weights and the bias.
- Inputs are no longer limited to Boolean values.

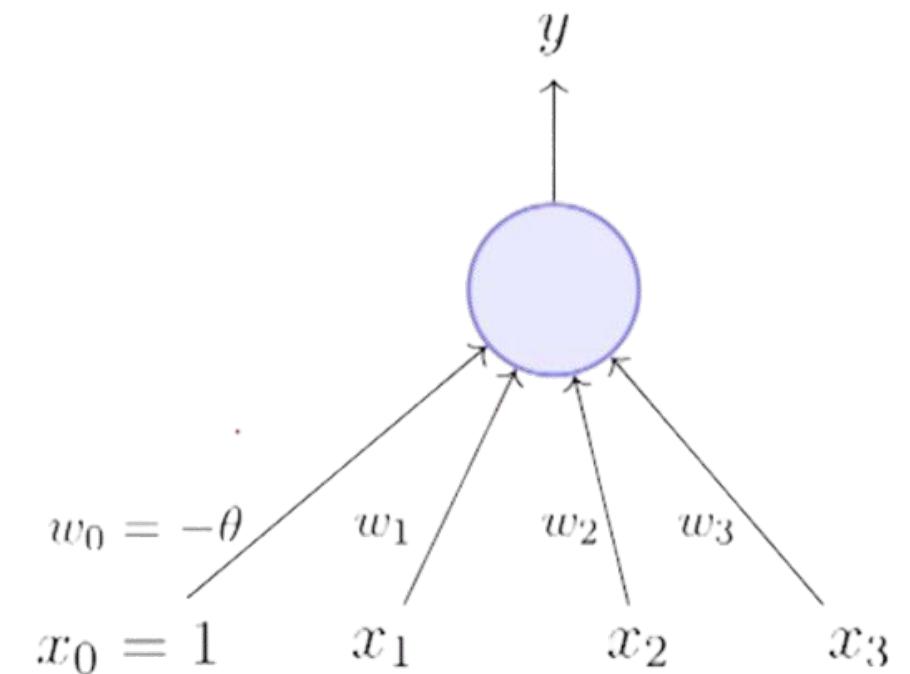


Perceptron Model (Minsky-Papert in 1969)

# UE22CS352A -Machine Learning

## Perceptron

- Consider a single perceptron,  $w_0$  is the bias and  $x_0 = 1$ .
- $w_1, w_2 \dots$  are weights associated with the inputs  $x_1, x_2 \dots$
- $y$  is the output.



$$y = 1 \quad if \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^n w_i * x_i < \theta$$



$$y = 1 \quad if \sum_{i=1}^n w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^n w_i * x_i - \theta < 0$$

A more accepted convention -

$$y = 1 \quad if \sum_{i=0}^n w_i * x_i \geq 0$$

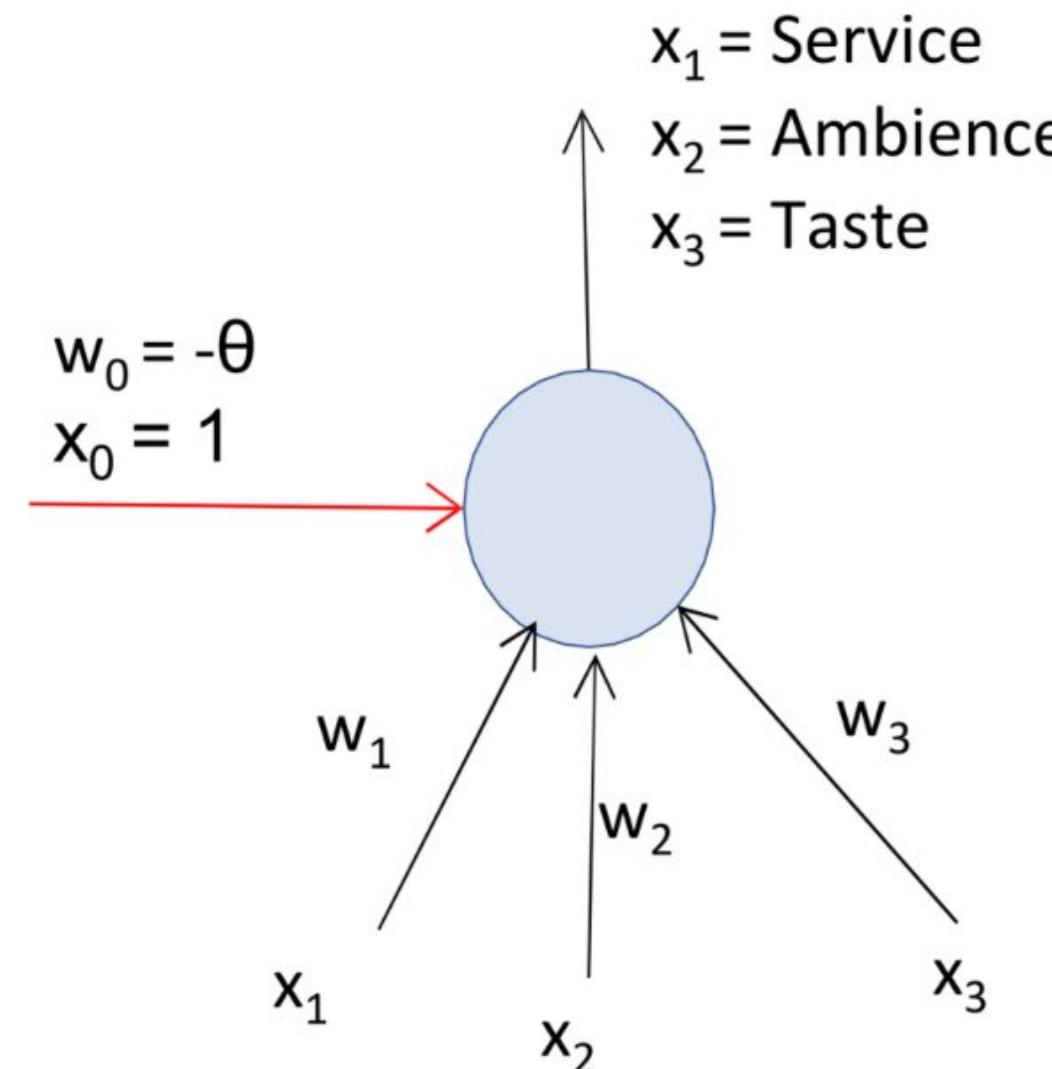
$$= 0 \quad if \sum_{i=0}^n w_i * x_i < 0$$

where,  $x_0 = 1$  and  $w_0 = -\theta$

$$y = 1 \ if \ w^t + b \geq 0$$

$$= 0 \ if \ w^t + b < 0$$

# Perceptron - an intuitive example



- Consider the task of predicting whether we would dine in at a restaurant.
- The decision is based on 3 inputs. We assign a higher weight to taste based on past dining experiences.
- $w_0$  is called the bias, or prejudice, in the decision-making process.
- A foodie with a low threshold may dine at any restaurant regardless of service, ambience, or taste [ $\theta = 0$ ].
- On the other hand, a food critic may only dine at restaurants with ratings of 4 stars or higher for service, taste, and ambience [ $\theta = 3$ ].
- The weights ( $w_1, w_2, \dots$ ) and the bias( $w_0$ ) would depend on the previous data (based on past dining experiences in this case).

# Perceptron vs McCulloch-Pitts Neuron

## McCulloch Pitts Neuron (assuming no inhibitory inputs)

$$\begin{aligned} y &= 1 \quad if \sum_{i=0}^n x_i \geq 0 \\ &= 0 \quad if \sum_{i=0}^n x_i < 0 \end{aligned}$$

## Perceptron

$$\begin{aligned} y &= 1 \quad if \sum_{i=0}^n \textcolor{red}{w}_i * x_i \geq 0 \\ &= 0 \quad if \sum_{i=0}^n \textcolor{red}{w}_i * x_i < 0 \end{aligned}$$

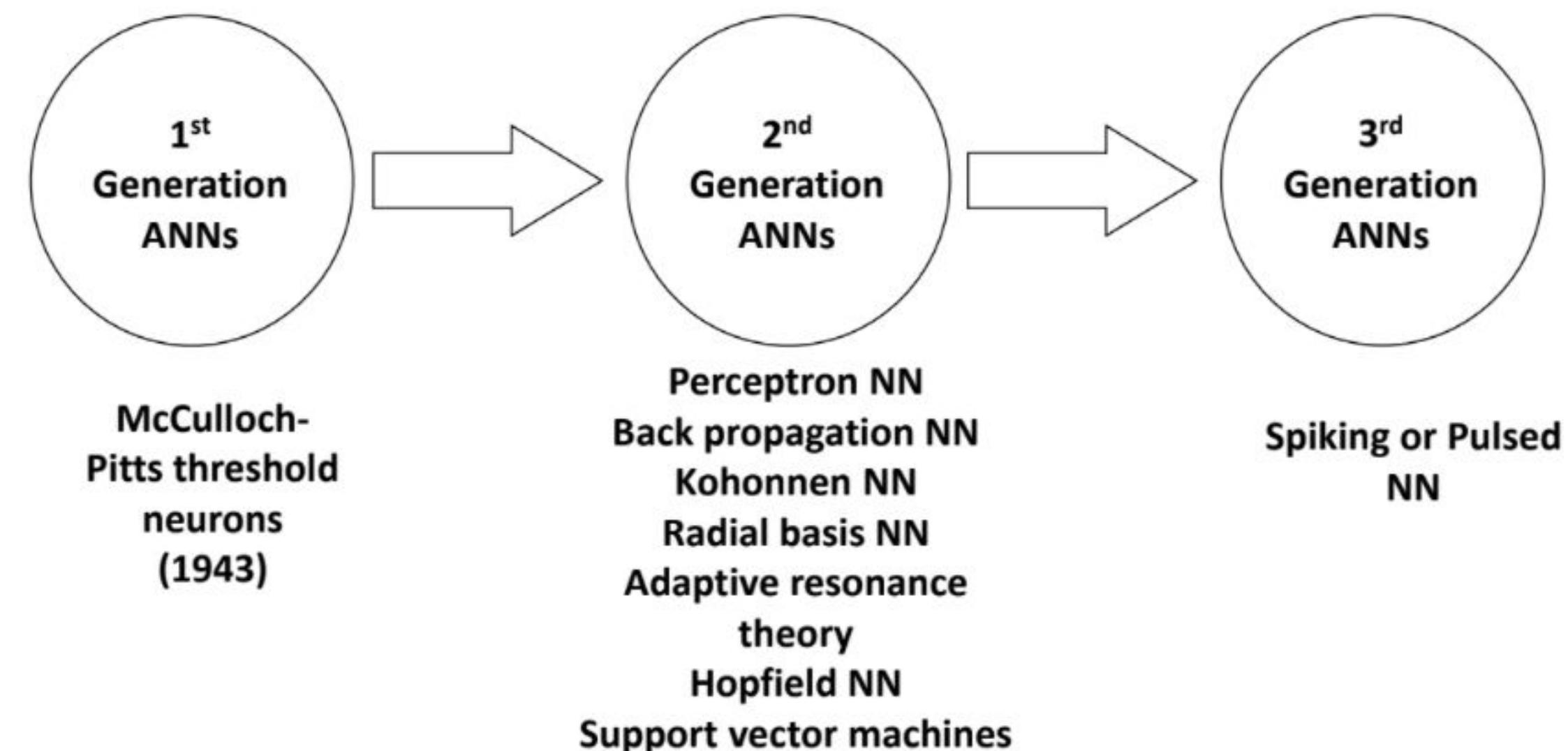
## Similarities

- Both perceptrons and MP neurons separate the input space into two halves.
- All inputs which produce a 1 lie on one side and all inputs which produce a 0 lie on the other side.
- The output can only be binary (0/1).

## Difference

The weights (including threshold) can be learned and the inputs can be real-valued for a perceptron.

# Evolution of Artificial Neural Networks



# Boolean Functions Using Perceptron - OR Function

| x1 | x2 | OR |                                     |
|----|----|----|-------------------------------------|
| 0  | 0  | 0  | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |
| 0  | 1  | 1  | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |
| 1  | 0  | 1  | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |
| 1  | 1  | 1  | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |

On expanding these, we get a system of linear inequalities

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \Rightarrow w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \Rightarrow w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \Rightarrow w_1 \geq -w_0$$

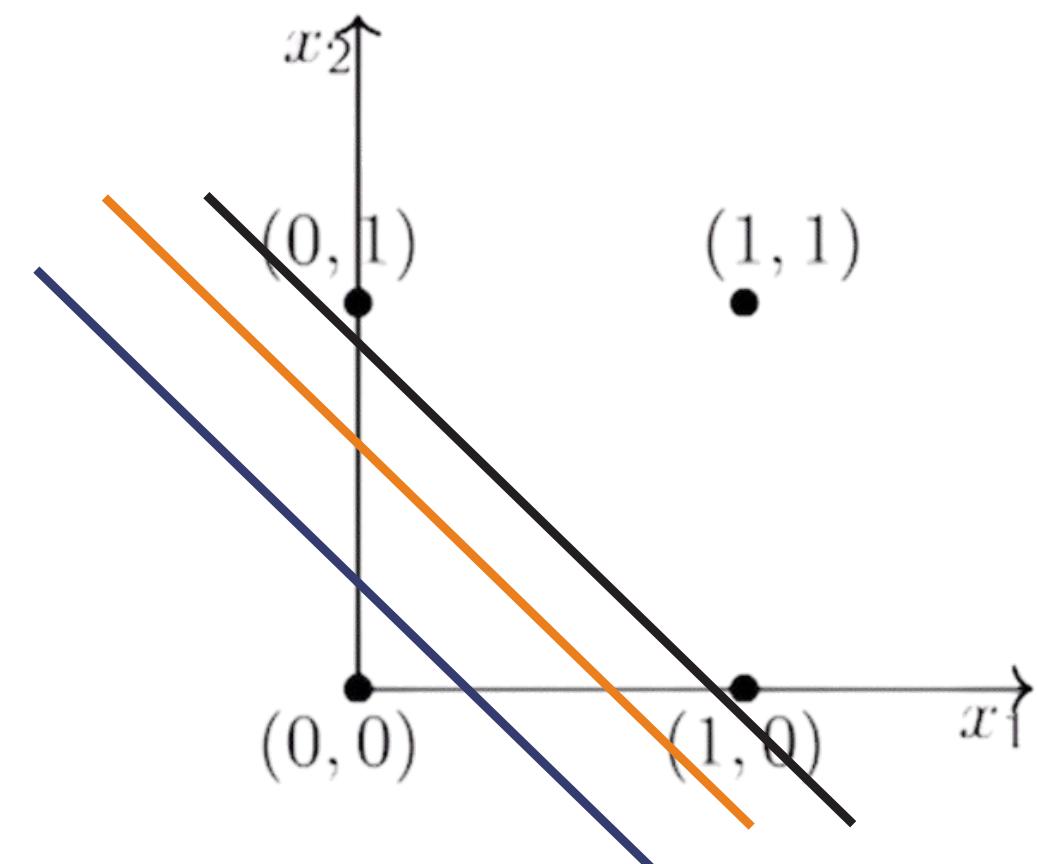
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \Rightarrow w_1 + w_2 \geq -w_0$$

# Boolean Functions Using Perceptron - OR Function

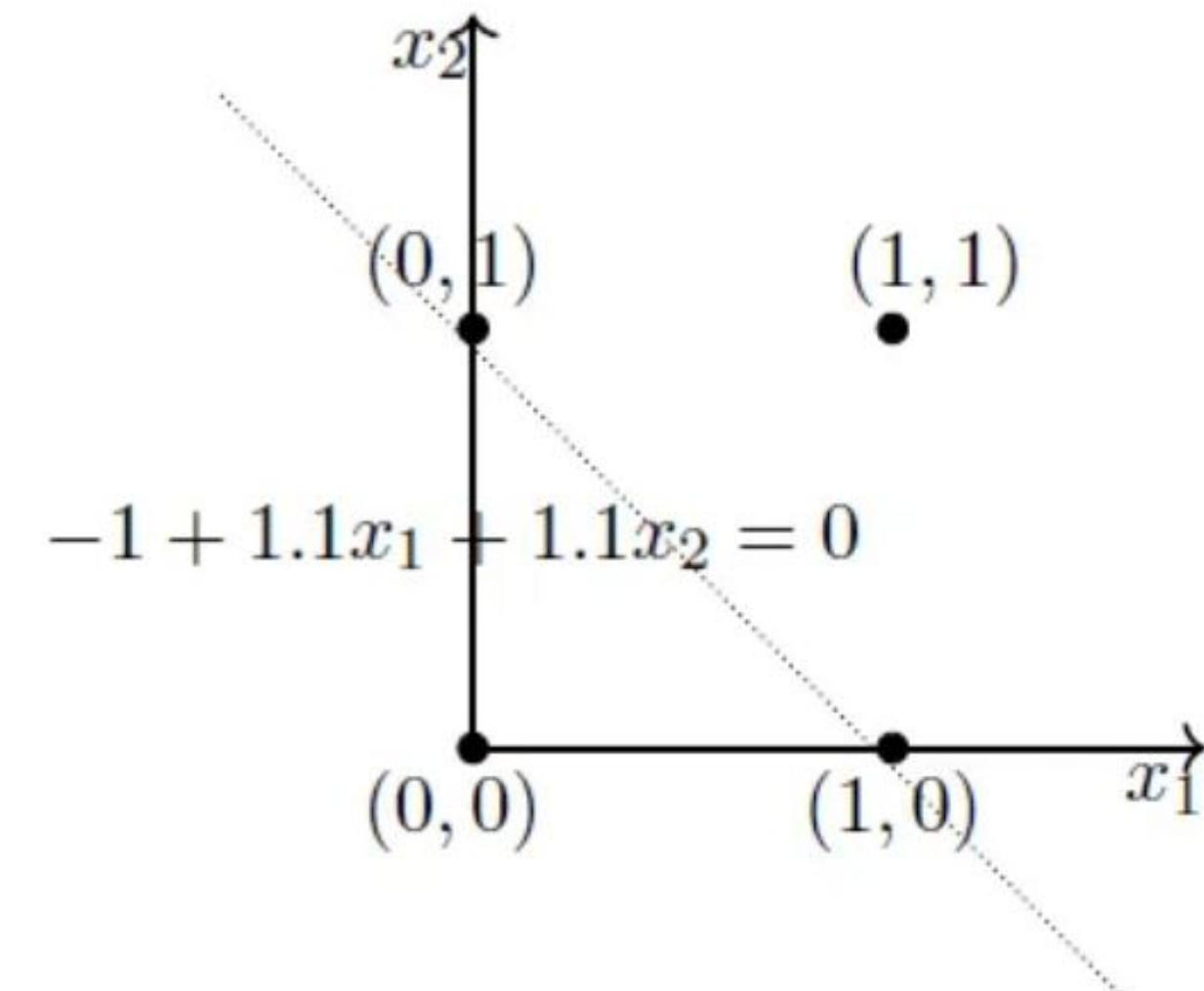
- On solving these, one such possible solution is

$$w_0 = -1, w_1 = 1.1, w_2 = 1.1$$

- Various other solutions are possible as well.



Other possible solutions



# Boolean Functions Using Perceptron - AND Function

| x1 | x2 | AND |                                     |
|----|----|-----|-------------------------------------|
| 0  | 0  | 0   | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |
| 0  | 1  | 0   | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |
| 1  | 0  | 0   | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |
| 1  | 1  | 1   | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |

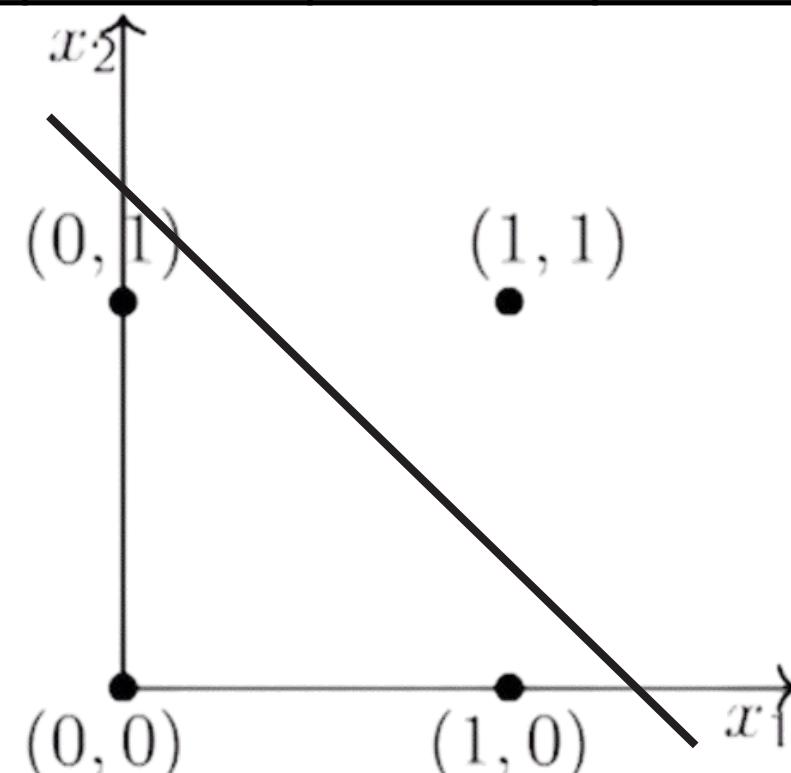
On expanding these, we get a system of linear inequalities

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \Rightarrow w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 < 0 \Rightarrow w_2 < -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 < 0 \Rightarrow w_1 < -w_0$$

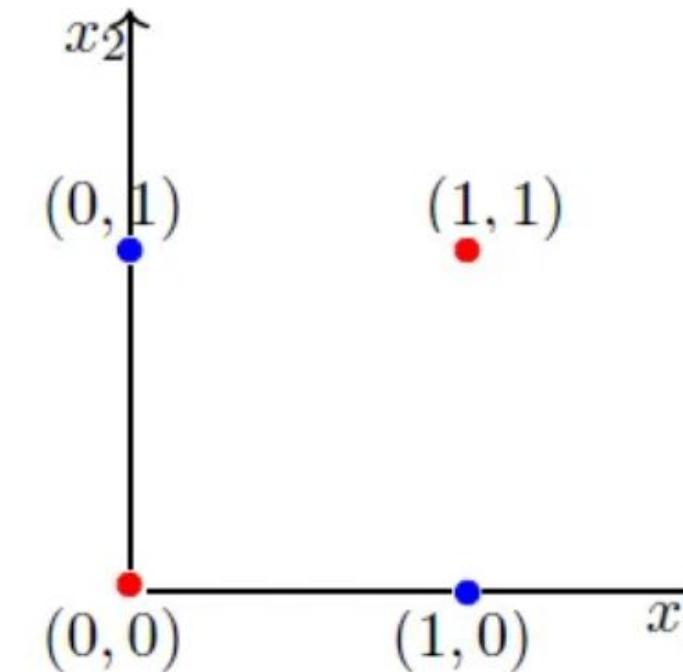
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \Rightarrow w_1 + w_2 > -w_0$$



One possible solution is  $w_0 = -2, w_1 = 1.5, w_2 = 1.5$

# Boolean Functions Using Perceptron - XOR Function

| x1 | x2 | XOR |                                     |
|----|----|-----|-------------------------------------|
| 0  | 0  | 0   | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |
| 0  | 1  | 1   | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |
| 1  | 0  | 1   | $w_0 + \sum_{i=1}^2 w_i x_i \geq 0$ |
| 1  | 1  | 0   | $w_0 + \sum_{i=1}^2 w_i x_i < 0$    |



It is impossible to draw a line that separates the blue points from the red ones.

On expanding these, we get a system of linear inequalities

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \Rightarrow w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \Rightarrow w_2 \geq -w_0$$

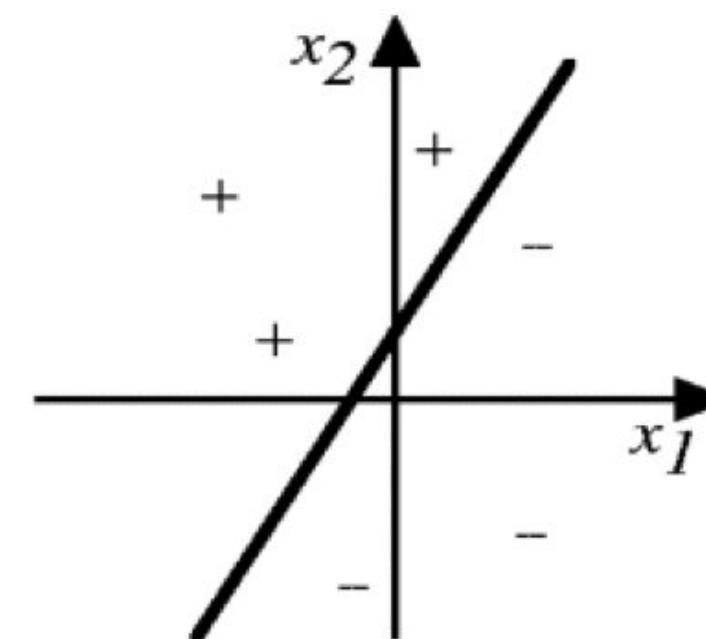
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \Rightarrow w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \Rightarrow w_1 + w_2 < -w_0$$

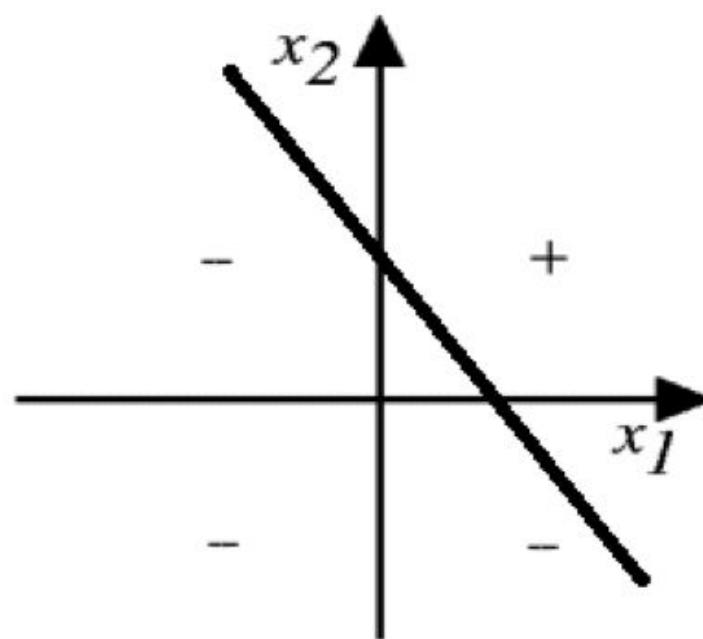
- The fourth condition contradicts the second and third one!
- Hence we cannot have a solution to this set of inequalities.
- A single perceptron cannot learn to separate the data that are non-linear in nature.

# Perceptron fails here

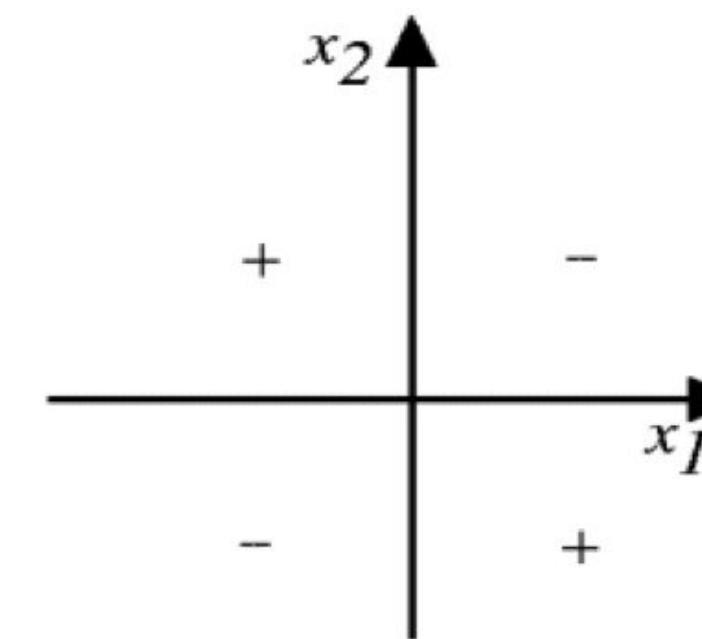
If the data is not linearly separable as in the XOR function, then a single perceptron is not enough to represent the functionality.



Linearly separable



Linearly separable

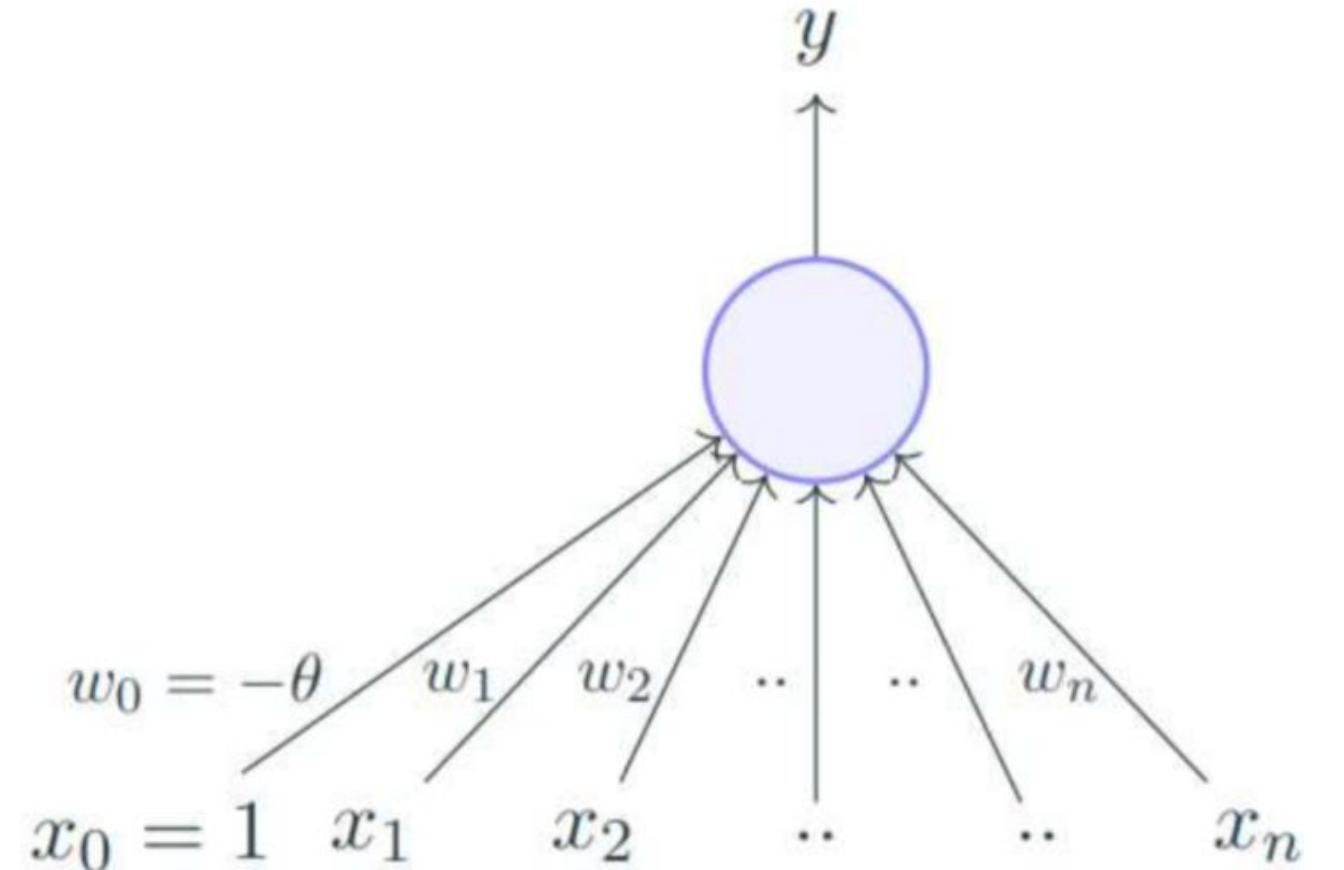


Not linearly separable

A single-layered perceptron can't solve problems that are not linearly separable.

# Perceptron Learning Algorithm

- Let's consider the problem of deciding whether to dine in at a restaurant.
- Suppose we have a list of  $m$  restaurants and a label (class) associated with each, indicating whether the user likes the cuisine, ambience, service, etc., i.e., a binary decision.
- Let's represent each restaurant with  $n$  features (Boolean/real, e.g., rating).
- We'll assume that the data is linearly separable and want a perceptron to learn how to make this decision.
- Essentially, we want the perceptron to find the equation of the separating plane (or determine the values of  $w_0, w_1, w_2, \dots, w_n$ ).



$x_1$  = likeCuisine  
 $x_2$  = likeAmbience  
 $x_3$  = likeService  
 $x_4$  = Ratings (scaled - 0 to 1)  
:  
 $x_n$  = CostFor2 (scaled to 0 to 1)

# A little math for better understanding

## Dot Product Of Two Vectors

Imagine you have two vectors of size  $n+1$ ,  $\mathbf{w}$  and  $\mathbf{x}$ , the dot product of these vectors ( $\mathbf{w} \cdot \mathbf{x}$ ) could be computed as follows.

$$\mathbf{w} = [w_0, w_1, w_2, \dots, w_n]$$

$$\mathbf{x} = [1, x_1, x_2, \dots, x_n]$$

$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T \mathbf{x} = \sum_{i=0}^n w_i * x_i$$

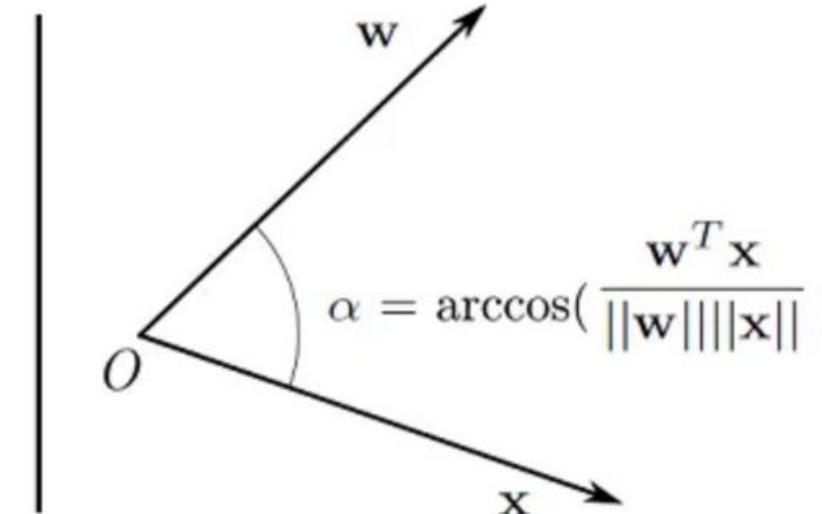
Their dot product quantifies how much one vector is going in the direction of the other.

## Angle Between Two Vectors

$$\mathbf{w}^T \mathbf{x} = \|\mathbf{w}\| \|\mathbf{x}\| \cos\alpha$$

You can get the angle between two vectors you know how to calculate vector magnitudes and their vanilla dot product

$$\cos\alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$$



When the dot product of two vectors is 0, they are

# Perceptron Learning Algorithm

The goal is to find a  $\mathbf{w}$  vector that can perfectly classify positive inputs and negative inputs .

---

**Algorithm:** Perceptron Learning Algorithm

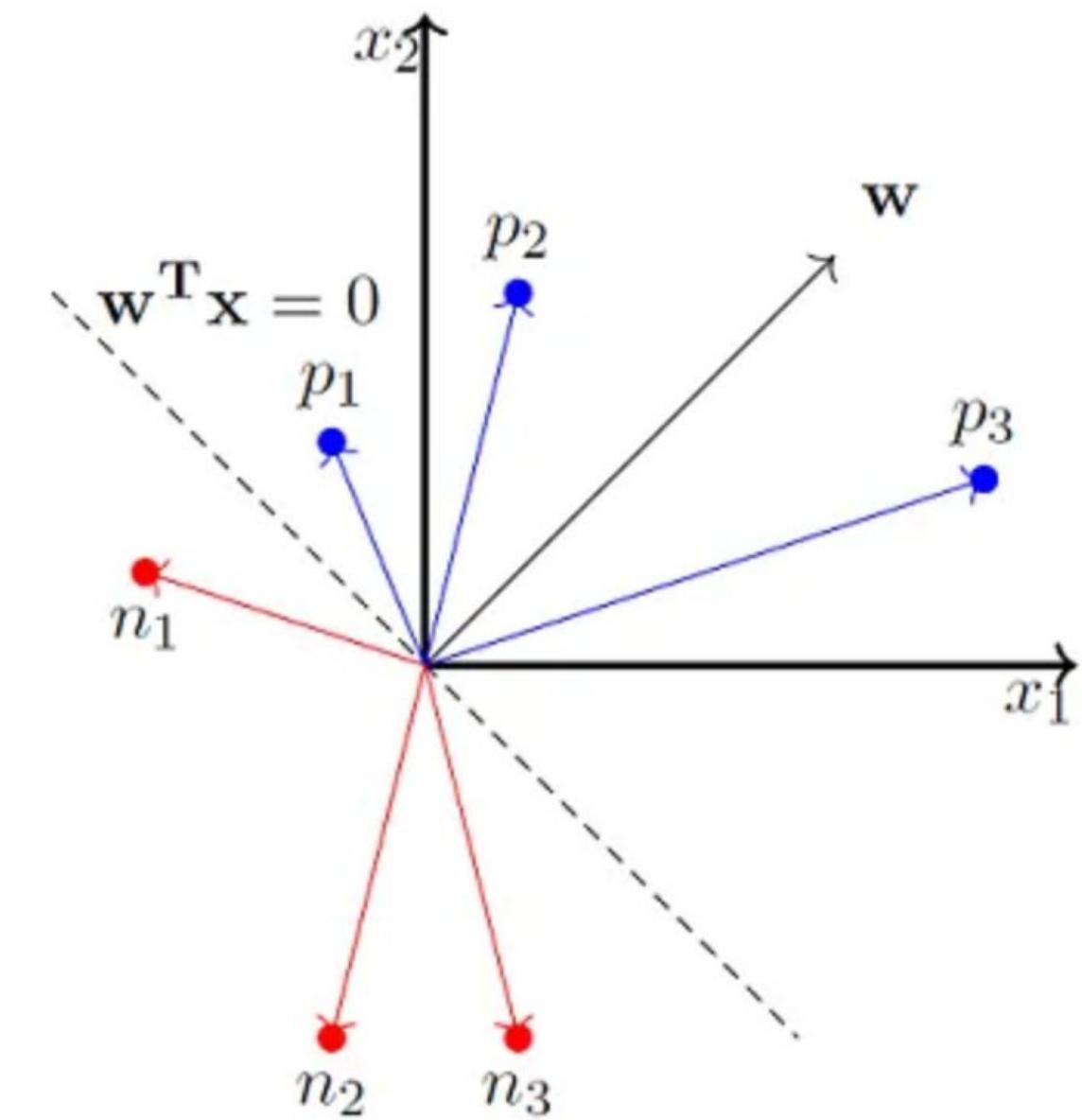
---

```

 $P \leftarrow$  inputs with label 1;
 $N \leftarrow$  inputs with label 0;
Initialize  $\mathbf{w}$  randomly;
while !convergence do
    Pick random  $\mathbf{x} \in P \cup N$  ;
    if  $\mathbf{x} \in P$  and  $\mathbf{w} \cdot \mathbf{x} < 0$  then
        |  $\mathbf{w} = \mathbf{w} + \mathbf{x}$  ;
    end
    if  $\mathbf{x} \in N$  and  $\mathbf{w} \cdot \mathbf{x} \geq 0$  then
        |  $\mathbf{w} = \mathbf{w} - \mathbf{x}$  ;
    end
end
//the algorithm converges when all the
inputs are classified correctly

```

---



# Intuition behind Perceptron Learning Algorithm

---

Consider two vectors  $w$  and  $x$  -

$$w = [w_0, w_1, w_2, \dots, w_d]$$

$$x = [1, x_1, x_2, \dots, x_d]$$

Taking the dot product

$$w \cdot x = w^T x = \sum w_i x_i$$

We can thus rewrite the perceptron rule as -

$$y = 1 \text{ if } w^T x \geq 0$$

$$= 0 \text{ if } w^T x < 0$$

Every point ( $x$ ) on this line satisfies  $w^T x = 0$

- The angle ( $\alpha$ ) between  $w$  and any point ( $x$ ) which lies on this line is 90 since

$$\cos \alpha = \frac{w^T x}{\|w\| \|x\|}$$

- Since the vector  $w$  is perpendicular to every point on the line it is actually perpendicular to the line itself.

We are interested in finding the line  $w^T x = 0$

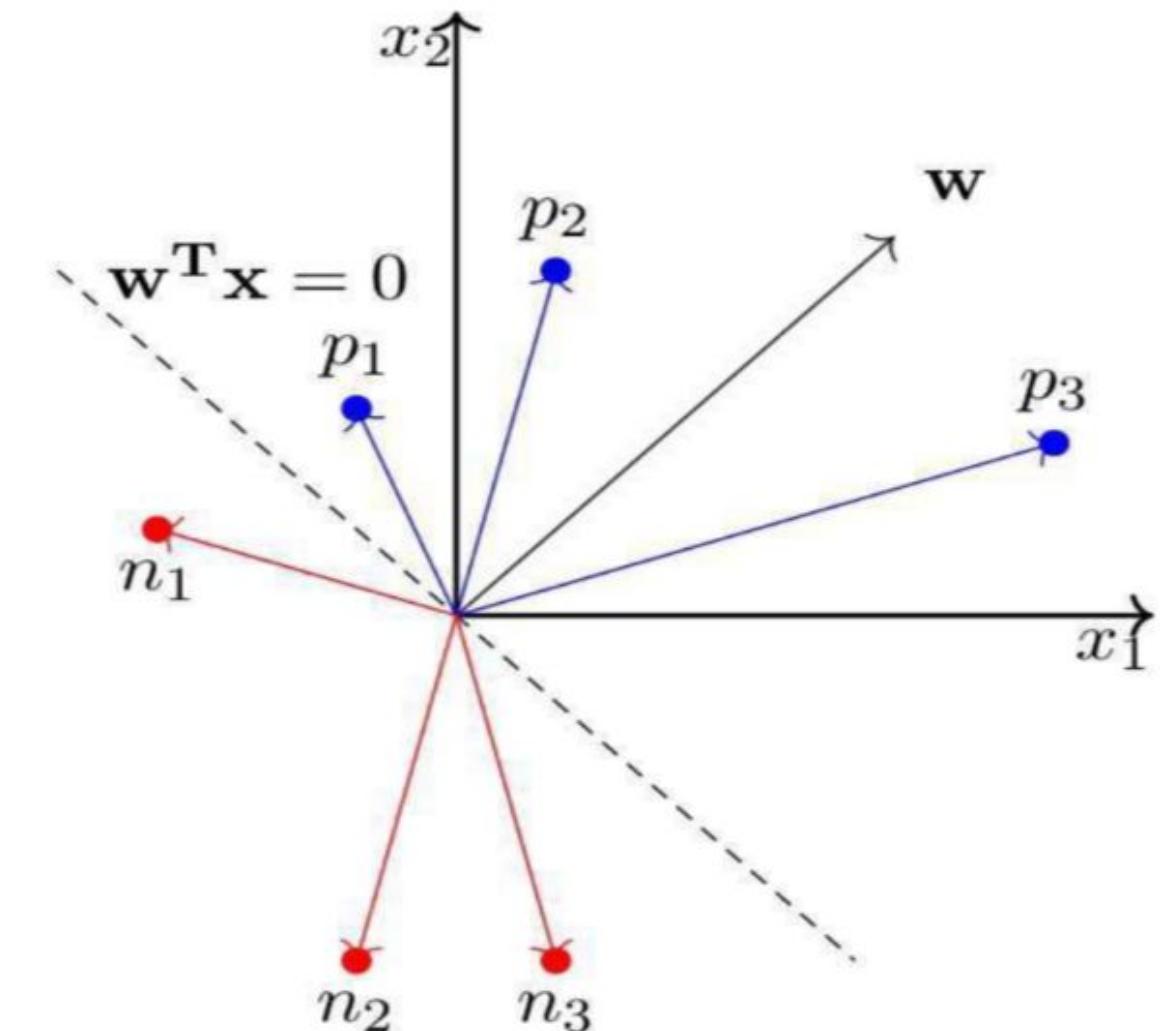
# Intuition behind Perceptron Learning Algorithm

- Consider some points which lie in the +ve half space of this line i.e.  $w^T x \geq 0$ . The angle between any such vector and w is  $< \text{abs}|\pm 90|$ .
- Consider some points which lie in the -ve half space of this line i.e.  $w^T x < 0$ . The angle between any such vector and w is  $> 90$ .

$$\cos\alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$$
 |  $\cos\alpha \propto \mathbf{w}^T \mathbf{x}$

So if  $\mathbf{w}^T \mathbf{x} > 0 \Rightarrow \cos\alpha > 0 \Rightarrow \alpha < 90$

Similarly, if  $\mathbf{w}^T \mathbf{x} < 0 \Rightarrow \cos\alpha < 0 \Rightarrow \alpha > 90$



- Keeping this in mind, let's revisit the algorithm.

# Intuition behind Perceptron Learning Algorithm

```

 $P \leftarrow$  inputs with label 1;
 $N \leftarrow$  inputs with label 0;
Initialize  $\mathbf{w}$  randomly;
while !convergence do
    Pick random  $\mathbf{x} \in P \cup N$  ;
    if  $\mathbf{x} \in P$  and  $\mathbf{w} \cdot \mathbf{x} < 0$  then
         $\mathbf{w} = \mathbf{w} + \mathbf{x}$  ;
    end
    if  $\mathbf{x} \in N$  and  $\mathbf{w} \cdot \mathbf{x} \geq 0$  then
         $\mathbf{w} = \mathbf{w} - \mathbf{x}$  ;
    end
end
//the algorithm converges when all the
inputs are classified correctly

```

$$\cos\alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$$

For  $\mathbf{x} \in P$  if  $\mathbf{w} \cdot \mathbf{x} < 0$  then it means that the angle ( $\alpha$ ) between this  $\mathbf{x}$  and the current  $\mathbf{w}$  is greater than  $90^\circ$  (but we want  $\alpha$  to be less than  $90^\circ$ )

What happens to the new angle ( $\alpha_{new}$ ) when  $\mathbf{w}_{new} = \mathbf{w} + \mathbf{x}$

$$\begin{aligned}
\cos(\alpha_{new}) &\propto \mathbf{w}_{new}^T \mathbf{x} \\
&\propto (\mathbf{w} + \mathbf{x})^T \mathbf{x} \\
&\propto \mathbf{w}^T \mathbf{x} + \mathbf{x}^T \mathbf{x} \\
&\propto \cos\alpha + \mathbf{x}^T \mathbf{x}
\end{aligned}$$

$$\cos(\alpha_{new}) > \cos\alpha$$

Thus  $\alpha_{new}$  will be less than  $\alpha$  and this is exactly what we want

# Intuition behind Perceptron Learning Algorithm

- When we are adding  $x$  to  $w$ , which we do when  $x$  belongs to the +ve half space and  $w \cdot x < 0$ , we are essentially increasing the  $\cos(\alpha)$  value, which means, we are decreasing the alpha value(the angle between  $w$  and  $x$ ) which is what we desire.
- A similar intuition works for the case when  $x$  belongs to the -ve half space and  $w \cdot x \geq 0$ . For error in the prediction of negative instances, the angle is less than 90 degrees, but it must be greater than 90 degrees

$(\alpha_{new})$  when  $w_{new} = w + x$

$$\begin{aligned} \cos(\alpha_{new}) &\propto w_{new}^T x \\ &\propto (w + x)^T x \\ &\propto w^T x + x^T x \\ &\propto \cos\alpha + x^T x \end{aligned}$$

$$\cos(\alpha_{new}) > \cos\alpha$$

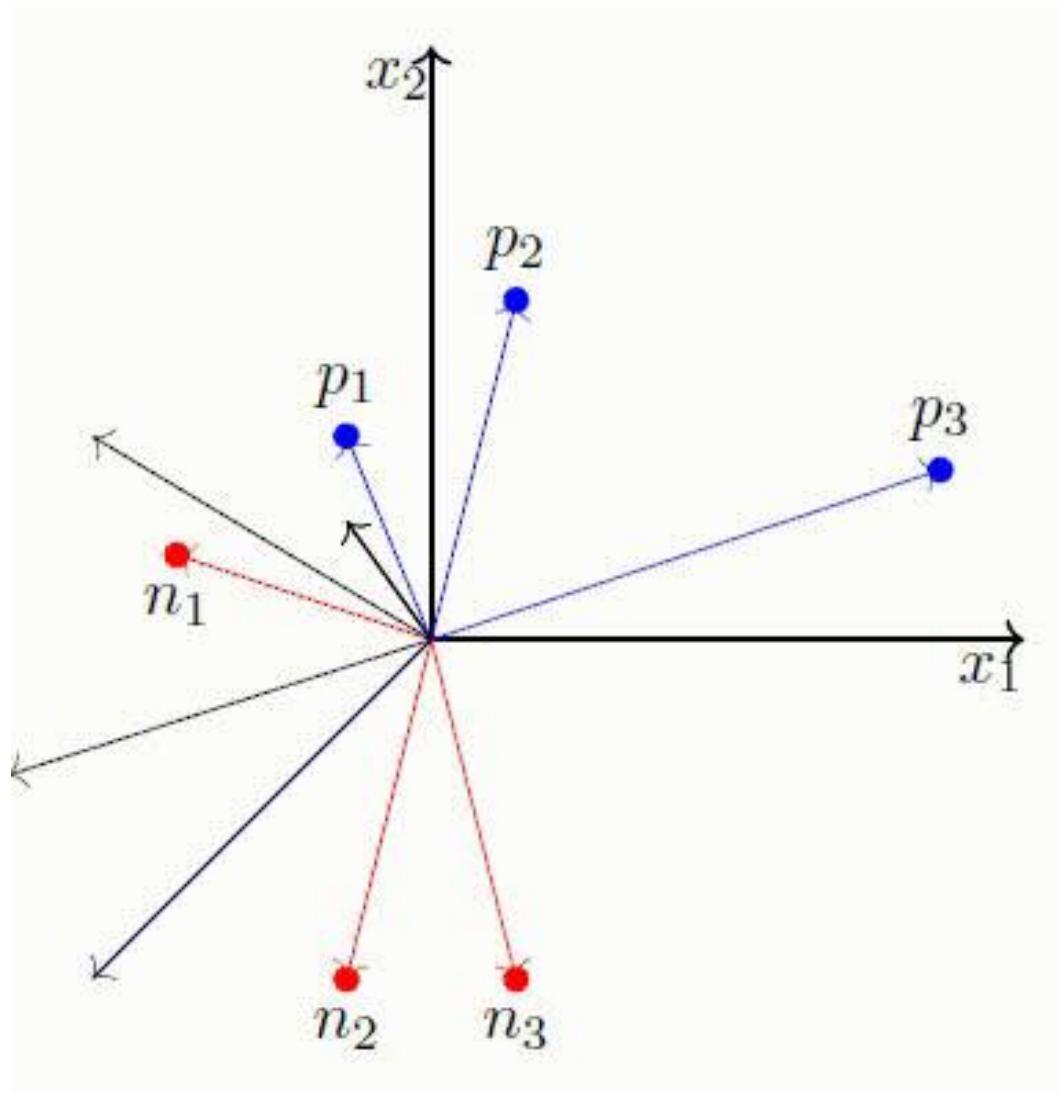
$(\alpha_{new})$  when  $w_{new} = w - x$

$$\begin{aligned} \cos(\alpha_{new}) &\propto w_{new}^T x \\ &\propto (w - x)^T x \\ &\propto w^T x - x^T x \\ &\propto \cos\alpha - x^T x \end{aligned}$$

$$\cos(\alpha_{new}) < \cos\alpha$$

# Intuition behind Perceptron Learning Algorithm

A simulation of how we might end up learning  $w$  that makes an angle less than 90 for positive examples and more than 90 for negative examples.



## Perceptron learning for linear surfaces

---

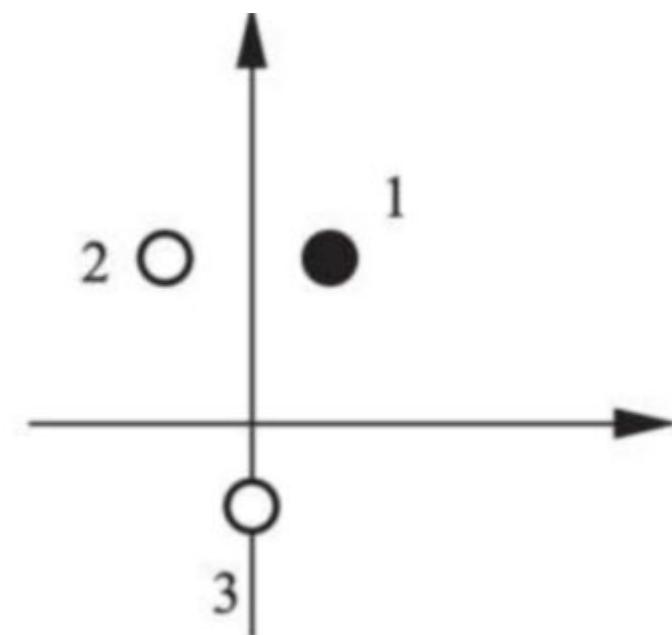
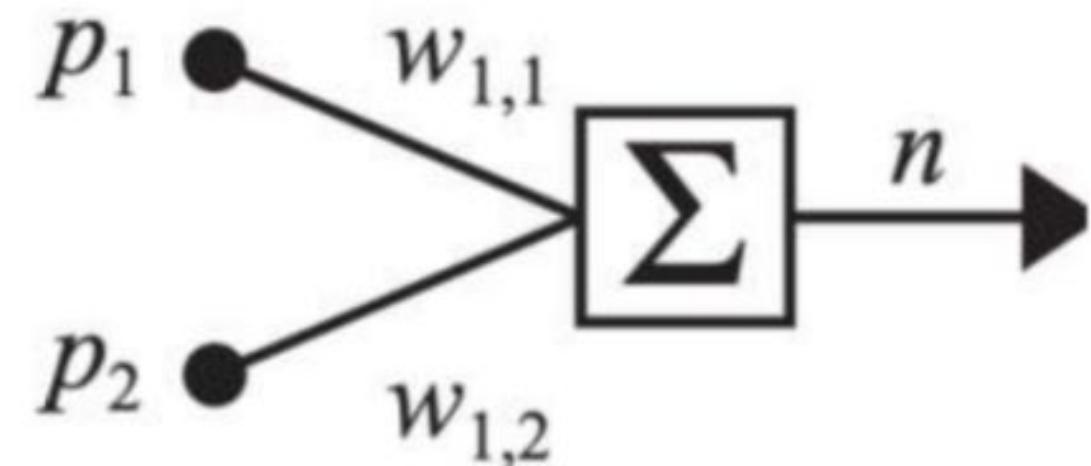
- A perceptron is a threshold linear unit (discrete-valued).
- What is a linear unit? A linear combination of weighted inputs (real-valued).
- A perceptron offers linear decision surfaces. This means a single perceptron can easily represent simple Boolean functions like AND, OR, NAND and NOR.

## Perceptron learning - Sample Test problem

Solve the following classification problem for a single neuron perceptron using the Perceptron learning algorithm. Apply each input vector in order, for as many repetitions as it takes to ensure that the problem is solved. Draw a graph of the problem for each weight vector to depict the decision boundary.

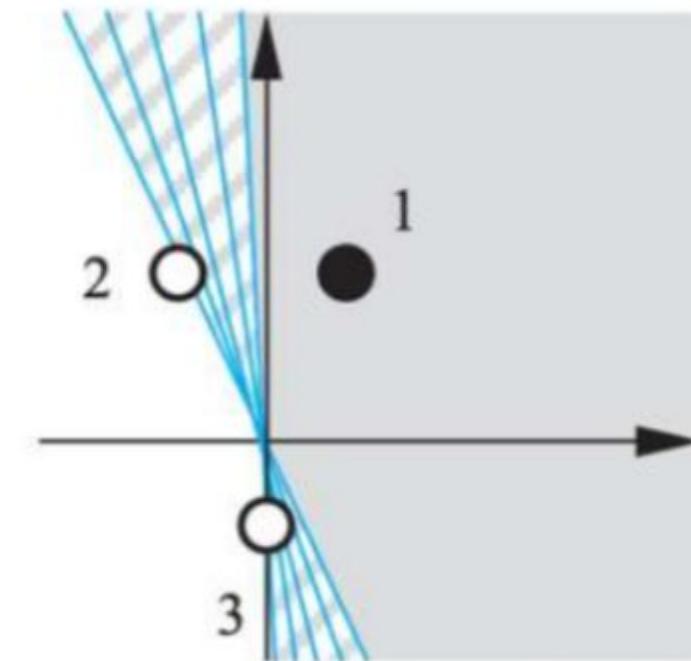
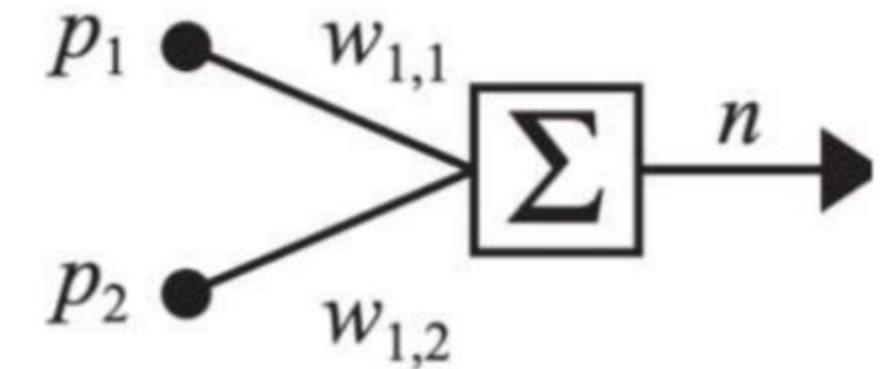
Assume initial weights  $w_{1,1} = 1$ ,  $w_{1,2}=-0.8$  and bias  $b = 0$ .

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$



## Perceptron learning - Sample Test problem

- The network for this problem should have two inputs and one output. To simplify our development of the learning rule, we will begin with a network without a bias.
- By removing the bias we are left with a network whose decision boundary must pass through the origin. We need to be sure that this network is still able to solve the test problem. There must be an allowable decision boundary that can separate the vectors and from the vector . The figure to the right illustrates that there are indeed an infinite number of such boundaries.



## Perceptron learning - Sample Test problem

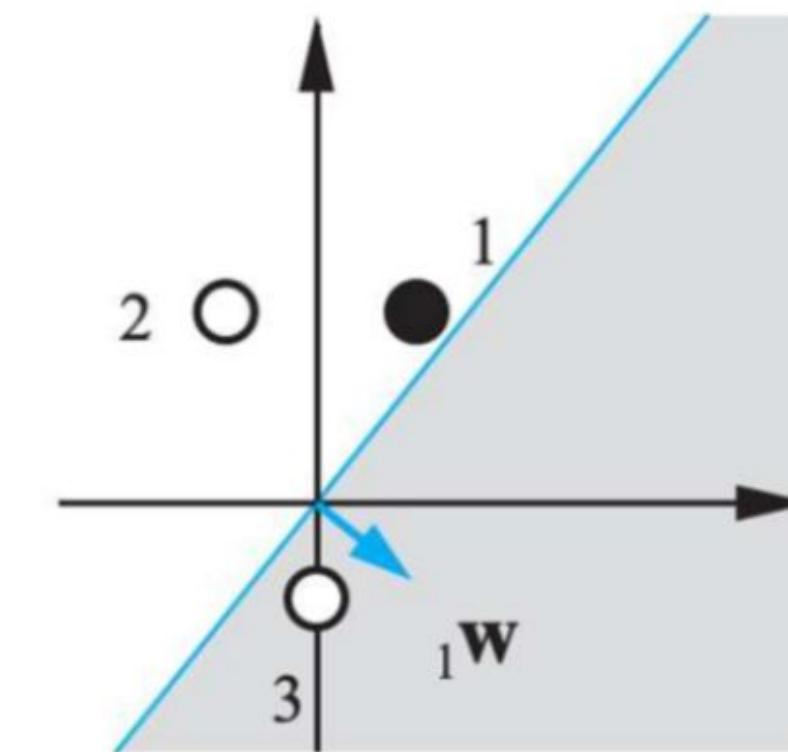
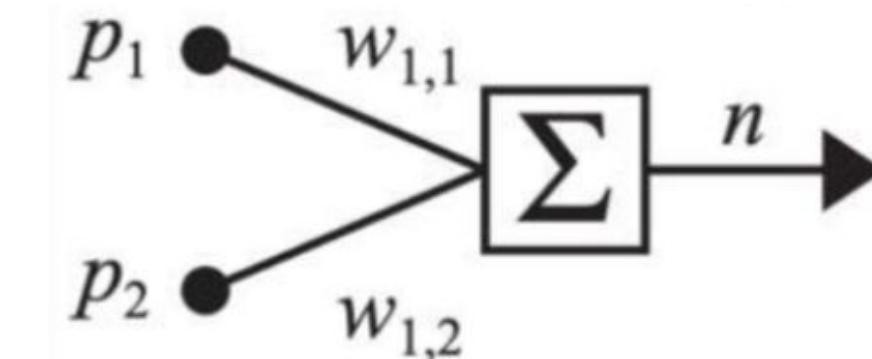
Given data :

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$

$$z = \mathbf{w}^T \mathbf{x} + b = \begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1.0 - 1.6 = -0.6 \not\geq 0 \Rightarrow y = 0 \text{ (error)}$$

Modify the weight vector as  $\mathbf{w} = \mathbf{w} + \mathbf{x}$

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}.$$



## Perceptron learning - Sample Test problem

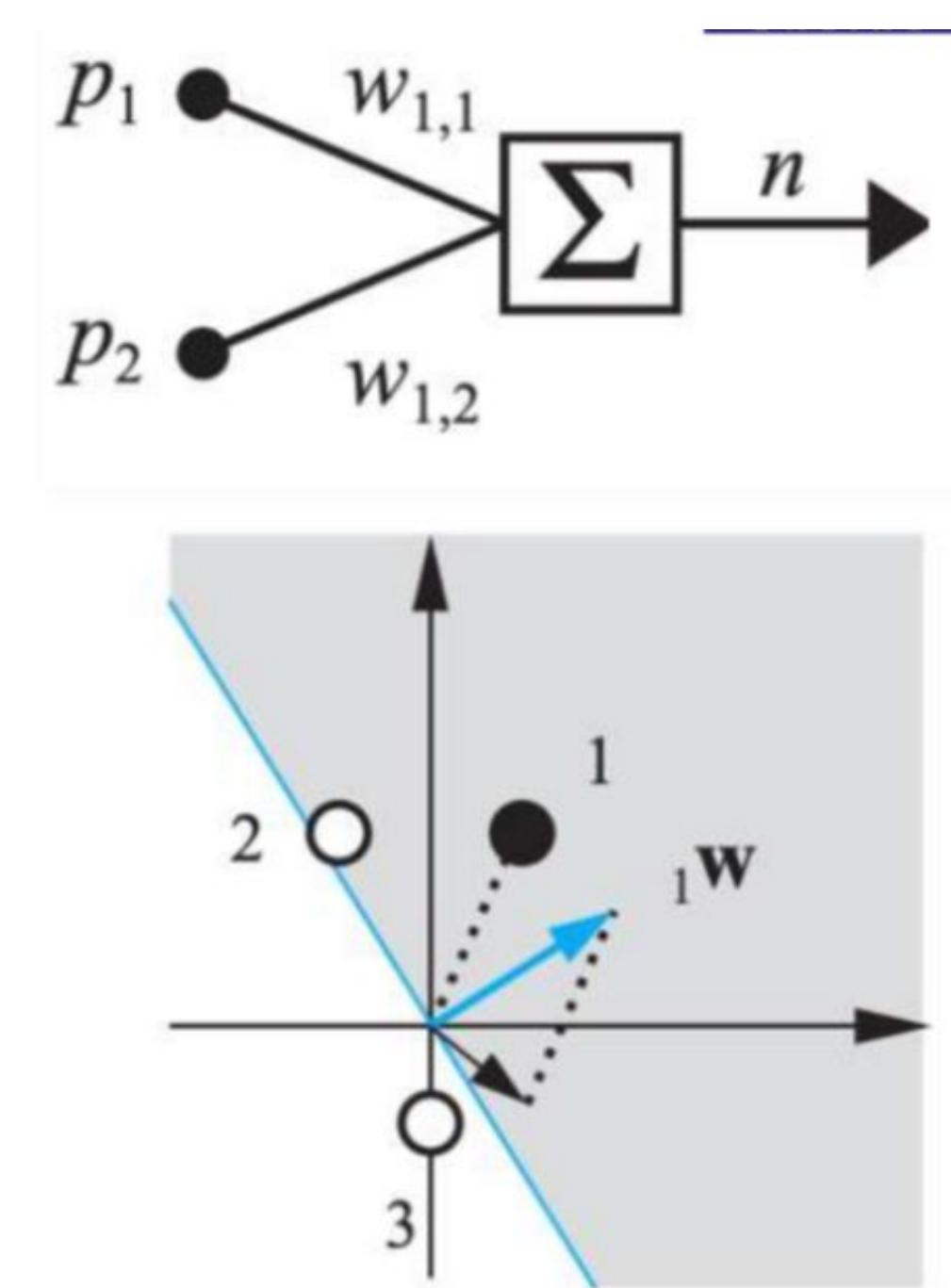
Given data :

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$

$$z = \mathbf{w}^T \mathbf{x} + b = [2.0 \ 1.2] \begin{bmatrix} -1 \\ 2 \end{bmatrix} = 0.4 < 0 \Rightarrow y = 1 \text{ (error)}$$

Modify the weight vector as  $\mathbf{w} = \mathbf{w} - \mathbf{x}$

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$



## Perceptron learning - Sample Test problem

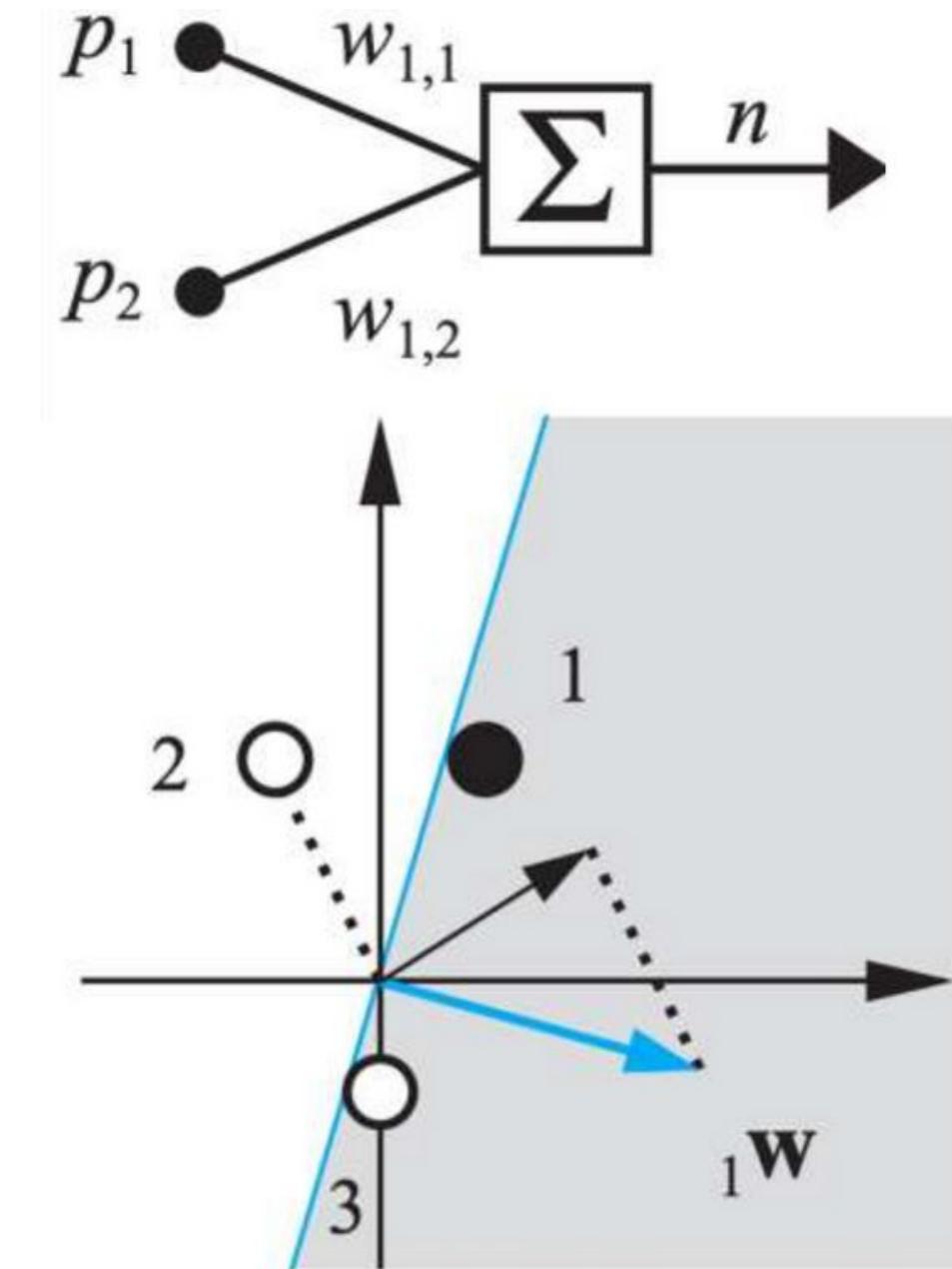
Given data :

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$

$$z = \mathbf{w}^T \mathbf{x} + b = \begin{bmatrix} 3.0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} = 0.4 < 0 \Rightarrow y = 1 \text{ (error)}$$

Modify the weight vector as  $\mathbf{w} = \mathbf{w} - \mathbf{x}$

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$



# Perceptron learning - Sample Test problem

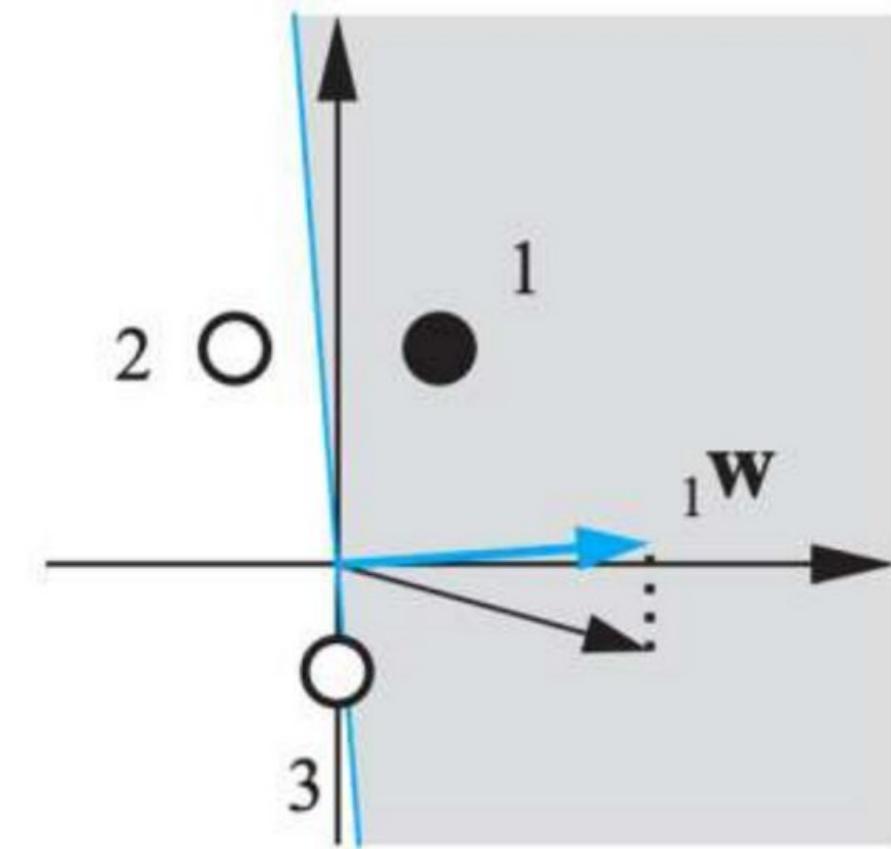
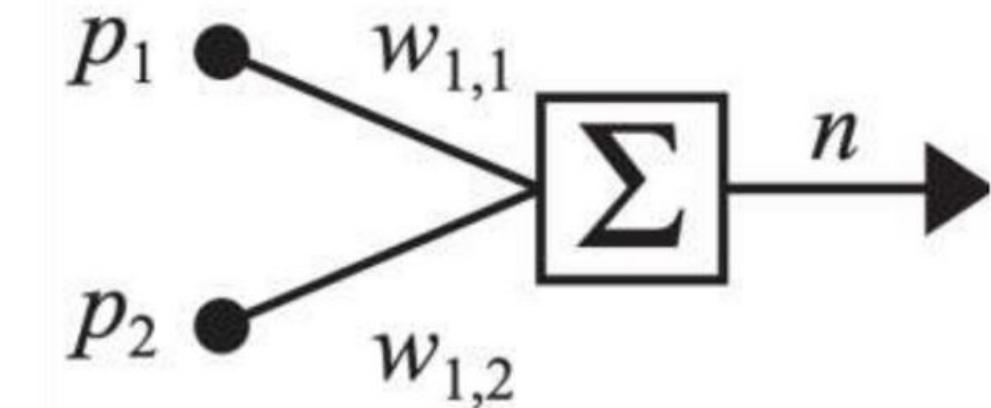
Given data :

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$

$$z = \mathbf{w}^T \mathbf{x} + 0 = [3 \ 0.2] \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 3.4 \geq 0 \Rightarrow y = 1$$

$$z = \mathbf{w}^T \mathbf{x} + 0 = [3 \ 0.2] \begin{pmatrix} -1 \\ 2 \end{pmatrix} = -2.6 < 0 \Rightarrow y = 0$$

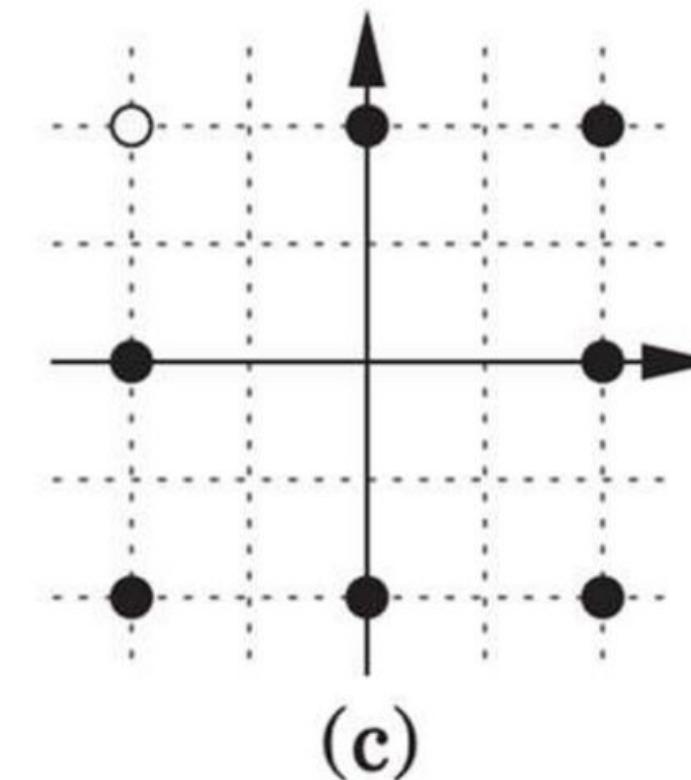
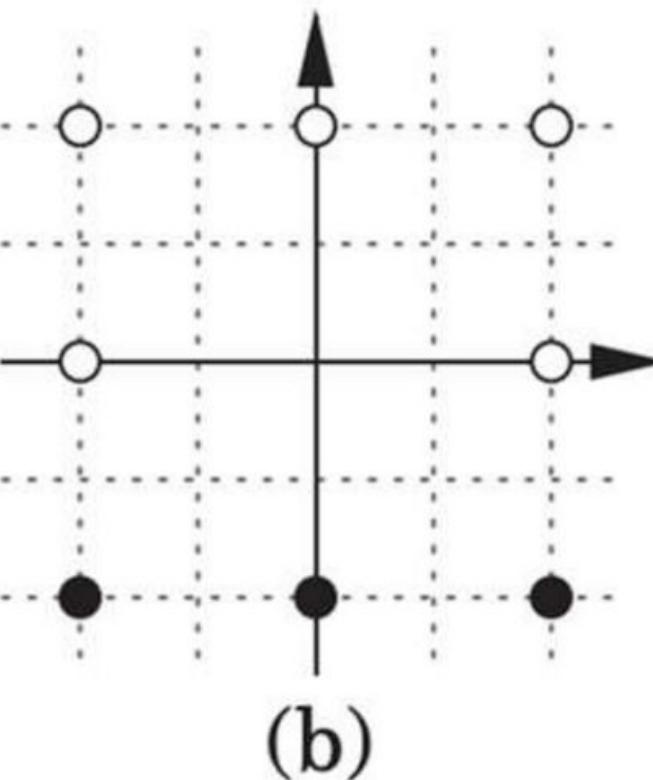
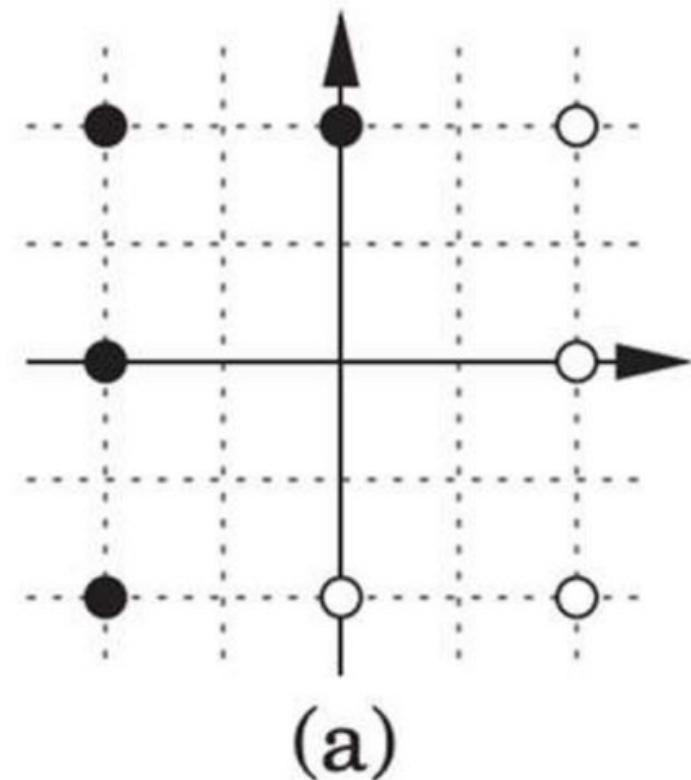
$$z = \mathbf{w}^T \mathbf{x} + 0 = [3 \ 0.2] \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -0.2 < 0 \Rightarrow y = 0$$



Algorithm converges! No more  
errors

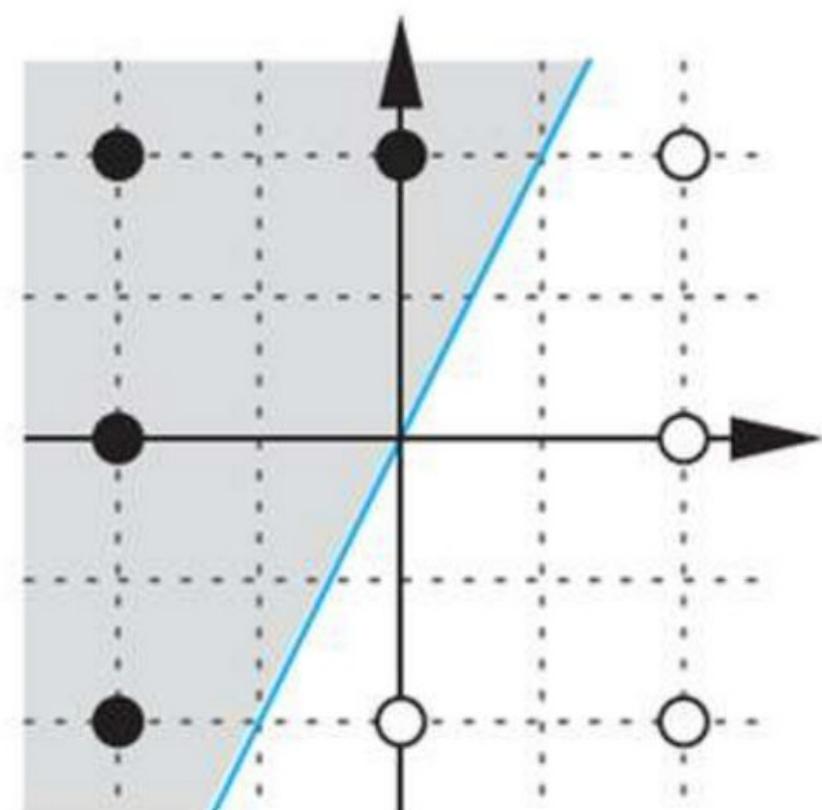
## Perceptron learning - Question

Solve the three simple classification problems shown in Figure by drawing a decision boundary. Find weight and bias values that result in single-neuron perceptrons with the chosen decision boundaries.

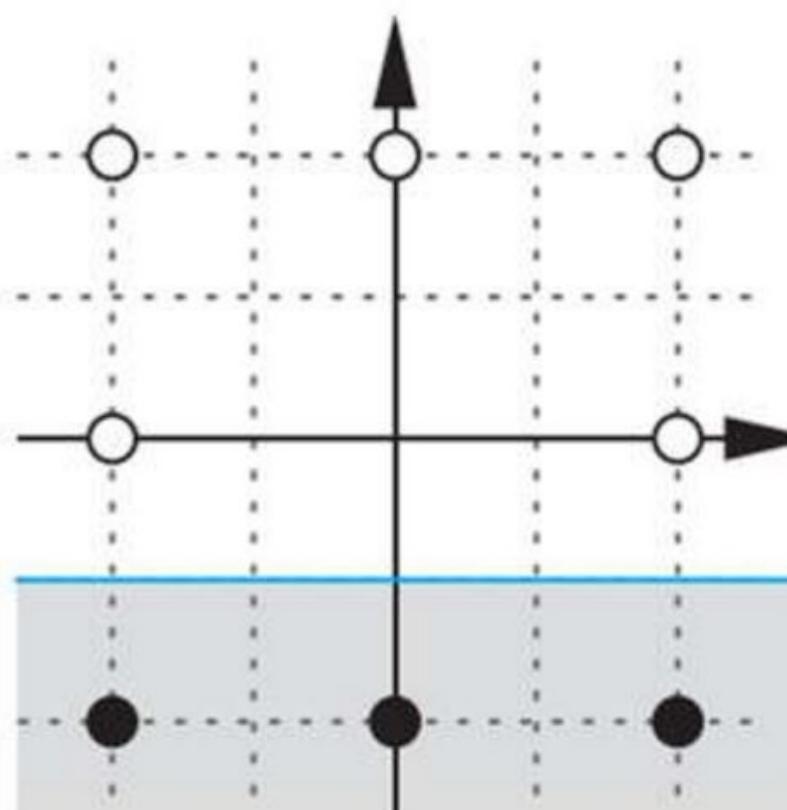


## Perceptron learning - Solution

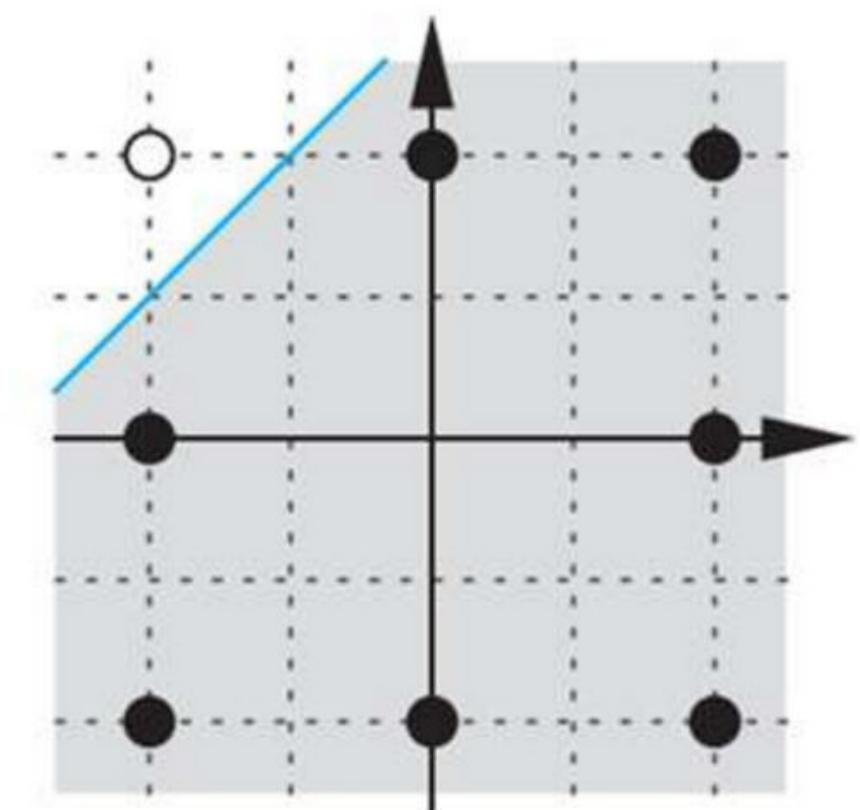
First we draw a line between each set of dark and light data points.



(a)



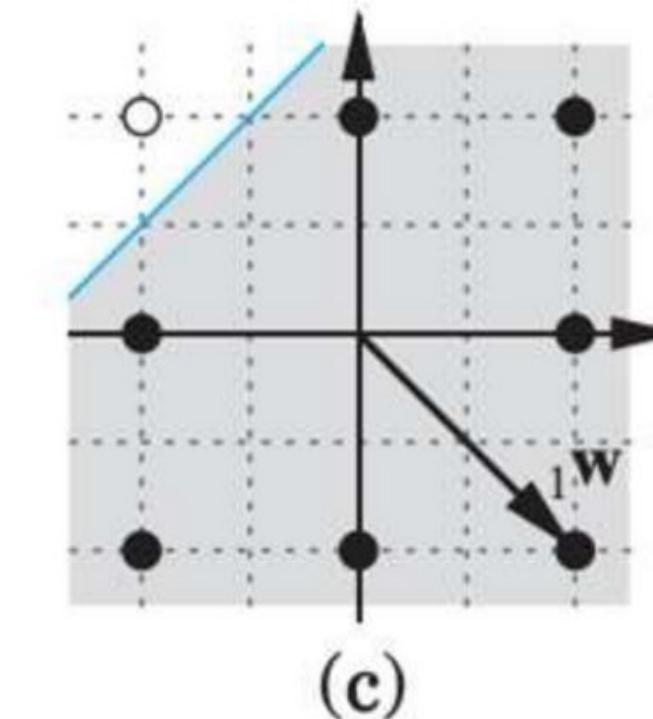
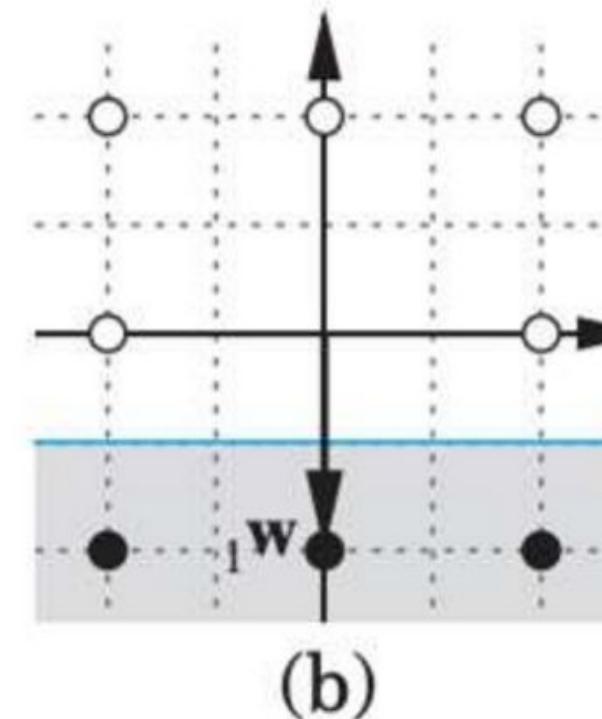
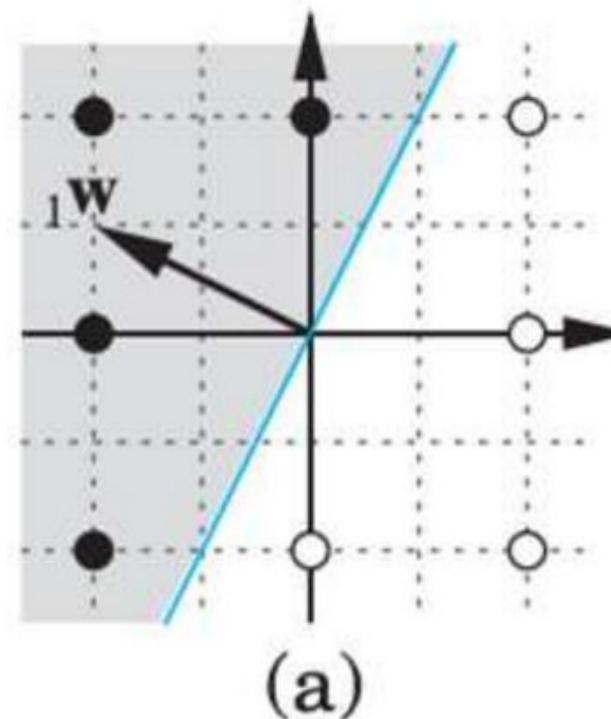
(b)



(c)

## Perceptron learning - Solution

The next step is to find the weights and biases. The weight vectors must be orthogonal to the decision boundaries, and pointing in the direction of points to be classified as 1 (the dark points). The weight vectors can have any length we like.



Here is one set of choices for the weight vectors:

$$(a) \mathbf{w}_1^T = [-2 \ 1], \ (b) \mathbf{w}_1^T = [0 \ -2], \ (c) \mathbf{w}_1^T = [2 \ -2].$$

# Perceptron learning - Solution

Now we find the bias values for each perceptron by picking a point on the decision boundary and satisfying Eq. (4.15).

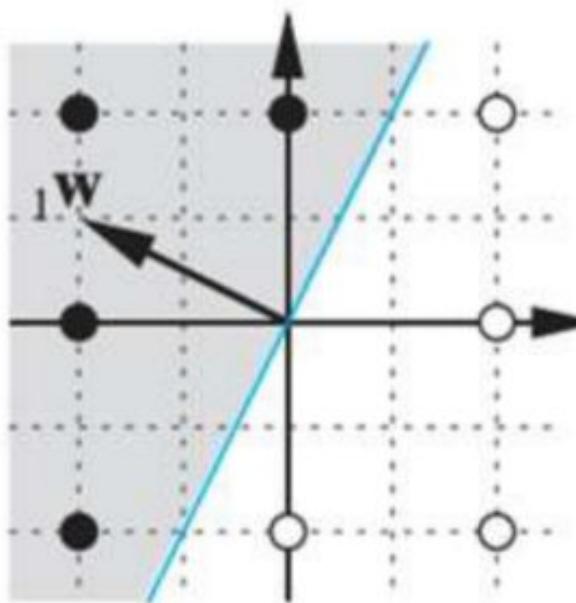
$$_1\mathbf{w}^T \mathbf{p} + b = 0$$

$$b = -_1\mathbf{w}^T \mathbf{p}$$

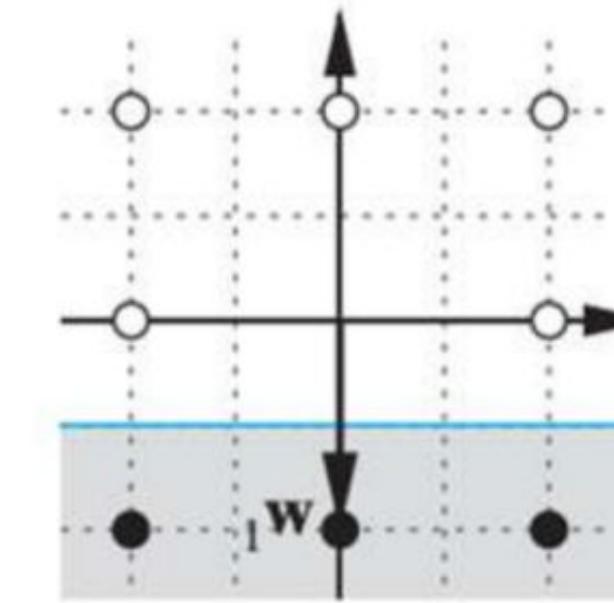
$$(a) \ _1\mathbf{w}^T = [-2 \ 1], \ (b) \ _1\mathbf{w}^T = [0 \ -2], \ (c) \ _1\mathbf{w}^T = [2 \ -2]$$

This gives us the following three biases:

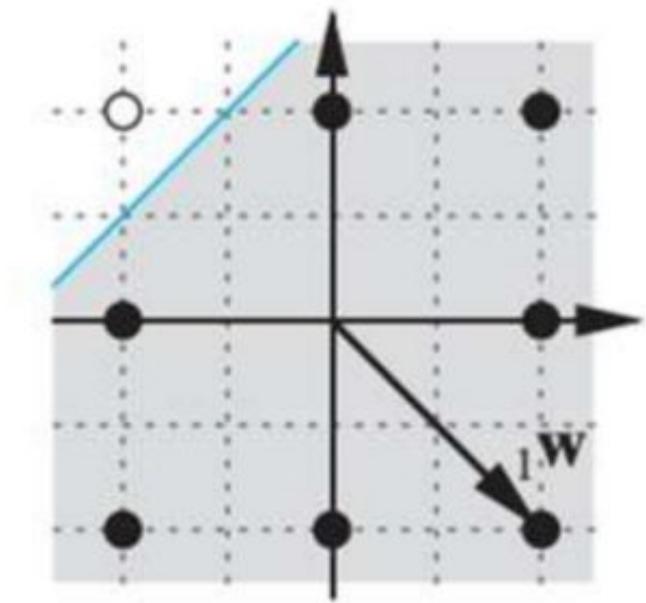
$$(a) b = -[-2 \ 1] \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0, (b) b = -[0 \ -2] \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -2, (c) b = -[2 \ -2] \begin{bmatrix} -2 \\ 1 \end{bmatrix} = 6$$



(a)



(b)



(c)

## Perceptron learning - Practice

Convert the classification problem defined below into an equivalent problem definition consisting of inequalities constraining weight and bias values.

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_2 = 1 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, t_3 = 0 \right\} \left\{ \mathbf{p}_4 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, t_4 = 0 \right\}$$

### Solution

1.  $W_0 + 2W_2 \geq 0$
2.  $W_0 + W_1 \geq 0$
3.  $W_0 - 2W_2 < 0$
4.  $W_0 + 2W_1 < 0$

## Perceptron learning - Practice

Solve the following classification problem with the perceptron rule. Apply each input vector in order, for as many repetitions as it takes to ensure that the problem is solved. Draw a graph of the problem only after you have found a solution.

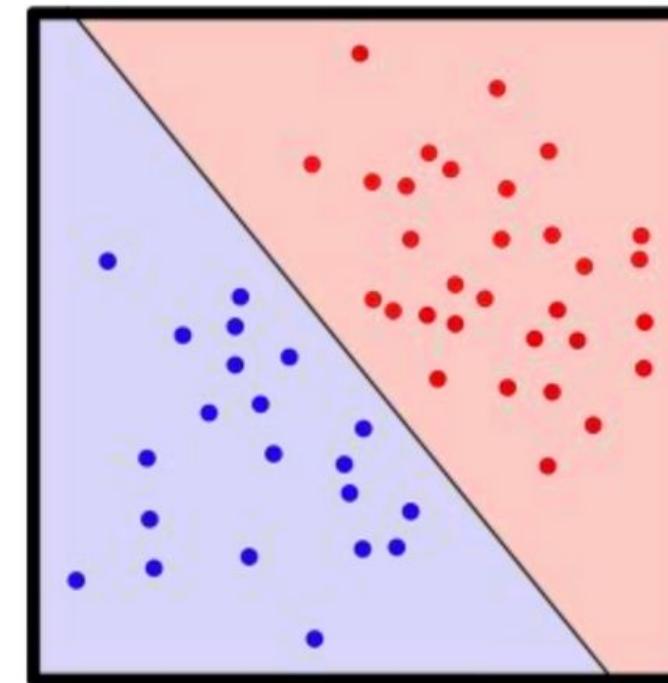
$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0 \right\} \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1 \right\} \left\{ \mathbf{p}_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0 \right\} \left\{ \mathbf{p}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}$$

**Use the initial weights and bias:**

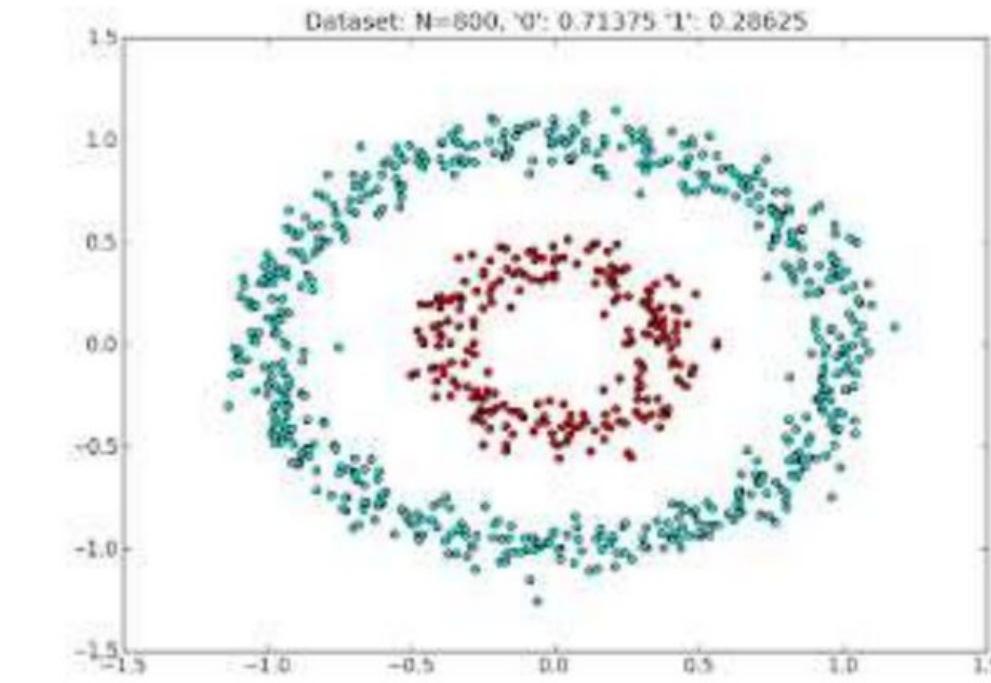
$$\mathbf{w}(0) = [0 \ 0] \quad b(0) = 0 .$$

# When does a single perceptron fail?

- If data is not linearly separable as we saw in the case of the XOR function, then a single perceptron is not enough to represent the functionality.
- In fact, most real world data is linearly inseparable. While a single perceptron cannot deal with such data, a network of perceptrons can!



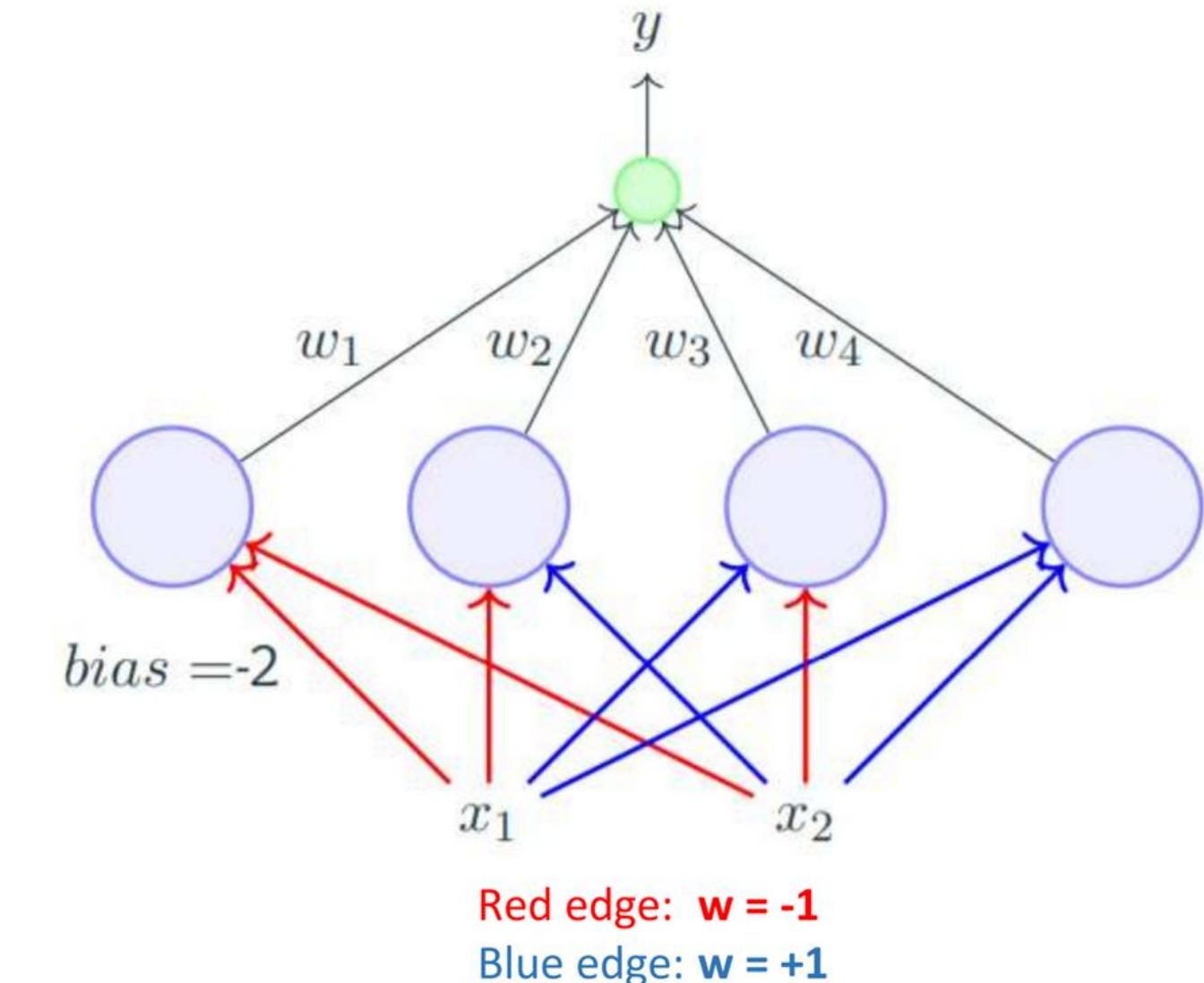
Linearly  
separable



Linearly  
inseparable

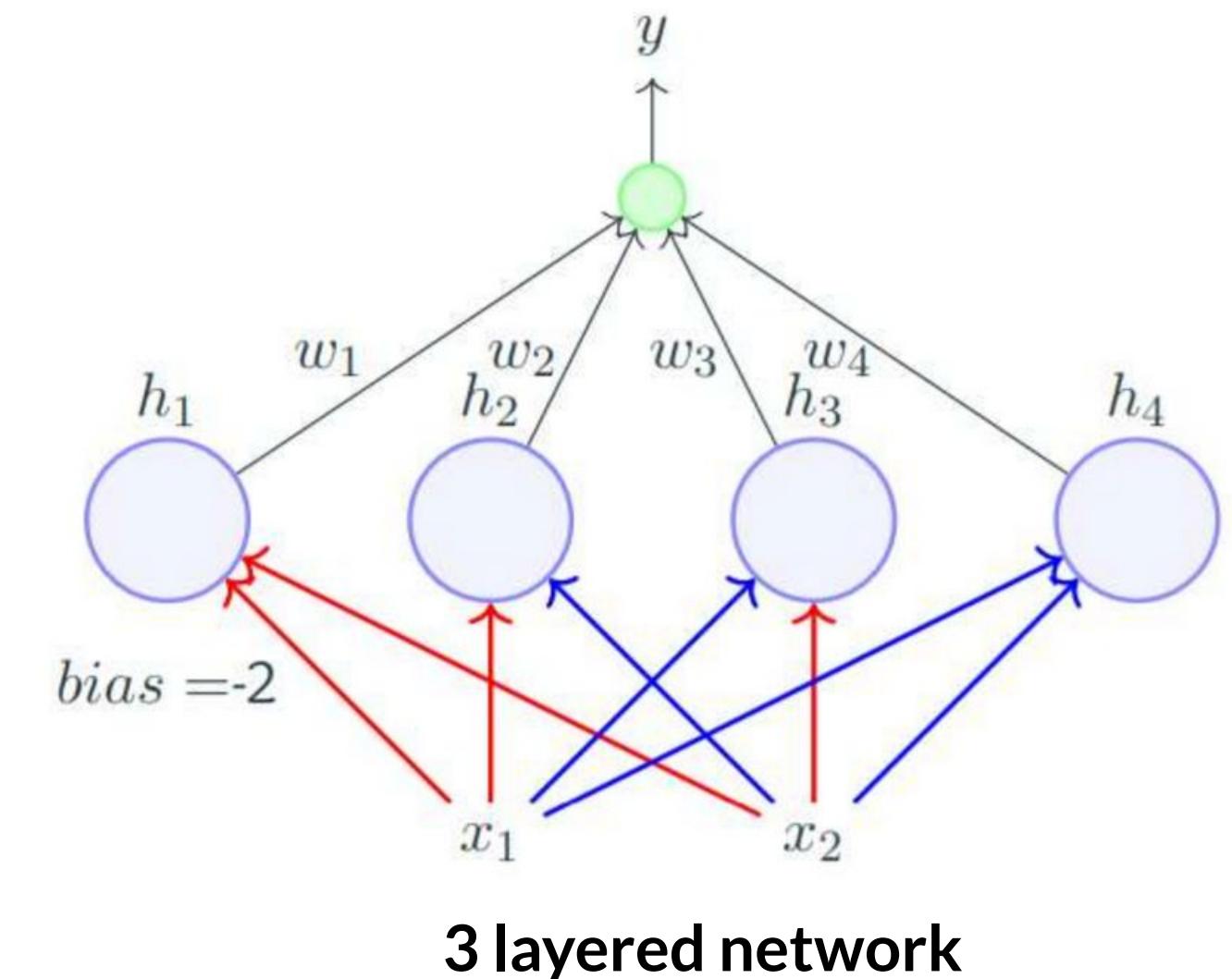
# Implementing any Boolean function using a network of perceptrons

- Assume True = +1 and False = -1.
- We consider 2 inputs and 4 perceptrons.
- Each input is connected to all the 4 perceptrons with specific weights.
- The bias ( $w_0$ ) of each perceptron is -2 (each perceptron will fire if the weighted sum of its input is  $\geq 2$ ).
- Each of these perceptrons is connected to an output perceptron by weights (need to be learned).
- The output of this perceptron ( $y$ ) is the output of this network.



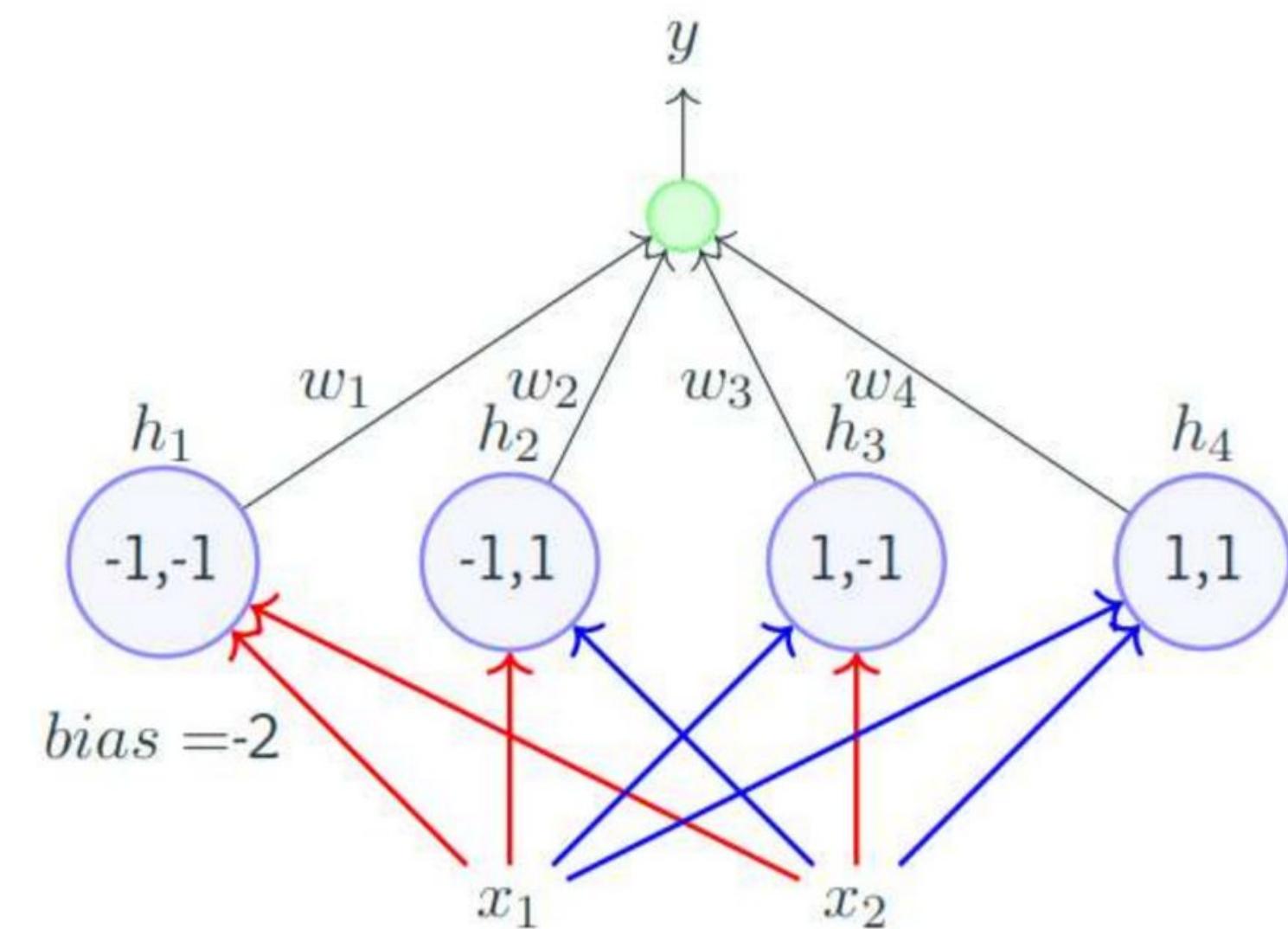
# Implementing any Boolean function using a network of perceptrons

- The input layer contains the inputs ( $x_1, x_2$ ).
- The middle layer containing the 4 perceptrons is called the hidden layer.
- The final layer containing one output neuron is called the output layer.
- The outputs of the 4 perceptrons in the hidden layer are denoted by  $h_1, h_2, h_3, h_4$ .
- The red and blue edges are called layer 1 weights.
- $w_1, w_2, w_3, w_4$  are called layer 2 weights.



# Implementing any Boolean function using a network of perceptrons

- We claim that this network can be used to implement any boolean function (linearly separable or not).
- In other words, we can find  $w_1, w_2, w_3, w_4$  such that the truth table of any boolean function can be represented by this network.
- Each perceptron in the middle layer fires only for a specific input (and no two perceptrons fire for the same input).
- Let us see why this network works by taking an example of the XOR function.



# Implementing any Boolean function using a network of perceptrons

- Let  $w_0$  be the bias output of the neuron. It will fire

if  $\sum_{i=1}^4 w_i h_i \geq w_0$ . Here,  $w_0 = 0$ .

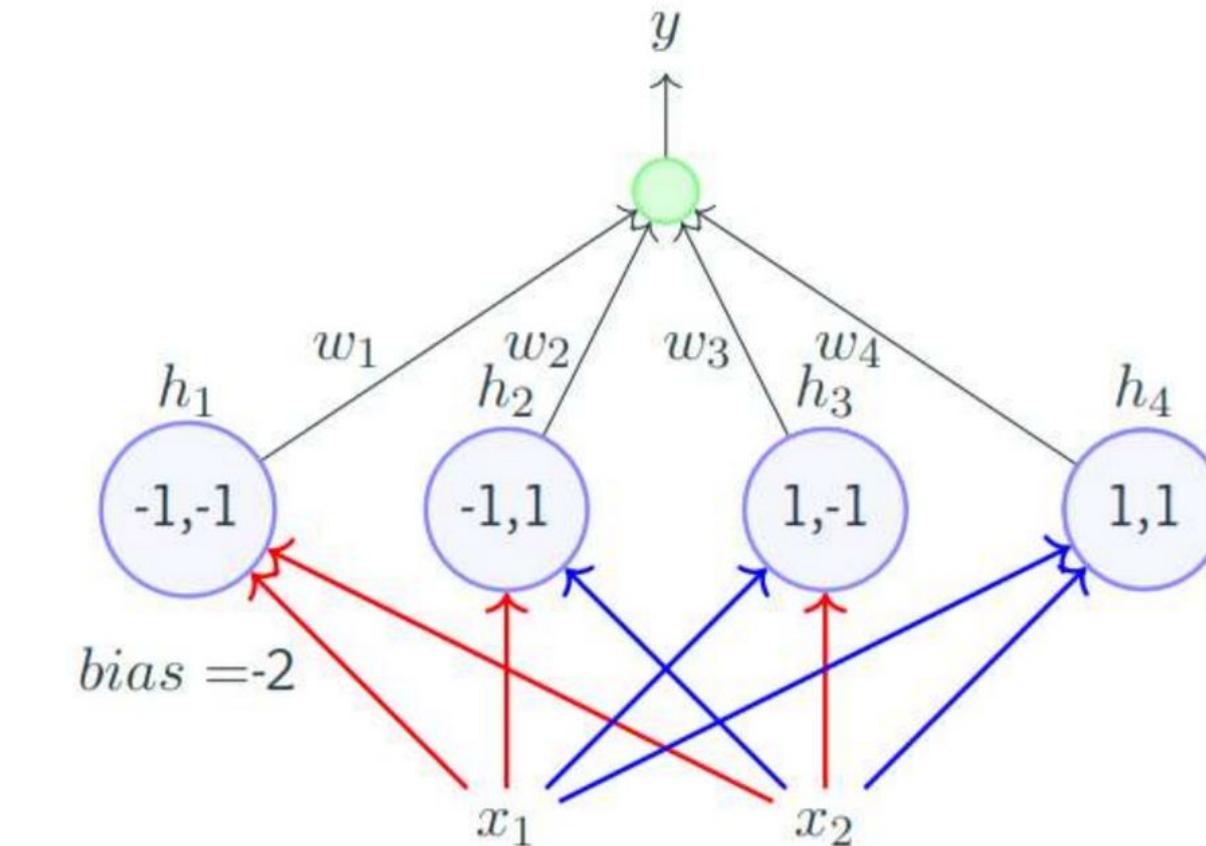
- This results in the following four conditions to implement XOR:

$$w_1 < w_0, w_2 \geq w_0, w_3 \geq w_0, w_4 < w_0.$$

- Unlike before, there are no contradictions now and the system of inequalities can be satisfied.

- Each  $w_i$  is now responsible for one of the 4 possible inputs and can be adjusted to get the desired output for that input.

- Each boolean function will result in a different set of non-contradicting inequalities which can be



| $x_1$ | $x_2$ | XOR | $h_1$ | $h_2$ | $h_3$ | $h_4$ | $\sum_{i=1}^4 w_i h_i$ |
|-------|-------|-----|-------|-------|-------|-------|------------------------|
| 0     | 0     | 0   | 1     | 0     | 0     | 0     | $w_1$                  |
| 0     | 1     | 1   | 0     | 1     | 0     | 0     | $w_2$                  |
| 1     | 0     | 1   | 0     | 0     | 1     | 0     | $w_3$                  |
| 1     | 1     | 0   | 0     | 0     | 0     | 1     | $w_4$                  |



# THANK YOU

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
**UNIVERSITY**

CELEBRATING 50 YEARS

# Machine Learning

## Multi Layer Perceptron

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**Teaching Assistants: Aneesh K B and Nettem Gayathri**

# Machine Learning

## Acknowledgement

---

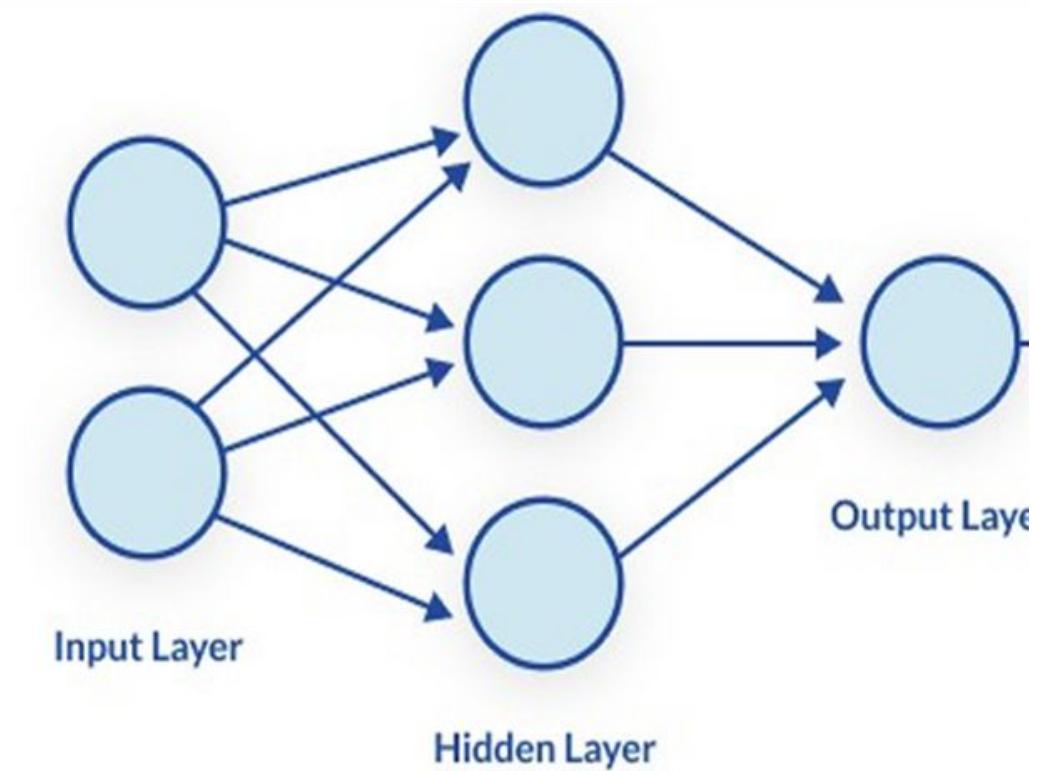


- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Multi Layer Network

A **Multilayer Perceptron (MLP)** is a perceptron that teams up with additional perceptron's, stacked in several layers, to solve complex problems.

Each perceptron in the first layer on the left (the input layer), sends outputs to all the perceptron's in the second layer (the hidden layer), and all perceptrons in the second layer send outputs to the final layer on the right (the output layer).



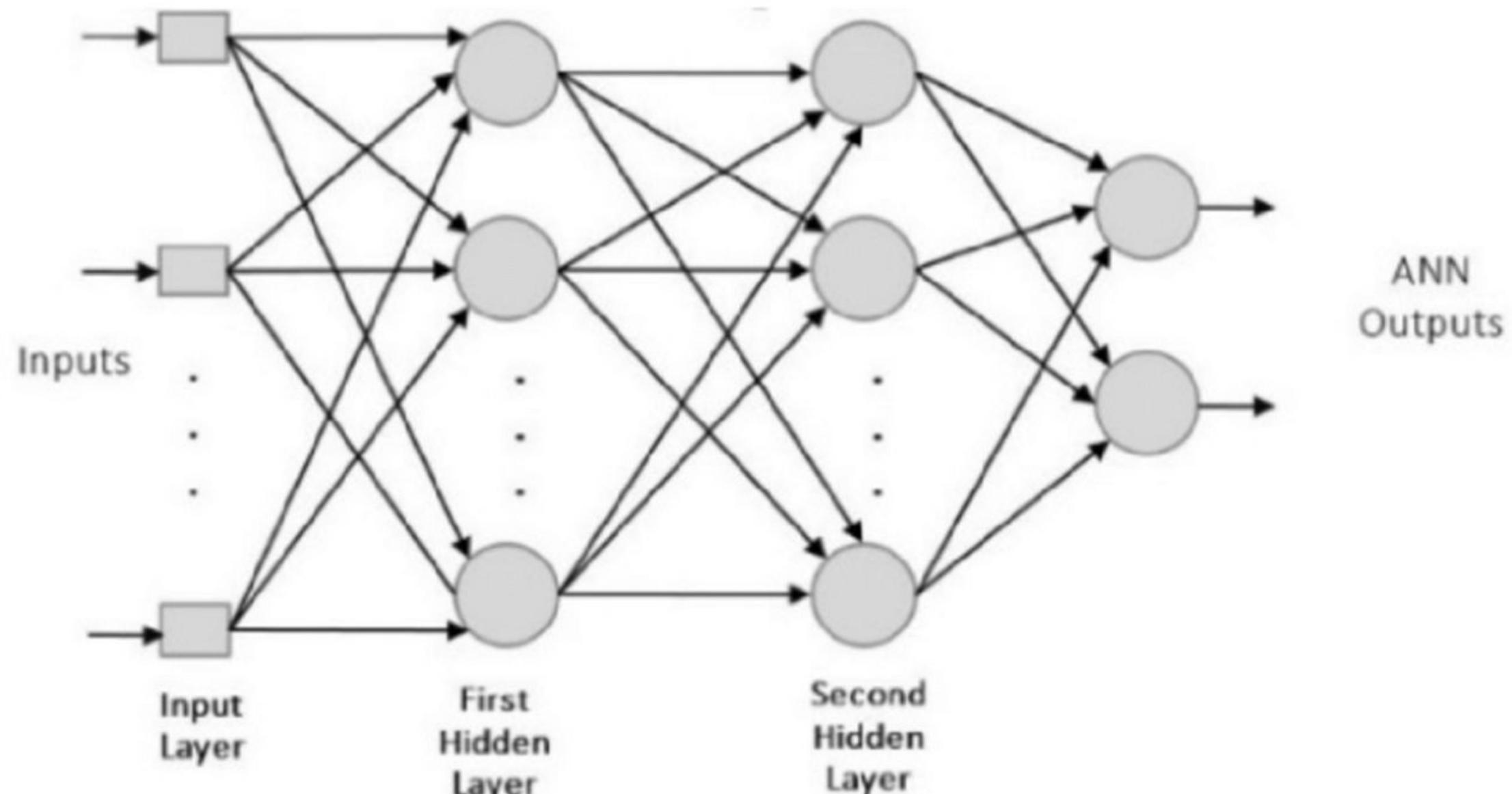
**Multi Layer  
Perceptron**

# Multi Layer Network

---

- A typical ANN consists of the input layer, hidden layer(s) and output layer.
- If it has more than 1 hidden layer, it is called a deep ANN (DNN).
- A fully connected multi-layer neural network is called a Multilayer Perceptron (MLP).
- Each layer has a collection of neurons, and the neurons in a layer interact with all neurons in other layers.
- We use the term nodes or units to represent the neurons in an ANN.
- The number of layers and the number of neurons are referred to as hyperparameters of a neural network.

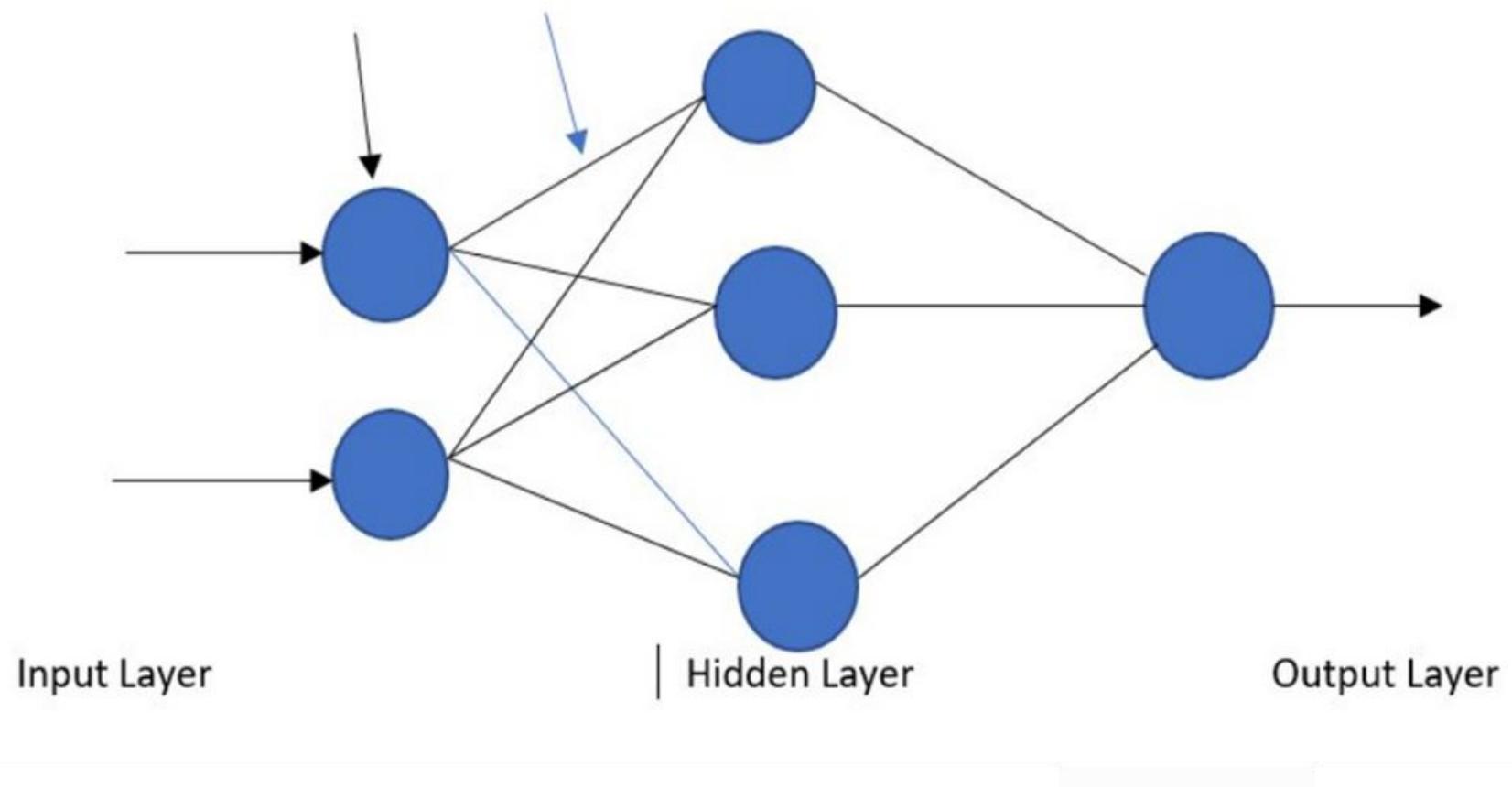
# Multi Layer Network



Multilayer perceptron ANN structure

## Structure of MLP

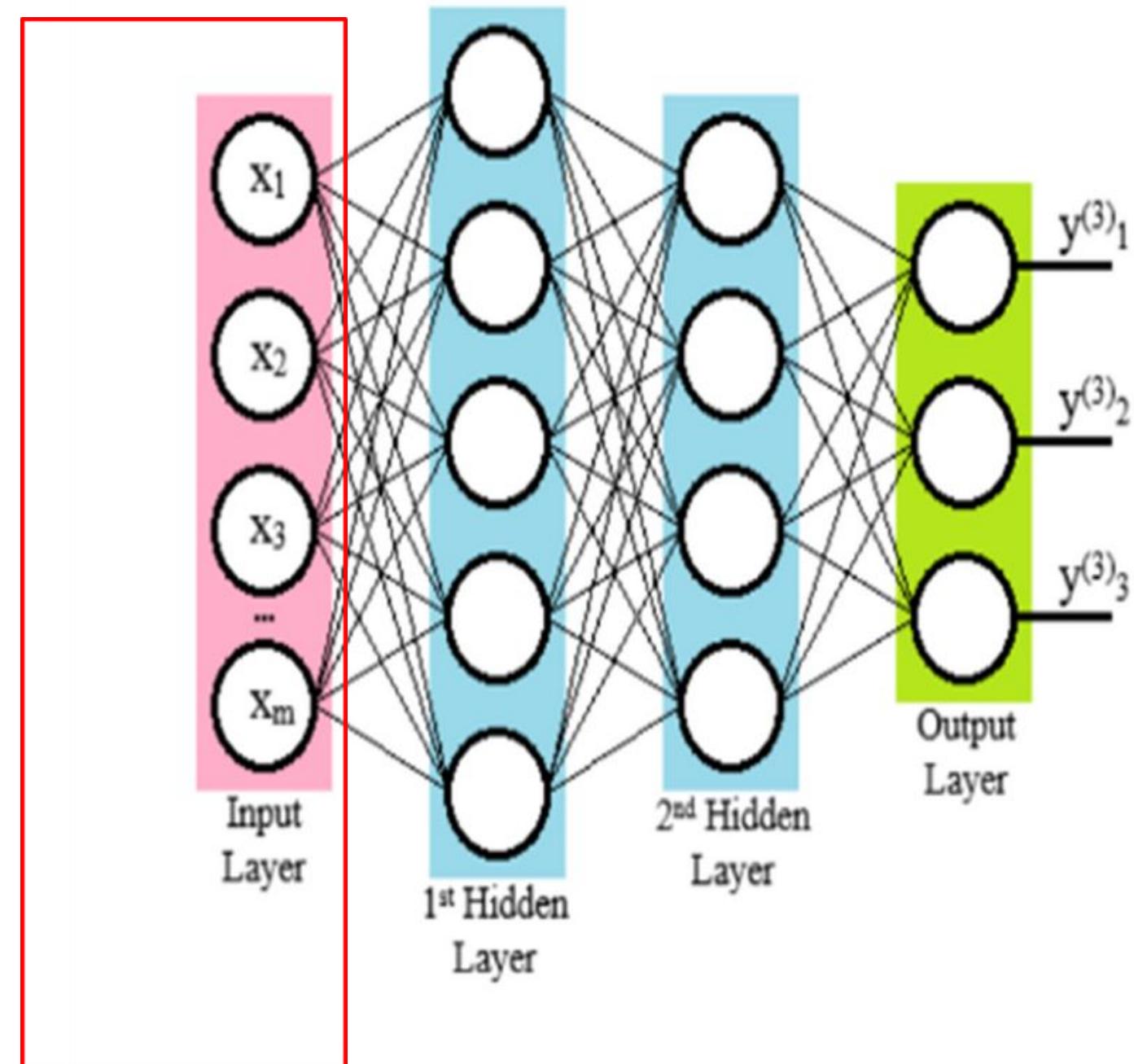
- This network has three main layers that combine to form a complete Artificial Neural Network.
- These layers are as follows:
  - Input Layer
  - Hidden Layer
  - Output Layer



**Structure of MLP**

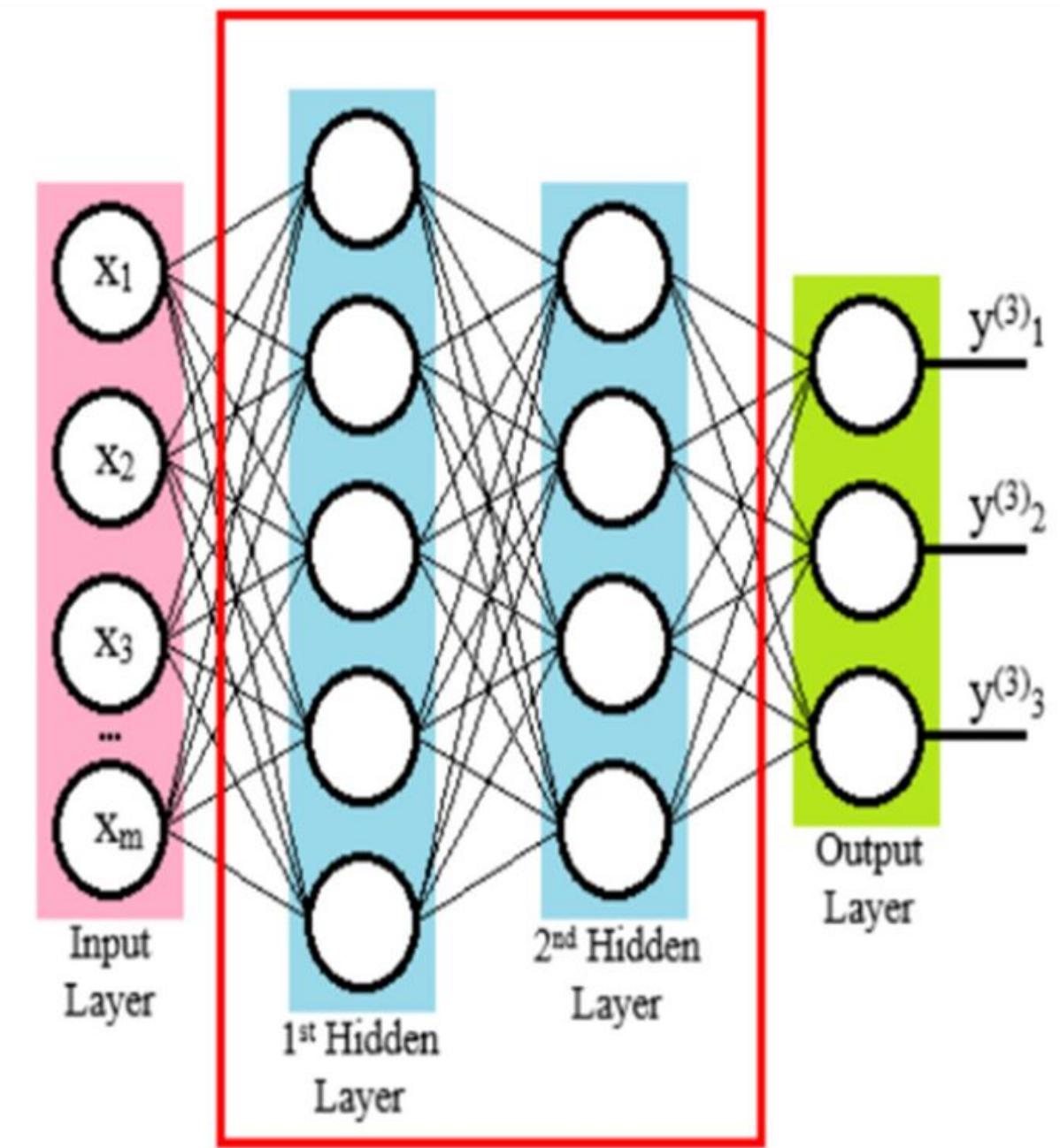
# Input Layer

- It is the initial or starting layer of the Multilayer perceptron.
- It takes input from the training data set and forwards it to the hidden layer.
- There are  $m$  input nodes in the input layer.
- The number of input nodes depends on the number of dataset features.



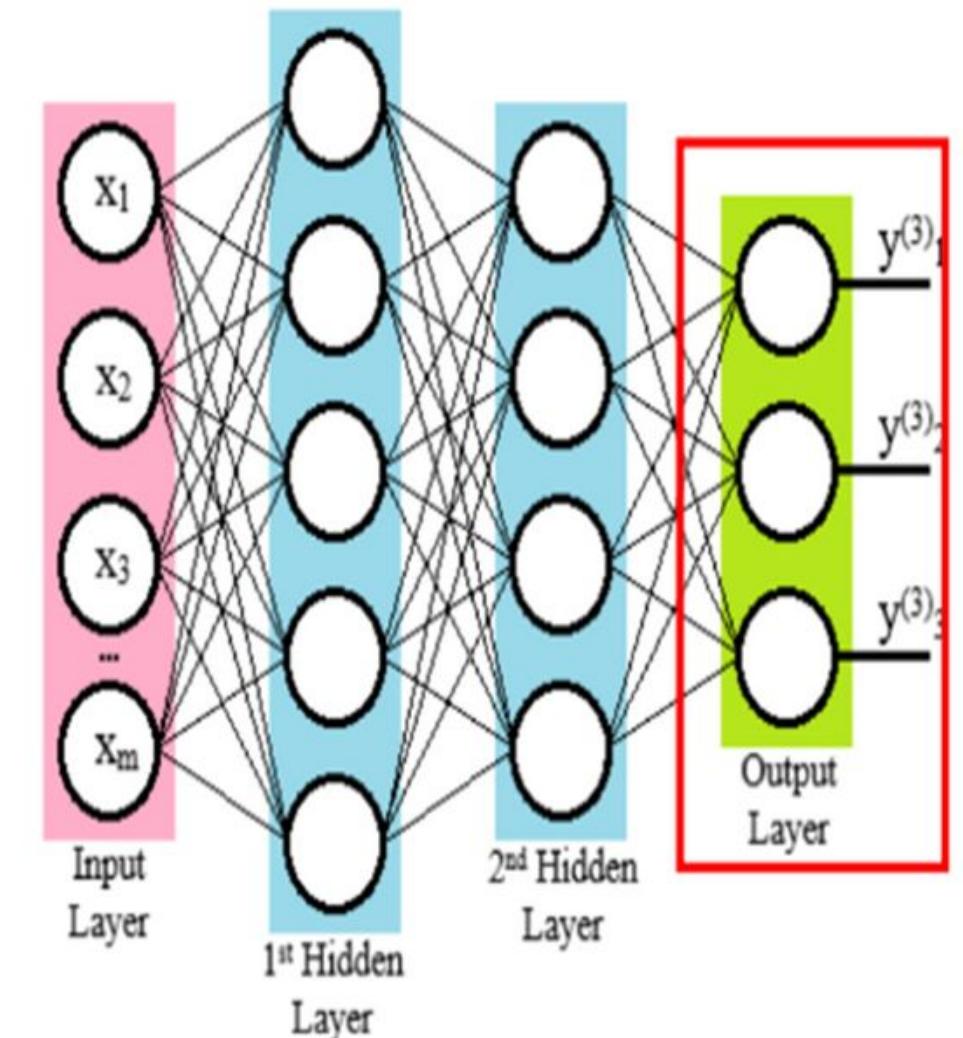
## Hidden Layer

- It is the heart of all Artificial neural networks. This layer comprises all computations of the neural network.
- The hidden layer is responsible for deriving complex relationships between input and output. It identifies patterns in the dataset.
- Several hidden layer nodes should be accurate as few nodes in the hidden layer make the model unable to work efficiently with complex data. More nodes will result in an overfitting problem.



# Output Layer

- This layer gives the estimated output of the Neural Network.
- The number of nodes in the output layer depends on the type of problem.
- For a single targeted variable, use one node. N classification problem, ANN uses N nodes in the output layer.



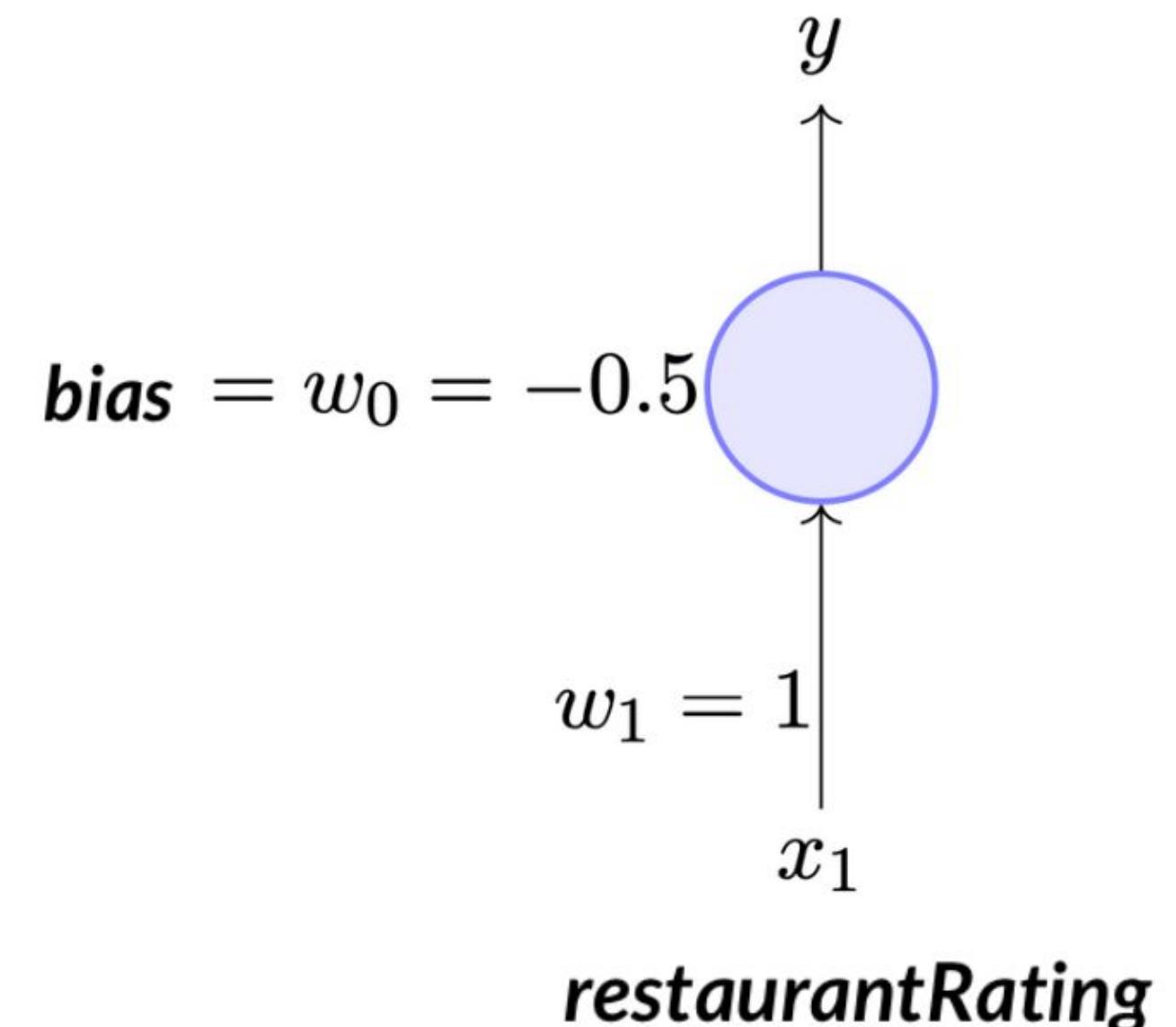
## Working of Multi Layer Perceptron

---

- The input node represents the feature of the dataset.
- Each input node passes the vector input value to the hidden layer.
- In the hidden layer, each edge has some weight multiplied by the input variable. All the production values from the hidden nodes are summed together. To generate the output the activation function is used in the hidden layer to identify the active nodes.
- The output is passed to the output layer.
- Calculate the difference between predicted and actual output at the output layer.
- The model uses backpropagation after calculating the predicted output.

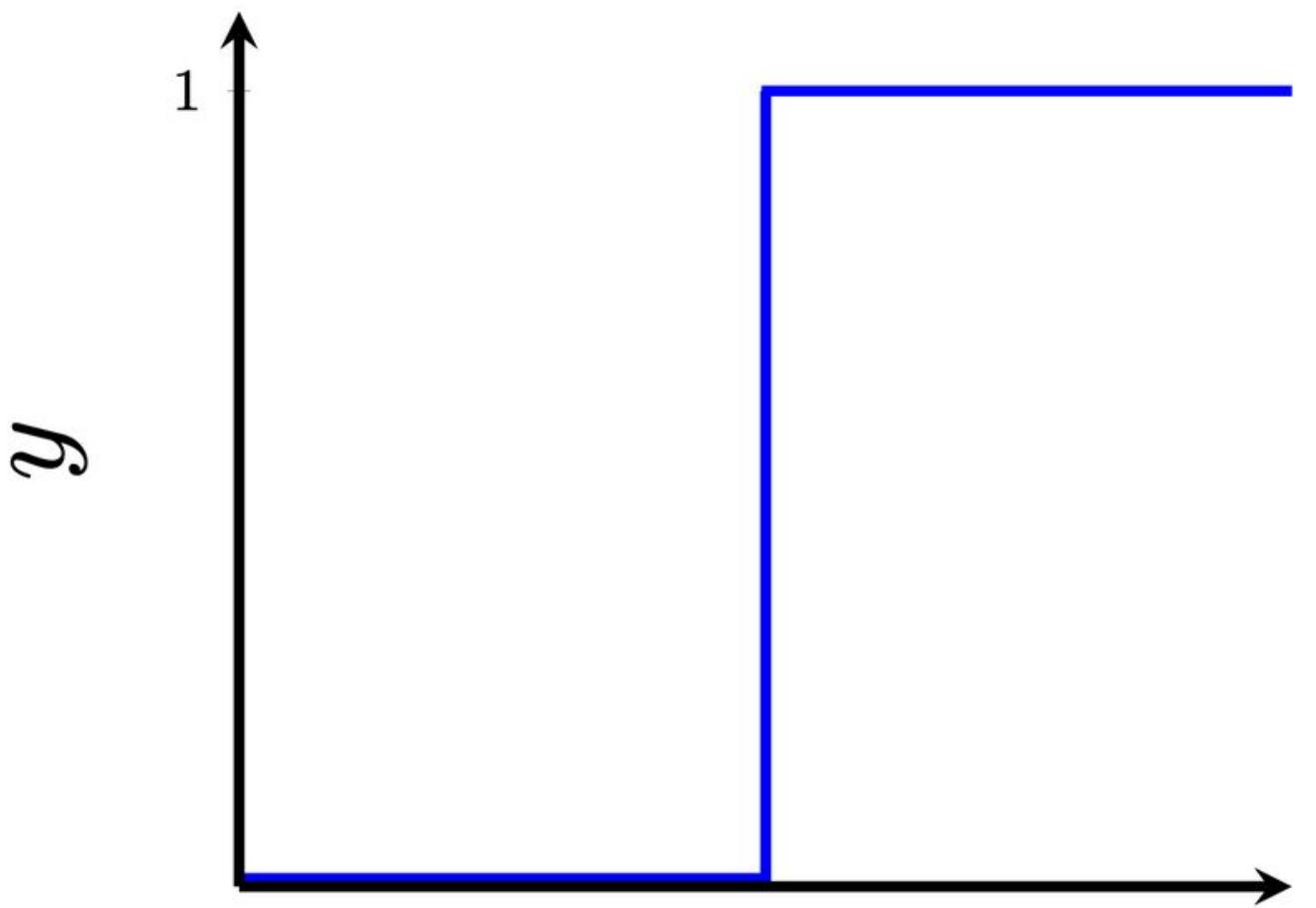
## Perceptron thresholding logic is harsh

- A perceptron will fire if the weighted sum of its inputs is greater than the threshold ( $-w_0$ )
- The thresholding logic used by a perceptron is very harsh!
- Let's consider our problem of deciding whether to dine in at a restaurant, based on just one input ( $x_1$  = restaurantRating, which lies between 0 and 1).
- If the threshold is 0.5 ( $w_0 = -0.5$ ) and  $w_1 = 1$ , what would be the decision for a restaurant with a restaurantRating of 0.51? (dine in).
- What about a movie with restaurantRating = 0.49 ? (don't dine in).
- It seems quite strict to decide to dine in at a restaurant with a rating of 0.51 but not at one with a rating of 0.49.



## Perceptron thresholding logic is harsh

- This behavior is not a characteristic of the specific problem we chose or the specific weight and threshold that we chose
- It is a characteristic of the perceptron function itself which behaves like a step function
- There will always be this sudden change in the decision (from  $\sum_{i=1}^n w_i x_i < -w_0$  to 1) when  $\sum_{i=1}^n w_i x_i$  crosses the threshold
- For most real-world applications we would expect a smoother decision function which gradually changes from 0 to 1



$$z = \sum_{i=1}^n w_i x_i$$

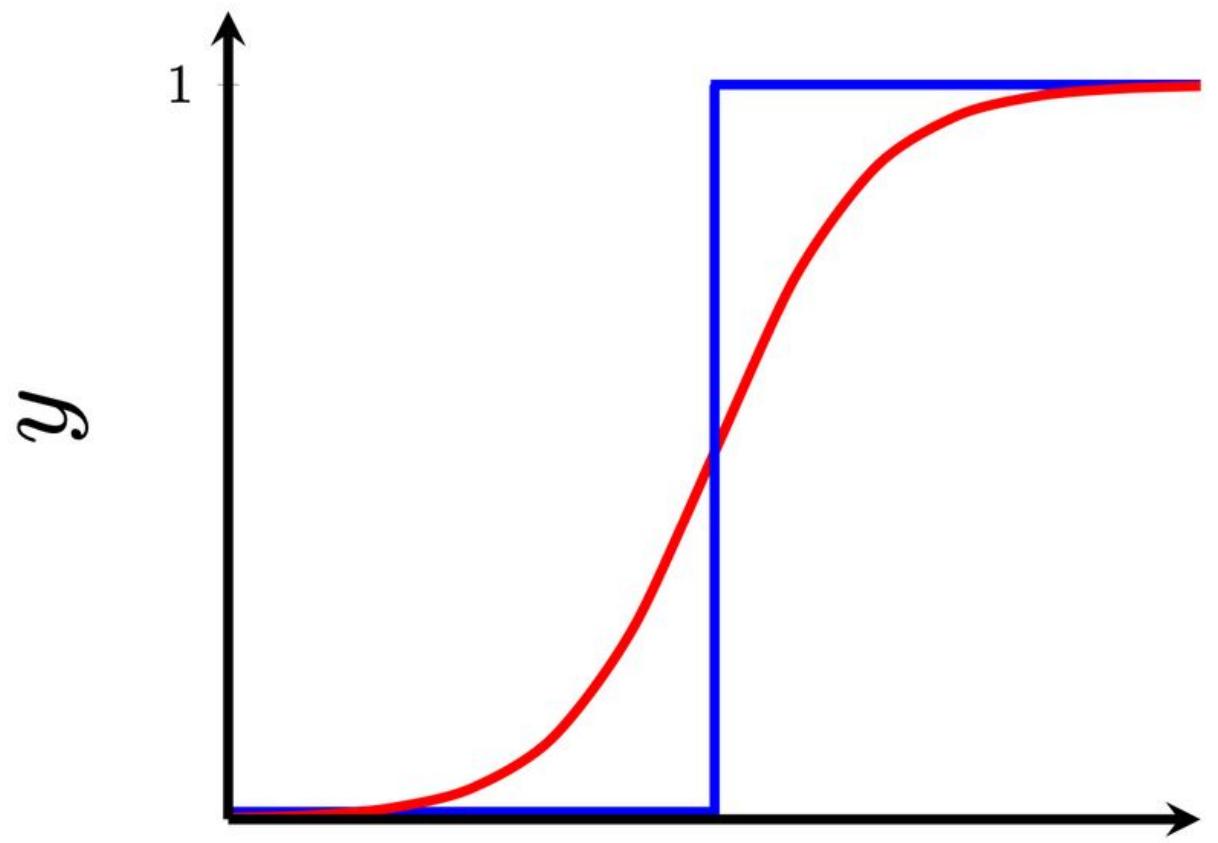
# Sigmoid Neuron

- Introducing sigmoid neurons where the output function is much smoother than the step function.
- Here is one form of the sigmoid function called the logistic function

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^n w_i x_i)}}$$

(- $w_0$ )

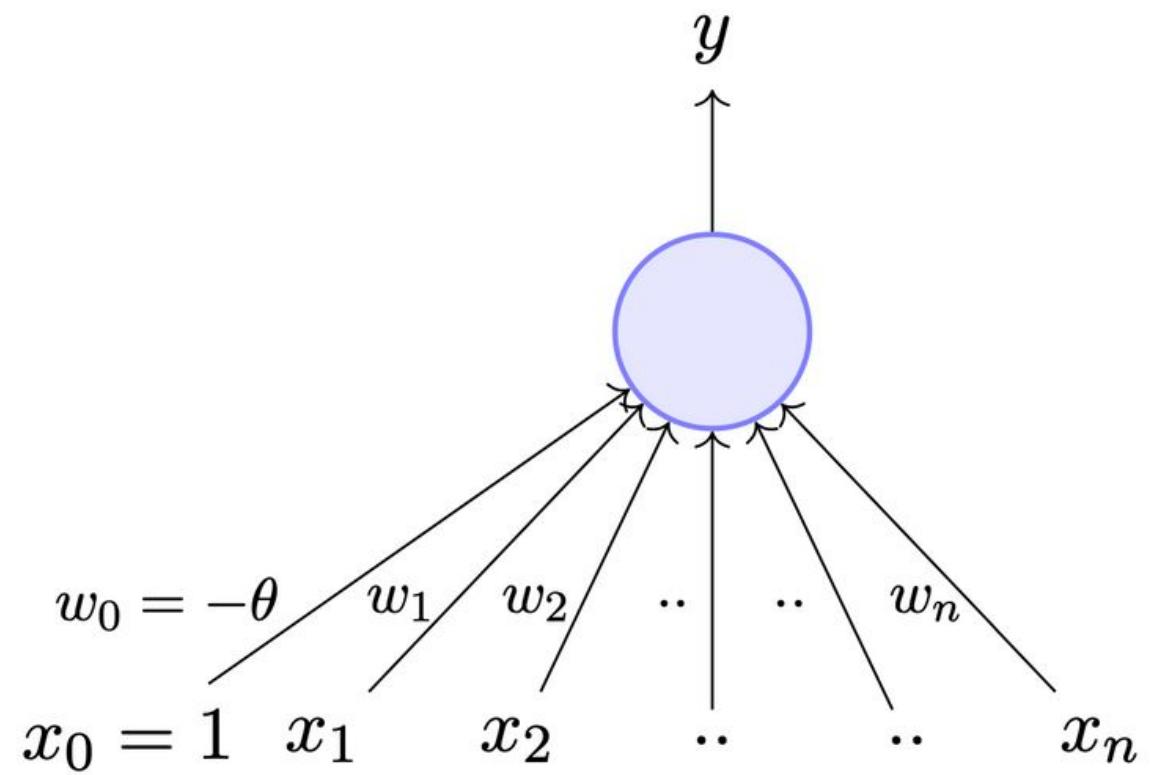
- We no longer see a sharp transition around the threshold
- Also, the output  $y$  is no longer binary but a real value between 0 and 1 which can be interpreted as a probability
- Instead of a like/dislike decision, we get the probability of



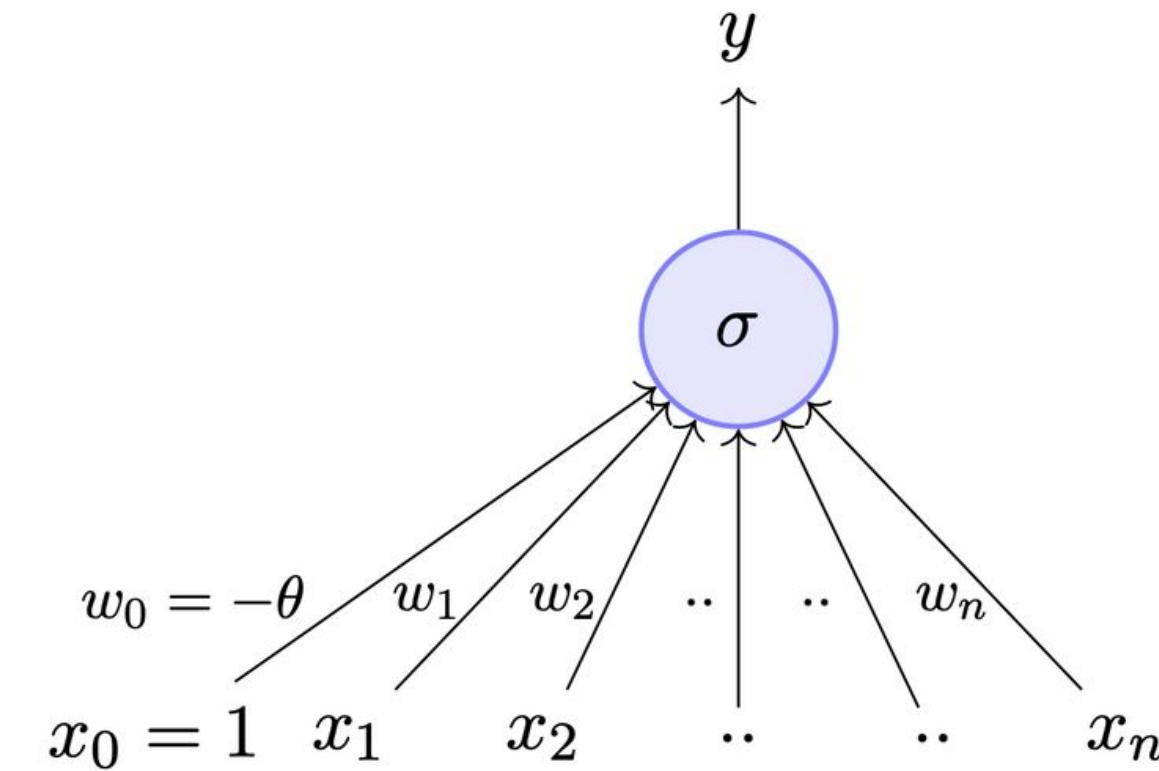
$$z = \sum_{i=1}^n w_i x_i$$

# Perceptron vs Sigmoid Neuron

**Perceptron**



**Sigmoid (logistic) Neuron**

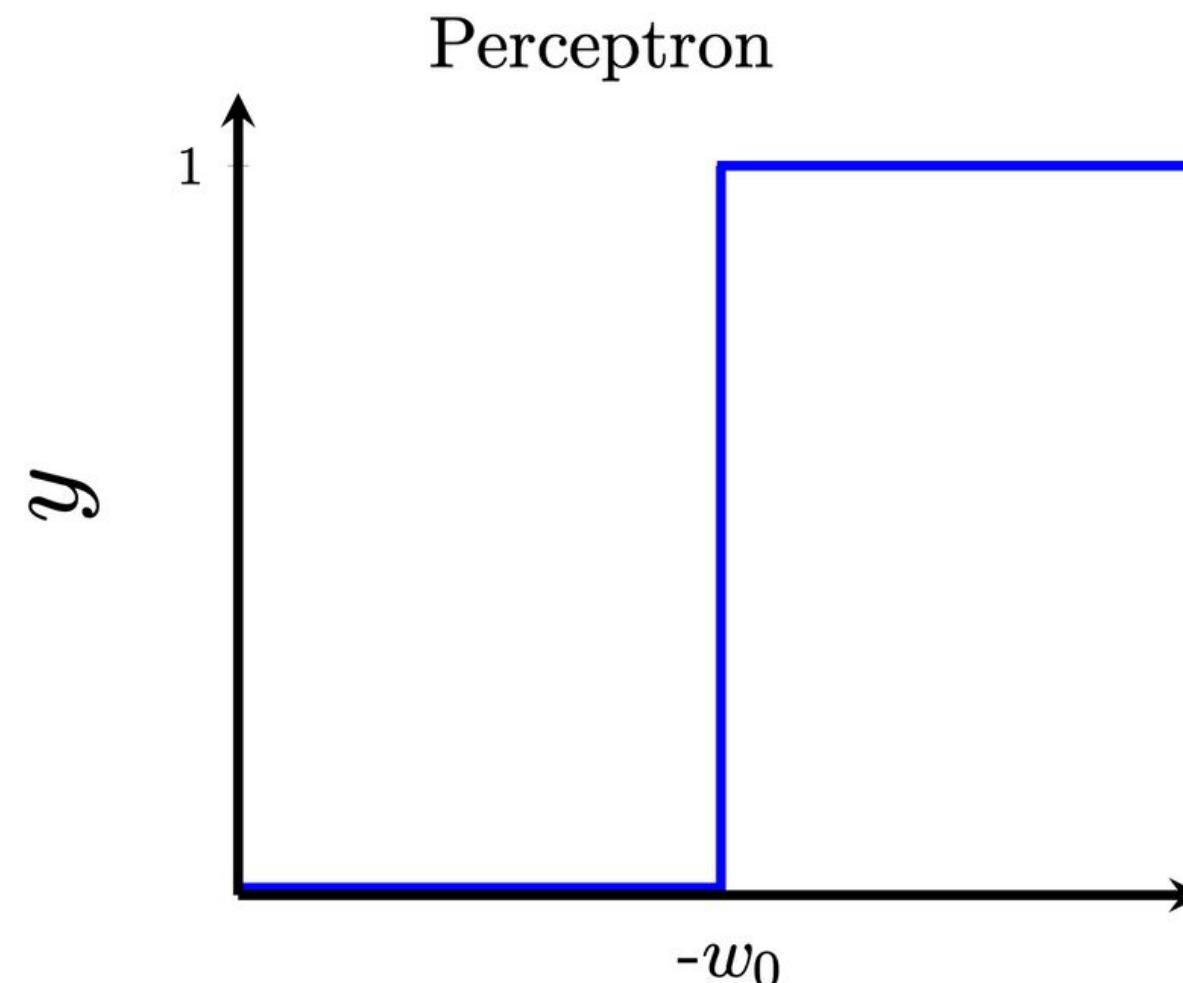


$$y = 1 \quad if \sum_{i=0}^n w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^n w_i * x_i < 0$$

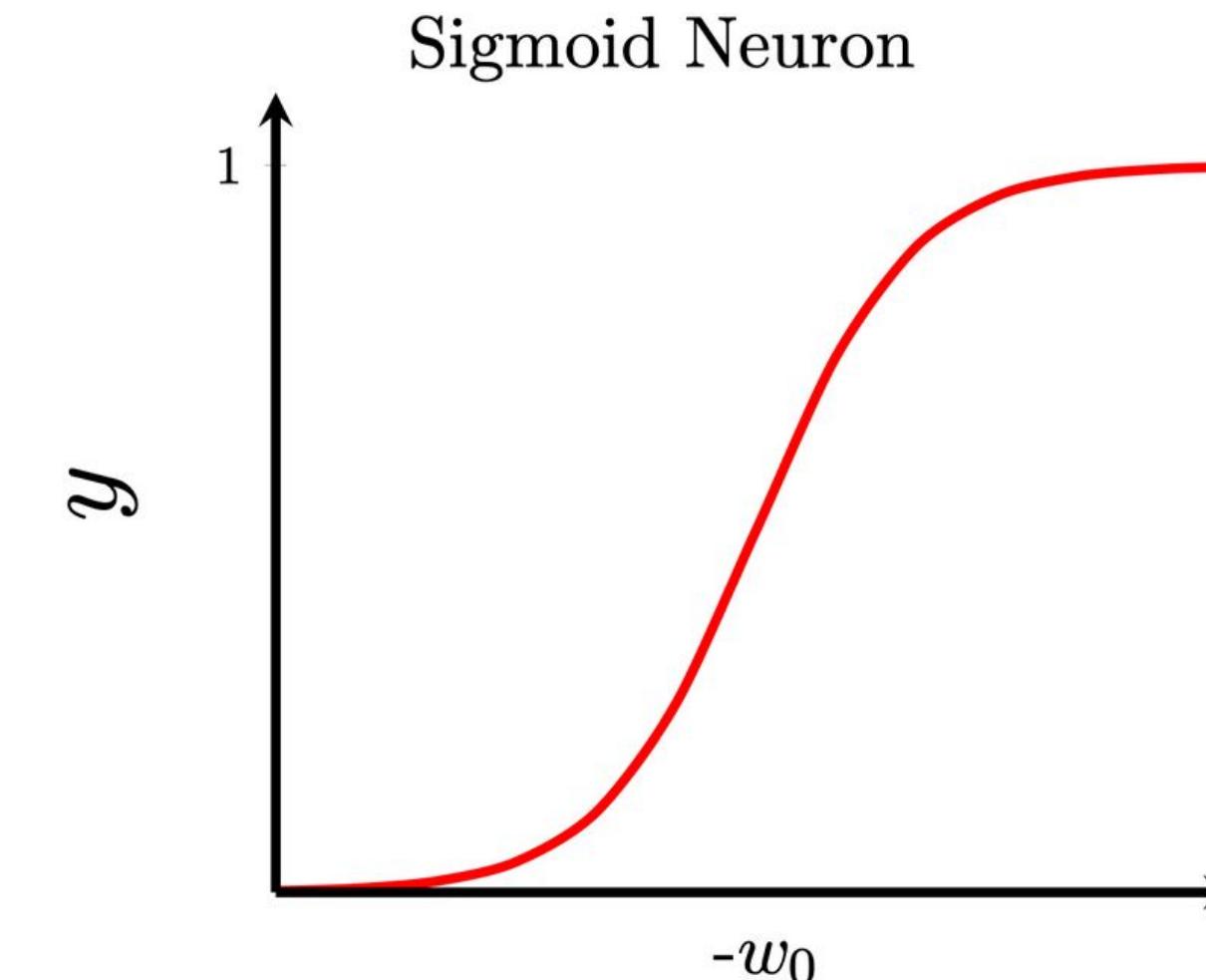
$$y = \frac{1}{1 + e^{-(\sum_{i=0}^n w_i x_i)}}$$

# Perceptron vs Sigmoid Neuron



$$z = \sum_{i=1}^n w_i x_i$$

Not smooth, not continuous (at  $w_0$ ), **not differentiable**



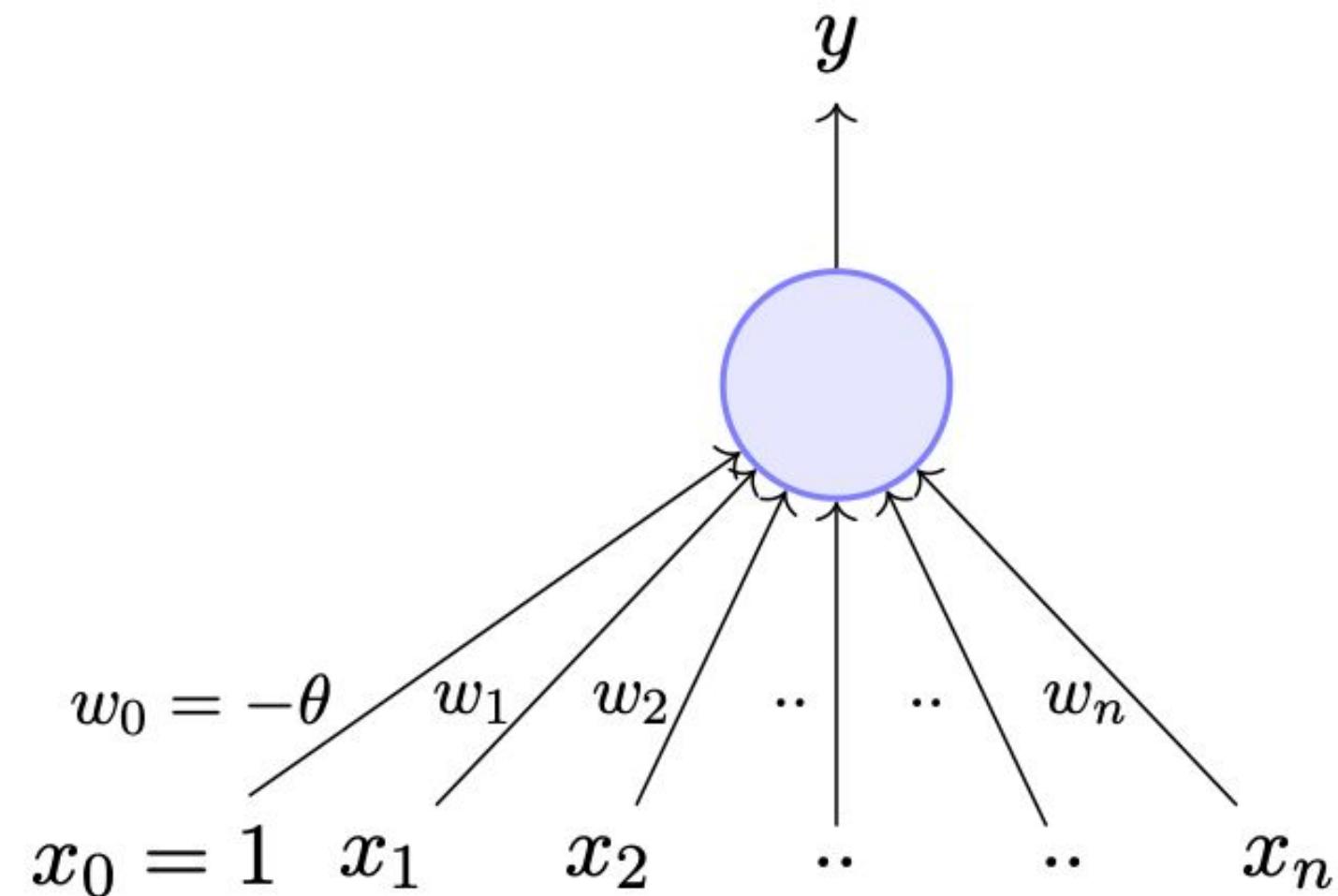
$$z = \sum_{i=1}^n w_i x_i$$

Smooth, continuous, **differentiable**

## Learning weights when output is real-valued

Well, just as we had an algorithm for learning the weights of a perceptron, we also need a way of learning the weights of a sigmoid neuron.

### Sigmoid (logistic) Neuron



# Components of a typical Supervised Machine Learning Set up

- **Data:**  $\{x_i, y_i\}_{i=1}^n$
- **Model:** Our approximation of the relation between x and y. For example,

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x})}}$$

or  $\hat{y} = \mathbf{w}^T \mathbf{x}$

or  $\hat{y} = \mathbf{x}^T \mathbf{W} \mathbf{x}$

or just about any function

- **Parameters:** In all the above cases, w is a parameter which needs to be learned from the data
- **Learning algorithm:** An algorithm for learning the parameters (w) of the model (for example, perceptron learning algorithm, gradient descent, etc.)
- **Objective/Loss/Error function:** To guide the learning algorithm - the learning algorithm should aim to minimize the loss function

## Example - Typical Supervised Machine Learning Set up

- **Data:**  $\{x_i = \text{restaurant}, y_i = \text{like/dislike}\}_{i=1}^n$
- **Model:** Our approximation of the relation between x and y (the probability of liking a restaurant).

$$\hat{y} = \frac{1}{1 + e^{-(\mathbf{w}^T \mathbf{x})}}$$

- **Parameters:**  $w$
- **Learning algorithm:** Gradient Descent
- **Objective/Loss/Error function:** One possible

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

$y$  and  $\hat{y}$

The learning algorithm should aim to find a  $w$  which minimizes the above function (squared error between  $y$  and  $\hat{y}$ )



# THANK YOU

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
**UNIVERSITY**

CELEBRATING 50 YEARS

# Machine Learning

## Forward propagation

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**

**Teaching Assistant: Nettem Gayathri (Semester VII)**

# Machine Learning

## Acknowledgement

---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Machine Learning

## Acknowledgement

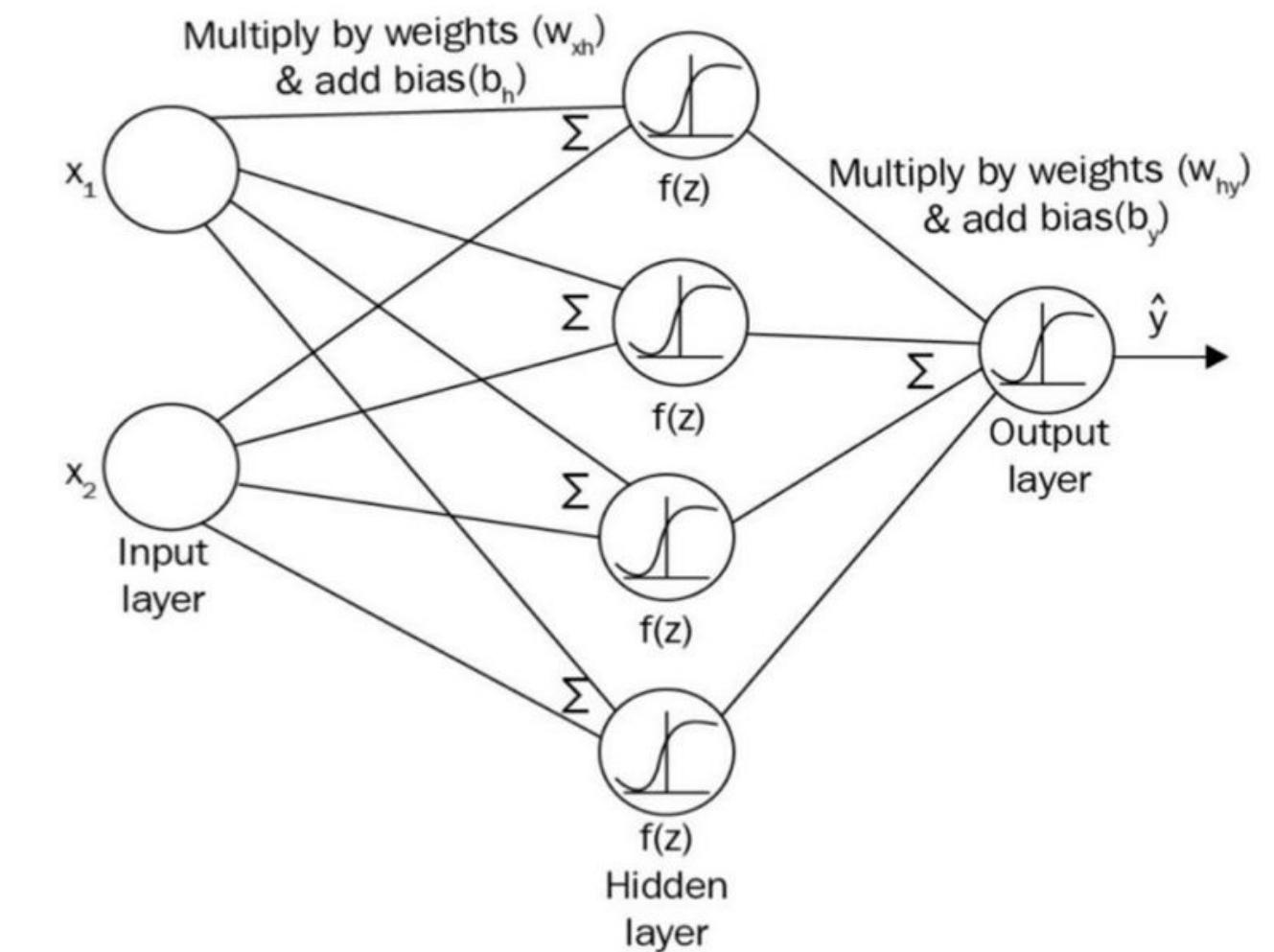
---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

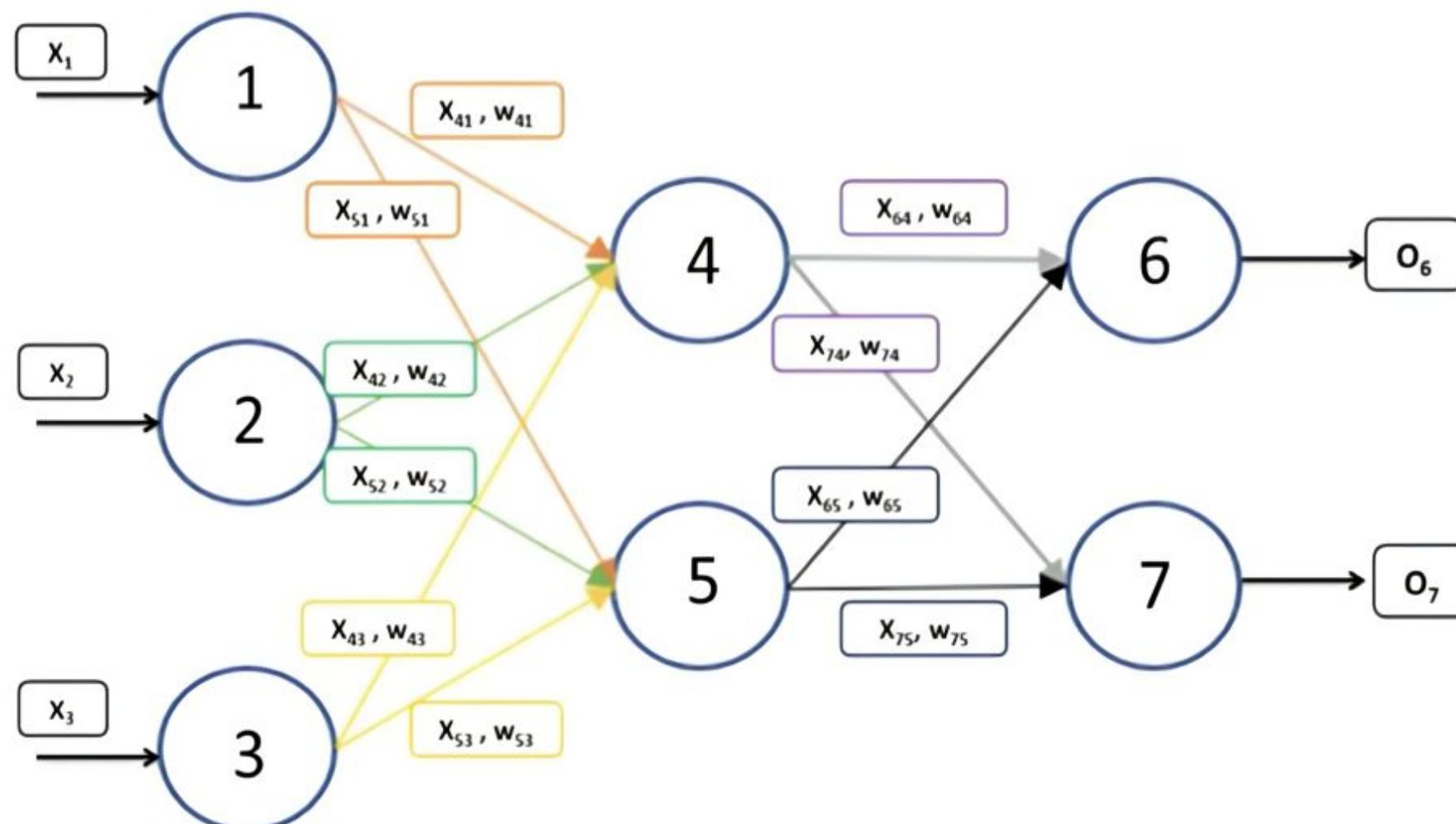
# Forward propagation

- In a neural network, the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.
- At each neuron in a hidden or output layer, the processing happens in two steps:
  - Preactivation: it is a weighted sum of inputs i.e. the linear transformation of weights w.r.t to inputs available. Based on this aggregated sum and activation function the neuron decides whether to pass this information further or not.
  - Activation: the calculated weighted sum of inputs is passed to the activation function. An activation function is a mathematical function that adds non-linearity to the network. There are four commonly used and popular activation functions – sigmoid, hyperbolic tangent(tanh), ReLU and Softmax.



# Forward propagation

- In an ANN, the total number of layers = No. of hidden layers + Output layers.



Layer 0

Input  
layer

Layer 1

Hidden  
layer

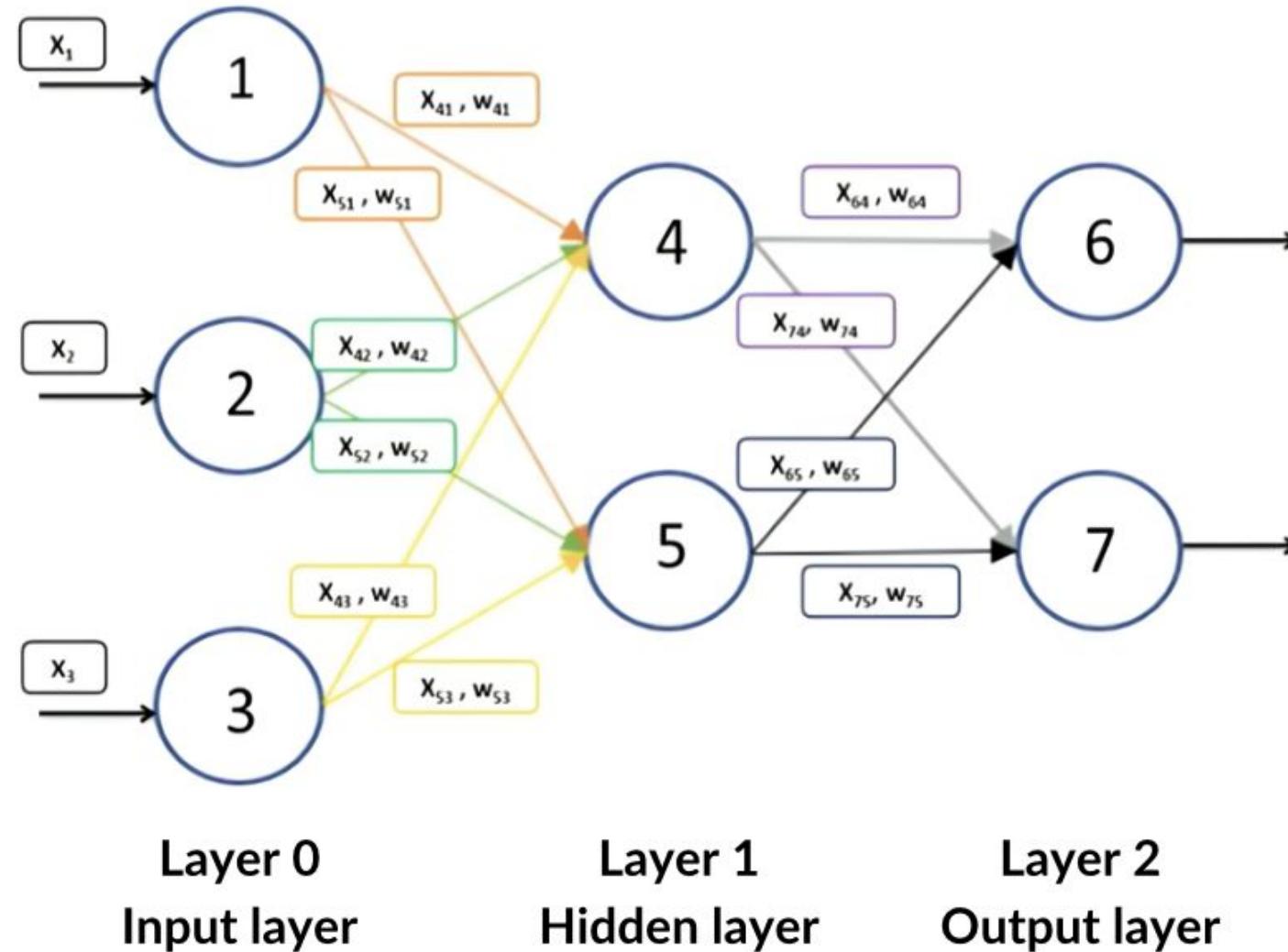
Layer 2

Output  
layer

- Consider a 2-layer neural network -
  - 3 inputs -  $x_1, x_2, x_3$ .
  - 2 hidden layer units.
  - 2 output layer units.
- Each input is multiplied by its corresponding weights, and a bias term is added.
- The computed value is then passed through an activation function in the hidden layer.
- This activated output is propagated through the hidden layer, eventually contributing to the final output.

# Forward propagation

- We begin by initializing the weight matrix. The dimensions of this matrix are determined by the number of units in the previous layer multiplied by the number of units in the current layer.
- In real-world scenarios, we don't know the relative importance of each input, so we randomly initialize the weights and biases.

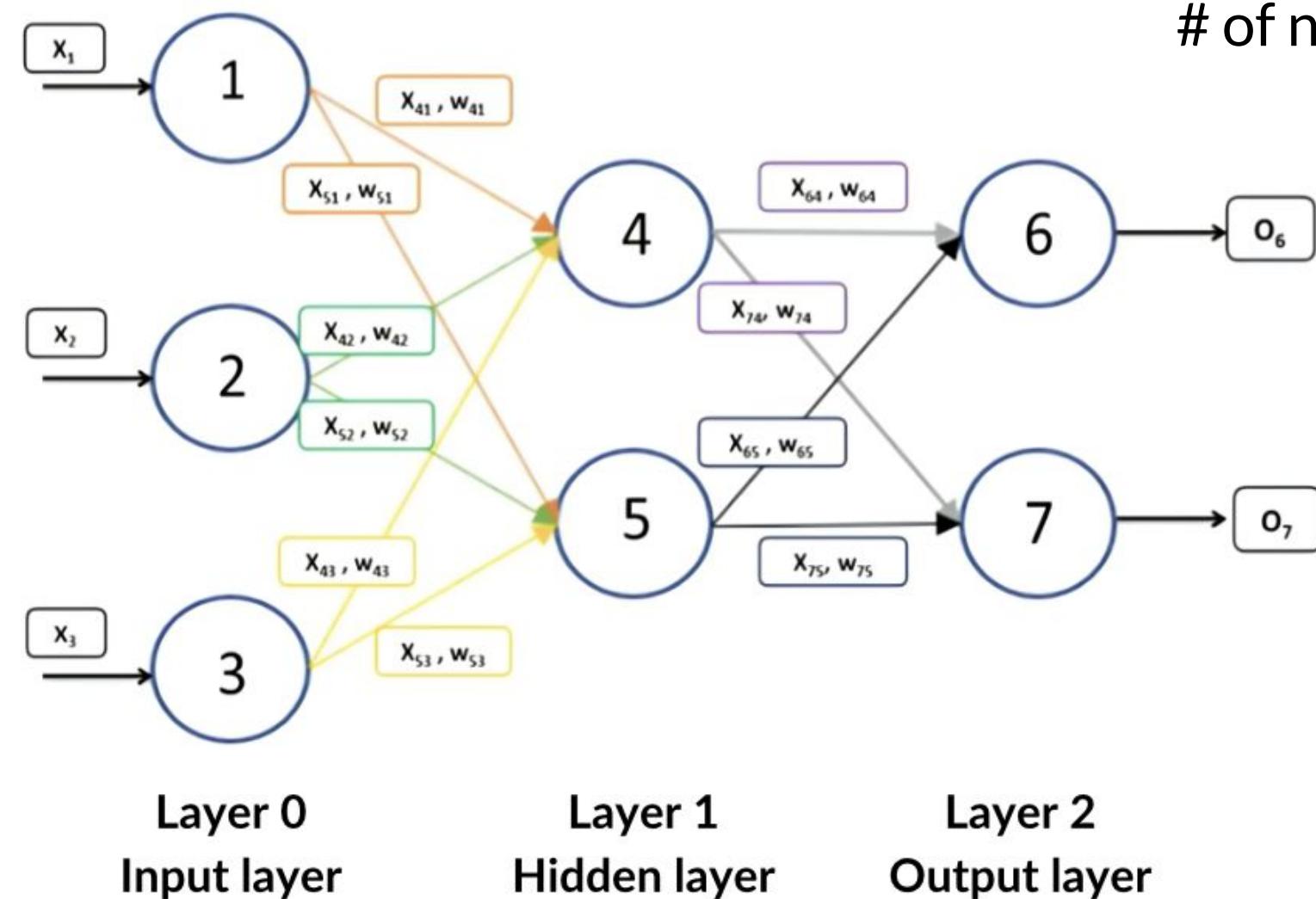


- For instance, if our input layer has 3 units and the next layer (a hidden layer) has 2 units, the weight matrix will have dimensions of 3 rows and 2 columns. We'll denote this matrix as  $W1$ . If we have additional layers, we'll introduce additional weight matrices, such as  $W2$ ,  $W3$ , and so on.
- In general, if a layer L has N neurons and the subsequent layer  $L+1$  has M neurons, the corresponding weight matrix will be an N-by-M matrix, with N rows and M columns.

# Forward propagation

- Shape of the weight matrix =

# of neurons in the previous layer \* # of neurons in the current layer



$$W = \begin{bmatrix} w_{41} & w_{51} \\ w_{42} & w_{52} \\ w_{43} & w_{53} \end{bmatrix}_{3 \times 2}$$

in  $w_{ji}$ ,

$j = \text{current layer neuron number}$

$i = \text{previous layer neuron number}$

## Notations used

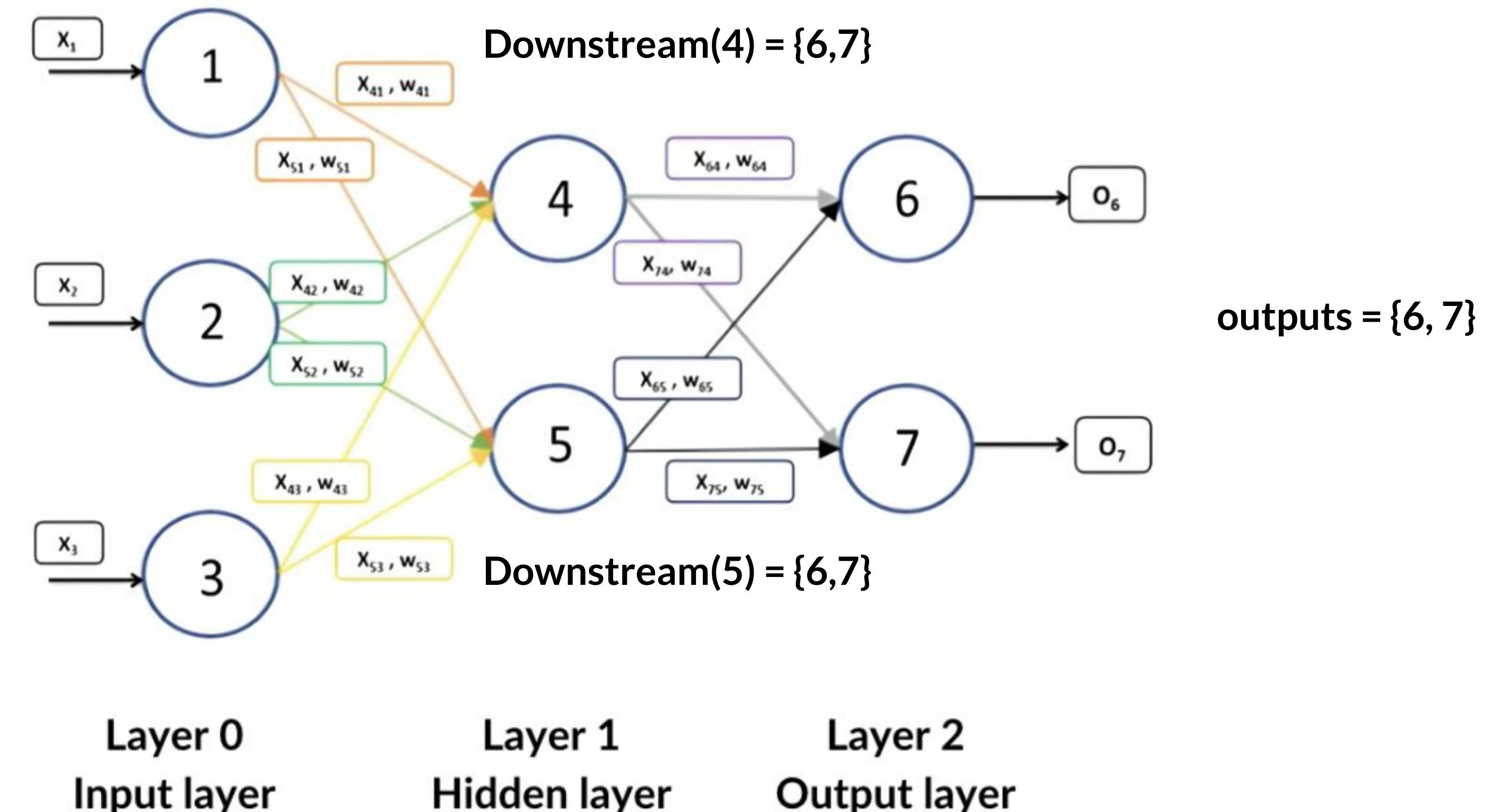
- $x_{ji}$  = the ith input to unit j
- $w_{ji}$  = the weight associated with the ith input to unit j
- $net_j = \sum_i w_{ji}x_{ji}$  (the weighted sum of inputs for unit j)
- $o_j$  = the output computed by unit j
- $t_j$  = the target output for unit j
- $\sigma$  = the sigmoid function
- **outputs** = the set of units in the final layer of the network
- **Downstream(j)** = the set of units whose immediate inputs include the output of unit j

Applicable only to Neurons in Output Layer:  $o_j$  and  $t_j$

Applicable only to Neurons in Hidden Layer:  $Downstream(j)$

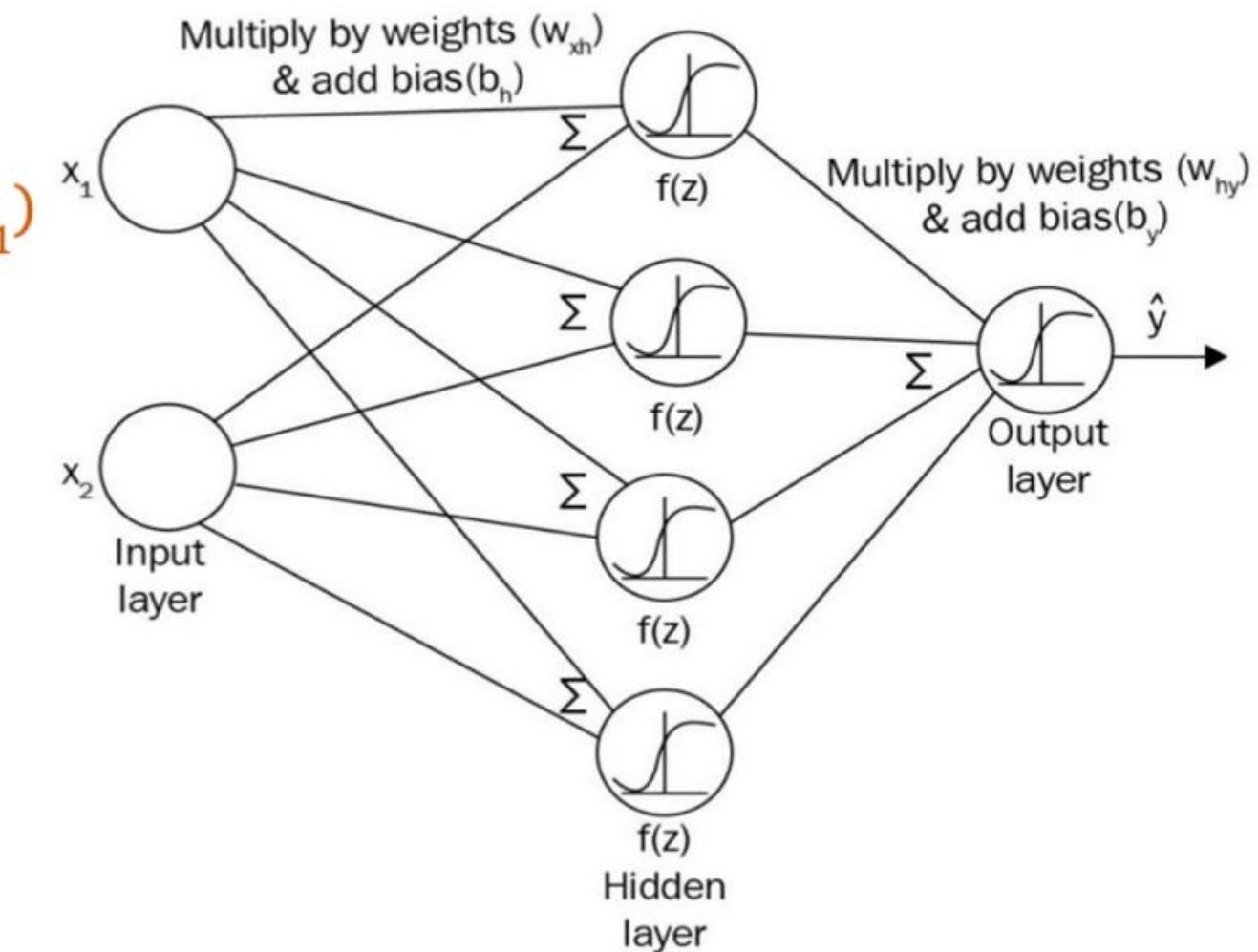
# Downstream

Downstream(j) where j is a neuron in the hidden layer

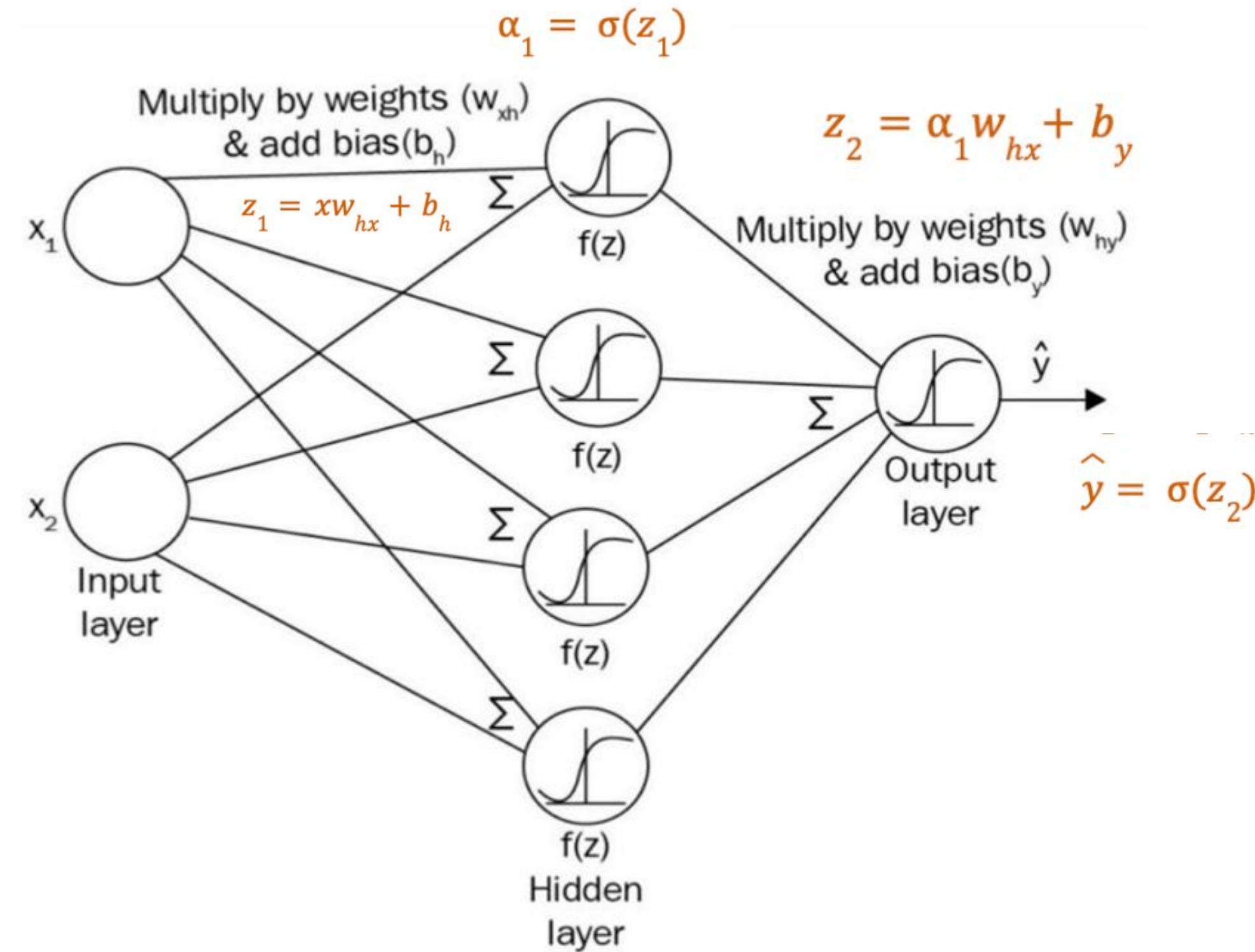


# Forward propagation

- Let the weight and bias between input and the hidden layer be represented as  $w_{hx}$  and  $b_h$  respectively.  $z_1 = xw_{hx} + b_h$
- This is passed through the hidden layer, where we apply the sigmoid activation.  $\alpha_1 = \sigma(z_1)$
- Our inputs to the output layer are now ready.
- Multiply  $\alpha_1$  with the weight matrix  $w_{hy}$  and add the bias  $b_y$ . (weight matrix and bias between hidden layer and output layer).  $z_2 = \alpha_1 w_{hx} + b_y$
- This is passed to the sigmoid activation in the output layer.
- We get the final output as  $\hat{y} = \sigma(z_2)$

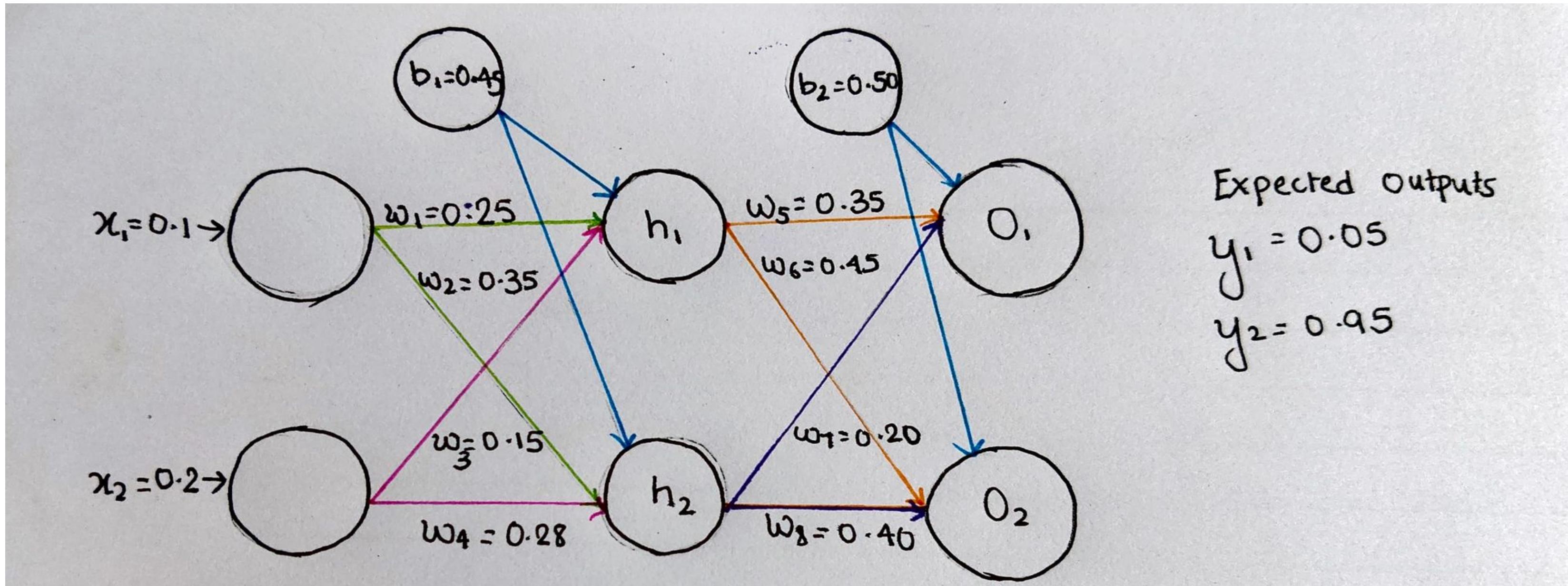


# Forward propagation



# Forward propagation - Numericals

## Question 1



# Forward propagation - Numericals

Output Calculation for hidden layer

$$\begin{aligned} \text{net}_1 &= w_1 * x_1 + w_3 * x_2 + b_1 \\ &= 0.25 * 0.1 + 0.15 * 0.2 + 0.45 \\ &= 0.505 \end{aligned}$$

$$\begin{aligned} \text{net}_2 &= w_2 * x_1 + w_4 * x_2 + b_2 \\ &= 0.35 * 0.1 + 0.28 * 0.2 + 0.45 \\ &= 0.541 \end{aligned}$$

Using sigmoid as the activation function:

$$\begin{aligned} \text{Out}(h_1) &= \sigma(\text{net}_1) = \frac{1}{1 + e^{-(\text{net}_1)}} = \frac{1}{1 + e^{-0.505}} \\ &= 0.623633 \end{aligned}$$

$$\begin{aligned} \text{Out}(h_2) &= \sigma(\text{net}_2) = \frac{1}{1 + e^{-(\text{net}_2)}} = \frac{1}{1 + e^{-0.541}} \\ &= 0.632045 \end{aligned}$$

# Forward propagation - Numericals

Output calculation at the Output layer

$$\begin{aligned}
 \text{net } 3 &= w_5 * \text{Out}(h_1) + w_7 * \text{Out}(h_2) + b_2 \\
 &= 0.35 * 0.623633 + 0.45 * 0.632045 \\
 &\quad + 0.50 \\
 &= 0.844680
 \end{aligned}$$

$$\begin{aligned}
 \text{net } 4 &= w_6 * \text{Out}(h_1) + w_8 * \text{Out}(h_2) + b_2 \\
 &= 0.45 * 0.623633 + 0.4 * 0.632045 \\
 &\quad + 0.50 \\
 &= 1.0334528
 \end{aligned}$$

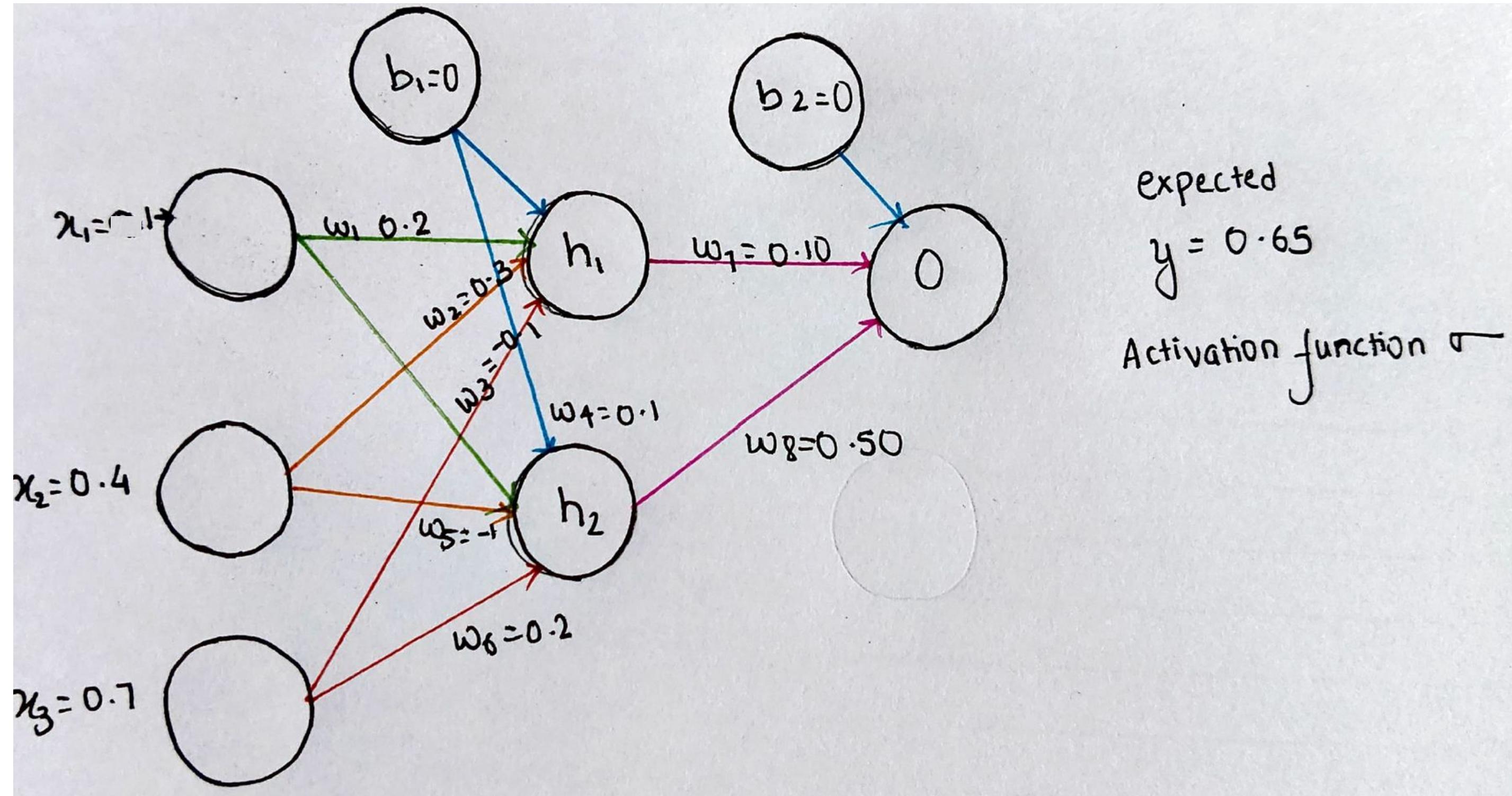
$$\begin{aligned}
 O_1 &= \sigma(\text{net}_3) = \frac{1}{1 + e^{-0.844680}} \\
 &= 0.699450
 \end{aligned}$$

$$\begin{aligned}
 O_2 &= \sigma(\text{net}_4) = \frac{1}{1 + e^{-1.0334528}} \\
 &= 0.737584
 \end{aligned}$$

$O_1 = 0.699450$ 
 $O_2 = 0.737584$

# Forward propagation - Numericals

## Question 2



# Forward propagation - Numericals

Output calculation for hidden layer

$$\begin{aligned} \text{net}_1 &= w_1 * x_1 + w_2 * x_2 + w_3 * x_3 + b_1 \\ &= 0.2 * 1 + 0.3 * 0.4 + (-0.1) * 0.7 + 0 \\ &= 0.25 \end{aligned}$$

$$\sigma(\text{net}_1) = \frac{1}{1+e^{-0.25}} = 0.5622$$

$$\begin{aligned} \text{net } 2 &= w_4 * x_1 + w_5 * x_2 + w_6 * x_3 + b_2 \\ &= 0.1 * 1 + (-1) * 0.4 + 0.2 * 0.7 + 0 \\ &= -0.16 \end{aligned}$$

$$\sigma(\text{net}_2) = \frac{1}{1+e^{-0.16}} = 0.4601$$

Output calculation for output layer

$$\begin{aligned} \text{net}_0 &= w_7 * \sigma(\text{net}_1) + w_8 * \sigma(\text{net}_2) + b_2 \\ &= 0.10 * 0.5622 + 0.50 * 0.4601 + 0 \\ &= 0.28627 \end{aligned}$$

$$\begin{aligned} \sigma(\text{net}_0) &= \frac{1}{1+e^{-0.28627}} \\ \hat{y} &= 0.57108 = 0.5711 \end{aligned}$$

# Forward propagation - Numericals

$$\begin{aligned} \text{LOSS} &= \frac{1}{2} (y - \hat{y})^2 \\ &= \frac{1}{2} (0.65 - 0.5711)^2 \\ &= \frac{1}{2} (0.08)^2 \\ &= 3.113 \times 10^{-3} \end{aligned}$$

Notice how the output is not the same as the expected value.

We use backpropagation to adjust the weights and biases so that the network's predictions get closer to the expected output, thereby reducing the error and improving accuracy.



# THANK YOU

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



**PES**  
UNIVERSITY

CELEBRATING 50 YEARS

# Machine Learning

## Gradient Descent

---

**Dr. Surabhi Narayan, Professor  
Department of Computer Science and  
Engineering**

# Machine Learning

## Acknowledgement

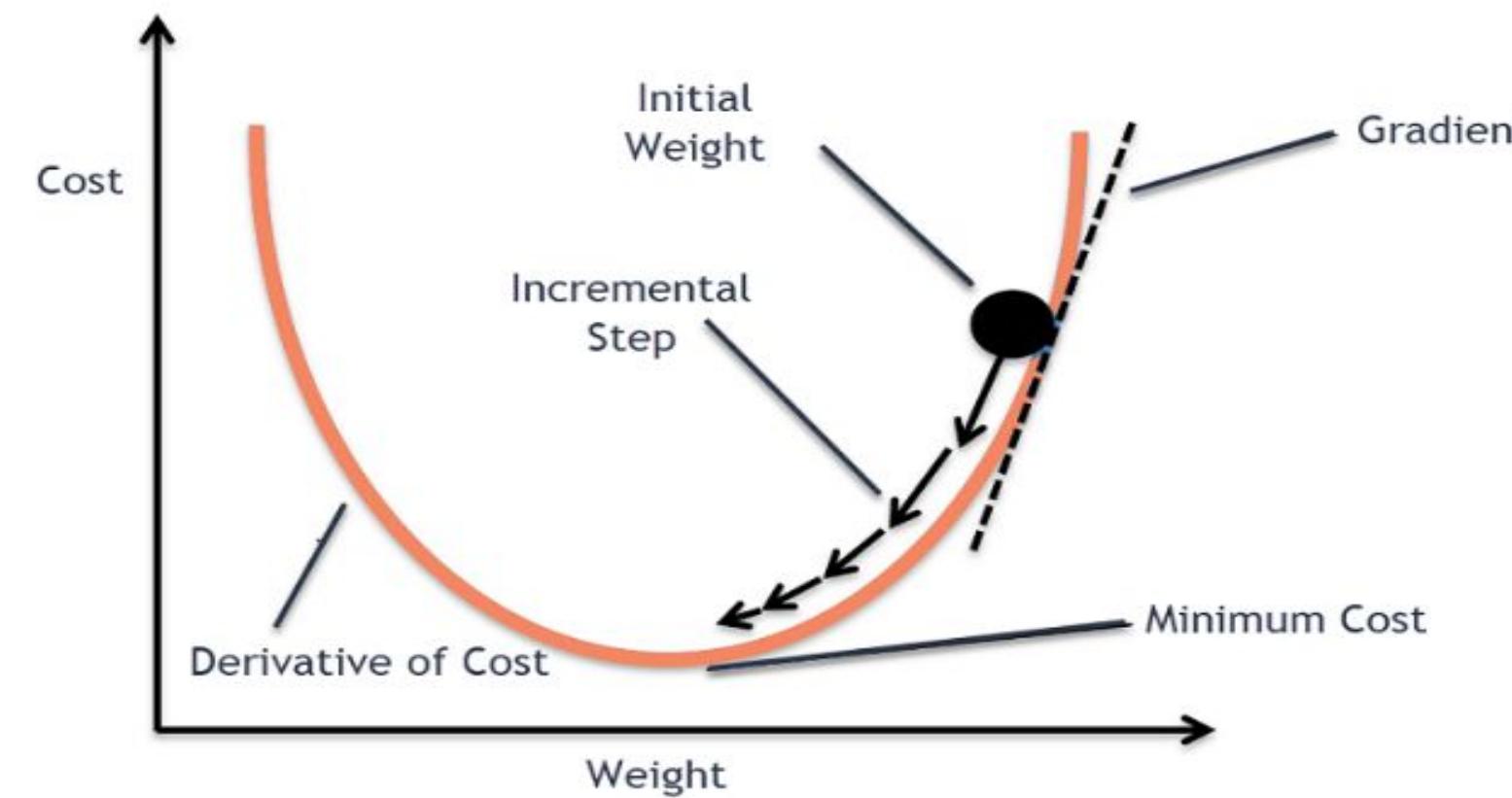
---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Gradient Descent

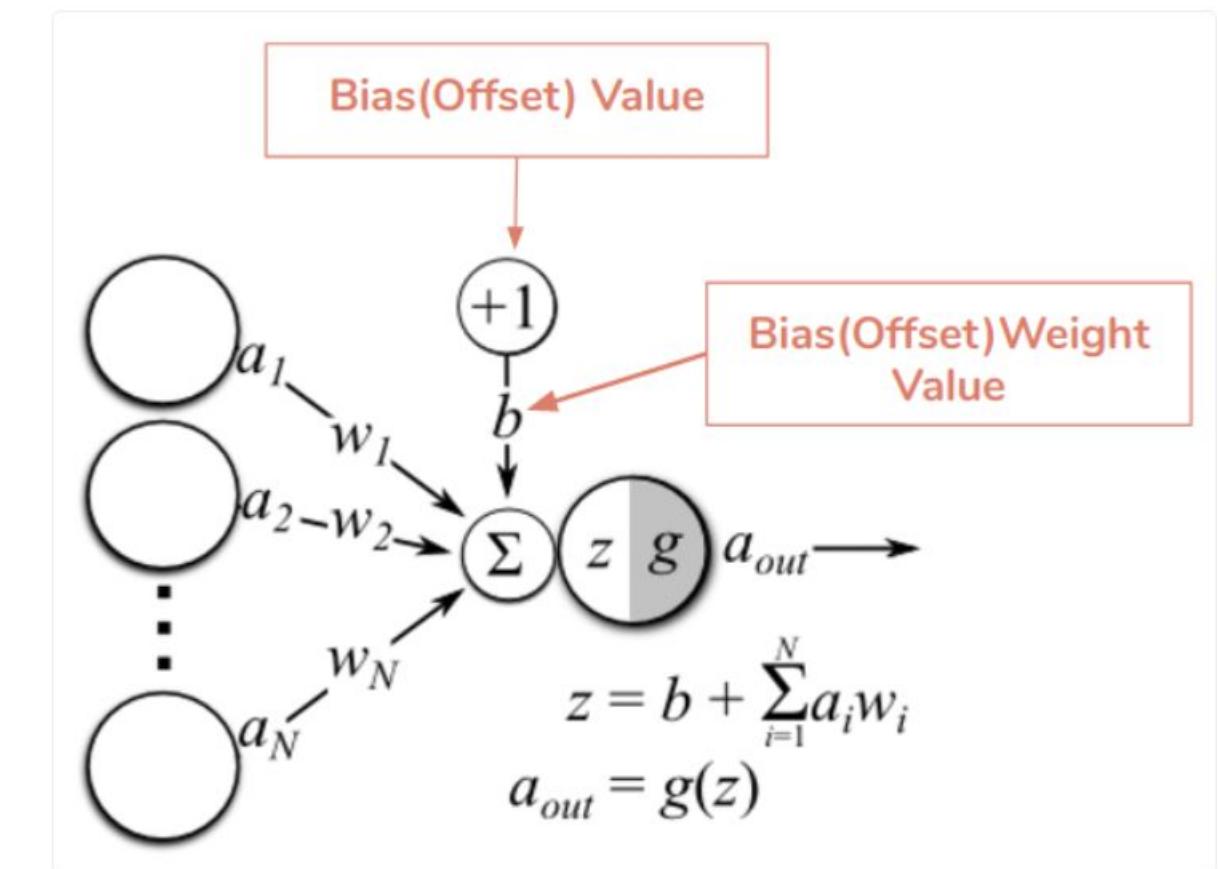
- Gradient descent (GD) is an iterative first-order optimization algorithm, used to find a local minimum/maximum of a given function.
- This method is commonly used in machine learning (ML) and deep learning (DL) to minimize a cost/loss function (e.g. in a linear regression).



# Learning Parameters

- In an ANN, each neuron in a layer is connected to some or all of the neurons in the next layer. When the inputs are transmitted between neurons, the weights are applied to the inputs along with the bias.
- **Weights** control the signal (or the strength of the connection) between two neurons. In other words, a weight decides how much influence the input will have on the output.
- **Biases**, which are constant, are an additional input into the next layer that will always have the value of 1. Bias units are not influenced by the previous layer (they do not have any incoming connections) but they do have outgoing connections with their own weights. The bias unit guarantees that even when all the inputs are zeros there will still be an activation in the neuron

$$o = \sum (\text{weight} * \text{input}) + \text{bias}$$



# Cost Function

- Also known as **error function**, it is a function that measures the performance of a model for any given data. Cost function quantifies the error between predicted values and expected values and presents it in the form of a single real number.
- After making a hypothesis with initial parameters, we calculate the Cost function. And with a goal to reduce the cost function, we modify the parameters by using the Gradient descent algorithm over the given data. Here's the mathematical representation for it:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

## Notations

---

- Output (prediction):

$$o(x_1, \dots, x_n) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

is represented as  $o(\vec{x}) = \vec{w} \cdot \vec{x}$

where the weights and biases are represented as a single vector  $\vec{w}$

- The training error is represented as  $E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$

where t is the target, and o is the prediction

# Gradient Descent

How can we calculate the direction of steepest descent along the error surface? This direction can be found by computing the derivative of  $E$  with respect to each component of the vector  $\vec{w}$ . This vector derivative is called the *gradient* of  $E$  with respect to  $\vec{w}$ , written  $\nabla E(\vec{w})$ .

$$\nabla E(\vec{w}) = \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right] \quad (4.3)$$

# Gradient Descent

Notice  $\nabla E(\vec{w})$  is itself a vector, whose components are the partial derivatives of  $E$  with respect to each of the  $w_i$ . *When interpreted as a vector in weight space, the gradient specifies the direction that produces the steepest increase in  $E$ .* The negative of this vector therefore gives the direction of steepest decrease. For example, the arrow in Figure 4.4 shows the negated gradient  $-\nabla E(\vec{w})$  for a particular point in the  $w_0, w_1$  plane.

# Gradient Descent

Since the gradient specifies the direction of steepest increase of  $E$ , the training rule for gradient descent is

$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

where

$$\Delta\vec{w} = -\eta \nabla E(\vec{w}) \quad (4.4)$$

$\eta$  is a positive constant called the learning rate. It is the step size in the gradient descent search.

There is a negative sign as we want to move in the direction of decreasing loss ( $E$ ).

# Gradient Descent

So, each weight  $w_i$  should be modified based on its gradient as

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

# Gradient Descent

$\frac{\partial E}{\partial w_i}$  n be derived in the following manner:

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d) \\ \frac{\partial E}{\partial w_i} &= \sum_{d \in D} (t_d - o_d) (-x_{id})\end{aligned}$$

# **Gradient Descent**

Therefore, the weight update rule for gradient descent is

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

# Gradient Descent Algorithm

## GRADIENT-DESCENT(*training-examples*, $\eta$ )

*Each training example is a pair of the form  $\langle \vec{x}, t \rangle$ , where  $\vec{x}$  is the vector of input values, and  $t$  is the target output value.  $\eta$  is the learning rate (e.g., .05).*

- Initialize each  $w_i$  to some small random value
- Until the termination condition is met, Do
  - Initialize each  $\Delta w_i$  to zero.
  - For each  $\langle \vec{x}, t \rangle$  in *training-examples*, Do
    - Input the instance  $\vec{x}$  to the unit and compute the output  $o$
    - For each linear unit weight  $w_i$ , Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i \quad (\text{T4.1})$$

- For each linear unit weight  $w_i$ , Do

$$w_i \leftarrow w_i + \Delta w_i \quad (\text{T4.2})$$



**PES**  
UNIVERSITY

CELEBRATING 50 YEARS

# THANK YOU

---

**Dr. Surabhi Narayan, Professor  
Department of Computer Science and  
Engineering surabhinarayan@pes.edu**



# Machine Learning

## Activation Functions

---

**Dr. Surabhi Narayan, Professor  
Department of Computer Science and Engineering**

**Teaching Assistant: Nettem Gayathri (Semester VII)**

# Machine Learning

## Acknowledgement

---



- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

# Activation Functions

- In a neural network, individual neurons function with designated weights, biases, and corresponding activation functions.
- The activation function plays a critical role by introducing non-linearity into the output of a neuron, determining whether a neuron should be activated by calculating the weighted sum and adding bias.

$$Y = \text{Activation}(\sum \text{weight} * \text{input}) + \text{bias}$$

- This non-linear transformation is crucial for enabling the network to learn complex patterns from data, which would not be possible if only linear transformations were used. Without activation functions, the neural network would essentially operate as a simple linear regression model, limiting its ability to solve complex problems.
- These functions can be linear or non-linear, depending on their mathematical form, and are vital for determining the neural network's outputs across various domains.
- During the neural network's operation, adjustments to neuron weights and biases are made based on discrepancies observed at the output layer, a process known as back-propagation. Activation functions are essential for making back-propagation feasible by facilitating the propagation of error signals. These gradients, produced by the activation functions, are pivotal in updating weights and biases, thus refining the network's overall performance.

# Linear Activation Function

- Also known as a straight-line function, where the activation is directly proportional to the input, meaning the output is simply the weighted sum from the neurons.
- The function is represented by the equation:  $f(x)=ax+c$ .
- This function has no specific range and can take values from  $-\infty$  to  $+\infty$ .
- Using this function in all nodes causes the neural network to behave like a linear regression model, with the final layer being a linear function of the first layer.
- Drawback: During backpropagation, the function's constant derivative leads to a fixed rate of error change, which can disrupt the backpropagation process and hinder learning.

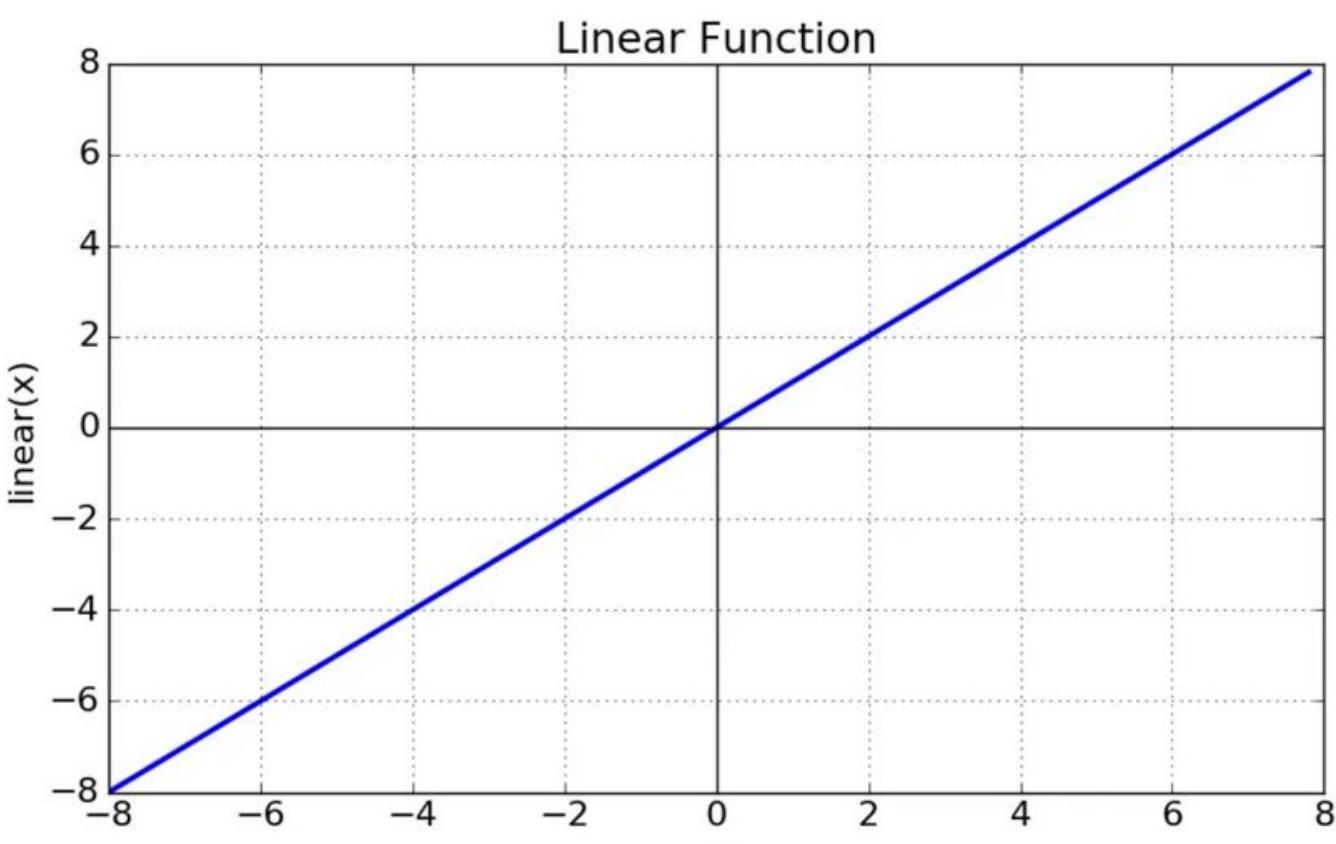


Fig: Linear Activation Function

# Non-Linear Activation Functions

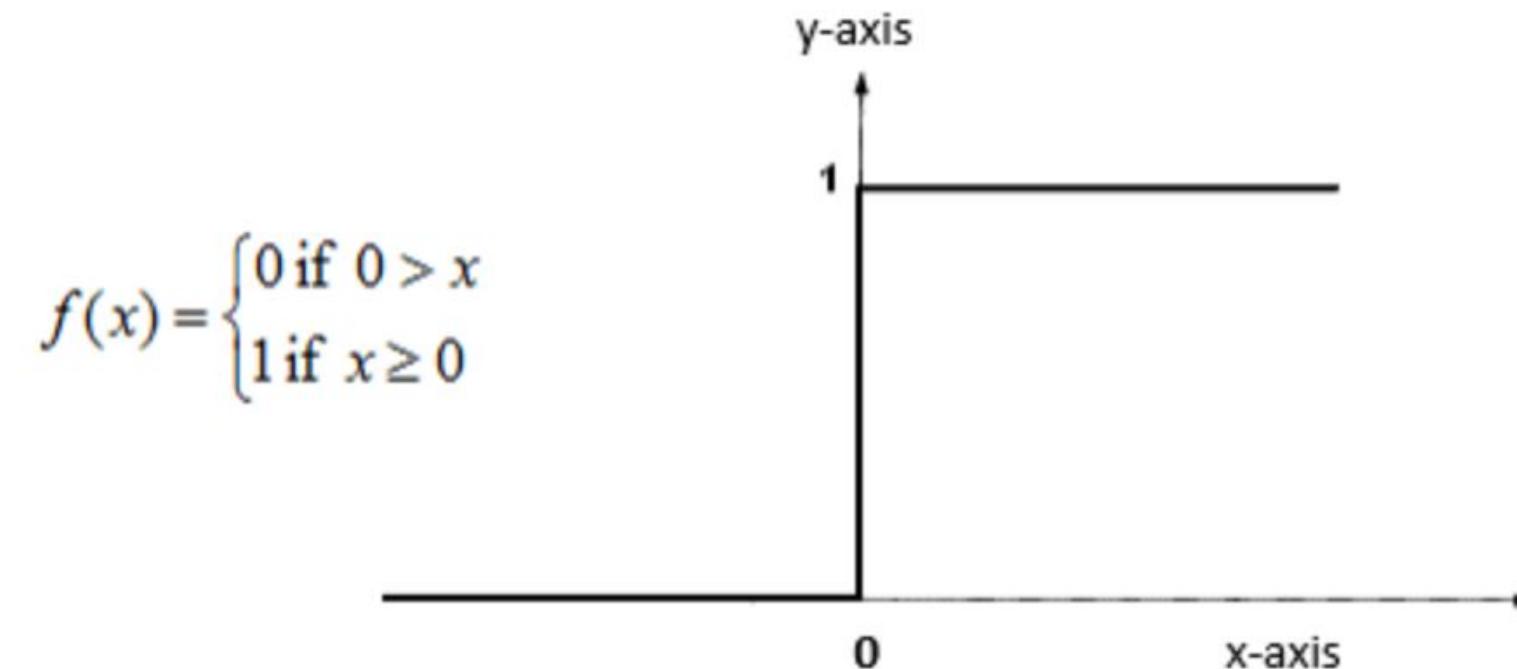
- These are the most commonly used activation functions in neural networks.
- They enable the model to adapt to a wide variety of data and make it easier to distinguish between different outcomes.
- Non-linear activation functions are categorized based on their range or the shape of their curve, including:
  - **Sigmoid (Logistic):** Maps input values to a range between 0 and 1, making it useful for probability-based predictions.
  - **Tanh (Hyperbolic Tangent):** Similar to the sigmoid but outputs values between  $-1$  and  $1$ , offering stronger gradients.
  - **ReLU (Rectified Linear Unit):** Outputs the input directly if positive, otherwise, it returns zero. It's widely used due to its simplicity and efficiency in deep networks.
  - **Leaky ReLU:** A variant of ReLU that allows a small, non-zero gradient when the input is negative, preventing neuron death.
  - **Softmax:** Typically used in the output layer for multi-class classification, it converts logits to probabilities.

# Step Function

- Step Function is one of the simplest kind of activation functions.
- In this, we consider a threshold value and if the value of net input say  $y$  is greater than the threshold then the neuron is activated. Mathematically,

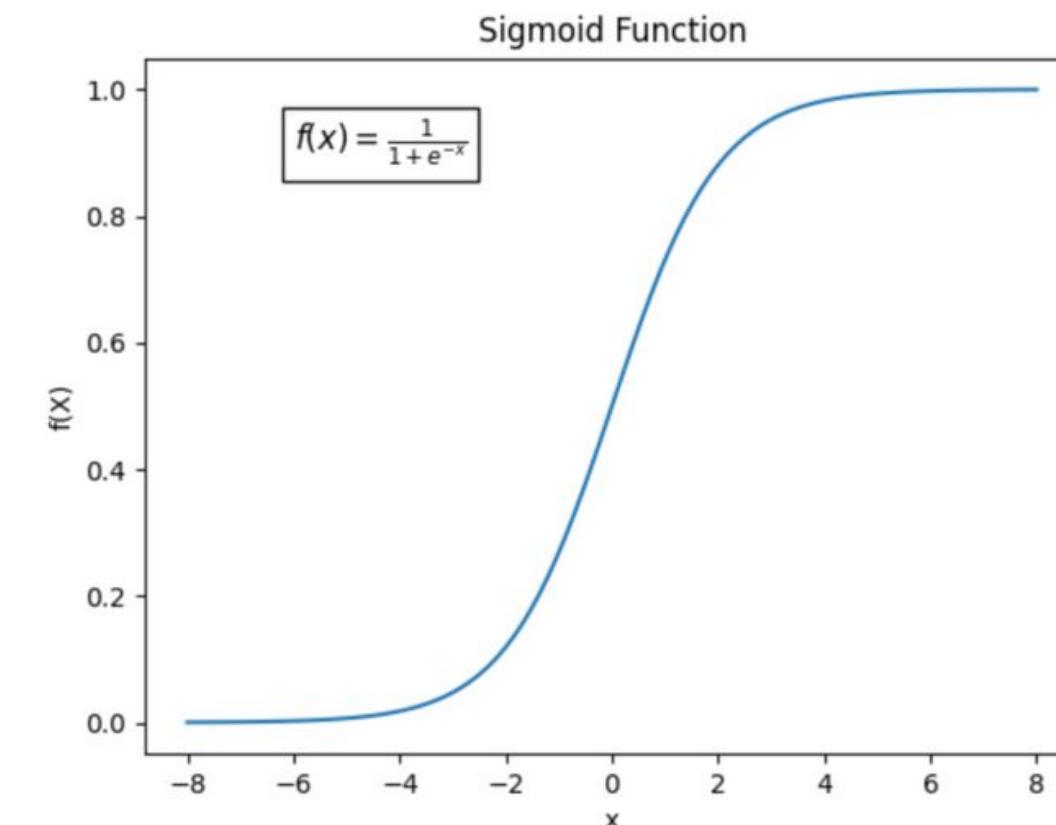
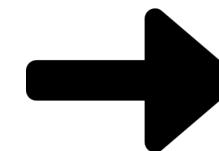
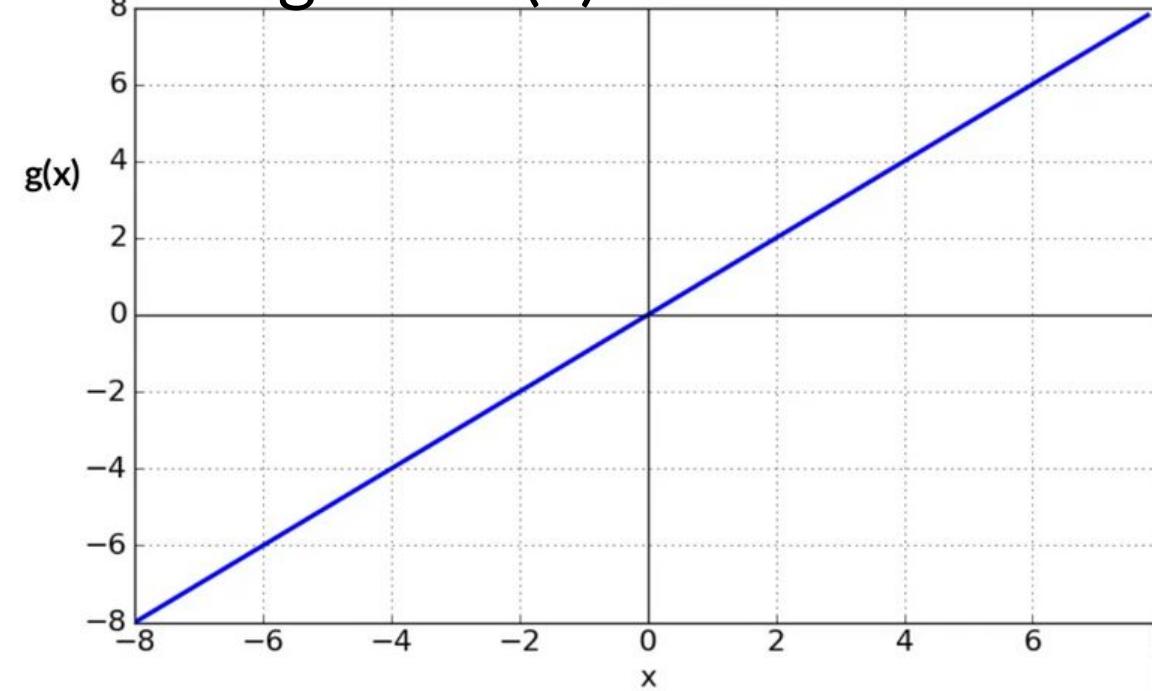
$$f(x) = 1, \text{if } x \geq 0$$

$$f(x) = 0, \text{if } x < 0$$



# Sigmoid Function

- The sigmoid function is one of the most commonly used activation functions.
- Formula :  $f(x) = \frac{1}{1+e^{-x}}$
- The output of a sigmoid function ranges between 0 and 1. Since output values bound between 0 and 1, it normalizes the output of each neuron. (Range : [0,1])
- Let  $g(x)$  be linear function i.e  $g(x) = x$
- Then the sigmoid  $f(x)$  does the following transformation



# Derivative of Sigmoid

Let's denote the sigmoid function as  $\sigma(x) = \frac{1}{1 + e^{-x}}$ .

$$\begin{aligned}
 \frac{d}{dx} \sigma(x) &= \frac{d}{dx} \left[ \frac{1}{1 + e^{-x}} \right] \\
 &= \frac{d}{dx} (1 + e^{-x})^{-1} \\
 &= -(1 + e^{-x})^{-2} (-e^{-x}) \\
 &= \frac{e^{-x}}{(1 + e^{-x})^2} \\
 &= \frac{1}{1 + e^{-x}} \cdot \frac{e^{-x}}{1 + e^{-x}} \\
 &= \frac{1}{1 + e^{-x}} \cdot \frac{(1 + e^{-x}) - 1}{1 + e^{-x}} \\
 &= \frac{1}{1 + e^{-x}} \cdot \left( \frac{1 + e^{-x}}{1 + e^{-x}} - \frac{1}{1 + e^{-x}} \right) \\
 &= \frac{1}{1 + e^{-x}} \cdot \left( 1 - \frac{1}{1 + e^{-x}} \right) \\
 &= \sigma(x) \cdot (1 - \sigma(x))
 \end{aligned}$$

The derivative of the sigmoid is  $\frac{d}{dx} \sigma(x) = \sigma(x)(1 - \sigma(x))$ .

# Sigmoid Function

- The function is monotonic, which implies it is either entirely non-decreasing or non-increasing.
- Specially used for models where we have to predict the probability as an output. Since the probability of anything exists only between the range of 0 and 1.
- Smooth gradient, preventing “jumps” in output values.
- The function is differentiable. That means we can find the slope of the sigmoid curve at any two points.

## Disadvantages of Sigmoid Function

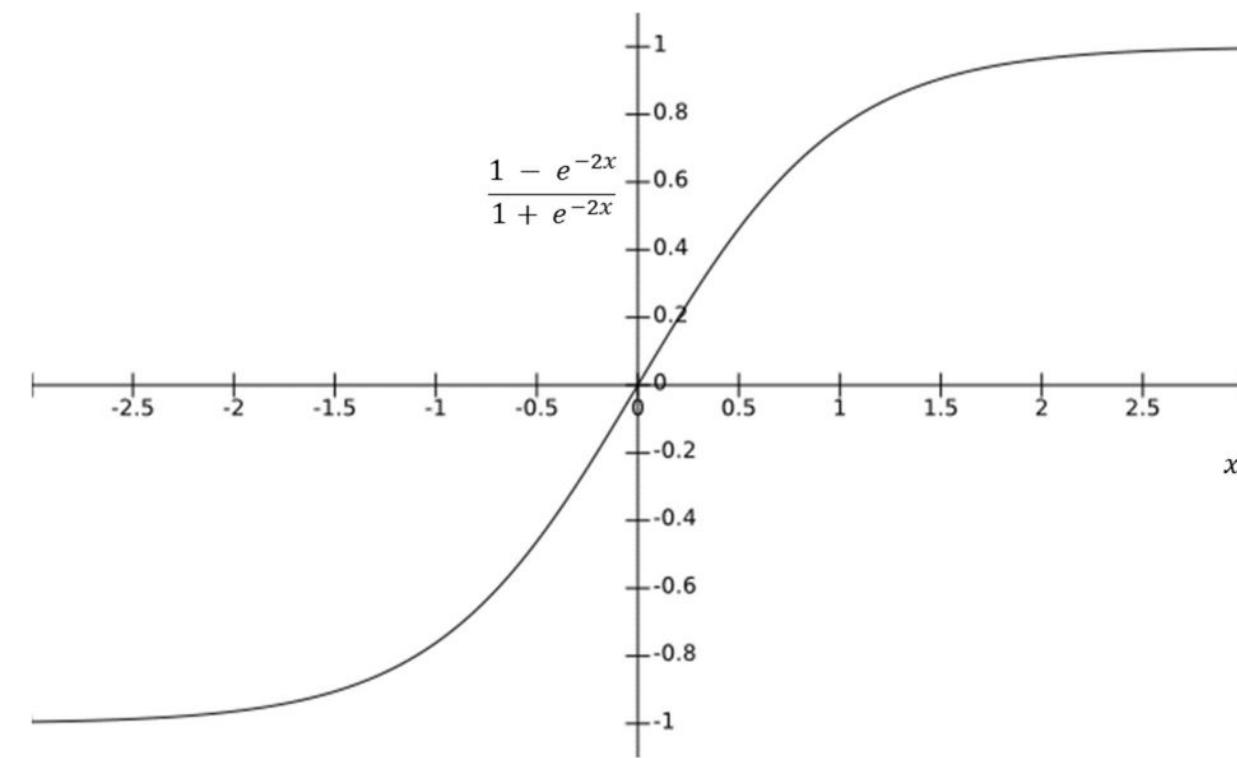
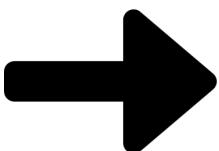
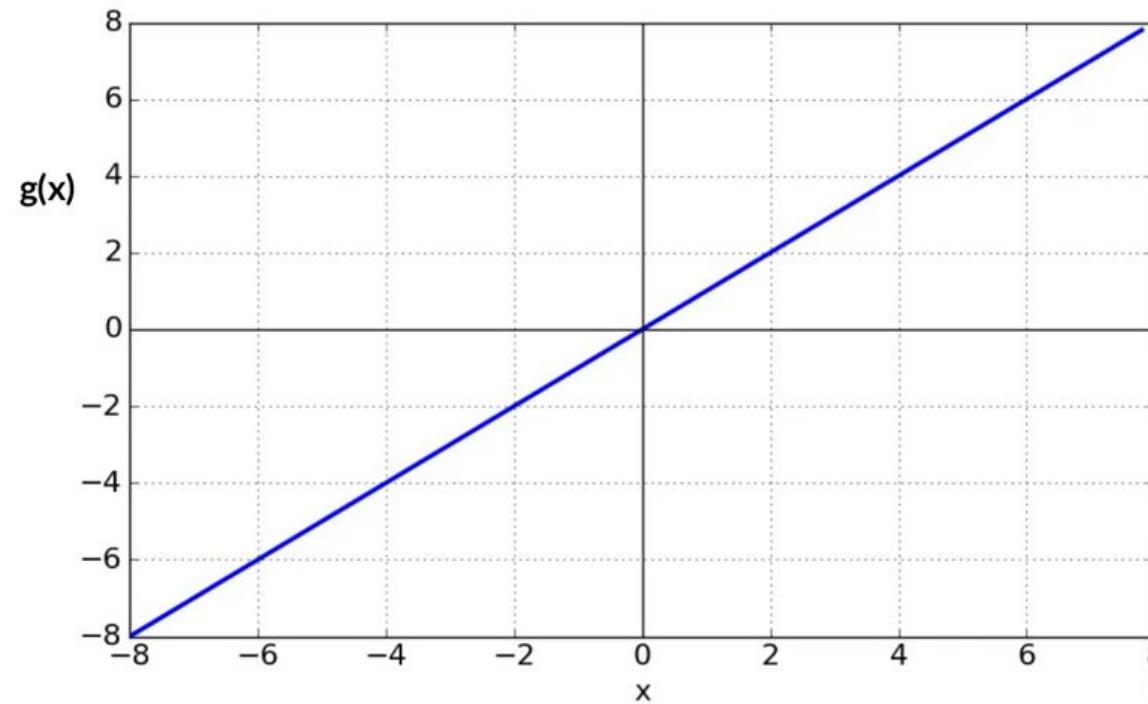
- Prone to gradient vanishing (when the sigmoid function value is either too high or too low, the derivative becomes very small i.e.  $<< 1$ . This causes vanishing gradients and poor learning for deep networks.)
- The function output is not centered on 0, which will reduce the efficiency of the weight update.
- The sigmoid function performs exponential operations, which is slower for computers.

# Tanh Function

- The hyperbolic tangent function outputs the value between -1 to +1 and is expressed as follows:

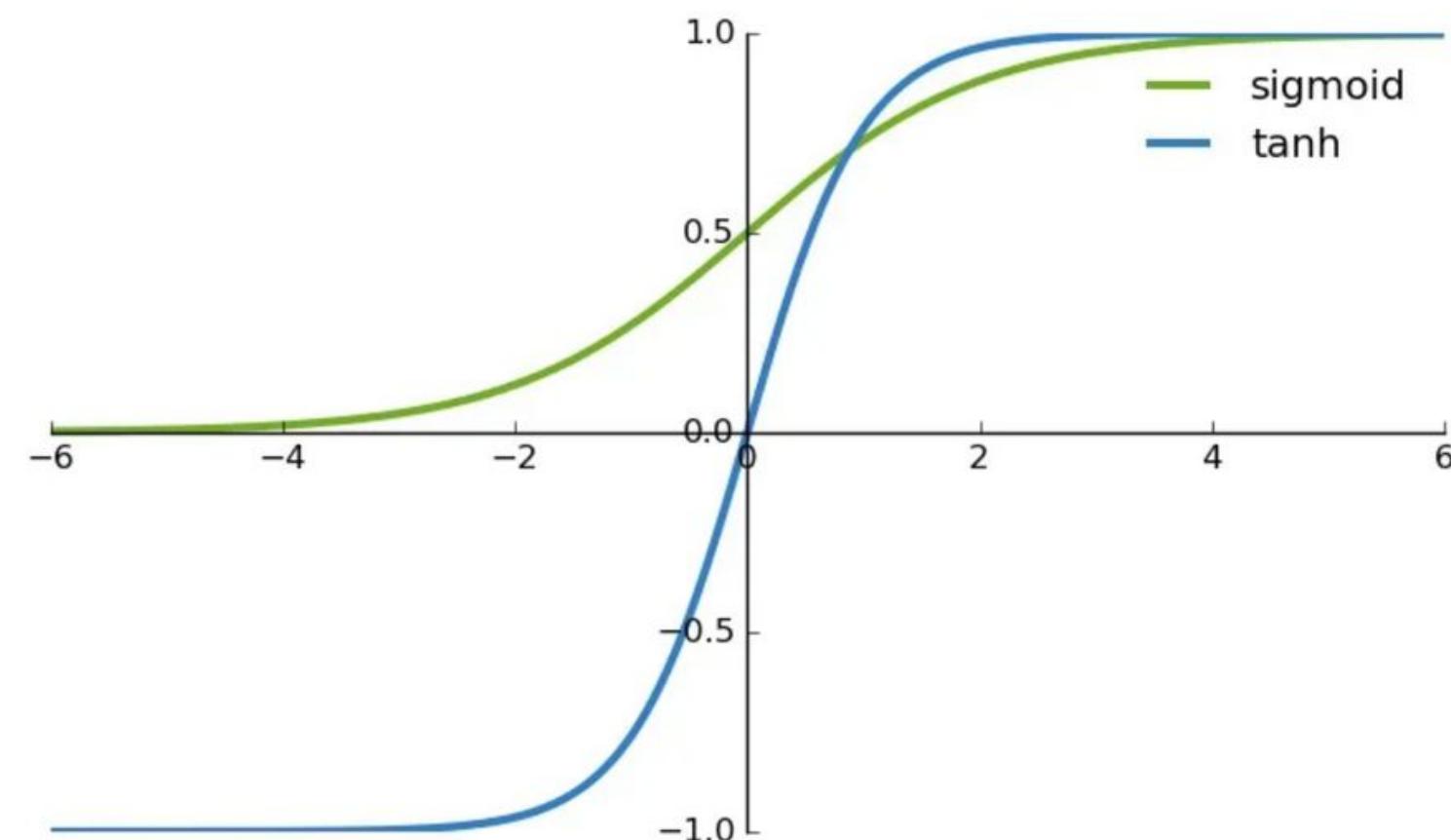
$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

- Let  $g(x)$  be linear function i.e  $g(x) = x$
- The tanh  $f(x)$  does the following transformation



# Tanh Function

- The tanh function is a differentiable function often used for classification tasks with two classes.
- It is similar to the sigmoid function but offers certain advantages.
- The output interval of tanh is 1, and the whole function is 0-centric, which is better than sigmoid.
- The major advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.



# Derivative of Tanh

---

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\frac{d}{dx} \tanh(x) = \frac{d}{dx} \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$= \frac{[(e^x + e^{-x}) * (e^x + e^{-x})] - [(e^x - e^{-x}) * (e^x - e^{-x})]}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$= 1 - \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right)^2$$

$$= 1 - \tanh^2(x)$$

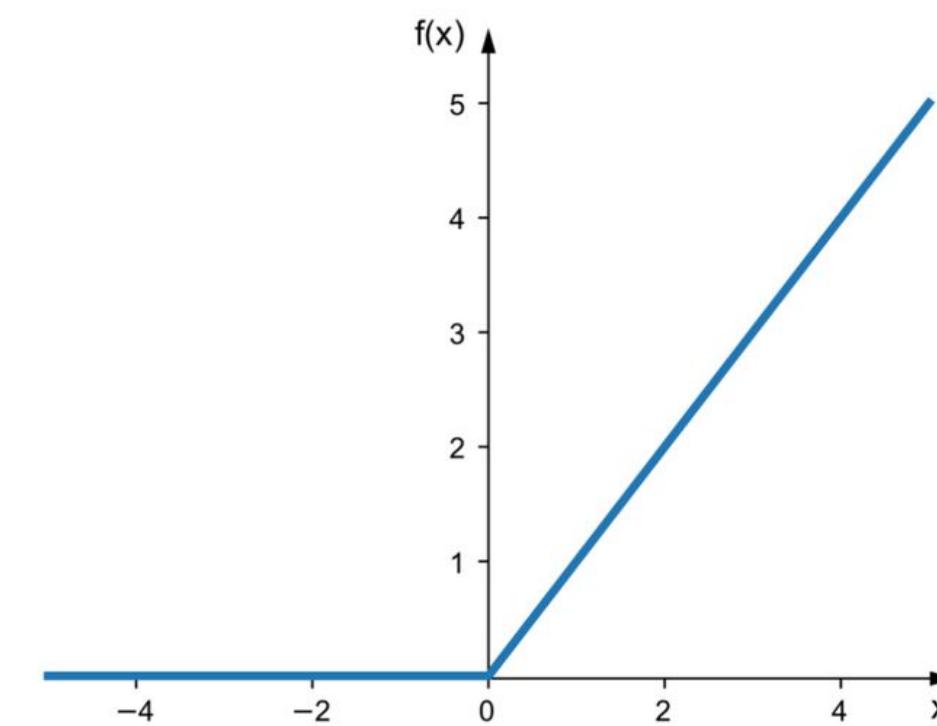
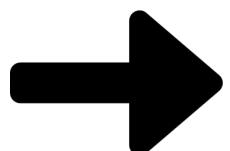
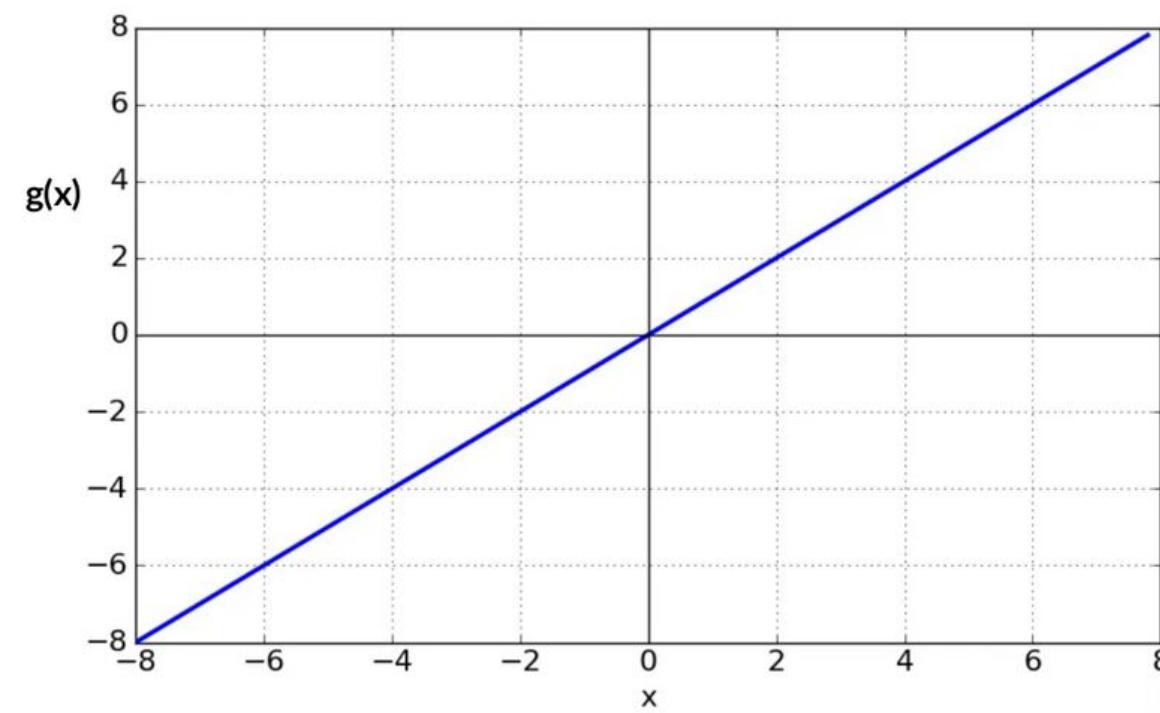
The derivative of tanh is  $\frac{d}{dx} \tanh(x) = 1 - \tanh^2(x)$

## ReLU Function

- The Rectified Linear Unit function is another one of the most commonly used activation functions.
- The ReLU is half rectified (from the bottom).  $f(x)$  is zero when  $x$  is less than zero and  $f(x)$  is equal to  $x$  when  $x$  is above or equal to zero.

$$\sigma(x) = \begin{cases} \max(0, x) & , x \geq 0 \\ 0 & , x < 0 \end{cases}$$

- Range:  $[0, \infty)$
- let  $g(x)$  be some function
- the ReLU function  $f(x)$  does the following transformation



# Dying ReLU

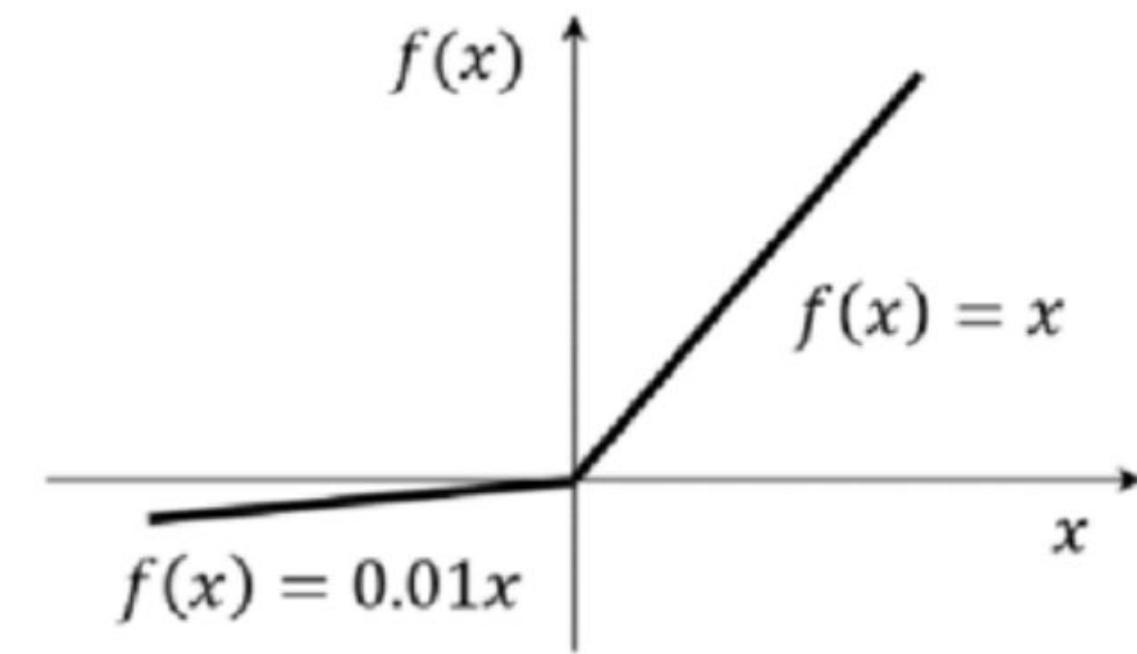
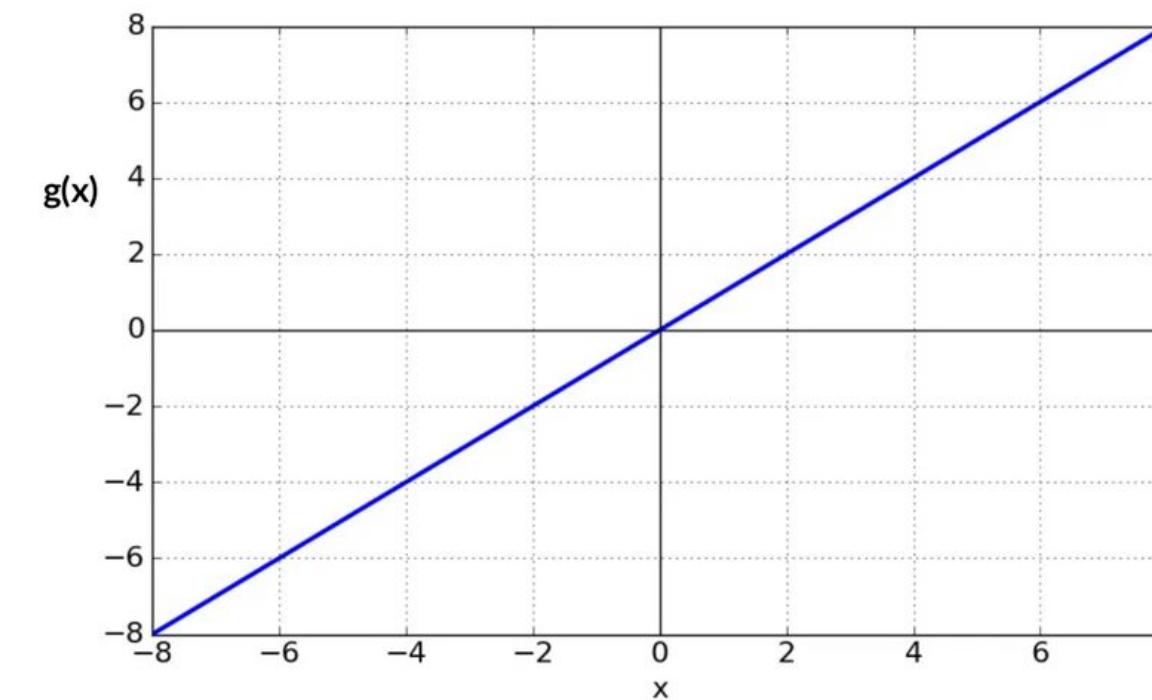
- When the input is negative, ReLU is completely inactive, which means that once a negative number is entered, ReLU will die.
- In the forward propagation process, it is not a problem. Some areas are sensitive and some are insensitive. But in the back propagation process, if the input is a negative number, the gradient will be completely zero.
- The snag for being zero for all negative values is a problem called dying RELU, and a neuron is said to be dead if it always outputs zero.

# Leaky ReLU Function

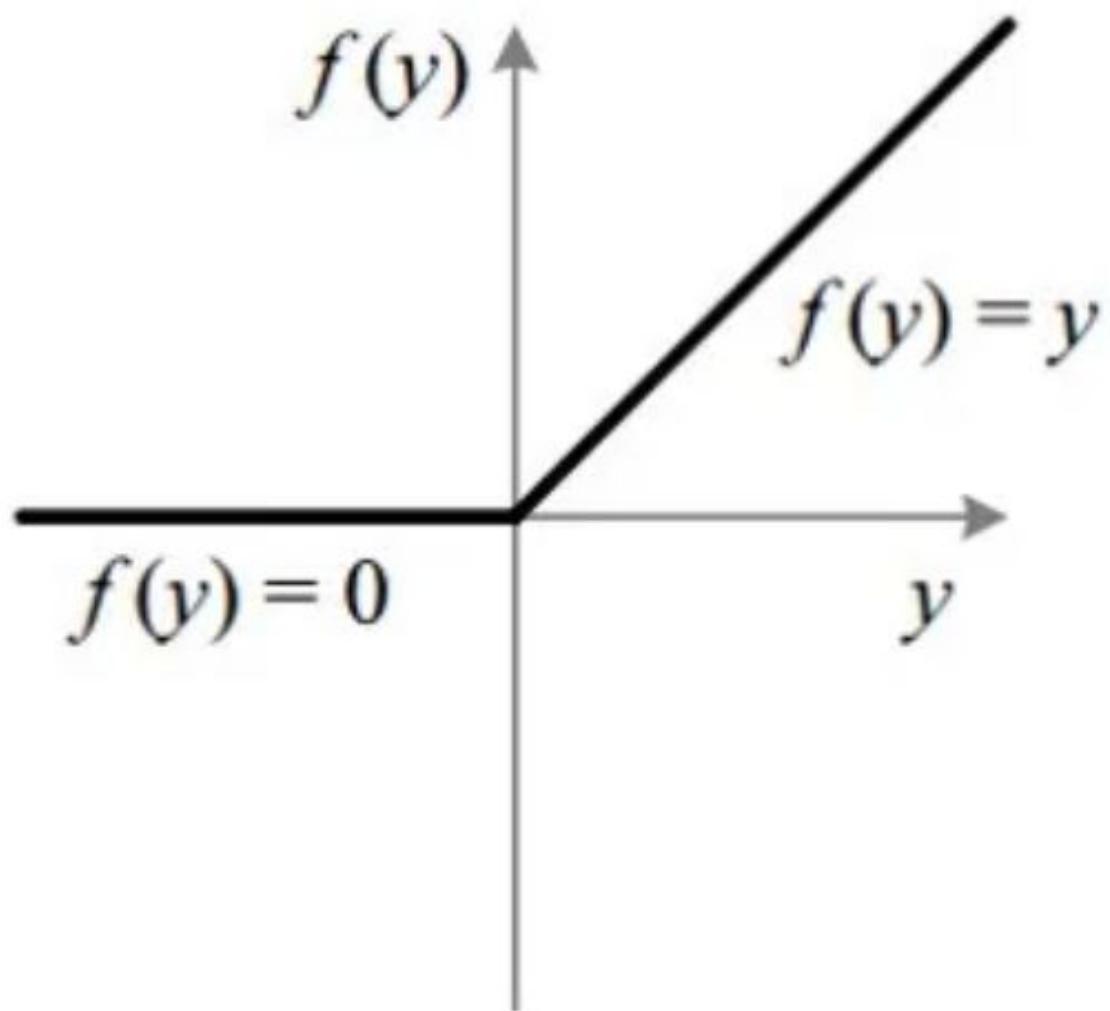
- The Leaky Rectified Linear is a variant of the ReLU that solves the dying ReLU problem.

$$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

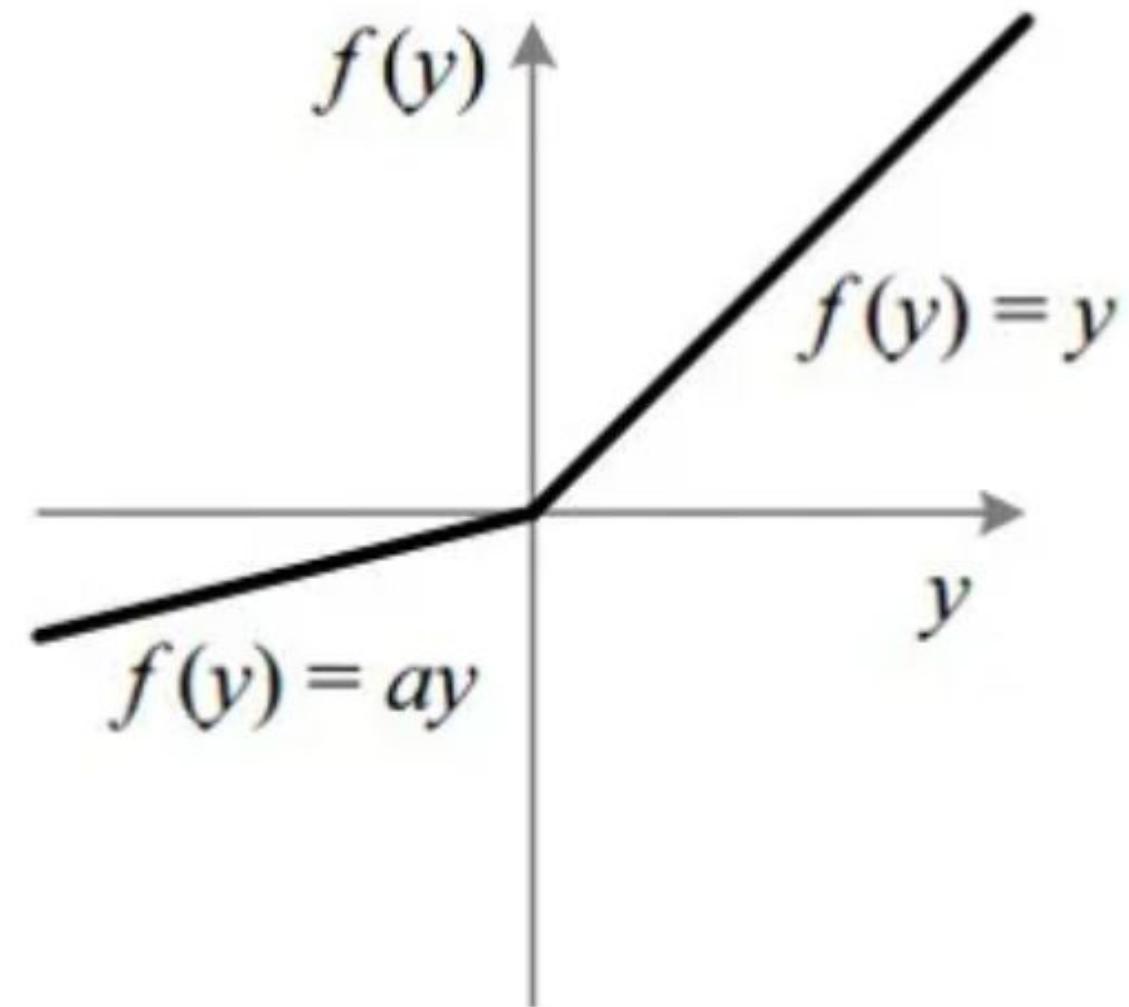
- The leaky ReLU adjusts the problem of zero gradients for negative value, by giving a very small linear component of  $x$  to negative inputs ( $0.01x$ ).
- The value of  $\alpha$  is typically set to some low value say 0.01.
- Instead of setting some default value to  $\alpha$ , pass them as a parameter to a neural network and make the network learn the optimal value for it.
- The range of the Leaky ReLU is  $(-\infty, \infty)$ .
- let  $g(x)$  be some function the Leaky ReLU function



# ReLU VS Leaky ReLU



ReLU

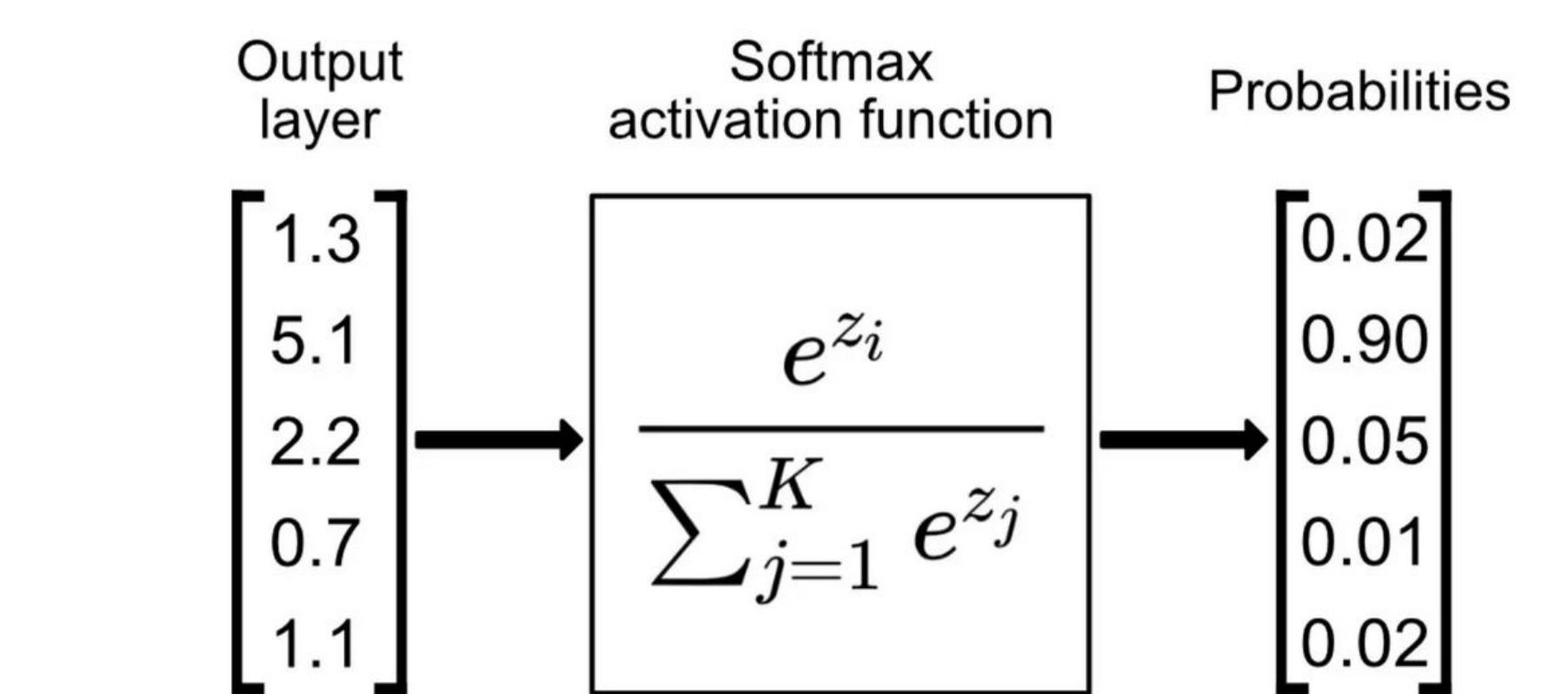
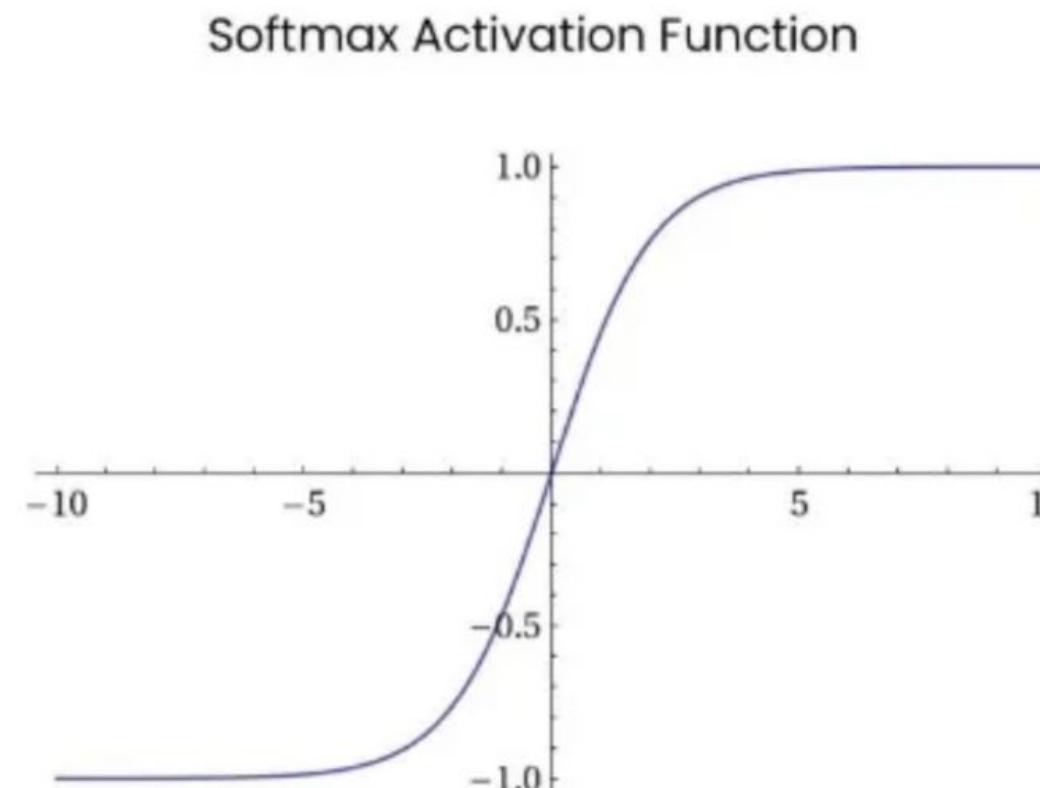


Leaky ReLU

# Softmax Function

- It is a generalization of the sigmoid function i.e. when the number of classes is two, it becomes the same as the sigmoid function.
- Used in the final layer of an ANN while performing multi-class classification tasks.
- The Softmax activation function calculates the relative probabilities of each class for being the output.
- Thus, the sum of the softmax values will always be equal to 1.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



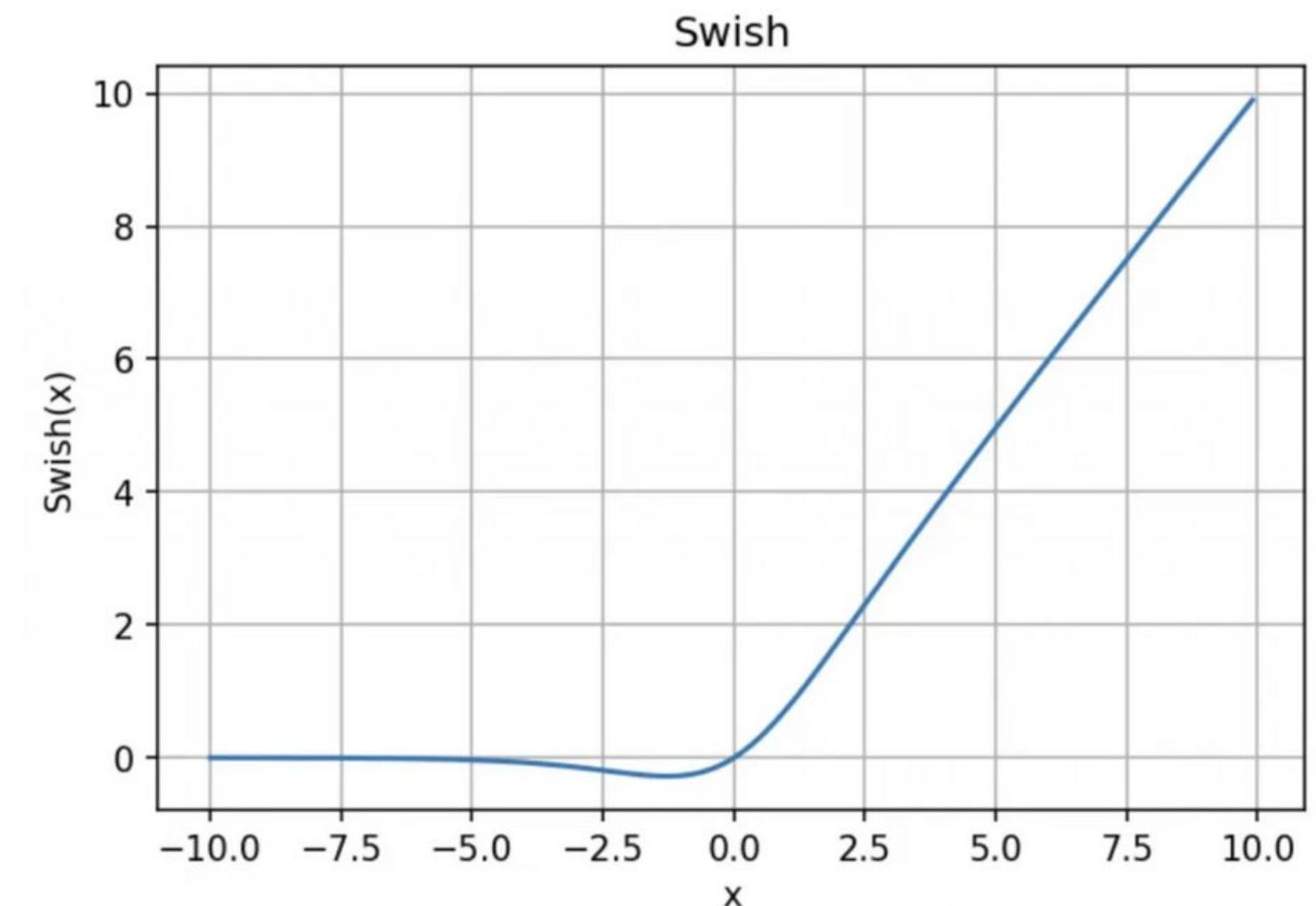
$z$  represents the values from the neurons of the output layer.

# Swish Function

- Swish's design was inspired by the use of sigmoid functions for gating in LSTMs and highway networks. We use the same value for gating to simplify the gating mechanism, which is called self-gating.
- It has a range of  $(-\infty, \infty)$ , is differentiable, introduces non-linearity, and produces a smooth output by retaining the input and applying a sigmoid.

$$\text{Swish}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}$$

- $\text{Swish}(x)$ : Swish function output
- $x$ : Input to the function
- $\sigma(x)$ : Sigmoid function
- $e$ : Base of the natural logarithm



## Swish Function

- The Swish function is perfect for hidden layers in neural networks due to its ability to preserve information flow and promote smoother gradients, leading to improved training performance and convergence.
- Swish is increasingly used in deep learning models where traditional activation functions like ReLU may fall short, providing better accuracy and efficiency in complex tasks.
- However, while Swish addresses some limitations of older activation functions like the vanishing gradient problem and output saturation, its computational complexity can be higher due to the inclusion of the sigmoid component. Despite this, the balanced performance of Swish makes it a strong candidate in modern neural network architectures.

## Which activation functions to use?

---

It's challenging to recommend a single activation function that works universally for all scenarios. There is no "one size fits all" solution.

- Calculus can slow down computations, so it's essential to evaluate how difficult it is to compute the derivative, assuming it is even differentiable.
- Consider how quickly the network converges with the chosen activation function.
- Assess the smoothness of the function.
- Determine whether the function satisfies the conditions of the Universal Approximation Theorem.

## Which activation functions to use?

Given the complexity of factors, it's impractical to weigh all of them before choosing an activation function.

Instead, we typically follow general guidelines:

- Sigmoid: Best suited for binary classification problems and remains popular in networks with a single hidden layer.
- ReLU: Widely used in deep neural networks due to its effectiveness in yielding better results.
- Leaky ReLU: Ideal when dealing with dead neurons in the network, as it allows a small gradient for negative inputs.
- ReLU Usage: This should be applied only in hidden layers, as it sets all negative values to zero.



# THANK YOU

---

**Dr. Surabhi Narayan, Professor**  
**Department of Computer Science and Engineering**  
**[surabhinarayan@pes.edu](mailto:surabhinarayan@pes.edu)**



# Machine Learning

## Backward Propagation

---

**Dr. Surabhi Narayan, Professor  
Department of Computer Science and  
Engineering**

**Teaching Assistants - Aneesh K B (7th semester)**

# What is a backpropagation algorithm?

---

- A backpropagation algorithm, or backward propagation of errors, is an algorithm that's used to help train neural network models. The algorithm adjusts the network's weights to minimize any gaps -- referred to as errors -- between predicted outputs and the actual target output.
- Weights are adjustable parameters that determine the strength of the connections between artificial neurons -- also referred to as nodes -- in different layers of a neural network. Specifically, weights determine how much influence the output of one neuron has on the input of the next neuron, which can directly influence the network's output and performance

# Objective of BackPropagation Algorithm

---

- Backpropagation algorithms are used extensively to train feedforward neural networks, such as convolutional neural networks, in areas such as deep learning.
- A backpropagation algorithm is pragmatic because it computes the gradient needed to adjust a network's weights more efficiently than computing the gradient based on each individual weight.
- It enables the use of gradient methods, such as gradient descent and stochastic gradient descent, to train multilayer networks and update weights to minimize errors.

# Backward Propagation

---

# Backward Propagation

## Understanding weight update with a Toy example

# Backward Propagation

- Let us understand a few symbols before we get into the explanation
- Neuron # in Output layer (k):** Represents the index or number of the neuron in the output layer, denoted by  $k$ .
  - Target values at output neuron (t):** These are the target or desired values for the output neurons, represented by  $t$ .
  - Predicted values (O):** The values predicted by the model for the output neurons, denoted by  $O$ .
  - Error in prediction ( $t_k - O_k$ ):** This is the difference between the target value  $t_k$  and the predicted value  $O_k$  for each output neuron, representing the error in the prediction.

# Backward Propagation

→ Let us look at a few more symbols

- $x_i$  - Input value from  $i^{\text{th}}$  neuron in input layer
- $w_{ji}^k$  - weight b/w  $j^{\text{th}}$  neuron in  $(i+1)^{\text{th}}$  layer and  $i^{\text{th}}$  neuron in  $i^{\text{th}}$  layer.  
 $k$  is the level.

Weights are updated using gradient descent method

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \text{ where } \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$\eta$  = learning rate

$E_d$  = loss function

→ The -ve sign is due to moving downhill along the error surface i.e. to reduce error.

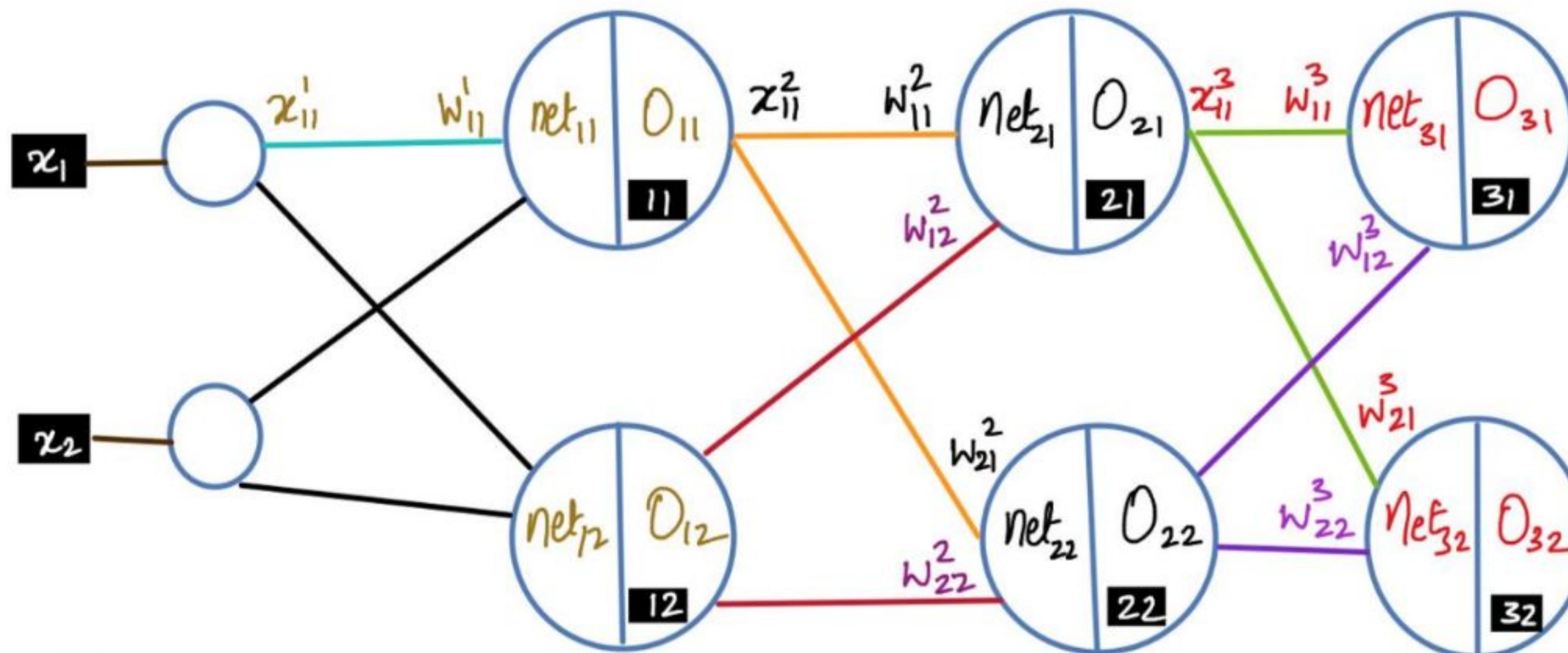
$$\text{net}_k = \sum_j w_{kj} x_{kj}$$

# Understanding Weight update using a Toy example

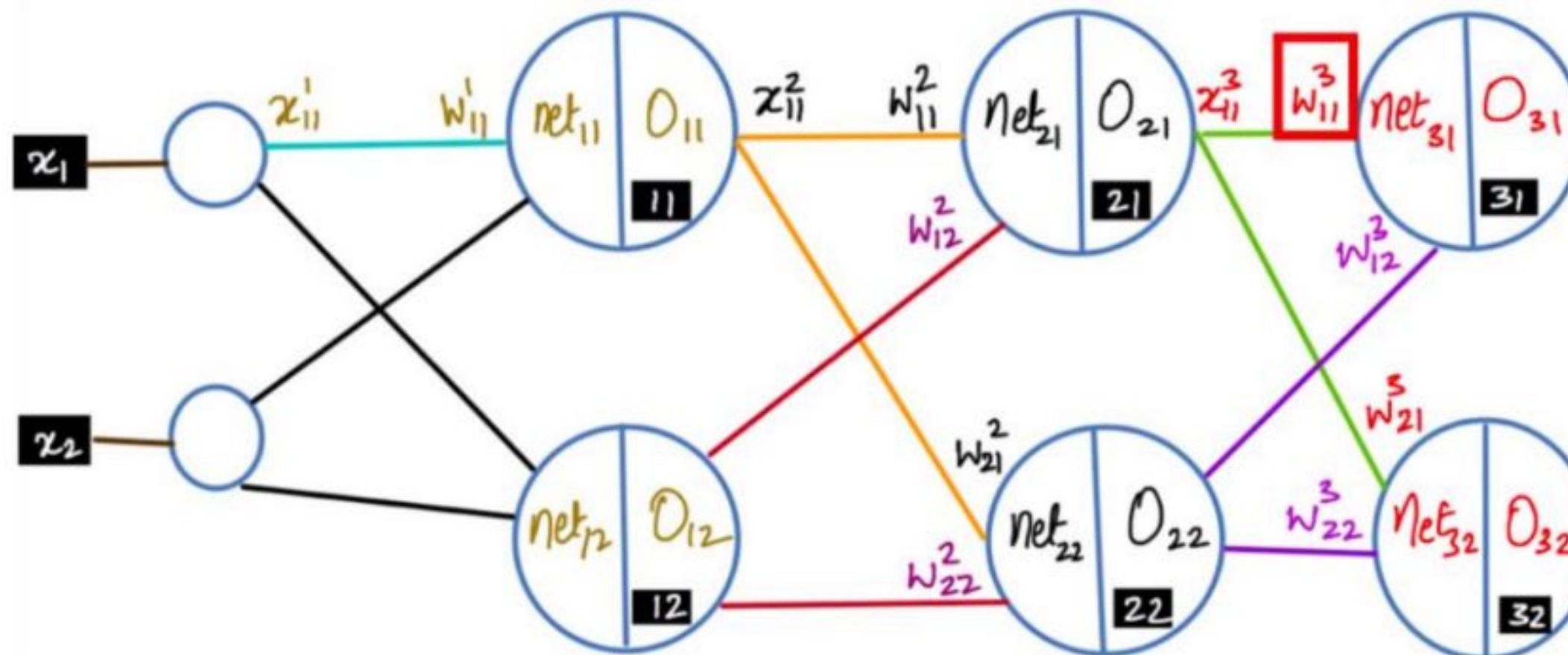
Weights are updated as using the gradient descent method.

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji} \quad \text{where}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$



# Updating Weight at Output layer



Now let us look at weight updation in output layer.

$$\omega_{11}^3 \leftarrow \omega_{31}^3 + \Delta \omega_{11}^3, \quad \Delta \omega_{11}^3 = -\eta \frac{\partial E_d}{\partial \omega_{11}^3}$$

$$\frac{\partial E_d}{\partial \omega_{11}^3} = \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial \omega_{11}^3} \quad \text{--- (1)}$$

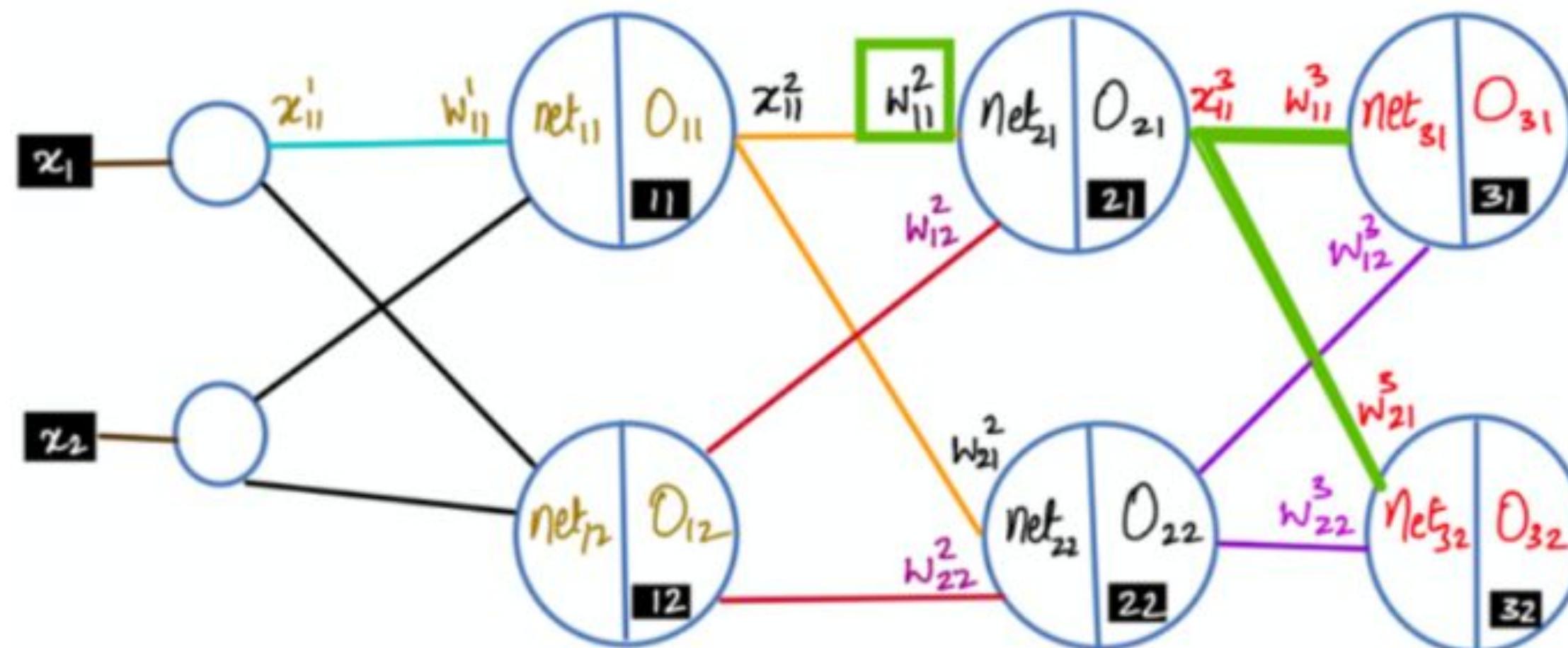
Now since  $O_{31} = f(\text{net}_{31})$ , by chain rule

$$\frac{\partial E_d}{\partial \text{net}_{31}} = \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial \text{net}_{31}} \quad \text{--- (2)}$$

From (1) & (2)

$$\frac{\partial E_d}{\partial \omega_{11}^3} = \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial \text{net}_{31}} \cdot \frac{\partial \text{net}_{31}}{\partial \omega_{11}^3}$$

# Updating Weight at Hidden layer 2



Since  $w_{11}^2$  is associated with both the neurons of the output layer.  
∴ The derivatives are added.

$$\frac{\partial E_d}{\partial w_{11}^2} = \left[ \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial net_{31}} \cdot \frac{\partial net_{31}}{\partial O_{21}} + \frac{\partial E_d}{\partial O_{32}} \cdot \frac{\partial O_{32}}{\partial net_{32}} \cdot \frac{\partial net_{32}}{\partial O_{21}} \right] * \left( \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_{11}^2} \right)$$

The update of  $w_{11}^2$  is given by  $w_{11}^2 = w_{11}^2 + \Delta w_{11}^2$

# Updating Weight at Hidden layer 2

$$\frac{\partial E_d}{\partial w_{11}^2} = \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w_{11}^2} + \frac{\partial E_d}{\partial O_{32}} \cdot \frac{\partial O_{32}}{\partial w_{11}^2}$$

Expanding  $\frac{\partial O_{31}}{\partial w_{11}^2}$  &  $\frac{\partial O_{32}}{\partial w_{11}^2}$  by chain rule

$$\frac{\partial O_{31}}{\partial w_{11}^2} = \frac{\partial O_{31}}{\partial net_{31}} \cdot \frac{\partial net_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_{11}^2}$$

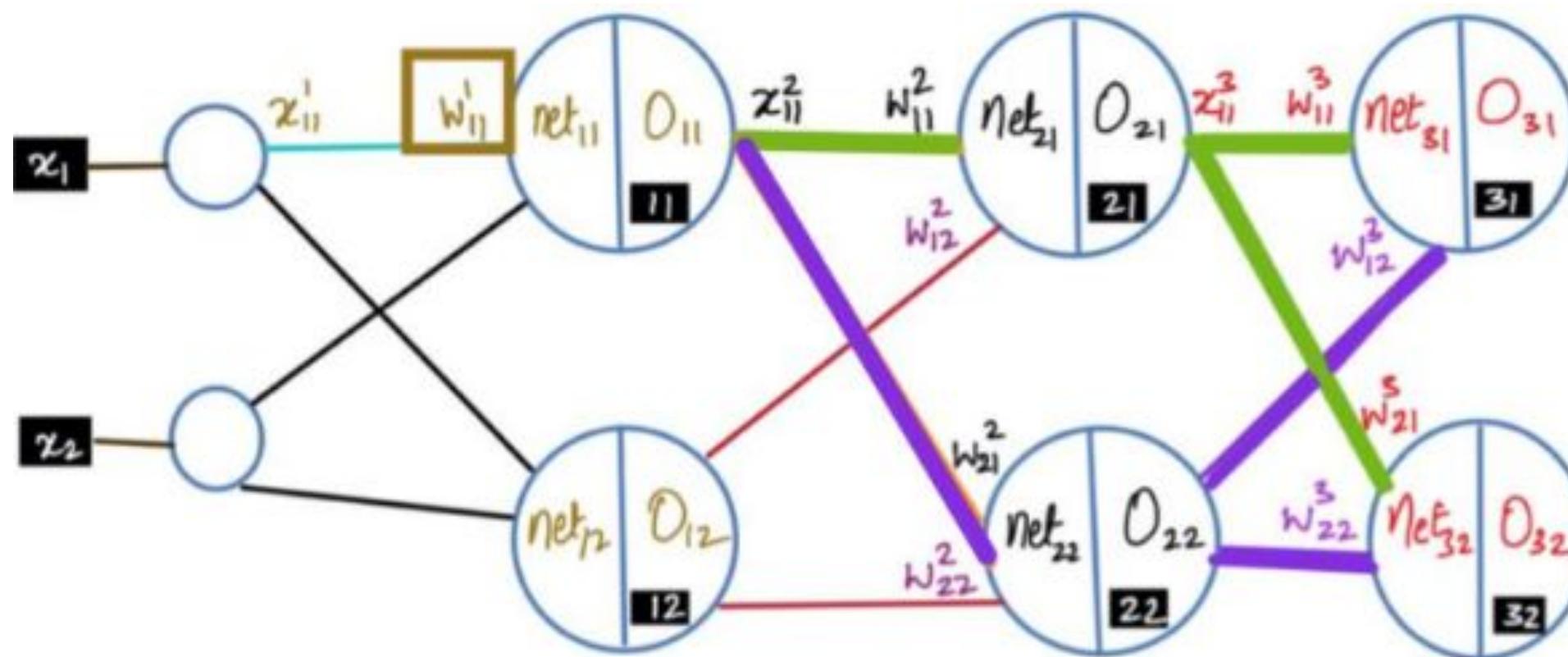
$$\frac{\partial O_{32}}{\partial w_{11}^2} = \frac{\partial O_{32}}{\partial net_{32}} \cdot \frac{\partial net_{32}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_{11}^2}$$

Now the final equation looks like

$$\frac{\partial E_d}{\partial w_{11}^2} = \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial net_{31}} \cdot \frac{\partial net_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_{11}^2}$$

$$+ \frac{\partial E_d}{\partial O_{32}} \cdot \frac{\partial O_{32}}{\partial net_{32}} \cdot \frac{\partial net_{32}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial w_{11}^2}$$

# Updating Weight at Hidden layer 1



$$\Delta w_{11} : -\eta \frac{\partial E_d}{\partial w_{11}}$$

$$w_{11} := w_{11} + \Delta w_{11}$$

$$\begin{aligned}
\frac{\partial E_d}{\partial w_{11}} &= \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial w_{11}} + \frac{\partial E_d}{\partial O_{32}} \cdot \frac{\partial O_{32}}{\partial w_{11}} \\
&= \frac{\partial E_d}{\partial O_{31}} \cdot \frac{\partial O_{31}}{\partial net_{31}} \left[ \frac{\partial net_{31}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial w_{11}} \right. \\
&\quad + \\
&\quad \left. \frac{\partial net_{31}}{\partial O_{22}} \cdot \frac{\partial O_{22}}{\partial net_{22}} \cdot \frac{\partial net_{22}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial w_{11}} \right] \\
&+ \\
\frac{\partial E_d}{\partial O_{32}} \cdot \frac{\partial O_{32}}{\partial net_{32}} \left[ \frac{\partial net_{32}}{\partial O_{21}} \cdot \frac{\partial O_{21}}{\partial net_{21}} \cdot \frac{\partial net_{21}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial w_{11}} \right. \\
&\quad + \\
&\quad \left. \frac{\partial net_{32}}{\partial O_{22}} \cdot \frac{\partial O_{22}}{\partial net_{22}} \cdot \frac{\partial net_{22}}{\partial O_{11}} \cdot \frac{\partial O_{11}}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial w_{11}} \right]
\end{aligned}$$

# Backward Propagation

---

## Backward Propagation Derivation Rules

# Updating weights to lower the error

Error computation for a single training instance considering error at all neurons in output layer.

$$\rightarrow E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

Weights are updated as using the gradient descent method.

$$\rightarrow w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

Notice the weight can influence the rest of the network only through  $net_j$ . Hence, we use the chain rule :



$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

# Updating weights to lower the error

we know  $net_j = \sum_i w_{ji}x_{ji}$  (the weighted sum of inputs for unit j )

$$\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$$

→  $\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$

→  $\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} x_{ji}$

→  $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$

All we need to do now is derive a convenient expression for the term

$$\frac{\partial E_d}{\partial net_j}$$

# Derivation of Back propagation Rule

To derive a convenient expression  
for the term :

$$\frac{\partial E_d}{\partial \text{net}_j}$$

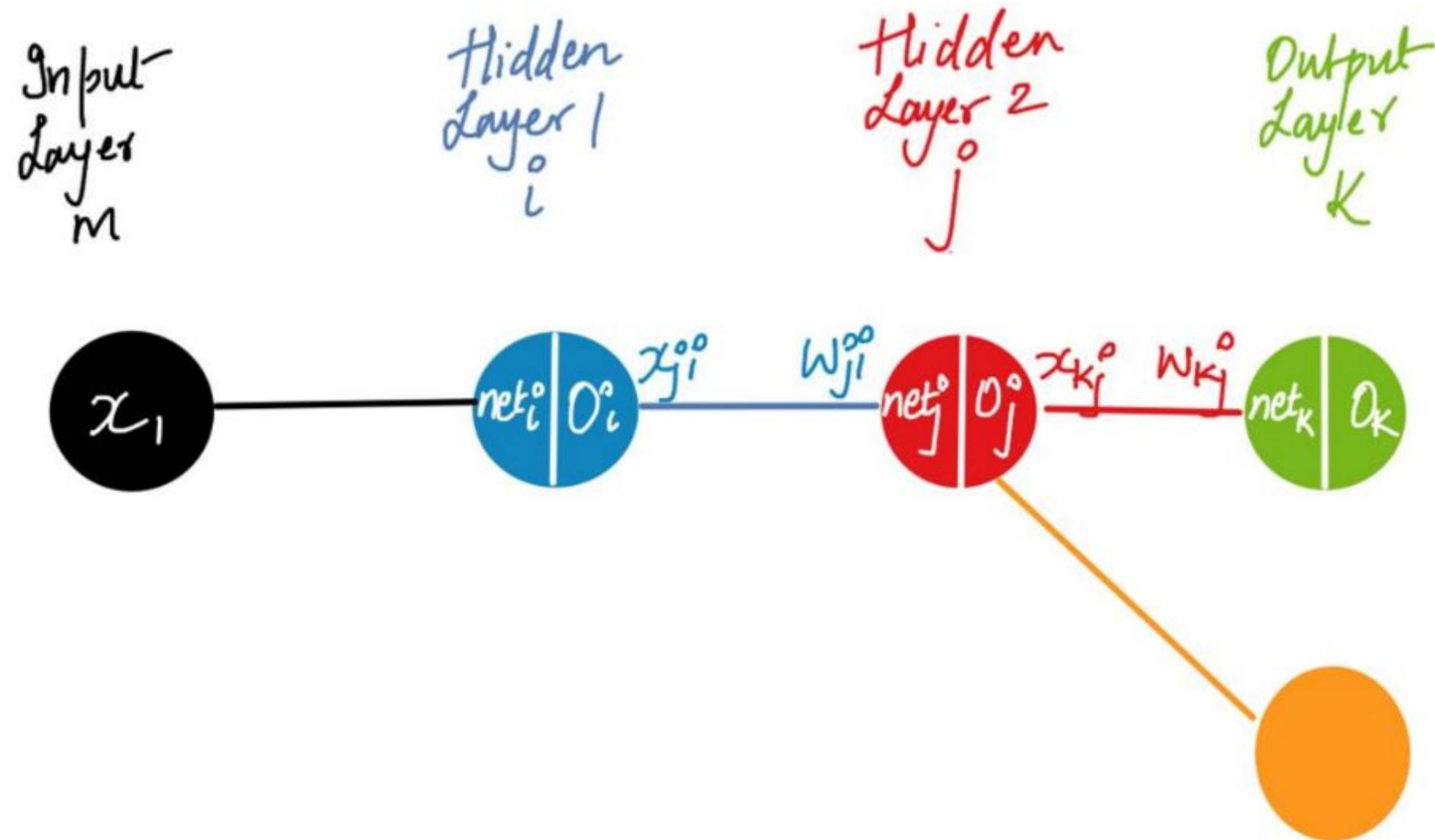
We consider two cases in turn:

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial \text{net}_j} x_{ji}$$

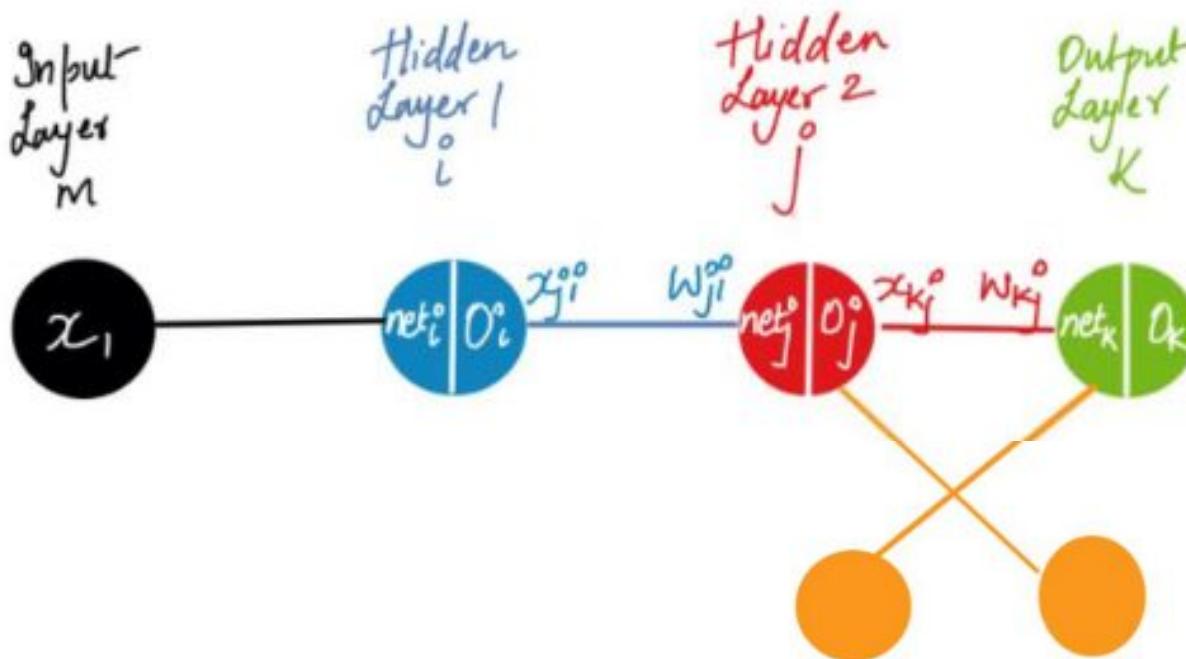
- Case 1, where unit j is an output unit for the network, and
- Case 2, where unit j is an internal unit of the network.

Use  $\delta_j = -\frac{\partial E_d}{\partial \text{net}_j}$  so the weight update looks like,  $\Delta w_{ji} = \eta \delta_j x_{ji}$

# Backward propagation



# Training rule for output unit weights



$$E_d = \frac{1}{2} \sum_{k \in \text{outputs}} (t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial \text{net}_K} = \frac{\partial E_d}{\partial o_K} \cdot \frac{\partial o_K}{\partial \text{net}_K}$$

$$\frac{\partial E_d}{\partial o_K} = -(t_k - o_k)$$

$$\frac{\partial o_K}{\partial \text{net}_K} = \frac{\partial (\sigma(\text{net}_K))}{\partial \text{net}_K} = o_K(1-o_K)$$

We know that

$$\omega_{kj} = \omega_{kj} + \Delta \omega_{kj}$$

$$\Delta \omega_{kj} = -\eta \frac{\partial E_d}{\partial \omega_{kj}}$$

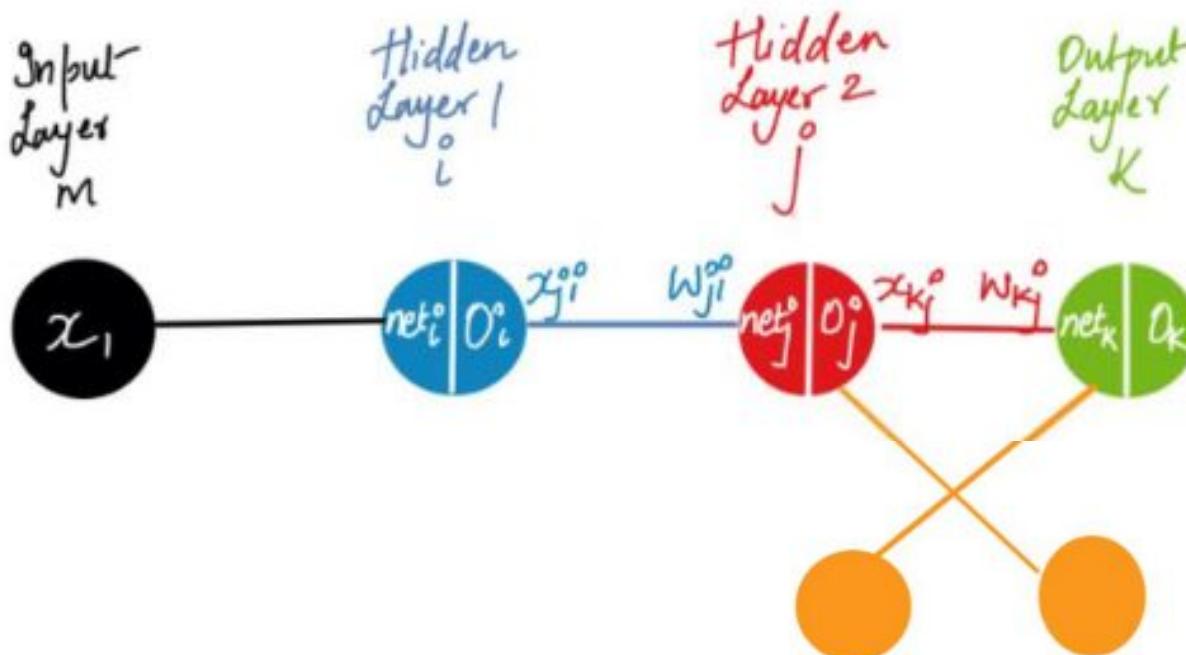
$$\Delta \omega_{kj} = -\eta \frac{\partial E_d}{\partial \text{net}_K} \frac{\partial \text{net}_K}{\partial \omega_{kj}}$$

$$\text{net}_K = \sum_j \omega_{kj} x_{kj}$$

$$\frac{\partial \text{net}_K}{\partial \omega_{kj}} = x_{kj}$$

$$\Delta \omega_{kj} = -\eta \frac{\partial E_d}{\partial \text{net}_K} \cdot x_{kj}$$

# Training rule for output unit weights

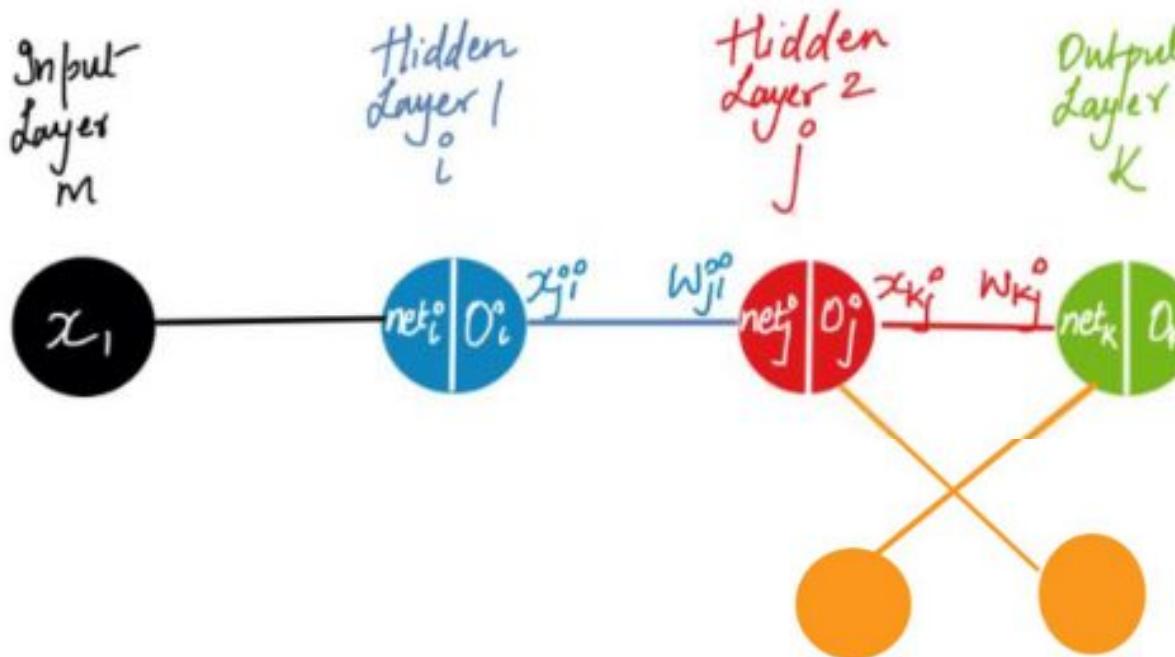


$$\begin{aligned}
\frac{dy}{dx} &= \frac{d\sigma(x)}{dx} = \frac{d}{dx} \left[ \frac{1}{1+e^{-x}} \right] \\
&= - (1+e^{-x})^{-2} \frac{d(1+e^{-x})}{dx} \\
&= - (1+e^{-x})^{-2} \left( \frac{d}{dx}(1) + \frac{d}{dx}(e^{-x}) \right) \\
&= - (1+e^{-x})^{-2} (0 + -e^{-x}) \\
&= \frac{e^{-x}}{(1+e^{-x})^2}
\end{aligned}$$

Adding  $1 \in -1$  to the numerator

$$\begin{aligned}
&\approx \frac{1+e^{-x}-1}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{(1+e^{-x})} \left[ 1 - \frac{1}{1+e^{-x}} \right] \\
&= \sigma(x) \left( 1 - \sigma(x) \right) \\
&= y(1-y)
\end{aligned}$$

# Training rule for output unit weights



$$\delta_k \leftarrow -\frac{\partial E_d}{\partial net_k}$$

$$\delta_k \leftarrow (t_k - O_k)^* O_k (1 - O_k)$$

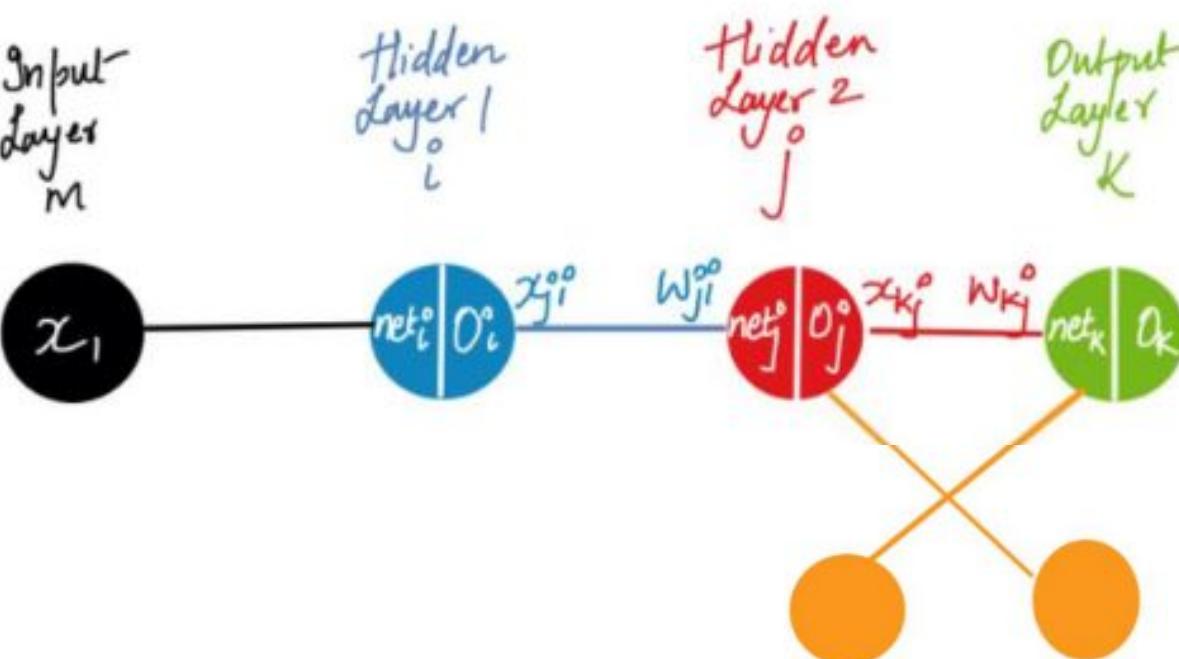
$$\frac{\partial E_d}{\partial net_k} = -(t_k - O_k)^* O_k (1 - O_k)$$

$$\Delta w_{kj} = -\eta^* - (t_k - O_k)^* O_k (1 - O_k)^* x_{kj}$$

$$\boxed{\Delta w_{kj} = \eta^* (t_k - O_k)^* O_k (1 - O_k)^* x_{kj}}$$

$$\Delta w_{kj} = \eta^* \delta_k^* x_{kj}$$

# Training rule for hidden layer 2 weights



$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in \text{Downstream}(j)} -\delta_k * w_{kj} * o_j(1-o_j)$$

$$\text{where } \delta_k = (t_k - o_k) * o_k(1-o_k)$$

$$\Delta w_{ji} = -\eta * \left[ \sum_{k \in \text{Downstream}(j)} -\delta_k * w_{kj} * o_j(1-o_j) \right] + x_{ji} \\ - \delta_j$$

$$\boxed{\Delta w_{ji} = \eta \delta_j x_{ji}}$$

SUKT

$$w_{ji} = w_{ji} + \Delta w_{ji}$$

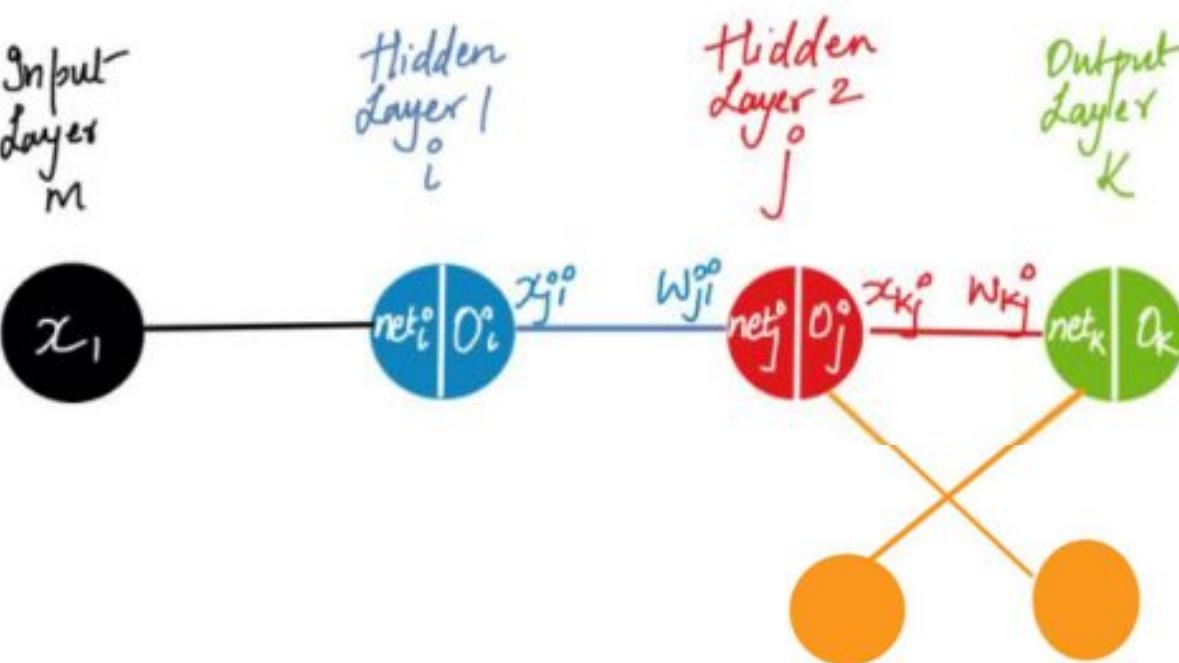
$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$net_k = \sum_j o_j \cdot w_{kj} \rightarrow \frac{\partial net_k}{\partial o_j} = w_{kj}$$

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in \text{Downstream}(j)} \frac{\partial E_d}{\partial o_k} \cdot \frac{\partial o_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j}$$

# Training rule for hidden layer 2 weights



SLKT

$$w_{ji} = w_{ji} + \Delta w_{ji}$$

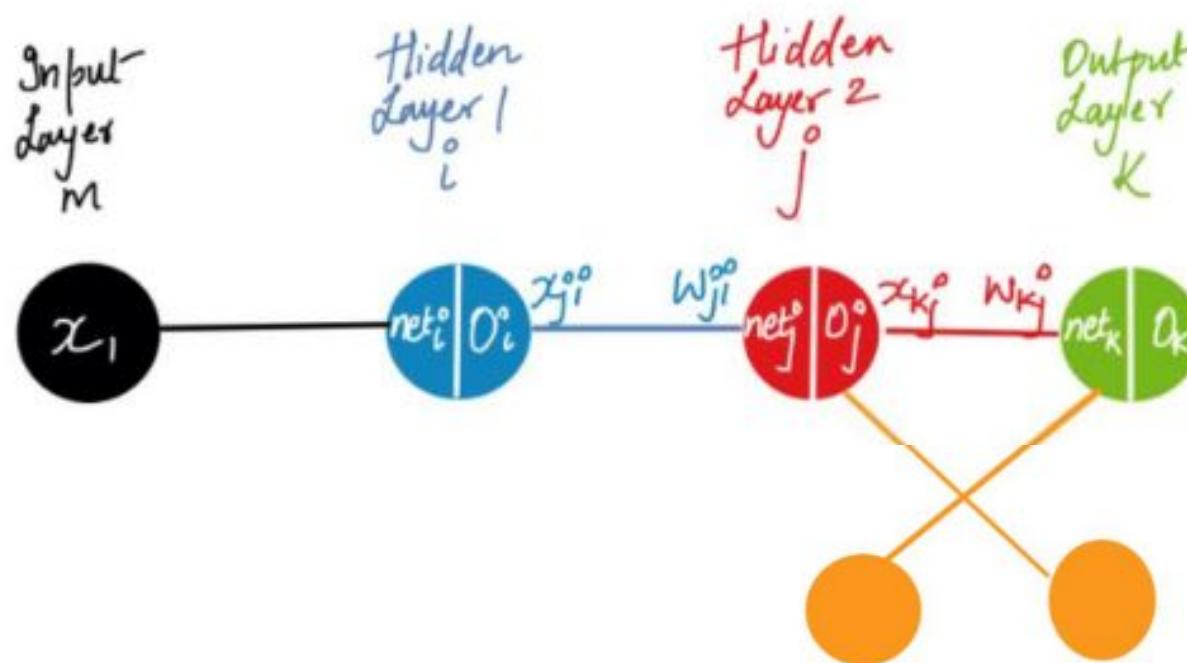
$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ji}}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} \cdot x_{ji}$$

$$net_k = \sum_j O_j \cdot w_{kj} \rightarrow \frac{\partial net_k}{\partial O_j} = w_{kj}$$

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in \text{downstream}(j)} \frac{\partial E_d}{\partial O_k} \cdot \frac{\partial O_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial net_j}$$

# Training rule for hidden layer 2 weights



$$\frac{\partial E_d}{\partial \text{net}_j} = \sum_{k \in \text{Downstream}(j)} -\delta_k * w_{kj} * O_j(1-O_j)$$

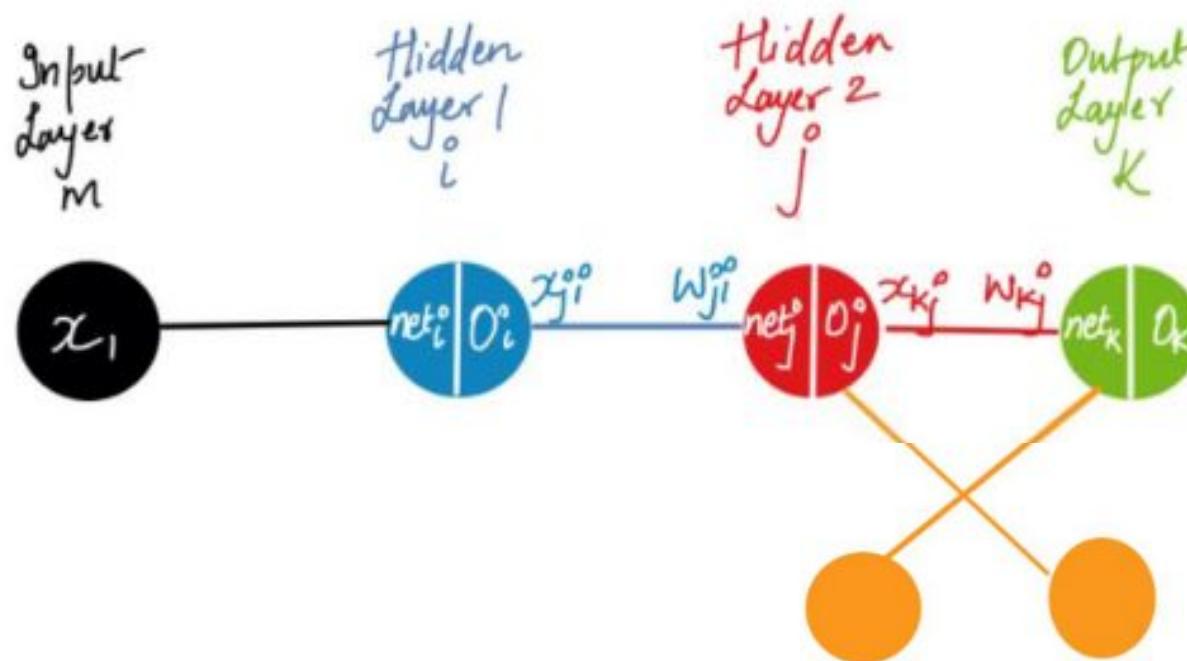
where  $\delta_k = (t_k - O_k) * O_k(1-O_k)$

$$\Delta w_{ji} = -\eta * \left[ \sum_{k \in \text{Downstream}(j)} -\delta_k * w_{kj} * O_j(1-O_j) + x_{ji} \right] - \delta_j$$

$\Delta w_{ji} = \eta \delta_j x_{ji}$

$$\frac{\partial E_d}{\partial \text{net}_j} = \sum_{k \in \text{Downstream}(j)} \frac{\partial E_d}{\partial O_k} \cdot \frac{\partial O_k}{\partial \text{net}_k} \cdot \frac{\partial \text{net}_k}{\partial O_j} \cdot \frac{\partial O_j}{\partial \text{net}_i}$$

# Training rule for hidden layer 2 weights



$$w_{im} = w_{im} + \Delta w_{im}$$

where

$$\Delta w_{im} = -\eta \frac{\partial E_d}{\partial w_{im}} \cdot x_{im}$$

$$\frac{\partial E_d}{\partial w_{im}} = \sum_{k \in \text{Downstream}(i)} -\delta_j * w_{ji} * O_i(1-O_i)$$

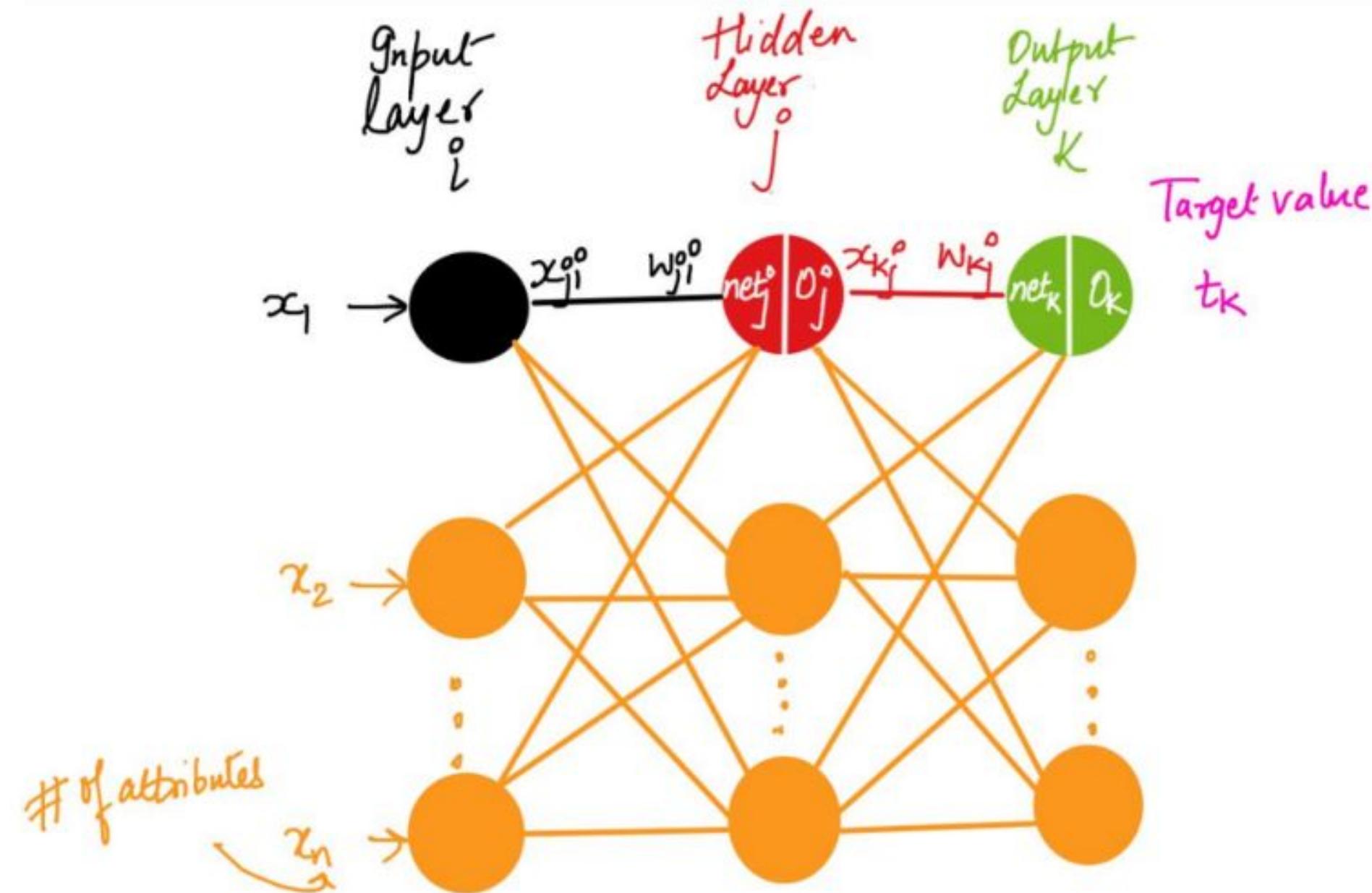
$$\Delta w_{im} = -\eta * \left[ \sum_{k \in \text{Downstream}(i)} -\delta_j * w_{ji} * O_i(1-O_i) \right] * x_{im}$$

$$-\delta_i$$

$$\Delta w_{im} = \eta * \delta_i * w_{im}$$

# Backward propagation Summary

So the typical set up will look like:



# Backward propagation Summary

## Weight update Equations

For output layer

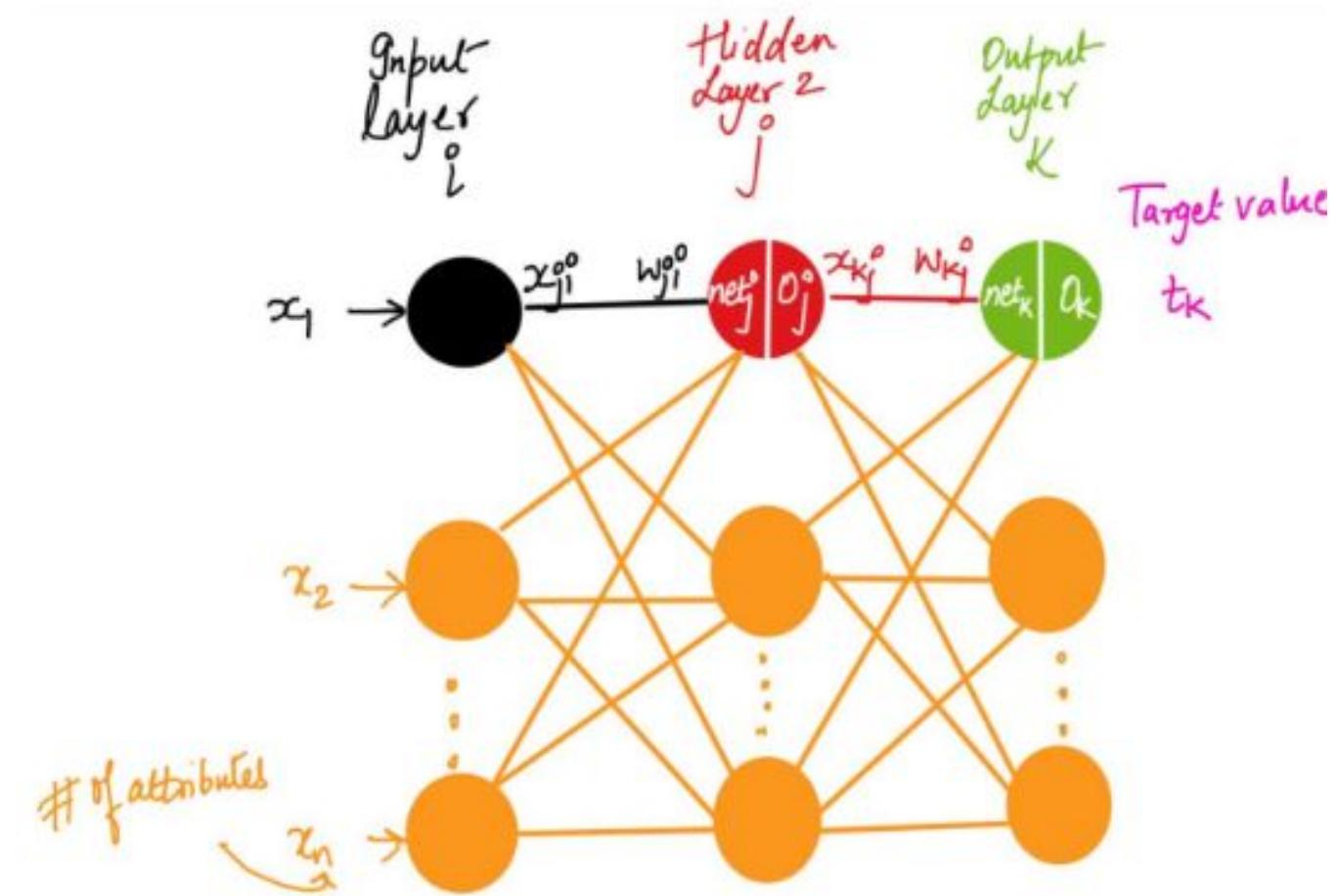
$$w_{kj} = w_{kj} + \Delta w_{kj}$$

where ,

$$\Delta w_{kj} = -\eta * -(t_k - o_k) * o_k (1-o_k) * x_{kj}$$

Simplifying

$$\boxed{\Delta w_{kj} = \eta * (t_k - o_k) * o_k (1-o_k) * x_{kj}}$$



# Backward propagation Summary

## Weight update Equations

For Hidden  
layer 2

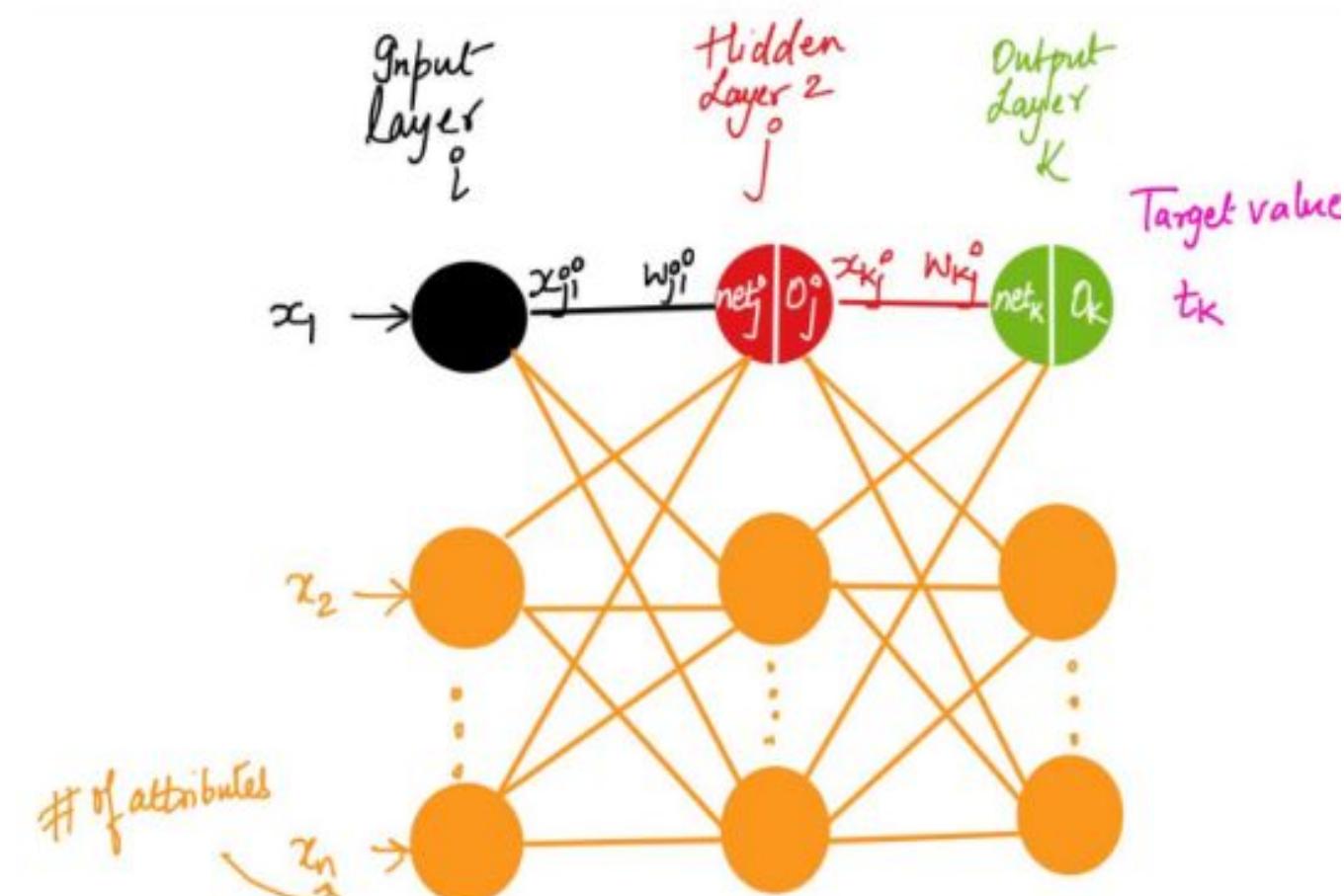
$$w_{ji} = w_{ji} + \Delta w_{ji}$$

where,

$$\Delta w_{ji} = -\eta * \left[ \sum_{k \in \text{Downstream}(j)} \delta_k * w_{kj} * o_j(1-o_j) \right] * x_{ji}$$

$$= -\eta * o_j(1-o_j) * x_{ji} \left( \sum_{k \in \text{Downst}(j)} \delta_k * w_{kj} \right)$$

$$\Delta w_{ji} = -\eta * o_j(1-o_j) * x_{ji} \left[ \sum_{k \in \text{Downst}(j)} -(t_k - o_k) * o_k(1-o_k) * w_{kj} \right]$$



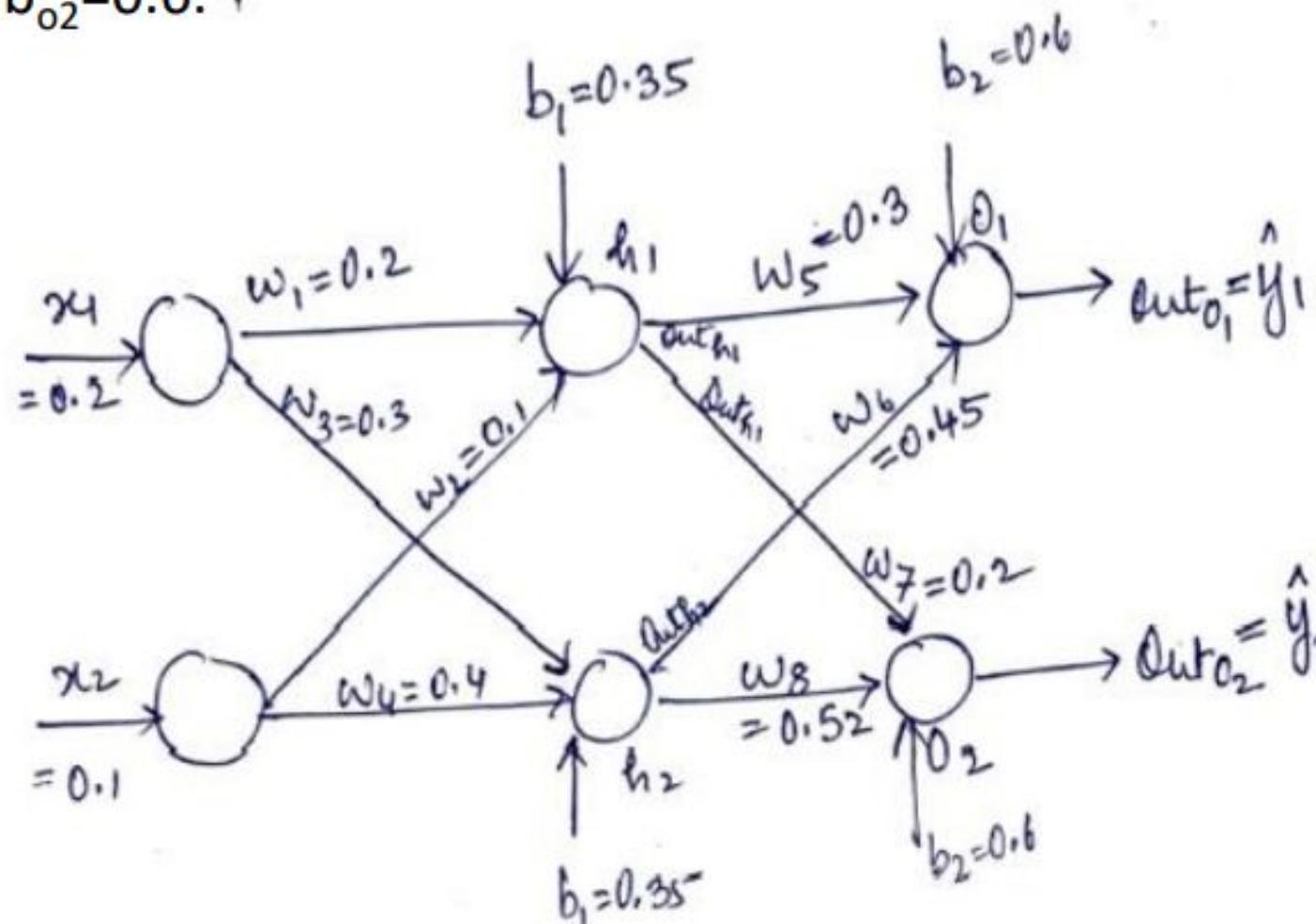
# Backward propagation Problem

---

# Backward Propagation Problem

# Backward Propagation Problem

Consider the input feature vector as [0.2, 0.1] with initial weights as  $w_1=0.2, w_2=0.1, w_3=0.3, w_4=0.4, w_5=0.3, w_6=0.45, w_7=0.2, w_8=0.52$  and biases as  $b_{h1}=0.35, b_{h2}=0.35, b_{o1}=0.6, b_{o2}=0.6$ .



Target at  $O_1 = 0.01, O_2 = 0.99, \eta = 0.5$

Forward pass,

$$\begin{aligned} \text{net}_{h1} &= w_1x_1 + w_2x_2 + b_1 \\ &= (0.2)(0.2) + (0.1)(0.1) + 0.35 \\ &= 0.04 + 0.01 + 0.35 = 0.4 \end{aligned}$$

Apply activation function Sigmoid,

$$\sigma(\text{net}_{h1}) = \frac{1}{1+e^{-0.4}} = \frac{1}{1+0.6703} = 0.5986 = \text{out}_{h1}$$

Similarly,

$$\sigma(\text{net}_{h2}) = \frac{1}{1+e^{-0.45}} = \frac{1}{1+0.6376} = 0.6106 = \text{out}_{h2}$$

# Backward Propagation Problem

Carrying out same process for O<sub>1</sub>,

$$\begin{aligned} \text{net}_{O_1} &= \text{out}_{h1} * w_5 + \text{out}_{h2} * w_6 + b_2 \\ &= (0.5986)(0.3) + (0.6106)(0.45) + 0.6 \\ &= 0.17958 + 0.27477 + 0.6 \\ &= 1.0543 \end{aligned}$$

Total Error, E<sub>total</sub> =  $\frac{1}{2} \sum (t - \hat{y})^2$

$$E_{O_1} = \frac{1}{2} (0.01 - 0.7415)^2 = 0.2675$$

$$E_{O_2} = \frac{1}{2} (0.99 - 0.7383)^2 = 0.0316$$

$$E_{\text{total}} = E_{O_1} + E_{O_2} = 0.2991 \quad (1)$$

## Output Layer

$$\begin{aligned} \text{net}_{O_2} &= \text{out}_{h1} * w_7 + \text{out}_{h2} * w_8 + b_2 \\ &= (0.5986)(0.2) + (0.6106)(0.52) + 0.6 \\ &= 0.11972 + 0.3175 + 0.6 \\ &= 1.0372 \end{aligned}$$

$$\text{out}_{O_1} = \sigma(\text{net}_{O_1}) = \frac{1}{1+e^{-\text{net}_{h1}}} = \frac{1}{1+e^{-1.0543}} = 0.7415$$

$$\text{out}_{O_2} = \sigma(\text{net}_{O_2}) = \frac{1}{1+e^{-1.0372}} = 0.7383$$

Consider w<sub>5</sub>, so we need to compute

$$\frac{\partial E_{\text{total}}}{\partial w_5} \text{ (rate of change of Error wrt } w_5)$$

i.e we would be interested in knowing how much change in w<sub>5</sub> affects the total error, E<sub>total</sub>

# Backward Propagation Problem

Now applying chain rule,

$$\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$

From (1),

$$\begin{aligned}\frac{\partial E_{\text{total}}}{\partial out_{o1}} &= \frac{1}{2} \times 2 (t_{o1} - out_{o1}) (-1) \\ &= -(t_{o1} - out_{o1}) \\ &= -(0.01 - 0.7415) = 0.7315\end{aligned}$$

Now,  $\frac{\partial out_{o1}}{\partial net_{o1}} = ?$

As  $out_{o1} = \frac{1}{1+e^{-net_{o1}}}$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1} (1 - Out_{o1})$$

$$= (0.7415)(1 - 0.7415) = 0.1916$$

Now,  $\frac{\partial net_{o1}}{\partial w_5} = ?$

As  $net_{o1} = out_{h1} \times w_5 + out_{h2} \times w_6 + b_2$

Hence,  $\frac{\partial net_{o1}}{\partial w_5} = out_{h1} = 0.5986$

$$\begin{aligned}\text{And, } \frac{\partial E_{\text{total}}}{\partial w_5} &= 0.7315 \times 0.1916 \times 0.5986 \\ &= 0.08389\end{aligned}$$

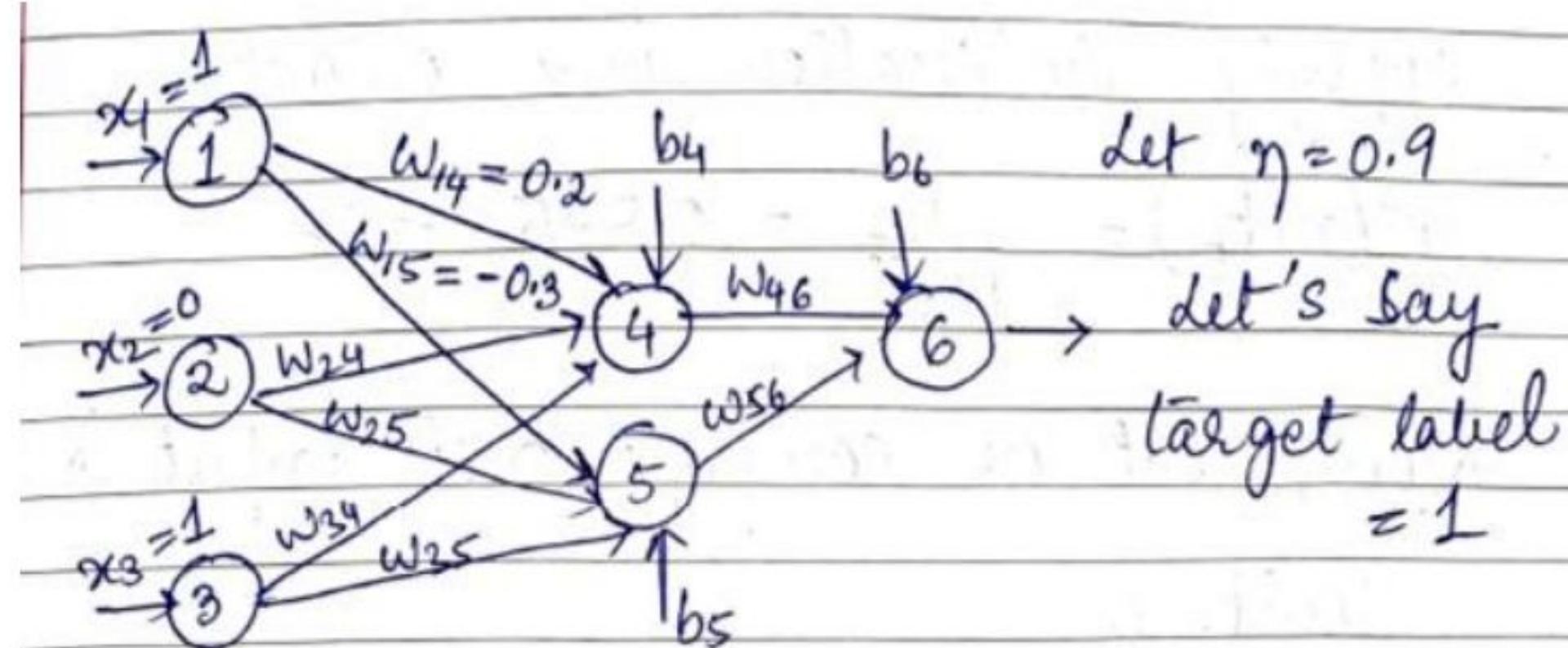
Decrease in the error/adjust in the weights

$$w_{5(\text{new})} = w_{5(\text{old})} - \eta \frac{\partial E_{\text{total}}}{\partial w_{5(\text{old})}}$$

$$= 0.3 - 0.5(0.08389)$$

$$= 0.3 - 0.041945 = 0.258055$$

# Backward Propagation Problem

Q  
2

Consider the input feature vector as  $[1, 0, 1]$  with other values as shown in figure. And train the neural network for the given feature vector with complete 1<sup>st</sup> iteration.

Target at O = 1,  $\eta = 0.9$ ,  $w_{14} = 0.2$ ,  $w_{15} = 0.3$ ,  $w_{24} = 0.4$ ,  $w_{25} = 0.1$ ,  $w_{34} = -0.5$ ,  $w_{35} = 0.2$ ,  $w_{46} = -0.3$ ,  $w_{56} = -0.2$ ,  $b_4 = -0.4$ ,  $b_5 = 0.2$ ,  $b_6 = 0.1$

# Backward Propagation Problem

**Q** Let's compute net input to unit 4

$$\begin{aligned} \text{net}_4 &= w_{14}x_1 + w_{24}x_2 + w_{34}x_3 + b_4 \\ &= 0.2 + 0 + 1(-0.5) + (-0.4) \\ &= -0.7 \end{aligned}$$

Apply activation function Sigmoid on  $\text{net}_4$ ,

$$\sigma(\text{net}_4) = \frac{1}{1+e^{-0.7}} = 0.332 = O_4 = \text{output of unit 4}$$

$$\begin{aligned} \text{net}_5 &= 1(-0.3) + 0 + 1(0.2) + (0.2) \\ &= 0.1 \end{aligned}$$

Apply activation function Sigmoid on  $\text{net}_5$ ,

$$\sigma(\text{net}_5) = \frac{1}{1+e^{-0.1}} = 0.525 = O_5$$

Let's compute net input to unit 6

$$\begin{aligned} \text{net}_6 &= \sigma(\text{net}_4) * w_{46} + \sigma(\text{net}_5) * w_{56} + b_6 \\ &= 0.332(-0.3) + (0.525)(-0.2) + (0.1) \\ &= -0.105 \end{aligned}$$

Apply activation function Sigmoid on  $\text{net}_6$ ,

$$O_6 = 0.474$$

So, 0.474 is the final output of the network

# Backward Propagation Problem

Q  
2

## Backpropagate

Calculation of error at each node

Consider Unit 6, since unit 6 is output unit, hence error term  $\delta_6$

$$\delta_6 = O_6 \times (1 - O_6) \times (t_6 - O_6)$$

$$\delta_6 = 0.474 \times (1 - 0.474) \times (1 - 0.474)$$

$$= 0.1311$$

Now error at unit 5 (hidden node)

$$\delta_5 = O_5 \times (1 - O_5) \sum_{k \in \text{outputs}} w_{kh} \delta_k$$

$$= O_5 \times (1 - O_5) (w_{65} \delta_6)$$

$$= 0.525 \times (1 - 0.525) ((-0.2)(0.1311)) = -0.0065$$

Now error at unit 4

$$\delta_4 = O_4 \times (1 - O_4) (w_{46} \delta_6)$$

$$= 0.332 \times (1 - 0.332) ((-0.3)(0.1311)) = -0.0087$$

# Backward Propagation Problem

Q  
2

New weight term corresponding to  $w_{46}$

$$\begin{aligned} W_{46(\text{new})} &= w_{46(\text{old})} + \eta \delta_6 o_4 \\ &= -0.3 + 0.9(0.1311)(0.332) \\ &= -0.261 \end{aligned}$$

Similarly,

$$\begin{aligned} W_{56(\text{new})} &= w_{56(\text{old})} + \eta \delta_6 o_5 \\ &= -0.2 + 0.9(0.1311)(0.525) \\ &= -0.138 \end{aligned}$$

Now,

$$\begin{aligned} W_{14(\text{new})} &= w_{14(\text{old})} + \eta \delta_4 o_1 \quad [O_1 = x_1] \\ &= 0.2 + 0.9(-0.0087)(1) \\ &= 0.192 \end{aligned}$$

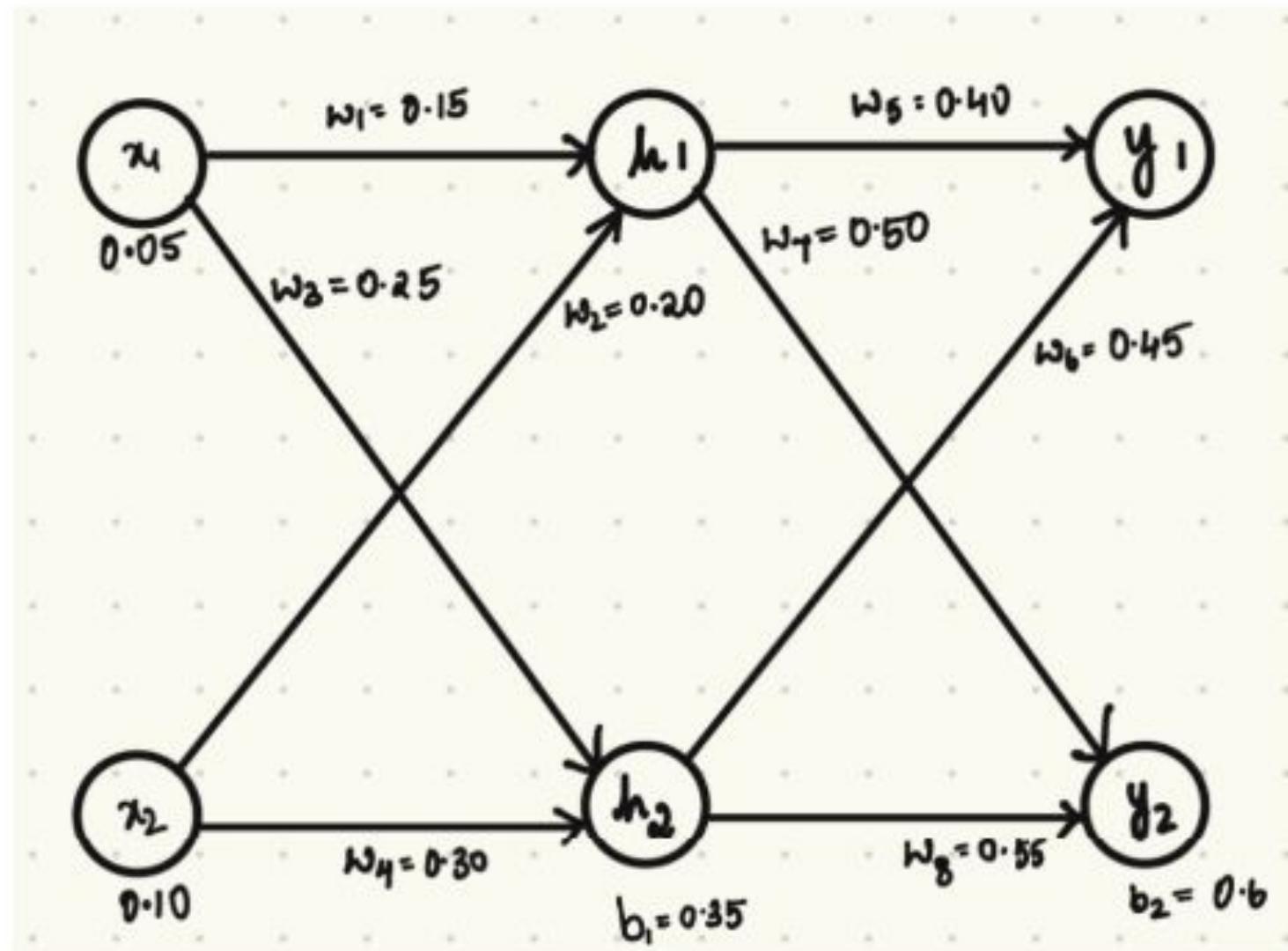
$$\begin{aligned} W_{15(\text{new})} &= w_{15(\text{old})} + \eta \delta_5 o_1 \\ &= -0.3 + 0.9(-0.0065)(1) \\ &= -0.306 \end{aligned}$$

Similarly, calculate updated weights,  $w_{24}, w_{25}, w_{34}, w_{35}$

# Backward Propagation Problem

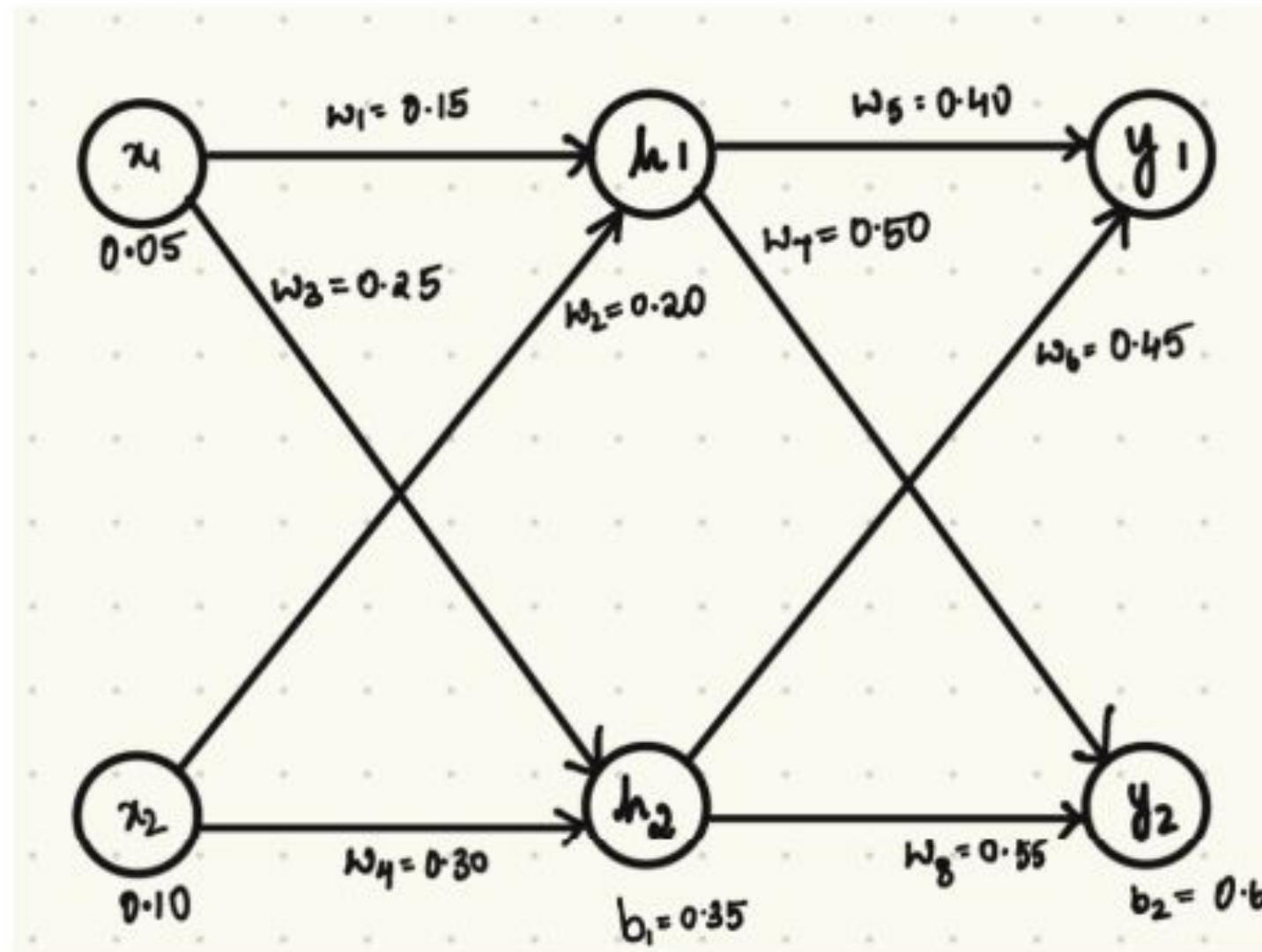
Q

3



Find the new weights in the Given Figure after 1 iteration

# Backward Propagation Problem

Q  
3

Target at  $O_1 = 0.01, O_2 = 0.99, \eta = 0.5$

Forward pass,

$$\begin{aligned} \text{net}_{h1} &= w_1x_1 + w_2x_2 + b_1 \\ &= (0.05)(0.15) + (0.10)(0.20) + 0.35 \\ &= 0.3775 \end{aligned}$$

$$\begin{aligned} \text{net}_{h2} &= w_3x_1 + w_4x_2 + b_1 \\ &= (0.05)(0.25) + (0.10)(0.30) + 0.35 \\ &= 0.3925 \end{aligned}$$

Apply activation function Sigmoid,

$$\sigma(\text{net}_{h1}) = \frac{1}{1+e^{-0.3775}} = 0.5932 = \text{out}_{h1}$$

Similarly,

$$\sigma(\text{net}_{h2}) = \frac{1}{1+e^{-0.3925}} = 0.5968 = \text{out}_{h2}$$

# Backward Propagation Problem

Q  
3

Carrying out same process for  $O_1$ ,

$$\begin{aligned} \text{net}_{O1} &= \text{out}_{h1} * w_5 + \text{out}_{h2} * w_6 + b_2 \\ &= (0.5932)(0.4) + (0.5968)(0.45) + 0.6 \\ &= 1.059 \end{aligned}$$

$$\begin{aligned} \text{net}_{O2} &= \text{out}_{h1} * w_7 + \text{out}_{h2} * w_8 + b_2 \\ &= (0.5932)(0.5) + (0.5968)(0.55) + 0.6 \\ &= 1.225 \end{aligned}$$

$$\text{out}_{o1} = \sigma(\text{net}_{h1}) = \frac{1}{1+e^{-\text{net}_{o1}}} = \frac{1}{1+e^{-1.059}} = 0.7513$$

$$\text{out}_{o2} = \sigma(\text{net}_{h2}) = \frac{1}{1+e^{-1.0372}} = 0.7729$$

$$\text{Total Error, } E_{\text{total}} = \frac{1}{2} \sum (t - \hat{y})^2$$

$$E_{O1} = \frac{1}{2} (0.01 - 0.7513)^2$$

$$E_{O2} = \frac{1}{2} (0.99 - 0.7729)^2$$

$$E_{\text{total}} = E_{O1} + E_{O2} = 0.2983 \quad (1)$$

# Backward Propagation Problem

Q  
3

Backpropagation (Output Layer):

Applying chain rule,

$$\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial net_{o1}} \times \frac{\partial net_{o1}}{\partial w_5}$$

From (1),

$$\begin{aligned}\frac{\partial E_{\text{total}}}{\partial out_{o1}} &= \frac{1}{2} \times 2 (to_1 - out_{o1}) (-1) \\ &= -(to_1 - out_{o1}) \\ &= -(0.01 - 0.7513) = 0.7413\end{aligned}$$

Now,  $\frac{\partial out_{o1}}{\partial net_{o1}} = ?$

$$\text{As } out_{o1} = \frac{1}{1+e^{-net_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial net_{o1}} = Out_{o1} (1 - Out_{o1})$$

$$= (0.7513)(1 - 0.7513) = 0.1868$$

Now,  $\frac{\partial net_{o1}}{\partial w_5} = ?$

$$\text{As } net_{o1} = out_{h1} \times w_5 + out_{h2} \times w_6 + b_2$$

$$\text{Hence, } \frac{\partial net_{o1}}{\partial w_5} = out_{h1} = 0.5932$$

$$\begin{aligned}\text{And, } \frac{\partial E_{\text{total}}}{\partial w_5} &= 0.7413 \times 0.1868 \times 0.5932 \\ &= 0.0821\end{aligned}$$

Decrease in the error/adjust in the weights

$$w_{5(\text{new})} = w_{5(\text{old})} - \eta \frac{\partial E_{\text{total}}}{\partial w_{5(\text{old})}}$$

$$\begin{aligned}&= 0.4 - 0.5(0.0821) \\ &= 0.3589\end{aligned}$$

Similarly, calculate  $w_{6(\text{new})}, w_{7(\text{new})}, w_{8(\text{new})}$

$$w_{6(\text{new})} = 0.4086$$

$$w_{7(\text{new})} = 0.5113$$

$$w_{8(\text{new})} = 0.5613$$

# Backward Propagation Problem

Q  
3

Backpropagation (Hidden Layer):

Applying chain rule,

$$\frac{\partial E_{\text{total}}}{\partial w_1} = \frac{\partial E_{\text{total}}}{\partial net_{h1}} \times \frac{\partial net_{h1}}{\partial h_1} \times \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial E_{\text{total}}}{\partial net_{h1}} = \frac{\partial E_1}{\partial net_{h1}} + \frac{\partial E_2}{\partial net_{h1}}$$

$$= \left( \frac{\partial E_1}{\partial net_{y1}} \times \frac{\partial net_{y1}}{\partial y_1} \times \boxed{\frac{\partial y_1}{\partial net_{h1}}} \right) + \left( \frac{\partial E_2}{\partial net_{y2}} \times \frac{\partial net_{y2}}{\partial y_2} \times \boxed{\frac{\partial y_2}{\partial net_{h1}}} \right)$$

Except the highlighted, all other values have been calculated before, hence there is no need to recalculate them

$$\frac{\partial y_1}{\partial net_{h1}} = ?$$

$$\text{As } y_1 = net_{h1} \times w_5 + net_{h2} \times w_6 + b_2$$

$$\text{Hence, } \frac{\partial y_1}{\partial net_{h1}} = w_5 = 0.4$$

$$\frac{\partial y_2}{\partial net_{h1}} = ?$$

$$\text{As } y_1 = net_{h1} \times w_7 + net_{h2} \times w_8 + b_2$$

$$\text{Hence, } \frac{\partial y_2}{\partial net_{h1}} = w_7 = 0.5$$

$$\frac{\partial E_{\text{total}}}{\partial net_{h1}} = \frac{\partial E_1}{\partial net_{h1}} + \frac{\partial E_2}{\partial net_{h1}} = 0.0364$$

$$\begin{aligned} \text{And, } \frac{\partial net_{h1}}{\partial h_1} &= net_{h1} (1 - net_{h1}) \\ &= 0.5932 (1 - 0.5932) \\ &= 0.2418 \end{aligned}$$

# Backward Propagation Problem

Q  
3

Now,  $\frac{\partial h_1}{\partial w_1} = ?$

As  $h_1 = x_1 \times w_1 + x_2 \times w_3 + b_1$

Hence,  $\frac{\partial h_1}{\partial w_1} = w_1 = 0.05$

And,  $\frac{\partial E_{\text{total}}}{\partial w_1} = 0.0364 \times 0.2418 \times 0.05$   
 $= 0.043$

Decrease in the error/adjust in the weights

$$\begin{aligned} w_{1(\text{new})} &= w_{1(\text{old})} - \eta \frac{\partial E_{\text{total}}}{\partial w_{1(\text{old})}} \\ &= 0.15 - 0.5(0.0043) \\ &= 0.1497 \end{aligned}$$

Similarly, calculate  $w_{2(\text{new})}, w_{3(\text{new})}, w_{4(\text{new})}$

$$\begin{aligned} w_{2(\text{new})} &= 0.1195 \\ w_{3(\text{new})} &= 0.2497 \\ w_{4(\text{new})} &= 0.2995 \end{aligned}$$

# Backward propagation algorithm

**Propagate the input forward through the network**

1. Input the instance  $\mathbf{x}$  to the network and compute the output  $\mathbf{o}_u$  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  $d_k$

$$\delta_k \leftarrow (t_k - o_k) * \theta_k(1-\theta_k)$$

3. For each hidden unit  $h$ , calculate its error term  $d_h$

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$$

4. Update each network weight  $w_{ji}$

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\text{Use } \delta_j = -\frac{\partial E_d}{\partial net_j}$$

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

# Vanishing Gradients Problem with Sigmoid

---

## Vanishing Gradients

As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, the gradients often get smaller and smaller and approach zero which eventually leaves the weights of the initial or lower layers nearly unchanged. As a result, the gradient descent never converges to the optimum. This is known as the vanishing gradients problem.

## Overfitting – Analogous to Decision Trees case

---

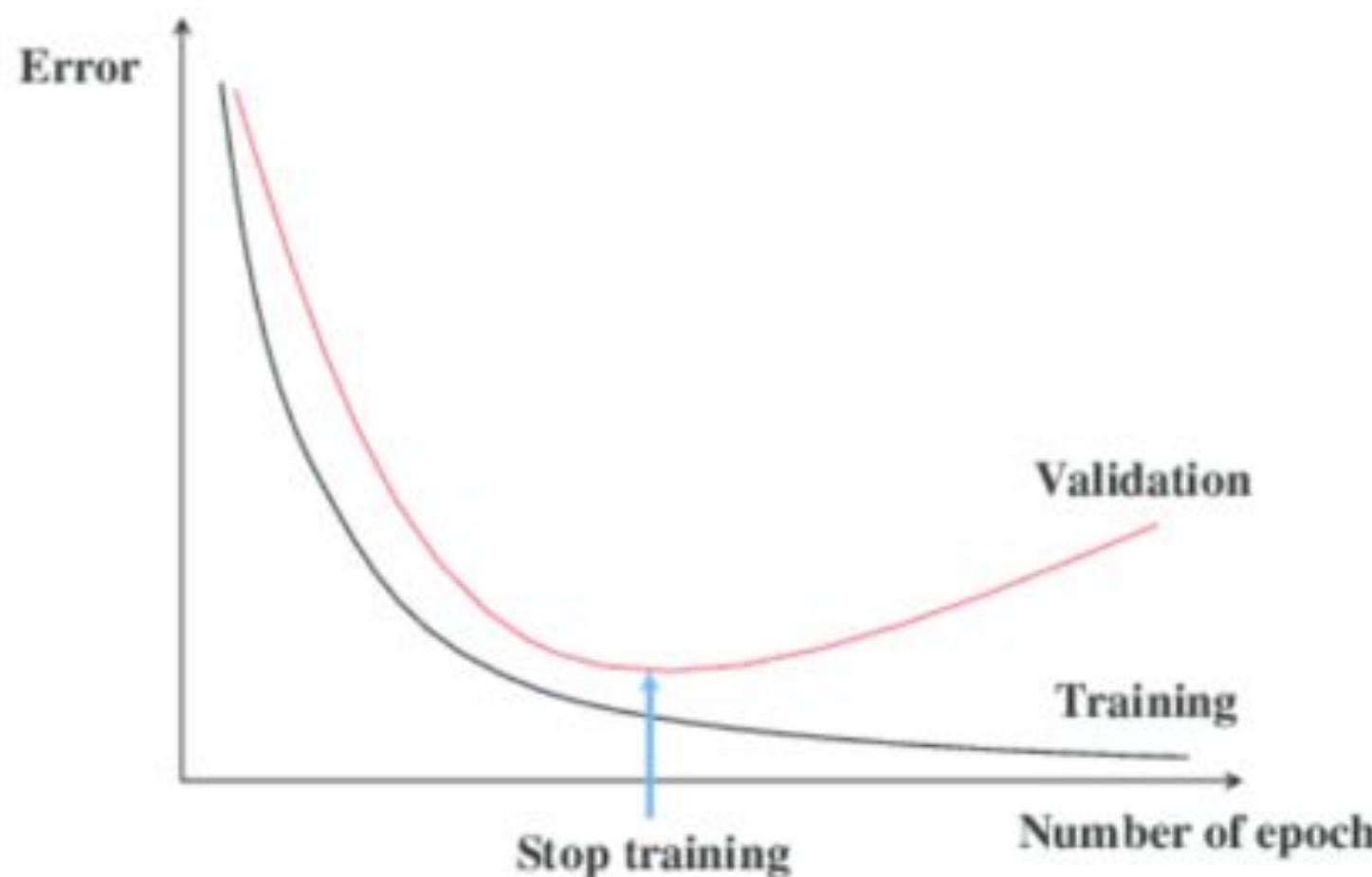
- With several weights to update over several iterations, the backpropagation algorithm tends to overfit.
- Overfitting occurs because the weights are being tuned to fit noise of the training examples that are not representative of the general distribution of examples.

Possible solutions:

- Maintain a separate validation set, apart from training and test sets.
- Measure the accuracy over the validation set at intervals.
- Choose the set of weights that produce the least error on the validation set.

## Overfitting – Analogous to Decision Trees case

Early termination: This regularization technique updates the model to make it better fit the training data with each iteration. After a certain number of iterations, new iterations improve the model. After that point, however, the model begins to overfit the training data. Early stopping refers to stopping the training process before that point.





**PES**  
UNIVERSITY

CELEBRATING 50 YEARS

# THANK YOU

---

**Dr. Surabhi Narayan, Professor  
Department of Computer Science and  
Engineering surabhinarayan@pes.edu**



**PES**  
UNIVERSITY

# **UE21CS352A**

# **Machine Learning**

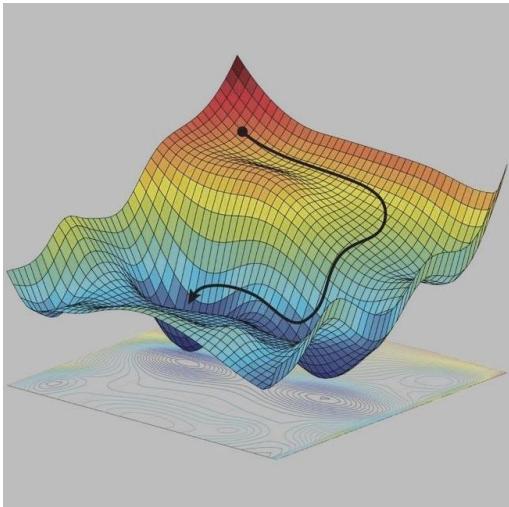
---

**Dr. Surabhi Narayan**

# Optimization

*Optimization is a process of finding optimal parameters for the model, which significantly reduces the error function.*

Optimization algorithms used for training of deep models differ from traditional optimization algorithms in several ways. In most machine learning scenarios we care about some performance measure say , P but we optimise P only indirectly by reducing a different cost function  $J(\theta)$  in the hope that doing so will improve P.



The following are the optimization algorithms discussed in this course :

- Mini Batch gradient descent
- Stochastic gradient descent
- Gradient descent with momentum
- RMS Prop
- Adam

# A Recap of Gradient Descent

Gradient Descent searched the hypothesis space of all possible weight vectors to find the best fit for all training examples. Say we have a cosy function  $J(w)$  and we wish to minimise this cost function:

1. We randomly initialize the weight vectors
2. We calculate the loss and update  $W_{\text{new}} = W - \alpha * \frac{\delta J(W)}{\delta W}$  direction of steepest descent.  
The equation to do so is :
3.  $W \leftarrow \text{repeat till we converge at a minima}$

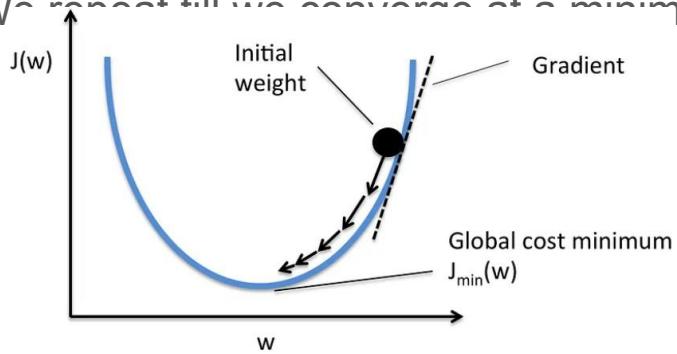
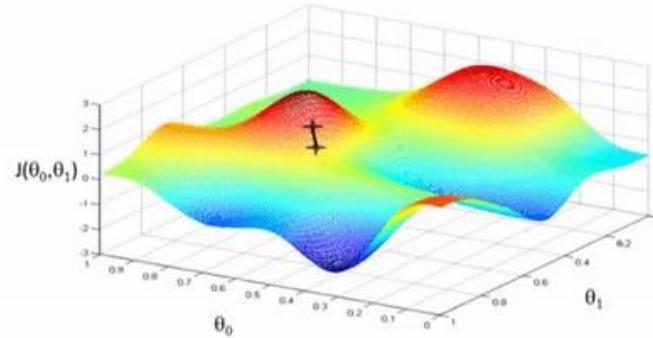


Figure 2: Example of minimizing  $J(w)$ ; (Source: MLextend)



Andrew Ng

# Batch Gradient Descent

Batch gradient descent is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated

## **Upsides**

- Fewer updates to the model means this variant of gradient descent is more computationally efficient than stochastic gradient descent.
- The decreased update frequency results in a more stable error gradient and may result in a more stable convergence on some problems.
- The separation of the calculation of prediction errors and the model update lends the algorithm to parallel processing based implementations.

## **Downsides**

- The more stable error gradient may result in premature convergence of the model to a less optimal set of parameters.
- The updates at the end of the training epoch require the additional complexity of accumulating prediction errors across all training examples.
- Commonly, batch gradient descent is implemented in such a way that it requires the entire training dataset in memory and available to the algorithm.
- Model updates, and in turn training speed, may become very slow for large datasets.

# Stochastic Gradient Descent

Stochastic gradient descent, often abbreviated SGD, is a variation of the gradient descent algorithm that calculates the error and updates the model for each example in the training dataset.

## **Upsides**

- The frequent updates immediately give an insight into the performance of the model and the rate of improvement.
- This variant of gradient descent may be the simplest to understand and implement, especially for beginners.
- The increased model update frequency can result in faster learning on some problems.
- The noisy update process can allow the model to avoid local minima (e.g. premature convergence).

## **Downsides**

- Updating the model so frequently is more computationally expensive than other configurations of gradient descent, taking significantly longer to train models on large datasets.
- The frequent updates can result in a noisy gradient signal, which may cause the model parameters and in turn the model error to jump around (have a higher variance over training epochs).
- The noisy learning process down the error gradient can also make it hard for the algorithm to settle on an error minimum for the model.

# Mini Batch Gradient Descent

---

Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model coefficients. Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.

## Upsides

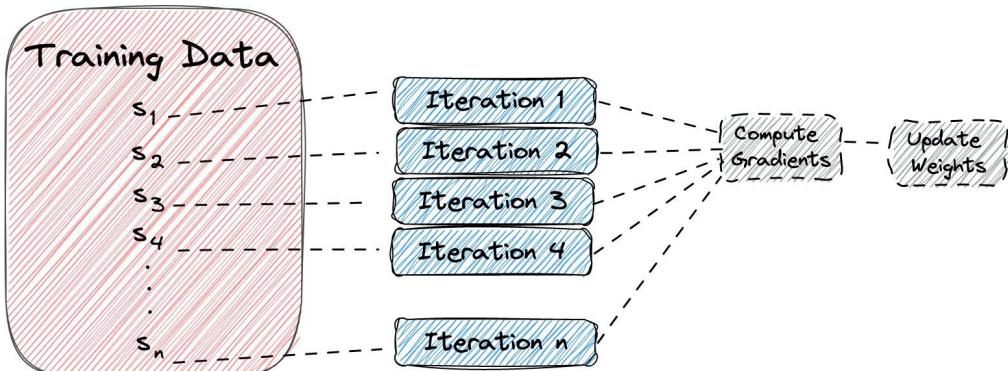
- The model update frequency is higher than batch gradient descent which allows for a more robust convergence, avoiding local minima.
- The batched updates provide a computationally more efficient process than stochastic gradient descent.
- The batching allows both the efficiency of not having all training data in memory and algorithm implementations.

## Downsides

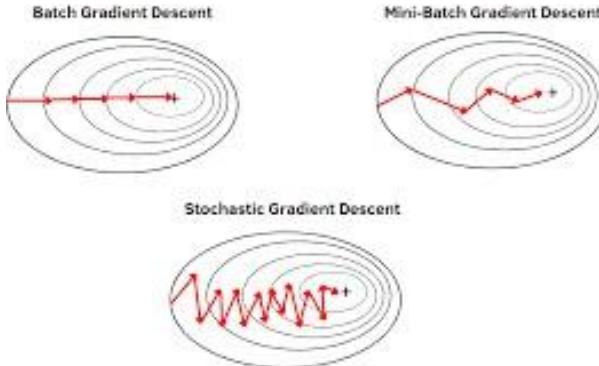
- Mini-batch requires the configuration of an additional “mini-batch size” hyperparameter for the learning algorithm.
- Error information must be accumulated across mini-batches of training examples like batch gradient descent

# How to choose the best value for Mini Batch size

- If there is a small training set batch gradient descent is preferred.
- Mini batch size is a hyperparameter and it is a good idea to experiment with a range of values to get the best fit for your model.
- Mini-batch sizes, commonly called “batch sizes” for brevity, are often tuned to an aspect of the computational architecture on which the implementation is being executed. Such as a power of two that fits the memory requirements of the GPU or CPU hardware like 32, 64, 128, 256, and so on.



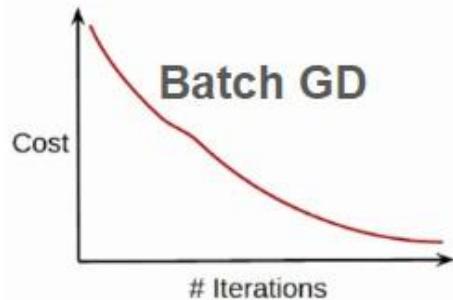
# Convergence in different versions of Gradient Descent



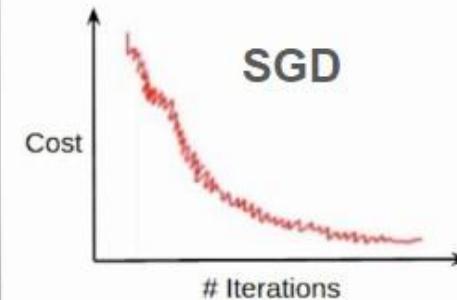
- Batch gradient descent takes small steps towards the minima and will converge to a minima.
- While Stochastic gradient descent is noisy and oscillates near the minima . It never actually

## Comparison: Cost function

- Cost function reduces smoothly



- Lot of variations in cost function



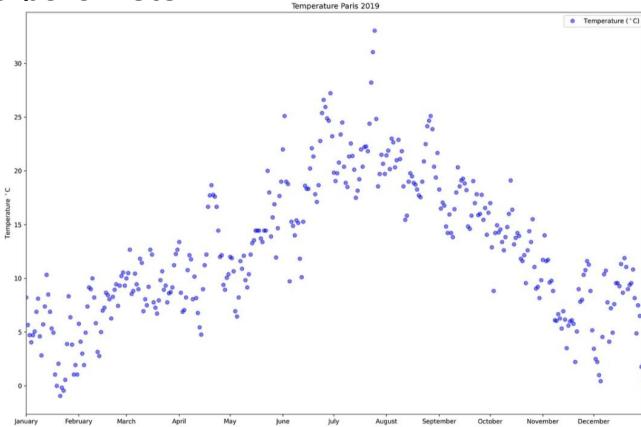
- Smoother cost function as compared to SGD



# Exponentially Weighted Averages

The Exponentially Weighted Moving Average (EWMA) is commonly used as a smoothing technique in time series. However, due to several computational advantages (fast, low-memory cost), the EWMA is behind the scenes of many optimization algorithms in deep learning, including Gradient Descent with Momentum, RMSprop, Adam, etc.

Let's understand with an example , say we have the some data for temperatures across multiple days in a city and we wish to approximate the next day's temperature from this data. Thus let the estimated temperature be  $V_t$  , let the previous estimate be  $V_{t-1}$  ,  $O_t$  be the temperature for day t and  $\beta$  be a hyperparameter.



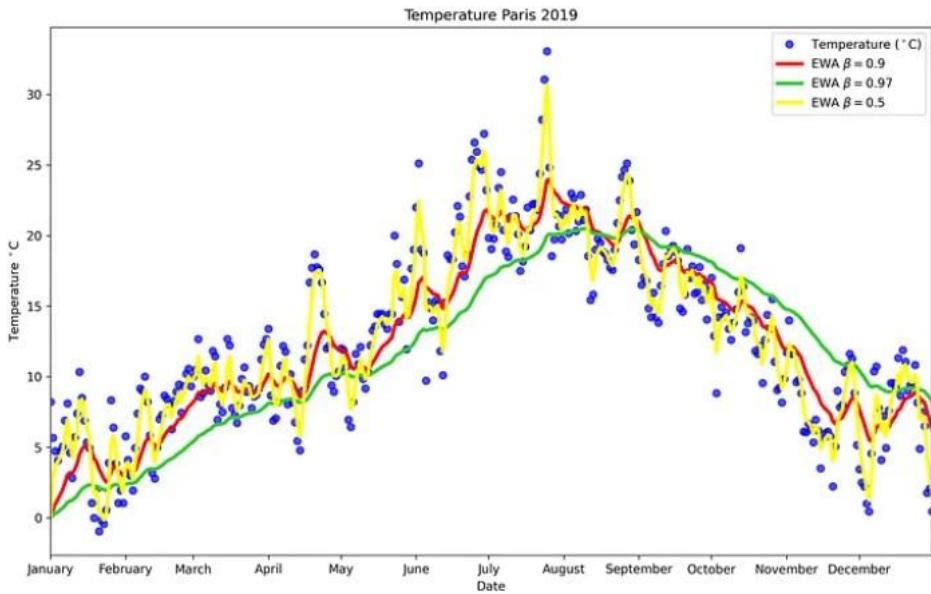
$$V_t = \beta * V_{t-1} + (1 - \beta) * O_t$$

Here ,  $\beta$  determines how important the previous value is (the trend), and  $(1-\beta)$  how important the current value is.

# Exponentially Weighted Averages

Here,  $V_t$  is calculated by approximating over  $1/(1-\beta)$  days of temperature. Thus if  $\beta$  is chosen to be 0.9 then we approximate over the last 10 days if 0.5 then the last 2 days and so on.

| $\beta$ | Days |
|---------|------|
| 0.9     | 10   |
| 0.98    | 50   |
| 0.5     | 2    |



**Note :** As the value of  $\beta$  increases the curve becomes smoother and has less noise.

However with large values of  $\beta$  as we are averaging over a larger window the formula adapts more slowly to changes in data as high weightage is given to older values

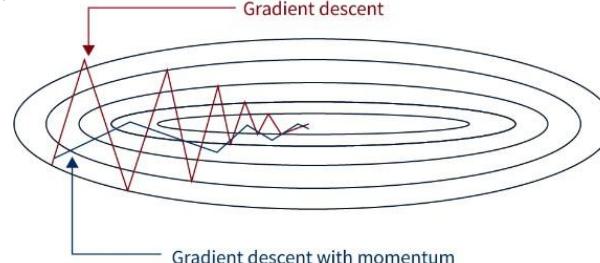
# Gradient Descent with Momentum

*In this method we compute the exponentially weighted average of the gradients and use this new value to calculate the weights instead*

A problem with the gradient descent algorithm is that the progression of the search can bounce around the search space based on the gradient. For example, the search may progress downhill towards the minima, but during this progression, it may move in another direction, even uphill, depending on the gradient of specific points (sets of parameters) encountered during the search.

This can slow down the progress of the search, especially for those optimization problems where the broader trend or shape of the search space is more useful than specific gradients along the way.

One approach to this problem is to add history to the parameter update equation based on the gradient encountered in the previous updates



# Gradient Descent with Momentum

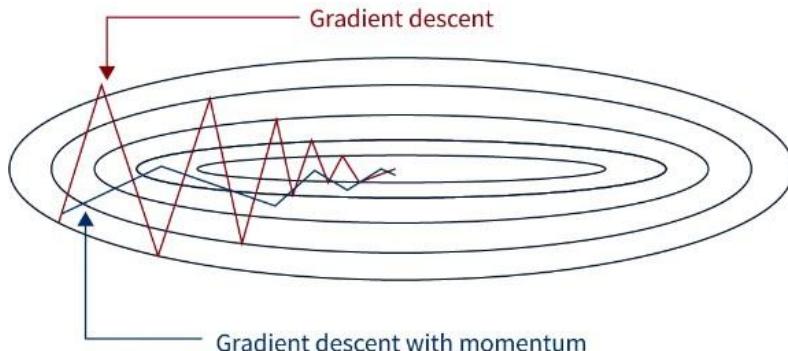
1. First we calculate  $dW$  (where  $w$  is the weight) for the current mini batch

1. Then we compute  $V_{dw} = \beta * V_{dw} + (1-\beta) * dw$

1. Then we update our weights as

$$W_{\text{new}} = W - \alpha * V_{dw}$$

**Using a very high value for  $\beta$  we can dampen out the oscillations which arise in gradient descent !**



# Gradient Descent with Momentum

## Upsides:

1. Momentum has the effect of damping down the change in the gradient and, in turn, the step size with each new point in the search space.
1. Momentum is most useful in optimization problems where the objective function has a large amount of curvature (e.g. changes a lot), meaning that the gradient may change a lot over relatively small regions of the search space.
1. It is also helpful when the gradient is estimated, such as from a simulation, and may be noisy, e.g. when the gradient has a high variance.
2. Finally, momentum is helpful when the search space is flat or nearly flat, e.g. zero gradient. The momentum allows the search to progress in the same direction as before the flat spot and helpfully cross the flat region.

## Downsides :

1. It can overshoot the global minimum and converge to a local minimum instead.
2. Another disadvantage is that the momentum term can cause the optimization process to oscillate around the global minimum.

## Root Mean Square or RMS prop

We know that while implementing gradient descent we end up with lots of oscillations  
 Let the horizontal direction be w and the vertical direction be b. In this case we wish to speed up learning in the w direction and slow down learning in the b direction.

for iteration t:

1. Calculate the derivative  $dW$  for the current mini batch ‘
2. ~~Calculate an exponentially weighted average of the squares of derivatives~~

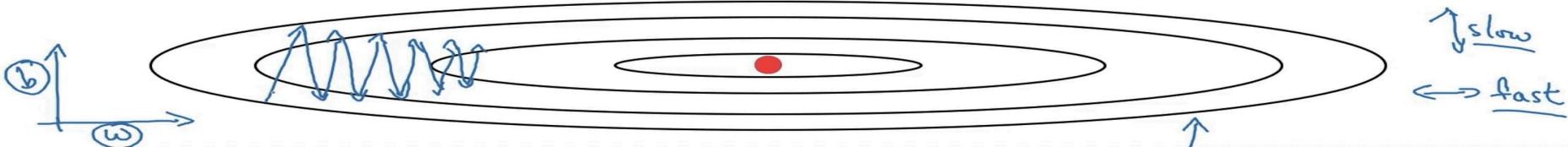
$$S_{dW} = \beta_2 * S_{dW} + (1 - \beta_2) * (dW)^2$$

NOTE :  $\beta_2$  and  $\beta$  are separate hyperparameters

$$S_{db} = \beta_2 * S_{db} + (1 - \beta_2) * (db)^2$$

1. Then we update the weights       $W_{\text{new}} = W - \alpha * \frac{dW}{\sqrt{SdW}}$  and       $b_{\text{new}} = b - \alpha * \frac{db}{\sqrt{Sdb}}$   
 as:

## RMSprop



# RMS prop intuition

- In this example we wish to dampen out oscillations in the b direction and to learn faster in the x direction.
- Thus as  $S_{dw}$  is a small number while  $S_{db}$  is a large number.

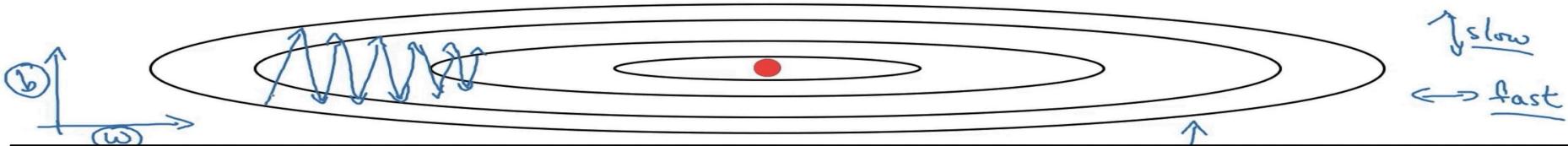
$$S_{dw} = \beta_2 * S_{dw} + (1 - \beta_2) * (dW)^2 \leftarrow \text{SMALL}$$

$$S_{db} = \beta_2 * S_{db} + (1 - \beta_2) * (db)^2 \leftarrow \text{LARGE}$$

$$W_{\text{new}} = W - \alpha * \frac{dW}{\sqrt{S_{dw}}} \quad \text{and} \quad b_{\text{new}} = b - \alpha * \frac{db}{\sqrt{S_{db}}}$$

- Thus by dividing by a small  $S_{dw}$  we can increase the magnitude of update in the W direction and while dividing with a large  $S_{db}$  we can dampen the oscillations in the b direction.

## RMSprop



Note : if  $S_{dw}$  is very close to zero in order to avoid the W term becoming very large we add a small term  $\epsilon$  to the denominator

$$W_{\text{new}} = W - \alpha * \frac{dW}{\sqrt{S_{dw} + \epsilon}}$$

# RMS Prop

## **Upsides:**

1. Faster convergence: RMSprop can converge faster than SGD by adapting the learning rate based on the magnitude of the gradients for each parameter.
2. Robustness to different learning rates: RMSprop is more robust to different learning rates for different parameters, which can be helpful in deep learning models with many parameters.
3. Adaptive learning rate: RMSprop adaptively scales the learning rate based on the history of the squared gradients, which can improve the convergence speed and stability of the optimization process

## **Downsides:**

1. RMSProp can sometimes lead to slow convergence.
2. It is sensitive to the learning rate.

# Adam optimizer

Adam or Adaptive moment estimation optimizer is a combination of gradient descent with momentum and the RMSprop optimizer

1. First we initialize  $V_{dw} = 0, V_{db} = 0, S_{dw} = 0, S_{db} = 0$
2. for iteration t :
  - a. Compute  $dW$  and  $db$  for the current mini batch
  - b. Then compute :
    - i. A momentum like

$$\text{upd } V_{dw} = \beta_1 * V_{dw} + (1 - \beta_1) * dW$$

$$V_{db} = \beta_1 * V_{db} + (1 - \beta_1) * db$$

- i. A RMSprop like

$$S_{dw} = \beta_2 * S_{dw} + (1 - \beta_2) * (dW)^2$$

$$S_{db} = \beta_2 * S_{db} + (1 - \beta_2) * (db)^2$$

**Note :** Adam optimization applies bias correction to the first and second moment estimates to ensure that they are unbiased estimates of the true values.

So, the bias corrected terms:

$$V_{dw}^C = \frac{V_{dw}}{1 - \beta_1^t} \quad V_{db}^C = \frac{V_{db}}{1 - \beta_1^t}$$

$$S_{dw}^C = \frac{V_{dw}}{1 - \beta_2^t} \quad S_{db}^C = \frac{V_{db}}{1 - \beta_2^t}$$

# Adam optimizer

Thus the weights will be updated as :

$$W_{\text{new}} = W - \alpha * \frac{V_{dW}^C}{\sqrt{S_{dW}^C} + \epsilon} \quad \text{and} \quad b_{\text{new}} = b - \alpha * \frac{V_{db}^C}{\sqrt{S_{db}^C} + \epsilon}$$

## Adam Configuration Parameters

- **alpha.** Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training
- **beta1.** The exponential decay rate for the first moment estimates (e.g. 0.9).
- **beta2.** The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).
- **epsilon.** Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

Further, learning rate decay can also be used with Adam.

# Adam optimizer

## Advantages :

1. **Adaptive Learning Rates:** Unlike fixed learning rate methods like SGD, Adam optimization provides adaptive learning rates for each parameter based on the history of gradients. This allows the optimizer to converge faster and more accurately, especially in high-dimensional parameter spaces.
1. **Momentum:** Adam optimization uses momentum to smooth out fluctuations in the optimization process, which can help the optimizer avoid local minima and saddle points.
1. **Bias Correction:** Adam optimization applies bias correction to the first and second moment estimates to ensure that they are unbiased estimates of the true values.
1. **Robustness:** Adam optimization is relatively robust to hyperparameter choices and works well across a wide range of deep learning architectures.

# Adam optimizer

## Disadvantages :

**Memory intensive:** Adam needs to store moving averages of past gradients for each parameter during training and hence it requires more memory than some other optimization algorithms, particularly when dealing with very large neural networks or extensive datasets.

**Slower convergence in some cases:** While Adam usually converges quickly, it might converge to flawed solutions in some cases or tasks. In such scenarios, other optimization algorithms like *SGD (stochastic gradient descent)* with momentum or *Nesterov accelerated gradient (NAG)* may perform better.

**Hyperparameter sensitivity:** Although Adam is less sensitive to hyperparameter choices than some other algorithms, it still has hyperparameters like the learning rate, beta1, and beta2. Choosing inappropriate values for these hyperparameters could impact the performance of the algorithm.

# Learning Rate Decay

Learning rate decay is a technique used in machine learning models to train modern neural networks. It involves starting with a large learning rate and then gradually reducing it until local minima is obtained.

→ Say, we are implementing mini batch gradient descent with a relatively small batch size

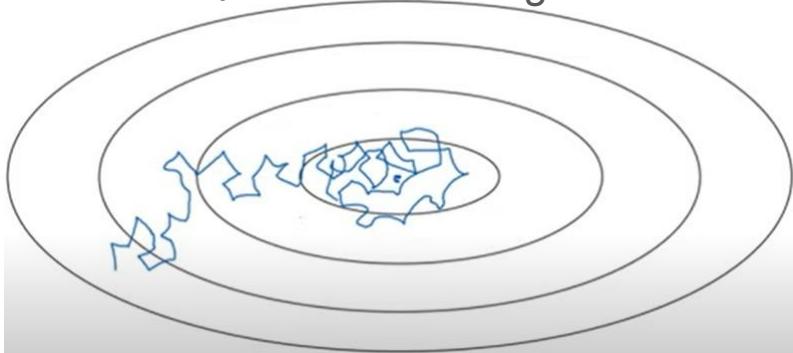
and the gradient takes noisy steps as shown below. It also does not converge at the minima but oscillates around the minimum value.

→ In such a situation slowly reducing the learning rate ( $\alpha$ ) value as we approach the minima is advantageous as the final value would oscillate closer to the minima if

This could be done by setting :

$$\alpha = \frac{1}{1 + \text{decay rate} * \text{epoch number}} * \alpha_0$$

Here decay rate is yet another hyperparameter while is  $\alpha_0$  the initial learning rate value.



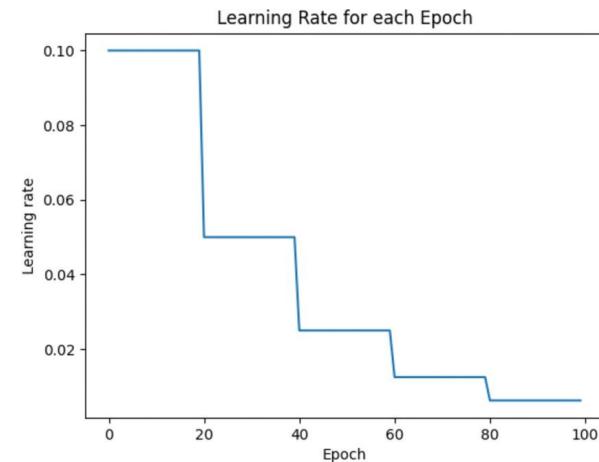
## Other learning rate decay methods

1. Exponential decay:  $\alpha = \text{decay rate}^{\text{epoch number}} * \alpha_0$

2. Epoch number Based decay:

$$\alpha = \frac{\text{constant}}{\sqrt{\text{epoch number}}} * \alpha_0$$

3. Discrete Step decay : In this method learning rate is decreased in some discrete steps after every certain interval of time , for example you are reducing learning rate to its half after every 10 secs..



4. Manual decay : In this method practitioners manually examine the performance of algorithm and decrease the learning rate manually day by day or hour by

# Resources

---

- <https://www.scaler.com/topics/momentum-based-gradient-descent/>
- <https://medium.com/mlearning-ai/exponentially-weighted-average-5eed00181a09>
- <https://medium.com/nerd-for-tech/optimizers-in-machine-learning-f1a9c549f8b4>
- <https://www.analyticsvidhya.com/blog/2021/03/variants-of-gradient-descent-algorithm/>
- <https://machinelearningmastery.com/gradient-descent-with-momentum-from-scratch/>
- <https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/>
- [https://youtube.com/playlist?list=PLkDaE6sCZn6Hn0vK8co82zjQtT2Nkqc&si=7Z\\_y1LQXzn7fZF1I](https://youtube.com/playlist?list=PLkDaE6sCZn6Hn0vK8co82zjQtT2Nkqc&si=7Z_y1LQXzn7fZF1I)
- <https://arxiv.org/pdf/1412.6980.pdf>



**PES**  
UNIVERSITY

**Thank You**

---



**PES**  
**UNIVERSITY**

CELEBRATING 50 YEARS

# Machine Learning

**Surabhi Narayan**

---

Department of Computer Science and Engineering

**Teaching Assistant : V. Guna Chowdary**

# **Machine Learning**

---

## **Support Vector Machines**

**Surabhi Narayan**

Department of Computer Science and Engineering

# Machine Learning

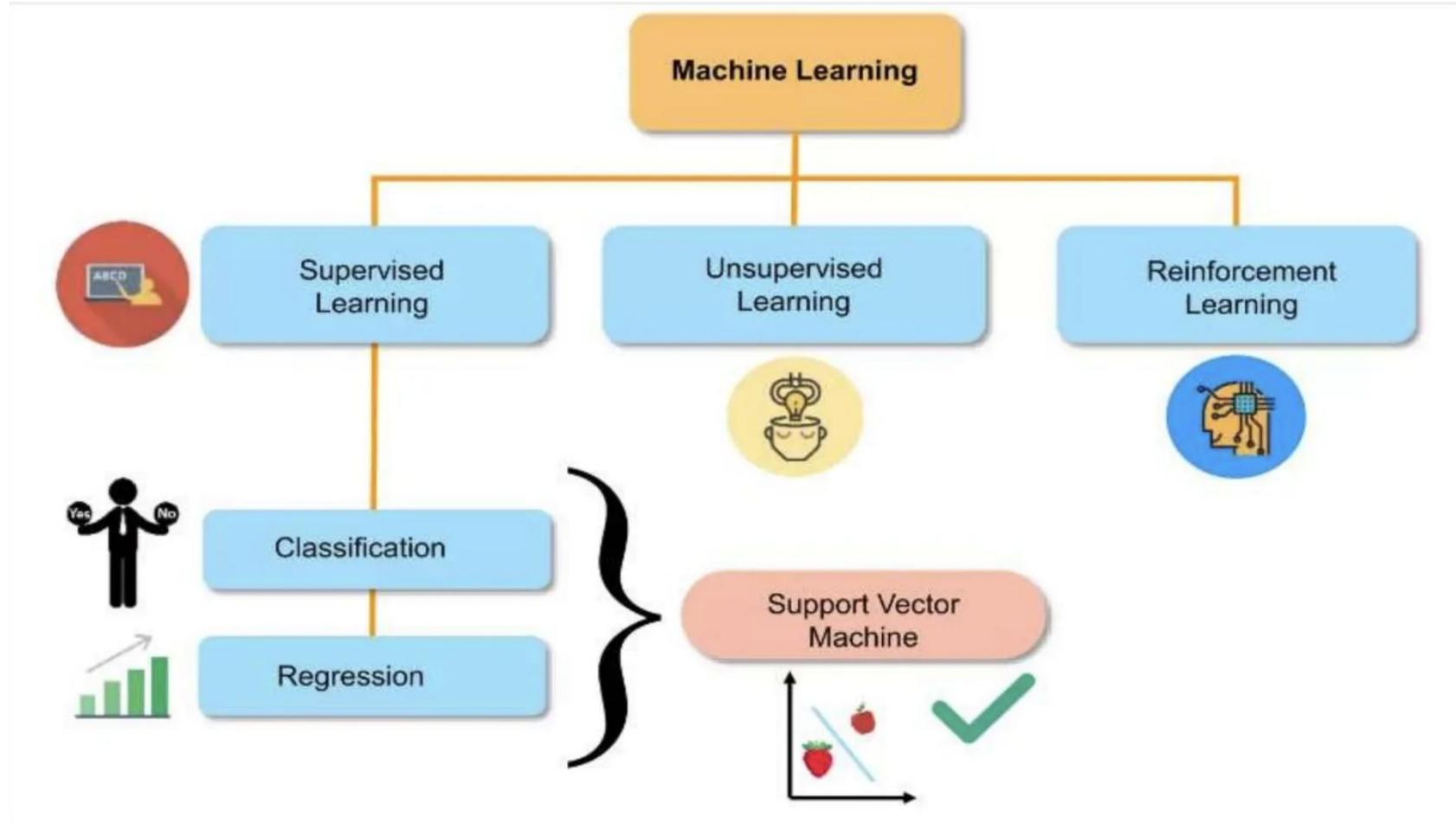
## Acknowledgement

---

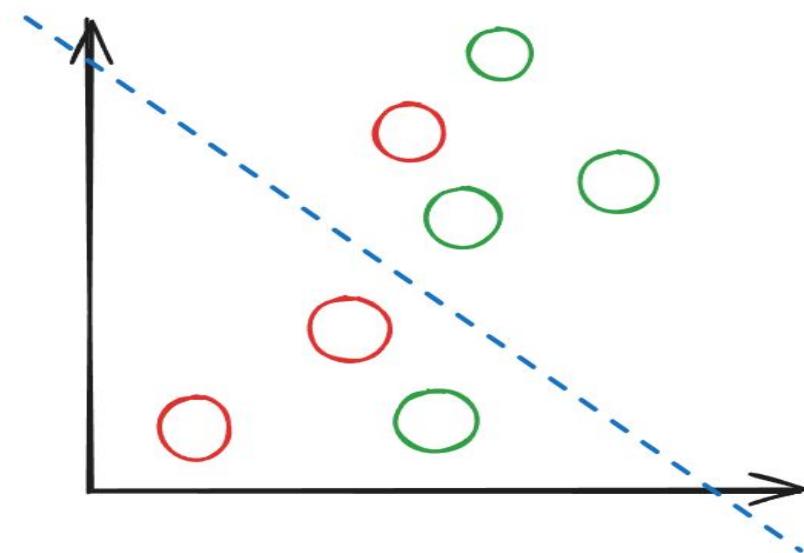


- Heartfelt gratitude to Dr. Rajnikanth who has instilled a deep insight among the faculty in Machine Learning
- Department of CSE for conducting the train the trainer programs multiple times
- Faculty for their contribution towards syllabus and course content preparation
- Wonderful Teaching Assistants who have worked tirelessly towards smooth conduction of the course

## Support Vector Machines : Introduction

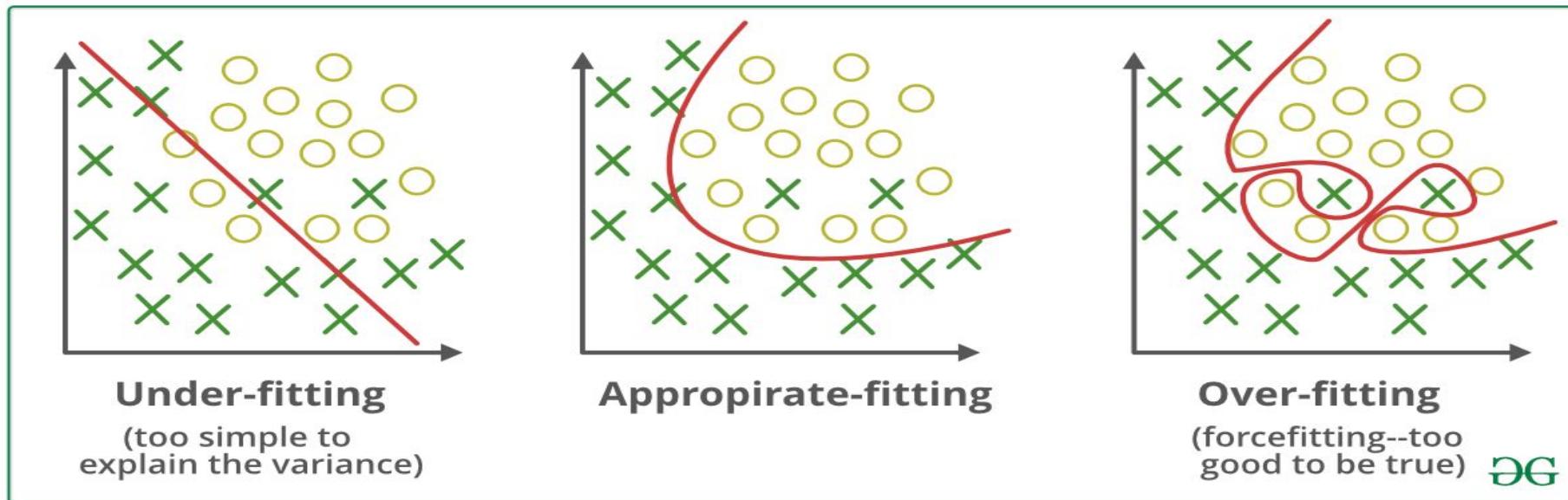


Suppose you have a 2D plane with dots representing data points. Some dots are positive (e.g., green) and some are negative (e.g., red). You want to draw a line that best separates the green dots from the red dots.



## SVM: Why SVM? Neural Networks

- Complex and slow: Lot of computational power and time to get the line just right.
- Overfitting: might draw a line that fits perfectly to the training dots, including any noise, but fails to generalize well to new dots.
- Data hungry: needs a lot of dots (data) to learn from, which you might not always have.

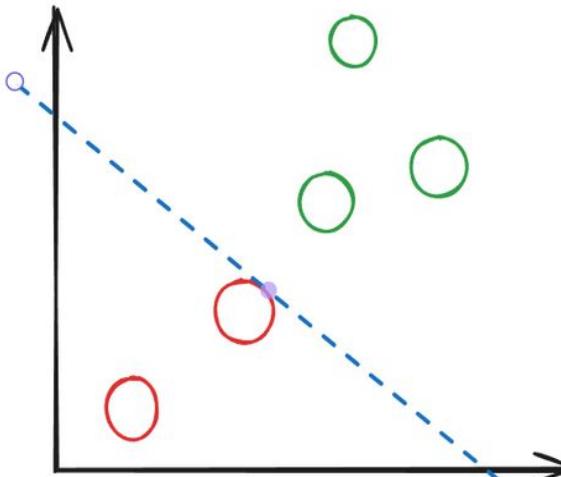


- Overfitting: tree might make too many specific decisions, creating a very jagged line that fits the training dots too well but doesn't work for new dots.
- Greedy Decisions: makes decisions step-by-step, choosing the best split at each step, which can lead to suboptimal overall separation.
- Bias: might favor certain splits that don't necessarily help in drawing the best overall line.

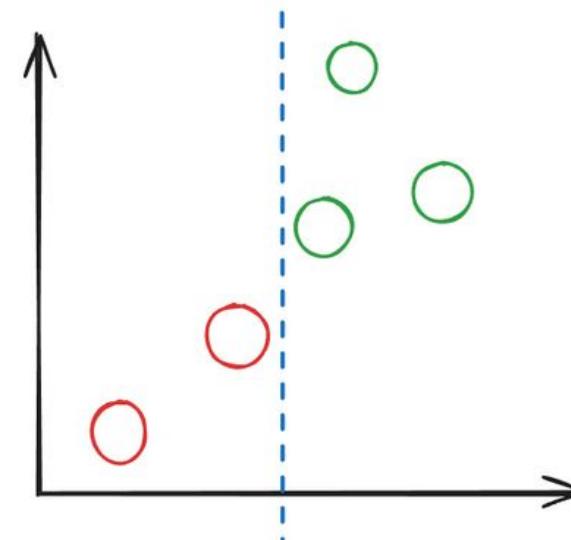
Thus, we need a better way to separate the green and red dots. We need a method that:

- Efficiently finds the best line.
- Avoids making too specific decisions (overfitting).
- Considers the overall best separation.

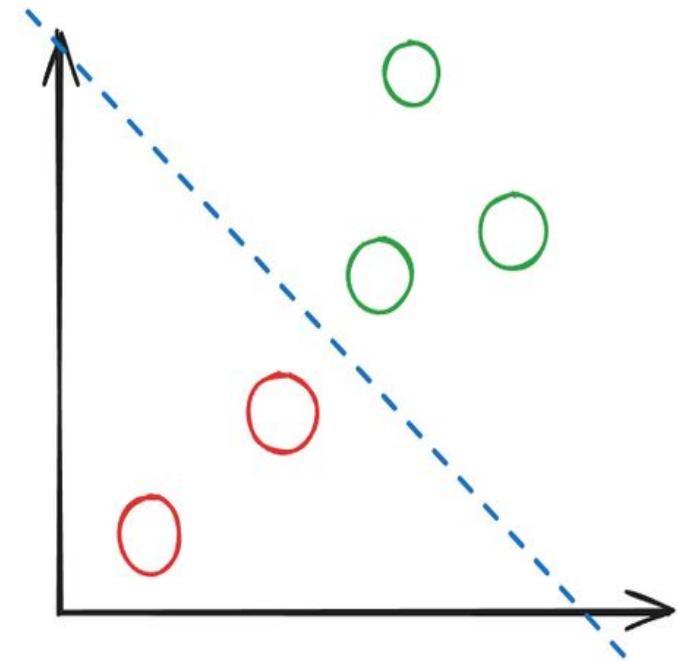
## SVM: Why SVM?



Separates the samples but too close to the negative samples.

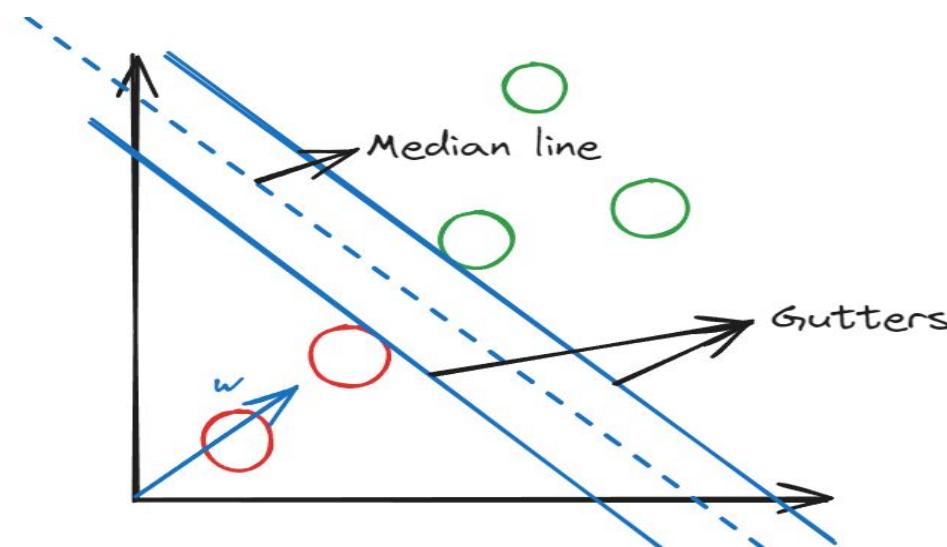


Probably not the best way to separate samples.



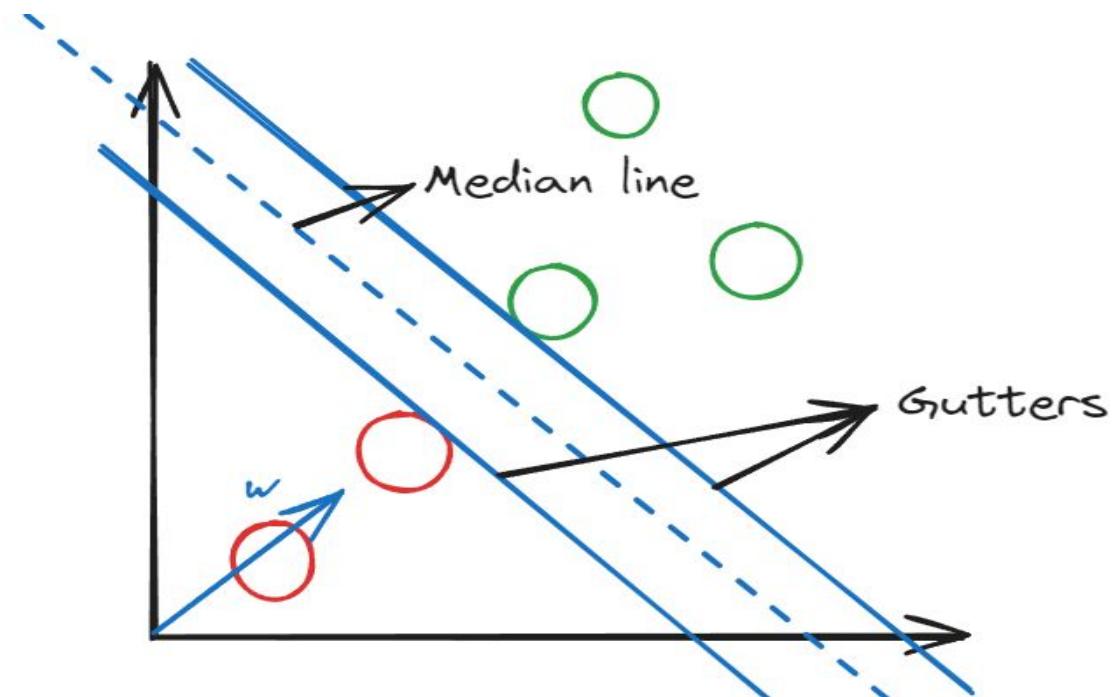
seems like the best way to separate the samples.  
(Samples closest to the decision boundary are equally spaced from it.)

- It is a supervised machine learning algorithm that helps in both classification and regression problems. It finds an optimal boundary known as hyperplane between different classes
- It is a vector-space-based machine-learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data (possibly discounting some points as outliers or noise)



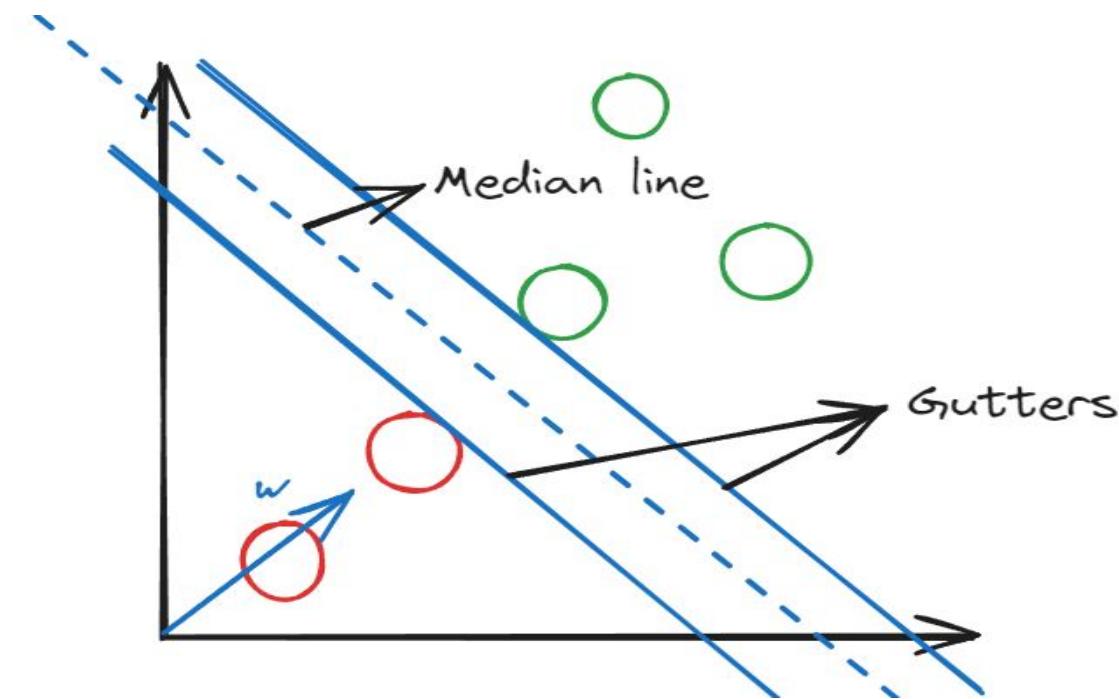
Here  $w$  is a vector of any length that is perpendicular to the median line.

Our purpose is to maximize the distance between the gutters.

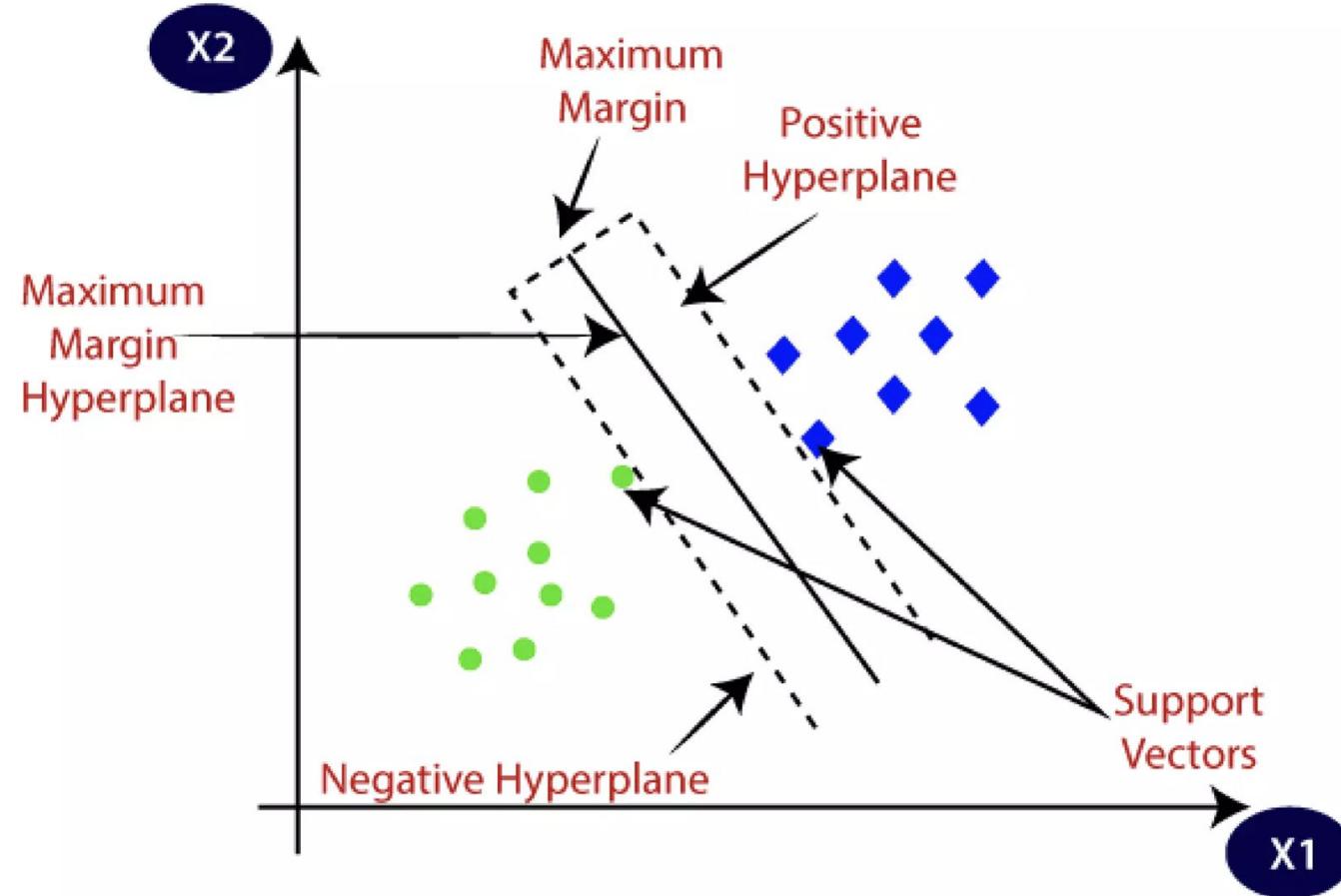


## Linearly Separable Case

- If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.
- The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane is the hyperplane that lies halfway between them.

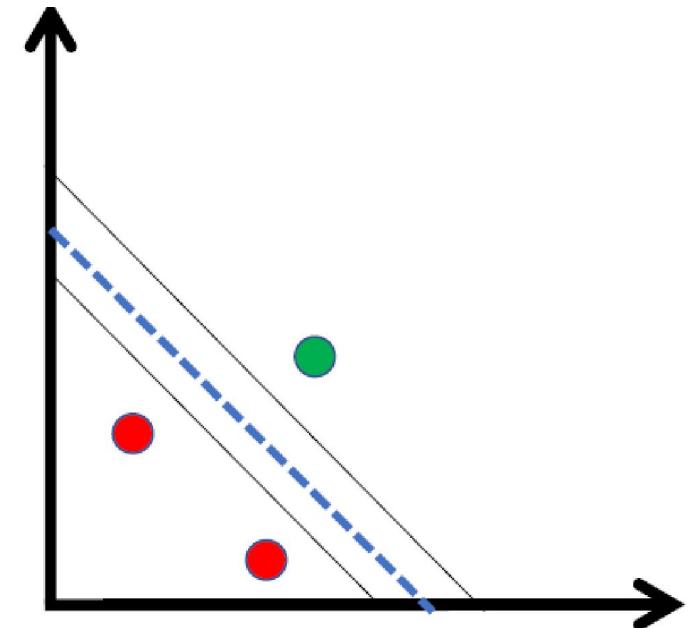


## Linearly Separable Case



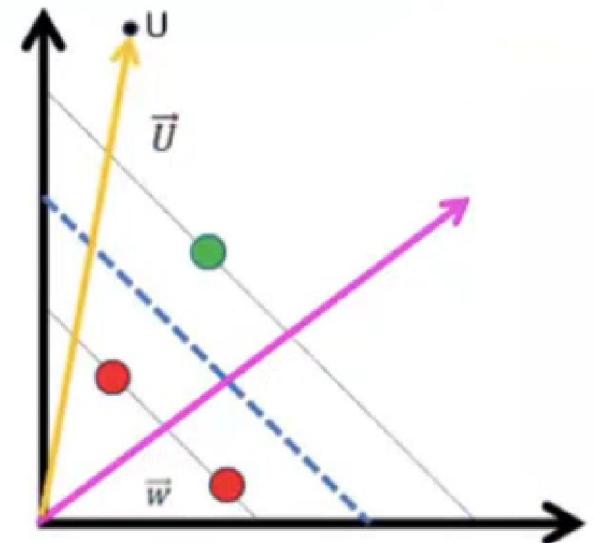
## Linearly Separable Case

- Assume we have labeled binary data
- We want to find a hyper plane that separates the samples into its respective class
- Hyper plane should have a margin as wide as possible



## Linearly Separable Case

- Let us imagine that by some magic we got that decision boundary
- We need a rule to classify the points.
- Take a vector of any length that is perpendicular to the gutter/margin
- Consider an unknown point  $U$  and a vector  $\vec{U}$  cap that points to it
- to find if  $U$  belongs to class 1 or class 2, project  $U$  on to the vector that is perpendicular to the margin



## Linearly Separable Case

- If

$$\vec{W} \cdot \vec{U} \geq c \quad +ve\ sample$$

$$\vec{W} \cdot \vec{U} - C \geq 0 \quad +ve\ sample$$

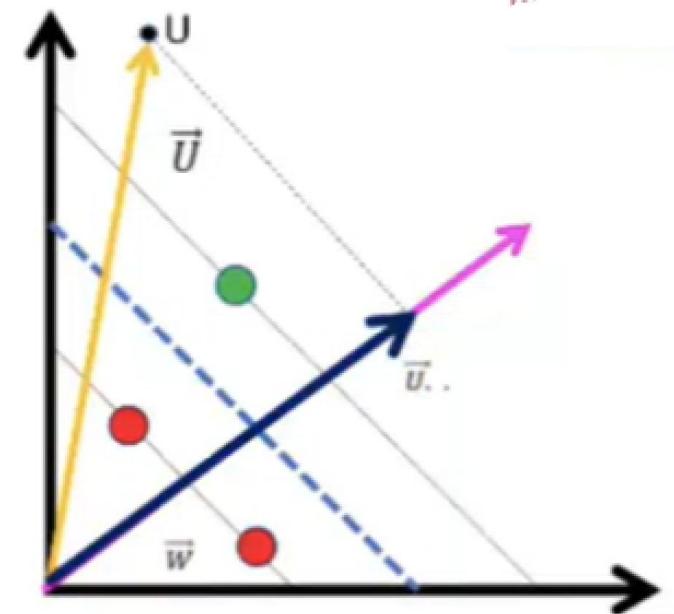
C is the length of the vector from origin

$$\vec{W} \cdot \vec{U} + b \geq 0 \quad +ve\ sample$$

**Decision Rule (first understanding)**

$$b = -C$$

We do not know what is w and b



## Linearly Separable Case

- so now if we take the same perpendicular w and dot with some positive sample X+

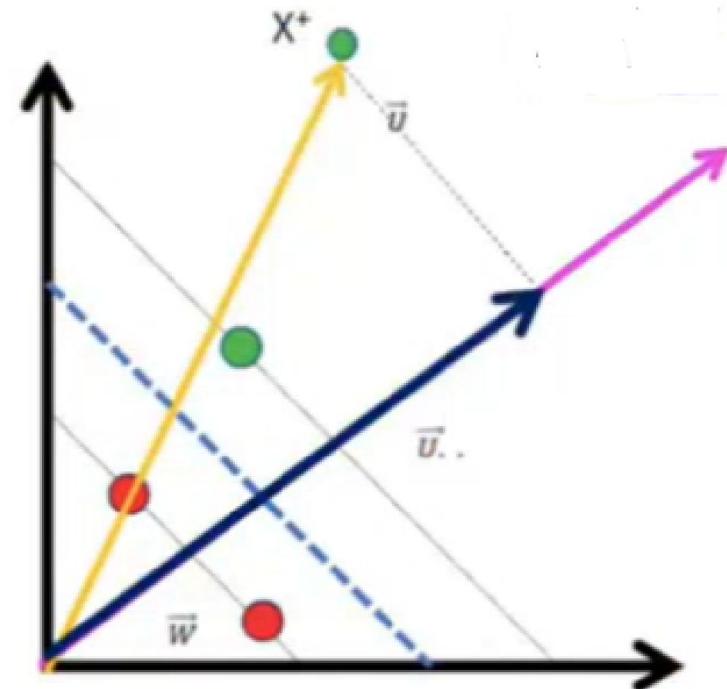
- Here we insist that

$$\vec{W} \cdot \vec{X^+} + b \geq 1$$

- In other words, with a positive sample we are going to insist that the decision function gives the value of 1 or greater

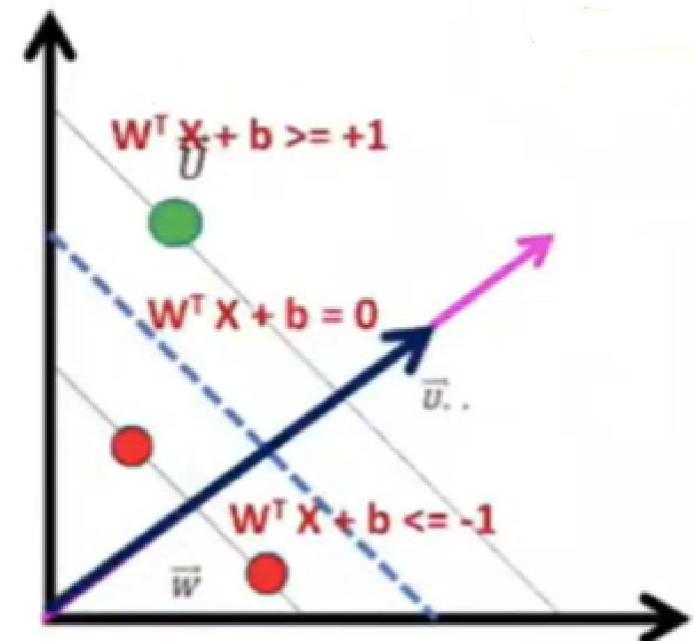
- Like wise with W dot negative sample

$$\vec{W} \cdot \vec{X^-} + b \leq -1$$



## Linearly Separable Case

- Two hyper planes:  $W^T X + b \geq +1$  and  $W^T X + b \leq -1$
- Evidently, the Decision Boundary is mid-way, so it must be defined by  $W^T X + b = 0$
- The three hyper planes are parallel to each other  $\rightarrow W$  can be same.  
(The general equation for a hyper plane is  $W^T X + b = 0$  where  $W$  is a vector that is normal to the hyper plane)
- How can we be sure that we will be able to get a common  $b$  also?
- Simple manipulation of the equations does allow us to get the equations for the supporting hyper planes in the required form!



## Linearly Separable Case

---

- Let the hyper planes be represented by general equations:  $W^T X + b_1 = 0$  and  $W^T X + b_2 = 0$
- Let  $C = (b_1 + b_2) / 2$  and  $D = (b_1 - b_2) / 2$
- so,  $b_1 = (C + D)$  and  $b_2 = (C - D)$
- Hyper planes:  $W^T X + (C + D) = 0$  and  $W^T X + (C - D) = 0$   
 $W^T X + C = -D$  and  $W^T X + C = D$   
 $(1/D) W^T X + (C / D) = -1$  and  $(1/D) W^T X + (C / D) = 1$   
Let  $(1 / D) W = M$  and  $(C / D) = b$
- Hyper planes are:  $M^T X + b = -1$  and  $M^T X + b = 1$
- Renaming the weight vector, we get the equations for the hyper planes in the desired form

$$W^T X + b = -1$$

$$W^T X + b = +1$$

# Stochastic Models and Machine Learning

## SVM

Introduce  $y_i$ , such that

- $y_i$  such that  $y_i = +1$  for +ve sample
- $y_i = -1$  for -ve sample

$$\vec{w} \cdot \vec{x} + b \geq 1$$

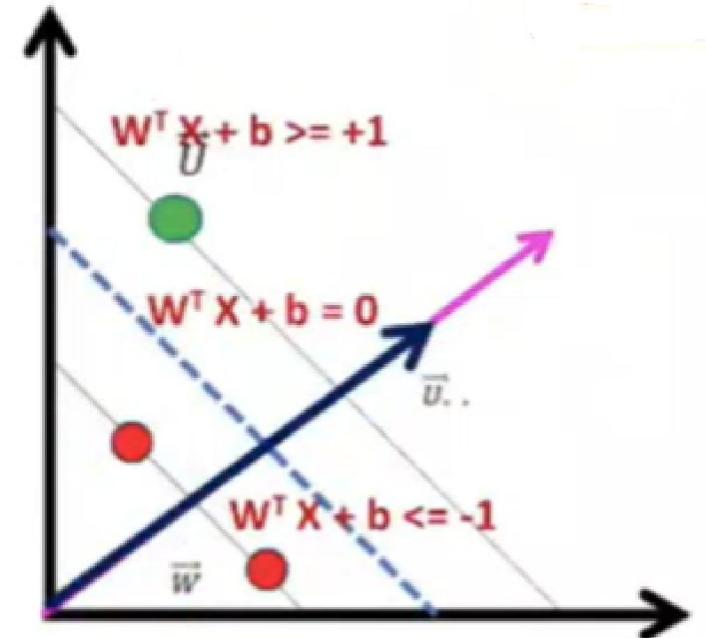
$$\vec{w} \cdot \vec{x} + b \leq -1$$

$$-(\vec{w} \cdot \vec{x}) - b \geq 1$$

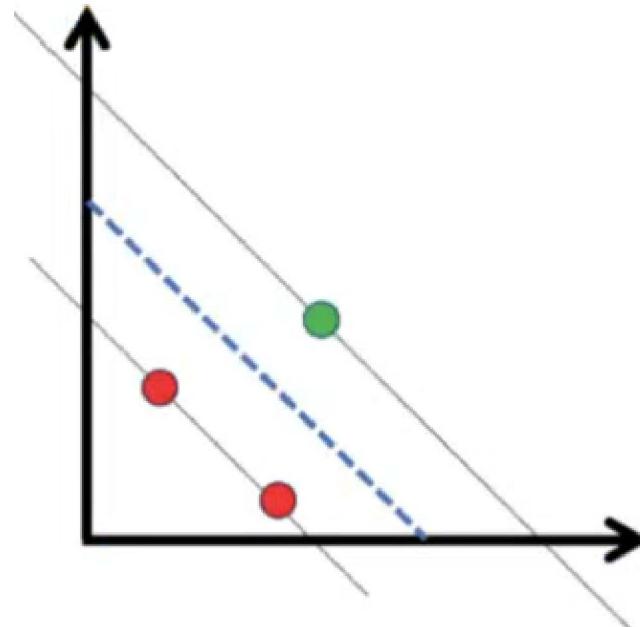
$$-1(\vec{w} \cdot \vec{x} + b) \geq 1$$

For negative sample

$$y(\vec{w} \cdot \vec{x} + b) \geq 1$$



We get the same equation...  
One equation to be solved



$$y_i(\vec{W} \cdot \vec{X} + b) \geq 1$$

$$y_i(\vec{W} \cdot \vec{X} + b) - 1 \geq 0$$

$y_i(\vec{W} \cdot \vec{X} + b) - 1 = 0$  , for  $\vec{X}_i$  in gutter

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0$$

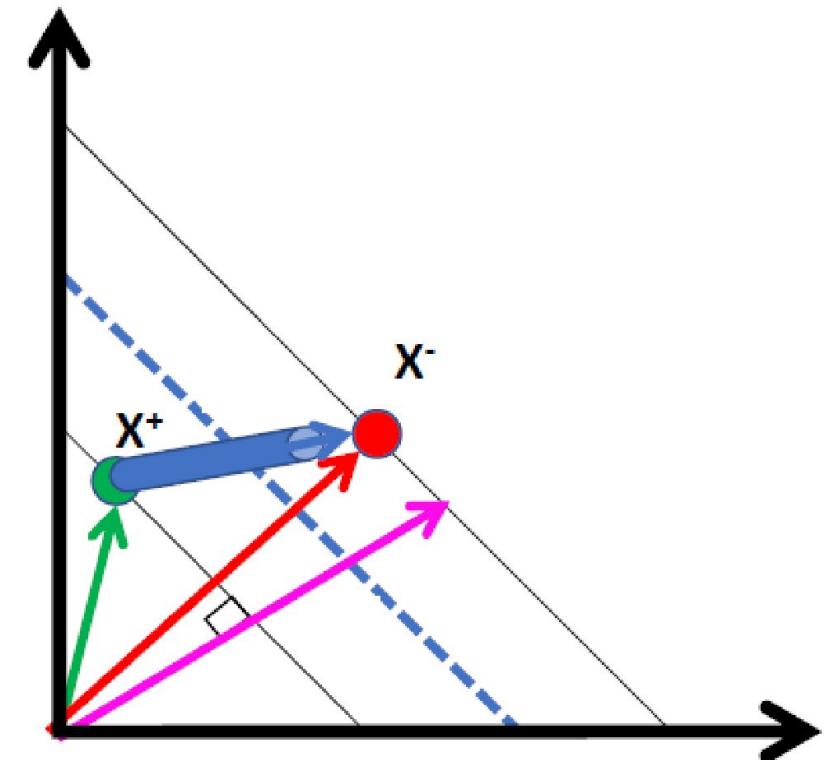
Combined decision rule

If you remember, our purpose was to figure out a way to maximize the distance between the two gutters. For this we need an equation that expresses the distance between the two gutters.

If a point is exactly on the gutter ( or in the gutter ), the decision rule will be,

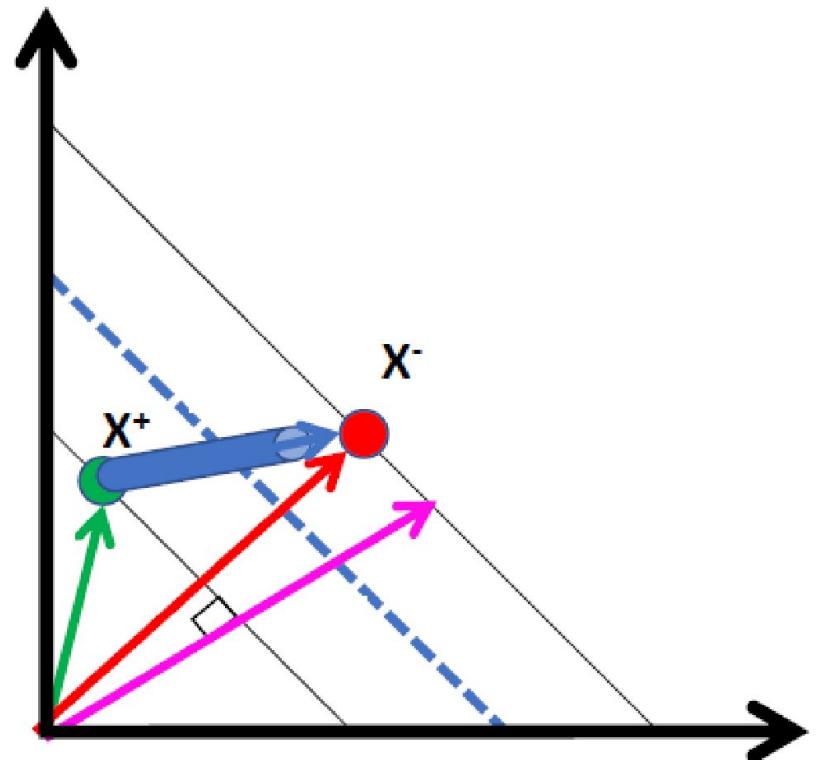
$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

- We have an  $x^-$  on one margin and an  $x^+$  on the other margin. →
- The difference between these 2 vectors is  $(x^+ - x^-)$  as indicated by the blue, when projected onto the vector  $W$ , would indicate the width of the street.



- We known that for + support vector the equation ( $y_i=+1$ )
- We known that for - support vector the equation ( $y_i=-1$ )
- What we want to compute is

$$\frac{\vec{W}}{\|W\|} \cdot (\vec{X}^+ - \vec{X}^-) = \text{width}$$



- Width =  $(W / ||W||) \cdot (X^+ - X^-)$ ; Let us name  $W / ||W||$ , the unit vector in the direction of  $W$  as  $w$ .
- Width =  $w \cdot X^+ - w \cdot X^-$
- $W \cdot X^+ + b = 1 \rightarrow W \cdot X^+ = (1 - b) \rightarrow (W / ||W||) \cdot X^+ = (1 - b) / ||W|| \rightarrow w \cdot X^+ = (1 - b) / ||W||$
- $W \cdot X^- + b = -1 \rightarrow W \cdot X^- = (-1 - b) \rightarrow (W / ||W||) \cdot X^- = (-1 - b) / ||W|| \rightarrow w \cdot X^- = (-1 - b) / ||W||$
- So, Width =  $w \cdot X^+ - w \cdot X^- \rightarrow \text{Width} = \{(1 - b) / ||W||\} - \{(-1 - b) / ||W||\}$
- Thus, Width =  $(1 - b + 1 + b) / ||W|| = 2 / ||W||$
- Width =  $2 / ||W||$

### Alternative approach to deriving Width

- The normal distance from Origin to a hyper plane that is defined by  $\mathbf{W}^T \mathbf{X} = b$  is:  
 $b / ||\mathbf{W}||$

- $\mathbf{W}^T \mathbf{X} + b = +1$  i.e.,  $\mathbf{W}^T \mathbf{X} = 1 - b$ ;     $\mathbf{W}^T \mathbf{X} + b = -1$  i.e.,  $\mathbf{W}^T \mathbf{X} = -1 - b$ ;

→

$$OM = (1-b) / ||\mathbf{W}||$$

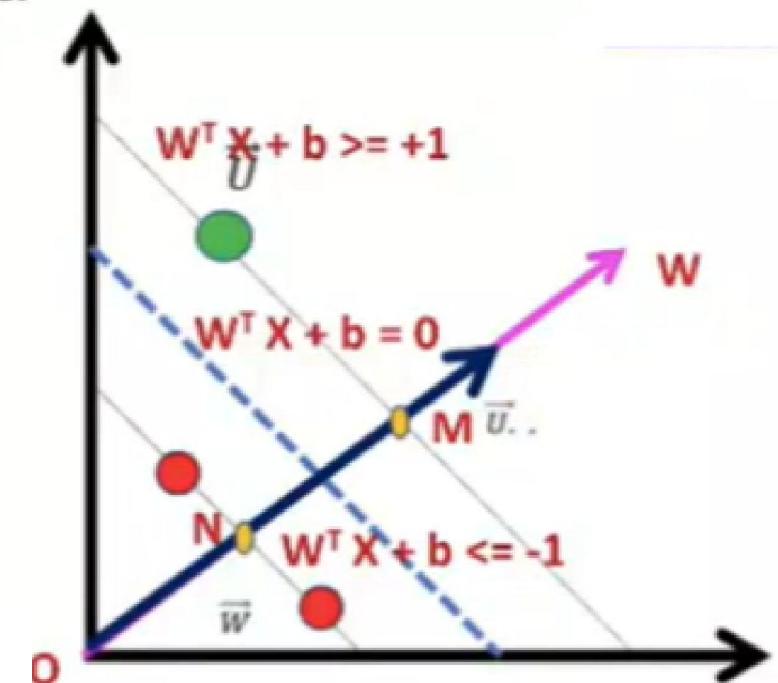
$$ON = (-1-b) / ||\mathbf{W}||$$

$$NM = OM - ON$$

(OM, ON, and NM are scalars, not vectors!)

- Width of the Channel =  $NM = \{ (1-b) / ||\mathbf{W}|| \} - \{ (-1-b) / ||\mathbf{W}|| \}$

$$= (1 - b + 1 + b) / ||\mathbf{W}|| = 2 / ||\mathbf{W}||$$



So finally, the width is expressed as

$$\frac{2}{|w|}$$

And this should be maximized, so we can also say that its reciprocal should be minimized.

$$\frac{1}{|w|} \Rightarrow |w|$$

Maximizing  $2/w$  is the same as maximizing  $1/w$  which is the same as minimizing  $w$

Minimizing  $w$  would be the same as minimizing,

$$|\omega| = \frac{1}{2} |\omega|^2$$

(we do this for it to become easier to differentiate later)

- so we got an expression here of which we would like to find minimum/extremum of
- and some constraints to honour

$$\begin{aligned} & \min \frac{1}{2} ||W||^2 \\ \text{s.t } & y_i(\vec{w} \cdot \vec{x}_i + b) - 1 \geq 0 \end{aligned}$$

- so what do we do ????
- we use Lagrange Multipliers

## So what is Lagrange Multipliers

Let us say we wish to maximize the function value  $f(x_1, x_2)$  subject to a single equality constraint  $g(x_1, x_2) = c$ .

**Max  $f(x_1, x_2)$**

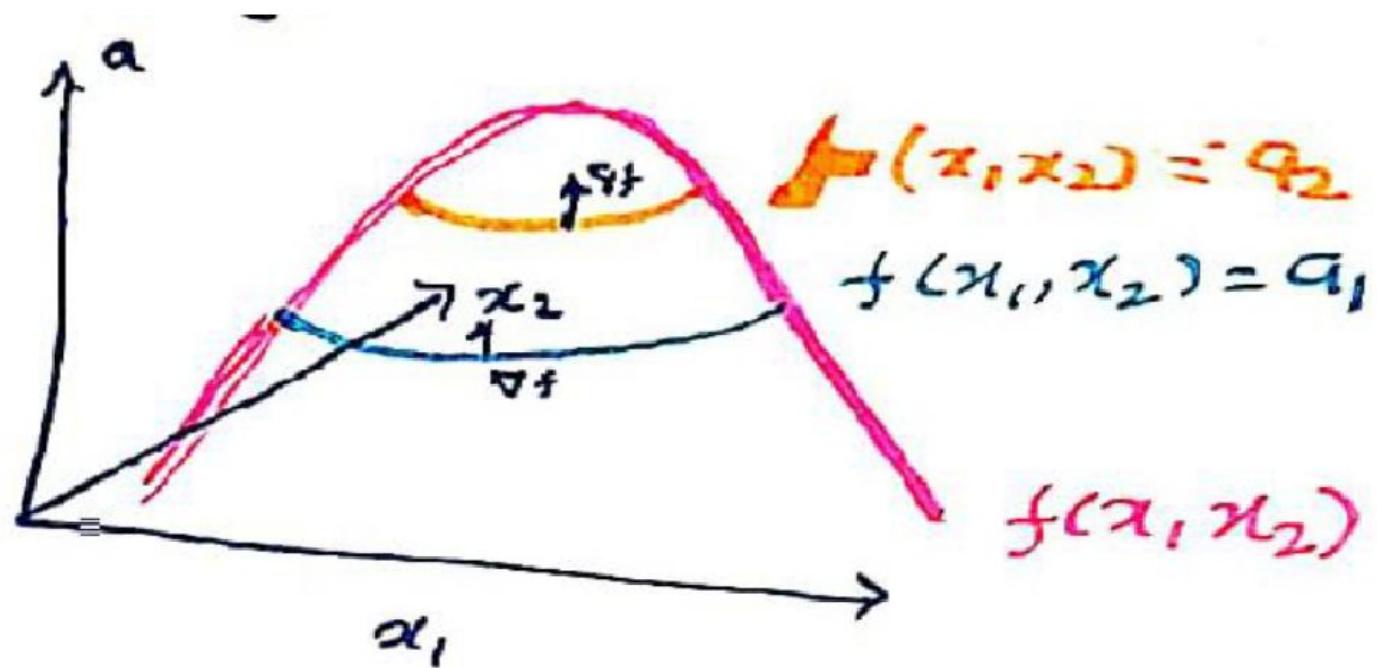
**subject to constraint  $g(x_1, x_2) - c = 0$  [Popular way of writing the constraint]**

(Remember in our SVM problem is a minimization problem with multiple inequality constraints ( $\geq 0$ )

However, for minimization same ideas will follow )

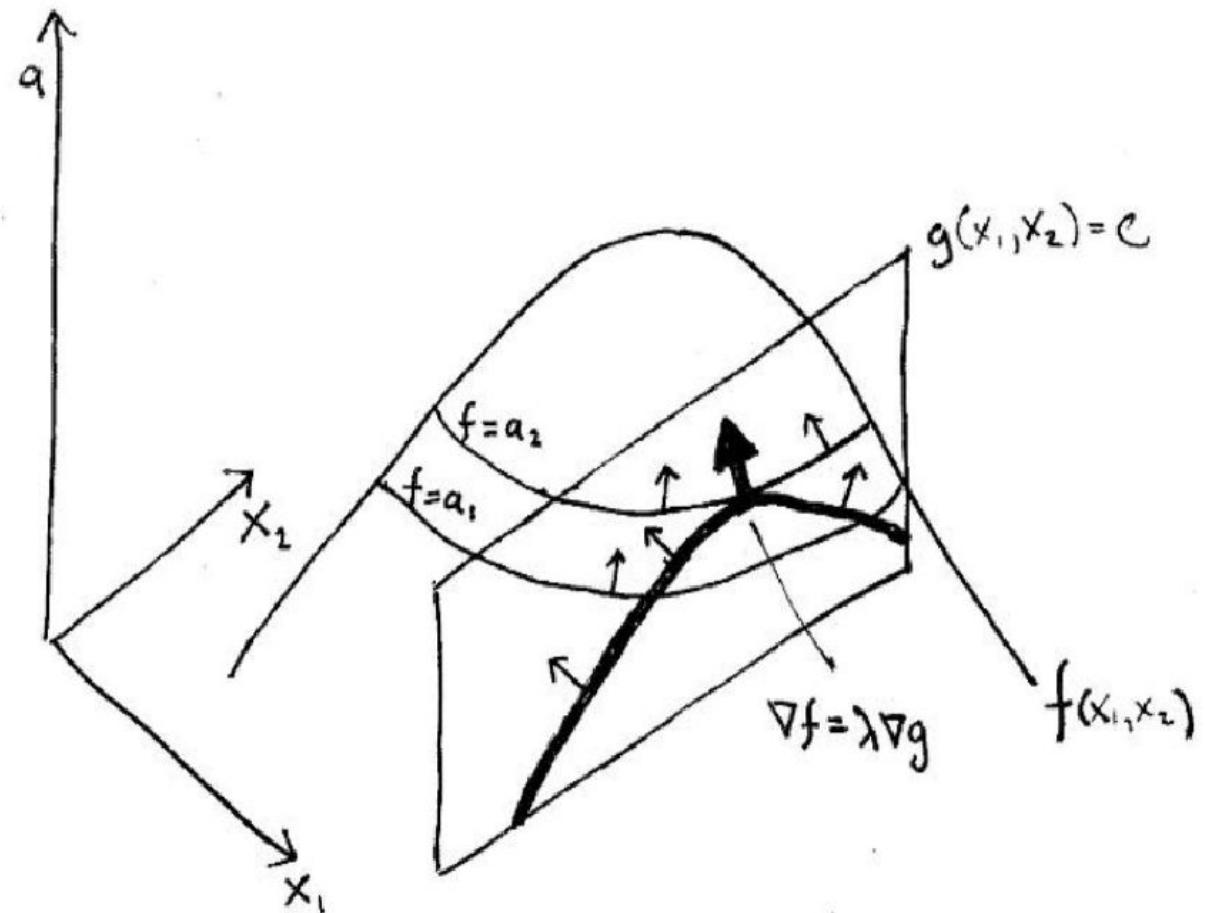
## Lagrange Multiplier

- Let's say there is a function  $f(x_1, x_2)$  that we wish to maximize subject to a function  $g(x_1, x_2) = c$
- Please note the equality constraint which will need to be relaxed later.
- We need to take the  $\nabla f$  which gives us the steepest direction
- Let us visualize the plot  $f = a_1$  all along the blue line and  $f=a_2$  along the orange line
- The function  $f$  being represented by the pink line



## Linearly Separable Case

- We start climbing the hill moving from left to right on the constraint curve
- At each point we check  $\nabla g$  and check  $\nabla f$
- When  $\nabla g > \nabla f$  we keep moving
- At a point where  $\nabla g = \nabla f$  we have reached the maximum
- Any further move will result in declining  $f$



## Lagrange Multipliers

---

We know, gradient of a function, points at the direction of steepest ascent. So in order to maximize  $f(x_1, x_2)$  we must move in the direction of the gradient.

While moving along the gradient, we cannot leave the constraint surface. Throughout the constraint surface, i.e. for any point on the constraint surface,  $g(x_1, x_2) - c = 0$ .

We can say, all the points on the constraint surface form a level surface(because value  $g(x_1, x_2) - c = 0$  for any point  $x_1, x_2$  on the constraint surface..

The optimal point (where value of  $f(x_1, x_2)$  is maximum must remain on the constraint surface.

That means for that optimal point  $(x_1^*, x_2^*)$  also, the value of  $g(x_1^*, x_2^*) - c = 0$ .

Let us take an example :

**maximize :  $f(X)$**

**subject to constraint :  $g(X) = 0$**

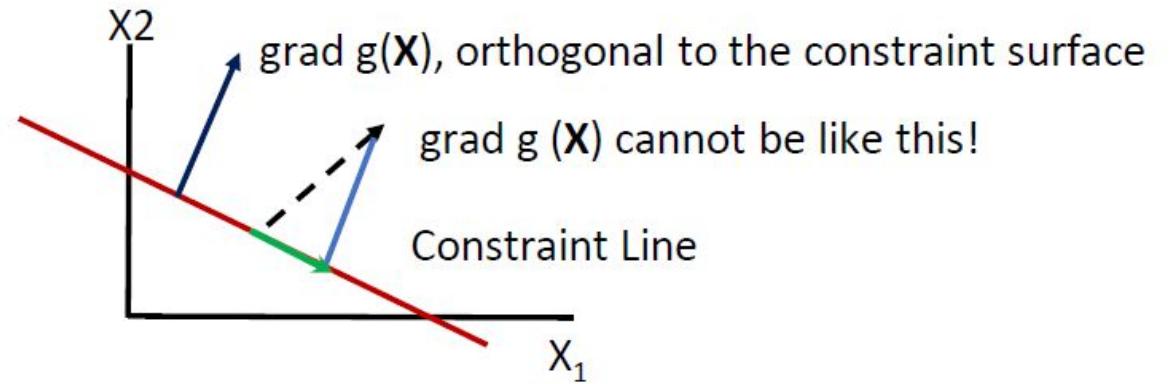
(equality constraint only for now, inequalities will be considered soon)

Assume  $X$  is  $d$ -dimensional, then  $g(X) = 0$  is a  $d - 1$  dimensional surface.

Let us take an example :

**maximize :  $f(\mathbf{X})$**

**subject to constraint :  $g(\mathbf{X}) = 0$**



Gradient  $\nabla g(\mathbf{X})$  is orthogonal to the constraint surface at any point on the constraint surface.

Else, projection of  $\nabla g(\mathbf{X})$  onto the constraint surface would be non-zero, implying the value of  $g(\mathbf{X})$  would increase moving along the direction of that projection. This is not possible, as  $g(\mathbf{X})$  remains unchanged on the constrained surface.

Let us take an example :

**maximize :  $f(\mathbf{X})$**

$\mathbf{X}$  is a vector of input values ( $x_1, x_2, x_3, \dots, x_n$ )

**subject to constraint :  $g(\mathbf{X}) = 0$**

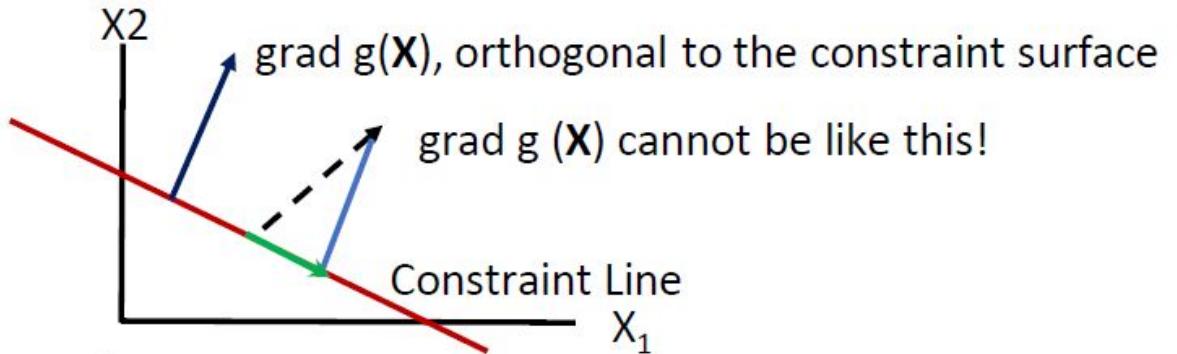
Let  $\mathbf{X}$  and  $\mathbf{X} + \boldsymbol{\epsilon}$  be nearby points on the constraint surface.

$\boldsymbol{\epsilon}$  is a vector located wholly on the constraint surface.

Using Taylor Series,

$$g(\mathbf{X} + \boldsymbol{\epsilon}) \approx g(\mathbf{X}) + \boldsymbol{\epsilon}^T \nabla g(\mathbf{X})$$

But,  $g(\mathbf{X} + \boldsymbol{\epsilon}) = g(\mathbf{X}) \Rightarrow \boldsymbol{\epsilon}^T \nabla g(\mathbf{X}) \approx 0 \Rightarrow \nabla g(\mathbf{X})$  is orthogonal to the constraint surface  $g(\mathbf{X}) = 0$ .



## Lagrange Multipliers

---

Let us take an example :

**maximize :  $f(X)$**

**subject to constraint :  $g(X) = 0$**

Let  $X^*$  be the point on the constraint surface  $g(X) = 0$  which maximizes  $f(X)$ .

Following the same logic, we conclude  $\nabla f(X^*)$  is orthogonal to the constraint surface.

Else, projection of  $\nabla f(X^*)$  onto the constraint surface would be non-zero, implying the value of  $f(X)$  would increase moving along the direction of that projection. This is not possible, as we are claiming  $X^*$  is the optimal point where max of  $f(X)$  occurs.

Thus at  $X^*$  the  $\nabla g(X^*)$  and  $\nabla f(X^*)$  are parallel to each other. (same or opposite direction)

## Lagrange Multipliers

---

Since at optimal point  $X^*$  the  $\nabla g(X^*)$  and  $\nabla f(X^*)$  are parallel to each other, we can say they are scalar multiples of one another i.e.,

$$\nabla f(X^*) = \lambda \nabla g(X^*)$$

where  $\lambda$  is a scalar and is called Lagrange Multiplier, **could be +ve or ve but not zero.**

Lagrange wrote down a special new function which takes in all the same input variables as  $f$  and  $g$ , along with  $\lambda$ , thought of now as a variable rather than just a constant/scalar:

$$L(X, \lambda) = f(X) - \lambda g(X)$$

find  $X^*$  by partially differentiating  $L$  wrt to  $X$  and  $\lambda$  and set it equal to zero.

We will have  $d+1$  equations,  $d+1$  unknowns.

if not interested in value of  $\lambda$ , do not solve for it. (Method of Undetermined Multipliers!!)

## Lagrange Multipliers

---

Few sources use the equation as:

$$L(X, \lambda) = f(X) - \lambda g(X)$$

Few sources use the equation as:

$$L(X, \lambda) = f(X) + \lambda g(X)$$

Both ways are fine.

## Lagrange Multipliers

---

Multiple constraints can be handled in a similar fashion by introducing the required number of Lagrange Multipliers. (One for each constraint.)

$$L(X, \lambda) = f(X) - \lambda_1 g_1(X) - \lambda_2 g_2(X) - \lambda_3 g_3(X) - \dots - \lambda_k g_k(X)$$

Function minimizations works in the same way.

We are finding a **stationary point** only when we take first order derivatives! To check that it is maximum or minimum, we need second-order derivatives! We will not get in to that!!!

We are still dealing with only Equality Constraints. We will look at Inequality constraints next.

- Assume the inequality constraint is of the form:  $g(X) \geq 0$  [like in our SVM problem]
- Lagrangian is still:  $L(X, \lambda) = f(X) - \lambda g(X)$
- Constrained stationary point  $X^*$  may lie in the region of  $g(X) > 0$  or it may lie on the surface of  $g(X) = 0$ .

(We do not as yet know which is the case! Let us consider each possibility.)

## Lagrange Multipliers

---

Case 1:  $X^*$  is in the region  $g(X) > 0$ :

- The constraint is **inactive!** We need to solve only  $\nabla f(X) = 0$ .
- Same as solving the Lagrangian with  $\lambda$  set to 0.  
(As  $\lambda$  is set to 0,  **$\lambda * g(X)$  will also become 0.**)

$$L(X, \lambda) = f(X)$$

Case 2:  $X^*$  is on the surface  $g(X) = 0$ :

- Same as the case discussed earlier i.e., solve the Lagrangian with  $\lambda$  not equal to 0.
- $\nabla g(X)$  is orthogonal to the surface  $g(X) = 0$  as discussed earlier.
- Again,  $\nabla f(X)$  must be orthogonal to the surface  $g(X) = 0$  but, crucially, it must be opposite in direction from the region  $g(X) > 0$ . (Else,  $f(X)$  can be increased by moving in to the region of  $g(X) > 0$  which we considered as Case 1.)
- Hence  $\nabla f(X) = -\lambda \nabla g(X)$ , with  $\lambda > 0$ .  
( $\nabla g(X) = 0$ ; so, still  **$\lambda \text{ grad } g(X) = 0$ .**  $\lambda * g(X) = 0$ )

## Lagrange Multipliers

- We must solve the Lagrangian as earlier (Partial derivatives of L wrt X as well as  $\lambda$  must be set to 0 and the resulting D+1 equations must be solved) but with some additional constraints:

$$\begin{aligned} g(X) &\geq 0; \\ \lambda &\geq 0; \\ \lambda g(X) &= 0 \end{aligned}$$

- For points on the Constraint Surface,  $\lambda$  is  $> 0$  whereas for points above the surface, constraint is inactive and  $\lambda = 0$ . [DUAL FEASIBILITY CONDITION]
- For points on the Constraint Surface,  $\lambda g(X) = 0$  because  $g(X) \neq 0$ . For points above the constraint surface,  $\lambda g(X) = 0$  because  $\lambda = 0$  (constraint is inactive) [COMPLEMENTARY SLACKNESS]

**Karush – Kuhn – Tucker (KKT) Conditions.**

**Multiple constraints are handled as discussed already but with KKT conditions added.**

## Lagrange Multipliers

- We saw previously that the SVM optimization problem is:

$$\underset{w, b}{\text{minimize}} \frac{1}{2} \|w\|^2$$

$$\text{subject to } y_i(\vec{W} \cdot \vec{X}_i + b) - 1 \geq 0 \quad i = 1, \dots, m$$

- We have one objective function to minimize:

$$f(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2$$

- Subject to  $m$  constraint functions:

$$g(w, b) = y_i(\vec{W} \cdot \vec{X}_i + b) - 1, \quad i=1, \dots, m$$

- We introduce the Lagrangian function:

$$L = f(w) - \sum_{i=1}^m \alpha_i g_i(w, b)$$

$$L = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i [y_i(\vec{W} \cdot \vec{X}_i + b) - 1]$$

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i(w^T x_i + b) - 1)$$

So now we know that W is going to be linear sum of **SOME** of the vectors (samples)

Why did I tell **SOME** ??

Because those points/instances which do not lie on gutter ,they will have Lagrangian multiplier=0  
and those vectors who have Lagrangian Multiplier !=0 will lie on gutter and **guess what???**

**THESE ARE CALLED SUPPORT VECTORS**

## Lagrange Multipliers

Differentiate  $\mathcal{L}$  with respect to  $w, b$ , and set the differential to zero:

- For  $w$ :

$$\frac{\partial}{\partial w} \mathcal{L}(w, b, \alpha) = w - \sum_{i=1}^m \alpha_i y_i x_i = 0$$
$$\implies w = \sum_{i=1}^m \alpha_i y_i x_i$$

- For  $b$ :

$$\frac{\partial}{\partial b} \mathcal{L}(w, b, \alpha) = 0 - \sum_{i=1}^m \alpha_i y_i$$
$$\implies \sum_{i=1}^m \alpha_i y_i = 0$$

If we can solve for  $\alpha$  (dual problem), then we have a solution for  $w, b$  (primal problem)

## Lagrange Multipliers

---

$$\begin{aligned}\frac{\partial \frac{1}{2}w^T w}{\partial w} &= \left[ \frac{\partial \frac{1}{2}w^T w}{\partial w_1}, \frac{\partial \frac{1}{2}w^T w}{\partial w_2}, \dots, \frac{\partial \frac{1}{2}w^T w}{\partial w_m} \right] \\ &= \left[ \frac{\partial \frac{1}{2}[w_1, w_2, \dots, w_m]^T [w_1, w_2, \dots, w_m]}{\partial w_1}, \frac{\partial \frac{1}{2}[w_1, w_2, \dots, w_m]^T [w_1, w_2, \dots, w_m]}{\partial w_2}, \dots, \frac{\partial \frac{1}{2}[w_1, w_2, \dots, w_m]^T [w_1, w_2, \dots, w_m]}{\partial w_m} \right] \\ &= \left[ \frac{\partial \frac{1}{2}(w_1^2 + w_2^2 + \dots + w_m^2)}{\partial w_1}, \frac{\partial \frac{1}{2}(w_1^2 + w_2^2 + \dots + w_m^2)}{\partial w_2}, \dots, \frac{\partial \frac{1}{2}(w_1^2 + w_2^2 + \dots + w_m^2)}{\partial w_m} \right] \\ &= [2 * \frac{1}{2}w_1, 2 * \frac{1}{2}w_2, \dots, 2 * \frac{1}{2}w_m] = 2 * \frac{1}{2} * [w_1, w_2, \dots, w_m] = 2 * \frac{1}{2} * w = w\end{aligned}$$

Similarly in other cases, we always differentiate a scalar function over a vector  $w$  here, which results in a vector of derivatives over each element  $w_j, 1 \leq j \leq m$ .

Rewrite the Lagrangian objective. Lets put these results back into  $\mathcal{L}$  equation in order to eliminate  $w, b$ :

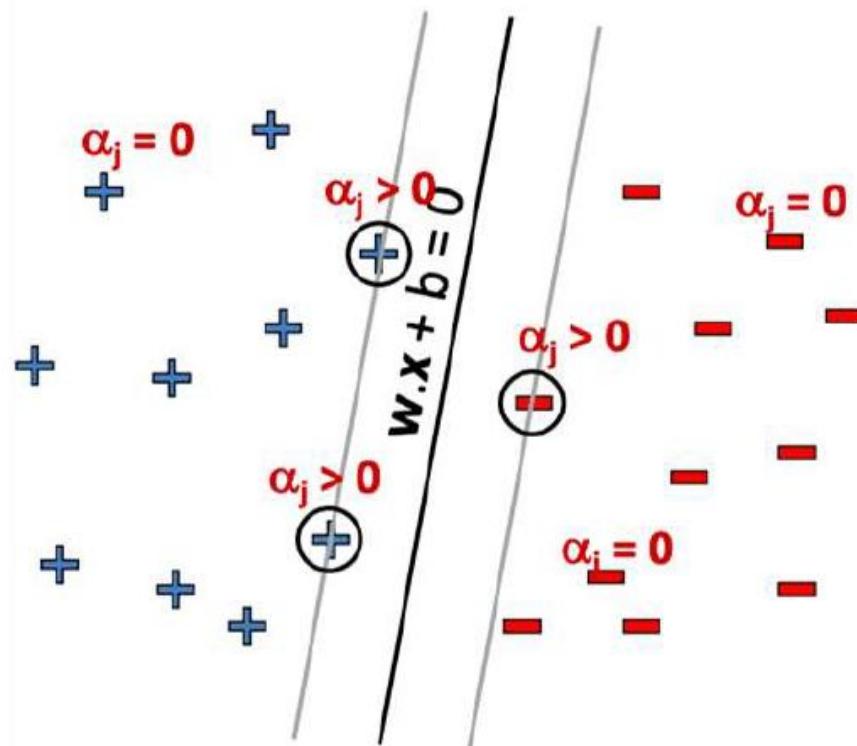
$$w = \sum_{i=1}^m \alpha_i y^i x^i$$

$$\begin{aligned}\mathcal{L}(w, b, \alpha) &= \frac{\|w\|^2}{2} - \sum_{i=1}^m \alpha_i (y^i (w^T x^i + b) - 1) \\ &= \frac{1}{2} (w^T w) - \sum_{i=1}^m \alpha_i y^i w^T x^i - \sum_{i=1}^m \alpha_i y^i b + \sum_{i=1}^m \alpha_i \\ &= \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^i y^j x^i x^j - \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^i y^j x^i x^j + \sum_{i=1}^m \alpha_i\end{aligned}$$

$$\mathcal{L}(w, b, \alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y^i y^j x^i x^j$$

## Lagrange Multipliers

## Dual SVM: Sparsity of dual solution



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few  $\alpha_j$ s can be non-zero : where constraint is active and tight

$$(\mathbf{w} \cdot \mathbf{x}_j + b)y_j = 1$$

**Support vectors** – training points j whose  $\alpha_j$ s are non-zero

**SVM-DUAL OPTIMIZATION PROBLEM**

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Subject to Constraints:  
for all  $i = 1$  to  $m$  (where  $m$  is the number of Training instances)

Constraint 1 ( $C_1$ ):  $\alpha_i \geq 0$

Constraint 2 ( $C_2$ ):  $\sum_{i=1}^m \alpha_i y_i = 0$

Constraint 3 ( $C_3$ ):  $w^T x^+ + b = +1$

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad \left[ \sum_{i=1}^m \alpha_i y_i x_i \cdot x \right] + b = +1$$

Constraint 4 ( $C_4$ ):  $w^T x^- + b = -1$

$$\left[ \sum_{i=1}^m \alpha_i y_i x_i \cdot x \right] + b = -1$$

## Lagrange Multipliers

---

- We previously stressed on equality constraints when introducing Lagrange Multipliers.
- But our SVM is a case of inequality constraints
- When dealing with inequality constraints, there is an additional requirement. The solution must also satisfy the **Karush-Kuhn-Tucker (KKT)** conditions.
- The KKT conditions are first-order necessary conditions for a solution of an optimization problem to be optimal.

$$\begin{aligned}g(X) &\geq 0; \\ \lambda &\geq 0; \\ \lambda g(X) &= 0\end{aligned}$$

- Stationarity condition:

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial \mathcal{L}}{\partial b} = - \sum_{i=1}^m \alpha_i y_i = 0$$

- The stationary condition tells us that the selected point must be a stationary point.
- It is a point where the function stops increasing or decreasing.
- When we have constraints, we use the gradient of the Lagrangian.

## Lagrange Multipliers

---

- Primal feasibility condition:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \text{for all } i = 1, \dots, m$$

Looking at this condition, you should recognize the constraints of the primal problem. It makes sense that they must be enforced to find the minimum of the function under constraints.

- Dual feasibility condition:

$$\alpha_i \geq 0 \quad \text{for all } i = 1, \dots, m$$

This tells only those instances which lie on the gutter will have multipliers greater than 0 and for rest of the instances, the value of the Lagrange Multiplier will be 0.

- Complementary slackness condition:

$$\alpha_i[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0 \quad \text{for all } i = 1, \dots, m$$

From the complementary slackness condition, we see that either  $\alpha_i = 0$  or  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$ .

- Complimentary slackness requires only one of the two following cases hold true for a constraint:
  - The constraint is binding (i.e. equality)
  - The Lagrangian multiplier of this constraint is zero

**Support vectors** are examples having a positive Lagrange multiplier. They are the ones for which the constraint  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$  is **active**. (We say the constraint is active when  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$ ).

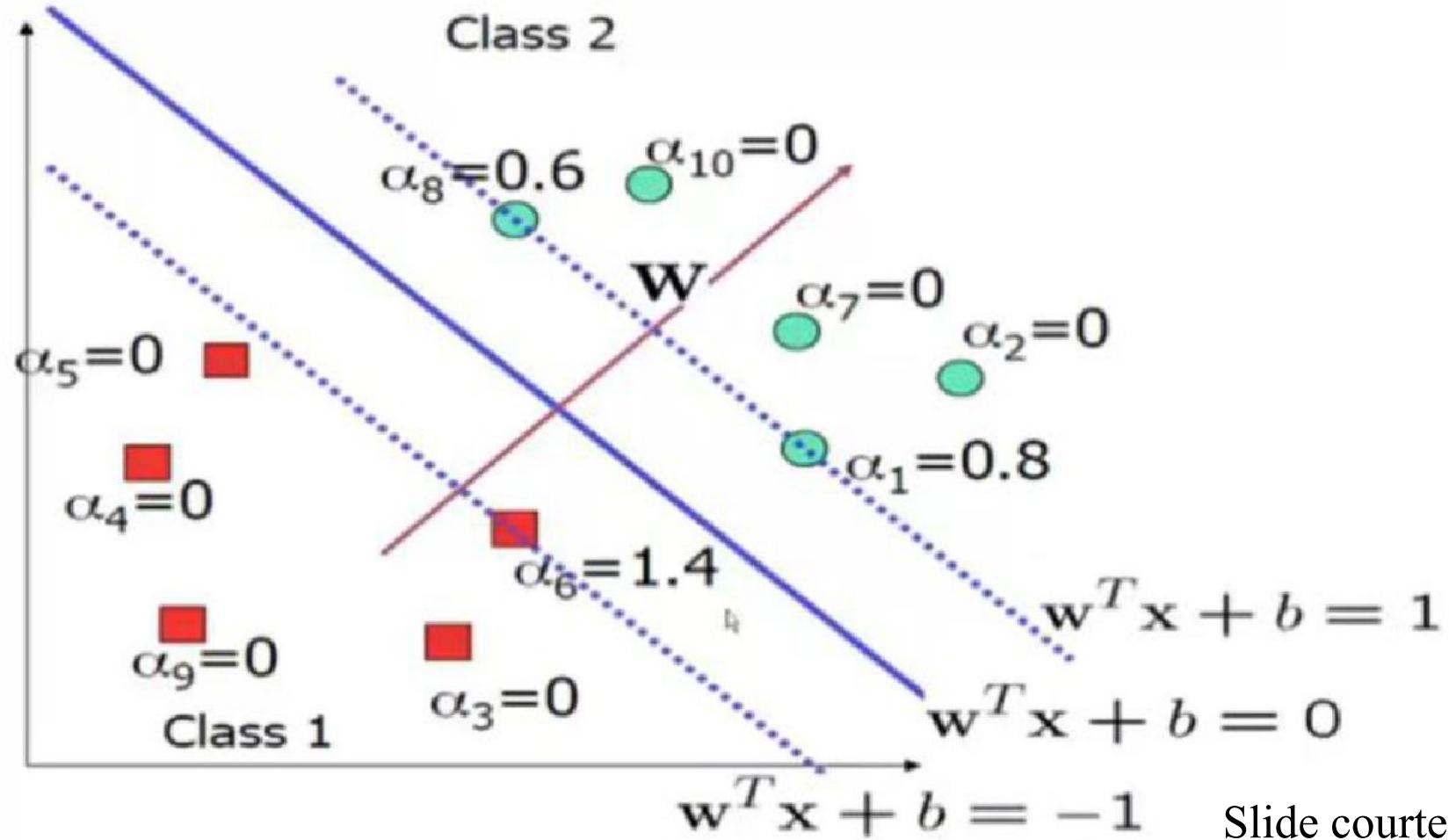
# So why solve the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal, (specially in high dimensions  $d \gg n$ )

- The classifier is a separating hyperplane.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points  $x_i$  are support vectors with non-zero Lagrangian multipliers.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products.

### SVM-DUAL OPTIMIZATION PROBLEM

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i^\top x_j$$

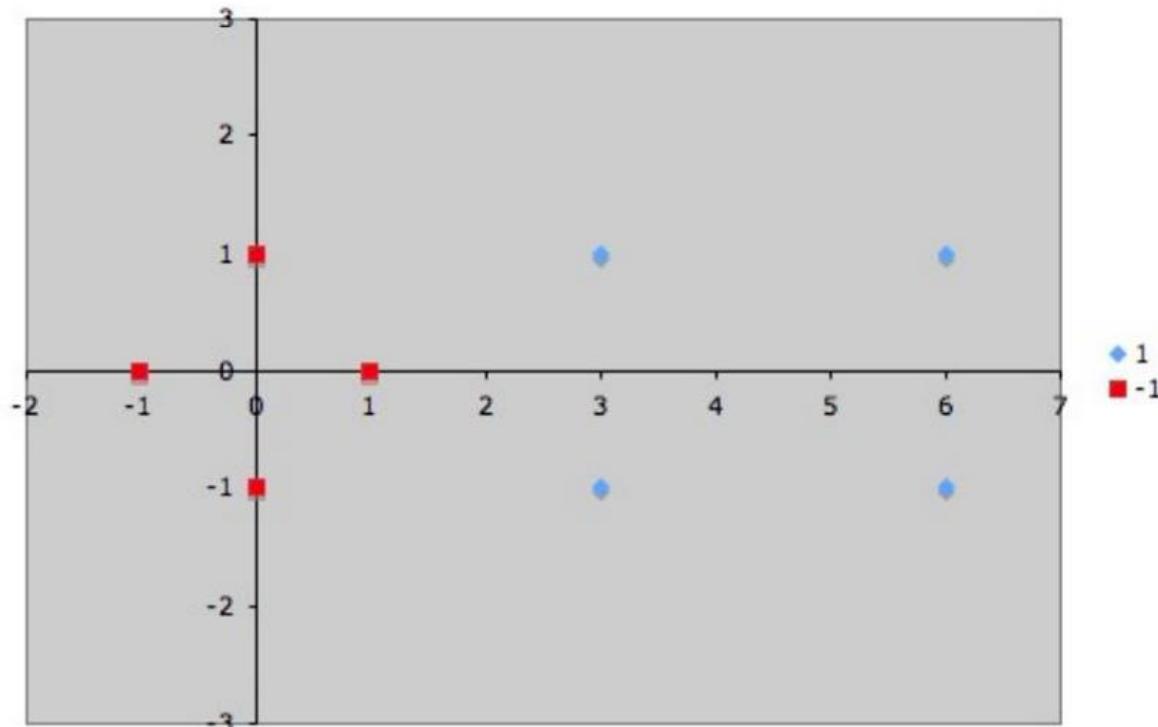


1. Is the data linearly separable?
2. Discover a simple SVM that accurately discriminates the two classes.

| point  | class |
|--------|-------|
| (3,1)  | +     |
| (3,-1) | +     |
| (6,1)  | +     |
| (6,-1) | +     |
| (1,0)  | -     |
| (0,1)  | -     |
| (0,-1) | -     |
| (-1,0) | -     |

1. Is the data linearly separable?

Plot the data and examine whether the data is linearly separable or not. The data provided is linearly separable.



| point  | class |
|--------|-------|
| (3,1)  | +     |
| (3,-1) | +     |
| (6,1)  | +     |
| (6,-1) | +     |
| (1,0)  | -     |
| (0,1)  | -     |
| (0,-1) | -     |
| (-1,0) | -     |

2. Discover a simple SVM that accurately discriminates the two classes.

Ideally we must solve the dual problem or primal problem, because we don't know which instances are support vectors. (SMO algorithm gives an efficient way of solving the dual problem of the support vector machine optimization problem --- out of scope)

But for these toy examples, we are able to manually inspect which are the support vectors, we can use the binding constraints in order to solve for w and b:

$$\mathbf{w}^T \mathbf{X} + b = -1$$

$$\mathbf{w}^T \mathbf{X} + b = 1$$

### Two ways to Solve :

Way 1 : Directly substitute the support vectors in above equations and get w and b.

Way 2 : Find Lagrange Multipliers and then find w and b.

Slide courtesy : Prof. Preet Kanwal

2. Discover a simple SVM that accurately discriminates the two classes.

**Way 1 - Solving for w and b using the binding constraint equations :**

$$\mathbf{w}^T \mathbf{X} + b = -1$$

$$\mathbf{w}^T \mathbf{X} + b = 1$$

Here, X is a support vector.

for our question, we have the following support vectors:-

$$s_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad s_2 = \begin{bmatrix} 3 \\ 1 \end{bmatrix} \quad s_3 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$y_1 = -1 \quad y_2 = +1 \quad y_3 = +1$$

we get 3 equations

$$\mathbf{w}^T s_1 + b = -1$$

$$\mathbf{w}^T s_2 + b = +1$$

$$\mathbf{w}^T s_3 + b = +1$$

final equations :

$$w_1 + b = -1$$

$$3w_1 + w_2 + b = 1$$

$$3w_1 - w_2 + b = 1$$

2. Discover a simple SVM that accurately discriminates the two classes.

**Way 1 - Solving for w and b using the binding constraint equations :**

$$\mathbf{w}^T \mathbf{X} + b = -1$$

$$\mathbf{w}^T \mathbf{X} + b = 1$$

Here, X is a support vector.

final equations :

$$① \mathbf{w}_1 + b = -1$$

$$② 3\mathbf{w}_1 + \mathbf{w}_2 + b = 1$$

$$③ 3\mathbf{w}_1 - \mathbf{w}_2 + b = 1$$

Solving above equations

$$\text{eqn. } ② - ③$$

$$6\mathbf{w}_1 + 2b = 2$$

from eqn. ① we know ( $\mathbf{w}_1 = -1 - b$ )

$$\Rightarrow 6(-1 - b) + 2b = 2$$

$$\Rightarrow -6 - 6b + 2b = 2$$

$$\Rightarrow 4b = -8$$

$$\Rightarrow b = -2$$

Using eqn ① we get

$$\begin{aligned}\mathbf{w}_1 &= -1 - b \\ &= -1 - (-2)\end{aligned}$$

$$\mathbf{w}_1 = 1$$

using eqn. ③ we get

$$3\mathbf{w}_1 - \mathbf{w}_2 + b = 1$$

$$\Rightarrow 3(1) - \mathbf{w}_2 + (-2) = 1$$

$$\Rightarrow 1 - \mathbf{w}_2 = 1$$

$$\Rightarrow \mathbf{w}_2 = 0$$

$$\therefore \mathbf{w} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and bias} = -2$$

2. Discover a simple SVM that accurately discriminates the two classes.

### **Way 2 - Finding Lagrange Multipliers first and then solving for w and b :**

Step 1 : Identify the support vectors by manual inspection (could be provided in the question)

Step 2 : Write the system of equations using SVM Constraints (C2, C3, C4)

Step 4 : Solve the system of equations to find the value of Lagrange Multipliers and bias.

Step 5 : Use the Lagrange Multipliers to find the weight vector using the formula :

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

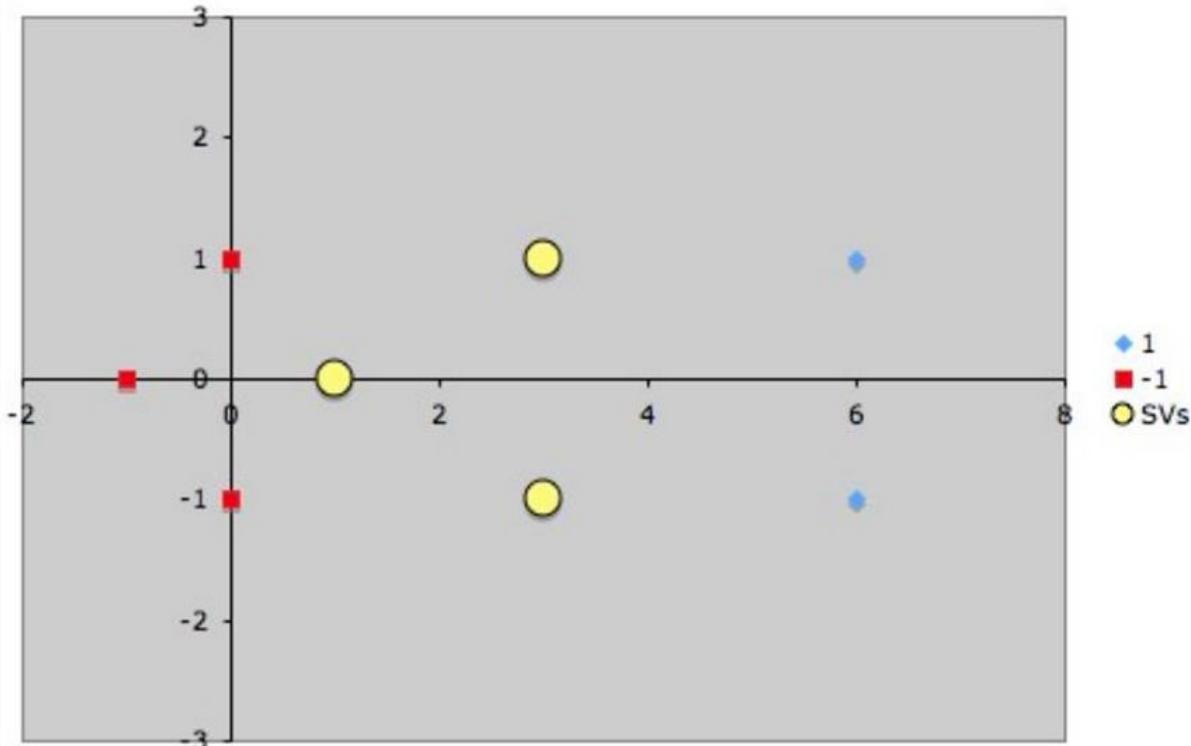
Step 6 : Report the weight vector and bias value as the Solution.

Step 7 : Given any new instance U to classify, plug the weight and bias value in the equation  $Z = w^T U + b$  and provide your prediction for the label of U as +ve if  $Z \geq 1$ , else if  $Z \leq -1$ , predict U as belonging to the -ve class.

Slide courtesy : Prof. Preet Kanwal

2. Discover a simple SVM that accurately discriminates the two classes.

Step 1 : Identify the support vectors by manual inspection



| point  | class |
|--------|-------|
| (3,1)  | +     |
| (3,-1) | +     |
| (6,1)  | +     |
| (6,-1) | +     |
| (1,0)  | -     |
| (0,1)  | -     |
| (0,-1) | -     |
| (-1,0) | -     |

2. Discover a simple SVM that accurately discriminates the two classes.

Step 2a : Write the system of equations using SVM constraints c2, c3, c4

Subject to Constraints:  
for all  $i = 1 \text{ to } m$  (where  $m$  is the number of Training instances)

$$\text{Constraint 1 } (C_1) : \alpha_i \geq 0$$

$$\text{Constraint 2 } (C_2) : \sum_{i=1}^m \alpha_i y_i = 0$$

$$\text{Constraint 3 } (C_3) : w^T x^+ + b = +1$$

$$w = \sum_{i=1}^m \alpha_i y_i x_i \quad \left[ \sum_{i=1}^m \alpha_i y_i x_i \cdot x \right] + b = +1$$

$$\text{Constraint 4 } (C_4) : w^T x^- + b = -1$$

$$\left[ \sum_{i=1}^m \alpha_i y_i x_i \cdot x \right] + b = -1$$

| point  | class |
|--------|-------|
| (3,1)  | +     |
| (3,-1) | +     |
| (6,1)  | +     |
| (6,-1) | +     |
| (1,0)  | -     |
| (0,1)  | -     |
| (0,-1) | -     |
| (-1,0) | -     |

2. Discover a simple SVM that accurately discriminates the two classes.

Step 2a : Plug the support vectors into the constraint equations c2, c3, c4

Because for other instances, Lagrange multiplier is Zero.

- $-\alpha_1 s_1 \cdot s_1 + \alpha_2 s_1 \cdot s_2 + \alpha_3 s_1 \cdot s_3 + b = -1$
- $-\alpha_1 s_2 \cdot s_1 + \alpha_2 s_2 \cdot s_2 + \alpha_3 s_2 \cdot s_3 + b = 1$
- $-\alpha_1 s_3 \cdot s_1 + \alpha_2 s_3 \cdot s_2 + \alpha_3 s_3 \cdot s_3 + b = 1$
- $-\alpha_1 + \alpha_2 + \alpha_3 + 0 \cdot b = 0$

| Support Vector | point  | class |
|----------------|--------|-------|
| s1             | (1,0)  | -1    |
| s2             | (3,1)  | +1    |
| s3             | (3,-1) | +1    |

Slide courtesy : Prof. Preet Kanwal

2. Discover a simple SVM that accurately discriminates the two classes.

Step 3 : Solve the equations to find the value of Lagrange Multipliers and bias

- $-\alpha_1 + 3\alpha_2 + 3\alpha_3 + b = -1$
- $-3\alpha_1 + 10\alpha_2 + 8\alpha_3 + b = 1$
- $-3\alpha_1 + 8\alpha_2 + 10\alpha_3 + b = 1$
- $-\alpha_1 + \alpha_2 + \alpha_3 + 0.b = 0$

| Support Vector | point  | class |
|----------------|--------|-------|
| s1             | (1,0)  | -1    |
| s2             | (3,1)  | +1    |
| s3             | (3,-1) | +1    |

Slide courtesy : Prof. Preet Kanwal

**SVM**

2. Discover a simple SVM that accurately discriminates the two classes.

Step 3 : Solve the equations to find the value of Lagrange Multipliers and bias

$$-\alpha_1 + 3\alpha_2 + 3\alpha_3 + b = -1 \quad \text{---(1)}$$

$$-3\alpha_1 + 10\alpha_2 + 8\alpha_3 + b = 1 \quad \text{---(2)}$$

$$-3\alpha_1 + 8\alpha_2 + 10\alpha_3 + b = 1 \quad \text{---(3)}$$

$$-\alpha_1 + \alpha_2 + \alpha_3 = 0 \quad \text{---(4)}$$

Eqn. (2) - (3)  $2\alpha_2 - 2\alpha_3 = 0 \Rightarrow \alpha_2 - \alpha_3 = 0 \Rightarrow \alpha_2 = \alpha_3$

Substitute  $\alpha_2 = \alpha_3$  in (4)  $\Rightarrow \alpha_1 = \alpha_3 + \alpha_3 \Rightarrow \alpha_1 = 2*\alpha_3$

Substitute  $\alpha_1 = 2\alpha_3$ ,  $\alpha_2 = \alpha_3$  in equation (1) & (2)

Eqn. 1 :  $-2\alpha_3 + 3\alpha_3 + 3\alpha_3 + b = -1 \Rightarrow 4\alpha_3 + b = -1 \quad \text{---(5)}$

Eqn. 2 :  $-6\alpha_3 + 10\alpha_3 + 8\alpha_3 + b = 1 \Rightarrow 12\alpha_3 + b = 1 \quad \text{---(6)}$

Eqn. (5) - (6)  $-8\alpha_3 = -2 \Rightarrow \alpha_3 = \frac{1}{4}$

$\alpha_1 = \frac{1}{2}, \alpha_2 = \frac{1}{4}, \alpha_3 = \frac{1}{4}, b = -2$

Slide courtesy : Prof. Preet Kanwal

2. Discover a simple SVM that accurately discriminates the two classes.

Step 3 : Solve the equations to find the value of Lagrange Multipliers and bias

solving this equations with 4 unknowns we get

- $b = -2$
- $\alpha_1 = \frac{1}{2}$
- $\alpha_2 = \frac{1}{4}$
- $\alpha_3 = \frac{1}{4}$

| Support Vector | point  | class |
|----------------|--------|-------|
| s1             | (1,0)  | -1    |
| s2             | (3,1)  | +1    |
| s3             | (3,-1) | +1    |

Slide courtesy : Prof. Preet Kanwal

2. Discover a simple SVM that accurately discriminates the two classes.

Step 4 : Find the weight vector

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

$$w = \frac{1}{2} x - 1 \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{4} \times 1 \times \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \frac{1}{4} \times 1 \times \begin{pmatrix} 3 \\ -1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

| Support Vector | point  | class |
|----------------|--------|-------|
| s1             | (1,0)  | -1    |
| s2             | (3,1)  | +1    |
| s3             | (3,-1) | +1    |

Final solution : Provide weight vector and bias value.

Slide courtesy : Prof. Preet Kanwal



**PES**  
**UNIVERSITY**

CELEBRATING 50 YEARS

**THANK YOU**

---

**Surabhi Narayan**

Department of Computer Science and Engineering

[surabhinarayan@gmail.com](mailto:surabhinarayan@gmail.com)