



Lecture on

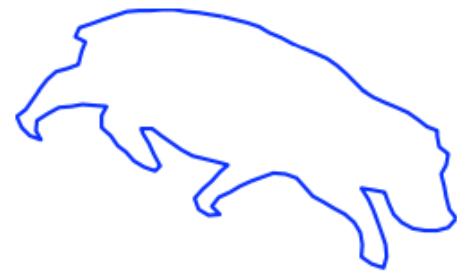
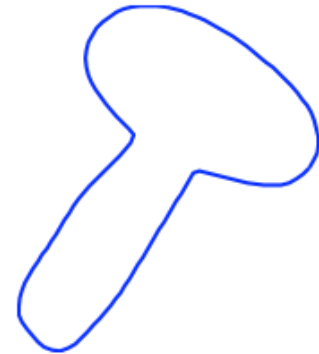
Linear Models for Classification







Animal or Vegetable

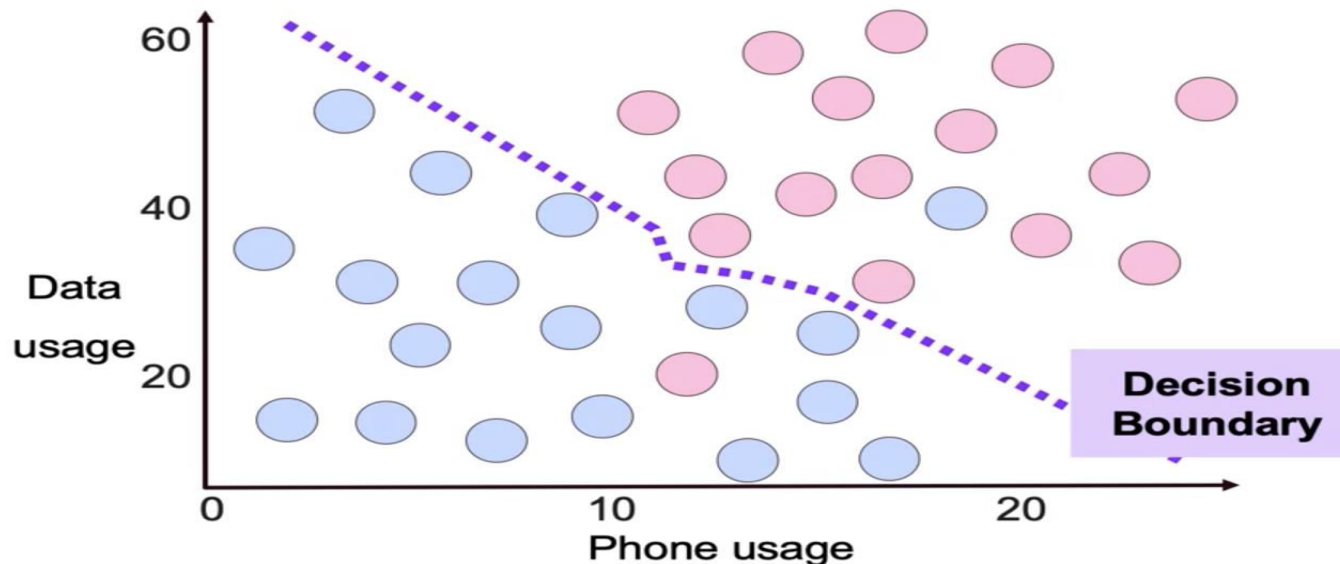


Classification: Introduction

- ❑ Classification is the task of assigning predefined disjoint categories to the objects.
- ❑ The basic idea behind the linear classifier is that the two values of the target class can be separated by a hyperplane in the feature space.
- ❑ If this can be done without error, the training set is called linearly separable.
 - ❑ Whether to play tennis or not
 - ❑ Whether to buy a iPhone or not on big billion day
 - ❑ Detect spam email
 - ❑ Whether a movie review is positive or not.
 - ❑ Whether placed at A company or not
 - ❑ Find the set of mobile < Rs 20,000 and rating is 5*

Classification: Introduction

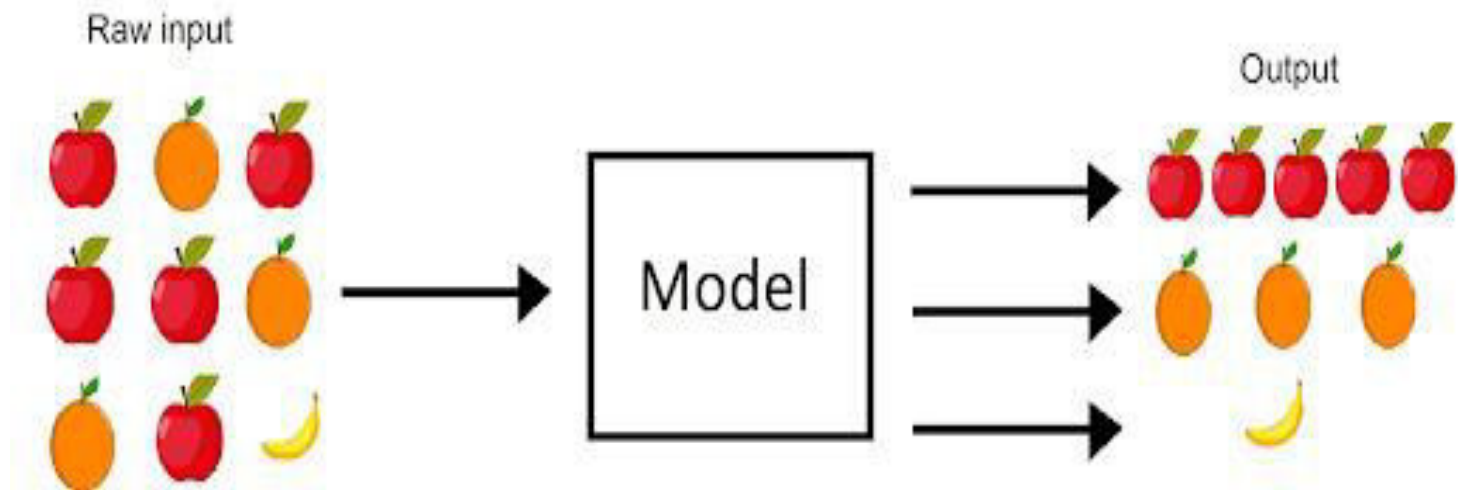
- Linear models for classification separate input vectors into classes using linear decision boundaries.
- Example-
 - Input vector x
 - Two discrete classes $C1$ and $C2$



Definition of Classification

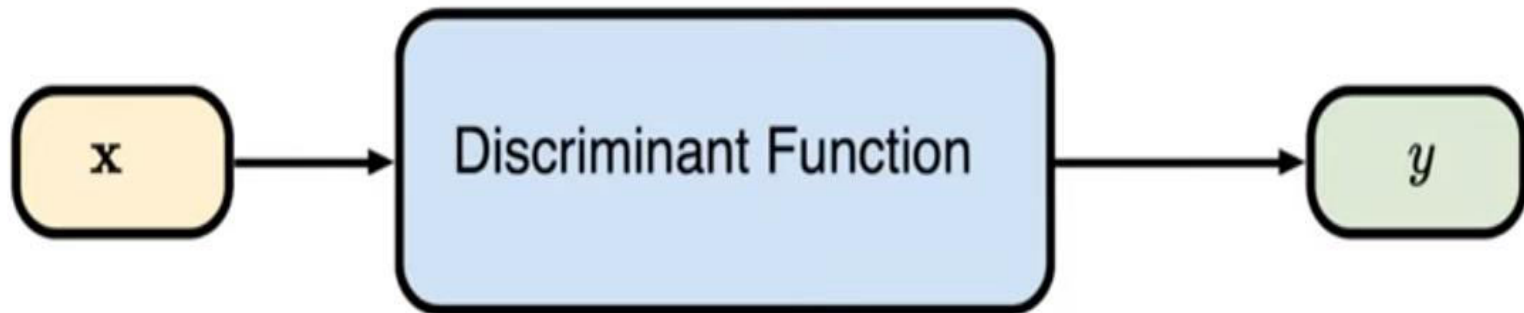
- The input is collection of records or data points.
- Each data point is represented by a tuple (x,y) .
- Where $X = \{x_1, x_2, x_3, x_4, \dots, x_n\}$
- $Y = y_1, y_2, y_3, y_4, \dots, y_n$
- Are the input feature and the classes respectively.
- $X \in \mathbb{R}^2$ is a vector- the set of observed variable.
- Let say, (x,y) are related by an unknown function
- The goal is to estimate the unknown function $g(.)$, it is known as classifier function
- Such that $g(x) = f(x)$

Contd..



Discriminant Function

- A discriminant function learn direct mapping between feature vector x and label y .
- Basically the discriminant function extended setup of the classification problem.



Contd..

■ **Binary classification**

- In binary classification set up with m features, the simplest the simplest discriminant function is very similar to the linear regression.
- It is the simplest and most commonly used function, since it has two class only.
- It is two class classification problem.

Contd..

- A linear regression model can be expressed as:

$$\text{Let, } X = \{x_1, x_2, x_3, \dots, x_n\}$$

$$W = \{w_1, w_2, w_3, \dots, w_n\}$$

Then,

$$\begin{aligned} y &= w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n \\ &= w_0 + W^T X \end{aligned}$$

Geometric Interpretation

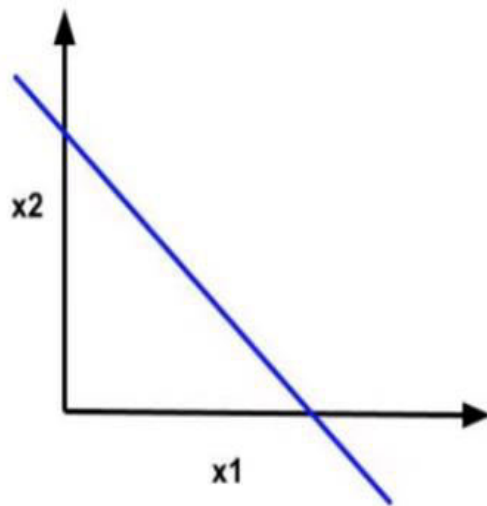
- Geometrically, the simplest discriminant function is given by

$$y = w_0 + W^T X$$

- It represents a hyper plane in $m-1$ dimensional space where m is number of features.

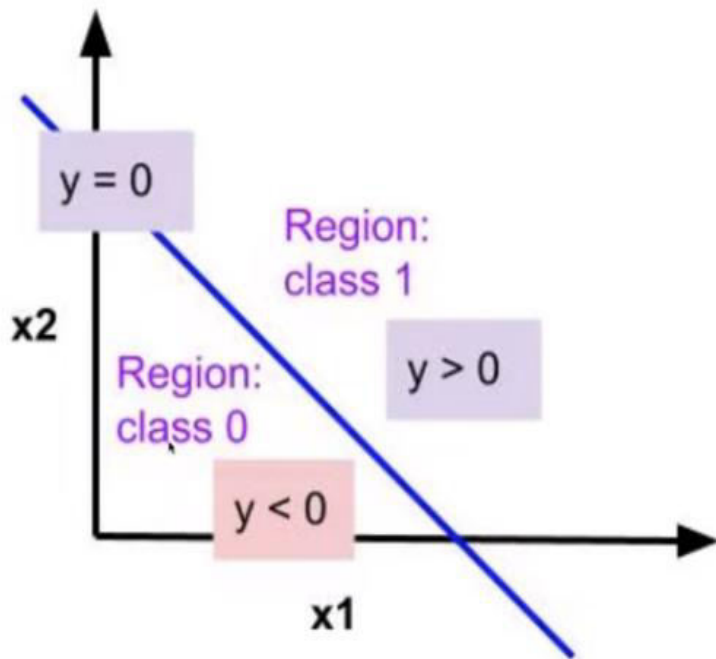
Contd..

- Lets look at the feature with their possible discriminant functions.



# features (m)	Discriminant function
1	Point
2	Line
3	Plane
4	Hyperplane in 3D space
...	...
m	Hyperplane in (m-1)-D space

Contd..

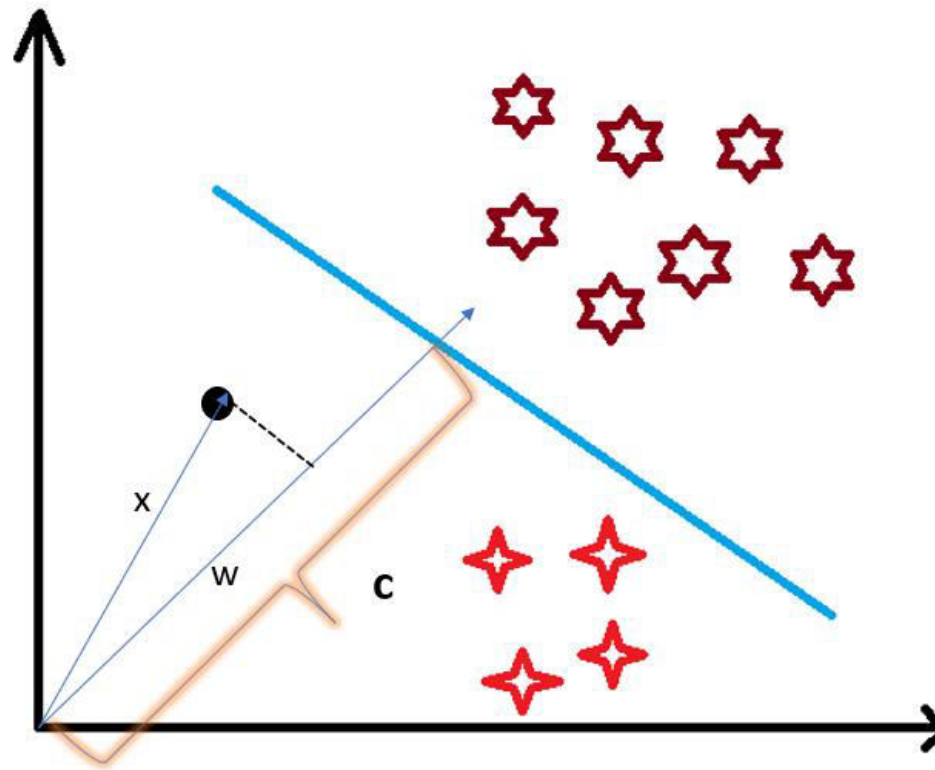


($m-1$)-D hyperplane (which is line in this case) divides feature space into two regions one for each class:

- Region for class 1, where $y > 0$
- Region for class 2, where $y < 0$

On decision boundary, $y = 0$

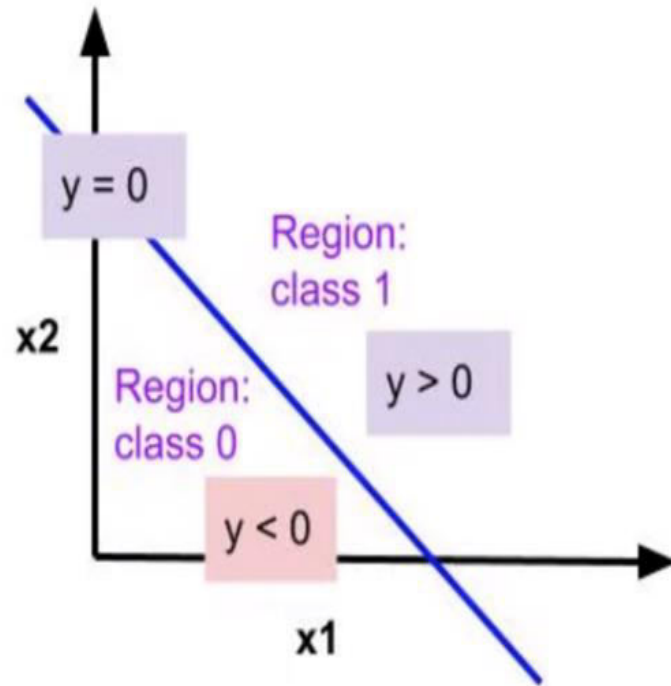
Contd..



Observation

- W_0 determines the location of decision surface.
- W determines orientation of the decision surface.
- Y gives signed measures of perpendicular distance of the point x from decision surface.
- Finally the decision surface divides feature space into two regions.

How classification can be done



Discriminant function assigns label 1 to an example with feature vector \mathbf{x} if $(w_0 + \mathbf{w}^T \mathbf{x}) > 0$ else assigns label 0.

$$y = \begin{cases} 1, & \text{if } w_0 + \mathbf{w}^T \mathbf{x} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Approaches to classification

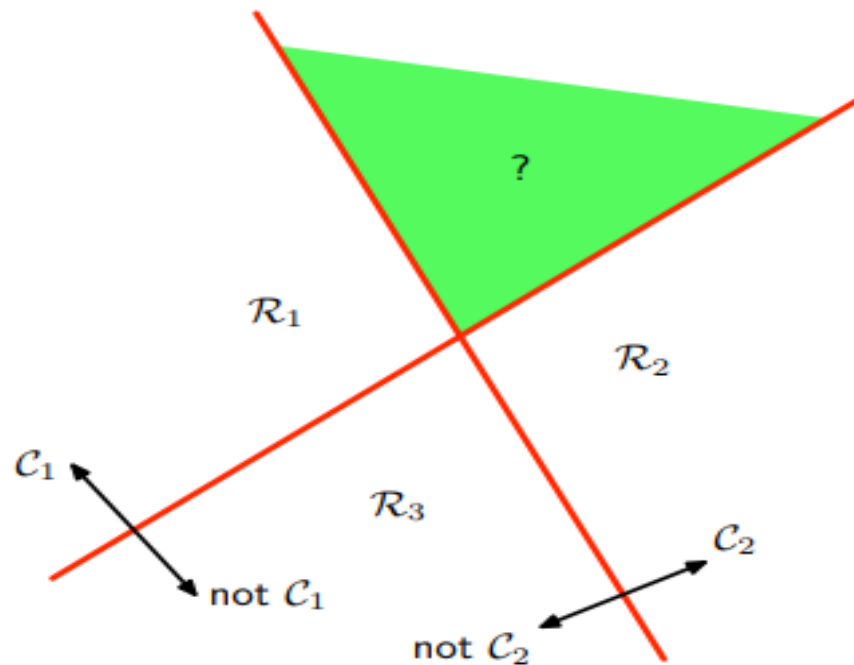
- **Discriminant function:** Directly assigns each data point x to a particular class C_i
- **Generative approach:** model class likelihoods, $p(x|C_i)$, and priors, $p(C_i)$; use Bayes' theorem to get posteriors

Multiple Class Classification

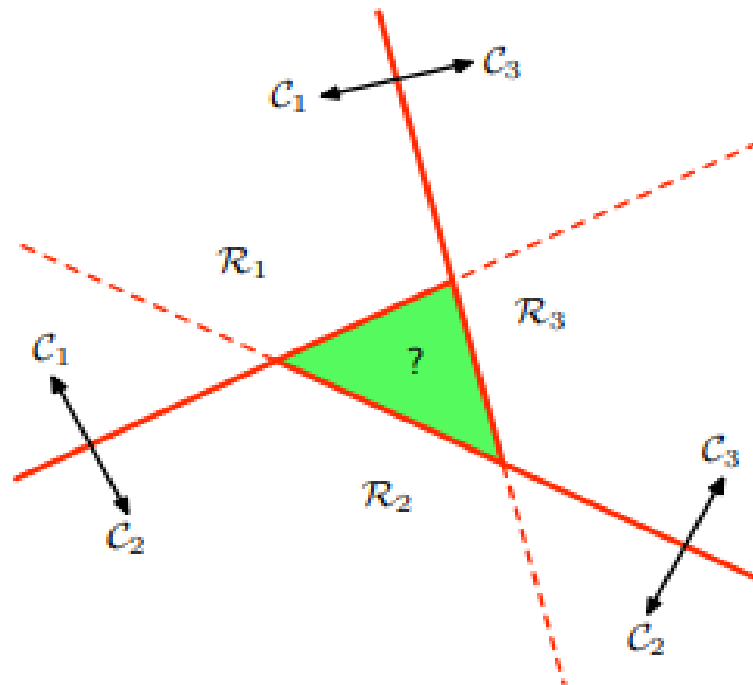
Assuming the number of classes to be $k > 2$, we can build discriminant functions in two ways:

- **One-vs-rest**: Build $k - 1$ discriminant functions. Each discriminant function solves two class classification problem: class c_k vs *not* c_k .
- **One-vs-one**: One discriminant function per pair of classes.
Total functions = $\binom{k}{2} = \frac{k(k-1)}{2}$

Issue with One Vs Rest



Issue with One Vs One



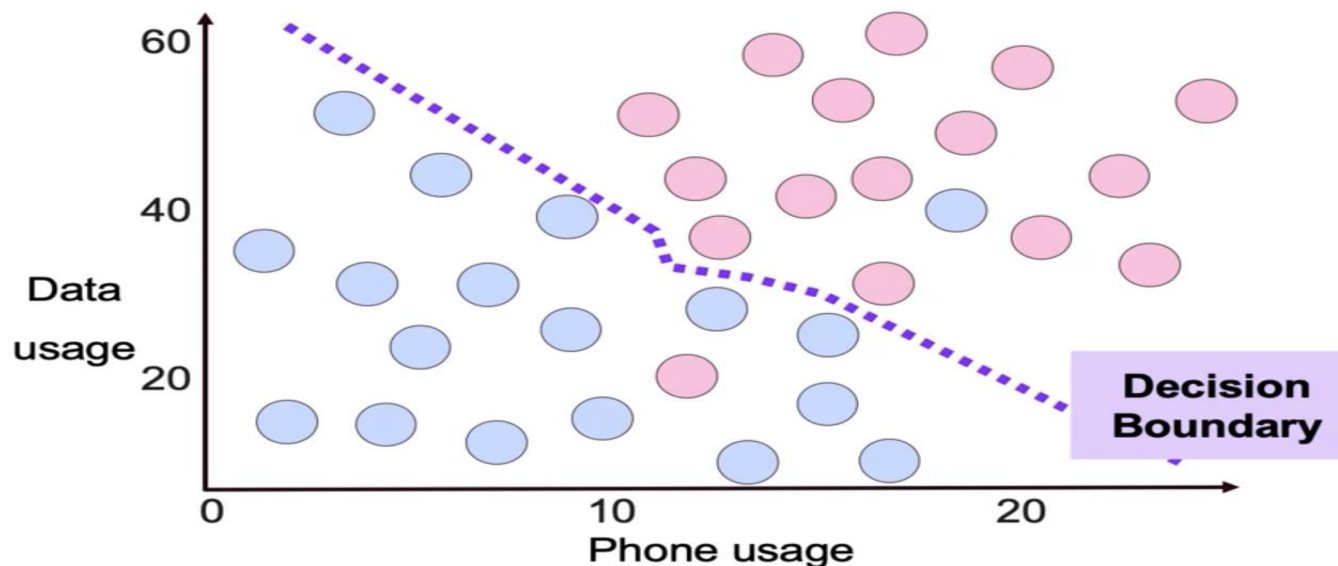


Lecture on

Linear Models for Classification: Parameter for classification

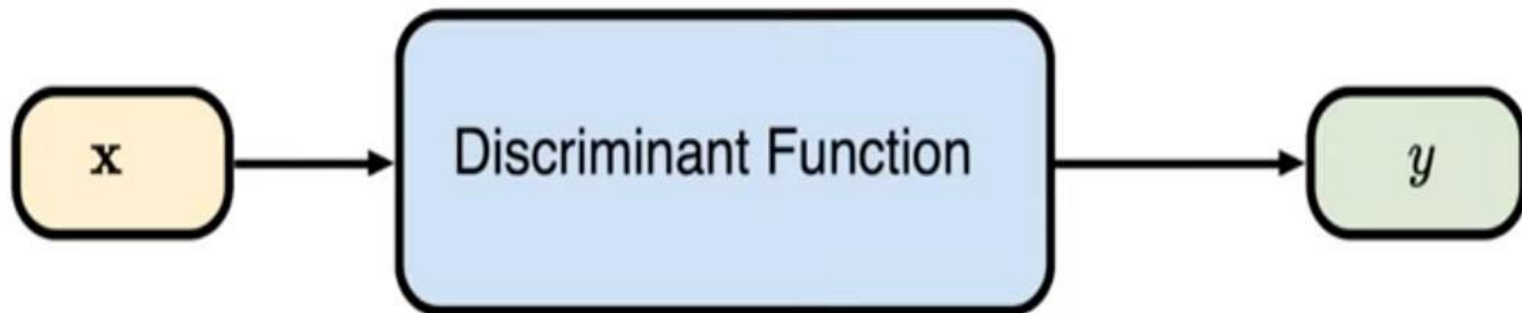
Classification: Introduction

- Linear models for classification separate input vectors into classes using linear decision boundaries.
- Example-
 - Input vector x
 - Two discrete classes $C1$ and $C2$

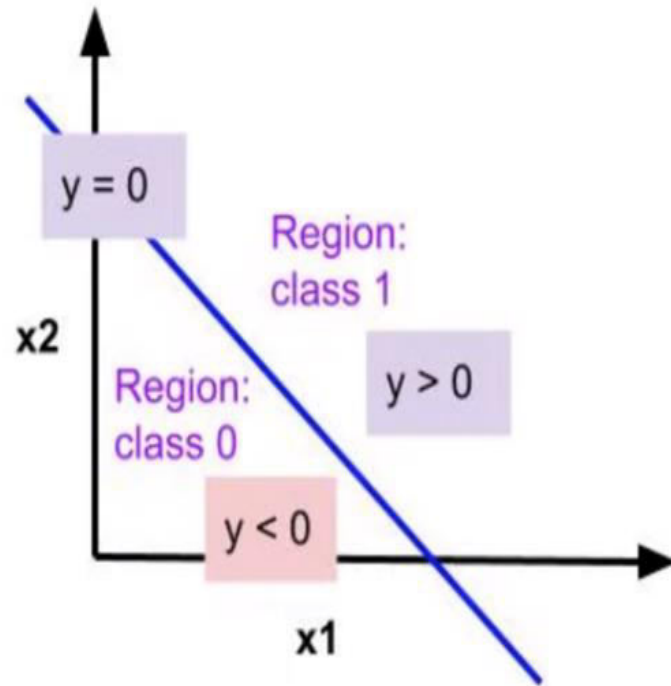


Discriminant Function

- A discriminant function learn direct mapping between feature vector x and label y .
- Basically the discriminant function extended setup of the classification problem.



How classification can be done



Discriminant function assigns **label 1** to an example with feature vector \mathbf{x} if $(w_0 + \mathbf{w}^T \mathbf{x}) > 0$ else assigns **label 0**.

$$y = \begin{cases} 1, & \text{if } w_0 + \mathbf{w}^T \mathbf{x} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Multiple Class Classification

Assuming the number of classes to be $k > 2$, we can build discriminant functions in two ways:

- **One-vs-rest**: Build $k - 1$ discriminant functions. Each discriminant function solves two class classification problem: class c_k vs *not* c_k .
- **One-vs-one**: One discriminant function per pair of classes.
Total functions = $\binom{k}{2} = \frac{k(k-1)}{2}$

Fixing issue

A single k -class discriminant comprising k linear functions as follows:

$$\begin{aligned} y_k &= w_{k0} + w_{k1}x_1 + \dots + w_{km}x_m \\ &= w_{k0} + \mathbf{w}_k^T \mathbf{x} \end{aligned}$$

Concretely:

$$\begin{aligned} y_1 &= w_{10} + \mathbf{w}_1^T \mathbf{x} \\ y_2 &= w_{20} + \mathbf{w}_2^T \mathbf{x} \\ &\vdots \\ y_k &= w_{k0} + \mathbf{w}_k^T \mathbf{x} \end{aligned}$$

Learning model parameter

- Now, we have a model of linear discriminant functions.
- We will study two approaches for estimating /learning the parameter of discriminant model-
 1. Least square for classification
 2. Fishers linear discriminant function
 3. Perceptron model

Evaluation measures

- Confusion matrix:-
- Precision, recall, F1 score
- Accuracy
- AUC etc.

Least square classification

- Least square classification is used for estimating parameter of discriminants function.
- Least square classification adapts linear regression model for classification.
- We use least square error as loss function.

Least square error

The error at i -th training point is calculated as follows:

$$\begin{aligned} e^{(i)} &= (\text{actual label} - \text{predicted label})^2 \\ &= \left(y^{(i)} - h_{\mathbf{w}}(\mathbf{x}^{(i)}) \right)^2 \\ &= \left(y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)} \right)^2 \end{aligned}$$

Optimization of LSE

- The optimization of least square error refers to the minimization of error.
- In this case the value of error must towards zero.
- The value of error is closely dependent on the value of weight vector w .
- This optimization of least square error for the parameter of discriminants function can be done in two ways-

1. By using normal equation

- To minimize the error, we calculate the partial derivative of loss function let say $J(w)$ w.r.t. weight vector w .
- Do the following:-

Set $\frac{\partial J(w)}{\partial w}$ to 0 and solve for w :

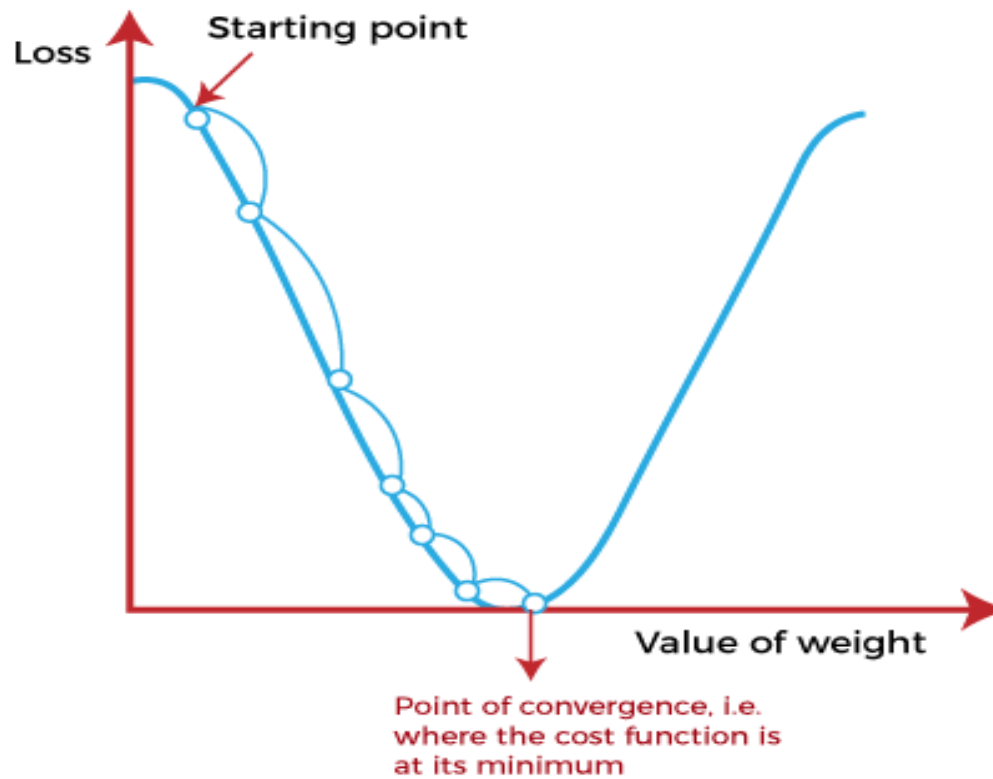
2. *By using gradient descent*

- Gradient Descent is known as one of the most commonly used optimization algorithms to train machine learning models by means of minimizing errors between actual and expected results.
- The main objective of using a gradient descent algorithm is to minimize the cost function using iteration.
- It can be derived as:

We derive **weight update rule** in vectorized format as follows:

$$\mathbf{w}_{k+1} := \mathbf{w}_k - \alpha \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

Contd..

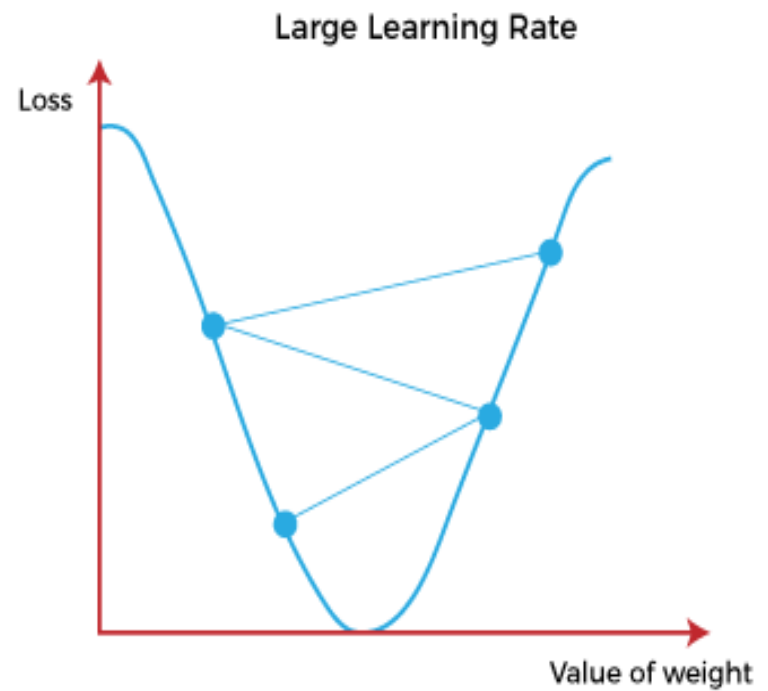
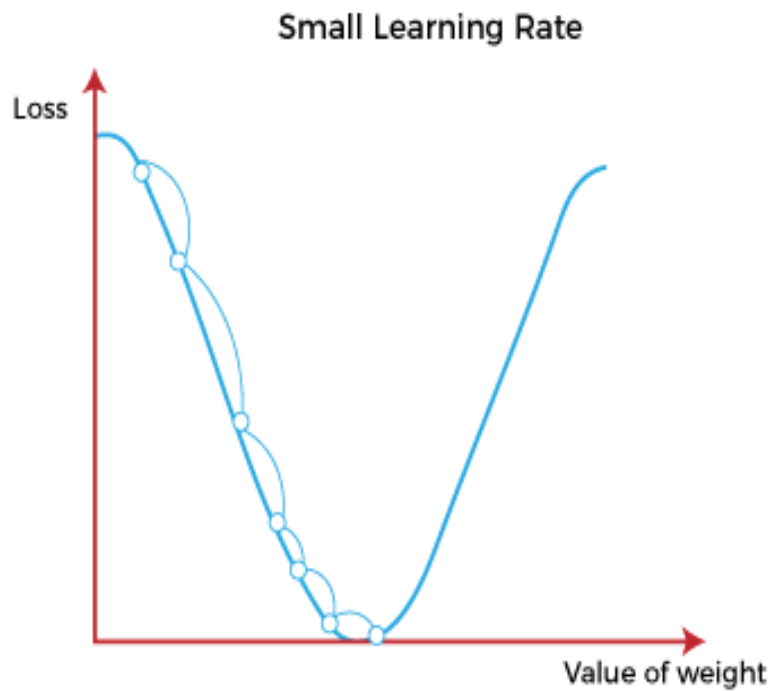


Contd..

■ Learning Rate:

- It is defined as the step size taken to reach the minimum or lowest point.
- This is typically a small value that is evaluated and updated based on the behavior of the cost function.
- If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum.
- At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.

Contd..



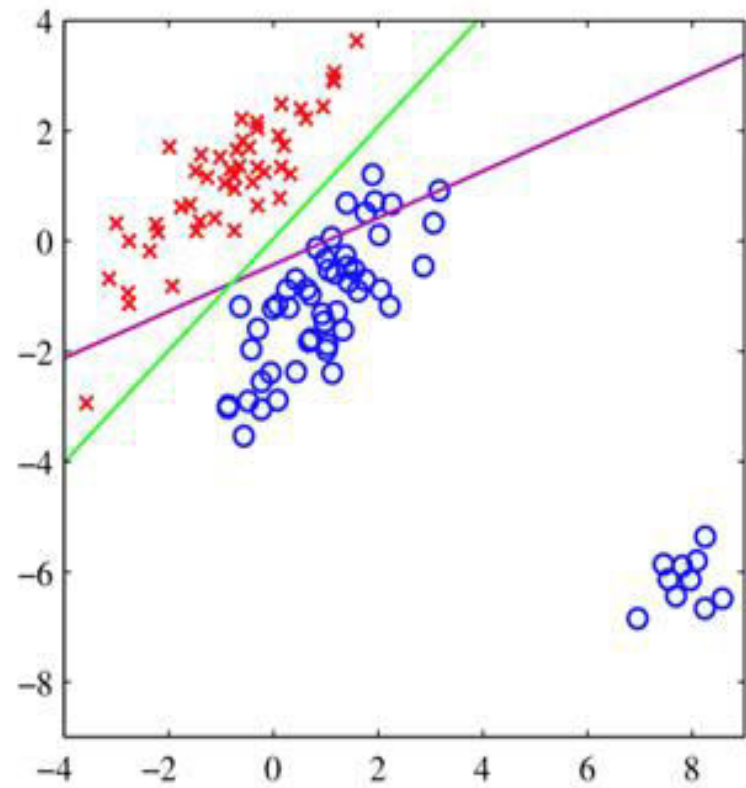
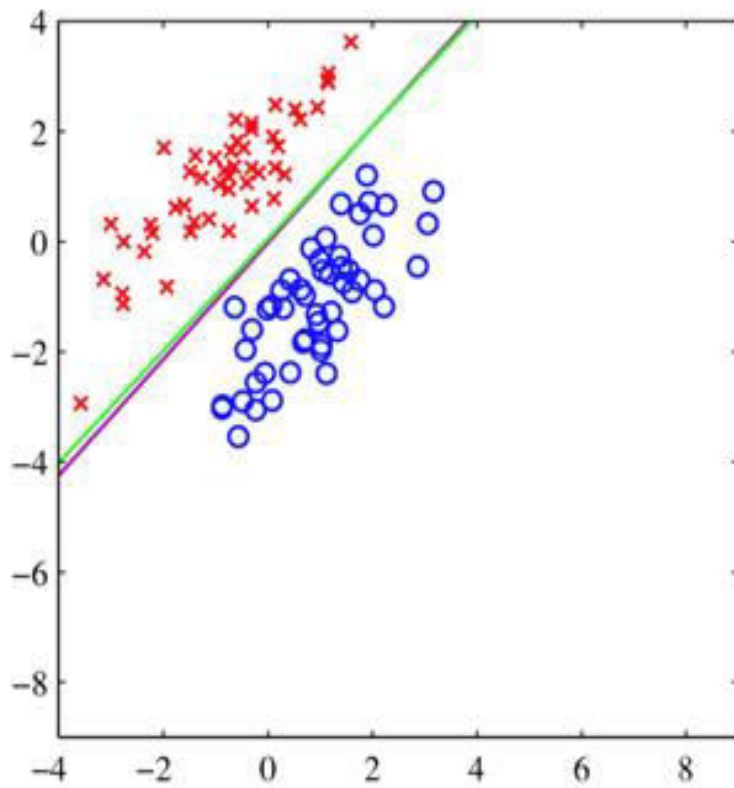
Performance of LSE

- The total loss is the sum of square of errors between actual and predicted labels at each training points.
- The least-squares problem has an analytical solution.
- When differentiating the error by w , then finding w for when the derivative is equal to zero yields the not feasible all time.

Contd..

- The least squares approach gives an exact closed form solution for discriminant function parameters. However, it has several problems:
 - It lacks robustness to outliers.
1. Additional data points away from the decision boundary produce significant change in the location of decision boundary.

Contd..

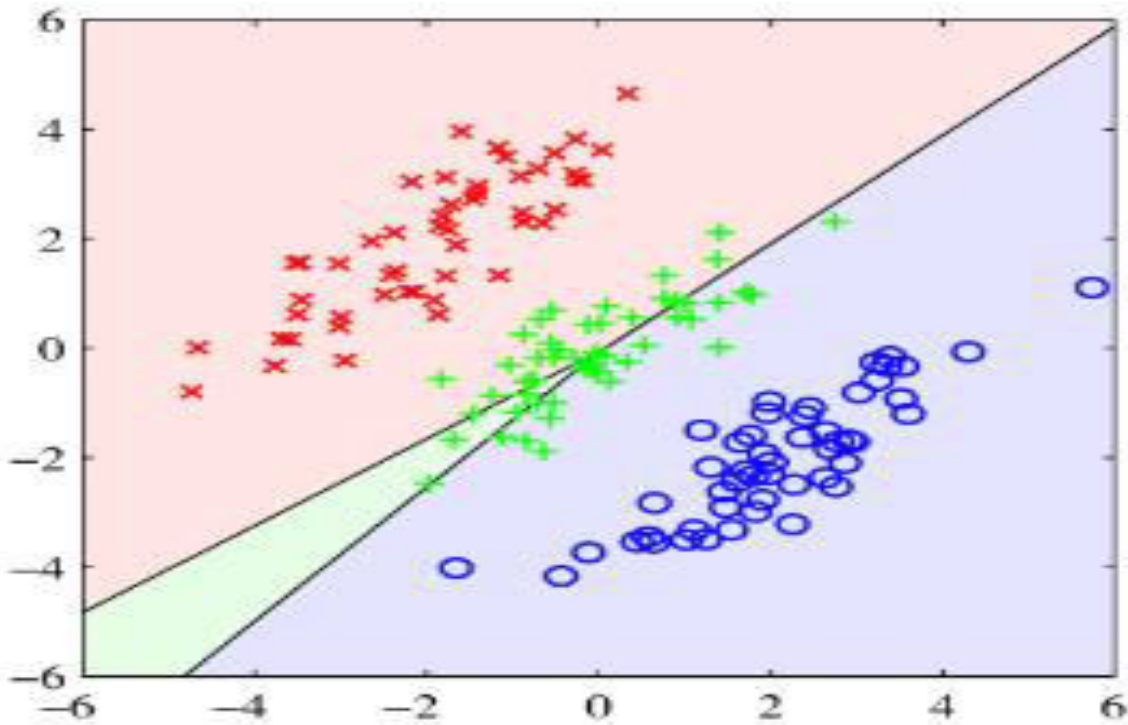




Contd..

2. For $K > 2$, some decision regions can become very small or are simply ignored.
 - This is called masking, and it happens a lot when we use regression for multi class classification.

Contd..





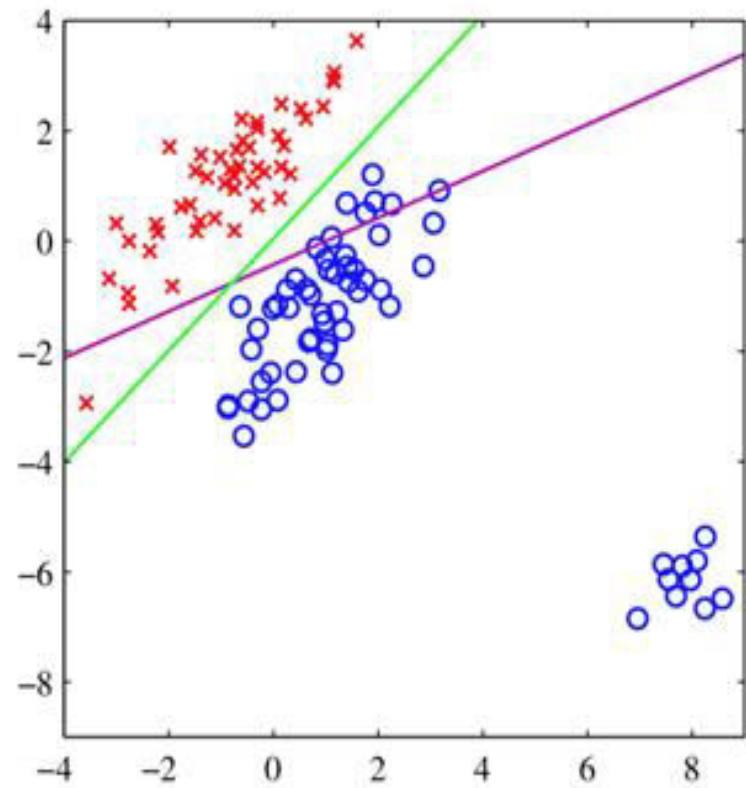
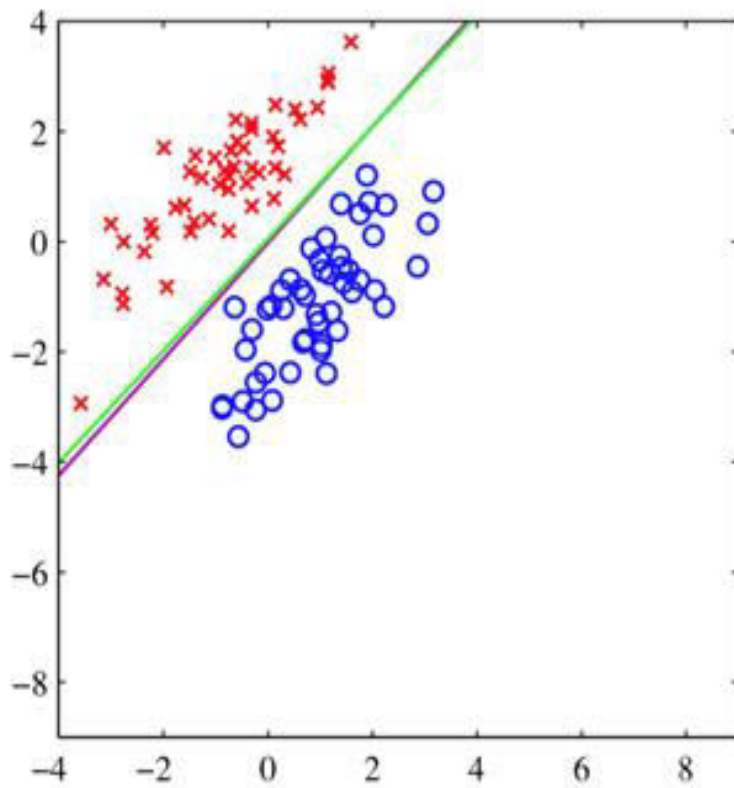
Lecture on

Linear Models for Classification: Perceptron Model

Contd..

- The least squares approach gives an exact closed form solution for discriminant function parameters. However, it has several problems:
 - It lacks robustness to outliers.
1. Additional data points away from the decision boundary produce significant change in the location of decision boundary.

Contd..

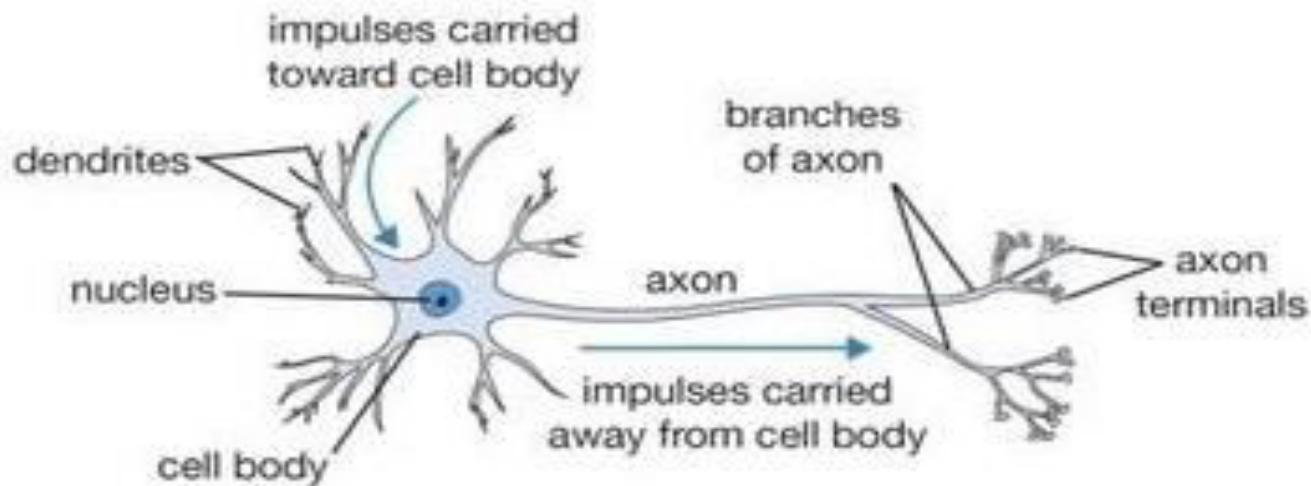




Contd..

2. For $K > 2$, some decision regions can become very small or are simply ignored.
 - This is called masking, and it happens a lot when we use regression for multi class classification.

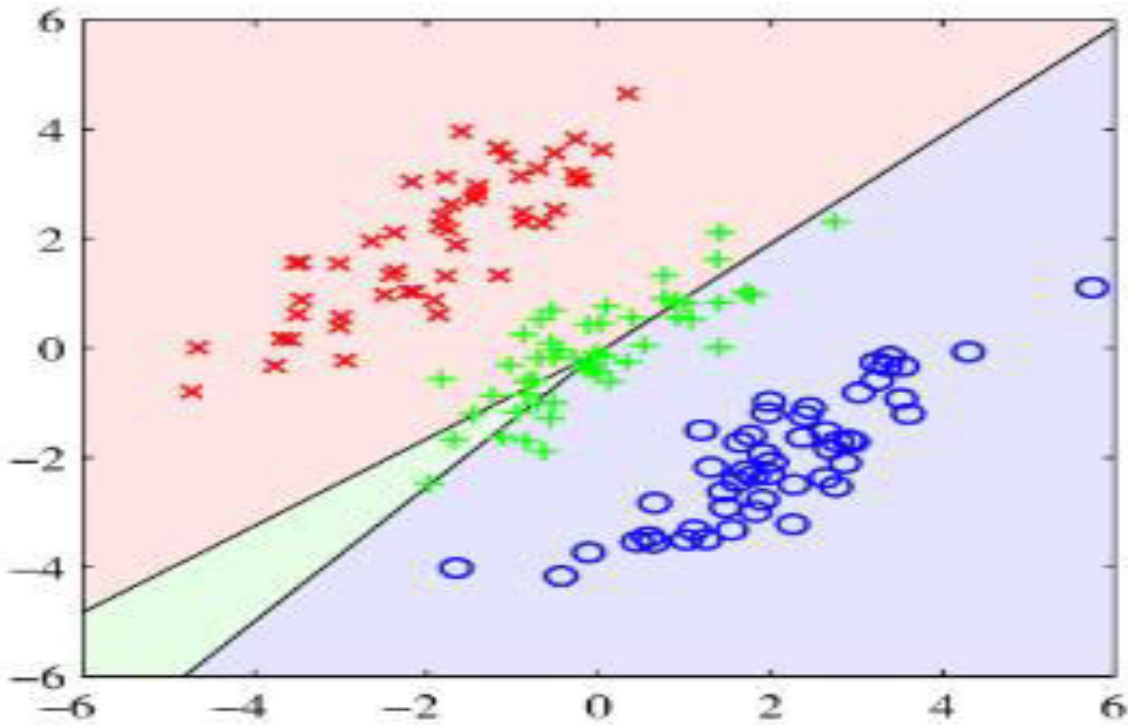
Biological Neuron



■ Components of BNN:

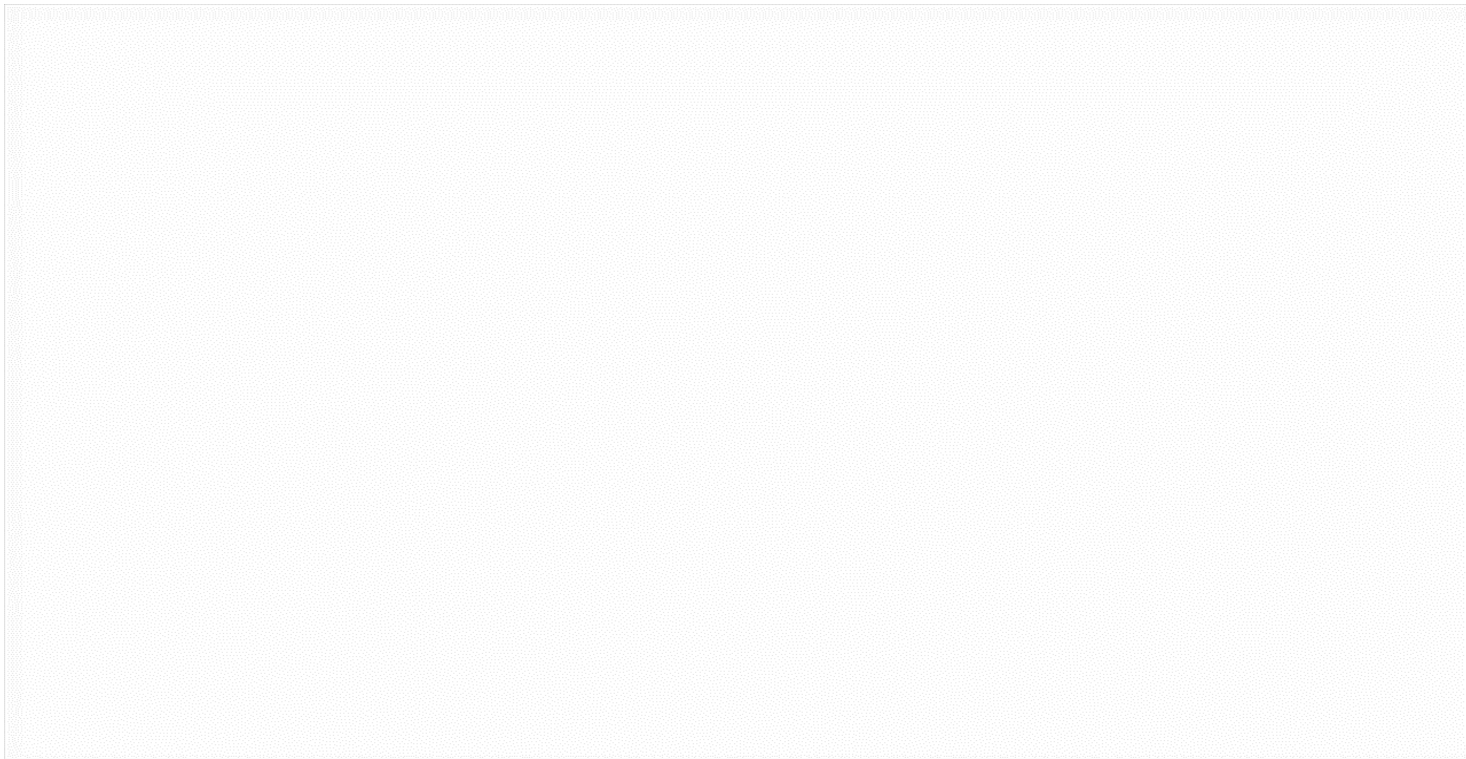
- Dendrites
- Axon
- Nucleus
- Cell Body
- Synapse

Contd..



*Nervous System: Get to know our nervous system a bit closer,
how does it works*

Link: [Click here to watch](#)





Biological Neuron

- ❑ A neuron (nerve cell) is the basic building block of the nervous system.
- ❑ A human brain consists of billions of neurons that are interconnected to each other. They are responsible for receiving and sending signals from the brain.
- ❑ Neurons are interconnected nerve cells in the human brain that are involved in processing and transmitting chemical and electrical signals.
- ❑ Dendrites are branches that receive information from other neurons.

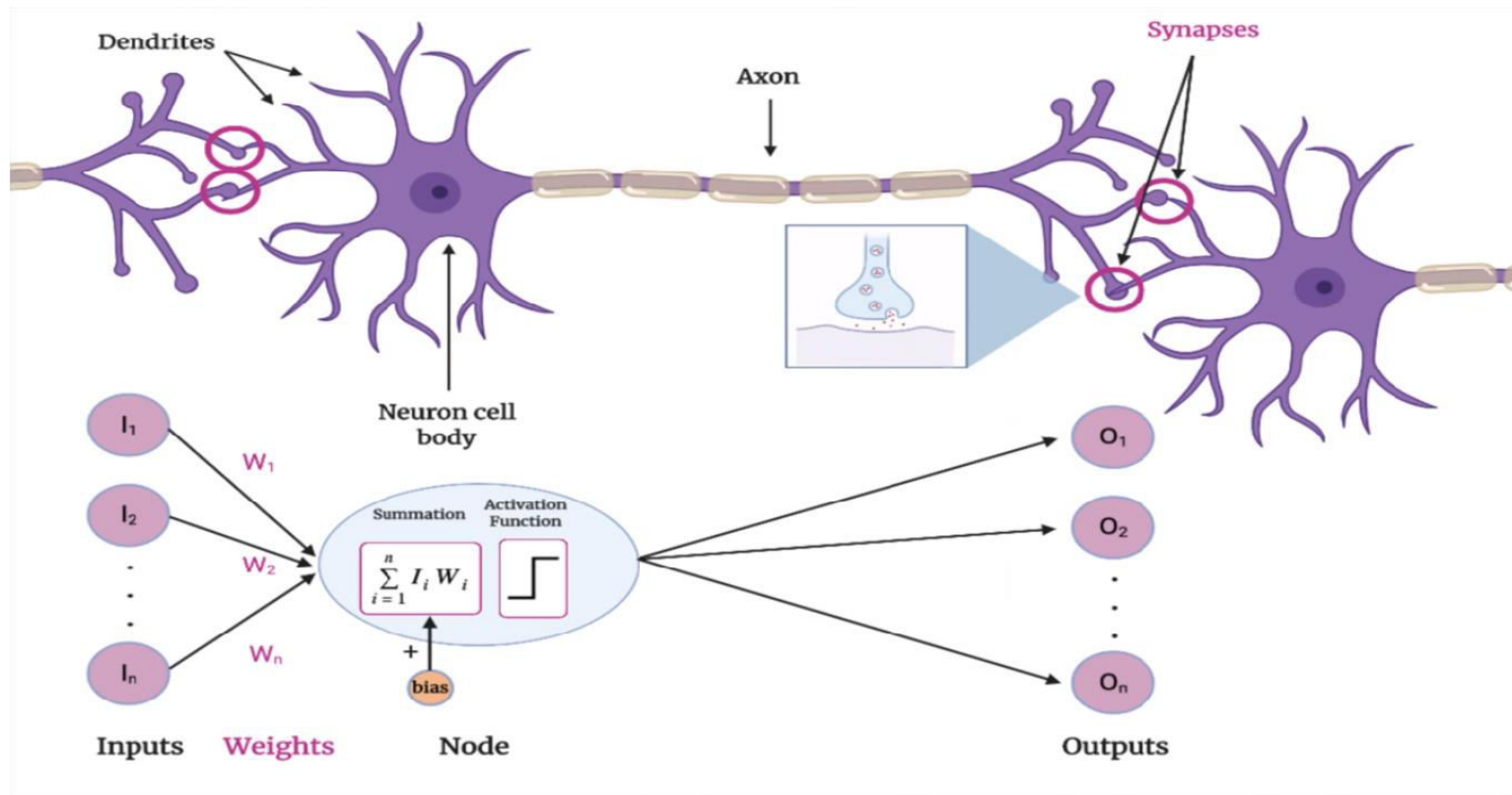
Contd..

- The transmission process involves an electrical impulse called 'action potential'.
- For the information to be transmitted, the input signals (impulse) should be strong enough to cross a certain threshold barrier, then only a neuron activates and transmits the signal further (output).

Perceptron Model

- Inspired by the biological functioning of a neuron, an American scientist Franck Rosenblatt came up with the concept of perceptron at Cornell Aeronautical Laboratory in 1957.
 - A neuron receives information from other neurons in form of electrical impulses of varying strength.
 - Neuron integrates all the impulses it receives from the other neurons.
 - If the resulting summation is larger than a certain threshold value, the neuron 'fires', triggering an action potential that is transmitted to the other connected neurons.

BNN vs ANN



Perceptron Model

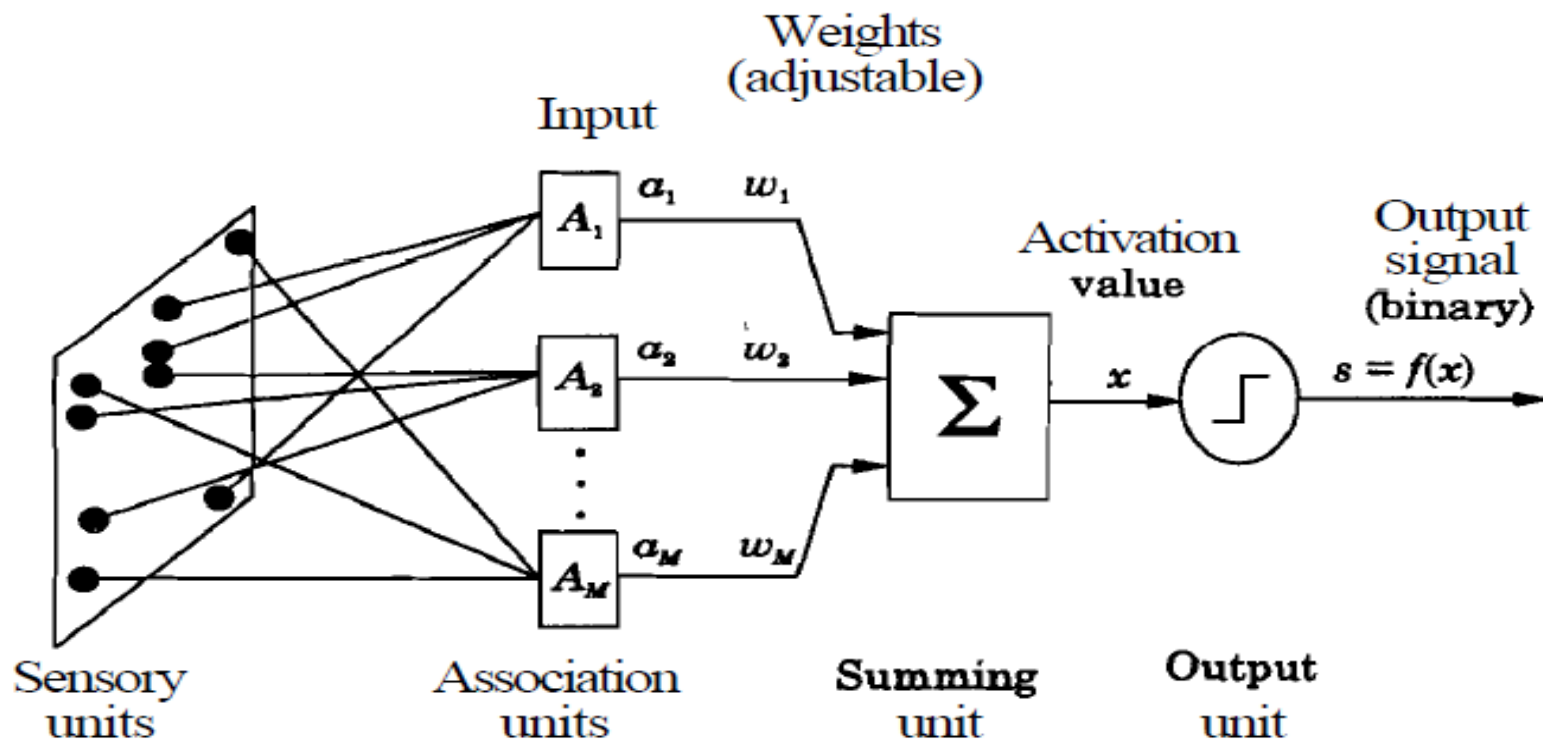


Figure Rosenblatt's perceptron model of a neuron.

Components of Perceptron

- Rosenblatt's perceptron is basically a binary classifier. The perceptron consists of following main parts:
 - Input nodes or input layer
 - Weights and Bias
 - Summation function
 - Activation function
 - Output
 - Weight update rule

Contd..

- **Input nodes or input layer:** The input layer takes the initial data into the system for further processing. Each input node is associated with a numerical value. It can take any real value.
- **Weights and bias:** Weight parameters represent the strength of the connection between units. Higher is the weight, stronger is the influence of the associated input neuron to decide the output. Bias plays the same as the intercept in a linear equation.
- **Activation function:** The activation function determines whether the neuron will fire or not. At its simplest, the activation function is a step function, but based on the scenario, different activation functions can be used.

Contd..

- **Summation Function:** The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.
- **Output:** The final output of the perceptron, is determined by the activation function's result. For example, in binary classification problems, the output might represent a predicted class (0 or 1).
- **Learning Algorithm (Weight Update Rule):** During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm. A common approach is the perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output.

Contd..

- The desired or target output (b) is compared with the actual binary output (s), and the error (e) is used to adjust the weights.
- The following equations describe the operation of the perceptron model of a neuron:

Activation:
$$x = \sum_{i=1}^M w_i a_i - \theta$$

Output signal:
$$s = f(x)$$

Error:
$$e = b - s$$

Weight change:
$$\Delta w_i = \eta e a_i$$

where η is the learning rate parameter.

Contd..

$$\begin{aligned} y &= 1 && \text{if } \sum_{i=1}^n w_i * x_i \geq \theta \\ &= 0 && \text{if } \sum_{i=1}^n w_i * x_i < \theta \end{aligned}$$

Rewriting the above,

$$\begin{aligned} y &= 1 && \text{if } \sum_{i=1}^n w_i * x_i - \theta \geq 0 \\ &= 0 && \text{if } \sum_{i=1}^n w_i * x_i - \theta < 0 \end{aligned}$$

Numerical 1

- Implement Perceptron model for AND Gate. Consider the weight for input vector is $w_1 = 1.2$ and $w_2 = 0.6$, threshold = 1, learning rate = 0.5

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Solution

- **For Training Instance 1:** A= 0, B= 0 and Target = 0

$$\sum W_i * X_i = W_1 * X_1 + W_2 * X_2$$

$$\text{Sum} = 1.2 * 0 + 0.6 * 0 = 0$$

- This is not greater than the threshold of 1, so the output = 0, Here the target is same as calculated output.

Contd..

- **For Training Instance 2:** A=0, B=1 and Target = 0

$$\sum W_i * X_i = W_1 * X_1 + W_2 * X_2$$

$$\text{Sum} = 0 * 1.2 + 1 * 0.6 = 0.6$$

- This is not greater than the threshold of 1, so the output = 0. Here the target is same as calculated output.

- **For Training Instance 3: A=1, B=0 and Target = 0**

$$\sum W_i * X_i = W_1 * X_1 + W_2 * X_2$$
$$w_i.x_i = 1 * 1.2 + 0 * 0.6 = 1.2$$

- This is greater than the threshold of 1, so the output = 1. Here the target does not match with the calculated output.
- Hence we need to update the weights.

Weight update

$$w_i = w_i + n(t - o)x_i$$

$$w_1 = 1.2 + 0.5(0 - 1)1 = 0.7$$

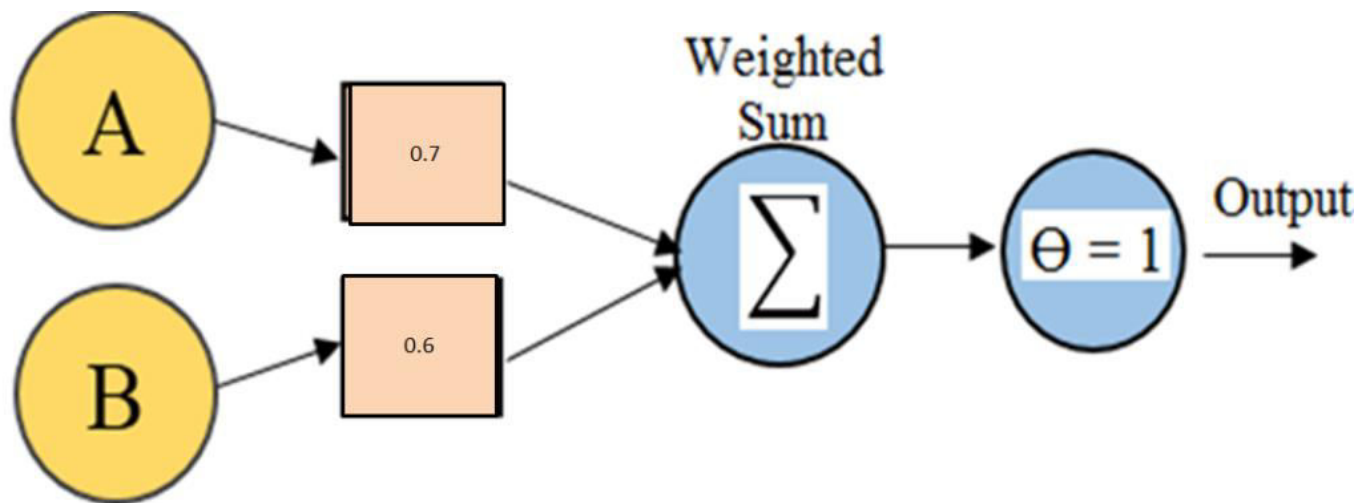
$$w_2 = 0.6 + 0.5(0 - 1)0 = 0.6$$

- Now, After updating weights are $w_1 = 0.7$, $w_2 = 0.6$ Threshold = 1 and Learning Rate $n = 0.5$

Contd..

- For Training Instance 1: $A=0$, $B=0$ and Target = 0
- For Training Instance 2: $A=0$, $B=1$ and Target = 0
- For Training Instance 3: $A=1$, $B=0$ and Target = 0
- For Training Instance 4: $A=1$, $B=1$ and Target = 1

Contd..



Home Assignment

- Implement Perceptron model for XOR gate. Consider initial weights are $w_1 = 1$, $w_2 = 2$, threshold = 1 and learning rate 1.5.

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Contd..

■ Learning Rate:

- It is defined as the step size taken to reach the minimum or lowest point.
- This is typically a small value that is evaluated and updated based on the behavior of the cost function.
- If the learning rate is high, it results in larger steps but also leads to risks of overshooting the minimum.
- At the same time, a low learning rate shows the small step sizes, which compromises overall efficiency but gives the advantage of more precision.



Lecture on

Linear Models for Classification: Perceptron Model

Perceptron Learning

- The desired or target output (b) is compared with the actual binary output (s), and the error (e) is used to adjust the weights.
- The following equations describe the operation of the perceptron model of a neuron:

Activation:
$$x = \sum_{i=1}^M w_i a_i - \theta$$

Output signal:
$$s = f(x)$$

Error:
$$e = b - s$$

Weight change:
$$\Delta w_i = \eta e a_i$$

where η is the learning rate parameter.

Numerical 1

- Implement Perceptron model for AND Gate. Consider the weight for input vector is $w_1 = 1.2$ and $w_2 = 0.6$, threshold = 1, learning rate = 0.5

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Weight update

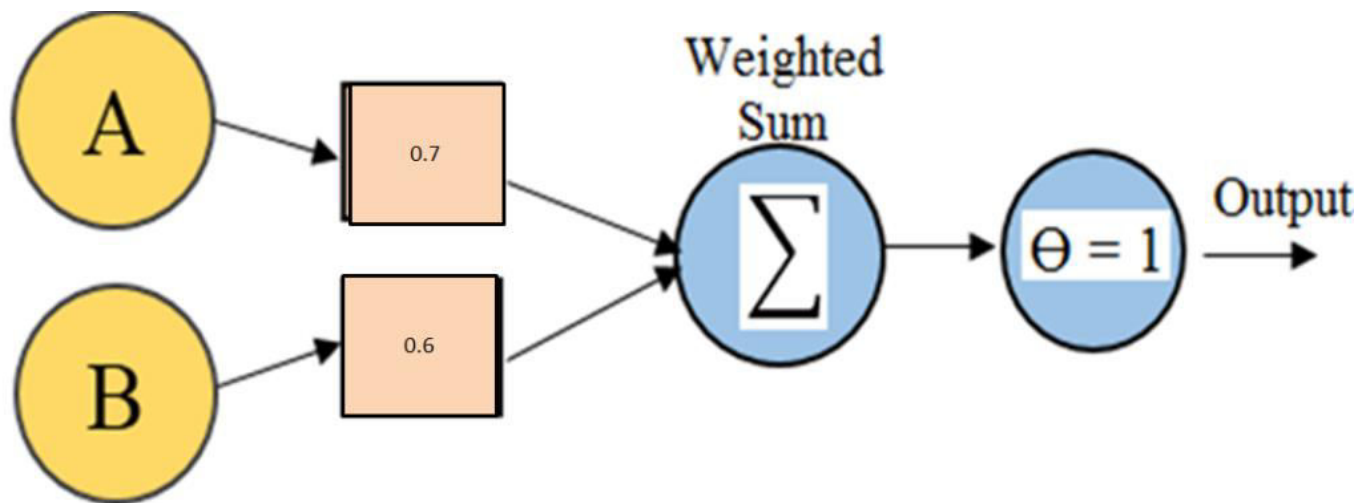
$$w_i = w_i + n(t - o)x_i$$

$$w_1 = 1.2 + 0.5(0 - 1)1 = 0.7$$

$$w_2 = 0.6 + 0.5(0 - 1)0 = 0.6$$

- Now, After updating weights are $w_1 = 0.7$, $w_2 = 0.6$ Threshold = 1 and Learning Rate $n = 0.5$


Contd..



Home Assignment 1

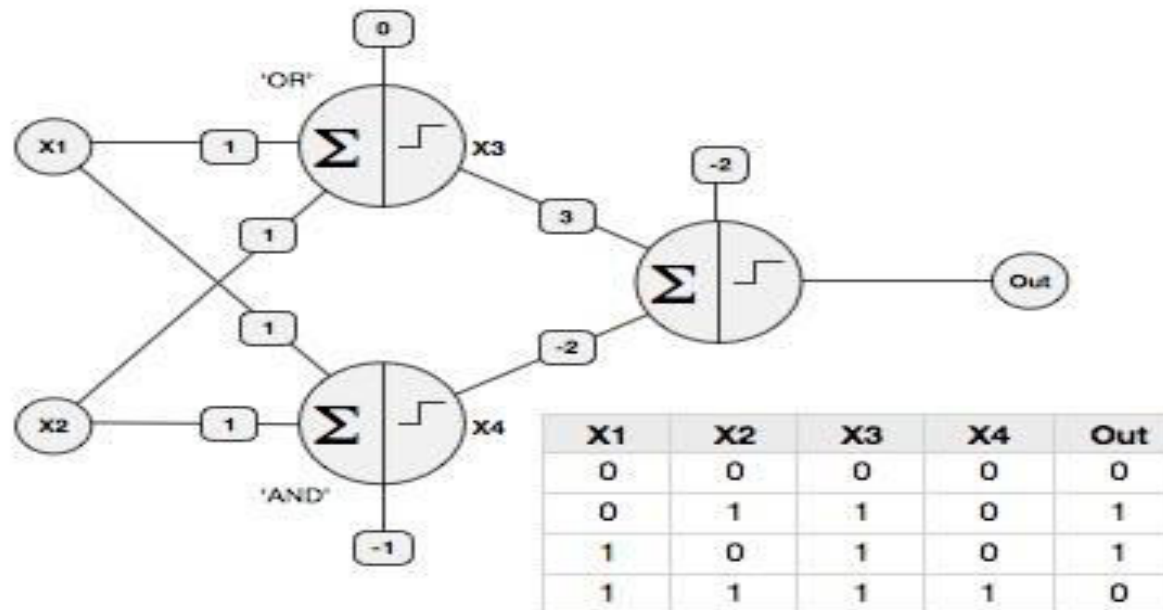
- Implement Perceptron model for XOR gate. Consider initial weights are $w_1 = 1$, $w_2 = 2$, threshold = 1 and learning rate 1.5.

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

- 
- Ex- OR gate can be implemented using following-
 - Non linear feature/functions
 - Function of NOR and AND
 - Function of own defined gate.

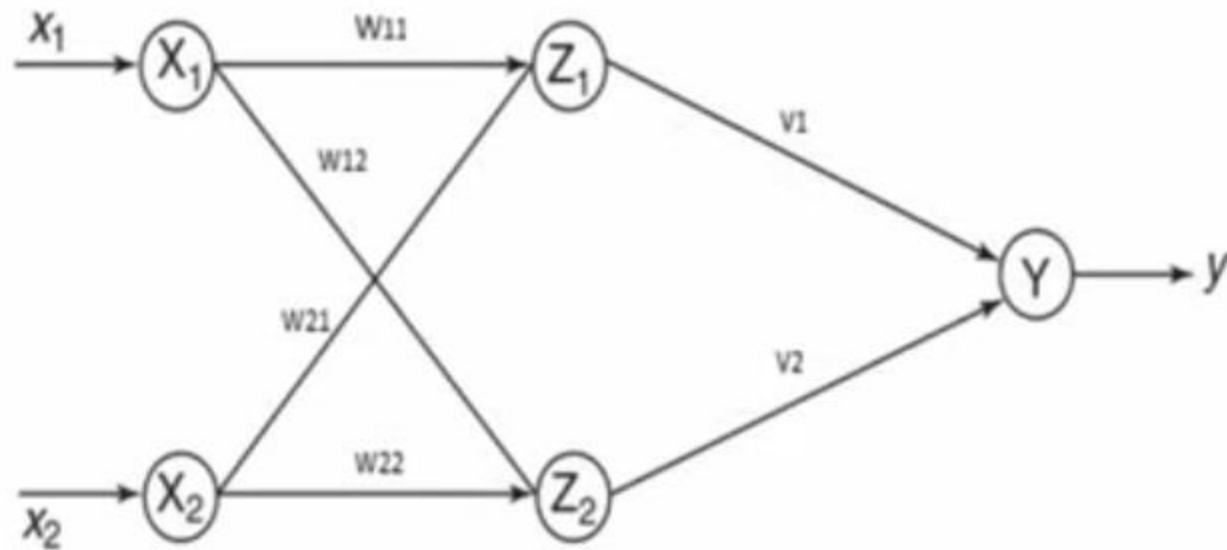
Contd..

- Ex OR gate can be defined as function of NOT and AND gate.
- $E\text{-OR} = \text{NOR}(\text{NOR}x1, x2), \text{AND}(x1, x2))$



- Lets take some own defined function to implement Ex-OR gate as:
- $Y = x_1 \cdot x_2' + x_1' \cdot x_2$
- Or
- $z_1 = x_1 \cdot x_2'$
- Where
- $Z_1 = x_1 \cdot x_2'$
- $Z_2 = x_1' \cdot x_2$
- $Y = z_1 \text{ OR } z_2$

Contd..



Contd..

$$y_{in} = z_1 v_1 + z_2 v_2$$

x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$

