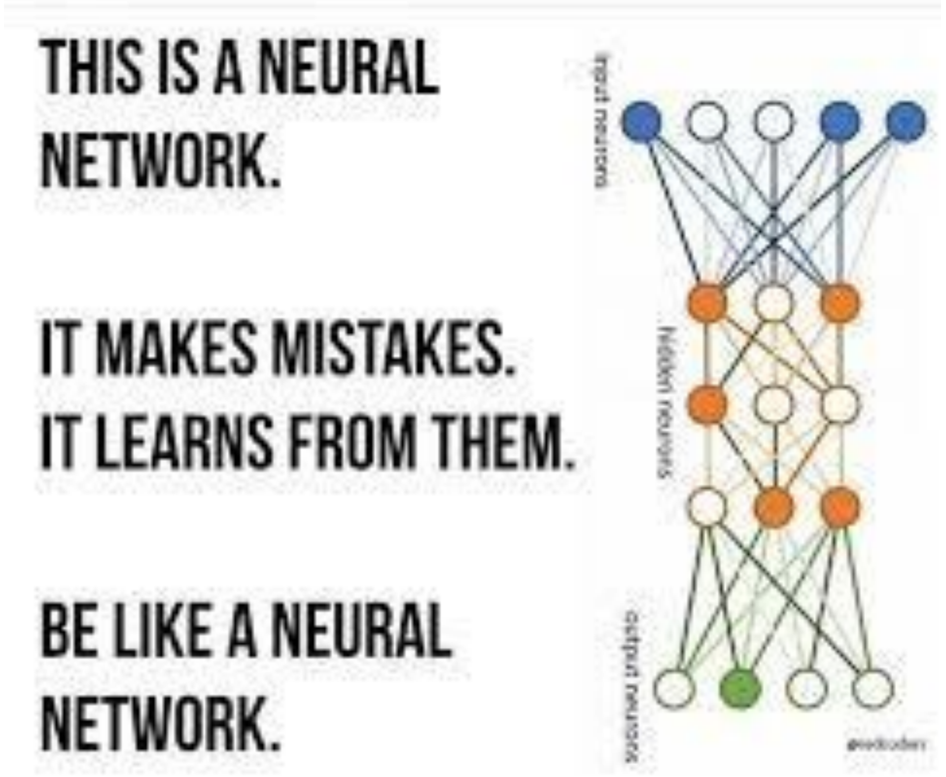
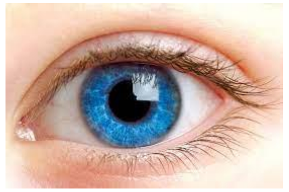
Introduction to Neural Networks

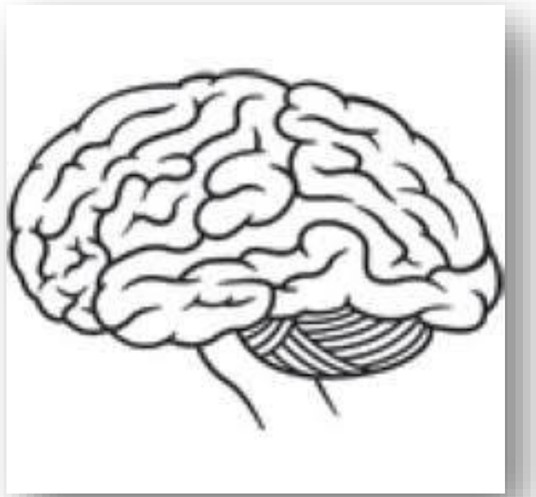


Neural Networks

- Neural networks are computing systems with interconnected nodes that work much like neurons in the human brain.
- Using algorithms, they can recognize hidden patterns and correlations in raw data, cluster and classify it, and – over time – continuously learn and improve.
- a neuron is just a node with many inputs and one output.
- A neural network consists of many interconnected neurons.
- A “simple” device that receives data at the input and provides a response.
- First, the neural network learns to correlate incoming and outcoming signals with each other — this is called learning.
- And then the neural network begins to work — it receives input data, generating output signals based on the accumulated knowledge.

Why only Human Brain

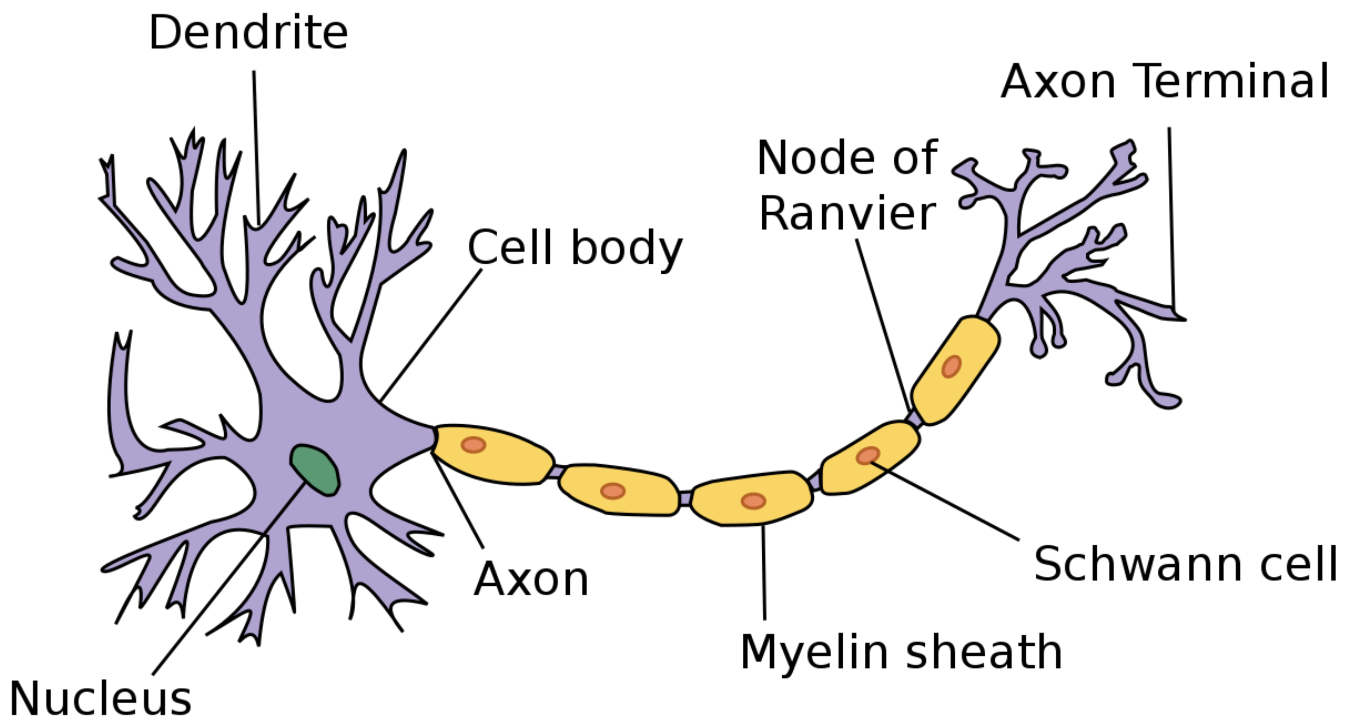




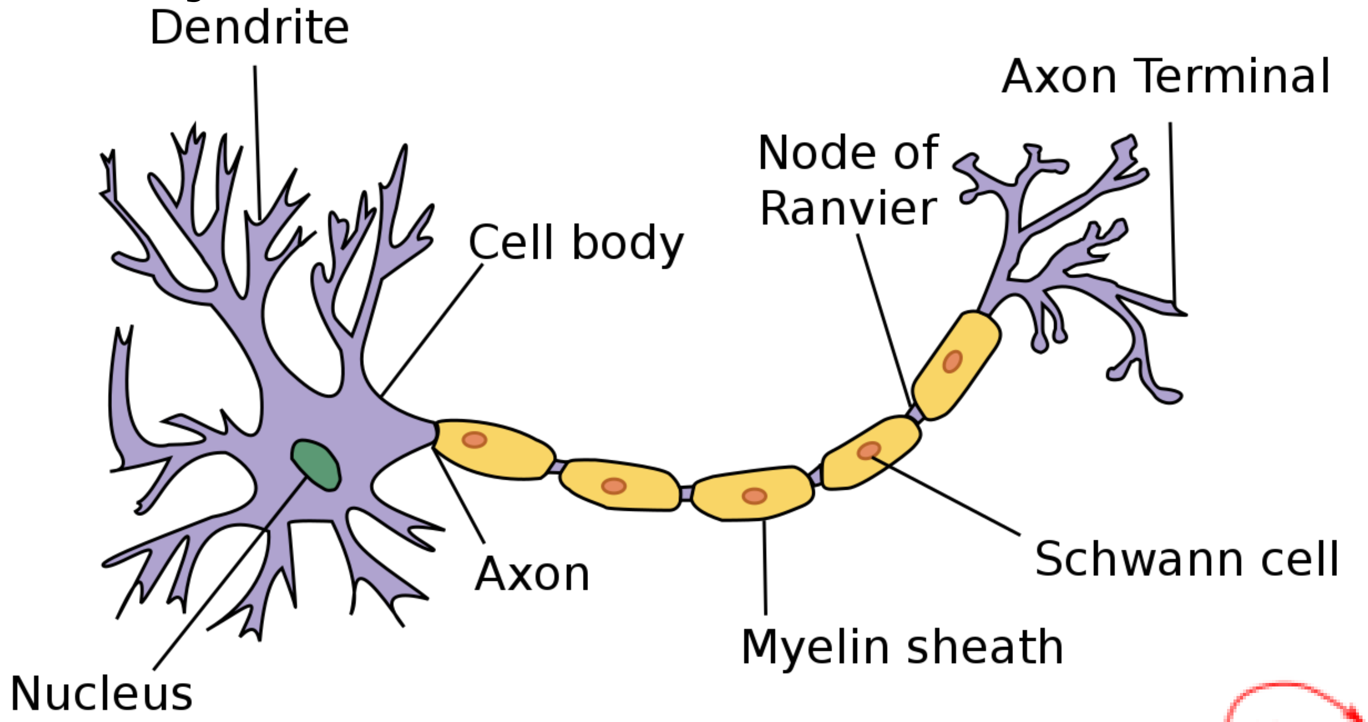
Audio Cortex

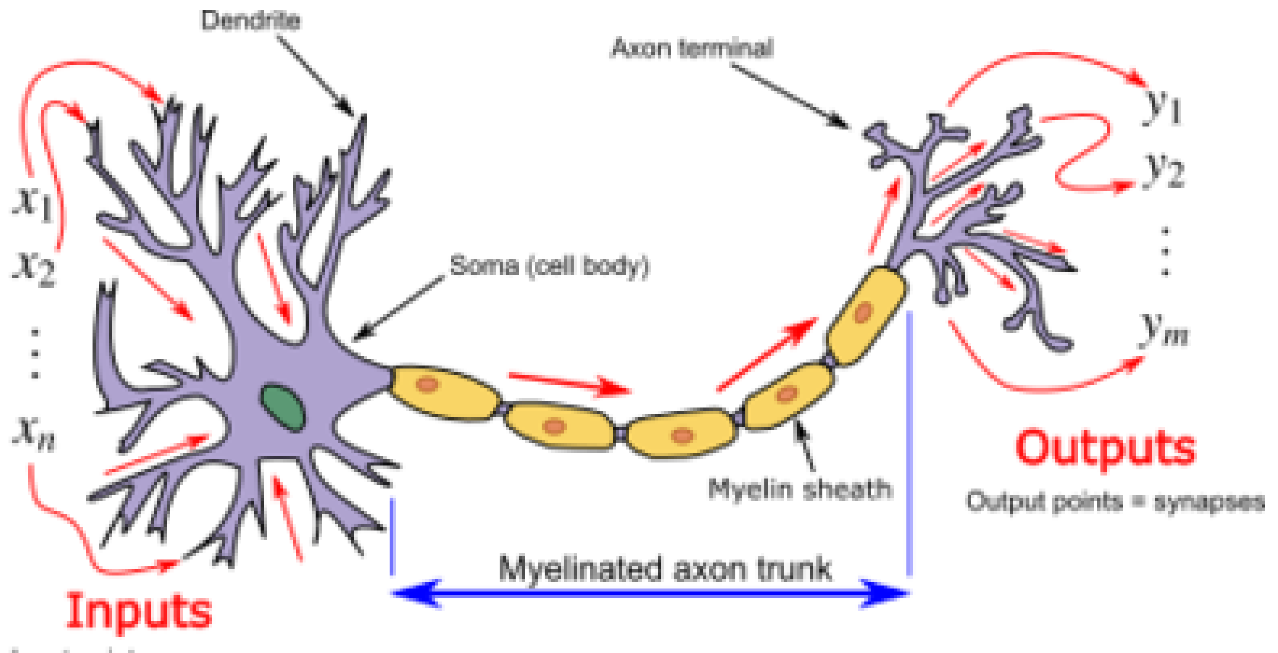
Visual Cortex

- The dendrite is where a neuron receives input from other neurons.
- The axon is the output of a neuron it transmits the signal to other neurons.
- The cell body contains a nucleus and genetic material, which controls the cell’s activities.
- Neurons communicate with each other by sending signals, called neurotransmitters, across a narrow space, called a synapse, between the axons of the sender neuron and dendrites of the receiver neuron.

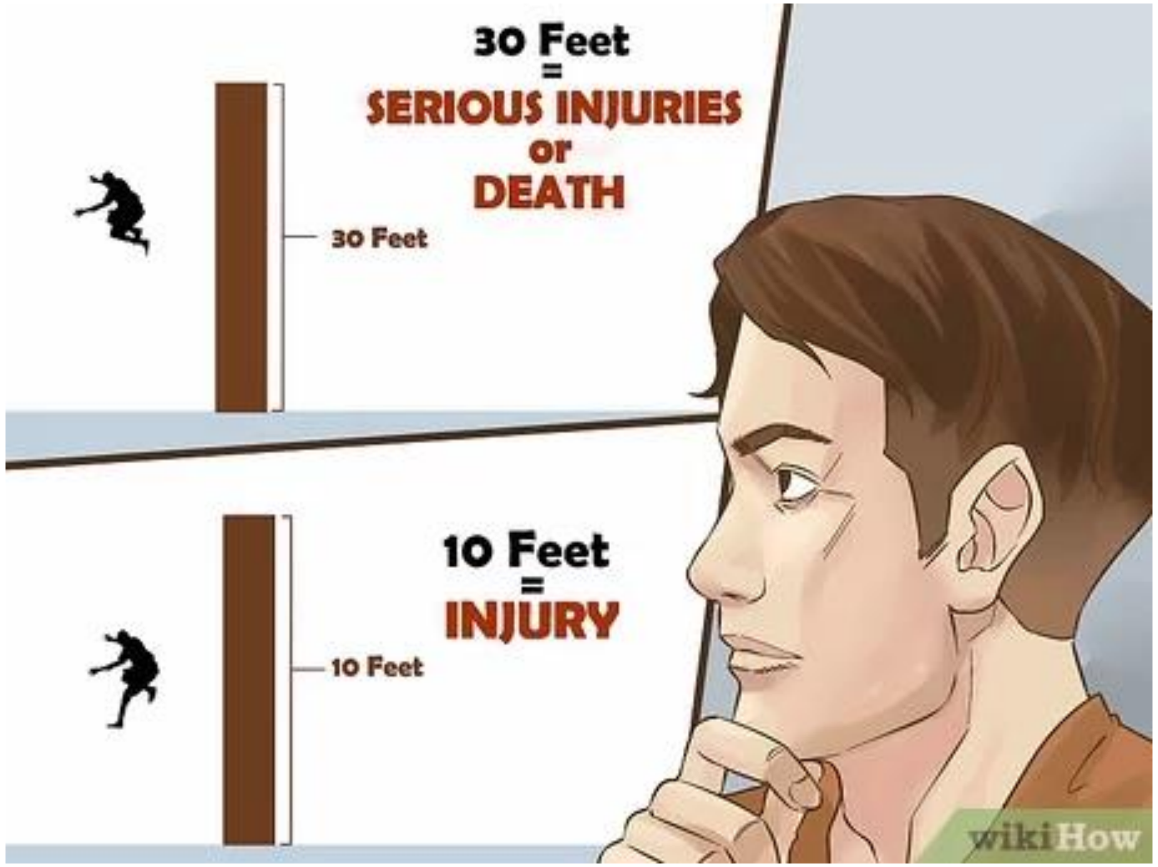


Why Neural Networks?





Analysis by Human brain



- Neural Networks are modelled to replicate Human Brain.
- The Key advantage of Neural Networks are that they are able to extract the features of the data automatically without a Programmer.
- First proposed in 1944 by Warren Mc Culloch and Walter Pitts.
- The main purpose is to develop Algorithms that mimics Human Brain

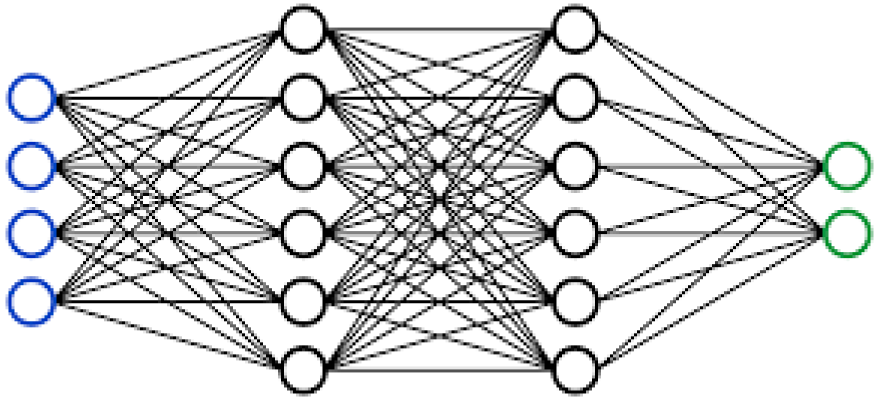
- The computational unit of Human Brain is called as Neuron.
- To provide computation we need,

  Input---------→Dendrites

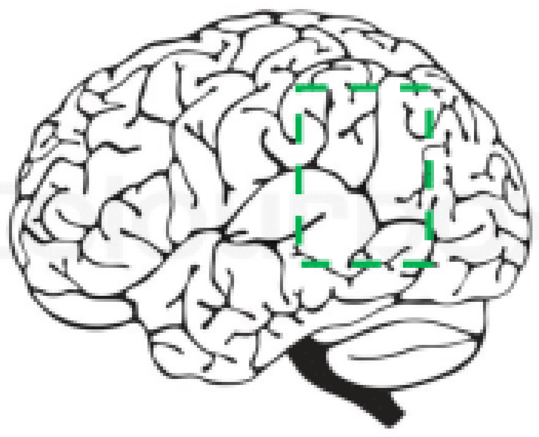
  Output-------→Axon

There are millions of Neurons that are interconnected, they transfer the data in the form of Electrical impulses.

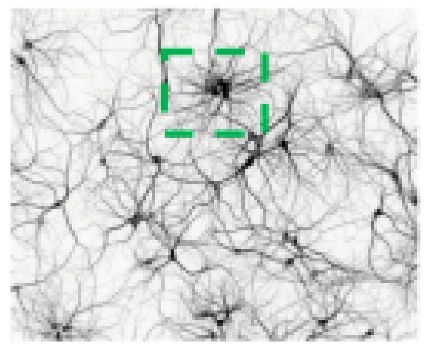
Interconnection of millions of Neurons forms a Brain



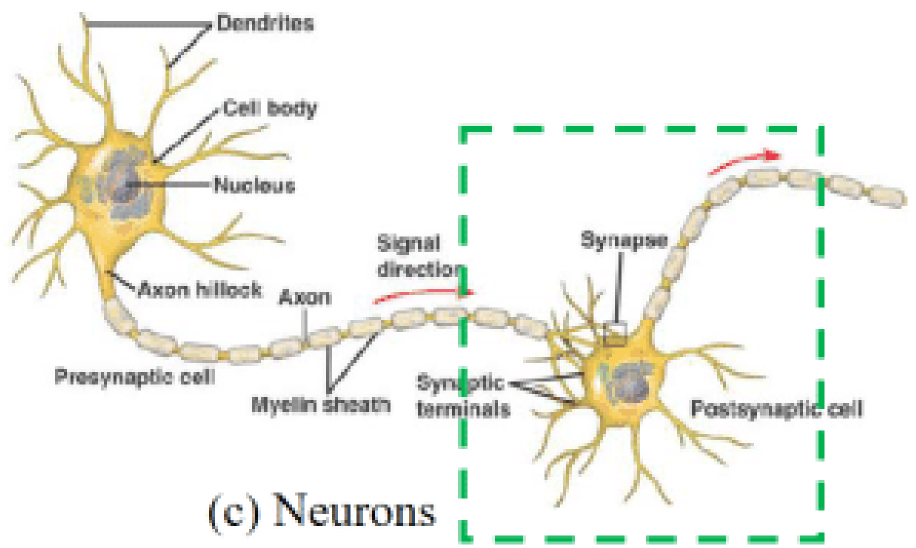
Artificial Neurons interconnected to form Artificial Neural Network



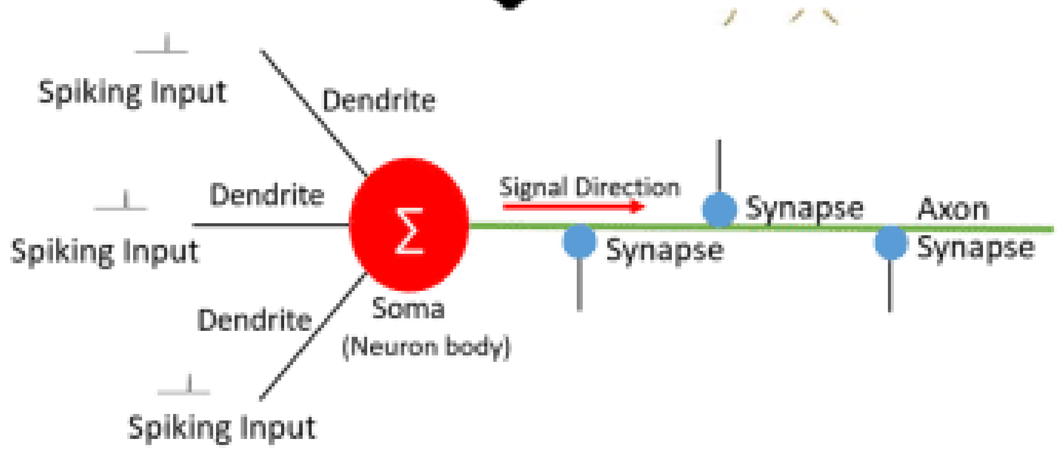
(a) Brain



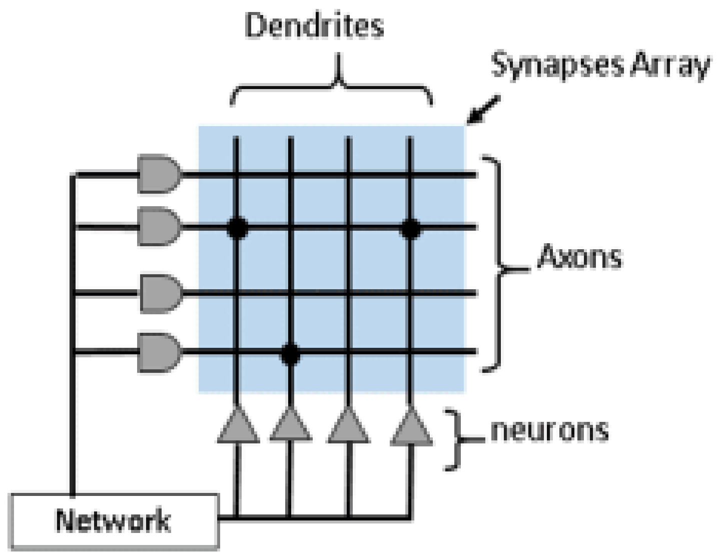
(b) Neuron network



(c) Neurons

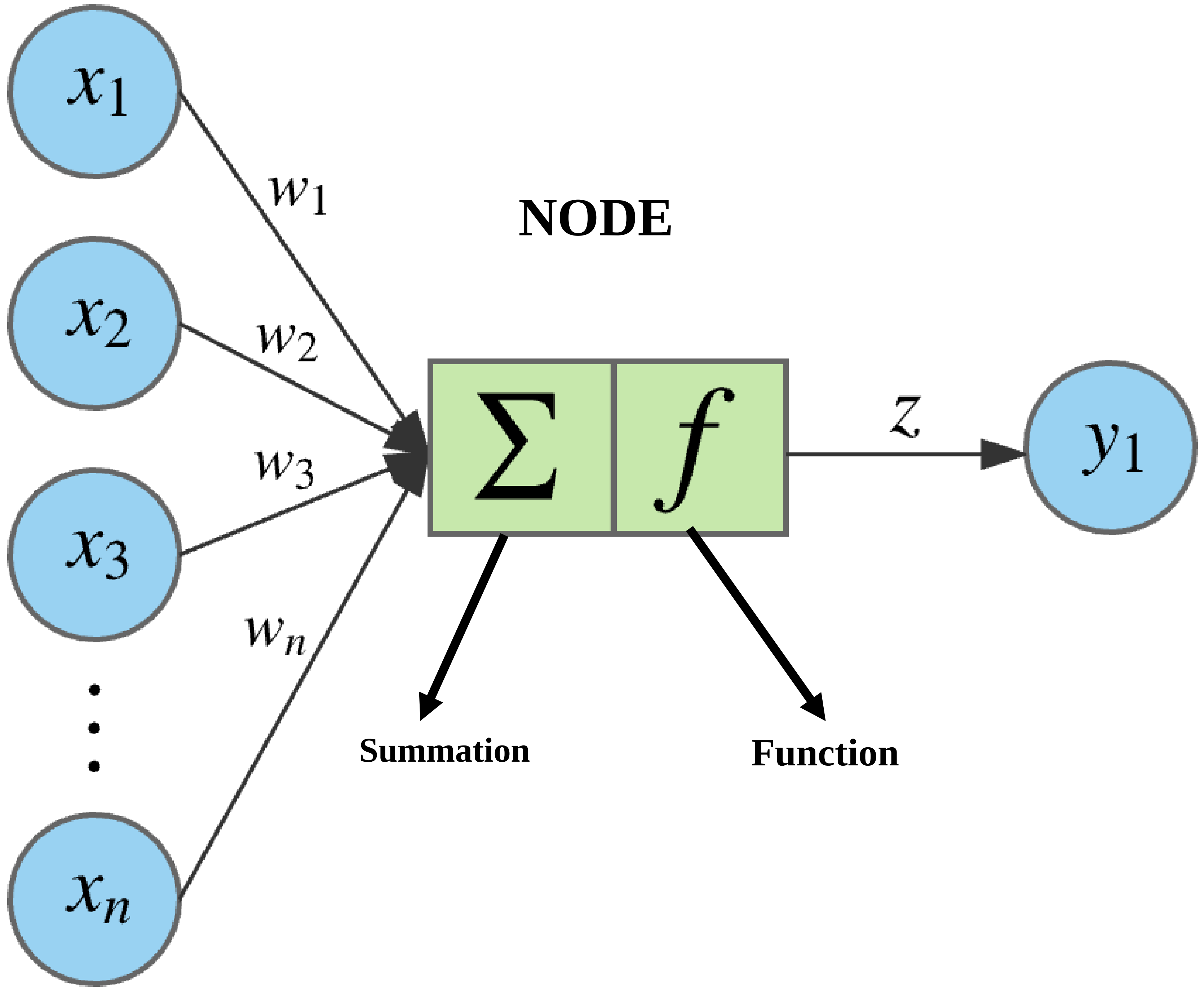


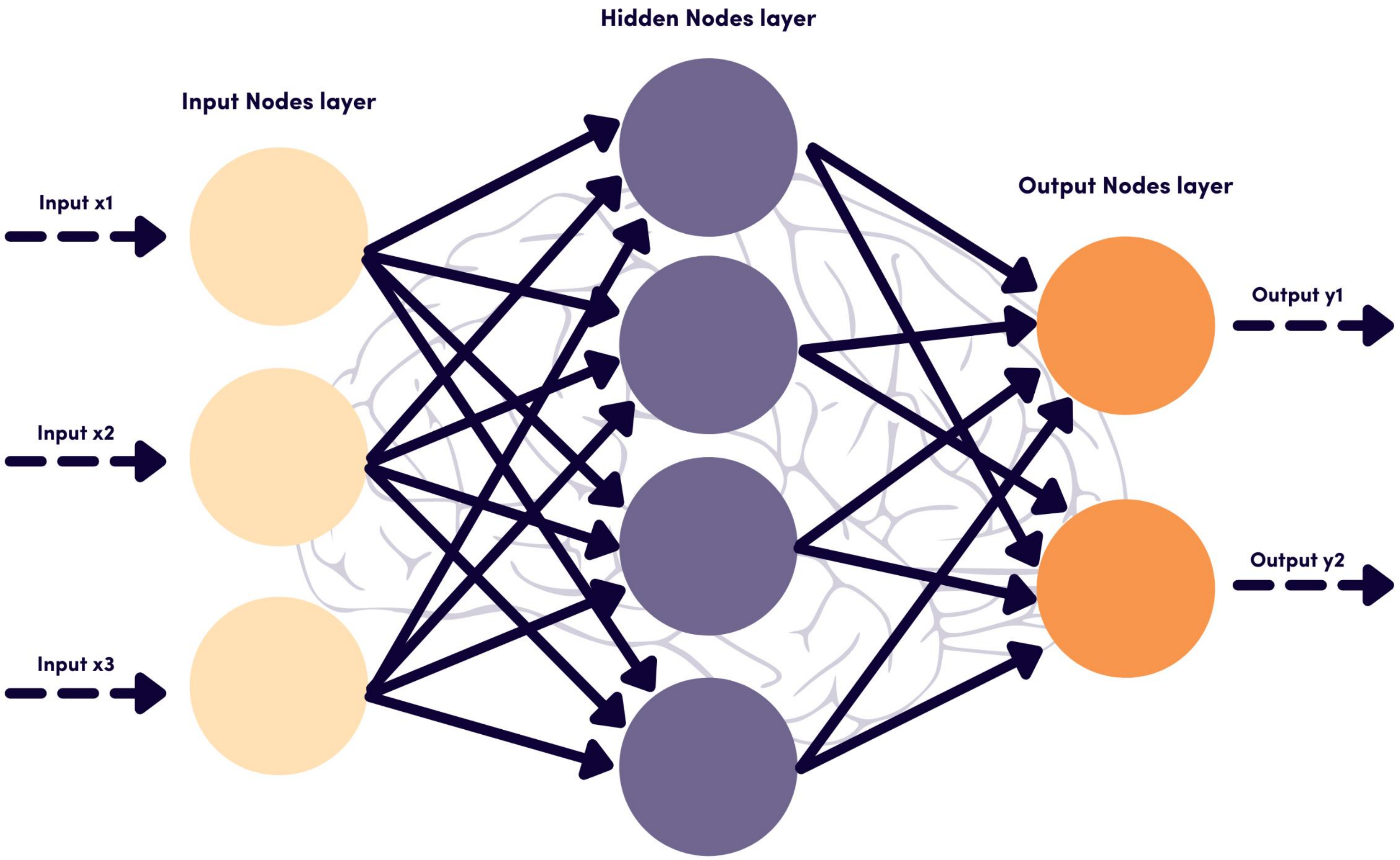
(d) Neuron Structure

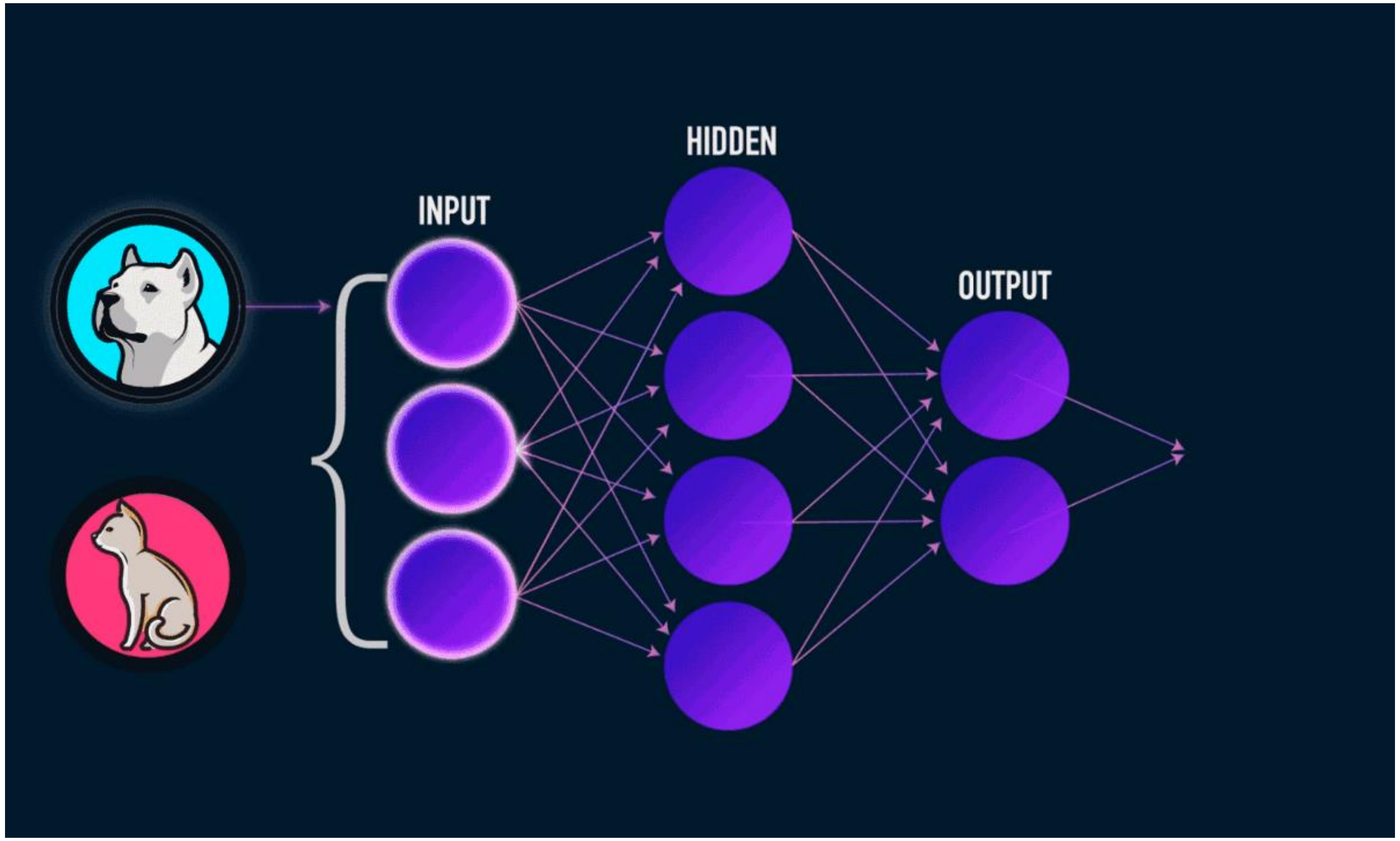


(e) Neuron Network

- When we receive an external stimulus like vision or sound, data travels as electrical signals through a path between neurons.
- The specific path is determined by the strength of inter-neuron connections, which itself is a cumulative result of all previous learning experiences.
- A neuron may get signals from many other neurons.
- If the sum of all input signals crosses its activation threshold, it transmits the signal to the next connection; otherwise the signal dies at that neuron.
- Thinking essentially involves taking the information from input neurons, progressively abstracting it through multiple connections among ‘thinking neurons’, finally leading to muscle instruction by output neurons.


x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_n
NODE
Σ
f
z
y_1
Summation
Function


Hidden Nodes layer
Input Nodes layer
Output Nodes layer
Input x1
Input x2
Input x3
Output y1
Output y2



dog and cat gif, the hidden layers are the ones responsible to learn that the dog picture is linked to the name dog, and it does this through a series of matrix multiplications and mathematical transformations to learn these mappings).

- input data, weights, a bias (or threshold), and an output. (Linear regression)

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + w_3 x_3 + bias$$

$$\text{output} = f(x) = \begin{cases} 1 \text{ if } \sum w_1 x_1 + b \geq 0 \\ 0 \text{ if } \sum w_1 x_1 + b < 0 \end{cases}$$

Types of Neural Networks

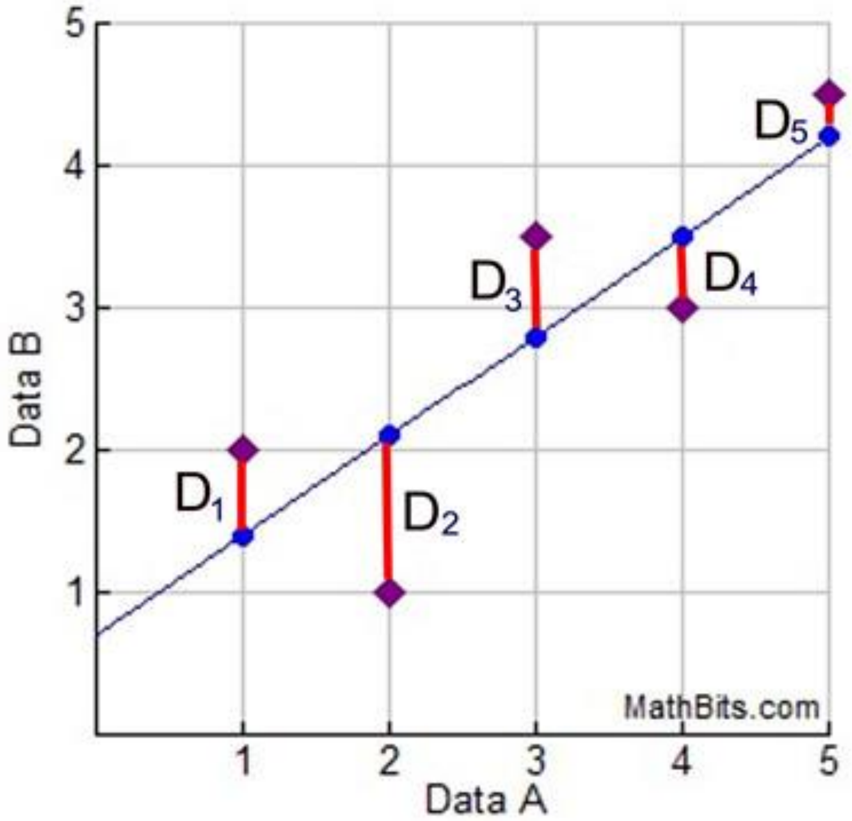
1. Artificial Neural Network(ANN/MLP)

2. Convolution Neural Network(CNN)

3. Recurrent Neural Networks(RNN)

Artificial Neural network

- Once an input layer is determined, weights are assigned.
- These weights help determine the importance of any given variable, with larger ones contributing more significantly to the output compared to other inputs.
- All inputs are then multiplied by their respective weights and then summed.
- Afterward, the output is passed through an activation function, which determines the output.
- If that output exceeds a given threshold, it “fires” (or activates) the node, passing data to the next layer in the network.
- This results in the output of one node becoming in the input of the next node. This process of passing data from one layer to the next layer defines this neural network as a feedforward network.



◆ Scatter Plot Points:

{(1,2), (2,1), (3,3½), (4,3), (5,4)}

● Regression Points

{(1,1.4), (2,2.1), (3,2.8), (4,3.5), (5,4.2)}

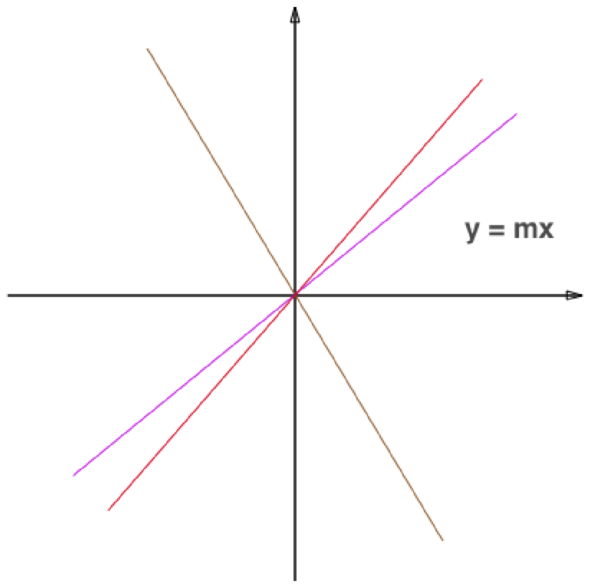
The Red Line Segments:

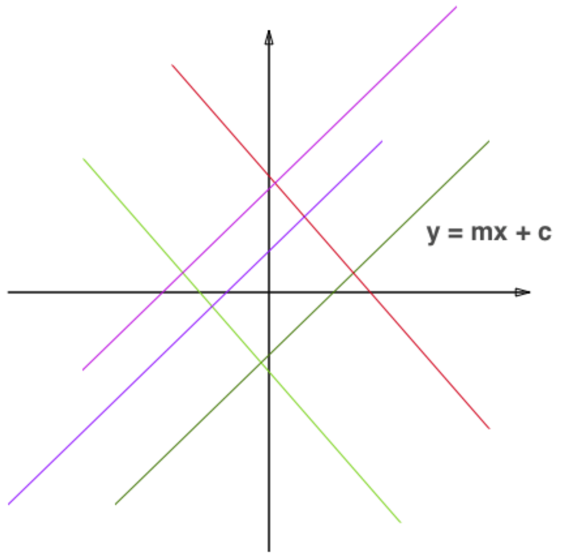
The red line segments represent the distances between the y-values of the actual scatter plot points, and the y-values of the regression equation at those points.

The lengths of the red line segments are called RESIDUALS.

Why bias???

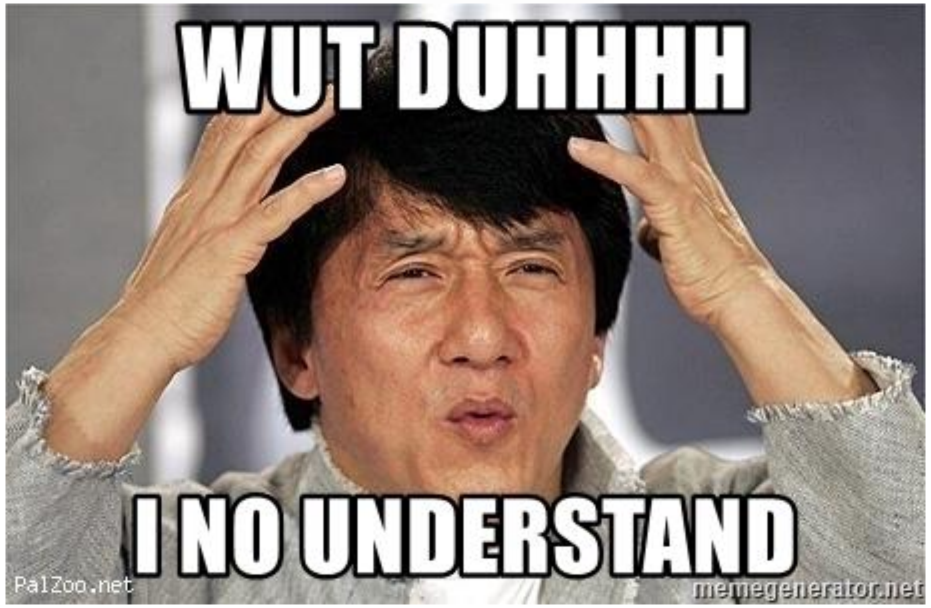
- Assume the model is having constraint to train itself and find a line which passes only through the origin.

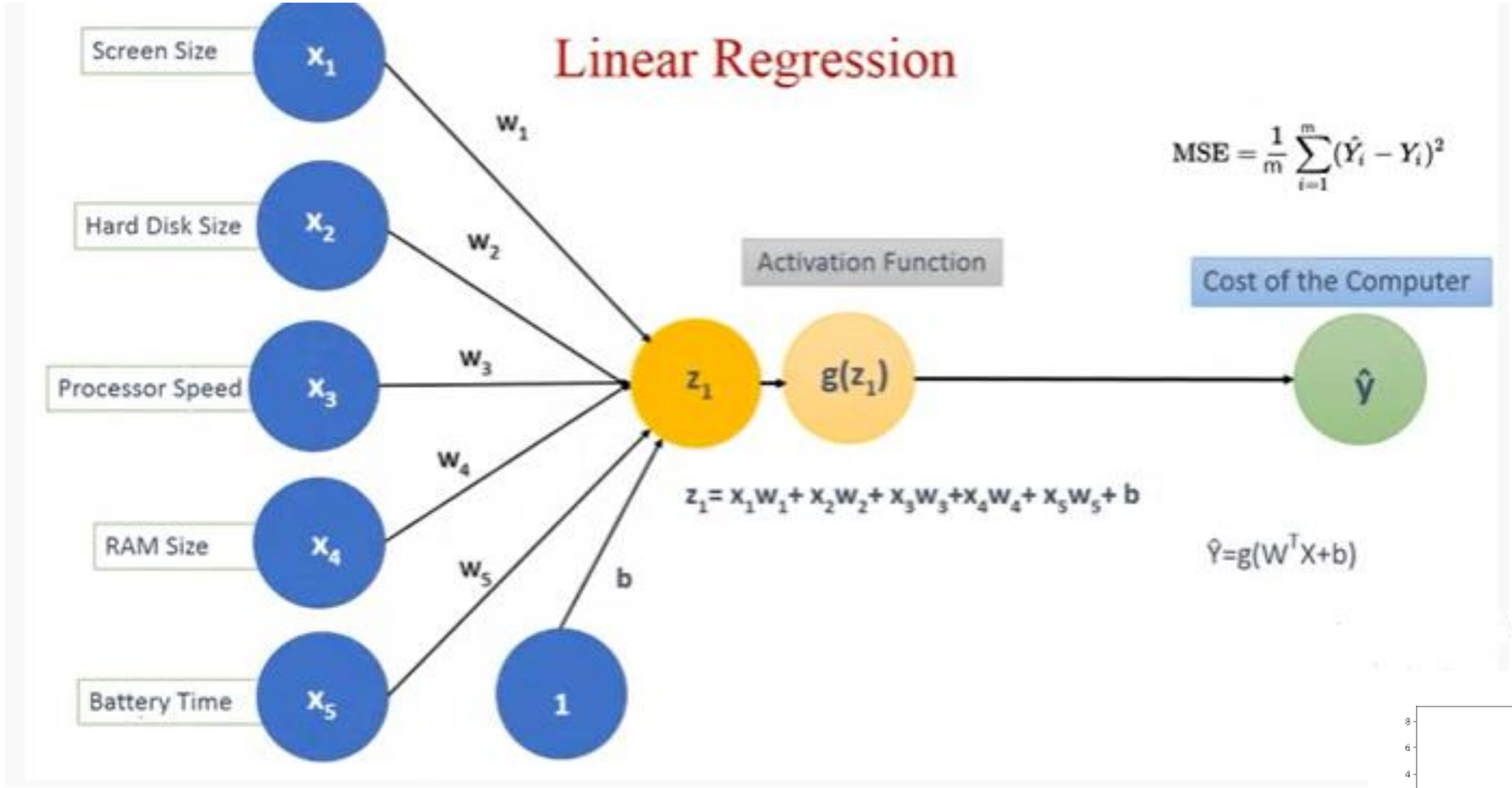


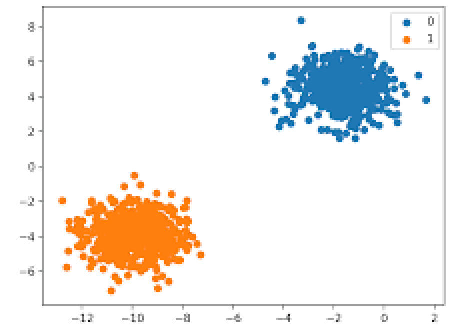


- **Bias** is a constant which helps the model in a way that it can fit best for the given data.
- In other words, **Bias** is a constant which gives freedom to perform best.

$$\theta_j = \theta_j - \alpha \times \frac{1}{m} \sum_{i=1}^{m} \underbrace{\left(h_\theta(x^{(i)}) - y^{(i)} \right)}_{\text{error}} \times \underbrace{x_j^{(i)}}_{\text{Features}}$$




Linear Regression
Screen Size
Hard Disk Size
Processor Speed
RAM Size
Battery Time
x_1
x_2
x_3
x_4
x_5
1
w_1
w_2
w_3
w_4
w_5
b
z_1
$g(z_1)$
Activation Function
Cost of the Computer
$\hat{y}$
$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{Y}_i - Y_i)^2$
$z_1 = x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 + x_5w_5 + b$
$\hat{Y} = g(W^T X + b)$


0
1

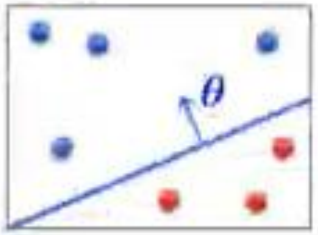
Perceptron

- In short, a perceptron is a single-layer neural network.
- They consist of four main parts including input values, weights and bias, net sum, and an activation function.
- A perceptron is a neural network unit that does a precise computation to detect features in the input data.
- Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.
- A Perceptron is an algorithm used for supervised learning of binary classifiers. Binary classifiers decide whether an input, usually represented by a series of vectors, belongs to a specific class.

Linear Classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$



$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \text{ where } \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

– Note that: $\boldsymbol{\theta}^\intercal \boldsymbol{x} > 0 \implies y = +1$

$\boldsymbol{\theta}^\intercal \boldsymbol{x} < 0 \implies y = -1$

- Perceptron – Linear Regression model with activation functions

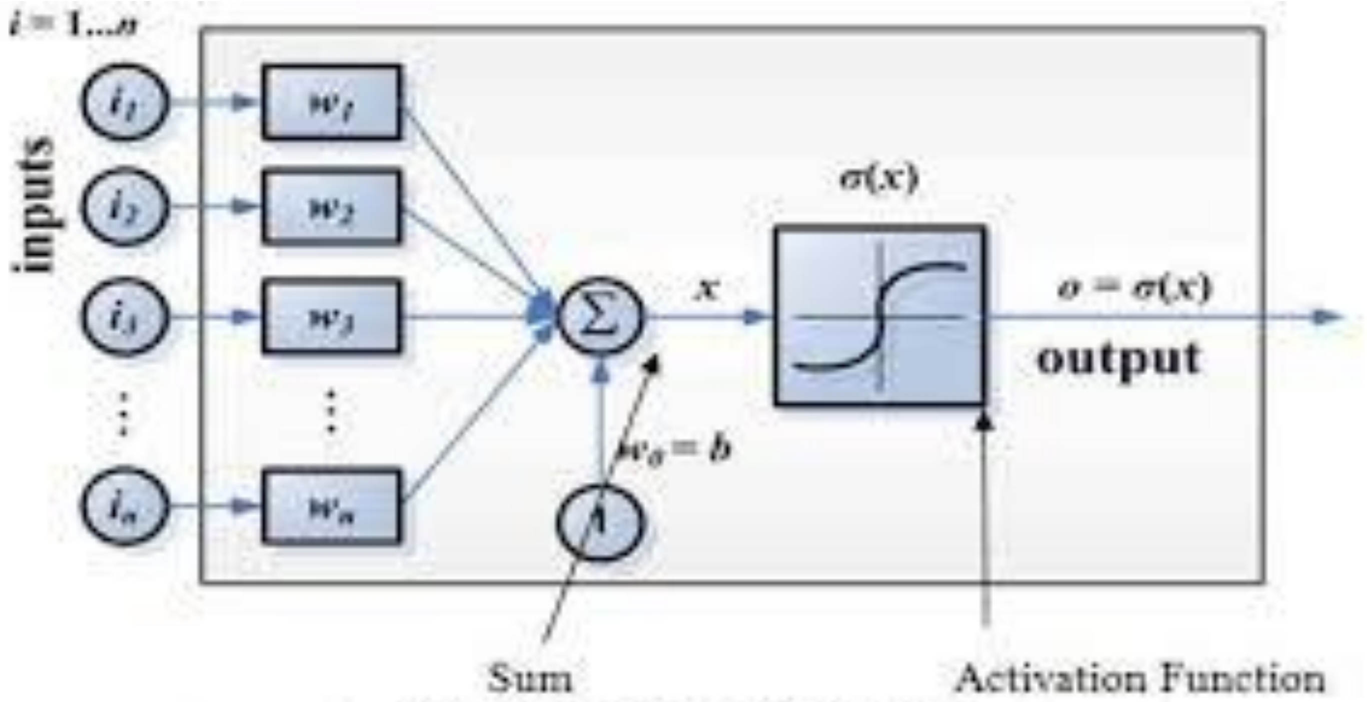
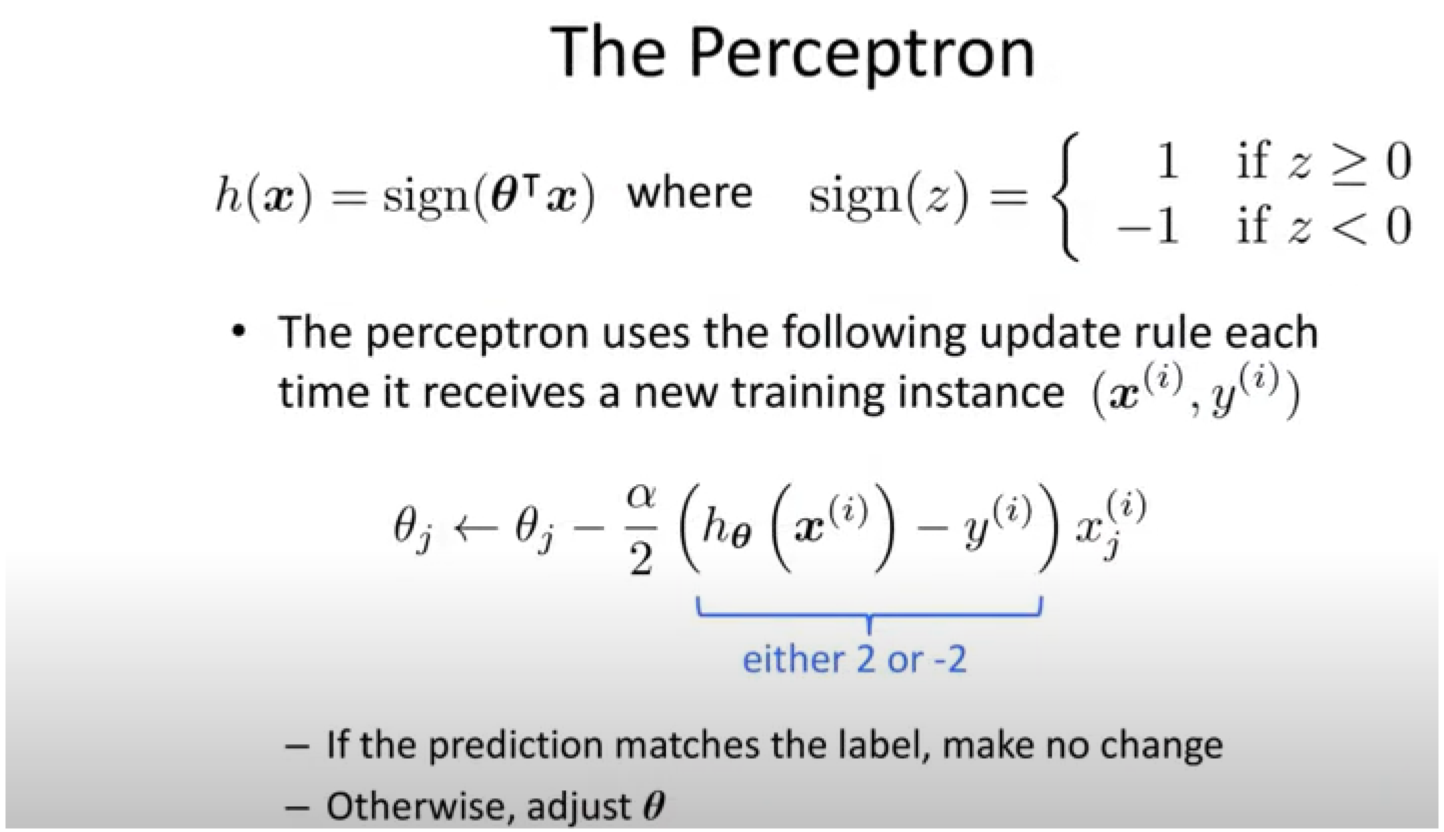


Figure 1: Structure of a typical Perceptron

an in the diagram show a typical perceptron will

Training the perceptron



The Perceptron

$$h(\boldsymbol{x}) = \operatorname{sign}(\boldsymbol{\theta}^\intercal \boldsymbol{x}) \quad \text{where} \quad \operatorname{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)}_{\text{either 2 or -2}} x_j^{(i)}$$

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{2}\underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

$\rightarrow$ Re-write as $\underbrace{\theta_j \leftarrow \theta_j + \alpha y^{(i)} x_j^{(i)}}$ (only upon misclassification)

The Perceptron

- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

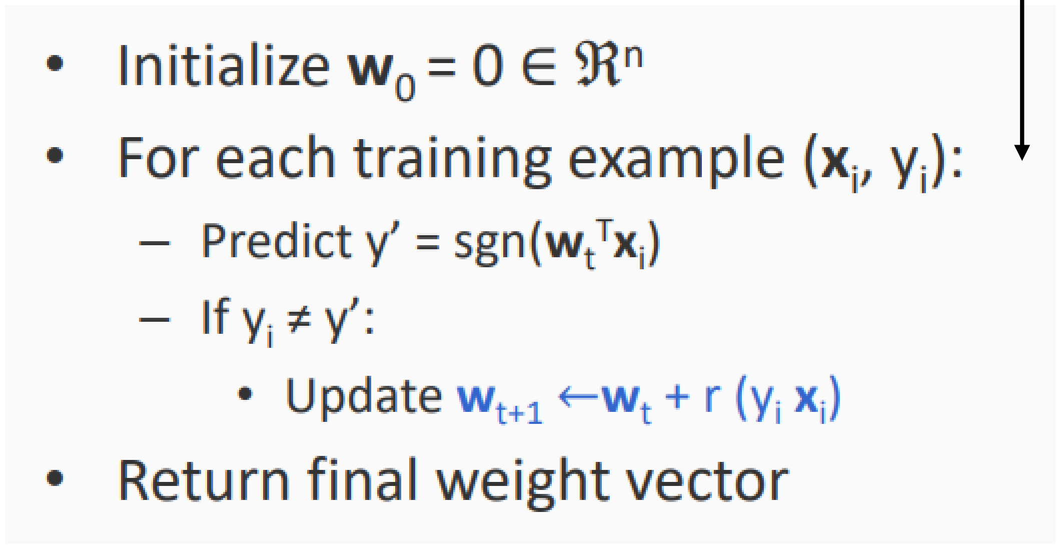
$$\theta_j \leftarrow \theta_j - \frac{\alpha}{2}\underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

- Re-write as $\theta_j \leftarrow \theta_j + \alpha y^{(i)} x_j^{(i)}$ (only upon misclassification)
  - Can eliminate $\cancel{\alpha}$ in this case, since its only effect is to scale $\boldsymbol{\theta}$ by a constant, which doesn't affect performance

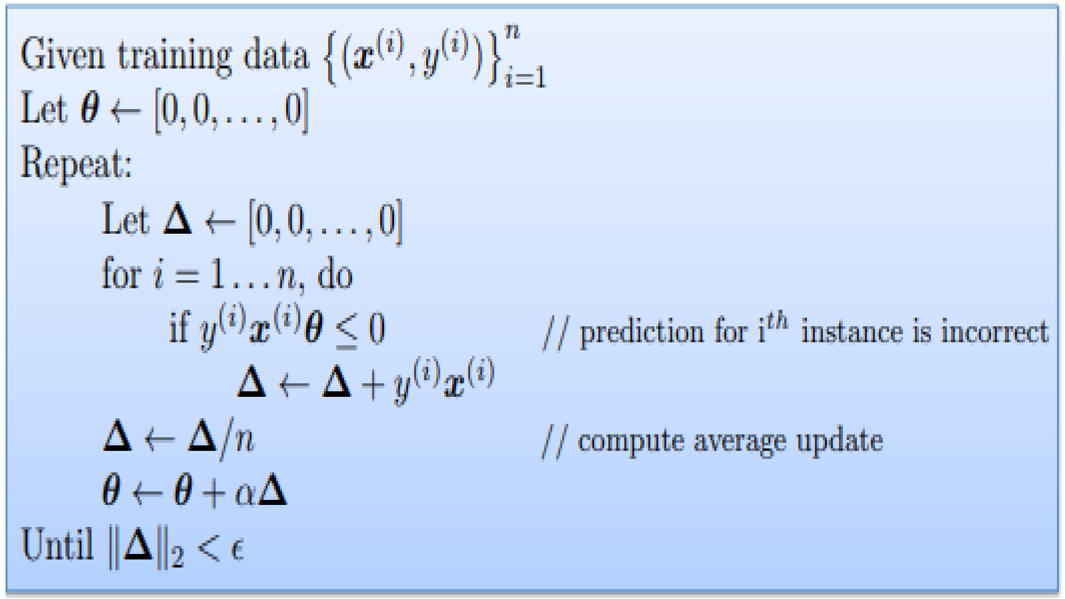
Perceptron Rule: If $\boldsymbol{x}^{(i)}$ is misclassified, do $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \underline{y}^{(i)}\underline{\boldsymbol{x}}^{(i)}$

Training

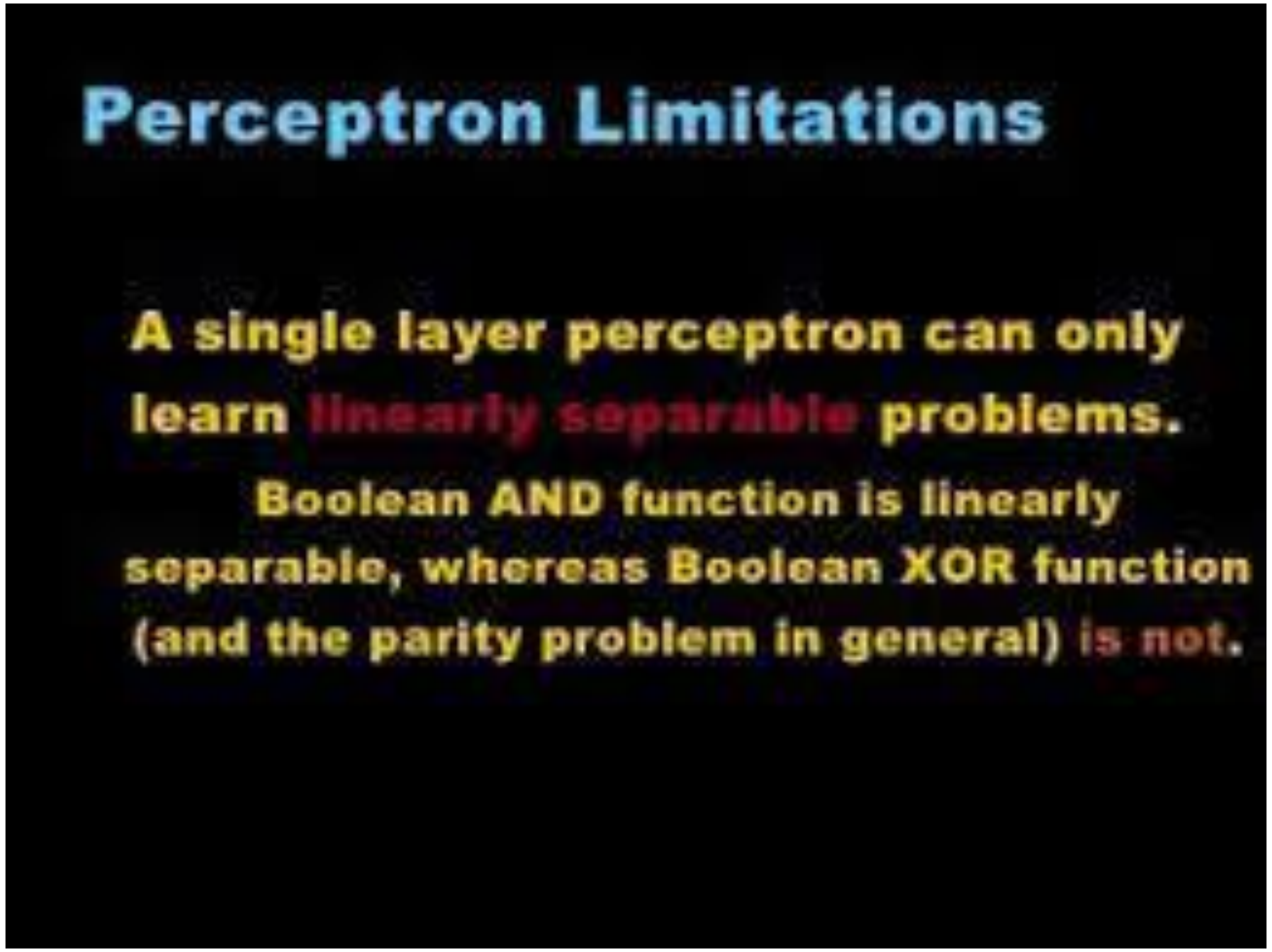
- Online training – single sample is used
- Batch Training- Entire dataset is used
- Mini batch learning- 10 instances



- Initialize $\mathbf{w}_0 = 0 \in \Re^n$
- For each training example $(\mathbf{x}_i, y_i)$:
  - Predict $y' = \text{sgn}(\mathbf{w}_t^T \mathbf{x}_i)$
  - If $y_i \neq y'$:
    - Update $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + r\,(y_i\,\mathbf{x}_i)$
- Return final weight vector



```
Given training data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    Let $\boldsymbol{\Delta} \leftarrow [0, 0, \ldots, 0]$
    for $i = 1 \ldots n$, do
        if $y^{(i)} \boldsymbol{x}^{(i)} \boldsymbol{\theta} \leq 0$        // prediction for i$^{th}$ instance is incorrect
            $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta} + y^{(i)} \boldsymbol{x}^{(i)}$
    $\boldsymbol{\Delta} \leftarrow \boldsymbol{\Delta}/n$        // compute average update
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \boldsymbol{\Delta}$
Until $\|\boldsymbol{\Delta}\|_2 < \epsilon$
```



Perceptron Limitations

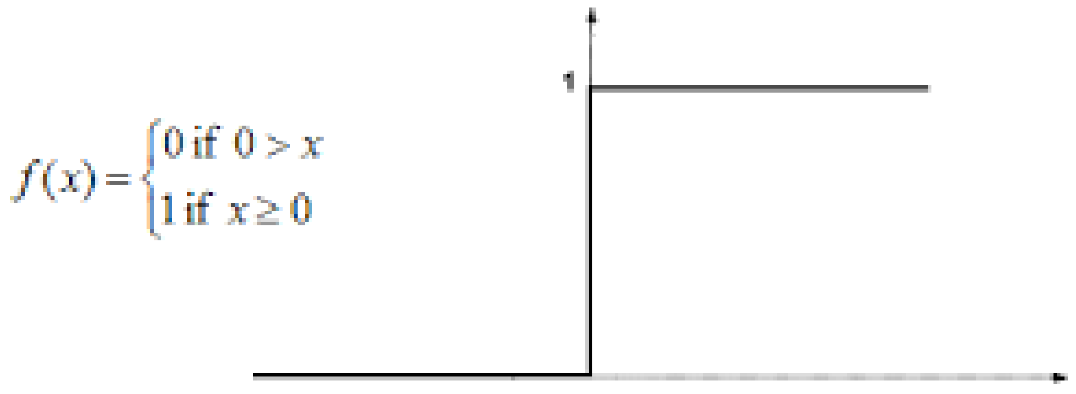
A single layer perceptron can only learn linearly separable problems.

Boolean AND function is linearly separable, whereas Boolean XOR function (and the parity problem in general) is not.

WE PRONOUNCE 22 AS TWENTY TWO, 33 AS THIRTY THREE, 44 AS FORTY FOUR,55 AS FIFTY FIVE
WHY NOT 11 AS ONETY ONE?

- It depends on θ^T .x

Unit step (threshold)



$$f(x) = \begin{cases} 0 \text{ if } 0 > x \\ 1 \text{ if } x \geq 0 \end{cases}$$