UNIT – 1
INTRODUCTION TO NEURAL NETWORK

Final Year
BTECH

Subject : **Deep Learning (PE3)**

Unit I : Contents

Introduction To Neural Network

Introduction, The architecture of an artificial neural network, Types of ANN architecture, Advantages and disadvantages of ANN, Perceptron, Sigmoid Neurons, Activation Functions, Loss Function.

Introduction, The architecture of an ANN, Types of ANN architecture, Advantages and Disadvantages of ANN

Sources:

- https://www.javatpoint.com/artificial-neural-network
- Introduction to Artificial Neural Systems by Jacek M. Zurada Yegnanarayana B, Artificial Neural Systems , PHP learning

What is Machine Learning?

- Artificial Intelligence (AI) systems learn by extracting patterns from input and output data.
- Machine Learning (ML) relies on learning patterns based on sample data. Programs learn from labeled data (supervised learning), unlabeled data (unsupervised learning), or a combination of both (semi-supervised learning).
- Artificial Intelligence (AI) came around in the middle of 1900s when scientists tried to envision intelligent machines. Machine Learning evolved in the late 1900s. This allowed scientists to train machines for AI.
- In the early 2000s, certain breakthroughs in multi-layered neural networks facilitated the advent of Deep Learning.

Deep Learning

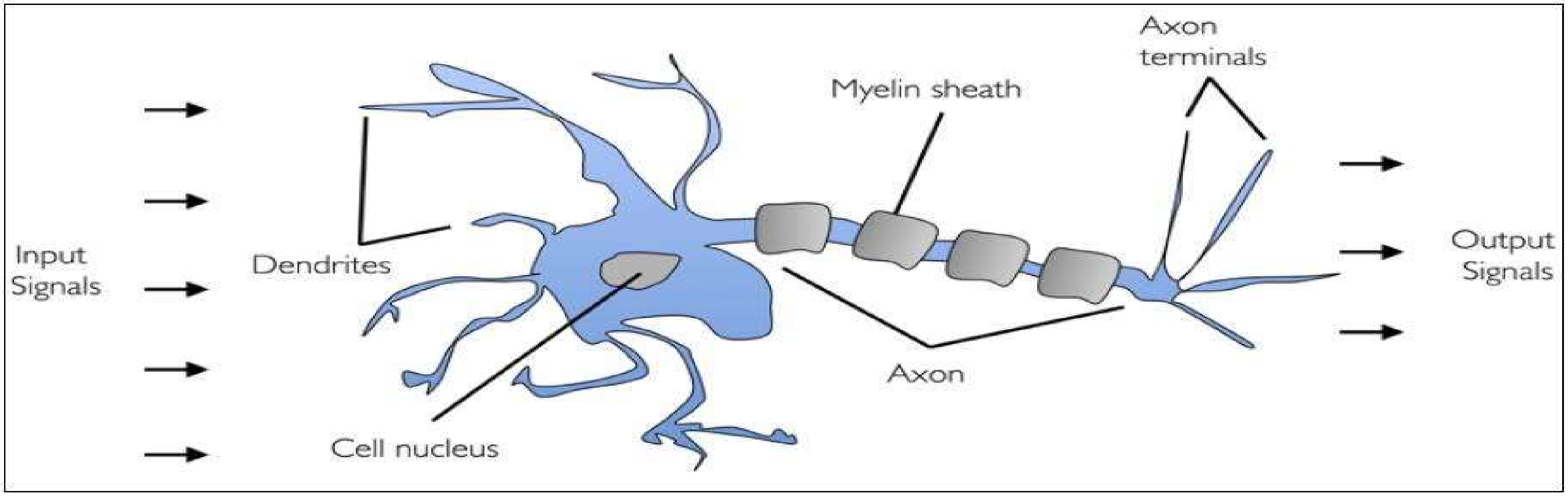
- DL –branch of ML, subset of AI
- It is a type of machine learning that works based on the structure and function of the human brain.
- Deep learning has gained massive popularity in scientific computing
- Its algorithms are widely used by industries that solve complex problems.
- All deep learning algorithms use different types of neural networks to perform specific tasks.
- Deep learning uses artificial neural networks to perform sophisticated computations on large amounts of data.

Deep Learning

- DL Models focus on accurate features by themselves
- DL model needs less or little guidance from programmer
- Useful in solving problem of dimensionality
- DL is used when input and output are huge in nos

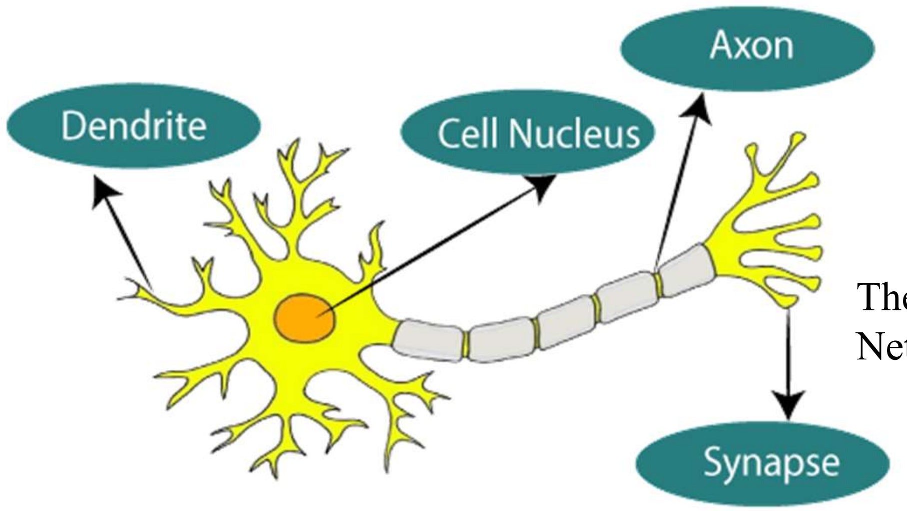
Introduction-ANN

- Biological Neuron
- A human brain has billions of neurons. **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.
- Cell nucleus or Soma processes the information received from dendrites. Axon is a cable that is used by neurons to send information. Synapse is the connection between an axon and other neuron dendrites.



What is Artificial Neural Network?

- The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain.
- Similar to the **human brain** that has **neurons interconnected** to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.

https://www.javatpoint.com/artificial-neural-network

Introduction-ANN

- Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943. This was called McCullock-Pitts (MCP) neuron. They described such a nerve cell as a simple logic gate with binary outputs.
- Multiple signals arrive at the dendrites and are then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon.
- ANN in the field of AI where it **attempts to mimic the network of neurons makes up a human brain** so that computers will have an option to understand things and make decisions in a human-like manner.
- It is designed by programming computers to behave simply like interconnected brain cells.
- There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000.

Artificial Neuron

- In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly.
- Human brain is made up of incredibly amazing parallel processors.
- A neuron is a mathematical function modeled on the working of biological neurons
- It is an elementary unit in an artificial neural network
- One or more inputs are separately weighted
- Inputs are summed and passed through a nonlinear function to produce output
- Every neuron holds an internal state called activation signal
- Each connection link carries information about the input signal
- Every neuron is connected to another neuron via connection link

What is Artificial Neural Network?

- ANN example-

Consider an example of a digital logic gate that takes an input and gives an output. "OR" gate,

It takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output.

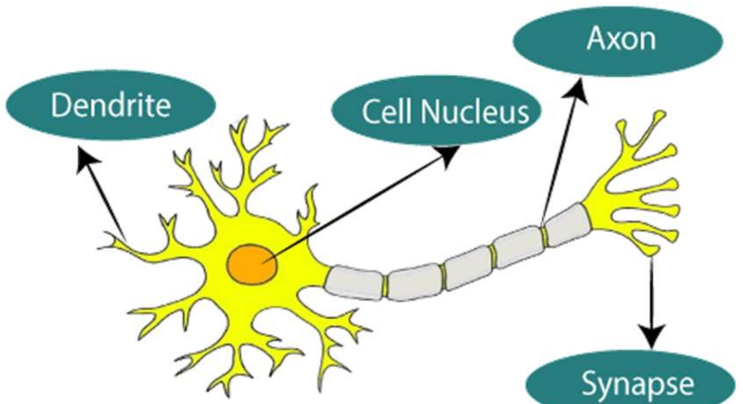
Here the output depends upon input.

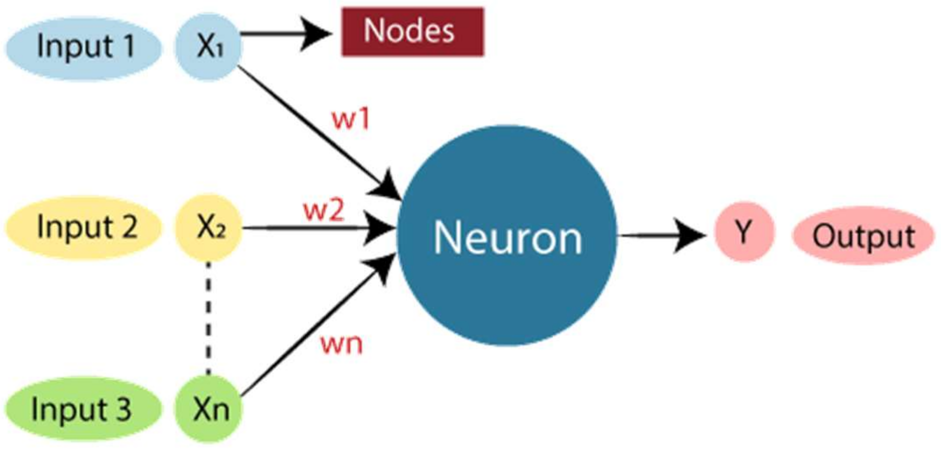
Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

Relationship between Biological neural network and artificial neural network

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output

Table 1: Neural network (biological and artificial)

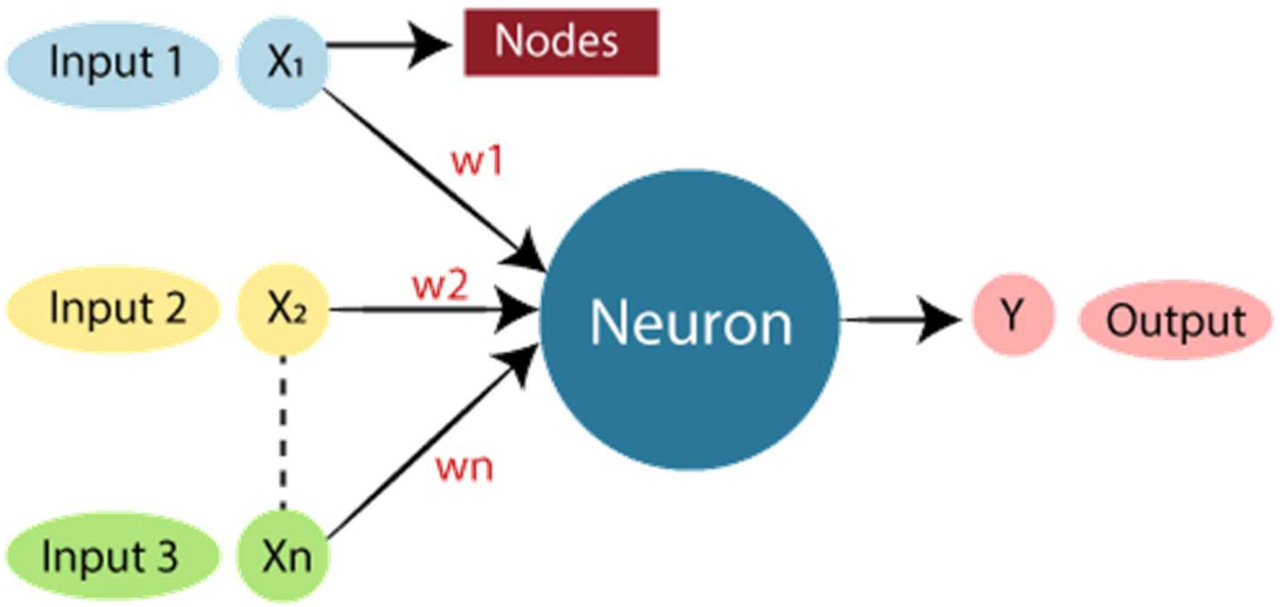




Typical Artificial Neural Network

- The typical Artificial Neural Network looks something like the given figure.
- Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.
- High level abstraction of neural input-output transformation

  Inputs → weighted sum of inputs → nonlinear function → output



The architecture of an artificial neural network

In order to define a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers.

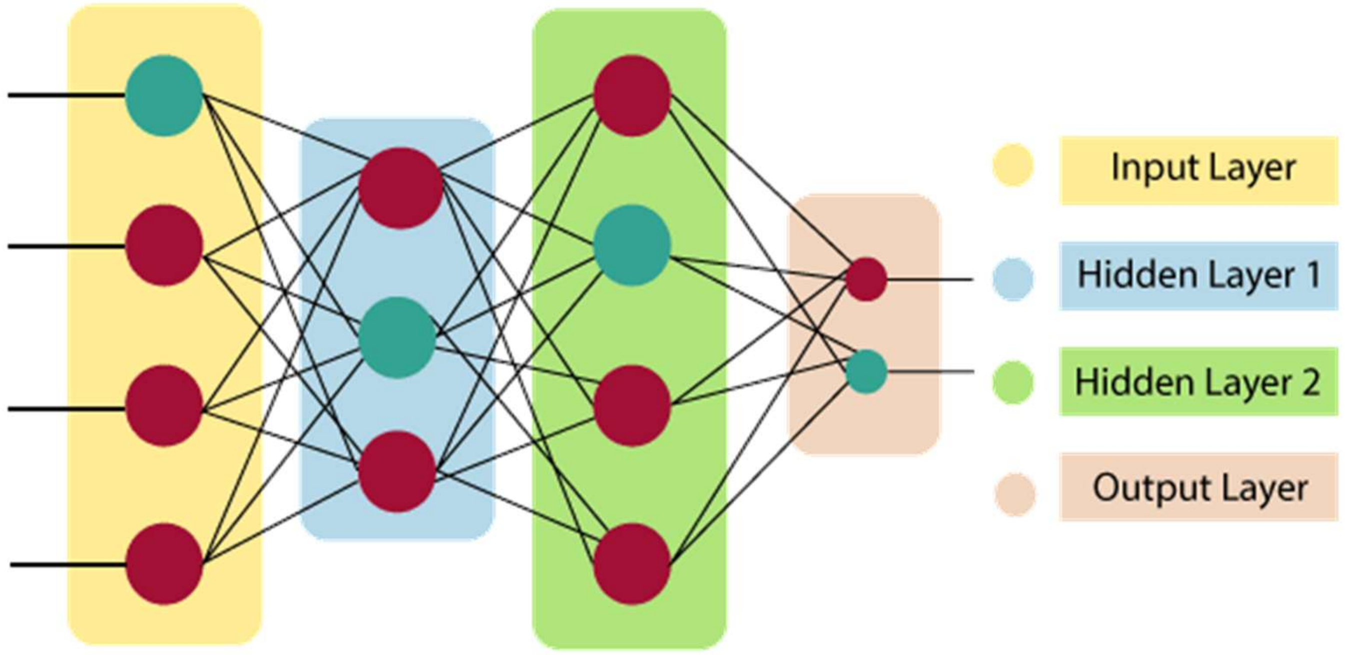


Fig 2 :Neural network' different architectures

Layers of Artificial Neural Network

Input Layer:

- As the name suggests, it accepts inputs in several different formats provided by the programmer.

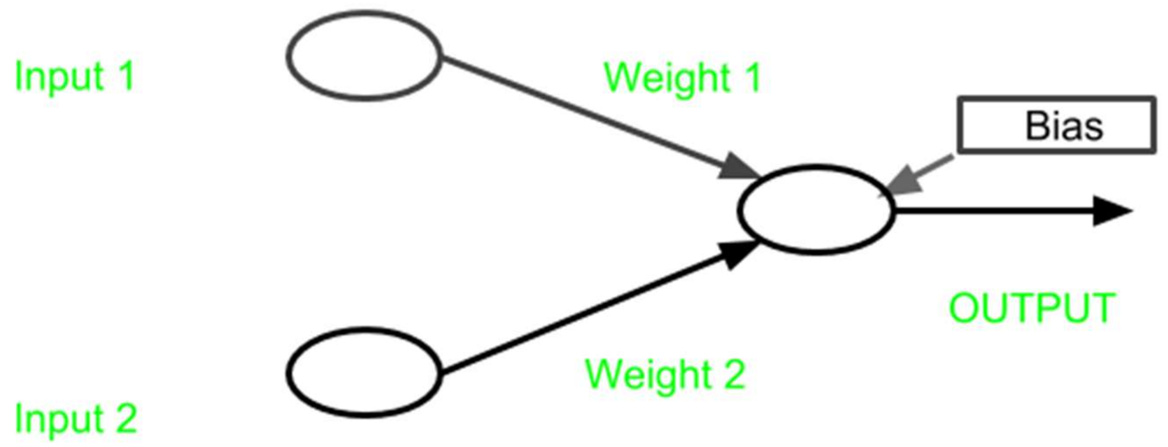
Hidden Layer:

- The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer:

- The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

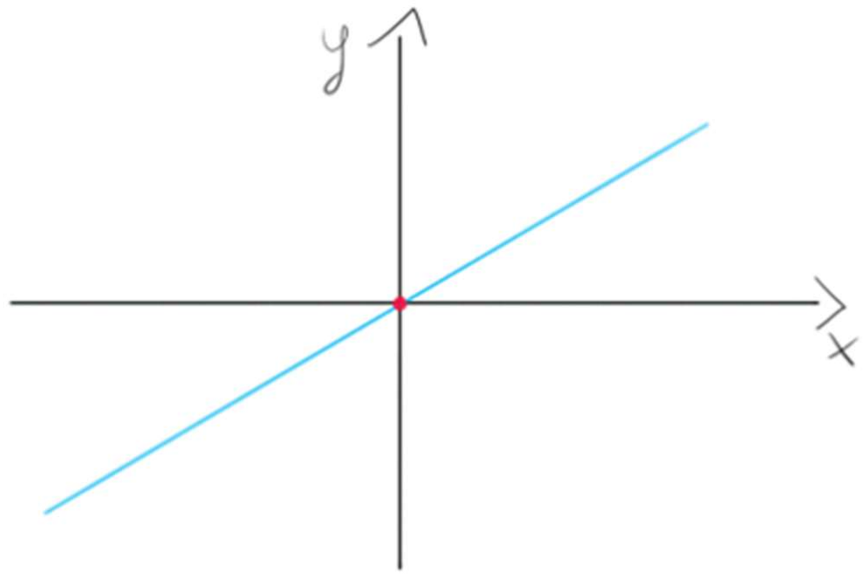
Bias in neural network

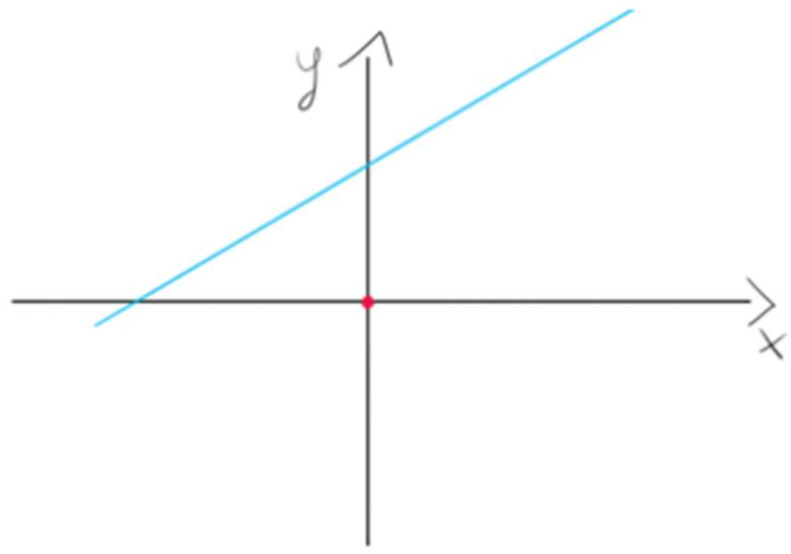


Output = weight1 * input1 + weight2 * input2 +bias

https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98

Need of Bais





Such a model is not very flexible. It means that the line needs to go through the point *(0, 0)*. A Slope of the line may change, however, it is tied to the coordinate system's origin.

That's the reason why we need bias neurons in neural networks. Without these spare bias weights, our model has quite limited "movement" while searching through solution space.

https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98

Artificial Neural Network

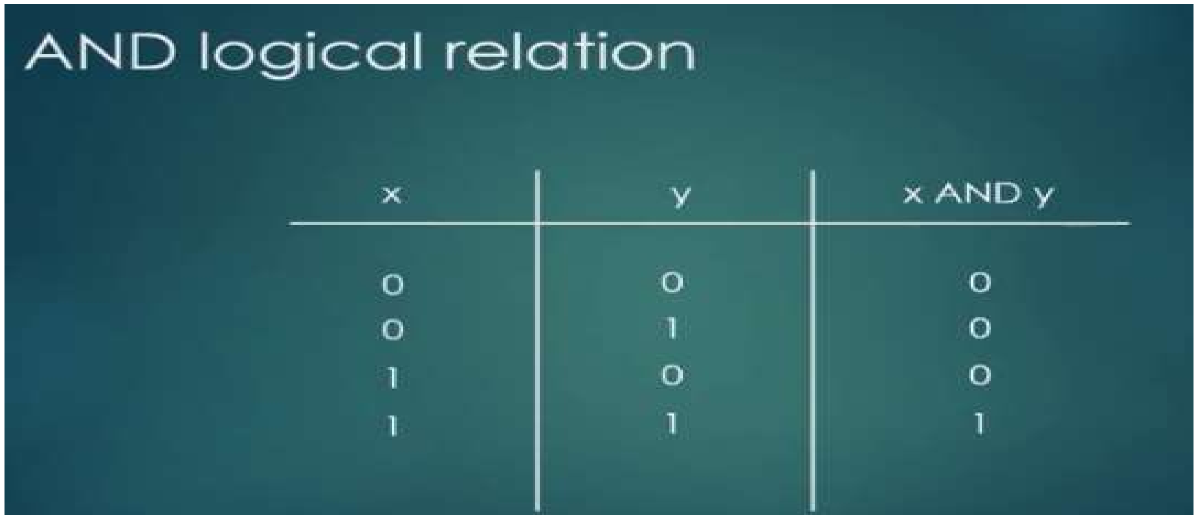
- The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.
- It determines weighted total is passed as an input to an activation function to produce the output. **Activation functions choose whether a node should fire or not.** Only those who are fired make it to the output layer.
- There are distinctive activation functions available that can be applied upon the sort of task we are performing.

$$\sum_{i=1}^{n} Wi * Xi + b$$

Artificial Neural Network

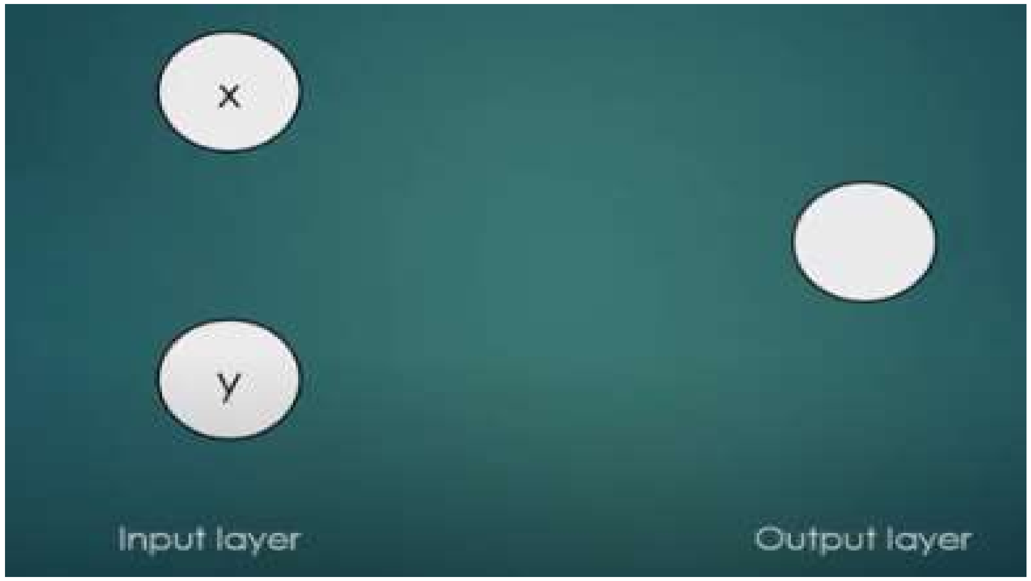
- Often used where data or functions are uncertain
  - Goal is to learn from a set of training data
  - And to generalize from learned instances to new unseen data
- Key attributes
  - Parallel computation
  - Distributed representation and storage of data
  - Learning (networks adapt themselves to solve a problem)
  - Fault tolerance (insensitive to component failures)

Artificial Neural Network

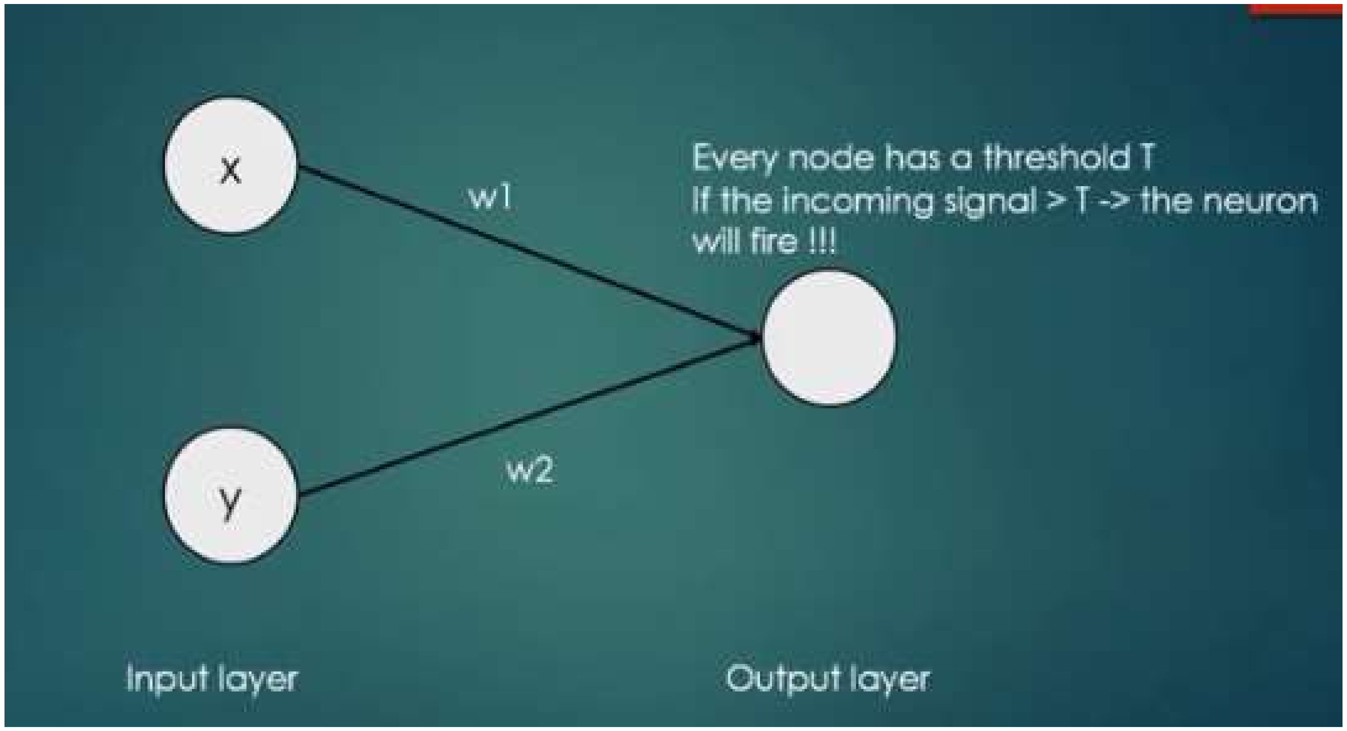


AND logical relation

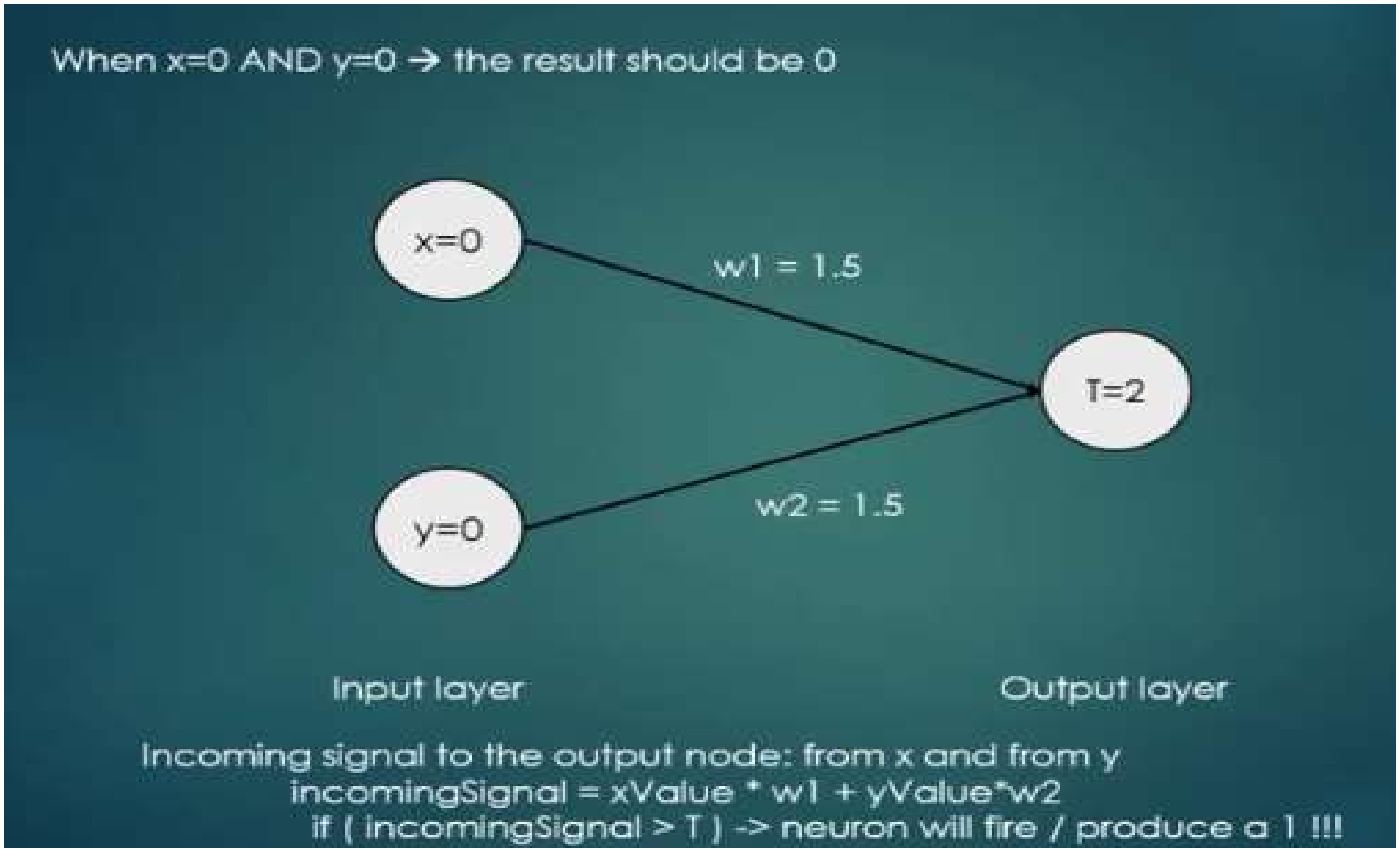
x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1



Artificial Neural Network

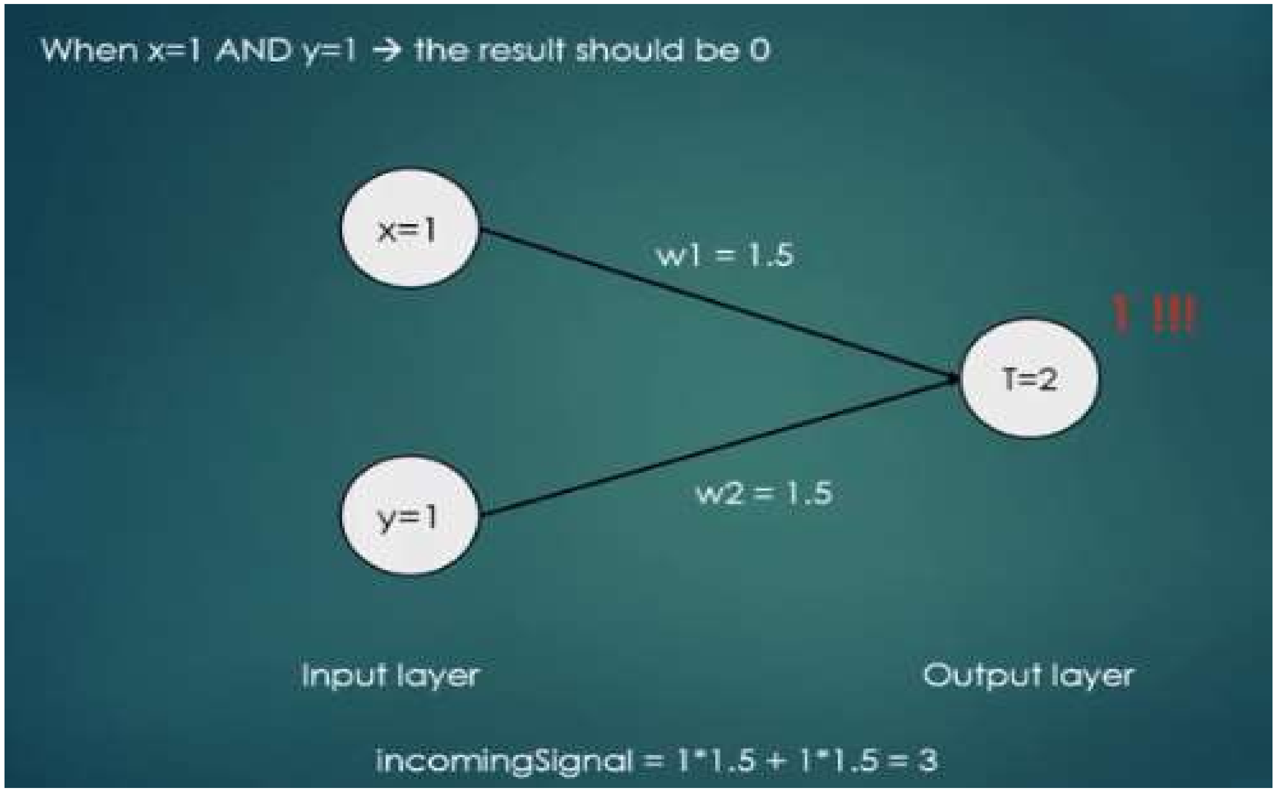

x
y
w1
w2
Every node has a threshold T
If the incoming signal > T -> the neuron will fire !!!
Input layer
Output layer

Artificial Neural Network

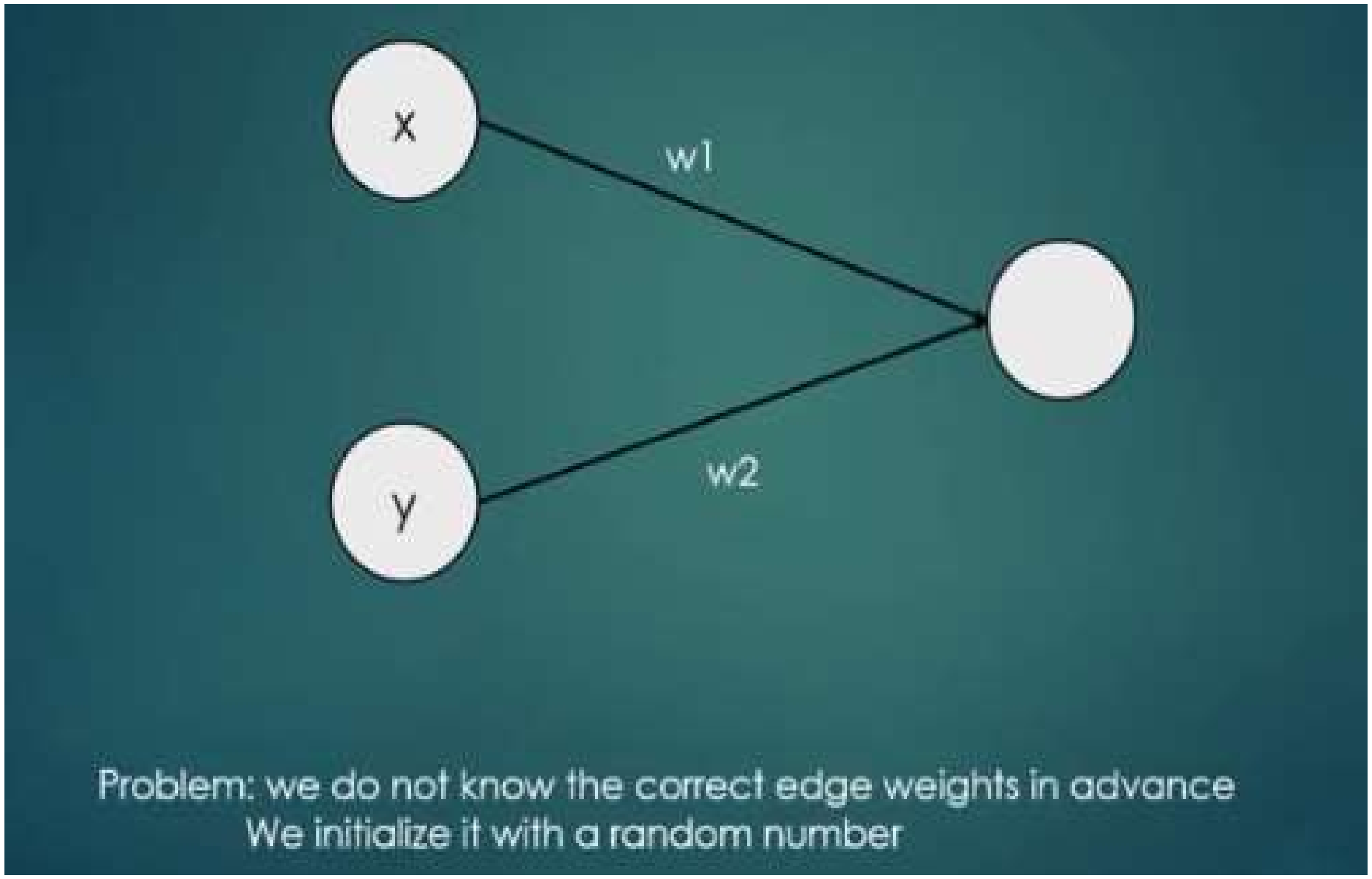

When x=0 AND y=0 → the result should be 0

Incoming signal to the output node: from x and from y
incomingSignal = xValue * w1 + yValue*w2
if (incomingSignal > T) -> neuron will fire / produce a 1 !!!

Artificial Neural Network

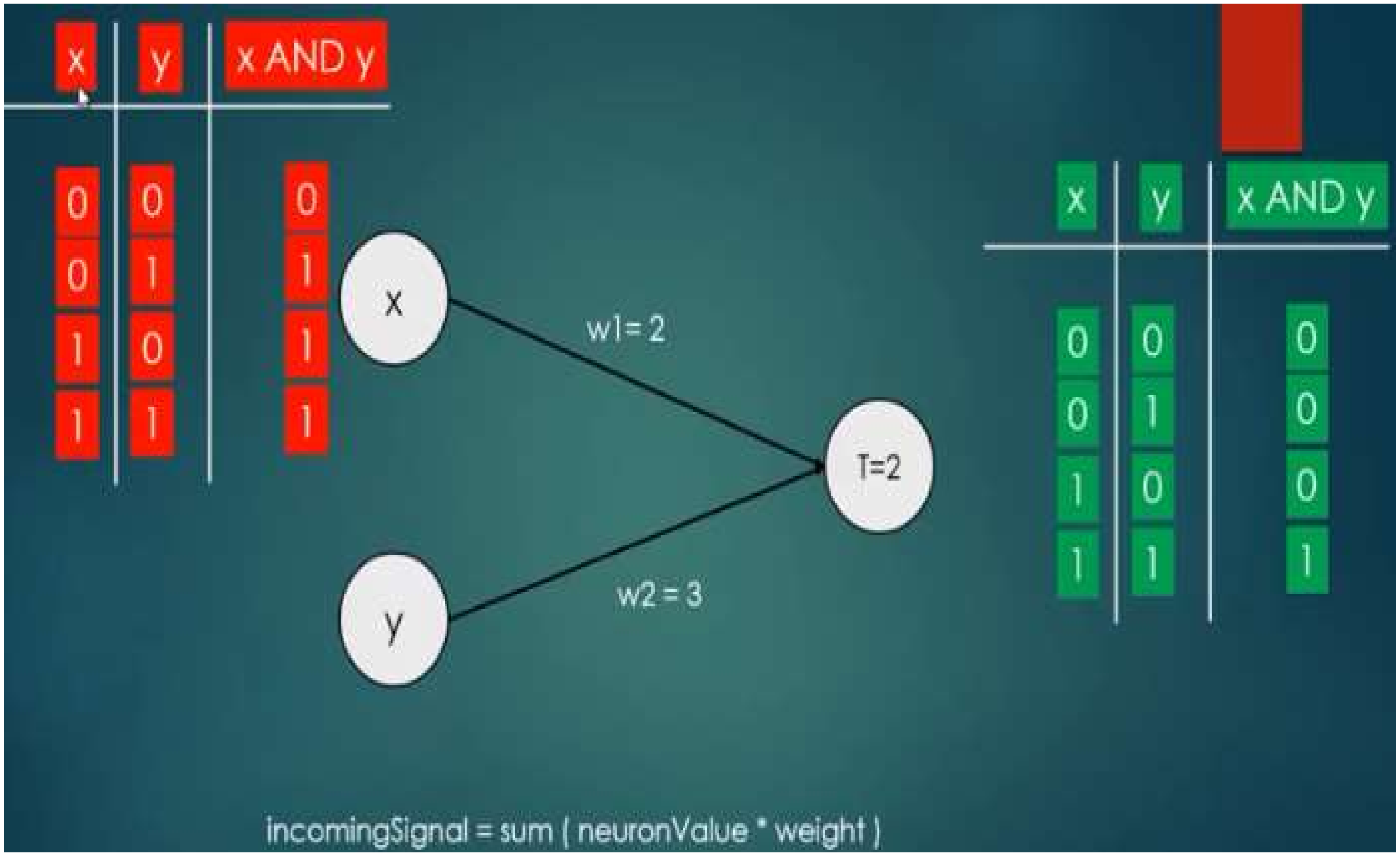

When x=1 AND y=1 → the result should be 0
x=1
y=1
w1 = 1.5
w2 = 1.5
T=2
1 !!!
Input layer
Output layer
incomingSignal = 1*1.5 + 1*1.5 = 3

Artificial Neural Network



Problem: we do not know the correct edge weights in advance
We initialize it with a random number

Artificial Neural Network



x	y	x AND y
0	0	0
0	1	1
1	0	1
1	1	1

x	y	x AND y
0	0	0
0	1	0
1	0	0
1	1	1

incomingSignal = sum (neuronValue * weight)

Types/Models of ANN Architecture

Feedforward Network

Feedback Network

Types/Models of ANN Architecture

Feedforward Network

- The generic feedforward network is characterized by the lack of feedback.
- This type of network can be connected in cascade to create a multilayer network.
- The output of a layer is the input to the following layer.
- Even though the feedforward network has no explicit feedback connection when x(t) is mapped into o(t), the output values are often compared with the "teacher's" information, which provides the desired output value, and also an error signal can be employed for adapting the network's weights.

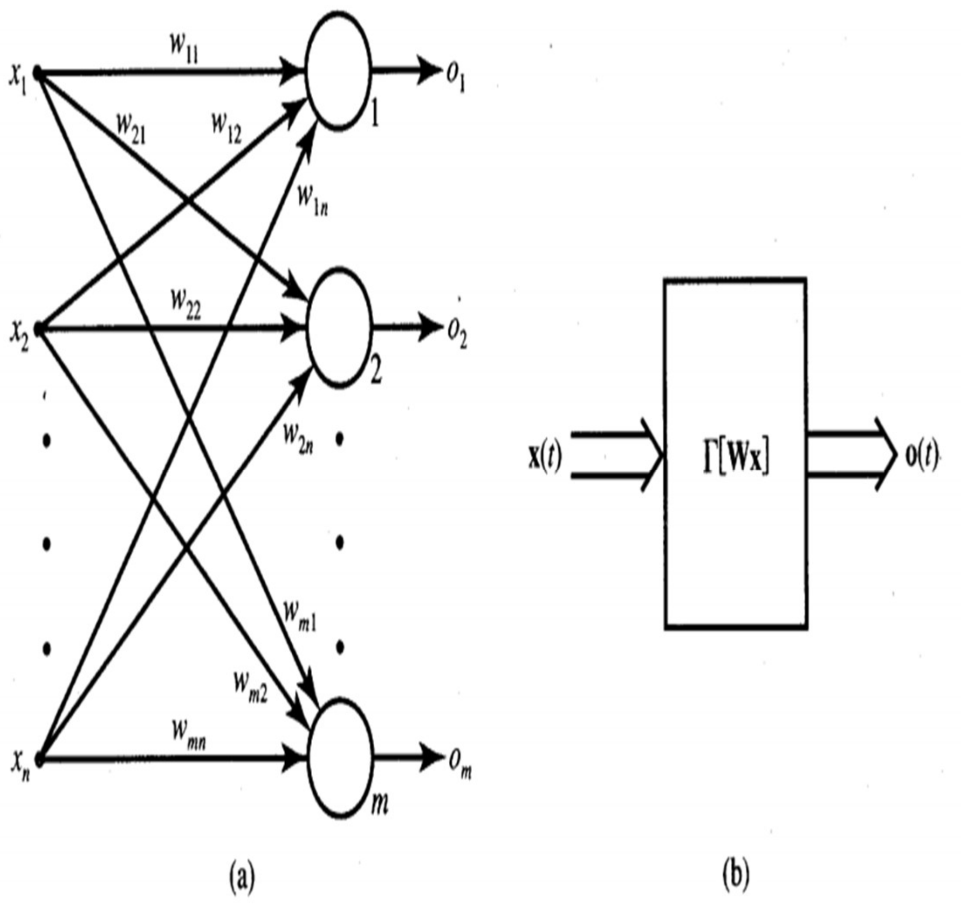


Figure 2.8 Single-layer feedforward network: (a) interconnection scheme and (b) block diagram.

Types/Models of ANN Architecture

Feedback Network

A feedback network can be obtained from the feedforward network by connecting the neurons' outputs to their inputs.

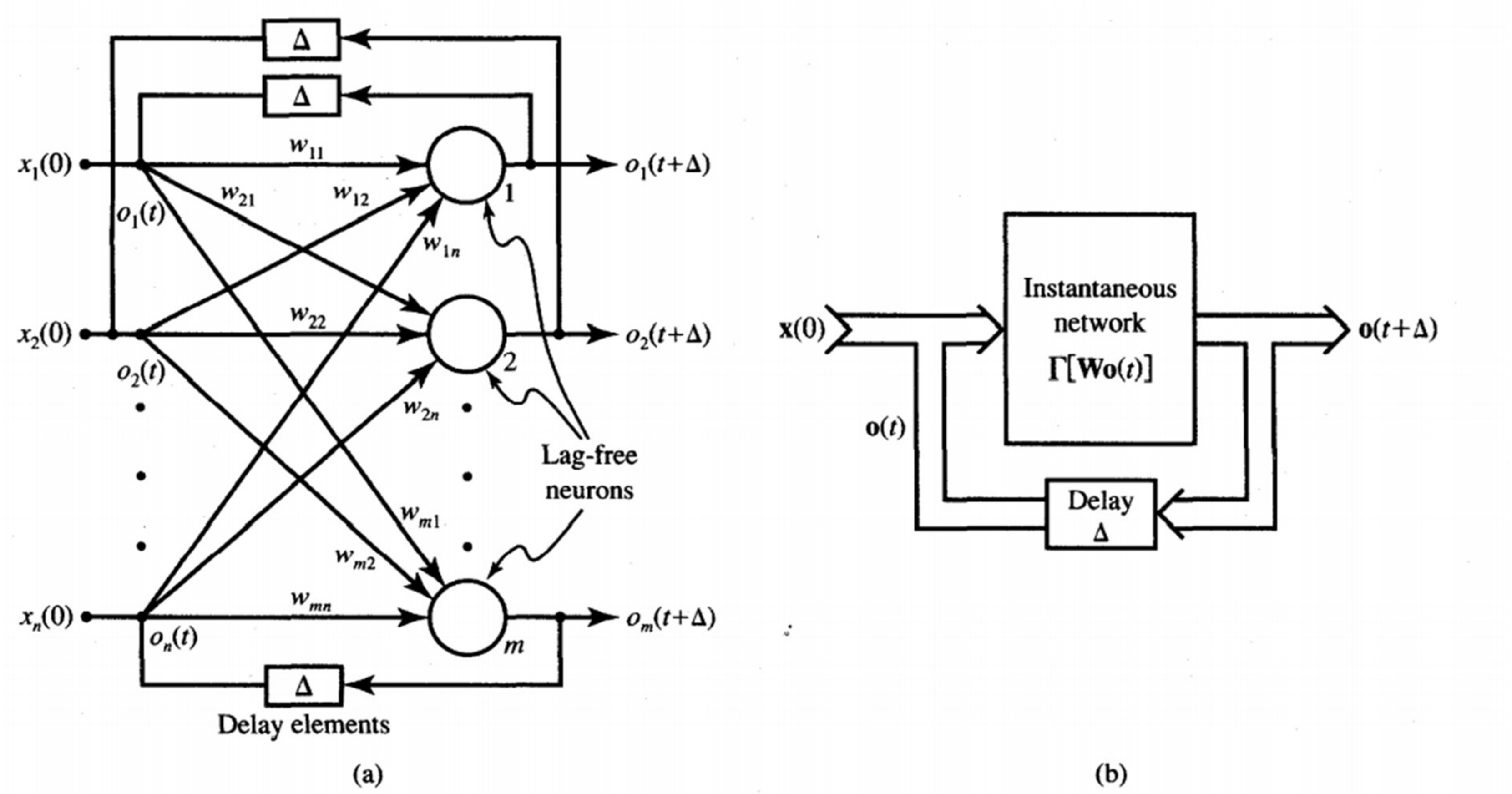


Figure 2.10 Single-layer discrete-time feedback network: (a) interconnection scheme and (b) block diagram.

Advantages of Artificial Neural Network (ANN)

- **Parallel processing capability:**

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

- **Storing data on the entire network:**

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

- **Capability to work with incomplete knowledge:**

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

- **Having a memory distribution:**

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

- **Having fault tolerance:**

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages of Artificial Neural Network (ANN)

- **Assurance of proper network structure:**

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

- **Unrecognized behavior of the network:**

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

- **Hardware dependence:**

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

- **Difficulty of showing the issue to the network:**

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

- **The duration of the network is unknown:**

The network is reduced to a specific value of the error, and this value does not give us optimum results.

1.1 Neural Computation (With Example)

- To inspect the performance of a simple classifier-
- Set of eight points, Po, P1, . . . , P7, in three-dimensional space is available.
- The set consists of all vertices of a three-dimensional cube as follows:

$$\{P_0(-1,-1,-1), P_1(-1,-1,1), P_2(-1,1,-1), P_3(-1,1,1),$$
$$P_4(1,-1,-1), P_5(1,-1,1), P_6(1,1,-1), P_7(1,1,1)\}$$

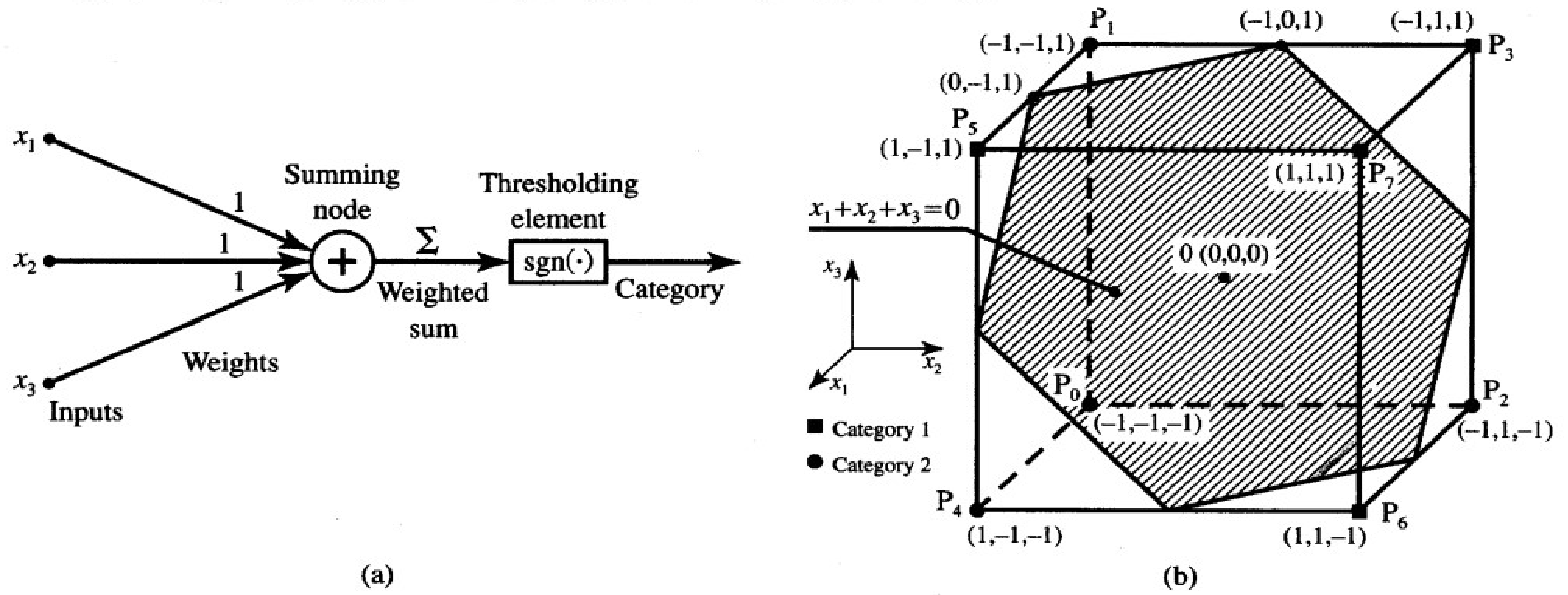


Figure 1.1 Partitioning of the set of cube vertices: (a) single-unit diagram and (b) implemented partitioning.

Introduction to Artificial Neural Systems by Jacek M. Zurada Yegnanarayana B, Artificial Neural Systems , PHP learning. (Page no. 3-8)

1.1 Neural Computation (With Example)

- Elements of this set need to be classified into two categories.
- The first category is defined as containing points with two or more positive ones
- The second category contains all the remaining points that do not belong to the first category.
- Accordingly, points P3, P5, P6, and P7 belong to the first category
- Remaining points to the second category. Classification of points P3, P5, P6, and P7 can be based on the summation of coordinate values for each point evaluated for category membership.
- For each point Pi (x,, x2, x3), where i = 0, . . . , 7, the membership in the category can be established by the following calculation:

$$\text{If } \operatorname{sgn}(x_1 + x_2 + x_3) = \begin{cases} 1, & \text{then category 1} \\ -1, & \text{then category 2} \end{cases}$$

Describes the decision function of the classifier designed by inspection of the set that needs to be partitioned

Introduction to Artificial Neural Systems by Jacek M. Zurada Yegnanarayana B, Artificial Neural Systems , PHP learning. (Page no. 3-8)

1.1 Neural Computation (With Example)

The unit from below Figure maps the entire three-dimensional space into just two points, 1 and - 1.
A question arises as to whether a unit with a "squashed" sgn function rather than a regular sgn function could prove more advantageous.
Assuming that the "squashed" sgn function has the shape as in Figure, the outputs take values in the range (- 1,l) and are generally more discernible than in the previous case.
Using units with continuous characteristics offers tremendous opportunities for new tasks that can be performed by neural networks. Specifically, the fine granularity of output provides more information than the binary f 1 output of the thresholding element.

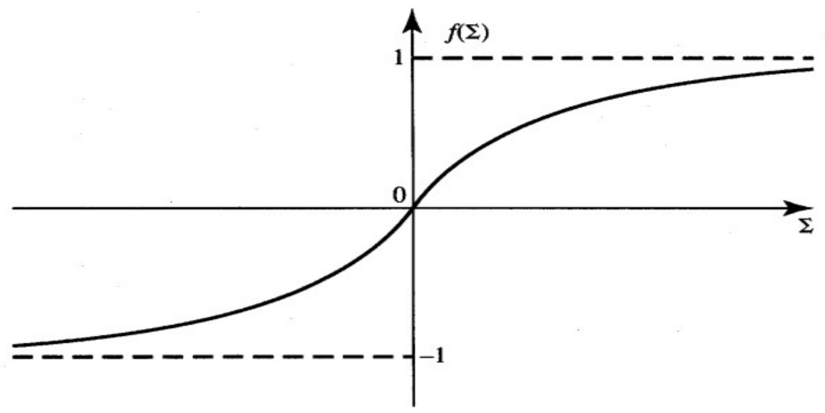


Figure 1.2 "Squashed" sgn function.

Perceptron, Sigmoid Neurons

Sources:

1. Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015 (Module I- Perceptron, Sigmoid Neurons)

2. https://www.simplilearn.com/what-is-perceptron-tutorial

Final Year
BTECH

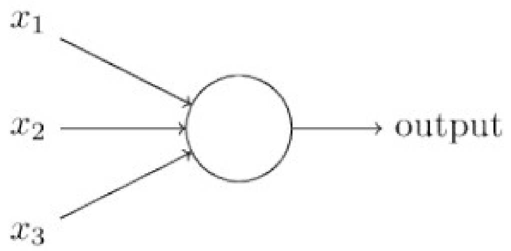
Subject : **Deep Learning (PE3)**

Perceptron

- A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.
- A type of artificial neuron called a perceptron.
- Perceptron was introduced by Frank Rosenblatt in 1957. He proposed a Perceptron learning rule based on the original MCP neuron.
- A Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time.

How do perceptron's work?

- A perceptron takes several binary inputs, x1 , x2 , . . ., and produces a single binary output

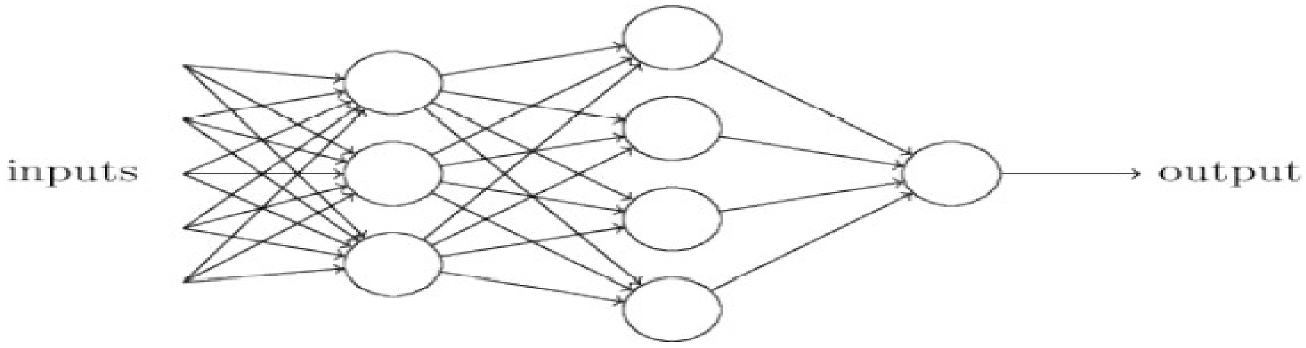


- In the example shown the perceptron has three inputs, x1 , x2 , x3 . In general it could have more or fewer inputs
- Rosenblatt proposed a simple rule to compute the output. He introduced weights, w1 ,w2 , . . ., real numbers expressing the importance of the respective inputs to the output.
- The neuron's output, 0 or 1, is determined by whether the weighted sum is less than or greater than some threshold value
- Just like the weights, the threshold is a $\sum_j w_j x_j$ real number which is a parameter of the neuron. To put it in more precise algebraic terms

$$\text{output} = \begin{cases} 0 & \text{if} \quad \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if} \quad \sum_j w_j x_j > \text{threshold} \end{cases}$$

Perceptron

- The first column of perceptron's – the first layer of perceptrons – is making three very simple decisions, by weighing the input evidence.
- Perceptrons in the second layer- Each of those perceptron's is making a decision by weighing up the results from the first layer of decision-making.
- Perceptron in the second layer can make a decision at a more complex and more abstract level than perceptron's in the first layer.
- And even more complex decisions can be made by the perceptron in the third layer.
- A many-layer network of perceptron's can engage in sophisticated decision making.



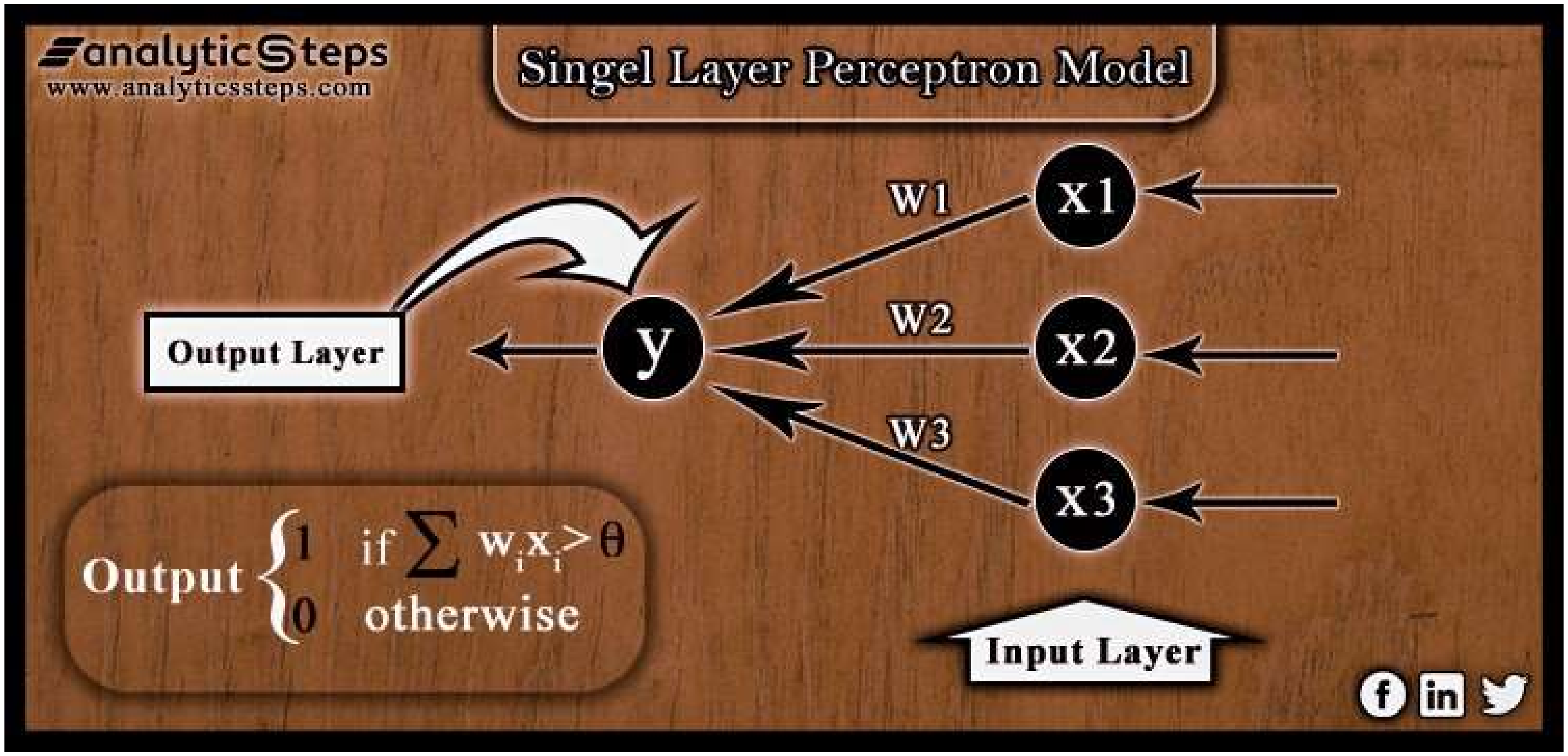
Perceptron

- There are two types of Perceptrons: Single layer and Multilayer.
- **Single layer Perceptrons** can learn **only linearly separable patterns**.
- **Multilayer Perceptrons** or feedforward neural networks with two or more layers have **the greater processing power**.
- The Perceptron algorithm learns the weights for the input signals in order to draw a **linear decision boundary**.
- This enables you to distinguish between the two linearly separable classes +1 and -1.
- Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data.

https://www.simplilearn.com/what-is-perceptron-tutorial

Perceptron-Single layer

- It includes a feed-forward network depends on a threshold transfer function in its model.
- It is the easiest type of ANN that able to analyze only linearly separable objects with binary outcomes(target) i.e. 1, and 0.

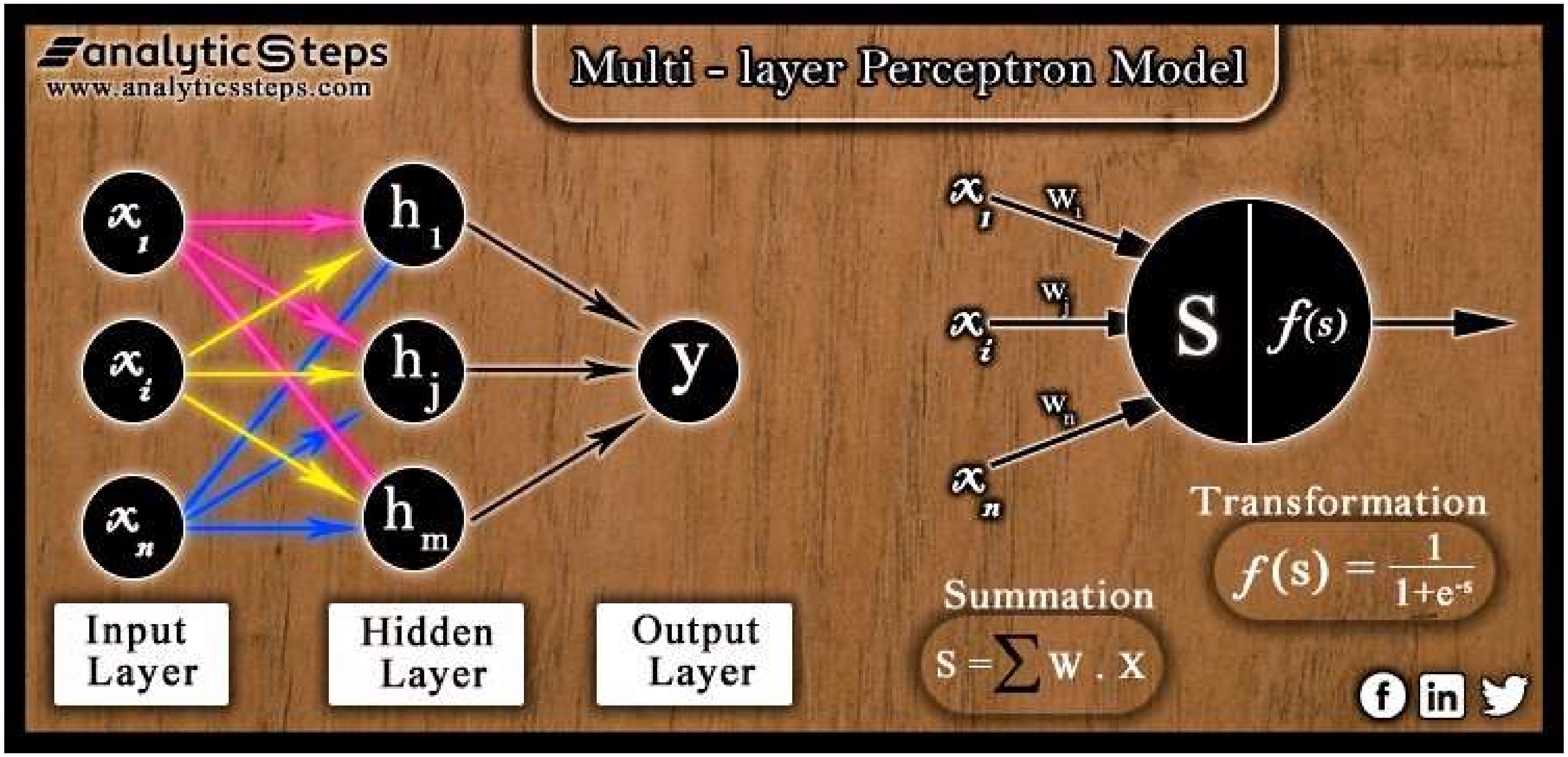

analyticSteps
www.analyticssteps.com
Singel Layer Perceptron Model
W1
W2
W3
x1
x2
x3
y
Output Layer
Input Layer
Output $\begin{cases} 1 & \text{if } \sum w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$

Perceptron-Single layer

- In single-layered perceptron model, its algorithm doesn't have previous information,
- Initially, **weights are allocated inconstantly**, then the algorithm adds up all the weighted inputs, if the added value is more than some pre-determined value(or, threshold value) then single-layered perceptron is stated as activated and delivered output as +1.
- **Multiple input values feed up** to the perceptron model, model executes with input values, and if the estimated value is the same as the required output, then the model performance is found out to be satisfied, therefore weights demand no changes. In fact, if the model doesn't meet the required result then few changes are made up in weights to minimize errors.

Perceptron-Multi layer

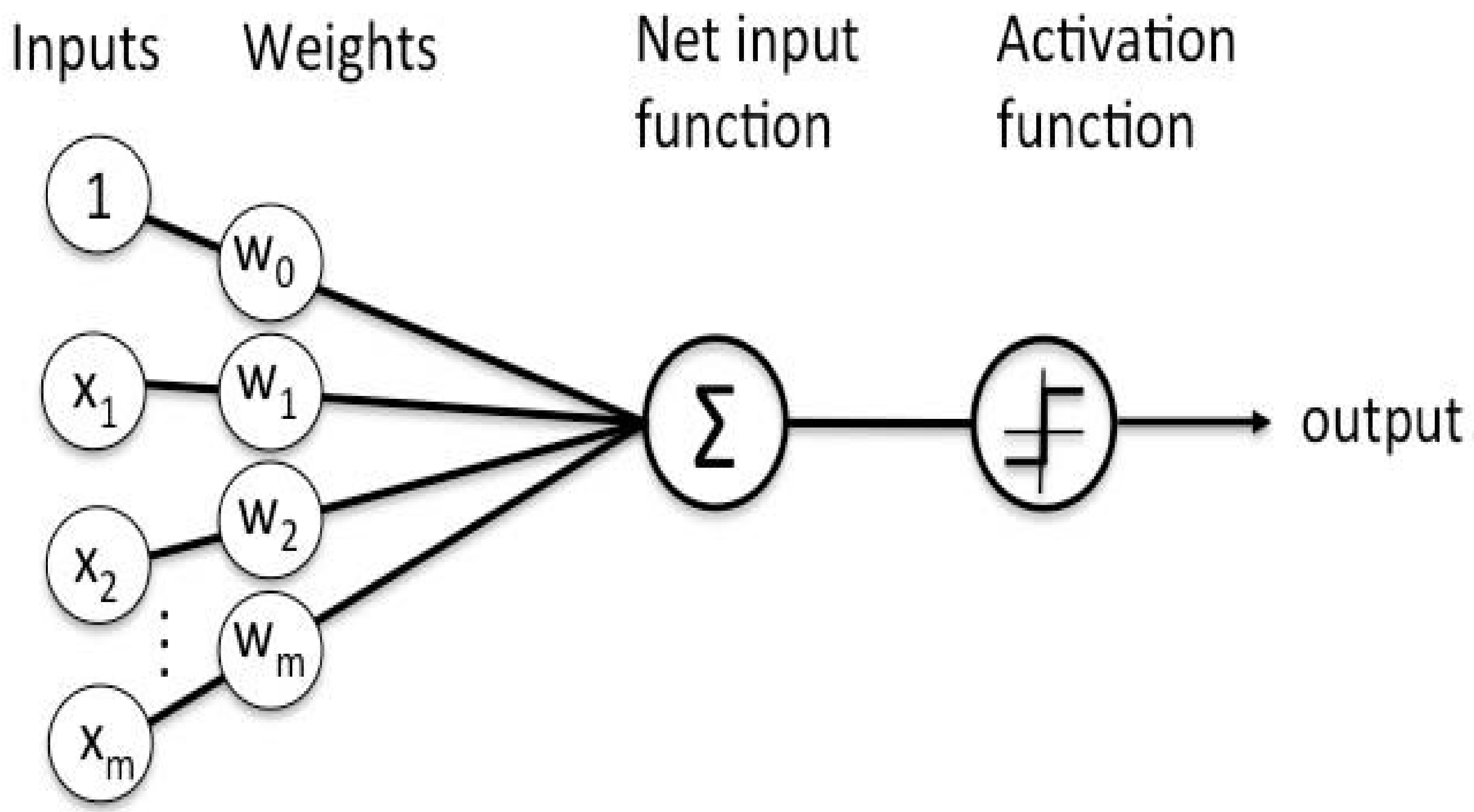
- It has a structure similar to a single-layered perceptron model with more number of hidden layers.
- It is also termed as a **Backpropagation algorithm**. It executes in two stages; the **forward stage** and the **backward stages**.



Perceptron-Multilayer

- In the forward stage, **activation functions are originated from** the input layer to the output layer,
- In the backward stage, the **error between the actual observed value and demanded given value is originated backward** in the output layer for modifying weights and bias values.
- In simple terms, multi-layered perceptron can be treated as a network of numerous artificial neurons overhead varied layers, the activation function is no longer linear, instead, non-linear activation functions such as Sigmoid functions, TanH, ReLU activation Functions, etc are deployed for execution.

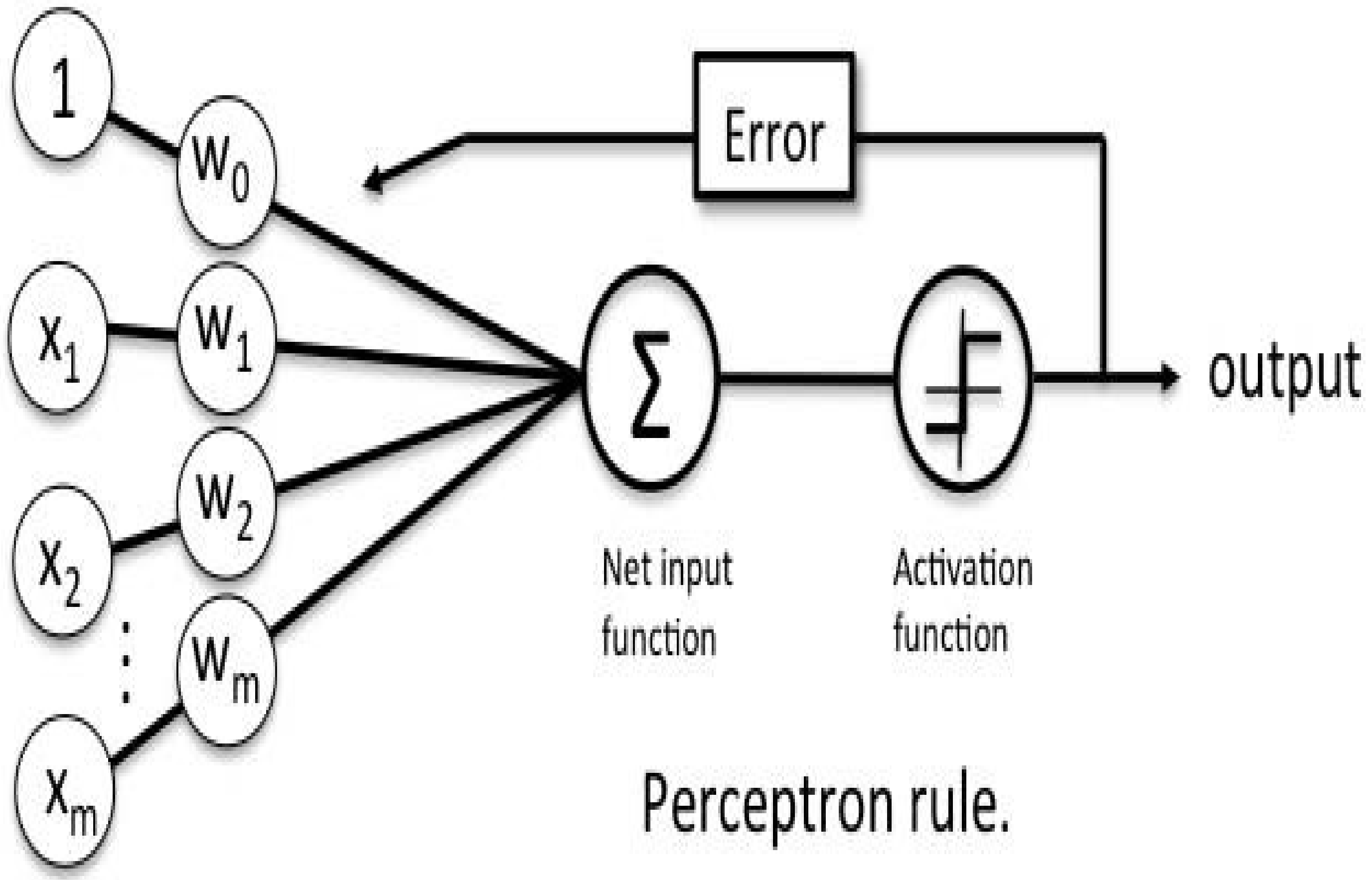
Perceptron


Inputs
Weights
Net input function
Activation function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

Perceptron Learning Rule

- Perceptron Learning Rule states that the **algorithm would automatically learn the optimal weight coefficients**.
- The input features are then **multiplied with these weights to determine if a neuron fires or not**.
- The Perceptron receives multiple input signals, and if the **sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an outpu**t.
- In the context of **supervised learning and classification**, this can then be used to **predict the class of a sample**.

Perceptron Learning Rule



Perceptron rule.

Perceptron Function

- Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)”is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

- In the equation given above:

“w” = vector of **real-valued weights**

“b” = **bias (an element that adjusts the boundary away from origin without any dependence on the input value)**

“x” = vector of input x values

$$\sum_{i=1}^{m} w_i x_i$$

“m” = number of inputs to the Perceptron

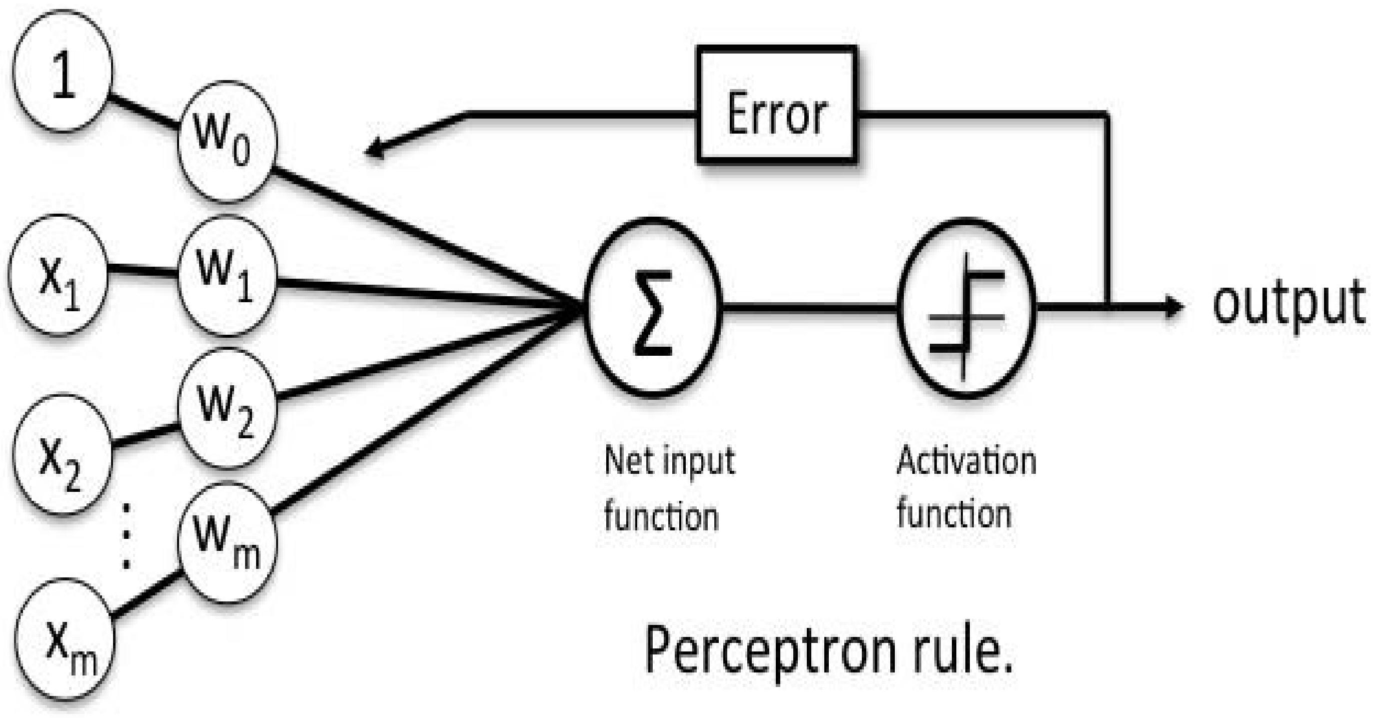
The output can be represented as “1” or “0.” It can also be represented as “1” or “-1” depending on which activation function is used.

Inputs of a Perceptron

- A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result
- A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False.
- The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

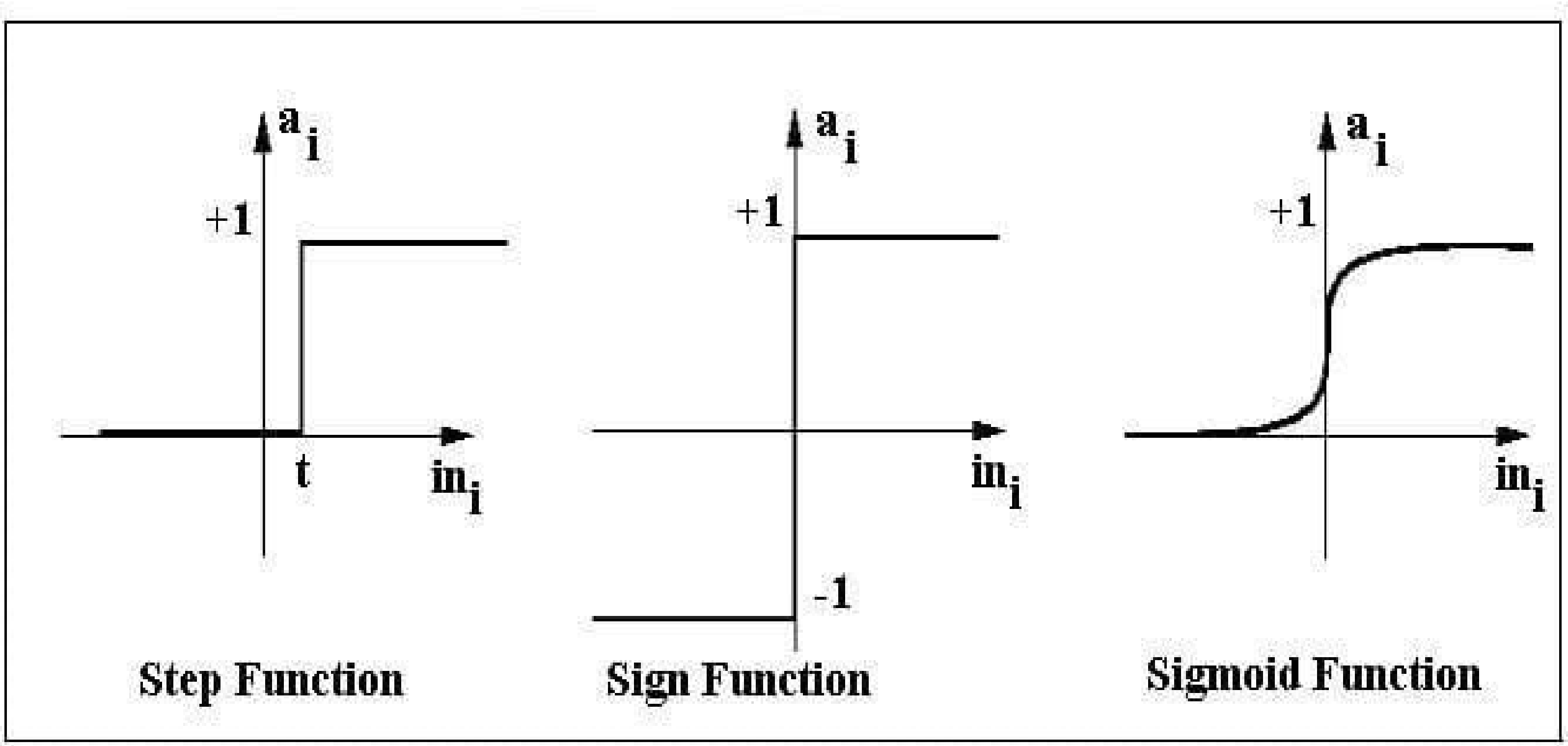
Inputs of a Perceptron



Perceptron rule.

Activation Functions of Perceptron

- The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not.



Activation Functions of Perceptron

- E.g.

If ∑ wixi> 0 => then final output “o” = 1 (issue bank loan)

Else, final output “o” = -1 (deny bank loan)

- Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

- Perceptron with a Boolean output:
- Inputs: x1…xn
- Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

- Weights: wi=> contribution of input xi to the Perceptron output;
- w0=> bias or threshold
- If ∑w.x > 0, output is +1, else -1. The neuron gets triggered only when weighted input reaches a certain threshold value.

Output of Perceptron

- An output of +1 specifies that the neuron is triggered. An output of -1 specifies that the neuron did not get triggered.
- "sgn" stands for sign function with output +1 or -1.

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

Error in Perceptron

- In the Perceptron Learning Rule, the predicted output is compared with the known output. If it does not match, the error is propagated backward to allow weight adjustment to happen.

Perceptron: Decision Function

- A decision function φ(z) of Perceptron is defined to take a linear combination of x and w vectors.

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

- The value z in the decision function is given by:

$$z = w_1 x_1 + \ldots + w_m x_m$$

- The decision function is +1 if z is greater than a threshold θ, and it is -1 otherwise.

$$\phi(z) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & otherwise \end{cases}$$

Perceptron: Decision Function

- Bias Unit
- For simplicity, the threshold θ can be brought to the left and represented as w0x0, where w0= -θ and x0= 1.

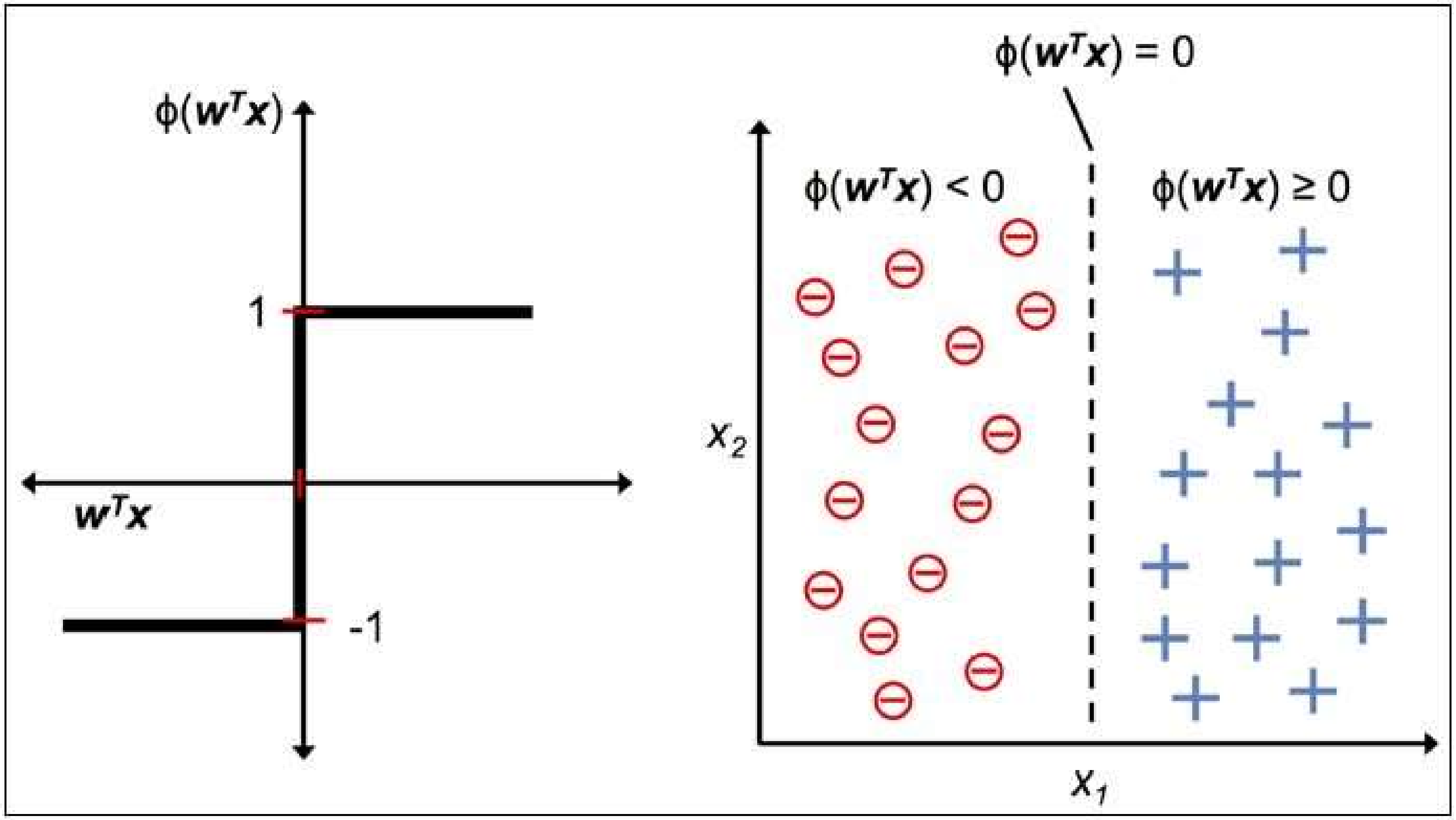
$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

- The value w0 is called the bias unit.
- The decision function then becomes:

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$

Perceptron: Decision Function

- Output
- The figure shows how the decision function squashes wTx to either +1 or -1 and how it can be used to discriminate between two linearly separable classes.



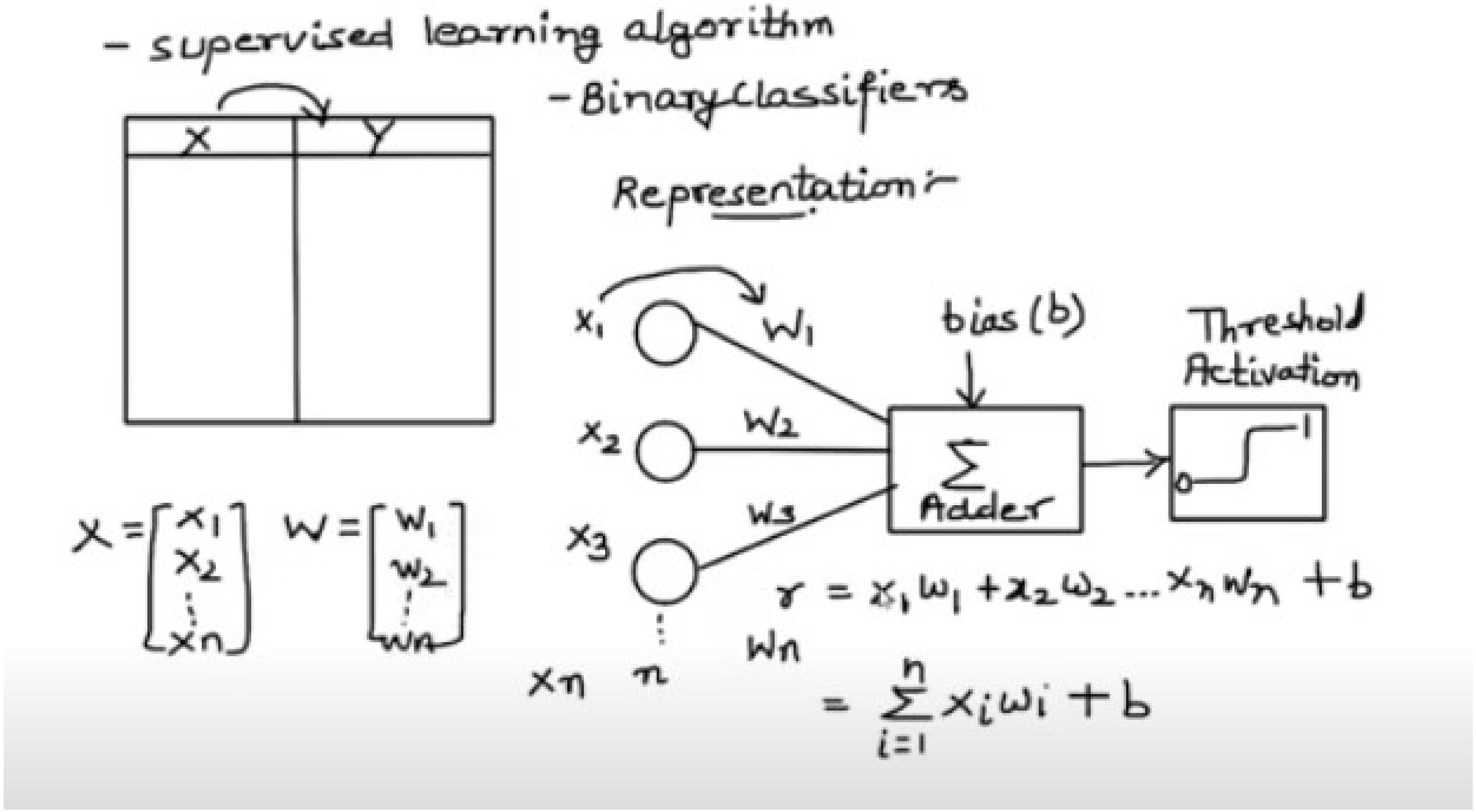
Perceptron

- Perceptron has the following characteristics:
- Perceptron is an algorithm for Supervised Learning of single layer binary linear classifier.
- Optimal weight coefficients are automatically learned.
- Weights are multiplied with the input features and decision is made if the neuron is fired or not.
- Activation function applies a step rule to check if the output of the weighting function is greater than zero.
- Linear decision boundary is drawn enabling the distinction between the two linearly separable classes +1 and -1.
- If the sum of the input signals exceeds a certain threshold, it outputs a signal; otherwise, there is no output.
- Types of activation functions include the sign, step, and sigmoid functions.

Summary Perceptron

- The activation function to be used is a subjective decision based on the problem statement and the form of the desired results.
- If the learning process is slow or has vanishing or exploding gradients, change the activation function to see if these problems can be resolved.
- An artificial neuron is a mathematical function conceived as a model of biological neurons, that is, a neural network.
- A Perceptron is a neural network unit that does certain computations to detect features or business intelligence in the input data. It is a function that maps its input “x,” which is multiplied by the learned weight coefficient, and generates an output value ”f(x).
- ”Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.
- Single layer Perceptrons can learn only linearly separable patterns.
- Multilayer Perceptron or feedforward neural network with two or more layers have the greater processing power and can process non-linear patterns as well.
- Perceptrons can implement Logic Gates like AND, OR, or XOR.

Perceptron



- supervised learning algorithm

- Binary Classifiers

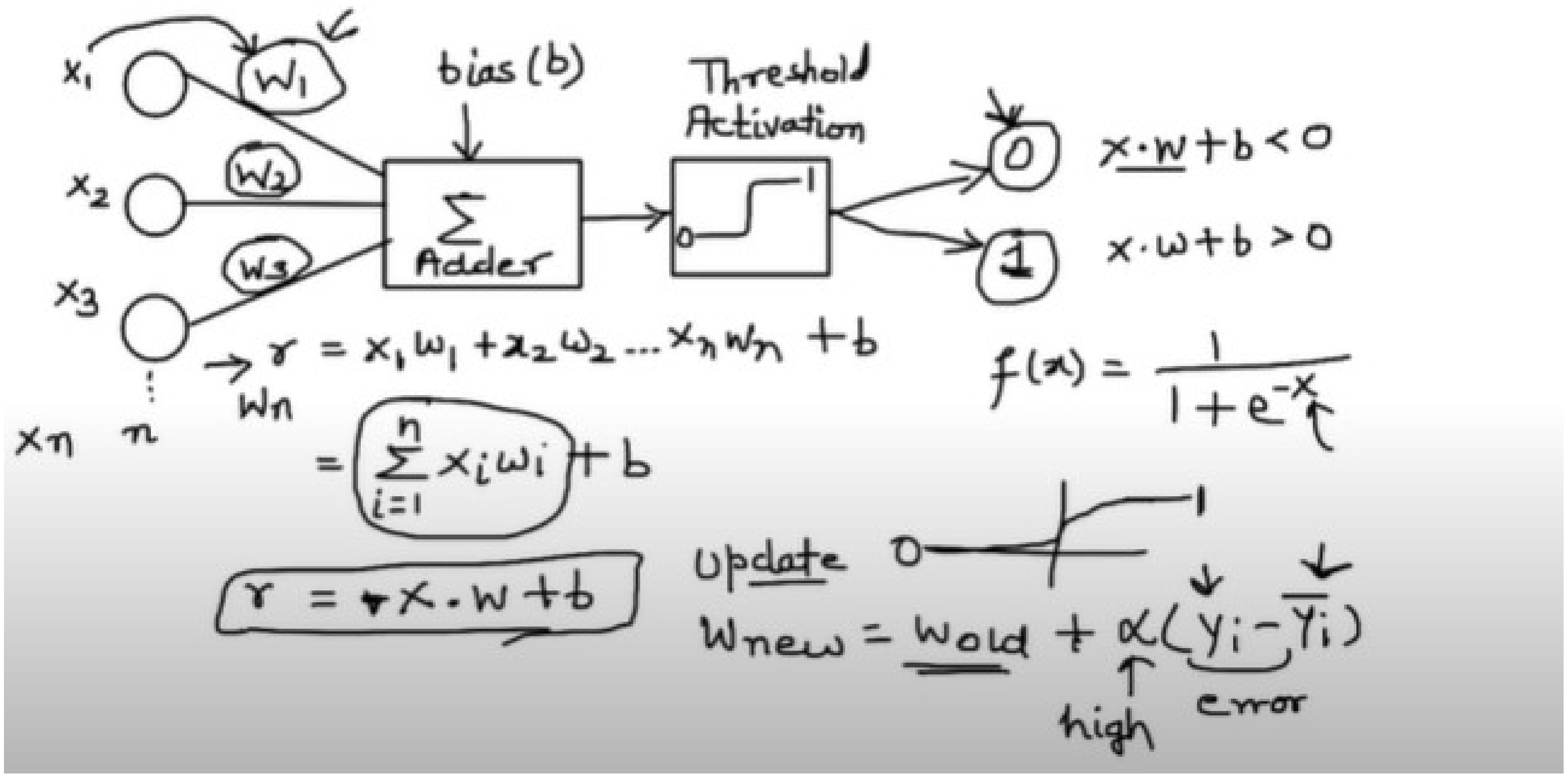
Representation:-

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

$$r = x_1 w_1 + x_2 w_2 \dots x_n w_n + b$$

$$= \sum_{i=1}^{n} x_i w_i + b$$

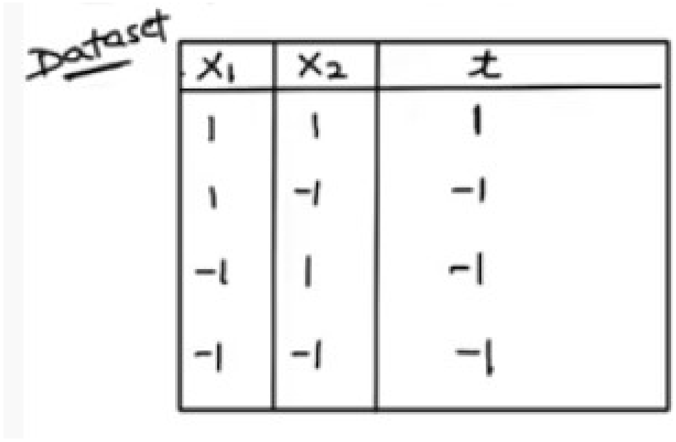
Perceptron



x_1
x_2
x_3
x_n
n
w_1
w_2
w_3
w_n
bias (b)
Σ Adder
Threshold Activation
1
0
0
1
$x \cdot w + b < 0$
$x \cdot w + b > 0$
$r = x_1 w_1 + x_2 w_2 \dots x_n w_n + b$
$= \sum_{i=1}^{n} x_i w_i + b$
$r = x \cdot w + b$
$f(x) = \frac{1}{1+e^{-x}}$
0
1
update
$W_{new} = W_{old} + \alpha (y_i - \bar{y}_i)$
high
error

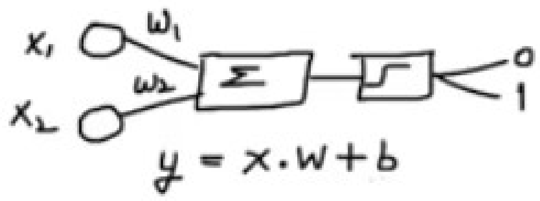
Perceptron Example :

Implementation AND function using Perceptron Network for bipolar input (1,-1) and target

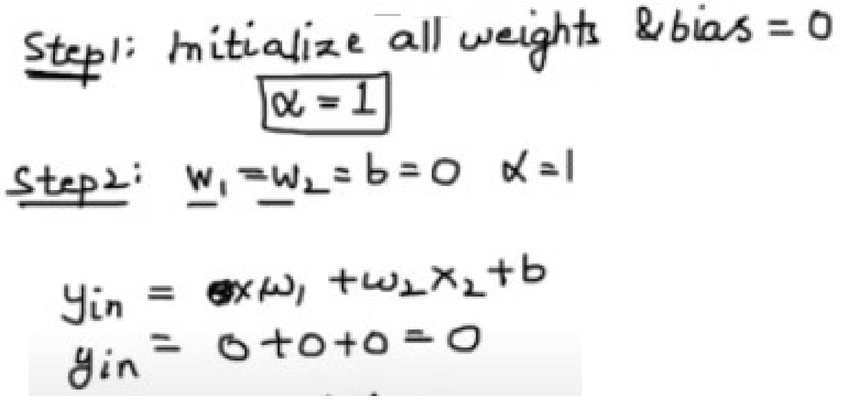


Dataset

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



Iteration 1



Step1: Initialize all weights & bias = 0

$\boxed{\alpha = 1}$

Step2: $w_1 = w_2 = b = 0 \quad \alpha = 1$

$y_{in} = x_1 w_1 + w_2 x_2 + b$

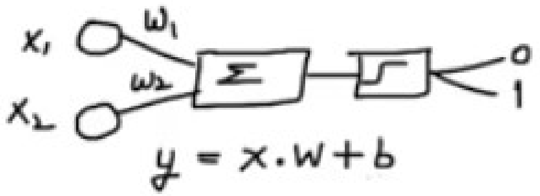
$y_{in} = 0 + 0 + 0 = 0$

Perceptron Example :

Implementation AND function using Perceptron Network for bipolar input (1,-1) and target

Dataset

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



$y = x.w + b$

Step 3) Activation / Sigmoid :-

$$f(y_{in}) = \begin{cases} 1 & y_{in} > 0 \\ 0 & y_{in} \leq 0 \end{cases}$$

$$y = f(y_{in}) = f(0) = 0$$

$$\rightarrow \boxed{y = 0} \leftarrow$$

Step 4) Update the weights :- $\overset{\text{error}}{\Delta w}$

$$\boxed{\begin{aligned} w_{new} &= w_{old} + \alpha t x_i \\ b_{new} &= b_{old} + \alpha t \end{aligned}}$$

$$\boxed{\begin{aligned} w_1 &= 0 + (1)(1)(1) = 1 \\ w_2 &= 0 + (1)(1)(1) = 1 \end{aligned}}$$

$$b_{new} = 0 + (1)(1) = 1$$

$$\boxed{b_{new} = 1}$$

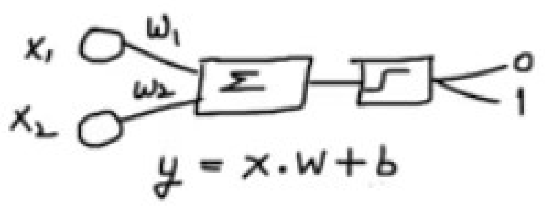
Perceptron Example :

Implementation AND function using Perceptron Network for bipolar input (1,-1) and target

Iteration 2

Dataset

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



$y = x \cdot w + b$

$x_1 = 1 \quad x_2 = -1 \quad t = -1$

$y_{in} = x_1 w_1 + x_2 w_2 + b$

$y_{in} = (1)(1) + (-1)(1) + 1$

$\boxed{y_{in} = 1}$

$f(y_{in}) = f(1) = 1$

$\boxed{y = 1}$

$w_1 = 1 + (1)(-1)(1)$

$\boxed{w_1 = 0}$

$w_2 = 1 + (1)(-1)(-1)$

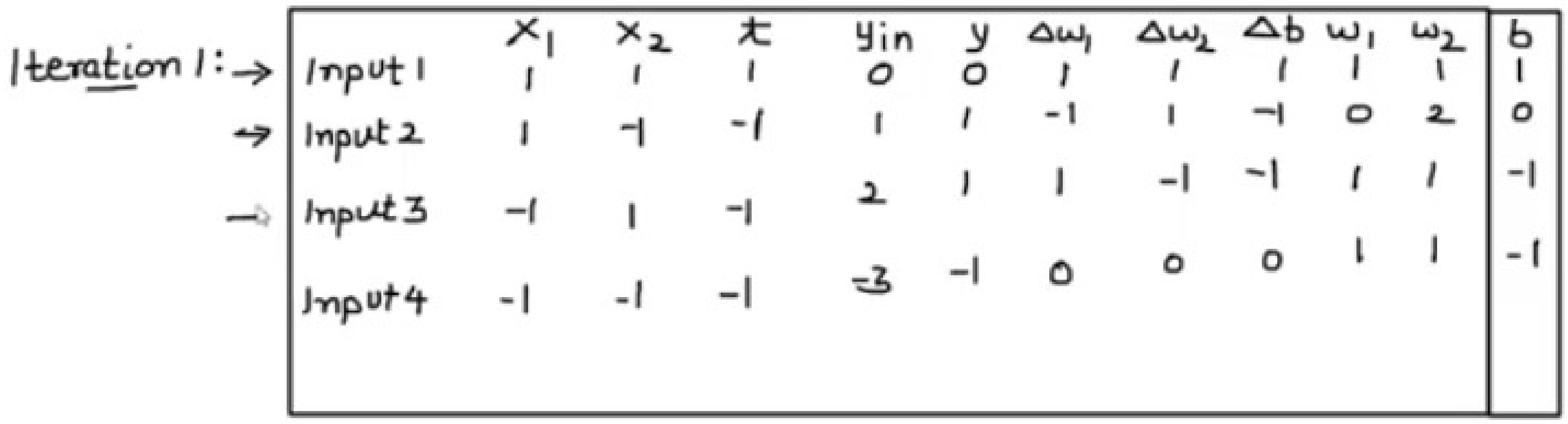
$\boxed{w_2 = 2}$

$\rightarrow b_{new} = 1 + (1)(-1)$

$\rightarrow b_{new} = 1 - 1 = 0$

Perceptron Example :

Implementation AND function using Perceptron Network for bipolar input (1,-1) and target



Iteration 1:

	x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
Input 1	1	1	1	0	0	1	1	1	1	1	1
Input 2	1	-1	-1	1	1	-1	1	-1	0	2	0
Input 3	-1	1	-1	2	1	1	-1	-1	1	1	-1
Input 4	-1	-1	-1	-3	-1	0	0	0	1	1	-1

Iteration 2 – Home work

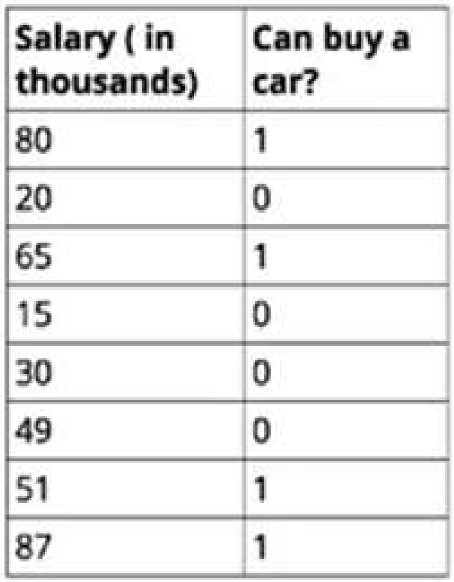
Sigmoid Neuron

- Sigmoid neurons are the building block of the deep neural networks.
- Sigmoid neurons are similar to perceptrons, but they are slightly modified such that the output from the sigmoid neuron is much smoother than the step functional output from perceptron.

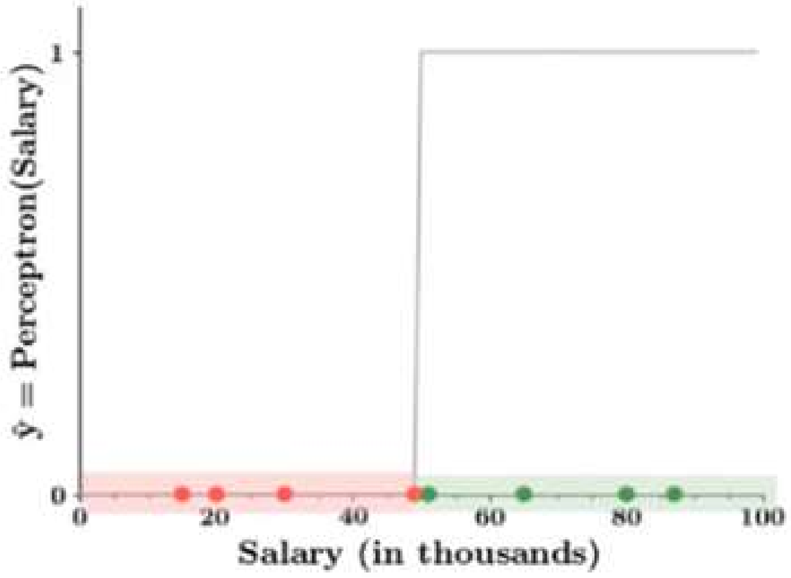
https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7

Why Sigmoid Neuron

- Consider the decision making process of a person, whether he/she would like to purchase a car or not based on only one input X1—Salary and by setting the threshold $b(W_o) = -10$ and the weight $W_1 = 0.2$.
- The output from the perceptron model will look like in the figure shown below.



Salary (in thousands)	Can buy a car?
80	1
20	0
65	1
15	0
30	0
49	0
51	1
87	1

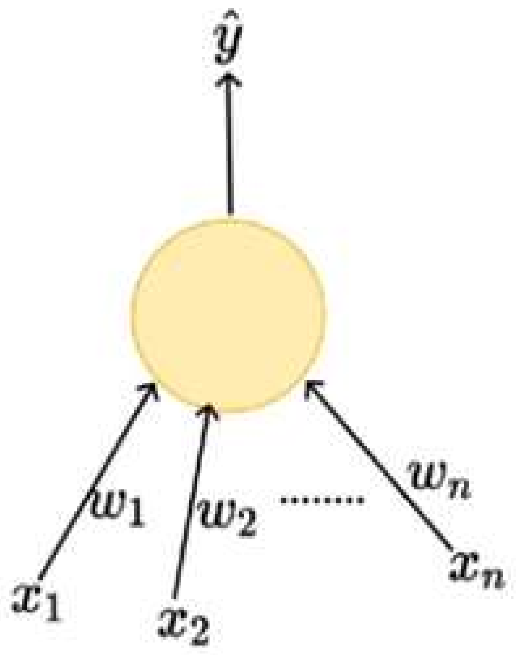


Why Sigmoid Neuron

- Perceptron model takes several real-valued inputs and gives a single binary output.
- In the perceptron model, every input xi has weight wi associated with it.
- The weights indicate the importance of the input in the decision-making process.
- The model output is decided by a threshold W_o if the weighted sum of the inputs is greater than threshold W_o output will be 1 else output will be 0.
- In other words, the model will fire if the weighted sum is greater than the threshold.

Why Sigmoid Neuron

- From the mathematical representation, we might say that the thresholding logic used by the perceptron is very harsh

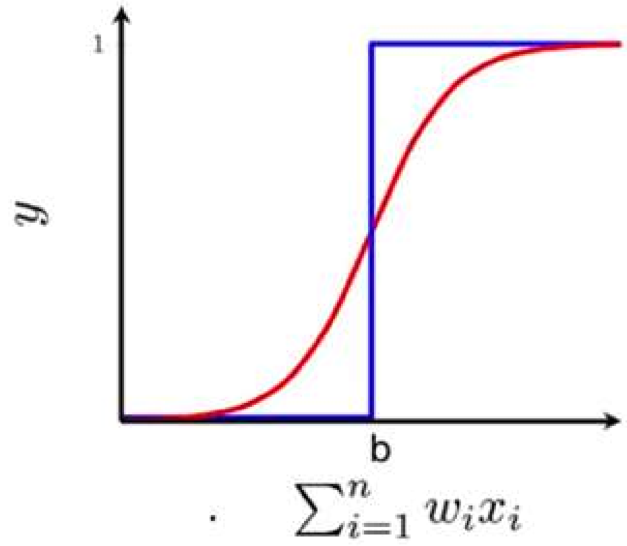


$$\hat{y} = 1 \text{ if } \sum_{i=1}^{n} w_i x_i \geq b$$

$$\hat{y} = 0 \text{ otherwise}$$

Sigmoid Neuron

- In sigmoid neurons where the output function is much smoother than the step function.
- In the sigmoid neuron, a small change in the input only causes a small change in the output as opposed to the stepped output.
- There are many functions with the characteristic of an "**S**" shaped curve known as sigmoid functions. The most commonly used function is the logistic function.



$$y = \frac{1}{1+e^{-(w^T x+b)}}$$

Sigmoid Neuron

- The **inputs to the sigmoid neuron can be real numbers** unlike the boolean inputs in MP Neuron and the **output will also be a real number** between 0–1.
- In the sigmoid neuron, trying to regress the **relationship between X and Y in terms of probability**.
- Even though the output is between 0–1, we can still use the sigmoid function for binary classification tasks by choosing some threshold.

Sigmoid Neuron

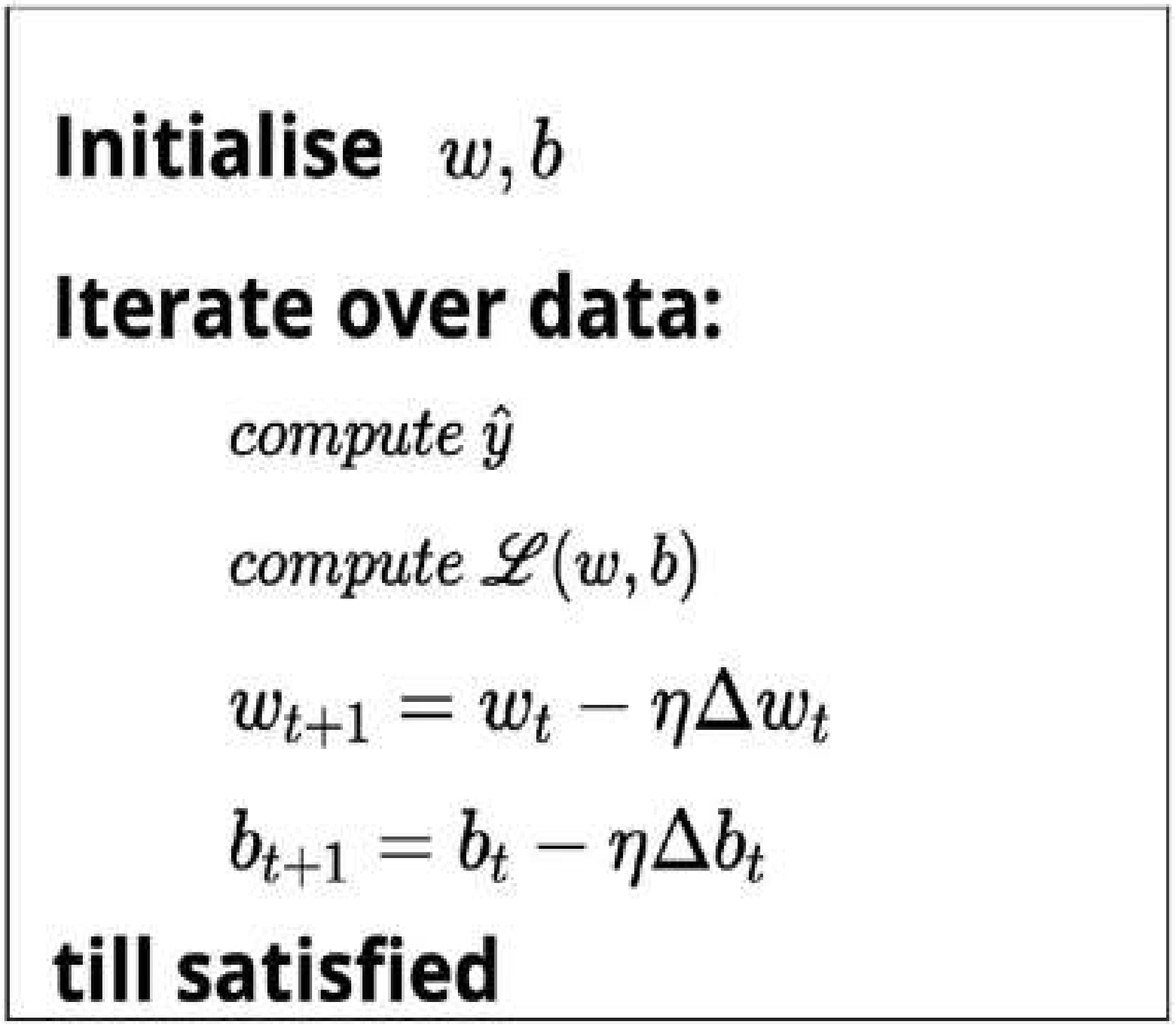
- Learning Algorithm
- algorithm for learning the parameters **w and b of the sigmoid neuron model by using the gradient descent algorithm**.

Find w and b such that:

$$\underset{w,b}{\text{minimize}}\, \mathscr{L}(w,b) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

- The objective of the learning algorithm is to determine **the best possible values for the parameters, such that the overall loss (squared error loss) of the model is minimized** as much as possible.

Sigmoid Neuron



```
Initialise $w, b$
Iterate over data:
    $compute\ \hat{y}$
    $compute\ \mathscr{L}(w, b)$
    $w_{t+1} = w_t - \eta \Delta w_t$
    $b_{t+1} = b_t - \eta \Delta b_t$
till satisfied
```

Sigmoid Neuron

- Initialize w and b randomly
- Iterate over all the observations in the data, for each observation find the corresponding predicted outcome using the sigmoid function
- Compute the squared error loss.
- Based on the loss value, update the weights such that the overall loss of the model at the new parameters will be less than the current loss of the model.
- Loss optimization

$$\mathscr{L}(w, b) > \mathscr{L}(w + \eta\Delta w, b + \eta\Delta b)$$

Activation Functions, Loss Function

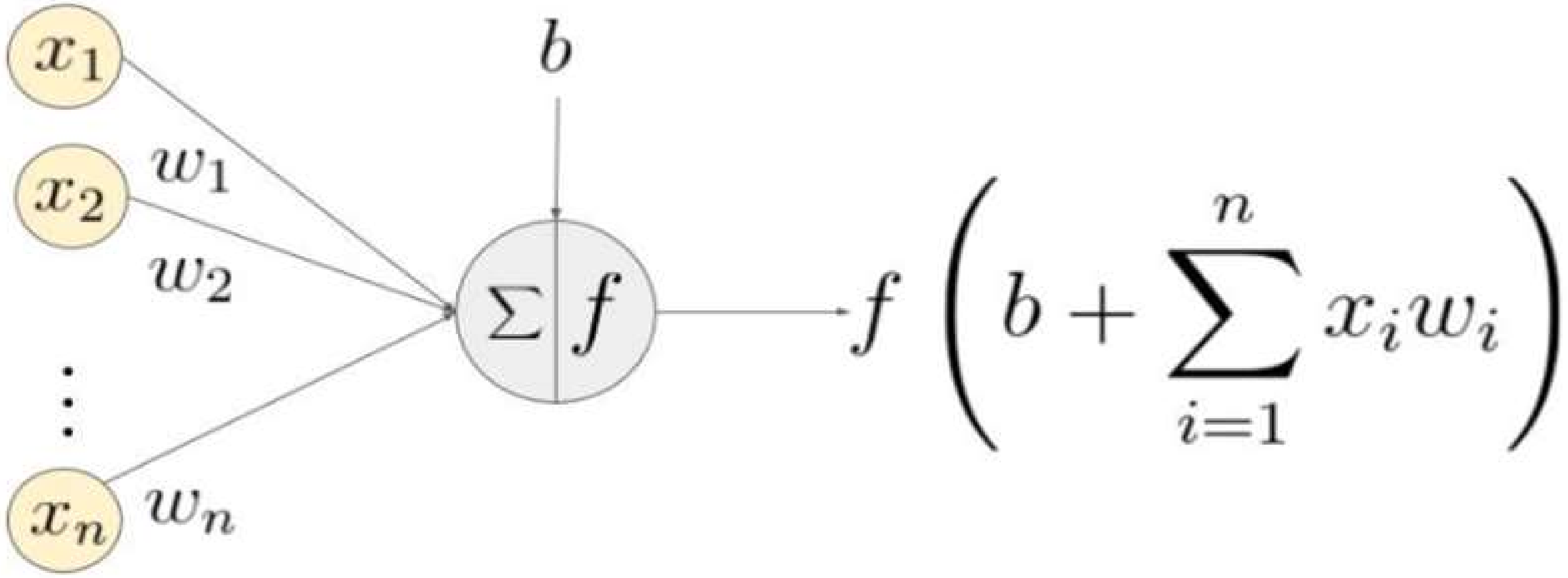
Sources:

1. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
2. https://medium.com/@abhigoku10/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8
3. https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de
4. https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

A Simple Neural Network

A Neuron [2]**:**

- (x_1,x_2, …x_n) - input signal vector
- (w_1,w_2,…w_n) - weights
- accumulation (i.e. summation + addition of bias b)
- an activation function f is applied to this sum



https://medium.com/@abhigoku10/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8

Activation Function

- An activation function is a very important feature of an Artificial Neural Network to learn and understand the complex patterns
- *A* mathematical equation that determine the *output of node*
- It helps to normalize the output of each neuron to a range between 1 and 0 or between -1 and 1
- *It is also known as* ***Transfer Function***

Activation Function

- The function
  - attached to each neuron in the network
  - determines whether neuron should be activated (“fired”) or not,
  - based on whether each neuron’s input is relevant for the model’s prediction
- Computationally efficient (calculated across thousands/millions of neurons for each data sample)
- The need for speed has led to the development of new functions such as ReLu

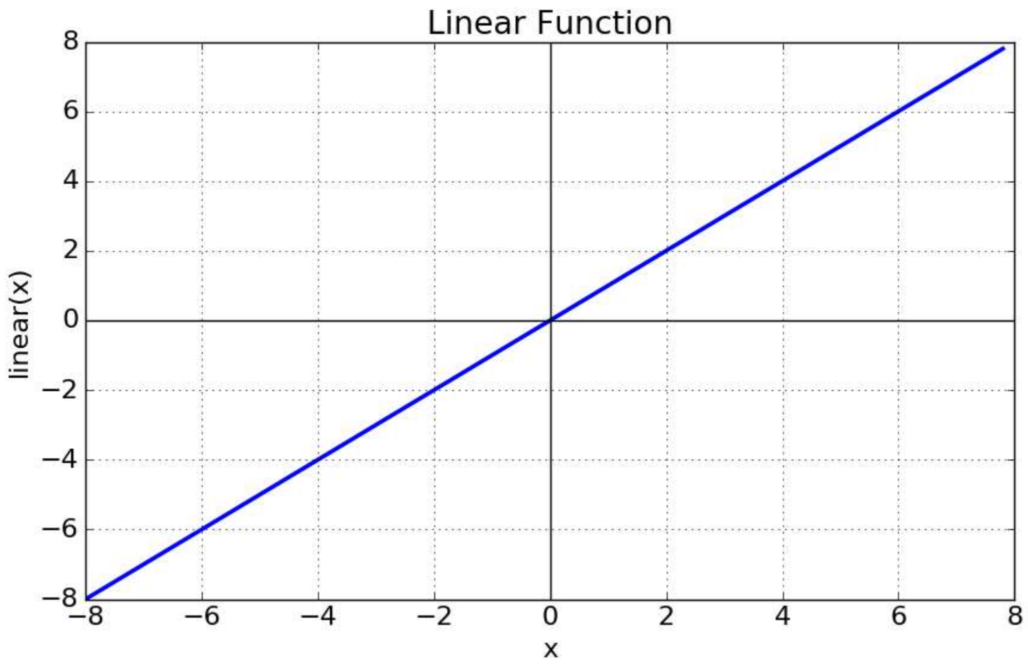
Activation Function (Types)

- The Activation Functions [1] can be basically divided into 2 types-
  - Linear Activation Function
  - Non-linear Activation Functions

https://medium.com/@abhigoku10/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8

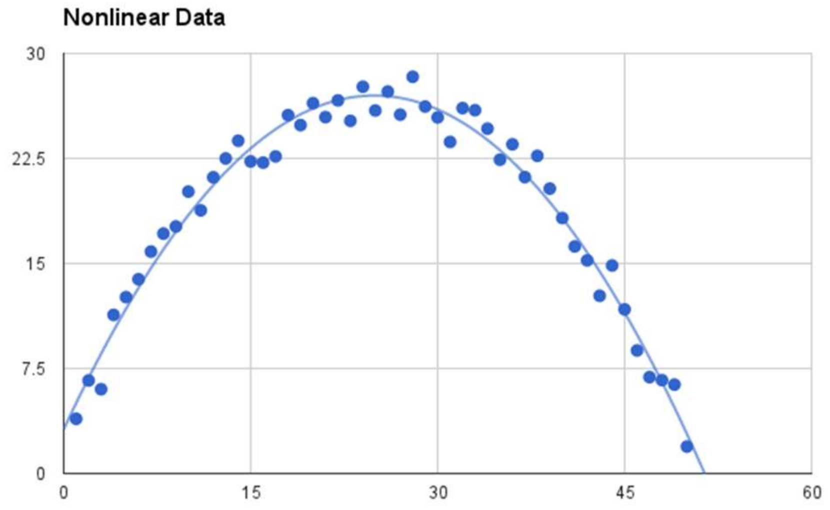
Activation Function (Linear)

- Linear Activation Function:
  - **Equation :** f(x) = x
  - **Range :** (-infinity to infinity)
  - It doesn't help with the complex data



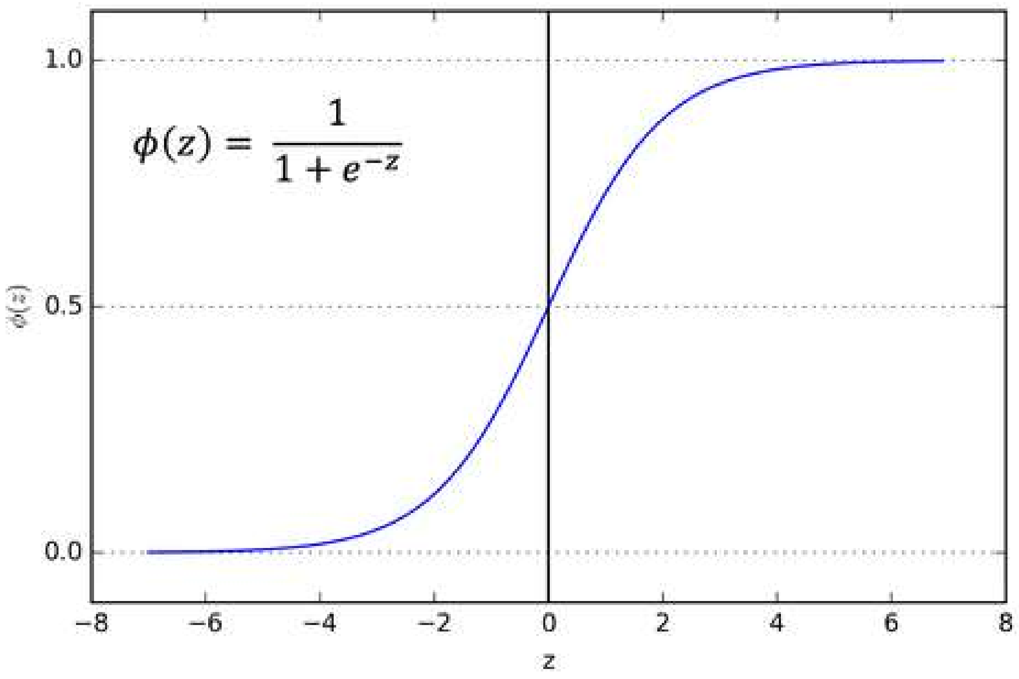
Activation Function (Non-Linear)

- Non-linear Activation Functions
  - The model generalizes or adapts with variety of data (images, video, audio, and have high dimensionality)
  - It allows backpropagation (derivative function which is related to the inputs)
  - Create a deep neural network ("stacking" of multiple layers of neurons)



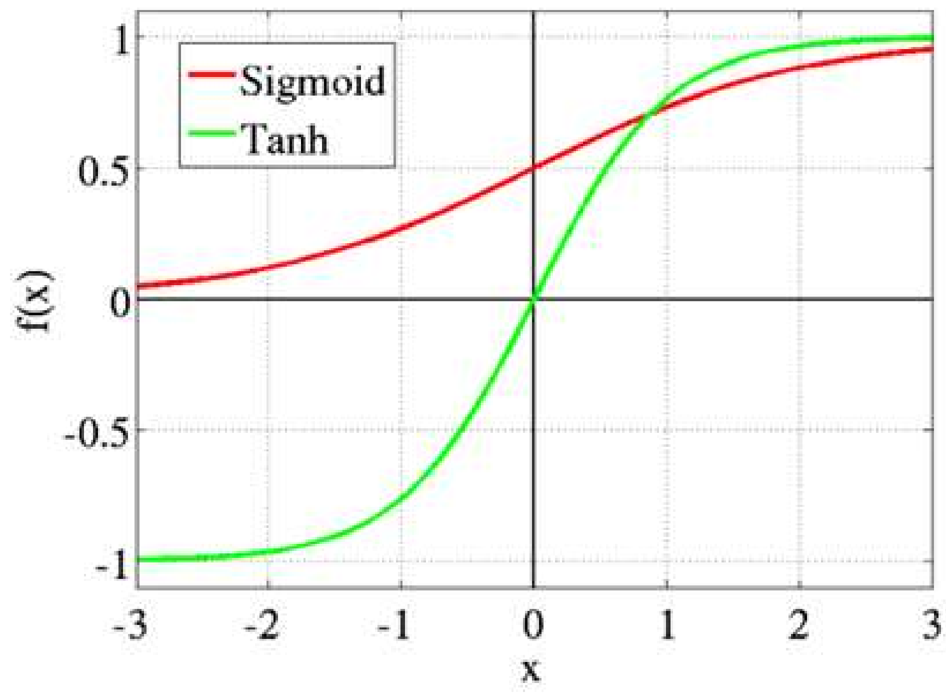
Activation Function (Non-Linear)

- The Nonlinear Activation Functions are mainly divided on the basis of their **range or curves**
- **Sigmoid or Logistic Activation Function**
  - A S-shape curve
  - Predict the probability
  - (0 and 1)
  - N/W can stuck at the training time



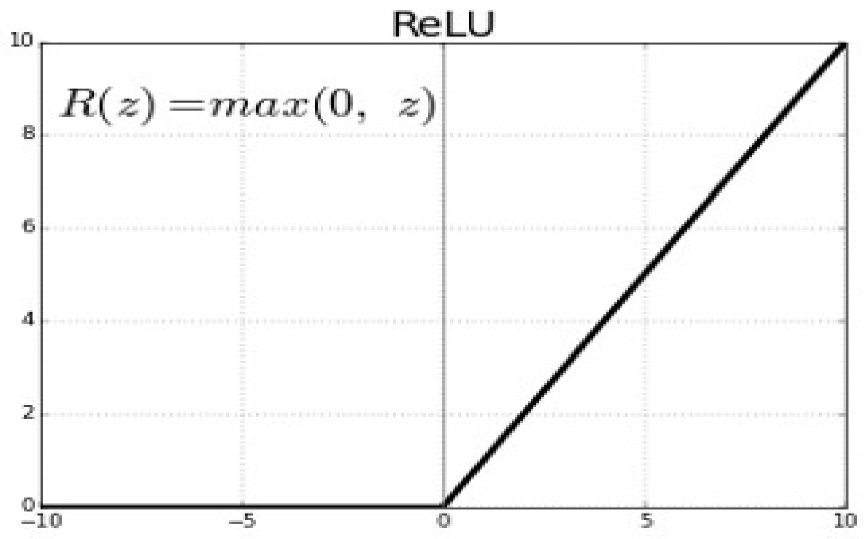
Activation Function (Non-Linear)

- **Tanh or hyperbolic tangent Activation Function**
  - tanh is also sigmoidal (S - shaped) with range -1 to 1
  - the negative inputs will also be mapped
  - *Mostly tanh &logistic sigmoid are used in feed-forward n/w*



Activation Function (Non-Linear)

- **ReLU (Rectified Linear Unit) Activation Function**
  - Widely used function (DL and CNN)
  - ReLU is half rectified (from bottom)
  - f(z) is zero when z is less than zero
  - f(z) is equal to z when z is above or equal to zero



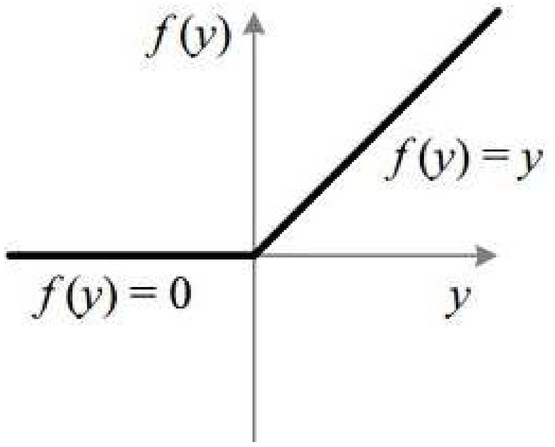
Activation Function (Non-Linear)

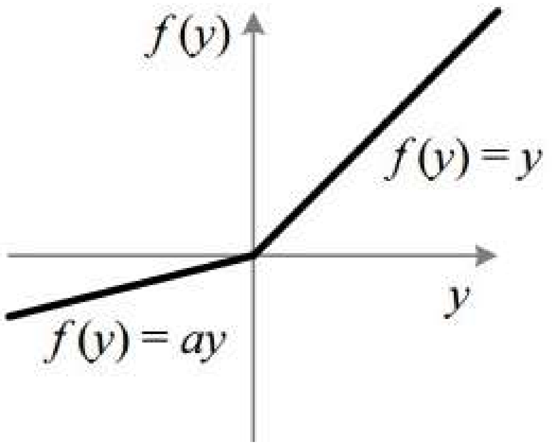
- **ReLU Advantages**
  - **Computationally efficient**—allows the network to converge very quickly
  - **Non-linear**—although it looks like a linear function, ReLU has a derivative function and allows for backpropagation

- **Disadvantages**
  - **The Dying ReLU problem**—when inputs approach zero, or are negative, the gradient of the function becomes zero, the network cannot perform backpropagation and cannot learn

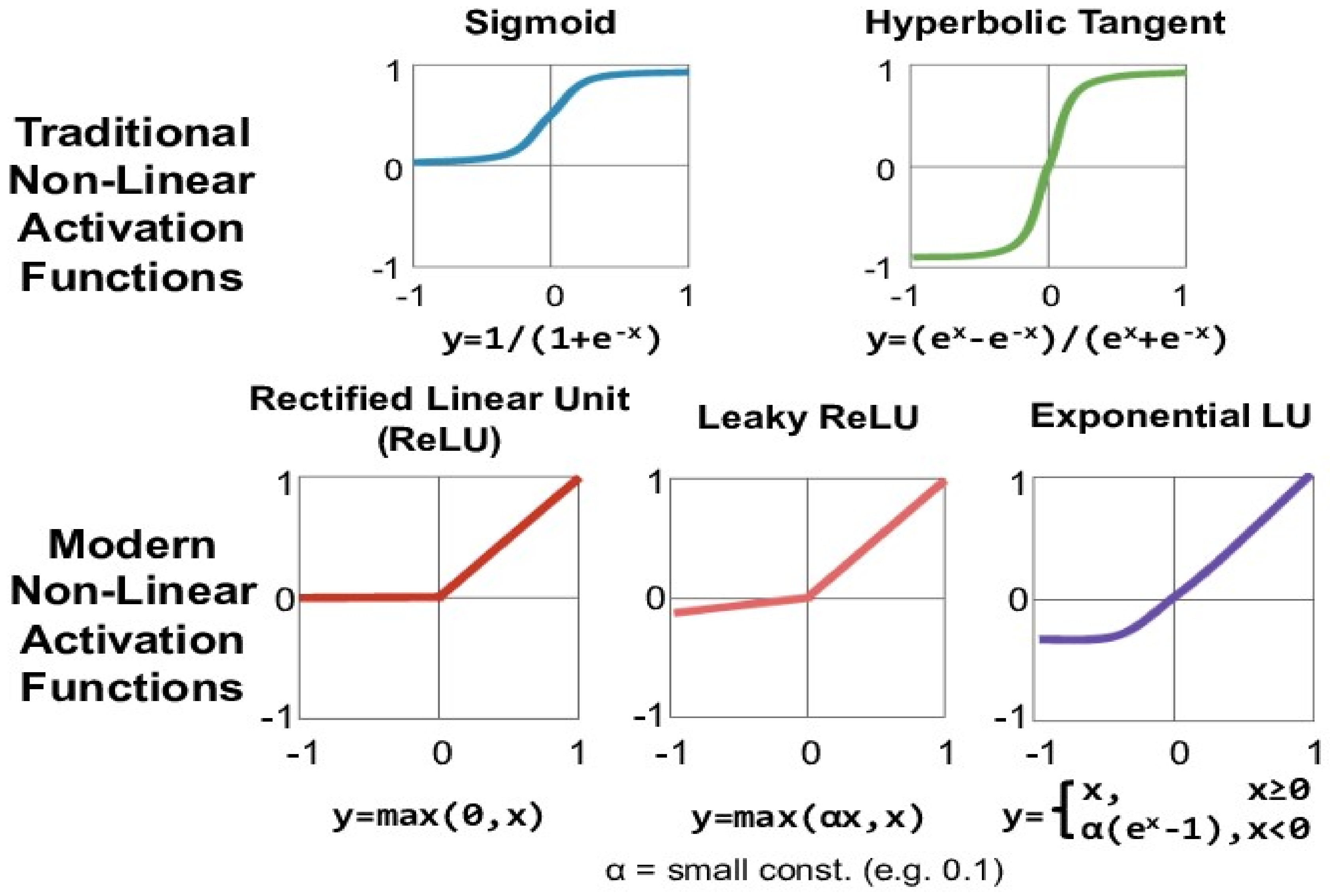
Activation Function (Non-Linear)

- **Leaky ReLU**
  - Attempt to solve the dying ReLU problem (a small positive slope in the negative area enables backpropagation)
  - The leak helps to increase the range of the ReLU function
  - f(x) = ax for x<0 and f(x) = x for x>0
  - Range **:** (0.01 to infinity)
  - When **a is not 0.01** then it is called **Randomized ReLU**.





Activation Function (Non-Linear)



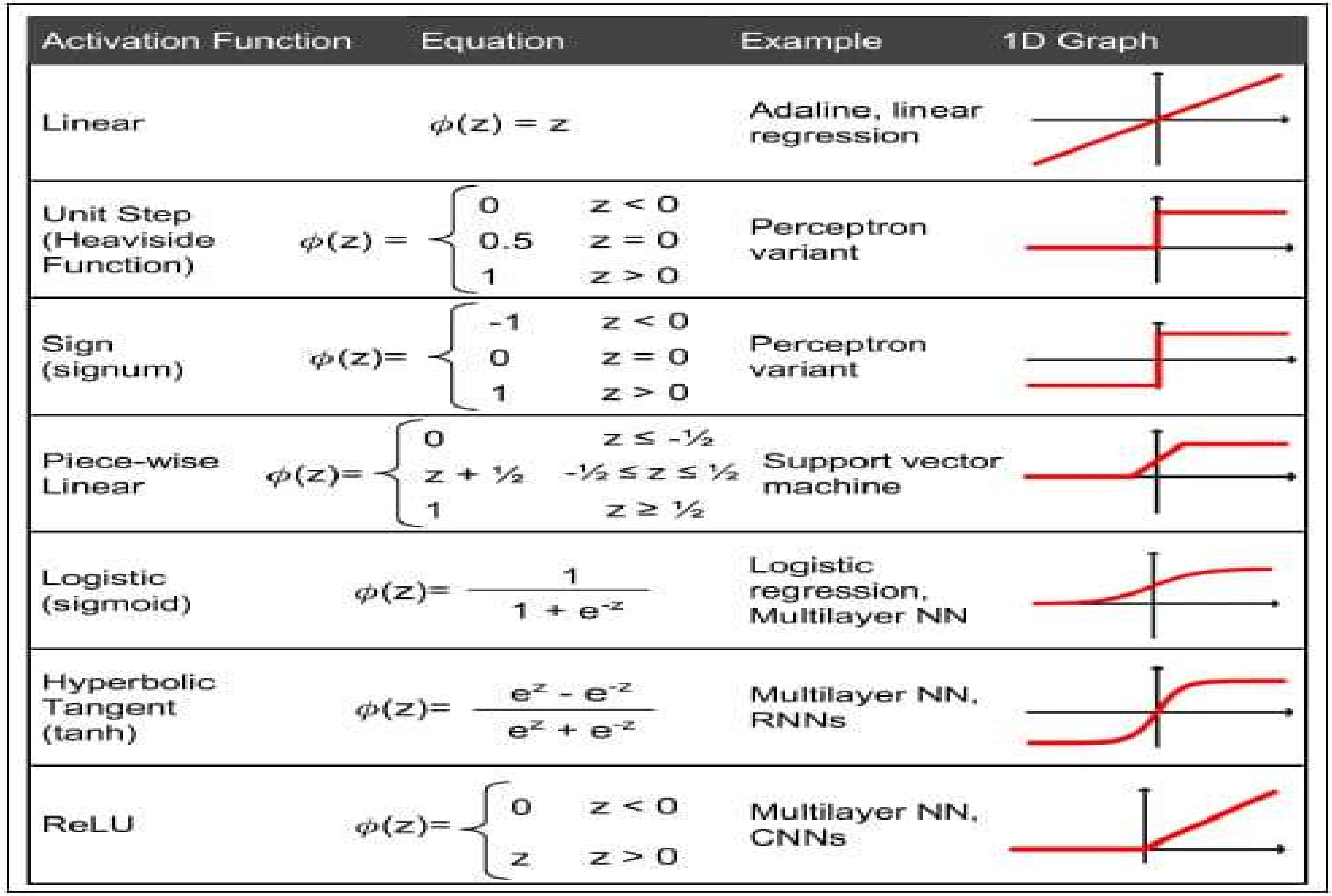
Src: Sze, Vivienne & Chen, Yu-Hsin & Yang, Tien-Ju & Emer, Joel. (2017). Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proceedings of the IEEE. 105.

Activation Function

- Heuristics to apply activation function
  - Sigmoid functions and their combinations generally work better in the case of classification problems
  - Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
  - Tanh is avoided most of the time due to dead neuron problem
  - ReLU activation function is widely used as it yields better results
  - In case of dead neurons in the networks, the leaky ReLU function is the best choice
  - ReLU function should only be used in the hidden layers

Summary of Activation Functions

- Various activation functions that can be used with Perceptron are shown here.



Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \le -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \le z \le \frac{1}{2} \\ 1 & z \ge \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

https://www.simplilearn.com/what-is-perceptron-tutorial

Loss Function

- In a supervised deep learning context the **loss function**
  - measures the quality of a particular set of parameters
  - based on how well the output of the network agrees with the ground truth labels in the training data

- **Loss function** is a method of evaluating "how well the algorithm models the dataset"

Nomenclature

loss function

=

cost function

=

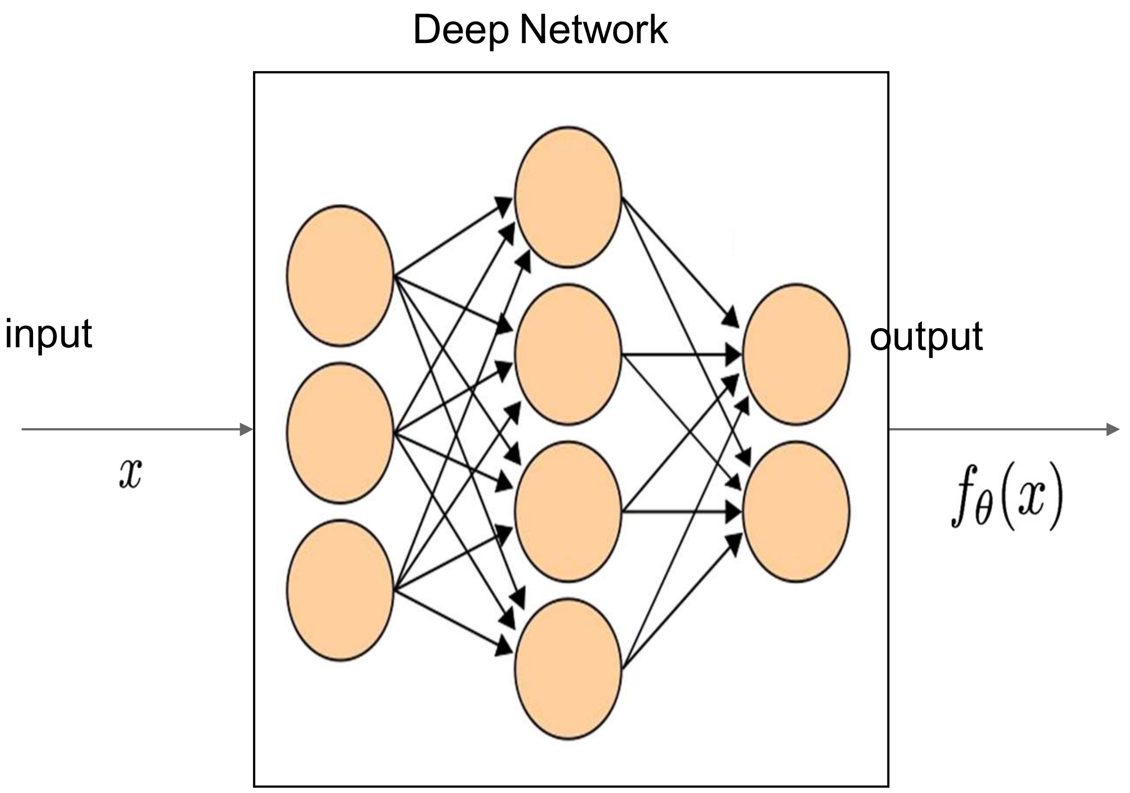
objective function

=

error function

Loss function

How good is the network with the training data?



labels (ground truth)

input

$$\mathcal{L}(w) = distance(f_\theta(x), y)$$

error

parameters (weights, biases)

Loss function

- If the predictions are totally off
  - Loss function will output a higher number
- If the predictions are pretty good
  - Loss function output a lower number
- Tune the algorithm to try and improve the model
  - Loss function will tell if its improving or not

- ‘Loss’ helps to understand how much the predicted value differ from actual value

https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de

Types of Loss function

- Regression Loss Function:
  - Regression models deals with predicting a continuous value
    - Ex. floor area, number of rooms, size of rooms, predict the price of the room.
  - The loss function used in the regression problem is called "Regression Loss Function"

Types of Loss function

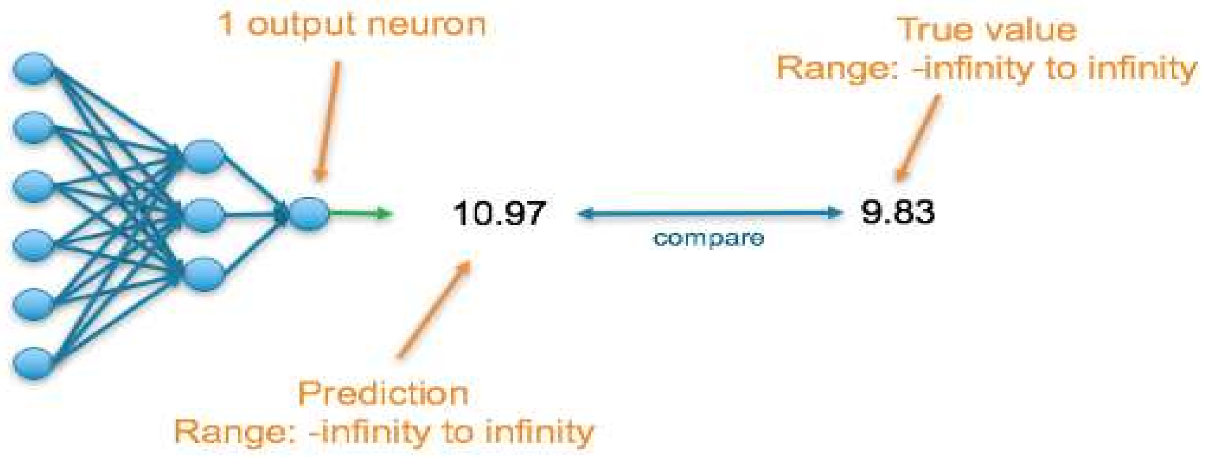
- Binary Classification Loss Functions:
  - Binary classification is a prediction algorithm where the output can be either 0 or 1
  - The output of binary classification algorithms is a prediction score (mostly)
  - So the classification happens based on the threshold the value (default value is 0.5)
    - If the prediction score > threshold then 1 else 0.

Types of Loss function

- Multi-class Classification Loss Functions:
  - Multi-Class classification are those predictive modeling problems where there are more target variables/class
  - It is just the extension of binary classification problem

Loss Function

- Regression: Predicting a numerical value
  - *E.g. predicting the price of a product*
  - The final layer of the neural network will have one neuron and the value it returns is a continuous numerical value
  - Compare the true value with predicted value



https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

Loss Function

- **Mean squared error (MSE)**
  - The average squared difference between the predicted value and the true value

$$\mathrm{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{test set}} \left(\underbrace{y_i}_{\text{predicted vaue}} - \underbrace{\hat{y}_i}_{\text{actual value}}\right)^2$$

Loss Function

- **Root Mean Square error (RMSE)**
  - Root Mean Square error is the extension of MSE
  - Its the average of square root of sum of squared differences between predictions and actual observations

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

Loss Function

- Binary Cross Entropy Loss Function
  - Binary cross entropy measures how far away from the true value (which is either 0 or 1) the prediction is for each of the classes
  - It averages these class-wise errors to obtain the final loss
  - Cross entropy is the difference between two probability distributions **p** and **q**, where **p** is our true output and **q** is our estimate of this true output
  - This difference is applied to neural networks

$$H(x) = \sum_{i=1}^{N} \overset{\text{true label}}{p}(x)\log \underset{\text{estimate}}{q}(x)$$

References

- https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
- https://medium.com/@abhigoku10/activation-functions-and-its-types-in-artifical-neural-network-14511f3080a8
- https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these-91167825a4de
- https://towardsdatascience.com/deep-learning-which-loss-and-activation-functions-should-i-use-ac02f1c56aa8

References

- https://www.simplilearn.com/what-is-perceptron-tutorial
- https://towardsdatascience.com/sigmoid-neuron-deep-neural-networks-a4cd35b629d7
- https://www.javatpoint.com/artificial-neural-network