Introduction to Neural Network

CSE451:Neural Network & Fuzzy Logic

MD Tamim Hossain
Lecturer
Department of Computer Science and Engineering
Premier University

"A baby learns to crawl, walk and then run. We are in the crawling stage when it comes to applying machine learning."

-**Dave Waters**

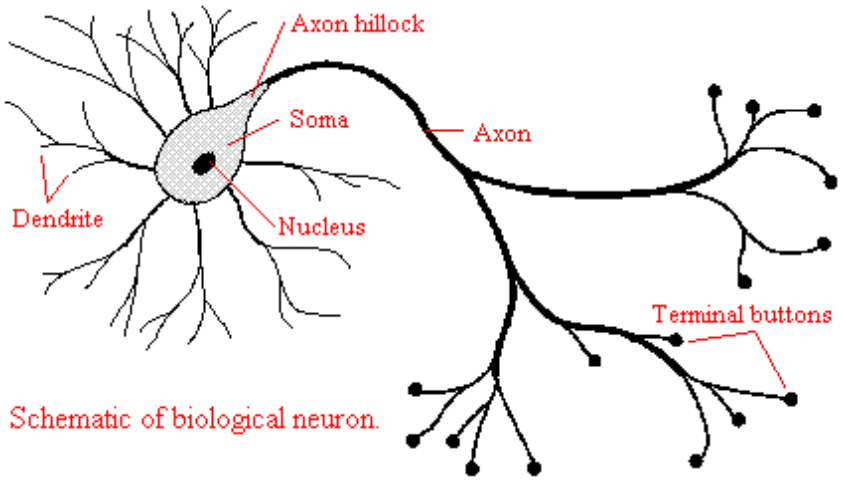
Why Artificial Neural Networks?

- Human brain has many desirable characteristics not presented in modern computers:
  - Massive parallelism,
  - Distributed representation and computation,
  - Learning ability,
  - Generalization ability,
  - Adaptability,
  - Inherent contextual information processing,
  - Fault tolerance, and
  - Low energy consumption

It is hope that devices based on biological neural networks will process some of these characteristics

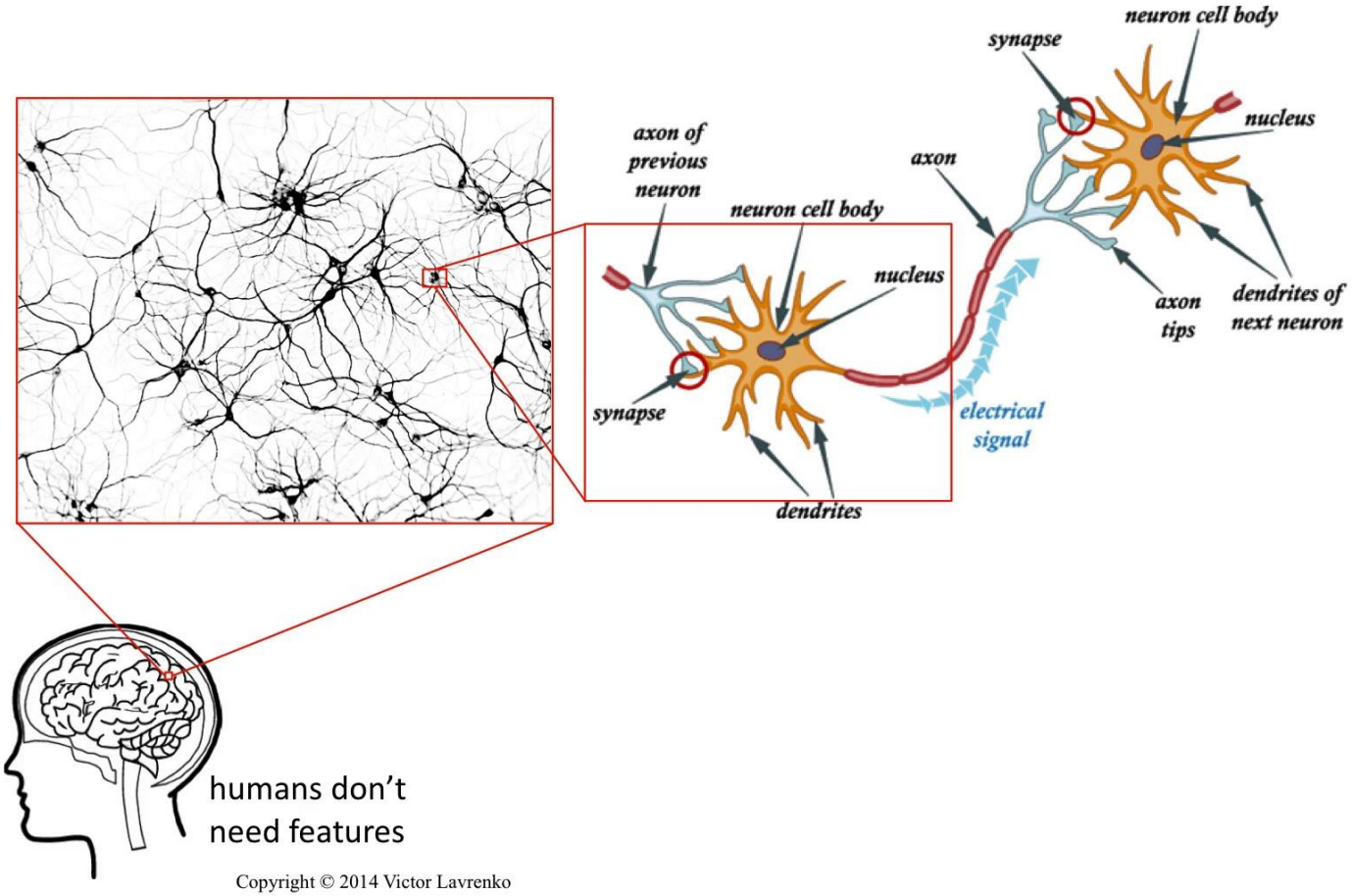
Biological Neuron to Artificial Neuron

- ANNs are inspired by biological neural networks



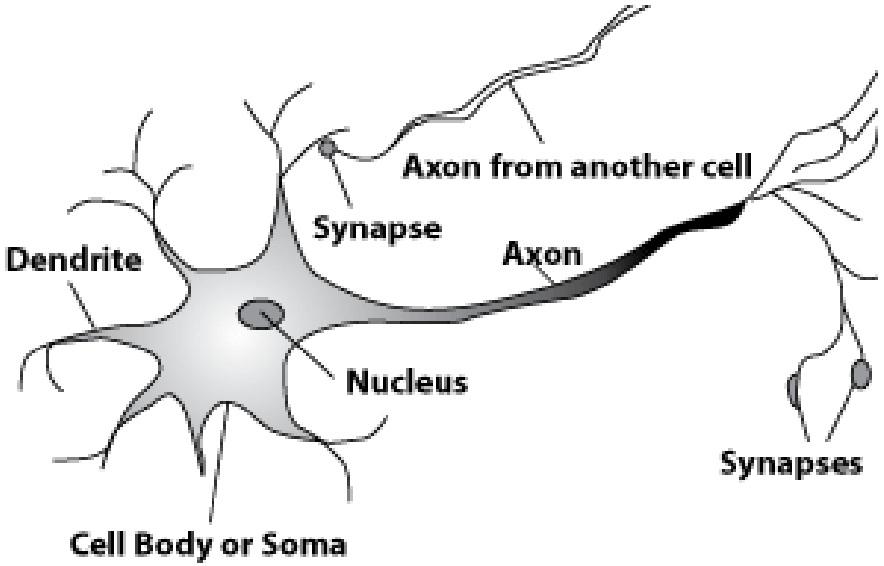
- A neuron (or nerve cell) is a special biological cell, the essence of life, with information processing ability
- It has been estimated that human brain contains around 10^11 neurons and each one is connected to 10^3 to 10^4 other neurons

Neuron : Learning Element in Brain



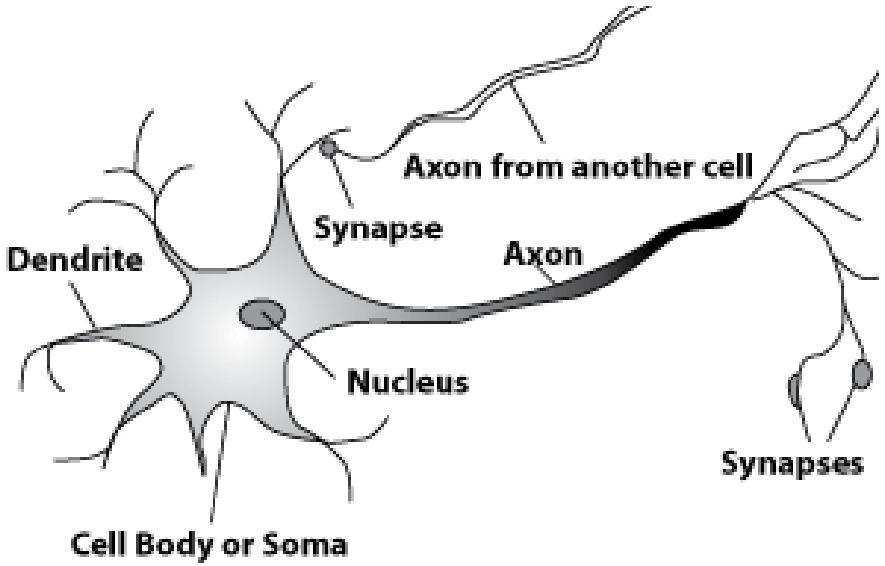
Biological neuron

- It is composed of a cell body, or soma, and two types of out
- Two types of out reaching tree branches: the axon and the dendrites
- A nucleus that contains information about hereditary traits
- A plasma



Biological neuron

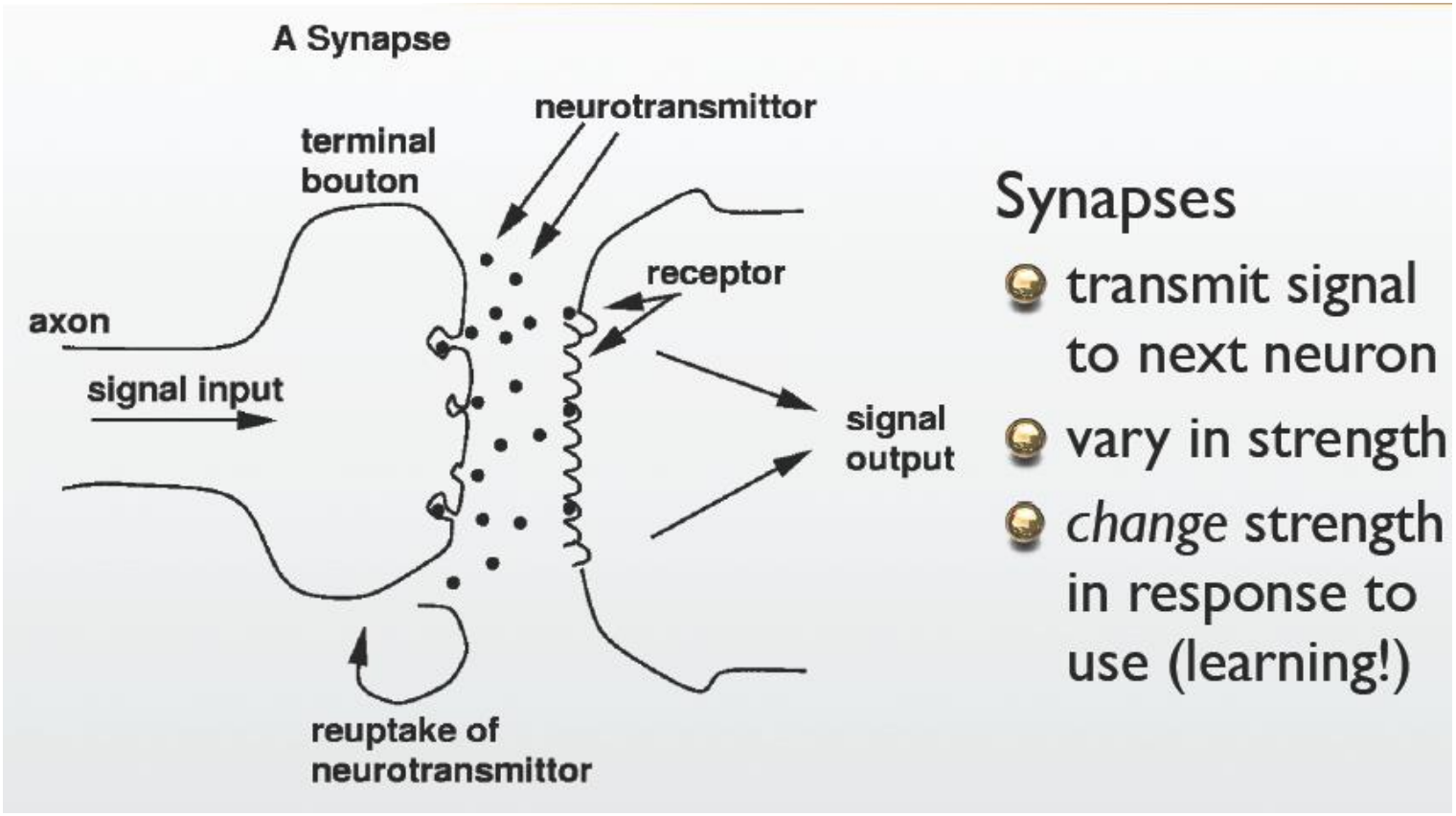
- A neuron receives signals (impulses) from other neurons through its **dendrites** (receivers)
- Transmits signals generated by its cell body along the **axon** (transmitter), which



Synapses

- A synapse is an elementary structure and functional unit between two neurons (an axon strand of one neuron and a dendrite of another)
- When the impulse reaches the synapse's terminal, certain chemicals called **neuro transmitters** are released.
- Synapse's effectiveness can be adjusted by the signals passing through it so that the synapses can learn from the activities in which they participate
- Dependence on history acts as a memory, which is possibly responsible for human memory.

Synapses


A Synapse
neurotransmittor
terminal bouton
receptor
axon
signal input
signal output
reuptake of neurotransmittor
Synapses
transmit signal to next neuron
vary in strength
change strength in response to use (learning!)

In a Word

Dendrite: Receives signals from other neurons

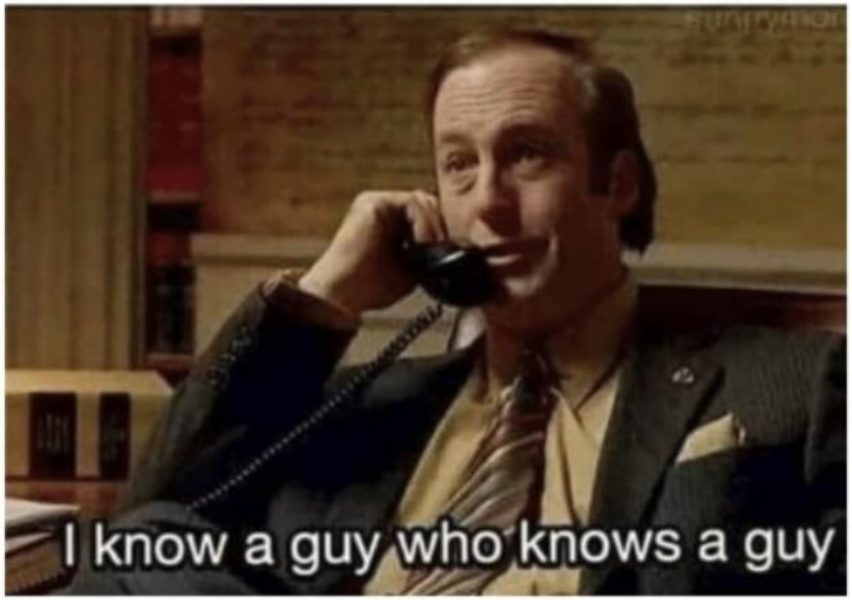
Soma: Processes the information

Axon: Transmits the output of this neuron

Synapse: Point of connection to other neurons

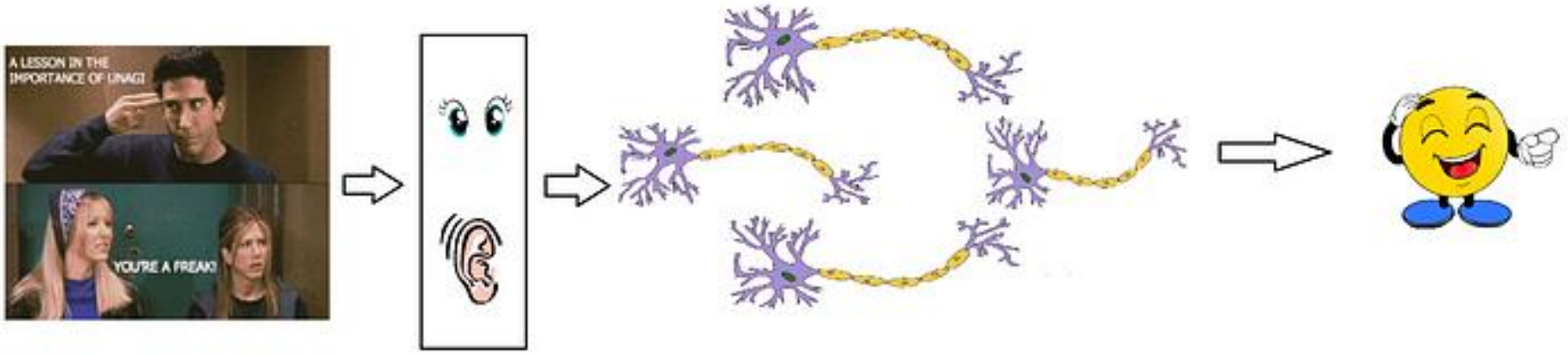
How Neural Networks work?

Neurons:

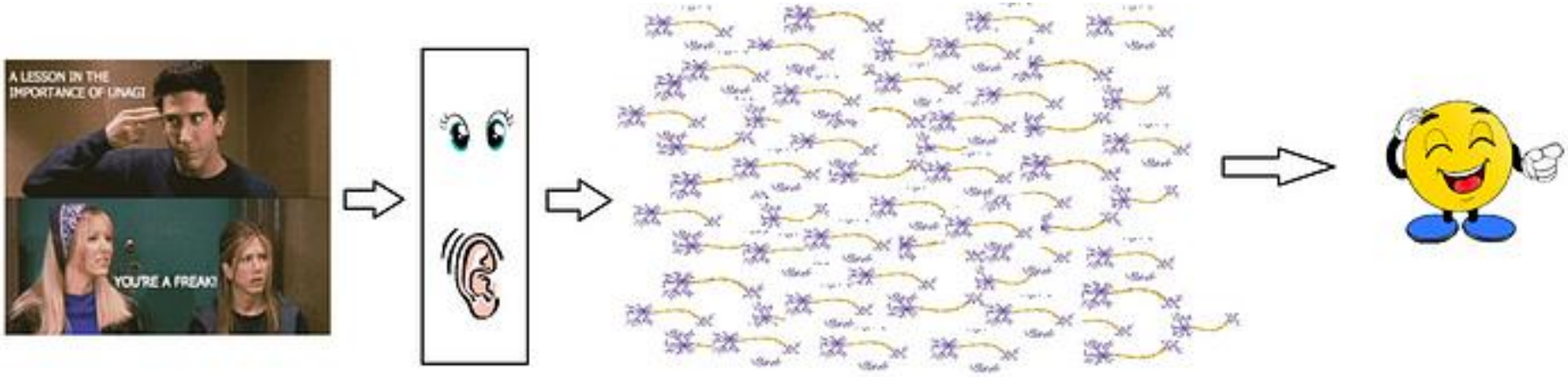


ProgrammerHumor.io

Laugh or Not Laugh?

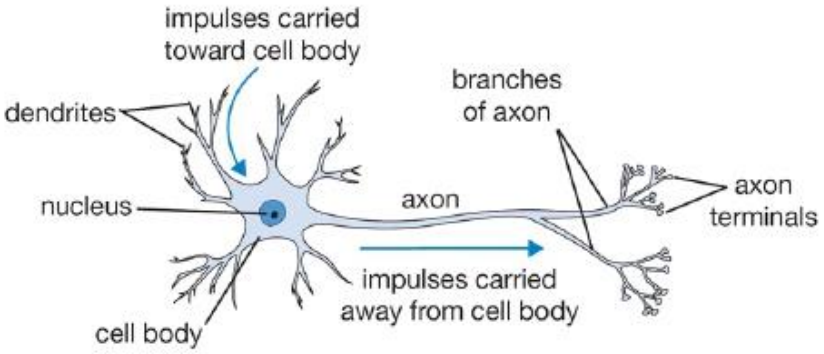


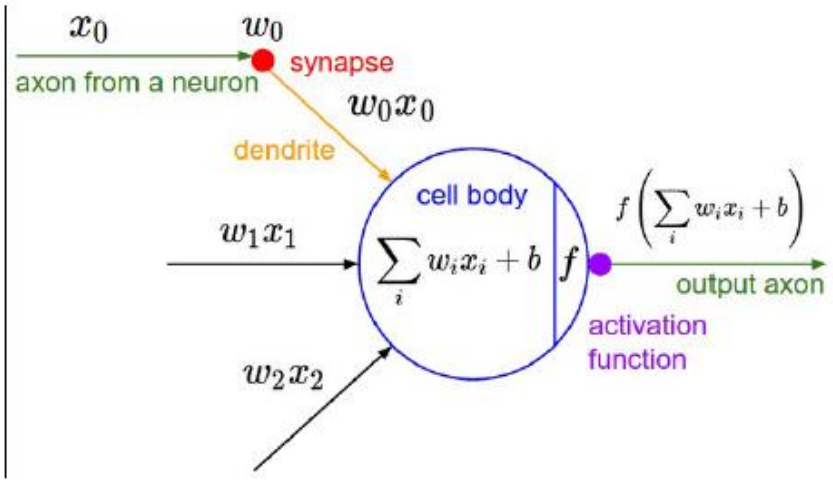
Laugh or Not Laugh?



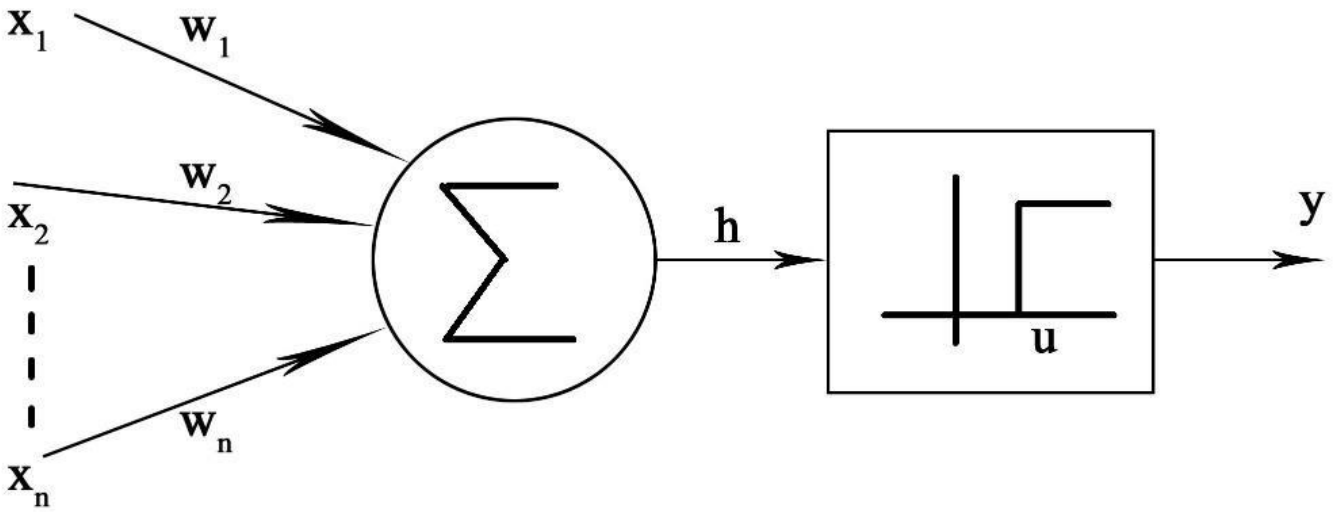
McCulloch-Pitts model of a Neuron

- ✓ McCulloch (NeuroScientist) and Pitts (Logician) proposed a highly simplified computational model of a Neuron
- ✓ It is based from Biological Neuron. It is called as M-P Neuron



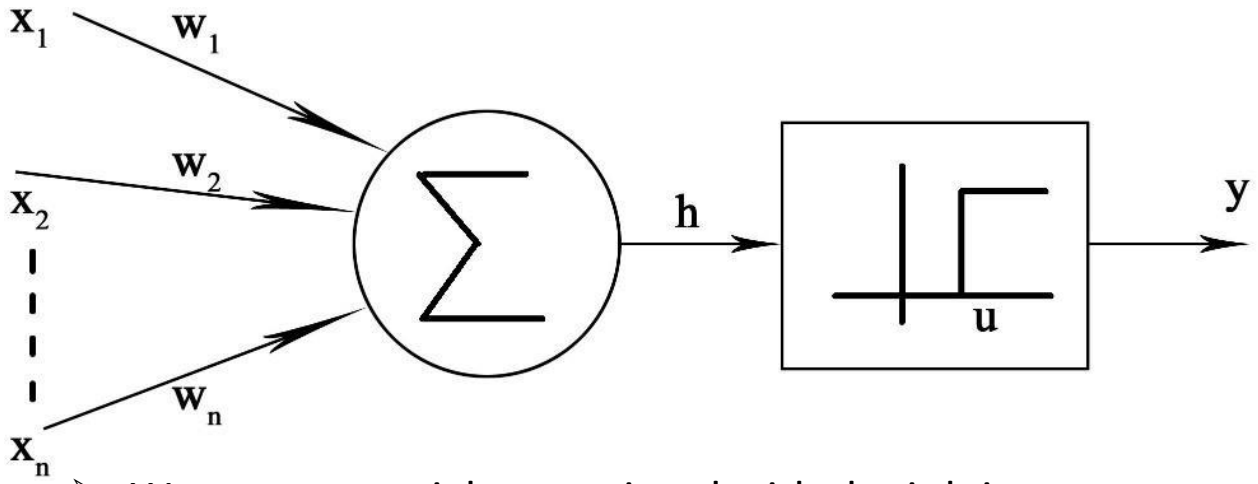


McCulloch-Pitts model of a Neuron



Wires and interconnections = Axons and dendrites
Connection weights = Synapses
Threshold function = Activity in a soma

McCulloch-Pitts model of a Neuron



- W =synapse weight associated with the j th input.
- Positive weights correspond to excitatory synapses
- Negative weights model inhibitory ones

- Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs
- Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together.

Watch a random football game or not?

- Inputs and outputs are Boolean(0 & 1)

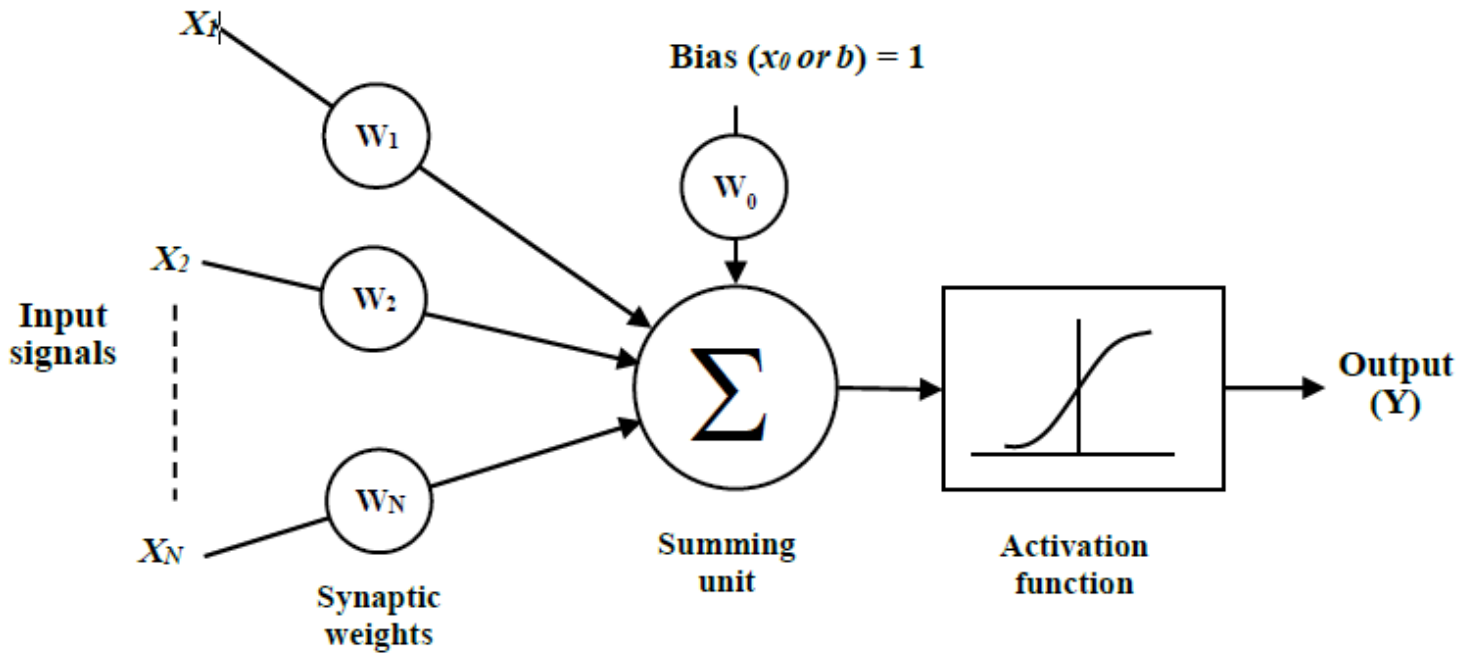
- x_1 could be isPremierLeagueOn (I like Premier League more)
- x_2 could be isItAFriendlyGame (I tend to care less about the friendlies)
- x_3 could be isNotHome - Inhibitory inputs
- x_4 could be isManCityPlaying (I am a big Man City fan)

$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

Theta = 2 (Threshold)

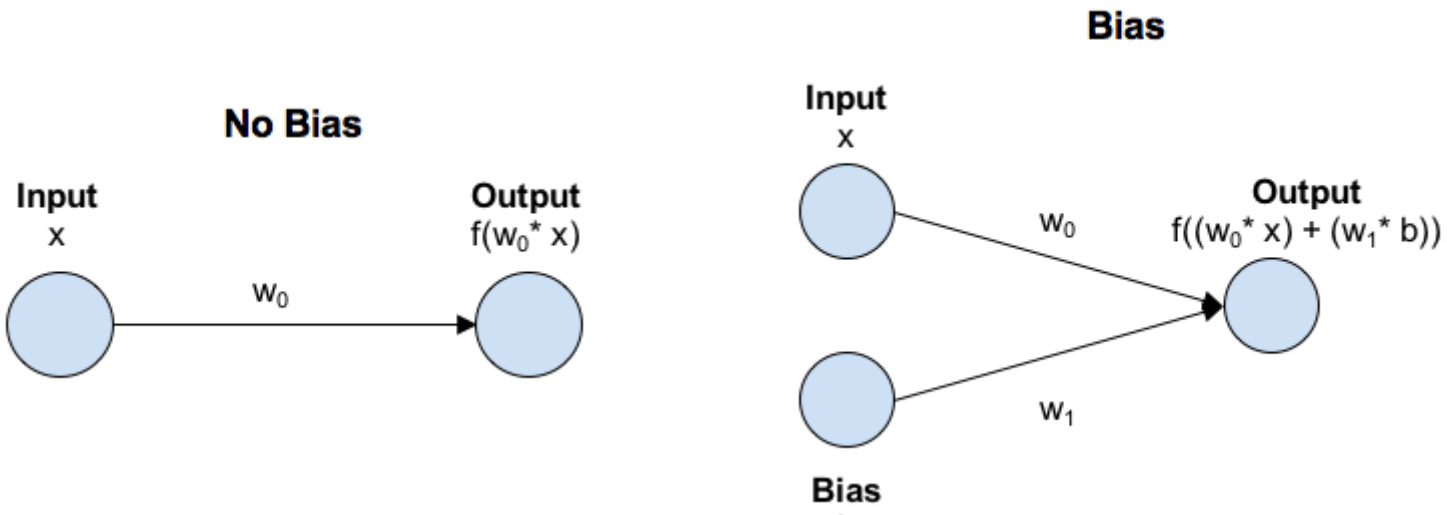
Generalized model of an artificial neuron



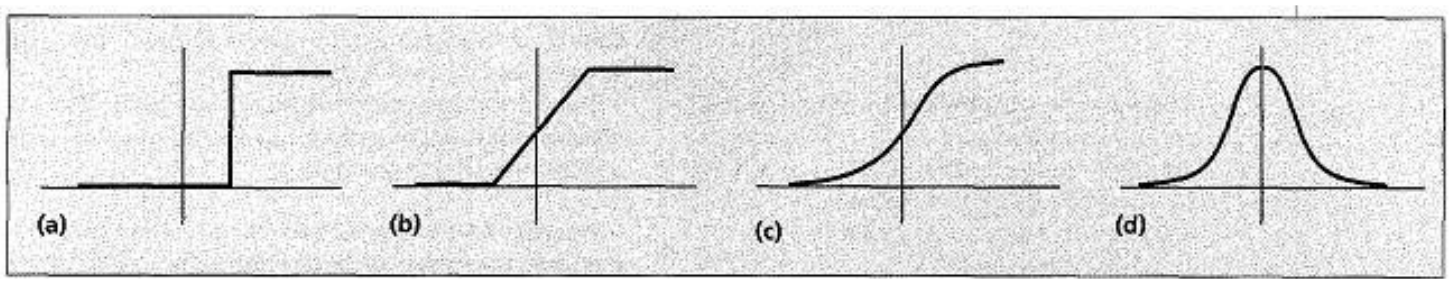
Role of the BIAS in Neural Networks

- A bias value allows you to shift the activation function to the left or right, which may be critical for successful learning

$$y = mx + c$$



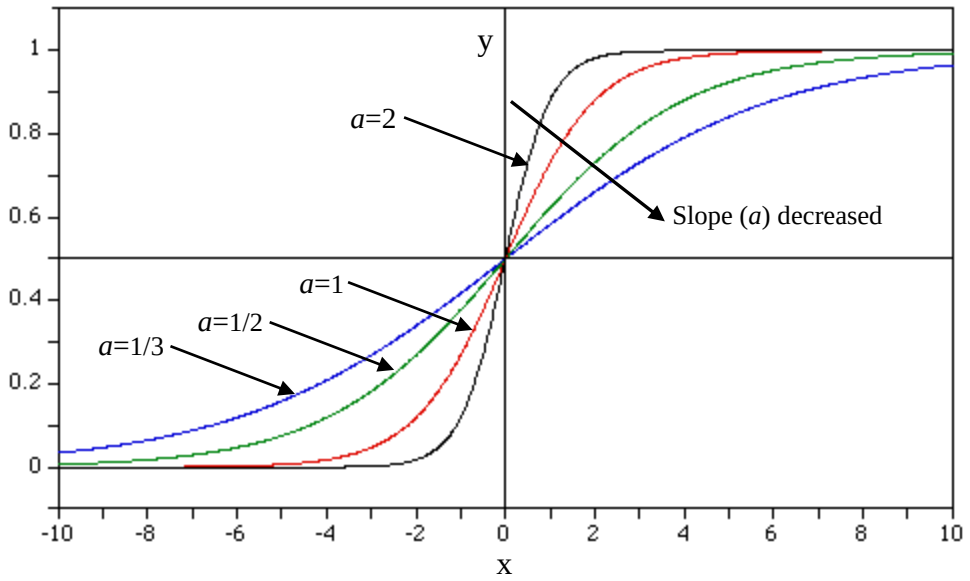
Different types of activation functions



Different types of activation functions: (a) threshold, (b) piecewise linear,(c) sigmoid, and (d) Gaussian

Sigmoid Activation function

$$y = \frac{1}{1 + e^{-x}}$$



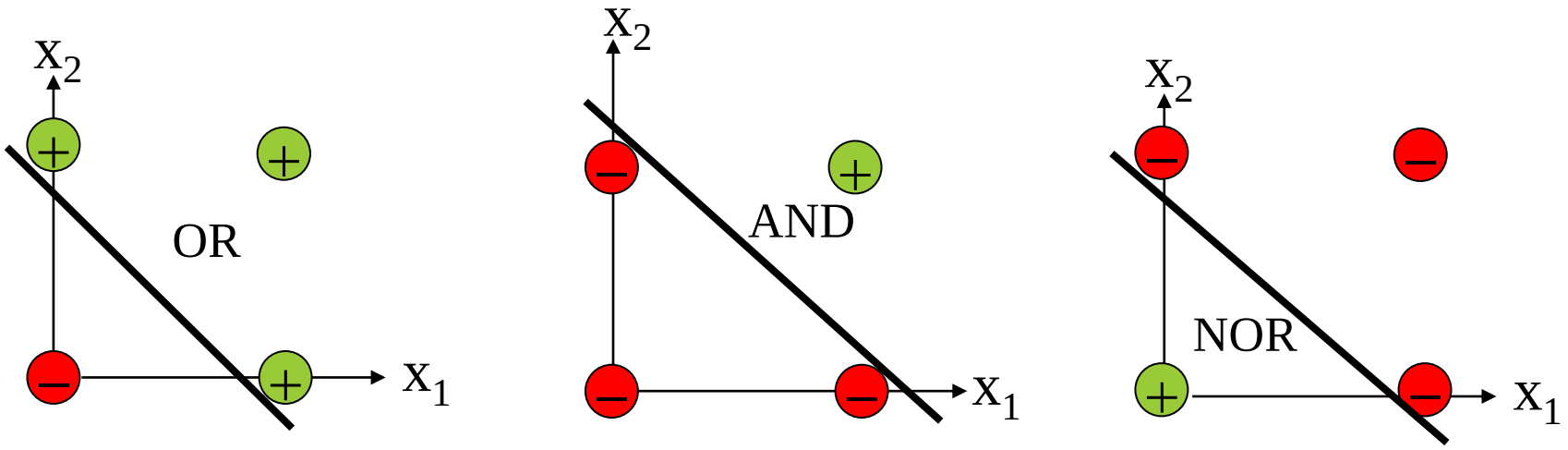
Sigmoid function with various slope (a) values.

Solving Logical Problem with a Single Neuron

- Any such single boundary problem (e.g., AND, NOR, NAND) may be solved by single neuron - linearly separable problem
- Single neuron is able to draw a single hyperplane to separate the input space in two portions

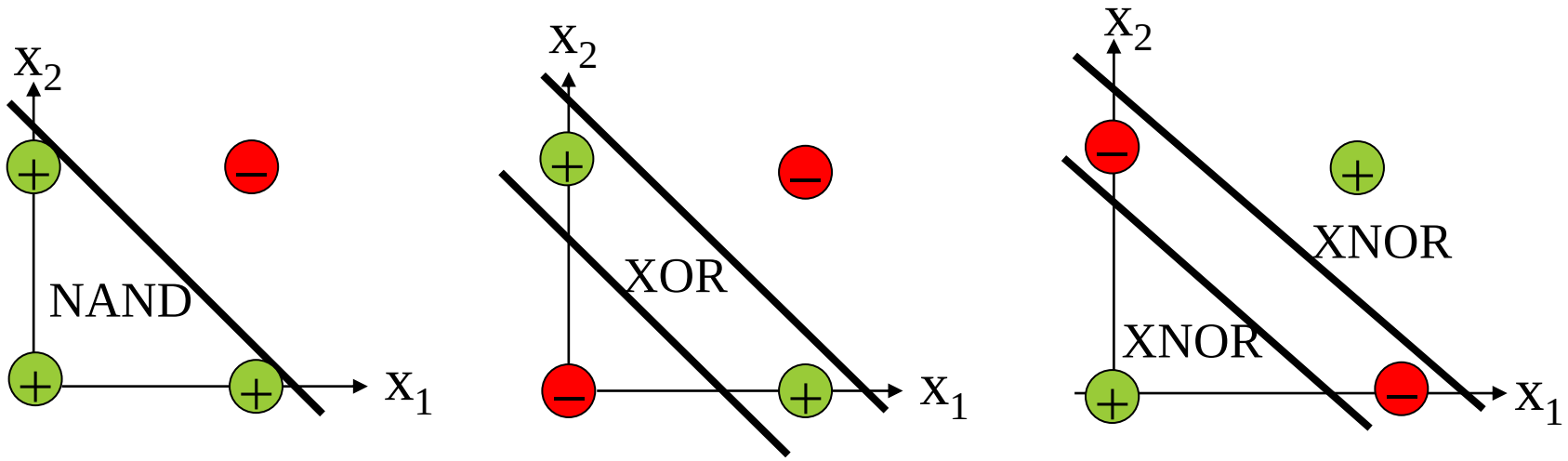
Input		Output (Y) for logic gate operation					
X_1	X_2	OR	AND	NOR	NAND	XOR	XNOR
0	0	0	0	1	1	0	1
0	1	1	0	0	1	1	0
1	0	1	0	0	1	1	0
1	1	1	1	0	0	0	1

Logical Operations



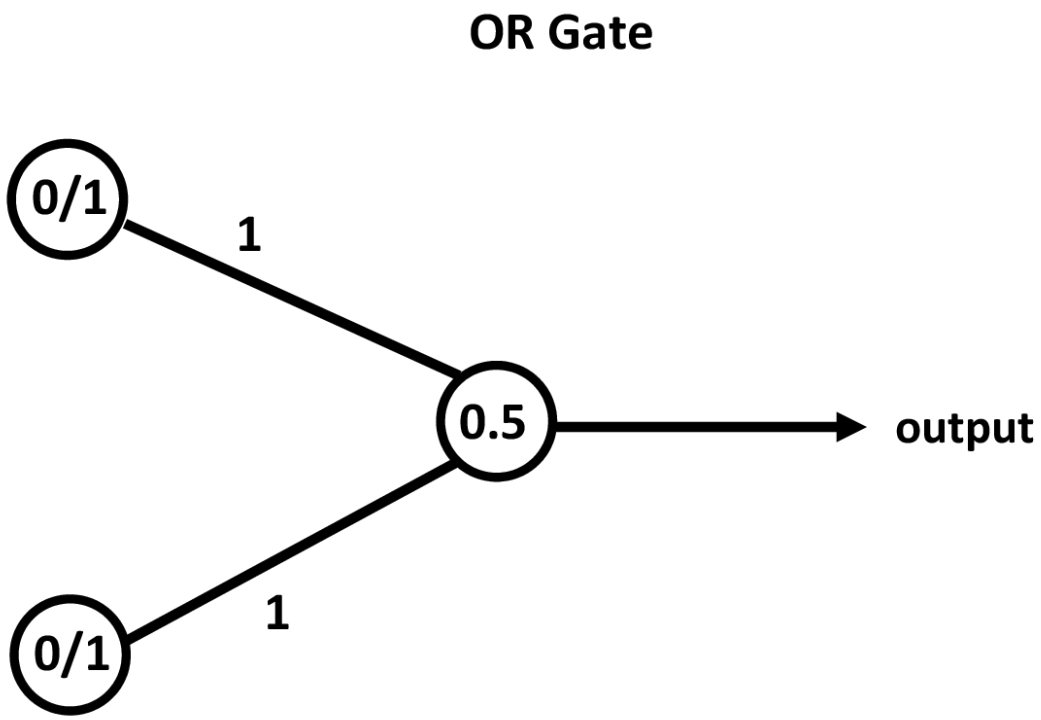
Boundary Line for Logic gates

Logical Operations

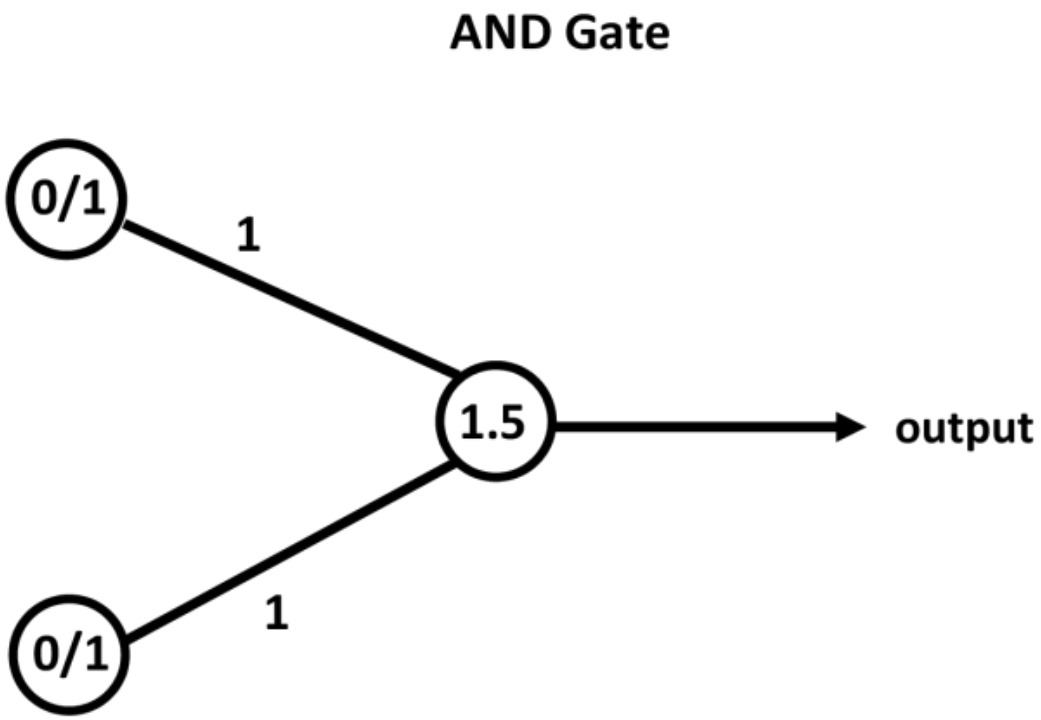


Boundary Line for Logic gates

OR Gate



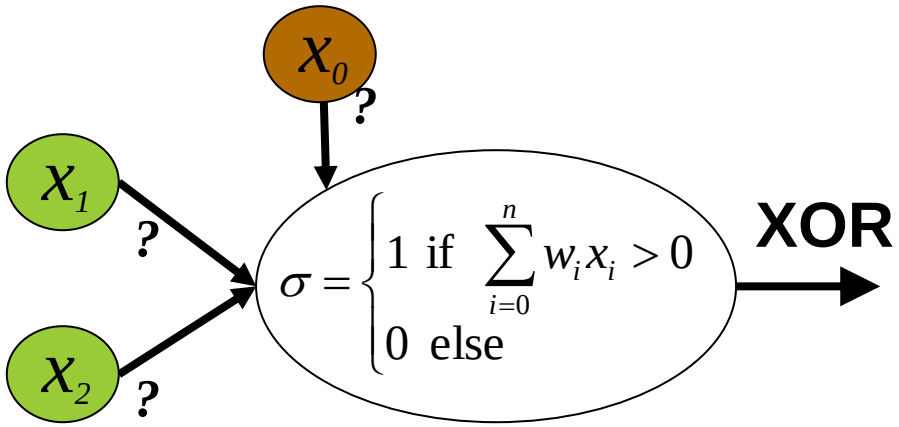
AND Gate

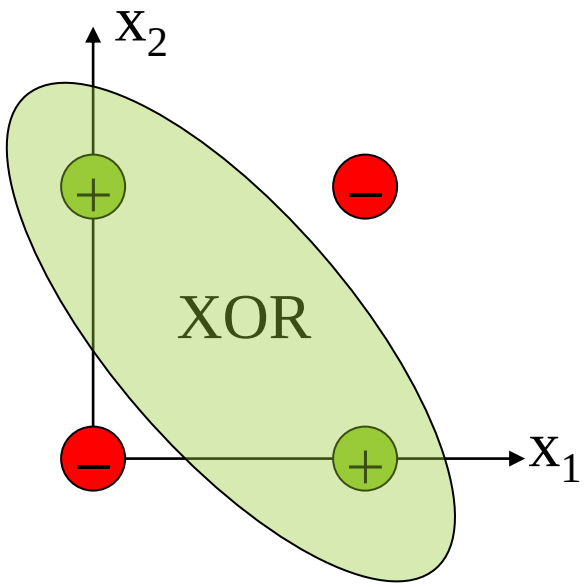


Single decision line logic operations

Logic Operation	W_1	W_2	W_0
OR	+1	+1	- 0.5
AND	+1	+1	- 1.5
NOR	- 1	- 1	+ 0.5
NAND	- 1	- 1	+ 1.5

Nonlinear Problem

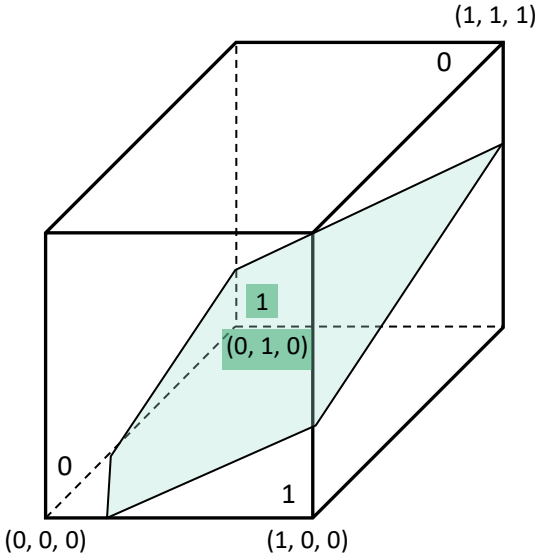




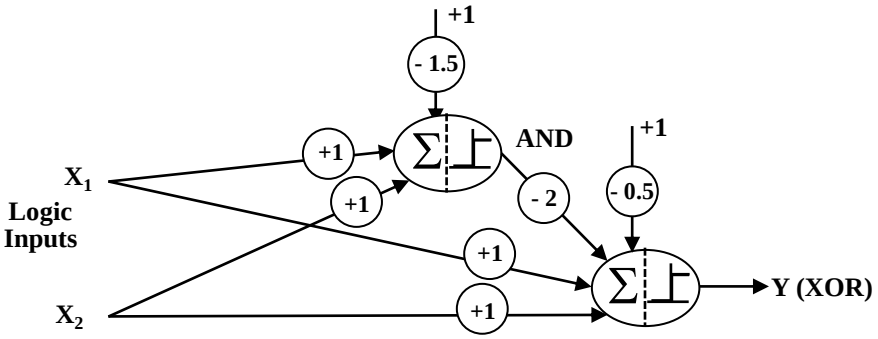
No arrangement work, not linearly separable. Require two boundary lines.

- **A nonlinear problem of lower dimension may be a linearly separable if transfer to higher dimension.**

XOR solution in Higher Dimension



a) Third dimension uplift XOR (1, 1) as (1, 1, 1) in three dimension.



b) XOR solution with additional input from AND gate.

An additional dimension (input) converts the problem to a linearly separable.

XOR solution in Higher Dimension

- The truth table that the response of OR and XOR is same for three inputs (0, 0), (0, 1) and (1, 0).

- XOR output is taken from the three input OR unit and the additional input is the outcome of AND unit using the inputs

XOR solution with HN

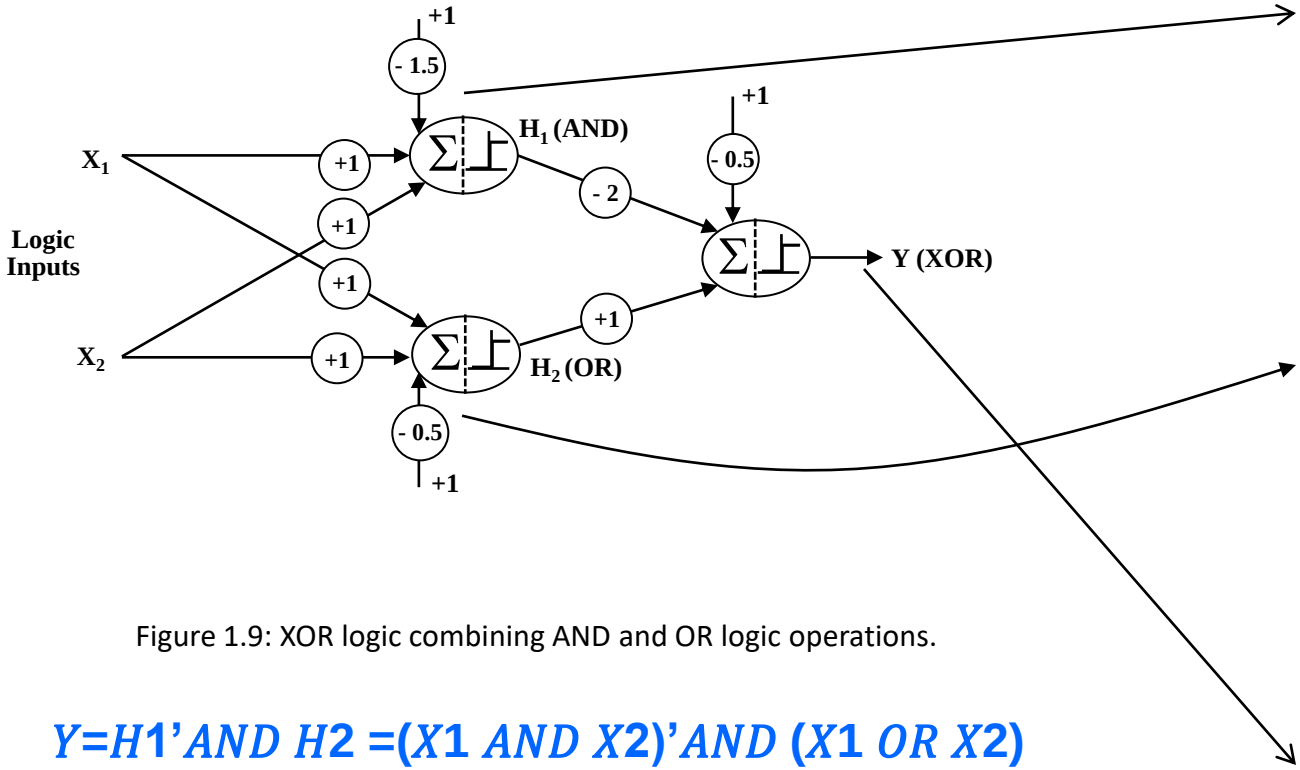
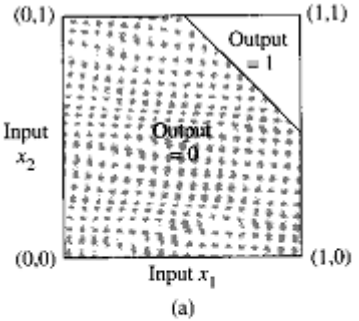
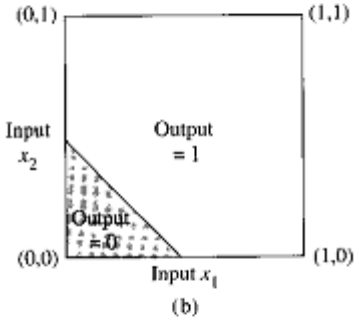


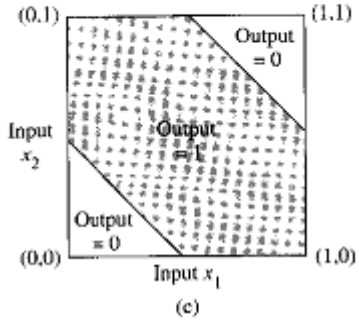
Figure 1.9: XOR logic combining AND and OR logic operations.



AND



OR

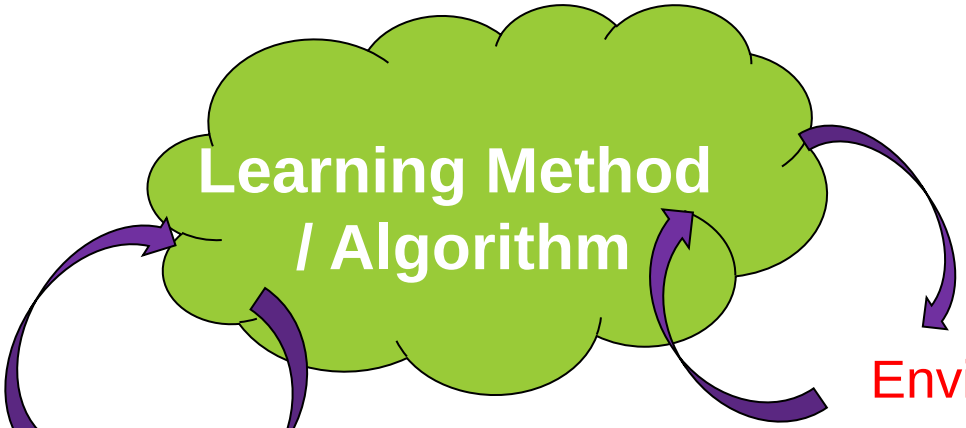


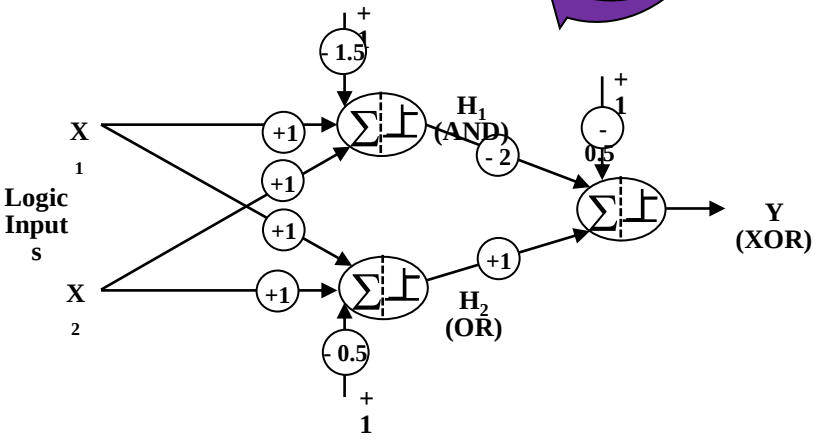
XOR

$Y = H1' \; AND \; H2 = (X1 \; AND \; X2)' \; AND \; (X1 \; OR \; X2)$

XOR logic through manipulation by hidden neurons

Learning





X1	X2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Learning

The ability to learn is a fundamental trait (characteristic) of intelligence.

- ANN-> Updating NN architecture and connection weights to perform a specific task efficiently.

- ANN ability to automatically learning from examples makes them attractive.

- To understand or design a learning process need: a model of environment or information is available to NN and learning rules.

- A learning algorithm -> procedure for adjusting weights

- Three main learning paradigms-> Supervised (learning with a teacher), unsupervised and hybrid

- Reinforcement learning is variant of supervised learning receives critique on the correctness of network output instead of correct answer.

Learning

- Learning theory must address three fundamental and practical issues:

1. Capacity – how many patterns can be stored , what functions and decision boundaries a NN can perform

2. Sample Complexity – Number training patterns needed to train

3. Computational Complexity – Time required for a learning algorithm to estimate a solution

X1	X2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Multilayer Feed-Forward Neural Networks

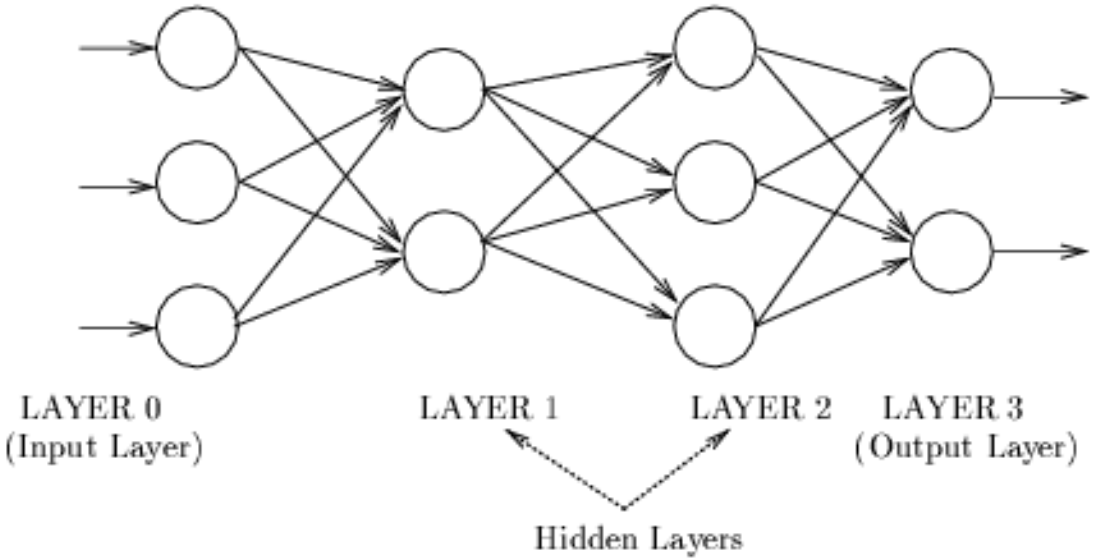
Feedforward Propagation - The flow of information occurs in the forward direction. The input is used to calculate some intermediate function in the hidden layer, which is then used to calculate an output.

Multilayer Feed-Forward Neural Networks

- Multi-layer feed forward neural network with several layers of neurons is the most common form of artificial neural network (NN)
- Solve various real world problems such as classification.
- Also known as a multilayer perceptron (MLP)
- An input layer, multiple intermediate layers and finally an output layer.
- The intermediate layers don't have input or output to the external world, and are called hidden layers.
- The purpose of hidden layer is to increase functional adaptability of a network.
- As the complexity in the relationship between the input data and the desired output increases, the number of neurons in the hidden layer should also increase.

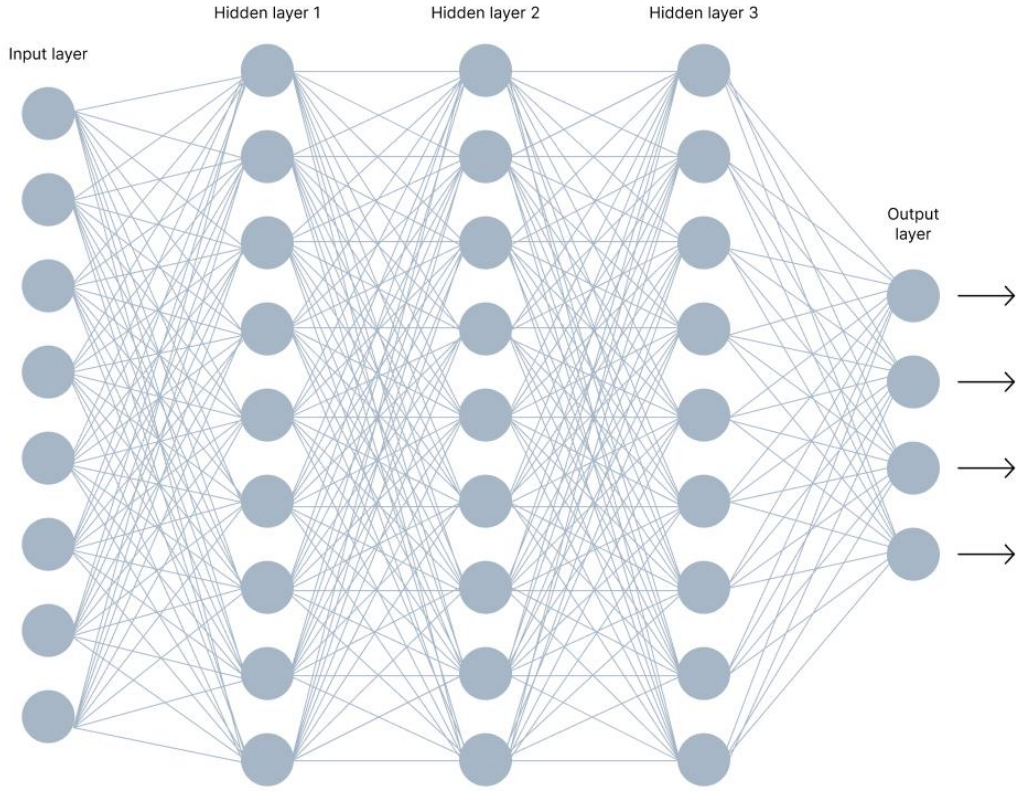
Multilayer Feed-Forward Neural Networks

- *Feed -forward Networks*
  - ***A connection is allowed from a node in layer* i *only to nodes in layer* i + *1.***
  - ***Most widely used architecture.***



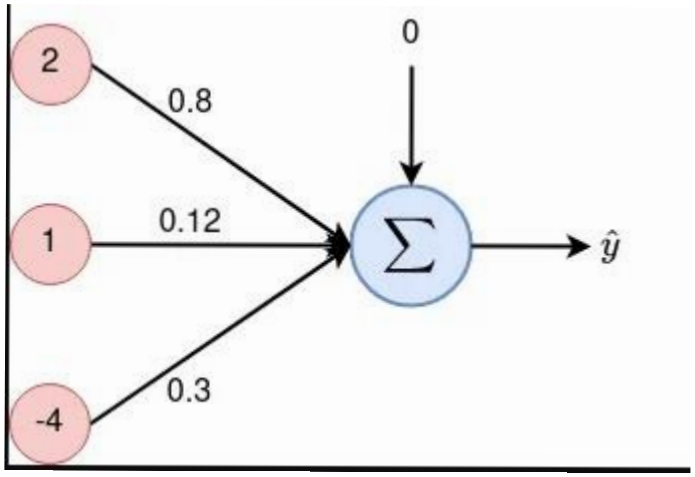
Conceptually, nodes at higher levels successively abstract features from preceding layers

Neural Networks Architecture



V7 Labs

A simple example



A simple example

$$x_1 \to x_1 w_1 = 2 \times 0.8 = 1.6$$
$$x_2 \to x_2 w_2 = 1 \times 0.12 = 0.12$$
$$x_3 \to x_3 w_3 = -4 \times 0.3 = -1.2$$

First operation: input values are weighted.

$$x \mapsto \left(\sum_{i=1}^{3} x_i w_i \right) + b = (.6 + 0.12 - 1.2) + 0 = 0.52$$

Second operation: sum weighted inputs and the bias

Sigmoid Activation Func

$$g(x) = \frac{1}{1 + e^{-x}} = \frac{1}{1 + e^{-0.52}} = 0.627$$

Third operation: apply sigmoid function.

Activation Function

Activation Function as Gate

- Mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.

What is Activation Function?

- The activation function decides whether a neuron should be activated or not
- By calculating the weighted sum and further adding bias to it
- Introduce non-linearity into the output of a neuron.

Why do we need Non-linear activation function?

- The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Why do Neural Networks Need an Activation Function?

- The purpose of an activation function is to add non-linearity to the neural network.

  Let's suppose we have a neural network working without the activation functions.

- Every neuron will only be performing a linear transformation
- Neural network becomes simpler, learning any complex task is impossible

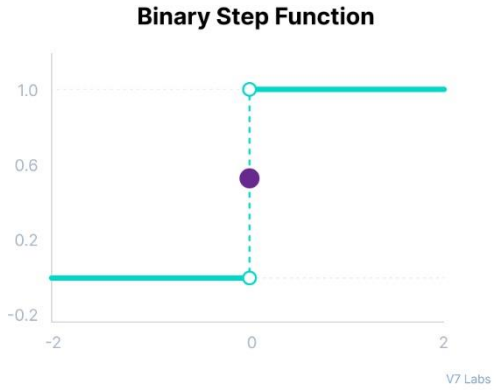
Types of Activation Functions

- 3 Types of Neural Networks Activation Functions

1. Binary Step Function
2. Linear Activation Function
3. Non-Linear Activation Functions

Binary Step Function

- Depends on a threshold value that decides whether a neuron should be activated or not.
- If the input is greater than threshold value, then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.



Binary step

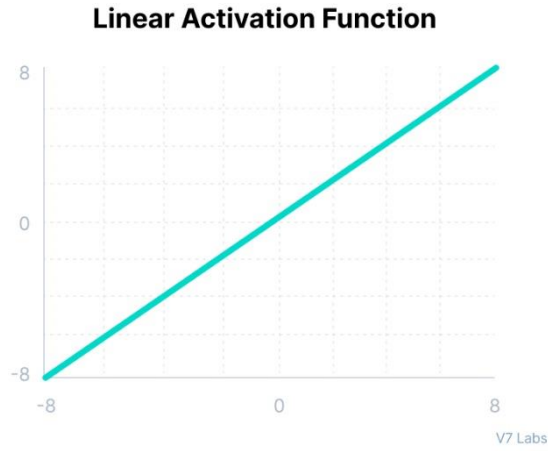
$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Limitations of binary step function

- It cannot provide multi-value outputs—for example, it cannot be used for multi-class classification problems.
- The gradient of the step function is zero, which causes a hindrance in the backpropagation process.

Linear Activation Function

- Also known as "no activation," or "identity function" (multiplied x1.0)
- Activation is proportional to the input.
- It simply spits out the value it was given.



Linear

$$f(x) = x$$

Limitations of Linear Activation

A linear activation function has two major problems :

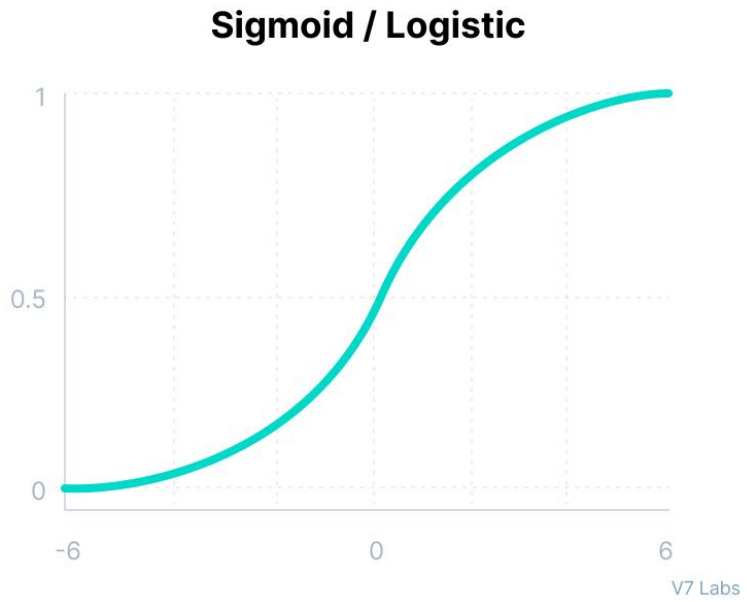
- It’s not possible to use backpropagation as the derivative of the function is a constant and has no relation to the input x
- All layers of the neural network will collapse into one if a linear activation function is used

Non-Linear Activation Functions

- Non-linear activation functions solve the following limitations of linear activation functions:
  - They allow backpropagation because now the derivative function would be related to the input
  - They allow the stacking of multiple layers of neurons as the output would now be a non-linear combination of input passed through multiple layers.

Sigmoid / Logistic Activation Function

- This function takes any real value as input and outputs values in the range of 0 to 1.



Sigmoid / Logistic

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid / Logistic Activation Function

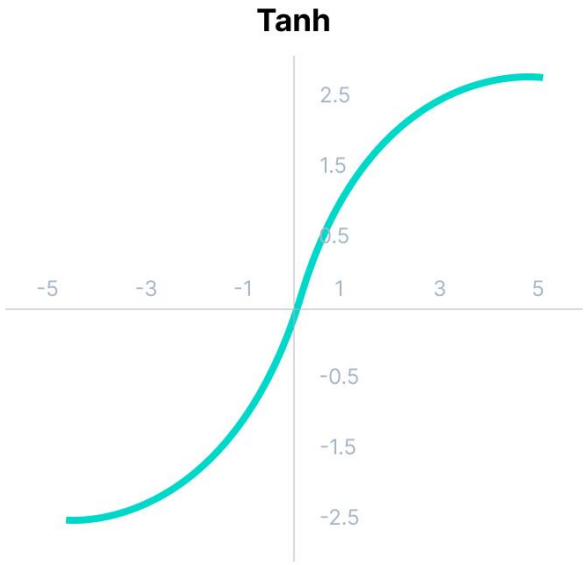
Why sigmoid/logistic activation function is one of the most widely used functions?

- It is commonly used for models where we have to predict the probability as an output.

- The function is differentiable and provides a smooth gradient

Tanh Function (Hyperbolic Tangent)

- Similar to the sigmoid/logistic activation function
- Difference in output range of -1 to 1

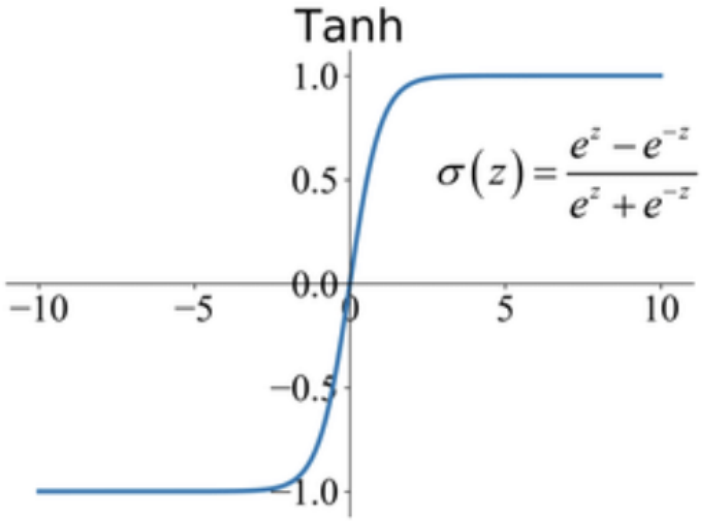


$$Tanh$$

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

Advantages

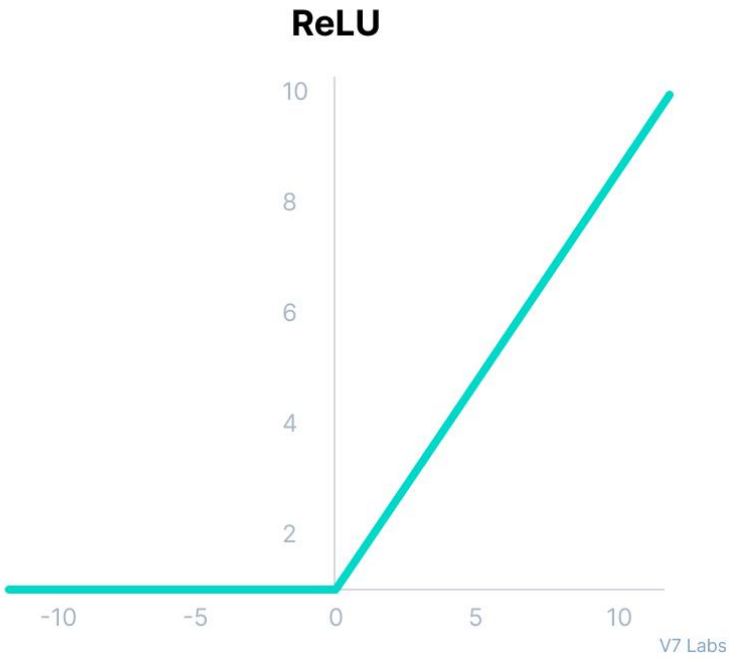
- The output of the tanh activation function is Zero centered; hence we can easily map the output values as strongly negative, neutral, or strongly positive.



ReLU Function

- ReLU stands for Rectified Linear Unit
- Gives an impression of a linear function
- The main catch here is that the ReLU function does not activate all the neurons at the same time.
- The neurons will only be deactivated if the output of the linear transformation is less than 0

ReLU Function



$$ReLU$$

$$f(x) = max\,(0, x)$$

ReLU Function

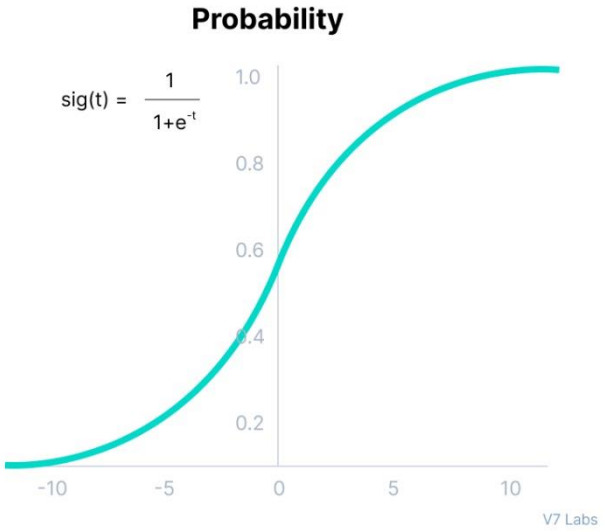
The advantages of using ReLU as an activation function are as follows:

- Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient
- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear, non-saturating property

Softmax Function

Lets see the building block—the sigmoid/logistic activation function

- Let's suppose we have five output values of 0.8, 0.9, 0.7, 0.8, and 0.6, respectively. How can we move forward with it?
- The answer is: We can't.
- The above values don't make sense as the sum of all the classes/output probabilities should be equal to 1.



Softmax Function

- Softmax function is described as a combination of multiple sigmoids.
- It calculates the relative probabilities
- The SoftMax function returns the probability of each class.
- It is most commonly used as an activation function for the last layer of the neural network in the case of multi-class classification

Softmax Function

Softmax

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

V7 Labs

Simple example

- Assume that you have three classes, meaning that there would be three neurons in the output layer. Now, suppose that your output from the neurons is [1.8, 0.9, 0.68].
- Applying the softmax function over these values to give a probabilistic view will result in the following outcome: [0.58, 0.23, 0.19].
- The function returns 1 for the largest probability index while it returns 0 for the other two array indexes. Here, giving full weight to index 0 and no weight to index 1 and index 2. So the output would be the class corresponding to the 1st neuron(index 0) out of three.

How to choose the right Activation Function?

- You need to match your activation function for your output layer based on the type of prediction problem that you are solving
- ReLU activation function should only be used in the hidden layers.
- Sigmoid/Logistic and Tanh functions should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).

Thank You