ANN Introduction

Zahoor Tanoli (PhD)

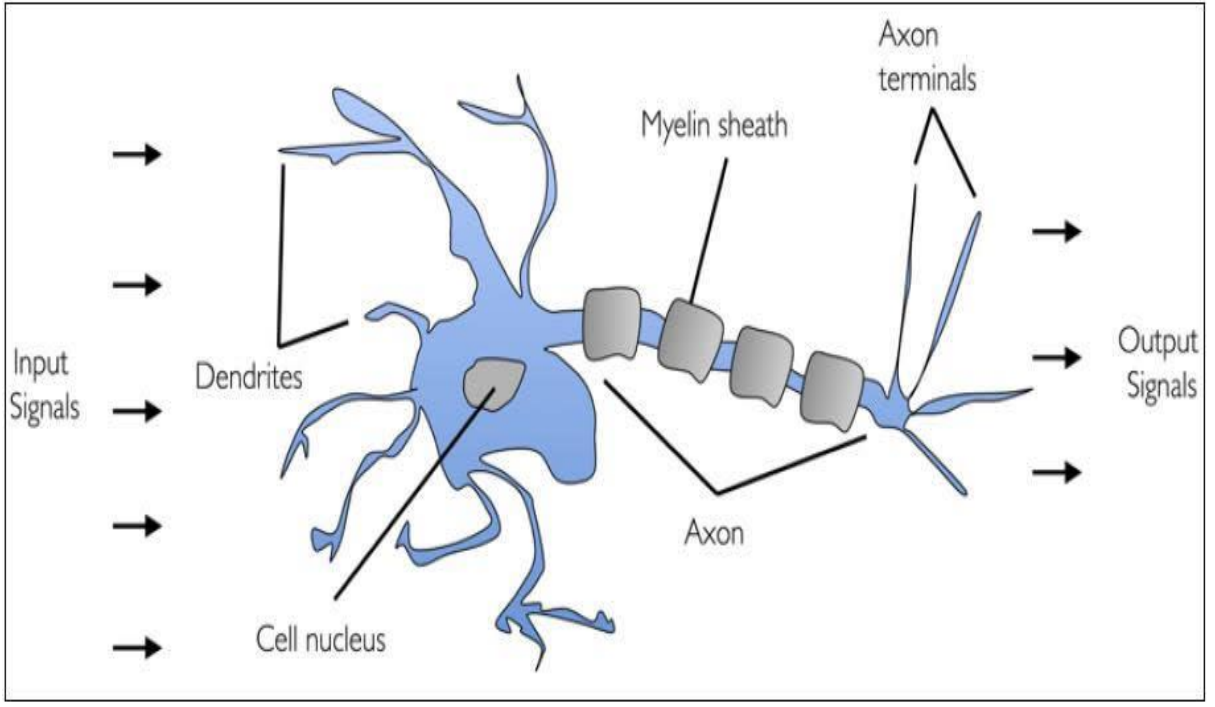
COMSATS Attock Campus

Biological Neuron

- Neurons are interconnected nerve cells in the human brain that are involved in:
  - Processing and transmitting chemical and electrical signals
- A human brain has billions of neurons
- Dendrites are branches that receive information from other neurons

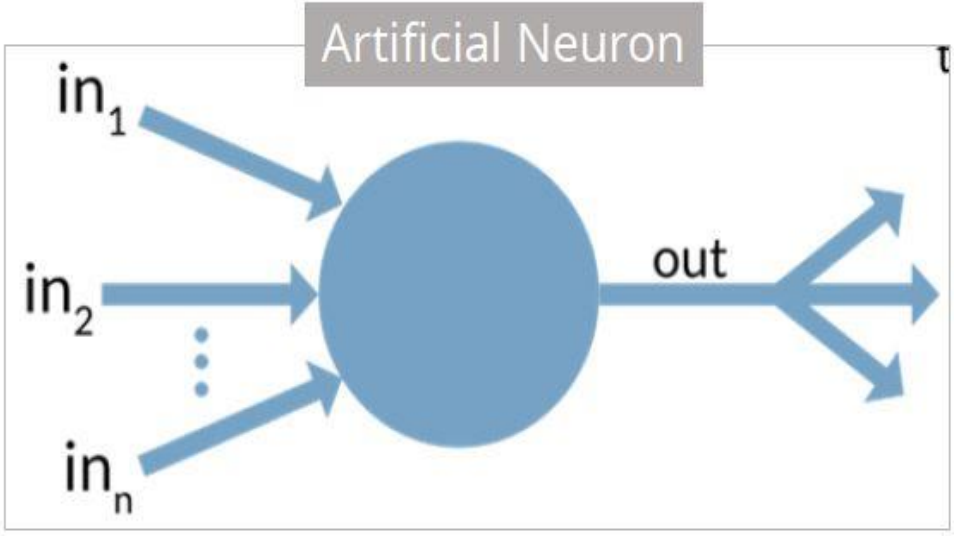
Biological Neurons

- 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



Artificial Neuron

- An artificial neuron is a mathematical function based on a model of biological neurons, where each neuron takes:
  - Inputs
  - Weighs them separately
  - Sums them up and
  - Passes this sum through a nonlinear function to produce output



Biological Neuron vs. Artificial Neuron

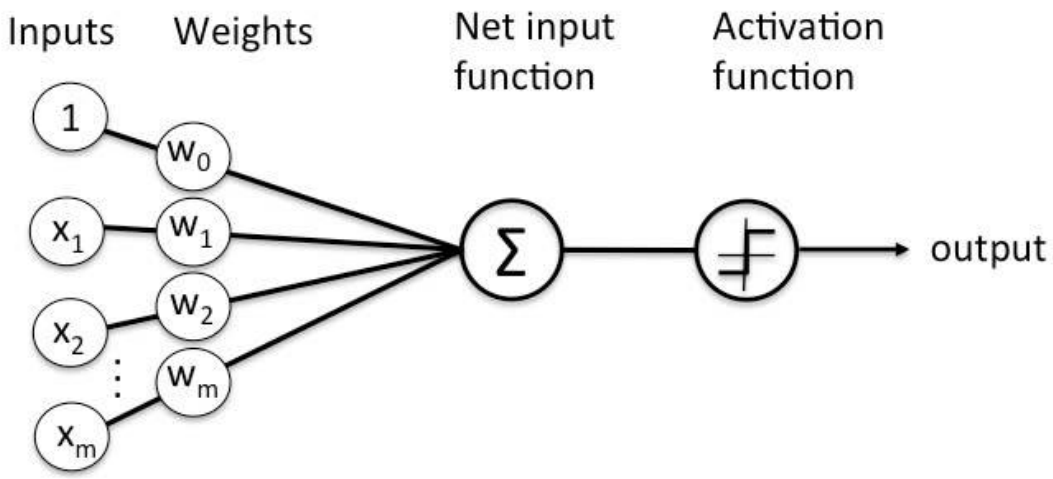
Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

Artificial Neuron at a Glance

- 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

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
- Perceptron was introduced by Frank Rosenblatt in 1957
- 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

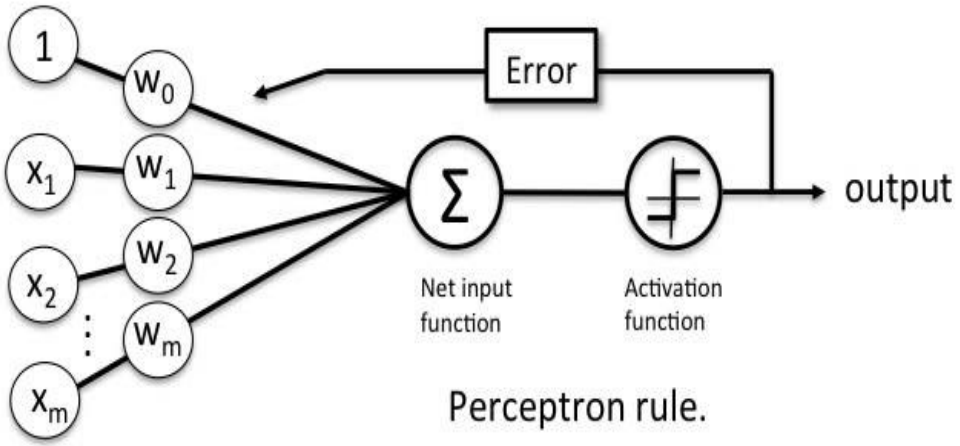


Types of Perceptron

- There are two types of Perceptrons:
  - Single layer
  - 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

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 output
- In the context of supervised learning and classification, this can then be used to predict the class of a sample



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

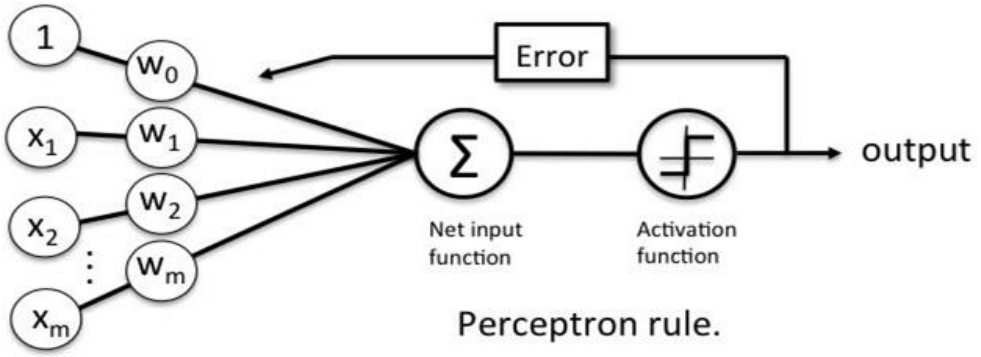
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

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

  - “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
- “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

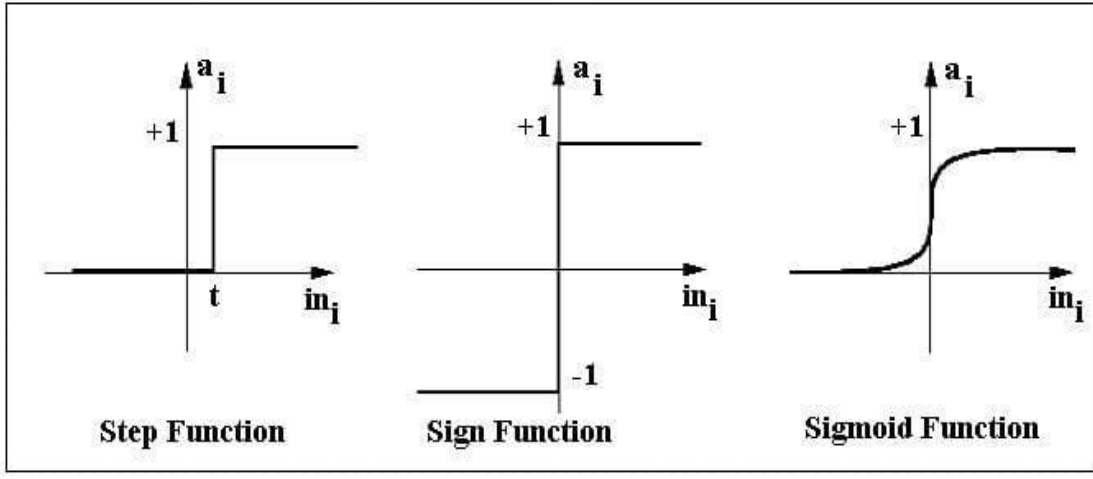
- Perceptron accepts inputs
- Adjust weight values
- Applies the transformation function to output the final result
- It has only two values: Yes and No or True and False
- The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up



Perceptron rule.

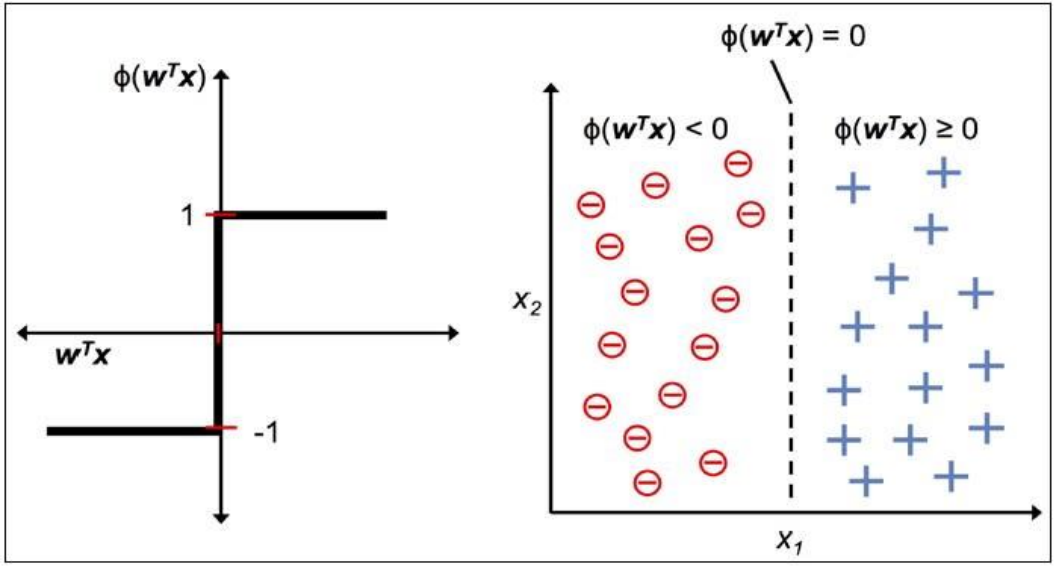
Activation Functions of Perceptron

- 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
- 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.



Error and Output of Perceptron

- 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 Summary

- Perceptron is an algorithm for Supervised Learning
- 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.

Running Example

- Going to work on AND Gate problem
- The gate returns if and only if both inputs are true
- We are going to set weights randomly
- Let's say that w1 = 0.9 and w2 = 0.9

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

Round 1

- We apply 1st instance to the perceptron. x1 = 0 and x2 = 0
- $\Sigma = x1 * w1 + x2 * w2 = 0 * 0.9 + 0 * 0.9 = 0$
- Suppose that activation threshold would be 0.5
- Sum unit was 0 for the 1st instance. So, activation unit would return 0 because it is less than 0.5.
- Similarly, its output should be 0 as well. We will not update weights because there is no error in this case
- Let's focus on the 2nd instance. x1 = 0 and x2 = 1.
- Sum unit: $\Sigma = x1 * w1 + x2 * w2 = 0 * 0.9 + 1 * 0.9 = 0.9$

Round-1 (Error)

- Activation unit will return 1 because sum unit is greater than 0.5 but Output of this instance should be 0
- Not predicted correctly, so will update weights based on the error
- ε = actual – prediction = 0 – 1 = -1
- Add error times learning rate value to the weights and let learning rate would be 0.5
- $w1 = w1 + \alpha * \varepsilon = 0.9 + 0.5 * (-1) = 0.9 – 0.5 = 0.4$
- $w2 = w2 + \alpha * \varepsilon = 0.9 + 0.5 * (-1) = 0.9 – 0.5 = 0.4$

Round-1 (3rd and 4th instance)

- The 3rd instance. x1 = 1 and x2 = 0
- Sum unit: Σ = x1 * w1 + x2 * w2 = 1 * 0.4 + 0 * 0.4 = 0.4
- Activation unit will return 0 this time because output of the sum unit is 0.4 and it is less than 0.5 (No update)
- 4th instance is x1 = 1 and x2 = 1
- Sum unit: Σ = x1 * w1 + x2 * w2 = 1 * 0.4 + 1 * 0.4 = 0.8
- Activation unit will return 1 because output of the sum unit is 0.8
- Actual value is 1 is predicted correctly. We will not update

Round-2 (1st instance)

- In previous round, 1st instance was classified correctly. Let's apply feed forward for the new weight values.
- Remember that 1st instance is x1 = 0 and x2 = 0
- Sum unit: $\Sigma = x1 * w1 + x2 * w2 = 0 * 0.4 + 0 * 0.4 = 0$
- Activation unit will return 0 because sum unit is 0 and it is less than the threshold value 0.5. The output of the 1st instance should be 0 as well. This means that the instance is classified correctly.
- No update in weights

Round-2 (2nd Instance)

- Feed forward for the 2nd instance. x1 = 0 and x2 = 1
- Sum unit: $\Sigma = x1 * w1 + x2 * w2 = 0 * 0.4 + 1 * 0.4 = 0.4$
- Activation unit will return 0 because sum unit is less than the threshold 0.5.
- Output will be 0 and means that it is classified correctly and we will not update weights
- We've applied feed forward calculation for 3rd and 4th instances already for the current weight values in the previous round-1. They were classified correctly

Learning Term

- Updating weights means learning in the perceptron
- We set weights to 0.9 initially but it causes some errors
- Then, we update the weight values to 0.4. In this way, we can predict all instances correctly
- Luckily, the best weights in 2 round