Artificial Neural Networks
Unit-5
Dr H C Vijayalakshmi

References

1. https://www.bing.com/search?pglt=41&q=mcculloch+pitts+model&cvid=df84765ab37b4038acd22c530fafc67c&aqs=edge.5.69i57j0l8.12113j0j1&FORM=ANNTA1&PC=ACTS&ntref=1#

2. https://www.bing.com/search?pglt=41&q=mcculloch+pitts+model&cvid=df84765ab37b4038acd22c530fafc67c&aqs=edge.5.69i57j0l8.12113j0j1&FORM=ANNTA1&PC=ACTS&ntref=1#

Artificial Neural Network

It is an information processing system, which is constructed and implemented to model the human brain.

The aim of NN is to mimic the human ability to adapt to changing circumstances and current environments.

In other words NN is a machine learning approach inspired by the way in which the brain performs a particular learning task:

- Knowledge about the learning task is given in the form of examples.
- Inter neuron connection strengths (weights) are used to store the acquired information (the training examples).
- During the learning process the weights are modified in order to model the particular learning task correctly on the training examples
- Neural Network models: **perceptron, feed-forward, radial basis function, support vector machine..**

In fact, the human brain is a highly complex structure viewed as a massive, highly interconnected network of simple processing elements called neurons. On an average human brain has around 10^{11} neurons.

Artificial neural networks (ANNs) or simply we refer it as neural network (NNs), which are simplified models (i.e. imitations) of the biological nervous system, and obviously, therefore, have been motivated by the kind of computing performed by the human brain.

The behavior of a biological neural network can be captured by a simple model called artificial neural network.

Structure of a Neuron

Dendrite : A bush of very thin fibre. Axon : A long cylindrical fibre.

Soma : It is also called a cell body, and just like as a nucleus of cell.

Synapse : It is a junction where axon makes contact with the dendrites of neighboring dendrites and receive messages at the dendrites

Message is sent quickly down the axon using electrical impulses

What happens when the signal reaches the end of the axon?

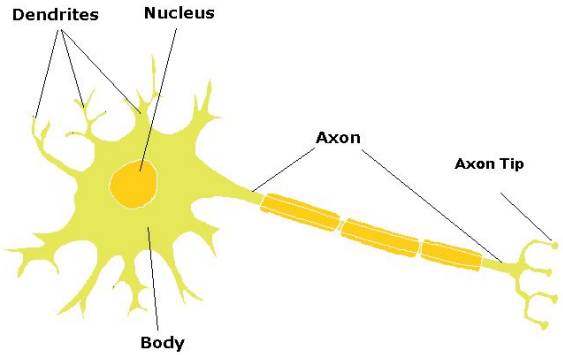


Image taken from img460.imageshack.us

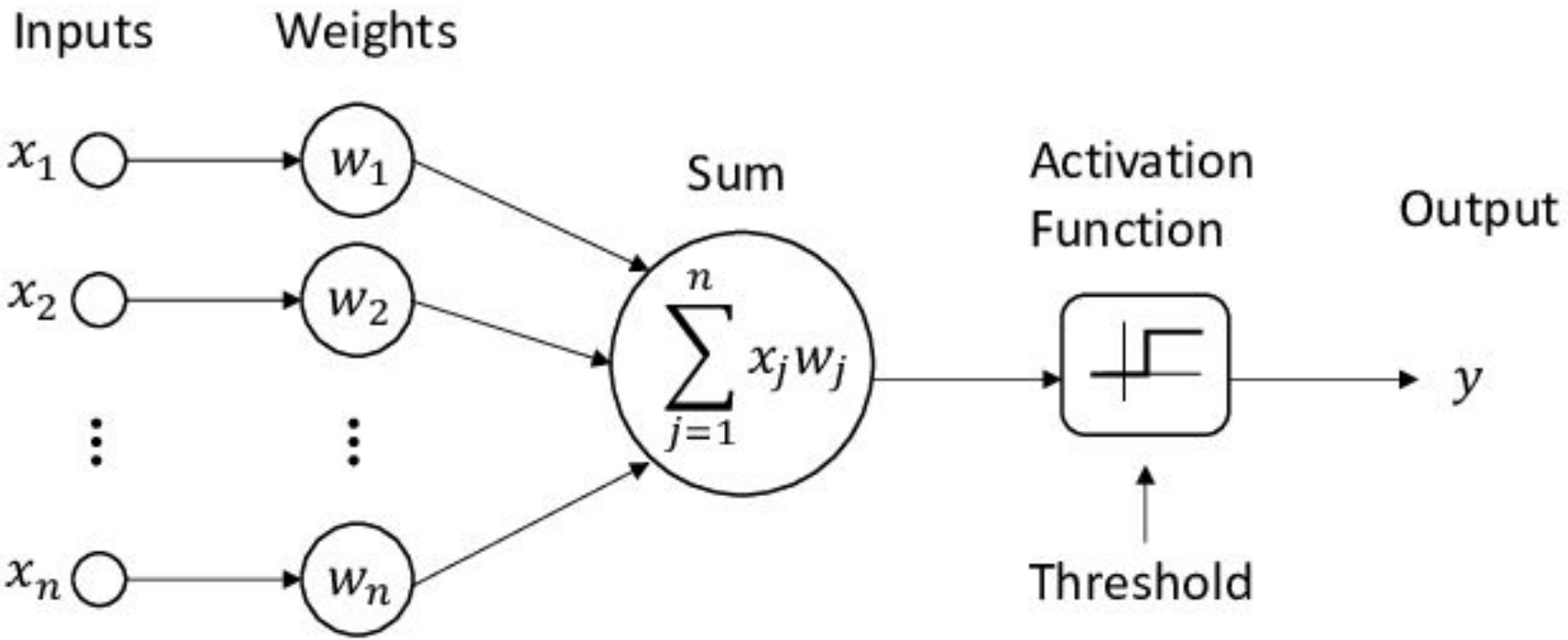
A neuron is a part of an interconnected network of nervous system and serves the following.

Compute input signals Transportation of signals (at a very high speed) Storage of information Perception, automatic training and learning

We also can see the analogy between the biological neuron and artificial neuron.

Truly, every component of the model (i.e. artificial neuron) bears a direct analogy to that of a biological neuron.

It is this model which forms the basis of neural network (i.e. artificial neural network). The representation of an ANN is as shown in next slide.



Here,

x1, x2, · · · , xn are the n inputs to the artificial neuron.

w1, w2, · · · , wn are weights attached to the input links.

Note that, a biological neuron receives all inputs through the dendrites, sums them and produces an output if the sum is greater than a threshold value.

The input signals are passed on to the cell body through the synapse, which may accelerate or retard an arriving signal.

It is this acceleration or retardation of the input signals that is modeled by the weights.

An effective synapse, which transmits a stronger signal will have a correspondingly larger weights while a weak synapse will have smaller weights.

Thus, weights here are multiplicative factors of the inputs to account for the strength of the synapse.

$$u = \sum_{j=1}^{n} w_j x_j$$

Hence, the total input say I received by the soma of the artificial neuron is

$$I = w1x1 + w2x2 + \cdots + wnxn$$

To generate the final output y, the sum is passed to a filter φ called transfer function, which releases the output. That is, $y = \varphi(I)$

Differences between ANN and BNN :

Biological Neural Networks (BNNs) and Artificial Neural Networks (ANNs) are both composed of similar basic components, but there are some differences between them.

Neurons: In both BNNs and ANNs, neurons are the basic building blocks that process and transmit information. However, BNN neurons are more complex and diverse than ANNs. In BNNs, neurons have multiple **dendrites** that receive input from multiple sources, and the **axons** transmit signals to other neurons, while in ANNs, neurons are simplified and usually only have a single output.

Synapses: In both BNNs and ANNs, synapses are the points of connection between neurons, where information is transmitted. However, in ANNs, the connections between neurons are usually fixed, and the strength of the connections is determined by a set of weights, while in BNNs, the connections between neurons are more flexible, and the strength of the connections can be modified by a variety of factors, including learning and experience.

Neural Pathways: In both BNNs and ANNs, neural pathways are the connections between neurons that allow information to be transmitted throughout the network. However, in BNNs, neural pathways are highly complex and diverse, and the connections between neurons can be modified by experience and learning. In ANNs, neural pathways are usually simpler and predetermined by the architecture of the network.

Advantages of NN

1. ANNs exhibits mapping capabilities, that is, they can map input patterns to their associated output pattern.
2. The ANNs learn by examples. In other words, they can identify new objects previously untrained.
3. The ANNs possess the capability to generalize. This is the power to apply in application where exact mathematical model to problem are not possible.The ANNs are robust system and fault tolerant. They can therefore, recall full patterns from incomplete, partial or noisy patterns.
4. The ANNS can process information in parallel, at high speed and in a distributed manner. Thus a massively parallel distributed processing system made up of highly interconnected (artificial) neural computing elements having ability to learn and acquire knowledge is possible.

Applications of NN

1. Voice recognition
2. Weather Prediction
3. Strategies for Games, businessand war
4. Fraud Detection
5. Data mining
6. Medical Diagnosis, Photo and fingerprint recognition

Neuron :The neuron is the basic information processing unit of a NN. It consists of:

1 A set of synapses or connecting links, each link characterized by a weight:

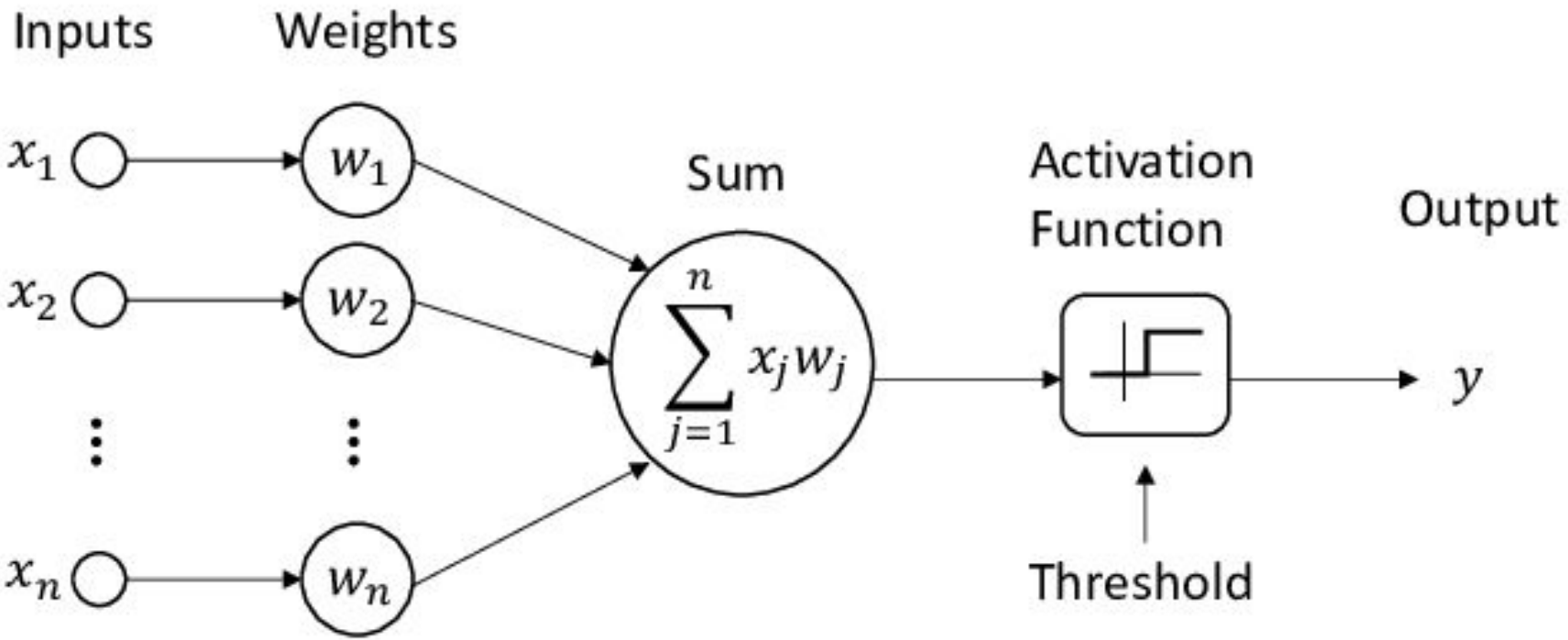
$W_1, W_2, \ldots, W_m$

2 An adder function (linear combiner) which computes the weighted sum of the inputs:

$$\mathbf{u} = \sum_{j=1}^{\mathrm{m}} \mathbf{w}_{\mathrm{j}} \mathbf{x}_{\mathrm{j}}$$

3 Activation function (squashing function) φ for limiting the amplitude of the output of the neuron.

$$\mathrm{y} = \varphi(\mathrm{u} + b)$$



Here,

x1, x2, · · · , xn are the n inputs to the artificial neuron.

w1, w2, · · · , wn are weights attached to the input links.

Each neuron consists of three major components:

A set of **'i' synapses having weight wi.** A signal xi forms the input to the i-th synapse having weight wi. The value of any weight may be positive or negative. A positive weight has an extraordinary effect, while a negative weight has an inhibitory effect on the output of the summation junction.

A **summation junction** for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction can be expressed as follows:

A threshold **activation function** (or simply **the activation function,** also known as **squashing function**) results in an output signal only when an input signal exceeding a specific threshold value comes as an input. It is similar in behaviour to the biological neuron which transmits the signal only when the total input signal meets the firing threshold.

What is an activation function and why use them?

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

Why do we need Non-linear activation function?

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Types of Activation Functions –

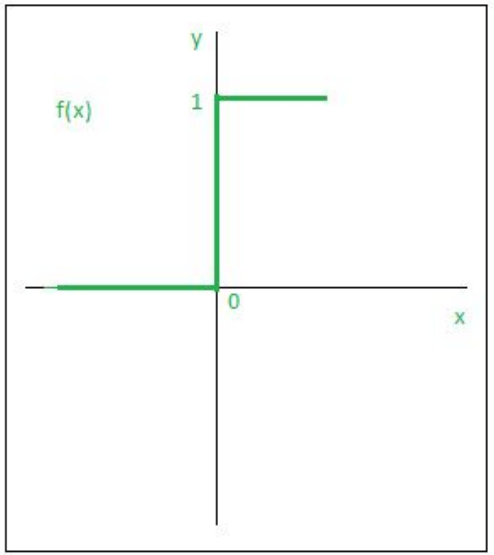
Several different types of activation functions are used in Deep Learning. Some of them are explained below:

Step Function:

Step Function is one of the simplest kind of activation functions. In this, we consider a threshold value and if the value of net input say **y** is greater than the threshold then the neuron is activated.

Mathematically,

1. f(x) = 1 if x>= 0
2. f(x) = 0 if x < 0



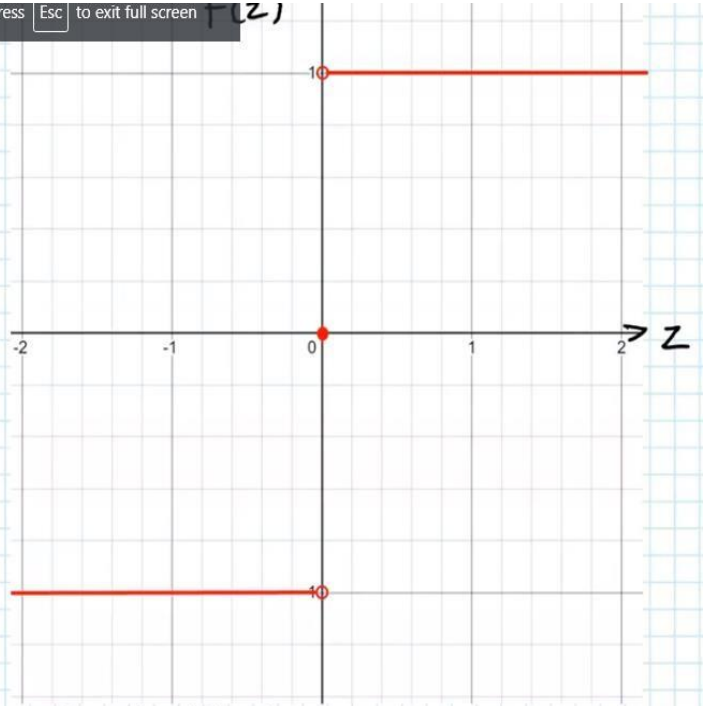
Sign function

2. Signum (sgn or sign) function

$$f(z) = \begin{cases} -1 & z < 0 \\ 1 & z > 0 \end{cases}$$

Range = Possible Outputs

$\{-1, 1\}$



Used in: Hidden Layer, Output Layer for Classification

Sigmoid Function:

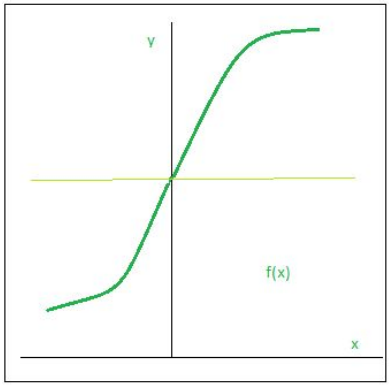
Sigmoid function is a widely used activation function. It is defined as: $f(x) = 1/(1+e^{-x})$. This is a smooth function and is continuously differentiable.

The biggest advantage that it has over step and linear function is that it is non-linear. This is an incredibly cool feature of the sigmoid function.

This essentially means that when there are multiple neurons

having sigmoid function as their activation function –the output is non linear as well.

The function ranges from 0-1 having an S shape.



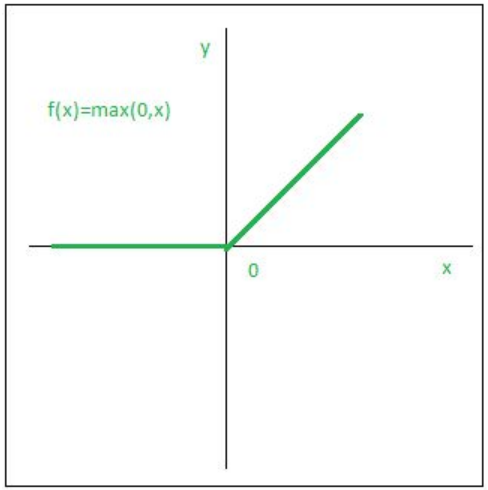
ReLU:

The ReLU function is the Rectified linear unit. It is the most widely used activation function. It is defined as: **f(x) = Max(0,x)**

The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time.

What does this mean ?

If you look at the ReLU function if the input is negative it will convert it to zero and the neuron does not get activated.

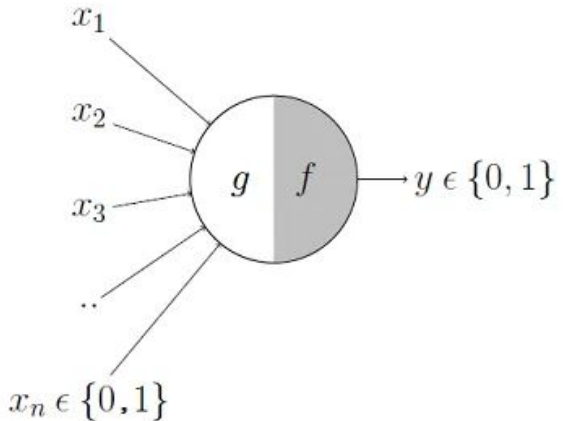


It is very well known that the most fundamental unit of neural networks is called an *artificial neuron*(Mc Culloch-Pitts) */perceptron* (Rosenblatt). But the very first step towards the *Neuron* was taken in 1943 by McCulloch and Pitts, by mimicking the functionality of a biological neuron.

It may be divided into 2 parts. The first part, ***g*** takes an input (ahem dendrite ahem), performs an aggregation and based on the aggregated value the second part, ***f*** makes a decision.

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

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



We can see that $g(x)$ is just doing a sum of the inputs — a simple aggregation. And ***theta*** here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the ***theta*** is 2 here. This is called the Thresholding Logic.

A single Mc Culloch pitts neuron can be used to represent boolean fns which are linearly separable.

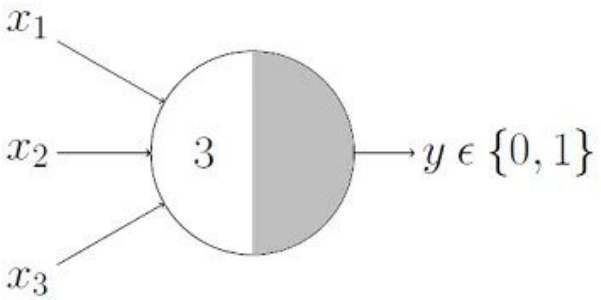
Linear Separability : There exists a line(plane) such that all inputs which produce 1 lie on one side of the (plane) line and all other inputs which produce 0 lie on the other side of the line(plane)

In 2 dimensional space it is a plane. In higher dimensional space it is a hyper plane.

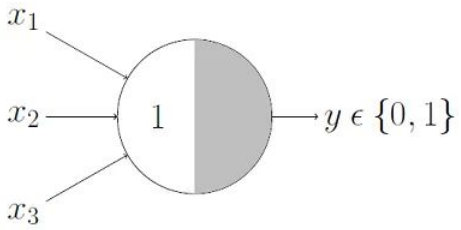
This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3*** if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

AND Gate

An AND function neuron would only fire when ALL the inputs are ON i.e., ***g*(x)** ≥ 3 here.



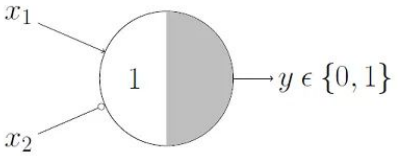
OR Gate An OR function neuron would only fire when ALL the inputs are ON i.e., $g(\mathbf{x}) \geq 1$ here.



A Function With An Inhibitory Input

Here, we have an inhibitory input i.e., ***x_2*** so whenever ***x_2*** is 1, the output will be 0. Keeping that in mind, we know that ***x_1 AND !x_2*** would output 1 only when ***x_1*** is 1 and ***x_2*** is 0 so it is obvious that the threshold parameter should be 1.

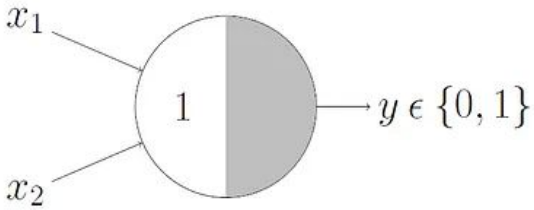
Lets verify that, the *g*(x) i.e., *x_1* + *x_2* would be ≥ 1 in only 3 cases:
Case 1: when *x_1* is 1 and *x_2* is 0
Case 2: when *x_1* is 1 and *x_2* is 1
Case 3: when *x_1* is 0 and *x_2* is 1



$x_1\ AND\ !x_2$*

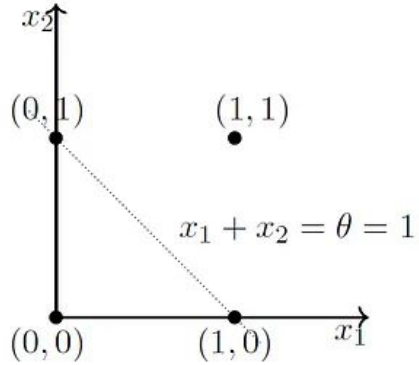
OR Function

OR function's thresholding parameter ***theta*** is 1, for obvious reasons. The inputs are obviously boolean, so only 4 combinations are possible — (0,0), (0,1), (1,0) and (1,1). Now plotting them on a 2D graph and making use of the OR function's aggregation equation i.e., ***x_1 + x_2 ≥ 1*** using which we can draw the decision boundary as shown in the graph below.

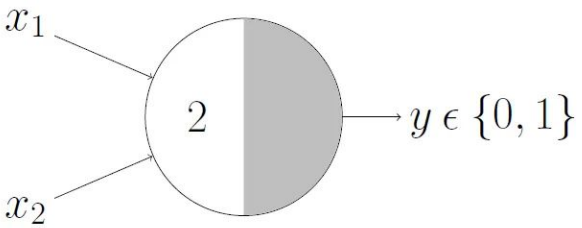


OR function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$$

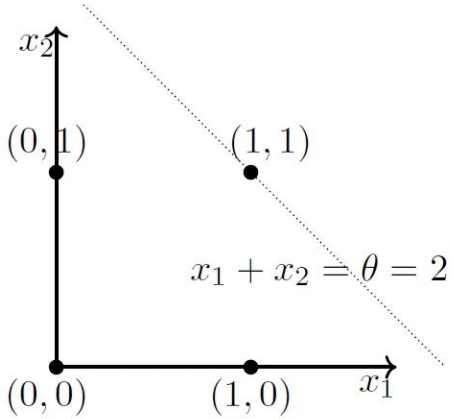


AND Gate In this case, the decision boundary equation is ***x_1 + x_2 =2***. Here, all the input points that lie ON or ABOVE, just (1,1), output 1 when passed through the AND function M-P neuron. It fits! The decision boundary works!

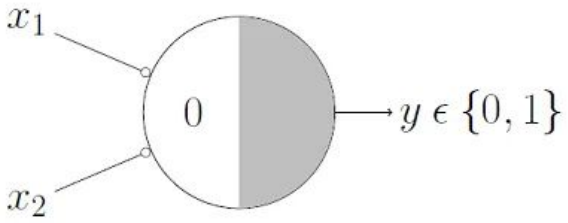


AND function

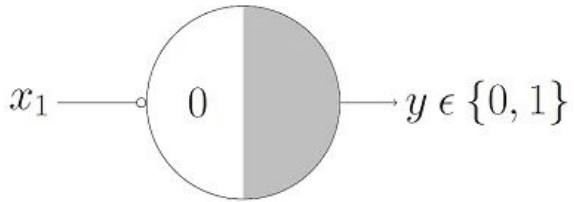
$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$



NOR Function : For a NOR neuron to fire, we want ALL the inputs to be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input.



NOT Function For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0.



Perceptrons- 1958- Frank RosenBlatt

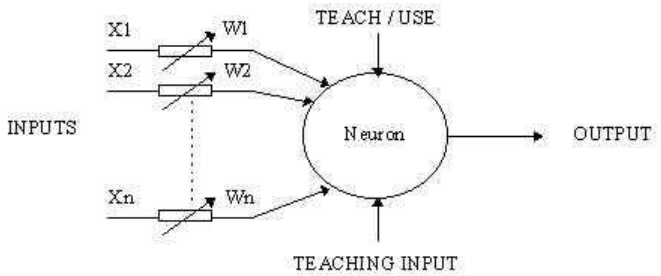
What about non-boolean(Say, real inputs)?

We need a way of learning these. What if some of these inputs have to be given more importance (weightage) than the others. And also which are not linearly separable.

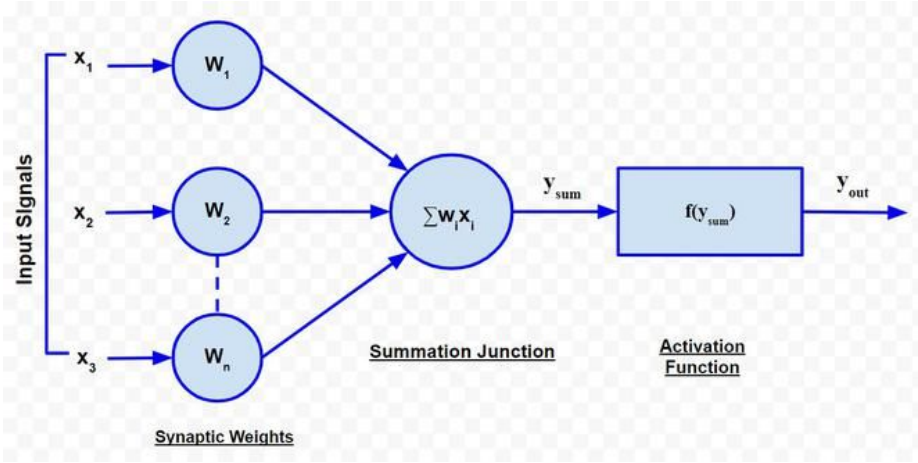
A perceptron is a more general computational mode.

Main differences: Introduction of Numerical wts for inputs and a mechanism for learning these wts.

$$u = \sum_{j=1}^{m} w_j x_j$$



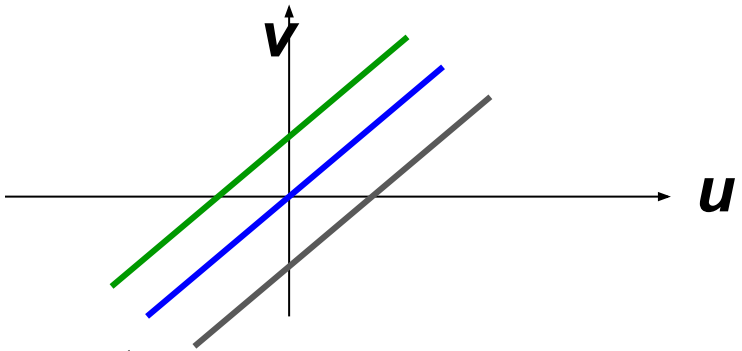
Mathematically: neuron fires if $X_1W_1 + X_2W_2 + X_3W_3 + \ldots >$ Theta



The values are boolean for Mc Cullochs neuron where as it can be real for perceptron.

Bias of a Neuron

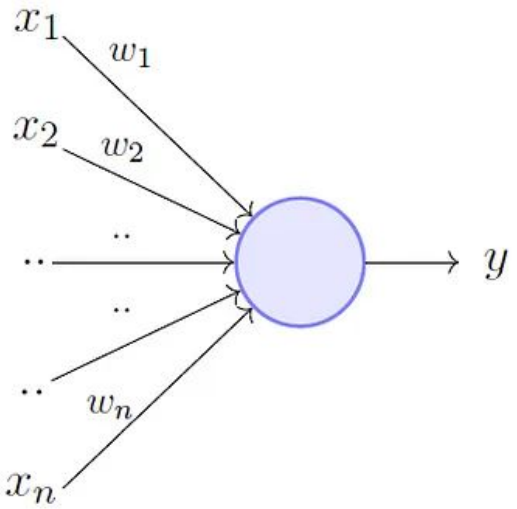
- Bias w_0 represents the prior (prejudice)
  - $v = u + w_0$
- v is the **induced field** of the neuron.



$$u = \sum_{j=1}^{m} w_j x_j$$

The wts $w_1\ w_2\ w_3 \ldots\ldots\ldots\ w_n$ ***and bias*** w_0 ***depends on the data***

PERCEPTRON



$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

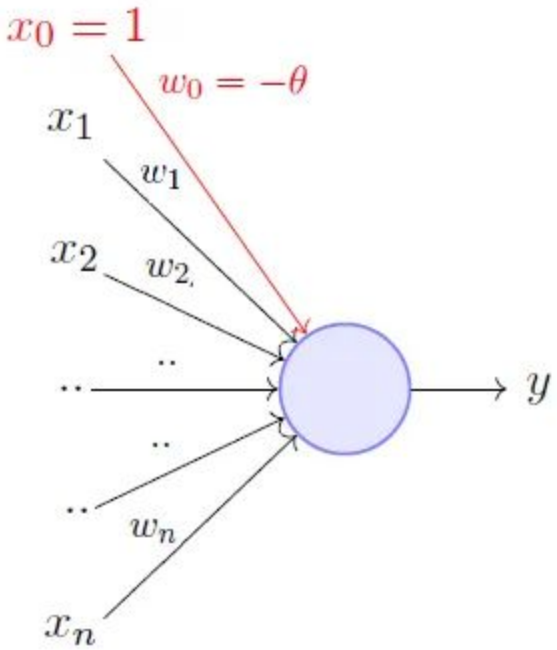
Rewriting the above,

$$y = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$

$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

The perceptron model is a more general computational model than McCulloch-Pitts neuron. It takes an input, aggregates it (weighted sum) and returns 1 only if the aggregated sum is more than some threshold else returns 0. Rewriting the threshold as shown above and making it a constant input with a variable weight, we would end up with something like in the next slide:

A single perceptron can only be used to implement **linearly separable** functions. It takes both real and boolean inputs and associates a set of **weights** to them, along with a **bias**



A more accepted convention,

$$y = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$

$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

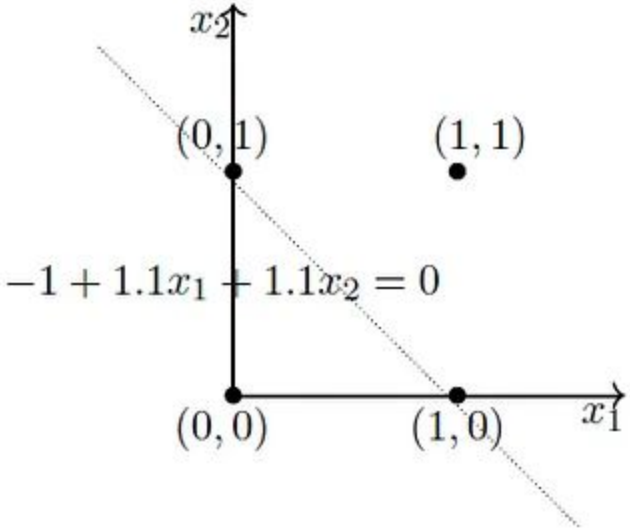
OR Function Using A Perceptron

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

One possible solution is

$w_0 = -1,\ w_1 = 1.1,\ w_2 = 1.1$



Perceptron Learning Algorithm

Our goal is to find the **w** vector that can perfectly classify positive inputs and negative inputs in our data.

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

Limitations of Perceptrons: (i) The output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function.

(ii) Perceptrons can only classify linearly separable sets of vectors. If a straight line or a plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable. If the vectors are not linearly separable, learning will never reach a point where all vectors are classified properly

The Boolean function XOR is not linearly separable (Its positive and negative instances cannot be separated by a line or hyperplane). Hence a single layer perceptron can never compute the XOR function. This is a big drawback that once resulted in the stagnation of the field of neural

- Perceptron works only with linearly separable classes.
- Perceptron can only classify linearly separable sets of vectors.
- The output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function.
- Perceptron doesn’t have the ability to learn a simple logical function like ‘XOR’.