An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process.

Units of Neural Network:

Nodes(units):

Nodes represent a cell of neural network.

Links:

Links are directed arrows that show propagation of information from one node to another node.

Activation:

Activations are inputs to or outputs from a unit.

Weight:

Each link has weight associated with it which determines strength and sign of the connection.

Activation function:

A function which is used to derive output activation from the input activations to a given node is called activation function.

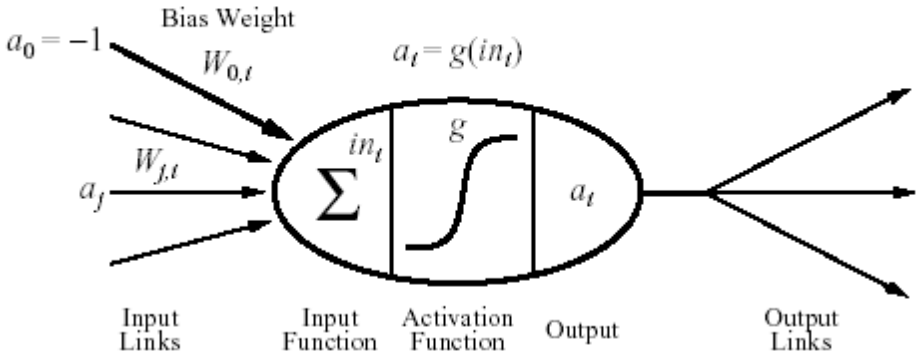
Bias Weight:

Bias weight is used to set the threshold for a unit. Unit is activated when the weighted sum of real inputs exceeds the bias weight.

Simple Model of Neural Network

A simple mathematical model of neuron is devised by McCulloch and Pit is given in the figure given below:

$$a_i \leftarrow g(in_i) = g\left(\sum_j W_{j,i} a_j\right)$$

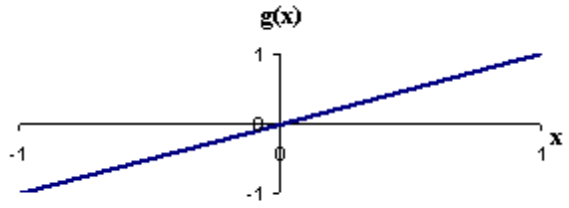


It fires when a linear combination of its inputs exceeds some threshold.

Activation function typically falls into one of three categories:

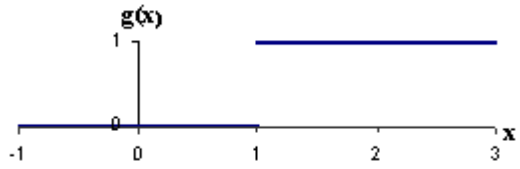
- Linear
- Threshold (*Heaviside function)*
- Sigmoid
- Sign

For **linear activation functions**, the output activity is proportional to the total weighted output.
g(x) = k x + c, where k and x are constant



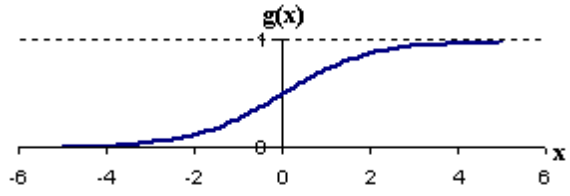
For **threshold activation functions**, the output are set at one of two levels, depending on whether the total input is greater than or less than some threshold value.
g(x) = 1 if x>= k

= 0 if x < k



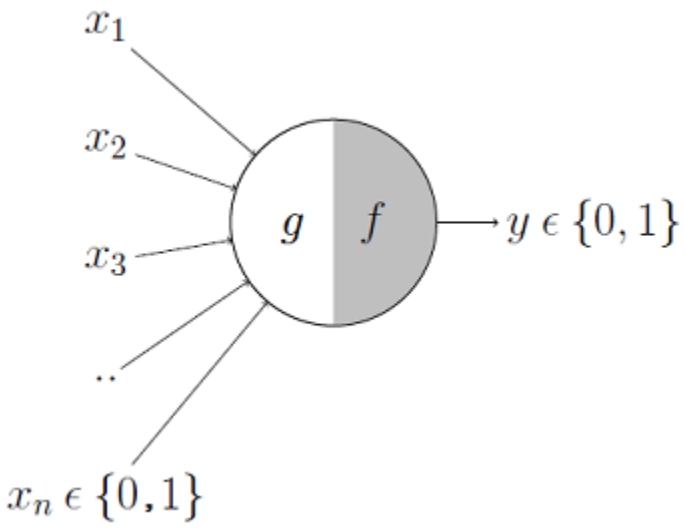
For **sigmoid activation functions**, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units. It has the advantage of differentiable.

g(x) = 1/ (1 + e-x)



McCulloch Pits neuron model

The first computational model of a neuron was proposed by Warren McCulloch (neuroscientist) and Walter Pitts (logician) in 1943.



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.

Lets suppose that I want to predict my own decision, whether to watch a random football game or not on TV. The inputs are all boolean i.e., {0,1} and my output variable is also boolean {0: Will watch it, 1: Won't watch it}.

- So, ***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* (Can't watch it when I'm running errands. Can I?)
- ***x_4*** could be *isManUnitedPlaying* (I am a big Man United fan. GGMU!) and so on.

These inputs can either be *excitatory* or *inhibitory*. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if ***x_3*** is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so ***x_3*** is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

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

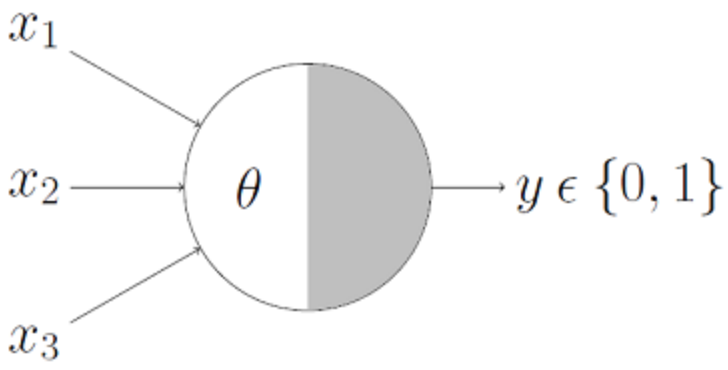
$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

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.

Boolean Functions Using M-P Neuron

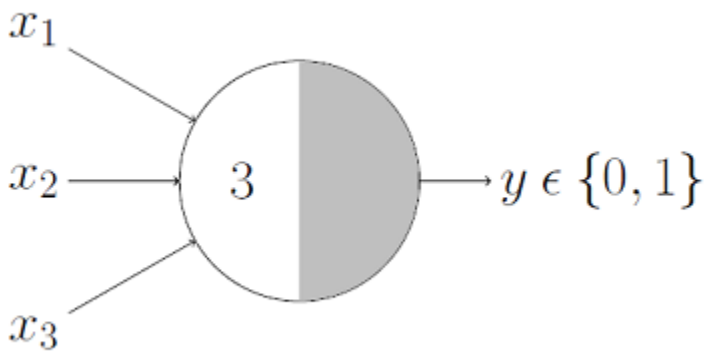
So far we have seen how the M-P neuron works. Now lets look at how this very neuron can be used to represent a few boolean functions. Mind you that our inputs are all boolean and the output is also boolean so essentially, the neuron is just trying to learn a boolean function. A lot of boolean decision problems can be cast into this, based on appropriate input variables— like whether to continue reading this post, whether to watch Friends after reading this post etc. can be represented by the M-P neuron.

M-P Neuron: A Concise Representation



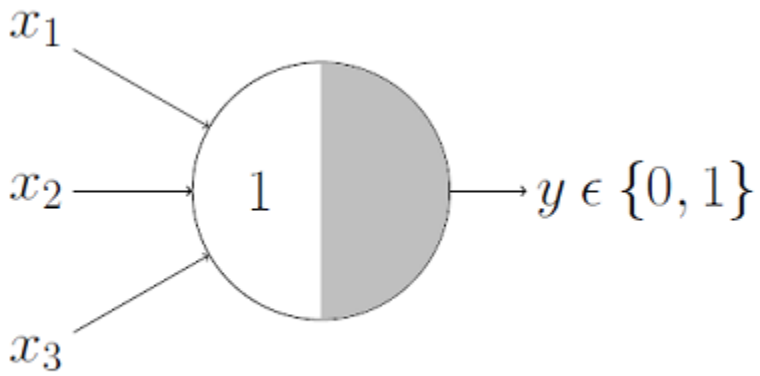
This representation just denotes that, for the boolean inputs ***x_1***, ***x_2*** and ***x_3***if the ***g*(x)** i.e., **sum ≥ theta**, the neuron will fire otherwise, it won't.

AND Function



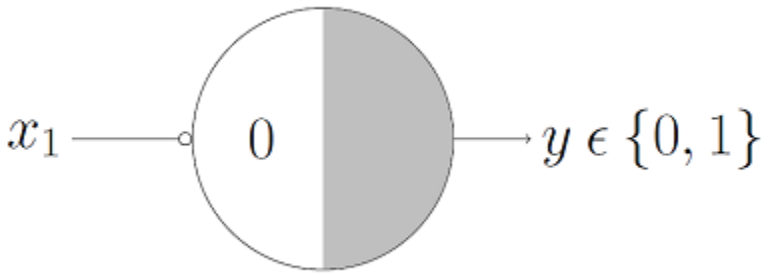
An AND function neuron would only fire when ALL the inputs are ON i.e., ***g*(x)** ≥ 3 here.

OR Function



I believe this is self explanatory as we know that an OR function neuron would fire if ANY of the inputs is ON i.e., ***g*(x)** ≥ 1 here.

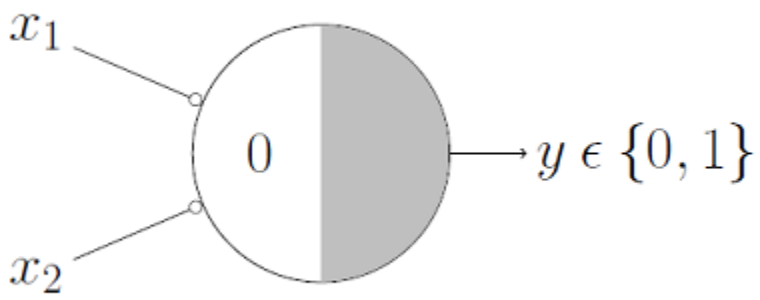
NOT Function



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. It works!

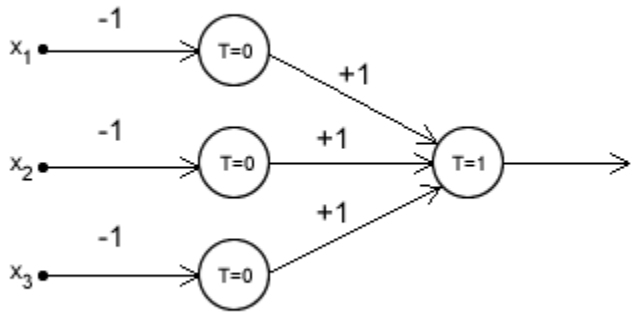
Can any boolean function be represented using the M-P neuron? Before you answer that, lets understand what M-P neuron is doing geometrically.

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.

NAND Gate



This figure shows how to create a 3-input NAND gate with these neurons.

A NAND gate gives a zero only when all inputs are 1. This neuron needs 4 neurons. The output of the first three is the input for the fourth neuron.

The McCulloch-Pitts model is no longer used. These NOR and NAND gates already have extremely efficient circuits. So it's pointless to redo the same thing, with less efficient models. The point is to use the "interconnections" and the advantages it has.

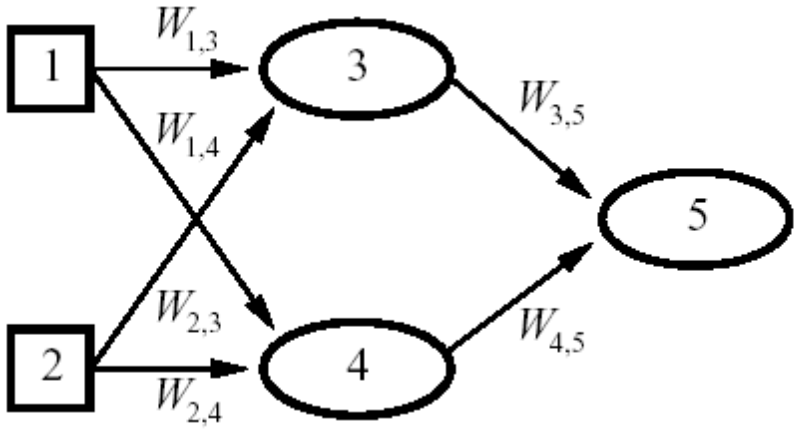
Feed-forward networks:

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down.

Feedback networks (Recurrent network:)

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent.

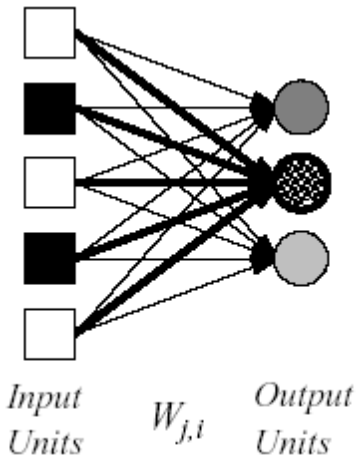
Feed-forward example



Here;
a5 = g(W3;5 a3 +W4;5 a4)
= g(W3;5 g(W1;3 a1 +W2;3 a2) + W4;5 g(W1;4 a1 +W2;4 a2)

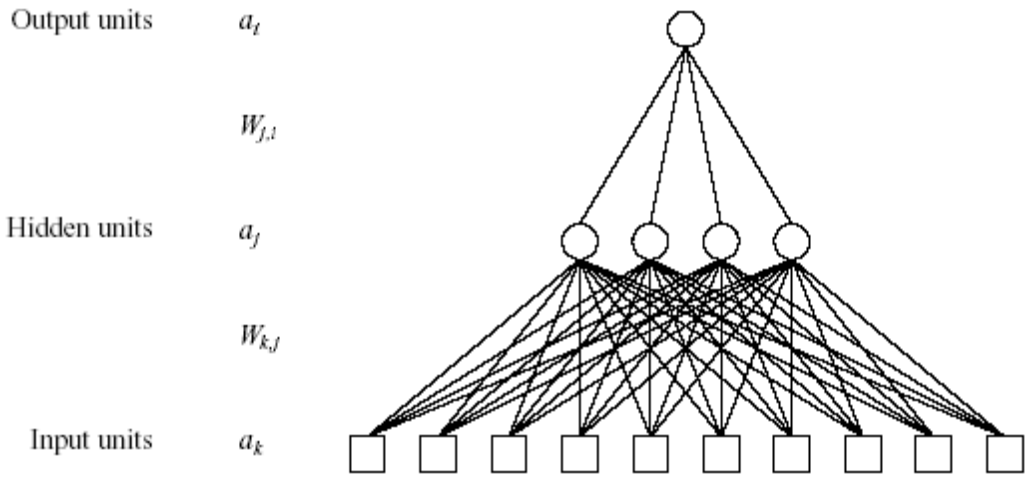
Types of Feed Forward Neural Network:
Single-layer neural networks

A neural network in which all the inputs connected directly to the outputs is called a single-layer neural network, or a perceptron network. Since each output unit is independent of the others each weight affects only one of the outputs.



Multilayer neural networks

The neural network which contains input layers, output layers and some hidden layers also is called multilayer neural network. The advantage of adding hidden layers is that it enlarges the space of hypothesis. Layers of the network are normally fully connected.



Once the number of layers, and number of units in each layer, has been selected, training is used to set the network's weights and thresholds so as to minimize the prediction error made by the network

Training is the process of adjusting weights and threshold to produce the desired result for different set of data.