Assignment No: 01

Aim: Implement basic logic gates using Mc-Culoch-Pitts or Hebbnet neural networks.

Objectives:

1. To learn basic logic gates using Mc-Culoch-Pitts neural networks.

Software Requirements:

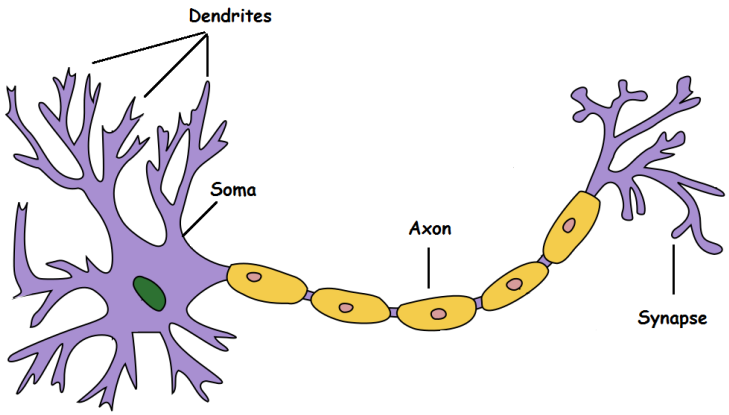
Ubuntu 18.04

Hardware Requirements:

Pentium IV system with latest configuration

Theory:

It is very well known that the most fundamental unit of deep neural networks is called an artificial neuron/perceptron. But the very first step towards the perceptron we use today was taken in 1943 by McCulloch and Pitts, by mimicking the functionality of a biological neuron.



Dendrite: Receives signals from other neurons

Soma: Processes the information

Axon: Transmits the output of this neuron

Synapse: Point of connection to other neurons

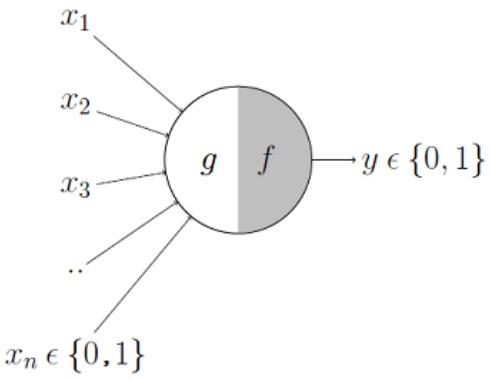
Basically, a neuron takes an input signal (dendrite), processes it like the CPU (soma), passes the output through a cable like structure to other connected neurons (axon to synapse to other neuron's dendrite). Now, this might be biologically inaccurate as there is a lot more going on out there but on a higher level, this is what is going on with a neuron in our brain—takes an input, processes it, throws out an output.

Our sense organs interact with the outer world and send the visual and sound information to the neurons. Let's say you are watching Friends. Now the information your brain receives is taken in by the "laugh or not" set of neurons that will help you make a decision on whether to laugh or not. Each neuron gets fired/activated only when its respective criteria (more on this later) is met like shown below.

McCulloch-Pitts Neuron

The first computational model of a neuron was proposed by Warren MuCulloch

(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:

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

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

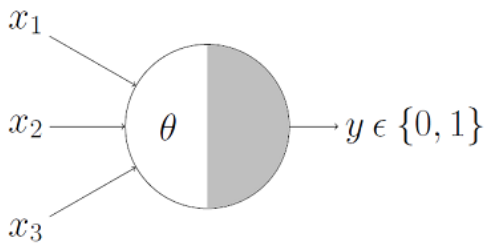
We can see that g(x) is just doing a sum of the inputs—a simple aggregation. And theta here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the theta is 2 here. This is called the Thresholding Logic.

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

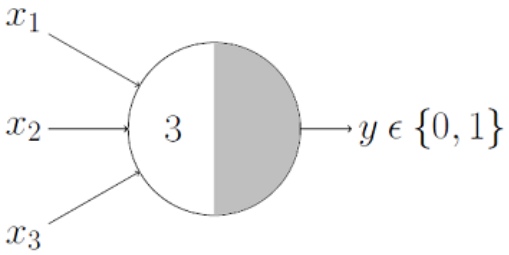
the output is also boolean so essentially, the neuron is just trying to learn a boolean function. A lot of boolean decision problems can be cast into this, based on appropriate input variables— like whether to continue reading this post, whether to watch Friends after reading this post etc. can be represented by the M-P neuron.

M-P Neuron: A Concise Representation



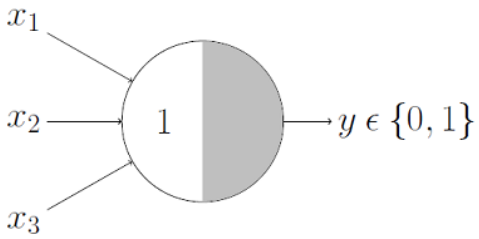
This representation just denotes that, for the boolean inputs x_1, x_2 and x_3 if the g(x) i.e., sum ≥ theta, the neuron will fire otherwise, it won't.

AND Function



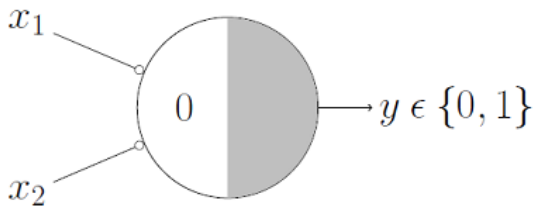
An AND function neuron would only fire when ALL the inputs are ON i.e., g(x) ≥ 3 here.

OR Function



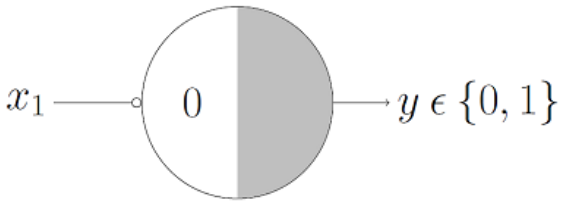
I believe this is self explanatory as we know that an OR function neuron would fire if ANY of the inputs is ON i.e., g(x) ≥ 1 here.

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. 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.

Conclusion:

Thus we learned implementation of basic logic gates using Mc-Culoch-Pitts or

Hebbnet neural networks.