

McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron



Akshay L Chandra · [Follow](#)

Published in Towards Data Science · 10 min read · Jul 24, 2018



2K



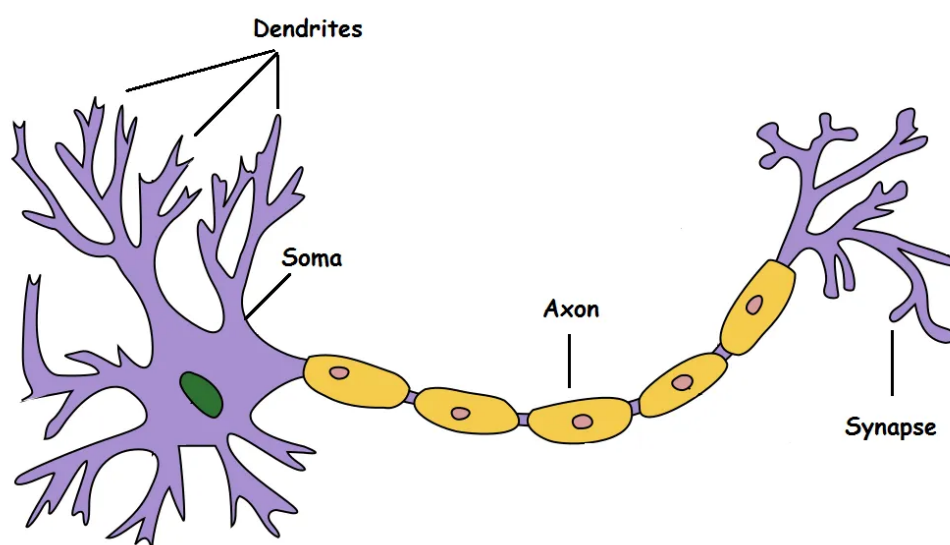
13



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.

Note: The concept, the content, and the structure of this article were directly taken from the awesome lectures and the material offered by Prof. Mitesh M. Khapra on NPTEL's Deep Learning course. Check it out!

Biological Neurons: An Overly Simplified Illustration



A Biological Neuron — [Wikipedia](#)

Dendrite: Receives signals from other neurons

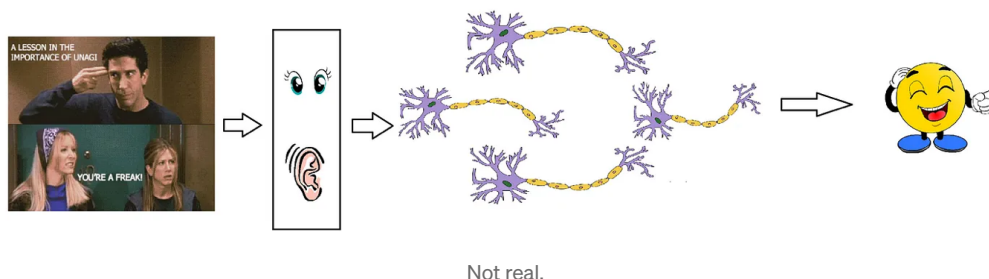
Soma: Processes the information

Axon: Transmits the output of this neuron

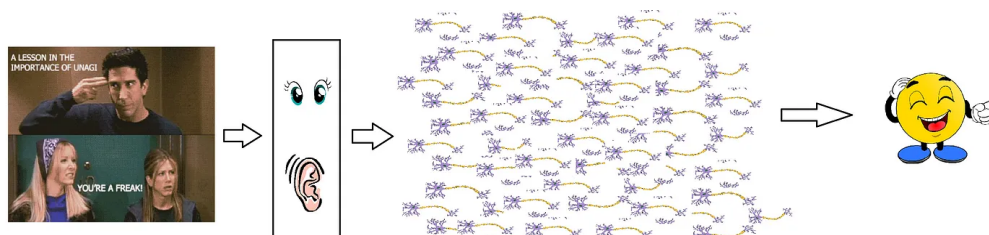
Synapse: Point of connection to other neurons

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.



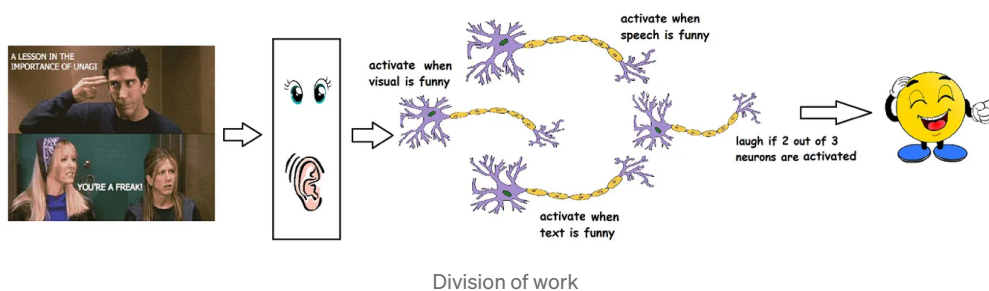
Of course, this is not entirely true. In reality, it is not just a couple of neurons which would do the decision making. There is a massively parallel interconnected network of 10^{11} neurons (100 billion) in our brain and their connections are not as simple as I showed you above. It might look something like this:



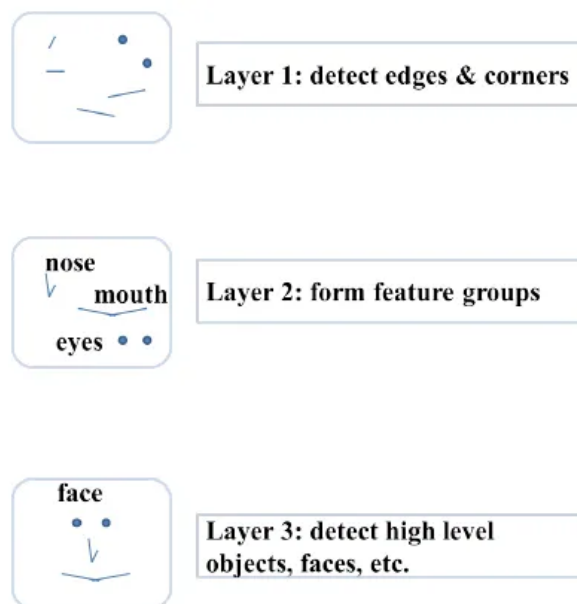
Still not real but closer.

Now the sense organs pass the information to the first/lowest layer of neurons to process it. And the output of the processes is passed on to the next layers in a hierarchical manner, some of the neurons will fire and some won't and this process goes on until it results in a final response — in this case, laughter.

This massively parallel network also ensures that there is a division of work. Each neuron only fires when its intended criteria is met i.e., a neuron may perform a certain role to a certain stimulus, as shown below.



It is believed that neurons are arranged in a hierarchical fashion (however, many credible alternatives with experimental support are proposed by the scientists) and each layer has its own role and responsibility. To detect a face, the brain could be relying on the entire network and not on a single layer.



Sample illustration of hierarchical processing. **Credits:** Mitesh M. Khapra's lecture slides

Now that we have established how a biological neuron works, let's look at



Search

Write

Sign up

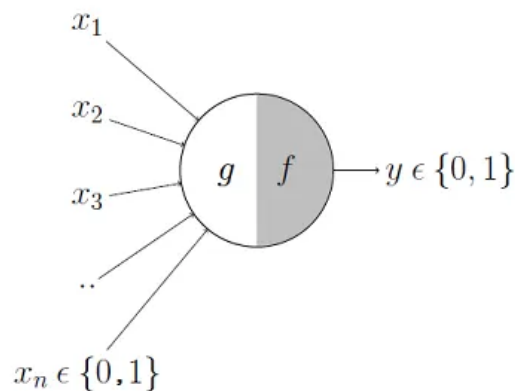
Sign In



Note: My understanding of how the brain works is very very very limited. The above illustrations are overly simplified.

McCulloch-Pitts Neuron

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



This is where it all began..

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 $\{0: \text{Will watch it}, 1: \text{Won't watch it}\}$.

- 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:

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

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

We can see that $g(\mathbf{x})$ 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 let's 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.

Top highlight

M-P Neuron: A Concise Representation

This representation just denotes that, for the boolean inputs x_1, x_2 and x_3 if the $g(\mathbf{x})$ i.e., $\text{sum} \geq \text{theta}$, 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., $g(\mathbf{x}) \geq 3$ 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., $g(\mathbf{x}) \geq 1$ here.

A Function With An Inhibitory Input

Now this might look like a tricky one but it's really not. 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 $x_1 \text{ AND } !x_2$ 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

But in both Case 2 and Case 3, we know that the output will be 0 because x_2 is 1 in both of them, thanks to the inhibition. And we also know that $x_1 \text{ AND } !x_2$ would output 1 for Case 1 (above) so our thresholding parameter holds good for the given function.

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, let's understand what M-P neuron is doing geometrically.

Geometric Interpretation Of M-P Neuron

This is the best part of the post according to me. Let's start with the OR function.

OR Function

We already discussed that the 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., $x_1 + x_2 \geq 1$ using which we can draw the decision boundary as shown in the graph below. Mind you again, this is not a real number graph.

We just used the aggregation equation i.e., $x_1 + x_2 = 1$ to graphically show that all those inputs whose output when passed through the OR function M-P neuron lie ON or ABOVE that line and all the input points that lie BELOW that line are going to output 0.

Voila!! The M-P neuron just learnt a linear decision boundary! The M-P neuron is splitting the input sets into two classes — positive and negative. Positive ones (which output 1) are those that lie ON or ABOVE the decision boundary and negative ones (which output 0) are those that lie BELOW the decision boundary.

Let's convince ourselves that the M-P unit is doing the same for all the boolean functions by looking at more examples (if it is not already clear from the math).

AND Function

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!

Tautology

Too easy, right?

I think you get it by now but what if we have more than 2 inputs?

OR Function With 3 Inputs

Lets just generalize this by looking at a 3 input OR function M-P unit. In this

case, the possible inputs are 8 points — (0,0,0), (0,0,1), (0,1,0), (1,0,0), (1,0,1),... you got the point(s). We can map these on a 3D graph and this time we draw a decision boundary in 3 dimensions.

| “Is it a bird? Is it a plane?”

Yes, it is a PLANE!

The plane that satisfies the decision boundary equation $x_1 + x_2 + x_3 = 1$ is shown below:

Take your time and convince yourself by looking at the above plot that all the points that lie ON or ABOVE that plane (positive half space) will result in output 1 when passed through the OR function M-P unit and all the points that lie BELOW that plane (negative half space) will result in output 0.

Just by hand coding a thresholding parameter, M-P neuron is able to conveniently represent the boolean functions which are linearly separable.

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

Limitations Of M-P Neuron

- What about non-boolean (say, real) inputs?
- Do we always need to hand code the threshold?

- Are all inputs equal? What if we want to assign more importance to some inputs?
- What about functions which are not linearly separable? Say XOR function.

I hope it is now clear why we are not using the M-P neuron today.

Overcoming the limitations of the M-P neuron, Frank Rosenblatt, an American psychologist, proposed the classical perception model, the mighty *artificial neuron*, in 1958. It is more generalized computational model than the McCulloch-Pitts neuron where weights and thresholds can be learnt over time.

More on *perceptron* and how it learns the weights and thresholds etc. in my future posts.

Conclusion

In this article, we briefly looked at biological neurons. We then established the concept of McCulloch-Pitts neuron, the first ever mathematical model of a biological neuron. We represented a bunch of boolean functions using the M-P neuron. We also tried to get a geometric intuition of what is going on with the model, using 3D plots. In the end, we also established a motivation for a more generalized model, the one and only *artificial neuron/perceptron* model.

Thank you for reading the article.

Live and let live!

A

Deep Learning

Artificial Neuron

Biological Neuron

McCulloch Pitts Neuron

Perceptron



Written by Akshay L Chandra

1.2K Followers · Writer for Towards Data Science

M.Sc. Student @ University of Freiburg

Follow



More from Akshay L Chandra and Towards Data Science

 Akshay L Chandra in Towards Data Science**Perceptron Learning Algorithm: A Graphical Explanation Of Why It**

This post will discuss the famous Perceptron Learning Algorithm, originally proposed by

8 min read · Aug 22, 2018



3.3K

11

 Adrian H. Raudaschl in Towards Data Science**Forget RAG, the Future is RAG-Fusion**

The Next Frontier of Search: Retrieval Augmented Generation meets Reciprocal

★ · 10 min read · Oct 6



2.2K

24

 Marco Peixeiro  in Towards Data Science**TimeGPT: The First Foundation Model for Time Series Forecasting**

Explore the first generative pre-trained forecasting model and apply it in a project

★ · 12 min read · Oct 24



1.6K

18

 Akshay L Chandra in Towards Data Science**Learning Parameters Part 5: AdaGrad, RMSProp, and Adam**

Let's look at gradient descent with adaptive learning rate.


6 min read · Sep 27, 2019



293

2

[See all from Akshay L Chandra](#)[See all from Towards Data Science](#)**Recommended from Medium**

 Prathammodi

Convolutional Neural Networks for Dummies

So you want to learn about Convolutional Neural Networks, CNNs, huh? Well, you've

6 min read · Oct 14

 Tiya Vaj

Adversarial Neural Network

An adversarial neural network, often referred to as an “adversarial network” or simply

2 min read · Oct 27



Lists

Natural Language Processing

786 stories · 363 saves

Practical Guides to Machine Learning


10 stories · 634 saves

New_Reading_List

174 stories · 174 saves

Staff Picks


495 stories · 414 saves

 Devanshu Tayal

Unleashing the Power of Neural Networks: Introducing Liquid

In the vast landscape of artificial intelligence, one technology stands tall, revolutionizing

5 min read · Jul 31

 Yanbo Ding

Key Concepts in Reinforcement Learning

In this blog, I will discuss fundamental key concepts and missions in reinforcement


5 min read · 2 days ago

 Navin

Implementing Multilayer Perceptron

A Perceptron is the most fundamental unit in neural networks. This is the building block of

5 min read · Jun 3

 Francesco Franco

How Does the Softmax Activation Function Work?

When you're creating a neural network for classification, you're likely trying to solve

12 min read · Oct 13

[See more recommendations](#)