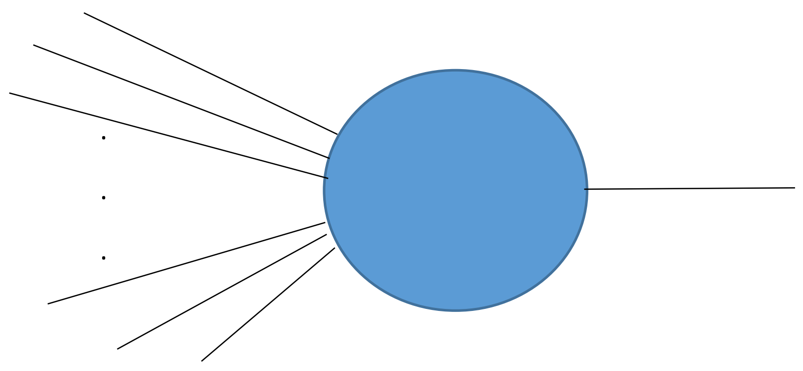
Perceptron:

The perceptron is one of the oldest concepts in neural networks. It was first introduced in 1943.
The goal of neural networks has always been to take inspiration from the human brain, particularly from the way neurons function. In short, our brain is made up of countless neurons connected to each other that exchange information. Everything we think and do is due to them. We receive inputs from the outside (input), they get activated, and we act accordingly (output). This is exactly how we should see a neural network, and in our case, the perceptron. The difference is simply that in neural networks, the inputs are just numbers. Lots of numbers.

Now, it's time to talk about the perceptron. The perceptron is nothing more than a single small neuron. Nothing more, nothing less. Let's imagine it as a small sphere. This sphere has many incoming connections and only one output. This means it receives n pieces of information, but only returns a single value.



So this is the perceptron. On the left, it receives many pieces of data, processes them, and returns a single result. The problem with the perceptron, however, is precisely its simplicity. In fact, it is almost unusable; the problems it can solve are very few.

Before talking about this, though, it's important to understand how a perceptron actually works.

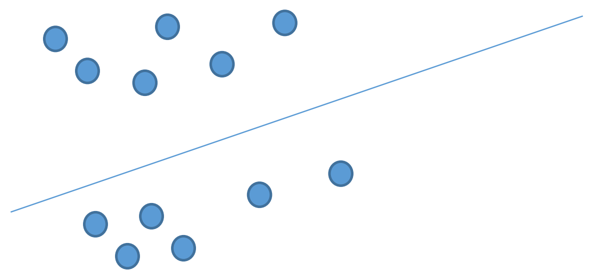
Suppose we receive n inputs. Each input has an associated weight. Every input is multiplied by its corresponding weight, all the multiplication results are summed, and this value is fed into the neuron. The neuron, based on the input it receives, will act accordingly. In our case, the perceptron can only return either 0 or 1.

Why do we use these weights? These weights are used to make the neuron “think”; it is crucial to find the correct values to assign to the weights. But how do we find these values? Fortunately, there’s no need to guess—they are found by training the model (though this requires writing some additional code). Thanks to training on a dataset, each weight acquires its correct value. Of course, both the dataset and the training process must be correct (the training code can be found in the file perceptron.py).

So this is how the perceptron works (in a very simple way). We can say that all of this forms the foundation of how more complex neural networks operate.

There’s just one question left: Why does it return 0 or 1? The perceptron only returns 0 or 1 because of the activation function defined within it. The activation function, as the name suggests, is the function that activates whenever the neuron receives an input. The function processes it and returns the result. In this case, the perceptron has a step function defined within it. That is, a piecewise function that returns 0 if the input is negative and 1 otherwise.

This is an example of a problem that can be solved with a perceptron, since the two classes can be easily separated by drawing a dividing line.



For all of this to work correctly, however, we must ensure that the model has been properly trained so that the weights have the correct values.