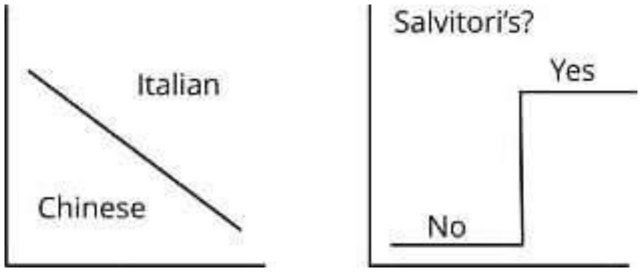
Module 1

Sigmoid Neurons

The Perceptron is an artificial neuron that produces a linear/binary output. The perceptron can distinguish between only two classes.

For example, if I provided the perceptron with a description of two restaurants (Italian and Chinese), and asked which I should choose to eat at, it would, at best, be able to provide a thumbs up or thumbs down to help me distinguish between two.

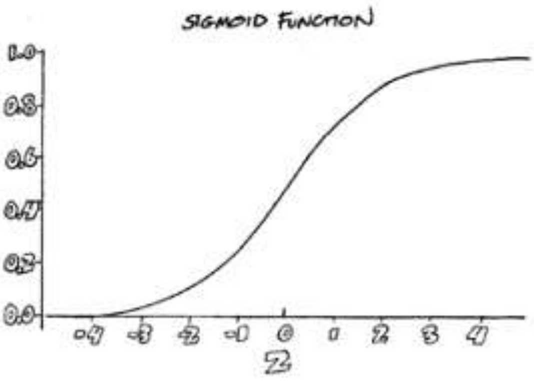
If you were to graph the output of the perceptron, it would look like this:



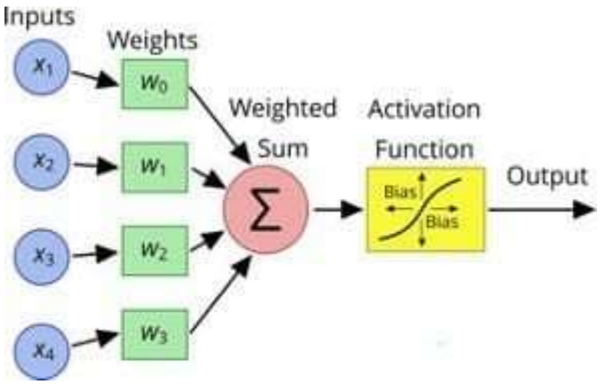
What I'd really like is a sliding scale that gives me some indication of the relative quality of the restaurant, maybe a ranking from 0 to 5. The closer I am to five, the more confident the network is in the quality of the restaurant. If it's close to zero then I should probably go somewhere else. For this purpose, I might try to use something different — a sigmoid neuron.

The Sigmoid Neuron

A sigmoid neuron can handle far more variation in values than the binary choices you get with a perceptron. In fact, you can put any two numbers you want in a sigmoid neuron and it will squeeze a sigmoid curve into that range. It's called a sigmoid neuron because its function's output forms an S-shaped curve when plotted on a graph; an S is almost like a line that's been squeezed into a smaller space.



The sigmoid neuron works like a perceptron, using weighted inputs. The key difference is the activation function. Instead of using a linear or stair-step function that results in binary classification (0 or 1), a sigmoid neuron uses the sigmoid function to deliver an infinite range of values between the specified upper and lower values.



If I were to use the sigmoid neuron to help me decide whether to eat at a certain restaurant, it would provide me with a clearer indication of the restaurant's quality.

Instead of giving a thumbs up or thumbs down, it provides a value within a range of values, which indicates the probability of the restaurant being a good choice.

A Key to Deep Learning

The sigmoid neuron is a key to deep learning — a subset of machine learning in which the neural network is capable of learning unsupervised from data that is unstructured or unlabeled. Using a learning algorithm, you can start by assigning random values to the weights and bias. When you feed in the data inputs, the neural network calculates the predicted outcome and computes the overall loss (squared error loss) of the model.

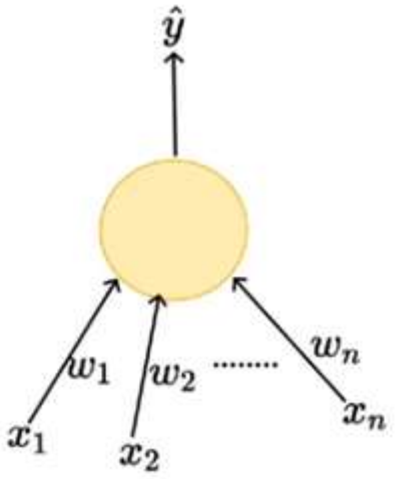
Based on the overall loss, the neural network adjusts the weights and biases and repeats its calculations. It continues this process until the overall loss is as close to zero as possible, meaning that the output is as accurate as possible.

The complexity of the mathematics involved in learning functions is beyond the scope of a general article, such as this one. The key takeaways here is that the sigmoid neuron takes the perceptron to the next level, enabling it to output a range of values between the specified upper and lower limits and that it enables a neural network to learn unsupervised — without having to be trained with a set of structured, labeled data.

Why Sigmoid Neuron

Before we go into the working of a sigmoid neuron, let's talk about the perceptron model and its limitations in brief.

Perceptron model takes several real-valued inputs and gives a single binary output. In the perceptron model, every input xi has weight wi associated with it. The weights indicate the importance of the input in the decision-making process. The model output is decided by a threshold **W**$_o$ if the weighted sum of the inputs is greater than threshold **W**$_o$ output will be 1 else output will be 0. In other words, the model will fire if the weighted sum is greater than the threshold.

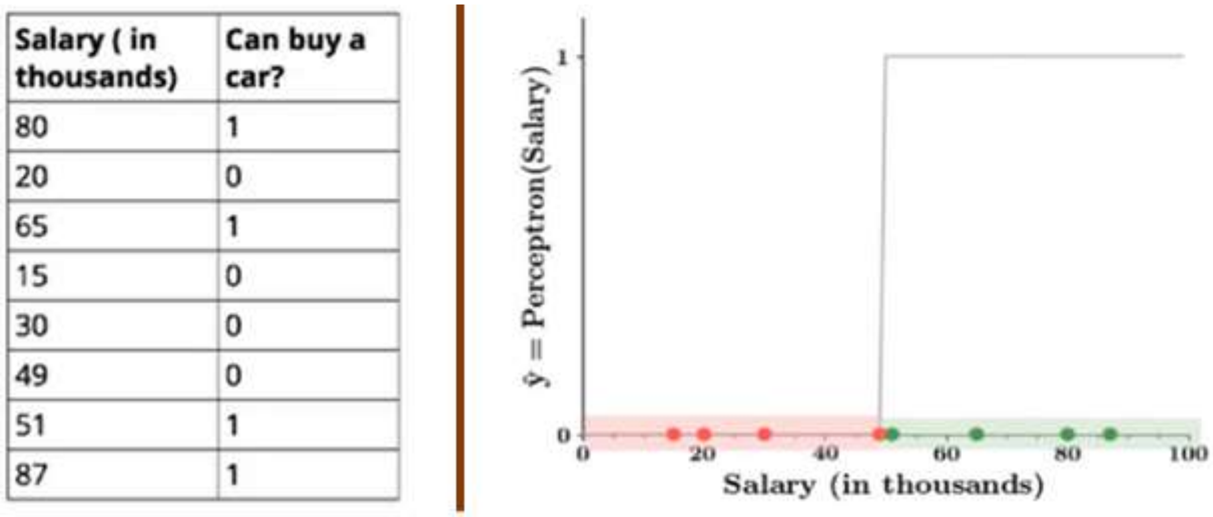


$$\hat{y} = 1 \text{ if } \sum_{i=1}^{n} w_i x_i \geq b$$

$$\hat{y} = 0 \text{ otherwise}$$

Perceptron (Left) & Mathematical Representation (Right)

From the mathematical representation, we might say that the thresholding logic used by the perceptron is very harsh. Let's see the harsh thresholding logic with an example. Consider the decision making process of a person, whether he/she would like to purchase a car or not based on only one input X1—Salary and by setting the threshold **b**(**W**$_o$) = -10 and the weight **W**$_1$ = 0.2. The output from the perceptron model will look like in the figure shown below.



Salary (in thousands)	Can buy a car?
80	1
20	0
65	1
15	0
30	0
49	0
51	1
87	1

Data (Left) & Graphical Representation of Output(Right)

Red points indicates that a person would not buy a car and green points indicate that person would like to buy a car. Isn't it a bit odd that a person with 50.1K will buy a car but someone with a 49.9K will not buy a car? The small change in the input to a perceptron can sometimes cause the output to completely flip, say from 0 to 1. This behavior is not a characteristic of the specific problem we choose or the specific weight and the threshold we choose. It is a characteristic of the perceptron neuron itself which behaves like a step function. We can overcome this problem by introducing a new type of artificial neuron called a *sigmoid* neuron.