MLT-03 Summary Document

Overview

This document summarizes the lecture MLT-03 on Machine Learning. This lecture will help you understand neural networks.

The lecture covers the following topics:

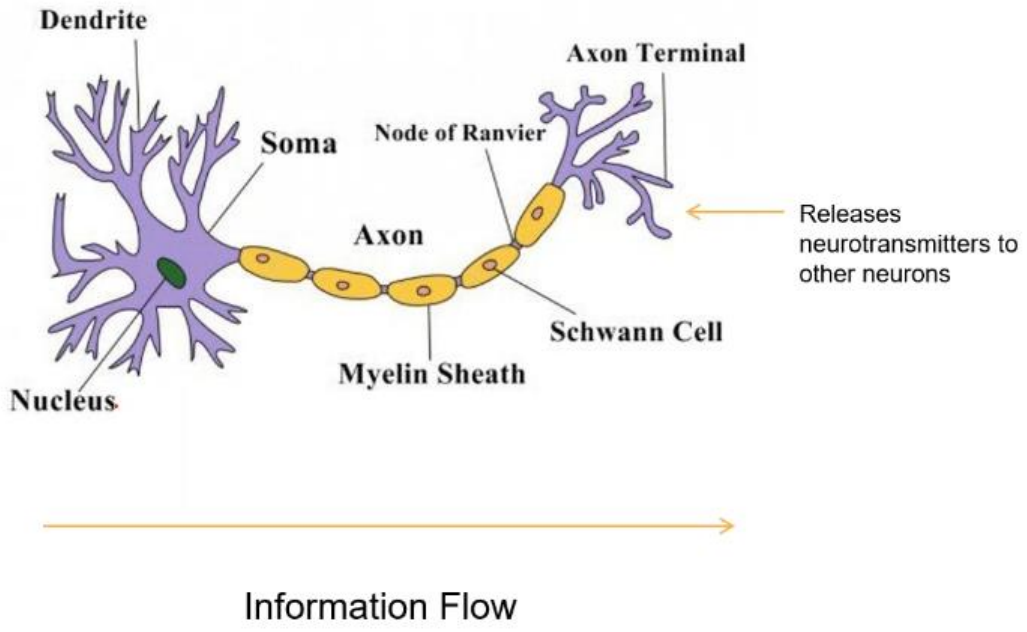
- Introduction to Neural networks
- A Typical Neuron and Application
- Deep Feed Forward Network
- The Architecture
- Types of Activation Functions
- Training Neural Network
- Gradient Descent and Backpropagation
- Application of Neural Networks in Practice
- Sentiment Analysis

Introduction to Neural networks

Neural networks belong to the deep learning classification in the artificial intelligence domain. In this lecture, we will specialise in the context of supervised learning, i.e., we will explain neural networks when there's a target variable to base the model training.

A Typical Neuron and Application

A cell in a brain can be graphed as the following:



- The dendrites take the signals from the external world. The signal then gets aggregated in the nucleus (the centre). Next the signal goes through the Axon and it gets finally to the Axon Terminal, where the information of the signal is passed to another neuron through the terminal.

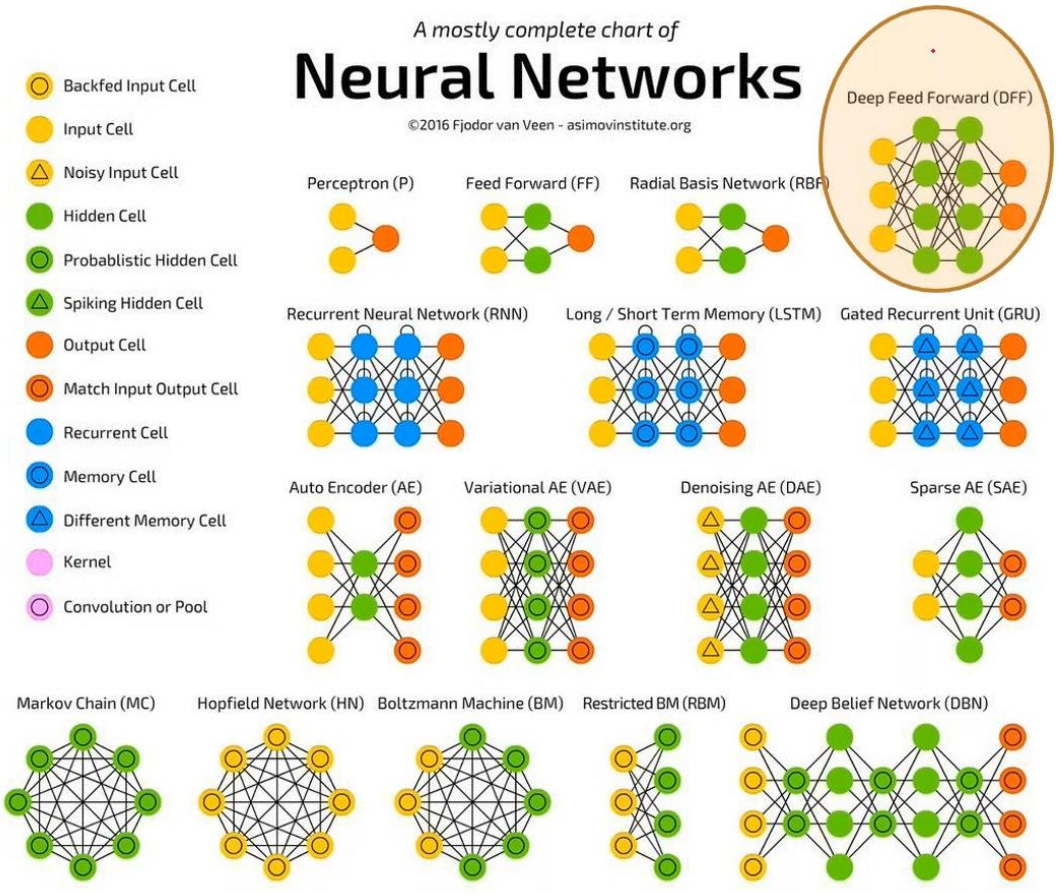
A similar process happens in an artificial neural network. The whole process is described as a circle, which:

- Receives inputs
- Transform the inputs into a signal
- Gives an output based on the above.

Applications of neural networks include, but are not limited to, the following:

- Speech and handwriting recognition
- Self-driving cars
- Time series predictions

Deep Feed Forward Network



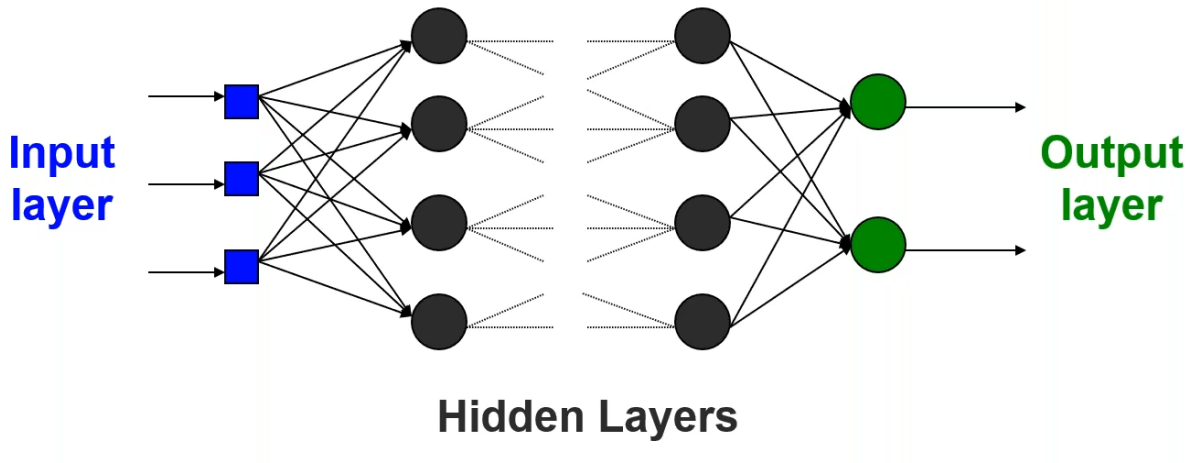
There are many kinds of neural networks. The basic neural network architecture is the perceptron. The perceptron (P as in the figure above), consists of only an input and output layer. In the input layer (yellow circles), you will find

the inputs (or the features' data). The nucleus of the neural network will be located in the orange circle. In this last circle, we will have the model applied with the inputs, and it will give the output based on the trained model.

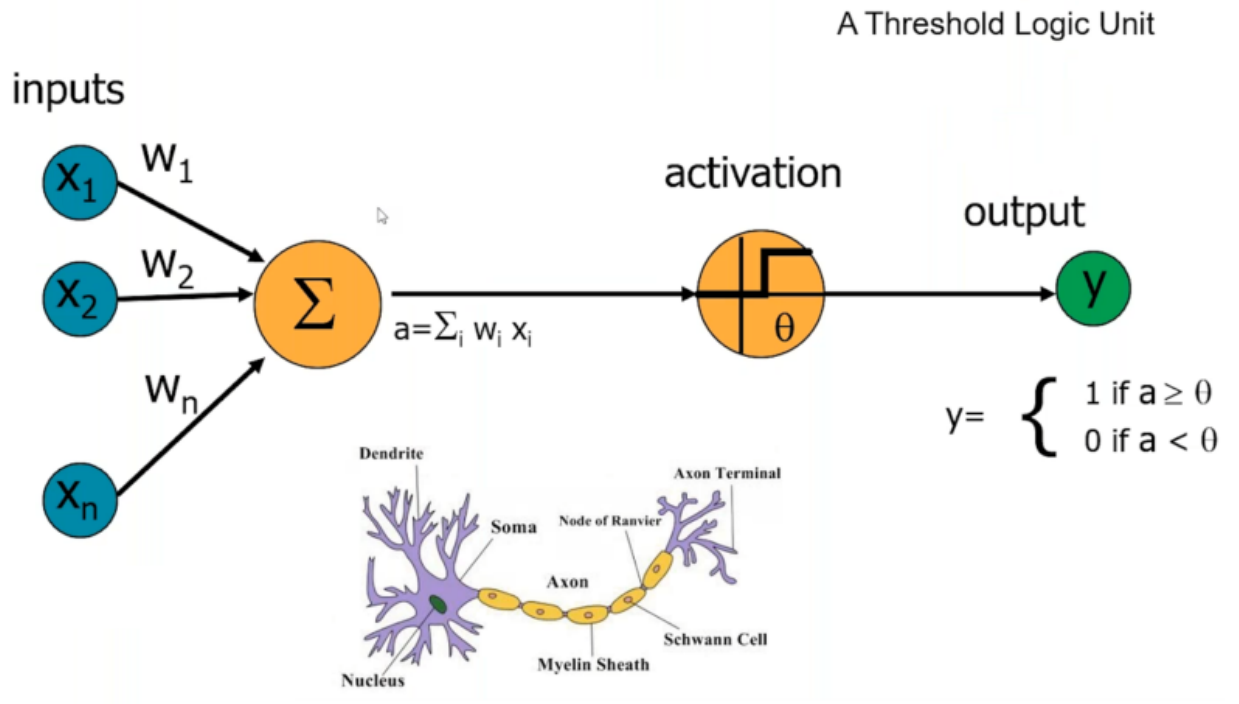
Deep feed-forward neural networks are one of them When the whole network becomes deeper than a simple perceptron, as you can see in the figure above. Besides, it is feed-forward because the information flows in a single direction, "forward".

Let's dive deeper into this type of neural network and check its architecture.

The Architecture

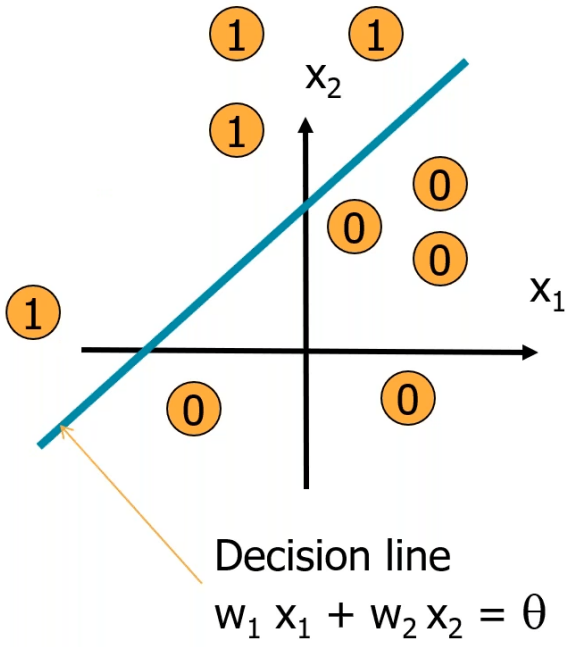


The architecture is as follows. Each circle, as described above, can be interpreted as a neural network. There are 3 layers. First, we have the inputs, then we use these inputs to get signals with the hidden layers. This intermediate layer then gives a result, the output layer, which can be a prediction of a stock price movement based on OHLC data (input data).



Let us try to refresh with the brain neurons.

The dendrites is like the input layer, the neuron receives the input information here. In the nucleus, we have the Σ, which is a model (in the neuron) to interpret the input information. Then, we use an activation function (the axon), which transforms the model into something that can be changed to a desired output. Finally we have the output, which is located in the axion terminal and ready to be used and passed to another neuron. The architecture, as you saw previously, is built with many circles (neurons) connected each of them with the other neurons in a "forward" fashion.



A TLU works as a linear classifier

Similar to SVM?

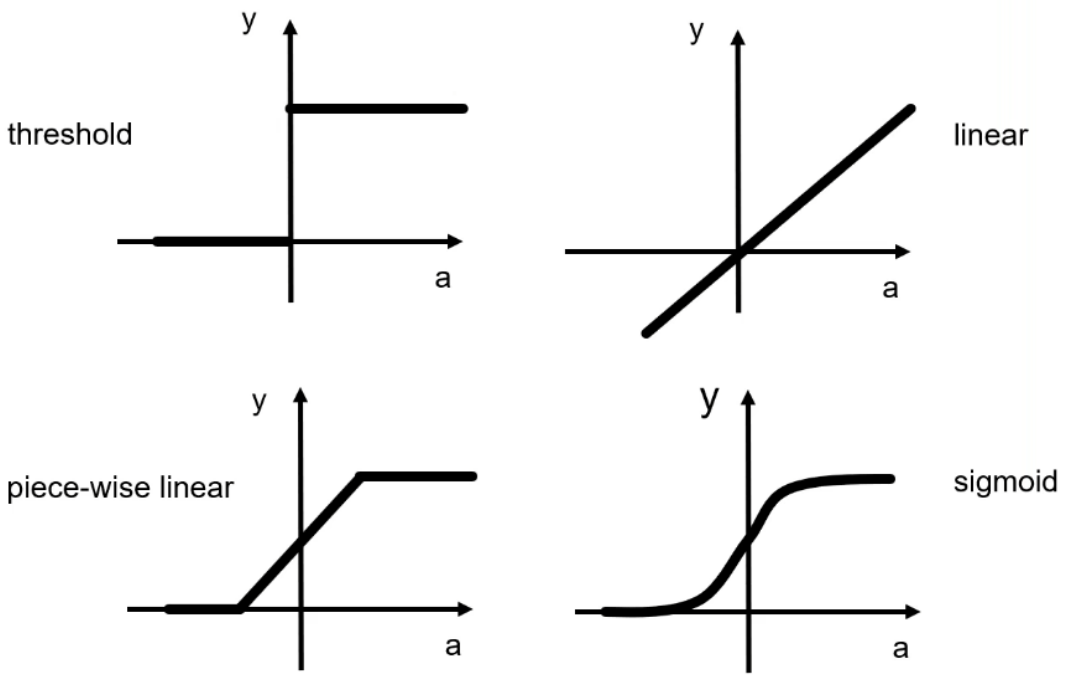
How do you identify the weights and threshold?

The activation function, together with the Σ model, works similarly as the support vector machine model.

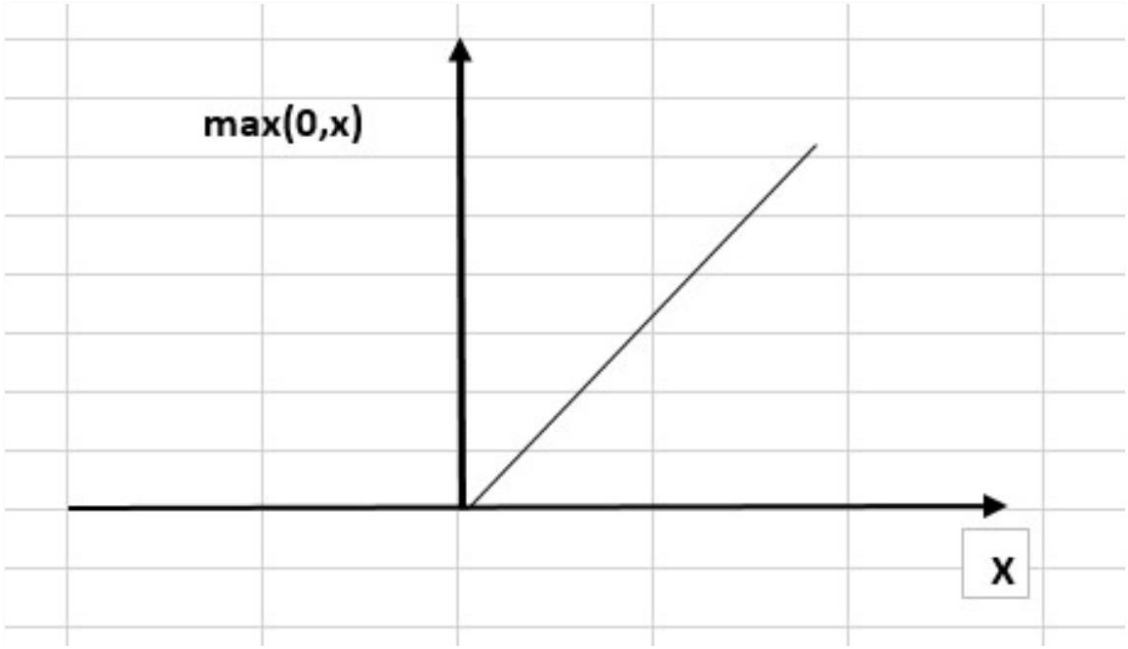
The output, for the neuron, can be a classification target or a ratio type variable. If it's a ration type target, threshold logic unit (TLU) is not used; otherwise, it is used. The latter makes the neuron able to output a classification target variable.

Types of Activation Functions

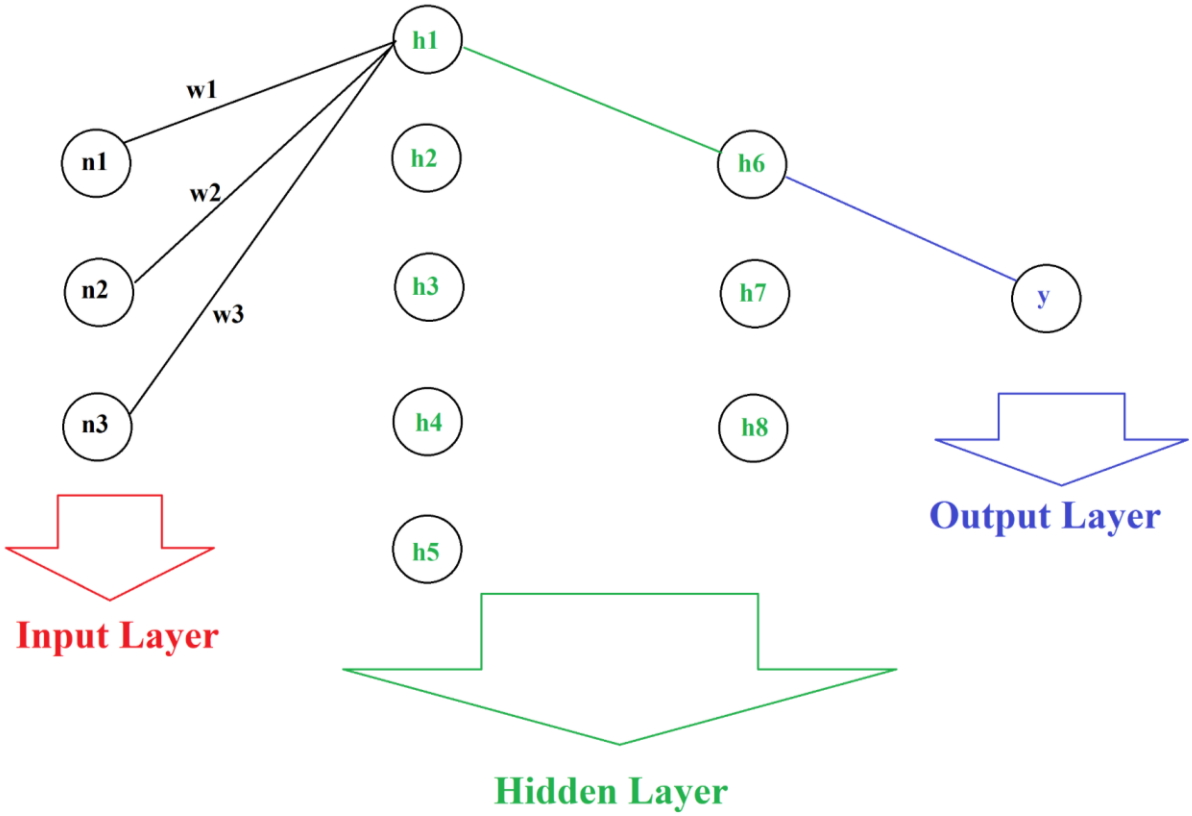
These types can be as follows:



One famous activation function is the ReLU function. The corresponding figure can be found below:



Training Neural Network



In the input layer, you have the inputs, each input "n", will have its unique weight "w". The product of each input together with its weight the corresponding sum of all of them makes the Σ model.

This model is analysed, activated and produces an output on each hidden layer neuron (from h1 to h5). Then these 5 neurons will produce outputs for the following 3 neurons (from h6 to h8), which in turn, will used the output of the previous neurons as inputs, and produce also an output. Finally the output is produced by a single neuron in the output layer.

How do you choose the weights? You have to train the model with the input variables to output the desired target variable. Each neuron will have its unique set of weights. You train the model and get an error between the predicted output and the target variable for the whole target vector. You will create a "loss" function with the sum of squared errors (it's a way, there are many more) to update it in order to reduce on each iteration this loss function. The final weights that result from this minimisation are the ones chosen as the best parameters of the model.

In order to make this optimisation for a neural network, there are several methods, such as the Backpropagation algorithm. You minimise the error, mentioned previously, subject to gradient of the error function.

Gradient Descent and Backpropagation

Do you remember calculus? The gradient descent algorithm is about getting the first derivative of a function (in this case our function is the loss function) and we equalise the first derivatives to zero to minimise the loss function. There are many forms to create a loss function, let's show an example of it:

$$loss\ function\ = \sum_{i=1}^{T} \quad (y_i\ - \hat{y}_i\ )^2$$

Where:
y_i : The target variable at point i.
$\hat{y}_i$: The predicted target variable at point i that results from the neural network model output.

The control variables are going to be the weights, i.e., you will update these weights on each iteration that you get the first derivatives in order to train the model. You will go through the whole iteration process until the derivative gets really close to zero. Once you get this point. You will have the best parameters (weights) for your model.

Application of Neural Networks in Practice

The professor takes you here through a python code to see an implementation of a neural network in practice.

Sentiment Analysis

You can use text from news, articles, etc as inputs to create a classification target variable. In order to do that, you have to change the text into numbers. Then, use these numbers as inputs for the neural network to produce an output.

Imagine you have in a text the following sentences:

- I liked the service
- It was horrible
- Waiting area was not so clean
- Wonderful experience

First you filter out the stop words, like: “the”, “of”, “that”, etc. Then you can use the “Term Frequency-Inverse Document Frequency”. We explain this as follows:

Imagine you have 1 million documents of the words of interest are “a”, “person”, “personal”, “information”. Each word will be present with a certain number of times on the 1 million documents.

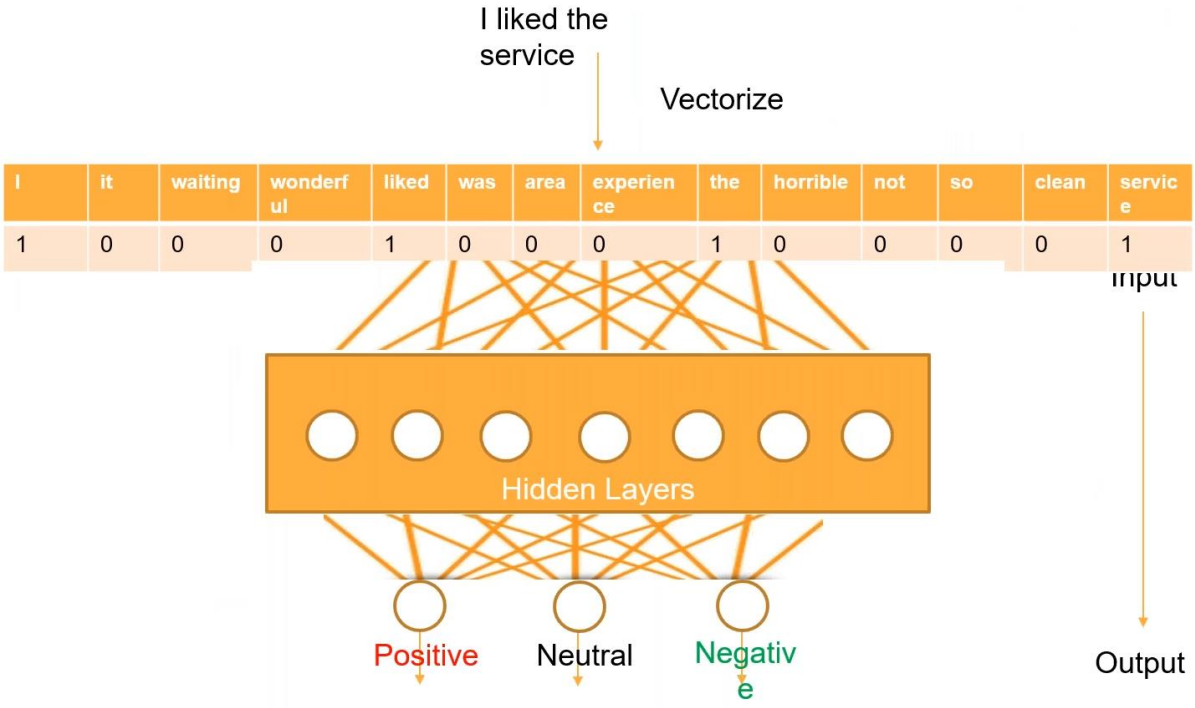
- Total number of documents/text: 1000,000
  - Term of interest:

a	present in 1000,000	$w = \log \frac{1000,000}{1000,000} = 0$
person	present in 10,000	$w = \log \frac{1000,000}{10,000} = 2$
personal	present in 1000	$w = \log \frac{1000,000}{1000} = 3$
information	present in 100	$w = \log \frac{1000,000}{100} = 4$

You will assign a weight for these important words, as follows:

Weight = total number of documents/number of documents in which appear the word. This whole process is called "vectorise a text".

Then you create a neural network as the following figure:


I liked the service
Vectorize
I
it
waiting
wonderful
liked
was
area
experience
the
horrible
not
so
clean
service
1
0
0
0
1
0
0
0
1
0
0
0
0
1
Input
Hidden Layers
Positive
Neutral
Negative
Output