This module is divided into the following topics:

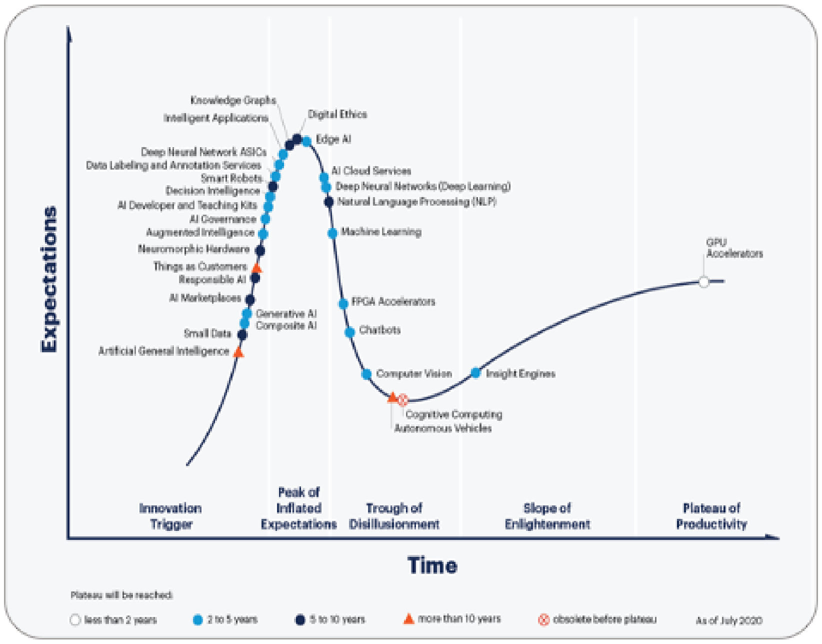
- Background of neural networks
- Inspiration from the human brain
- Architecture of neural networks
- Forward pass and its implementation in Numpy
- Loss functions
- Backpropagation
- Introduction to the Keras library
- Sentiment classification of IMDb movie reviews using the Keras library

Background of Neural Nets

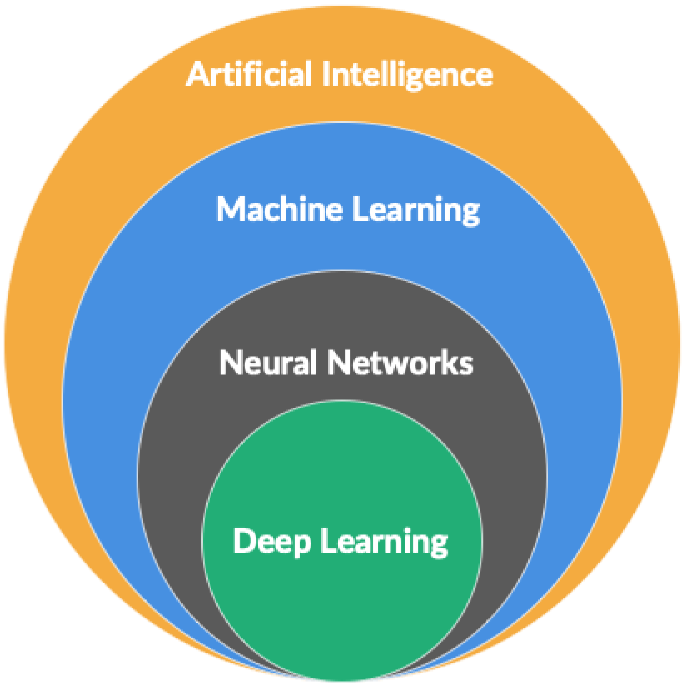
The popularity of neural networks is deeply linked to their history, which is exactly where we will start from. The following topics will be covered next:

1. Inspiration from the human brain: You will take a deeper look into how the human brain has inspired the structure of artificial neural networks.
2. Artificial neuron and its components: You will learn about the different components of a neural network such as inputs, outputs, weights, biases and activation functions.
3. Architecture of neural networks: You will develop an understanding of how multiple neurons come together to form a neural network. You will understand the concepts of fully connected neural networks.
4. Forward pass: You will learn how information passes through a neural network from the input layer to the output layer. You will look at this as a series of matrix multiplication using Numpy and Excel.

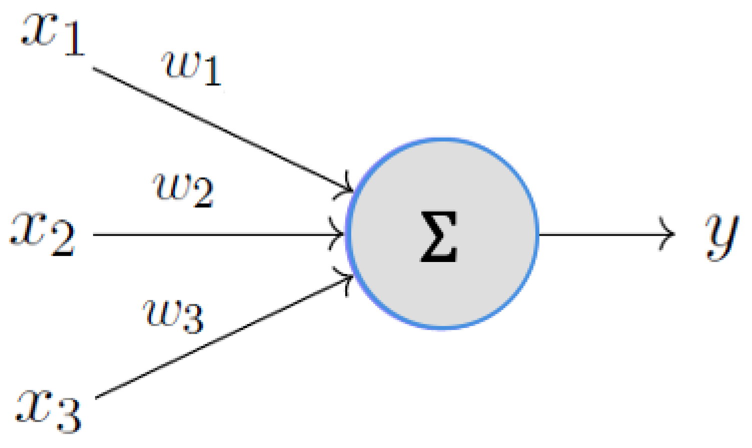
The Gartner cycle helped you understand the expectations from neural networks over time.



You can read more about the Gartner Hype Cycle and other trends in this link.
Although AI, Machine Learning, Neural Networks and Deep Learning are used interchangeably, there is a huge difference in the meaning of these terms. They are subsets of one another, as portrayed in the diagram below.



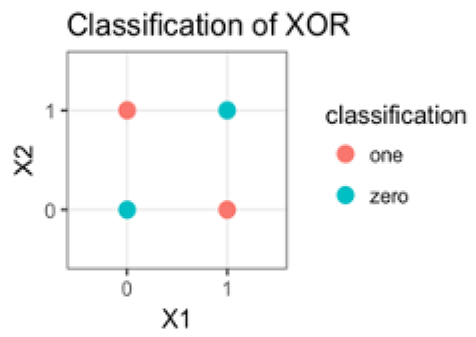
It all started with Rosenblatt's Perceptron. A single perceptron has certain inputs, weights and biases. The weighted sum of these weights and inputs combined with the biases forms the output of the perceptron, as shown in the diagram below.



$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

The Perceptron Convergence theorem guarantees that if classes are linearly separable, the perceptron will learn a linear boundary within finite steps.

There was a lot of hype around perceptrons, but it soon fell into the trough of disillusionment as in the case of the Gartner Hype Cycle.
It turned out that a perceptron could not solve a simple problem of XOR.
Although perceptron's convergence theorem guaranteed that two linearly separable classes can be separated using a single straight line, the XOR problem required two straight lines to separate two classes. As you can see in the diagram below, a single line cannot separate two classes, as it would require more than one line to do so, which means that we would require more than one perceptron for seperation.



In the 1980s, the AI winter started when there was no innovation in the field of Artificial Intelligence.
However, in the following decade, artificial intelligence was reintroduced as a concept. Let's watch the next video to find out who was responsible for the revival of AI.

The most prominent figures who were responsible for the revival of artificial intelligence are Yan Lecun, Geoffrey Hinton, Yoshua Bengio and Andrew Ng.

Lecun, Hinton and Bengio received the ACM Turing Award for their work in deep learning and are referred to as “The Godfathers of AI”.

Yan Lecun is the founding father of Convolutional Neural Networks.

Geofrrey Hinton co-authored the highly cited paper on backpropagation algorithms for training neural networks. He has also designed AlexNet, a breakthrough in computer vision, which led to the popularity of Convolutional Neural Networks in general.

Yoshua Bengio is noted for his work on artificial neural networks. He is the founder of Element AI, a Montreal-based AI-incubation firm.

Andrew Ng is the co-founder of Google Brain and is known for his video series on Deep Learning.

Today, we can see how artificial intelligence is democratised. Everyone can deploy and train neural networks using graphic cards, open source codes and MOOCs.

Artificial Neuron: Inspired from the Human Brain

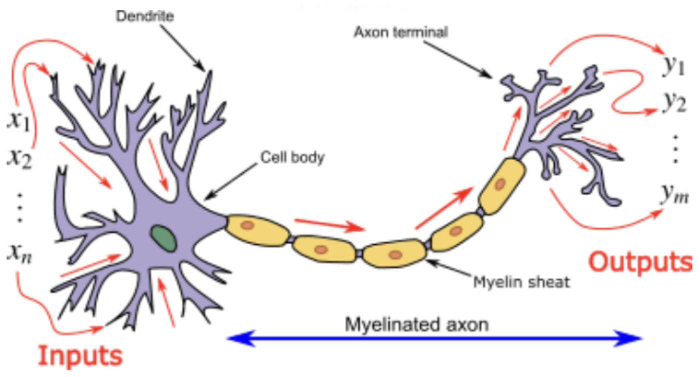
Neural networks are designed to mimic the logic and reasoning behind human activities such as driving a car or taking part in a conversation. These tasks require the different parts of a system to interact with each other to produce a final result (driving according to traffic rules/responding in the proper manner), which is what Jaidev referred to as distributed learning.

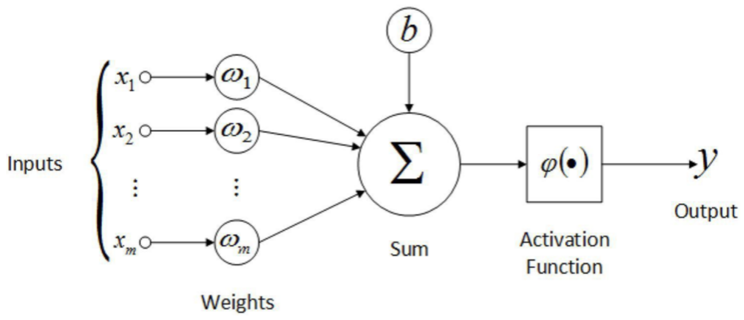
A task such as driving is quite complex with the involvement of several parameters, including traffic rules (road lines, speed limit), traffic (proximity, horn sound) and path planning. The brain needs to coordinate all of these inputs accordingly in order to drive the car successfully. Similarly, neural networks need to coordinate various inputs accordingly in order to achieve a defined output.

In order to mimic these human activities, the very architecture of an artificial neuron has been heavily inspired from the biological neuron. This analogy resides in the overall structure of the nervous system, including the brain, the spinal cord and the nerve endings, right down to the most fundamental unit, a nerve cell.

Here is a brief comparison between the elementary units of the neural network (neuron) and the brain (brain cell).

Brain Cell	Neuron
Dendrite	Inputs
Nucleus	Summation (logical operation)
Myelin sheath	Activation functions
Axon terminal	Outputs





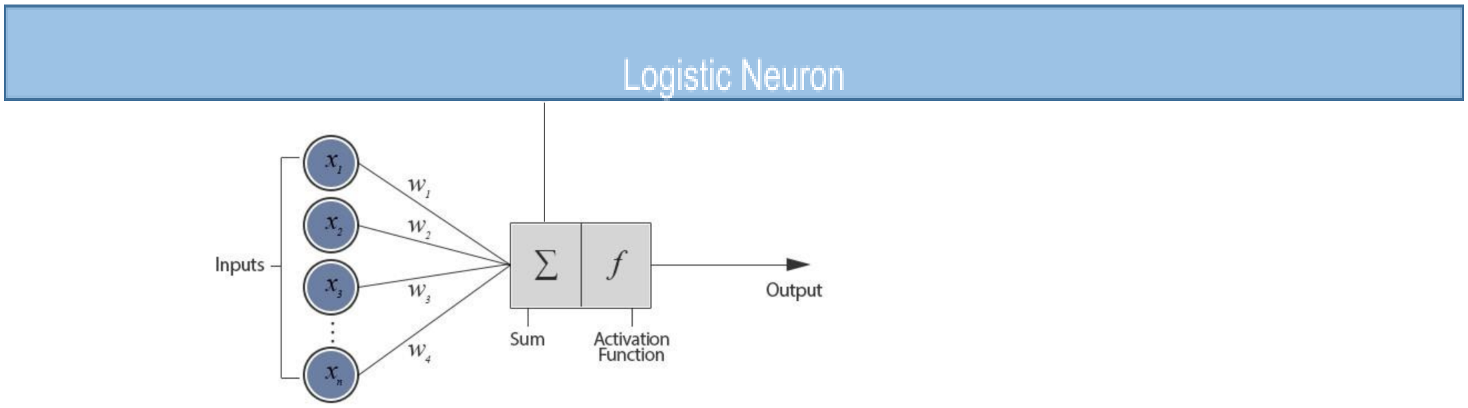
Notice how similar these two images are. Both contain a set of inputs, a carrier (with memory), which alters this input, a logical unit for elementary processing and, finally, an output. These basic units are further connected to form a bigger neural network, which can carry out more complex operations.

Some popular applications of neural networks include:

- Speech, text and language detection
  - Help users learn a new language more proficiently
  - Call intelligence
  - Customer care virtual assistant

- Computer vision
    - Autonomous driving
    - Virtual coaching assistant for sports
    - Gesture recognition for video games

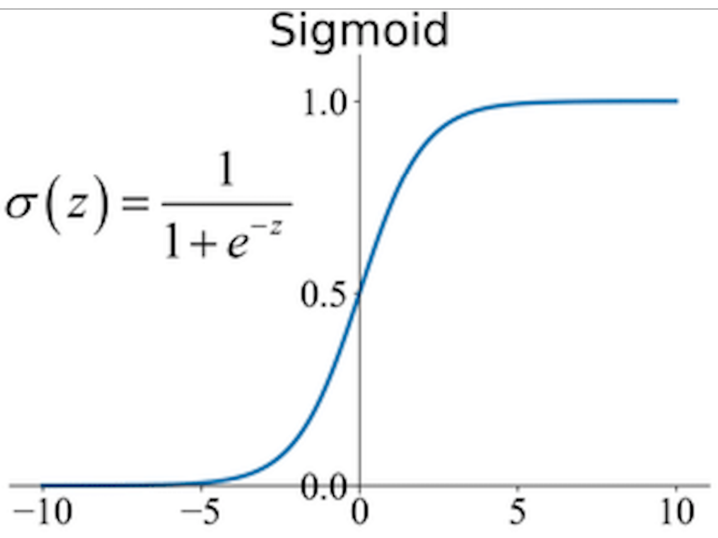
- Anomaly detection
    - Disease/Disorder recognition in medical patients
    - Banking fraud



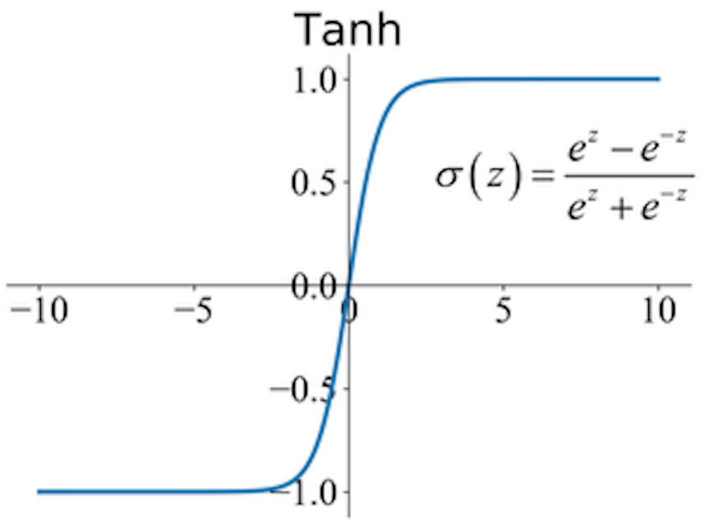
The first step that occurs in a logistic neuron is the dot product of each individual input with its corresponding weight. Next, these dot products are summed together and added to a bias term, as shown in the figure given below. At the last stage, the number exiting the summation goes through an activation function to transform it into a number in the predetermined range (for example, 0 to 1).

With this broad definition, several classes of the activation function, including tanH, Relu and Softmax, exist. Each of these functions does a slightly different manipulation, which can be seen from the shapes of their graph.

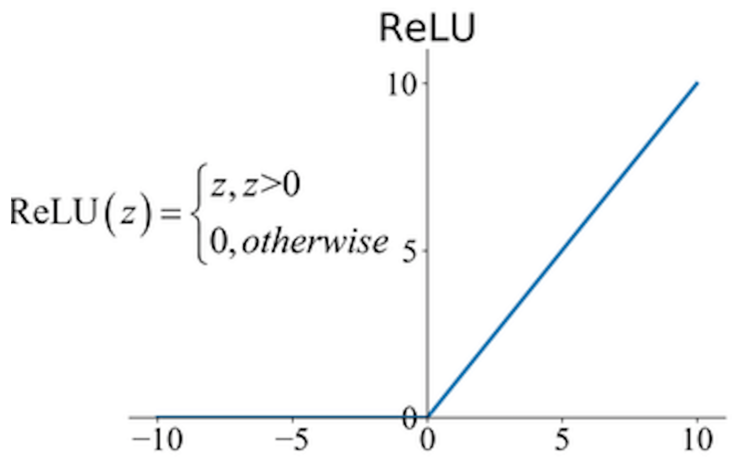
Sigmoid activation function transform any number to a value between 0 and 1.



The Tanh activation function is quite similar in shape to the sigmoid function but operates in the range of -1 to +1. You must have also noticed how the graph of Tanh is much steeper than that of sigmoid.



The Relu activation function transforms all negative numbers to 0 and maintains the value for positive numbers as it is.

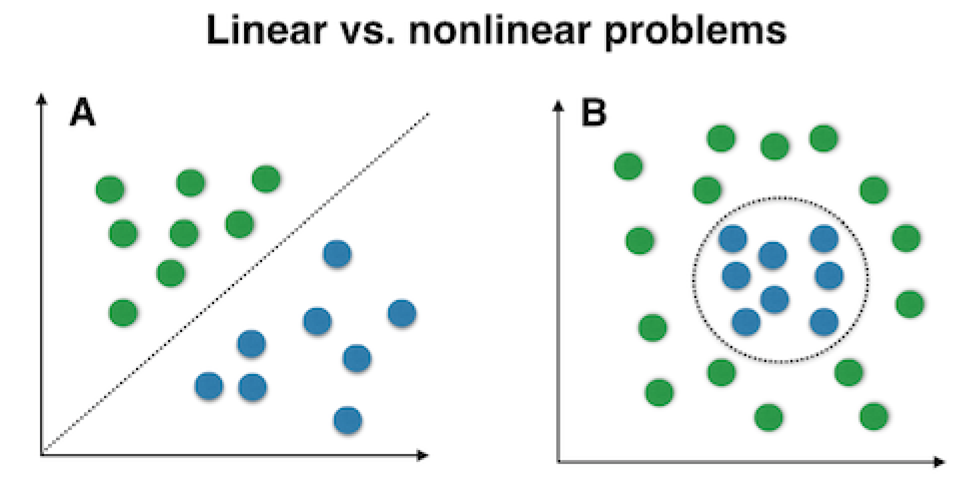


All these operations together help to solve logical problems with fixed parameters. For example, deciding which restaurant to eat at or planning a trip with friends.

Architechture of Neural Network

As you learnt in the previous segment, a network of biological neurons forms the nervous system. Similarly, groups of artificial neurons accumulate to form neural networks. These groups of logistic neurons are generally required when the given data is not linearly separable for classification.

To separate the classes that cannot be separated by a single neuron, we would require a network of neurons.
Another example of non-linear separability can be seen in the diagram below.



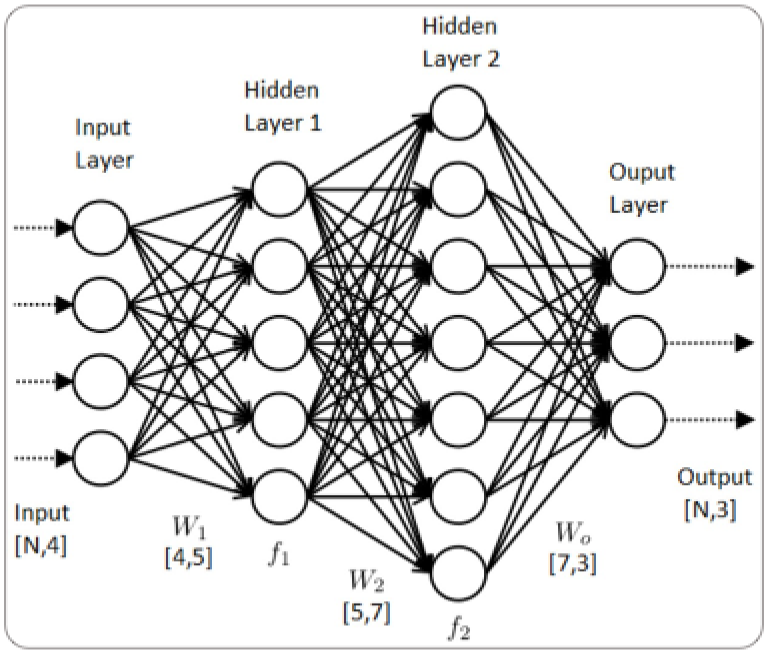
Input Data Matrix can be generally represented as done below, where n represents the number of features and m represents the number of samples. In the Iris data set, the number of features n was 4, including sepal length, sepal width, petal length and petal width.

$$X = \begin{bmatrix} X_{11} & X_{21} & \cdots & X_{n1} \\ X_{12} & X_{22} & \cdots & X_{n2} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ x_{1m} & x_{2m} & \cdots & x_{nm} \end{bmatrix}$$

We designed the neural network that had 4 input neurons because we had 4 features in the Iris data set and 3 neurons in the output layer because we had 3 classes, setosa, virginica and versicolor. The neurons in the hidden layer are arbitrarily decided.
Now, we understand the different components in a fully connected neural network:

- Input Layer
- Weight Matrix (W1)
- Hidden Layer 1
- Activation function of layer 1 (f1)
- Weight Matrix (W2)
- Hidden Layer 2
- Activation function of layer 2 (f2)
- Weight Matrix (Wo)
- Output Layer
- Activation function of output layer (f0)

The dimensions of the weight matrix are deduced based on the number of neurons in the previous layer and the number of neurons in the next layer.
Number of rows of weight matrix = Number of neurons in the previous layer
Number of columns of weight matrix = Number of neurons in the next layer



Forward Pass

You learnt how one input sample goes through the first layer of the network. For an input X, the output of the first layer is calculated in the following manner:

- Input to the first layer: I1in=<X,W1>+b1
- Output of the first layer:I1out=f1(I1in)

The output of the second hidden layer is calculated in the following manner:

- Input to the second layer: I2in=<I1out,W2>+b2
- Output of the second layer:I2out=f2(I2in)

Loss function

The variables inside a neural network are set using a special function known as the loss function. The loss function compares the output given by the neural network with the actual output and sets the variables in order to reduce the difference between these two outputs. This phenomenon of reducing an equation to optimise a set of values is known as *error backpropagation*.

an *optimizer* continually tries new weights and biases until it reaches its goal of finding the optimal values for the model to make accurate predictions. With each *iteratio*n of change, the loss of error or cost (i.e., the difference between the prediction of the model and the actual ground-truth) is incrementally reduced until a sufficient *tolerance* is achieved. The mantra being, "*the larger the cost, the more the error in the neural network*".

The loss function essentially maps an input and output to a real number and, can be depicted as follows:

L(y,^y)=f:(y,^y)→R

The term 'loss function' is commonly confused with the term 'cost function'. The accepted norm is that the loss function computes the loss for a single training iteration, whereas the cost function computes the average loss across the entire training data.

Some loss functions are designed for a classification task, whereas others are designed for a regression task. Regression is used to predict a continuous value in the form of a probability for an integer, for example, predicting a number between a range such as 50 to 200. On the other hand, classification is used to predict a discrete value in the form of a class, for example, segregating the images of fruits according to apples, oranges and bananas. However, loss functions work with the probabilities of labels, not labels themselves.

The loss function is dependent on two factors: predicted output(p) and actual output(y). In classification problem the loss function is as follows:
L(y,p)=−1n∑ni=1[yilog(pi)+(1−yi)log(1−pi)]

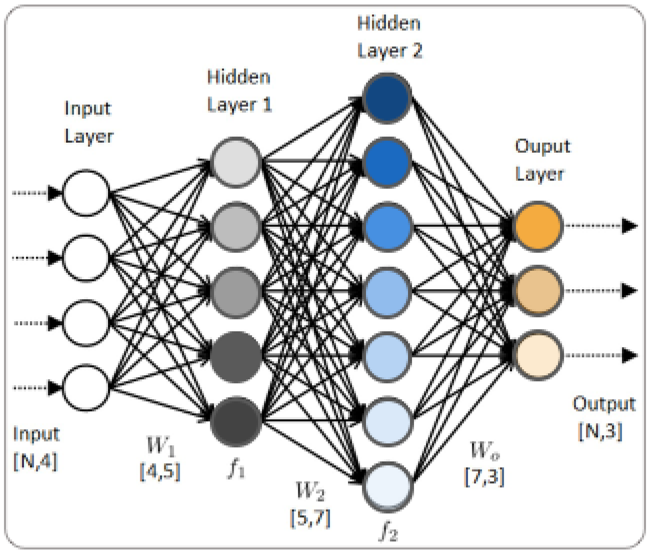
For each iteration, this function calculates the loss between the predicted output and the actual output. These costs are then summed together and negated in order to calculate the final loss.
L(y,^y)=−1/n∑(y−^y)2

Backpropogation

The loss function passes from one layer to the next, backwards through the neural network, optimising the values of the weights and biases. At each step of the way, a different name (Final Loss, Penultimate loss, First layer loss) is given to this ever-changing loss to encapsulate its volatile nature.

The loss decides the incremental changes that need to be made in order to train the model accurately. Little by little, the network starts learning and changing itself until the accuracy is within the tolerance limit.

As you can see in the picture given below, the loss is different for different layers. You can see the difference in the loss based on the colour of the neuron. We can assume that when the colour is light, the loss is less and vice versa.



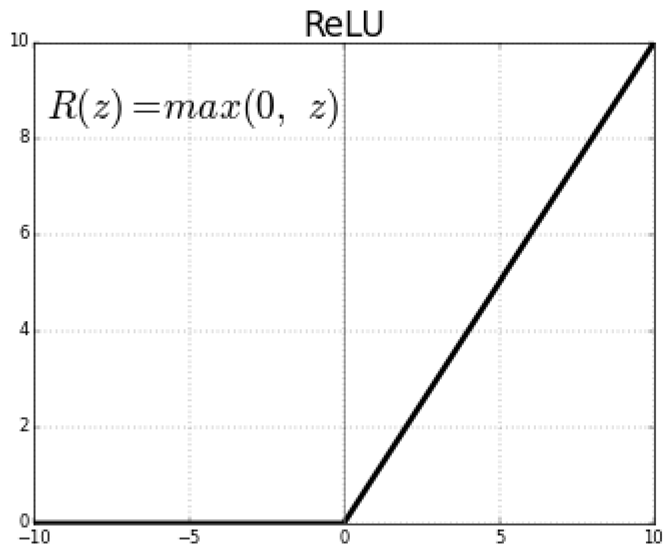
Keras is a powerful open source Python tool built on top of Tensorflow. It can be used to build neural networks. TensorFlow is an end-to-end open source platform for machine learning developed by Google.
Some salient features of Keras include:

- Modularity: It consists of models in the forms of graphs and sequences.
- Data sets: It contains a variety of pre-installed clean data.
- Pre-trained models which can be readily used

Moreover, Keras is a high-level language, which means that you do not have to deal with the nitty-gritty or mathematics of stitching a network together. Infact, any particular type of layer can be called with just one line of code (of course, with the right dependencies).

Looking at the data set will help you decide the basic architecture of the neural network. The input of the net will depend on the features of the input data, and the output will correspond to the number of classes.

The other important aspect that we defined in the sequential function is the activation function.
As we know, the input layer does not have any activation function. The first and second hidden layers have the relu activation function, which can be represented as done below.



The output layer contains the softmax activation function, which is used for classification problems. The softmax function will turn a vector of K values into a vector of K values whose values range from 0 and 1.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

This can be interpreted as probabilities for the classes.

Previously, we saw that the log loss function was used for classification problems. In this case, we have used an equivalent function called the sparse categorical cross entropy as our loss. There are different loss functions for classification. You can read about the same in the link.

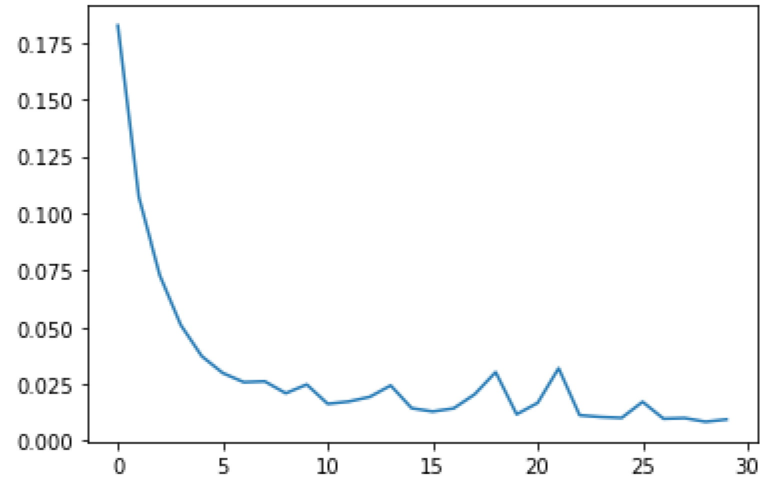
We have also used the most common optimizer, Stochastic Gradient Descent algorithm. Please go through the link to read about the other optimisers available in Keras.

To measure the performance of the network, we have used metrics such as categorical accuracy, which calculates the accuracy for each of the classes separately. This is done as the data is highly imbalanced.

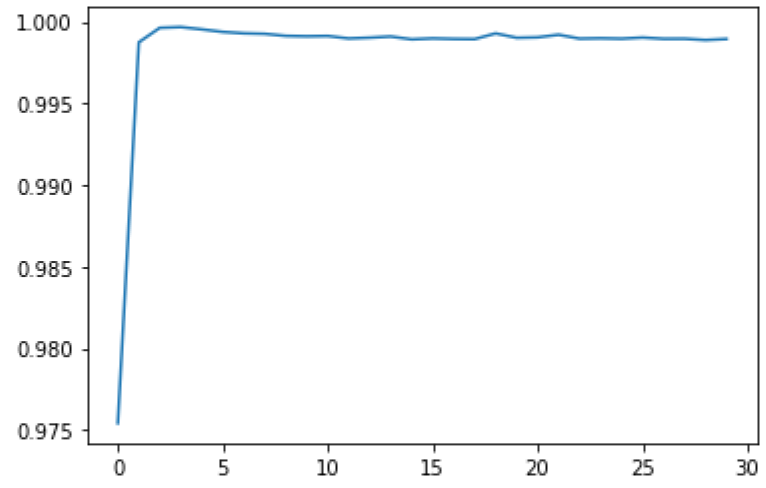
To summarise, we saw the underlying mathematics of our model.The plot model is used to depict the shapes of the weights and biases in each layer in order to view the dimensions at each stage.
Now, we have defined and debugged the model. Let's proceed to the most important step, training the model. The model will be trained using a backpropagation algorithm.

One sampling of the entire batch is known as epoch. As the neural network is trained, the loss of the model decreases and the accuracy tends to increase after every epoch.

Loss vs Epochs



Accuracy vs Epochs



Summary
Although neural networks are a powerful tool for training models, you need to have a clear understanding where it can be used and where it cannot be used.
They can become computationally expensive very fast.

We applied neural networks to solve a fairly simple problem of credit card fraud detection. Next, we will apply neural networks to a widely used Natural Language Processing problem of Sentiment Analysis.

The applications of sentiment analysis are broad and powerful. Therefore, the ability to extract insights from social data or reviews is a practice that is being widely adopted by organisations across the world.
It helps businesses to understand what their customers are talking about and for curating their products to cater to their needs.

The sentiment analysis of social media channels has helped politicians understand where their voters stand on certain issues. It can also be used to analyse market shifts and understand stock prices.

IMDB movie reviews contain test data and train data, each containing positive and negative sentiments.
Since this data is unstructured, we need to preprocess the data before feeding it into neural networks for training. Remember that neural networks can only take numerical input. We need to convert the text data into some numerical representation.

Number of neurons in the input layer = Number of words in vocabulary or Number of columns in the TFIDF matrix
Number of neurons in the output layer = 1
As we are applying the sigmoid activation function in the output layer, we can assume that it has 1 neuron even though it has two classes.
We can define the threshold in the output and define that the review belongs to one class if it is below the specified threshold and vice versa.

We are using the log loss or binary cross entropy function to measure the loss. As the data is not biased, which was the case in the credit card fraud detection case study, we can use the cross entropy function instead of the sparse categorical entropy function.