Artificial Neural Networks 1o1

Keshav Singh

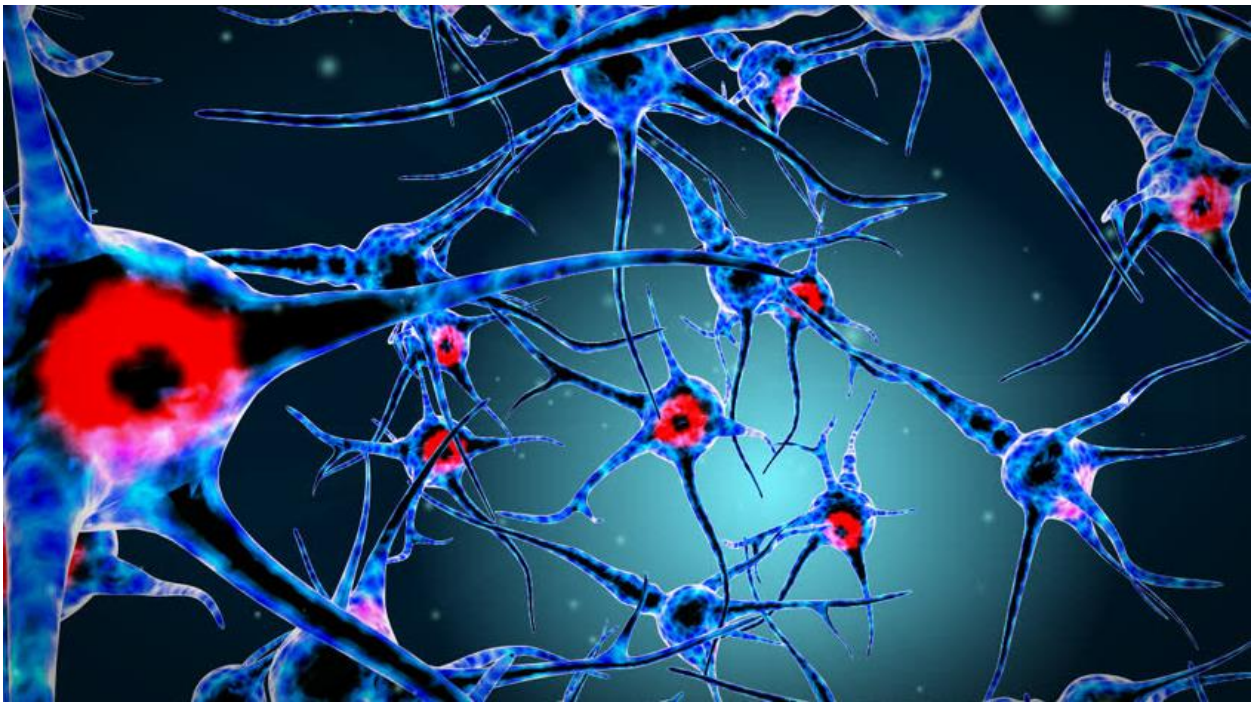
1 GETTING STARTED

Humans have drawn inspiration from nature and surroundings. I feel the greatest true source of our excellence has been our capability to learn and draw various rules and inferences from our habits, traits, behaviors. The fascination of humans to invent something that can think, behave and do things almost as quickly as us, I feel that's the back ground from where artificial neural network (ANN) draws its inspiration from.

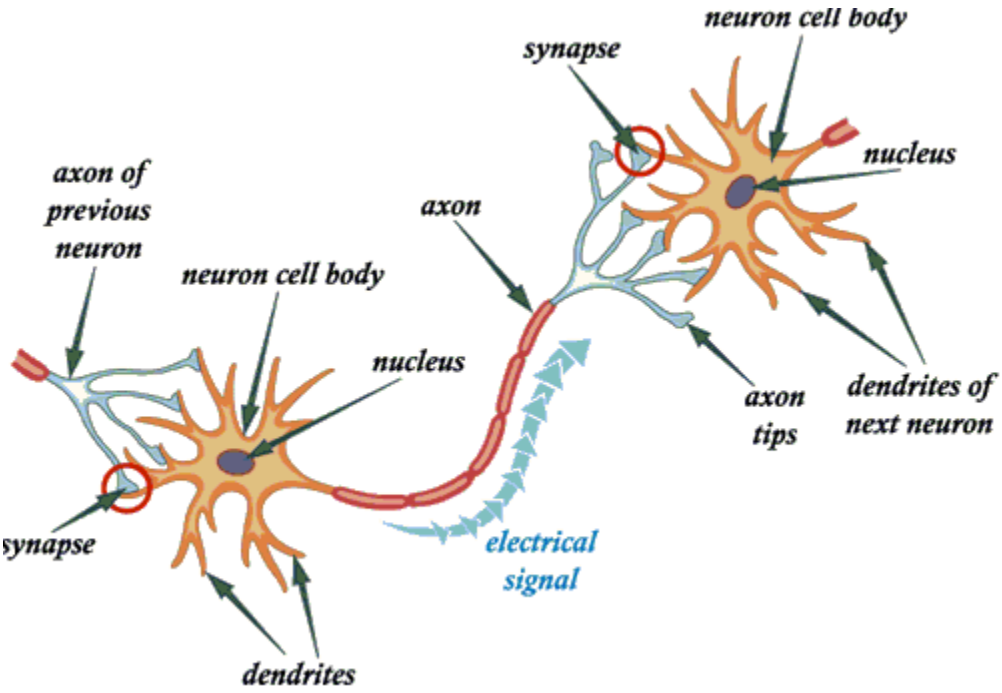
I reckon the attempt of machine learning and AI is to understand perhaps how a robot can learn from us and walk on the streets, or pack and deliver goods, how a driverless car can ride us home while abiding the traffic rules almost as we do. Neural network opens doors of interesting possibilities for humans. Deep learning draws its inspiration from the biology of functioning of a human brain.

I am no biologist yet let me attempt to give you a biological back ground of ANN.

Human brains under a microscope divulges a well-constructed network, below image can be considered a pictorial representation of the same. It's a network of neurons all well connected, transferring signals neuron to neuron for every decision we make or action we humans take.



If we were to zoom into it, we would observe each neuron connection looks like below. Each neuron has a bunch of short little tentacles extending in various directions called dendrites. It also has a long tentacles called axon. The axon ends on branches called synapse which in turn hooks up to the dendrites of the next neuron.



Our brain literally consists of billions and billions of networks of neurons.

<u>How do they function?</u> These dendrites happen to be sensitive to certain kind of chemicals (pardon my knowledge of biology). When there are certain variety of chemical signals around these dendrites they absorb them. When a certain amount of chemical is accumulated at the cell and it gets overwhelmed (let me call it crossing the threshold), it fires. It translates to generating an electrical pulse all the way down the tail – axon, into the synapse and these are picked up as a signal by the dendrites of the next cell. So essentially we have a system which looks as to how much chemical I have around me and if it crosses my threshold I shoot. Boom! These are then picked up by the next cell.

The 2 interesting components we find:

a.) Accumulator ($\sum$)

b.) Trigger ($\int$)

I am not a biologist, so I seek apology in advance if I may have trampled basic biology while attempting to be able to back my mathematical models.

2 REVISITING LOGISTIC REGRESSION

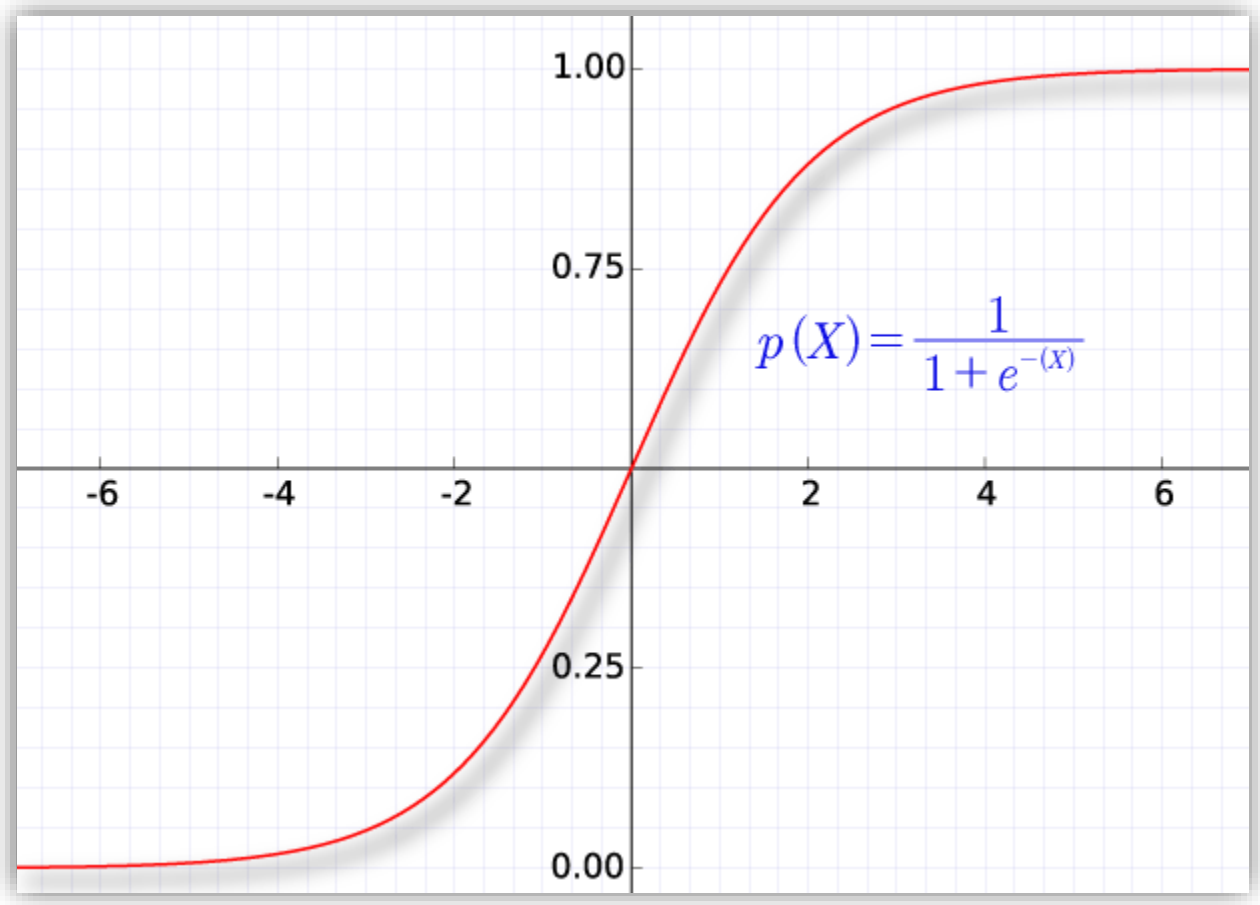
Logistic decision making will not seem unknown to the folks already from ML/AI field of study. Mathematician David Cox formulated the famous logit function, where the probability of occurrence or dichotomous categorical outcome was represented as a linear combination of a function.

Ln(P/1-P) = β0 + β1X1 + β2X2 + .. + βnXn

OR β0 + β1X1 + β2X2 + .. + βnXn = t

P = e^t/(1+e^t)

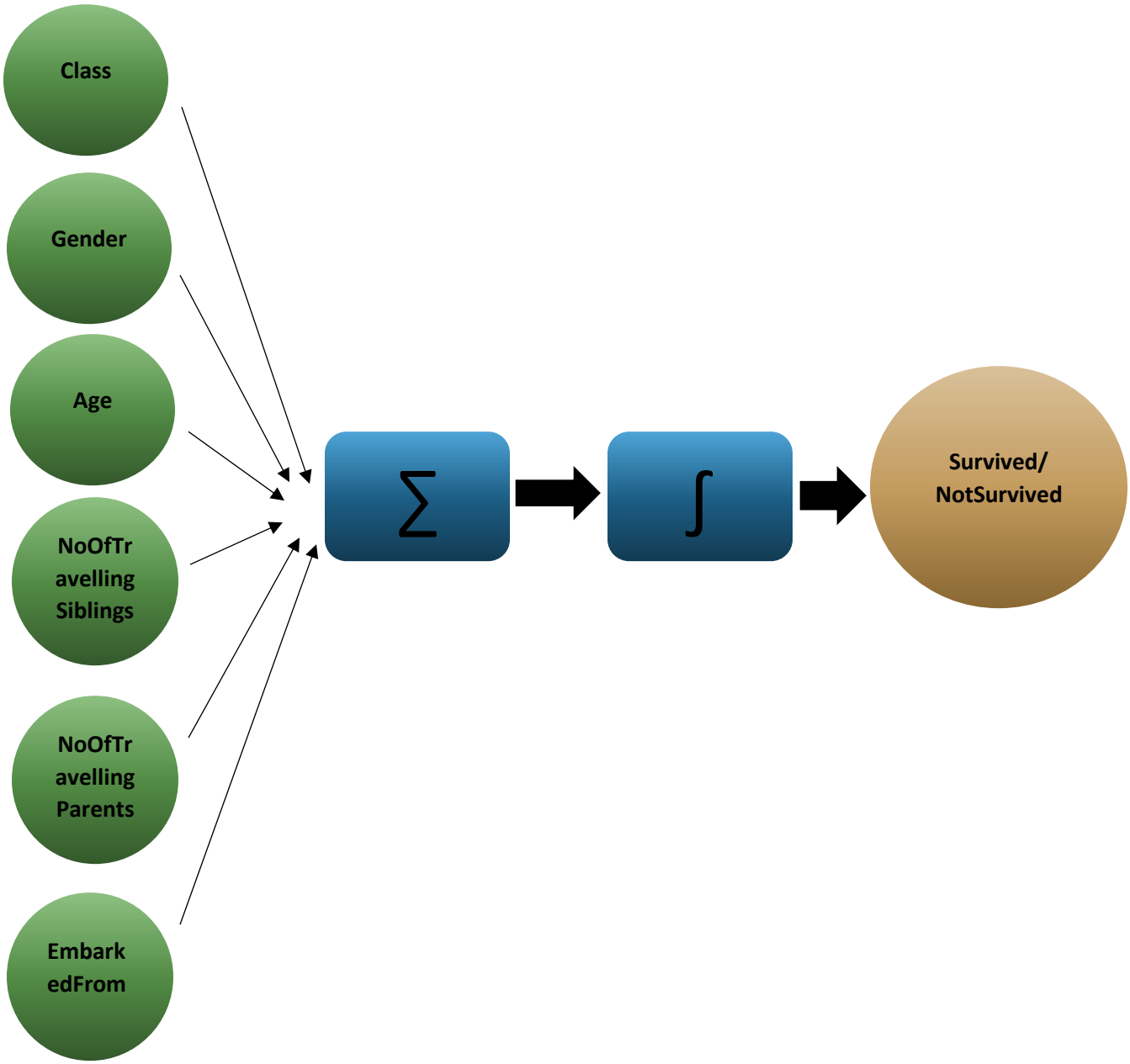
The function has the following distribution and is also coined sigmoid function due to its (∫) shape.



We must already be familiar with the "Titanic" dataset for predicting "survivors" based on predictors

- **Class**
- **Gender**
- **Age**
- **NoOfTravellingSiblings**
- **NoOfTravellingParents**
- **EmbarkedFrom**

Essentially this forms the fundamental of logistic regression.



t = β0 + β1*Class + β2*Gender + β3*Age + β4*NoOfTravellingSiblings + β5*NoOfTravellingParents + β6*EmbarkedFrom
We get inputs from 6 predictors into what is called as accumulator, the β's let's call them weights *W with some bias*. The accumulator function calls for summation of

$$\sum WnXn + B$$

The accumulator ∑ spits out a probability number, the sigmoid function (∫) based on the threshold decides to call the output as "Survived" or "Will not survive". For instance setting a threshold or 0.5 or 50%, when the summation function spits any value greater than 0.5 the sigmoid function calls it results as "Will survive" the titanic disaster. Essentially we can say the sigmoid is triggered only once the threshold of 0.5 is reached by the inputs.

We can draw a clear correlation with the explanation of the human brain cell. Just to reiterate the dendrites absorb the electrical pulse and keep accumulating them until the threshold is reached. An electrical pulse is triggered once the threshold is reached. This forms the basis of the intuition for the neural networks. The decision boundary of the logistic model is a single line which is divided by the threshold point into "Survived" and "Not survived" classes.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

For the function above if *b* were to be a –ive real number the threshold point would be such that *w.x weight times the input* is a number greater than bias *b.* According to our intuition so far this will result in the electrical pulse to be triggered and result 1 as output.

3 PERCEPTRON AND HISTORY

Neural networks is quite an interesting area of machine learning, simultaneously one of the oldest and newest areas of research. Late in 1940 researchers tried to model the human brain. We have revisited the logistic regression to be able to infer the intuition of perceptron. Basing perceptron researchers solved a number of problems. But soon it turned out that the perceptron was limited in capabilities to be able to solve only certain kinds of problems (linear classification problems to be specific.). It couldn't solve simple things like solving a XOR problem. Soon after that the research in neural networks basically died. With a lot of hype around it, turned out it couldn't solve basic and simple XOR problems.

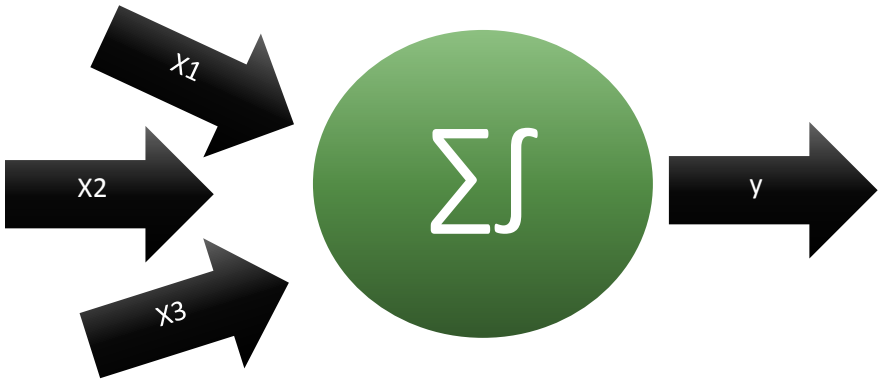
They were re-born, yes in 1980's researchers figured out that putting multiple perceptron together into a network could solve complex classification & prediction problems. The breakthrough came with the development of backward propagation, symmetric networks. This led to a great deal of excitement. These models were now able to solve all kinds of learning problems but the learning needed time or compute power. The computing power

with only as much in the period, neural networks were unable to evolve and flourish to say the least.

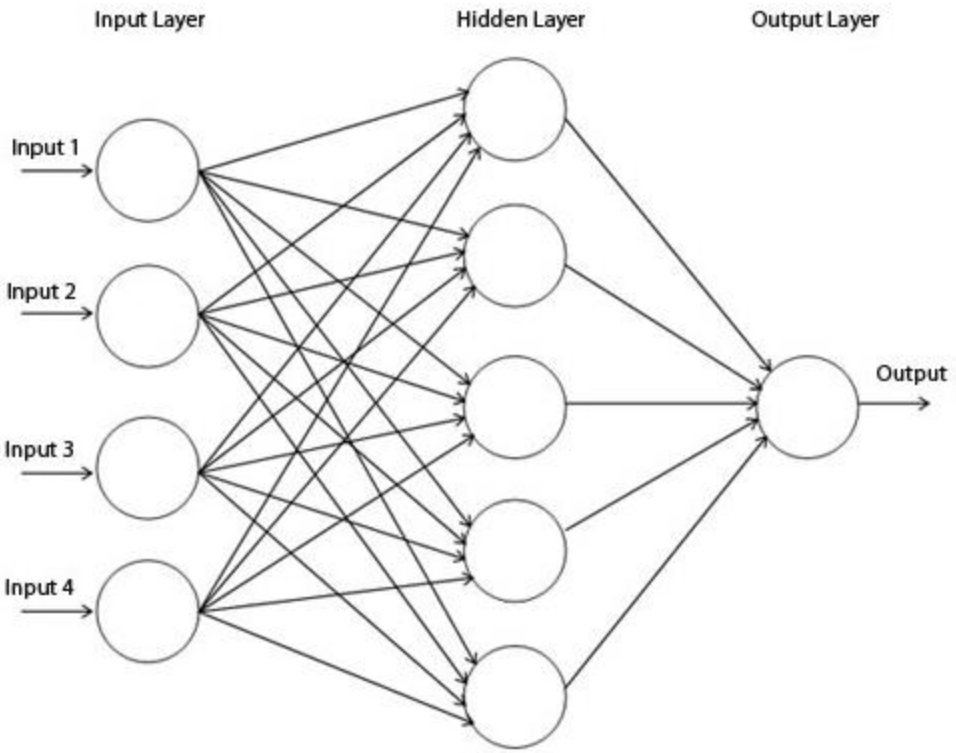
5 years back with cloud computing, big data and GPUs of processing power making a revolutionary entry into the technological world. Neural networks made a re-entry the 2nd time. It's now the back bone of all the research for some of the very complex ML problems right from driverless car, speech and image recognition.

4 ARTIFICIAL NEURAL NETWORK MODEL

The basic unit of ANN is a perceptron or single neuron.



The output is just the squashed function of the summation of inputs. It would literally mean the linear classifier or the output from the logistic regression. Or conversely the logistic classifier is neural network with 1 neuron.
But when we begin to arrange a layer of neurons together into a network we get a neural net. Especially when we have the inputs feeding into several neurons and they form inputs in turn to neurons in one direction we get a feed forward neural network.

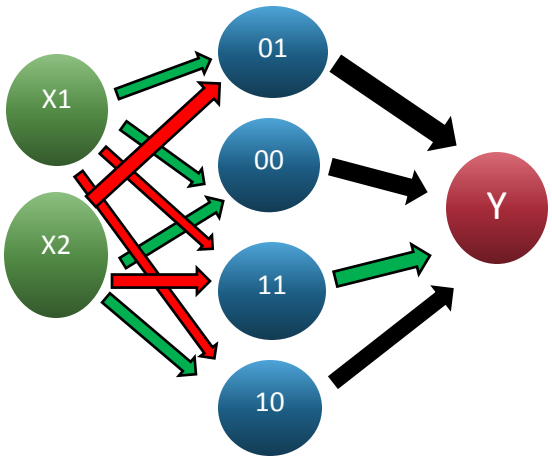


They are used for classification or regressions. It's called feed forward because the information from 1 layer of units is passed to the next layer of units and then they compute and pass on until the output layer. The output classifies or regresses the class label or value respectively based on the weights and biases at each layer neurons. The layers if more than one between input and output conventionally are referred as deep hidden layer.

Solving AND with Neural Network –

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

For an input of 2 attributes and a hidden layer of 2^(inputs) i.e. 4 let's build our first NN.



The NN have quite a representational power, a NN with a single layer we can represent any bounded continuous function to quite a degree of accuracy. And to start with a slightly simpler Boolean function we can represent virtually any Boolean function into a neural net with 100 % accuracy.
The AND Boolean function case for 2 inputs can be solved with a hidden layer of 4 nodes. We can represent an input value of 0 as GREEN and 1 as RED. The network with the Weight vectors of W1, W2, W3 and W4 on to output Y will optimize the solution for AND gate.

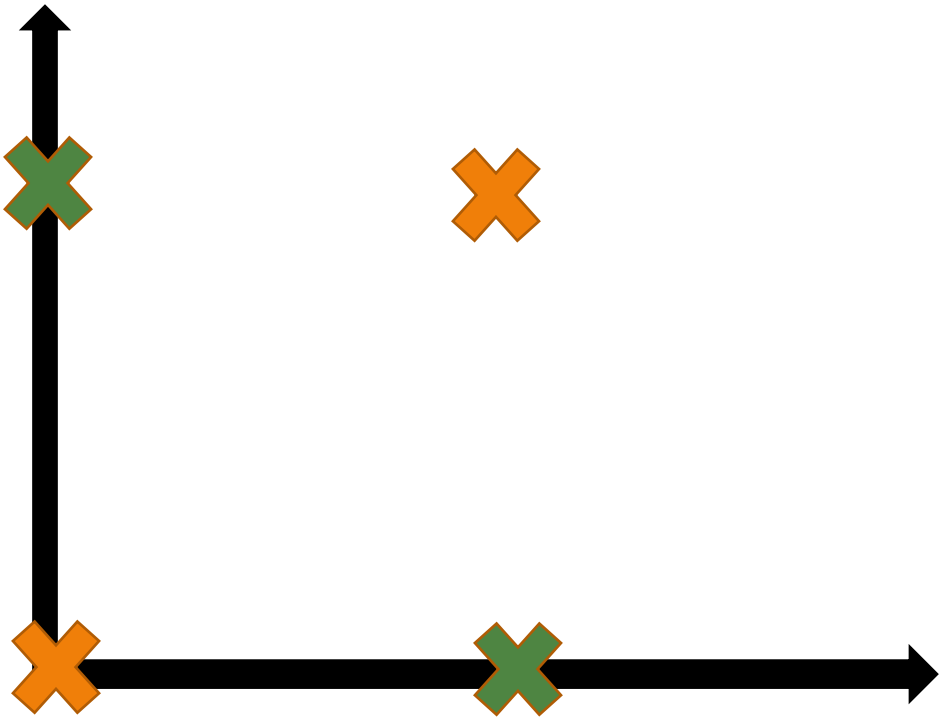
And this network can not only solve for AND but for *any Boolean function*. Now that's quite a statement and power of NN. The hidden unit here encode all possible combinations for 1s and 0s and solves for the desired function.

5 XOR GATE PROBLEM – SOLVED BY NEURAL NETWORK

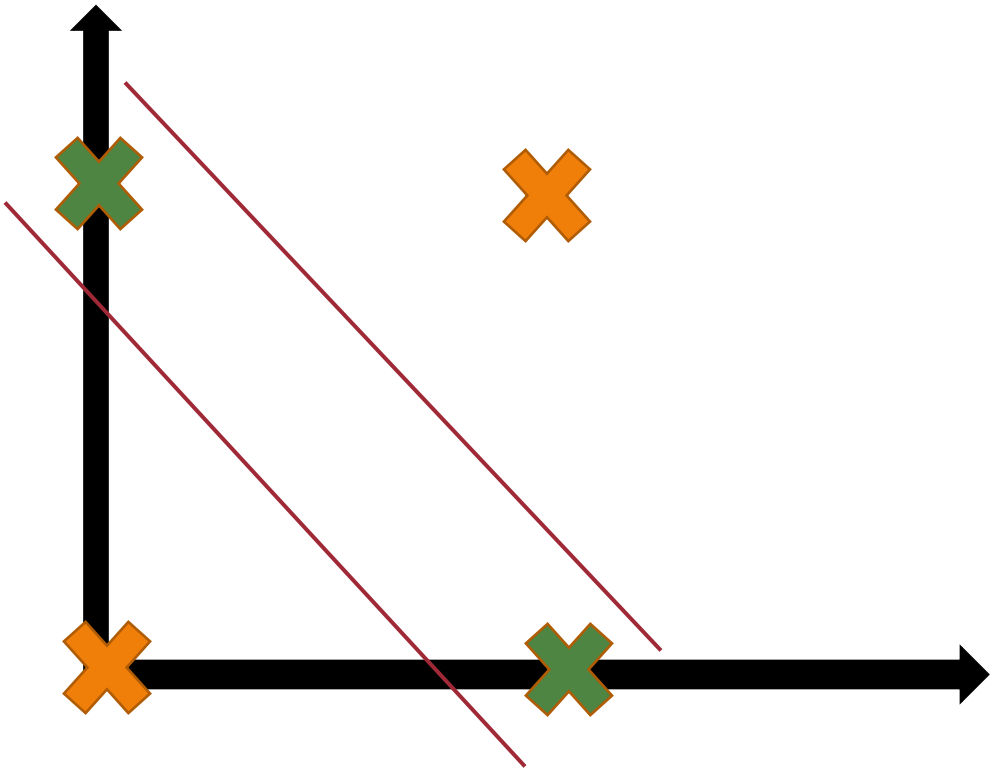
XOR gate is a digital logic gate that sets the signal as high only when 1 of the inputs are high. Compared against an OR gate XOR is also called as TRUE OR.

A	B	A OR B	A	B	A XOR B
1	1	1	1	1	0
1	0	1	1	0	1
0	1	1	0	1	1
0	0	0	0	0	0

A linear classifier can't solve this. To visualize this on a graph, it would look something like below. Now a straight line cannot classify one from the other class label.



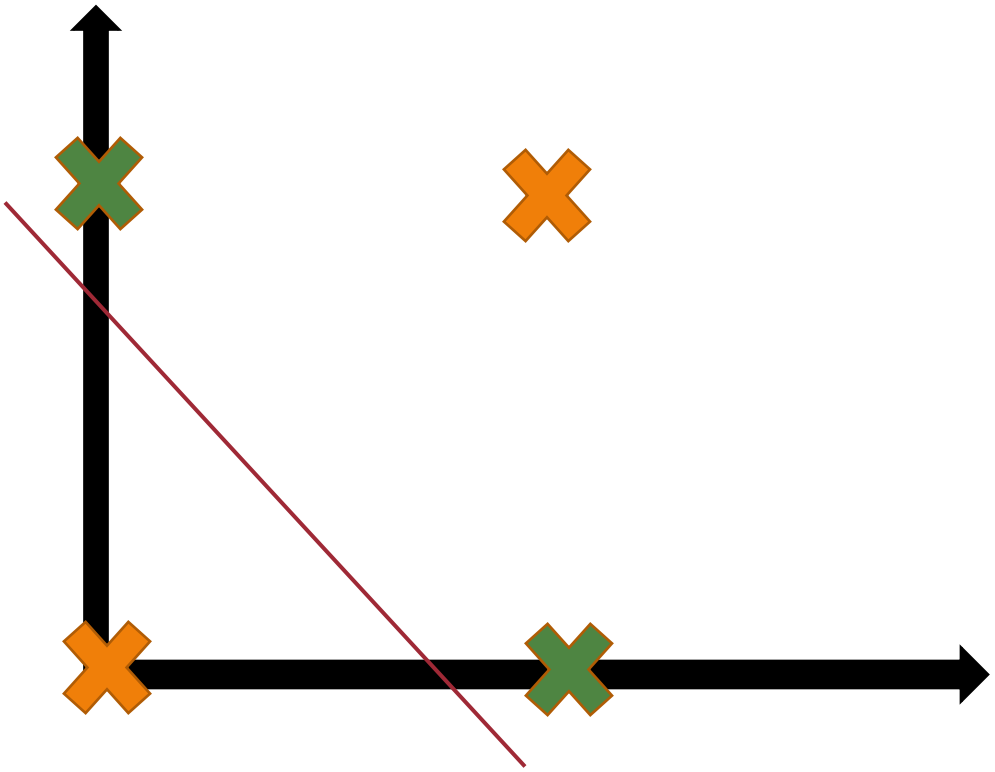
To do so we need not 1 but 2 lines. And hence we need 2 layers of perceptron.



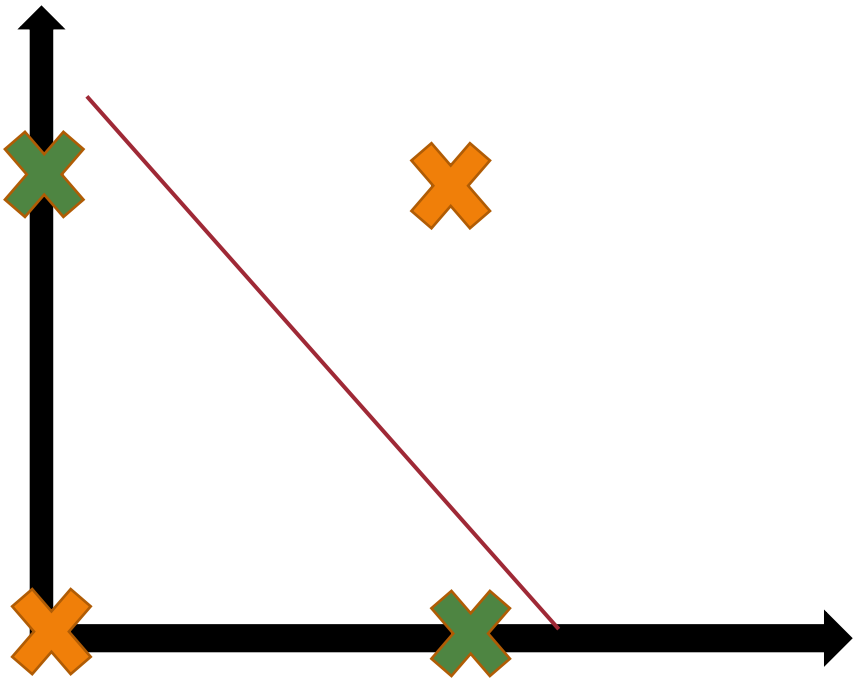
There is no way to solve this with a single neuron, however with 2 layers neurons we can solve this problem.

<u>Layer 1</u>

A	B	A OR B
1	1	1
1	0	1
0	1	1
0	0	0



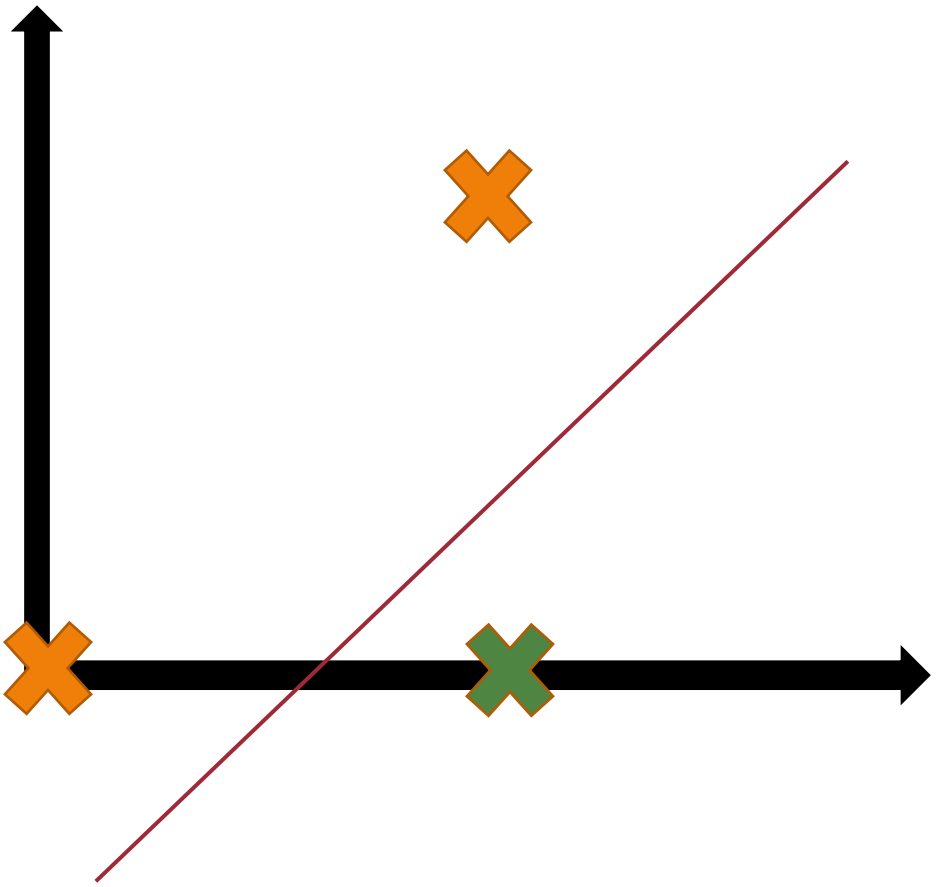
A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

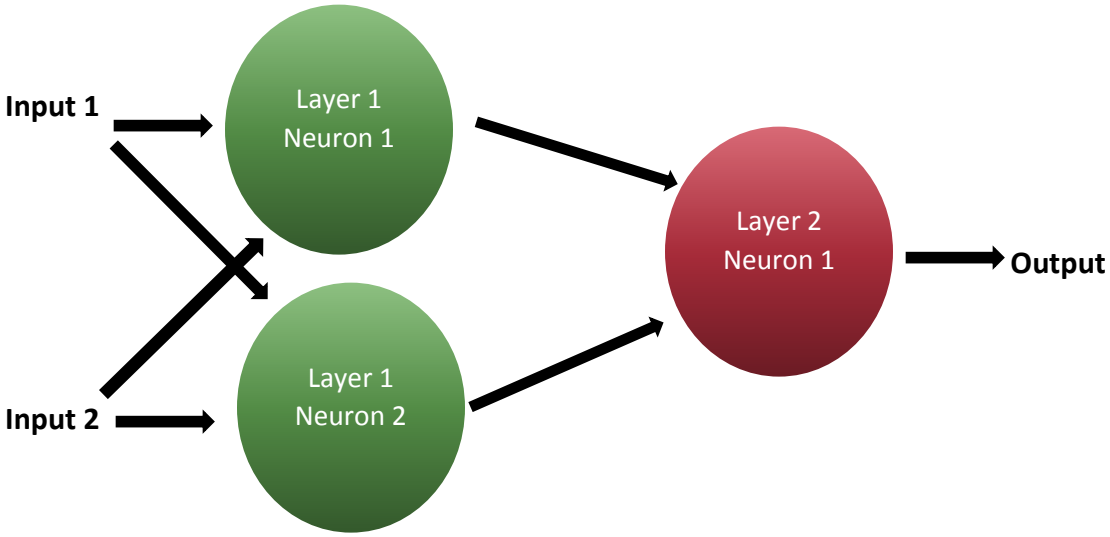


Layer 2

On layer 2 interestingly we redefine our axis and plot the outputs from layer 1 neuron 1 and the output from layer 1 Neuron 2. Interestingly we began with solving a 4 point classification now leaves us with only 3 points on the board. As (1, 0) are common outputs from both 1 & 2 neurons.
At the second layer the classification becomes easy as the linear classifier will now be able to separate them into 2 classes.

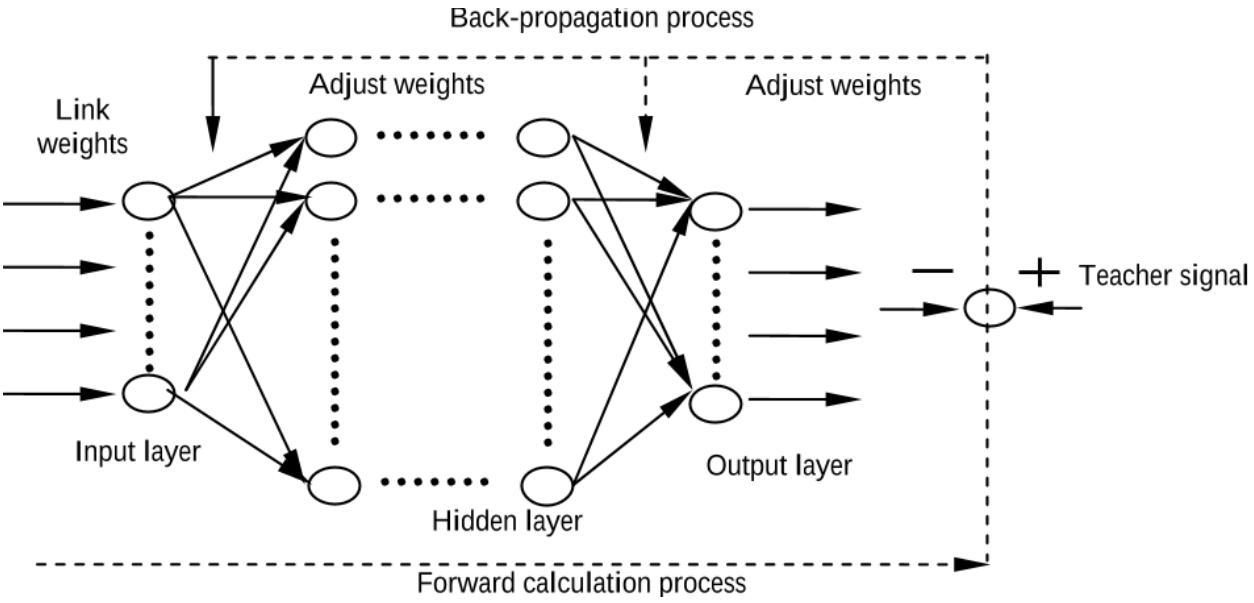
Output From Layer 1 Neuron 1	Output from Layer 1 Neuron 2	Output Layer 2
1	1	0
1	0	1
1	0	1
0	0	0




Input 1
Input 2
Layer 1
Neuron 1
Layer 1
Neuron 2
Layer 2
Neuron 1
Output

6 BACKWARD PROPAGATION

Back propagation can be considered as an optimization problem. Basically it entails what the hidden units (neurons) should look like to achieve the results closely explaining and/or learning from the training data. The figure briefly explains the back propagated neural network with hidden layer and nodes. It's an online algorithms which basically starts with a set of weights for each units and receives the inputs and the output data point at a time and then computes the values for each node on the hidden layers. It results in an output Y, this output may be quite different to the Y* (the output in the training dataset). Hence Y-Y* would be the Error $\dot{\varepsilon}$. In this each case it's just running the sigmoid and forward feed.



$\dot{\varepsilon}$ Error Y-Y* on the output will now be pushed back to adjust or optimize for the backward propagation algorithm, to come as close possible to the true label or output. This acts as synthetic training example for each layers backward.

Derivative solves this optimization problem. Basically $\partial\dot{\varepsilon}/\partial k$ where k is the node on a hidden layer tells whether the weight needs to be higher or lower to the current. On each layer and node, based on the error $\dot{\varepsilon}$, the derivative of the error w.r.t the weight tells us whether the value needs to be increased or decreased and overall this just keeps relaying until we solve over iterations and get an optimized value as close to the output in the training data as possible. The final weights determine the model and perform the predictions.

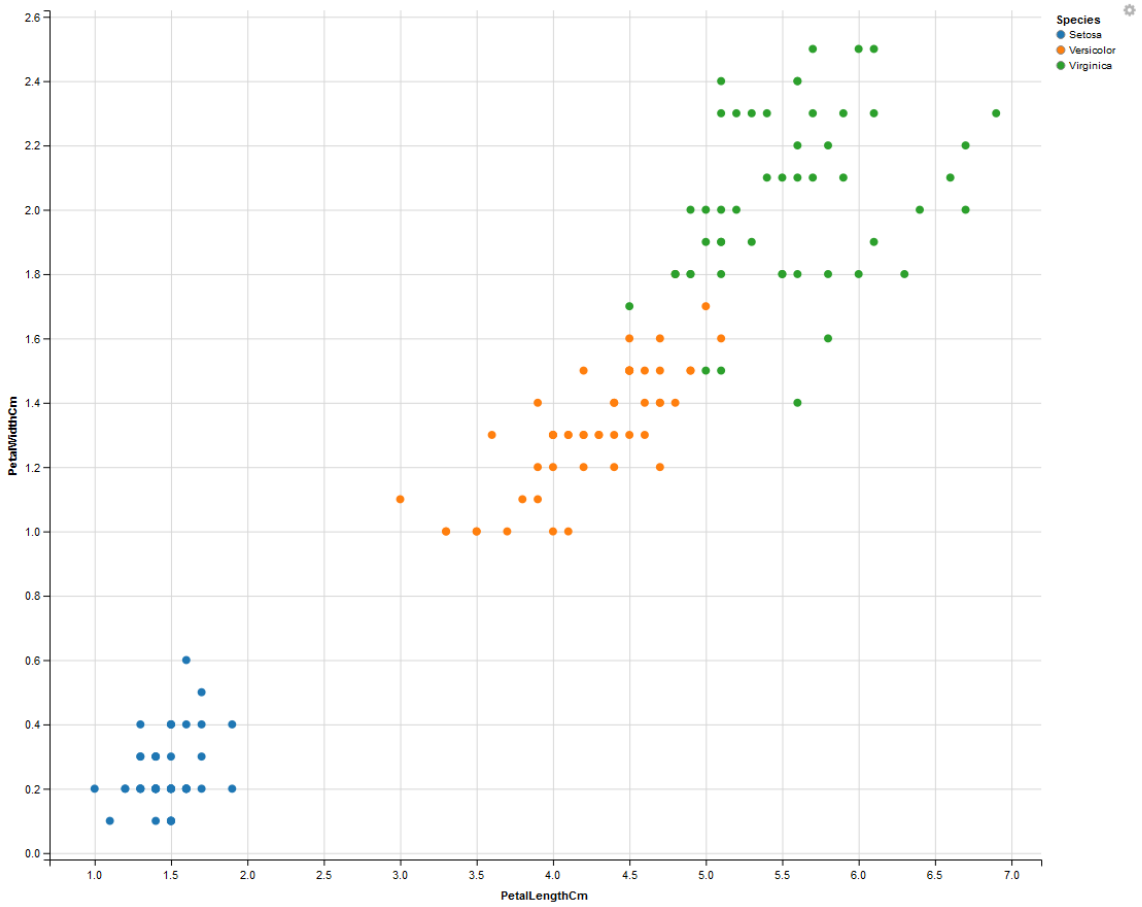
For complex problems the training time could be quite lengthy and GPUs of compute may be required to solve the massive scale problems.

7 MULTICLASS NEURAL NETWORK

A striking question arises, how do we solve the multi-class prediction problems? Well in that case let's pick up the famous IRIS dataset. Just to revisit we have a dataset of 3 species of IRIS flowers namely "Sentosa", "Virginica", "Versicolor" and their sepal length and width and petal length and width. A sample of 5 records looks as below.

SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5.0	3.6	1.4	0.2	Setosa

Plotting the petal details on R the graph looks as below.



To be able to solve this classification problem and train a Neural Network to be able to predict species based on sepal and petal details we flatten the results.

The code to do so in R

```
irisdf<-cbind(iris[,2:5],class.ind(iris$Species))
head(irisdf)
```

```
  SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm Setosa Versicolor Virginica
1           5.1          3.5           1.4          0.2      1          0         0
2           4.9          3.0           1.4          0.2      1          0         0
3           4.7          3.2           1.3          0.2      1          0         0
4           4.6          3.1           1.5          0.2      1          0         0
5           5.0          3.6           1.4          0.2      1          0         0
6           5.4          3.9           1.7          0.4      1          0         0
```

We flatten the output by transforming the Species attribute with the output species classes. For the records with output as "Sentosa" the data is flagged as 1 and respectively is for the rest.

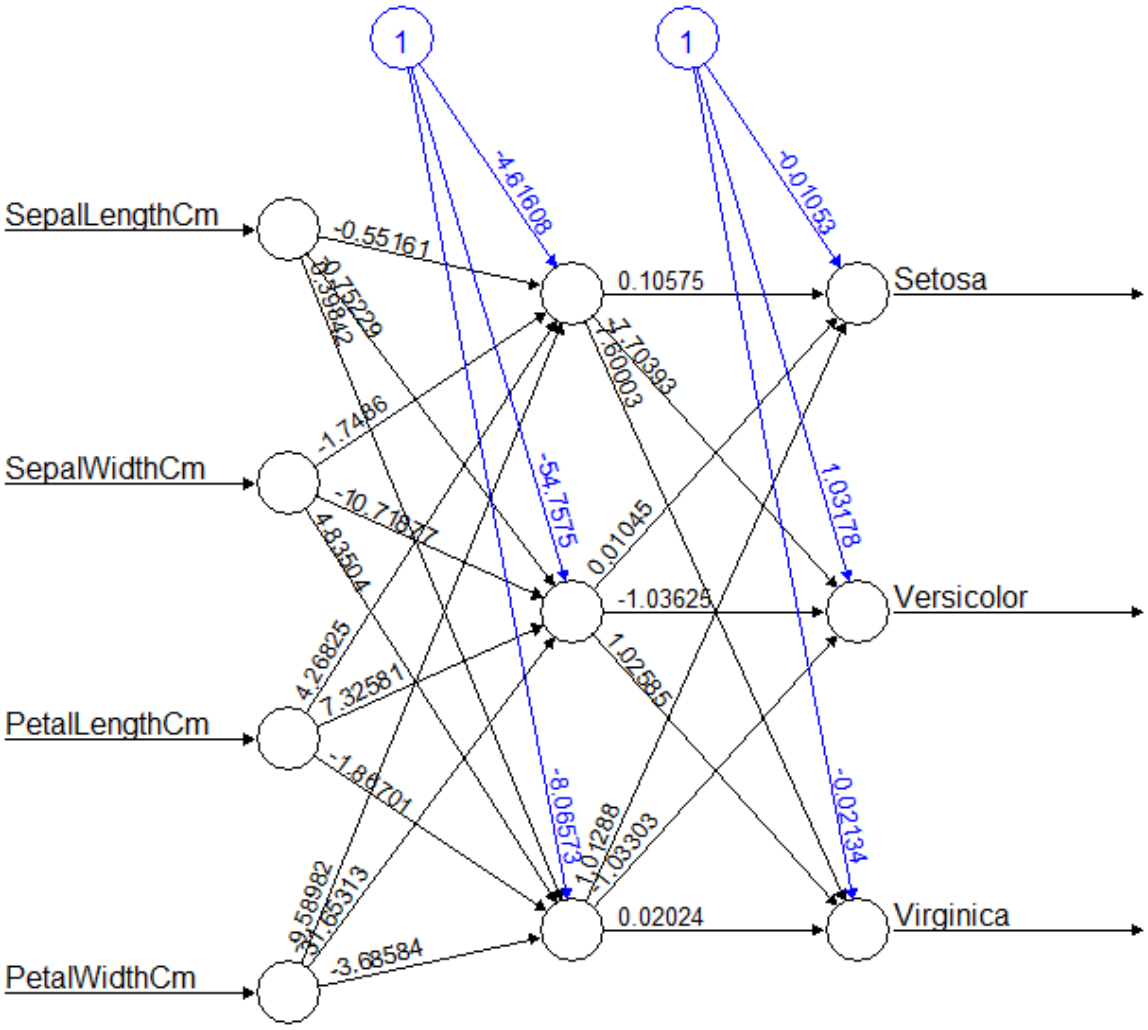
```
set.seed(2)
proportion <- 0.80 # Set to Split
index <- sample(1:nrow(irisdf), round(proportion*nrow(irisdf)))
train_iris <- irisdf[index, ]
test_iris <- irisdf[-index, ]
head(train_iris)
head(test_iris)

NROW(train_iris)
NROW(test_iris)

iris_n<-neuralnet(Setosa + Versicolor + Virginica ~ SepalLengthCm
             +SepalWidthCm+PetalLengthCm+PetalWidthCm
             , train_iris ,hidden = c(3))
plot(iris_n)
```

Next we sample the data into training and test sample in 80:20 ratio and train neural net with 1 hidden layer of 3 nodes.

Upon training the plot looks as below.



Error: 0.434176 Steps: 16243

As we observe the trained neural network has 4 inputs of the attributes and 1 hidden layer for their respective weights (indicated in black) and biases (indicated in blue). And output predictions.

Based on a set of input the NN basically predicts a set for 3 values i.e. the output species (Sentosa, Versicolor or a Virginica) chances for set of an input values. A sample predictions code and output.

```
pred_test<-compute(iris_n,test_iris[,1:4])

predtestResult <- pred_test$net.result
predtestResult<-as.data.frame(predtestResult)

colnames(predtestResult)<-c("Setosa","Versicolor","Virginica")
```

```
       Setosa       Versicolor         Virginica
1.0001246575  0.0002281266586 -0.00035598506666
1.0012668178 -0.0012541517470 -0.00001569593707
0.9895750610  0.0093682644634  0.00105299853778
1.0009436883 -0.0017105580655  0.00076400422076
1.0023339198 -0.0012561689526 -0.00108095670459
0.9990615592 -0.0030469978489  0.00398320900347
```

For each record the max value is considered the winner or the predicted class.

```
result_IRIS <- colnames(predtestResult)[apply(predtestResult,1,which.max)]
result_IRIS<- as.data.frame(result_IRIS)

IRIS_testdataset<-colnames(test_iris[,5:7])[apply(test_iris[,5:7],1,which.max)]
IRIS_testdataset<-as.data.frame(IRIS_testdataset)
table(result_IRIS$result_IRIS,IRIS_testdataset$IRIS_testdataset)
```

```
            Setosa Versicolor Virginica
 Setosa         11          0         0
 Versicolor      0          6         1
 Virginica       0          0        12
```

The scoring of the model against the 20 % of the test dataset indicate that there has been an accuracy of 29/30 or 96.67 % accurate with only 1 record being incorrectly labeled or predicted.

Wrapping up, with the humongous leap in the computing capabilities, neural network and deep leaning opens up quintessential possibilities of very interesting inventions ahead. As Theo Jansen aptly put –

"The walls between art and engineering exist only in our minds."