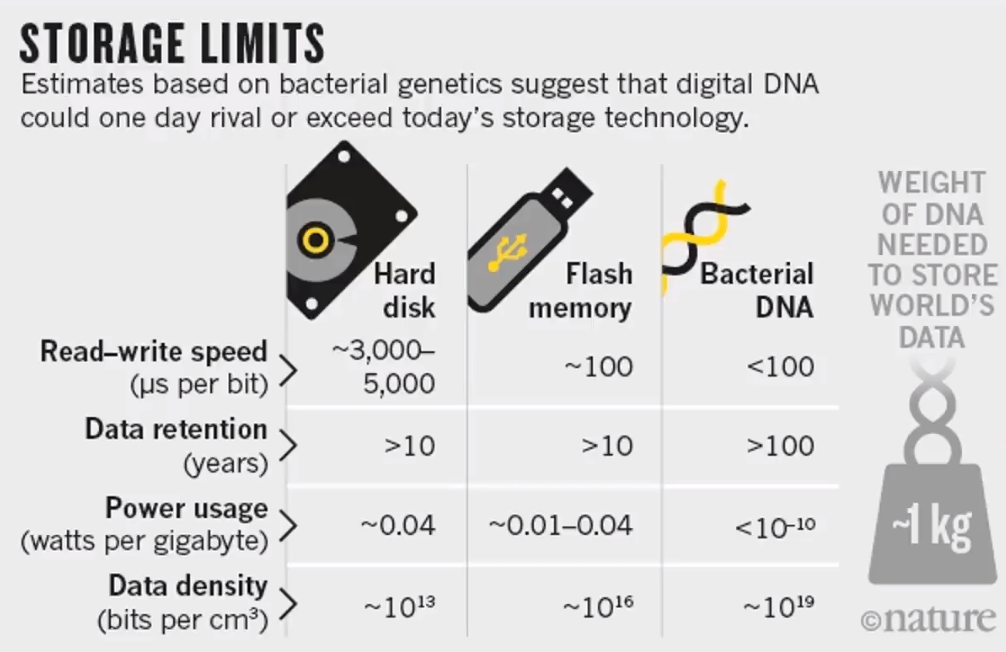
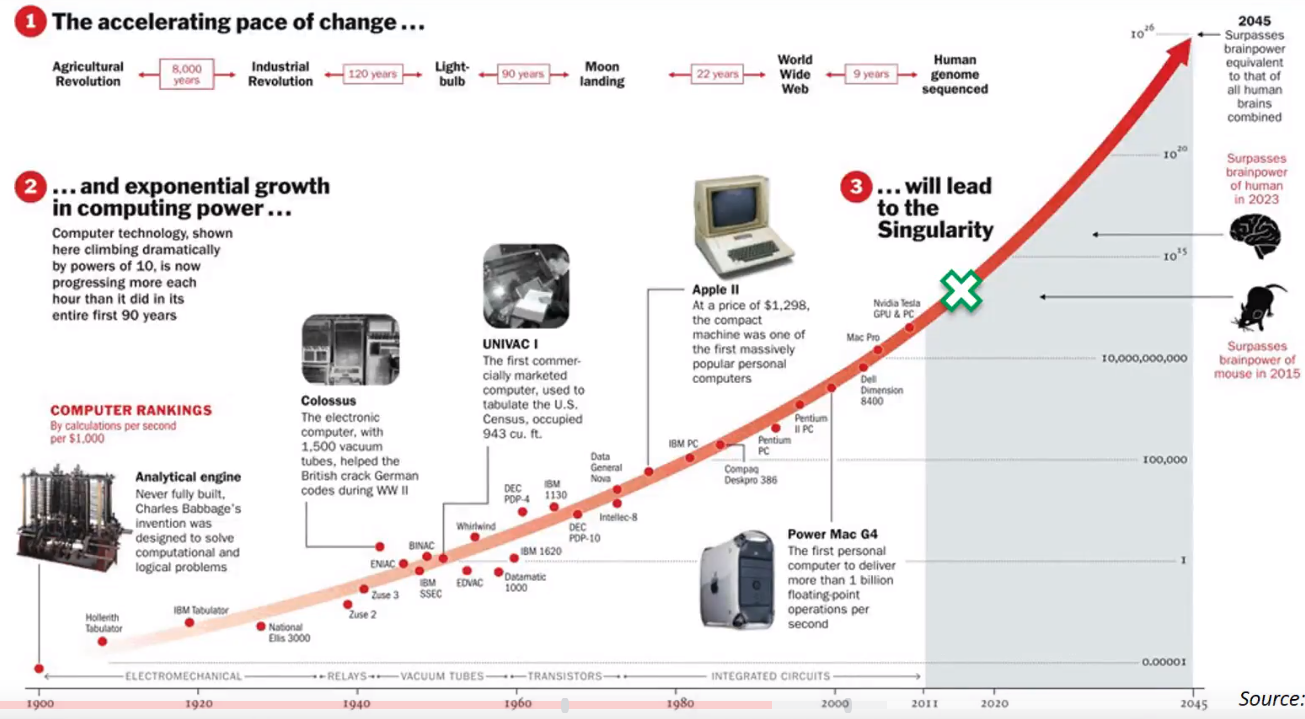
**Neural Networks –**

Neural Networks were doing rounds in 1970s but there was no proper storage technology to support it. As well as, there was not enough data to train sophisticated neural networks.

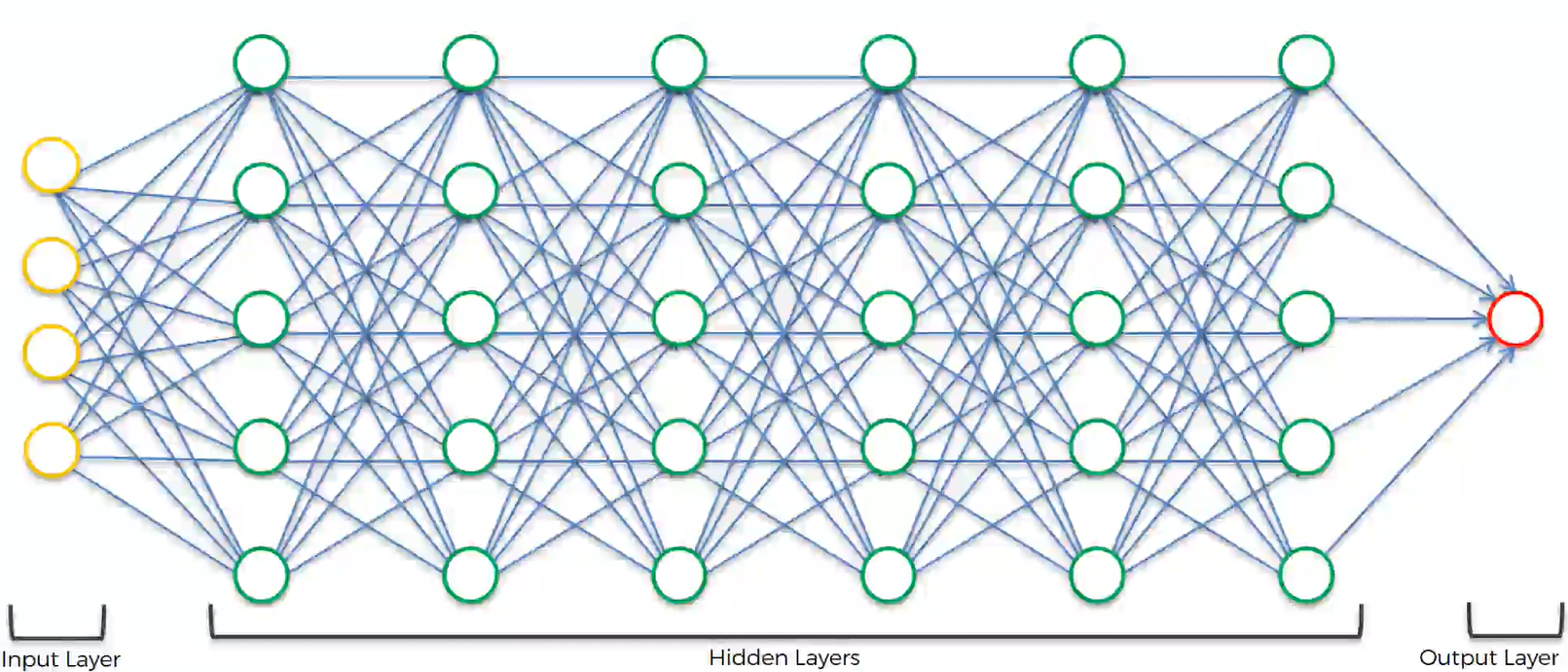


Also processing capacity of computers is evolving – Moore’s law



Deep learning – Trying to mimic and recreate how human brain works.

Neural Network Structure – It has an input layer, output layer and hidden layers. Input layer neurons are connected to hidden layer neurons and hidden layer neurons to output layer neurons. When there are many hidden layers involved, then it is called deep learning.

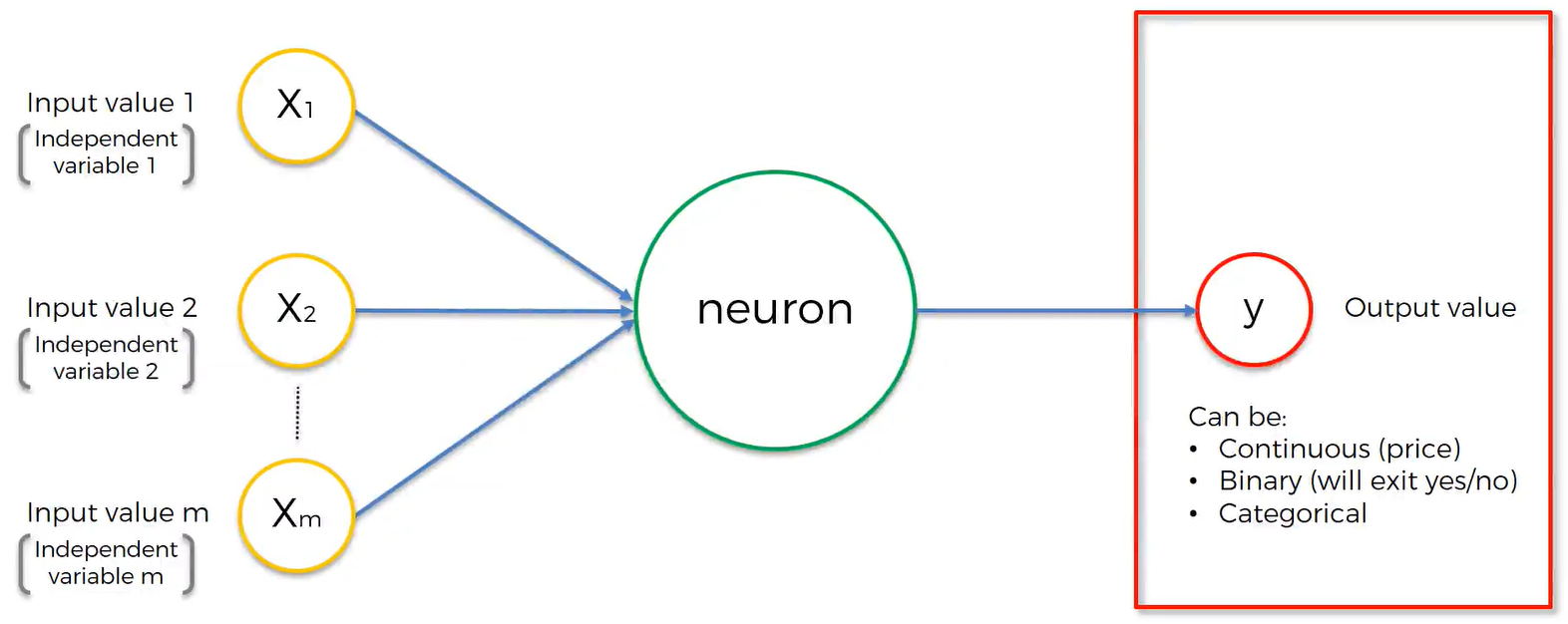


A list of cost function used in Neural Networks alongside Applications, click [here.](https://stats.stackexchange.com/questions/154879/a-list-of-cost-functions-used-in-neural-networks-alongside-applications)

Neural Network with [Gradient Descent](https://iamtrask.github.io/2015/07/27/python-network-part2/). (Neural Network in 13 lines of Python Part -2 Gradient Descent)

[Backpropagation Algorithm](http://neuralnetworksanddeeplearning.com/chap2.html) (Free online book Neural Networks and Deep Learning by Michael Nielsen)

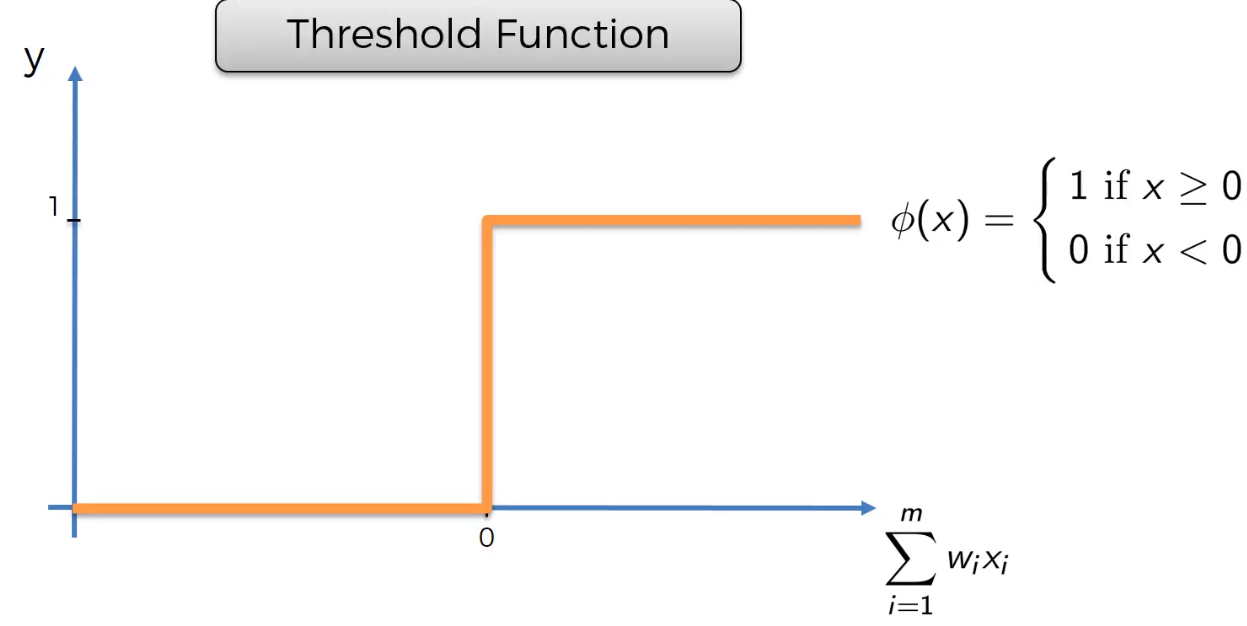
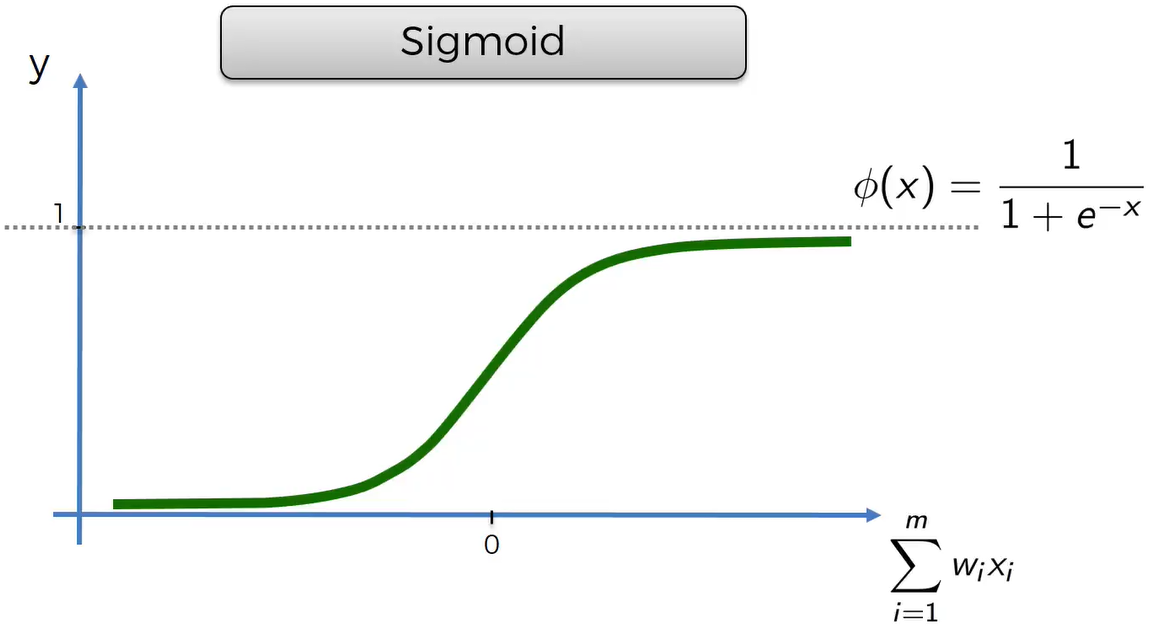
1. Neuron –

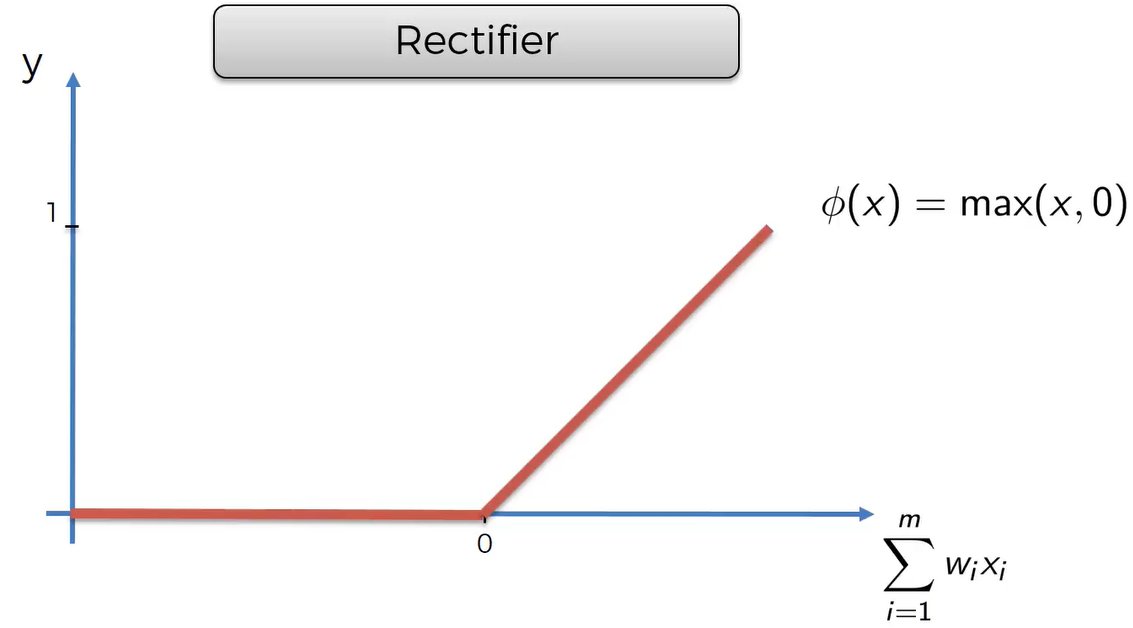
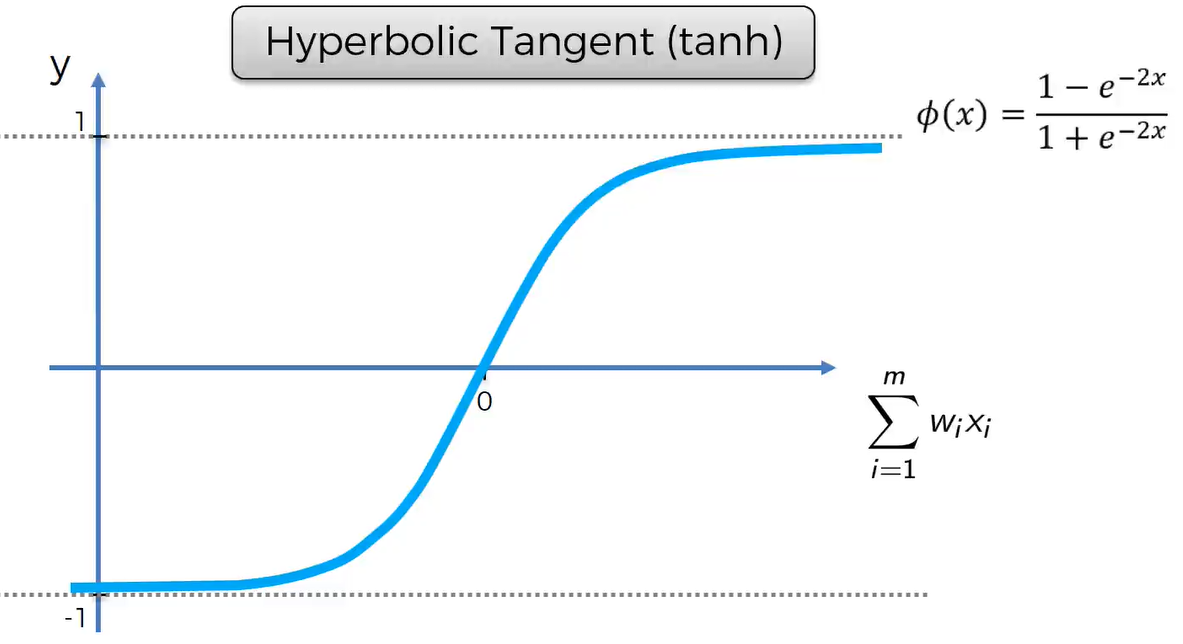


This input layer is for one observation (dealing with one row at a time) and the output is for the same observation. It deals with different attributes of a single observation. Weights are how ANN learn. By adjusting the weights for every single case – what feature/ signal is important and what is not important. When you are training ANN, weights are the one that get adjusted through the training process. In the neuron inside the hidden layer, first all the input values multiplied by the weights get added up and then you apply a activation function on the summation (Which is applied to the whole layer).

2. Activation Function

There are 4 distinct types of activation function –

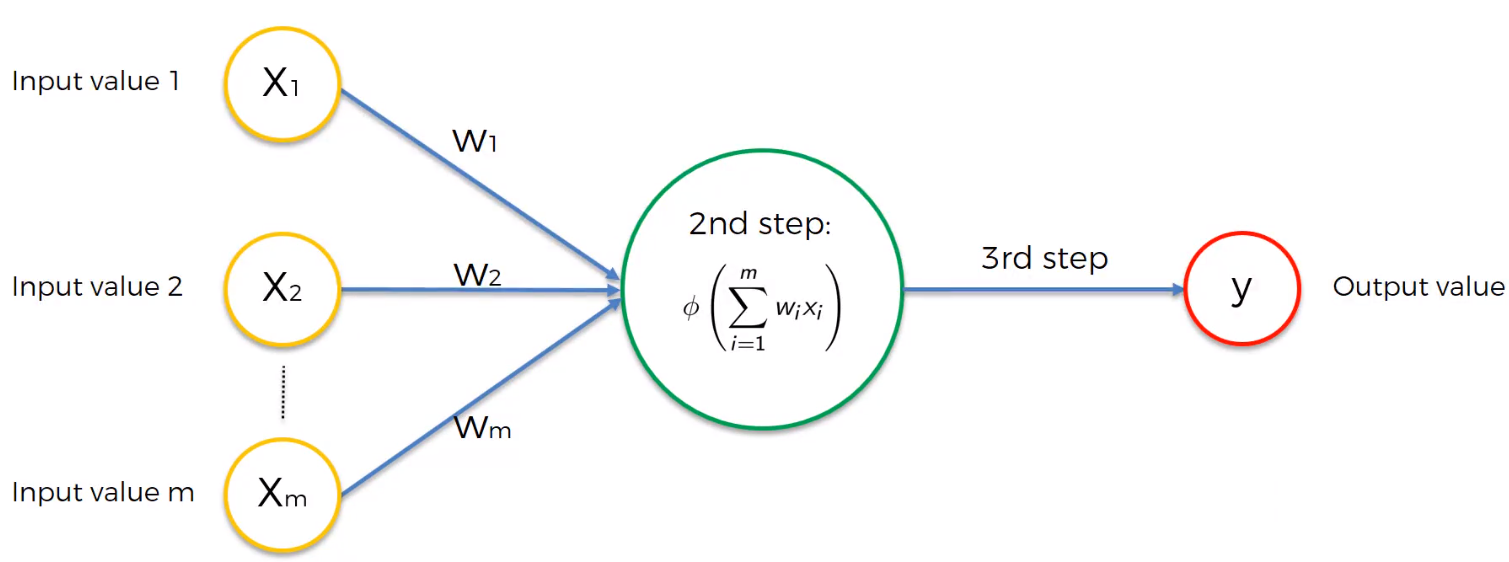
 

If the dependent variable is binary, which Activation function would you use?

We can use either use Threshold function (It is between 0 and 1) or Sigmoid Function (It gives the probability of the dependent variable being 0 or 1).

Usually, we apply Rectifier function in the hidden layer and sigmoid function in the output layer.

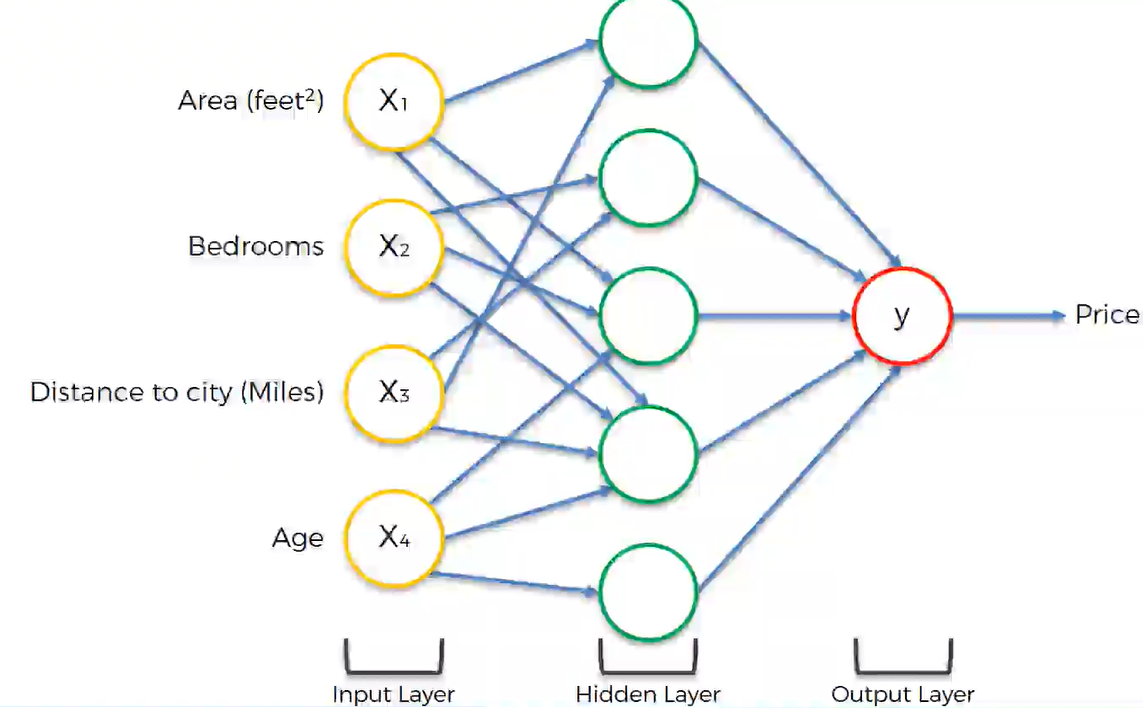
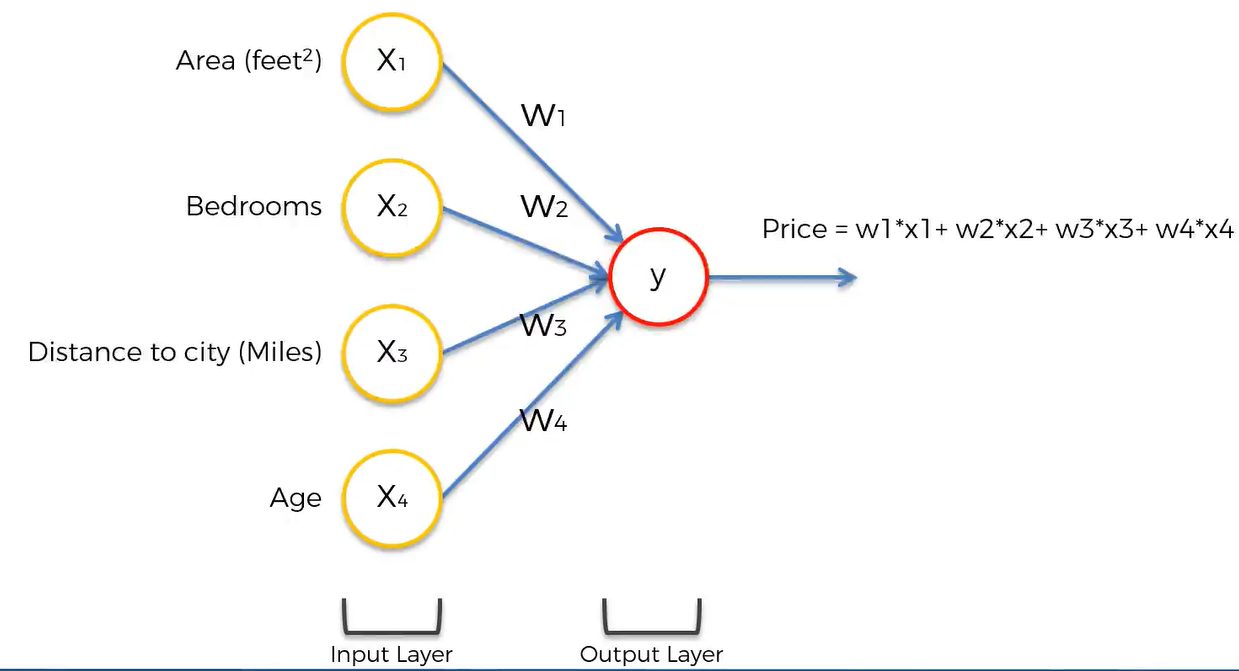


3. Working of Neural Networks with example

Consider Property value dataset with each observation having 4 input parameters – Area, Bedrooms count, Distance to city and Age of the property. Basic Neural Network – Input layer and output layer with no hidden layers. Output of the ANN is the price of the property.

In NN, the hidden layers give lot of power and accuracy. Consider the neural network is already trained.

All the neurons in input layer to each neuron in hidden layer. Certain neurons get activated for certain inputs from input layer neurons. Say, the first neuron is focused on Area, and bedrooms. Second neuron may pick 3 parameters – Area, Bedrooms, Age of property. It will create a new parameter which is the combination of the 3 parameters.

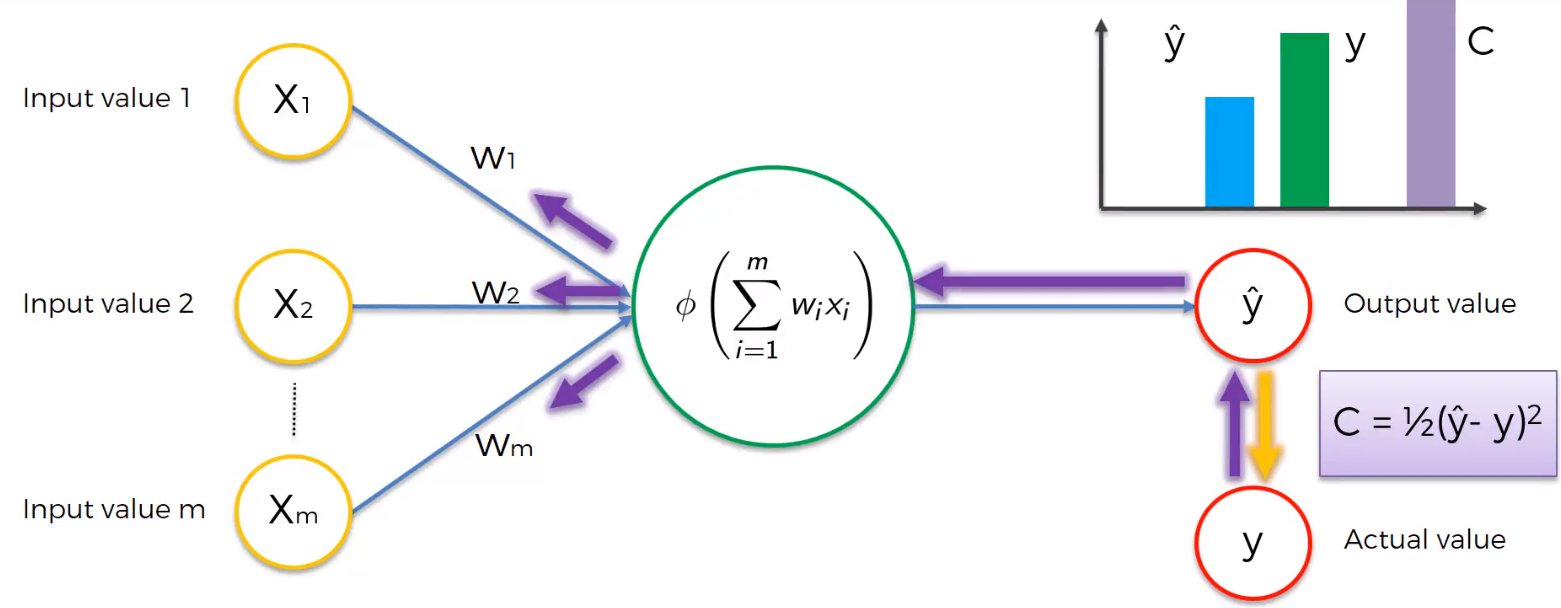


4. How Neural Networks learn?

ANN should understand what it needs to do on its own by creating a facility. We will avoid putting in rules manually.

Consider ANN with only one hidden layer called as Single Layer Feed Forward Neural Network or a Perceptron. The output value is y^ and y is the Actual value.

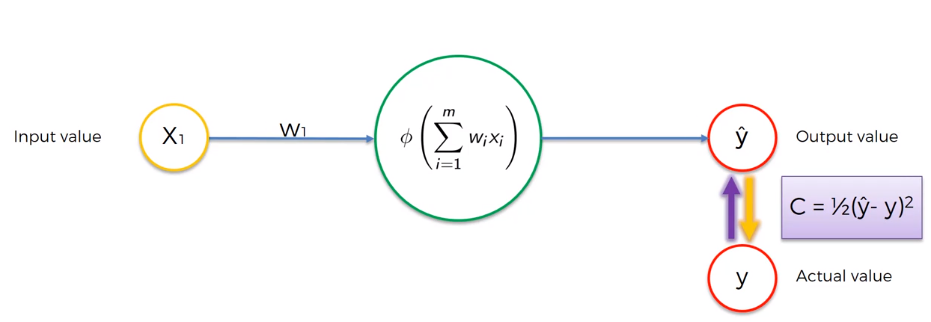
Consider some input values, that is supplied to the perceptron, then activation function is applied and we have a output y^. On comparing actual value to the output value, we get a difference. We calculate a cost function which is half the square of the difference between the original and actual value. The cost function tells us about the error we have in the prediction and our goal is to minimize the cost function. Lesser the cost function, closer the y^ is to y. Once the cost function is calculated, we feed this info back into the NN and it goes to weights and the weights get updated.



We are dealing with only one row – one observation. All the observations share the weights.

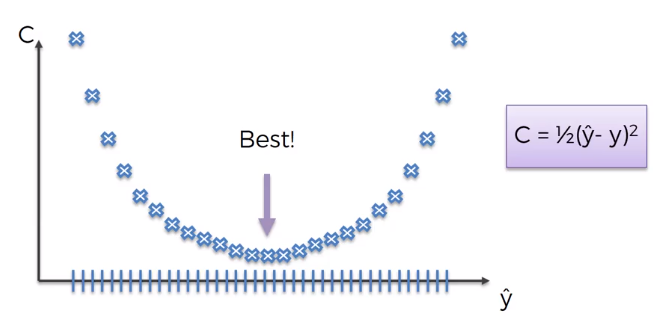
5. Gradient Descent

Neural Networks use Back Propagation to learn. The value returned by cost function is back propagated to the NN, the weights the adjusted accordingly. How these weights are adjusted?

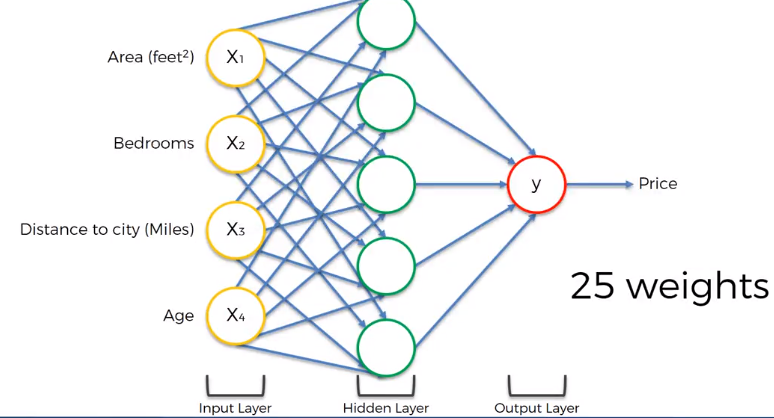


How to minimize the cost function?

1. Brute Force Method – Try different values of weights and calculate the cost function for all the values.

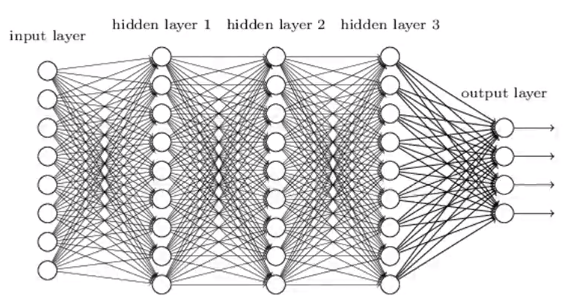


The one with minimum cost function value gives the best set of weights. But if you increase the number of weights, and the number of synapses in NN, then you will face Curse of Dimensionality. Consider the following below network –



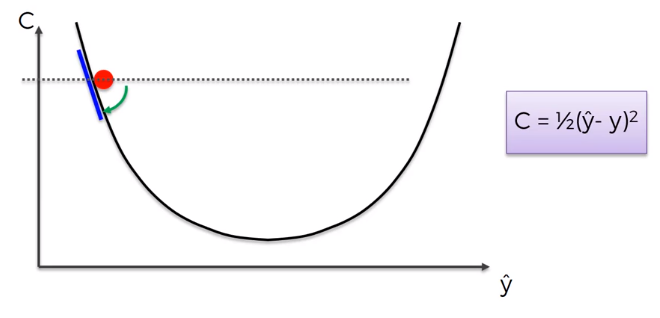
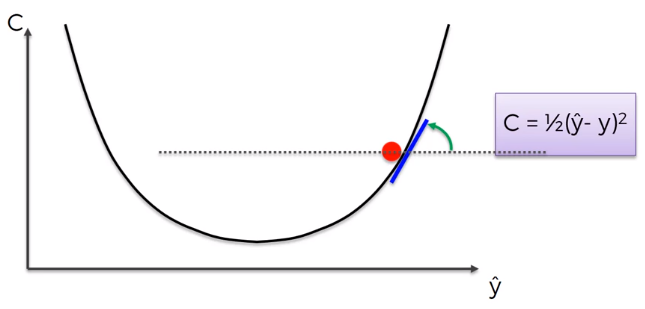
There are 25 set of weights in total. Suppose we have 1000 different values for each weight, then the total number of combinations would be - 100025  = 1075  Combinations. The world’s fastest Super Computer – Sunway TaihuLight. It can perform 93 PFLOPS – 93 X 1015 floating operations per second. Let each combination be one PFLOP. It would still require 1075/ (93 X 1015) = 1.08 X 1058 seconds = 3.42 X 1050 years. So, this is not possible.

Suppose NN looks something like this –

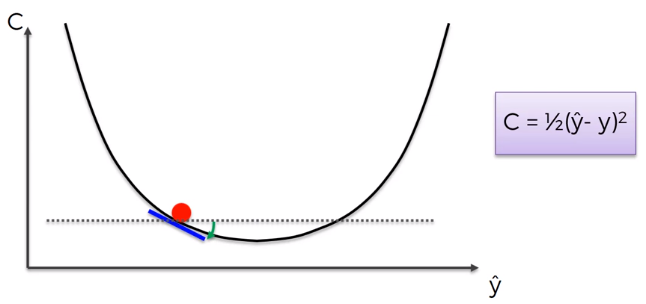


The time taken to process will be greater than that. One solution to this problem is - Gradient Descent also called as Batch Gradient Descent Method.

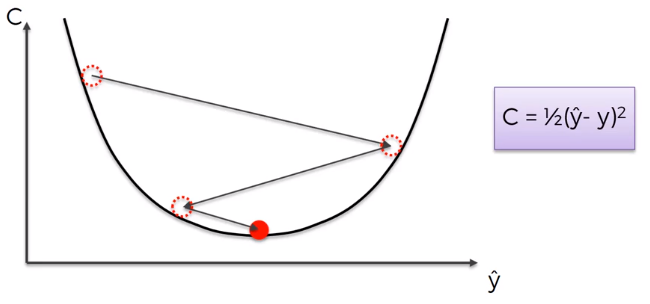
Gradient Descent Technique – Consider a point (combination of weights) in a 2D plane and the calculate the angle of cost function at that point which is called Gradient where you differentiate the cost function and find the slope at that specific point and find if the slope is positive or negative. If the slope is negative, it means that you are going downhill and if the slope is positive, it means you are going uphill. So, as shown in the below figure, the slope is downhill, now move and consider the next point which is likely to be uphill.

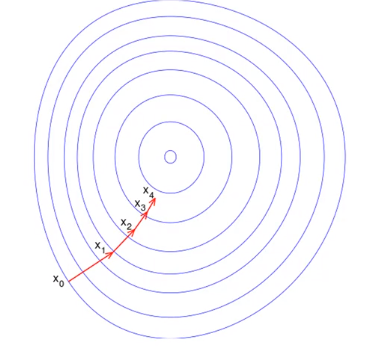
Consider a third and fourth points, where you finally get the optimum point.

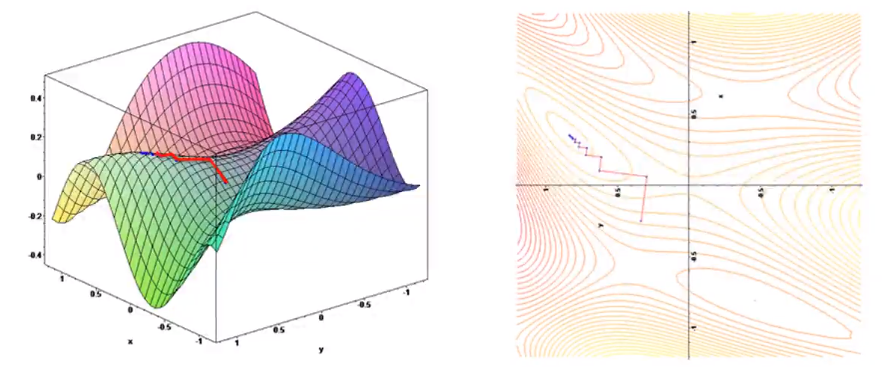
It is a zigzag type of approach –



Gradient Descent applied in 2D space –

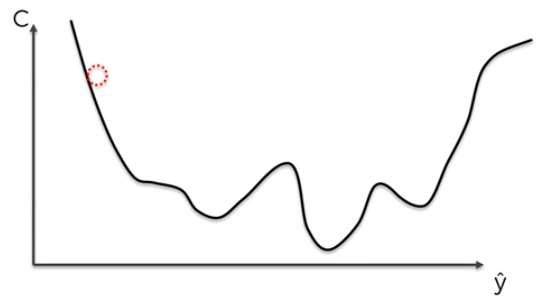


Gradient Descent in 3D space and when it is projected onto 2D space –

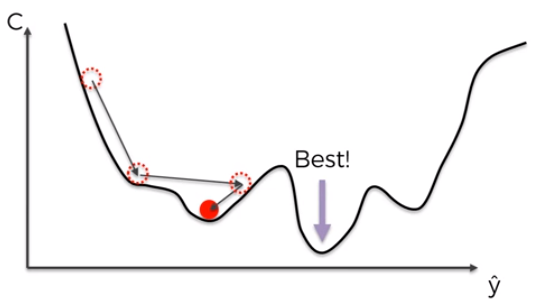


6. Stochastic Gradient Descent

Problem with Gradient Descent – This method requires the cost function to be convex. Convex function is one which has one global minimum. What happens if the cost function is not convex. Example shown below –



This could happen if our cost function is different which is obvious. And also in cases where cost function is projected onto multiple dimensions. In such cases, we end calculating local minimum of cost function rather than global minimum.



So, the solution is stochastic gradient descent that doesn’t require the cost function to be convex.

Difference between Gradient Descent and Stochastic Gradient Descent –

In the former, we calculate the cost function and adjust the weights for an entire set of observations i.e. we plug in entire set of observations onto the neural network, calculate the cost and adjust weights. That is why it is called Batch Gradient Descent Method. We take whole batch from sample for the process. We adjust the weights after we run all of the rows in NN. It is a deterministic algorithm. Every time you run batch gradient descent method, you get same result for bunch of weights.

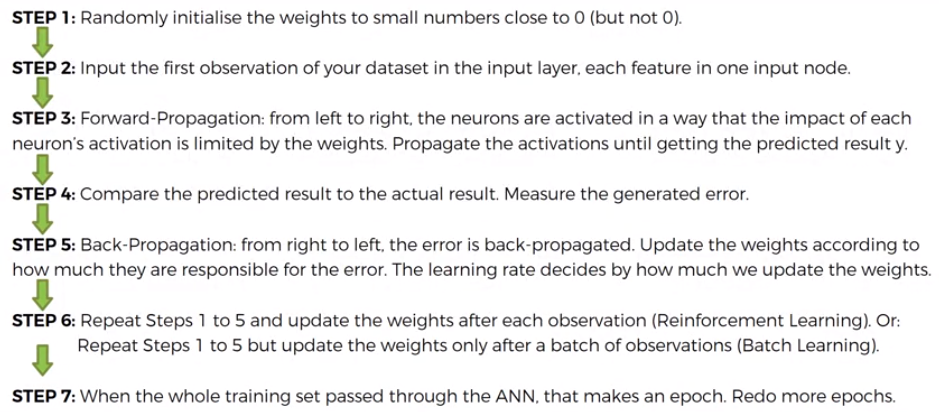
In the latter, we take the rows one by one. We consider an observation, run the NN, calculate the cost function and back propagate it to the NN and adjust the weights. We repeat the same steps for subsequent rows. We adjust the weights for each observation/ every single row one by one. The advantage here is that, stochastic gradient descent method helps us to avoid local minimums because it has much higher fluctuations because it is considering one row at a time and it is most likely to find the global minimum. Also, it is faster than Batch gradient descent method because it doesn’t have to load entire dataset into memory. It is a stochastic algorithm because it is random. Here, every time you run stochastic gradient descent method, you may not get same set of results because each observation is selected randomly every time.

Mini Batch Gradient Descent – You run batches of rows at a time and you update the weights.

7. Backpropagation

Forward Propagation – Information is entered into the input layer and is propagated forward to get y^ (output values) and then we compare those to the actual values in the training set and calculate the errors. Then the errors are back propagated through the network in the opposite direction and that allows us to train the network by adjusting the weights. Back propagation allows us to adjust all of the weights simultaneously. Key thing to remember – You know which part of the error each of the weights in the NN is responsible for.

Step by step walkthrough of what happens in the training of NN with Stochastic Gradient Descent–



What is Keras and What is TensorFlow?

Keras is a wrapper. Think of Keras like the body of a car with pedals, steering wheel, seat, etc.

TensorFlow (or Theano, or CNTK) is the engine of the car.

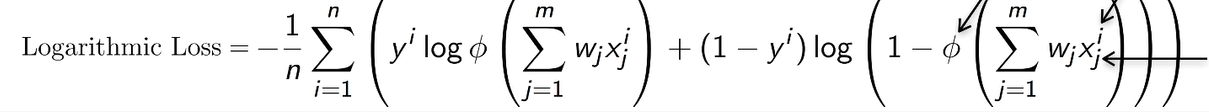
When we drive the car, we are actually using the engine, but we access the control of the engine through the body of the car. In other words, keras provides us easy access to the complicated TensorFlow interface, helping us prototype models with very minimum effort.

Number of nodes in hidden layer = Average of number of nodes in Input layer and output layer.

Parameter Tuning – Using techniques like K-Fold cross validation to test the model with different parameters. It will help us choose optimal number of parameters in the hidden layer.

Loss function for Stochastic Gradient Method is same as loss function for logistic regression.

For a simple perceptron (single neuron), if we use a Sigmoid Activation function, then what we obtain is a logistic regression model. The loss function for logistic regression model is logarithmic function shown below –

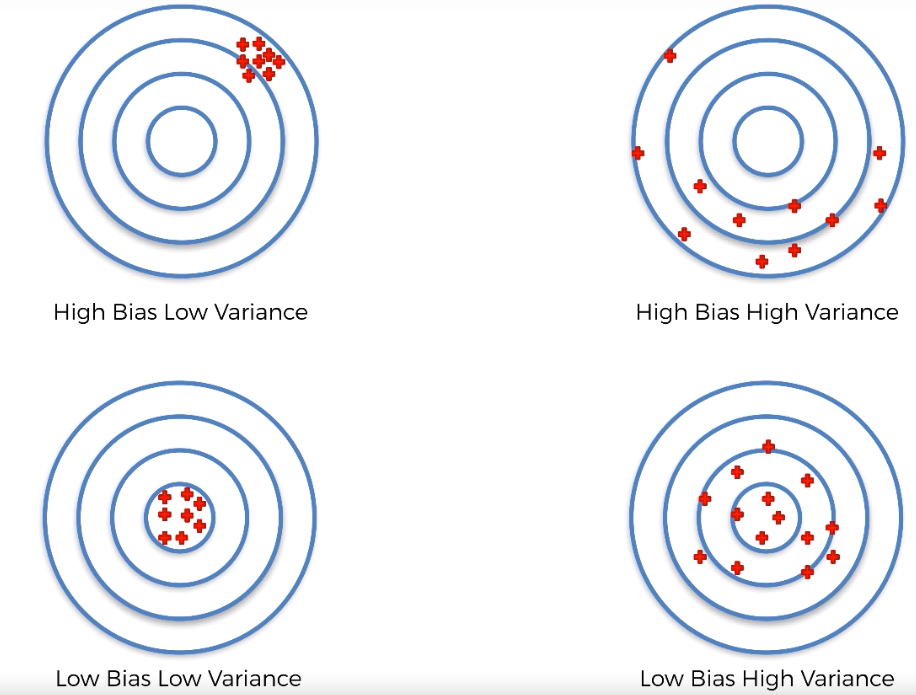


If the dependent variable has a binary outcome, them this logarithmic loss function is called – Binary\_CrossEntropy.

If the dependent variable has more than two categories as outcome, them this logarithmic loss function is called – Categorical\_CrossEntropy.

Measuring the accuracy of the model on one single test set is not the most relevant way to model’s performance because by retraining the model several times, we get different accuracies. This solution is called cross-validation.

Bias-Variance Tradeoff – We are trying to train a model that will not only be accurate but also it should not have too much variance of accuracy when we train it several times. When we run the model one test set and then run the model on a different test set, we get different accuracy. So, judging the model’s performance only on one test set is not relevant. So, we use K-Fold cross validation technique that will fix the variance problem.



Low Bias = High Accuracy

Dropout Regularization – Solution for overfitting in deep learning. Overfitting in deep learning is when your model was trained too much on the training set that it becomes much less performance on the test set. We can observe this when we have large differences in the accuracies between the training set and the test set. One way to detect overfitting is, you have much higher accuracy on the training set than the test set. Another way to detect overfitting is when you observe high variance when applying k-fold cross validation. When the model learns too much, it will succeed on new observations in the test set when the correlations are similar to what it has learnt from the training set. When the correlations are dissimilar, the model fails on certain test sets. So, when you get vector of accuracies in 10-fold cross validation, we get some high accuracies, few low accuracies and hence high variance.

Working mechanism of Dropout Regularization – At each iteration of the training some neurons of ANN are randomly disabled to prevent them from being too dependent on each other when they learn the correlations and therefore by overriding these neurons the artificial neural network learns several independent correlations in the data because each time the configuration of the neurons is different, and we get independent correlations of the data. Neurons work more independently. That prevents the neurons from learning too much and therefore prevents overfitting. If the model has overfitting problem, then you should apply dropout to all the hidden layers.

Parameter Tuning – Finding the best values for hyper-parameters for the model like best epochs, batch size, best optimizer.

There are 2 types of parameters in ANN model -

1. Parameters that are learned from model during training (weights)

2. Other parameters that stay fixed – Number of epochs, batch size, optimizer and number of neurons in the layer. When we train ANN, we train it with fixed values of these hyper-parameters, but there may be better values for these parameters. We use grid search technique to get best values of these hyperparameters and achieve great accuracy.