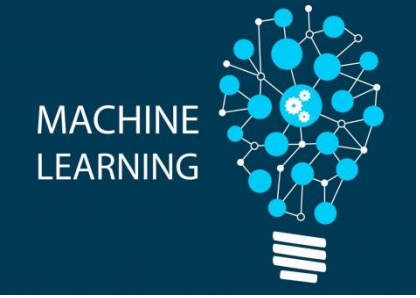
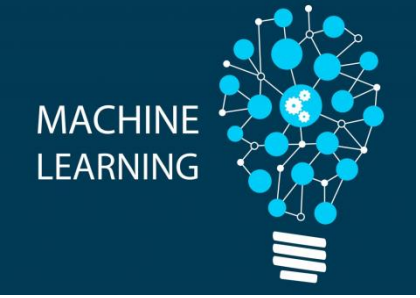


Deep Learning

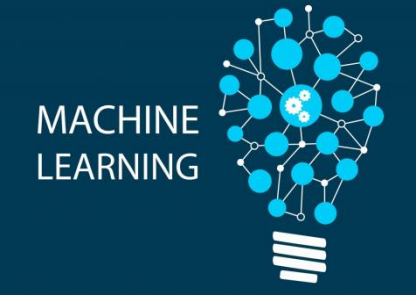
Diptangshu Banik

Twitter - @dipbanik



Overview

- Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.
- One of the most powerful branch of Machine Learning.
- As the amount of data increases, deep learning becomes more and more relevant.



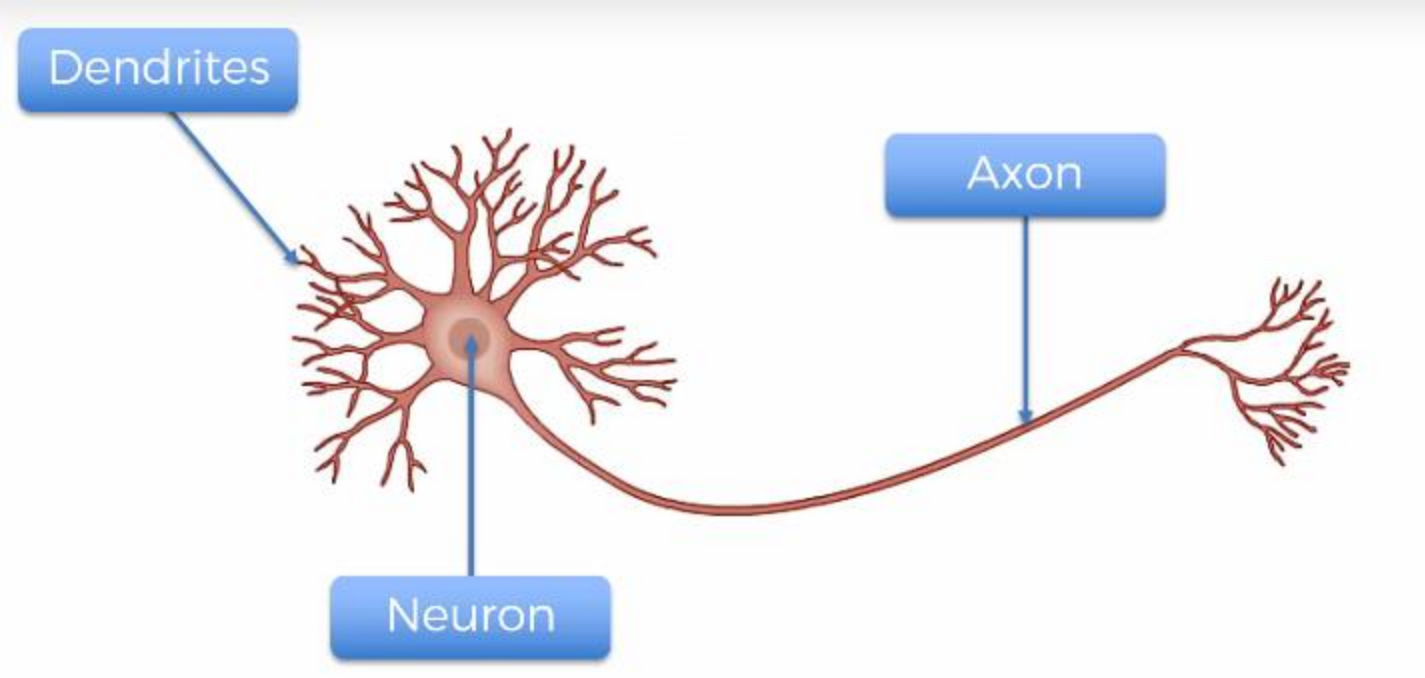
Examples

- Artificial Neural Networks for Regression and Classification
- Convolutional Neural Networks for Computer Vision
- Recurrent Neural Networks for Time Series Analysis
- Self Organizing Maps for Feature Extraction
- Deep Boltzmann Machines for Recommendation Systems
- Auto Encoders for Recommendation Systems

MACHINE LEARNING

Artificial Neural Network

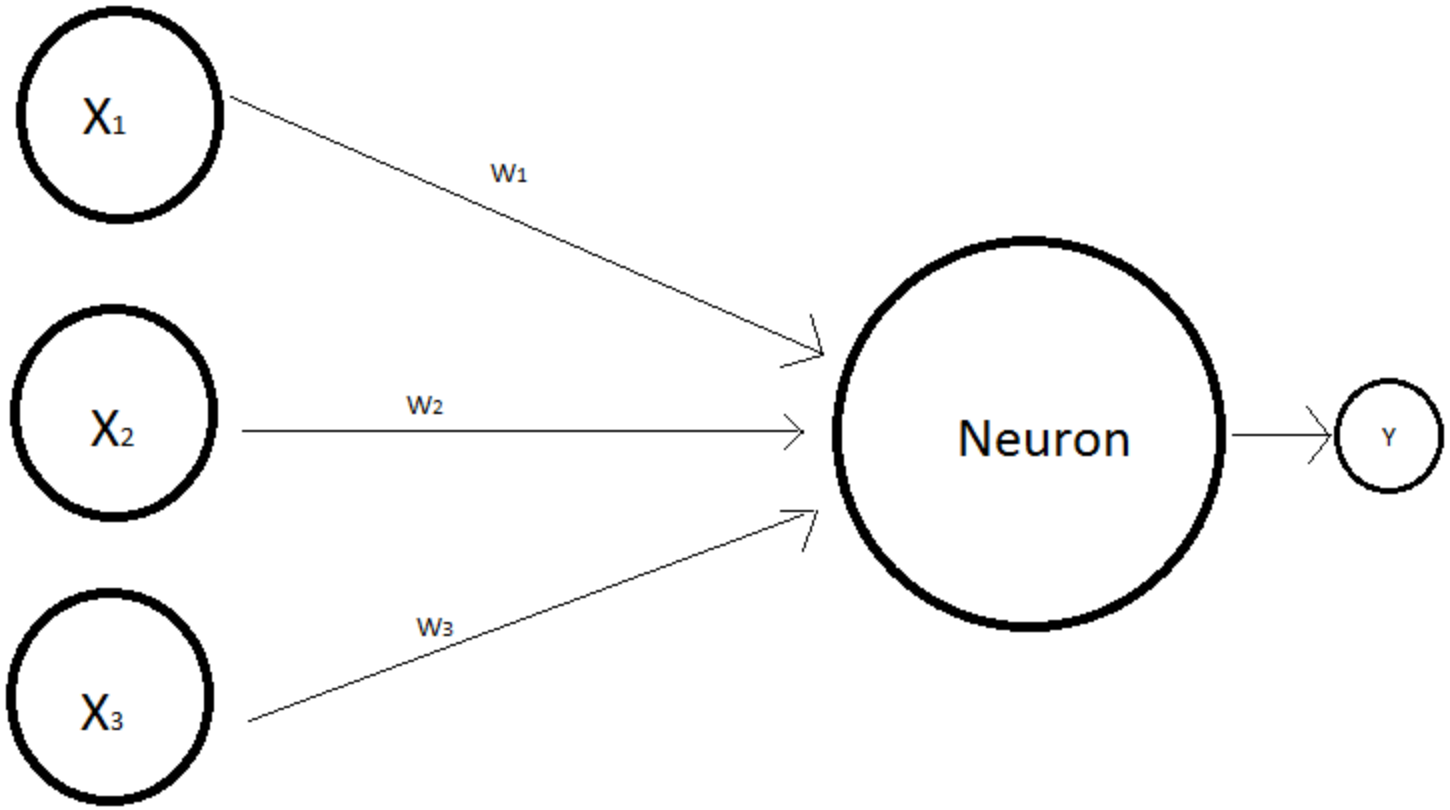
- Multiple neurons work together to perform a powerful deep learning algorithms.
- Synapse (connection between two neurons are called Synapse.

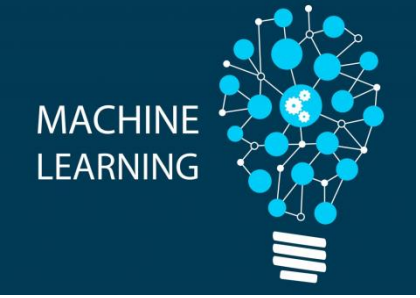


MACHINE LEARNING

Artificial Neural Network

- How Input and output works with neurons.

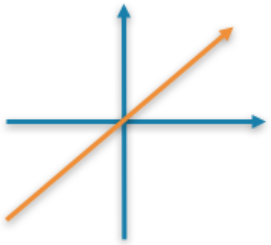




The activation function

- An activation function is a non-linear function applied by a neuron to introduce non-linear properties in the network.

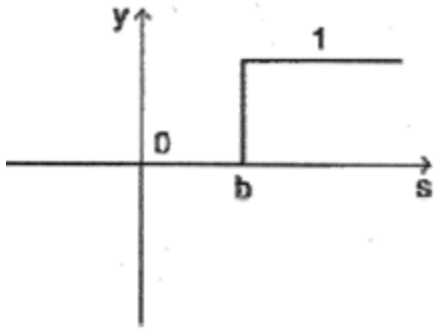
- Linear Activation Function :

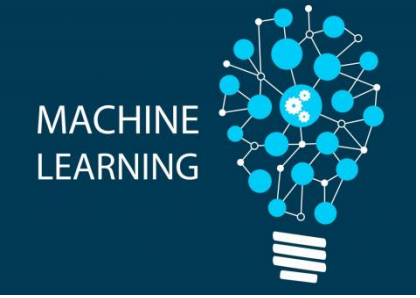


$f(z) = az$

Range: -infinity to infinity

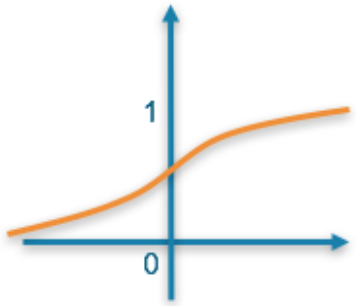
- Threshold function : 1 or 0.





The activation function

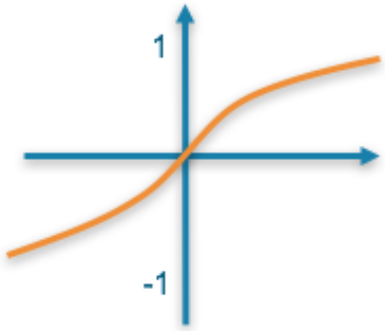
- Sigmoid Function :



$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

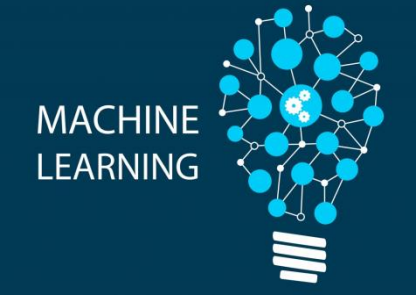
Range: 0 to 1

- Hyperbolic Tangent (*tan* h) function –



$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

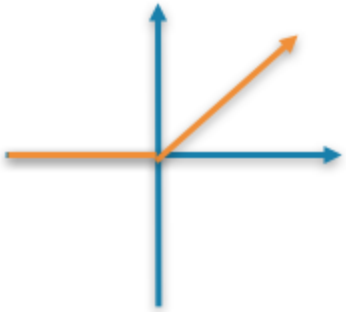
Range: -1 to 1



The activation function

- Rectified Linear Unit

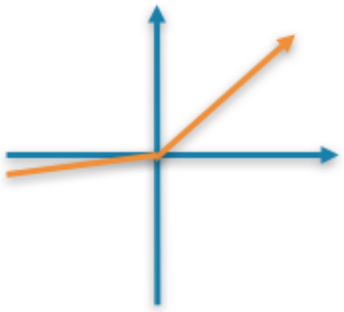
(ReLU) Activation Function



$$relu(z) = \max(0, z)$$

Range: 0 to infinity

- Softmax Activation Function : Range : 0-1
  - Used only in output layer.
- Leaky ReLU Activation Function



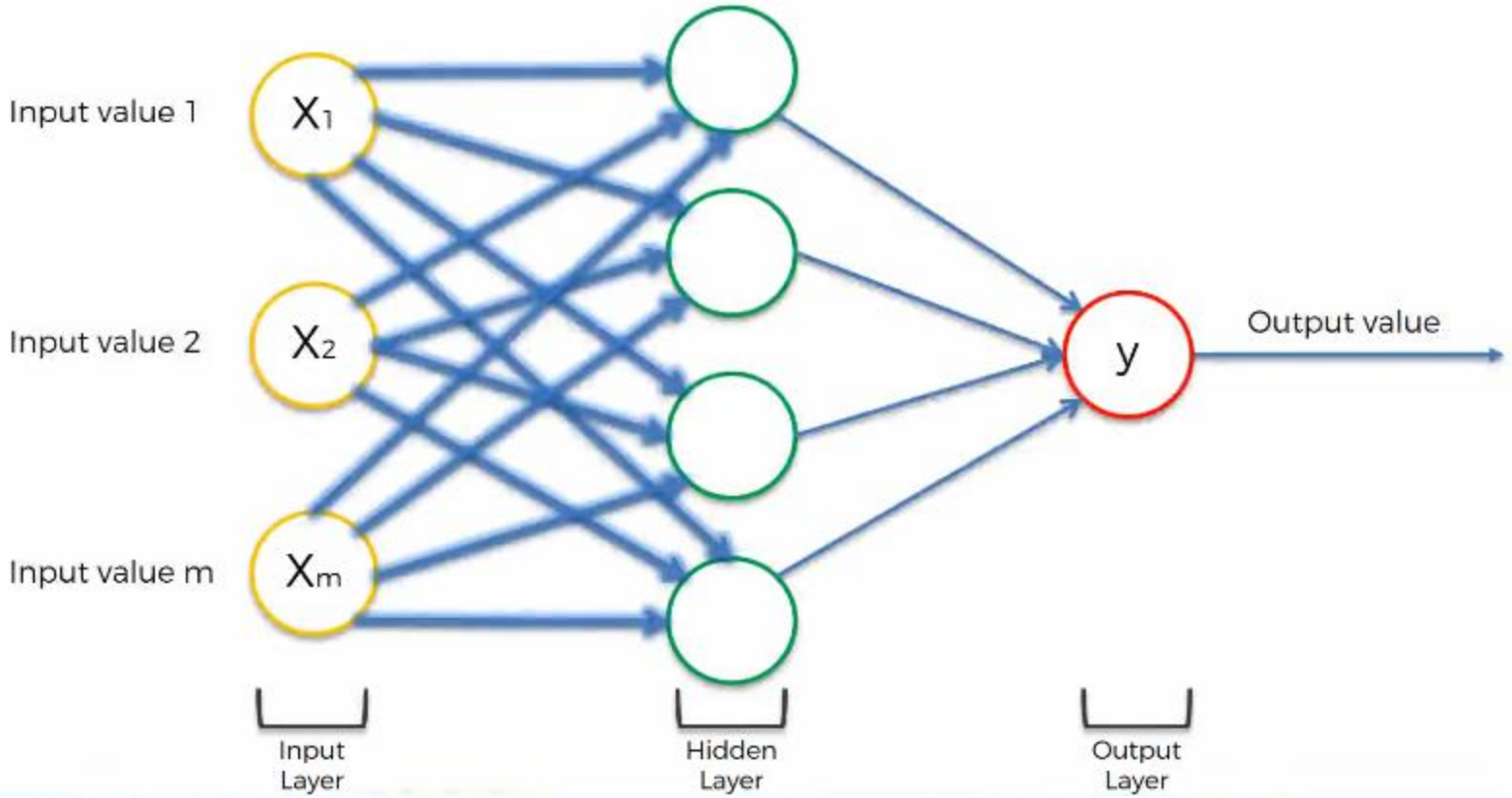
$$leakyrelu(z) = \begin{cases} 0.01z & for\ z < 0 \\ z & for\ z \geq 0 \end{cases}$$

Range: -infinity to infinity

MACHINE LEARNING

The hidden layer

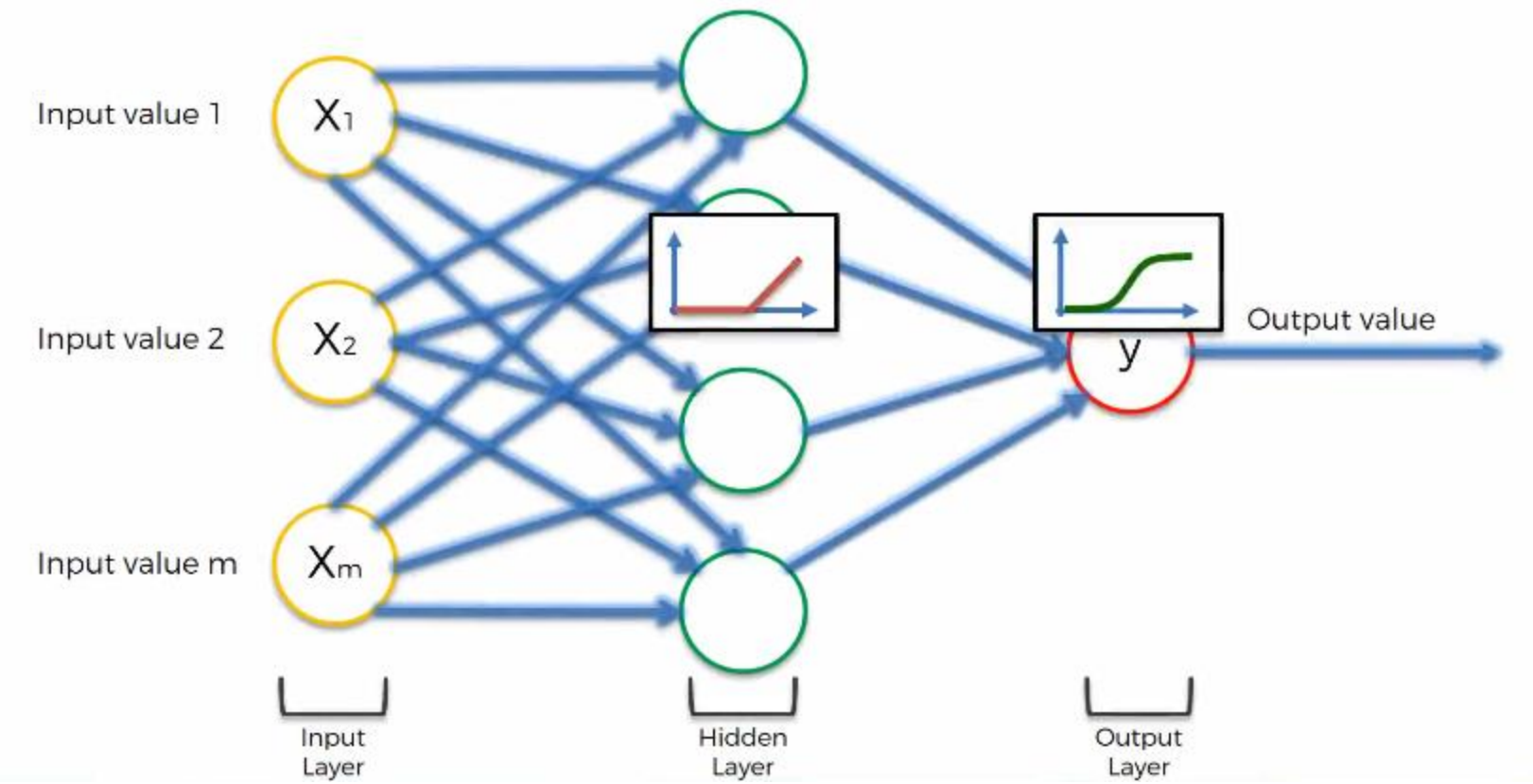
- The hidden layers are present in between the input and output layers. Can be multiple.



MACHINE LEARNING

Application of activation function

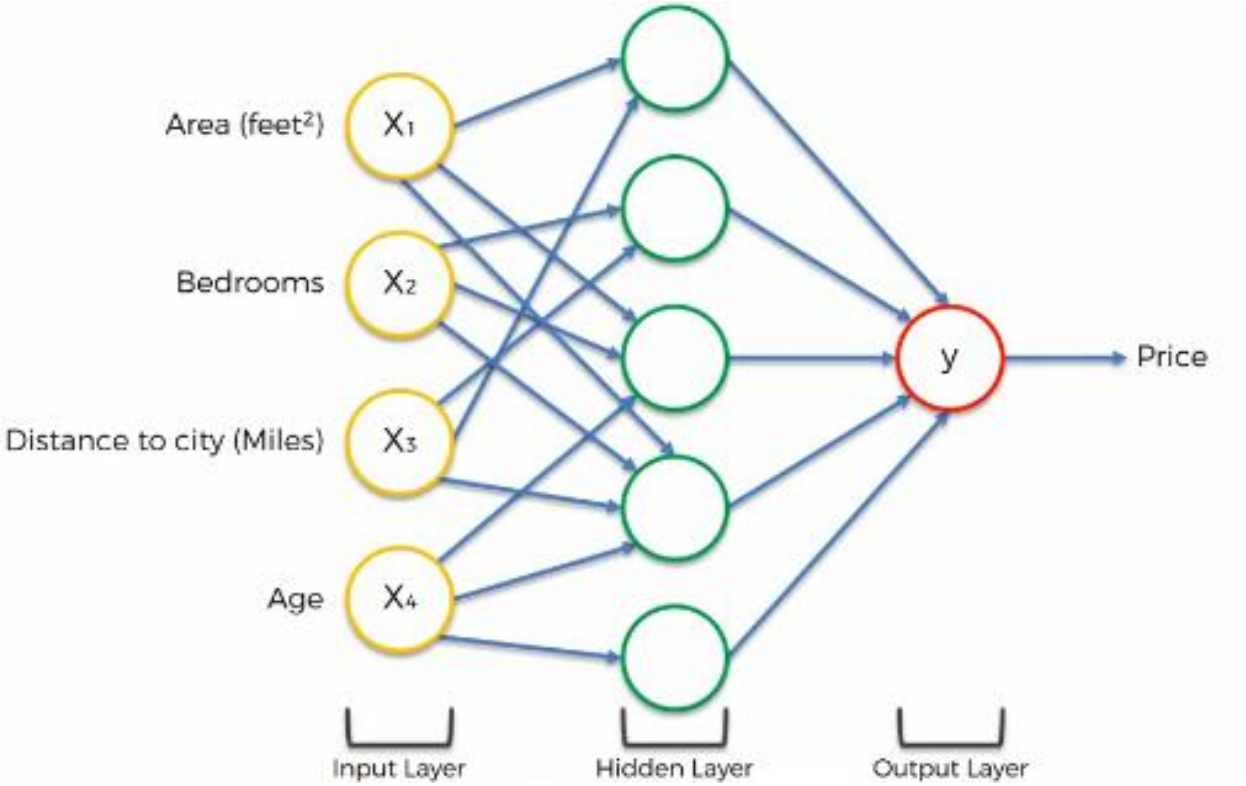
- An activation function is applied to each layer of the model.
- Hidden layer applies Rectifier function and output layer applies Sigmoid function.



MACHINE LEARNING

Example Final Model

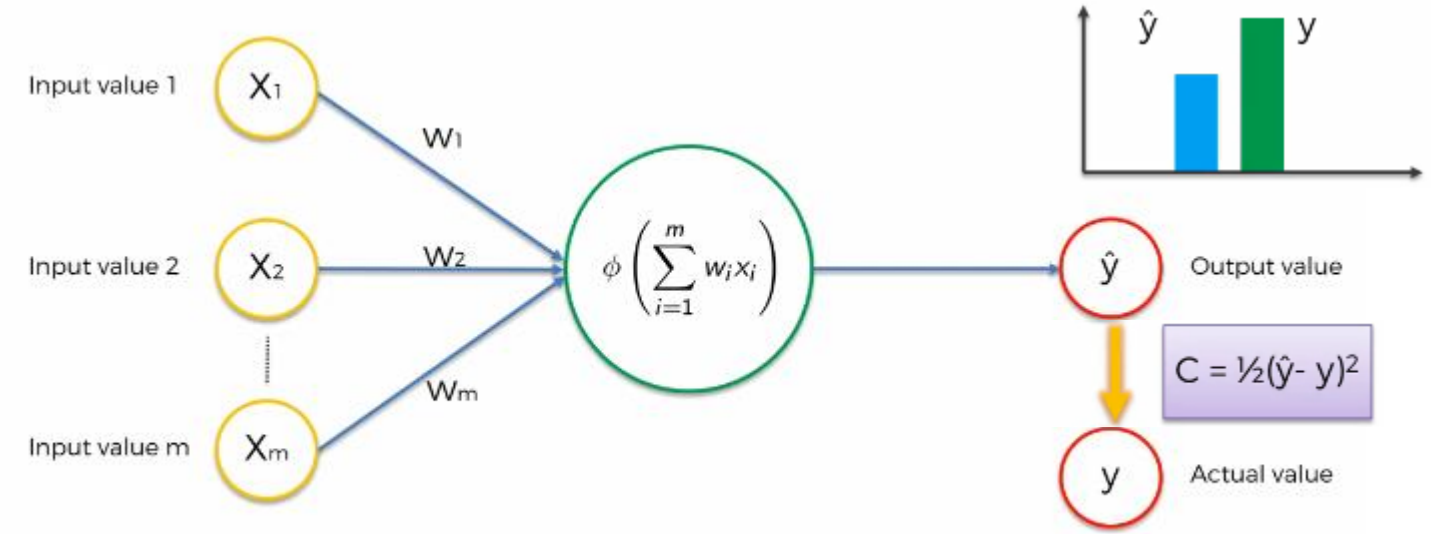
- In the final model, all inputs need not necessarily be a part of the neuron in the hidden layers.



MACHINE LEARNING

How do Neural Networks work?

- This is a Perceptron or Single Layer Feed Forward Network.
- C is the cost function which is the difference between actual and predicted values of y.
- Goal is to keep updating the weights with multiple **epochs** so that the cost function is minimal.



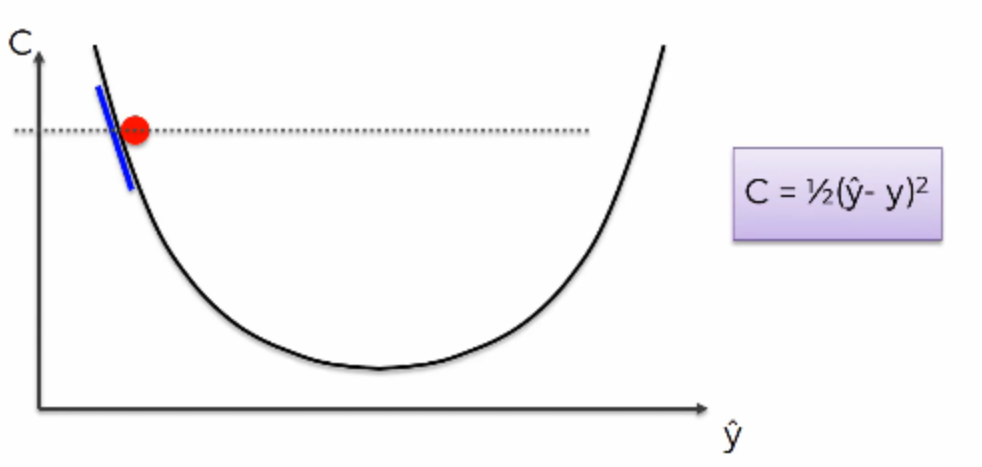
MACHINE LEARNING

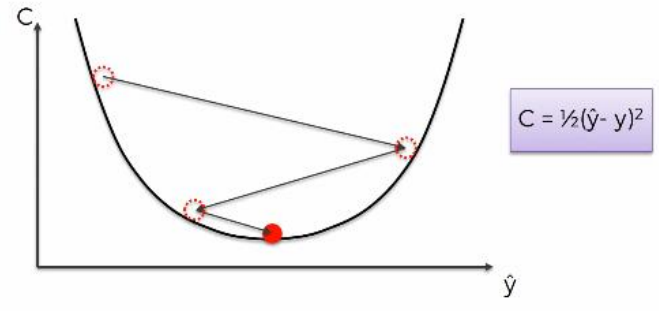
Gradient Descent

- To find the lowest point of the curve, Brute force method – Not feasible due to Curse of Dimensionality
  - 4 inputs, 1 hidden layer – 25 weights – 10^{75} combinations .

- Gradient Descent uses slope to identify the lowest point of the cost function curve.

- Slope = Negative, go right.
- Slope = Positive, go left.

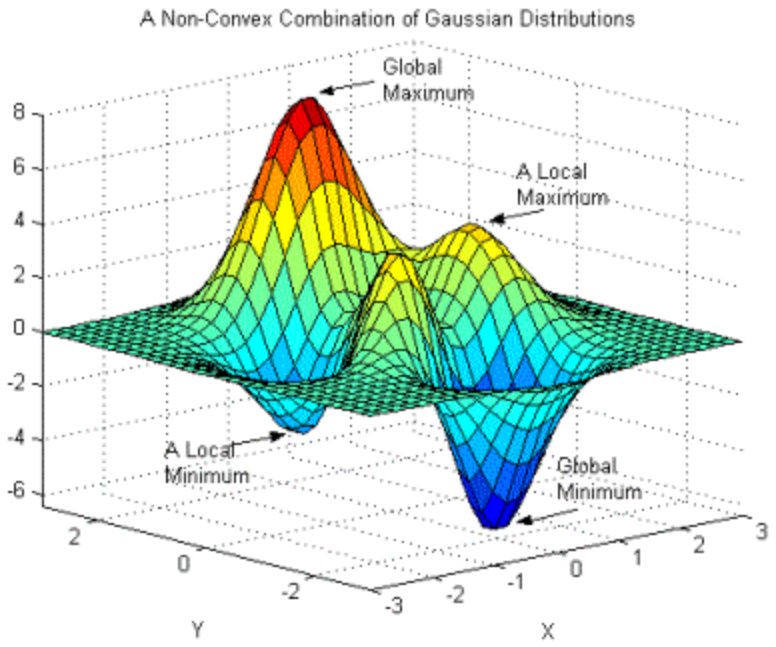


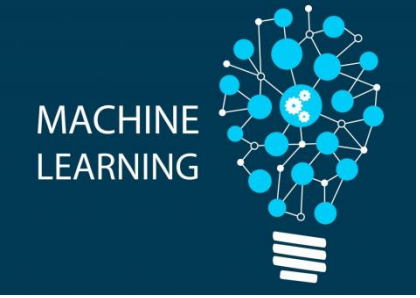


MACHINE LEARNING

Stochastic Gradient Descent

- Gradient Descent updates its weights after each epochs(also called batch Gradient Descent).
- Stochastic Gradient Descent updates its weights on execution of every row of data that it executes.
- Stochastic Gradient Descent does not stick to the local minima but actually considers the global minima. Very useful with multi-dimensional data.
- All weights are adjusted simultaneously in case of backpropagation.





Steps for ANN

- Step 1 : Randomly initialize the weights to small numbers close to 0.
- Step 2 : Input the first observation of your dataset in the input layer, each feature in one input node.
- Step 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.
- Step 4: Compare the predicted result to the actual result. Measure the generated error.
- Step 5: Back-Propagation: from right to left. the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.
- Step 6: Repeat Step 1 to 5 and update the weights after each observation (Reinforcement Learning) Or Repeat steps 1 to 5 but update the weights only after a batch of observations(Batch learning).
- Step 7: When the whole training set passed through ANN, that makes an epoch. Redo more epochs.