Artificial Neural Networks

Pallapu Mohan Krishna
Research Scholar
MFSDSAI
IIT Guwahati

Introduction

- The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain.
- There are several kinds of artificial neural networks, which are implemented based on the mathematical operations and a set of parameters required to determine the output
- Types of ANN are:

  Feed Forward
  Radial Basis Function Network
  RNN or Feedback
  Hop Field Network
  Self-Organizing maps
  Competitive Network etc.,

Single Layer Perceptron

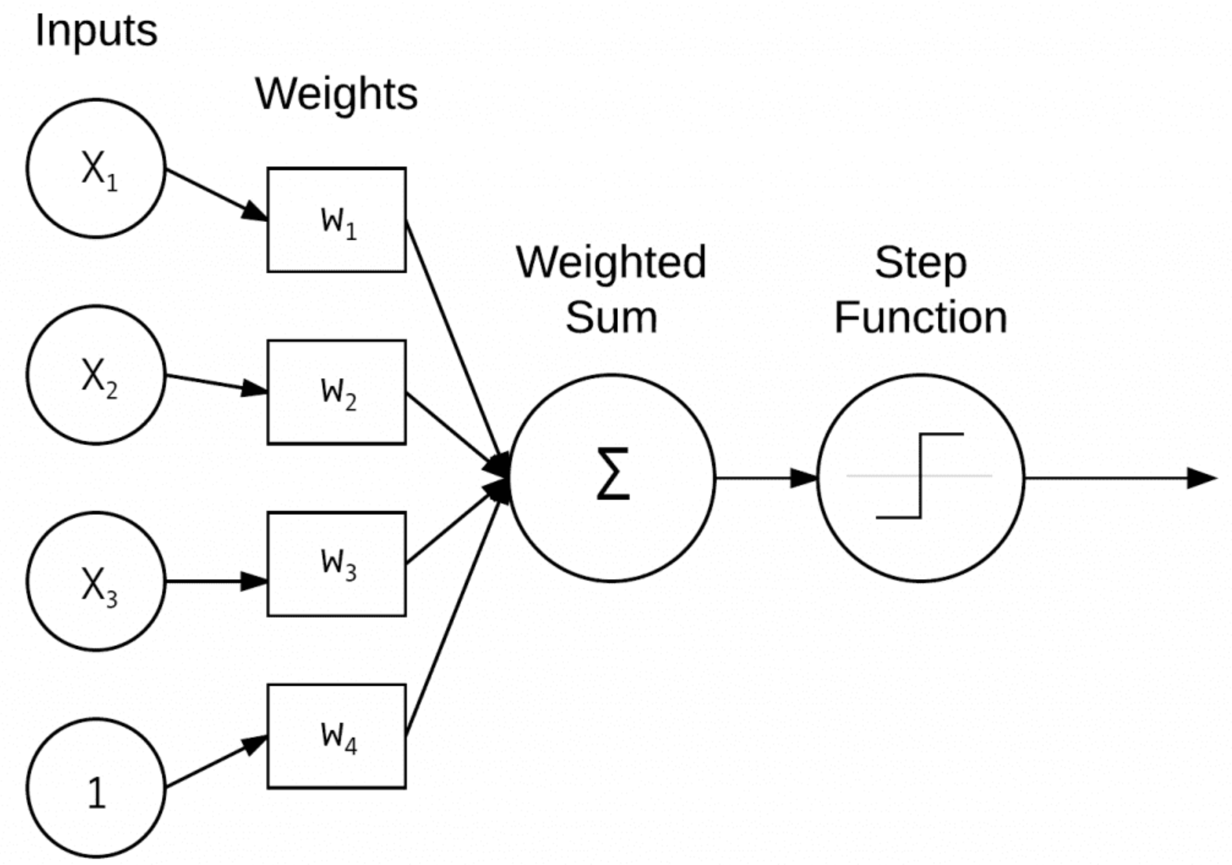


Image Credits : PyImageSearch

- While training ,we have to decide these connection weights/weight vector to classify unknown feature vectors between two classes say ***w1*** and ***w2***
- While training the network, the aim is to set weight vectors in such a way that this sum of squared error will be minimized. Say **D** as output and **d** is target output.

$$Error\, E = \frac{1}{2}(D - d)^2$$

- We can use gradient descent procedure.

$$\frac{\partial E}{\partial W_i} = (D - d)\frac{\partial (D-d)}{\partial W_i} = (D - d)\,.X_i$$

- Initially we start with random weight vectors and values are quite small

$$W_i(0) \leftarrow initialization$$
$$W_i(k+1) = W_i(k) - \eta\,(D - d)\,.X_i$$

- Termination Criteria of this iterative process is either in complete pass all the training samples are properly or I get an error that I can tolerate.
- Lets see an example of AND,OR,XOR gate using SLP

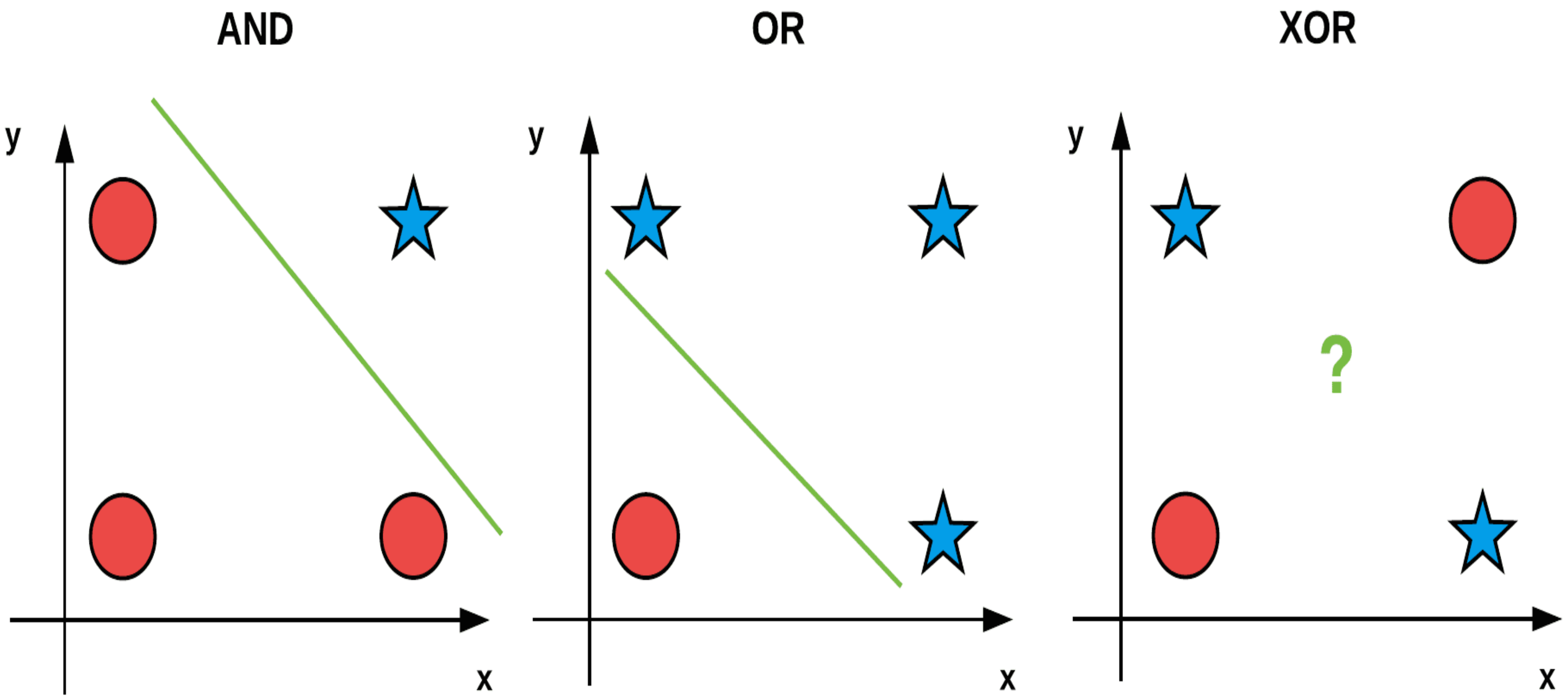


Image Credits : PyImageSearch

- So, SLP can classify AND ,OR features linearly but for XOR a single layer neural network cannot give a solution for Simple problem like XOR.
- So we have to go for MLP

Multi Layer Perceptron

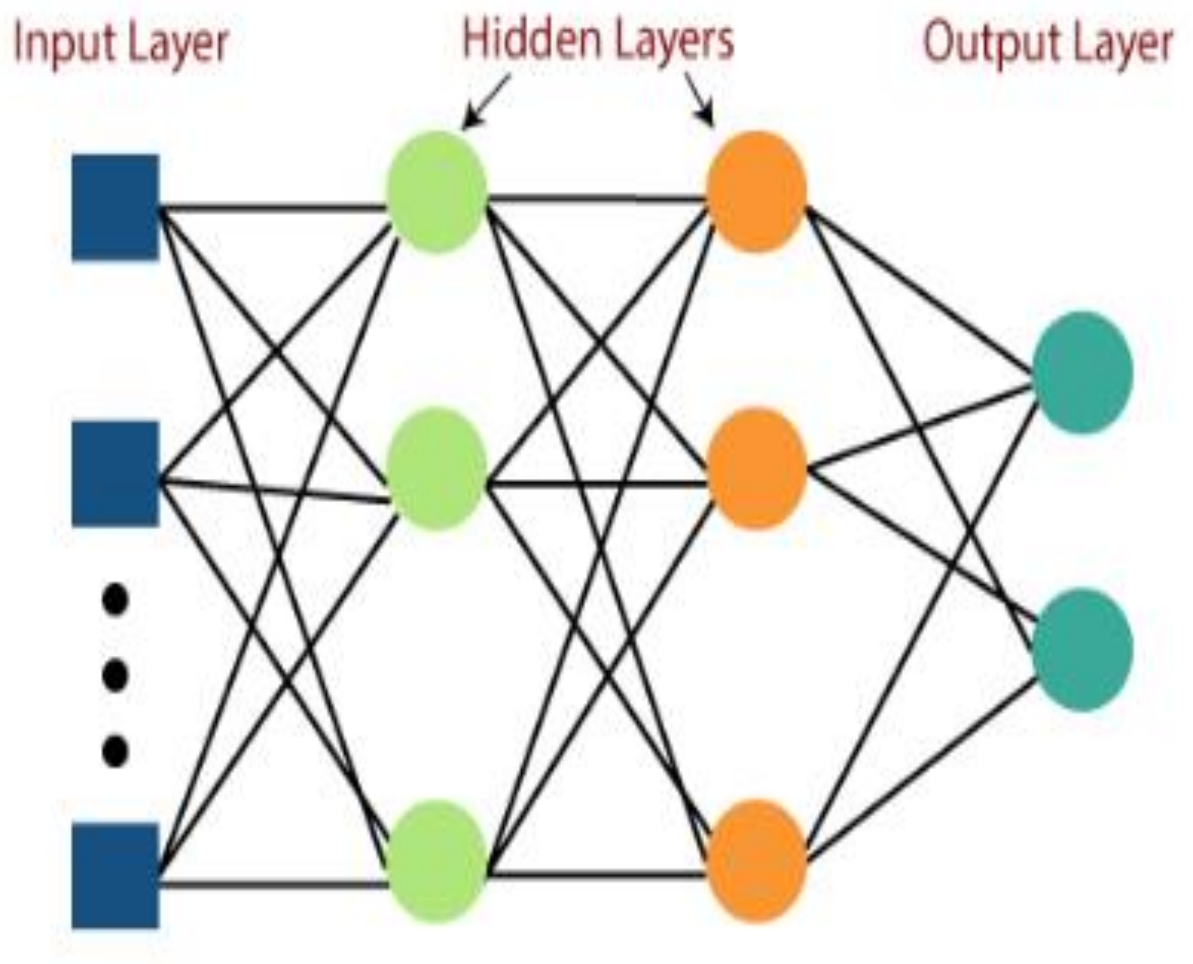


Image Credits : Analytics Vidya

- In SLP only output has non linearity i.e hard non linearity or threshold non linearity.
- But in MLP , except input layer every other layer has non linearity.

- Its differentiable where as in case of hard nonlinearity that is not differentiable at 0.
- For training of MLP , again Gradient Descent is choosed.

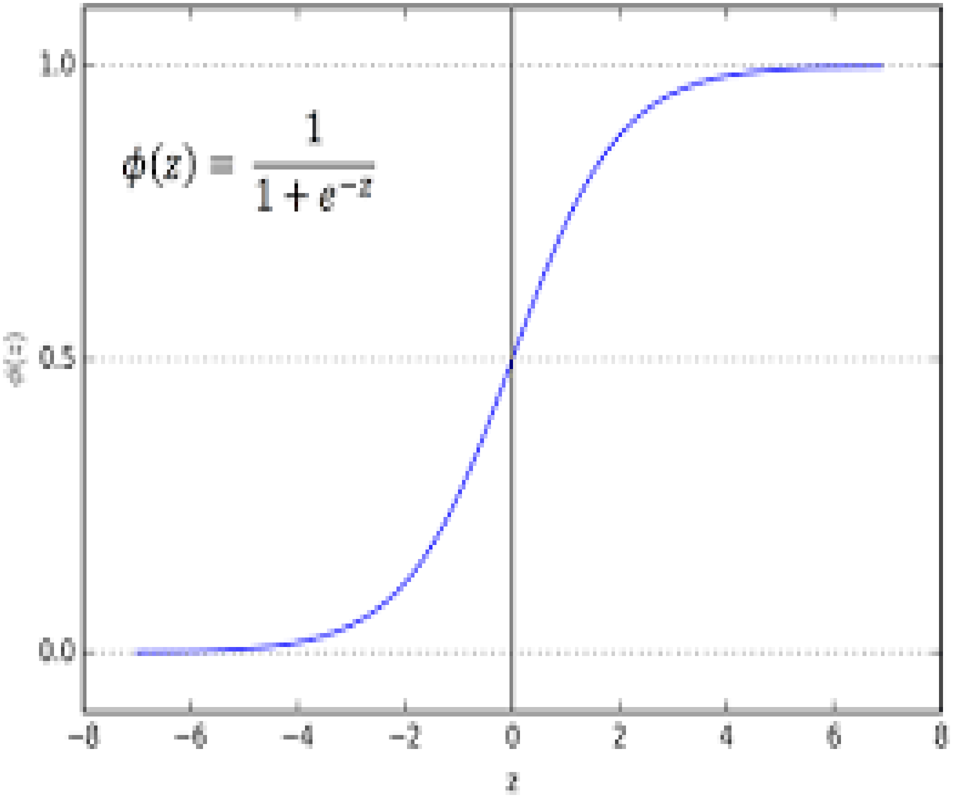


Image Credits : Medium , by Jaehoon Jang

Backpropagation learning Algo:

1. Initialize Weights randomly
2. Feed training samples
3. Feed forward pass: for i=0.......k-1 layers

   For every node compute output

4. Back Propagation: compute output gradient for every node in output layer and back propagate to previous layer nodes.

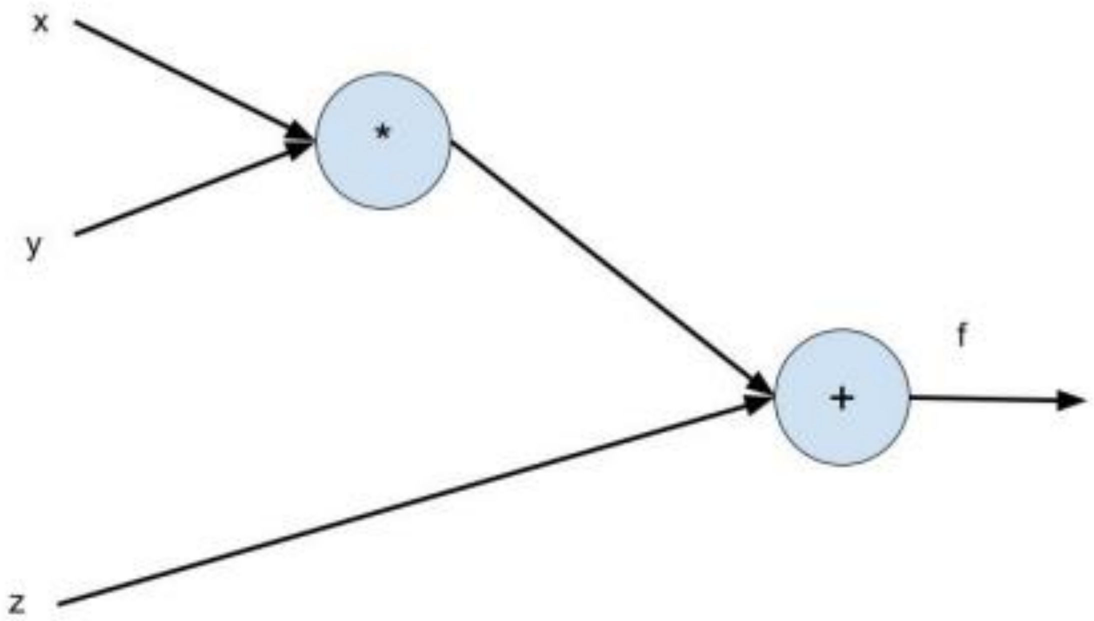
5. Update the weights:

$$W_i(k+1) = W_i(k) - \eta(J)$$

6. Repeat steps 2-5 until convergence.

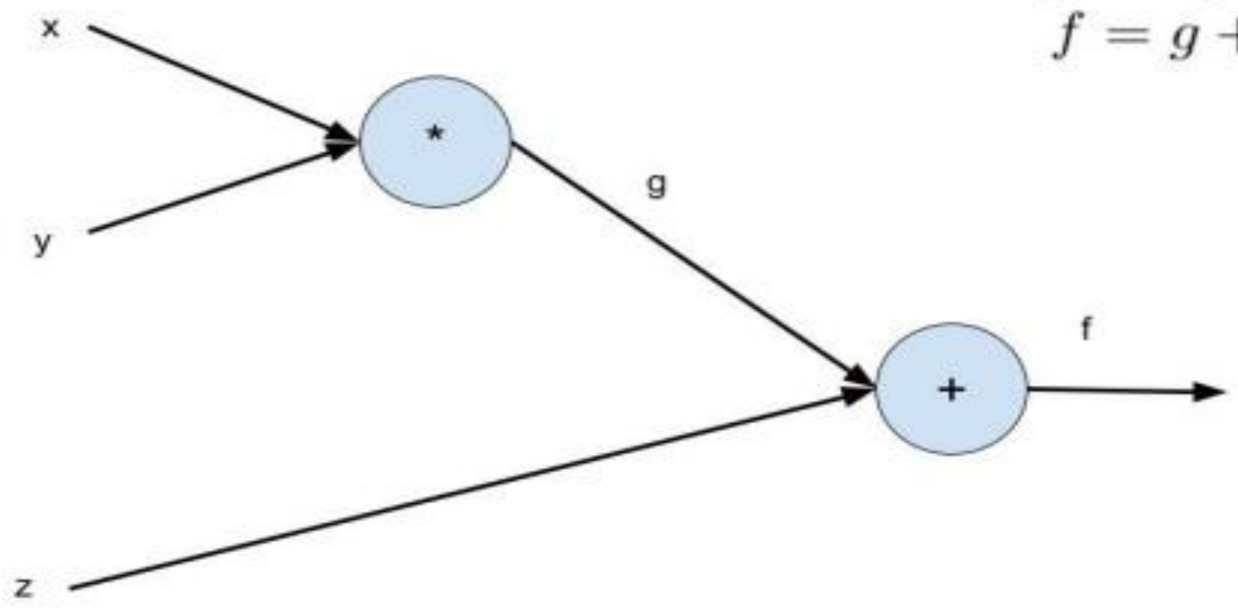
Example:

$$f(x, y, z) = xy + z$$



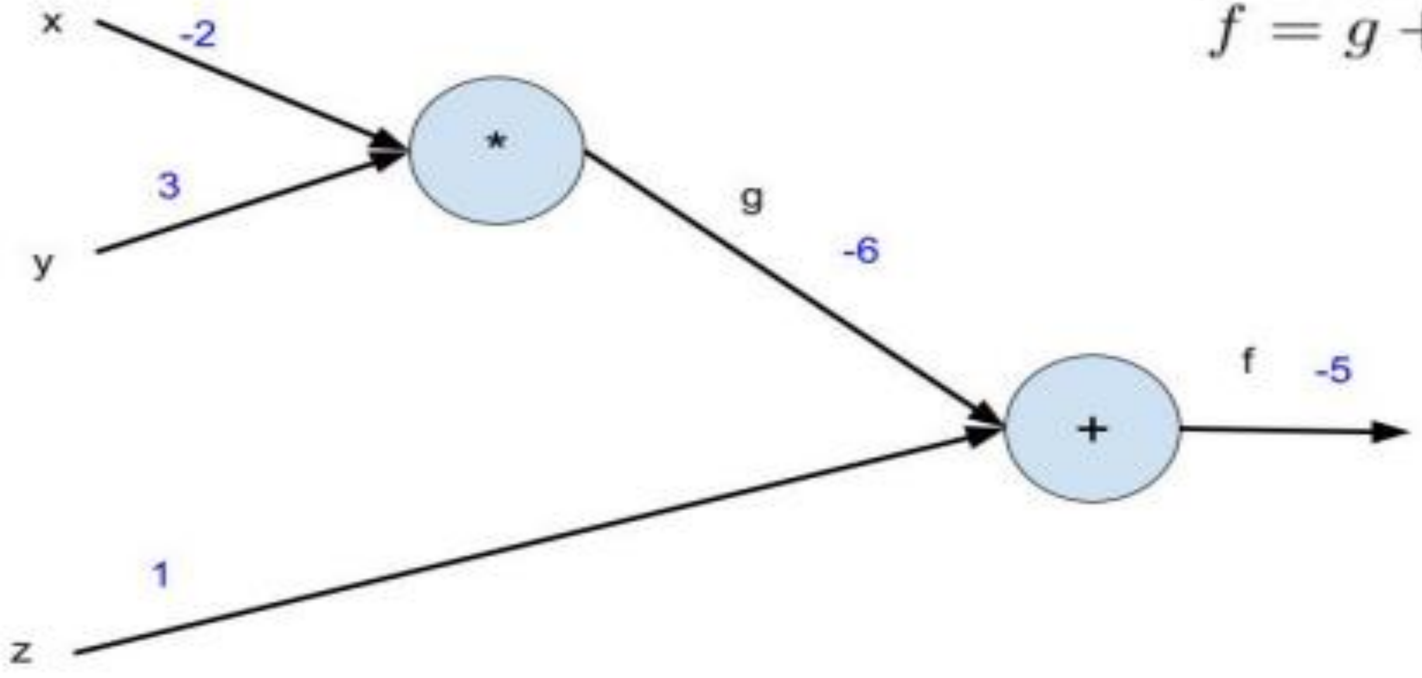
Credits : DL4CV , Dr. Kondareddy Mopuri

$$f(x, y, z) = xy + z$$
$$g = xy$$
$$f = g + z$$



Credits : DL4CV , Dr. Kondareddy Mopuri

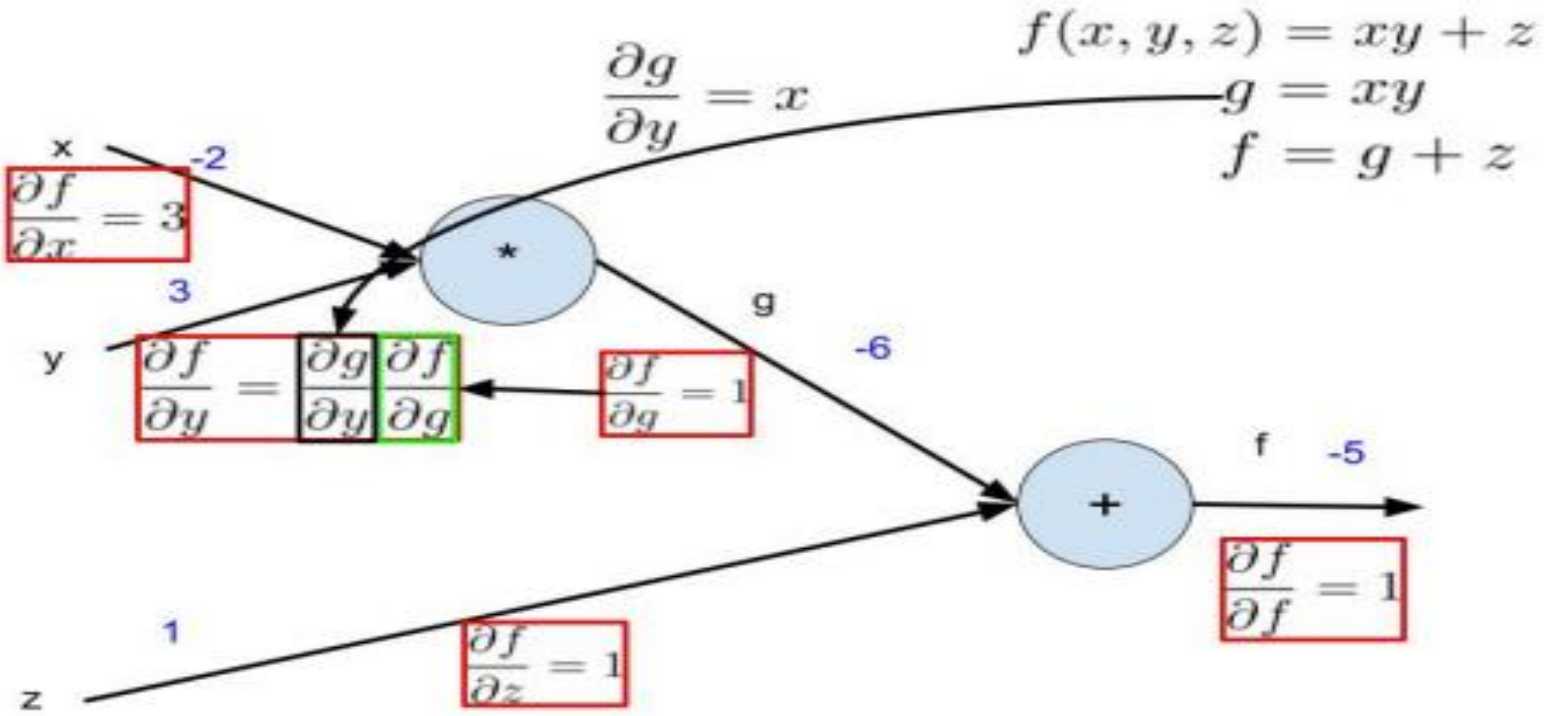
$$f(x, y, z) = xy + z$$
$$g = xy$$
$$f = g + z$$



Credits : DL4CV , Dr. Kondareddy Mopuri

BackwardPass

- Compute the derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

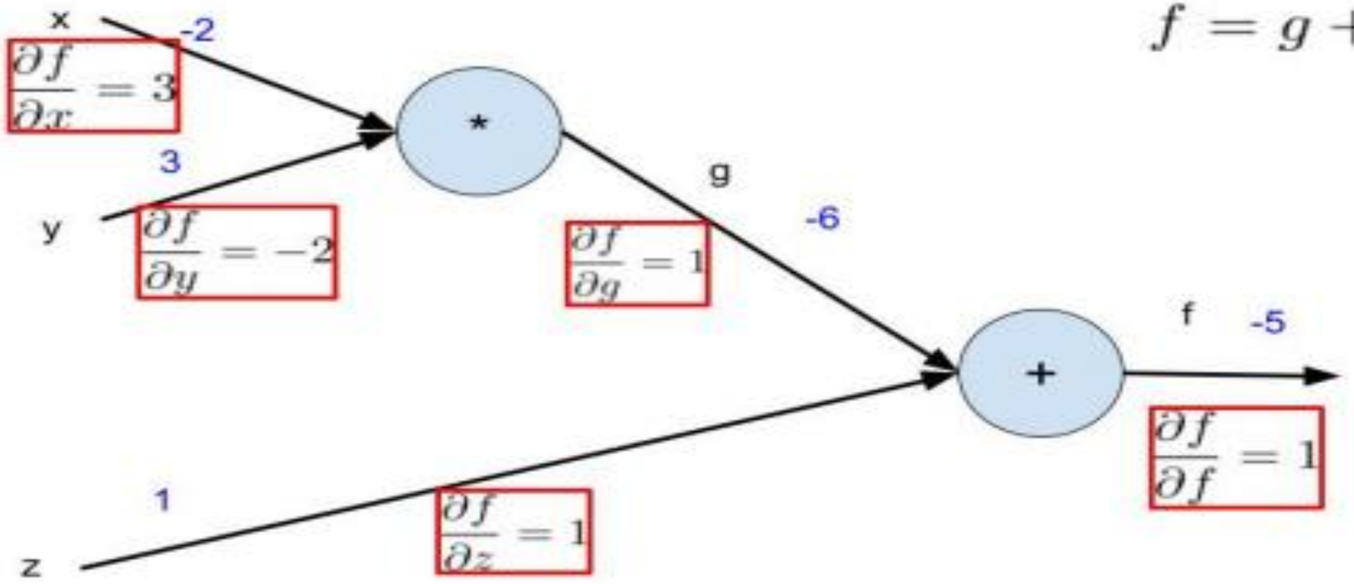


Credits : DL4CV , Dr. Kondareddy Mopuri

Backward pass

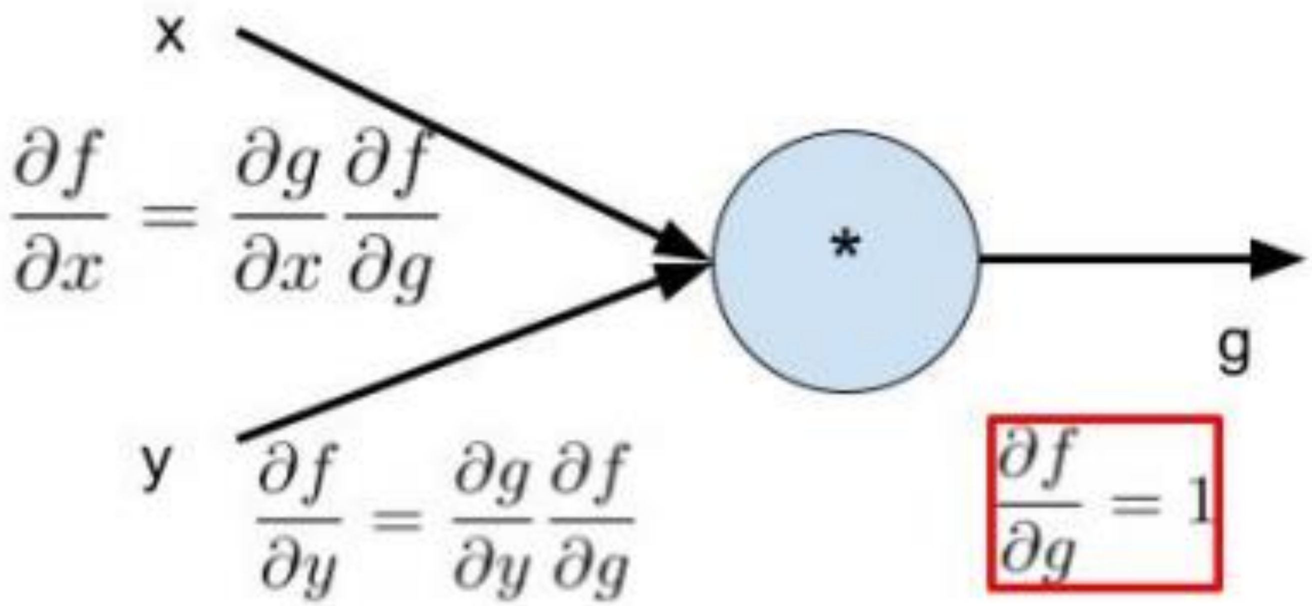
- Compute the derivatives $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = xy + z$$
$$g = xy$$
$$f = g + z$$



Credits : DL4CV , Dr. Kondareddy Mopuri

- down steam gradient = local gradient × upstream gradient



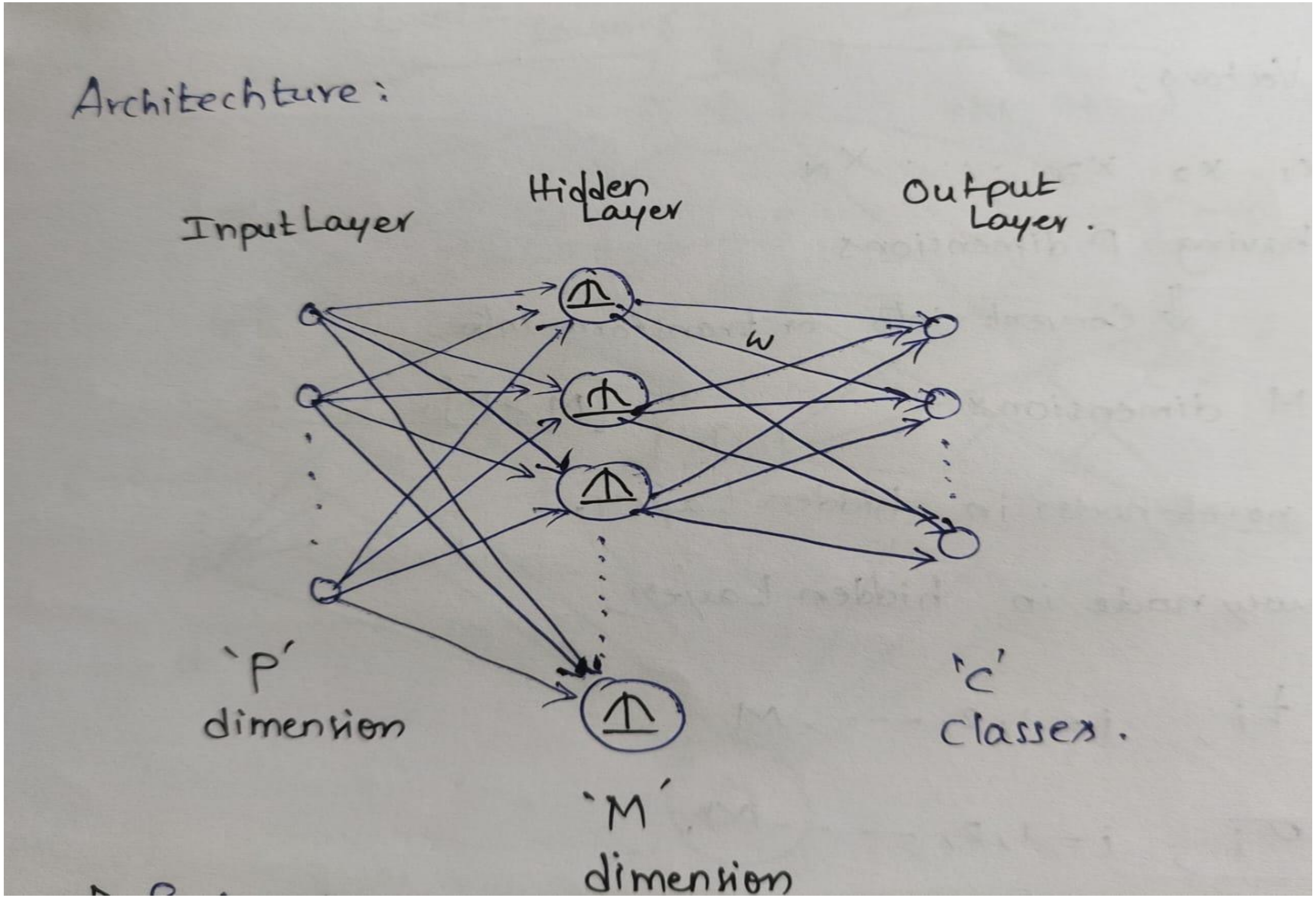
$$g_d = g_l \times g_u$$

Credits : DL4CV , Dr. Kondareddy Mopuri

RBF Networks

- In MLP , more hidden layers represents a non-linear boundary by set of piece wise linear boundaries.
- RBF performs a non-linear transformation over input vectors before the input vectors are fed for classification. Using such non-linear transformations, it is possible to convert linearly non separable problem into a linearly separable problem.
- In RBF , for each feature vector X , we impose some M number of RBF's, each of them produce a real values. I.e M number of real valued components.

RBF Architechture



Training Procedure:

- Part 1: While training hidden layer, for each nodes we have to find receptor t, spread sigma.

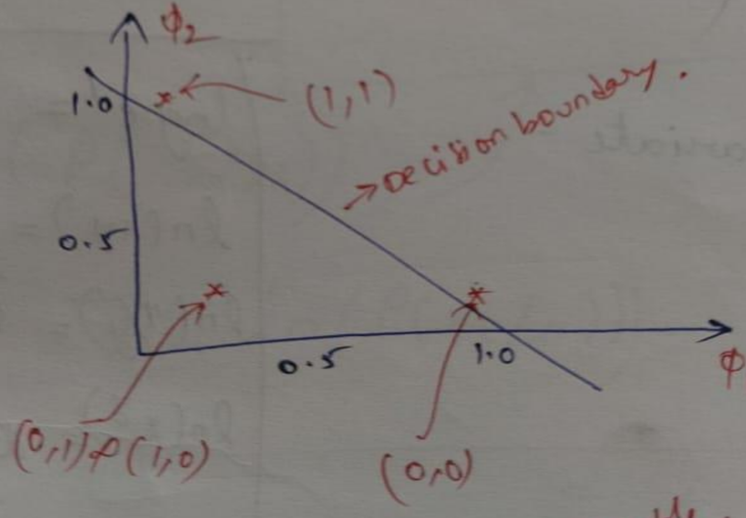
$$\phi_i(X) = e^{\frac{-\|X - t_i\|^2}{2\sigma^2}}$$

- Part 2: The weight vectors which connects outputs of hidden layers nodes to the output layer nodes. Simply it is a linear combination of outputs of hidden layer nodes.

- For example we can analyze XOR by using RBF network
- Take 0,0 and 1,1 as receptors and perform RBF on every feature vector so that it will transform into some non linear space for linear separability.
- Here we have 2 RB functions and each of them calculates the output by taking gaussian function as choice of Radial Basis Function.

$\underline{x}$		$\phi_1(\underline{x})$	$\phi_2(\underline{x})$
0	0	1	0.135
0	1	0.367	0.367
1	0	0.367	0.367
1	1	0.135	1

After the mapping we will get,



Linearly separable.

Thank You