Day 2

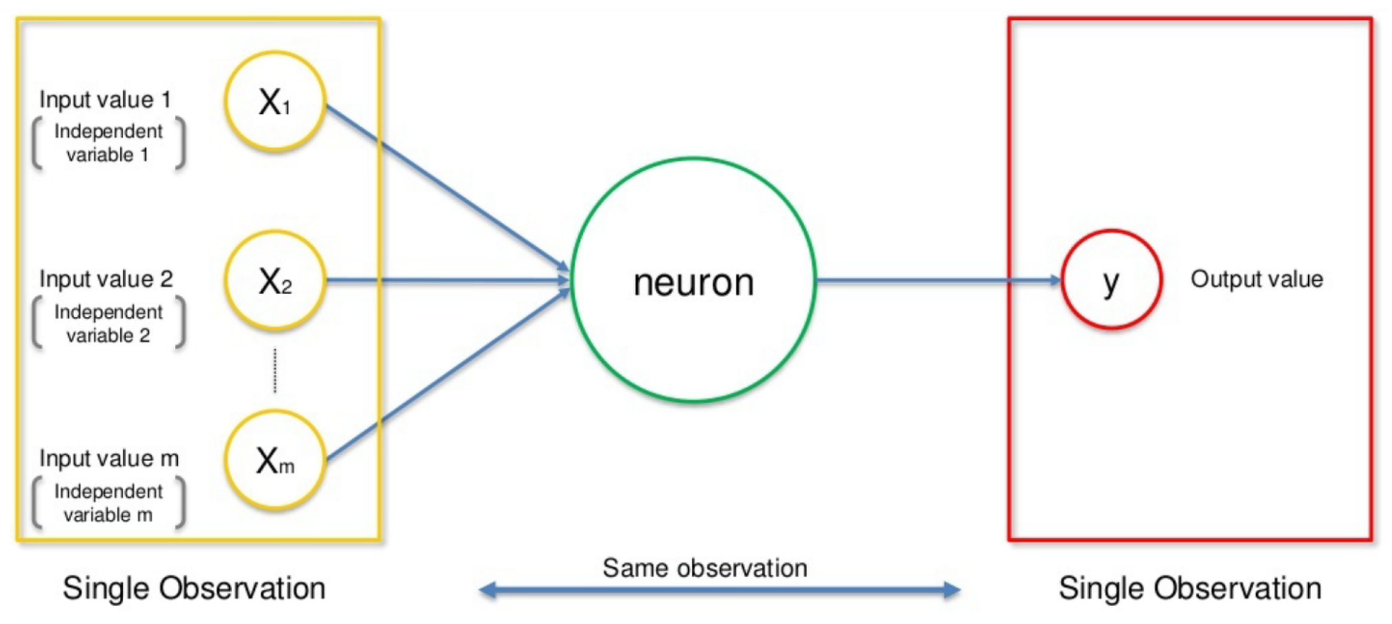
Agenda :

1. Introduction to Deep Learning
2. ANN
3. Perceptron
4. Multiplayer Perceptron
5. Notation
6. How to train Single layer Neuron
7. Optimisation
8. MCQs.

Artificial Neural Networks:

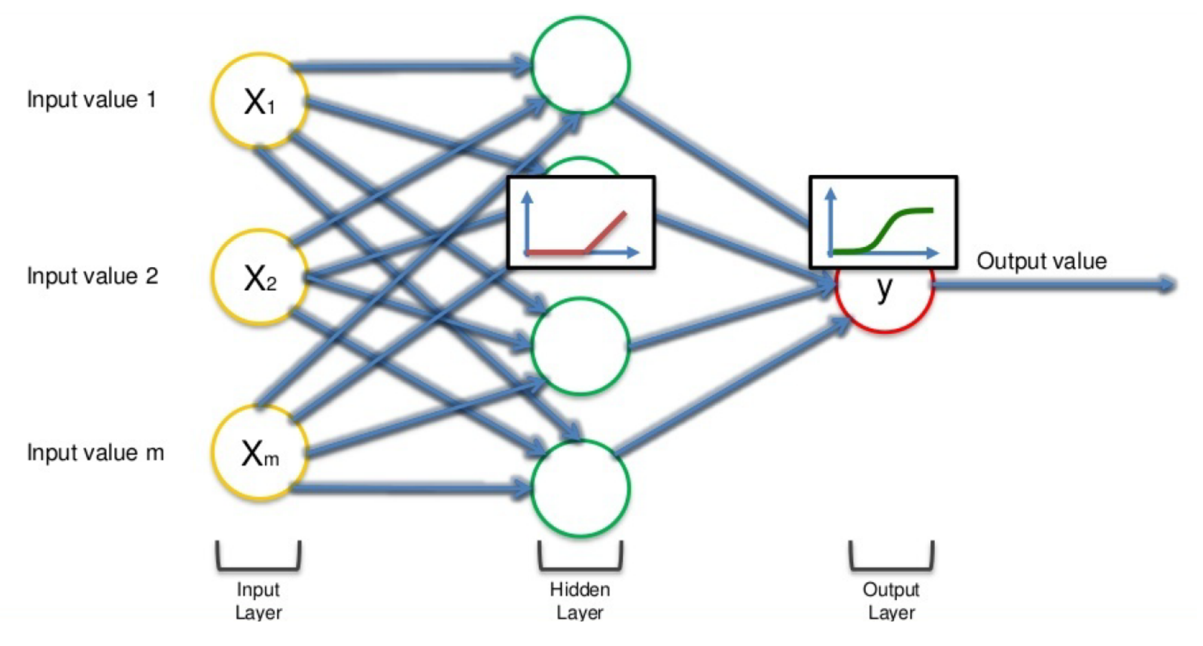
-- Artificial Neural Networks (ANN)are the basic algorithms and also simplified methods used in Deep Learning (DL) approach. We have come across more complicated and high-end models in the DL approach. However, ANN is a vital element of the progressive procedure and is the first stage in the DL algorithm.

Neural Network – Basic foundation to understand:

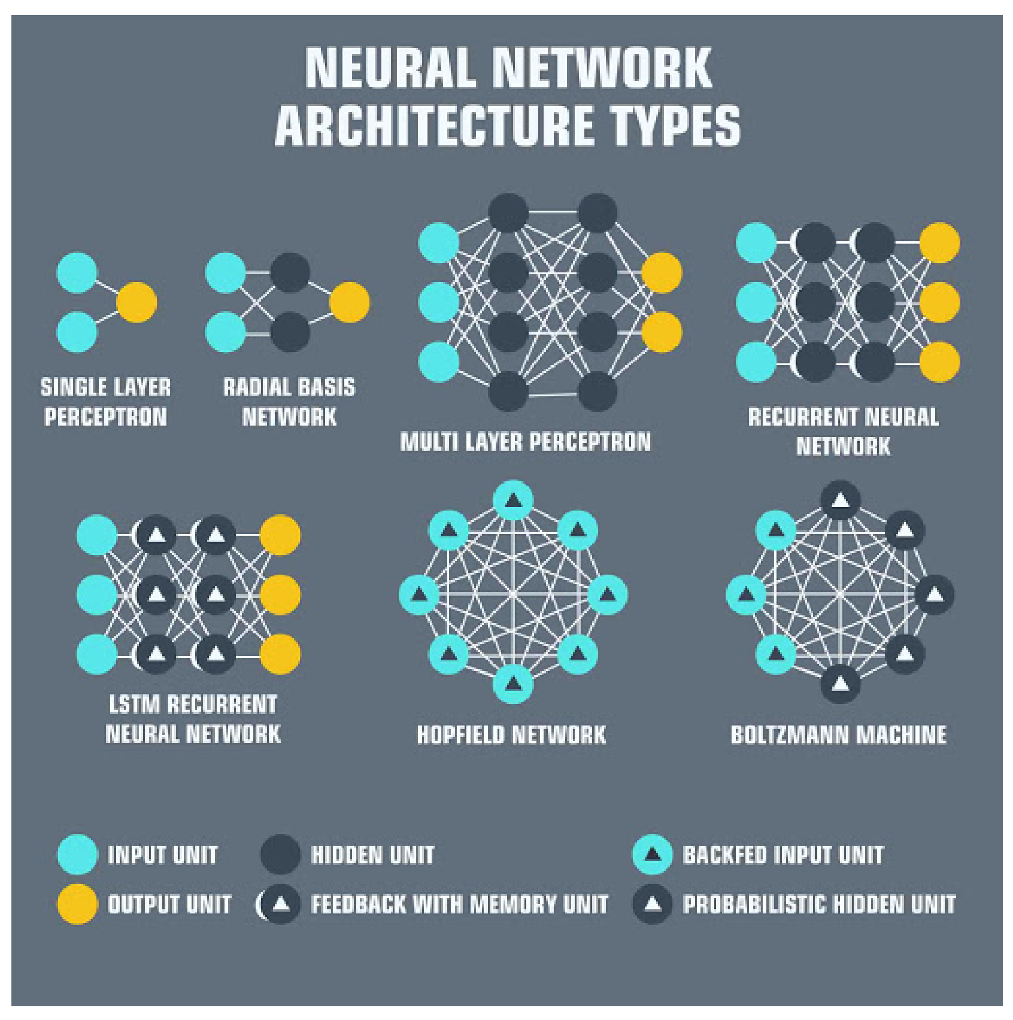


To summarize in short, how neural network takes place

1. Signals from multiple neurons like x1, x2, x3... xn, along with associated weights w1, w2, w3... wn are transmitted to the successive neurons. (Input layers)
2. So after we are receiving multiple signals (here consider it as neurons), with weights associated with it, we will be summing up all the signals with weights, and that's our step 1 as shown in the above figure (Hidden layers).
3. Then it applies the activation function as step 2 as shown in the above figure (Hidden layers).
4. From the above step neuron understand if it needs to pass the signal or not.
5. If the neuron passes the signal (that's to another neuron), that's our output (Output layers).
6. The above process keeps on repeating until the last neuron or the part of our body here.

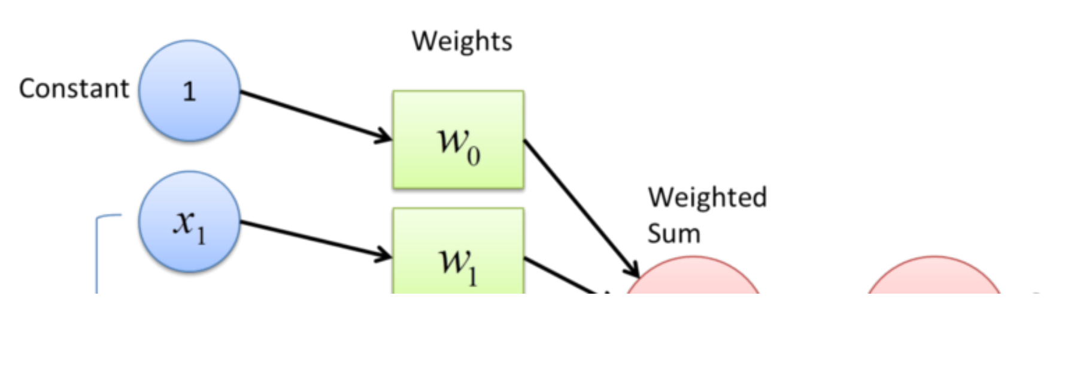


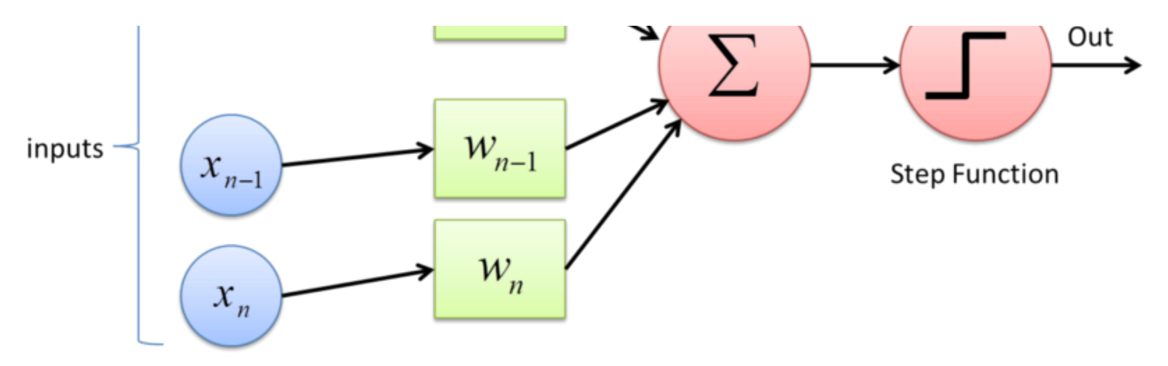
Finally, NN is a sequence of algorithms that imitate human brain functions to recognize connections between huge volumes of data. This is the basis for the DL approach, and NN can be categorized as stated below based on the architecture,



Perceptron

-- Perceptron is a single layer neural network and a multi-layer perceptron is called Neural Networks.

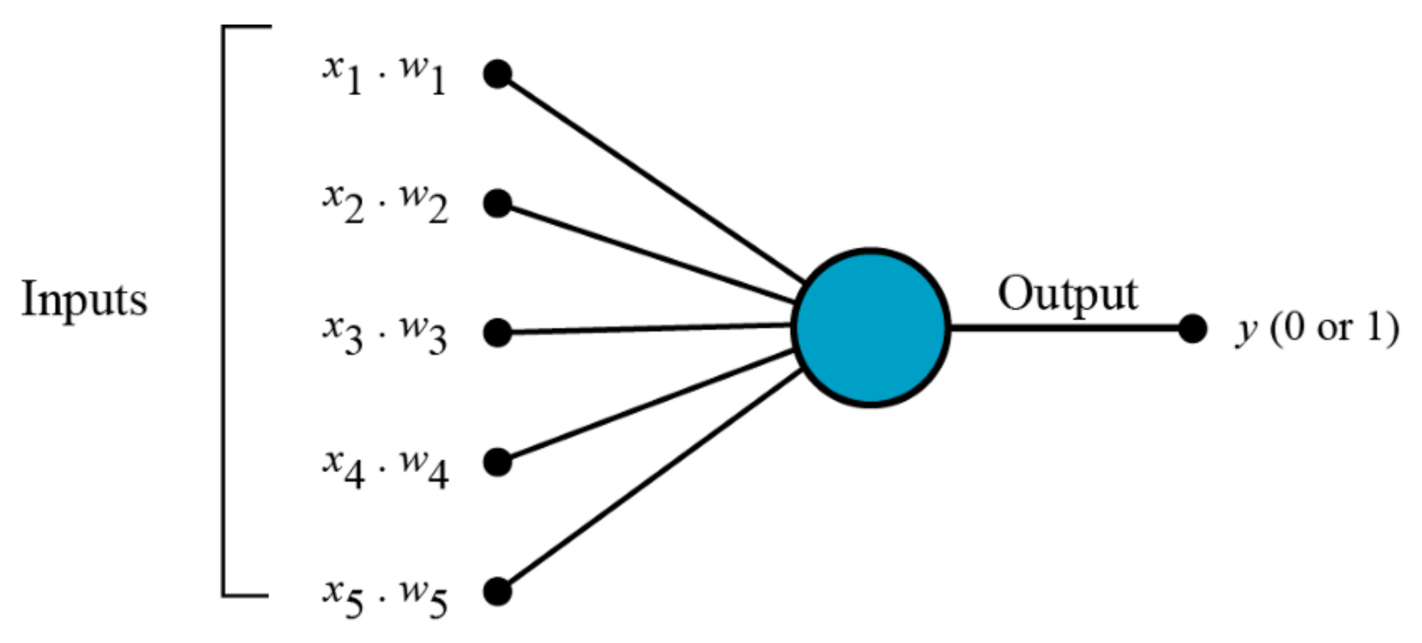




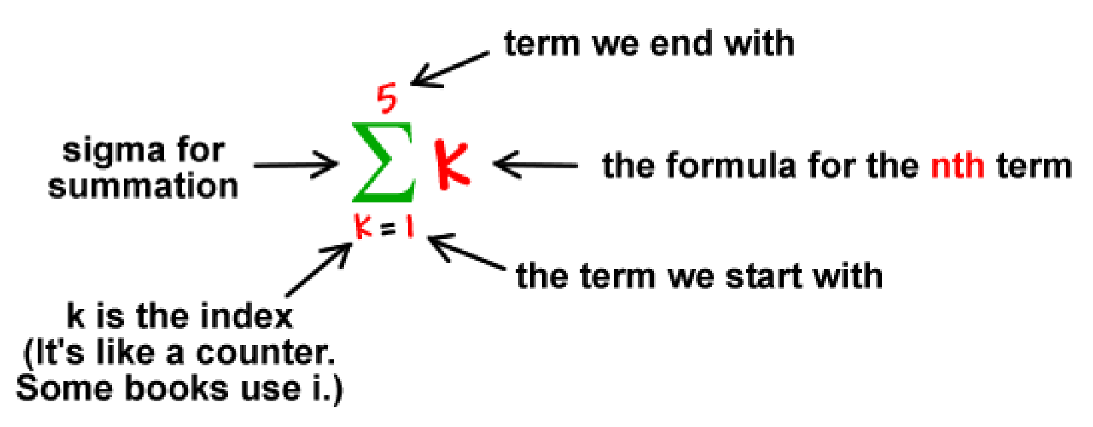
But how does it work?

The perceptron works on these simple steps

a. All the inputs x are multiplied with their weights w. Let's call it k.



b. Add all the multiplied values and call them Weighted Sum.

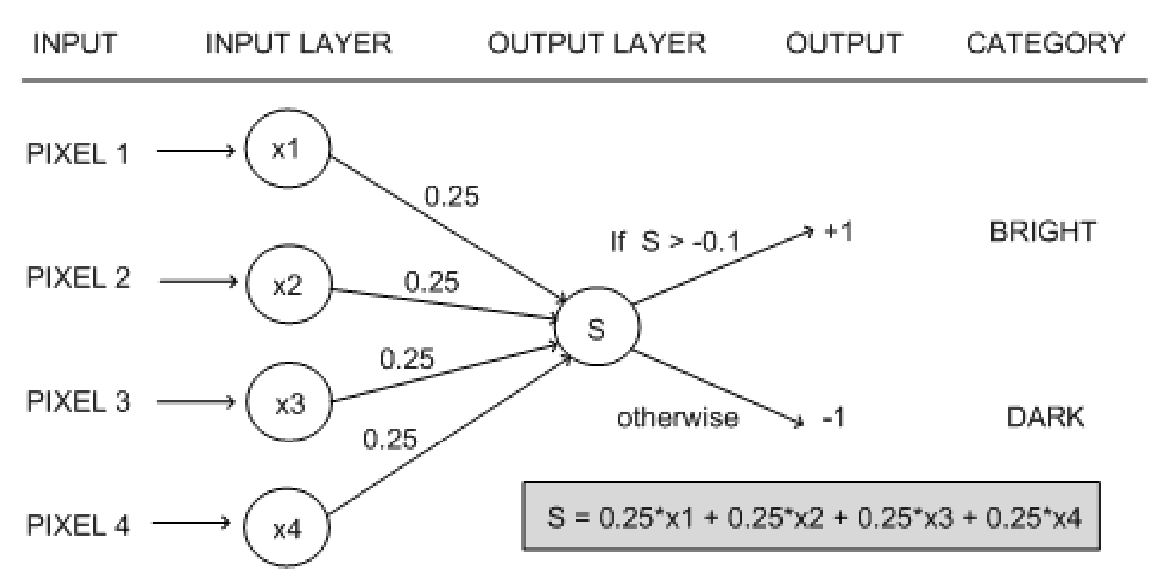

term we end with
sigma for summation
$\sum_{k=1}^{5} k$
the formula for the nth term
the term we start with
k is the index (It's like a counter. Some books use i.)

c. Apply that weighted sum to the correct Activation Function.

Why do we need Weights and Bias?

-- Weights shows the strength of the particular node.

-- A bias value allows you to shift the activation function curve up or down.

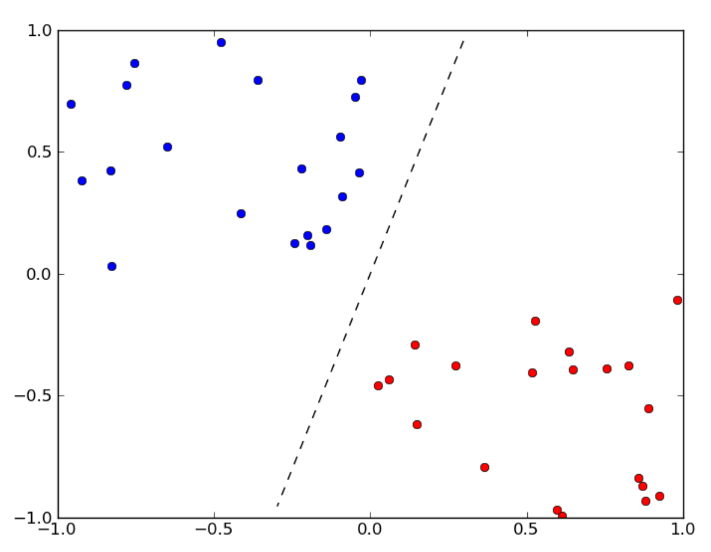


Why do we need Activation Function?

--- In short, the activation functions are used to map the input between the required values like (0, 1) or (-1, 1).

Where we use Perceptron?

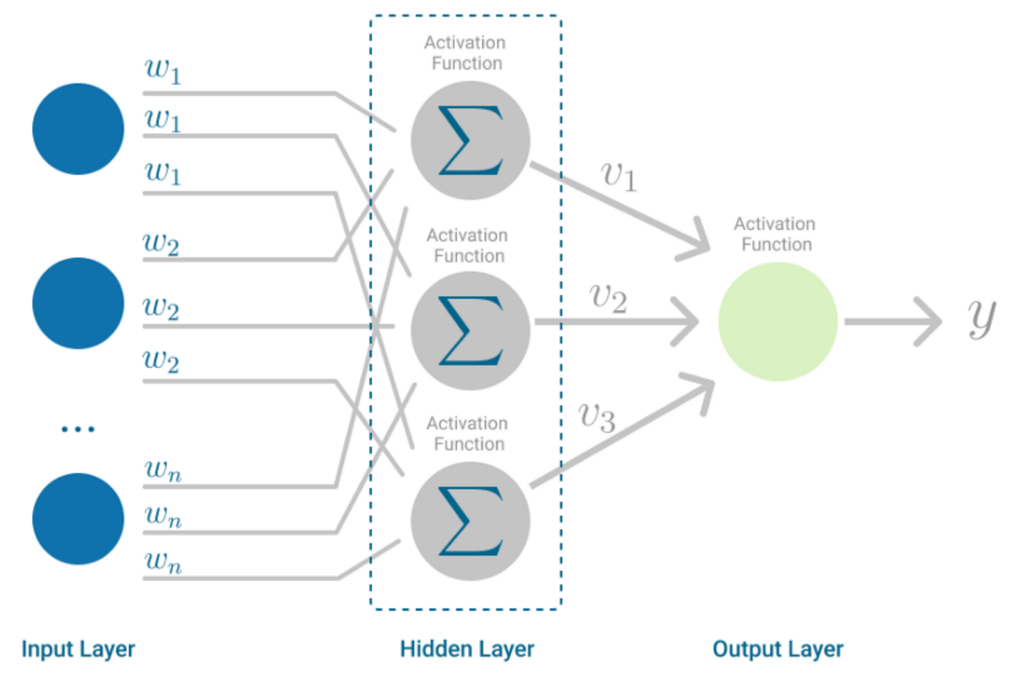
-- Perceptron is usually used to classify the data into two parts. Therefore, it is also known as a Linear Binary Classifier.



Multi Layer Perceptron

--- The Multilayer Perceptron was developed to tackle this limitation. It is a neural network where the mapping between inputs and output is non-linear.

--- A Multilayer Perceptron has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function.

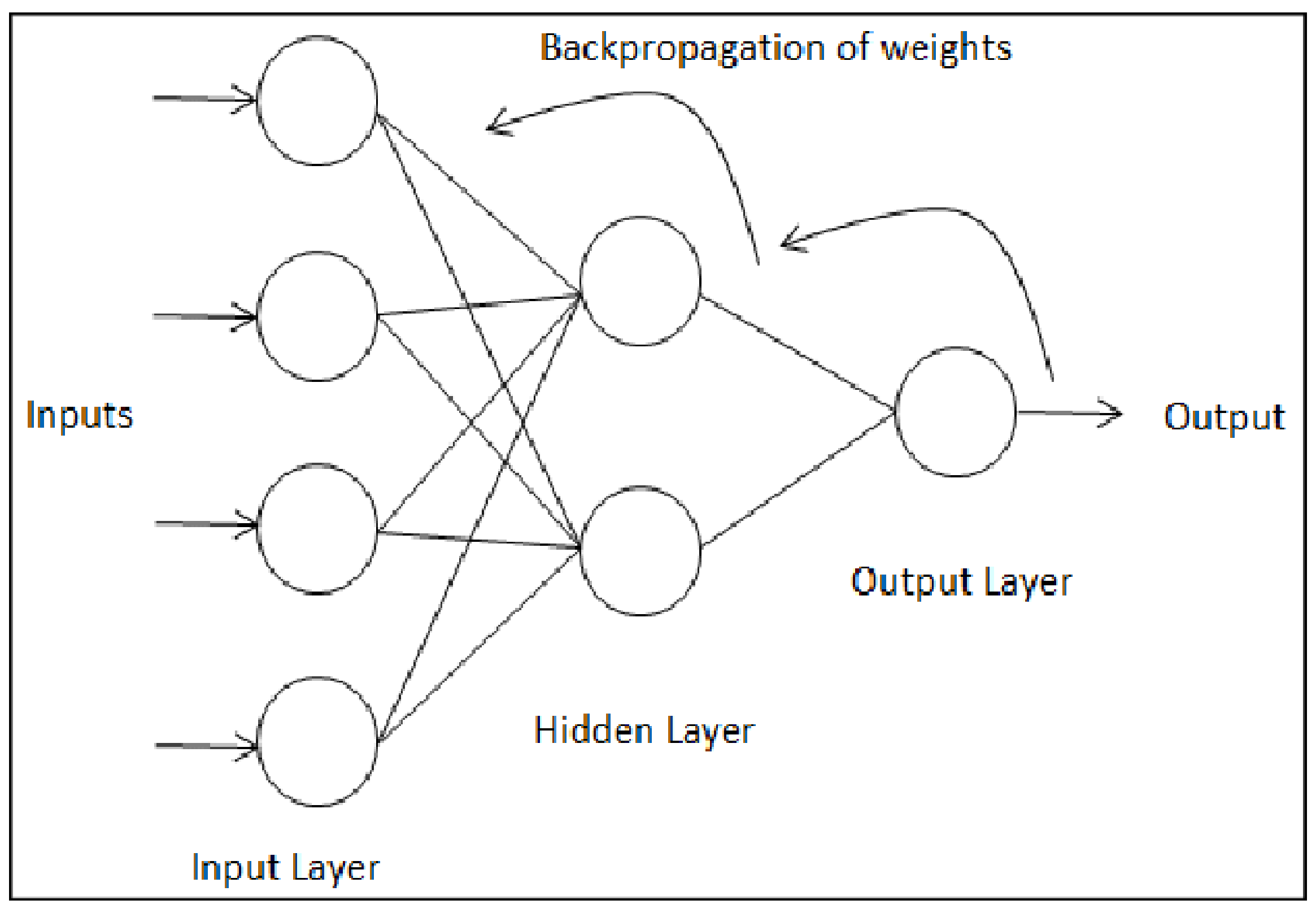


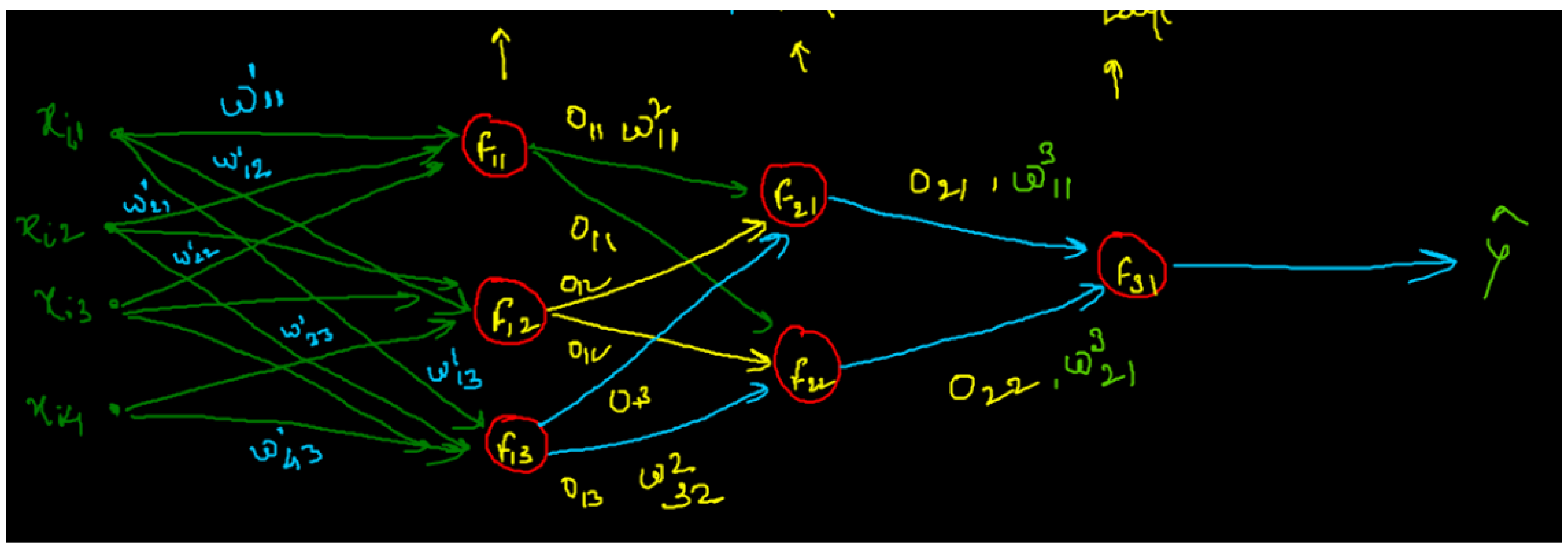
-- Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer.

-- Each layer is feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.

-- If the algorithm only computed the weighted sums in each neuron, propagated results to the output layer, and stopped there, it wouldn't be able to learn the weights that minimize the cost function. If the algorithm only computed one iteration, there would be no actual learning.

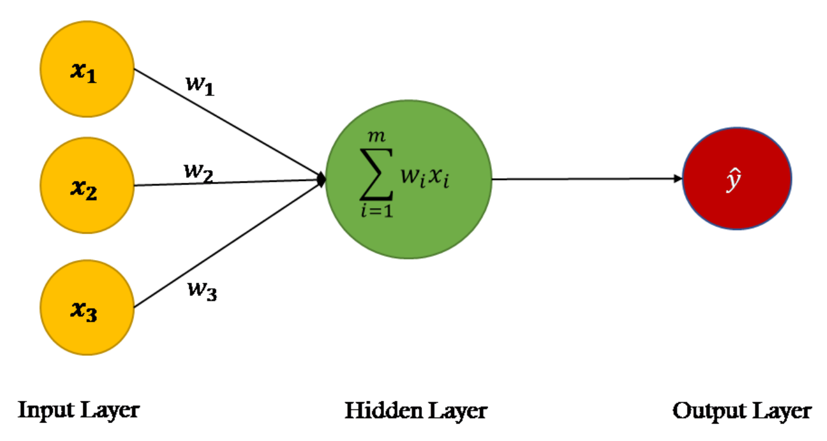
Notation





How to train Single Neuron Model :

Aim : Find the best edge weight.



Step 1. Define Loss function :

$$L = \sum (Y - Y_p)^2 + \text{Reg}$$

Step 2. Optimization :

$$\text{Min} \sum_{i=1}^{n} (Y - Y_p)^2 + \text{Reg}$$

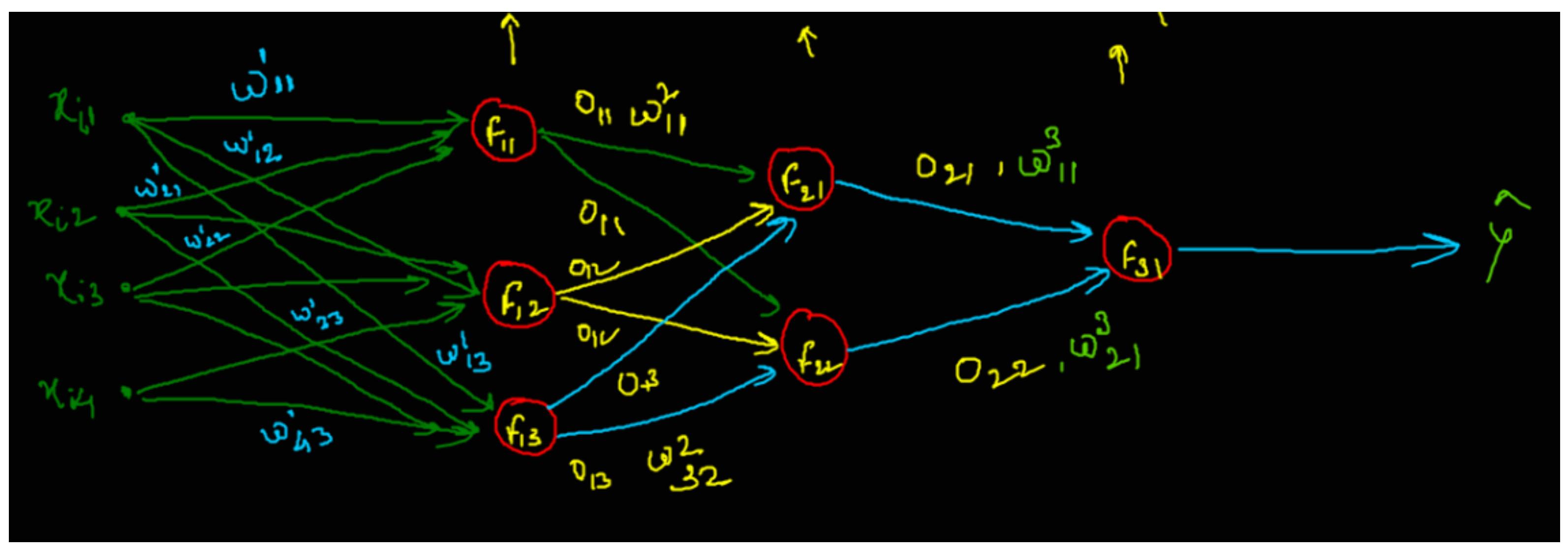
$Y_p = f(w^T * x_i)$

Step-3

→ Refer to Optimisation PdF.

Memoization

Is there any Operation that is used many times repeatedly.
It's Good idea to compute it once and store it and reuse it.



We learnt Chain Rule + memoization

Backpropagation

Let D = $\{X_i, Y_i\}$

Step-1 Initialize W_{ij}^{k}

Step-2 For each X_i in D (randomly select any(x_i, y_i) then run) :

(a) Pass X_i forward through the neuron
(b) Compute (Y_p, Y_i)
(c) Compute all the derivative using chain rule and memoization.
(d) Update weight from end to start of the network.

Step-3 Repeat step 2 until the convergence $(W_n == W_o)$

$$\boxed{(W)_n = (W)_o - n(dL/dW)}$$

Epoch :

- Each trail to learn from the input dataset is called an epoch.
- So an epoch refers to one cycle through the full training dataset. Usually, training a neural network takes more than a few epochs. Increasing the number of epochs doesn't always mean that the network will give better results.

Iteration :

- For each complete epoch, we have several iterations. Iteration is the number of batches or steps through partitioned packets of the training data, needed to complete one epoch.

Batch :

- Batch is the number of training samples or examples in one iteration. The higher the batch size, the more memory space we need.