- Used for Supervised learning
- Typical learning algo for MLP network also called BACK PROPOGATION ALGORITHM
- I/P layer receives I/P signals to be processed, then it is given to hidden layer.
- Hidden layer & O/P layer performs computational task
- O/P layer gives prediction or classification O/P
- Hidden layer is the computation Engine
- Arbitary no. of hidden layers are placed in b/w I/P layer & O/P layer, are the true computation Engine of MLP
- Data flow in forward direction (I/P ⟶ O/P)
- The neurons in MLP are trained with the back Propogation learning algo which can Solve problems which are not linearly Separable.
- The major areas of MLP are :
  1. Pattern Recognition.
  2. Classification
  3. Prediction & Approximation.
- The computation taking place at Every neuron in hidden layer & O/P layer.

Hidden: $h(x) = S(b(1) + w(1))$ — I/P $\xrightarrow{W1}$ h(x)

hidden layer; b: bias; w: weight; S: Activation func:

Output: $O(x) = G(b(2) + w(2)h(x))$ — h(x) $\xrightarrow{W_2}$ O/P

output layer

- b(1), b(2) are bias vectors.
- w(1), w(2) are weight matrices.

• S & G are Activation function.

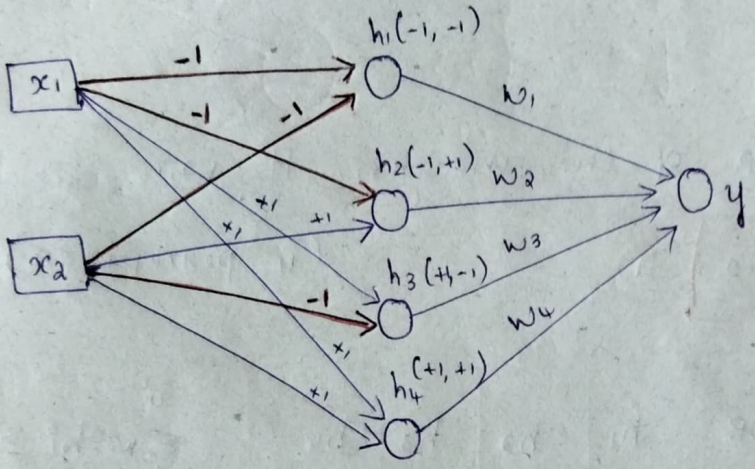
• Set of parameters to learn are

$$Set(\theta) = \{b(1), b(2), w(1), w(2)\}$$

by using back Propogation Algorithm.

• For MLP, we typically use TanH or Sigmoid activation function.

→ 2 I/P 4 Perceptron O/P layer



bias $w_0 = (-2)$

Black edge represents $w = -1$

blue edge indicates $w = +1$

Bias of each perceptron $w_0 = -2$.

* Each perceptron will fire only if weighted sum of I/P is greater than or equal to -2.
* Each of these perceptron in the hidden layer is connected to the perceptron in output layer by weights.
* All these weight should be learnt.
* ~~Red~~ Black & blue edges are called layer1 weights.
* w_1, w_2, w_3, w_4 are called layer2 weights.
* Checking this n/w can be used to implement any Boolean func. (Linearly Separable or not)

> Each perceptron in the hidden layer fires

only for the specific input

(for h_1, $x_1 = -1$ & $x_2 = -1$ h_3 $x_1 = +1$ & $x_2 = -1$
h_2, $x_1 = -1$ & $x_2 = +1$ h_4 $x_1 = +1$ & $x_2 = +1$
)

XOR Operation

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

w_0 is the bias o/p of the neuron. It will fire if

$$\sum_{i=1}^{4} w_i h_i \geq w_0 \quad (1)$$

(4 perceptron)

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^{4} w_i h_i \geq w_0$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3
1	1	0	0	0	0	1	w_4

} weight factors

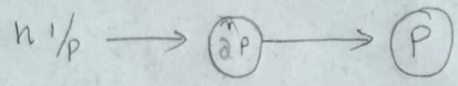
* Those results in the 4 condition to implement

$w_1 < w_0 (0)$
$w_2 \geq w_0 (1)$
$w_3 \geq w_0 (1)$
$w_4 < w_0 (0)$

* Each w_i is now responsible for one of the 4 possible inputs & can be adjusted to get the desired o/p for that i/p.

Theorem:

Any Boolean function of 'n' input can be represented exactly by a network of perceptron containing one hidden layer with 2^n perceptrons & one output layer containing one perceptron



Eg: A person would like to buy a car or not with single input x ⟶ Salary.

Sal in $\frac{K}{1000}$	y (buy or not)
80	1
20	0
65	1