Supervised Learning Networks

Perceptron Network

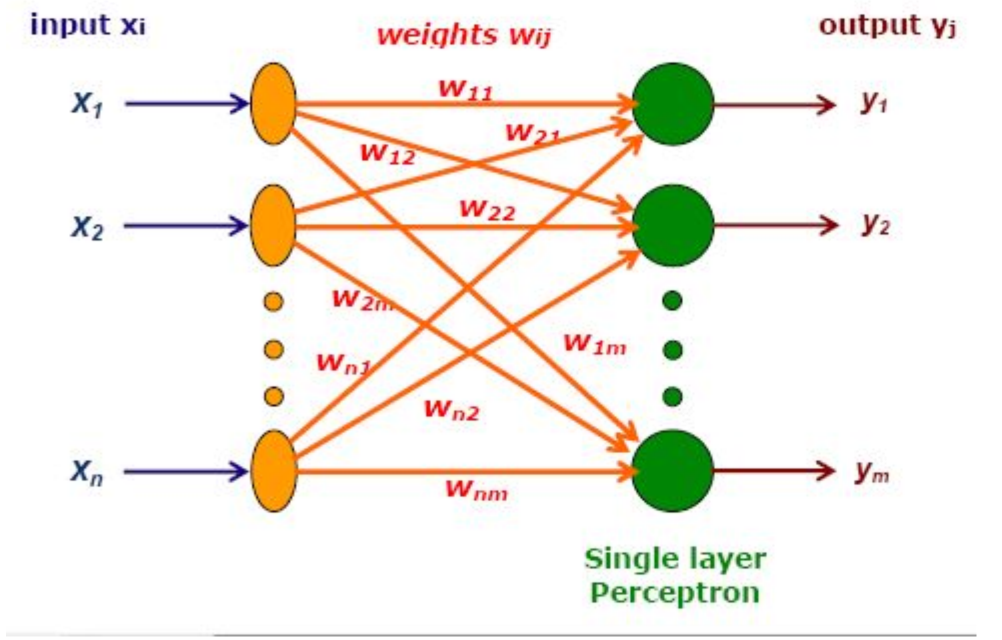
- Perceptron network comes under single layer feed forward networks.
- The different types of perceptron are designed by Rosenblastt and Minsky-Papert.
- Original network consists of 3 units
- Sensory unit (input unit)
- Associator unit (hidden unit)
- Response unit (output unit)
- Sensory and Associator unit has fixed weights having values 1,0,-1.

- Binary activation function is used in sensory and associator unit
- The output signals sent from associator unit to the response unit are only binary.
- Output of perceptron is given by

  y=f(yin)
- Perceptron learning rule is used in the weight updation between associator unit and response unit.
- The error calculation is based on the comparison of the values of targets with those of calculated outputs.
- Eventually learning rules reaches near optimal or optimal solution in finite number of steps.
- Response unit act as pattern recognizer while associator act as a feature detectors.

Single layer perceptron

- An arrangement of one input layer of neurons feed forward to one output layer of neurons is known as Single Layer Perceptron.



$$y_j = f(net_j) = \begin{cases} 1 & \text{if } net_j \geq 0 \\ 0 & \text{if } net_j < 0 \end{cases} \quad \text{where } net_j = \sum_{i=1}^{n} x_i \, w_{ij}$$

Learning Algorithm: Training Perceptron

- The training of perceptron is a supervised learning algorithm where weights are adjusted to minimize error when ever the output does not match the desired output.

- If the output is correct the of weight is done.

  i.e. $W_{ij}^{K+1} = W_{ij}^{K}$

- If the output is 1 but should have been 0 then the weights are decreased on the active input link

  i.e. $W_{ij}^{K+1} = W_{ij}^{K} - \alpha . x_i$

- If the output is 0 but sl he weights are increased on the active input link

  i.e. $W_{ij}^{K+1} = W_{ij}^{K} + \alpha . x_i$

Where W^{K+1} is the new adjusted weight, W^{K} is old weight.

Xi is the input and α is the learning rate parameter.

α small leads to slow and α large leads to fast learning.

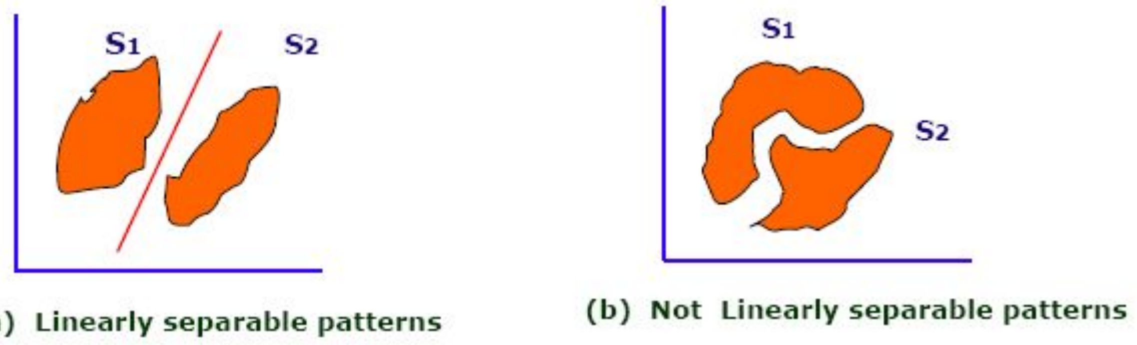
Perceptron and linearly separable task

- Perceptron can not handle tasks which are not separable.

- Sets of points in 2-D space are linearly separable if the sets can be separated by straight line.

- Generally, a set of points in n-dimensional space are linearly separable if there is a hyper plane of (n – 1) dimensions separates the sets.

  Example:



(a) Linearly separable patterns (b) Not Linearly separable patterns

Algo for training process for perceptron network with single output

Step 1: initialize the weights & the bias. Also initialize the learning rate α ($0 < \alpha \leq 1$).
Step-2: perform steps 3-7 until the final stopping condition is false.
Step-3: perform steps 4-6 for each training pair indicated by **s:t**
Step 4: The input layer containing input units is applied with identity activation functions: $x_i = s_i$
Step 5: calculate the output of the network. To do so, first obtain the net input

$$Y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

Where 'n' is the number of input neurons in the input layer. Then apply activation function over the net input calculated to obtain the output:

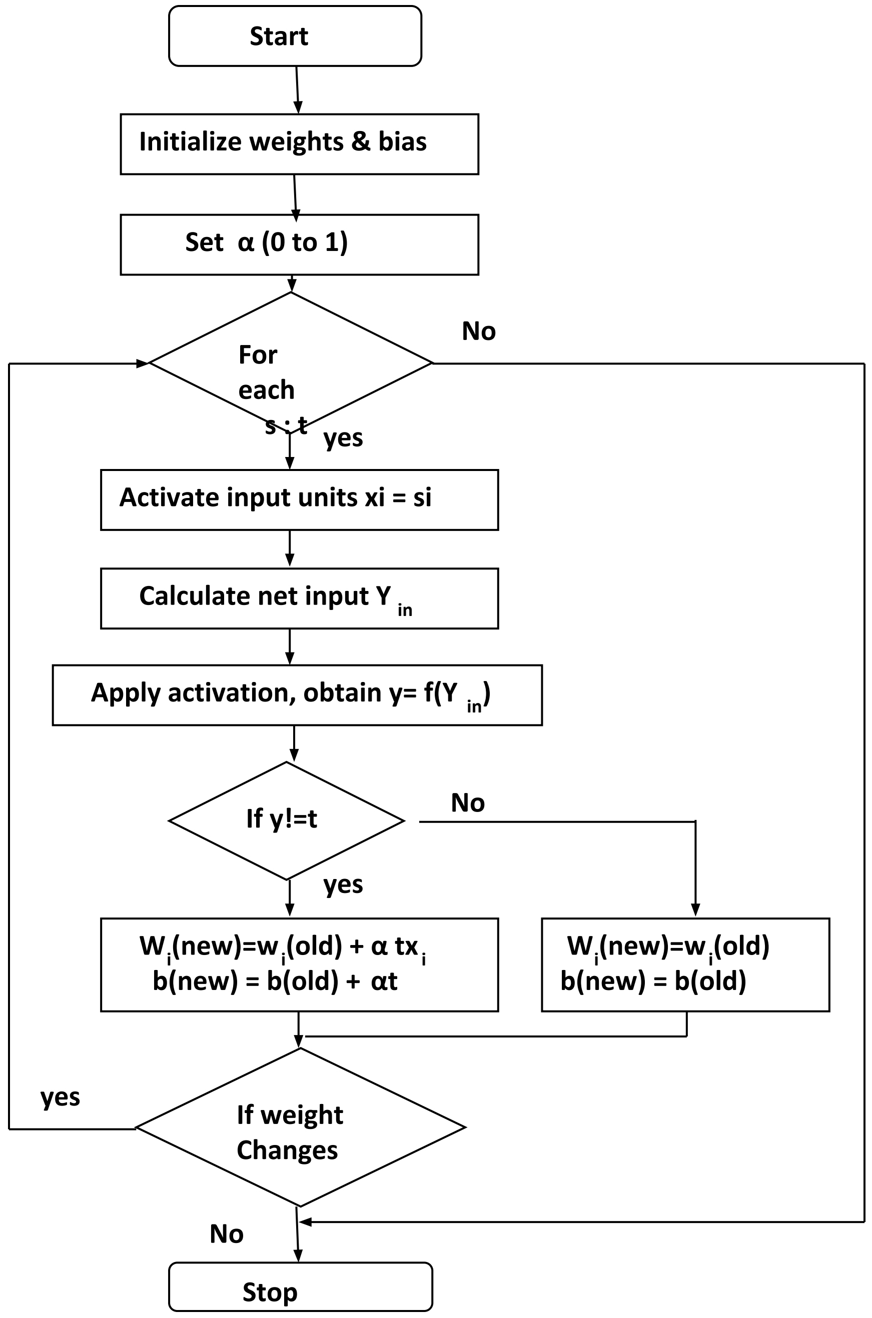
$$Y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \Phi \\ 0 & \text{if } -\Phi <= y_{in} <= \Phi \\ -1 & \text{if } y_{in} < -\Phi \end{cases}$$

Step 6: weight and bias adjustment: Compute the value of the actual (calculated) output and desired (target) out put.

if $y != t$, then $w_i\,(new) = w_i(old) + \alpha\, t x_i$ and $b\,(new) = b(old) + \alpha t$
Else, we have $w_i\,(new) = w_{ij}(old)$ $b\,(new) = b(old)$

Step 7: Train the network until there is no weight change. This is stopping condition.

Flow chart for training process for perceptron network with single output



Start
Initialize weights & bias
Set α (0 to 1)
For each s : t
No
yes
Activate input units xi = si
Calculate net input Y_{in}
Apply activation, obtain y= f(Y_{in})
If y!=t
No
yes
W_i(new)=w_i(old) + α tx_i
b(new) = b(old) + αt
W_i(new)=w_i(old)
b(new) = b(old)
If weight Changes
yes
No
Stop

Perceptron Network testing Algorithm

It is best to test the network performance once the training process is complete.
For efficient performance of the network, it should be trained with more data.

Step 0: The initial weights to be used here are taken from the training algorithms (the final weights obtained during training)

Step1: For each input vector X to be classified, perform Step 2-3.

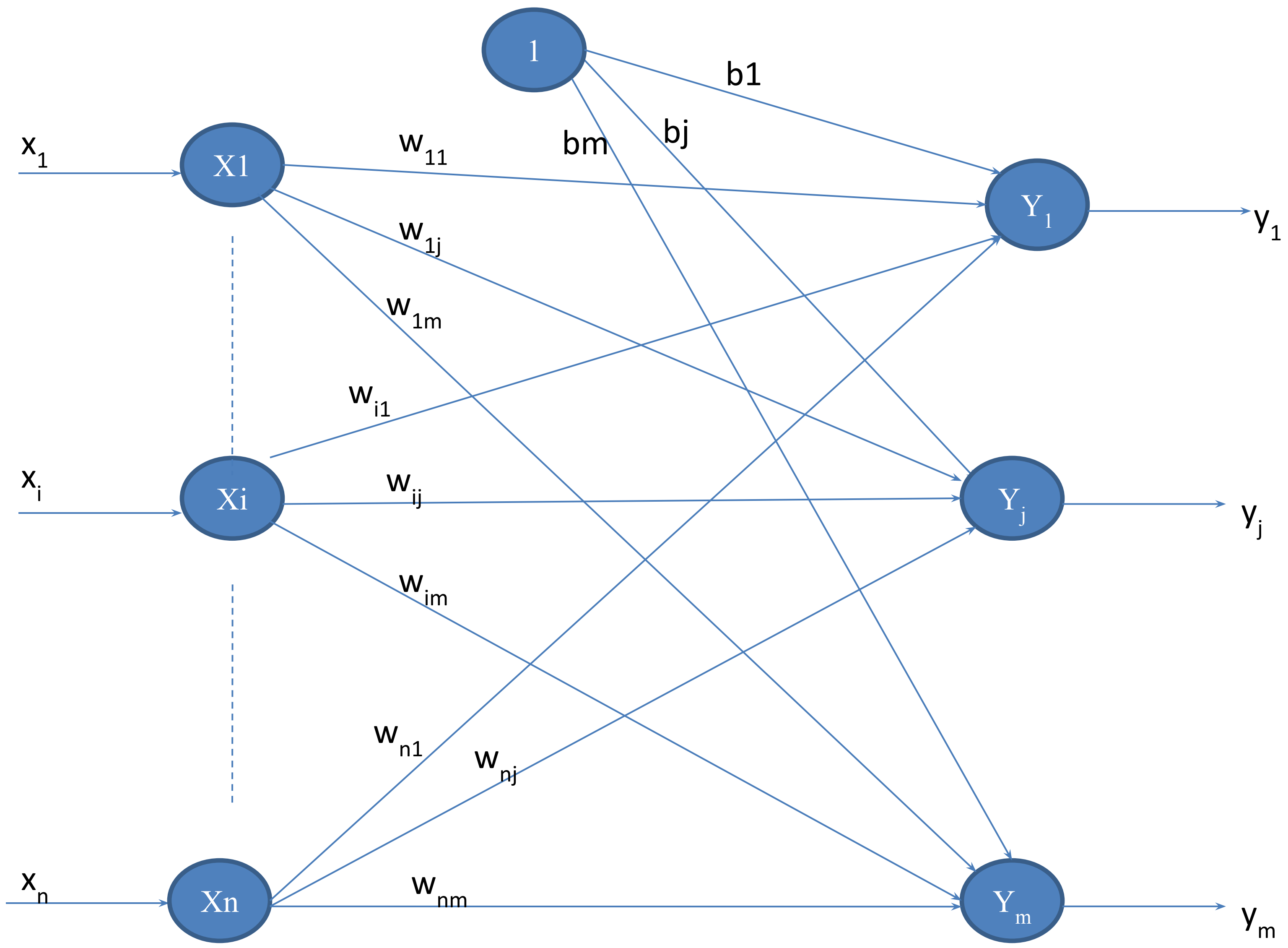
Step2: Set activations of the input unit

Step 3: Obtain the response output unit

$$Y_{in} = \sum_{i=1}^{n} x_i w_i$$

$$Y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \Phi \\ 0 & \text{if } -\Phi <= y_{in} <= \Phi \\ -1 & \text{if } y_{in} < -\Phi \end{cases}$$

Network Architecture for perceptron network for several output classes



Perceptron Training Algorithm for Multiple Output Classes

Step 0: Initialize the weights, biases and learning rate suitably.
Step 1: Check for stopping condition; if it is false, perform Steps 2-6.
Step 2: Perform Steps 3-5 for each bipolar or binary training vector pair s:t.
Step 3: Set activation function (identity) of each input unit $i = 1$ to n:

$$x_i = s_i$$

Step 4: Calculate output response of each output unit $j = 1$ to m: first, the net input is calculated:

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_i$$

Then activations are applied over the net input to calculate the output response

$$y_j = f(y_{inj}) = \begin{cases} 1 & \text{if } y_{inj} > \Phi \\ 0 & \text{if } -\Phi <= y_{inj} <= \Phi \\ -1 & \text{if } y_{inj} < -\Phi \end{cases}$$

Step 5: Make adjustment in weight and bias for $j=1$ to m and $i = 1$ to n

if $t_j \,!= y_j$,

then $w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha\, t_j x_i$

$b_j(\text{new}) = b_j(\text{old}) + \alpha t_j$

Else, we have

$w_{ij}(\text{new}) = w_{ij}(\text{old})$

$b_j(\text{new}) = b_j(\text{old})$

Step 6: Test for the stopping condition, i.e., if there is no change in weights then stop the training process, else start again from Step 2.

It can be noticed that after training, the net classifies each of the training vectors.

Perceptron netwrok can be used for liear seperaility concept.

Seperating or decision line ay be bbased on value of threshold.

Threshold used in activation function must be a non-negative value.