Supervised Learning Network.

Perceptron Networks

The perceptron was first introduced by **Mr. Frank Rosenblatt** in 1957.

A Perceptron is an algorithm for supervised learning of binary classifiers. There are two types of Perceptrons: Single layer and Multilayer.

- **Single layer** - Single layer perceptrons can learn only linearly separable patterns
- **Multilayer** - Multilayer perceptrons or feedforward neural networks with two or more layers have the greater processing power

The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

This enables you to distinguish between the two linearly separable classes +1 and -1.

Perceptron algorithm (Single layer)

Step 0: initialize the weights and the bias(for easy calculation they can be set to zero). also initialize the learning rate α(0, α, 1) for simpicity α is set to 1.

Step 1: Perform step 2 to 6 until the final stopping condition is false.

Step 2: Perform step 3 to 5 each training pari indicated by s:t

Step 3: The input layer containing input units is applied with identity activation function: $x_i = s_i$

Step 4: calculate the output of the network. to do so first obtain the net input:

$$y_{in} = \sum_{i}^{n} x_i . w_i + \mathrm{b}$$

where n is the number of inputs neurons in the input layer. Then apply activation over the net input calculated to obtain the output

$$f(y_{in}) = \begin{cases} 1 & if\, y_{in} > \theta \\ 0 & if\ -\theta \leqslant y_{in} \leqslant \theta \\ -1 & if\, y_{in} < -\theta \end{cases}$$

Step 5: weight and bias adjustment: compare the value of the actual (calculated) output and desire (targer) output

```
if y ≠ t, then
        $w_i(new) = w_i(old) + \alpha t x_i$
        $b(new) = b(old) + \alpha t$
else we have
        $w_i(new) = w_i(old)$
        $b(new) = b(old)$
```

Step 6: Train the network until there is no weight change. This is the stopping condition for the network. If this condition is not met, then start again form step 2.

Perceptron algorithm (Multiple layer)

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 (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 as

$$y_{inj} = \sum_{i}^{n} x_i . w_{ij} + b_j$$

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

$$f(y_{in}) = \begin{cases} 1 & if\, y_{in} > \theta \\ 0 & if\, -\theta \leqslant y_{in} \leqslant \theta \\ -1 & if\, y_{in} < -\theta \end{cases}$$

Step 5: Make adjustment in weights and bias for j = l to m and i = l to n.

if $y_j \neq t_j$, then

w_{ij}(new) = w_{ij}(old)+
$\alpha t_j x_i$ b_j(new) =
b_j(old)+ αt_j

else we have

w_{ij}(new) = w_{ij}(old)
b_j(new) = b_j(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.