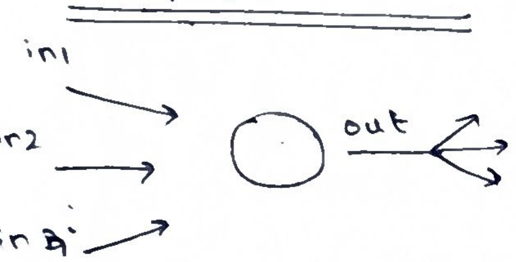
Soft Computing

Unit - 2

Neural Networks

Supervised Learning Neural Networks : Perceptrons - Adaline - Backpropagation Multilayer Perceptrons - Radial Basis Function Networks - Unsupervised Learning Neural Networks - Competitive Learning Networks - Kohonen Self - Organizing Networks - Learning Vector Quantizations - Hebbian Learning

* Artificial Neurons



→ An artificial neuron has the following characteristics:

(i) A neuron is a mathematical function modeled on the working of biological neurons - an elementary unit in an ANN.

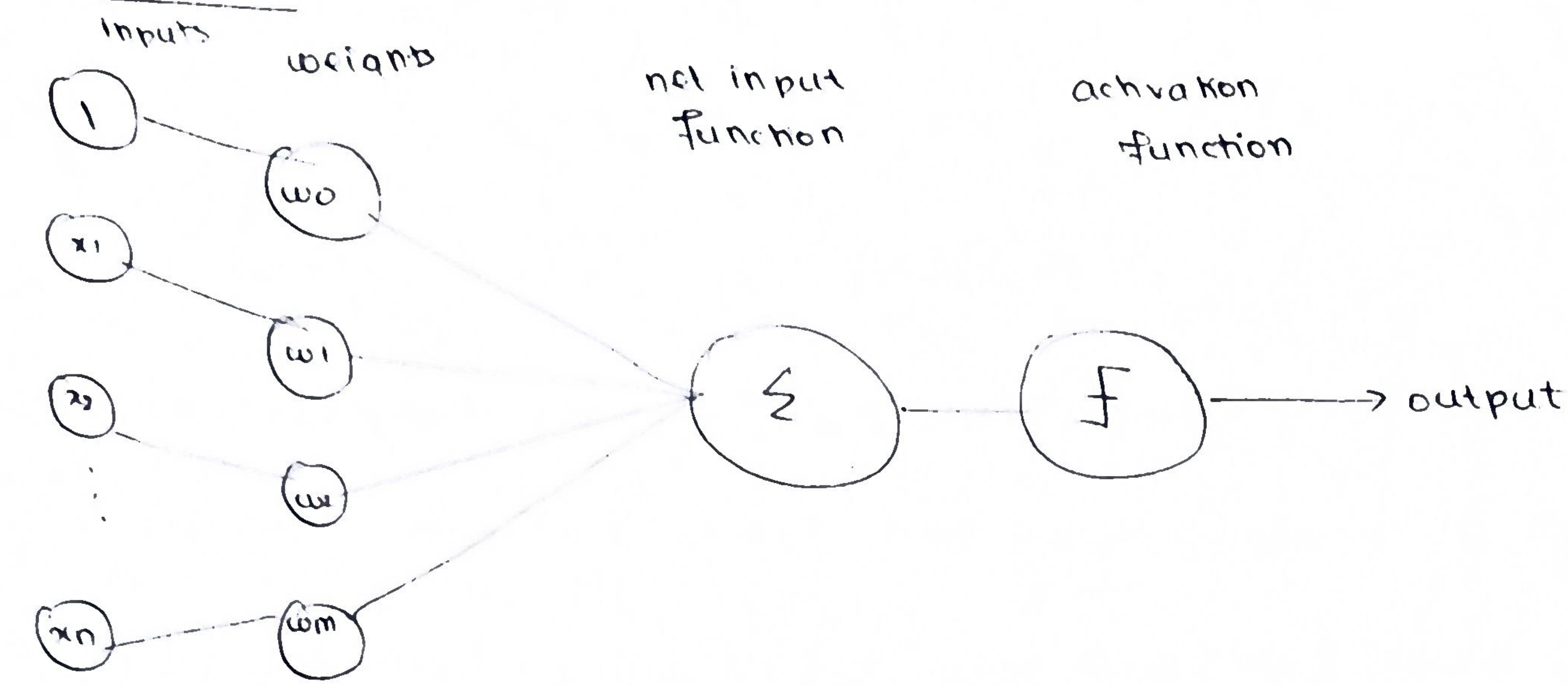
→ One or more inputs are separately weighted

→ Inputs are summed and passed through a non- linear function to produce an output

→ Every neuron holds an internal state called the activation signal.

→ Each connection link carries info. about the input signal - neurons are connected to one another by connection links

* Perceptron



Basic Components of Perceptron

① Input Layer - consists of one or more input neurons, which receive input signals from the external world or from other layers of the neural network

② Weights - each input neuron is associated with a weight, which represents the strength of the connection between the input and output neuron.

③ Bias - A bias term is added to the input layer to provide the perceptron with additional flexibility in modelling complex patterns in the input data.

④ Activation Function - determines the output of the perceptron based on the weighted sum of the inputs and the bias term. Common activation functions are : (i) step function
(ii) sigmoid function
(iii) RELU function

5) Output - o/p of the perceptron is a single binary value, either 0 or 1, which indicates the class or category to which the input data belongs.

6) Training Algorithm - train using a supervised learning algorithm such as the perceptron learning algorithm or backpropagation.

→ The weights and biases of the perceptron are adjusted to minimize the error between the predicted output and the true output for a given set of training examples.

Perceptron can be used for binary classification

* Types of Perceptrons

A. Single Layer - can learn only linearly separable patterns

B. Multilayer - can learn about two or more layers having greater processing power.

* Working of Perceptron

Step1: Multiply all input values with the corresponding weight values and then add to calculate the weighted sum.

i.e $\sum w_i x_i = w_1 x_1 + w_2 x_2 + \ldots$

Step2 Add a bias term to the weighted sum to improve the model's performance.

Step 3 : An activation function is applied with the weighted sum to give a binary or a continuous output.

$$y = f(\sum w_i x_i + b)$$

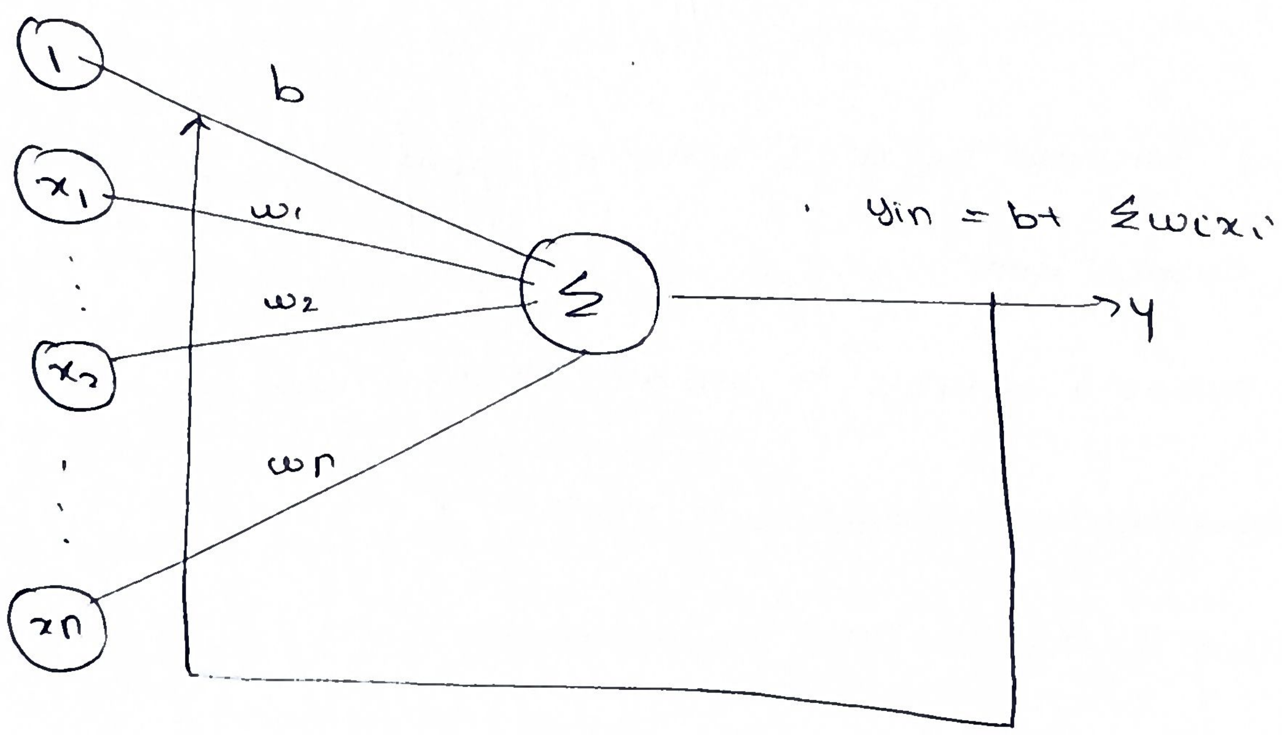
* Adaline - Adaptive Linear Neuron

→ The units with activation function are called linear units

→ A network with a single linear unit is called an adaptive linear neuron

→ The IO-relation is linear

Architecture



Error : $E_i = \sum (t - y_{in})^2$

Weight updation : $w_i(new) = w_i(old) + \alpha (t - y_{in}) x_i$

$$b(new) = b(old) + \alpha (t - y_{in})$$

* Backpropagation Multilayer Perception

→ Each training example is a pair of the form (x, t), where (x) is the vector of network input values and t is the target value.

→ η = learning rate

n_i = no. of network inputs

n_{hidden} = no. of units in the hidden layer

n_{out} = no. of output units

→ Input from unit i to unit j is denoted as x_{ji} and the weight from unit i to unit j is denoted as w_{ji}.

Algorithm

→ Create a feed forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.

→ Initialize the network weights to small random ~~nos~~ nos

→ Until the termination condition is met, do

For each (x, t) in training examples, do

1. Input the instance x to the network & compute the output o_u of every unit u in the network

2. Propagate the error backward

For each network unit k, calculate the error term δ_k

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

For each network unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k \in ops} w_{h,k}\,\delta_k$$

→ Update weights

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

$$\Delta w_{ji} = \eta\,\delta_j\,x_{ji}$$

* Unsupervised Learning

— when there is no information available regarding the desired outputs, unsupervised learning networks update weights only on the basis of input patterns.

* Competitive Learning Network

→ A scheme to achieve unsupervised data clustering or classification

→ The no. of inputs is the input dimensions & the no. of outputs is equal to the number of clusters that the data is to be divided into.

→ A cluster's center position is specified by the weight vector connected to the corresponding output unit.

→ When an input x is presented to the network, the wt vector closest to x rotates towards it.

→ Consequently, weight vectors move towards those areas where the most inputs appear, and eventually the weight vectors become the cluster centers for the data set.

Similiarity Metric

→ The input vectors $x = [x_1, x_2, x_3]^T$ and the weight vector $w_j = [w_{j1}, w_2 ..]^T$ are assumed to be normalized to unit length.

→ The activation value a is calculated as:

$$a_j = \sum_{i=1}^{3} w_{ji} x_i = x^T w_j = w_j^T x$$

→ The o/p unit w/ the highest activation must be selected for further processing - which is what is implied by competitive.

→ Weights are updated according to the competitive & winner-take all rule.

$$w_v(t+1) = \frac{w_v(t) + \eta(x(t) - w_v(t))}{\| w_v(t) + \eta(x(t) - w_v(t)) \|}$$

← includes normalization

Dissimilarity Metric

→ use the Euclidean distance

$$a_j = \left(\sum_{i=1}^{3} (x_i - w_{ij})^2 \right)^{0.5} = \|x - w_j\|$$

→ update weights of the output unit with the smallest activation

$$w_k(t+1) = w_k(t) + \eta\left(x(t) - w_k(t)\right)$$

* Limitations of Competitive Learning?

→ wt. vectors are initialized to random values, and may be far from the input vector, and subsequently never get updated.

→ prevent by initializing wts. from input data samples

→ use leaky learning - update the wts of both the winning & losing units, but w/ a significantly smaller rate for the losers

→ A large initial value of learning rate explores the data space widely - later on, progressively smaller values refines the weights

$$\begin{cases} \eta(t) = \eta_0 e^{-\alpha t} & \alpha > 0 \\ \eta(t) = \eta_0 t^{-\alpha} & \alpha \leq 1 \\ \eta(t) = \eta(1-\alpha t) & 0 < \alpha < (\max\{t\})^{-1} \end{cases}$$

* Problems with the learning rate

→ If the learning rate is a constant, competitive learning does not guarantee stability in forming clusters.

→ However, if the learning rates decrease w/ time - they may become too small to update cluster centers when new data of a diff. probability is presented

→ called the stability - plasticity dilemma -

a learning agent should be plastic or adaptive in reactive to changing environments

should also be stable enough to preserve knowledge acquired previously

* Kohonen Self - Organizing Networks

Unit 2 Numericals

Percephon

① Find the wts required to perform the following classification using a percephon network.

The vectors [1, 1, 1, 1] and [-1, 1, -1, -1] belong to class 1

vectors [1, 1, 1, -1] and [1, -1, -1, 1] belong to class -1

learning rate = 1

initial wt = 0 & bias = 0

activation fn: $y = f(y_{in}) = \begin{cases} 1, & y_{in} > 0 \\ 0, & y_{in} = 0 \\ -1, & y_{in} < 0 \end{cases}$

Ans $\boxed{\text{Epoch 1}}$ $\alpha = 1$ $w = \alpha t x$

$w_{new} = old + \Delta w$

x_1	x_2	x_3	x_4	t	y_{in}	output	Δw_1	Δw_2	Δw_3	Δw_4	w_1	w_2	w_3	w_4
											0	0	0	0
1	1	1	1	1	0	0	1	1	1	1	1	1	1	1
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	0	2	0	0
1	1	1	-1	-1										

After row 1: ~~Δb~~ = $\Delta b = \alpha t = 1 \times 1 = 1$ $b_{new} = b_{old} + \Delta b = 1$

After row 2: $\Delta b = \alpha t = 1 \times 1 = 1$ $b_{new} = 2$

(2) Adaline network - OR-gate

x_1	x_2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

weights = 0.1

learning rate = 0.1

Least square error = ?

bias = 0.1

Epoch 1

(i) $y_{in} = b + \sum w_i x_i$

$= b + w_1 x_1 + w_2 x_2$

$= 0.1 + 0.1 + 0.1 = 0.3$

$t - y_{in} = 0.7$

$w_{inew} = w_{iold} + \Delta w_i$

$\Delta w_i = (t - y_{in}) \alpha x_i$

$w_1 = (0.7)(0.1)(1) = 0.17$

$w_2 = (0.7)(0.1)(1) = 0.17$

$b_{new} = b(old) + \Delta b$

$= b(old) + \alpha(t - y_{in})$

$= 0.1 + (0.1)(0.7)$

$= 0.17$

$E = (t - y_{in})^2 = 0.49$

find sum after all epochs

(3)

x_1	x_2	x_3	x_4	t
1	1	1	1	1
-1	1	-1	-1	1
1	1	1	-1	-1
1	-1	-1	1	-1

$\alpha = 1$

$\Delta w = \alpha t x$

$\Delta b = \alpha t$

$w_{new} = w_{old} + \Delta w$

$b_{new} = b_{old} + \Delta b$

initial weights $= 0$

Epoch 1

(i) $x_1 = 1 \quad w_1 = 0$

$x_2 = 1 \quad w_2 = 0$

$x_3 = 1 \quad w_3 = 0$

$x_4 = 1 \quad w_4 = 0$

$t = 1$

$\Sigma w_i x_i = 0$

y_{in} output $= 0 \neq t$

$\Delta w_1 = \alpha t x_1 = 1 \Rightarrow w_1 = 1$

$\Delta w_2 = \alpha t x_2 = 1 \Rightarrow w_2 = 1$

$\Delta w_3 = \alpha t x_3 = 1 \Rightarrow w_3 = 1$

$\Delta w_4 = \alpha t x_4 = 1 \Rightarrow w_4 = 1$

$\Delta b = \alpha t = (1)(1) = 1$

$b = b_{old} + \Delta b \quad \boxed{b = 1}$

(ii) $x_1 = -1 \quad w_1 = 1 \quad w x_1 = -1$

$x_2 = 1 \quad w_2 = 1 \quad w_2 x_2 = 1$

$x_3 = -1 \quad w_3 = 1 \quad w_3 x_3 = -1$

$x_4 = -1 \quad w_4 = 1 \quad w_4 x_4 = -1 \quad b = 1$

-1

$\boxed{y_{in} = -1} \neq t$

$\Delta w_1 = \alpha t x_1 = -1$ $\quad w_1 = 0$

$\Delta w_2 = \alpha t x_2 = 1 \Rightarrow w_2 = 2$

$\Delta w_3 = \alpha t x_3 = -1$ $\quad w_3 = 0$

$\Delta w_4 = \alpha t x_4 = -1$ $\quad w_4 = 0$

$\Delta b = \alpha t$

$= +1$

$b_{new} = b_{old} + \Delta b$

$= 1 + +1 = 2$ $\quad \boxed{b = 2}$

(iii) $x_1 = 1$ $\quad w_1 = 0$ $\quad w_1 x_1 = 0$

$x_2 = 1$ $\quad w_2 = 2$ $\quad w_2 x_2 = 2$

$x_3 = 1$ $\quad w_3 = 0$ $\quad w_3 x_3 = 0$

$x_4 = -1$ $\quad w_4 = 0$ $\quad w_4 x_4 = 0$ $\quad \boxed{b = 2}$

output $= 4$

$\boxed{y_{in} = 1}$ $\neq -1$

t
-1

$\Delta w_1 = \alpha t x_1 = -1$ $\quad w_1 = -1$

$\Delta w_2 = \alpha t x_2 = -1$ $\quad w_2 = 1$

$\Delta w_3 = \alpha t x_3 = -1$ $\quad w_3 = -1$

$\Delta w_4 = \alpha t x_4 = 1$ $\quad w_4 = 1$

$\Delta b = \alpha t = (1)(-1) = -1$

$b_{new} = -1 + 2 = 1$

(ii) $x_1 = 1$ $w_1 = -1$ $w_1 x_1 = -1$

$x_2 = -1$ $w_2 = 1$ $w_2 x_2 = -1$

$x_3 = -1$ $w_3 = -1$ $w_3 x_3 = 1$

$x_4 = 1$ $w_4 = 1$ $w_4 x_4 = 1$

$\boxed{b = 1}$

output = 1

$\boxed{y_{in} = 1}$

$\alpha = 1$ $t = -1$

$\Delta w_1 = -1$ $w_1 = -2$

$\Delta w_2 = 1$ $\Rightarrow$ $w_2 = 2$

$\Delta w_3 = 1$ $w_3 = 0$

$\Delta w_4 = -1$ $w_4 = 0$

$\Delta b = \alpha t$

$= (1)(-1) = \underline{\underline{-1}}$

$b = -1 + 1 = 0$

* Adaline for OR gate

x_1	x_2	t
1	+1	1
1	-1	1
-1	1	1
-1	-1	-1

weights = 0.1

learning rate = 0.1

lse = 2

bias = 0.1

$\boxed{\text{Epoch1}}$

(i) $x_1 = 1$ $w_1 = 0.1$

$x_2 = 1$ $w_2 = 0.1$

$t = 1$

$w_1 x_1 = 0.1$

$w_2 x_2 = 0.1$

$b = 0.1$

Total = $\underline{\underline{0.3}}$

$t - y_{in} = 1 - 0.3 = 0.7$

$\boxed{E_1 = (t - y_{in})^2 = 0.49}$

weight updation

$\Delta w_1 = \alpha (t - y_{in}) x_1 = 0.1 \times 0.7 \times 1 = 0.07$

$\Delta w_2 = \alpha (t - y_{in}) x_2 = 0.1 \times 0.7 \times 1 = 0.07$

new weights

$w_1 = 0.07 + 0.1 = 0.17$

$w_2 = 0.07 + 0.1 = 0.17$

$\boxed{\begin{aligned} w_1 &= 0.17 \\ w_2 &= 0.17 \end{aligned}}$

bias updation

$\Delta b = \alpha (t - y_{in}) = 0.1 \times 0.7$

$= 0.07$

$b_{new} = 0.07 + 0.1$

$\boxed{b = 0.17}$

(ii) $x_1 = 1$ $w_1 = 0.17$ $w_1 x_1 = 0.17$

$x_2 = -1$ $w_2 = 0.17$ $w_2 x_2 = -0.17$

$b = 0.17$

$y_{in} = 0.17$

$E = t - y_{in} = 0.83$

$\boxed{E_2 = (t - y_{in})^2 = 0.6889}$

weight updation

$\Delta w_1 = \alpha (t - y_{in}) x_1 = 0.1 \times 0.83 \times 1 = 0.083$

$\Delta w_2 = \alpha (t - y_{in}) x_2 = 0.1 \times 0.83 \times -1 = -0.083$

$$\boxed{\begin{aligned} w_1 &= \cancel{0.0114}\ 0.253 \\ w_2 &= +0.087 \end{aligned}}$$

<u>bias</u>

$\Delta b = \cancel{0.83}\ \alpha(t - y_{in})$

$= (0.1)(0.83)$

$= 0.083$

$b = 0.083 + 0.17 = 0.253$

$\boxed{b = 0.253}$

(iii) $x_1 = -1$ $\quad w_1 = 0.253 \quad = -0.253$

$x_2 = 1$ $\quad w_2 = 0.087 \quad = 0.087$

$t = 1$ $\quad b = 0.253$

$\boxed{y_{in} = 0.087}$

$t - y_{in} = 0.913$

$\boxed{E = (t - y_{in})^2 = 0.8335}$

$\Delta w_1 = \alpha(t - y_{in})x_1 = 0.1 \times 0.913 \times -1 = -0.0913$

$\Delta w_2 = \alpha(t - y_{in})x_2 = 0.1 \times 0.913 \times 1 = +0.0913$

$$\boxed{\begin{aligned} w_1 &= 0.1617 \\ w_2 &= 0.1783 \end{aligned}}$$

$\Delta b = \alpha(t - y_{in}) = 0.1 \times 0.913 = 0.0913$

$\boxed{b = 0.3443}$