Dikshita Kambri

118A2044

BE EXTC A2

EXPERIMENT 3

PERCEPTRON NETWORKS

EXPERIMENT 3

PERCEPTRON NETWORKS

AIM: Program for Perceptron network training for an AND function using Python. Assume bipolar inputs and target output of the network.

APPARATUS: PC with Python software.

THEORY: The architecture of perceptron network is shown in Fig. 1. It is a single layer network since it has only one layer of interconnections between the input and the output neurons. This network perceives the input signal received and performs the classification [2], [3].

The perceptron algorithm can be used for either binary or bipolar input vectors, having bipolar targets, threshold being fixed and variable bias.

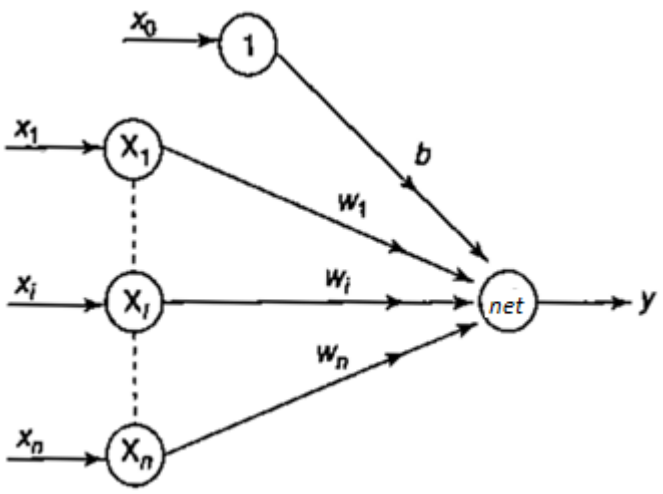

Fig. 1. Single Layer Perceptron Architecture [2]

The algorithm of a perceptron network [2] is as follows:

Step 0: Initialize weights and the bias (initially it can be set to zero). Also initialize the learning rate α (0 to 1). For simplicity α is set to 1.

Step 1: Perform Steps 2-6 until the final stopping condition is false.

Step 2: Perform Steps 3-5 for each training pair indicated by s:t, where s are the sensory inputs and t is the target output

Step 3: The input layer containing input units is applied with identity activation functions:

$$x_i = s_i$$

Step 4: Calculate the output of the network. To do so, first obtain the net input:

$$net = b + \sum_{i=1}^{n} (x_i w_i)$$

where "n" is the number of input neurons in the input layer. Then apply activations over the net input calculated to obtain the output:

$$y = f(net) = 1 \ \ if \ net \ > T$$

$$y = f(net) = 0 \ \ if \ -T \ \leq net \ \leq T$$

$$y = f(net) = -1 \ \ if \ net \ < T$$

Step 5: Weight and bias adjustment: Compare the value of the actual (calculated) output (y) and desired (target) output (t).

If $y \neq t$ 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 from Step 2.

This algorithm is not sensitive to the initial values of the weights or the value

OUTPUT:

1)AND Gate

```
[2]     for i in range (0,4):

            yin[i]=b+x1[i]*w1+x2[i]*w2
            if yin[i]>0:
                y[i]=1
            elif yin[i]==0:
                y[i]=0
            else:
                y[i]=-1
            if (y[i]!=t[i]):
                w1=w1+alpha*t[i]*x1[i]
                w2=w2+alpha*t[i]*x2[i]
                b=b+alpha*t[i]
        epoch=epoch+1

        if (np.array_equal(y,t)):
            print("MODEL HAS CONVERGED")
            c=1
```

```
MODEL HAS CONVERGED
```

```
[4] print("AND")
    print("w1 final",w1)
    print("w2 final",w2)
    print("b final",b)
    print(epoch)
```

```
AND
w1 final 1
w2 final 1
b final -1
2
```

2) OR Gate

```
[3]     for i in range (0,4):

            yin[i]=b+x1[i]*w1+x2[i]*w2
            if yin[i]>0:
                y[i]=1
            elif yin[i]==0:
                y[i]=0
            else:
                y[i]=-1
            if (y[i]!=t[i]):
                w1=w1+alpha*t[i]*x1[i]
                w2=w2+alpha*t[i]*x2[i]
                b=b+alpha*t[i]
        epoch=epoch+1

        if (np.array_equal(y,t)):
            print("MODEL HAS CONVERGED")
            c=1
```

```
MODEL HAS CONVERGED
```

```
[5] print("OR gate")
    print("w1 final",w1)
    print("w2 final",w2)
    print("b final",b)
    print(epoch)
```

```
OR gate
w1 final 2
w2 final 2
b final -1
5
```

3) AND NOT gate:

```
[3]  while(c==0):
       for i in range (0,4):
         yin[i]=b+x1[i]*w1+x2[i]*w2
         if yin[i]>0:
           y[i]=1
         elif yin[i]==0:
           y[i]=0
         else:
           y[i]=-1
         if (y[i]!=t[i]):
           w1=w1+alpha*t[i]*x1[i]
           w2=w2+alpha*t[i]*x2[i]
           b=b+alpha*t[i]
       epoch=epoch+1

       if (np.array_equal(y,t)):
         print("MODEL HAS CONVERGED")
         c=1
```

```
MODEL HAS CONVERGED
```

```
[4]  print("AND NOT gate")
     print("w1 final",w1)
     print("w2 final",w2)
     print("b final",b)
     print(epoch)
```

```
AND NOT gate
w1 final 0
w2 final -2
b final 0
2
```

CONCLUSION:

Performed the Perceptron network for AND, OR and AND NOT gate for bipolar input and target.

The output showed that AND gate required 2 epochs, OR gate required 5 epochs and AND NOT required 2 epochs for given order of input, threshold and learning rate.