K. J. Somaiya College of Engineering, Mumbai-77

Batch:A1 Roll No.: 16010120015

Experiment No. : 03

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Perceptron learning algorithm.

Objective: To write a program to implement Logic gate using perceptron network.

Expected Outcome of Experiment:

CO1: Identify and describe soft computing techniques and their roles

Books/ Journals/ Websites referred:

- J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
- Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
- S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI, 2003.
- http://library.thinkquest.org/C007395/tqweb/history.html

Pre Lab/ Prior Concepts:

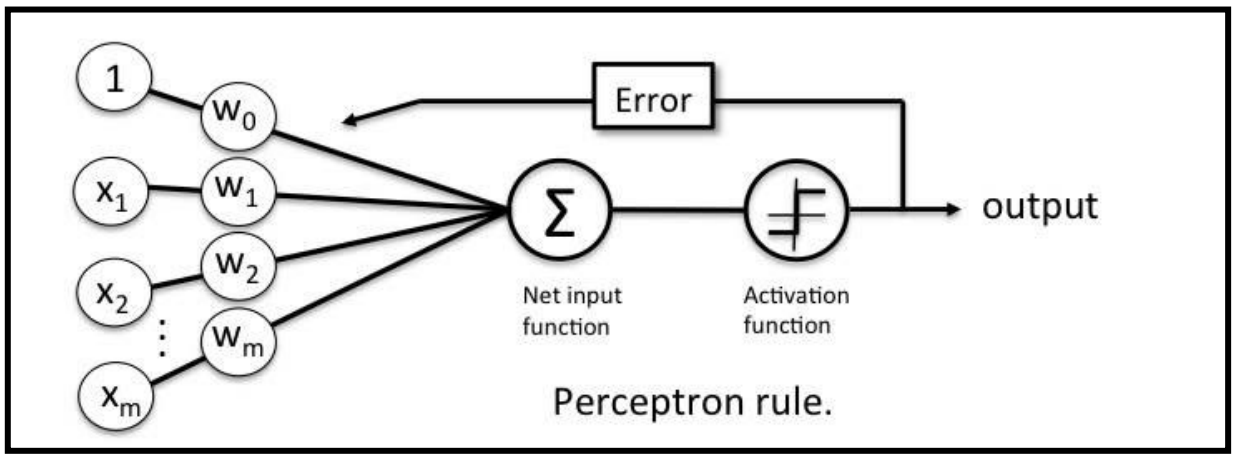
Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features-

1) A set of processing units;
2) An activation state for each unit, which is equivalent to the output of the unit;
3) Connections between the units. Generally, each connection is defined by a weight w_{jk} that determines the effect that the signal of unit *j* has on unit *k;*
4) A propagation rule, which determines the effective input of the unit from its external inputs;
5) An activation function, which determines the new level of activation based on the effective input and the current activation;
6) An external input (bias, offset) for each unit;
7) A method for information gathering (learning rule);
8) An environment within which the system can operate, provide input signals and, if necessary, error signals.

Implementation Details:

Perceptron learning concept:

Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients. The input features are then multiplied with these weights to determine if a neuron fires or not.



The Perceptron receives multiple input signals, and if the sum of the input signals exceeds a certain threshold, it either outputs a signal or does not return an output. In the context of supervised learning and classification, this can then be used to predict the class of a sample.

AND Function:

```
import numpy as np
import TableIt

def perceptron(x,w,t,b):
    output = []
    i = 0
    while i < 4:
        v = np.dot(w,x[i])
        print("v: {}",v)
        v = v + b
        print("v+b: {}",v)
        if v >= t:
            output.append(1)
        else:
            output.append(0)
        i += 1
    return output

print("*Options*")
print("1)AND Gate")
print("2)OR Gate")
print("3)NAND Gate")
print("4)NOR Gate")
choice = int(input("Enter the index of logic gate that you want to check for: "))
print(choice)

w=[]
w1 = float(input("Enter weight 1 value: "))
w.append(w1)
w2 = float(input("Enter weight 2 value: "))
w.append(w2)
t = float(input("Enter threshold value: "))
b = float(input("Enter bias value: "))

x = [[0,0],[0,1],[1,0],[1,1]]
result = perceptron(x,w,t,b)

orResult = [0,1,1,1]
andResult = [0,0,0,1]
norResult = [1,0,0,0]
```

```
nandResult = [1,1,1,0]

if choice==1:
    if result==andResult:
        print("This configuration behaves like AND Gate!")
    else:
        print("This configuration doesnt behave like AND Gate!")

if choice==2:
    if result==orResult:
        print("This configuration behaves like OR Gate!")
    else:
        print("This configuration doesnt behave like OR Gate!")

if choice==3:
    if result==nandResult:
        print("This configuration behaves like NAND Gate!")
    else:
        print("This configuration doesnt behave like NAND Gate!")

if choice==4:
    if result==norResult:
        print("This configuration behaves like NOR Gate!")
    else:
        print("This configuration doesnt behave like NOR Gate!")
table = [
    ["x1","x2","result"],
    ["0","0",result[0]],
    ["0","1",result[1]],
    ["1","0",result[2]],
    ["1","1",result[3]],
]
TableIt.printTable(table, useFieldNames=True)
```

OUTPUT:

```
*Options*
1)AND Gate
2)OR Gate
3)NAND Gate
4)NOR Gate
>>Weights: [0.3, 0.5]
>>Threshold: 0.15
>>Bias: 0.1
This configuration doesnt behave like AND Gate!
+-------------------------+
| x1     | x2     | result |
+--------+--------+--------+
| 0      | 0      | 0      |
| 0      | 1      | 1      |
| 1      | 0      | 1      |
| 1      | 1      | 1      |
+-------------------------+
```

Algorithm:

- For the first input pattern, x1=0, x2=0 and t=0.1, with weights and bias, w1=0.3, w2=0.5 and b=0.15
- Calculate the net input:
  yin = b +x1w1 +x2w2 =0.1+0*0.3+0*0.5 =0.1
- The output y is computed by applying activations over the net input calculated:
  y= f(yin) = {1 $if\ yin$ >= threshold and 0 $if\ yin$ < threshold}
- Here, we have taken θ = 0.15 Hence when yin = 0.1, y = 0.
- The same procedure is repeated for all the input patterns

Conclusion: the application of neural networks in various fields gives us a better understanding of how things are organized and the way they function. Thus, we have successfully implemented perceptron training algorithm for AND function.

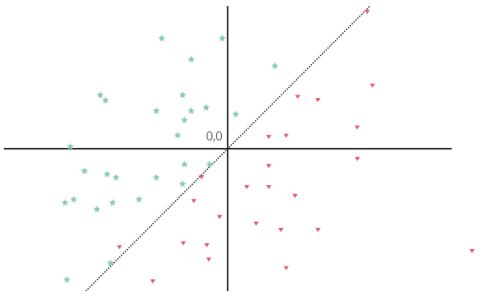
Post Lab Descriptive Questions :

1. How is linear separability implemented using perceptron networking training

Linear separability is the concept wherein the separation of input space into regions is based on whether the network response is positive or negative.

A decision line is drawn to separate positive and negative responses. The decision line may also be called as the decision-making Line or decision-support Line or linear separable line. The necessity of the linear separability concept was felt to clarify classify the patterns based upon their output responses.

Generally, the net input calculated to the output unit is given as -



yin=b+∑ni=1(xiwi)yin=b+∑i=1n(xiwi)

A perceptron is a single neuron model that may be a forerunner to a large network.

Perceptrons perform the computations to output binary values 0 or 1 (the output values can also be -1 and 1 depending on the activation function used).

ingle perceptron consists of inputs, weights, bias, and an activation function to serve the output.

Let us assume the output of the perceptron to be y, and let x be an input to the perceptron with a single input.

Then y = w1*x + b, where **w1** is the weight for the input, and **b** is the bias.

Let us further simplify this problem by assuming w1 to be 1, and b to be 0.

This makes our equation to be y = x, which is the equation of the straight line passing through the origin.

2. Mention the application of perceptron network.

Data Encryption - Data Security / Data Loss Protection

Data encryption is a variation of data compression. The difference is that while data compression is designed to retain the original shape of data, encryption is doing the opposite - it conceals the content of data and makes it incomprehensible in the encoded form.

Data Analytics for Business

Presenting data in an accessible form is as important as understanding the insights behind it. Because of that, data visualization is one of the most viable tools in depicting the state of things and explaining complex data in simple terms.

Image Recognition, Object detection, Route Adjustment

Drones of all forms are slowly, but surely establishing themselves are viable multi-purpose tools. After all, if you can train a robotic assembly line to construct cars with laser-focused precision - why can't try to teach artificial intelligence to drive it.

Customer Ranking - User Profiling - CRM

Customer engagement is a high priority for any company that is interested in a continuous and consistent relationship with their customers. The key is in the value proposition design that is relevant to the target segments and appropriate calls to action that motivate customers to proceed.

.

Date: _____10/10/2022__ **Signature of faculty in-charge**