Spring_22_NNDL_Lab_1

January 10, 2022

1 LAB 1 : Realisation of Logic Gates and Linear Regression Equation

Name :
Roll Number :
Referrence Material : Page no. 26-29, Artificial Neural Network by B. Yegnanarayana

2 Problem 1 : Demonstrate the realization of NAND gate, NOR gate, and XOR gate using McCulloch Pitts (MP) and Rosenblatt's Perceptron model.

1. Write down the truth table of the logic gates.
2. For MP model: compute the appropriate weight, such that the neuron provide logic gate output.
3. For Perceptron: use truth table values as input and output to learn the weights through weight update equation.

2.1 Write down the Objectives, Hypothesis and Experimental description for the above problem

=== Write your answer here ===

2.2 Programming :

Please write a program to demonstrate the same

[6]:
```
## Part A : MP model

import numpy as np
import matplotlib.pyplot as plt


def mp_model(x,gate):

  ## Write the activation function and define weights and bias for all three␣
 ↪gates

  if gate == "NAND":
```

```
    ## Write your code here

  if gate == "NOR":

    ## Write your code here

  if gate == "XOR":

    ## Write your code here


  return out


inp_list = ## Create a list consisting of inputs to logic gates (Binary, i.e␣
 ↪(0,0),(0,1),(1,0),(1,1))
req_gates = ## Create a list of all gates (NAND,NOR,XOR)

for gate in req_gates:
  print('Results for ' + gate + ' gate : ')
  for inp in inp_list:
    print("Input is : " + str(inp))
    out = mp_model(inp,gate)
    print("Logic Gate output is : ",out)

  print('====================================')
```

```
Results for NAND gate :
Input is : (0, 0)
Logic Gate output is :  1
Input is : (0, 1)
Logic Gate output is :  1
Input is : (1, 0)
Logic Gate output is :  1
Input is : (1, 1)
Logic Gate output is :  0
====================================
Results for NOR gate :
Input is : (0, 0)
Logic Gate output is :  1
Input is : (0, 1)
Logic Gate output is :  0
Input is : (1, 0)
Logic Gate output is :  0
Input is : (1, 1)
Logic Gate output is :  0
```

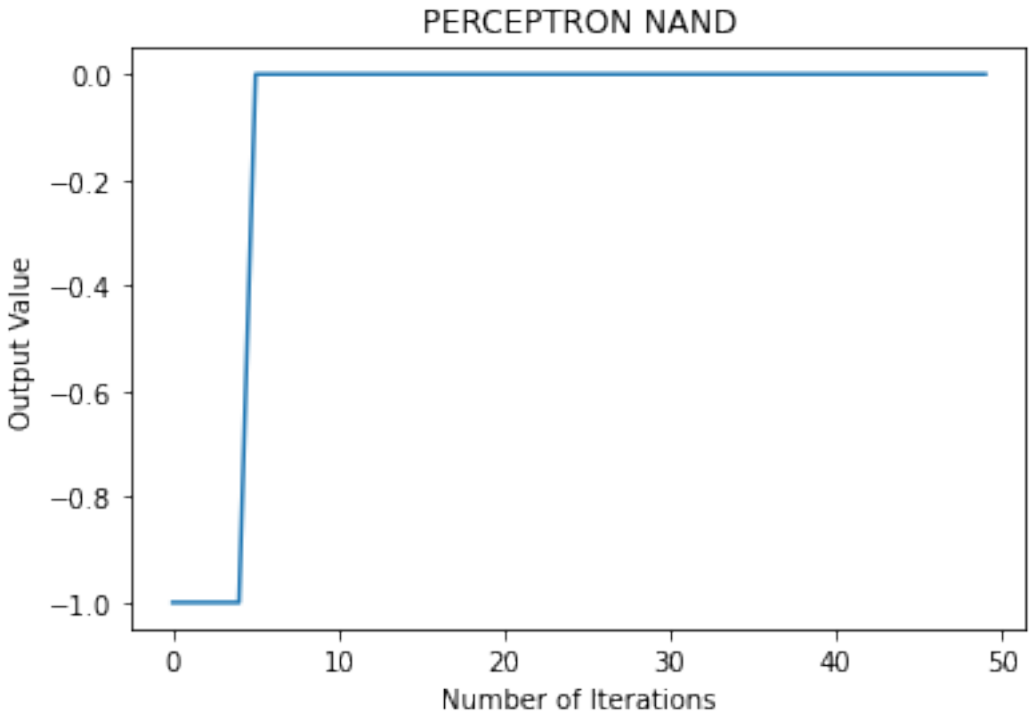
```
====================================
Results for XOR gate :
Input is : (0, 0)
Logic Gate output is :  0
Input is : (0, 1)
Logic Gate output is :  1
Input is : (1, 0)
Logic Gate output is :  1
Input is : (1, 1)
Logic Gate output is :  0
====================================
```

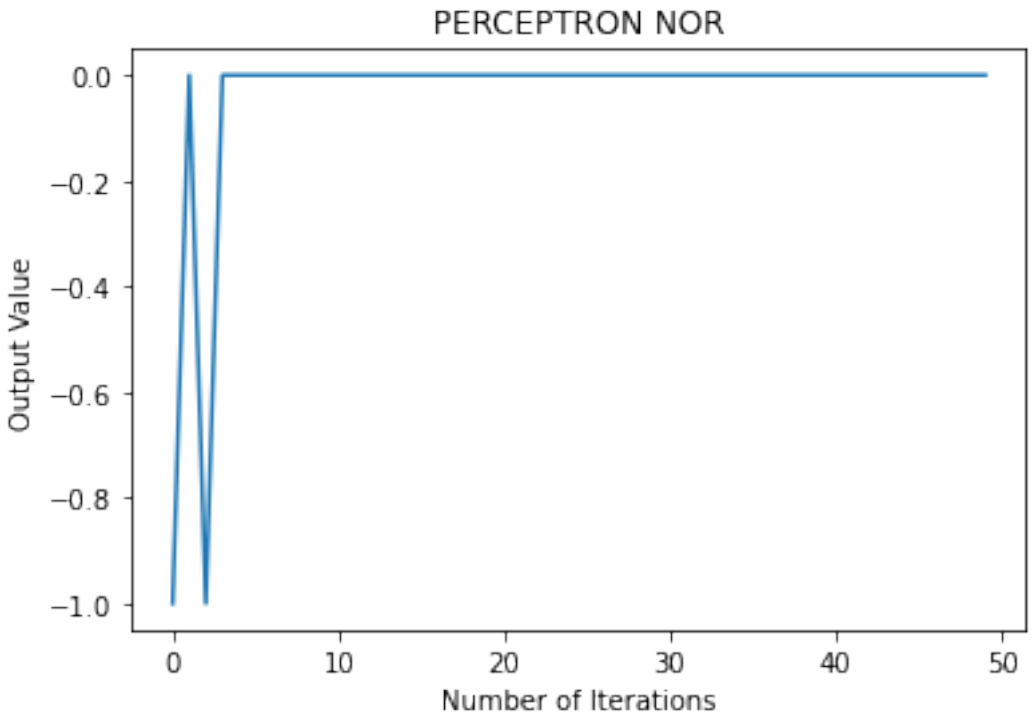
[17]:
```
## Part B : Rosenblatt's Percept

import numpy as np
import matplotlib.pyplot as plt

def␣
 ↪rosenblatt_perceptron(inp_list,output,learning_rate,w_initial,num_epochs,gate):
 ↪

  error = []

  for epoch in range(num_epochs):

    ## Write the learning code here, return new weights and error, Save the␣
 ↪error value after each iteration to plot a error v/s iteration graph in case␣
 ↪of all 3 gates

  return w_new,error

inp_list = ## Create a list consisting of inputs to logic gates (Binary, i.e␣
 ↪(0,0),(0,1),(1,0),(1,1))
nand_output = ## Define NAND gate target output
nor_output = ## Define NOR gate target output
xor_output = ## Define XOR gate target output

learning_rate = ## Define a learning rate
w_initial = ## Initialise weights
num_epochs = ## Set number of epochs

## For NAND gate
w_new,error =␣
 ↪rosenblatt_perceptron(inp_list,nand_output,learning_rate,w_initial,num_epochs,"NAND")

## For NOR gate
```

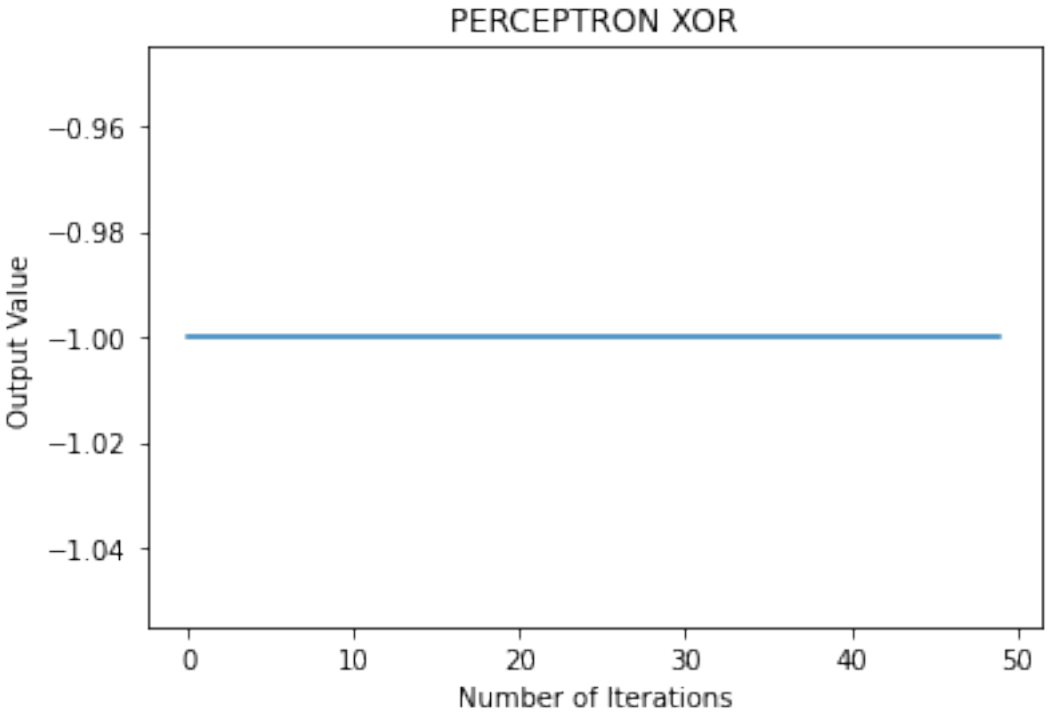
```
w_new,error =␣
 ↪rosenblatt_perceptron(inp_list,nor_output,learning_rate,w_initial,num_epochs,"NOR")

## For XOR gate
w_new,error =␣
 ↪rosenblatt_perceptron(inp_list,xor_output,learning_rate,w_initial,num_epochs,"XOR")
```

```
Weights for NAND :
[ 1.00000000e-01 -2.77555756e-17 -1.00000000e-01]
Weights for NOR :
[ 0.1 -0.1 -0.1]
Weights for XOR :
[ 1.00000000e-01 -1.00000000e-01  2.77555756e-17]
```

[17]: `Text(0, 0.5, 'Output Value')`




PERCEPTRON NOR
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
Output Value
0
10
20
30
40
50
Number of Iterations


PERCEPTRON XOR
−0.96
−0.98
−1.00
−1.02
−1.04
Output Value
0
10
20
30
40
50
Number of Iterations

2.3 Inferences and Conclusion : State all the key observations and conclusion

=== Write your answer here ===

3 Problem 2 : Demonstrate the realization of y = 1 + 2x1 + 2x2 using Widrow's Adaline model.

1. Generate some N no. of data points using equation y = 1 + 2x1 + 2x2.
2. Use the input and output data to train the Adaline model, after training the Adaline model should provide appropriate y as output value for any arbitrary input.

3.1 Write down the Objectives, Hypothesis and Experimental description for the above problem

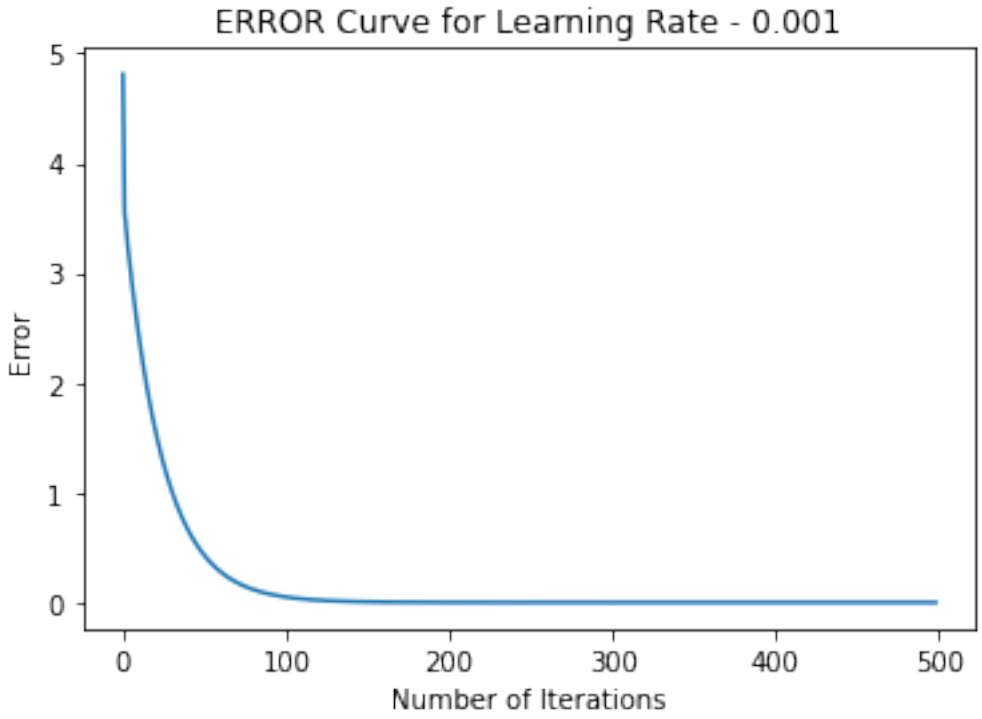
=== Write your answer here ===

3.2 Programming :

Please write a program to demonstrate the same

[20]:
```
## Widrow Adaline model : Perform the experiment for different weight␣
↪inialisations and learning rates and state your observations

def widrow_adaline(inp,learning_rate,w_initial,num_epochs,N):

  error = []

  for epoch in range(num_epochs):

    ## Write the learning(training) loop here and return the new weights, also␣
↪save the error after each iteration


  return w_new,error


N = ## Set the number of input points
learning_rate = ## Define a learning rate
w_initial = ## Initialise weights
num_epochs = ## Set number of epochs

## Generate the input below

inp = ## An input array of the shape (3,100)

w_new,error = widrow_adaline(inp,learning_rate,w_initial,num_epochs)
```

```
Final Weights for Learning rate - 0.001 are :-
[[1.214284 ]
 [2.1428595]
 [1.9285718]]
```



3.3 Inferences and Conclusion : State all the key observations and conclusion

=== Write your answer here ===