Machine Learning End Lab Exam Question 1

Name: Sri Sai Vijaya Aditya Nittala

Roll No.: 177163

Section: A

Implement given boolean funtion using suitable Neural Network:

> Y = (A + AB)(B + BC)(C + AB)

This boolean function can be simplified as follows:

> Y = (A + AB)(B + BC)(C + AB) = (A)(B)(C + AB) = ABC + (AB)(AB) = ABC + AB = AB(C + 1) = AB

Now, this boolean function is nothing but an AND function

```
## IMPORTING NECESSARY LIBRARIES
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score ## USED TO FIND ACCURACY ONLY


## TRUTH TABLE FOR 3 BOOLEAN VARIABLES
X = [0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1]
X = np.reshape(X, (8, 3))

## -1 INDICATES 0 AND 1 INDICATES 1
y = [-1, -1, -1, -1, -1, -1, 1, 1]
y = np.reshape(y, (8, 1))

print("Truth table for the given boolean function : ")
print("A\tB\tC\tY")
print("-----"*5)
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        print("{}\t".format(X[i, j]), end = '')
    print("{}".format(0 if y[i, 0] == -1 else 1))
```

```
Truth table for the given boolean function :
A       B       C       Y
-------------------------
0       0       0       0
0       0       1       0
0       1       0       0
0       1       1       0
1       0       0       0
```

```
     1       0       1       0
     1       1       0       1
     1       1       1       1
```

```
## USED FOR PREDICTIONS
def threshold(Z):
    for i in range(Z.shape[0]):
        if Z[i][0] > 0:
            Z[i][0] = 1
        else:
            Z[i][0] = -1
    return Z


''' Back propagation implemetation '''

def backpropagation(w, b, X, Z, y, a):
    w = w - a*np.dot((Z - y).T, X)/Z.shape[0]
    b = b - a*np.sum(Z - y)/Z.shape[0]
    return w, b


'''Forward propagation implementation'''

def train(X, y, w, b, a, iters):
    for i in range(iters):
        Z = np.dot(X, w.T) + b
        Z = threshold(Z)
        w, b = backpropagation(w, b, X, Z, y, a)
    return w, b


## FINDS ACCURACY OF THE MODEL
def find_accuracy(X, Y, w, b):
    Z = np.dot(X, w.T) + b
    Z = threshold(Z)
    return accuracy_score(Y, Z)


w = np.zeros(3)                 # Weights of the model
w = np.expand_dims(w, axis=0)
b = -1                          # bias for the neuron
a = 0.1                         # learning rate

print("Training model for given boolean function : ")
w, b = train(X, y, w, b, a, 1000)

print("Parameters obtained after training : ")
print("Bias : {}".format(b))
print("Weights : \n{}\n".format(w))
```

```
     Training model for given boolean function :
```

```
Parameters obtained after training :
Bias : -0.6499999999999997
Weights :
[[0.35 0.35 0.15]]
```

```
accuracy = find_accuracy(X, y, w, b)
print("Accuracy of model : {}%".format(accuracy*100))
```

```
Accuracy of model : 100.0%
```