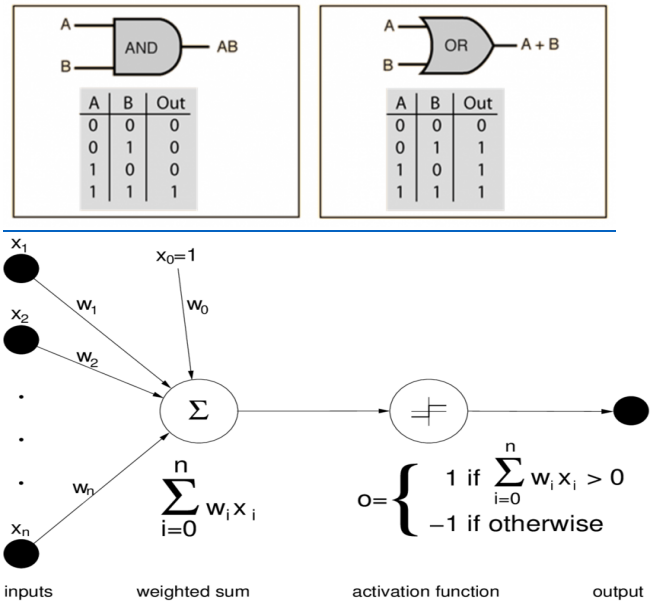
Aim : Train a Perceptron to Learn the AND and OR Gate

Theory :

Perceptron is capable of learning decision boundaries in a binary classification scenario. For example, if your inputs are 1-Dimensional or 2-Dimensional, we can actually visualize different parts of our perceptron. Below, in the 1-Dimensional case you can see how the input and the bias unit are linearly combined using the weights, resulting in Z. Then you can see that we apply the thresholding function on top of Z to create our binary classifier. This classifier outputs +1 for instances of one class, and -1 for the instances of the other class.



Code :

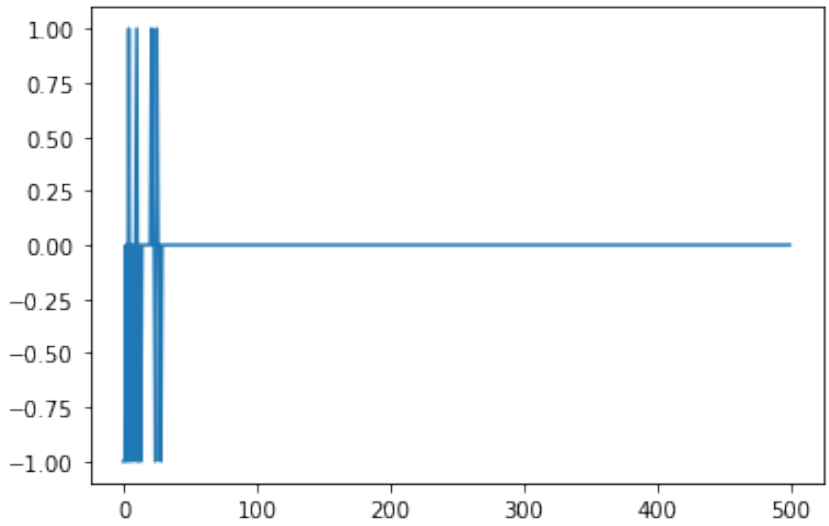
And Gate :

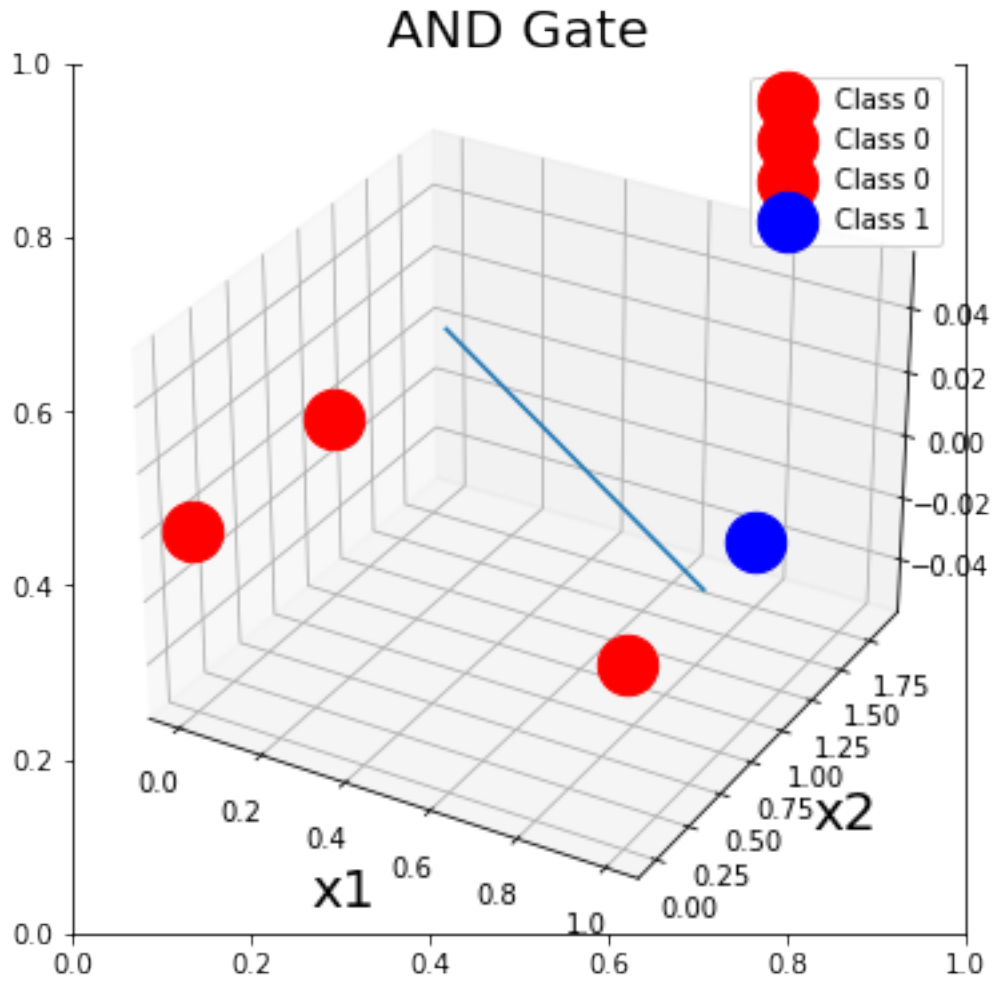
```
import numpy as np
from numpy import random
import matplotlib.pyplot as plt

eta=0.5
b=-0.3
n=200

def unit_step(z):
    if z > 1:
        return 1
    else:
        return 0

def perceptron(x,w):
    v = np.dot(w,x)+b
    y_o = unit_step(v)
```

```
    return y_o

x = np.array([[0,0],[0,1],[1,0],[1,1]])
w = random.rand(2)
y = np.array([0,0,0,1])

for p in range(n):
    for i in range(4):
        y_o = perceptron(x[i],w)
        error = y[i]-y_o
        if(error != 0):
            for j in range(len(w)):
                w[j] += eta*error*x[i][j]

for i in range(4):
    print("{} -> {}".format(x[i],perceptron(x[i],w)))
print("Weights obtained= {}".format(w))

y_x=[]
for i in range(4):
    y_x.append(-w[0]/w[1]*x[i][0]-b/w[1])


fig = plt.figure(figsize=(6, 6))
area=500
plt.title('AND Gate', fontsize=20)
ax = fig.add_subplot(111,projection='3d')
ax.scatter(0, 0, s=area, c='r', label="Class 0")
ax.scatter(0, 1, s=area, c='r', label="Class 0")
ax.scatter(1, 0, s=area, c='r', label="Class 0")
ax.scatter(1, 1, s=area, c='b', label="Class 1")
ax.plot(x[:,0], y_x)
plt.xlabel('x1', fontsize=20)
plt.ylabel('x2', fontsize=20)
plt.grid()
plt.legend()
plt.show()
```

Output : And with bias -5

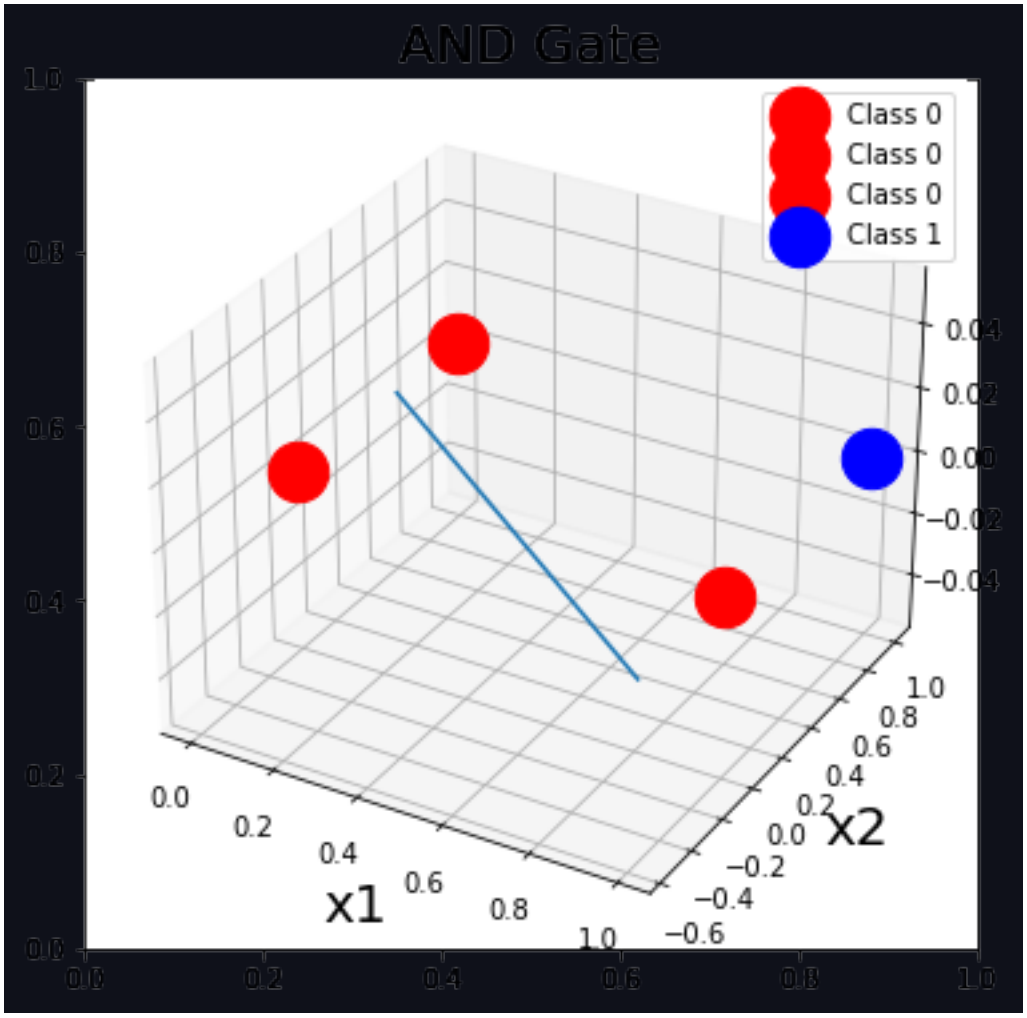
```
[0 0] -> 0
[0 1] -> 0
[1 0] -> 0
[1 1] -> 1
Weights obtained= [1.36891538 0.98142447]
```





And with bias -0.3

```
[0 0] -> 0
[0 1] -> 0
[1 0] -> 0
[1 1] -> 1
Weights obtained= [0.84705938 0.88287752]
```

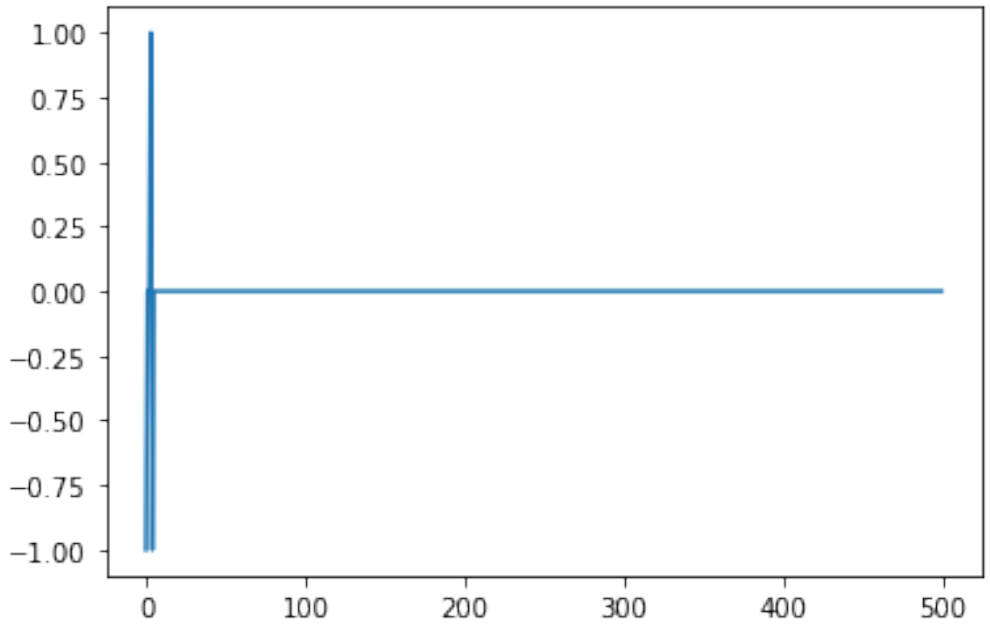


OR Gate :

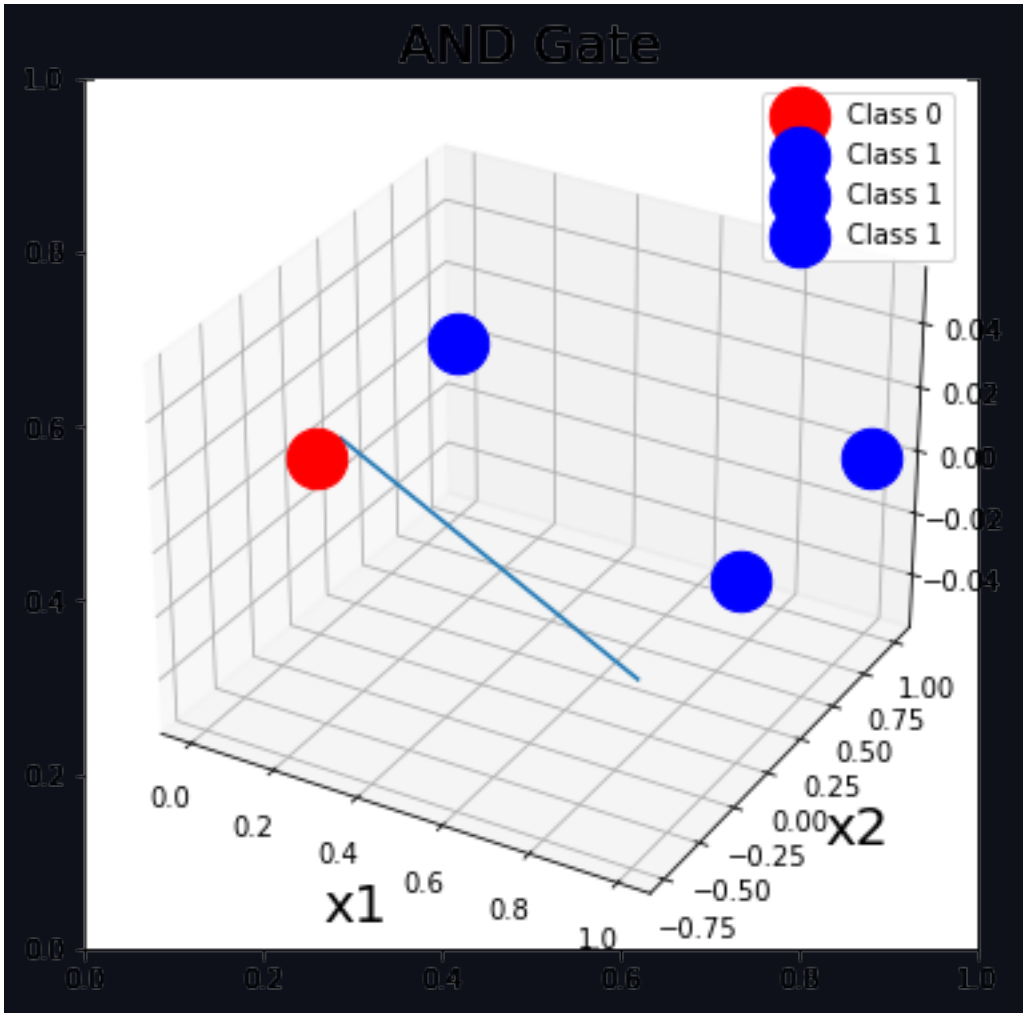
```
import numpy as np
from numpy import random
import matplotlib.pyplot as plt

eta=0.5
b=-0.3
n=200

def unit_step(z):
    if z > 1:
        return 1
    else:
        return 0

def perceptron(x,w):
    v = np.dot(w,x)+b
    y_o = unit_step(v)
    return y_o
```

```
x = np.array([[0,0],[0,1],[1,0],[1,1]])
w = random.rand(2)
y = np.array([0,1,1,1])

for p in range(n):
    for i in range(4):
        y_o = perceptron(x[i],w)
        error = y[i]-y_o
        if(error != 0):
            for j in range(len(w)):
                w[j] += eta*error*x[i][j]

for i in range(4):
    print("{} -> {}".format(x[i],perceptron(x[i],w)))
print("Weights obtained= {}".format(w))

y_x=[]
for i in range(4):
    y_x.append(-w[0]/w[1]*x[i][0]-b/w[1])


fig = plt.figure(figsize=(6, 6))
area=500
plt.title('AND Gate', fontsize=20)
ax = fig.add_subplot(111,projection='3d')
ax.scatter(0, 0, s=area, c='r', label="Class 0")
ax.scatter(0, 1, s=area, c='b', label="Class 1")
ax.scatter(1, 0, s=area, c='b', label="Class 1")
ax.scatter(1, 1, s=area, c='b', label="Class 1")
ax.plot(x[:,0], y_x)
plt.xlabel('x1', fontsize=20)
plt.ylabel('x2', fontsize=20)
plt.grid()
plt.legend()
plt.show()
```

OR with bias -5

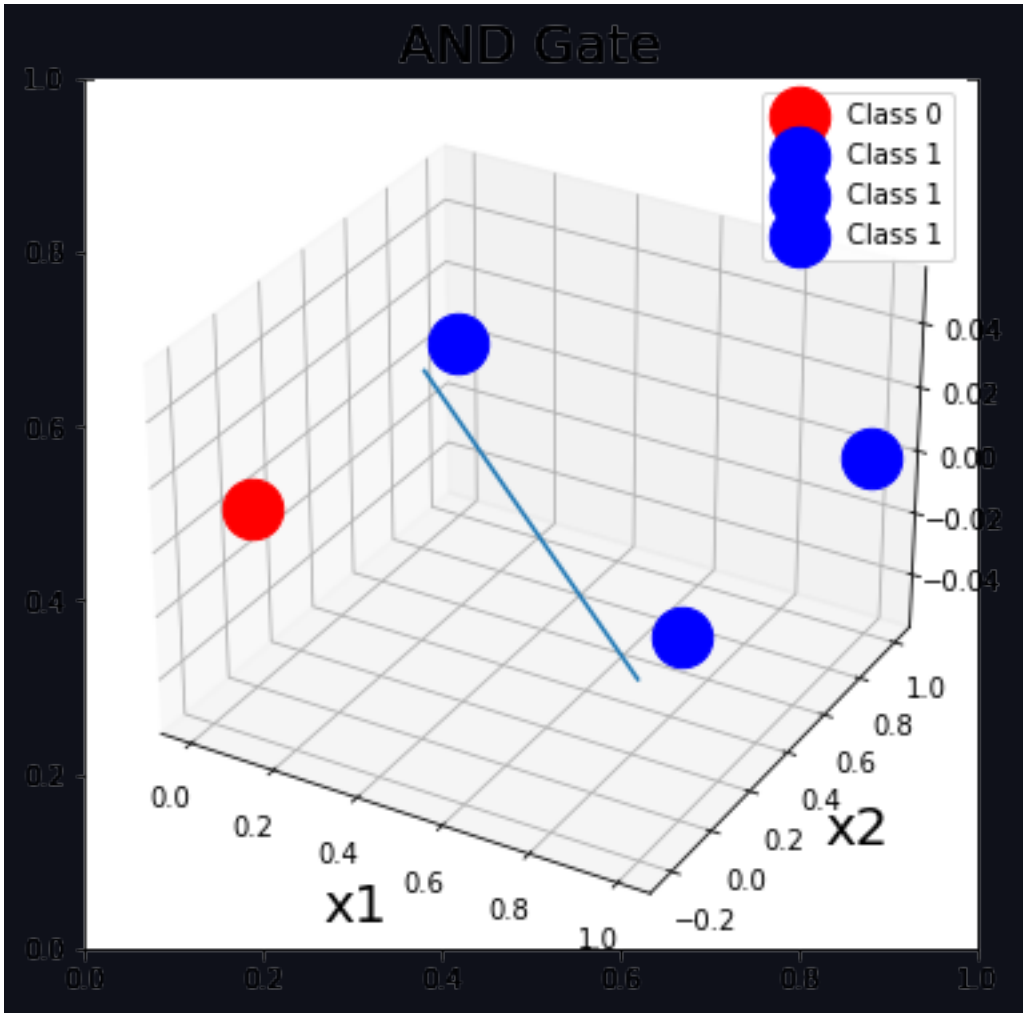



```
[0 0] -> 0
[0 1] -> 1
[1 0] -> 1
[1 1] -> 1
Weights obtained= [6.30753949 6.03873356]
```



OR with bias -0.3

```
[0 0] -> 0
[0 1] -> 1
[1 0] -> 1
[1 1] -> 1
Weights obtained= [1.5216746 1.59220334]
```



DISCUSSION :

1. AND Gate: For the AND gate, with a bias of -0.3 and w = [0.84705938 0.88287752] the regression line doesn't separate the values properly. Setting a bias = -5 and w = [1.36891538 0.98142447] the regression line now properly separates the two divisions.
2. OR Gate: For or gate with the initial bias of -0.3 and w = [1.5216746 1.59220334], the regression line obtained is shown in first OR output. With the bias=-5 and w = [6.30753949 6.03873356] the regression line moved up, showing that there is more than one solution to the problem.

CONCLUSION:

In this experiment, AND and OR gates perceptron models were successfully run. 3D Regression line were plotted to verify that the classification between the two classes representing -1 and +1.

By :

Heramba Panda

118EI0372