| Name: | Tufan Kundu |
|---|---|
| Registration no: | 24MDT0184 |
| Course Name: | Deep Learning Lab |
| Course Code: | PMDS603P |
| Experiment: | 1 |
| Date: | 17 July,2025 |

## Q1. Using McCulloch Pitts model discussed in the class, write a Python code to implement the OR and AND Boolean functions. Also, plot the boundary input points and the linear classifier that we get in that case.

### OR GATE

In [1]:

```
x1 = [0,0,1,1]
x2 = [0,1,0,1]
def or_func(x1,x2):
    x_or = []
    for i in range(len(x1)):
        if x1[i] == 0 and x2[i] == 0:
            x_or.append(0)
        else:
            x_or.append(1)
    return x_or
x_or = or_func(x1,x2)
g_or = [x1[i]+x2[i] for i in range(len(x1))]
```

In [2]:

```
y_or = []
for i in range(len(x1)):
    if g_or[i]>=1:
        y_or.append(1)
    else:
        y_or.append(0)
```

In [3]:

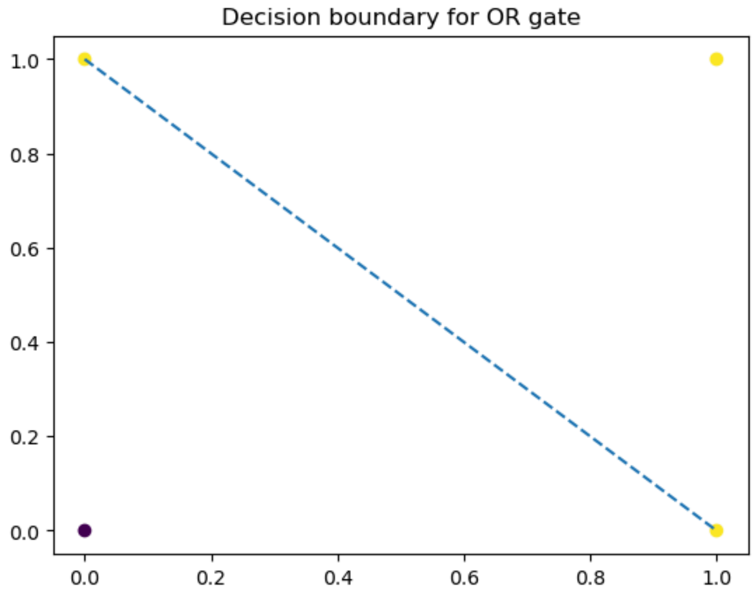
```
y_or
```

Out[3]:

```
[0, 1, 1, 1]
```

In [4]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.scatter(x1,x2, c = y_or)
x = np.linspace(0, 1, 100)
y = 1 - x
plt.plot(x, y, label='x + y = 1', linestyle = '--')
plt.title("Decision boundary for OR gate")
plt.show()
```



### AND GATE

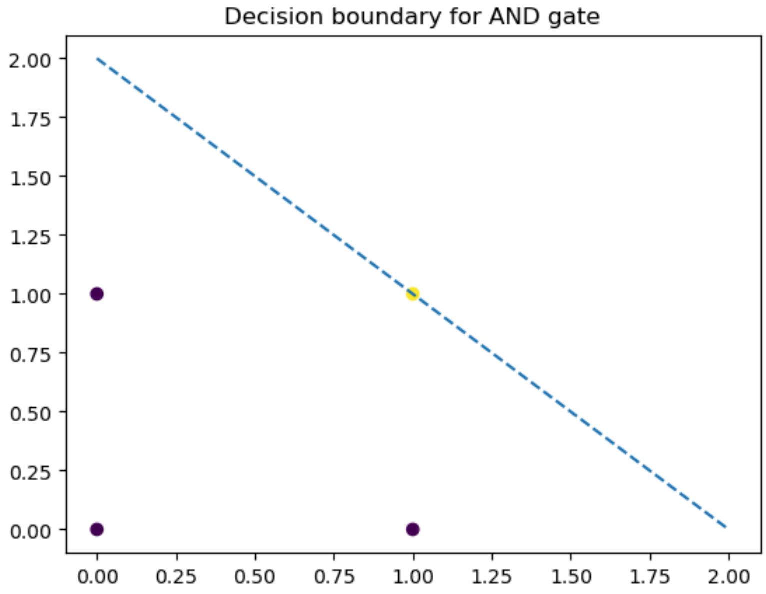
In [5]:
```
x1 = [0,0,1,1]
x2 = [0,1,0,1]
def and_func(x1,x2):
    x_and = []
    for i in range(len(x1)):
        if x1[i] == 1 and x2[i] == 1:
            x_and.append(1)
        else:
            x_and.append(0)
    return x_and
x_and = and_func(x1,x2)
g_and = [x1[i]+x2[i] for i in range(len(x1))]
```

In [6]:
```
y_and = []
for i in range(len(x1)):
    if g_and[i]>=2:
        y_and.append(1)
    else:
        y_and.append(0)
```

In [7]:
```
y_and
```

Out[7]:
```
[0, 0, 0, 1]
```

In [8]:
```
plt.scatter(x1,x2, c = y_and)
x = np.linspace(0, 2, 100)
y = 2 - x
plt.plot(x, y, label='x + y = 2', linestyle = '--')
plt.title("Decision boundary for AND gate")
plt.show()
```



# Q2. Write a Python code to implement the Perceptron Learning Algorithm to implement OR, AND, NAND, NOR logic Gates and report the weights and bias. Also, print the inputs and outputs after training the weights properly. Further plot the linear classifier that you have obtained as well. Note that here the input vectors x should be extended with a x0 term, which is taken as 1. Assume we have two inputs x1 and x2 so the x vector would be [1, x1, x2]

In [9]:
```
x1_input = np.array([0,0,1,1])
x2_input = np.array([0,1,0,1])
x_bias = np.ones(shape=(x1_input.shape), dtype = 'int')
```

In [10]:
```
x = np.concatenate([[x_bias],[x1_input],[x2_input]])
```

In [11]:
```
x = x.T
```

In [21]:
```
w = np.random.rand(x.shape[1])
```

In [22]:
```
x
```

```
Out[22]: array([[1, 0, 0],
                [1, 0, 1],
                [1, 1, 0],
                [1, 1, 1]])
```

```
In [23]: w
```

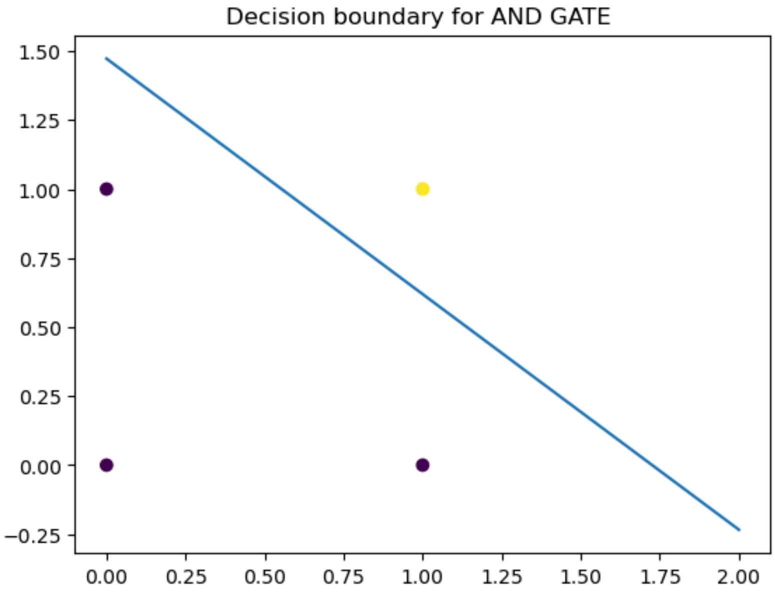
```
Out[23]: array([0.29940949, 0.40591265, 0.47580566])
```

```
In [24]: x_or = np.array([0,1,1,1])
         x_and = np.array([0,0,0,1])
         x_nand = np.array([1,1,1,0])
         x_nor = np.array([1,0,0,0])
```

```
In [25]: def perceptron(x,y,w):
             for i in range(100):
                 n = np.random.randint(0,4)
                 y_pred = np.dot(w,x[n])
                 if y[n]==1 and y_pred<0:
                     w = w+x[n]
                 elif y[n] == 0 and y_pred>=0:
                     w = w-x[n]
             return w
```

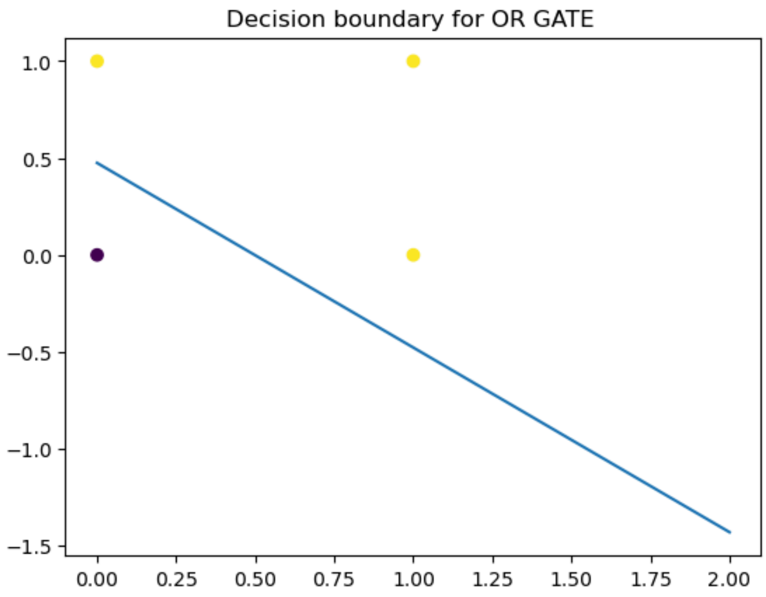
```
In [32]: w_updated_and = perceptron(x,x_and,w)
```

```
In [33]: plt.scatter(x.T[1],x.T[2], c= x_and)
         x1 = np.linspace(0,2)
         x2 = (-w_updated_and[0]- w_updated_and[1]*x1)/w_updated_and[2]
         plt.plot(x1,x2)
         plt.title("Decision boundary for AND GATE ")
         plt.show()
```



```
In [34]: w_updated_or = perceptron(x,x_or,w)
```

```
In [36]: plt.scatter(x.T[1],x.T[2], c= x_or)
         x1 = np.linspace(0,2)
         x2 = (-w_updated_or[0]- w_updated_or[1]*x1)/w_updated_or[2]
         plt.plot(x1,x2)
         plt.title("Decision boundary for OR GATE ")
         plt.show()
```

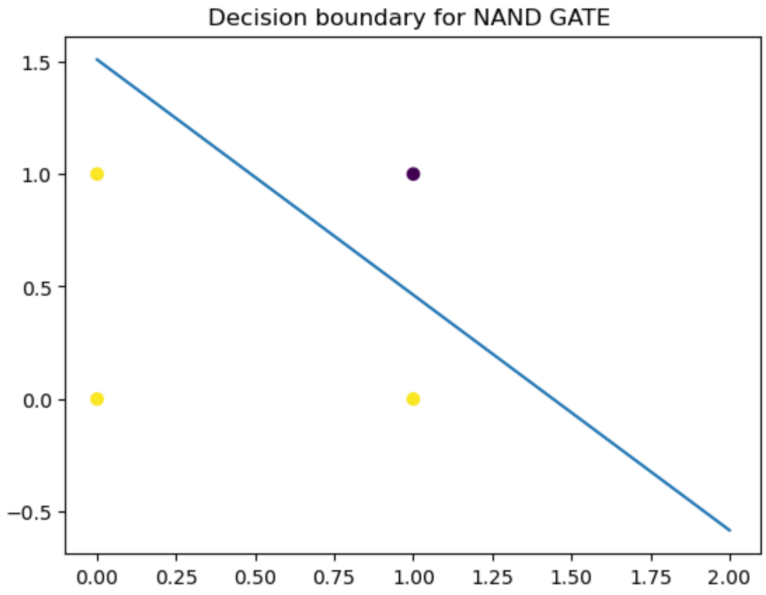


In [37]:

```
w_updated_nand = perceptron(x,x_nand,w)
```

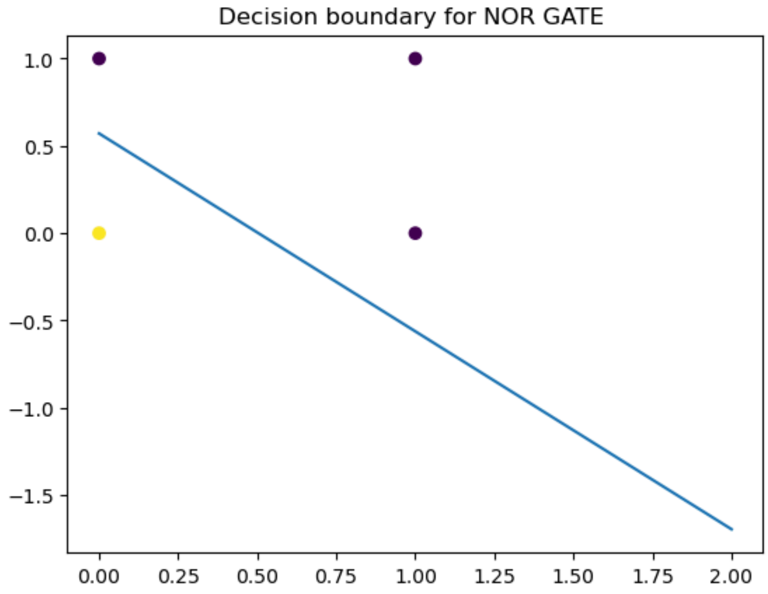
In [38]:

```
plt.scatter(x.T[1],x.T[2], c= x_nand)
x1 = np.linspace(0,2)
x2 = (-w_updated_nand[0]- w_updated_nand[1]*x1)/w_updated_nand[2]
plt.plot(x1,x2)
plt.title("Decision boundary for NAND GATE ")
plt.show()
```



In [39]:

```
w_updated_nor = perceptron(x,x_nor,w)
plt.scatter(x.T[1],x.T[2], c= x_nor)
x1 = np.linspace(0,2)
x2 = (-w_updated_nor[0]- w_updated_nor[1]*x1)/w_updated_nor[2]
plt.plot(x1,x2)
plt.title("Decision boundary for NOR GATE ")
plt.show()
```


Decision boundary for NOR GATE
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00