In [1]:
```
import numpy as np
```

In [3]:
```
np.random.seed(42)
weights = np.random.rand(3,1)
bias = np.random.rand(1)
lr = 0.05
```

In [4]:
```
input_set = np.array([[0,1,0],
[0,0,1],
[1,0,0],
[1,1,0],
[1,1,1],
[0,1,1],
[0,1,0]])
labels = np.array([[1,0,0,1,1,0,1]])
labels = labels.reshape(7,1)
```

In [5]:
```
def sigmoid(x):
    return 1/(1+np.exp(-x))
```

In [7]:
```
def sigmoid_derivative(x):
    return sigmoid(x)*(1-sigmoid(x))
```

In [9]:
```
for epoch in range(25000):
    inputs = input_set
    XW = np.dot(inputs,weights) + bias
    z = sigmoid(XW)
    error = z - labels
    #print(error.sum())
    dcost = error
    dpred = sigmoid_derivative(z)
    z_del = dcost*dpred
    inputs = input_set.T
    weights = weights - lr * np.dot(inputs,z_del)
```

In [11]:
```
for num in z_del:
    bias = bias- lr*num
inputs = input_set
XW = np.dot(inputs ,weights)+bias
z = sigmoid(XW)
error = z - labels
```

In [14]:
```
#print(error.sum())
dcost = error
dpred = sigmoid_derivative(z)
z_del = dcost*dpred
inputs = input_set.T
weights = weights -lr*np.dot(inputs,z_del)
```

In [16]:
```
for num in z_del:
    bias = bias -lr*num
single_pt = np.array([1,1,0])
print("weight vector:")
print(weights)
print("Bias vector:")
print(bias)
y = sigmoid(np.dot(single_pt , weights)+ bias)
print("Output Y:")
print(y)
if y<0.5:
    print("Class 0")
else:
    print("Class 1")
```

```
weight vector:
[[-0.37730908]
 [ 6.04411305]
 [-6.42754087]]
Bias vector:
[0.57948483]
Output Y:
[0.99806612]
Class 1
```

In [ ]: