In [6]:
```
import numpy as np

def NN(m1, m2, w1, w2, b):
    z = m1*w1 + m2*w2 + b
    return sigmoid(z)

def sigmoid(x):
    return 1 /(1+ np.exp(-x))

w1 = np.random.randn()
w2 = np.random.randn()
b = np.random.randn()

NN(3, 1.5, w1, w2, b)
```

Out[6]:
```
0.014982592247174329
```

In [1]:

```
import numpy as np

def sigmoid(x):
    return 1/ (1+ np.exp(-x))

training_inputs = np.array([[0, 0, 1],
                            [1, 1, 1],
                            [1, 0, 1],
                            [0, 1, 1]])
training_outputs = np.array([[0, 1, 1, 0]]).T

np.random.seed(1)

synaptic_weights = 2 * np.random.random((3,1))-1

print('Random starting synaptic weights: ')
print(synaptic_weights)

for iteration in range(1):
    input_layer = training_inputs

    outputs = sigmoid(np.dot(input_layer, synaptic_weights))

    print ('Outputs after training:')
    print(outputs)
```

```
Random starting synaptic weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Outputs after training:
[[0.2689864 ]
 [0.3262757 ]
 [0.23762817]
 [0.36375058]]
```

In [5]:

```
import numpy as np

def sigmoid(x):
    return 1/ (1+ np.exp(-x))

def sigmoid_derivative(x):
    return x*(1-x)

training_inputs = np.array([[0, 0, 1],
                            [1, 1, 1],
                            [1, 0, 1],
                            [0, 1, 1]])
training_outputs = np.array([[0, 1, 1, 0]]).T

np.random.seed(1)

synaptic_weights = 2 * np.random.random((3,1))-1

print('Random starting synaptic weights: ')
print(synaptic_weights)

for iteration in range(50000):
    input_layer = training_inputs

    outputs = sigmoid(np.dot(input_layer, synaptic_weights))

    error = training_outputs - outputs
    adjustments = error * sigmoid_derivative(outputs)
    synaptic_weights += np.dot(input_layer.T, adjustments)

print('Synaptic weights after training')
print (synaptic_weights)
print ('Outputs after training:')
print(outputs)
```

```
Random starting synaptic weights:
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Synaptic weights after training
[[11.30926129]
 [-0.20509237]
```

```
 [-5.45001623]]
Outputs after training:
[[0.0042779 ]
 [0.99650925]
 [0.99715469]
 [0.00348742]]
```

In [7]:

In [9]:

In [ ]: