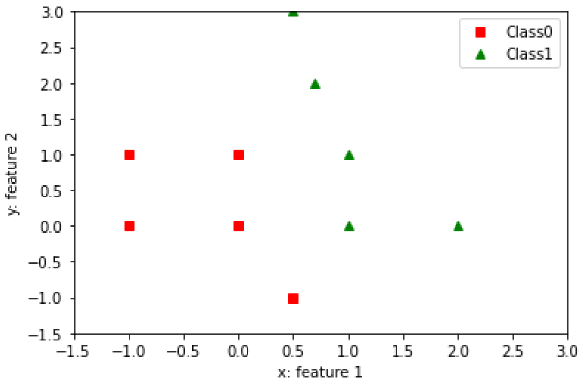
```
import numpy as np
import matplotlib.pyplot as plt

class NeuralNetwork(object):
  def __init__ (self):
    np.random.seed(1)
    self.weight_matrix = 2 * np.random.random((3,1)) - 1
    self.learning_r = 1

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

  def forward_propagation(self, inputs):
    outs=np.dot(inputs, self.weight_matrix)
    return self.sigmoid(outs)

  def train(self, inputs_train, labels_train, num_train_iterations = 10, lr = 1):
    N = inputs_train.shape[0]
    self.learning_r = lr
    cost_func = np.array([])

    for iteration in range(num_train_iterations):
      outputs = self.forward_propagation(inputs_train)
      error = labels_train - outputs
      adjustment = (self.learning_r/N)*np.sum(np.multiply(error,inputs_train), axis = 0)
      cost_func = np.append(cost_func, (1/2*N)*np.sum(np.power(error,2)))
      self.weight_matrix[:,0] += adjustment
      print('Iteration #'+str(iteration))
      plot_fun_thr(inputs_train[:,1:3], labels_train[:,0],
                   self.weight_matrix[:,0], classes)

    plot_cost_function(cost_func, num_train_iterations)

  def pred(self,inputs):
    prob=self.forward_propagation(inputs)
    preds=np.int8(prob>=0.5)
    return preds

def plot_cost_function(J, iterations):
    x = np.arange(iterations, dtype = int)
    y = J
    plt.plot(x,y)
    plt.axis([-1, x.shape[0]+1, -1,np.max(y)+1])
    plt.title('Learning Curve')
    plt.xlabel('x:  iteration number')
    plt.ylabel('y:  J(0)')
    plt.show

def plot_fun(features,labels,classes):
    #plotting data points
    plt.plot(features[labels[:]==classes[0],0],features[labels[:]==classes[0],1], 'rs',
           features[labels[:]==classes[1],0], features[labels[:]==classes[1],1], 'g^')
    plt.axis([-1.5,3,-1.5,3])
    plt.xlabel('x: feature 1')
    plt.ylabel('y: feature 2')
    plt.legend(['Class'+str(classes[0]), 'Class'+str(classes[1])])
    plt.show

def plot_fun_thr(features,labels,thre_parms,classes):
    #plotting data points
    plt.plot(features[labels[:]==classes[0],0],features[labels[:]==classes[0],1], 'rs',
           features[labels[:]==classes[1],0], features[labels[:]==classes[1],1], 'g^')
    plt.axis([-1.5,3,-1.5,3])
    x1 = np.linspace(-1,2,50)
    x2 = -(thre_parms[1]*x1+thre_parms[0])/thre_parms[2]    #ax1 + bx2 + c = 0 --> -(ax1+c)/b
    plt.plot(x1,x2, '-r')
    plt.xlabel('x: feature 1')
    plt.ylabel('y: feature 2')
    plt.legend(['Class'+str(classes[0]), 'Class'+str(classes[1])])
    plt.legend(['Class'+str(classes[0]), 'Class'+str(classes[1])])
    plt.show
```

```
features=np.array([[1,1], [1,0], [0,1], [0.5,-1], [0.5, 3], [0.7, 2], [-1,0], [-1,1], [2,0], [0,0]])
labels = np.array([1,1,0,0,1,1,0,0,1,0])
classes=[0,1]
```

```
plot_fun(features,labels,classes)
```



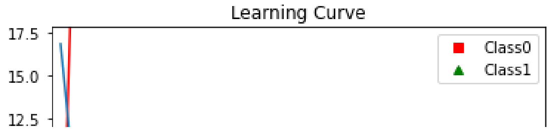
```
bias=np.ones((features.shape[0],1))
features = np.append(bias,features,axis =1)
print('Features matrix')
print(features)
print('Features matrix shape')
print(features.shape)
```

```
     Features matrix
     [[ 1.   1.   1. ]
      [ 1.   1.   0. ]
      [ 1.   0.   1. ]
      [ 1.   0.5 -1. ]
      [ 1.   0.5  3. ]
      [ 1.   0.7  2. ]
      [ 1.  -1.   0. ]
      [ 1.  -1.   1. ]
      [ 1.   2.   0. ]
      [ 1.   0.   0. ]]
     Features matrix shape
     (10, 3)
```

```
neural_network = NeuralNetwork()
print ('Random weights at the start of training')
print (neural_network.weight_matrix)
neural_network.train(features, np.expand_dims(labels,axis=1), 50)
```

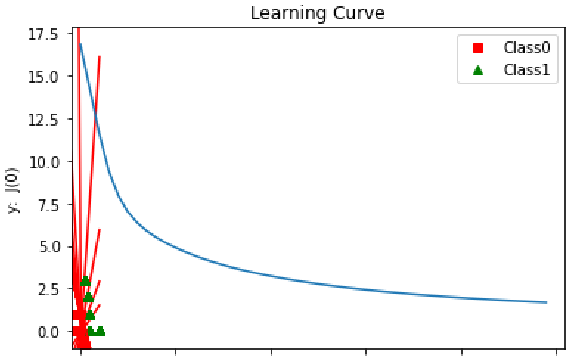
```
print('New weights after training')
print(neural_network.weight_matrix)
```

```
Random weights at the start of training
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Iteration #0
Iteration #1
Iteration #2
Iteration #3
Iteration #4
Iteration #5
Iteration #6
Iteration #7
Iteration #8
Iteration #9
Iteration #10
Iteration #11
Iteration #12
Iteration #13
Iteration #14
Iteration #15
Iteration #16
Iteration #17
Iteration #18
Iteration #19
Iteration #20
Iteration #21
Iteration #22
Iteration #23
Iteration #24
Iteration #25
Iteration #26
Iteration #27
Iteration #28
Iteration #29
Iteration #30
Iteration #31
Iteration #32
Iteration #33
Iteration #34
Iteration #35
Iteration #36
Iteration #37
Iteration #38
Iteration #39
Iteration #40
Iteration #41
Iteration #42
Iteration #43
Iteration #44
Iteration #45
Iteration #46
Iteration #47
Iteration #48
Iteration #49
New weights after training
[[-2.16960572]
 [ 3.46870407]
 [ 1.34808305]]
```



```
neural_network = NeuralNetwork()
print ('Random weights at the start of training')
print (neural_network.weight_matrix)
neural_network.train(features, np.expand_dims(labels,axis=1), 50, 0.5)


print('New weights after training')
print(neural_network.weight_matrix)
```

```
Random weights at the start of training
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Iteration #0
Iteration #1
Iteration #2
Iteration #3
Iteration #4
Iteration #5
Iteration #6
Iteration #7
Iteration #8
Iteration #9
Iteration #10
Iteration #11
Iteration #12
Iteration #13
Iteration #14
Iteration #15
Iteration #16
Iteration #17
Iteration #18
Iteration #19
Iteration #20
Iteration #21
Iteration #22
Iteration #23
Iteration #24
Iteration #25
Iteration #26
Iteration #27
Iteration #28
Iteration #29
Iteration #30
Iteration #31
Iteration #32
Iteration #33
Iteration #34
Iteration #35
Iteration #36
Iteration #37
Iteration #38
Iteration #39
Iteration #40
Iteration #41
Iteration #42
Iteration #43
Iteration #44
Iteration #45
Iteration #46
Iteration #47
Iteration #48
Iteration #49
New weights after training
[[-1.47953369]
 [ 2.56651734]
 [ 1.08127652]]
```



```
neural_network = NeuralNetwork()
print ('Random weights at the start of training')
print (neural_network.weight_matrix)
```

```
neural_network.train(features, np.expand_dims(labels,axis=1), 50, 0.1)


print('New weights after training')
print(neural_network.weight_matrix)
```

```
Random weights at the start of training
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Iteration #0
```

```
neural_network = NeuralNetwork()
print ('Random weights at the start of training')
print (neural_network.weight_matrix)
neural_network.train(features, np.expand_dims(labels,axis=1), 50, 0.01)


print('New weights after training')
print(neural_network.weight_matrix)
```

```
Random weights at the start of training
[[-0.16595599]
 [ 0.44064899]
 [-0.99977125]]
Iteration #0
Iteration #1
Iteration #2
Iteration #3
Iteration #4
Iteration #5
Iteration #6
Iteration #7
Iteration #8
Iteration #9
Iteration #10
Iteration #11
Iteration #12
Iteration #13
Iteration #14
Iteration #15
Iteration #16
Iteration #17
Iteration #18
Iteration #19
Iteration #20
Iteration #21
Iteration #22
Iteration #23
Iteration #24
Iteration #25
Iteration #26
Iteration #27
Iteration #28
Iteration #29
Iteration #30
Iteration #31
Iteration #32
Iteration #33
Iteration #34
Iteration #35
```