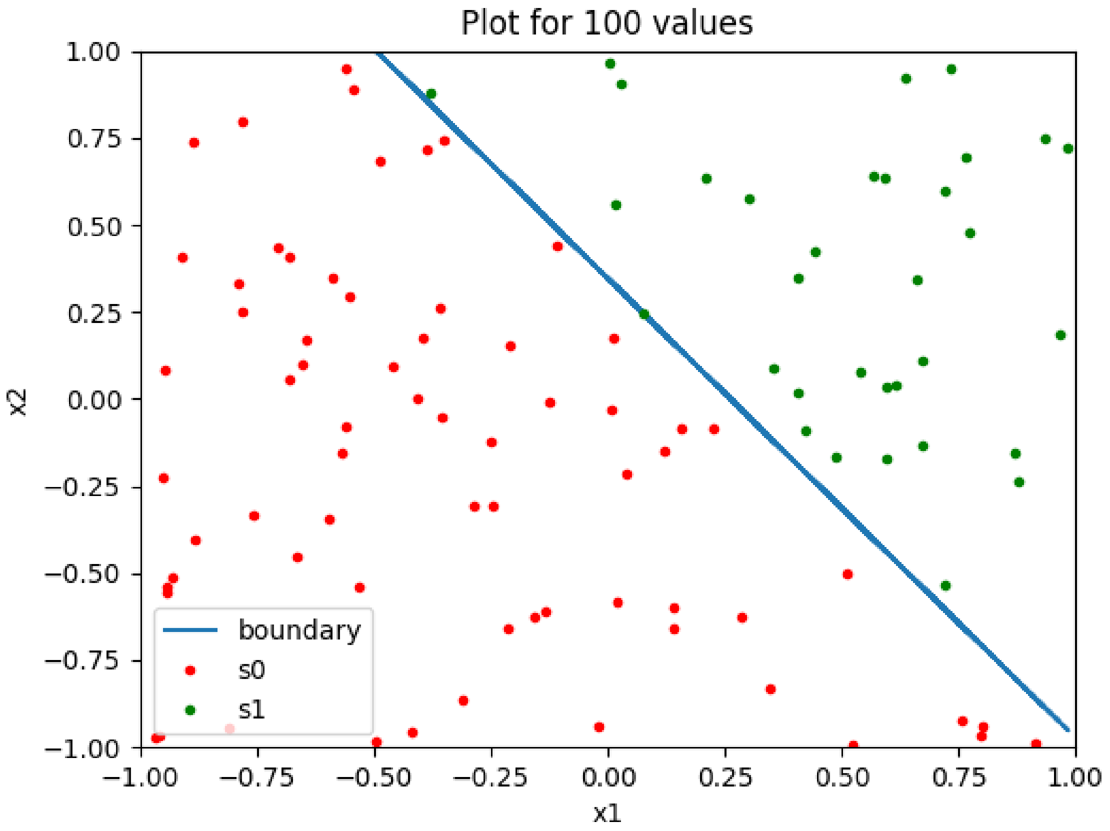
Assignment 2:



The above plot is for S0 and S1 where S0 are red dots and S1 are green dots, for 100 values of [x1 ,x2]. The blue line is the equation w0 + w1x1 + w2x2 = 0

Here are the weights picked:

w0 = -0.1828178779437994

w1 = 0.6948674738744653

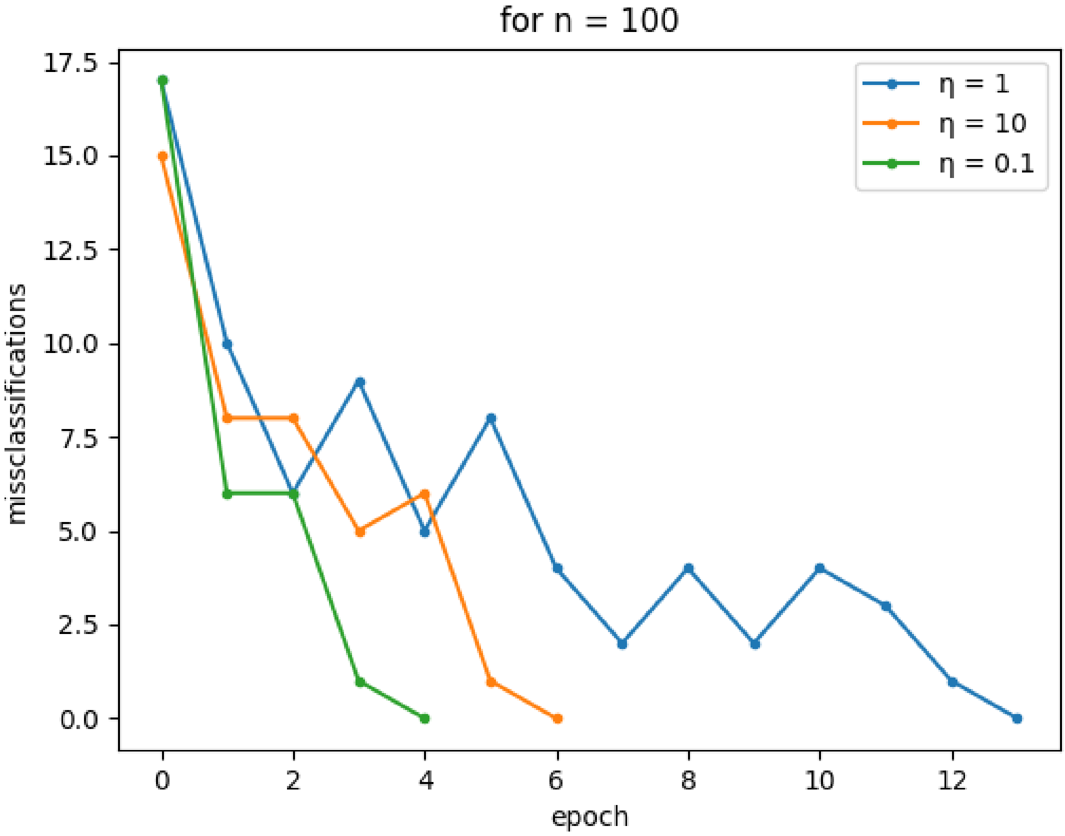
w2 = 0.5275492379532281

$w'_0$ = -0.4898619485211566

$w'_1$= -0.009129825816118098

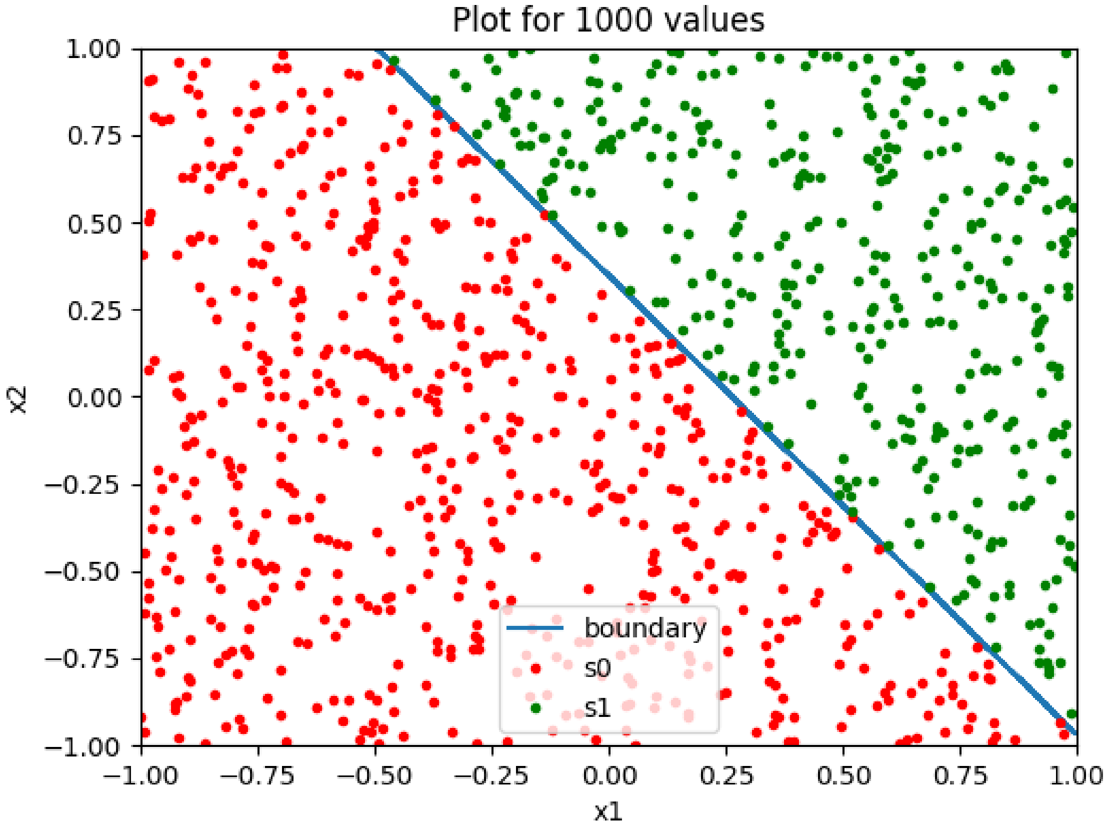
$w'_2$= -0.10101787042252375

Graph for epochs vs misclassifications for n = 100:

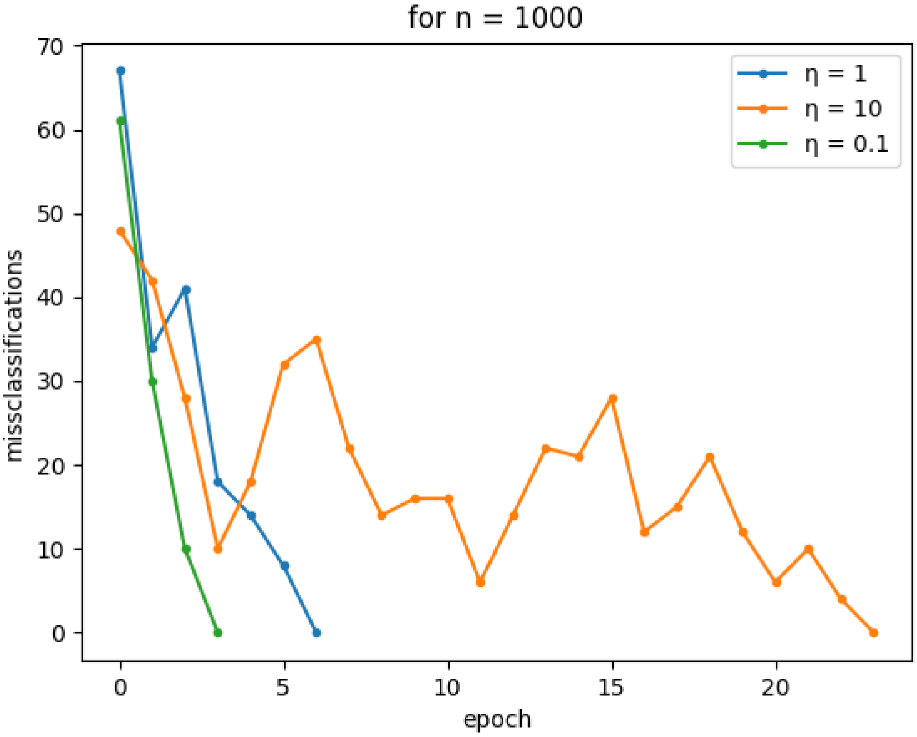


For eta = 0.1, the number of epochs is the least. For Eta = 1, the number of epochs is the highest. It takes longer to converge for eta = 1 than for eta = 10 and the least amount of convergence time is with eta = 0.1

The following plot shows S1 and S0 for n = 1000 values of X. The legend is similar to the one above.



Graph for epochs vs misclassifications for n = 1000:



The rate of convergence is the fastest for eta = 0.1 just as before but this time the slowest convergence happens with eta = 10. There are times when the number of misclassifications increase in some epochs in the case of eta = 10 which signifies there may be some overshooting of weights happening.

If we had started with different w0, w1, w2, S, ${w'}_0$, ${w'}_1$, ${w'}_2$: we would get completely different results as the effect of eta varies with all these factors. It is not the same for all weights and input data set.

Code:

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


rand.seed(1)
w0 = rand.uniform(-0.25, 0.25)
w1 = rand.uniform(-1, 1)
w2 = rand.uniform(-1, 1)


print('w0 = ' + str(w0))
print('w1 = ' + str(w1))
print('w2 = ' + str(w2))


w0new = rand.uniform(-1, 1)
w1new = rand.uniform(-1, 1)
w2new = rand.uniform(-1, 1)


print('w0new = ' + str(w0new))
print('w1new = ' + str(w1new))
print('w2new = ' + str(w2new))


def getOutput(x1, x2, w0, w1, w2):
        return w0 + w1*x1 + w2*x2


def training(s, eta, n, s0, s1, w0, w1, w2):
    print('initial weights are: \n' + 'w0 = ' + str(w0) + '\nw1 = ' + str(w1) + '\nw2 = ' + str(w2))

    missc_list = []
    epoch = 0
    while(True):
        epoch += 1
        #print(epoch)
        miscalculations = 0
        for x, y in s.T:
            if getOutput(x, y, w0, w1, w2) < 0 and [x, y] not in s0:
                miscalculations += 1
                w0 += eta
                w1 += eta*x
                w2 += eta*y
            elif getOutput(x, y, w0, w1, w2) >= 0 and [x, y] not in s1:
                miscalculations += 1
                w0 -= eta
                w1 -= eta*x
                w2 -= eta*y
        if miscalculations == 0:
            missc_list.append(0)
            break
        missc_list.append(miscalculations)

    print('eta = ' + str(eta) + '\n' + 'n = ' + str(n) + '\n' + 'final weights are: \n' + 'w0 = ' + str(w0) + '\nw1 = ' + str(w1) + '\nw2 = ' + str(w2))
    print(missc_list)
    plt.plot(missc_list, marker='.', linestyle='-', label = 'η = '+str(eta))
    plt.xlabel('epoch')
    plt.ylabel('missclassifications')
    plt.title('for n = ' + str(n))

    plt.legend()


def assignment(n, w0, w1, w2):

    s = np.empty([2, n])


    for i in range(n):

      s[0][i] = rand.uniform(-1, 1)

      s[1][i] = rand.uniform(-1, 1)


    print(w0)


    s0x = []

    s0y = []

    s1x = []

    s1y = []


    for i in range(n):

        if getOutput(s[0][i], s[1][i], w0, w1, w2) < 0:

            s0x.append(s[0][i])

            s0y.append(s[1][i])

        else:

            s1x.append(s[0][i])

            s1y.append(s[1][i])


    s0 = []

    s1 = []


    for i in range(len(s0x)):

        s0.append([s0x[i], s0y[i]])

    for i in range(len(s1x)):
        s1.append([s1x[i], s1y[i]])

    plt.plot(s[0], (-s[0]*w1 - w0)/w2, linestyle ='solid', label = 'boundary')

    plt.plot(s0x, s0y, 'r.', label = 's0')

    plt.plot(s1x, s1y, 'g.', label = 's1')

    plt.xlim(-1, 1)
    plt.ylim(-1, 1)
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.title('Plot for '+str(n)+' values')
    plt.legend()
    plt.show()

    print(s.T.shape)

    training(s, 1, n, s0, s1, w0new, w1new, w2new)
    training(s, 10, n, s0, s1, w0new, w1new, w2new)
    training(s, 0.1, n, s0, s1, w0new, w1new, w2new)



assignment(100, w0, w1, w2)
plt.show()
assignment(1000, w0, w1, w2)
plt.show()
```