CSE 634: Project

I All source codes

For Question 1 named neuron_q1.py

```
import numpy as np
import numpy.linalg
import scipy
import cython
import matplotlib.pyplot as plt




M1 = np.array([[3],[4]]) # create the array for mean
M2 = np.array([[2],[3]])
C  = np.array([[1,0],[0,2]]) # create the array for C
C_inverse = numpy.linalg.inv(C) #inverse C
W_0 = 0.5*((M2.T).dot(C_inverse).dot(M2)-(M1.T).dot(C_inverse).dot(M1)) # Calculate b =W0
W_1_2 = C_inverse.dot(M1-M2) #calculate W=(w1,w2)
read_two_cols = np.loadtxt('data2.txt', usecols=(0,1))
print read_two_cols
read_desired_ouput = np.loadtxt('data2.txt', usecols=2)
#print read_desired_ouput

add_col = np.ones((200,1))
data= np.append(add_col,read_two_cols,1)
W = np.append(W_0,W_1_2,0)
print W

X = data.T
Y = W.T.dot(X)
print Y

##iterate over output array to classify

actual_output = np.array([])


for i in np.nditer(Y):
    if i >= 0:
        actual_output = numpy.append(actual_output,1)
    else:
        actual_output = numpy.append(actual_output,-1)

counter = 0 #counter for misclassification

#compare  actual_putput and desired_ouput array

for i, j in zip(np.nditer(actual_output),np.nditer(read_desired_ouput)):
    if i!=j:
        counter +=1

print 'number of misclassification: ' + str(counter)

#plot the scatter points with bayes classification and LMS classification
x = np.loadtxt('data2.txt', usecols=0)
y= np.loadtxt('data2.txt', usecols= 1)
```

```
x1 = numpy.linspace(-15,15,100)
y1 = x1*(-0.5) +4.25
y2 = x1*(0.00560961)/(-0.25895751) + (0.25895751/-0.25895751) # W=[-0.25895751 -0.00560961
0.14153423]

fig, ax= plt.subplots()
for output in np.unique(read_desired_ouput):
       mask = read_desired_ouput == output
       ax.plot(x[mask], y[mask], linestyle='none', marker='o', label=read_desired_ouput)
ax.plot(x1,y1)
ax.plot(x1,y2)
plt.show()
```

For Question 2 named neuron_q2.py

```
import numpy as np
import numpy.linalg
import scipy
import cython
import matplotlib.pyplot as plt

epoch = np.arange(200) #create the array range = 0...199



read_two_cols = np.loadtxt('data2.txt', usecols=(0,1))
read_desired_ouput = np.loadtxt('data2.txt', usecols=2)
add_col = np.ones((200,1))
data= np.concatenate((add_col,read_two_cols),1)
weights_array = np.zeros(shape=(200,3))
MSE_Array = np.zeros(shape=(200,1))
p = 200





def LMS(desired_signal,learning_rate):
    # initialize Weight value [0,1,2]
    weights = np.array([0,1,2])
    print weights
    error = np.zeros(200)
    error_absolute = np.zeros(200)


    #LMS algorithm
    for n in range(200):
        error[n] = desired_signal[n] - weights.T.dot(data[n])
        weights = weights + learning_rate*data[n]*error[n]
        weights_array[n] = weights # add weight value to a new array
    # Mean-square- error vs epoch

    for j in range(200):
        diff =0.0
        for i in range(200):
                diff += (desired_signal[i]-weights_array[j].dot(data[i]))**2

        MSE = diff/(2*p)

        MSE_Array[j] = MSE



    #Plot the MSE vs epoch
    plt.plot(epoch,MSE_Array)
    plt.show()

    return weights_array[np.argmin(MSE_Array)]



#learning rate = 0.01
min_weight = LMS(read_desired_ouput, 0.02)
print 'learning rate =0.01 which has Weights value: '+str(min_weight)

#learning rate = 0.05
min_weight_2 = LMS(read_desired_ouput, 0.05)
print 'learning rate =0.05 which has Weights value: '+str(min_weight_2)
#learning rate = 0.1
min_weight_3= LMS(read_desired_ouput, 0.1)
print 'learning rate =0.1 which has Weights value: '+str(min_weight_3)




#Calculate the number of misclassification
stored_output = min_weight.reshape(min_weight.shape + (1,)).T.dot(data.T)
actual_output = np.array([])

for i in np.nditer(stored_output):
    if i >= 0:
        actual_output = numpy.append(actual_output,1)
    else:
        actual_output = numpy.append(actual_output,-1)

counter = 0 #counter for misclassification

#compare  actual_putput and desired_ouput array

for i, j in zip(np.nditer(actual_output),np.nditer(read_desired_ouput)):
    if i!=j:
        counter +=1

print 'number of misclassification: ' + str(counter)
```

II. The value of the weight vector found by the Bayes method W0 = -4.25 ; W1=1;W2 = 0.5
[[-4.25]

```
 [ 1.  ]
 [ 0.5 ]]
```

III. The value of the weight vector found by the LMS method W0 = -0.25895751; W1= -0.00560961; W2=0.14153423 at learning rate = 0.01

```
[-0.25895751 -0.00560961  0.14153423]
```

IV The 3 plots in item B5

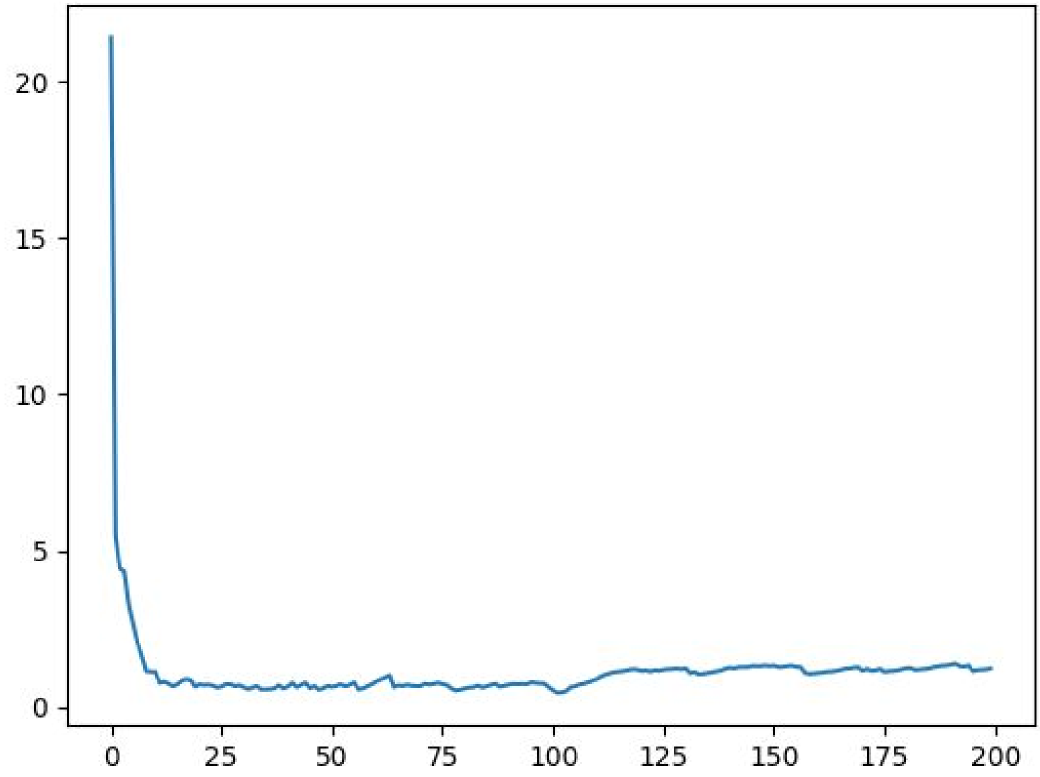

Figure 1: Learning rate = 0.01

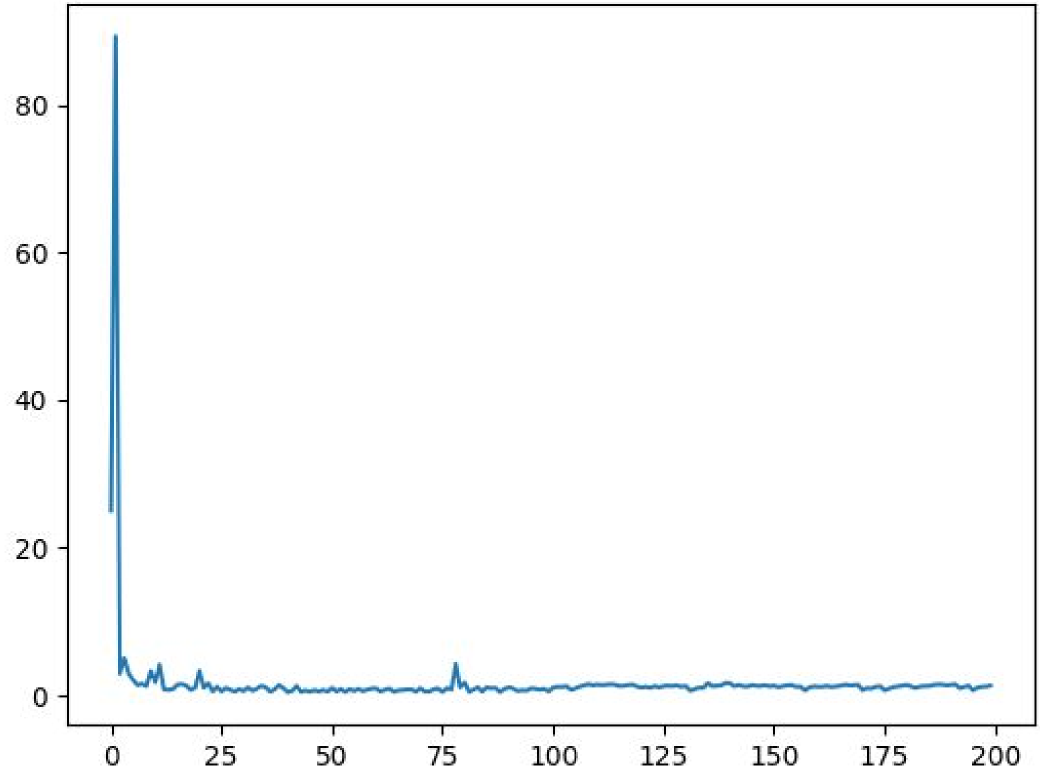

Figure 2: Learning rate = 0.05

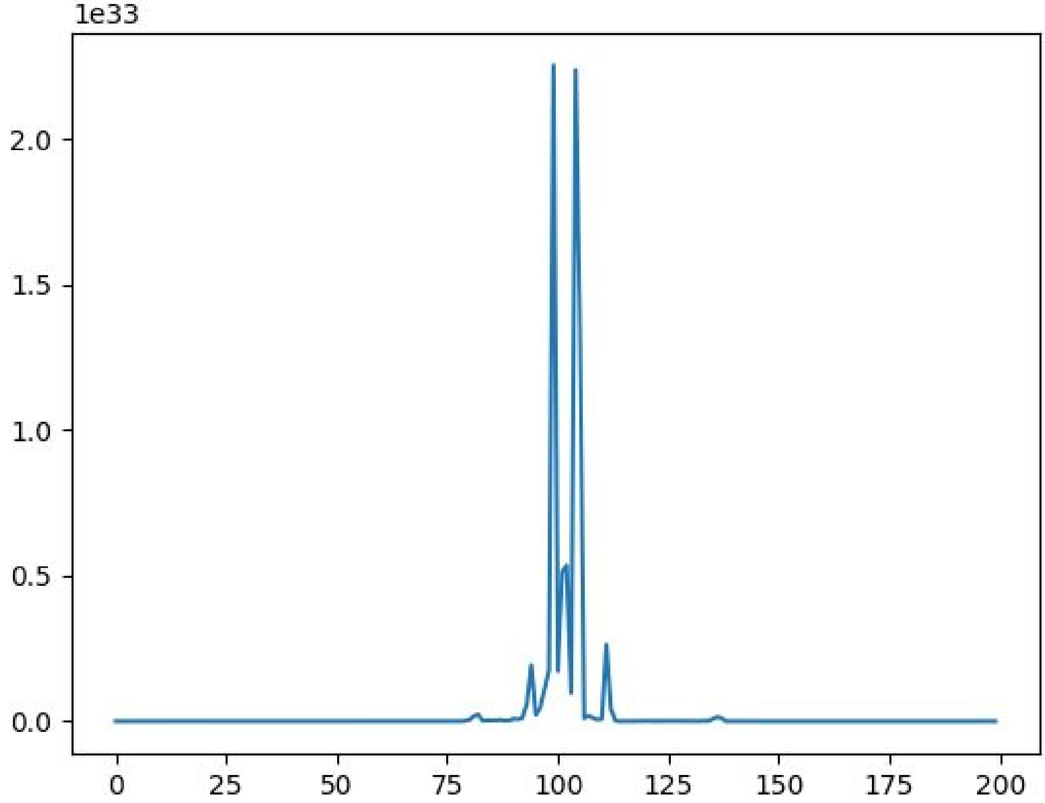

Figure 3: Learning rate = 0.1

V

For Bayes classification: The number of misclassification is 51 which is about 26%
For LMS classification: The number of misclassification is 102 which is about 51%
→ Bayes classification is better than LMS

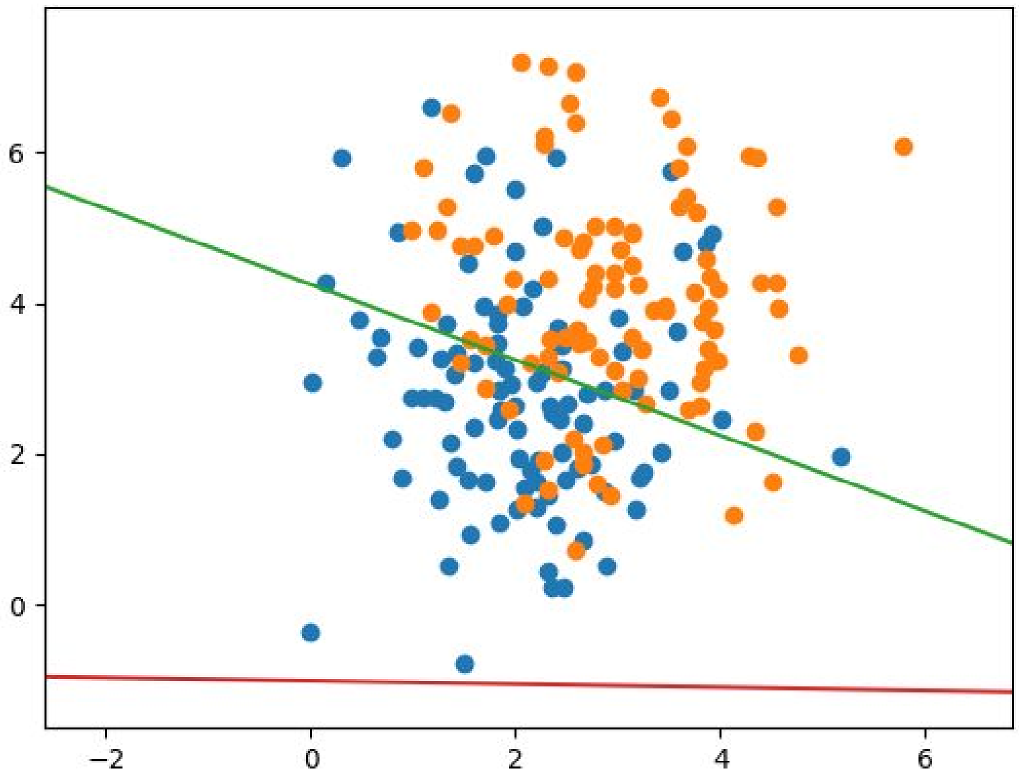

Figure 5: Bayes vs LMS

The greenline (above) is Bayes classification

The readline (botttom) is LMS classification