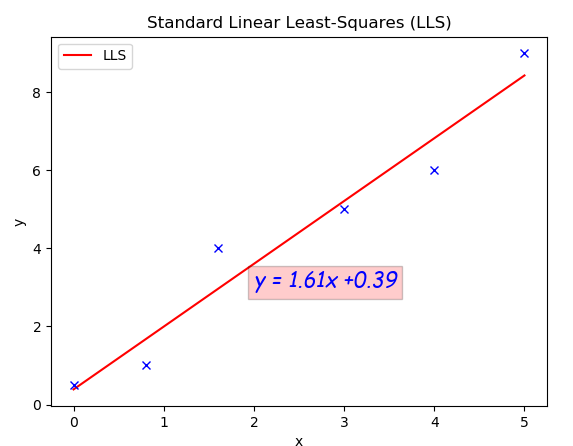
Q4a



W = 1.61, b = 0.39

############################source code for Q4a####################

import numpy as np

import matplotlib.pyplot as plt

x = np.array([[1,0],[1,0.8],[1,1.6],[1,3],[1,4],[1,5]])

d = np.array([[0.5] , [1], [4], [5], [6], [9]])

w = np.mat(np.linalg.inv((np.mat(np.transpose(x)))\*(np.mat(x))))\*np.mat(np.transpose(x))\*np.mat(d)

# w0 = 0.38733906 w1 = 1.60944206

X = np.arange(0,6,1)

y = w[1,0]\*X + w[0,0]

plt.figure()

plt.title('Standard Linear Least-Squares (LLS)')

plt.xlabel('x')

plt.ylabel('y')

plt.plot(X,y,color= 'red',label = "LLS")

plt.plot(x[:,1:2],d[:,:],'bx')

plt.text(2, 3, "y = "+str("%.2f" %w[1,0]) +"x +"+str("%.2f" %w[0,0]), size = 15,\

         family = "fantasy", color = "b", style = "italic", weight = "light",\

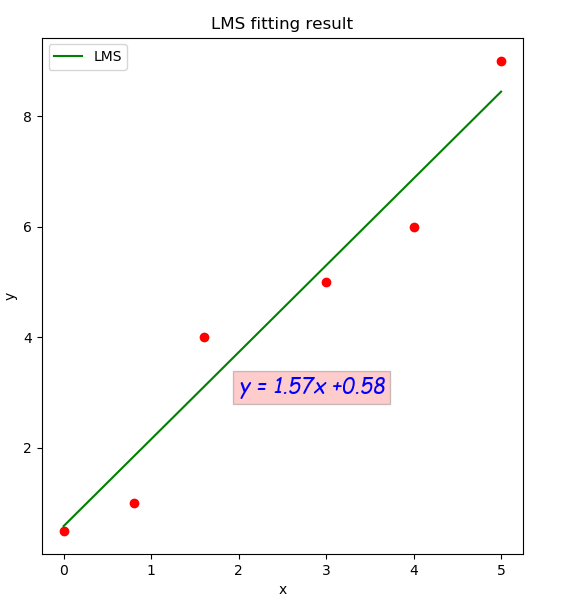
         bbox = dict(facecolor = "r", alpha = 0.2))

plt.legend()

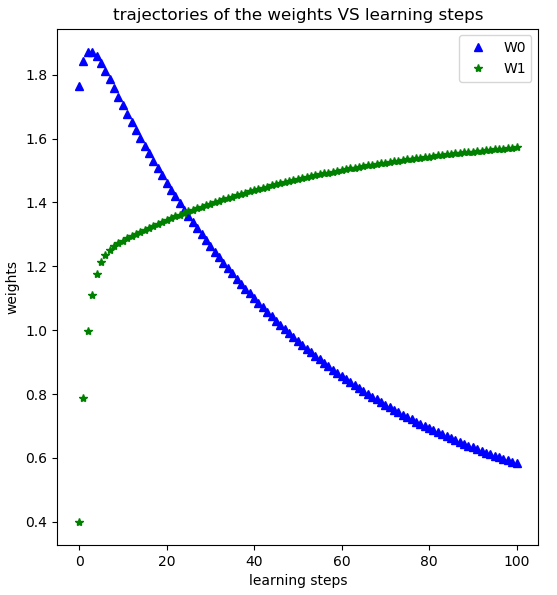
plt.show()

############################end of Q4a###########################

Q4b.



W=1.57, b = 0.58



The weights will not converge. the weights will keep updated until the times of iteration are met.

########################source code for Q4b################

import numpy as np

import matplotlib.pyplot as plt

x = np.array([[1,0],[1,0.8],[1,1.6],[1,3],[1,4],[1,5]])

d = np.array([[0.5] , [1], [4], [5], [6], [9]])

#learning rate

learn\_rate = 0.009

# 100 epochs

iteration = 100

def train(x,iteration,learn\_rate):

    #initial weights

    np.random.seed(0)

    weights = np.random.randn(1,x.shape[1])

    W = weights

    for i in range(iteration):

        for j in range(x.shape[0]):

            error = d[j] - np.mat(x[j,:])\*np.transpose(weights)

            weights = weights + learn\_rate\*error\*x[j,:]

        W = np.append(W,weights,axis = 0)

        # print("Updated weights is " + str(weights))

    # print(str(iteration) + " times of epochs")

    return W

W\_x = train(x,iteration,learn\_rate)

# print(W\_x)

X = np.arange(0,6,1)

y = W\_x[W\_x.shape[0]-1,1]\*X + W\_x[W\_x.shape[0]-1,0]

plt.figure()

plt.subplot(1,2,1)

plt.title('LMS fitting result')

plt.xlabel('x')

plt.ylabel('y')

plt.plot(x[:,1:2],d,'ro')

plt.text(2, 3, "y = "+str("%.2f" %W\_x[W\_x.shape[0]-1,1]) +"x +"+str("%.2f" %W\_x[W\_x.shape[0]-1,0]), size = 15,\

         family = "fantasy", color = "b", style = "italic", weight = "light",\

         bbox = dict(facecolor = "r", alpha = 0.2))

plt.plot(X,y,color= 'green',label = "LMS")

plt.legend()

plt.subplot(1,2,2)

plt.title('trajectories of the weights VS learning steps')

plt.xlabel('learning steps')

plt.ylabel('weights')

plt.plot(range(W\_x.shape[0]),W\_x[:,0:1],'b^',range(W\_x.shape[0]),W\_x[:,1:2],'g\*')

plt.legend(('W0','W1'),loc = 'upper right')

plt.show()

########################end of Q4b#######################

Q4c.

The final results obtained by LLS and LMS methods are very close. The results obtain from LLS method is unique. The results obtained from LMS method are iterated from initial weights, iteration stops when a convergence criterion is satisfied. Different initial weights can result different weights.

Q4d.

A smaller learning rate make small changes on weights each update, this allow the model to learn a more optimal set of weights. Whereas, a larger learning rate can make rapid changes each update, this will result a sub-optimal set of weights. If learning rate is too large or too small, the model may result in a failure train.

######################source code for Q4d#####################

import numpy as np

import matplotlib.pyplot as plt

x = np.array([[1,0],[1,0.8],[1,1.6],[1,3],[1,4],[1,5]])

d = np.array([[0.5] , [1], [4], [5], [6], [9]])

#learning rate

learn\_rate = 0.009

# 100 epochs

iteration = 100

def train(x,iteration,learn\_rate):

    #initial weights

    np.random.seed(0)

    weights = np.random.randn(1,x.shape[1])

    W = weights

    for i in range(iteration):

        for j in range(x.shape[0]):

            error = d[j] - np.mat(x[j,:])\*np.transpose(weights)

            weights = weights + learn\_rate\*error\*x[j,:]

        W = np.append(W,weights,axis = 0)

        # print("Updated weights is " + str(weights))

    # print(str(iteration) + " times of epochs")

    return W

W\_x = train(x,iteration,learn\_rate)

# print(W\_x)

X = np.arange(0,6,1)

y = W\_x[W\_x.shape[0]-1,1]\*X + W\_x[W\_x.shape[0]-1,0]

plt.figure()

plt.title('LMS fitting result at '+str(learn\_rate)+' learning rate')

plt.xlabel('x')

plt.ylabel('y')

plt.plot(x[:,1:2],d,'ro')

plt.text(2, 3, "y = "+str("%.2f" %W\_x[W\_x.shape[0]-1,1]) +"x +"+str("%.2f" %W\_x[W\_x.shape[0]-1,0]), size = 15,\

         family = "fantasy", color = "b", style = "italic", weight = "light",\

         bbox = dict(facecolor = "r", alpha = 0.2))

plt.plot(X,y,color= 'green',label = "LMS")

plt.legend()

plt.show()

#####################end of Q4###############################