Theory Questions.
Ss4328, CS383 hw3

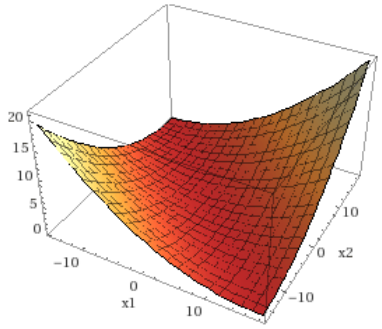
1. In this question, we add a column on ones to X to add bias. Then we try to find the LSE estimate.

   X = [-2,-5,-3,0,-8,-2,1,5,-1,6,] y=[1,-4,1,3,11,5,0,-1,-3,1]
   Data x-xinv =[-1.1,-4.1,-2.1,0.9,-7.,-1.1,1.9,5.9,-0.1,6.9]
   Data y-yinb = []-0.4,-5.4,-0.4,1.6,9.6,3.7,-1.4,-2.4,-4.4444]
   x-xinv . y-yinv =[0.44,22.14,0.84,1.44,-68.2,-4.0,-2.7,-14.188,0.45,-2.77]

   equation => y=mx+c
   1.4 = intercept+ -0.412(-0.9)
   interept = 1.0286

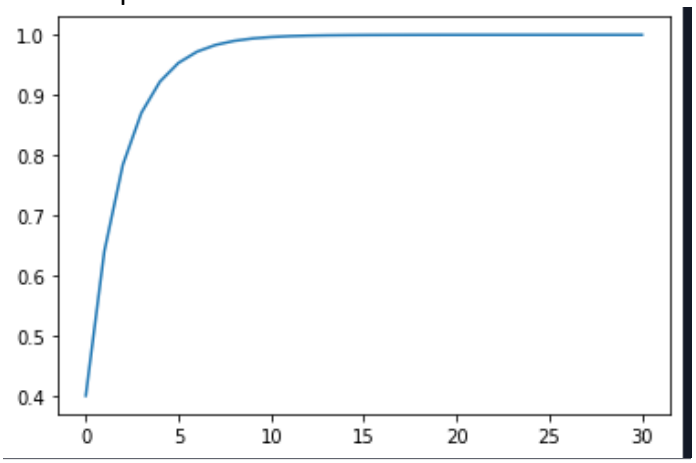
   Basically, we follow the steps we did in Part 3 linear regression. To reach the eqn above.

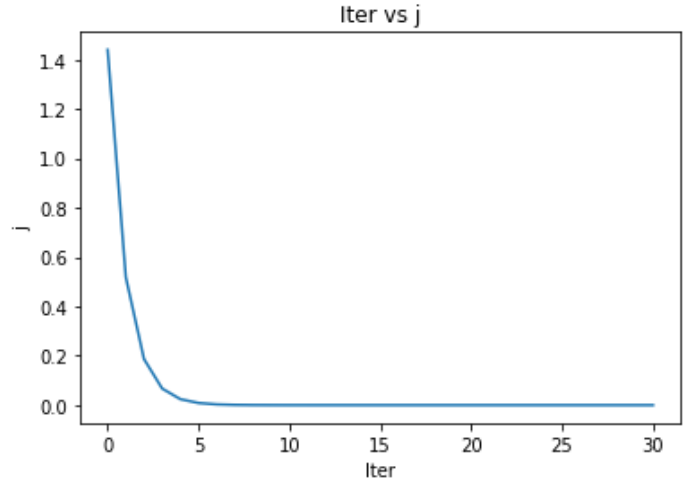
2.
   a. Basically we have to differentiate wrt x1, x2 both of which's derivative gives us 2(x1+x2-2)

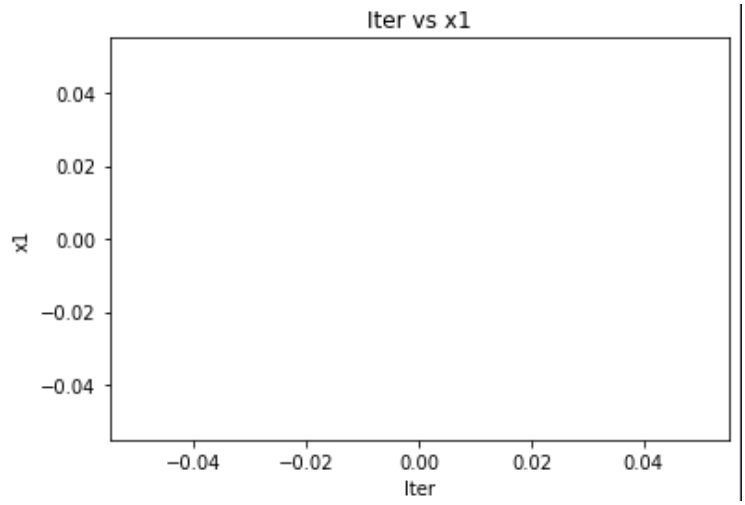


   b. Gradient 3-d plot:
   c. For the values of x1 and x2 that minimize J, we have to find the global minima for the given equation. So x1+x2-2 =0 => x1+x2 = 2.
   So here, x1 is min 0 when x2 is 2 and similarly X2 is min=0 when x1 =2

Plots for part 2







Part 4 Results

The data for the fold verification done 20x validations is:
[15.03356438 17.56930347 19.1857595 15.03356438 17.56930347 19.1857595
 15.03356438 17.56930347 19.1857595 15.03356438 17.56930347 19.1857595
 15.03356438 17.56930347 19.1857595 15.03356438 17.56930347 19.1857595
 15.03356438 17.56930347]

Mean: 17.166731597514257
Std dev: 1.69478770834235

Report:

For part 2, I followed the gradient descent as described in the slides. Basically, here we try to find local minima for the descend and stop when the change in our variables is extremely small (provided) or when we exhaust a given number of iterations. Then matplotlib plots the graphs needed from the prompt.

For part 3, I used pandas to load and align the data(feature->age becomes last feature which becomes y). Then I shuffled the data, created training and testing matrices from the data. I then add bias which is a column of ones. And finally , I computed the weights by using the formula “theta = np.dot(np.dot(np.linalg.inv(XtX), Xt), yTrain)” where theta is the weight collection. Then I computed the RMSE as asked in the prompt which is

RMSE = np.sqrt((np.mean(np.square(actual_y-pred_y),axis=0)))

For part 4, I abstracted the functionality from part 3. Then I made functions for RMSE, Linre, getPredictedValues, and Sq.Errors.

Then I did S-fold validation. For 20 times, I randomized the data and created random folds based on the given value of ‘s’. For each fold as testing set, I calculated the weights and sample squared errors. Then I calculated the mean of each fold iteration sample squared error, and fed it to a numpy array which kept tracks for s values. Then once S-fold is done for a value of 1->20, I calculate the RMSE for the fold mean, and then store it in a 20-sized RMSE array.

At the end, I print the RMSE array, along with its mean and standard deviation.