**Problem assignment 4**

**Due: Thursday, February 15, 2018**

**Ang Li**

**Linear Regression**

In this problem set we use the Boston Housing dataset from the CMU StatLib Library that concerns prices of housing in Boston suburbs. A data sample consists of 13 attribute values (indicating parameters like crime rate, accessibility to major highways etc.) and the median value of housing in thousands we would like to predict.

The data are in the file *housing.txt*, the description of the data is in the file housing desc.txt on the course web page.

Part 1. Exploratory data analysis.

Examine the dataset *housing.txt* using Matlab. Answer the following questions.

(a) How many binary attributes are in the data set? List the attributes.

*There is 1 binary attribute in the dataset: column 4: CHAS*

(b) Calculate and report correlations in between the first 13 attributes (columns) and

the target attribute (column 14). What are the attribute names with the highest

positive and negative correlations to the target attribute?

*The correlation between feature 1-13 with target attribute:*

*cor\_vec =*

*Columns 1 through 9*

*-0.3883 0.3604 -0.4837 0.1753 -0.4273* ***0.6954*** *-0.3770 0.2499 -0.3816*

*Columns 10 through 13*

*-0.4685 -0.5078 0.3335* ***-0.7377***

*The largest positive correlation is: attribute 6 (RM: average number of rooms per dwelling)*

*The highest negative correlation is: attribute 13 (MEDV: Median value of owner-occupied homes in $1000's)*

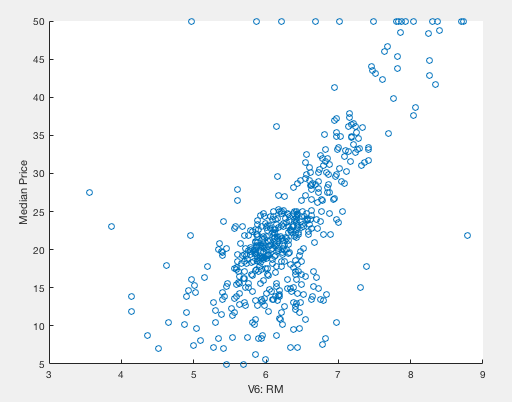
(c) Note that the correlation is a linear measure of similarity. Examine scatter plots

for attributes and the target attribute using the function you wrote in problem set 1.

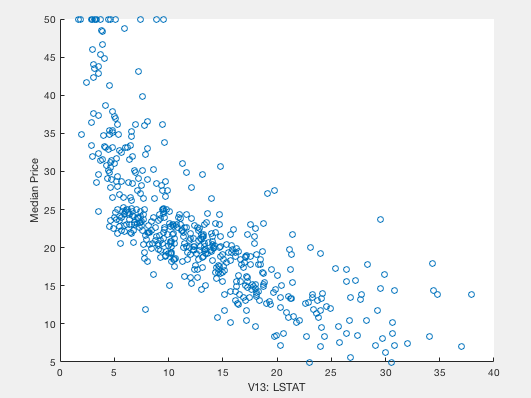
Which scatter plot looks the most linear, and which looks the most nonlinear? Plot

these scatter plots and briefly (in 1-2 sentences) explain your choice.

As correlation examined the linear similarity, so attribute 6 (RM) should most likely have the linear relationship with the median price, as demonstrated in the plot below



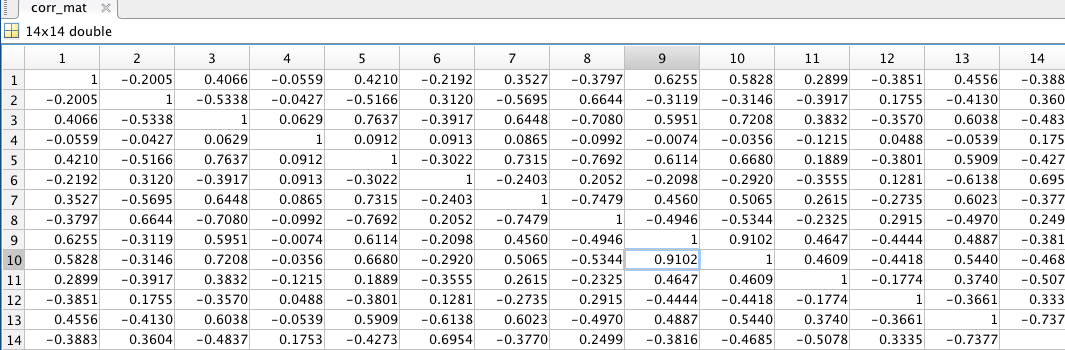
The attribute 13 (LSTAT: % lower status of the population) seems to be non-linear as demonstrated in the following plot, which seems to be curve instead of a line.



(d) Calculate all correlations between the 14 columns (using the corrcoef function).

Which two attributes have the largest mutual correlation in the dataset?

As shown in the screenshot below, attribute 9 (RAD: index of accessibility to radial highways) and 10 (TAX: full-value property-tax rate per $10,000) show to have the highest correlation



Part 2. Linear regression.

Our goal is to predict the median value of housing based on the values of 13 attributes.

For your convenience the data has been divided into two datasets: (1) a training dataset *housing\_train.txt* you should use in the learning phase, and (2) a testing dataset *housing­\_test.txt*

to be used for testing.

Assume that we choose a linear regression model to predict the target attribute. Using Matlab:

(a) Write a function ***LR\_solve*** that takes X and y components of the data (X is a matrix of inputs where rows correspond to examples) and returns a vector of coefficients w with the minimal mean square fit. (Hint: you can use backslash operator ’/’ to do the least squares regression directly; check Matlab’s help).

Code LR\_solve.m

(b) Write a function ***LR­\_predict*** that takes input components of the test data (X) and a fixed set of weights (w), and computes vector of linear predictions y.

Code LR\_predict.m

(c) Write and submit the program ***main4\_2.m*** that loads the train and test set, learns the weights for the training set, and computes the mean squared error of your predictor on both the training and testing data set. See rules for submission of programs on the course webpage.

>> main4\_2

(d) in your report please list the resulting weights, and both mean square errors. Compare the errors for the training and testing set. Which one is better?

W = [w0, w1, w2, …, w13] =

[

39.5843

-0.1011

0.0459

-0.0027

3.0720

-17.2254

3.7113

0.0072

-1.5990

0.3736

-0.0158

-1.0242

0.0097

-0.5860

]

MSR\_train = 22.0813

MSR\_test = 22.6383

Testing error is higher

**Part 3. Online gradient descent**

The linear regression model can be also learned using the gradient descent method.

One concern when using the gradient descent method is that it may become unstable when fed with unnormalized data.

To deal with the issue you are given two matlab functions: *compute\_norm\_parameters* and *normalize* that are able to respectively calculate the normalization coefficients from the data matrix and apply them to un-normalized data matrix.

(a) Implement and submit an online gradient descent procedure for finding the regression coefficients **w**. Your program should:

1. start with zero weights (all weights set to 0 at the beginning);
2. update weights using the annealed learning rate 2/t, where t denotes the t-th update step. Thus, for the first data point the learning rate is 2, for the second it is 2/2 = 1, for the 3-rd is 2/3 and so on;
3. repeat the update procedure for 1000 steps reusing the examples in the training data if necessary (hint: the index of the i-th example in the training set of size n can be obtained by (i mod n) operation);
4. return the final set of weights.

(b) Write and submit a program ***main4­\_3.m*** that runs the gradient procedure on the data

and at the end prints the mean test and train errors. Please note that your program

should normalize the x (input) part of the data before running the method.

Run your program and report the results. Give the mean errors for both the training and test set. Is the result better or worse than the one obtained by solving the regression problem exactly.

MSR of online gradient decent:

MSR\_train = 39.8159

MSR\_test = 53.1975

The result is worse than the previous one obtained by solving the regression problem exactly

(c) Try to run your ***main4\_3.m*** program from part b. without data normalization, that in

on the original un-normalized data. Please report on what you observed?

Without normalization, the derivative (dw) become very large even at the step 10, which results also large W.

W keeps becoming larger and larger. The derivative (dw) becomes inf number, which results W start to report NaN.

(d) Modify ***main4\_3.m*** from part c. such that it lets you to progressively observe changes

in the mean train and test errors during the execution of the gradient descend procedure.

Use functions init progress graph and add to progress graph on the course web page. The

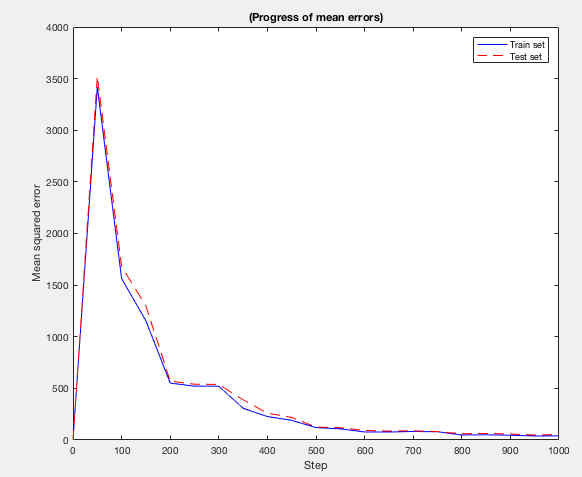
init progress graph initializes the graph structure and add to progress graph lets you add

new data entries on-fly to the graph. Using the two functions plot the mean squared errors

for the training and test test for every 50 iteration steps.

Submit the program and include the graph in the report.

Modified main4\_3.m



(d) Experiment with the gradient descent procedure. Try to use: fixed learning rate (say

0.05, 0.01), or different number of update steps (say 500 and 3000). You may want to

change the learning rate schedule as well.

Try for example 2/sqrt(n). Report your results and any interesting behaviors you observe.

Keep the learning rate still as annealed learning rate 2/t:

Adjust number of iterations:

Steps= 500, the algorithm cannot converge well

MSR\_train\_500interation = 120.3173

MSR\_test\_500interation = 125.2429

Steps=3000, much better results

MSR\_train\_3000interation = 11.5197

MSR\_test\_3000interation = 15.5116

Adjust the learning rate to keep it = 0.05 and 0.01

Steps = 1500:

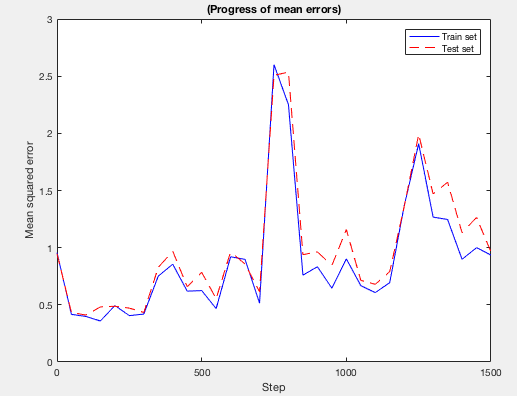
MSR\_train\_0.01 = 0.3986

MSR\_test\_0.01 = 0.4543

MSR\_train\_0.05 = 0.9351

MSR\_test\_0.05 = 0.9614

Also test with the learning rate 0.1, MSR turn out to become very large and unstable



Adjust the learning rate to 2/sqrt(n), the performance also not stable, worse than learning rate 2/t.

Conclusion on the observation:

Learning rate impact the performance of the algorithm very much.

Compare with the stable small learning rate (0.01, 0.05), annealed style learning rate starting with a large learning rate seems not very suitable for this dataset. Started large learning rate will somehow push the w into a place that hard to converge back.

**Part 4. Regression with polynomials.**

Assume we are not happy with the predictive accuracy of the linear model and we decided

to explore a more complex model for predicting housing values.

Assume we decided to use a quadratic polynomial to model the relation between y and x:



(a) Write a function *extendx* that takes an input x and returns an expanded x that includes all linear and degree two polynomials.

>> *extend.m*

(b) What happened to the binary attribute after the transformation?

Still binary attribute

(c) Write and submit a Matlab program ***main4\_4.m*** that computes the regression coefficients for the extended input and both the train and test errors for the result.

>> main4\_4.m

Use the learning rate = 0.01, iterate 3000 times

MSR\_train = 0.2521

MSR\_test = 0.2911

>> W

W =

Columns 1 through 8

0.0137 -0.1355 -0.0992 -0.2671 -0.0105 -0.0924 -0.0496 -0.0308

Columns 9 through 16

-0.3435 0.0568 -0.1704 -0.1013 0.2133 -0.8043 0.0421 0.1102

Columns 17 through 24

0.2588 -0.0105 -0.1324 0.5186 0.0535 0.1340 0.0560 -0.0071

Columns 25 through 27

-0.0703 -0.1573 0.4595