

Problem assignment 3

Due: Wednesday, February 12, 2003

Regression.

In this problem set we use the Bouston 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 there in the data set? List the attributes.
- (b) Calculate and report the correlations 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 target attribute?
- (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.
- (d) Calculate all correlations between the 14 columns (using the `corrcoef` function). Which two attributes have the largest mutual correlation in the dataset?

Part 2. Linear regression.

Our goal in this problem is to predict the median value of housing based on the values of 13 attributes. For the 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 to use a linear regression model to predict the target attribute. Using Matlab:

- (a) Write a function *LR_solve* that takes X components of the data and Y component of the data and returns a vector of coefficients with the least square fit. (Hint: you can use backslash operator '/' to do the least squares regression directly; check Matlab's help).
- (b) Write a function *LR_predict* that takes input components of the data (\mathbf{X}) and a fixed set of weights (\mathbf{w}), and computes vector of linear predictions \mathbf{y} .
- (c) Write and submit program main2.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.
- (d) in your report list the resulting weights, both mean square errors. Compare the errors for the training and testing set. Which one is better?

Part 3. Gradient descent

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

(a) Implement a gradient descent procedure for finding the regression coefficients. Your program should:

- start with zero weights (all weights set to 0 at the beginning);
- 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;
- repeat the update procedure for 1000 steps reusing the data in the training set if necessary;
- return the final set of weights.

(b) Write a program main3.m that runs the gradient procedure on the **normalized training data** and at the end prints the mean test and train errors. Run it 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 exactly.

(c). Try to run the gradient descent on unnormalized data. What happened?

(d) Modify `main3.m` from part b. such that it lets you to progressively observe changes in the mean train and test errors. 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.

(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 many want to change the learning rate schedule as well. Try for example $2/\sqrt{n}$. Report your results and any interesting behaviors you observe.

Part 4. Regression with polynomials.

Assume we are not happy with the predictive accuracy of the linear model and you decide to explore a more complex model for predicting housing values. Assume we decide to use a quadratic polynomial to model the relation between y and \mathbf{x} :

$$f(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{i=1}^{13} w_i x_i + \sum_{i=1}^{13} \sum_{j=i}^{13} w_{ij} x_i x_j.$$

- (a) Write a function `extendx` that takes an input \mathbf{x} and returns an extended \mathbf{x} that includes all linear and degree two polynomials.
- (b) What happened to the binary attribute after the transformation?
- (c) Write and submit a Matlab program that computes the regression coefficients for the extended input and both train and test errors for the result.
- (d) Report both errors in your report and compare them with the results in part 2. What do you see? Which method would you use for the prediction? Why? Please do not turn in the weights for this part in your report.