Boston University
Department of Electrical and Computer Engineering

ENG EC 500 B1 (Ishwar) Introduction to Learning from Data

Matlab Exercise 3

Issued: Thu 5 Nov 2015 **Due:** 9am Wed 18 Nov 2015 PHO440 box + Blackboard
Required reading: Your notes from lectures and additional notes on website on classification.

- This homework assignment requires some programming background in MATLAB. Please refer to the following link for an introduction (or review) of MATLAB:
`http://www.math.ucsd.edu/~bdriver/21d-s99/matlab-primer.html`
- The maximum expected run time of code for any problem in this assignment is 10-15 minutes. All problems except 3.2(a) and 3.4(a) make use of built-in matlab functions.
- You will be making two submissions: (1) A paper submission in the box outside PHO440. (2) An electronic submission of all your matlab code to blackboard (in a single zipped file appropriately named as described below).
- **Paper submission:** This must include all plots, figures, tables, numerical values, derivations, explanations (analysis of results and comments), and also printouts of all the matlab .m files that you either created anew or modified. Submit color printouts of figures and plots whenever appropriate. Color printers are available in PHO307 and PHO305. Be sure to annotate figures, plots, and tables appropriately: give them suitable *titles* to describe the content, label the *axes*, indicate *units* for each axis, and use a *legend* to indicate multiple curves in the plots. Please also explain each figure properly in your solution.
- **Blackboard submission:** All the matlab .m files (and only .m files) that you either create anew or modify must be appropriately named and placed into a **single** directory which should be zipped and uploaded into the course website. Your directory must be named as follows: `<yourBUemailID>_matlab2`. For example, if your BU email address is `mary567@bu.edu` you would submit a single directory named: `mary567_matlab2.zip` which contains all the matlab code (and only the code).
- For this assignment, only submit **one** .m file per problem with the following naming convention: `<yourBUemailID>_matlab3_<prb_num>.m`. For example if your id is `charles500` and the .m file is for problem 3.1, then you must submit a file named: `charles500_matlab3_1.m`. Note that the dot . in 3.1 is replaced with an underscore (this is important). *Running the .m file for a problem should generate the results for all parts of that problem, i.e., it should be self-contained.*

Problem 3.1 Exploring Boston Housing Data with Regression Trees: In this problem, we explore estimating housing values in Boston based on a set of 13 features. The following 13 features are used to estimate the output, MEDV (median value of of owner-occupied homes in $1000's): CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT. Details about these features can be found at this website: `https://archive.ics.uci.edu/ml/datasets/Housing`

This data is provided in `housing_data.mat` where training and testing pairs are specified as the variables (`(Xtrain, ytrain)` and `(Xtest, ytest)`). The names of the features and the output are listed in `feature_names` and `output_name` respectively.

(a) Use MATLAB's built-in functions to learn a regression tree using the training data provided. Visualize and submit an image of the regression tree that has a minimum of 20 observations per leaf.
MATLAB functions: `classregtree, fitrtree, eval, test, view`.

(b) For the trained regression tree from part (a) that has a minimum of 20 observations per leaf, what is the estimated MEDV value for the following test feature vector: CRIM = 5, ZN = 18, INDUS = 2.31, CHAS = 1, NOX = 0.5440, RM = 2, AGE = 64, DIS = 3.7, RAD = 1, TAX = 300, PTRATIO = 15, B = 390, LSTAT = 10?

(c) Plot the mean absolute error (MAE) of the training and testing data as a function of the minimum observations per leaf ranging from 1 to 25 (one of many ways to change the sensitivity of a tree). What trend do you notice?

Problem 3.2 Ordinary Least Squares (OLS) versus Robust Linear Regression:

(a) **Implement** ordinary least squares (OLS) linear regression for input (`xData`) and output (`yData`) provided in the `linear_data.mat` file. Do **not** use MATLAB's `robustfit` or `regress`. (i) Will the input data matrix (`xData`) yield a unique solution? Why or why not? (ii) If $h_{\text{OLS}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}_{\text{OLS}} + b_{\text{OLS}}$, report the values of $\mathbf{w}_{\text{OLS}}$ and b_{OLS} and the resulting mean-squared-error (MSE) $\frac{1}{n}\sum_{j=1}^{n} |y_j - h_{\text{OLS}}(\mathbf{x}_j)|^2$ and mean absolute deviation (MAD) $\frac{1}{n}\sum_{j=1}^{n} |y_j - h_{\text{OLS}}(\mathbf{x}_j)|$.

(b) Use MATLAB's `robustfit` function to implement robust linear regression for the input and output in `linear_data.mat`. The function `robustfit` minimizes $\sum_{j=1}^{n} \rho(y_j - \mathbf{w}^\top \mathbf{x}_j - b)$ over $\mathbf{w}, b$, for a specified loss function $\rho(\cdot)$. When using `robustfit`, set the `const` parameter in `robustfit` to 'on' to include the constant term (the default). When `const` is set to 'on', `robustfit` includes 1 as a new component in every feature vector. Run MATLAB's `robustfit` on the data provided for the following loss functions: `'cauchy'`, `'fair'`, `'huber'`, and `'talwar'`. Use the default value for the parameter `tune` in `robustfit`. (i) Compare the MSE and mean absolute deviation (MAD) of OLS (from part(a)) with corresponding values for robust linear regression for different loss functions. (ii) Report the values of $\mathbf{w}_{\texttt{huber}}, b_{\texttt{huber}}$. (iii) Create a plot that compares all these methods, OLS and robust linear regression for all loss functions (show data points, and the fitted lines of all 5 methods). What do you notice?
MATLAB function: `robustfit`

Problem 3.3 Overfitting and Ridge Regression: For this problem use the data provided in `quad_data.mat`. Training data corresponds to `xtrain` and `ytrain`. Testing data corresponds to `xtest` and `ytest`.

(a) Use OLS to fit polynomials of degree d, $d = 1, \ldots, 14$, to the training data (`xtrain` and `ytrain`). For this, use MATLAB's `ridge` function with the vector of ridge parameters (regularization parameters) `k` set to zero in order to reduce ridge regression to OLS. Also set the parameter `'scaled'` to zero in order to ensure that the coefficients estimated are on the same scale as the original data. (i) On one figure, plot the training points, with polynomial curves of degree 2, 6, 10 and 14 overlaid on top. Label accordingly. (ii) On another figure, plot the mean-squared-error (MSE) of the training and testing data as a function of polynomial degree (1 to 14). What do you observe?
Note: Polynomials of degree d in the variable x_j are of the form: $w_0 + \sum_{i=1}^{d} w_i (x_j)^i$, where x_j is the given sample.

(b) By adding l_2 regularization to linear regression (making it ridge regression) the effects of over-fitting can be alleviated. Fix the polynomial degree to 10. (i) Again, on one figure, plot the mean-squared-error (MSE) of the training and testing data as a function of $\log \lambda$ for $\log \lambda$ ranging from -25 to 5 in

increments of 1. (ii) On another figure, plot the testing points along with the original (no-regularization OLS) degree 10 polynomial fit and the l_2-regularized degree 10 fit with that obtained the smallest testing MSE. What do you observe?
MATLAB function: `ridge`.

Problem 3.4 Lasso vs Ridge: In this problem, we explore the trade-offs between l_1-regularization (lasso) and l_2-regularization (ridge) in a multi-dimensional prostate cancer dataset `prostateStnd.mat`. In this dataset, 8 features: lcavol - log(cancer volume), lweight - log(prostate weight), age, lbph - log(benign prostatic hyperplasia amount), svi (seminal vesicle invasion), lcp - log(capsular penetration), gleason (Gleason score), and pgg45 (percentage Gleason scores 4 or 5) are used to estimate lpsa (log (prostate specific antigen)). In the same fashion as the last problem, training and test data are provided: `(Xtrain, ytrain)`, `(Xtest, ytest)`. The first 8 features correspond to the first 8 entries of `names`. The last and ninth entry of `names` is the output in `(ytest, ytrain)`.

(a) **Implement** cyclic coordinate descent for lasso (shooting algorithm). Update coefficients at most 100 times or until coefficients have converged (no change). Do **not** use MATLAB's `lasso`. (i) Plot the lasso coefficient of each feature (8 total – yielding 8 curves) as a function of $\log\lambda$ for $\log\lambda$ ranging from -5 to 10 in steps of 1. Label the plot accordingly. (ii) In another figure, plot the mean-squared-error (MSE) of both the training and the test data as a function of $\log\lambda$.

(b) What happens to the lasso coefficients as λ gets larger? What are the 2 most meaningful features with lasso (last 2 features to converge)? How can this information be used?

(c) Use MATLAB's ridge regression on this dataset. (i) Again, plot the ridge coefficients of each feature (8 total – yielding 8 curves) against $\log\lambda$ for values of $\log\lambda$ ranging from -5 to 10 in steps of 1. Label the plot accordingly. (ii) In another figure, plot the mean-squared-error (MSE) of both the training and testing data.

(d) What happens to the ridge coefficients as λ becomes larger? How is this different from lasso?