CS589: Machine Learning - Spring 2019

Homework 1: Regression

Assigned: Thursday, Feb 7. Due: Thursday, Feb 21 at 11:55pm

Getting Started: In this assignment, you will train and evaluate different regression models on two datasets. Please install `Python 3.6` via `Anaconda` on your personal machine. For this homework (and most likely for this course) you will only be using `numpy`, `scipy`, `sklearn` and `matplotlib` packages. Download the homework file `HW01.zip` via `Moodle`. Unzipping this folder will create the directory structure shown below,

```
HW01
--- Data
    |--AirFoil
    |--AirQuality
--- Submission
    |--Code
    |--Figures
    |--Predictions
       |--AirFoil
       |--AirQuality
    |--Report
```

The data files for each data set are in `Data` directory respectively. You will write your code under the `Submission/Code` directory. Make sure to put the deliverables (explained below) into the respective directories.

Deliverables: This assignment has three types of deliverables: a report, code files, and Kaggle submissions.

- **Report:** The solution report will give your answers to the homework questions (listed below). Try to keep the maximum length of the report to 5 pages in 11 point font, including all figures and tables. Reports longer than five pages will only be graded up until the first five pages. You can use any software to create your report, but your report must be submitted in PDF format.

- **Code:** The second deliverable is the code that you wrote to answer the questions, which will involve implementing a regression models. Your code must be `Python 3.6` (no iPython notebooks or other formats). You may create any additional source files to structure your code. However, you should aim to write your code so that it is possible to re-produce all of your experimental results exactly by running `python run_me.py` file from the `Submissions/Code` directory.

- **Kaggle Submissions:** We will use `Kaggle`, a machine learning competition service, to evaluate the performance of your regression models. You will need to **register** on `Kaggle` using a `umass.edu` email address to submit to Kaggle (you can use any user name you like). You will generate test prediction files, save them in Kaggle format (helper code provided called `Code/kaggle.py`) and upload them to Kaggle for scoring. Your scores will be shown on the Kaggle leaderboard, and **12%**

of your assignment grade will be based on how well you do in these competitions. The Kaggle links for each data set are given under respective questions.

Submitting Deliverables: When you complete the assignment, you will upload your report and your code using the `Gradescope.com` service. Place your final code in `Submission/Code`, and the Kaggle prediction files for your best-performing submission only for each data set in Submission/Predictions/<Data Set>/best.csv. If you used Python to generate report figures, place them in Submission/Figures. Finally, create a **zip** file of your submission directory, `Submission.zip` (NO rar, tar or other formats). Upload this *single zip file* on Gradescope as your solution to the `HW01-Regression-Programming` assignment. Gradescope will run checks to determine if your submission contains the required files in the correct locations. Finally, upload your pdf report to the `HW01-Regression-Report` assignment. When you upload your report please make sure to select the correct pages for each question respectively. Failure to select the correct pages will result in point deductions. The submission time for your assignment is considered to be the later of the submission timestamps of your code, report and Kaggle submissions. *Some students have requested to be able to submit an IPython notebook to save time. This is permissible if you just export the notebook to a .pdf and upload that .pdf to gradescope. However, the same rules apply as if you prepared the report using any other writing system! In particular, your code should not be in the notebook itself (other than perhaps a tiny command like plot_answer_2b() before each plot.*

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Sharing your solutions with other students is considered cheating. Posting your code to public repositories like GitHub is also considered cheating. Any detected cheating will result in a grade of -100% on the assignment for all students involved, and potentially a grade of F in the course.

Task:
Regression: The regression task consists on finding a model that, for every input, outputs a value. While for classification tasks this output is one of several possible classes, for regression problems it is a real number. An example is shown in Fig. 1 [1], in which the blue dots are the data, and the red line is the learned model used to predict, given an input value (x axis) the corresponding output value (y axis).

Model selection: For each trained model you will try different parameters. Which set of parameters provides the best performance? Is the one that provides a lower training error? Not necessarily, lower training error does not mean lower testing error (or better generalization). For this homework we will be using mean absolute error (MAE). One way to estimate the best model is via model selection. In other words, after training a model one would like to know how well will that model generalize, *i.e.* how well will the model perform on new unseen data. This cannot be known, but can be estimated via cross-validation. This consists of splitting the training data into K pieces, training using a subset of $K-1$ pieces, and evaluating the final performance on the unused piece. This is repeated K times, leaving out for testing one different piece each time. The average of the accuracy obtained in this K simulations can be used as an estimation for the out-of-sample error.

Note: You cannot use sklearn's inbuilt gridsearchcv function. One of the goals of this assignment is for you to write one yourself. You are allowed to use Kfold function from sklearn, We will perform checks and points will be deducted accordingly.

[1] Image taken from https://en.wikipedia.org/wiki/Regression_analysis

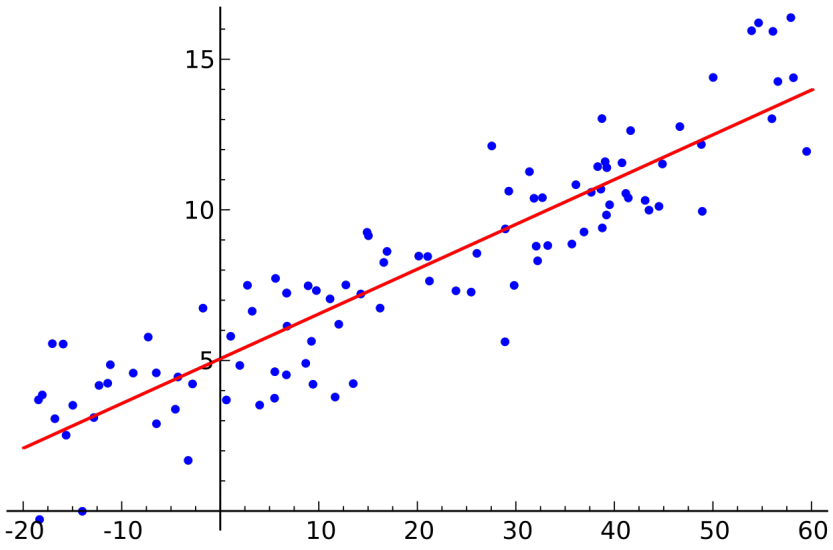

Figure 1: Example of linear regression.

Models: You will train decision trees with different maximum depths, nearest neighbors with different number of neighbors, and linear models with different regularization parameters.

Data Sets: In this assignment, you will experiment with two different datasets: AirFoil and AirQuality. The basic properties of these data sets are shown below. Additional details are given in the `README.txt` files contained in each data set directory.

Dataset	Training Cases	Test Cases	Dimensionality
AirFoil	751	752	5
AirQuality	5614	3743	7

Each data set has been split into a training set and a test set and stored in NumPy binary format. The provided `Submission/Code/run_me.py` file provides example code for reading in all data sets. The following are the required Kaggle links to the respective competitions:

- `https://inclass.kaggle.com/c/hw1-air-foil`
- `https://inclass.kaggle.com/c/hw1-air-quality`

Questions:

1. (*26 points*) Decision trees:

a. (*6 pts*) What is the criteria used to select a variable for a node when training a decision tree? Is it optimal? If yes, explain why it is optimal. If no, explain why is the optimal ordering not used.

b. (*10 pts*) For the air foil dataset train 5 different decision trees using the following maximum depths {3, 6, 9, 12, 15}. Using 5-fold cross-validation, estimate the output of sample error for each model, and report them using a table. Measure the time (in milliseconds) that it takes to perform cross-validation with each model and report the results using a graph (make sure to label both axis; points will be deducted for missing labels). Choose the model with lowest estimated out of sample error, train it with the full training set and predict the target outputs for the samples in the test set. Following this Kaggleize your predictions (i.e. write them out in Kaggle CSV file), upload your submission to Kaggle to `https://inclass.kaggle.com/c/hw1-air-foil` and report the MAE on the public leaderboard. Is the predicted out of sample error close to the test error? Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

c. (*10 pts*) Repeat the previous question (1.b) with the air quality dataset using the following maximum depths {20, 25, 30, 35, 40}. Use Kaggle url `https://inclass.kaggle.com/c/hw1-air-quality`.

2. (*15 points*) Cross-validation:

a. (*9 pts*) Suppose that training a model on a dataset with K samples takes K units of time, and that you are given a dataset with N samples. In order to perform cross-validation, you split this dataset into chunks of M samples (that is, N/M subsets). Ignoring the time that it takes to evaluate the model, what is the time complexity of performing cross-validation with this partition? Give the answer using *big-O* notation with respect to N and M. What happens if $M = 5$? And when $M = N/2$?

b. (*6 pts*) Can you mention one advantage of making M small?

3. (*16 points*) Nearest neighbors:

a. (*8 pts*) For the air foil dataset train 5 different nearest neighbors regressors using the following number of neighbors {3, 5, 10, 20, 25}. Using 5-fold cross-validation, estimate the out of sample error for each model, and report them using a table. Choose the model with lowest estimated out of sample error, train it with the full training set, predict the outputs for the samples in the test set and report the MAE (follow the steps as question 1 to report MAE). Is the predicted out of sample error close to the real one? Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

b. (*8 pts*) Repeat the previous question with the air quality.

4. (*26 points*) Linear model:

a. (*6 pts*) What is the purpose of penalties (regularization) used in Ridge and Lasso regression?

b. (*10 pts*) Train a Ridge and a Lasso linear model using the air foil dataset with the following regularization constants $\alpha = \{10^{-6}, 10^{-4}, 10^{-2}, 1, 10\}$ for each. Using 5-fold cross-validation, estimate the out of sample error for each model, and report them using a table. Choose the model with lowest estimated out of sample

error (out of the 10 trained models), train it with the full training set, predict the target outputs for the samples in the test set and report the MAE (follow the steps as question 1 to report MAE). Make sure that your report clearly states which model was chosen and what was the predicted out of sample error for it.

c. (*10 pts*) Repeat the previous question with the air quality dataset for $\alpha = \{10^{-4}, 10^{-2}, 1, 10\}$ (8 models only).

5. (*12 points*) Kaggle Competition:

a. (*6 pts*) Train a regression model of your choice from either decision trees, nearest neighbors or linear models on the air foil dataset. Pick ranges of hyperparameters that you would like to experiment with (depth for decision trees, number of neighbors for nearest neighbors and regularization constants for linear models). Also pick k in k-fold cross-validation used to tune hyperparameters. Your task is to make predictions on the test set, kagglize your output and submit to kaggle public leadership score (limited to ten submissions per day). Make sure to list your choice of regression model, hyperparameter range, k in k-folds, your final hyperparameter values from cross-validation and best MAE. Save the predictions associated to the best MAE under `Submissions/Predictions/<Data set>/best.csv`. Kaggle submission should be made to `https://inclass.kaggle.com/c/hw1-air-foil`.

b. (*6 pts*) Repeat the previous question with the air quality. Kaggle submission should be made to `https://inclass.kaggle.com/c/hw1-air-quality`.

6. (*5 points*) Code Quality:

a. (*5 pts*) Your code should be sufficiently documented and commented that someone else (in particular, the TAs and graders) can easily understand what each method is doing. Adherence to a particular Python style guide is not required, but if you need a refresher on what well-structured Python should look like, see the Google Python Style Guide: `https://google.github.io/styleguide/pyguide.html`. You will be scored on how well documented and structured your code is.