Assignment 2

September 2020

Instructions

- This assignment should be completed individually.
- Do not look at solutions to this assignment or related ones on the Internet.
- The files related to the assignment are present in `lab2-rollno.zip` folder. Extract it and upload it on moodle in the same .zip format after completion and after replacing the string "rollno" with your actual roll number. For example, if you roll number is 00405036, then single zip folder that you will upload will be named "lab2-00405036.zip". Also collate all the CS337 based theory solutions into ONE pdf file named `answers.pdf`. Include `answers.pdf` inside the zip folder mentioned above and submit the zip folder.
- Answers to all subjective questions need to be placed in single pdf `answers.pdf` including all plots and figures and uploaded.
- Only add/modify code between `TODO` and `END TODO` unless specified otherwise
- Python files to submit - `p1.py`,`utils.py`, `p2.py` and `p3.py`
- This Assignment carries a total of 4 marks for CS337 Theory and 11 marks for CS335 Lab

1 LASSO and ISTA

Lasso regression uses the ℓ_1 penalty term and stands for Least Absolute Shrinkage and Selection Operator. It is a widely used tool for achieving sparse solutions to optimization problems. The penalty applied for ℓ_1 is equal to the absolute value of the magnitude of the coefficients.

$$L(w) = \sum_{i=1}^{n} \|y_i - x_i.w\|_2^2 + \lambda \|w\|_1^1$$

In matric form (using matrix X here instead of Φ as used in the class, LASSO optimizes

$$\min_w \|y - Xw\|_2^2 + \lambda \|w\|_1$$

Similar to ridge regression, a lambda value of zero results in the basic OLS equation, however given a suitable lambda value, lasso regression can drive some coefficients to zero. The larger the value of lambda the more features are shrunk to zero. This can help eliminate some features entirely and give us a subset of predictors that helps mitigate multi-collinearity and model complexity. Predictors not shrunk towards zero signify that they are important and thus ℓ_1 regularization allows for feature selection (sparse selection).

In this problem, we will implement the Iterative Soft-Thresholding Algorithm (ISTA) for LASSO.

1.1 CS 337: Theoretical Question

Prove that the solution to LASSO is the MAP estimate of Linear regression subject to the Laplacian prior on weights. (1.5 marks)

1.2 CS 335: Lab Questions

(a) Complete the function `ista()` in the file `p1.py`. The file `utils.py` is same as that of previous assignment and you may reuse the code written before. Make sure you write an efficient vectorized implementation (5000 iterations should take less than 10 seconds). You have to stop the algorithm on convergence, *i.e.*, when the norm of the difference between the weights of successive iterations falls below $\epsilon = 10^{-4}$. You may change the values of λ (the learning rate) and the maximum number of iterations if needed. (3 marks)

(b) Plot the train and test MSEs for at least 15 different values of λ ranging from 0.1 to 6, with learning rate 0.001 and maximum iterations 10,000, in a single figure. Explain the plot. What value of λ is optimal according to you? Why? (1.5 marks)

(c) Create a scatter plot of the weights obtained from both LASSO and Ridge regressions in separate figures and compare them. You can use code and the value of λ for Ridge from previous assignment. For LASSO, use the optimal λ found in previous part and run the algorithm for enough number of iterations to achieve convergence. Briefly justify your observations. (1.5 marks)

2 Multi-class Classification using 1 vs rest Perceptron

A Perceptron keeps a weight vector w_y corresponding to each class y. Given a feature vector f, we score each class y with

$$score(f, y) = \sum_i f_i w_{y_i}$$

where i iterates over the dimensions in the data. Then we choose the class with the highest score as the predicted label for data instance f.

Learning the weights of the Perceptron

In the basic multi-class Perceptron, we scan the entire training data, one instance at a time. When we come to an instance (f, y), we find the label with the highest score: $y' = \arg\max_{y''} score(f, y'')$, while breaking ties arbitrarily. We compare y' to the true label y. If $y' = y$, we have correctly classified the instance, and we do nothing. Otherwise, we have wrongly predicted y' instead of the true label y. This means that w_y has scored f lower, and/or $w_{y'}$ has scored f higher, than what would have been ideally desirable. To avert this error in the future, we update these two weight vectors accordingly: $w_y = w_y + f$ and $w_{y'} = w_{y'} - f$

2.1 CS 337: Theoretical Problem

An alternative to the 1-vs-rest perceptron defined above is to have a 1-vs-1 perceptron, one for each pair of classes. Compare this proposed 1-vs-1 approach to the previously described 1-vs-rest approach for multi-class classification. Outline and briefly justify the advantages and disadvantages of each. (1 mark)

2.2 CS 335: Lab question

Complete the `pred` and `train` functions in `p2.py`. You can modify `max_iter` if needed. (3 marks)

3 Bias Variance Trade-off

3.1 CS 337: Theoretical Questions

Indicate the effect of each of the following on the bias and/or variance of the learned model along with an informal reasoning. (1.5 marks)

(a) Increasing the value of λ in lasso regression.

(b) Adding a higher number of training examples for a perceptron

(c) Adding more features in the lasso regression task such that the new features are some linear function of the original features. You can assume that Lasso is being solved using the ISTA algorithm.

3.2 CS 335: Lab Questions

Using the code you developed in task 2, compute the test set error for the perceptron using $10, 100, 1000, 10000, 50000, 80000$ samples (This can be achieved by using a command line option while running `p2.py`). Make sure you keep the test set fixed. Plot the test set error vs number of train samples, and report your observations in terms of the bias and variance. Also attach the plot in your `answers.pdf` file. (0.5 marks)

Complete the function `prepare_data(X,degree)` in the file `p3.py` to prepare data for a higher order polynomial regression, following the instructions in the function. We then fit 4 linear regression models on this augmented data, sampling 40 points at a time from the dataset. Theoretically, for what value of `degree` do you obtain the least train error on each run? Plot graphs of the test error for `degree` between 1 and 6, and between train error and `degree` and for the 4 different train sample subsets and explain your observations in terms of the bias variance trade-off and overfitting. You may use the graph generated by running `p3.py` to develop your explanations. The degree of the polynomial fitted can be controlled with a command line parameter. Add the plots of error vs degree to your report as well. (1.5 marks)