Assignment #2: Penalized Regression Methods STA9890 Statistical Learning for Data Mining

Assignment Parameters:
Date Assigned: 2024-02-20
Date Due: 2024-03-05 @ 5:45pm

Submission Mechanism(s):

- Blackboard (strongly preferred)
- Email to instructor: michael.weylandt@baruch.cuny.edu
  Email submissions must be titled *exactly* as `STA9890-S2024-HW2-LASTNAME,FIRSTNAME.pdf`

Question 0: Recommended Reading (Ungraded)

Recommended Reading:

- ISLR: Chapters 3 and 6
- ESL: Section 3 - 3.4

Please contact me if you need additional background reading recommendations.

Question 1: Data Analysis Pipeline (15 points)

This is an *issue spotter* question. Below, I describe (in words) a hypothetical application of the ML techniques we have discussed in this class. Your task is to find *at least* 3 mistakes in the pipeline and to i) describe the problem (2 points each) and ii) say how that stage should have been performed (3 points).

This problem comes with two opportunities for extra credit:

- Additional issues: for any additional issues beyond the three you are required identify, extra credit will be given (up to 5 points per issue).

  Please mark the three mistakes you are most sure about as your 'main answers' so I can distinguish them from your EC attempts.

  To dissuade random guessing, extra credit will only be given if all guesses are at least partially correct. (*I.e.*, if you guess 25 things, 20 of which are completely wrong, you can still get full credit, but you won't get extra credit.)

- Simulations to demonstrate the effect of mistakes in the ML pipeline.

  For each of your three 'main answers', you can provide a simulation that demonstrates the effect of that mistake on the conclusions of the ML pipeline. Each simulation is worth up to 5 points depending on how well it is implemented.

Scenario:

> Bernard wants to use a ML pipeline to improve the credit lending business of his bank; his goal is to use his experience lending to businesses and high-net-worth clients as the basis for new retail division. The bank staff has gone back through the bank's records and collected the following information from previous loans:
>
> - Applicant's Annual Income at Application
> - Applicant's Credit Score at Application
> - Purpose of Loan:
>   - 1: Home Purchase Mortgage
>   - 2: Mortgage Refinance

    – 3: Unsecured Personal Loan
    – 4: Loan to Start and/or Expand a Small Business
- Amount of Loan
- Duration of Loan
- Was the Loan Successfully Paid Off? (Response)

Bernard arranges this data into a 5 column $\boldsymbol{X}$ matrix and uses the 'Was the Loan Successfully Paid Off?' feature as the response variable $\boldsymbol{y}$. He fits an ordinary least squares (OLS) model to this data. To see how well his business would have performed, he computes his profit as if he had made loans to any client with $\hat{y}_i > 0.5$, which turns out to be about 20% of the total number of applications: based on his historical estimates, Bernard predicts that he would make an average of $\$100,000$ in profit for each loan given.

After working with his business development office, Bernard estimates that he will receive 10,000 loans in the first year of his retail business. At 20% approval rate, he expects to make 2,000 loans over the next year, for a forecast profit of two hundred million dollars. Because the retail business is expected to cost one hundred million dollars to start up (new employees, IT investments, advertising, and a new group in the legal and compliance divisions), Bernard predicts 100% ROI in one year to his board; the board ultimately approves the new venture.

Question 2: Non-Negative Least Squares (15 points)

In class, we have considered *penalized* regression methods, which combine the MSE loss with a suitable penalty function. We can also add regularization using *constraints*, the most famous form of which is *non-negative least squares*:

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \text{ such that } \beta_i \geq 0 \text{ for all } i = 1, \dots p$$

In this question, you will explore the properties of NNLS and compare it to OLS, Ridge Regression, and Lasso.

Q2(a). Implement NNLS using `CVXR` or similar software (3 points)

Q2(b). Find conditions on $\hat{\boldsymbol{\beta}}_{\text{OLS}}$ such that $\hat{\boldsymbol{\beta}}_{\text{OLS}} = \hat{\boldsymbol{\beta}}_{\text{NNLS}}$ (2 points)

Q2(c). When can NNLS give a *sparse* solution? (2 points)

Q2(d). Suppose $\boldsymbol{X}$ is a matrix of IID Standard Gaussian elements. Use a simulation to estimate the bias of NNLS when: (5 points)

- All elements of $\boldsymbol{\beta}_*$ are drawn $\beta_i^* \overset{\text{IID}}{\sim} \mathcal{U}([-3, -1])$
- All elements of $\boldsymbol{\beta}_*$ are drawn $\beta_i^* \overset{\text{IID}}{\sim} \mathcal{U}([-0.25, 0.25])$
- All elements of $\boldsymbol{\beta}_*$ are drawn $\beta_i^* \overset{\text{IID}}{\sim} \mathcal{U}([1, 3])$

Q2(e). Compare the variance of OLS and NNLS in these three scenarios. Which one achieves the lower total MSE? (3 points)

Question 3: Variable Selection & Linear Models (20 points)

Download the SRBCT microarray data from the course website. This is a gene expression data set from a childhood cancer study with $n = 83$ patients and $p = 2308$ genes. Your response (outcome) is the gene expression profile of the gene p53, a major oncogene that acts as a tumor suppressor.

Your goal is to select other genes whose expression profiles are associated with p53 so as to find other possible genes to target in drug therapies.

1. Visualize regularization paths for the following methods:
    (a) Elastic net
    (b) Lasso

(c) SCAD

(d) MC+

If you're using `R`, the first two methods are implemented in the `glmnet` package and the second two in the `ncvreg` package.

2. Reflection. Interpret the results. What are the top genes selected by each method? Are they different? If so, why? Which regularization paths look most variable? Why is this the case? If you had to report to a scientist the top 10 genes associated with p53, which ones would you report? Why?

Question 4: Predictive Improvements in Ridge Regression (20 points)

In class, we discussed the *Ridge MSE Existence Theorem*, which roughly states that:

Theorem. Suppose $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ is a fixed matrix and $\boldsymbol{y}$ is generated as $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{\beta}_*, \sigma^2 \boldsymbol{I}_{n \times n})$ for some (unknown) $\boldsymbol{\beta}_* \in \mathbb{R}^p$, $\sigma^2 \in \mathbb{R}_{>0}$. There exists a $\lambda > 0$ such that

$$\mathbb{E}[\|\boldsymbol{\beta}_* - \hat{\boldsymbol{\beta}}_{\text{Ridge}}\|_2^2] < \mathbb{E}[\|\boldsymbol{\beta}_* - \hat{\boldsymbol{\beta}}_{\text{OLS}}\|_2^2]$$

where the expectation is taken over the randomness in $\boldsymbol{y}$ and hence $\hat{\boldsymbol{\beta}}_{\text{OLS}}, \hat{\boldsymbol{\beta}}_{\text{Ridge}}$.

In this problem, you will demonstrate this theorem in *one of two* ways (your choice):

- Simulation: Design a simulation to demonstrate the MSE Existence Theorem.

  Hint: This will be easier if the columns of $\boldsymbol{X}$ *are strongly correlated.*

- Theory: Prove the theorem above. Add any additional assumptions necessary to the statement of the theorem.

  Hints: The RHS of the inequality can be computed in closed form. For the LHS, compute the (squared) bias and the variance separately and add them together. Express $\boldsymbol{X}$ in terms of its SVD and many terms will cancel. You may find it useful to differentiate that quantity with respect to λ.

Question 5: Coordinate Descent Methods for the Lasso (30 points)

In the previous homework, you used a *gradient descent* method to fit OLS and Ridge Regression. Gradient Descent proceeds by taking incremental steps in the direction away from the gradient of the objective function.[1] This works well when our objective function is differentiable, but what can we do for non-differentiable objective functions, *e.g.* Lasso regression? Enter *coordinate descent*!

Coordinate Descent (CD) is similar to gradient descent, but it only updates one coordinate (element of $\boldsymbol{\beta}$) at a time. By looping through all the coordinates many times over, CD eventually converges to the optimal solution. CD methods are often easier to implement than GD because the one-dimensional update problems can be solved in closed form.

Formally, CD methods look something like this:

Algorithm 1 Skeleton of a CD Method

```
• Initialize: $\boldsymbol{\beta}^{(0)} = \mathbf{0}_p$, $k = 0$
• Repeat Until Convergence: ($\|\boldsymbol{\beta}^{(k)} - \boldsymbol{\beta}^{(0)}\| \leq \epsilon$)
    – Copy $\tilde{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(k)}$
    – For $i = 1, \dots p$:
        * $\tilde{\beta}_i = \arg\min_{\beta_i} \mathsf{Objective}(\beta_i)$
    – Set $\boldsymbol{\beta}^{(k+1)} = \tilde{\boldsymbol{\beta}}$
    – Set $k := k + 1$
```

[1] Gradient *ascent* (*i.e.*, following the gradient) maximizes a function, but in our "loss + penalty" framework, we seek to minimize our objectives.

Note that the changes in each element of $\boldsymbol{\beta}$ happen "immediately" so that the new values of $\tilde{\boldsymbol{\beta}}$ are present in the next coordinat update. In this problem, you will implement CD for the lasso regression problem:

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$

Q5(a). Suppose that our current working estimate of $\boldsymbol{\beta}$ is $\boldsymbol{\beta}^{(k)}$. Find the value of β_i that minimizes the objective function, *i.e.*, solve

$$\arg\min_{\beta_i} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 = \frac{1}{2}\left\|\left(\boldsymbol{y} - \sum_{j\neq i}\boldsymbol{x}_j^\top\beta_j\right) - \boldsymbol{x}_i^\top\beta_i\right\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1$$

Hint: Recall the Soft Threshold function we discussed in class (5 points)

Q5(b). Using your solution to the previous section, implement a basic CD algorithm for the lasso. Demonstrate it on simulated data with a sparse $\boldsymbol{\beta}_*$. (10 points)

Q5(c). Basic CD can be slow, but there are two easy ways to speed it up. Implement these both to give a CD function that computes the lasso solution for a grid of λ values: (10 points)

- "Active Set": Most of the time, the variables selected by Lasso do not change (or, if they do change, only one or two new variables are selected). CD can proceed by updating only the selected (non-zero) elements of $\tilde{\boldsymbol{\beta}}$ and then performing a 'full loop' (over all elements $\boldsymbol{\beta}$) when you seem to have converged.
- "Warm Starts": We typically want to solve the lasso problem for many λ values all at once. Because the lasso solution $\hat{\boldsymbol{\beta}}_{\text{Lasso}}(\lambda)$ varies slowly with λ, you can speed CD up by starting at the value of $\boldsymbol{\beta}$ from the previous λ instead of starting at $\mathbf{0}$. Use this warm-start technique to solve the lasso problem at a grid of 100 values: you can use the following code to generate a suitable default grid:

```
LAMBDA_MAX <- max(abs(crossprod(X, y)))

LAMBDA_GRID <- seq(0.001 * LAMBDA_MAX, LAMBDA_MAX, length.out=100)
```

Compare how long it takes your efficient CD to compute the lasso path as compared to basic CD. (It should be faster!)

Q5(d). Compare the CD results to `CVXR` on the data from Q3 and show that your results are (nearly) equal (5 points)