Programming Assignment I

Shaun Pritchard

Florida Atlantic University

CAP 5625

October -18-2021

M.DeGiorgio

Programming Assignment I

The scope of this project was to build ridge regression with batch gradient descent algorithm from scratch while implementing a grid based search 5K-fold Cross Validation to test the effect of the tuning parameters on the inferred regression coefficients. For this assignment I implemented the vectorize algorithm approach as follows:

Algorithm 1 (vectorized):

Step 1. Choose learning rate α and fix tuning parameter λ

Step 2. Generate N-dimensional centered response vector $\mathbf{y}$ and $N \times p$ standardized

(centered and scaled to have unit standard deviation) design matrix $\mathbf{X}$

Step 3. Randomly initialize the parameter vector β = [β1, β2, … , βp]

Step 4. Update the parameter vector as

$\beta := \beta - 2\alpha[\lambda\beta - \mathbf{X}T(\mathbf{y} - \mathbf{X}\beta)]$

Step 5. Repeat Step 4 for 105 iterations or until convergence (vector β does not change) Step 6. Set the last updated parameter vector as β = [β1, β2, … , βp]

Deliverables:

This project required six deliverables produced from the algorithms I created using the python programming language. The deliverables are as follows.

- Deliverable 1: build graph of dataset N=9 features tuning parameter effect on inferred Ridge regression
- Deliverable 2: Illustrate the effect of the tuning parameter on the cross-validation error
- Deliverable 3: Indicate the value of λthat generated the smallest CV(5)error
- Deliverable 4: retrain the model of N=400 observations and provide the estimates of the p=9 best-fit model parameters.
- Deliverable 5 Provide all your source code and instructions

- ❖ Deliverable 6 Implement the assignment using statistical or machine learning libraries in a language of your choice and compare all the results.

Instructions:

The algorithms in this assignment were created using python Google Colaboratory Jupyter notebooks. I created two separate notebooks, one for deliverables one through five and another notebook for deliverables 6. This report will show the comparisons of the algorithms I developed from scratch with the python scikit learn library implementations.

The contents of this assignment including this report and as per deliverables required and produced are as follows: separate Jupyter notebook files (which can be ran with interpreter online or offline), a PDF print out for each containing the working code, results, and testing output, and separate raw python (.py) script files. Also, output image files for deliverables 1 and 2 which are not required are included for good measure.

All files are included and are also accessible online for viewing and commenting at the following URLs:

From scratch implementation:

https://colab.research.google.com/drive/13lsR-yuHv1YKGIXqCLHilKlwUVZcunp0?usp=sharing

Using liberties implementation:

https://colab.research.google.com/drive/1W0aaP4C2_QJo4NTQ-mrODOepyaWxnQa-?usp=sharing

Note: For using standard python scripts on a local machine make sure the required imports are installed using pip, pip3, or anaconda. Also, rename the file-path if importing the dataset from another location other than the original implementation.

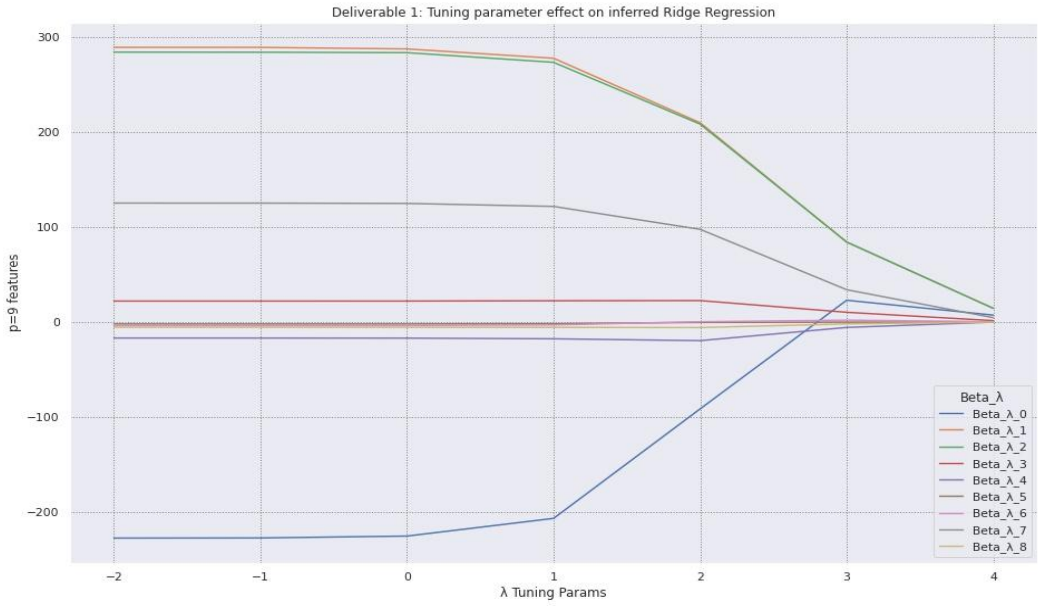
Files and directories:

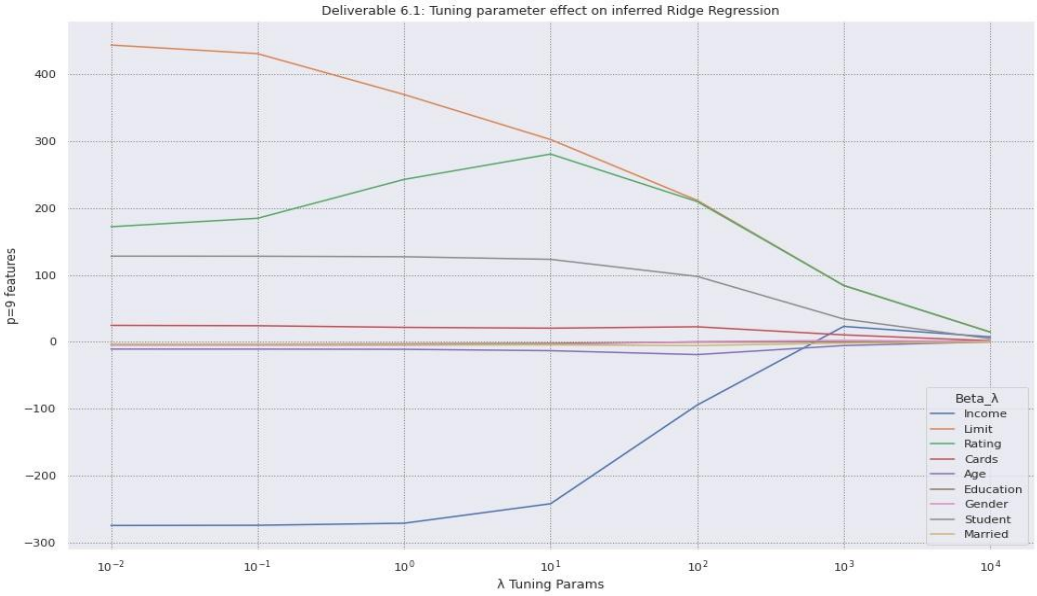
- Assignment Instructions := instructions as per assignment for CAP 5625
- Dataset := contains the working dataset for this project

- From Libraries Implementation directory := PDF of code and results, ipynb file of code, .py script file of code, and output images for deliverables specific to the code used with libraries only.
- From Scratch Implementation directory := PDF of code and results, ipynb file of code, .py script file of code, and output images for deliverables specific to the code project from scratch only
- Instructions and comparison file : = this file in word and pdf format

From scratch vs. libraries for ridge regression:

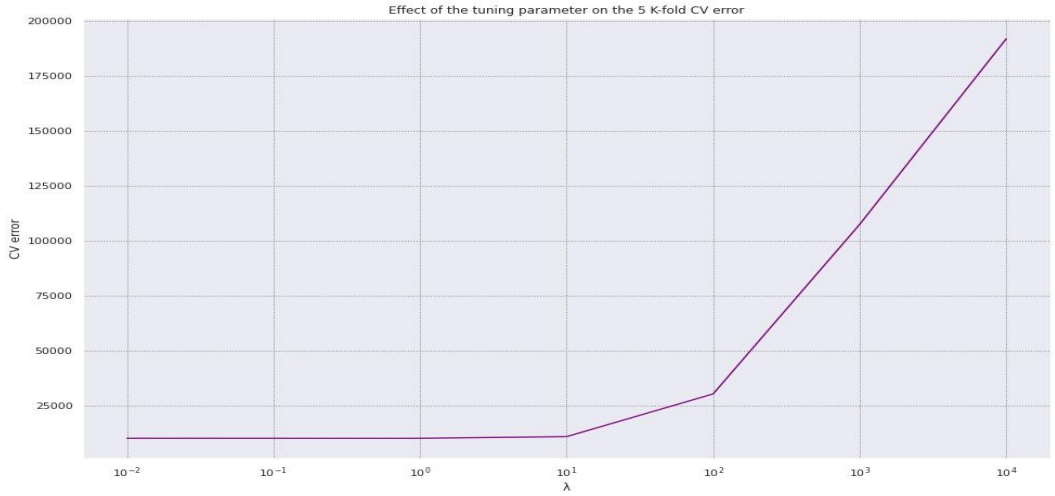
Using basic python numpy and pandas libraries as (D1), I created an algorithm from scratch. Scikit-Learn (D2) was used to implement the comparison for deliverable 6. As a first step, we compute the initial ridge regression algorithm with batch gradient descent. The results of these two algorithms were comparable as seen in the output for the λ tuning parameters As follows:

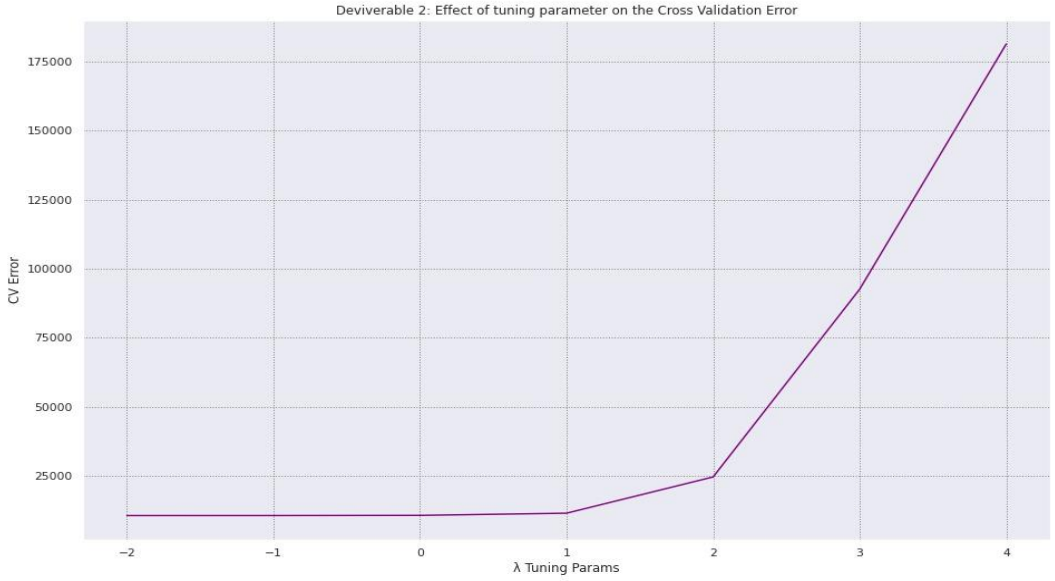




One major difference was found with the convergence: the third feature of the D2 algorithm, labeled "Ratings," which compared to the D1 algorithm I created from scratch seemed to compute more precision on the actual tuning parameter errors. The results show minor variations in convergence overall.

In addition, we observe a similar result with respect to the tuning parameter eros on the ridge regression algorithm with 5 k-fold cross-validation. The visual results were almost identical.





In addition, when comparing the best MSE score and best tuning parameters score, the results from the algorithm built from scratch and the scikit learn library algorithm were quite different. Based on the results of the D1 algorithm, the best MSE error value was 159943.40 and the sampling parameter was 10.0, however, scikit-learn D2 algorithm produced -10080.82 and the sampling parameter was 1.0.

Scikit-learn D2 algorithm results displayed MSE error value of -10080.82 and λ tuning parameter of 1.0, which is an improvement in performance over the D1 algorithm created from scratch.

D1 Algorithm results:

```
Best CV error of λ = 159943.40640125488
Best tuning param of λ = 10.0
```

D2 Algorithm results (scikit learn):

```
Best CV error of λ -10080.822276681745
Best tuning params of λ {'alpha': 1.0}
```

Finally the best fit model parameters showed variation for both algorithms are as follows.

D1 Algorithm results:

```
Best fit model parameters [[-206.56414602]
 [ 277.74769578]
 [ 273.0386694 ]
 [  22.40550326]
 [ -17.62639158]
 [  -1.74306883]
 [  -3.03950781]
 [ 121.65485588]
 [  -5.61543757]]
```

D2 Algorithm results (scikit learn):

```
array([-271.23122766,  369.52441617,  242.60689635,   21.49206153,
        -11.27075014,   -3.04525924,   -5.09410044,  127.07827162,
         -3.97552834])
```

The two results were very close, but it appeared that the scikit-learn algorithm performed better overall. In addition, I decided to compute a second instance of the cross validation errors just to see how it compared. Based on the absolute value of ridge regression and cross validation error results similar to LASSO regression, the observations confirm the cross validation error of -80.761 and tuning parameter of 0.01. The approximation results were significantly different.

λ that generated the smallest CV(5)error
CV Error: -80.761
Best tuning params of λ : {'alpha': 0.01}