CAP 5625: Programming Assignment 2

Due on Canvas by Wednesday, November 3, 2021 at 11:59pm

Preliminary instructions

You may consult with other students currently taking CAP 5625 at FAU on this programming assignment. If you do consult with others, then you must indicate this by providing their names with your submitted assignment. However, all analyses must be performed independently, all source code must be written independently, and all students must turn in their own independent assignment.

> Though it should be unnecessary to state in a graduate class, I am reminding you that you may **not** turn in code (partial or complete) that is written or inspired by others, including code from other students, websites, past code that I release from prior assignments in this class or from past semesters in other classes I teach, or any other source that would constitute an academic integrity violation. All instances of academic integrity violations will receive a zero on the assignment and will be referred to the Department Chair and College Dean for further administrative action.

You may choose to use whatever programming language you want. However, you must provide clear instructions on how to compile and/or run your source code. I recommend using a modern language, such as Python, R, or Matlab as learning these languages can help you if you were to enter the machine learning or artificial intelligence field in the future.

All analyses performed and algorithms run must be written from scratch. That is, you may not use a library that can perform coordinate descent, cross validation, elastic net, least squares regression, optimization, etc. to successfully complete this programing assignment (though you may reuse your relevant code from Programming Assignment 1). The goal of this assignment is not to learn how to use particular libraries of a language, but it is to instead understand how key methods in statistical machine learning are implemented. With that stated, I will provide 10% extra credit if you additionally implement the assignment using built-in statistical or machine learning libraries (see Deliverable 6 at end of the document).

Brief overview of assignment

In this assignment you will still be analyzing the same credit card data from $N = 400$ training observations that you examined in Programming Assignment 1. The goal is to fit a model that can predict credit balance based on $p = 9$ features describing an individual, which include an individual's income, credit limit, credit rating, number of credit cards, age, education level, gender, student status, and marriage status. Specifically, you will perform a penalized (regularized) least squares fit of a linear model using elastic net, with the model parameters obtained by coordinate descent. Elastic net will permit you to provide simultaneous parameter shrinkage (tuning parameter $\lambda \geq 0$) and feature selection (tuning parameter $\alpha \in [0,1]$). The two tuning parameters λ and α will be chosen using five-fold cross validation, and the best-fit model parameters will be inferred on the training dataset conditional on an optimal pair of tuning parameters.

Data

Data for these observations are given in `Credit_N400_p9.csv`, with individuals labeled on each row (rows 2 through 401), and input features and response given on the columns (with the first row representing a header for each column). There are six quantitative features, given by columns labeled "Income", "limit", "Rating", "Cards", "Age", and "Education", and three qualitative features with two levels labeled "Gender", "Student", and "Married".

Detailed description of the task

Recall that the task of performing an elastic net fit to training data $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$ is to minimize the cost function

$$J(\beta, \lambda, \alpha) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \left(\alpha \sum_{j=1}^{p} \beta_j^2 + (1 - \alpha) \sum_{j=1}^{p} |\beta_j| \right)$$

where y_i is a centered response and where the input p features are standardized (*i.e.*, centered and divided by their standard deviation). Note that we cannot use gradient descent to minimize this cost function, as the component $\sum_{j=1}^{p} |\beta_j|$ of the penalty is not differentiable. Instead, we use coordinate descent, where we update each parameter k, $k = 1, 2, \dots, p$, in turn, keeping all other parameters constant, and using sub-gradient rather than gradient calculations. To implement this algorithm, depending on whether your chosen language can quickly compute vectorized operations, you may implement coordinate descent using either Algorithm 1 or Algorithm 2 below (choose whichever you are more comfortable implementing). Note that in languages like R, Python, or Matlab, Algorithm 2 (which would be implemented by several nested loops) may be much slower than Algorithm 1. Also note that if you are implementing Algorithm 1 using Python, use numpy arrays instead of Pandas data frames for computational speed. For this assignment, assume that we will reach the minimum of the cost function within a fixed number of steps, with the number of iterations being 1000.

Algorithm 1 (vectorized):

Step 1. Fix tuning parameters λ and α

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. Precompute $b_k, k = 1,2, \dots, p$, as

$$b_k = \sum_{i=1}^{N} x_{ik}^2$$

Step 4. Randomly initialize the parameter vector $\beta = [\beta_1, \beta_2, \dots, \beta_p]$

Step 5. For each k, $k = 1,2, \dots, p$:

compute

$$a_k = \mathrm{x}_k^T (\mathbf{y} - \mathbf{X}\beta + \mathrm{x}_k \beta_k)$$

and set

$$\beta_k = \frac{\mathrm{sign}(a_k)\left(|a_k| - \frac{\lambda(1-\alpha)}{2}\right)_+}{b_k + \lambda\alpha}$$

Step 6. Repeat Step 5 for 1000 iterations or until convergence (vector β does not change)

Step 7. Set the last updated parameter vector as $\hat{\beta} = [\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p]$

```
**Algorithm 2 (non-vectorized):**
**Step 1.** Fix tuning parameters $\lambda$ and $\alpha$
**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.** Precompute $b_k, k = 1,2,\dots,p$, as
        $b_k = \sum_{i=1}^{N} x_{ik}^2$
**Step 4.** Randomly initialize the parameter vector $\beta = [\beta_1, \beta_2, \dots, \beta_p]$
**Step 5.** For each $k$, $k = 1,2,\dots,p$:
            compute
                $a_k = \sum_{i=1}^{N} x_{ik} \left( y_i - \sum_{j=1, j \neq k}^{p} x_{ij}\beta_j \right)$
            and set
                $\beta_k = \frac{\operatorname{sign}(a_k)\left(|a_k| - \frac{\lambda(1-\alpha)}{2}\right)_+}{b_k + \lambda\alpha}$
**Step 6.** Repeat Step 5 for 1000 iterations or until convergence (vector $\beta$ does not change)
**Step 7.** Set the last updated parameter vector as $\hat{\beta} = [\hat{\beta}_1, \hat{\beta}_2, \dots, \hat{\beta}_p]$
```

Note that we define

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \qquad x_+ = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

and we use the notation x_k as the kth column of the design matrix $\mathbf{X}$ (the kth feature vector). This vector by definition is an N-dimensional column vector.

When randomly initializing the parameter vector, I would make sure that the parameters start at small values. A good strategy here may be to randomly initialize each of the β_j, $j = 1,2,\dots,p$, parameters from a uniform distribution between -1 and 1.

Effect of tuning parameter on inferred regression coefficients

You will consider a discrete grid of nine tuning parameter values $\lambda \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$ where the tuning parameter is evaluated across a wide range of values on a log scale, as well as six tuning parameter values $\alpha \in \left\{0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right\}$. For each tuning parameter value pair, you will use coordinate descent to infer the best-fit model. Note that when $\alpha = 0$, we obtain the lasso estimate, and when $\alpha = 1$, we obtain the ridge regression estimate.

Deliverable 1: Illustrate the effect of the tuning parameter on the inferred elastic net regression coefficients by generating six plots (one for each α value) of nine lines (one for each of the $p = 9$ features), with the y-axis as $\hat{\beta}_j$, $j = 1,2,\dots,9$, and the x-axis the corresponding log-scaled tuning parameter value $\log_{10}(\lambda)$ that generated the particular $\hat{\beta}_j$. Label both axes in all six plots. Without the log scaling of the tuning parameter λ, the plots will look distorted.

Choosing the best tuning parameter

You will consider a discrete grid of nine tuning parameter values $\lambda \in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}, 10^{5}, 10^{6}\}$ where the tuning parameter is evaluated across a wide range of values on a log scale, as well as six tuning parameter values $\alpha \in \left\{0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right\}$. For each tuning parameter value pair, perform five-fold cross validation and choose the pair of λ and α values that give the smallest

$$\mathrm{CV}_{(5)} = \frac{1}{5}\sum_{i=1}^{5} \mathrm{MSE}_i$$

where MSE_i is the mean squared error on the validation set of the ith-fold.

Note that during the five-fold cross validation, you will hold out one of the five sets (here 80 observations) as the Validation Set and the remaining four sets (the other 320 observations) will be used as the Training Set. On this Training Set, you will need to center the output and standardize (center and divided by the standard deviation across samples) each feature. These identical values used for centering the output and standardizing the input will need to be applied to the corresponding Validation Set, so that the Validation set is on the same scale. Because the Training Set changes based on which set is held out for validation, each of the five pairs of Training and Validation Sets will have different centering and standardization parameters.

Deliverable 2: Illustrate the effect of the tuning parameters on the cross validation error by generating a plot of six lines (one for each α value) with the y-axis as $\mathrm{CV}_{(5)}$ error, and the x-axis the corresponding log-scaled tuning parameter value $\log_{10}(\lambda)$ that generated the particular $\mathrm{CV}_{(5)}$ error. Label both axes in the plot. Without the log scaling of the tuning parameter λ, the plots will look distorted.

Deliverable 3: Indicate the pair of values λ and α that generated the smallest $\mathrm{CV}_{(5)}$ error.

Deliverable 4: Given the optimal λ and α pair, retrain your model on the entire dataset of $N = 400$ observations and provide the estimates of the $p = 9$ best-fit model parameters. How do these estimates compare to the estimates obtained from ridge regression ($\alpha = 1$ under optimal λ for $\alpha = 1$) and lasso ($\alpha = 0$ under optimal λ for $\alpha = 0$) on the entire dataset of $N = 400$ observations?

Deliverable 5: Provide all your source code that you wrote from scratch to perform all analyses (aside from plotting scripts, which you do not need to turn in) in this assignment, along with instructions on how to compile and run your code.

Deliverable 6 (extra credit): Implement the assignment using statistical or machine learning libraries in a language of your choice. Compare the results with those obtained above, and provide a discussion as to why you believe your results are different if you found them to be different. This is worth up to 10% additional credit, which would allow you to get up to 110% out of 100 for this assignment.