STAT 542/CS 598 Fall 2021

Coding Assignment 2

Due Monday, September 27

Part I: Lasso Implementation (2pt)

Implement Lasso using the Coordinate Descent (CD) algorithm and apply your algorithm on the Boston housing data.

- First, load the transformed Boston Housing Data, `Coding2_myData.csv`.
- Next write your own function `MyLasso` to implement CD, which should output estimated Lasso coefficients similar to the ones returned by `R` with option "`standardized = TRUE`".

  In case you don't know where to start, you can follow the structure given in the Appendix (page 3) to prepare your function. In our script, we run a fixed number of iterations, "`maxit = 100`," which is enough for this assignment. You could set it to be a bigger number, or change it to a `while` loop that stops when some convergence criterion is satisfied.
- Test your algorithm with the following lambda sequence.

```
lam.seq = exp(seq (-1, -8, length.out = 80))
myout = MyLasso(X, y, lam.seq, maxit = 50)
```

- Produce a path plot for the 13 non-intercept coefficients along the lambda values in log scale.
- Check the accuracy of your algorithm against the output from `glmnet`. The maximum difference between the two coefficient matrices should be less than 0.005.

```
lasso.fit = glmnet(X, y, alpha = 1, lambda = lam.seq)
max(abs(coef(lasso.fit) - myout))
```

  Students who use **Python** for this assignment can find the target Lasso coefficients returned by `coef(lasso.fit)` in file `Coef_Lasso.dat` on Campuswire.

Part II: Simulation Study (3pt)

Consider the following seven procedures:

- Full: run a linear regression model using all features
- Ridge.min and Ridge.1se: Ridge regression using lambda.min or lambda.1se
- Lasso.min and Lasso.1se: Lasso using lambda.min or lambda.1se

- L.Refit: Refit the model selected by Lasso using lambda.1se
- PCR: principle components regression with the number of components chosen by 10-fold cross validation

1. Download `BostonData2.csv` from Campuswire.

   The first 14 columns are the same as the transformed Boston Housing data we used in Part I with "Y" being the response variable. I add 78 more predictors, which are the quadratic terms of the 12 numerical predictors (excluding the binary predictor `chas`) and all pairwise interactions between the 12 numerical predictors.

   a) Repeat the following simulation 50 times: In each iteration, randomly split the data into two parts, 75% for training and 25% for testing. For **each of the seven procedures**, fit a model based on the training data and obtain a prediction on the test data, record the mean squared prediction error (MSPE) on the test data.

   b) Summarize your results on MSPE graphically, e.g., using boxplot or stripchart, and comment your results.

2. Download `BostonData3.csv` from Campuswire.

   The first 92 columns are the same as `BostonData2.csv`, and the remaining 500 columns are artificially generated noise features.

   Repeat (a-b) above for the **seven procedures except Full**, and comment your results.

What to Submit

- A Markdown (or Notebook) file in HTML format, which should contain all necessary code and the corresponding output/results.
- Set the seed at the beginning of your code to be the last 4-dig of your University ID. So once we run your code, we can get the same result.
- Name your file starting with `Assignment_2_xxxx_netID..`, where "xxxx" is the last 4-dig of your University ID and make sure the same 4-dig is used as the seed in your code.

  For example, the submission for Max Chen with netID 'mychen12' and UID '672757127' should be named as `Assignment_1_7127_mychen12_MaxChen.html`. You can add whatever characters after your netID.

Appendix

In case you don't know where to start with your own function `MyLasso`, you can follow the structure given below.

```
One_var_lasso = function(r, x, lam){
    ###############
    # YOUR CODE
    ###############
}
MyLasso = function(X, y, lam.seq, maxit = 100){
    # X: n-by-p design matrix without the intercept
    # y: n-by-1 response vector
    # lam.seq: sequence of lambda values
    # maxit: number of updates for each lambda

    n = length(y)
    p = dim(X)[2]
    nlam = length(lam.seq)
    # arrange lam.seq from large to small
    lam.seq = sort(lam.seq, decreasing = TRUE)

    ##############################
    # YOUR CODE
    # Center and scale X
    # Center y
    # Record the corresponding means and scales
    ##############################

    # Initialize coef vector b and residual vector r
    b = XXX
    r = XXX
    B = XXX
    # Triple nested loop
    for(m in 1:nlam){
        lam = XXX # assign lambda value
        for(step in 1:maxit){
            for(j in 1:p){
                r = r + (X[,j]*b[j])
                b[j] = One_var_lasso(r, X[, j], lam)
                r = r - X[, j] * b[j]
            }
         }
        B[m, -1] = b
    }

    ##############################
    # YOUR CODE
    # Scale back the coefficients and update the intercepts B[, 1]
     ##############################

    return(t(B))
 }
```

Note: You need to write your own function `One_var_lasso` to solve the one-variable Lasso for β_j. Check hints given on the next page.

In the CD algorithm, at each iteration, we repeatedly solve a one-dimensional Lasso problem for β_j while holding the other $(p-1)$ coefficients at their current values:

$$\min_{\beta_j} \sum_{i=1}^{n} (y_i - \sum_{k \neq j} x_{ik}\hat{\beta}_k - x_{ij}\beta_j)^2 + \lambda \sum_{k \neq j} |\hat{\beta}_k| + \lambda|\beta_j|, \tag{1}$$

which is equivalent to solving

$$\min_{\beta_j} \sum_{i=1}^{n} (r_i - x_{ij}\beta_j)^2 + \lambda|\beta_j|,$$

where

$$r_i = y_i - \sum_{k \neq j} x_{ik}\hat{\beta}_k. \tag{2}$$

How to solve (1)? In class we have discussed how to find the minimizer of

$$f(x) = (x-a)^2 + \lambda|x|,$$

which is given by

$$x^* = \arg\min_x f(x) = \text{sign}(a)(|a| - \lambda/2)_+ = \begin{cases} a - \lambda/2, & \text{if } a > \lambda/2; \\ 0, & \text{if } |a| \leq \lambda/2; \\ a + \lambda/2, & \text{if } a < -\lambda/2. \end{cases} \tag{3}$$

We can write (1) in the form of $f(x)$ and then use the solution given above.

Define two $n \times 1$ vectors: $\mathbf{r} = (r_1, \dots, r_n)^t$ with its i-th element being r_i defined in (2) and $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})^t$ with its i-th element being x_{ij}. Then

$$\begin{pmatrix} r_1 - x_{1j}\beta_j \\ r_2 - x_{2j}\beta_j \\ \cdots \\ r_n - x_{nj}\beta_j \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \cdots \\ r_n \end{pmatrix} - \begin{pmatrix} x_{1j} \\ x_{2j} \\ \cdots \\ x_{nj} \end{pmatrix} \beta_j = \mathbf{r} - \mathbf{x}_j\beta_j.$$

So we can rewrite the objective function in (1) as

$$\sum_{i=1}^{n} (r_i - x_{ij}\beta_j)^2 + \lambda|\beta_j| = \|\mathbf{r} - \mathbf{x}_j\beta_j\|^2 + \lambda|\beta|_j. \tag{4}$$

The first term above is like the RSS from a regression model with only one predictor (whose coefficient is β_j) without the intercept. The corresponding LS estimate is given by

$$\hat{\beta}_j = \mathbf{r}^t\mathbf{x}_j / \|\mathbf{x}_j\|^2.$$

Then we have

$$\begin{aligned}\|\mathbf{r}-\mathbf{x}_j\beta_j\|^2 &= \|\mathbf{r}-\mathbf{x}_j\hat{\beta}_j+\mathbf{x}_j(\beta_j-\hat{\beta}_j)\|^2 \\ &= \|\mathbf{r}-\mathbf{x}_j\hat{\beta}_j\|^2+\|\mathbf{x}_j(\beta_j-\hat{\beta}_j)\|^2+ \\ &\quad 2\times\text{inner-product-between } (\mathbf{r}-\mathbf{x}_j\hat{\beta}_j) \text{ and } \mathbf{x}_j(\beta_j-\hat{\beta}_j) \\ &= \|\mathbf{r}-\mathbf{x}_j\hat{\beta}_j\|^2+\|\mathbf{x}_j(\beta_j-\hat{\beta}_j)\|^2, \qquad (5)\end{aligned}$$

where the last equality is due to the fact that the inner product term is zero since the two vectors are orthogonal[1]

Note that the first term of (5) has nothing do with β_j. So to minimize (1) or equivalently (4) with respect to β_j, we can ignore the first term and instead minimize

$$\begin{aligned}\|\mathbf{x}_j(\beta_j-\hat{\beta}_j)\|^2+\lambda|\beta_j| &= \|\mathbf{x}_j\|^2(\beta_j-\hat{\beta}_j)^2+\lambda|\beta_j| \\ &= \|\mathbf{x}_j\|^2\left((\beta_j-\hat{\beta}_j)^2+\frac{\lambda}{\|\mathbf{x}_j\|^2}|\beta_j|\right) \\ &\propto (\beta_j-\hat{\beta}_j)^2+\frac{\lambda}{\|\mathbf{x}_j\|^2}|\beta_j|.\end{aligned}$$

Now we can use (3), the solution we have derived for $f(x)$, with

$$a=\hat{\beta}_j=\mathbf{r}^t\mathbf{x}_j/\|\mathbf{x}_j\|^2, \quad \lambda=\lambda/\|\mathbf{x}_j\|^2.$$

[1] This is because $(\mathbf{r}-\mathbf{x}_j\hat{\beta}_j)$ represents the residual vector from a regression model with $\mathbf{x}_j$ being a column (actually the only column) of the design matrix, and therefore it is orthogonal to $\mathbf{x}_j$.