STAT/BIST 5665 Homework 5

Fall 2019

Due: Wednesday, October 30.
Note: please try to use R markdown for any problem that involves coding. Submission of code is required.

1. Show the MSE of the ridge estimator on page 31 of Lecture 7 notes. Compare with the MSE of the least squares estimator.

2. Simulation to compare the prediction performance of least squares, lasso and ridge.

   (a) Generate $(y_i, \mathbf{X}_i)$, $i = 1, \dots, 200$, based on page 63 of the Lecture 7 notes.

   (b) Use 100 data points to fit least squares, lasso and ridge regression. You may use cross validation to choose the tuning parameter, which is available in `glmnet` package.

   (c) Use another 100 points for testing, i.e., compute $\|\mathbf{y}_{test} - \mathbf{X}_{test}\widehat{\beta}\|^2$ for each estimator.

   (d) Repeat the above for at least 100 times, and report the average prediction error for each estimator. Compare and make conclusion.

3. Consider the linear regression model,

$$y_i = x_{i1}\beta_1 + \cdots + x_{ip}\beta_p + \epsilon_i, \qquad 1 \leqslant i \leqslant n,$$

   or in matrix form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon,$$

   where $\mathbf{y} = (y_1, \dots, y_n)^\top$, $\mathbf{x}_j = (x_{1j}, \dots, x_{nj})^\top$, $\mathbf{X}_{n\times p} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$, and $\epsilon = (\epsilon_1, \dots, \epsilon_n)^\top$.
   The lasso (least absolute shrinkage and selection operator) method estimates the coefficient vector $\boldsymbol{\beta}$ by minimizing the following objective function,

$$\min_{\boldsymbol{\beta}\in R^p} \left\{ L(\boldsymbol{\beta}) \equiv \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

   where $\|\cdot\|$ denotes the ℓ_2-norm, $\lambda \geqslant 0$ is the tuning parameter controlling the degree of penalization.

1) Consider a simple one-dimensional problem:

$$\widehat{\theta} = \arg\min_{\theta \in R} \left\{ \frac{1}{2}(z-\theta)^2 + \lambda|\theta| \right\},$$

where $z \in R$ is a known value. Proof that the minimizer $\widehat{\theta}$ is given by the soft-threshold operator,

$$\widehat{\theta} = \mathcal{S}(z;\lambda),$$

where

$$\mathcal{S}(z;\lambda) = \operatorname{sgn}(z)(|z|-\lambda)_+ = \begin{cases} z-\lambda, & \text{if } z > \lambda \\ 0, & \text{if } |z| \leqslant \lambda \\ z+\lambda, & \text{if } z < -\lambda. \end{cases}$$

2) Now consider solving the lasso problem using an iterative algorithm. Suppose the current value of the coefficient vector is $\boldsymbol{\beta}^{(t)} = (\beta_1^{(t)}, \ldots, \beta_p^{(t)})^T$. Consider minimizing L with respect to a single β_j for some $1 \leqslant j \leqslant p$, given other coefficients held fixed at $\beta_k^{(t)}, k \neq j$, i.e.,

$$\beta_j^{(t+1)} = \arg\min_{\beta_j} L(\beta_j \mid \beta_k^{(t)}, k \neq j).$$

Using 1) to show that

$$\beta_j^{(t+1)} = \mathcal{S}(\tilde{z}_j; \lambda n(\mathbf{x}_j^T\mathbf{x}_j)^{-1}),$$

where

$$\tilde{z}_j = (\mathbf{x}_j^T\mathbf{x}_j)^{-1}\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}^{(t)} + \mathbf{x}_j\beta_j^{(t)}).$$

$\mathcal{S}(\cdot;\lambda)$ is called the soft-thresholding operator.

3) Coordinate descent algorithm optimizes a target function with respect to a single parameter at a time, iteratively cycling through all parameters until convergence is reached. Implement the coordinate descent algorithm for lasso as a function in R or other programming environment of your choice.

Note that in lecture notes, the details on an efficient implementation of the coordinate descent algorithm are given.

Submit your code electronically, and write documentations on how the function should be used.

Using any existing implementation from internet or existing R packages is strictly prohibited.

Remark, for the stopping criterion, let $\boldsymbol{\beta}^{(s)}(\lambda)$ and $\boldsymbol{\beta}^{(s+1)}(\lambda)$ be the solution at the sth and $s+1$th cycle. You can stop based on the absolute difference criterion,

$$\|\boldsymbol{\beta}^{(s)}(\lambda) - \boldsymbol{\beta}^{(s+1)}(\lambda)\| \leqslant \epsilon,$$

where ϵ is a small positive constant, say, 10^{-4}. Alternatively, you can stop based on the relative difference criterion,

$$\frac{\|\boldsymbol{\beta}^{(s)}(\lambda) - \boldsymbol{\beta}^{(s+1)}(\lambda)\|}{\|\boldsymbol{\beta}^{(s)}(\lambda)\|} \leqslant \epsilon.$$

Numerically, to make it more stable, it is commonly to use

$$\frac{\|\boldsymbol{\beta}^{(s)}(\lambda) - \boldsymbol{\beta}^{(s+1)}(\lambda)\|}{\|\boldsymbol{\beta}^{(s)}(\lambda)\| + \epsilon} \leqslant \epsilon.$$

4) Verify that you get the same solutions as those produced by the R package `glmnet`.

```
set.seed(123)
library(glmnet)

n <- 100
p <- 300
sigma <- 0.5
beta <- c(rep(1, 5), rep(0, p - 5))
X <- matrix(nrow = n, ncol = p, rnorm(n * p))
e <- rnorm(n, mean = 0, sd = sigma)
y <- X %*% beta + e

lambda = exp(seq(log(1), log(0.005), length = 101))

ptm <- system.time(fit1 <- glmnet(
  X,
  y,
  intercept = FALSE,
  standardize = FALSE,
  lambda = lambda
))

plot(fit1)

ptm
```

You should report (1) a solution path plot, to compare with the one produced by `glmnet`; you may also compute the difference between your solutions and the `glmnet` solutions numerically; (2) report the computation time of your implementation, and compare with the time by `glmnet`; no need to beat it!