Code

PSL (F20) Coding Assignment 2

08/28/2020

Data Preparation

Load required R packages and apply proper transformations on the Boston Housing Data.

Hide

```
library(MASS)
library(glmnet)
myData = Boston
names(myData)[14] = "Y"
iLog = c(1, 3, 5, 6, 8, 9, 10, 14);
myData[, iLog] = log(myData[, iLog]);
myData[, 2] = myData[, 2] / 10;
myData[, 7] = myData[, 7]^2.5 / 10^4
myData[, 11] = exp(0.4 * myData[, 11]) / 1000;
myData[, 12] = myData[, 12] / 100;
myData[, 13] = sqrt(myData[, 13]);
X = as.matrix(myData[, -14])
y = myData$Y
lam.seq = c(0.30, 0.2, 0.1, 0.05, 0.02, 0.005)
```

CD for Lasso

Implement the Coordinate Descent algorithm for Lasso. Some part of the function `MyLasso` is blocked here, but your submission should include all code used to produce your results.

Hide

```
MyLasso = function(X, y, lam.seq, maxit = 50) {

    # 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
    # Center/Scale X
    # Center y

    n = length(y)
    p = dim(X)[2]
    nlam = length(lam.seq)

    ##############################
    # YOUR CODE:
    # Record the corresponding means and scales
    # For example,
    # y.mean = mean(y)
    # yc = centered y
    # Xs = centered and scaled X
    ##############################

    # Initilize coef vector b and residual vector r
    b = rep(0, p)
    r = yc
    B = matrix(nrow = nlam, ncol = p + 1)

    # Triple nested loop
    for (m in 1:nlam) {
        lam = 2 * n * lam.seq[m]
        for (step in 1:maxit) {
            for (j in 1:p) {
                r = r + (Xs[, j] * b[j])
                b[j] = one_var_lasso(r, Xs[, j], lam)
                r = r - Xs[, j] * b[j]
            }
        }
        B[m, ] = c(0, b)
    }

    ##############################
    # YOUR CODE:
    # Scale back the coefficients;
    # Update the intercepts stored in B[, 1]
    ##############################
}
```

Some Technical Details

- We use the following function to solve the Lasso estimate for β_j given other coefficients fixed; see the derivation in Coding2.pdf.

Hide

```
one_var_lasso = function(r, x, lam) {
    xx = sum(x^2)
    xr = sum(r * x)
    b = (abs(xr) - lam/2)/xx
    b = sign(xr) * ifelse(b > 0, b, 0)
    return(b)
}
```

- `glmnet` standardizes the data using a different definition of “standard deviation”, which is divided by **n**, while the command `sd(z)` in R divides the sum of squares by **(n-1)** where z is a n-by-1 data vector. We suggest to use `sd(z) * sqrt((n-1)/n)` to compute the standard deviation of a data vetor used in `myLasso`, since

$$\sqrt{\frac{\sum_i (z_i - \text{z.mean})^2}{n}} = \sqrt{\frac{\sum_i (z_i - \text{z.mean})^2}{n-1}} \cdot \sqrt{\frac{n-1}{n}} = \text{sd(z)} \cdot \sqrt{\frac{n-1}{n}}.$$

- Why we need to scale the lambda value by (2n) in `lam = 2*n*lam.seq[m]`? As detailed in Coding2.pdf, the `one_var_lasso` function is derived based on objective function

  $$\text{RSS} + \lambda|\beta|,$$

  while the objective function used in `glmnet` is

  $$\frac{1}{2n}\text{RSS} + \lambda|\beta| \propto \text{RSS} + 2n\lambda|\beta|$$

  So to compare results from the two algorithms on an equal footing, we need to scale our lambda value by (2n).

##Check the Accuracy

Hide

```
lam.seq = c(0.30, 0.2, 0.1, 0.05, 0.02, 0.005)
lasso.fit = glmnet(X, y, alpha = 1, lambda = lam.seq)
coef(lasso.fit)
```

```
14 x 6 sparse Matrix of class "dgCMatrix"
                     s0         s1           s2           s3
(Intercept)  3.16239335  3.5089461  3.855935763  3.778455800
crim         .           .          .            .          
zn           .           .          .            .          
indus        .           .          .            .          
chas         .           .          .            .          
nox          .           .          .            .          
rm           .           .          .            0.240416063
age          .           .          .            .          
dis          .           .          .            .          
rad          .           .          .            .          
tax          .           .          .           -0.055308804
ptratio      .           .         -0.004310305 -0.023647910
black        .           .          .            0.008515144
lstat       -0.03741741 -0.1388176 -0.237634045 -0.244558377
                      s4           s5
(Intercept)  3.542129253  3.925458057
crim         .            .          
zn           .            .          
indus        .           -0.005398865
chas         0.066692255  0.099119798
nox          .           -0.125162756
rm           0.417249062  0.459272519
age          .            .          
dis         -0.004681106 -0.120437573
rad          .            0.014980251
tax         -0.080579126 -0.149039129
ptratio     -0.033626958 -0.038211176
black        0.031234249  0.043402206
lstat       -0.243489936 -0.256708941
```

Hide

```
myout = MyLasso(X, y, lam.seq, maxit = 50)
rownames(myout) = c("Intercept", colnames(X))
myout
```

```
                [,1]       [,2]         [,3]         [,4]
Intercept  3.16239335  3.5089461  3.855935052  3.777919609
crim       0.00000000  0.0000000  0.000000000  0.000000000
zn         0.00000000  0.0000000  0.000000000  0.000000000
indus      0.00000000  0.0000000  0.000000000  0.000000000
chas       0.00000000  0.0000000  0.000000000  0.000000000
nox        0.00000000  0.0000000  0.000000000  0.000000000
rm         0.00000000  0.0000000  0.000000000  0.240724202
age        0.00000000  0.0000000  0.000000000  0.000000000
dis        0.00000000  0.0000000  0.000000000  0.000000000
rad        0.00000000  0.0000000  0.000000000  0.000000000
tax        0.00000000  0.0000000  0.000000000 -0.055335623
ptratio    0.00000000  0.0000000 -0.004303294 -0.023646161
black      0.00000000  0.0000000  0.000000000  0.008522442
lstat     -0.03741741 -0.1388176 -0.237638248 -0.244528823
                 [,5]         [,6]
Intercept  3.541573070  3.927045881
crim       0.000000000  0.000000000
zn         0.000000000  0.000000000
indus      0.000000000 -0.005275075
chas       0.066670285  0.099049491
nox        0.000000000 -0.123495061
rm         0.417613831  0.459679619
age        0.000000000  0.000000000
dis       -0.004686483 -0.119808332
rad        0.000000000  0.015118868
tax       -0.080634128 -0.149512107
ptratio   -0.033638505 -0.038234008
black      0.031255297  0.043452192
lstat     -0.243439646 -0.256682319
```

Compare the accuracy of my algorithm against the output from glmnet. The maximum difference between the two coefficient matrices is less than 0.005.

Hide

```
max(abs(coef(lasso.fit) - myout))
```

```
[1] 0.001667695
```