STA 545 Statistical Data Mining I, Fall 2020

Homework 5

Stella Liao

October 7, 2020

1. (25 points) Please use the datasets shown in Question 1 of your homework 4 to fit a PLS model on the set A, with the parameter M chosen by the set B. Report the value of M selected by the set B, the estimated regression coefficients of the original input variables, and the test error obtained. In addition, please fit a lasso regression model on the set A, with the tuning parameter $ chosen by the set B. Report the test error obtained, along with the the estimated regression coefficients of the original input variables.

1) Preparing data

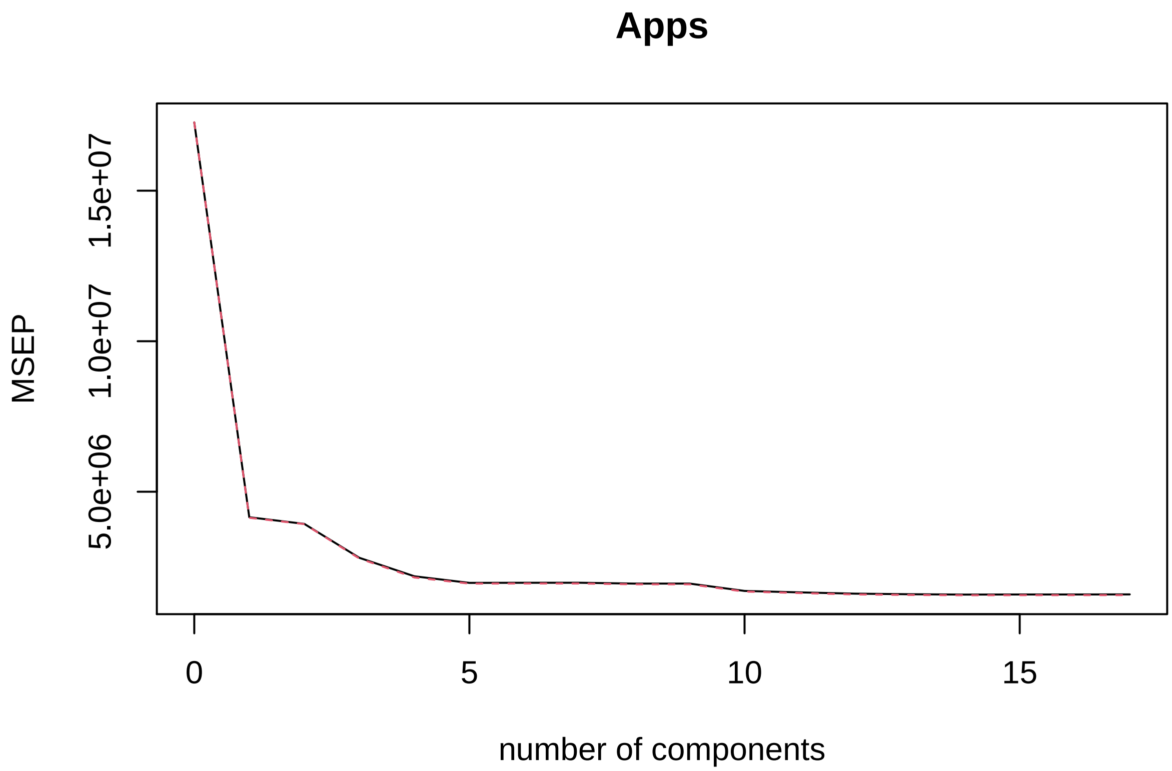
```
library(ISLR)
library(glmnet)
library(pls)

data(College)
set.seed(123)
train.num <- sample(777, size = 500, replace = FALSE)
train.College <- College[train.num,]
test.College <- College[-train.num,]

set.seed(1)
set.num <- sample(500, size = 250, replace = FALSE)
set.A <- College[set.num,]
set.B <- College[-set.num,]
X.set.A <- model.matrix(Apps ~ .,set.A)
X.set.B <- model.matrix(Apps ~ .,set.B)
test.matrix.College <- model.matrix(Apps ~ .,test.College)
grid <- 10 ^ seq(10, -2, length = 100)
```

2) PLS model

```
fit.PLS <- plsr(Apps ~ .,
                data = set.B,
                family = "gaussian",
                scale = F,
                validation = "CV")
validationplot(fit.PLS,val.type = "MSEP")
```




```r
summary(fit.PLS)
```

```
## Data:    X dimension: 527 17
##  Y dimension: 527 1
## Fit method: kernelpls
## Number of components considered: 17
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            4156     2038     1983     1675     1480     1404     1406
## adjCV         4156     2033     1981     1668     1467     1396     1397
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV        1406     1396     1396      1306      1287      1271      1263
## adjCV     1397     1388     1388      1298      1279      1260      1254
##        14 comps  15 comps  16 comps  17 comps
## CV         1259      1260      1260      1261
## adjCV      1250      1251      1251      1251
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       40.42    68.55    92.44    97.05    98.76    99.29    99.65    99.97
## Apps    77.46    82.01    86.86    90.42    90.81    90.86    90.93    90.97
##       9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
## X      100.00    100.00    100.00    100.00     100.0    100.00    100.00
## Apps    91.12     92.18     92.49     92.89      92.9     92.91     92.92
##       16 comps  17 comps
## X       100.00       100
## Apps     92.92        93
```

According to the output of `summary(fit.pcr2)`, we find that when `ncomp` = 16, the cross validation error is lowest.

```
fit.PLS2 <- pcr(Apps ~ .,
                data = set.A,
                family = "gaussian",
                scale = F,
                ncomp = 16)

pred.PLS <- predict(fit.PLS2,
                    test.College,
                    scale = F,
                    ncomp = 16)

# test error in PLS model
test.error.PLS <- mean((pred.PLS - test.College$Apps)^2)
test.error.PLS
```

```
## [1] 1726332
```

```
#the coefficients of the original outputs in the PLS model
as.data.frame(fit.PLS2$coefficients[,,16])
```

```
##             fit.PLS2$coefficients[, , 16]
## PrivateYes                    -1.16873440
## Accept                         1.31283568
## Enroll                        -0.44554162
## Top10perc                     13.32145875
## Top25perc                      2.71971884
## F.Undergrad                    0.05365231
## P.Undergrad                    0.09952674
## Outstate                      -0.03998232
## Room.Board                     0.13036972
## Books                         -0.23101502
## Personal                      -0.10137072
## PhD                           -2.75395703
## Terminal                      -4.59517609
## S.F.Ratio                     39.73244558
## perc.alumni                   -7.99349032
## Expend                         0.09091874
## Grad.Rate                      1.56692304
```

3) Lasso regression model

```
lasso.mod <- glmnet(X.set.A,
                    set.A$Apps,
                    family = "gaussian",
                    standardize = F,
                    alpha = 1,
                    lambda = grid)

cv.out = cv.glmnet(X.set.B,
                   set.B$Apps,
                   lambda = grid,
                   standardize = F,
                   alpha = 1)
```

```r
bestlam.B.lasso = cv.out$lambda.min
bestlam.B.lasso
```

```
## [1] 231.013
```

```r
lasso.pred = predict(lasso.mod,
                     s = bestlam.B.lasso,
                     newx = test.matrix.College)

# the test error in the lasso model
mean((lasso.pred - test.College$Apps)^2)
```

```
## [1] 1777035
```

```r
#the coefficients of the original outputs in the lasso model
lasso.coef = predict(lasso.mod,
                     type = "coefficients",
                     s = bestlam.B.lasso)
lasso.coef
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -516.12632721
## (Intercept)    .
## PrivateYes     .
## Accept         1.29229257
## Enroll        -0.33117710
## Top10perc     11.48847212
## Top25perc      2.38685707
## F.Undergrad    0.04720434
## P.Undergrad    0.10098330
## Outstate      -0.04880593
## Room.Board     0.14189798
## Books         -0.20688684
## Personal      -0.11410355
## PhD           -1.15822172
## Terminal      -2.93639397
## S.F.Ratio      8.28915054
## perc.alumni   -6.30300183
## Expend         0.08161484
## Grad.Rate      0.10310942
```

Problem 2

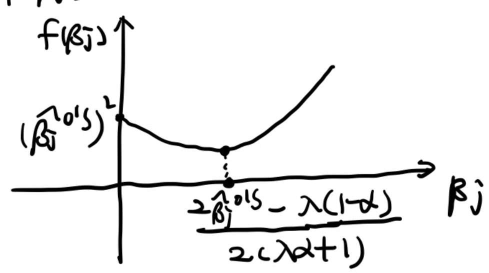
Define the loss function as.

$$f(\beta_j) = (\beta_j - \hat{\beta}_j^{ols})^2 + \lambda\alpha\beta_j^2 + \lambda(1-\alpha)|\beta_j|$$

1) when $\beta_j \geq 0$.

$$f(\beta_j) = (\beta_j - \hat{\beta}_j^{ols})^2 + \lambda\alpha\beta_j^2 + \lambda(1-\alpha)\beta_j$$
$$= (\lambda\alpha+1)\beta_j^2 + [\lambda(1-\alpha) - 2\hat{\beta}_j^{ols}]\cdot\beta_j + (\hat{\beta}_j^{ols})^2$$

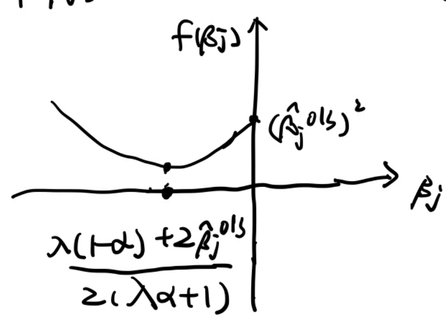
the graph of $f(\beta_j)$ would be like and $\hat{\beta}_j^{ols} > \frac{\lambda(1-\alpha)}{2}$



2) when $\beta_j < 0$.

$$f(\beta_j) = (\beta_j - \hat{\beta}_j^{ols})^2 + \lambda\alpha\beta_j^2 - \lambda(1-\alpha)\beta_j$$
$$= (\lambda\alpha+1)\beta_j^2 - [\lambda(1-\alpha) + 2\hat{\beta}_j^{ols}]\cdot\beta_j + (\hat{\beta}_j^{ols})^2$$

the graph of $f(\beta_j)$ would be like and $\hat{\beta}_j^{ols} < \frac{-\lambda(1-\alpha)}{2}$



Therefore, the solution is

$$\hat{\beta}_j = \begin{cases} \frac{2\hat{\beta}_j^{ols} - \lambda(1-\alpha)}{2(\lambda\alpha+1)} & \text{if } \hat{\beta}_j^{ols} > \frac{\lambda(1-\alpha)}{2} \\ 0 & \text{if } \hat{\beta}_j^{ols} \leq \left|\frac{\lambda(1-\alpha)}{2}\right| \\ \frac{\lambda(1-\alpha) + 2\hat{\beta}_j^{ols}}{2(\lambda\alpha+1)} & \text{if } \hat{\beta}_j^{ols} < \frac{-\lambda(1-\alpha)}{2} \end{cases}$$

And we could also use the coordinate descent algorithm for the lasso method to solve it.

Problem 3 (50 points) Please write your own R function for the coordinate descent algorithm to fit lasso regression models. Please use the prostate cancer data to compare the results from your own R function with the results from the glmnet R function in the glmnet R package.

1) using `glmnet()` function

```
prostate <- read.csv("prostate.csv")
# have a look at prostate data
head(prostate, n = 2 )
```

```
##   X     lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
##   train
## 1  TRUE
## 2  TRUE
```

```
# standardizing prostate data to make sure its mean equal to 0 and variance equal to 1
X.prostate = scale(as.matrix(prostate[,2:9]))
y.prostate = scale(as.vector(prostate[,10]))

set.seed(123)
lasso.mod.prostate <- glmnet(X.prostate,
                             y.prostate,
                             intercept = F,
                             standardize = T,
                             family="gaussian",
                             alpha = 1,
                             lambda = grid)

cv.out.prostate = cv.glmnet(X.prostate,
                            y.prostate,
                            alpha = 1,
                            standardize = T,
                            intercept = F,
                            lambda = grid)

bestlam.B.lasso.prostate = cv.out.prostate$lambda.min
coef.lasso.prostate = predict(lasso.mod.prostate, type = "coefficients", s = bestlam.B.lasso.prostate)
coef.lasso.prostate
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept)  .
## lcavol       0.54796374
## lweight      0.22164654
## age         -0.10768686
## lbph         0.10690468
## svi          0.24463535
## lcp         -0.06094245
## gleason      0.02015517
## pgg45        0.08356488
```

```
bestlam.B.lasso.prostate
```

```
## [1] 0.01
```

2) using coordinate descent algorithm [1]

The loss function for lasso problem is

$$f(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

According to the coordinate algorithm,

$$\min_{\beta_K} f(\beta_1, \beta_2, \cdots, \beta_K, \beta_{K+1}, \cdots, \beta_p)$$

$$\frac{\partial f(\beta)}{\partial \beta_K} = -\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j) x_{iK} + \frac{\partial(\lambda|\beta_K|)}{\partial \beta_K}$$

$$= -\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j) x_{iK} + \sum_{i=1}^{n} x_{iK}^2 \beta_K + \frac{\partial(\lambda|\beta_K|)}{\partial \beta_K}$$

$$= -r_K + z_K\beta_K + \frac{\partial(\lambda|\beta_K|)}{\partial \beta_K}$$

$$r_K = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j) x_{iK}$$

$$z_K = \sum_{i=1}^{n} x_{iK}^2$$

And easily, we could know that

$$\frac{\partial(\lambda|\beta_K|)}{\partial \beta_K} = \begin{cases} \lambda, & \beta_K > 0 \\ [-\lambda, \lambda], & \beta_K = 0 \\ -\lambda, & \beta_K < 0 \end{cases} \Rightarrow \frac{\partial(f(\beta))}{\partial \beta_K} = \begin{cases} -r_K + z_K\beta_K + \lambda, & \beta_K > 0 \\ [-r_K - \lambda, -r_K + \lambda], & \beta_K = 0 \\ -r_K + z_K\beta_K - \lambda, & \beta_K < 0 \end{cases}$$

we make $\frac{\partial(f(\beta))}{\partial \beta_K} = 0$, then we could get the minimum value

$$\beta_K^* = \begin{cases} (r_K - \lambda)/z_K, & r_K > \lambda \\ 0, & r_K < |\lambda| \\ (r_K + \lambda)/z_K, & r_K < -\lambda \end{cases}$$

In the soft-thresholding function $S(a,b) = \begin{cases} a-b, & a > b \\ 0, & a < |b| \\ a+b, & a < -b \end{cases}$

$$\beta_K^* = \frac{1}{z_K} S(r_K, \lambda)$$

11

```r
cd.lasso <- function(X, y, lambda = 0.1, max.iter = 1000,  tol = 1e-6 ){

  #to create soft-thresholding function
  soft.thresholding <- function(b, l){
    result = rep(0, length(b))
    result[b >  l] = b[b > l] - l
    result[b < -l] = b[b < -l] + l
    result
  }

  #to create function to calculate beta k star
  compute.bks <- function(k, X, y, beta, lambda){
        y.predict = X %*% beta
        rk = X[, k] %*% (y - y.predict+ X[, k] * beta[k])
        rk = rk / nrow(X)
        zk = colSums(X^2)[k]
        zk = zk / nrow(X)
        beta.k = soft.thresholding(rk, lambda)
        beta.k = beta.k / zk
        beta.k }
  #initialize some parameters
  tol.curr = 1
  iter = 1
  all.beta = rep(0, ncol(X))
  old.all.beta = rep(0, ncol(X))

  #update beta k
  while (tol < tol.curr && iter < max.iter) {
    for (k in 1:ncol(X)) {
        old.all.beta[k] = all.beta[k]
        all.beta[k] = compute.bks(k, X, y, all.beta, lambda)
    }
    tol.curr = abs(all.beta - old.all.beta)
    iter = iter + 1
  }
  #print all beta
  all.beta
}

coef.cd.lasso.prostate <- cd.lasso(X = X.prostate,
                                   y = y.prostate,
                                   lambda = bestlam.B.lasso.prostate,
                                   tol = 1e-12)

comparison <- data.frame("glmnet" = as.data.frame(summary(coef.lasso.prostate))$x,
                         "coordinate descent" = coef.cd.lasso.prostate)

predictors <- c("lcavol", "lweight", "age", "lbph", 'svi', "lcp" ,"gleason", "pgg45")
comparison <- cbind(predictors, comparison)

comparison
```

```
##   predictors      glmnet coordinate.descent
## 1     lcavol  0.54796374         0.54775369
```

```
## 2    lweight  0.22164654         0.22161781
## 3        age -0.10768686        -0.10755924
## 4       lbph  0.10690468         0.10682744
## 5        svi  0.24463535         0.24451606
## 6        lcp -0.06094245        -0.06062363
## 7    gleason  0.02015517         0.02016581
## 8      pgg45  0.08356488         0.08342290
```

We could see that the coefficients of those predictors in the two functions are almost same.

Reference

[1] https://www.scutmath.com/coordiante_descent_for_lasso.html