HW6

Jinhong Du - 12243476

2019/10/08

Contents

1. In this problem we will examine the performance of ridge regression and Lasso regression on the `seatpos` data set from Faraway. The response variable, `hipcenter`, measures the position of the seated driver's hips in the car. The covariates are `Age`, `Weight`, `HtShoes` (height in shoes), `Ht` (height without shoes), `Seated` (seated height), `Arm`, `Thigh`, `Leg` (arm / thigh / lower leg length).

(a) Examine the correlations among the covariates and comment on what you see, and how you might expect this to affect the regression.

As we can see, the covariates `HtShoes`, `Ht`, `Seated` and `Leg` are highly correlated with each other (absolute value of correlation larger than 0.9). So there is collinearity in the covariates. The estimated coefficients of these covariates in the regression may be nearly nonidentifiable.

```
library(faraway)
data(seatpos)
cor(seatpos[,-9])
```

```
##                 Age      Weight     HtShoes          Ht     Seated       Arm
## Age      1.00000000 0.08068523 -0.07929694 -0.09012812 -0.1702040 0.3595111
## Weight   0.08068523 1.00000000  0.82817733  0.82852568  0.7756271 0.6975524
## HtShoes -0.07929694 0.82817733  1.00000000  0.99814750  0.9296751 0.7519530
## Ht      -0.09012812 0.82852568  0.99814750  1.00000000  0.9282281 0.7521416
## Seated  -0.17020403 0.77562705  0.92967507  0.92822805  1.0000000 0.6251964
## Arm      0.35951115 0.69755240  0.75195305  0.75214156  0.6251964 1.0000000
## Thigh    0.09128584 0.57261442  0.72486225  0.73496041  0.6070907 0.6710985
## Leg     -0.04233121 0.78425706  0.90843341  0.90975238  0.8119143 0.7538140
##              Thigh         Leg
## Age     0.09128584 -0.04233121
## Weight  0.57261442  0.78425706
## HtShoes 0.72486225  0.90843341
## Ht      0.73496041  0.90975238
## Seated  0.60709067  0.81191429
## Arm     0.67109849  0.75381405
## Thigh   1.00000000  0.64954120
## Leg     0.64954120  1.00000000
```

(b) Begin by standardizing each covariate so that we don't run into issues of how the code treats each covariate. Center each covariate to have zero mean, and then rescale it so that $\sum_i X_{ij}^2 = n$.

```
seatpos[,-9] <- apply(seatpos[,-9], 2, function(x)x-mean(x))
seatpos[,-9] <- apply(seatpos[,-9], 2, function(x)x/sqrt(mean(x^2)))
```

(c) Now we run ridge regression.

```
library(MASS)
model = lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg, lambda = ????)
betahat = coef(model) # intercept, then coefficients on the covariates
```

Note that by convention, the intercept term is not penalized by the ridge regression procedure. Run this at $\lambda = 0, 0.1, 1, 2, 5, 10, 20, 50$, and comment on what you see for the fitted coefficients. In particular, what do you see happening to the coefficients on the covariates `HtShoes`, `Ht`, and `Seated`? Discuss.

As λ increases, the coefficient of `HtShoes` increases and gets closer to 0. As λ increases, the coefficient of `Ht` decreases at first and then increases. As λ increases, the coefficient of `Seated` decreses and gets away from 0.

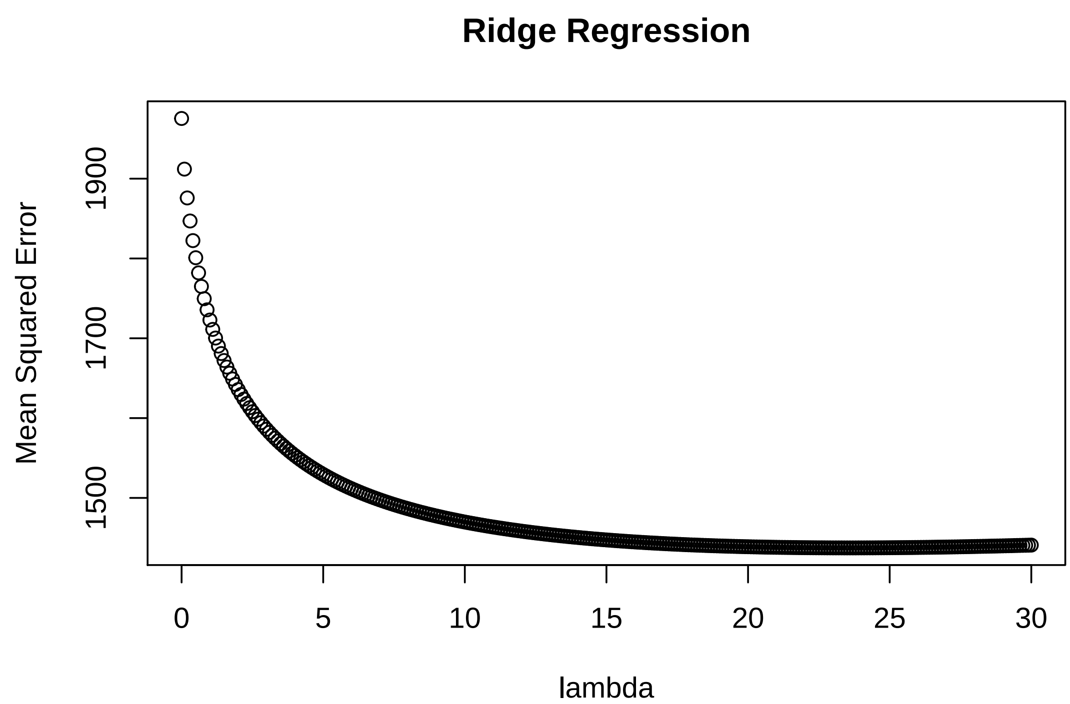
```
library(MASS)
lambda_list <- c(0, 0.1, 1, 2, 5, 10, 20, 50)
model <- lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg,
                  data=seatpos, lambda = lambda_list)
print(coef(model)) # intercept, then coefficients on the covariates
```

```
##                       Age     Weight    HtShoes         Ht     Seated       Arm
##  0.0 -164.8849 11.763894  0.9290423 -29.618082  6.630015  2.5974846 -4.418228
##  0.1 -164.8849 11.501678  0.9733863 -18.375451 -4.775794  2.2699896 -4.354180
##  1.0 -164.8849 11.209797  0.3796354 -11.749208 -9.785972  0.3649019 -4.640491
##  2.0 -164.8849 10.970371 -0.1942487 -10.701957 -9.759135 -0.9677302 -4.822986
##  5.0 -164.8849 10.237244 -1.3963694  -9.629934 -9.327971 -3.1470988 -4.940371
## 10.0 -164.8849  9.171969 -2.5307142  -8.974283 -8.878673 -4.7001831 -4.819280
## 20.0 -164.8849  7.612264 -3.5585050  -8.304574 -8.296598 -5.7236525 -4.571745
## 50.0 -164.8849  5.089632 -4.2130982  -7.115605 -7.138796 -5.8725225 -4.169567
##          Thigh        Leg
##  0.0 -4.370897 -21.626207
##  0.1 -4.120937 -21.397236
##  1.0 -4.293360 -20.059262
##  2.0 -4.495120 -18.783753
##  5.0 -4.809908 -16.052201
## 10.0 -4.992430 -13.486776
## 20.0 -5.015310 -10.990252
## 50.0 -4.652486  -8.169737
```

(d) Next we will use leave-one-out cross-validation to select a good value of λ. Fix a grid of λ values ranging between 0 and 20 (you should use a very fine grid, e.g. $0, 0.1, 0.2, \ldots$) For each data point $i = 1, \ldots, n$, run ridge with each λ value on the data set with point i removed, find β, then get the leave-one-out error for predicting Y_i. Average the squared error over all n choices of i. Plot the leave-one-out error against λ and find the best λ value. How do the estimated coefficients compare against least squares? Based on your leave-one-out analysis, does ridge appear to offer substantial improvement of the prediction error?

As we can see, as lambda increases, then leave-one-out error decreases when $\lambda < 23.6$ and then increases when $\lambda > 23.6$.

```
lambda_list <- seq(0,30,0.1)
p <- dim(seatpos)[2]
n <- dim(seatpos)[1]
n_lambda <- length(lambda_list)
MSE <- matrix(0, n, n_lambda)
beta <- array(0, c(n, n_lambda, p))
for(i in 1:n){
    model = lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg,
                     data=seatpos[-i,], lambda = lambda_list)
    y_pred <- as.matrix(cbind(const=1,seatpos[i,-9])) %*% t(coef(model))
    beta[i,,] <- coef(model)
    MSE[i,] <- (seatpos[i,9] - y_pred)^2
}
plot(lambda_list, apply(MSE,2,mean), xlab='lambda',
     ylab='Mean Squared Error', main='Ridge Regression')
```




```r
cat(lambda_list[which.min(apply(MSE,2,mean))])
```

```
## 23.6
```

```r
print(apply(beta[,which.min(apply(MSE,2,mean)),], 2, mean))
```

```
## [1] -164.897155    7.096162   -3.790556   -8.088862   -8.093368   -5.872812
## [7]   -4.502979   -4.981070  -10.345592
```

```r
print(apply(beta[,which.min(apply(MSE,2,mean)),], 2, median))
```

```
## [1] -164.890514    7.134557   -3.731478   -8.085774   -8.088057   -5.814973
## [7]   -4.410621   -4.976503  -10.301326
```

(e) Next we'll turn to the Lasso.

```r
library(glmnet)
model = glmnet(cbind(Age,Weight,HtShoes,Ht,Seated,Arm,Thigh,Leg), y = hipcenter, lambda = ????)
betahat = c(model$a0,as.matrix(model$beta)) # intercept, then coefficients on the covariates
```

Again, the intercept is not penalized. Run this at $\lambda = 0, 0.1, 1, 2, 5, 10, 20, 50$, and comment on what you see for the fitted coefficients. In particular, what do you see happening to the coefficients on the covariates HtShoes, Ht, and Seated? Discuss. [Note: the range of λ values that works well for ridge, will not necessarily be the right range of values for Lasso—the two λ's are penalizing different functions and are not comparable.]

In the output below, s7-s0 (reversed order) corresponds to $\lambda = 0, 0.1, 1, 2, 5, 10, 20, 50$. As λ increases, the coefficient of `HtShoes` increases and becomes 0 when $\lambda > 2$. As λ increases, the coefficient of `Ht` decreases at first and then increases. As λ increases, the coefficient of `Seated` decreses and becomes 0 when $\lambda > 0.1$.

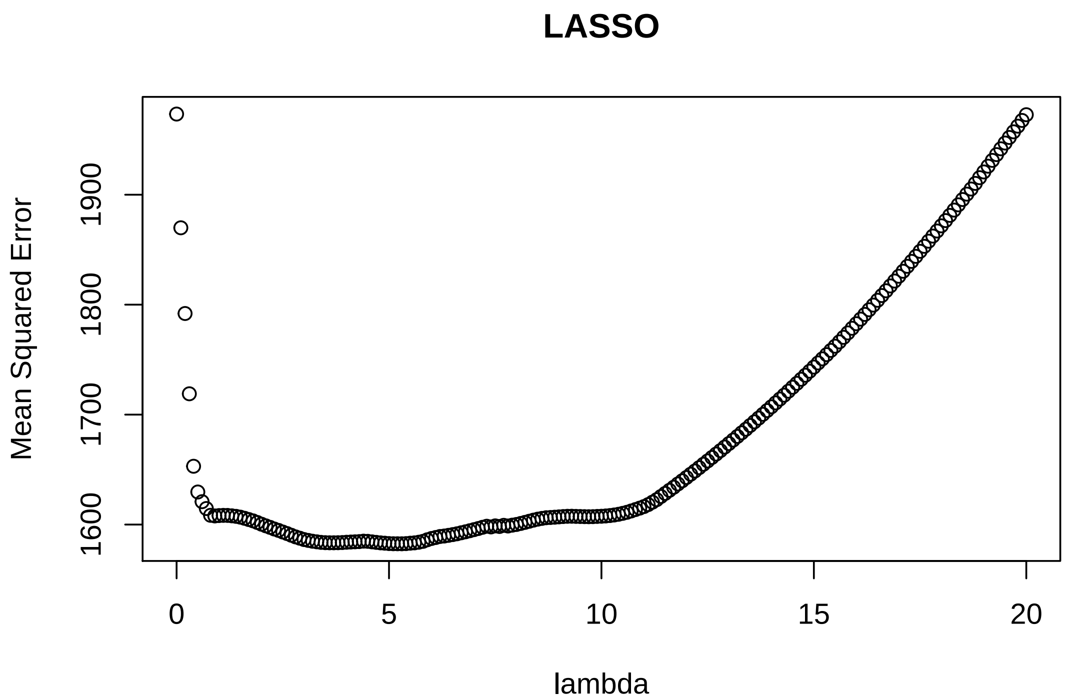
```
library(glmnet)
lambda_list <- c(0, 0.1, 1, 2, 5, 10, 20, 50)
model <- glmnet(x=as.matrix(seatpos[,-9]),  y = seatpos$hipcenter, lambda = lambda_list)
# glmnet will sort lambda to be decreasing
betahat <- c(model$a0,as.matrix(model$beta)) # intercept, then coefficients on the covariates
print(cbind(model$a0, t(model$beta)))
```

```
## 8 x 9 sparse Matrix of class "dgCMatrix"
##                    Age    Weight    HtShoes         Ht    Seated        Arm
## s0 -164.8849  .         .           .          .         .          .
## s1 -164.8849  .         .           .        -17.803728 .          .
## s2 -164.8849  .         .           .        -23.039970 .          .
## s3 -164.8849  4.068237 .           .        -24.501311 .          .
## s4 -164.8849  7.471686 .         -6.814971 -15.739548 .          .
## s5 -164.8849  9.514490 .        -14.643985  -6.302104 .         -2.060838
## s6 -164.8849 11.409939 0.4934133 -21.133922   .        0.9783011 -4.191980
## s7 -164.8849 11.656318 0.9687952 -24.220832   1.223035 2.4749935 -4.383074
##         Thigh       Leg
## s0  .          .
## s1  .        -10.13364
## s2  .        -15.36996
## s3  .        -18.86829
## s4 -2.946868 -21.59019
## s5 -3.851717 -21.83609
## s6 -4.368349 -21.73273
## s7 -4.258496 -21.64602
```

(f) Run the leave-one-out analysis again, now with Lasso, and answer the same questions as above.

As we can see, as lambda increases, then leave-one-out error decreases when $\lambda < 5.3$ and then increases when $\lambda > 5.3$.

```
lambda_list <- seq(0,20,0.1)
p <- dim(seatpos)[2]
n <- dim(seatpos)[1]
n_lambda <- length(lambda_list)
MSE <- matrix(0, n, n_lambda)
beta <- array(0, c(n, n_lambda, p))

for(i in 1:n){
    model <- glmnet(x= as.matrix(seatpos[-i,-9]),
                    y = seatpos$hipcenter[-i], lambda = lambda_list,
                    standardize=FALSE)
    betahat <- cbind(as.matrix(model$a0), t(as.matrix(model$beta)))

    y_pred <- as.matrix(cbind(const=1,seatpos[i,-9])) %*% t(betahat)
    beta[i,,] <- betahat
    MSE[i,] <- t((seatpos[i,9] - y_pred)^2)
}
ind <- which.min(rev(apply(MSE,2,mean)))
plot(rev(lambda_list), apply(MSE,2,mean), xlab='lambda',
     ylab='Mean Squared Error', main='LASSO')
```




```
cat(lambda_list[ind])
```

```
## 5.3
```

```
print(apply(beta[,n_lambda+1-ind,], 2, mean))
```

```
## [1] -164.90645997    3.80157982   -0.07195516   -0.68799077  -23.03211821
## [6]   -0.31891314   -0.02510893   -0.26318907  -18.75333052
```

```
print(apply(beta[,n_lambda+1-ind,], 2, median))
```

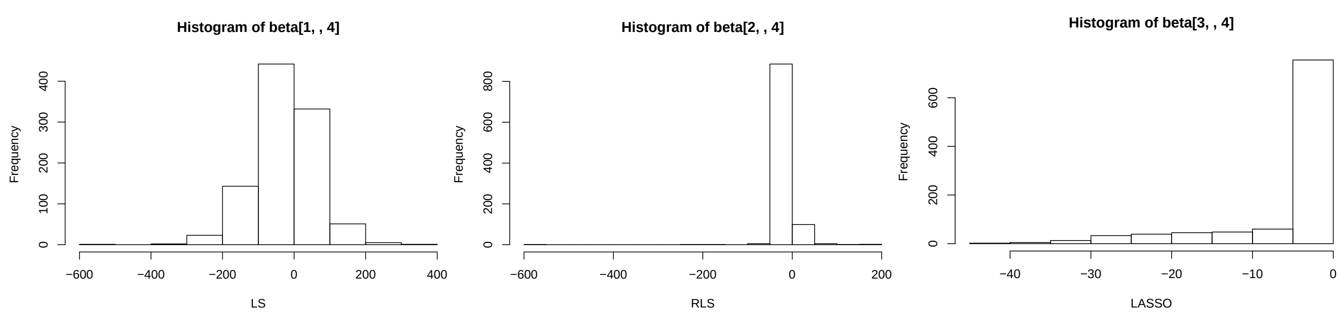
```
## [1] -164.83785    3.91882    0.00000    0.00000  -24.26899    0.00000    0.00000
## [8]    0.00000  -18.56649
```

(g) Finally, let's look at variability. Bootstrap the sample 1000 times and record β using (1) least squares, (2) Ridge (with the value of λ selected by the leave-one-out analysis for ridge), (3) Lasso (with the value of λ selected by the leave-one-out analysis for Lasso). Plot histograms of $\hat{\beta}_{\texttt{HtShoes}}$ and compare—what do you see? (Each of the three methods is its own plot.)

The bootstrapped $\hat{\beta}_{\texttt{HtShoes}}$ of the least squares can take values with large absolute values. While Ridge regression and Lasso shrink the bootstrapped $\hat{\beta}_{\texttt{HtShoes}}$ to have value closer to 0. Furthermore, there is much higher percent of bootstrapped $\hat{\beta}_{\texttt{HtShoes}}$'s to be 0 of LASSO than the one of Ridge regression.

```
n_boot <- 1000
n <- dim(seatpos)[1]
p <- dim(seatpos)[2]
beta <- array(0, c(3,n_boot,p))
lambda_list <- seq(0,20,0.1)
n_lambda <- length(lambda_list)
for(j in 1:n_boot){
```

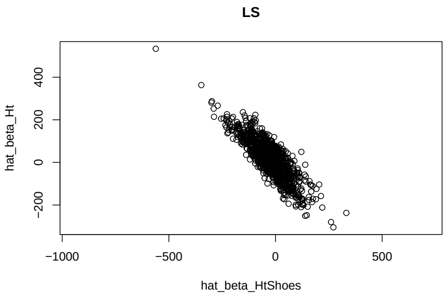
```
    boot_inds <- sample(n, n, replace=TRUE)

    model = lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg,
                     data=seatpos[boot_inds,], lambda = 0)
    beta[1,j,] <- coef(model)

    MSE <- matrix(0, n, n_lambda)
    for(i in 1:n){
        model = lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg,
                         data=seatpos[boot_inds,][-i,], lambda = lambda_list)
        y_pred <- as.matrix(cbind(const=1,seatpos[boot_inds,][i,-9])) %*% t(coef(model))
        MSE[i,] <- (seatpos[boot_inds,][i,9] - y_pred)^2
    }
    model = lm.ridge(hipcenter ~ Age+Weight+HtShoes+Ht+Seated+Arm+Thigh+Leg,
                     data=seatpos[boot_inds,],
                     lambda = lambda_list[which.min(apply(MSE,2,mean))])
    beta[2,j,] <- coef(model)

    MSE <- matrix(0, n, n_lambda)
    for(i in 1:n){
        model <- glmnet(x= as.matrix(seatpos[boot_inds,][-i,-9]),
                        y = seatpos[boot_inds,]$hipcenter[-i], lambda = lambda_list,
                        standardize=FALSE)
        betahat <- cbind(as.matrix(model$a0), t(as.matrix(model$beta)))

        y_pred <- as.matrix(cbind(const=1,seatpos[boot_inds,][i,-9])) %*% t(betahat)
        MSE[i,] <- t((seatpos[i,9] - y_pred)^2)
     }
    model <- glmnet(x= as.matrix(seatpos[boot_inds,-9]),
                    y = seatpos$hipcenter[boot_inds],
                    lambda = lambda_list[which.min(rev(apply(MSE,2,mean)))])
    beta[3,j,] <- cbind(model$a0,t(as.matrix(model$beta)))
}
hist(beta[1,,4], xlab='LS')
hist(beta[2,,4], xlab='RLS')
hist(beta[3,,4], xlab='LASSO')
```

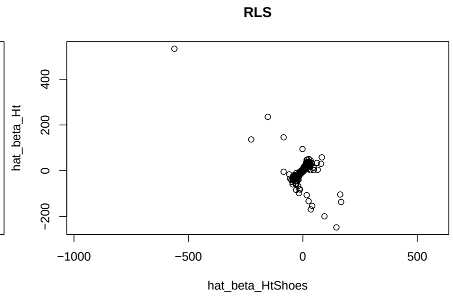


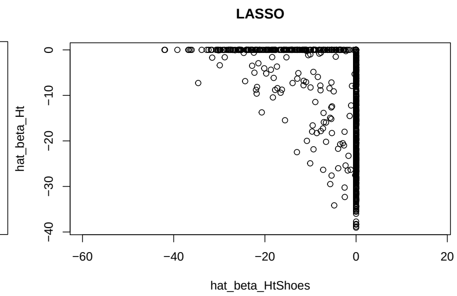
(h) Next plot scatterplots of $(\hat{\beta}_{\texttt{HtShoes}}, \hat{\beta}_{\texttt{Ht}})$. Discuss what you see for each plot, in detail. (Each of the three methods is its own plot. Each plot has 1000 points, one for each bootstrapped sample.)

In the following plots, the points of the least squares scatter in a wide range, while the points of the Ridge regression and LASSO seem to be more concentrated. What's more, the points of LASSO tends to lie right at the two axes, which means that $\hat{\beta}_{\texttt{HtShoes}}$ or $\hat{\beta}_{\texttt{Ht}}$ achieves 0 more easily than those of LS and Ridge regression.

```r
plot(beta[1,,4], beta[1,,5], xlab='hat_beta_HtShoes',
     ylab='hat_beta_Ht', main='LS', asp=1)
plot(beta[2,,4], beta[2,,5], xlab='hat_beta_HtShoes',
     ylab='hat_beta_Ht', main='RLS', asp=1)
plot(beta[3,,4], beta[3,,5], xlab='hat_beta_HtShoes',
     ylab='hat_beta_Ht', main='LASSO', asp=1)
```







2. (Faraway 8.4) Using the `trees` data, fit a model with $\log($`Volume`$)$ as the response and a second-order polynomial (including the interaction term) in `Girth` and `Height`. Determine whether the model may be reasonably simplified.

From the summary of the fit, we see that except `Grith`, the estimated coefficients of other covariates are not significant. Also, the estimated coefficient of `Girth`$*$`Height` is extremely insignificant, which implies that this term may be useless. Actually, we know that $\texttt{Volume} \propto \texttt{Girth}^2 \cdot \texttt{Height}$ and $\log \texttt{Volume} \propto 2 \log \texttt{Girth} + \log \texttt{Height}$. So it may be no interaction effect in this problem. Let's fit a reduce model with the interaction term removed.

Then, we remove the term $\texttt{Height}^2$ which has largest p value of t tests. After that , the p values of the coefficients in the reduced model are all significant. From the ANOVA result, the full model can be reasonably simplified to the reduced model.

```
data(trees)
model <- lm(log(Volume)~Height+Girth+I(Height*Girth)+I(Height^2)+I(Girth^2), trees)
summary(model)
```

```
##
## Call:
## lm(formula = log(Volume) ~ Height + Girth + I(Height * Girth) +
##     I(Height^2) + I(Girth^2), data = trees)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.159718 -0.041905 -0.003371  0.055167  0.133780
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1.9660208  2.0066922  -0.980  0.33660
## Height             0.0484196  0.0567321   0.853  0.40150
## Girth              0.2808126  0.0786856   3.569  0.00149 **
## I(Height * Girth) -0.0001975  0.0018089  -0.109  0.91395
## I(Height^2)       -0.0002022  0.0004186  -0.483  0.63326
## I(Girth^2)        -0.0042410  0.0032183  -1.318  0.19953
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08469 on 25 degrees of freedom
## Multiple R-squared:  0.9784, Adjusted R-squared:  0.9741
## F-statistic: 226.7 on 5 and 25 DF,  p-value: < 2.2e-16
```

```
reduce_model <- lm(log(Volume)~Height+Girth+I(Height^2)+I(Girth^2), trees)
summary(reduce_model)
```

```
##
## Call:
## lm(formula = log(Volume) ~ Height + Girth + I(Height^2) + I(Girth^2),
##     data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.16094 -0.04023 -0.00295  0.05474  0.13434
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) -1.9434909  1.9577536  -0.993  0.33000
## Height       0.0489426  0.0554449   0.883  0.38547
## Girth        0.2738856  0.0456299   6.002 2.45e-06 ***
## I(Height^2) -0.0002220  0.0003699  -0.600  0.55349
## I(Girth^2)  -0.0045434  0.0016059  -2.829  0.00887 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08307 on 26 degrees of freedom
## Multiple R-squared:  0.9784, Adjusted R-squared:  0.9751
## F-statistic: 294.5 on 4 and 26 DF,  p-value: < 2.2e-16
```

```
anova(reduce_model, model)
```

```
## Analysis of Variance Table
##
## Model 1: log(Volume) ~ Height + Girth + I(Height^2) + I(Girth^2)
## Model 2: log(Volume) ~ Height + Girth + I(Height * Girth) + I(Height^2) +
##     I(Girth^2)
##   Res.Df     RSS Df  Sum of Sq      F Pr(>F)
## 1     26 0.17941
## 2     25 0.17932  1 8.5465e-05 0.0119  0.914
```

```
reduce_model <- lm(log(Volume)~Height+Girth+I(Girth^2), trees)
summary(reduce_model)
```

```
##
## Call:
## lm(formula = log(Volume) ~ Height + Girth + I(Girth^2), data = trees)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.174348 -0.043284 -0.000147  0.059198  0.138282
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.783931   0.315182  -2.487  0.01935 *
## Height       0.015701   0.002759   5.690 4.80e-06 ***
## Girth        0.285333   0.040960   6.966 1.74e-07 ***
## I(Girth^2)  -0.004954   0.001435  -3.451  0.00185 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08208 on 27 degrees of freedom
## Multiple R-squared:  0.9781, Adjusted R-squared:  0.9757
## F-statistic: 402.1 on 3 and 27 DF,  p-value: < 2.2e-16
```

```
anova(reduce_model, model)
```

```
## Analysis of Variance Table
##
## Model 1: log(Volume) ~ Height + Girth + I(Girth^2)
## Model 2: log(Volume) ~ Height + Girth + I(Height * Girth) + I(Height^2) +
```

```
##     I(Girth^2)
##   Res.Df     RSS Df Sum of Sq      F Pr(>F)
## 1     27 0.18189
## 2     25 0.17932  2 0.0025722 0.1793 0.8369
```

3. You are welcome to collaborate in pairs or groups of three on this problem; if you choose to work in a group, please list your collaborators in your handed in HW.
Why does the Lasso lead to solutions β with values β_j that are exactly equal to zero? To study this, let's look at an extreme case—for very large values λ, the solution is actually all zeros. We will use the definition

$$\hat{\beta} = \arg\min \left\{ \frac{1}{2}\|Y - X\beta\|_2^2 + \lambda \sum_j |\beta_j| \right\}$$

for the Lasso. Here $X \in \mathbb{R}^{n\times p}$, $Y \in \mathbb{R}^n$, and the optimization is over $\beta \in \mathbb{R}^p$.

(a) Preliminary step: prove that for any X and Y and β,

$$|\langle Y, X\beta\rangle| \le \max_j |X_j^\top Y| \cdot \sum_j |\beta_j|,$$

where X_j is the jth column of the matrix X.

$$\begin{aligned}
|\langle Y, X\beta\rangle| &= \left|\langle Y, \sum_{j=1}^p \beta_j X_j\rangle\right| \\
&= \left|\sum_{j=1}^p \langle Y, \beta_j X_j\rangle\right| \\
&\le \sum_{j=1}^p |\langle Y, \beta_j X_j\rangle| \\
&\le \sum_{j=1}^p |\beta_j| \cdot |X_j^\top Y| \\
&\le \max_j |X_j^\top Y| \sum_{j=1}^p |\beta_j|
\end{aligned}$$

(b) Now suppose we choose an extremely large λ, satisfying $\lambda \ge \max_j |X_j^\top Y|$. Using the preliminary calculation above, prove that $\beta = \mathbf{0}_p = (0, \dots, 0)$. In other words, prove that for any $\beta \in \mathbb{R}^p$, we have

$$\mathbf{Loss}(\beta) \ge \mathbf{Loss}(\mathbf{0}_p)$$

where the loss function is

$$\mathbf{Loss}(\beta) = \frac{1}{2}\|Y - X\beta\|_2^2 + \lambda \sum_j |\beta_j|,$$

i.e. the function we're trying to minimize. If indeed $\mathbf{Loss}(\beta) \ge \mathbf{Loss}(\mathbf{0}_p)$ for every $\beta \in \mathbb{R}^p$ then this means that $\mathbf{0}_p$ is a minimizer (although we have not proved that it's the unique minimizer).

For $\lambda \geq \max_j |X_j^\top Y|$,

$$\begin{aligned}
\text{Loss}(\beta) &= \frac{1}{2}\|Y - X\beta\|_2^2 + \lambda \sum_j |\beta_j| \\
&= \frac{1}{2}Y^\top Y + \frac{1}{2}\beta^\top X^\top X\beta - \langle Y, X\beta\rangle + \lambda \sum_j |\beta_j| \\
&\geq \frac{1}{2}Y^\top Y + \frac{1}{2}\beta^\top X^\top X\beta - \max_j |X_j^\top Y| \sum_{j=1}^p |\beta_j| + \lambda \sum_j |\beta_j| \\
&= \frac{1}{2}Y^\top Y + \frac{1}{2}\beta^\top X^\top X\beta + (\lambda - \max_j |X_j^\top Y|) \sum_{j=1}^p |\beta_j| \\
&\geq \frac{1}{2}Y^\top Y \\
&= \text{Loss}(\mathbf{0}_p)
\end{aligned}$$

(c) Bonus question (this is completely optional): prove that $\mathbf{0}_p$ is the unique minimizer, i.e. if $\beta \neq \mathbf{0}_p$ then Loss$(\beta) >$ Loss$(\mathbf{0}_p)$. Hint: one way to do this is to split into cases, depending on whether $\|X\beta\|_2 = 0$ or > 0.

If $\beta \neq 0$, then $\|\beta\|_2 \neq 0$ and $\sum_{j=1}^p |\beta_j| > 0$.
If $\|X\beta\|_2 = 0$, then $X\beta = \mathbf{0}_n$,

$$\begin{aligned}
\text{Loss}(\beta) &= \frac{1}{2}Y^\top Y + \lambda \sum_j |\beta_j| \\
&> \frac{1}{2}Y^\top Y \\
&= \text{Loss}(\mathbf{0}_p)
\end{aligned}$$

If $\|X\beta\|_2 > 0$, then

$$\begin{aligned}
\text{Loss}(\beta) &\geq \frac{1}{2}Y^\top Y + \frac{1}{2}\beta^\top X^\top X\beta + (\lambda - \max_j |X_j^\top Y|) \sum_{j=1}^p |\beta_j| \\
&> \frac{1}{2}Y^\top Y \\
&= \text{Loss}(\mathbf{0}_p)
\end{aligned}$$

Therefore, $\mathbf{0}_p$ is the unique minimizer.