# MSSC 6250 Machine Learning Homework 2

Ridge, Lasso, and Splines

Henri Medeiros Dos Reis

- Deadline: **Friday, Mar 3 11:59 PM**
- Homework presentation date: **Tuesday, Mar 7**
- Please submit your work in **one PDF** file to **D2L** > **Assessments** > **Dropbox**. *Multiple files or a file that is not in pdf format are not allowed.*
- Any relevant code should be attached.
- Read **ISL** Chapter 5.1, 6.2, and 7.

## Exercises required for all students

1. **ISL** Sec. 5.4: 3

**Solution:**

*a-)* k-fold cross-validation is done by taking the set of observations and splitting into k non-overlapping random groups. Each one of these groups then are used as a validation, and the one left as the training set. Then to estimate the test error, we do the average of the k MSE estimates.

*b-)*

i. The validation set strategy is easier to implement, since it only requires splitting the training data into two groups.However, depending on which observations are included in the training and validation sets, the estimate of the test error rate might vary a lot.

ii. LOOCV is a specific instance of k-fold cross-validation. LOOCV is the approach that is the most expensive computationally wise, since the model needs to be fit n times. Besides that, LOOCV has lower bias but larger variance than the common k-fold cross-validation.

2. **ISL** Sec. 6.6: 3

**Solution:**

*a-)*

i. Decrease steadily. As s increases the constraint on beta decreases until it is non-existent. Then the RSS reduces until it becomes the least squares answer.

*b-)*

ii. Decrease initially, and then eventually start increasing in a U shape. When $s = 0$, all $\beta$ s are $0$, and we have the simplest model possible, which will have higher RSS. As we increase $s$, $beta$ s become non-zero values and model starts fitting well on test data, which makes RSS decreases. And if we continue increasing $s$ we will end-up overfitting.

*c-)*

iii. Steadily increase. When $s = 0$ the model is just the intercept, which has very small variance. Once the values of $\beta$s start to increase, we will get higher and higher values for variance, since it will start to overfit the model.

*d-)*

iv. Steadily decrease. When $s = 0$ the model is just the intercept, and likely to be far from the true value. Thus bias is high. As $s$ increases, more $\beta$ s become non-zero, making our model better at fitting the training data, which then decreases the bias.

*e-)*

v. Remains constant. The error is irreducible, since it is not dependent on the model, so changing the the $\beta$s values, by changing $s$, will not increase or decrease the irreducible error.

3. **ISL** Sec. 6.6: 4

**Solution:**

*a-)*

iii. Steadily increase. As $\lambda$ increases, $\beta$s decreases from the OLS $\beta$s until it reaches $0$. Then the RSS will increase until it reaches a maximum value, when $\beta$s reach $0$.

*b-)*

ii. Decrease initially, and then eventually start increasing in a U shape. When $\lambda = 0$, all $\beta$ s are the same as OLS. When lambda increases, the fit's flexibility decreases, which lowers the variance of predictions for causing an increase in bias. The test RSS will initially fall as a result of the improved prediction accuracy. Predictions will become more skewed when lambda rises above the ideal point because there is a considerably higher since we will start to overfit.

*c-*

iv. Steadily decreases. When $\lambda = 0$, The actual estimates depend more on the training data, which means that variance is high. when $\lambda$ increases, $\beta$ s start decreasing and model is simplified. The larger $\lambda$ grows, all $\beta$s start to reach 0, which predictions are constant with no variance.

*d-)*

iii. Steadily increases. When $\lambda = 0$, we have a OLS, which mean the least bias. When $\lambda$ increases, $\beta$ s start going towards zero, the model fits less accurately to training data, which causes bias to increase. The larger $\lambda$ grows, all $\beta$s start to reach 0, which predictions are constant and bias is the largest.

*e-)*

v. Remains constant. The error is irreducible, since it is not dependent on the model, so changing the the $\beta$s values, by changing $\lambda$, will not increase or decrease the irreducible error.

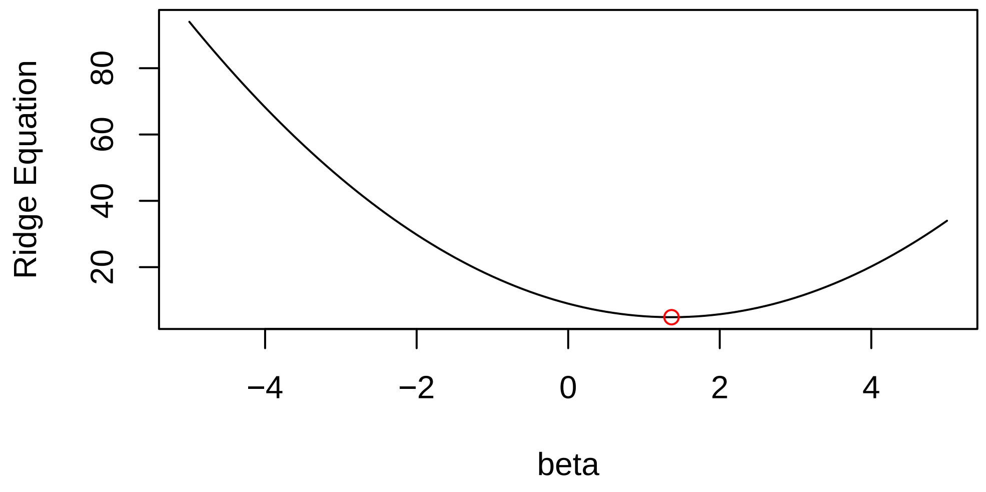
4. **ISL** Sec. 6.6: 6

**Solution:**

*a-)*

When $p = 1$, (6.12) is equal to $(y - \beta)^2 + \lambda\beta^2$.

```
y = 3
beta = seq(-5,5,0.1)
lambda = 1.2

eq_6.12 = (y - beta)2 + lambda*(beta2)


est_beta_6.14 = y/(1 + lambda)
est_val = (y - est_beta_6.14)2 + lambda*(est_beta_6.142)

plot(beta, eq_6.12, xlab="beta", ylab="Ridge Equation",type="l")
points(est_beta_6.14, est_val,col="red",type ="p")
```

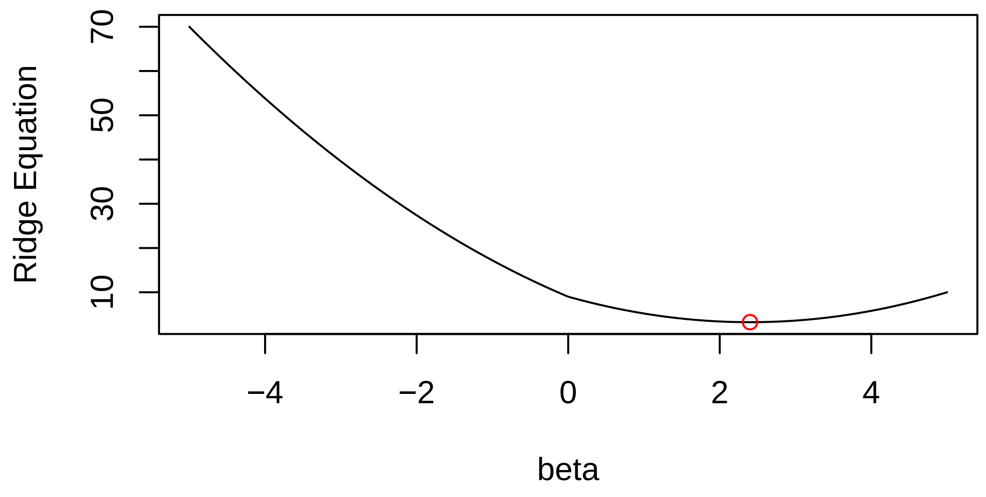


As we can see, the red point show that the minimum value occurs at $\beta = y/(1+\lambda)$.

*b-)*

For $p = 1$, (6.13) is equal to $(y-\beta)^2 + \lambda|\beta|$.

```
y = 3
beta = seq(-5,5,0.1)
lambda = 1.2

eq_6.13 = (y - beta)2 + lambda * abs(beta)


est_beta_6.15 = y-lambda/2
est_val = (y - est_beta_6.15)2 + lambda*abs(est_beta_6.15)

plot(beta, eq_6.13, xlab="beta", ylab="Ridge Equation",type="l")
points(est_beta_6.15, est_val,col="red",type ="p")
```



As we can see, the red point show that the minimum value occurs at $\beta = y - \lambda/2$.

5. **ISL** Sec. 6.6: 9 (a)-(d)

**Solution:**

*a-)*

```r
set.seed(1)
College <- read.csv("College.csv")
train_size <- round(dim(College)[1]*0.8)

train <- sample(1:dim(College)[1], train_size)
# make it negative, so it includes everything
# that is not on the train
test <- -train
College_train <- College[train, ]
College_test <- College[test, ]
#take out the name of  the universities
College_test <- College_test[-1]
College_train <- College_train[-1]
```

*b-)*

```r
lin_model <- lm(Apps~., data=College_train)
my_pred <- predict(lin_model, College_test)
```

```r
mean((College_test[, "Apps"] - my_pred)**2)
```

```
[1] 1578073
```

The test error is 1578073.

*c-)*

```r
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.2.2

Loading required package: Matrix

Loaded glmnet 4.1-6
```

```r
ridge_cv_fit <- cv.glmnet(x = data.matrix(College_train[-2]),
                          y = College_train[, "Apps"],
                          alpha = 0,
                          nfolds = 10)

best_lambda <- ridge_cv_fit$lambda.min

my_ridge_pred <- predict(ridge_cv_fit,
                         newx= data.matrix(College_test[-2]),
                         s=best_lambda)

mean((College_test[, "Apps"] - my_ridge_pred)**2)
```

```
[1] 1447148
```

The test error is 1447148.

*d-)*

```r
lasso_cv_fit <- cv.glmnet(x = data.matrix(College_train[-2]),
                          y = College_train[, "Apps"],
                          alpha = 1,
                          nfolds = 10)

best_lambda_lasso <- lasso_cv_fit$lambda.min
```

```r
my_lasso_pred <- predict(lasso_cv_fit,
                         newx= data.matrix(College_test[-2]),
                         s=best_lambda_lasso)

mean((College_test[, "Apps"] - my_lasso_pred)**2)
```

```
[1] 1565220
```

The test error is 1565220.

```r
coef(lasso_cv_fit, s = "lambda.min")
```

```
18 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept) -263.03842516
Private     -381.59445521
Accept         1.67633178
Enroll        -1.10351481
Top10perc     49.03123900
Top25perc    -12.50569717
F.Undergrad    0.06871478
P.Undergrad    0.06463208
Outstate      -0.07283358
Room.Board     0.13968385
Books          0.19871129
Personal       0.01595598
PhD           -9.63192851
Terminal      -0.22846447
S.F.Ratio     17.24863884
perc.alumni    0.59060285
Expend         0.05703165
Grad.Rate      5.61816590
```

6. **ISL** Sec. 7.9: 9

**Solution:**

*a-)*

```r
library(ISLR2)
```

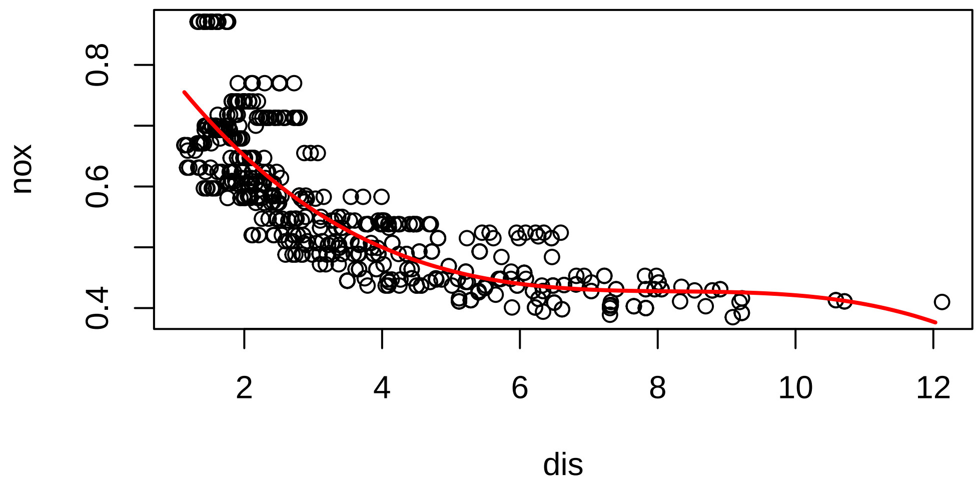
```
Warning: package 'ISLR2' was built under R version 4.2.2
```

```
Attaching package: 'ISLR2'


The following object is masked _by_ '.GlobalEnv':

    College
```

```r
boston_lm_fit <- lm(nox~poly(dis, 3), data=Boston)
summary(boston_lm_fit)
```

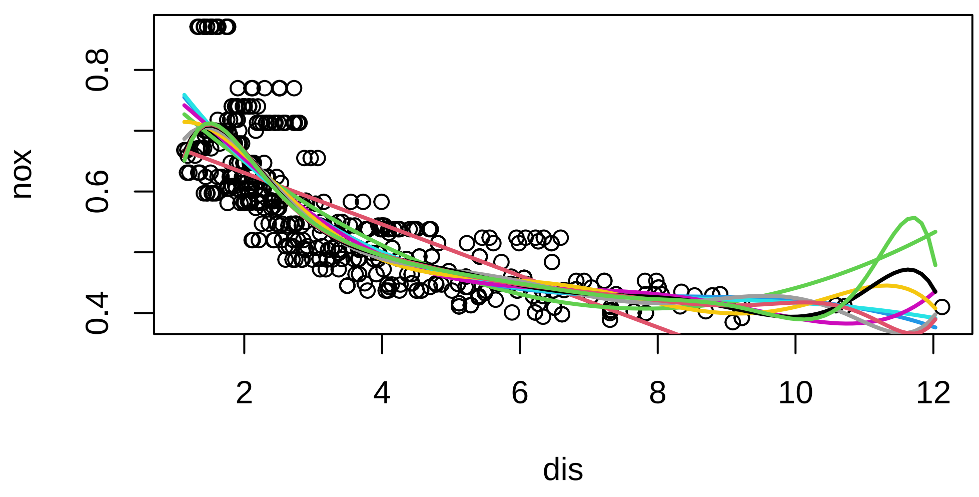
```

Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.121130 -0.040619 -0.009738  0.023385  0.194904 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,    Adjusted R-squared:  0.7131 
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

```r
dis_lim <- range(Boston["dis"])
dis_seq <- seq(from=dis_lim[1], to=dis_lim[2], by=0.1)

my_boston_pred <- predict(boston_lm_fit, list(dis=dis_seq))
plot(nox~dis, data=Boston)
lines(dis_seq, my_boston_pred, col="red", lwd=2)
```



*b-)*

```r
rss <- c()
my_fits <- c()
preds <- c()
for (i in 1:10) {
  curr_mod <-  lm(nox~poly(dis, i), data=Boston)
  my_fits <- c(my_fits, curr_mod)
  rss <- c(rss, sum(curr_mod$residuals**2))
  preds <- c(preds, predict(curr_mod, list(dis=dis_seq)))
}

plot(nox~dis, data=Boston)
indexes <- seq(1,110)

plot(nox~dis, data=Boston)
for(i in 1:10){
  lines(dis_seq, preds[indexes], col=i+1, lwd=2)
  indexes <- indexes+110
}
```

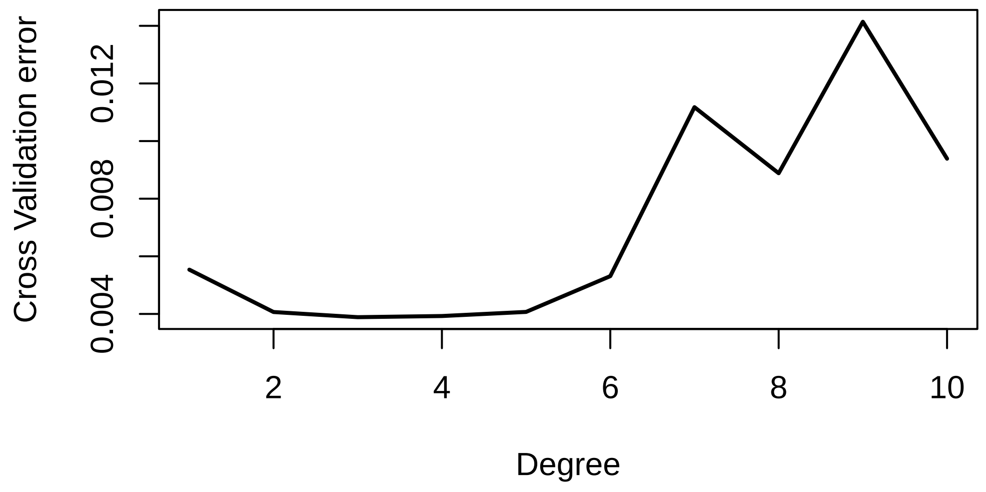


```
rss
```

```
 [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630
 [9] 1.833331 1.832171
```

*c-)*

```
library(boot)

delt <- c()
for (i in 1:10) {
  glm.fit <- glm(nox~poly(dis, i), data=Boston)
  delt <- c(delt, cv.glm(Boston, glm.fit, K=10)$delta[2])
}
plot(1:10, delt, xlab="Degree", ylab="Cross Validation error",
     type="l", lwd=2)
```



Using a 10-gold CV, we can see that the error decreases when using degrees from 1 to 3, and increases slowly from 3 until 6. So the best degree would be 3.

*d-)*

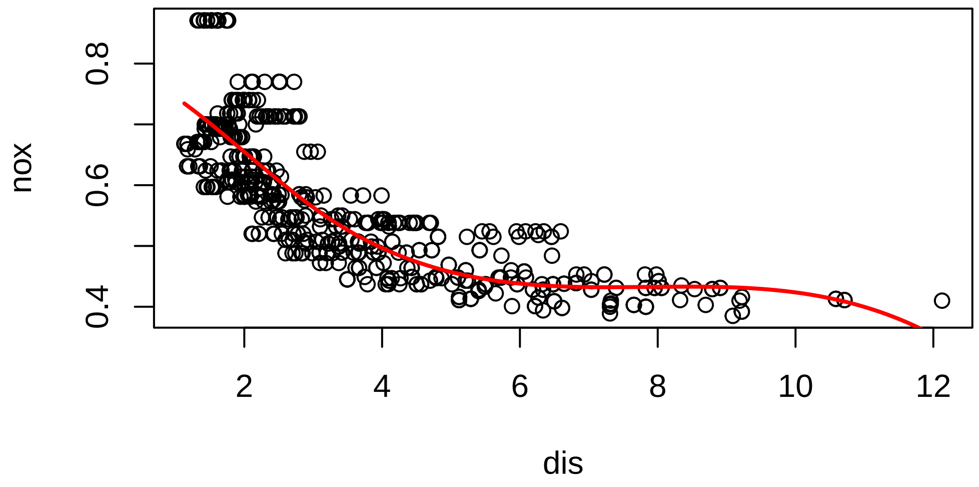
```
library(splines)
splines_mod <- lm(nox~bs(dis,degree = 3, df=4), data=Boston)
summary(splines_mod)
```

```
Call:
lm(formula = nox ~ bs(dis, degree = 3, df = 4), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.124622 -0.039259 -0.008514  0.020850  0.193891 

Coefficients:
                             Estimate Std. Error t value Pr(>|t|)    
(Intercept)                   0.73447    0.01460  50.306  < 2e-16 ***
bs(dis, degree = 3, df = 4)1 -0.05810    0.02186  -2.658  0.00812 ** 
bs(dis, degree = 3, df = 4)2 -0.46356    0.02366 -19.596  < 2e-16 ***
bs(dis, degree = 3, df = 4)3 -0.19979    0.04311  -4.634 4.58e-06 ***
bs(dis, degree = 3, df = 4)4 -0.38881    0.04551  -8.544  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.06195 on 501 degrees of freedom
Multiple R-squared:  0.7164,    Adjusted R-squared:  0.7142
F-statistic: 316.5 on 4 and 501 DF,  p-value: < 2.2e-16
```

The bs function chooses the knots automatically, since the degrees of freedom is already specified.

```
splines_pred <- predict(splines_mod, list(dis=dis_seq))
plot(nox~dis, data=Boston)
lines(dis_seq, splines_pred, col="red", lwd=2)
```



*e-)*

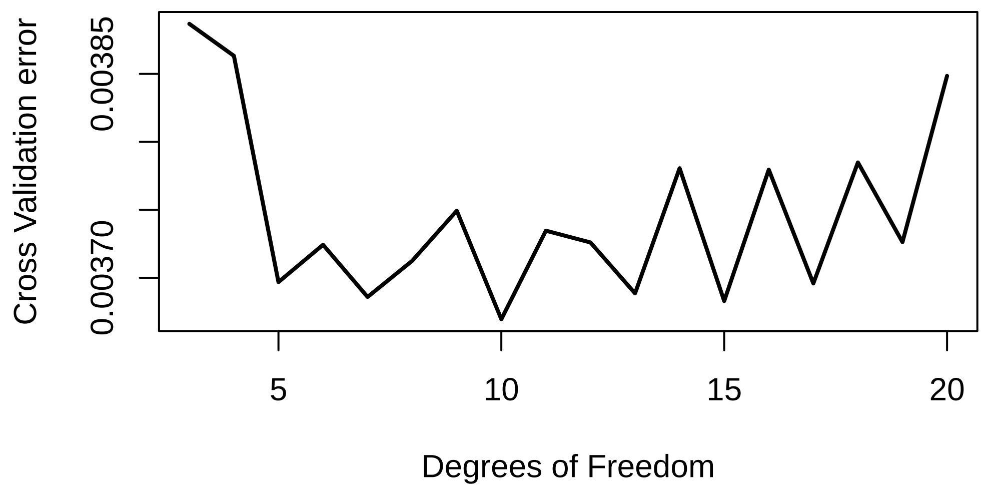
```
cross_df <-c()
for (i in 3:20) {
  my_model <- lm(nox~bs(dis,degree = 3, df=i), data=Boston)
  cross_df <- c(cross_df,sum(my_model$residuals**2))
}
cross_df
```

```
 [1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535
 [9] 1.796992 1.788999 1.782350 1.781838 1.782798 1.783546 1.779789 1.775838
[17] 1.774487 1.776727
```

The train RSS decreases from 3 to 9 degrees of freedom, then it increases for 10 , decreases until 19 degrees of freedom, then it increases with 20.

*f-)*

```r
cross_val <- c()
for (i in 3:20) {
  lm.fit <- glm(nox~bs(dis,degree = 3, df=i), data=Boston)
  cross_val <- c(cross_val,cv.glm(Boston, lm.fit, K=10)$delta[2])
}
plot(3:20, cross_val, lwd=2, type="l",
     xlab="Degrees of Freedom", ylab="Cross Validation error")
```



The best score by this cross validation with 10-fold is 13 degrees of freedom. However, this version is not as stable, there are a lot of ups and downs in the plot.

7. Suppose for a linear regression problem with $n = p$, $\mathbf{X} = \mathbf{I}$ and no intercept. Show that

   (a) The least squares problem can be simplified to finding $\beta_1, \ldots, \beta_p$ that minimize $\sum_{j=1}^{p} (y_j - \beta_j)^2$. What is least squares estimator $b_j$?

   (b) The ridge estimator is $\hat{\beta}_j^r = \frac{y_j}{1+\lambda} = \arg\min \sum_{j=1}^{p} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$.

   (c) The Lasso solution of $\sum_{j=1}^{p} (y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$ is

$$\hat{\beta}_j^l = \begin{cases} y_j - \lambda/2 & \text{if } y_j > \lambda/2 \\ y_j + \lambda/2 & \text{if } y_j < -\lambda/2 \\ 0 & \text{if } |y_j| \leq \lambda/2 \end{cases}$$

**Solution:**

*a-)*

The least squares problem $\min \|y - X\beta\|_2^2$, since $X = I$ and no intercept, then

$$\min \|y - \beta\|_2^2 = \min \sum_{i=1}^{n} (y_i - \beta_i)^2$$

Then by using the general solution of least squares, we have

$$b = (X^T X)^{-1} X^T y = Iy = y$$

So, to find the jth element $b_j$, we multiply on the left with the standard basis vector $e_j$

$$e_j^T b = e_j^T y \Rightarrow b_j = y_j$$

*b-)*

The ridge regression $\min \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2, y > 0$. Since $X = I, n = p$ and no intercept. Then

$$\min \|y - \beta\|_2^2 + \lambda \|\beta\|_2^2 = \min \sum_{i=1}^{n} (y_i - \beta_i)^2 + \sum_{i=1}^{n} \beta_i^2$$

Now we take the derivative with respect to $\beta_i$

$$\frac{\partial}{\partial \beta_i} (y_i^2 - 2y_i\beta_i + \beta_i^2 + \lambda\beta_i^2) = -2y_i + 2\beta_i + 2\lambda\beta_i$$

Setting it equal to zero and solving for $\beta_i$ gives

$$-y_i + \beta_i + \lambda\beta_i = 0 \Rightarrow \beta_i = \frac{y_i}{1 + \lambda}$$

*c-)*

The lasso estimator $\min \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$. Since $X = I, n = p$ and no intercept

$$\min \|y - \beta\|_2^2 + \lambda \|\beta\|_1 = \min \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n} |\beta_i|$$

Taking the derivative with respect to $\beta_i$

$$\frac{\partial}{\partial \beta_i} (y_i^2 - 2y_i\beta_i + \beta_i^2 + \lambda\beta_i) = -2y_i + 2\beta_i + \lambda$$

Setting it equal to zero and solving for i gives

$$2(\beta_i - y_i) + \lambda = 0 \Rightarrow \beta_i = \frac{-\lambda}{2} + y_i$$

Then

$$\beta_i = \begin{cases} y_i - \lambda/2 & \text{if } y_i > \lambda/2 \\ y_i + \lambda/2 & \text{if } y_i < -\lambda/2 \\ 0 & \text{if } |y_i| \leq \lambda/2 \end{cases}$$

## Exercises required for MSSC PhD students

1. **ISL** Sec. 6.6: 5

**Solution:**

*a-)*

The general Ridge Regression is the minimization of $\sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_j)^2 + \lambda \sum_{i=1}^{p}\hat{\beta}_i^2$

If we have, $\hat{\beta}_0 = 0$ and $n = p = 2$. we can write the minimization problem as $(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$

*b-)*

Since we have $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$. We can take derivatives of the expression in part a with respect to both $\hat{\beta}_1$ and $\hat{\beta}_2$ and setting them equal to zero so that we minimize.

$$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2) =$$
$$\lambda(\hat{\beta}_1^2+\hat{\beta}_2^2)-2\hat{\beta}_1 x_1 y_1-2\hat{\beta}_2 x_1 y_1-2\hat{\beta}_1 x_2 y_2-2\hat{\beta}_2 x_2 y_2+\hat{\beta}_1^2 x_1^2+\hat{\beta}_2^2 x_1^2+2\hat{\beta}_1\hat{\beta}_2 x_1^2+\hat{\beta}_1^2 x_2^2+\hat{\beta}_2^2 x_2^2+2\hat{\beta}_1\hat{\beta}_2 x_2^2+y_1^2+y_2^2$$

Then taking the derivative with respect to $\hat{\beta}_1$ and setting it equal to 0 we have

$$2\lambda\hat{\beta}_1 - 2x_1 y_1 - 2x_2 y_2 + 2\hat{\beta}_1 x_1^2 + 2\hat{\beta}_1 x_2^2 + 2\hat{\beta}_2 x_2^2 + 2\hat{\beta}_2 x_1^2 = 0$$

Solving for $\hat{\beta}_1$

$$\hat{\beta}_1 = \frac{x_1 y_1 + x_2 y_2 - \hat{\beta}_2(x_1^2 + x_2^2)}{\lambda + x_1^2 + x_2^2}$$

Now, let's repeat the process for $\hat{\beta}_2$

$$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2) =$$
$$\lambda(\hat{\beta}_1^2+\hat{\beta}_2^2)-2\hat{\beta}_1 x_1 y_1-2\hat{\beta}_2 x_1 y_1-2\hat{\beta}_1 x_2 y_2-2\hat{\beta}_2 x_2 y_2+\hat{\beta}_1^2 x_1^2+\hat{\beta}_2^2 x_1^2+2\hat{\beta}_1\hat{\beta}_2 x_1^2+\hat{\beta}_1^2 x_2^2+\hat{\beta}_2^2 x_2^2+2\hat{\beta}_1\hat{\beta}_2 x_2^2+y_1^2+y_2^2$$

Then taking the derivative with respect to $\hat{\beta}_1$ and setting it equal to 0 we have

$$2\lambda\hat{\beta}_2 - 2x_1 y_1 - 2x_2 y_2 + 2\hat{\beta}_2 x_1^2 + 2\hat{\beta}_2 x_2^2 + 2\hat{\beta}_1 x_2^2 + 2\hat{\beta}_1 x_1^2 = 0$$

Solving for $\hat{\beta}_1$

$$\hat{\beta}_2 = \frac{x_1 y_1 + x_2 y_2 - \hat{\beta}_1(x_1^2 + x_2^2)}{\lambda + x_1^2 + x_2^2}$$

Therefore, since the expressions for both are the same, then $\hat{\beta}_1 = \hat{\beta}_2$.

*c-)*

The general Ridge Regression is the minimization of $(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|)$

*d-)*

we can use a geometric interpretation of the solutions for the equation above. We can use the alternate form of Lasso constraint $|\hat{\beta}_1| + |\hat{\beta}_2| < s$. Which if we plot, take the shape of a diamond centered at origin. Next consider the squared optimization constraint $(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2$, which is the part of the equation we have not used yet. Since $x_{11} = x_{12}$, $x_{21} = x_{22}$, $x_{11} + x_{21} = 0$, $x_{12} + x_{22} = 0$ and $y_1 + y_2 = 0$ we can rewrite the problem as the minimization of

$$2(y_1 - (\hat{\beta}_1 + \hat{\beta}_2)x_{11})^2$$

If we let $b = \hat{\beta}_1 + \hat{\beta}_2$,

$$2(y_1 - (\hat{\beta}_1 + \hat{\beta}_2)x_{11})^2 = 2(y_1 - bx_{11})^2 = 2y_1^2 - 4y_1 b x_{11} + 2b^2 x_{11}^2$$

and take the derivative with respect to b, and set it equal to 0, we have

$$4yx_{11} + 4bx_{11}^2 = 0 \Rightarrow b = \frac{-4yx_{11}}{4x_{11}^2} = \frac{-y}{x_{11}}$$

$$\hat{\beta}_1 + \hat{\beta}_2 = \frac{-y}{x_{11}}$$

Therefore

$$\hat{\beta}_1 = \frac{-y}{x_{11}} - \hat{\beta}_2$$

$$\hat{\beta}_2 = \frac{-y}{x_{11}} - \hat{\beta}_1$$

This line is parallel to the edge of the Lasso diamond. So the solutions we found by minimizing, are limes of the functions $\hat{\beta}_2 = \frac{-y}{x_{11}} - \hat{\beta}_1$ and $\hat{\beta}_1 = \frac{-y}{x_{11}} - \hat{\beta}_2$ that touch the Lasso diamond at different points. As a result, the entire edge $\hat{\beta}_1 + \hat{\beta}_2 = s$ are solutions to the Lasso optimization problem.

Now solutions to the original Lasso optimization problem are contours of the function $\hat{\beta}_1 + \hat{\beta}_2 = \frac{-y}{x_{11}}$ that touch the Lasso-diamond $\hat{\beta}_1 + \hat{\beta}_2 = s$. Finally, as $\hat{\beta}_1$ and $\hat{\beta}_2$ very along the line $\hat{\beta}_1 + \hat{\beta}_2 = \frac{y_1}{x_{11}}$, these contours touch the Lasso-diamond edge $\hat{\beta}_1 + \hat{\beta}_2 = s$ at different points. As a result, the entire edge $\hat{\beta}_1 + \hat{\beta}_2 = s$ are solutions to the Lasso optimization problem.

Therefore, the Lasso problem does not have a unique solution.

2. **ISL** Sec. 7.9: 11

**Solution:**

*a-)*

```r
n=100
x1 <- rnorm(n)
x2 <- rnorm(n)
eps <- rnorm(n)

y <- 2 + 4.2*x1 +2.4*x2 + eps
```

*b-)*

```r
beta_1 <- 10000
```

*c-)*

```r
a1 <- y - beta_1 * x1
beta_2 <- lm(a1 ~ x2)$coef[2]
```

*d-)*

```r
a2 <- y - beta_2 * x2
beta_1 <- lm(a2 ~ x1)$coef[2]
```

*e-)*

```r
beta_0s <- c()
beta_1s <- c()
beta_2s <- c()

# do the first time manually
a2 <- y - beta_1 * x1
beta_2s[1] <- lm(a2 ~ x2)$coef[2]
a1 <- y - beta_2s[1] * x2
lm_coef <- lm(a1 ~ x1)$coef
beta_1s[1] <- lm_coef[2]
beta_0s[1] <- lm_coef[1]


for (i in 2:1000)
{
  a2 <- y - beta_1s[i-1] * x1
  beta_2s[i] <- lm(a2 ~ x2)$coef[2]

  a1 <- y - beta_2s[i-1] * x2
  lm_coef <- lm(a1 ~ x1)$coef
  beta_1s[i] <- lm_coef[2]
```

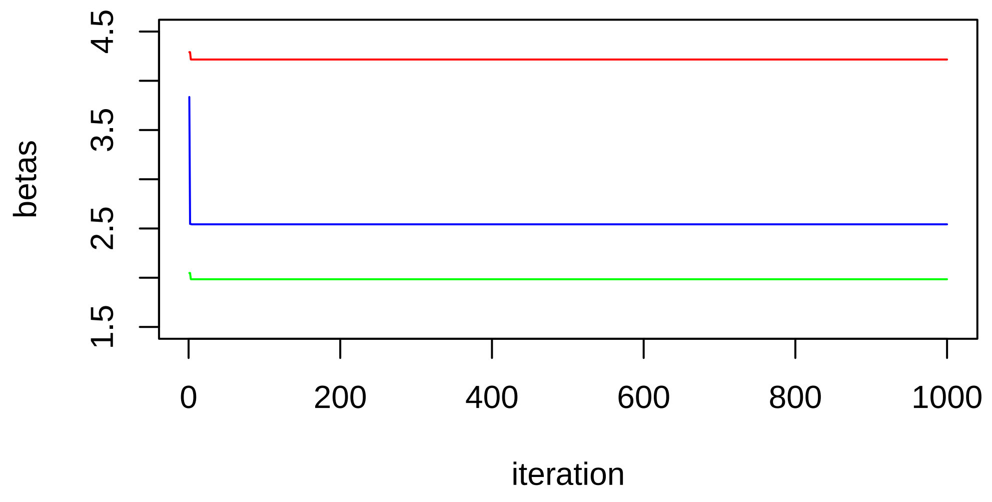
```
    beta_0s[i] <- lm_coef[1]
  }

  lm_coef <- lm(y ~ x1 + x2)$coefficients
  lm_coef
```

```
(Intercept)          x1          x2 
   1.984562    4.216038    2.542486 
```

```
  data.frame( c(beta_0s[1000], beta_1s[1000], beta_2s[1000]),
              as.numeric(lm_coef))
```

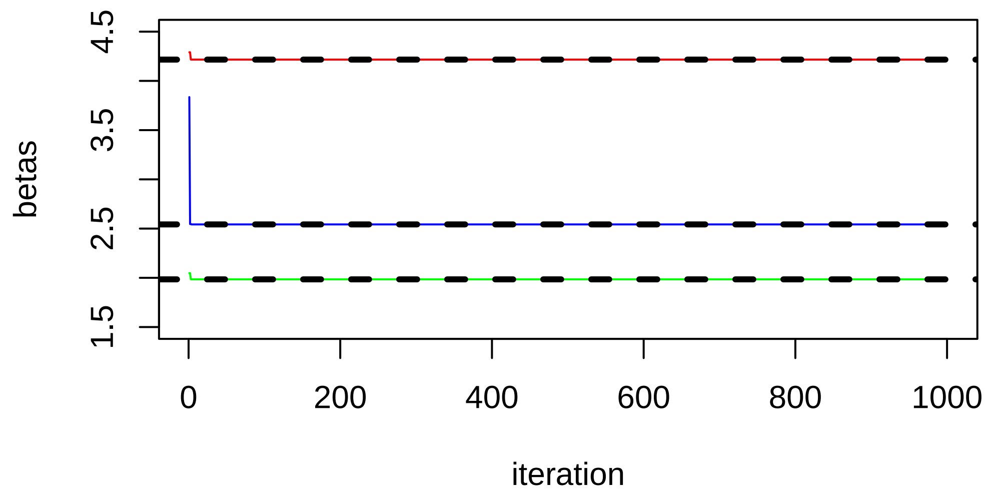
```
  c.beta_0s.1000...beta_1s.1000...beta_2s.1000.. as.numeric.lm_coef.
1                                       1.984562           1.984562
2                                       4.216038           4.216038
3                                       2.542486           2.542486
```

```
  plot(1:1000, beta_0s, type="l", xlab="iteration",
       ylab="betas",ylim=c(1.5, 4.5), col="green")
  lines(1:1000, beta_1s, col="red")
  lines(1:1000, beta_2s, col="blue")
```



*f-)*

```r
m_lin_model <- lm(y~x1+x2)
plot(1:1000, beta_0s, type="l", xlab="iteration",
     ylab="betas",ylim=c(1.5, 4.5), col="green")
lines(1:1000, beta_1s, col="red")
lines(1:1000, beta_2s, col="blue")
abline(h=m_lin_model$coef[1], lty=2, lwd=3, col="black")
abline(h=m_lin_model$coef[2], lty=2, lwd=3, col="black")
abline(h=m_lin_model$coef[3], lty=2, lwd=3, col="black")
```



It is possible to see that the multiple regression coefficients (dotted lines) are the same as the coefficients from back fitting.

g-)

```r
print(beta_0s[1:14])
```

```
[1] 2.048334 2.048334 1.984736 1.984736 1.984562 1.984562 1.984562 1.984562
 [9] 1.984562 1.984562 1.984562 1.984562 1.984562 1.984562
```

```r
print(beta_1s[1:14])
```

```
[1] 4.291008 4.291008 4.216243 4.216243 4.216038 4.216038 4.216038 4.216038
 [9] 4.216038 4.216038 4.216038 4.216038 4.216038 4.216038
```

```r
print(beta_2s[1:14])
```

```
 [1] 3.835490 2.546027 2.546027 2.542496 2.542496 2.542486 2.542486 2.542486
 [9] 2.542486 2.542486 2.542486 2.542486 2.542486 2.542486
```

We can see that it took around 10 iterations to converge.

3. Define the objective of the **elastic net** problem $J_1(\boldsymbol{\beta}) = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_2\|\boldsymbol{\beta}\|_2^2 + \lambda_1\|\boldsymbol{\beta}\|_1$ and the objective of Lasso $J_2(\boldsymbol{\beta}) = \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\boldsymbol{\beta}\|^2 + c\lambda_1\|\boldsymbol{\beta}\|_1$ where $c = (1+\lambda_2)^{-1/2}$, and

$$\tilde{\mathbf{X}} = c\begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2}\mathbf{I}_p \end{pmatrix}, \quad \tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ \mathbf{0}_{p\times 1} \end{pmatrix}.$$

Show that $J_1(c\boldsymbol{\beta}) = J_2(\boldsymbol{\beta})$. Therefore the elastic net problem can be solved using algorithms for Lasso on modified data.

**Solution:**

To show that $J_1(c\beta) = J_2(\beta)$, we substitute

$$\begin{gathered}
J_2(\beta) = \|\tilde{y} - \tilde{X}\beta\|_2^2 + c\lambda_1\|\beta\|_1 \\
= \| \begin{bmatrix} y \\ 0_{p\times 1} \end{bmatrix} - c\begin{bmatrix} X \\ \sqrt{\lambda_2}I_p \end{bmatrix} \beta\|_2^2 + c\lambda_1\|\beta\|_1 \\
= \| \begin{bmatrix} y - cX\beta \\ -c\sqrt{\lambda_2}I_p\beta \end{bmatrix} \|_2^2 + c\lambda_1\|\beta\|_1 \\
\\
= \|y - cX\beta\|_2^2 + \|c\sqrt{\lambda_2}\beta\|_2^2 + \lambda_1\|c\beta\|_1 \\
= \|y - Xc\beta\|_2^2 + \lambda_2\|c\beta\|_2^2 + \lambda_1\|c\beta\|_1 \\
= J_1(c\beta)
\end{gathered}$$

Therefore the elastic net problem can be solved using algorithms for Lasso on modified data.

## Optional Exercises

Let the ridge estimator be $\hat{\boldsymbol{\beta}}^r = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$. Show that

1. $\hat{\boldsymbol{\beta}}^r = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}'\mathbf{X})\mathbf{b}$, and hence $\hat{\beta}_j^r = \frac{b_j}{1+\lambda}$ is a special case when $\mathbf{X}'\mathbf{X} = \mathbf{I}$.

2. The bias $\mathrm{E}\left(\hat{\boldsymbol{\beta}}^r\right) - \boldsymbol{\beta} = \left[(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}'\mathbf{X}) - \mathbf{I}\right]\boldsymbol{\beta}$, and hence $\mathrm{Bias}(\hat{\beta}_j^r) = \frac{-\lambda}{1+\lambda}\beta_j$ when $\mathbf{X}'\mathbf{X} = \mathbf{I}$.

3. $\sum_{j=1}^{p}\left(\hat{\beta}_j^r - \beta_j\right)^2 = \lambda^2\boldsymbol{\beta}'\left(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}\right)^{-2}\boldsymbol{\beta} = \sum_{j=1}^{p}\frac{\lambda^2}{(d_j^2+\lambda)^2}\beta_j^2$.

4. $\text{Var}\left(\hat{\beta}_j^r\right) = \sigma^2\frac{d_j^2}{\left(d_j^2+\lambda\right)^2}$ where $d_j$ is the $j$-th singular value of $\mathbf{X}$. Hence $\text{Var}\left(\hat{\beta}_j^r\right) = \sigma^2\frac{1}{(1+\lambda)^2}$ when $\mathbf{X}'\mathbf{X} = \mathbf{I}$.

5. $\hat{\mathbf{y}}^r = \mathbf{X}\hat{\boldsymbol{\beta}}^r = \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y} = \sum_{j=1}^{p}\mathbf{u}_j\left(\frac{d_j^2}{d_j^2+\lambda}\mathbf{u}_j'\mathbf{y}\right)$ where $\mathbf{u}_j$ is the $j$-th column of the SVD of $\mathbf{X} = \mathbf{UDV}'$. Hence, a greater amount of shrinkage is applied to the coordinates of basis vectors with smaller $d_j^2$.

6. The trace of the projection matrix $\mathbf{H}_\lambda = \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'$ defines the **effective degrees of freedom**. In general, $\text{tr}(H) = \sum_{j=1}^{p}\frac{d_j^2}{d_j^2+\lambda}$. For linear regression where $\lambda = 0$, $\text{tr}(H) = p$.