HW1

Benjamin Panny

2023-11-12

Theory

Derive the solutions of OLS and ridge regression:

Prove OLS estimator formula

a. In OLS, we derive

$$\min_{\beta}(Y - X\beta)^T(Y - X\beta)$$

which is the sum of squared residuals.

Prove

$$\hat{\beta}^{OLS} = (X^TX)^{-1}X^TY$$

hint, for given A:

$$\frac{d}{d\beta}(A\beta) = A$$

$$\frac{d}{d\beta}(\beta^TA\beta) = 2A\beta$$

$$\begin{aligned}
e^Te &= (Y - X\hat{\beta})^T(Y - X\hat{\beta}) \\
&= (-\hat{\beta}^TX^T + Y^T)(Y - X\hat{\beta}) \\
&= -\hat{\beta}^TX^TY + \hat{\beta}^TX^TX\hat{\beta} + Y^TY - Y^TX\hat{\beta} \\
&= -2\hat{\beta}^TX^TY + \hat{\beta}^TX^TX\hat{\beta} + Y^TY \\
\frac{de^Te}{d\hat{\beta}} &= -2X^TY + 2X^TX\hat{\beta} \\
&-> \text{set to } 0 \\
2X^TY &= 2X^TX\hat{\beta} \\
X^TY &= X^TX\hat{\beta} \\
(X^TX)^{-1}X^TX\hat{\beta} &= (X^TX)^{-1}X^TY \\
I\hat{\beta} &= (X^TX)^{-1}X^TY \\
\hat{\beta} &= (X^TX)^{-1}X^TY
\end{aligned}$$

Prove Ridge Regression Estimator Formula

b. In ridge regression, we aim on

$$\min_{\beta}(Y - X\beta)^T(Y - X\beta) + \lambda\beta^T\beta$$

which is the sum of squared residuals.

Prove

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$

$$\begin{aligned} e^T e &= (Y - X\hat{\beta})^T (Y - X\hat{\beta}) + \lambda \hat{\beta}^T \hat{\beta} \\ &= (-\hat{\beta}^T X^T + Y^T)(Y - X\hat{\beta}) + \lambda \hat{\beta}^T \hat{\beta} \\ &= -\hat{\beta}^T X^T Y + \hat{\beta}^T X^T X \hat{\beta} + Y^T Y - Y^T X \hat{\beta} + \lambda \hat{\beta}^T \hat{\beta} \\ &= -2\hat{\beta}^T X^T Y + \hat{\beta}^T X^T X \hat{\beta} + Y^T Y + \lambda \hat{\beta}^T \hat{\beta} \end{aligned}$$

$$\begin{aligned} \frac{de^T e}{d\hat{\beta}} &= -2X^T Y + 2X^T X\hat{\beta} + 2\lambda\hat{\beta} \\ -> \text{set to } 0 \\ 2X^T Y &= 2X^T X\hat{\beta} + 2\lambda\hat{\beta} \\ X^T Y &= X^T X\hat{\beta} + \lambda\hat{\beta} \\ X^T Y &= (X^T X + I\lambda)\hat{\beta} \\ (X^T X + I\lambda)^{-1}(X^T X + I\lambda)\hat{\beta} &= (X^T X + I\lambda)^{-1} X^T Y \\ I\hat{\beta} &= (X^T X + I\lambda)^{-1} X^T Y \\ \hat{\beta} &= (X^T X + I\lambda)^{-1} X^T Y \end{aligned}$$

Prove OLS is scale invariant but ridge regressions is not

Scale invariant means that changing the scale of X does not change the prediction

In both Ridge and OLS, beta adjusts for the scaled X by minimizing the prediction error. However, in Ridge, large Betas are penalized because they increase the value of a function we intend to minimize. This penalization is not sensitive to the scale of X because lambda is not scaled in accordance with it. Therefore, we wind up with different beta estimates due to the Ridge beta^2 penalty

$$\min_{\beta} (Y - X\beta)^T (Y - X\beta) + \lambda \beta^T \beta$$

$$\hat{\beta} = (X^T X + I\lambda)^{-1} X^T Y$$

$$\min_{\beta} (Y - X\beta)^T (Y - X\beta)$$

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Prove relationship between Ridge, OLS, Lasso

X is an orthogonal matrix (i.e. the predictors are uncorrelated: Covariance = 0), prove

$$\hat{\beta}_j^{ridge} = \frac{\hat{\beta}_j^{OLS}}{1 - \lambda}$$

$$\hat{\beta}_j^{lasso} = sign(\hat{\beta}_j^{OLS})(\mid \hat{\beta}_j^{OLS} \mid -\lambda/2)_+$$

$$x_+ = 0 \text{ if } x < 0, \text{ and } x_+ = x \text{ if } x \geq 0$$

Properties of orthogonal matrix:

$$X^T X = X X^T = I$$

$$X^T = X^{-1}$$

$$\hat{\beta}_j^{OLS} = (X^T X)^{-1} X^T Y = (I)^{-1} X^T Y = X^T Y$$
$$\hat{\beta}_j^{ridge} = (X^T X + I\lambda)^{-1} X^T Y = (I + I\lambda)^{-1} X^T Y = (I + I\lambda)^{-1} \hat{\beta}_j^{OLS}$$

Proves part 2a.

$$ _j^{lasso} = sign(_j^{OLS})(*j^{OLS}- /2)+ \

x_+ = 0 x x_+ = x x $$

$$\hat{\beta}_j^{lasso} = sign(X^T Y)(\mid X^T Y \mid -\lambda/2)_+$$
$$\hat{\beta}_j^{lasso} = \min_{\beta}(Y - X\beta)^T (Y - X\beta),$$
$$s.t. \sum_{j=1}^{p} \mid \beta_j \mid \leq s$$

We can solve for the coefficient given three different scenarios

$$\hat{\beta}_j^{lasso} = \min_{\beta}(Y - X\beta)^T (Y - X\beta) + \lambda \mid \beta \mid$$
$$e^T e = -2\hat{\beta}^T X^T Y + \hat{\beta}^T \hat{\beta} + Y^T Y + \lambda \mid \hat{\beta} \mid$$
$$\text{when } \beta \text{ is positive}$$
$$\frac{de^T e}{d\hat{\beta}} = -2X^T Y + 2\hat{\beta} + \lambda$$
$$-> \text{set to } 0$$
$$2X^T Y - \lambda = 2\hat{\beta}$$
$$X^T Y - \lambda/2 = \hat{\beta}$$
$$\text{when } \beta \text{ is negative}$$
$$\frac{de^T e}{d\hat{\beta}} = -2X^T Y + 2\hat{\beta} - \lambda$$
$$-> \text{set to } 0$$
$$2X^T Y + \lambda = 2\hat{\beta}$$
$$X^T Y + \lambda/2 = \hat{\beta}$$
$$\text{when } \beta \text{ is zero}$$
$$\frac{de^T e}{d\hat{\beta}} = \hat{\beta} = 0$$

These three scenarios are accomplished by the single, given equation:

$$\hat{\beta}_j^{lasso} = sign(X^T Y)(\mid X^T Y \mid -\lambda/2)_+$$

Computation

Part 1, Carseats data set

This question is modified from Question 10 in Chapter 3.7 in ISLR. This question should be answered using the Carseats data set from the R package ISLR. (7 points)

```
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.3.2
```

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyve
rse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_co
nflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to forc
e all conflicts to become errors
```

```
cars <- Carseats
cars %>% glimpse
```

```
## Rows: 400
## Columns: 11
## $ Sales       <dbl> 9.50, 11.22, 10.06, 7.40, 4.15, 10.81, 6.63, 11.
85, 6.54, …
## $ CompPrice   <dbl> 138, 111, 113, 117, 141, 124, 115, 136, 132, 13
2, 121, 117…
## $ Income      <dbl> 73, 48, 35, 100, 64, 113, 105, 81, 110, 113, 78,
94, 35, 2…
## $ Advertising <dbl> 11, 16, 10, 4, 3, 13, 0, 15, 0, 0, 9, 4, 2, 11,
11, 5, 0, …
## $ Population  <dbl> 276, 260, 269, 466, 340, 501, 45, 425, 108, 131,
150, 503,…
## $ Price       <dbl> 120, 83, 80, 97, 128, 72, 108, 120, 124, 124, 10
0, 94, 136…
## $ ShelveLoc   <fct> Bad, Good, Medium, Medium, Bad, Bad, Medium, Goo
d, Medium,…
## $ Age         <dbl> 42, 65, 59, 55, 38, 78, 71, 67, 76, 76, 26, 50,
62, 53, 52…
## $ Education   <dbl> 17, 10, 12, 14, 13, 16, 15, 10, 10, 17, 10, 13,
18, 18, 18…
## $ Urban       <fct> Yes, Yes, Yes, Yes, Yes, No, Yes, Yes, No, No, N
o, Yes, Ye…
## $ US          <fct> Yes, Yes, Yes, Yes, No, Yes, No, Yes, No, Yes, Y
es, Yes, N…
```

Multiple Regression Model

a. Fit a multiple regression model to predict Sales using Price, Urban, and US.

```
glm1 <- cars %>% glm(Sales ~ Price + Urban + US, data = .)
glm1_sum <- summary(glm1)
glm1_sum
```

```
##
## Call:
## glm(formula = Sales ~ Price + Urban + US, data = .)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
## Price       -0.054459   0.005242 -10.389  < 2e-16 ***
## UrbanYes    -0.021916   0.271650  -0.081    0.936
## USYes        1.200573   0.259042   4.635 4.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.113219)
##
##     Null deviance: 3182.3  on 399  degrees of freedom
## Residual deviance: 2420.8  on 396  degrees of freedom
## AIC: 1865.3
##
## Number of Fisher Scoring iterations: 2
```

b. Provide an interpretation of each coefficient in the model. Be careful—some of the variables in the model are qualitative!

The dataset is 400 observations on 11 variables, in this model. Sales is the Unit sales (in thousands) at each location of child car seats. Price is the price company charges for car seats at each site. Urban is a factor indicating whether a store is in a urban or rural location. US is a factor with levels indicating whether the store is in the US or not.

Quantitative vars: Price - Qualitative vars: UrbanYes, USYes

The intercept coefficient is the average sales of the carseat outside of urban areas, outside of the US, and when the Price is equal to 0. It is equal to 13.04 (thousand), which is significantly different from zero. However, this would be more interpretable if the price variable were centered and/or normalized, because then the intercept would give the price in rural, non-US areas at the centered (e.g., average) price.

The price coefficient is significantly different form zero, indicating that each unit increase in price is associated with a 0.054 (thousand (aka 54 units)) lower sales on average, holding the other covariates constant.

The Urban indicator coefficient, is not statistically significantly different from zero, indicating no relationship between Urban neighborhood and rural neighborhoods, holding the other covariates constant.

The US indicator coefficient is statistically significantly different from zero, indicating that location in the US is associated with a 1.2 (thousand) higher sales count on average compared to location outside of the US, holding other covariates constant.

c. Write out the model in equation form, being careful to handle the qualitative variables properly.

$$\begin{aligned} Y &= X\beta + \epsilon \\ &= \beta_0 + \beta_{price}x_{price} + \beta_{urban}x_{urban} + \beta_{US}x_{US} + \epsilon \end{aligned}$$

d. For which of the predictors can you reject the null hypothesis $HH0$: $\beta\beta jj$ = 0?

For the price and US covariates, I can reject the null hypothesis that their beta coefficients are zero because their corresponding T-statistics are significantly unlikely to be drawn from the null T-distribution. ($p < 0.05$)

e. On the basis of your response to the previous question, fit a smaller model that only uses the predictors for which there is evidence of association with the

outcome.

```
glm2 <- cars %>% glm(Sales ~ Price + US, data = .)
glm2_sum <- summary(glm2)
glm2_sum
```

```
##
## Call:
## glm(formula = Sales ~ Price + US, data = .)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.03079    0.63098  20.652  < 2e-16 ***
## Price       -0.05448    0.00523 -10.416  < 2e-16 ***
## USYes        1.19964    0.25846   4.641 4.71e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.097921)
##
##     Null deviance: 3182.3  on 399  degrees of freedom
## Residual deviance: 2420.9  on 397  degrees of freedom
## AIC: 1863.3
##
## Number of Fisher Scoring iterations: 2
```

f. How well do the models in (a) and (e) fit the data?

The models have multiple and adjusted R-squareds that are virtually equal. A multiple R-squared of .2393 is fairly low, indicating almost 4/5ths of the variance in the outcome is unaccounted for.

```
anova(glm2, glm1)
```

```
## Analysis of Deviance Table
##
## Model 1: Sales ~ Price + US
## Model 2: Sales ~ Price + Urban + US
##   Resid. Df Resid. Dev Df Deviance
## 1       397     2420.9
## 2       396     2420.8  1  0.03979
```

There is no statistically significant evidence taht the model with the urban covariate performs any better than the simpler model.

g. Using the model from (e), obtain 95% confidence intervals for the coefficient(s).

The confidence intervals are:

```
confint(glm2)
```

```
## Waiting for profiling to be done...
```

```
##                   2.5 %      97.5 %
## (Intercept) 11.79410192 14.26748359
## Price       -0.06472849 -0.04422677
## USYes        0.69306864  1.70621725
```

Indicating that if we follow these same procedures again and again, then 95% of the time we will capture the true regression coefficients in the interval, and on this particular time none of our intervals contain zero.

h. Using the leave-one-out cross-validation and 5-fold cross-validation techniques to compare the performance of models in (a) and (e). What can you tell from (f) and (h)?

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
lmControl_5k <- trainControl(method = "repeatedcv",
                             number = 5,
                             repeats = 5)

lmControl_loo <- trainControl(method = "LOOCV")

lm2_5k <- train(formula(glm2),
                data = cars,
                method = "lm",
                trControl = lmControl_5k)

lm2_loo <- train(formula(glm2),
                 data = cars,
                 method = "lm",
                 trControl = lmControl_loo)

lm1_5k <- train(formula(glm1),
                data = cars,
                method = "lm",
                trControl = lmControl_5k)

lm1_loo <- train(formula(glm1),
                 data = cars,
                 method = "lm",
                 trControl = lmControl_loo)
```

```
resamps <- resamples(list(lm1_5k = lm1_5k,
                          lm2_5k = lm2_5k))

summary(resamps)
```

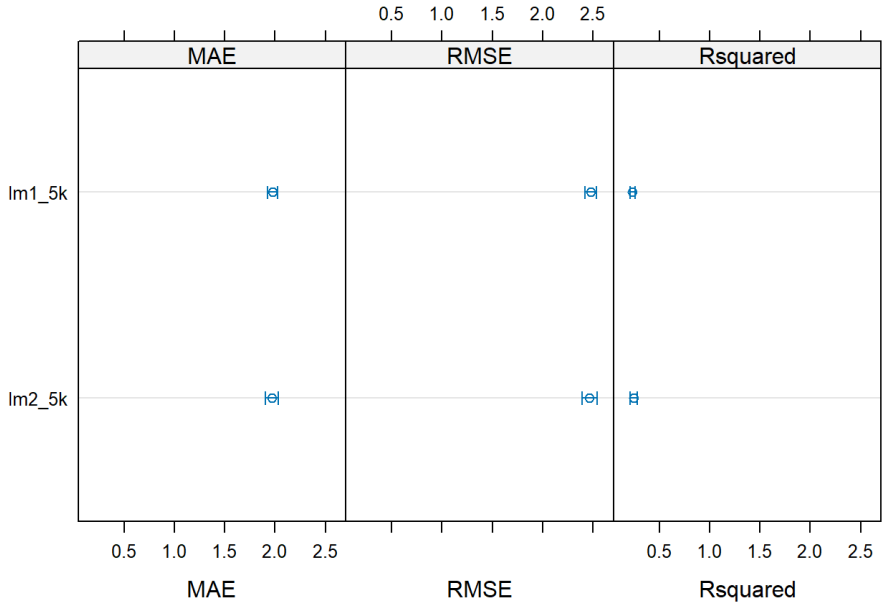
```
##
## Call:
## summary.resamples(object = resamps)
##
## Models: lm1_5k, lm2_5k
## Number of resamples: 25
##
## MAE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm1_5k 1.742238 1.909689 1.989179 1.973883 2.070330 2.169775    0
## lm2_5k 1.761842 1.825453 1.995489 1.968041 2.053171 2.289449    0
##
## RMSE
##            Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm1_5k 2.189821 2.373019 2.460704 2.475922 2.569105 2.719243    0
## lm2_5k 2.209968 2.298121 2.501796 2.466130 2.595917 2.912534    0
##
## Rsquared
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
NA's
## lm1_5k 0.08674554 0.1908068 0.2437311 0.2353970 0.2716111 0.3342987
0
## lm2_5k 0.05118121 0.1909123 0.2390956 0.2439518 0.3040147 0.4372463
0
```

```
summary(lm1_loo);  summary(lm2_loo)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9206 -1.6220 -0.0564  1.5786  7.0581
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
## Price       -0.054459   0.005242 -10.389  < 2e-16 ***
## UrbanYes    -0.021916   0.271650  -0.081    0.936
## USYes        1.200573   0.259042   4.635 4.86e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.472 on 396 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2335
## F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9269 -1.6286 -0.0574  1.5766  7.0515
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.03079    0.63098  20.652  < 2e-16 ***
## Price       -0.05448    0.00523 -10.416  < 2e-16 ***
## USYes        1.19964    0.25846   4.641 4.71e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.469 on 397 degrees of freedom
## Multiple R-squared:  0.2393, Adjusted R-squared:  0.2354
## F-statistic: 62.43 on 2 and 397 DF,  p-value: < 2.2e-16
```

```
dotplot(resamps)
```



Confidence Level: 0.95

Using cross-validation yields the same results as in comparing the models fit on the entire dataset. Cross-validation merely identifies the fact that the models can have different ranges in their performance metrics, such as their RMSE, and that on average they perform mostly the same. The same goes for LOOCV. The plots illustrate that the 95% confidence intervals around the performance metrics largely overlap for the two models, so it makes sense to go with the simpler model.

Part 2. Prostate data

This question pertains to a prostate microarray dataset. You can access it by load("prostate.Rdata"). It has been preprocessed to have 210 gene and 235 samples. Lpsa value is the clinical outcome we want to predict. (5 points)

Split dataset

a. Randomly divide the data into one training dataset and one testing dataset (1:1).

```
load('prostate.Rdata')
prostate <- data; rm(data)
prostate$x %>% dim
```

```
## [1] 235 210
```

```
prostate$y  %>% length()
```

```
## [1] 235
```

There are 235 samples of 210 genes and an associated Lpsa value.

```
p_tbl <- as_tibble(prostate$x) %>%
  mutate(lpsa = prostate$y)
trainIndex <- createDataPartition(p_tbl$lpsa, p = .5,
                                  list = FALSE,
                                  times = 1)
p_tbl_train <- p_tbl[trainIndex,]
p_tbl_test <- p_tbl[-trainIndex,]
```

OLS Model

b. Fit a linear model using OLS on the training dataset and calculate the test error in terms of RMSE. Report any problems you encountered.

$$\hat{\beta}^{OLS} = (X^T X)^{-1} X^T Y$$

```
ols_estimates <- function(X, Y){
  solve(t(X)%*%X)%*%t(X)%*%Y
}

train_matrix <- as.matrix(p_tbl_train)
# 211 is the lpsa column
train_X <- train_matrix[,-211]
train_Y <- train_matrix[,211]
train_beta_ols <- ols_estimates(train_X, train_Y)
```

```
## Error in solve.default(t(X) %*% X): system is computationally singul
ar: reciprocal condition number = 3.57635e-21
```

The system is computationally singular, meaning that at least two of the covariates are so highly correlated that their beta coefficients are unidentifiable. The determinant of the matrix must also be zero, since this is a necessary/sufficient condition for the matrix to be uninvertible, which is necessary for obtaining the OLS estimates. This can be circumvented by removing covariates with high correlation. Uncertainty around coefficients starts to skyrocket at $\rho = .8$ so I will use this for my cutoff at first.

```
train_cor <- cor(train_X)

train_cor[upper.tri(train_cor)] <- 0
diag(train_cor) <- 0

train_X_pruned <- train_X[, !apply(train_cor, 2, function(x) any(abs(x)
> 0.8))]
train_X_pruned %>% dim()
```

```
## [1] 118 151
```

149/210 (71%) covariates remain after this correlation threshold pruning. I will continue this procedure until I don't encounter singularities.

```
train_beta_ols <- ols_estimates(train_X_pruned, train_Y)
```

```
## Error in solve.default(t(X) %*% X): system is computationally singul
ar: reciprocal condition number = 1.14813e-20
```

```
train_X_pruned <- train_X[, !apply(train_cor, 2, function(x) any(abs(x)
> 0.7))]
train_X_pruned %>% dim()
```

```
## [1] 118  71
```

```
train_beta_ols <- ols_estimates(train_X_pruned, train_Y)
```

A cutoff of .7 (with 79/210 or 37.6% covariates) relieves the computational singularity error.

We then use the remaining covariates to filter those in the testing dataset, since we can only make predictions with those covariates used to train our model.

```
train_X_pruned_covs <- colnames(train_X_pruned)
test_matrix <- as.matrix(p_tbl_test)
test_X_pruned <- test_matrix[,train_X_pruned_covs]
test_X <- test_matrix[,-211]
test_Y <- test_matrix[,211]
pred_vector <- test_X_pruned%*%train_beta_ols
```

```
calc_rmse <- function(y, pred_y){
  sqrt( sum( (pred_y - y)^2 ) / length(y) )
}

(ols_rmse <- calc_rmse(test_Y, pred_vector))
```

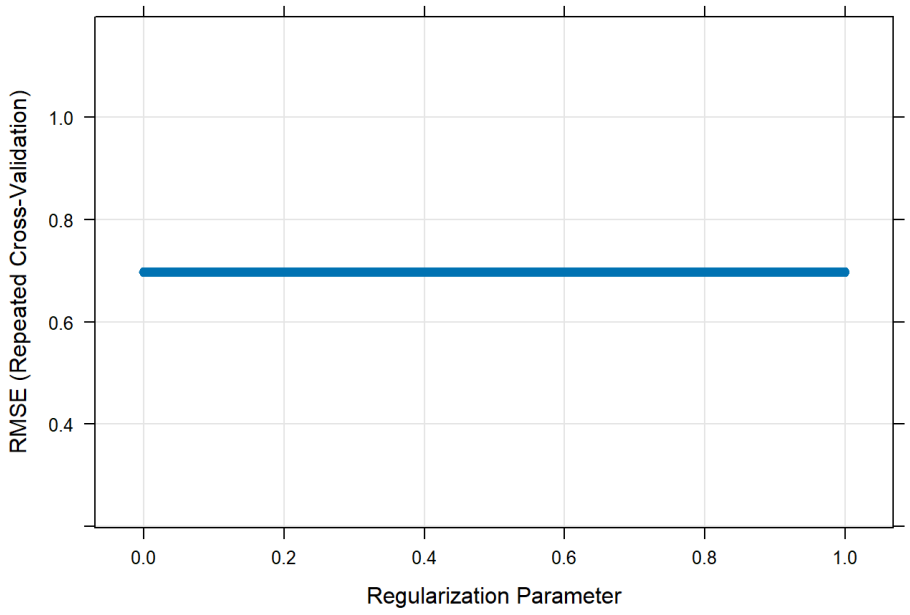
```
## [1] 1.078462
```

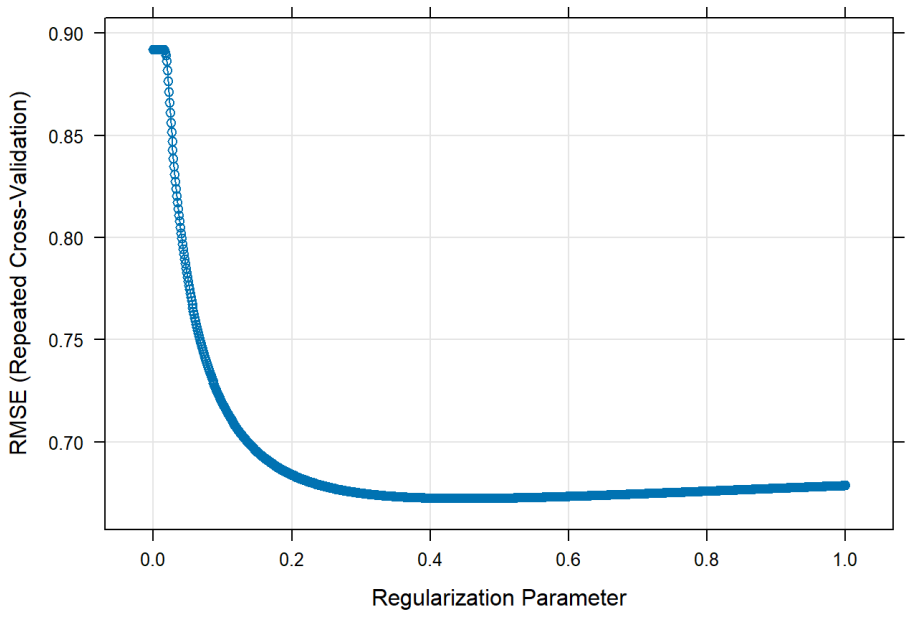
Ridge Model

c. Use ridge regression. Find the optimal lambda which will return the smallest cross validation error using the training data.

I know it says cross-validation, but I am using repeated cross-validation for fun.

```
ridgeControl <- trainControl(method = 'repeatedcv',
                             number = 10,
                             repeats = 10)

tuneRidge <- expand.grid(alpha = 0, lambda = seq(.0001,1,by=0.001))

ridge1 <- train(lpsa ~ .,
                data = p_tbl_train,
                method = "glmnet",
                trControl = ridgeControl,
                tuneGrid = tuneRidge)

ridge2 <- train(lpsa ~ .,
                data = p_tbl_train[,c(train_X_pruned_covs,'lpsa')],
                method = "glmnet",
                trControl = ridgeControl,
                tuneGrid = tuneRidge)

plot(ridge1);plot(ridge2)
```





```
ridge1$bestTune;ridge2$bestTune
```

```
##      alpha lambda
## 1000     0 0.9991
```

```
##     alpha lambda
## 448     0 0.4471
```

These results are interesting. It appears that when using the full dataset, any regularization parameter in the range of 0.0001 and 1 is equally good, while in the pruned training dataset, regularization needs to be kicked up in order to achieve a similar RMSE. While the average RMSE is just slightly better in ridge2, I will use the Ridge model in the full training dataset, since Ridge handles the pruning for me, and thus may be more sensitive to which covariates help and may generalize better in the test set. This corresponds to a lambda of 0.9991

Optimal Ridge Model from Scratch

d. Build the ridge regression model using the training data and the lambda in (c) and then predict test error in terms of RMSE.

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$

```
ridge_estimates <- function(X, Y, lambda){
  solve(t(X)%*%X + lambda * diag(nrow=ncol(X)))%*%t(X)%*%Y
}

train_beta_ridge <- ridge_estimates(train_X, train_Y, lambda=0.9991)

ridge_preds <- test_X %*% train_beta_ridge

ridge_rmse <- calc_rmse(y = test_Y, pred_y = ridge_preds)

ridge_rmse
```

```
## [1] 1.002593
```

The Ridge Test RMSE is better than the pruned OLS Test RMSE.

Lasso Model

e. Repeat steps in (c) and (d) using lasso. Derive the RMSE in the testing dataset.

I am using repeated CV again for fun.

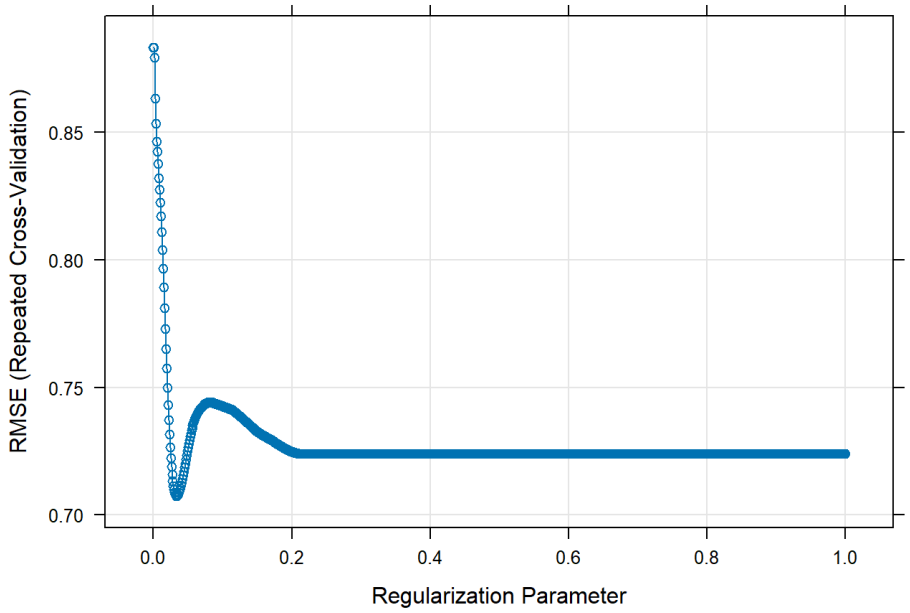
```
lassoControl <- trainControl(method = 'repeatedcv',
                             number = 10,
                             repeats = 10)

tunelasso <- expand.grid(alpha = 1, lambda = seq(.0001,1,by=0.001))

lasso1 <- train(lpsa ~ .,
                data = p_tbl_train,
                method = "glmnet",
                trControl = lassoControl,
                tuneGrid = tunelasso)
```

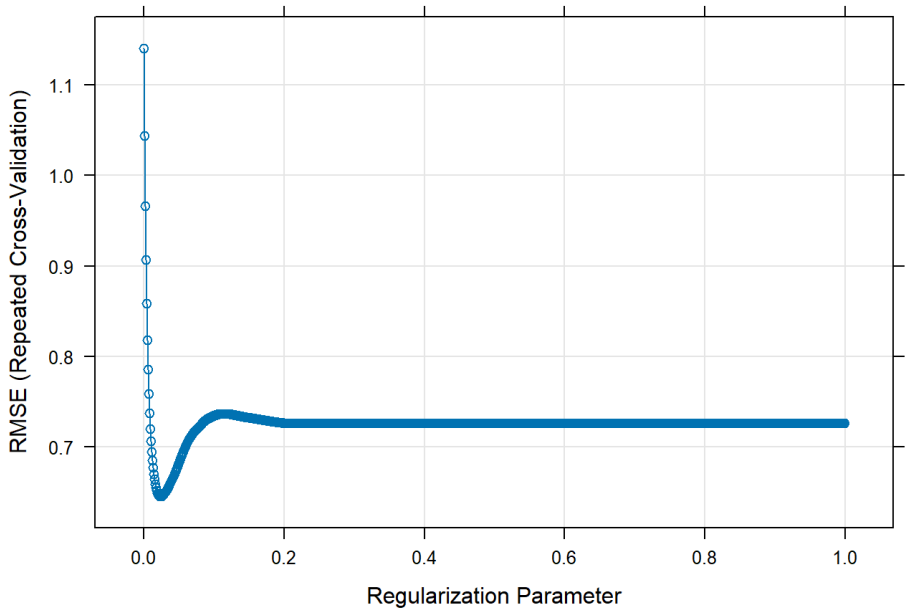
```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
trainInfo,
## : There were missing values in resampled performance measures.
```

```
lasso2 <- train(lpsa ~ .,
                data = p_tbl_train[,c(train_X_pruned_covs,'lpsa')],
                method = "glmnet",
                trControl = lassoControl,
                tuneGrid = tunelasso)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
trainInfo,
## : There were missing values in resampled performance measures.
```

```
plot(lasso1);plot(lasso2);
```






```
lasso1$bestTune;lasso2$bestTune
```

```
##    alpha lambda
## 35     1 0.0341
```

```
##    alpha lambda
## 24     1 0.0231
```

```
lasso_pred_y <- predict(lasso1, newdata=test_X)
lasso_rmse <- calc_rmse(y=test_Y, pred_y=lasso_pred_y)
```

```
tibble(beta_type = c("OLS", "Ridge", "Lasso"),
       rmse = c(ols_rmse, ridge_rmse, lasso_rmse)) %>%
  kableExtra::kable() %>% kableExtra::kable_styling()
```

beta_type	rmse
OLS	1.0784618
Ridge	1.0025927
Lasso	0.6601722

It is clear that the Lasso regularized model has the best Test set performance in terms of RMSE.