STAD80 Assignment 3

Vladislav Trukhin

February 12th, 2022

Contents

Question 1 **1**
Question 1.1.a . . . 1
Question 1.1.b . . . 3
Question 1.1.c . . . 4
Question 1.1.d . . . 5
Question 1.2 . . . 6

Question 2 **8**
Question 2.1 . . . 8
Question 2.2 . . . 9

Question 3 **11**
Question 3.1 . . . 11
Question 3.2 . . . 12

Question 4 **13**
Question 4.1 . . . 13
Question 4.2 . . . 14

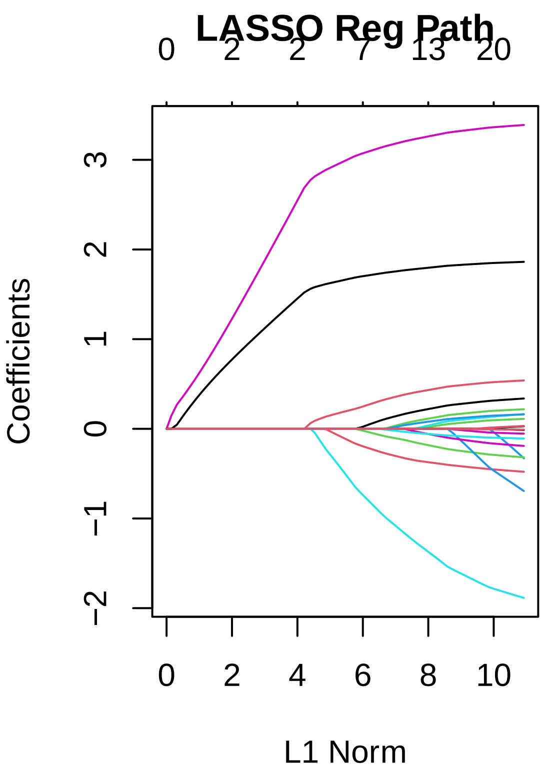
Question 1

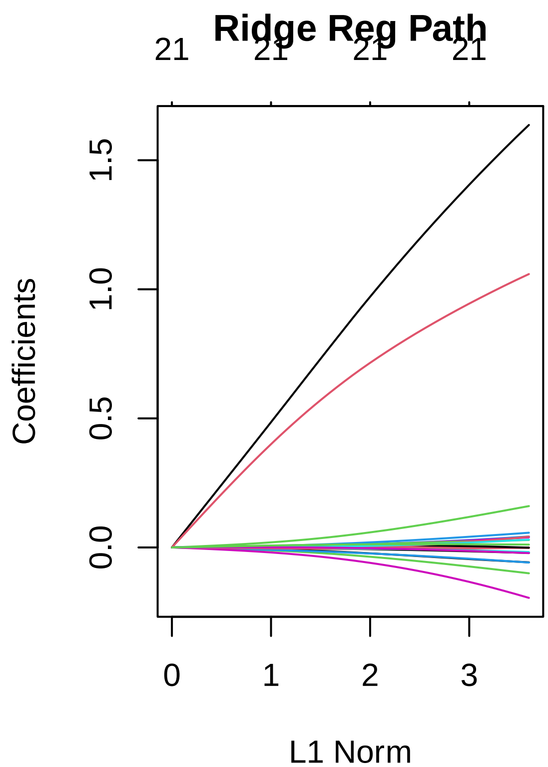
Question 1.1.a

```
par(mfrow=c(1,2))
load("/Users/vladislavtrukhin/Downloads/A3_data/q1.RData")

#Format training data
dataTrainAll$Region <- cbind(ifelse(dataTrainAll$Region == 1, 1, 0),
                             ifelse(dataTrainAll$Region == 3, 1, 0))
dataTrainAll$City <- cbind(ifelse(dataTrainAll$City == 1, 1, 0),
                           ifelse(dataTrainAll$City == 2, 1, 0),
                           ifelse(dataTrainAll$City == 3, 1, 0),
                           ifelse(dataTrainAll$City == 4, 1, 0),
                           ifelse(dataTrainAll$City == 5, 1, 0))
dataTrainAll$AdX <- cbind(ifelse(dataTrainAll$AdX == 1, 1, 0),
                          ifelse(dataTrainAll$AdX == 2, 1, 0))
dataTrainAll$Domain <- cbind(
                ifelse(dataTrainAll$Domain == '5Fa-expoBTTR1m58uG', 1, 0),
                ifelse(dataTrainAll$Domain == '5KFUl5p0Gxsvgmd4wspENpn', 1, 0),
                ifelse(dataTrainAll$Domain == 'trqRTuT-GNTYJNKbuKz', 1, 0),
```

```
                ifelse(dataTrainAll$Domain == 'trqRTu5Jg9q9wMKYvmpENpn', 1, 0))
dataTrainAll$Key_Page <- cbind(
  ifelse(dataTrainAll$Key_Page == '3a7eb50444df6f61b2409f4e2f16b687', 1, 0),
  ifelse(dataTrainAll$Key_Page == 'df6f61b2409f4e2f16b6873a7eb50444', 1, 0))
dataTrainAll$Ad_Vis <- cbind(ifelse(dataTrainAll$Ad_Vis == 1, 1, 0),
                             ifelse(dataTrainAll$Ad_Vis == 2, 1, 0))
dataTrainAll$Ad_Form <- cbind(ifelse(dataTest$Ad_Form == 1, 1, 0))
dataTrainAll$Ad_Width <- (dataTrainAll$Ad_Width-mean(dataTrainAll$Ad_Width))/
                            sd(dataTrainAll$Ad_Width)
dataTrainAll$Ad_Height <- (dataTrainAll$Ad_Height-mean(dataTrainAll$Ad_Height))/
                            sd(dataTrainAll$Ad_Height)
dataTrainAll$Floor_Price <-
  (dataTrainAll$Floor_Price-mean(dataTrainAll$Floor_Price))/
    sd(dataTrainAll$Floor_Price)
dataTrainAll$Click <- as.integer(dataTrainAll$Click != 0)
predictors_train <- cbind(dataTrainAll$Region,
                    dataTrainAll$City,
                    dataTrainAll$AdX,
                    dataTrainAll$Domain,
                    dataTrainAll$Key_Page,
                    dataTrainAll$Ad_Vis,
                    dataTrainAll$Ad_Form,
                    dataTrainAll$Ad_Width,
                    dataTrainAll$Ad_Height,
                    dataTrainAll$Floor_Price)

#Format test data
dataTest$Region <- cbind(ifelse(dataTest$Region == 1, 1, 0),
                         ifelse(dataTest$Region == 3, 1, 0))
dataTest$City <- cbind(ifelse(dataTest$City == 1, 1, 0),
                       ifelse(dataTest$City == 2, 1, 0),
                       ifelse(dataTest$City == 3, 1, 0),
                       ifelse(dataTest$City == 4, 1, 0),
                       ifelse(dataTest$City == 5, 1, 0))
dataTest$AdX <- cbind(ifelse(dataTest$AdX == 1, 1, 0),
                      ifelse(dataTest$AdX == 2, 1, 0))
dataTest$Domain <- cbind(
                    ifelse(dataTest$Domain == '5Fa-expoBTTR1m58uG', 1, 0),
                    ifelse(dataTest$Domain == '5KFUl5p0Gxsvgmd4wspENpn', 1, 0),
                    ifelse(dataTest$Domain == 'trqRTuT-GNTYJNKbuKz', 1, 0),
                    ifelse(dataTest$Domain == 'trqRTu5Jg9q9wMKYvmpENpn', 1, 0))
dataTest$Key_Page <- cbind(
          ifelse(dataTest$Key_Page == '3a7eb50444df6f61b2409f4e2f16b687', 1, 0),
          ifelse(dataTest$Key_Page == 'df6f61b2409f4e2f16b6873a7eb50444', 1, 0))
dataTest$Ad_Vis <- cbind(ifelse(dataTest$Ad_Vis == 1, 1, 0),
                         ifelse(dataTest$Ad_Vis == 2, 1, 0))
dataTest$Ad_Form <- cbind(ifelse(dataTest$Ad_Form == 1, 1, 0))
dataTest$Ad_Width <- (dataTest$Ad_Width-mean(dataTest$Ad_Width))/
                        sd(dataTest$Ad_Width)
dataTest$Ad_Height <- (dataTest$Ad_Height-mean(dataTest$Ad_Height))/
                        sd(dataTest$Ad_Height)
dataTest$Floor_Price <- (dataTest$Floor_Price-mean(dataTest$Floor_Price))/
                          sd(dataTest$Floor_Price)
```

```
dataTestRes$Click <- as.integer(dataTestRes$Click != 0)
predictors_test <- cbind(dataTest$Region,
                         dataTest$City,
                         dataTest$AdX,
                         dataTest$Domain,
                         dataTest$Key_Page,
                         dataTest$Ad_Vis,
                         dataTest$Ad_Form,
                         dataTest$Ad_Width,
                         dataTest$Ad_Height,
                         dataTest$Floor_Price)

#Plot regularization paths
lasso <- glmnet(predictors_train, dataTrainAll$Click, family="binomial",
                standardize=FALSE, alpha=1)
ridge <- glmnet(predictors_train, dataTrainAll$Click, family="binomial",
                standardize=FALSE, alpha=0)
plot(lasso)
title('LASSO Reg Path')
plot(ridge)
title('Ridge Reg Path')
```





Question 1.1.b

```
# Pulling data to figure out which coefficient is which on the reg path graphs
lasso$beta[,"s60"]
```

```
##          V1          V2          V3          V4          V5          V6
##  0.00000000  0.00000000 -0.37332014  0.02034307  0.00000000  0.03862679
```

```
##          V7          V8          V9         V10         V11         V12
##  0.00000000  0.00000000  0.00000000 -0.18408582  0.00000000 -1.37743055
##         V13         V14         V15         V16         V17         V18
## -0.05923494  0.22183780  0.43412044  0.11498391  0.08058022 -0.05764039
##         V19         V20         V21
##  3.26243672  1.79726921  0.00000000
```

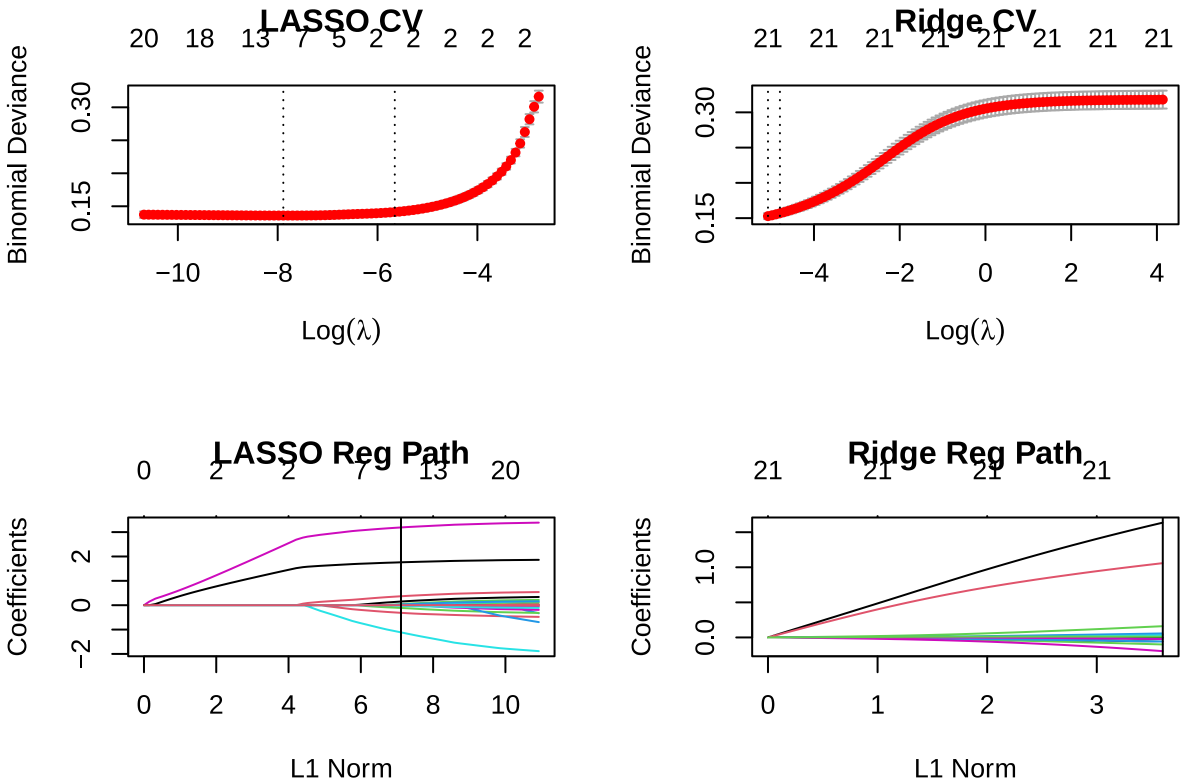
```
ridge$beta[,"s99"]
```

```
##            V1            V2            V3            V4            V5
## -0.0576417597 -0.0022104905 -0.1000908016  0.0424505564 -0.0198632993
##            V6            V7            V8            V9           V10
##  0.0390233525 -0.0213713002 -0.0005691764  0.0288367595 -0.0584429726
##           V11           V12           V13           V14           V15
## -0.0177808368 -0.1952993194 -0.0008595341  0.0419041119  0.1602612574
##           V16           V17           V18           V19           V20
##  0.0569908436  0.0301263809 -0.0220105017  1.6364782552  1.0589847255
##           V21
##  0.0109832008
```

Ad_Width, Ad_Height, and Domain "trqRTuT-GNTYJNKbuKz" where all chosen as influential features by LASSO, influential being measured by a relatively large coefficient magnitude. Ridge chose Ad_Width, Ad_Height as influential features.

Question 1.1.c

```
par(mfrow=c(2,2))

#Plot cross validation
lasso_cv <- cv.glmnet(predictors_train, dataTrainAll$Click, family="binomial",
                      standardize=FALSE, alpha=1, nfolds=5)
ridge_cv <- cv.glmnet(predictors_train, dataTrainAll$Click, family="binomial",
                      standardize=FALSE, alpha=0, nfolds=5)
plot(lasso_cv)
title('LASSO CV')
plot(ridge_cv)
title('Ridge CV')

#Plot regularization paths with chosen coefficients norm line
plot(lasso)
abline(v=sum(abs(coef(lasso_cv, s="lambda.min")[2:22])))
title('LASSO Reg Path')
plot(ridge)
abline(v=sum(abs(coef(ridge_cv, s="lambda.min")[2:22])))
title('Ridge Reg Path')
```



The regression paths graphs shows 2-3 of the original 21 features are dominant in predicting at least one click. The CV deviance spikes at certain thresholds of λ, with the values below the threshold corresponding with the relative dominance of a few features on the regression paths graphs. The reason why models with large degree of freedom do not tend to perform better in CV is because they are prone to over-fitting, which leads to a low training error at the expense of generalization, resulting in more deviance during CV calculation.

Question 1.1.d

```r
# Total classifications made
n <- sum(table(dataTestRes$Click))
n
```

```
## [1] 10000
```

```r
# Classification test error for ridge
predict_test_ridge  <- predict(ridge_cv, predictors_test, s="lambda.min")
# Error when true class was 0 but predicted class was 1
sum(as.integer(as.integer(predict_test_ridge > 0) - dataTestRes$Click == 1))
```

```
## [1] 22
```

```r
# Error when true class was 1 but predicted class was 0
sum(as.integer(as.integer(predict_test_ridge > 0) - dataTestRes$Click == -1))
```

```
## [1] 236
```

```r
# Classification test error for lasso
predict_test_lasso  <- predict(lasso_cv, predictors_test, s="lambda.min")
# Error when true class was 0 but predicted class was 1
sum(as.integer(as.integer(predict_test_lasso > 0) - dataTestRes$Click == 1))
```

```
## [1] 23
```

```r
# Error when true class was 1 but predicted class was 0
sum(as.integer(as.integer(predict_test_lasso > 0) - dataTestRes$Click == -1))
```

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

Question 1.2

```r
par(mfrow=c(1,2))
load("/Users/vladislavtrukhin/Downloads/A3_data/q1.RData")

# Format training data
dataTrainAll$AdX <- (dataTrainAll$AdX-mean(dataTrainAll$AdX))/
                      sd(dataTrainAll$AdX)
dataTrainAll$iPinYou_Bid <- (dataTrainAll$iPinYou_Bid
                             -mean(dataTrainAll$iPinYou_Bid))/
                              sd(dataTrainAll$iPinYou_Bid)
dataTrainAll$Comp_Bid <- (dataTrainAll$Comp_Bid-mean(dataTrainAll$Comp_Bid))/
                            sd(dataTrainAll$Comp_Bid)

# Fit linear regression
fit <- lm(Comp_Bid ~ AdX + iPinYou_Bid, dataTrainAll)
mle <- fit$coefficients[2:3]
mle #MLE
```

```
##          AdX iPinYou_Bid
##   -0.1664490   0.7721763
```

```r
# Fit lasso linear regression
lasso <- glmnet(cbind(dataTrainAll$AdX, dataTrainAll$iPinYou_Bid),
                dataTrainAll$Comp_Bid, family="gaussian", standardize=FALSE,
                alpha=1)

# Choose lasso solution
l1_norm_mle <- sum(abs(mle))
for (i in 1:ncol(lasso$beta)) {
  l1_norm_lasso <- sum(abs(lasso$beta[,i]))
  if (l1_norm_lasso > 0.5*l1_norm_mle) {
    break
  }
  picked_lasso <- lasso$beta[,i]
}
picked_lasso #Lasso
```

```
##        V1        V2
## 0.0000000 0.4382291
```

```r
# Fit ridge linear regression
ridge <- glmnet(cbind(dataTrainAll$AdX, dataTrainAll$iPinYou_Bid),
                dataTrainAll$Comp_Bid, family="gaussian", standardize=FALSE,
                alpha=0)

# Choose ridge solution
l2_norm_mle <- sum((mle)^2)
for (i in 1:ncol(ridge$beta)) {
```

```r
  l2_norm_ridge <- sum((ridge$beta[,i])^2)
  if (l2_norm_ridge > 0.5*l2_norm_mle) {
    break
  }
  picked_ridge <- ridge$beta[,i]
}
picked_ridge #Ridge
```

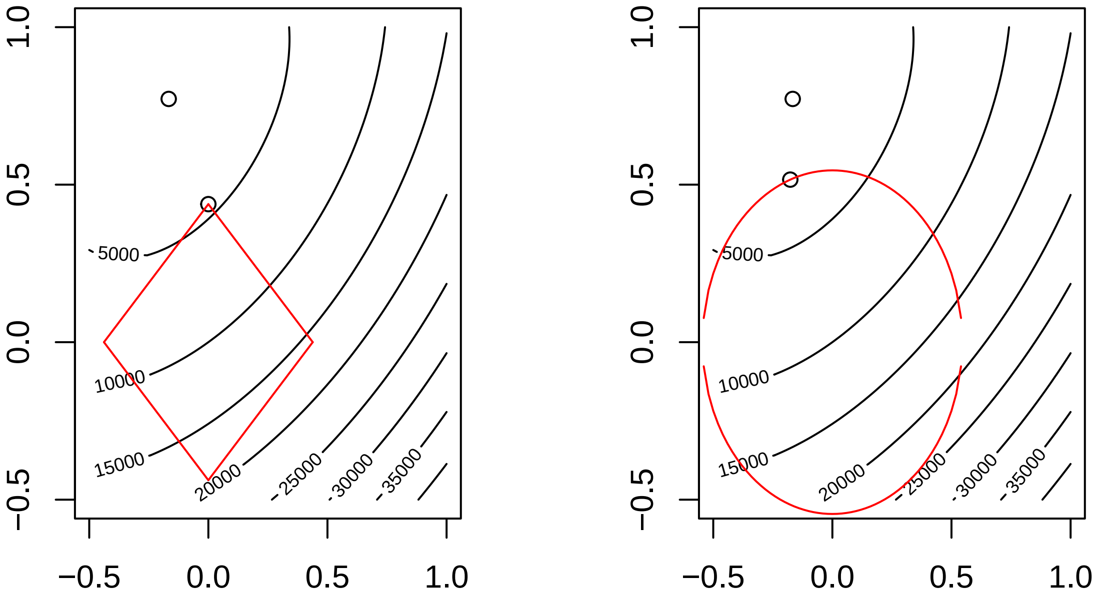
```
##         V1         V2
## -0.1765613  0.5160305
```

```r
# Discretized grid of mse
beta1 = seq(-.5,1,length.out=100)
beta2 = seq(-.5,1,length.out=100)
mse <- matrix(, nrow = 100, ncol = 100)
for (i in 1:100) {
  for (j in 1:100) {
    pred <- beta1[i]*dataTrainAll$AdX + beta2[j]*dataTrainAll$iPinYou_Bid
    mse[i,j] = sum((dataTrainAll$Comp_Bid - pred)^2)
  }
}

# Plot contour with mse and lasso
contour(beta1, beta2, mse, nlevels=10)
points(mle[1], mle[2])
points(picked_lasso[1], picked_lasso[2])
l1_norm_lasso <- sum(abs(picked_lasso))
plot(function(x){x=-x+l1_norm_lasso}, 0, l1_norm_lasso, add=TRUE, col = 'red')
plot(function(x){x=x-l1_norm_lasso}, 0, l1_norm_lasso, add=TRUE, col = 'red')
plot(function(x){x=x+l1_norm_lasso}, -l1_norm_lasso, 0, add=TRUE, col = 'red')
plot(function(x){x=-x-l1_norm_lasso}, -l1_norm_lasso, 0, add=TRUE, col = 'red')

# Plot contour with mse and ridge
contour(beta1, beta2, mse, nlevels=10)
points(mle[1], mle[2])
points(picked_ridge[1],picked_ridge[2])
l2_norm_ridge <- sum((picked_ridge)^2)
plot(function(x){sqrt(l2_norm_ridge-x^2)}, -1, 1, add=TRUE, col='red')
```

```
## Warning in sqrt(l2_norm_ridge - x^2): NaNs produced
```

```r
plot(function(x){-sqrt(l2_norm_ridge-x^2)}, -1, 1, add=TRUE, col='red')
```

```
## Warning in sqrt(l2_norm_ridge - x^2): NaNs produced
```



The MLE sits right in the epicenter of the level curves, which tells us that the MLE coefficients have the lowest MSE of all the possible coefficients.

LASSO sits on the top vertex of the diamond $L1$ norm, the vertex closest to the smallest level curve that touches the shape. The point is the lowest MSE possible with the condition that the point is along the shape.

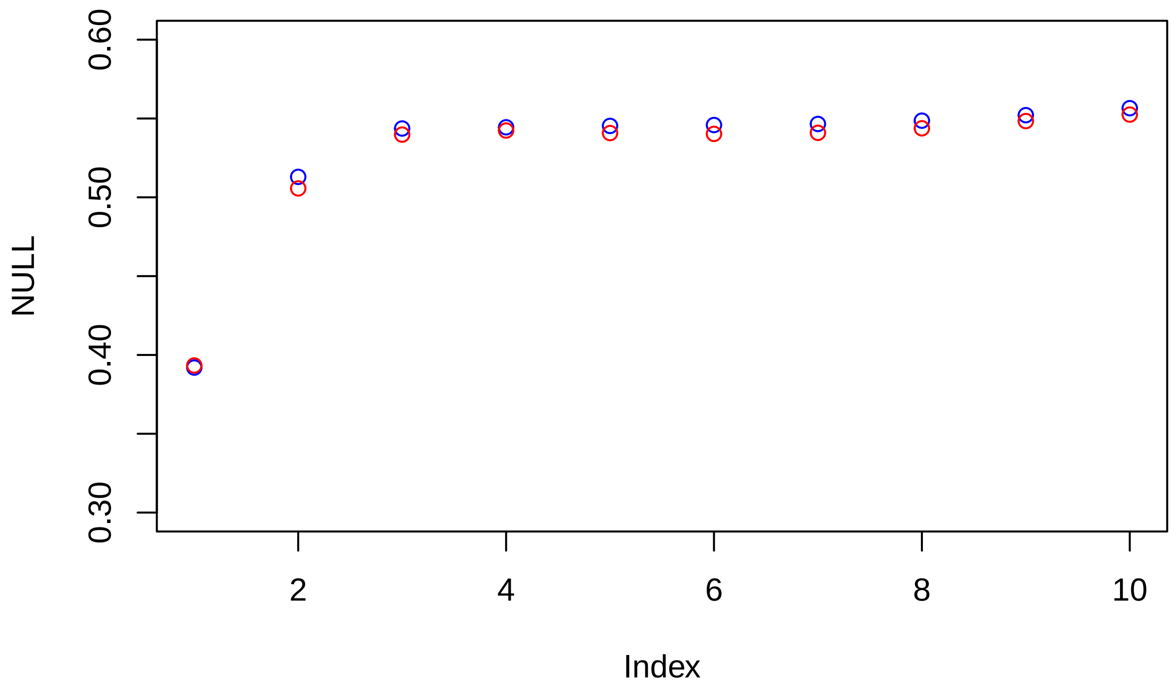
Ridge sits on the upper-left portion of the circle $L2$ norm, the point closest to the smallest level curve that touches the shape. The point is the lowest MSE possible with the condition that the point is along the shape.

Lasso favors sparsity as the restriction that the coordinate of coefficients be bounded by an $L1$ norm means they are geometrically restricted to the diamond shape, with the vertices of the shape, where some of the coefficients are 0, end up the most optimal solutions in terms of MSE.

Question 2

Question 2.1

```
train = read.csv("/Users/vladislavtrukhin/Downloads/_data_hw2/train.data.csv")
test = read.csv("/Users/vladislavtrukhin/Downloads/_data_hw2/test.data.csv")

# Plot R^2 wrt degree
plot(NULL, xlim=c(1, 10), ylim=c(0.3, 0.6))
for (i in 1:10) {
  fit <- lm(price ~ poly(log(sqft_living), i) + bedrooms + bathrooms, train)

  # R^2 train
  points(i, summary(fit)$r.squared, col='blue')

  # R^2 test
  predict_test <- predict(fit, test)
  rss <- sum((predict_test - test$price) ^ 2)
  tss <- sum((test$price - mean(test$price)) ^ 2)
  rsq <- 1 - rss/tss
  points(i, rsq, col='red')
}
```

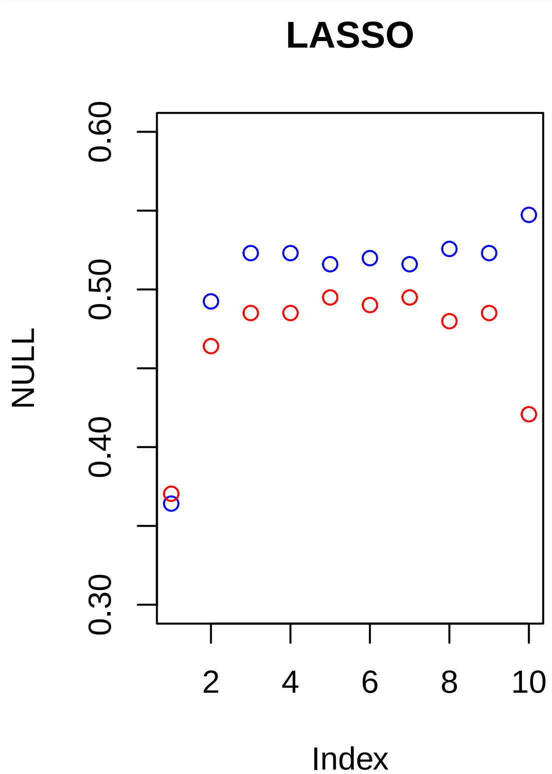


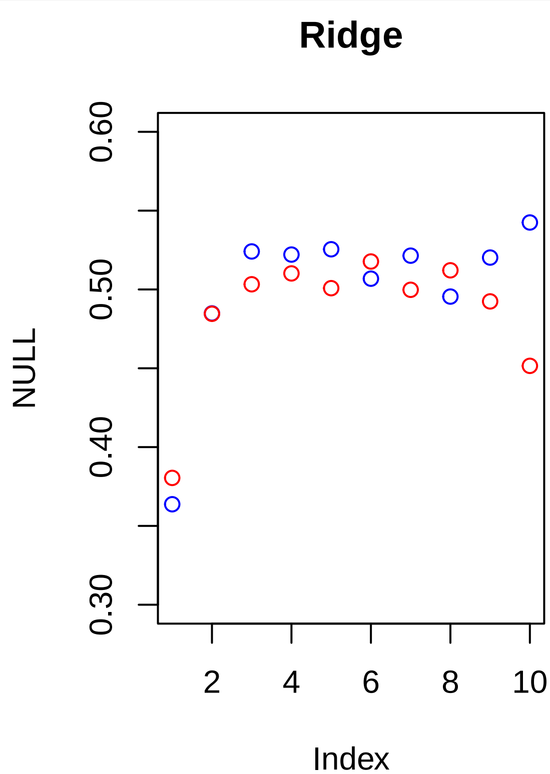
The plot is consistent with the concept that adding more features does not hurt the training R^2. Over-fitting can be seen, as past $k = 4$, the gap between the training and test R^2 begins to grow, showing that increases in k is improving the training R^2 without improving the test R^2 at the same rate.

Question 2.2

```
par(mfrow=c(1,2))

# Plot R^2 wrt degree, lasso
plot(NULL, xlim=c(1, 10), ylim=c(0.3, 0.6))
title("LASSO")
for (i in 1:10) {
  fit <- cv.glmnet(
    cbind(poly(log(train$sqft_living), i), train$bedrooms, train$bathrooms),
    train$price, alpha=1, nfolds=10)

  # R^2 train
  predict_train <- predict(fit, cbind(poly(log(train$sqft_living), i),
                                  train$bedrooms,
                                  train$bathrooms), s="lambda.1se")
  rss <- sum((predict_train - train$price) ^ 2)
  tss <- sum((train$price - mean(train$price)) ^ 2)
  rsq <- 1 - rss/tss
  points(i, rsq, col='blue')

  # R^2 test
  predict_test <- predict(fit, cbind(poly(log(test$sqft_living), i),
                                  test$bedrooms,
                                  test$bathrooms), s="lambda.1se")
  rss <- sum((predict_test - test$price) ^ 2)
  tss <- sum((test$price - mean(test$price)) ^ 2)
  rsq <- 1 - rss/tss
```

```r
  points(i, rsq, col='red')
}

# Plot R^2 wrt degree, ridge
plot(NULL, xlim=c(1, 10), ylim=c(0.3, 0.6))
title("Ridge")
for (i in 1:10) {
  fit <- cv.glmnet(
    cbind(poly(log(train$sqft_living), i), train$bedrooms, train$bathrooms),
    train$price, alpha=0, nfolds=10)

  # R^2 train
  predict_train <- predict(fit, cbind(poly(log(train$sqft_living), i),
                                    train$bedrooms,
                                    train$bathrooms), s="lambda.1se")
  rss <- sum((predict_train - train$price) ^ 2)
  tss <- sum((train$price - mean(train$price)) ^ 2)
  rsq <- 1 - rss/tss
  points(i, rsq, col='blue')

  # R^2 test
  predict_test <- predict(fit, cbind(poly(log(test$sqft_living), i),
                                    test$bedrooms,
                                    test$bathrooms), s="lambda.1se")
  rss <- sum((predict_test - test$price) ^ 2)
  tss <- sum((test$price - mean(test$price)) ^ 2)
  rsq <- 1 - rss/tss
  points(i, rsq, col='red')
}
```



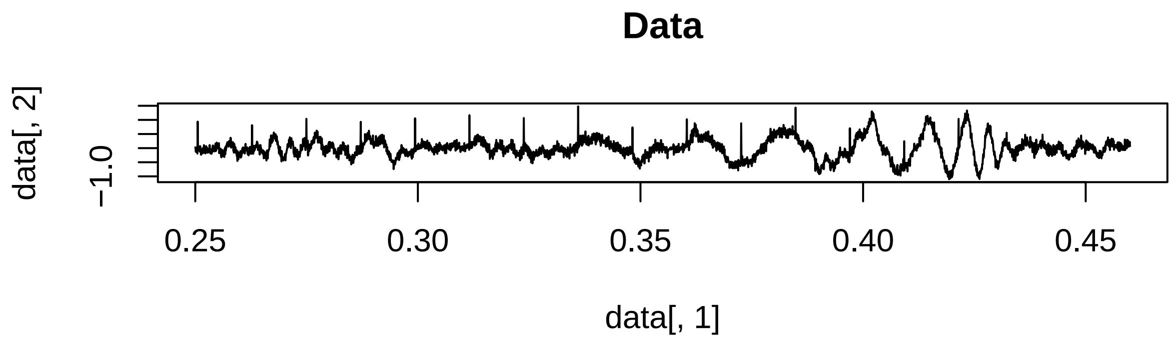


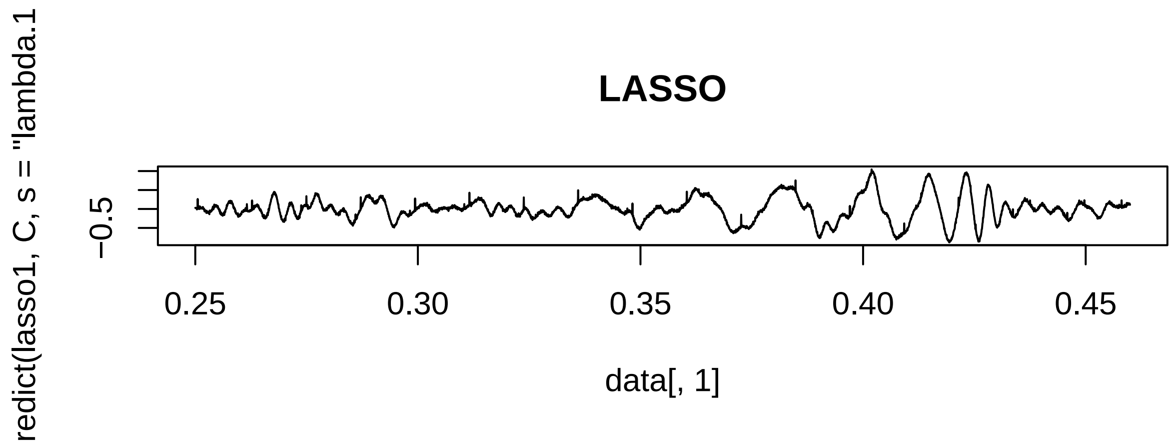
LASSO and Ridge help improve performance for models with large number of features, where they are able to simplify the model leading to better generalization capabilities. However, LASSO and Ridge harm performance when there are few features to choose from, with the norm regularization holding back the models ability to predict the output with its limited features.

Question 3

Question 3.1

```
par(mfrow=c(2,1))

data <- read.table(
            "/Users/vladislavtrukhin/Downloads/A3_data/LIGO.Hanford.Data.txt")
theory <- read.table(
            "/Users/vladislavtrukhin/Downloads/A3_data/LIGO.Hanford.Theory.txt")
set.seed(10)

# Plot time series
plot(data[,1], data[,2], type='l')
title("Data")

# Construct inverse discrete cosine transform matrix
Tn <- nrow(data)
C <- matrix(, nrow = Tn, ncol = Tn)
for (i in 1:nrow(C)) {
  for (j in 1:ncol(C)) {
    if (j == 1) {
      C[i, j] = sqrt(1/Tn)
    } else {
      C[i, j] = sqrt(2/Tn)*cos(pi*(2*i-1)*(j-1)/(2*Tn))
    }
  }
}

# Fit, w_hat, and plot predicted values
lasso1 <- cv.glmnet(C, as.matrix(data[2]), alpha=1, nfolds=10)

plot(data[,1], predict(lasso1, C, s="lambda.1se"), type='l')
title("LASSO")
```






```r
# Alternate estimator C_inv*y
C_inv = solve(C)

# Counting number of sparse coefficients in w_hat
sum(ifelse(abs(coef(lasso1, s="lambda.min")) < 0.01, 1, 0))
```

```
## [1] 2823
```

```r
# Counting number of sparse coefficients in C_inv*y
sum(ifelse(abs(C_inv%*%data[,2]) < 0.01, 1, 0))
```

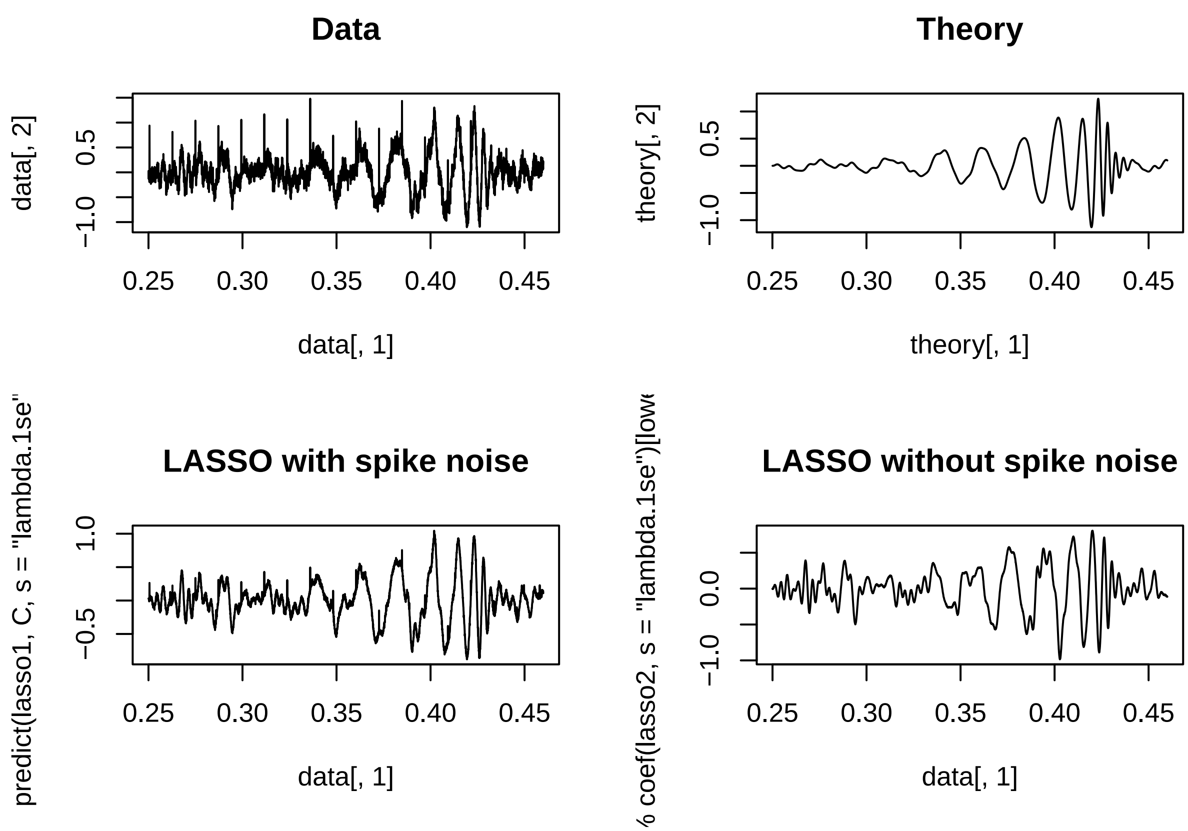
```
## [1] 234
```

$\hat{w}$ is more sparse than $C^{-1}y$ as seen by the count of the coefficients close to 0. $C^{-1}y$ could be less sparse since it directly uses y which is full of noise, obscuring the underlying model which leads to the coefficients predicting both the noise and the data, resulting in the coefficents becoming less sparse. This allows LASSO's $\hat{w}$ to beat out $C^{-1}y$ in sparsity via its sparsity property alone.

Question 3.2

```r
par(mfrow=c(2,2))

# Plot time series, data and theory
plot(data[,1], data[,2], type='l')
title("Data")
plot(theory[,1], theory[,2], type='l')
title("Theory")

# Plot previously predicted values via w_hat from 3.1
```

```
plot(data[,1], predict(lasso1, C, s="lambda.1se"), type='l')
title("LASSO with spike noise")

# Construct design matrix
Phi <- cbind(diag(Tn), C)

# Fit and plot predicted values
lasso2 <- cv.glmnet(Phi, as.matrix(data[2]), alpha=1, nfolds=10)

# With the spike and spectrum of waves modeled via estimators, reconstruct only
# the strength of waves by eliminating the spike estimator and working only with
# spectrum estimator
lower = Tn+1
upper = 2*Tn
plot(data[,1], C%*%coef(lasso2, s="lambda.1se")[lower:upper], type='l')
title("LASSO without spike noise")
```



The method in 3.2 that denoises the spike noise within the data is more meaningful. It is able to smooth out the rough spikes that were left untreated in the method in 3.1, aligning it closer to the theoretical simulation. This can be heard when playing the results as audio, with 3.2 results sounding smoother and closer to the theoretical audio than 3.1.

Question 4

Question 4.1

Suppose $\hat{\beta}$ is a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$ for an arbitrary $\lambda \geq 0$

Suppose $\hat{\beta}$ is not a minimizer for $\frac{1}{2n}||Y - X\beta||_2^2$ subject to $||\beta||_1 \leq C$

Let $C = ||\hat{\beta}||_1$, $\exists \beta^*$ that is a minimizer for $\frac{1}{2n}||Y - X\beta||_2^2$ that satisfies $||\beta^*||_1 \leq C = ||\hat{\beta}||_1$

Case 1:

Suppose $||\beta^*||_1 = ||\hat{\beta}||_1$

Then $\lambda||\beta^*||_1 = \lambda||\hat{\beta}||_1$

Since β^* is a minimizer for $\frac{1}{2n}||Y - X\beta||_2^2$ it is also a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$

$\hat{\beta}$ is also a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$

$\therefore \beta^* = \hat{\beta}$

$\therefore \hat{\beta}$ is a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2$ subject to $||\beta||_1 \leq C = ||\hat{\beta}||_1$

Contradiction

Case 2:

Suppose $||\beta^*||_1 < ||\hat{\beta}||_1$

Then $\lambda||\beta^*||_1 < \lambda||\hat{\beta}||_1$

Since β^* is a minimizer for $\frac{1}{2n}||Y - X\beta||_2^2$ it is also a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$

$\hat{\beta}$ is cannot be a minimizer of $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$ since $\lambda||\beta^*||_1 < \lambda||\hat{\beta}||_1$

Contradiction

Therefore for any $\lambda \geq 0$, $\exists C$ such that $\frac{1}{2n}||Y - X\beta||_2^2$ subject to $||\beta||_1 \leq C$ has the same minimizer as $\frac{1}{2n}||Y - X\beta||_2^2 + \lambda||\beta||_1$

Question 4.2

Add a new row to X, $X_{n+1} = [0, ..., 0]^\top$, such that $X \to \tilde{X}$, and a new row to Y, $Y_{n+1} = \sqrt{\alpha\lambda||\beta||_2^2}$, such that $Y \to \tilde{Y}$

Let $\tilde{\lambda} = \lambda(1 - \alpha)$

$||\tilde{Y} - \tilde{X}\beta||_2^2 + \tilde{\lambda}||\beta||_1 = (\tilde{Y} - \tilde{X}\beta)^\top(\tilde{Y} - \tilde{X}\beta) + \tilde{\lambda}||\beta||_1$

$= (\tilde{Y}^\top - \beta^\top\tilde{X}^\top)(\tilde{Y} - \tilde{X}\beta) + \tilde{\lambda}||\beta||_1$

$= \tilde{Y}^\top\tilde{Y} - \tilde{Y}^\top\tilde{X}\beta - \beta^\top\tilde{X}^\top\tilde{Y} + \beta^\top\tilde{X}^\top\tilde{X}\beta + \tilde{\lambda}||\beta||_1$

$= \tilde{Y}^\top\tilde{Y} - 2\tilde{Y}^\top\tilde{X}\beta + \beta^\top\tilde{X}^\top\tilde{X}\beta + \tilde{\lambda}||\beta||_1$

$= \tilde{Y}^\top\tilde{Y} - 2Y^\top X\beta + \beta^\top X^\top X\beta + \tilde{\lambda}||\beta||_1$

since $X_{n+1}^T\beta = 0 \Rightarrow \tilde{X}\beta = X\beta$ and $Y_{n+1}X_{n+1}^T\beta = 0 \Rightarrow \tilde{Y}^\top\tilde{X}\beta = Y^\top X\beta$

$= Y^\top Y + \alpha\lambda||\beta||_2^2 - 2Y^\top X\beta + \beta^\top X^\top X\beta + \tilde{\lambda}||\beta||_1$

since $Y_{n+1}^2 = \sqrt{\alpha\lambda||\beta||_2^2}^2 = \alpha\lambda||\beta||_2^2 \Rightarrow \tilde{Y}^\top\tilde{Y} = Y^\top Y + \alpha\lambda||\beta||_2^2$

$= Y^\top Y - 2Y^\top X\beta + \beta^\top X^\top X\beta + \alpha\lambda||\beta||_2^2 + \tilde{\lambda}||\beta||_1$

$= (Y - X\beta)^\top(Y - X\beta) + \alpha\lambda||\beta||_2^2 + \lambda(1 - \alpha)||\beta||_1$

substitute $\tilde{\lambda} = \lambda(1 - \alpha)$

$= ||Y - X\beta||_2^2 + \lambda(\alpha||\beta||_2^2 + (1 - \alpha)||\beta||_1)$

$\therefore$ Elastic-Net can be converted into a LASSO problem with the same optimal value via augmenting X and Y with one row of specific values