Statistical Learning HW 6 - Regularized Regression

Christian Conroy

April 4th, 2019

3 points # Book 2ab (3 points)

2. For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

(a) The lasso, relative to least squares, is:

i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

False. Lasso will either be the same as least squares for a λ of 0 or less flexible than least squares for a λ of greater than 0.

ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

False. Lasso will either be the same as least squares for a λ of 0 or less flexible than least squares for a λ of greater than 0.

iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

This is correct depending on the λ selected. As λ increases from 0 up to about 10, the increase in bias will be less than the decrease in variance. Because mean squared error is a function of the variance plus the squared bias, the MSE drops considerably. The smaller the MSE, the greater the prediction accuracy, and therefore less flexibility gives improved prediction accuracy when its increase in bias is less than its decrease in variance under ridge regression.

iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

Not correct. While a high λ may make the model less flexible, if the variance is rising significantly, then it means that a small change in the training data can cause a large change in the least squares coefficient estimate, thereby decreasing prediction accuracy.

(b) Repeat (a) for ridge regression relative to least squares.

i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

False. Ridge Regression will either be the same as least squares for a λ of 0 or less flexible than least squares for a λ of greater than 0.

ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

False. Ridge Regression will either be the same as least squares for a λ of 0 or less flexible than least squares for a λ of greater than 0.

iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

This is correct depending on the λ selected. As λ increases from 0 up to about 10, the increase in bias will be less than the decrease in variance. Because mean squared error is a function of the variance plus the squared bias, the MSE drops considerably. The smaller the MSE, the greater the prediction accuracy, and therefore less flexibility gives improved prediction accuracy when its increase in bias is less than its decrease in variance under ridge regression.

iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

Not correct. While a high λ may make the model less flexible, if the variance is rising significantly, then it means that a small change in the training data can cause a large change in the least squares coefficient estimate, thereby decreasing prediction accuracy.

Extra 50 (3 points)

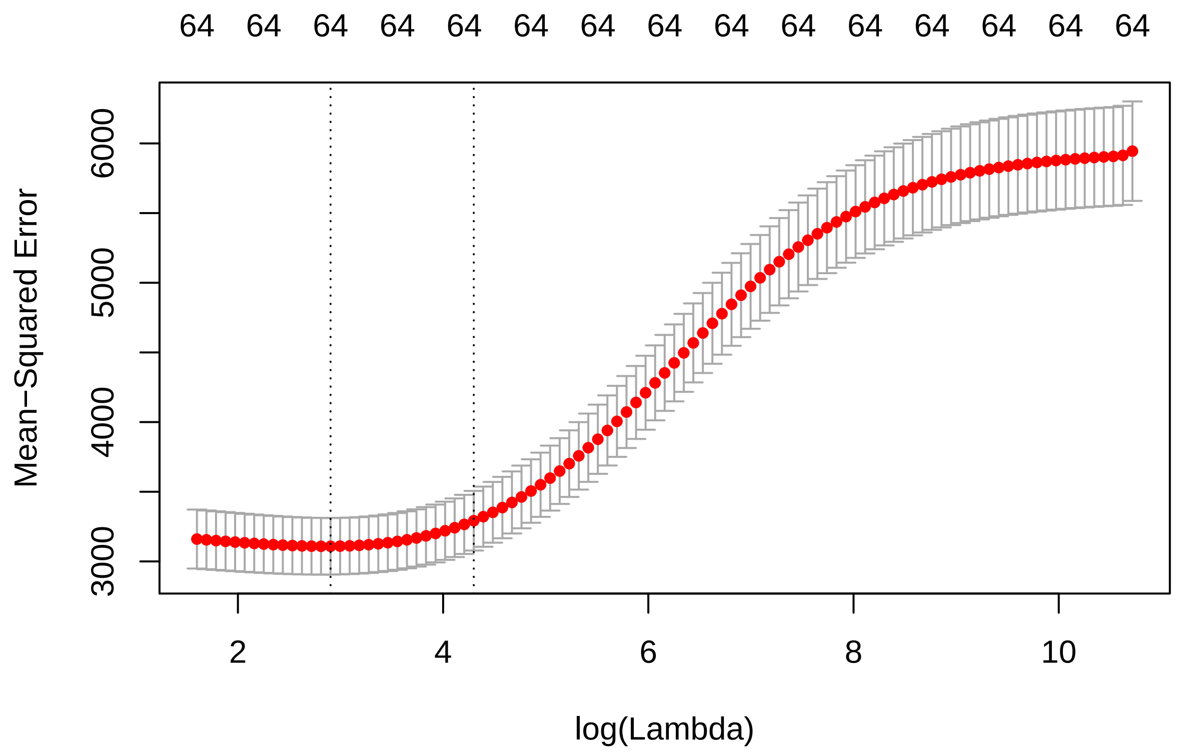
```r
load('diabetes.RData')
df <- as.data.frame(cbind(diabetes$y, diabetes$x2))
```

Use ridge regression and 10-fold cross validation to come up with a good model to predict y. Be sure to set the random seed before doing the cross validation.

```r
X.model <- model.matrix(V1 ~ .-1, data = df)
y <- df$V1

# remove the NAs:
y <- na.omit(y)

# Fit ridge
set.seed(1)
fit.ridge <- cv.glmnet(X.model, y, alpha = 0)
```

a) Plot the mean sqared error estimates and report ??1SE.

```r
plot(fit.ridge)
```



```r
fit.ridge$lambda.1se
```

```
## [1] 73.6
```

b) What is the RMS prediction error according to cross validation for this ???

```r
sqrt(fit.ridge$cvm[fit.ridge$lambda == fit.ridge$lambda.1se])
```

```
## [1] 57.4
```

Book 4abc (3 points)

4. Suppose we estimate the regression coefficients in a linear regression model by minimizing (Ridge)

for a particular value of ??. For parts (a) through (e), indicate which of i. through v. is correct. Justify your answer.

(a) As we increase ?? from 0, the training RSS will:

iii. Steadily increase.

Increasing ?? from 0 means increasing the penalty value and therefore increasing the RSS.

(b) Repeat (a) for test RSS.

ii. Decrease initially, and then eventually start increasing in a U shape.

At first, the variance decreases rapidly and MSE drops considerably, with very little increase in bias. Beyond some point though, the decrease in variance due to increasing ?? slows, and the shrinkage on the coefficients causes them to be significantly underestimated, resulting in a large increase in the bias. The middle point where the curve changes is essentially the best combination of under and over fitting.

(c) Repeat (a) for variance.

iv. Steadily decrease.

As the penalty term increases, the variance steadily decreases as coefficients asymptotically approach 0.

Book 9abcd (5 points)

In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) Split the data set into a training set and a test set.

```r
# Load The Data
data("College")

# Split into train and test
set.seed(12345)
train <- sample(nrow(College),(nrow(College) * .70), replace = FALSE)

College_train <- College[train,]
College_test <- College[-train,]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```r
# Fit the model on the training data
fit <- lm(Apps ~ ., data = College_train)

# Make predictions on the test dataset here
apps.predicted = predict(fit, newdata = College_test)

# Calculate the root mean squared error
sqrt(mean((apps.predicted - College_test$Apps)^2))
```

```
## [1] 996
```

(c) Fit a ridge regression model on the training set, with ?? chosen by cross-validation. Report the test error obtained.

```r
Train.model <- model.matrix(Apps ~ ., data = College_train)
Test.model <- model.matrix(Apps ~ ., data = College_test)

# Fit ridge
fit.ridge <- cv.glmnet(Train.model, College_train$Apps, alpha = 0)

# Get best lambda from fit ridge
fit.ridge.lambda <- fit.ridge$lambda.min

#Predict on Test
apps.predicted <- predict(fit.ridge, s = fit.ridge.lambda, newx = Test.model)

# Calc RMSE
sqrt(mean((apps.predicted - College_test$Apps)^2))
```

```
## [1] 1016
```

The test MSE is higher for ridge regression compared to least squares

(d) Fit a lasso model on the training set, with ?? chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```
# Fit lasso
fit.lasso <- cv.glmnet(Train.model, College_train$Apps, alpha = 1)

# Get best lambda from fit ridge
fit.lasso.lambda <- fit.lasso$lambda.min

#Predict on Test
apps.predicted <- predict(fit.lasso, s = fit.lasso.lambda, newx = Test.model)

# Calc RMSE
sqrt(mean((apps.predicted - College_test$Apps)^2))
```

```
## [1] 995
```

The test MSE is slightly lower compared to least squares and lower than that of Ridge Regression.

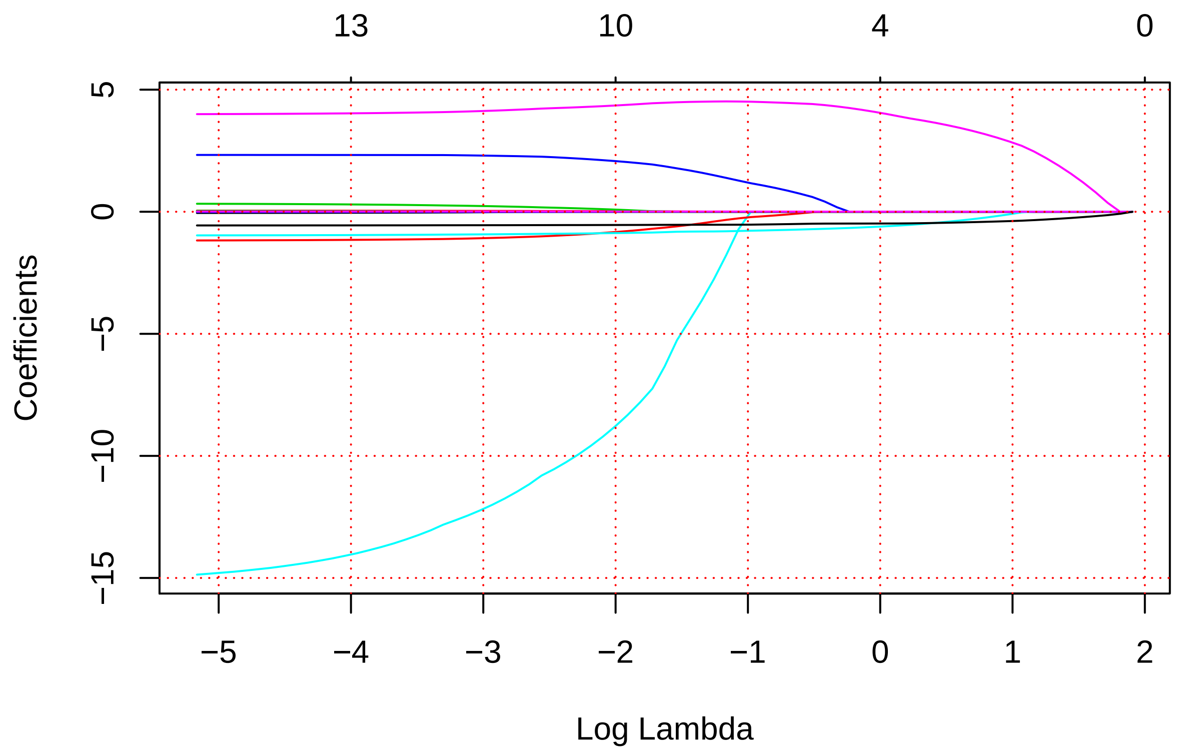
- Extra 49

Consider the Boston data. We want to predict medv from all other predictors, using the LASSO.

```
data("Boston")
```

a) Set up the LASSO and plot the trajectories of all coefficients. What are the last five variables to remain in the model?

```
# Split into train and test
set.seed(12345)
train <- sample(nrow(Boston),(nrow(Boston) * .70), replace = FALSE)

Boston_train <- Boston[train,]
Boston_test <- Boston[-train,]

Train.model <- model.matrix(medv ~ ., data = Boston_train)
Test.model <- model.matrix(medv ~ ., data = Boston_test)

lasso.mod <- glmnet(Train.model, Boston_train$medv, alpha=1)
plot(lasso.mod, xvar = "lambda")
grid(col = 2)
```



```r
# Evaluate last five to remain
Opt <- as.data.frame(as.matrix(lasso.mod$beta))
sapply(Opt, function(x) sum(x > 0))
```

```
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   1   2   2   2   2   2   3   3   3   3   3   3   3   3   3   3   4   4
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##   4   5   5   5   5   5   5   5   5   5   5   5   6   6   6   6   6   6
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71
##   6   6   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7
## s72 s73 s74 s75 s76
##   7   7   7   7   7
```

```r
rownames(Opt[Opt$s37 > 0,])
```

```
## [1] "zn"    "chas"  "rm"    "rad"   "black"
```

The last five variables to remain include zn, chas, rm, rad, and black.

b) Find the 1SE value of ??, using 10-fold cross-validation. What is the cross validation estimate for the residual standard error?

```r
# Fit lasso
fit.lasso <- cv.glmnet(Train.model, Boston_train$medv, alpha = 1)

# Get best lambda from fit ridge
fit.lasso.lambda <- fit.lasso$lambda.min
```

```r
#Predict on Test
medv.predicted <- predict(fit.lasso, s = fit.lasso.lambda, newx = Test.model)

# Report 1se
fit.lasso$lambda.1se
```

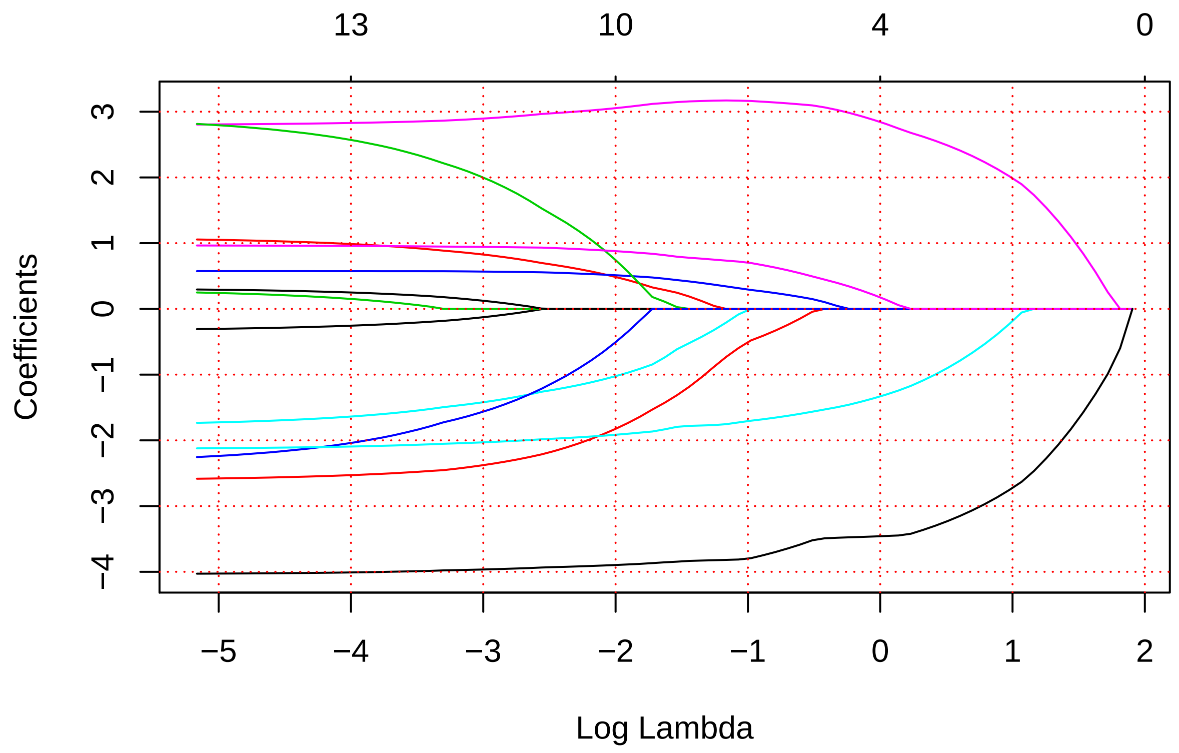
```
## [1] 1.26
```

```r
# Calc RMSE
sqrt(mean((medv.predicted - Boston_test$medv)^2))
```

```
## [1] 4.8
```

c) Rescale all predictors so that their mean is zero and their standard deviation is 1. Then set up the LASSO and plot the trajectories of all coefficients. What are the last five variables to remain in the model? Compare your answer to part a).

```r
# Scale all the predictors
Boston_train[, -c(14)] <- scale(Boston_train[, -c(14)])
Boston_test[, -c(14)] <- scale(Boston_test[, -c(14)])

Train.model <- model.matrix(medv ~ ., data = Boston_train)
Test.model <- model.matrix(medv ~ ., data = Boston_test)

lasso.mod <- glmnet(Train.model, Boston_train$medv, alpha=1)
plot(lasso.mod, xvar = "lambda")
grid(col = 2)
```



```
# Evaluate last five to remain
Opt <- as.data.frame(as.matrix(lasso.mod$beta))
sapply(Opt, function(x) sum(x > 0))
```

```
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   0   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##   1   2   2   2   2   2   3   3   3   3   3   3   3   3   3   3   4   4
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##   4   5   5   5   5   5   5   5   5   5   5   5   6   6   6   6   6   6
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71
##   6   6   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7   7
## s72 s73 s74 s75 s76
##   7   7   7   7   7
```

```
rownames(Opt[Opt$s37 > 0,])
```

```
## [1] "zn"    "chas"  "rm"    "rad"   "black"
```

The last five variables to remain still include zn, chas, rm, rad, and black.

d) Find the 1SE value of ??, using 10-fold cross-validation. What is the cross validation estimate for the residual standard error now? Does rescaling lead to a better performing model?

```
# Fit lasso
fit.lasso <- cv.glmnet(Train.model, Boston_train$medv, alpha = 1)

# Get best lambda from fit ridge
fit.lasso.lambda <- fit.lasso$lambda.min

#Predict on Test
medv.predicted <- predict(fit.lasso, s = fit.lasso.lambda, newx = Test.model)

# Report 1se
fit.lasso$lambda.1se
```

```
## [1] 0.657
```

```
# Calc RMSE
sqrt(mean((medv.predicted - Boston_test$medv)^2))
```

```
## [1] 4.88
```

We end up with the same MSE, but a lower L1SE.

- Extra 52

The LASSO also works for logistic models. We can therefore use it for the MNIST image classification data, available as mnist_all.RData that were used earlier. We want to distinguish between 1 and 8. Extract the relevant training data and place them in a data frame. Remove all variables (pixels) that have zero variance, i.e. pixels that have the same value for both digits. The response variable should have values 0 (for digit = 1) and 1 (for digit = 8).

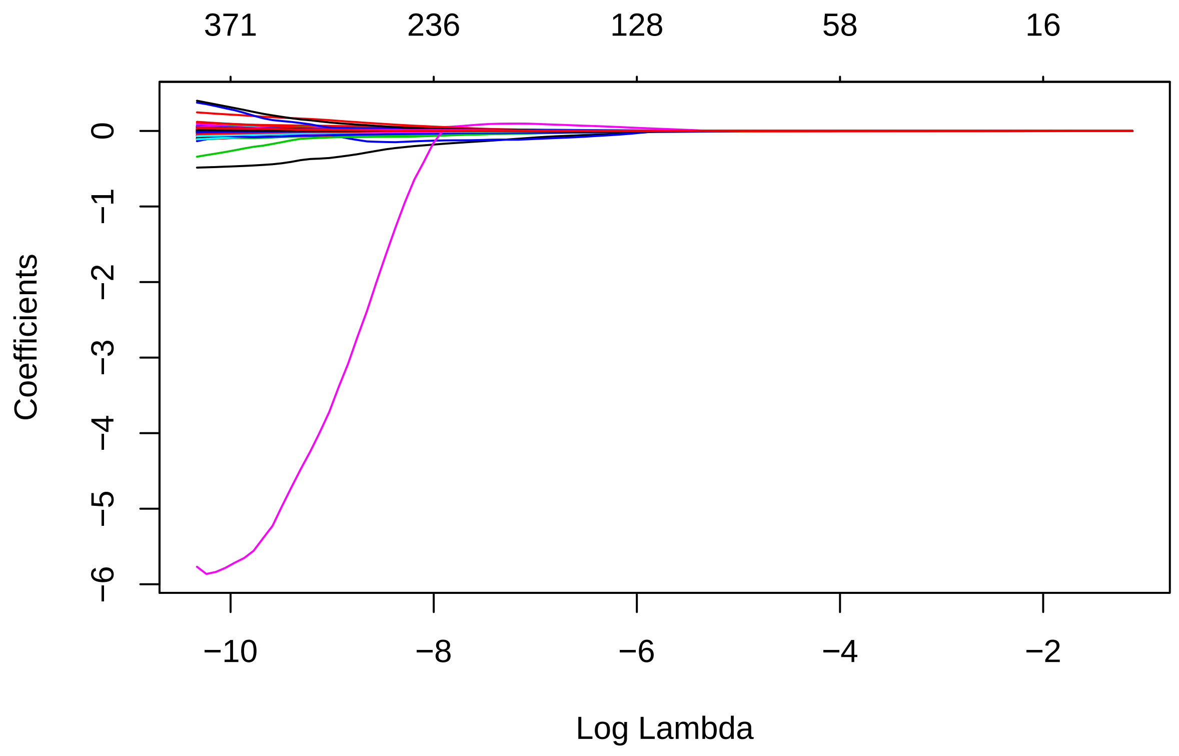
Look up the help file to find out how to use glmnet for logistic regression

```
mnist <-load('mnist_all.RData')

# Remove NAs
mnist <- na.omit(mnist)
```

```
# Train
mnist_train <- data.frame(train$n, train$x, train$y)
mnist_train <- mnist_train[mnist_train$train.y == 1 | mnist_train$train.y == 8,]
mnist_train$train.y <- as.factor(ifelse((mnist_train$train.y == 8), 1, 0))
mnist_train <- mnist_train[ - as.numeric(which(apply(mnist_train, 2, var) == 0))]

# Test
mnist_test <- data.frame(test$n, test$x, test$y)
mnist_test <- mnist_test[mnist_test$test.y == 1 | mnist_test$test.y == 8,]
mnist_test$test.y <- as.factor(ifelse((mnist_test$test.y == 8), 1, 0))
mnist_test <- mnist_test[ - as.numeric(which(apply(mnist_test, 2, var) == 0))]
```

a) Apply the LASSO and plot the result (trajectories against ??). There are several hundred trajectories, which is not helpful.

```
Train.model <- model.matrix(train.y ~ ., data = mnist_train)[,-621]
Test.model <- model.matrix(test.y ~ ., data = mnist_test)[,-534]

lasso.mod <- glmnet(Train.model, y= mnist_train$train.y, alpha=1, family="binomial", lambda = NULL)

plot(lasso.mod, xvar="lambda")
```



b) Identify the last ten variables that leave the model. Determine in which order they leave the model.

```
# Evaluate last ten to remain

Opt <- as.data.frame(as.matrix(lasso.mod$beta))
```

```
sapply(Opt, function(x) sum(x > 0))
```

```
##  s0  s1  s2  s3  s4  s5  s6  s7  s8  s9 s10 s11 s12 s13 s14 s15 s16 s17
##   0   4   4   6   7   8   9  11  13  13  15  18  19  20  21  21  23  23
## s18 s19 s20 s21 s22 s23 s24 s25 s26 s27 s28 s29 s30 s31 s32 s33 s34 s35
##  25  26  29  30  31  34  36  39  43  44  45  49  50  51  50  52  52  54
## s36 s37 s38 s39 s40 s41 s42 s43 s44 s45 s46 s47 s48 s49 s50 s51 s52 s53
##  57  60  60  70  74  77  81  84  86  87  88  90  96  97  96  98  99  99
## s54 s55 s56 s57 s58 s59 s60 s61 s62 s63 s64 s65 s66 s67 s68 s69 s70 s71
##  99 104 105 106 110 114 115 119 119 124 130 133 135 137 137 142 145 146
## s72 s73 s74 s75 s76 s77 s78 s79 s80 s81 s82 s83 s84 s85 s86 s87 s88 s89
## 147 152 155 156 159 162 161 163 164 165 167 165 167 174 178 179 184 186
## s90 s91 s92 s93 s94 s95 s96 s97 s98 s99
## 190 193 195 197 202 205 207 208 205 204
```

```
rownames(Opt[Opt$s7 > 0,])
```

```
##  [1] "X236" "X263" "X292" "X300" "X320" "X328" "X348" "X355" "X376" "X404"
## [11] "X410"
```

```
Opt[Opt$s7 > 0, c("s7")]
```

```
##  [1] 1.82e-03 1.58e-04 4.47e-04 1.19e-04 1.13e-03 7.99e-05 9.23e-04
##  [8] 1.57e-03 1.07e-03 1.36e-03 2.43e-04
```

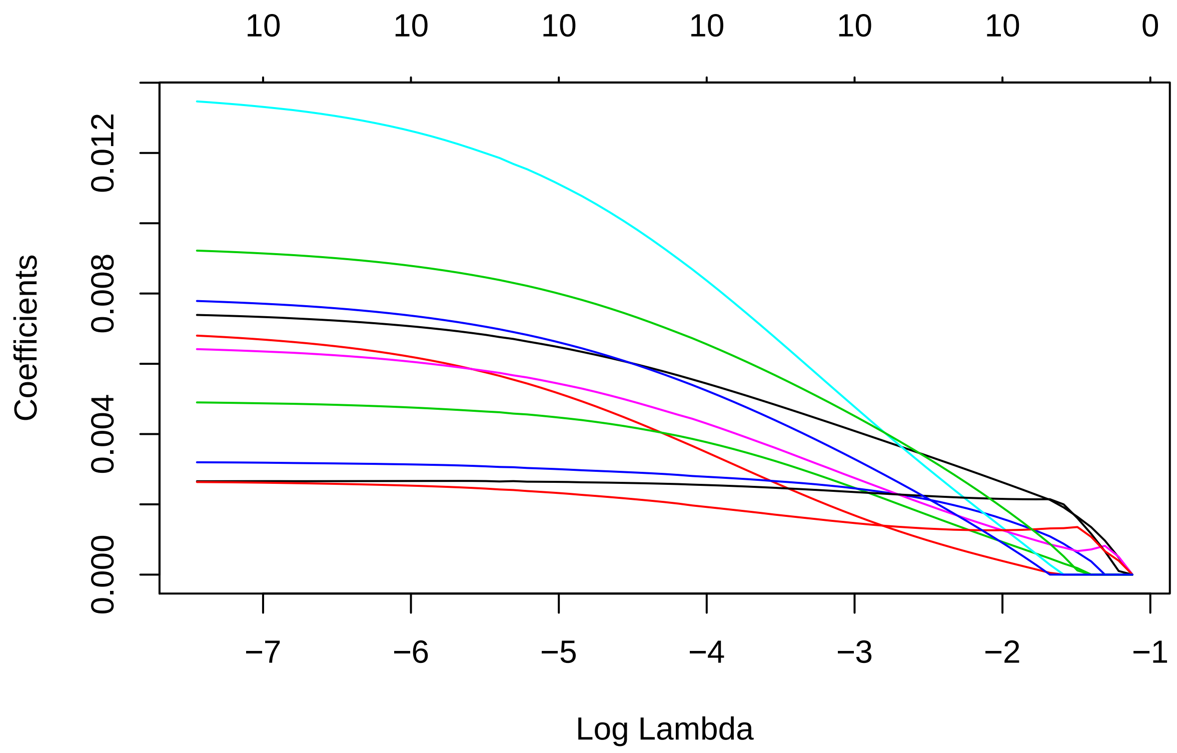
```
# 11 left in s7, so eliminated lowest coefficient to get to 10.
```

The last ten variables left are “X236” “X263” “X292” “X320” “X328” “X348” “X355” “X376” “X404” “X410”.

Why in the above is X264 in s6 but not in s7?

c) Find a way to make a trajectory plot of the coefficients only for the last ten variables that leave the model.

```
Train.model <- model.matrix(train.y ~ X236 + X263 + X292 + X320 + X328 + X348 + X355 + X376 + X404 + X4

lasso.mod <- glmnet(Train.model, y= mnist_train$train.y, alpha=1, family="binomial", lambda = NULL)

plot(lasso.mod, xvar="lambda")
```



10
10
10
10
10
10
0
Coefficients
0.000
0.004
0.008
0.012
−7
−6
−5
−4
−3
−2
−1
Log Lambda