Homework 4 - STAT 5443

Daniel Kiser

April 25, 2018

Problem 1

To compare the performance of Lasso, Ridge, and Elastic Net, we first create a simulated data set:

```
set.seed(1234)

n <- 100
p <- 500
p0 <- 150

B <- c(rep(5, p0), rep(0, p-p0))
X <- scale(matrix(rnorm(n*p, 0, 1), n, p))
epsilon <- rnorm(n, 0, 0.1)
fx <- X%*%B
y <- drop(fx + epsilon)
```

We also separate the data set into training and testing portions:

```
train_indices <- sample(seq(1:n), n/2)
trainX <- X[train_indices,]
trainy <- y[train_indices]
testX <- X[-train_indices,]
testy <- y[-train_indices]
```

Part 1:

We fit the three different regularization features and plot the solution path for each versus λ:

```
library(glmnet)
```

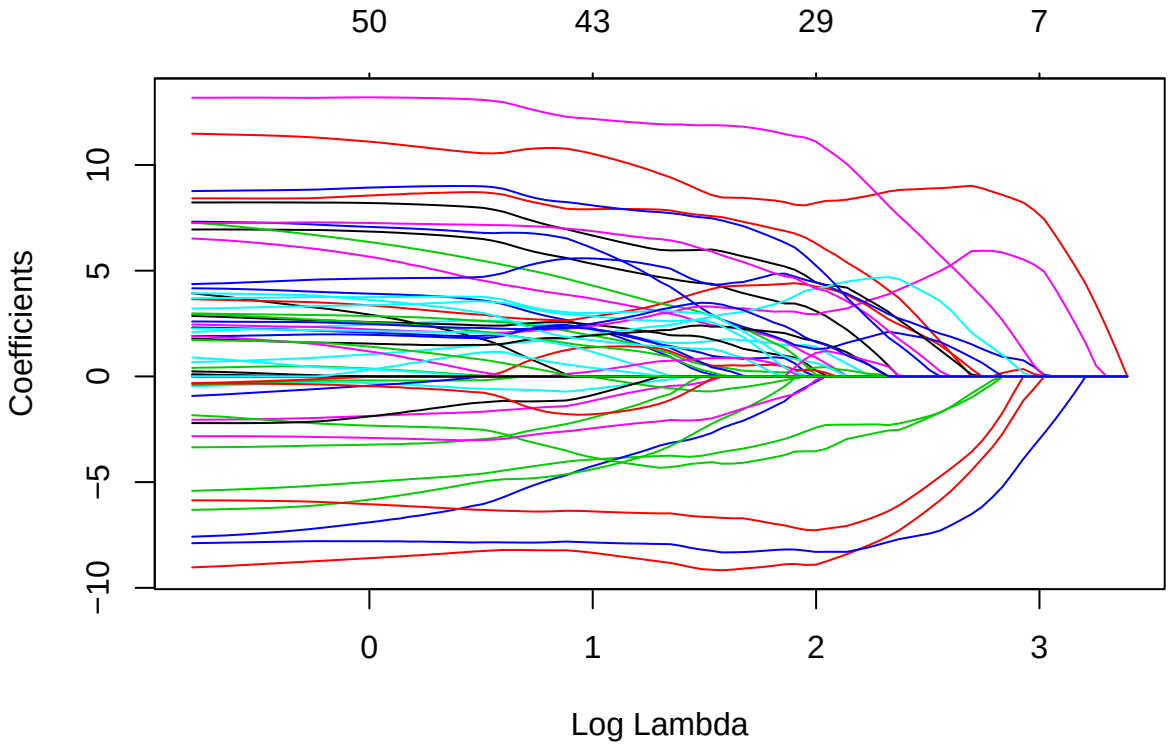
```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

Lasso

```
cv.lasso <- cv.glmnet(x = trainX, y = trainy, alpha = 1)
plot(cv.lasso$glmnet.fit, xvar = "lambda")
```

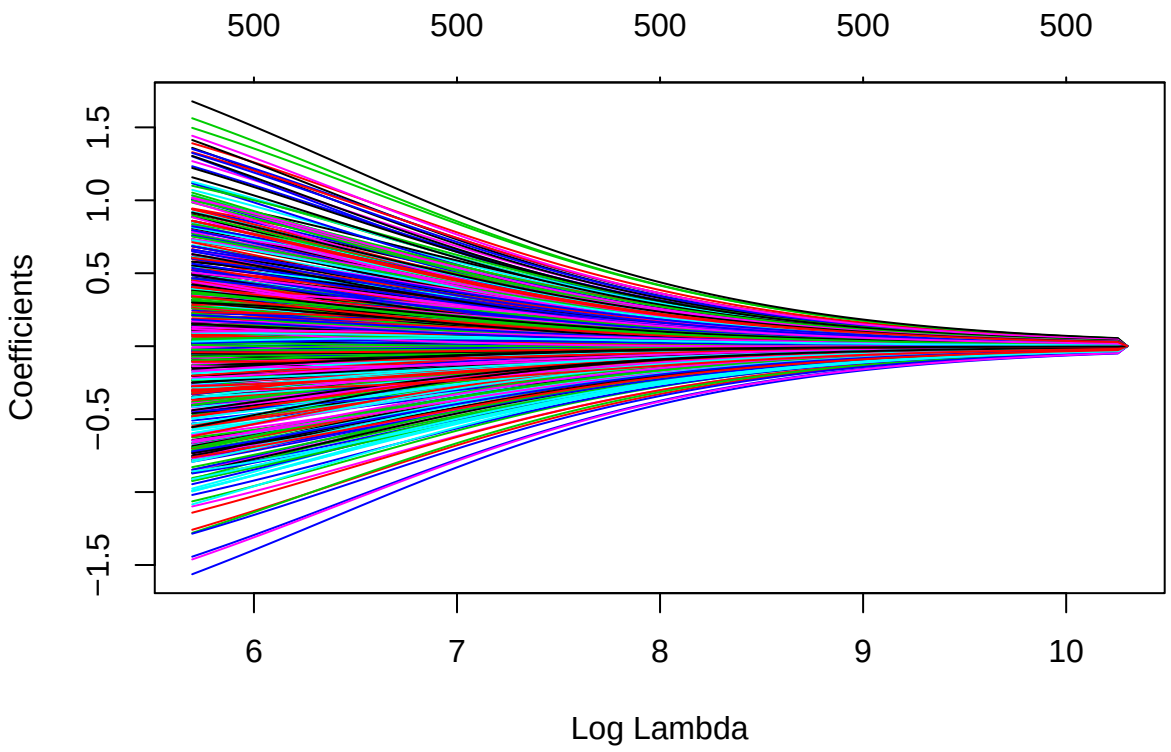



```
lasso_lambda <- cv.lasso$lambda.min
```

For the solution path for Lasso, we note that as λ increases, some coefficients are reduced to zero while other coefficients remain large. For a certain value of λ, all the coefficients are reduced to zero. In contrast, in Ridge regression all of the coefficients are shrunk but they never quite reach zero, even for large values of λ. This can be seen in the solution path for Ridge regression below:

Ridge

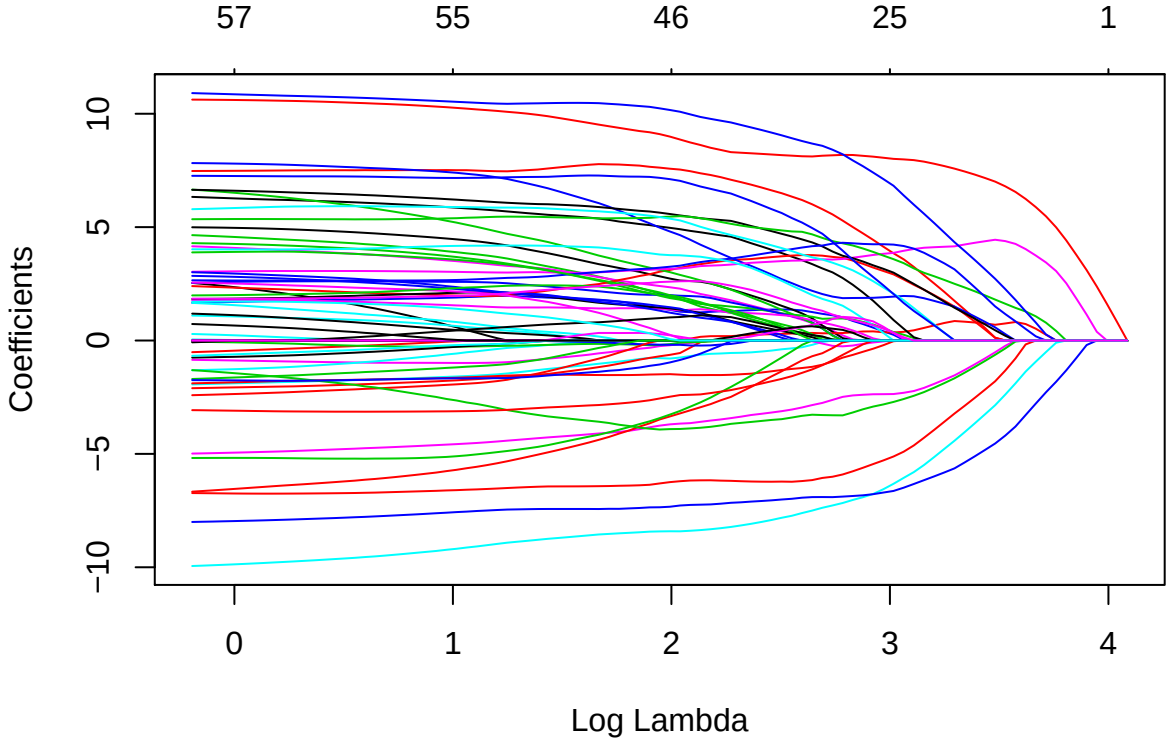
```
cv.ridge <- cv.glmnet(x = trainX, y = trainy, alpha = 0)
plot(cv.ridge$glmnet.fit, xvar = "lambda")
```




```
ridge_lambda <- cv.ridge$lambda.min
```

Elastic Net

```
cv.elastic <- cv.glmnet(x = trainX, y = trainy, alpha = 0.5)
plot(cv.elastic$glmnet.fit, xvar = "lambda")
```



```
elastic_lambda <- cv.elastic$lambda.min
```

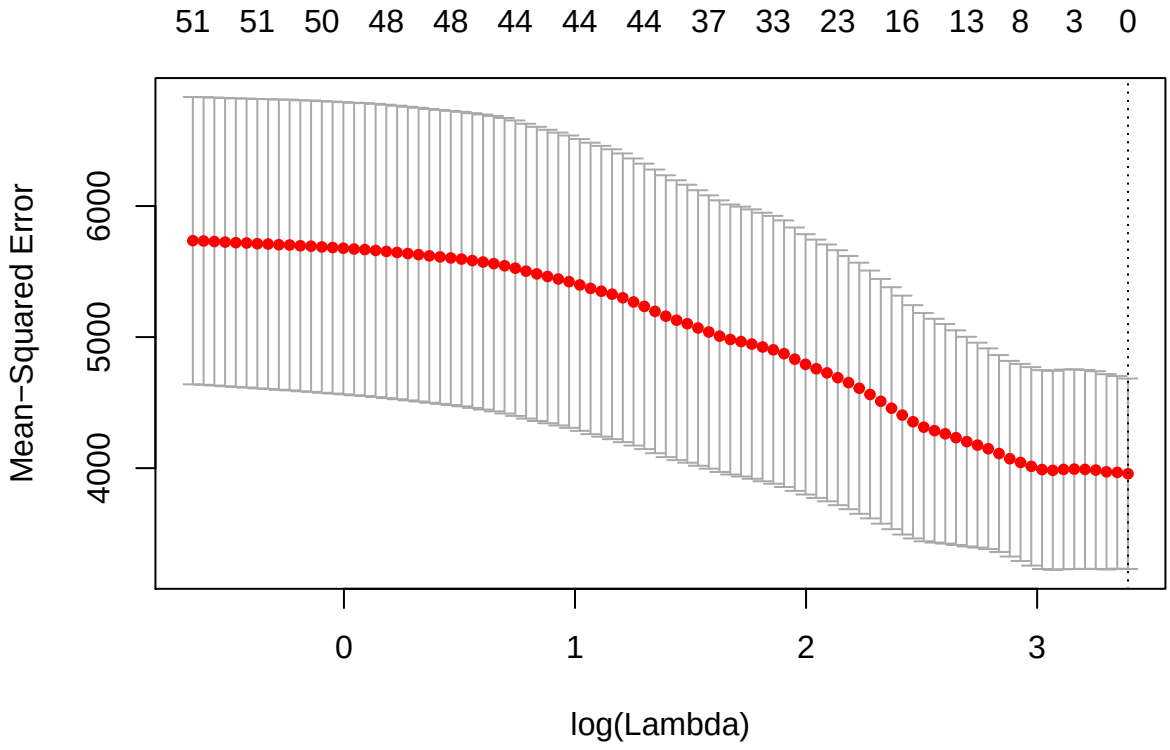
The solution path for Elastic Net looks very similar to that of Lasso, except that a larger value of λ is required to reduce all of the coefficients to zero.

Part 2:

We plot the cross-validation error for each method:

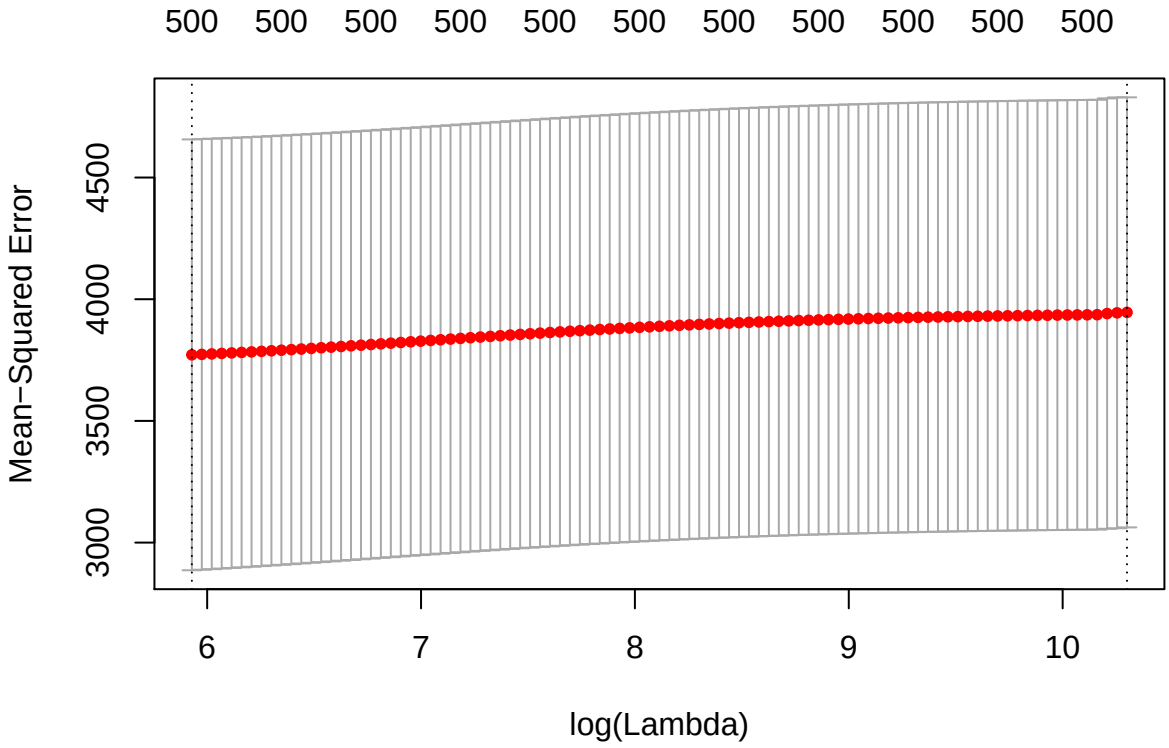
Lasso

```
plot(cv.lasso)
```



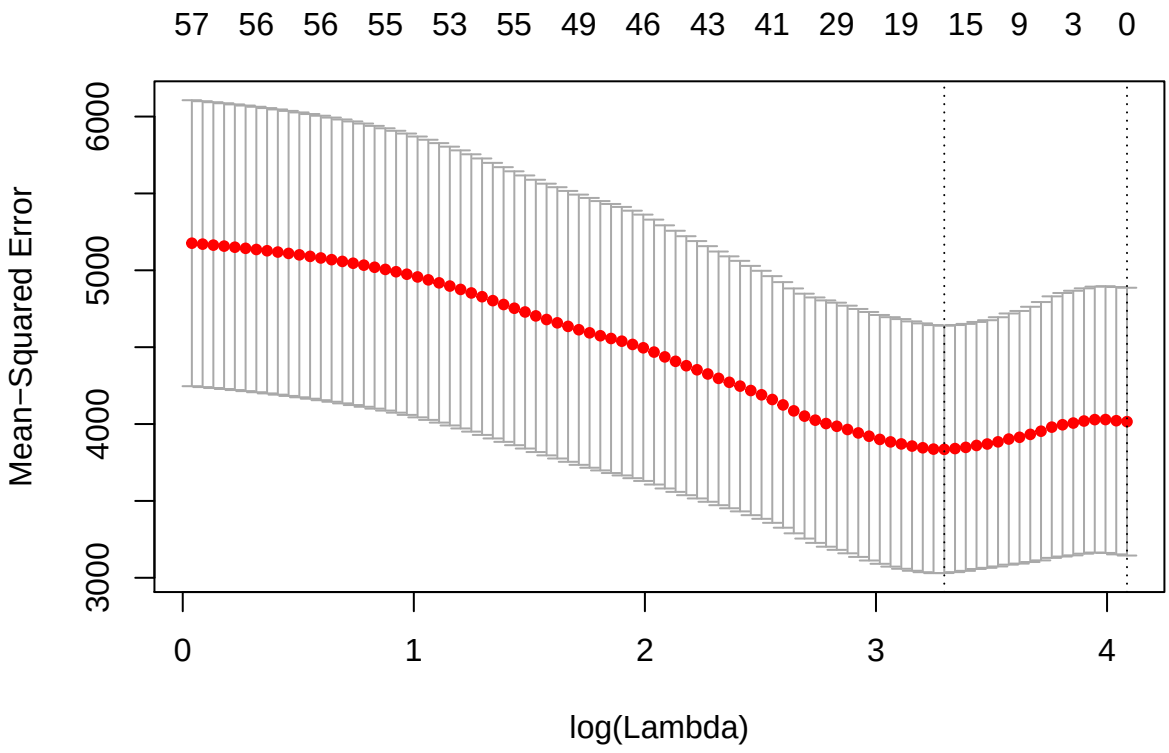
Ridge

```
plot(cv.ridge)
```



Elastic Net

```
plot(cv.elastic)
```



Part 3:

We note that there is some degree of variability in the estimated value of λ for each method that seems to be due to differences in how the folds are determined. This problem may be due to the fact that we only have 50 observations in our training set, making the MSE highly dependent on how the algorithm divides up the folds (which each include only 5 observations). Thus, we run cross-validation 40 times for each method and take the mode of the estimated values of λ as the best estimate:

```
M <- 40
lasso <- numeric(M)
ridge <- numeric(M)
elastic <- numeric(M)

# Perform cross-validation 40 times
for(i in 1:M) {
  cv.lasso <- cv.glmnet(x = trainX, y = trainy, alpha = 1)
  cv.ridge <- cv.glmnet(x = trainX, y = trainy, alpha = 0)
  cv.elastic <- cv.glmnet(x = trainX, y = trainy, alpha = 0.5)
  lasso[i] <- cv.lasso$lambda.min
  ridge[i] <- cv.ridge$lambda.min
  elastic[i] <- cv.elastic$lambda.min
}

# Function to find mode
findmode <- function(x) {
  uniqval <- numeric(length(x))
  count <- numeric(length(x))
  for(i in 1:length(x)) {
    if(x[i] %in% uniqval) {
      index <- match(x[i], uniqval)
      count[index] <- count[index] + 1
    } else {
      uniqval[i] <- x[i]
    }
  }
  mode_index <- match(max(count), count)
  mode <- uniqval[mode_index]
  return(mode)
}
lasso_lambda <- findmode(lasso)
ridge_lambda <- findmode(ridge)
elastic_lambda <- findmode(elastic)
```

Using our best estimates of λ, we measure the prediction error for each model.

```
lasso.pred <- predict(cv.lasso, s = lasso_lambda, newx = testX)
ridge.pred <- predict(cv.ridge, s = ridge_lambda, newx = testX)
elastic.pred <- predict(cv.elastic, s = elastic_lambda, newx = testX)
mean((lasso.pred - testy)^2)
```

```
## [1] 4008.47
```

```
mean((ridge.pred - testy)^2)
```

```
## [1] 3371.013
```

```r
mean((elastic.pred - testy)^2)
```

```
## [1] 3974.627
```

Since Ridge regression resulted in the smallest MSE, it appears that Ridge regression performed the best on this particular simulated dataset.

Problem 2

Since

$$\hat{\beta}^{Lasso} = argmin_{\beta \in R^n} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda \sum_{i=1}^{n} |\beta_i| = argmin_{\beta \in R^n} \sum_{i=1}^{n} [(y_i - \beta_i)^2 + \lambda |\beta_i|]$$

we find $\hat{\beta}_j^{Lasso}$ by minimizing

$$(y_j - \beta_j)^2 + \lambda |\beta_j|$$

with respect to β_j. First we consider the case where $\beta_j > 0$. In that case,

$$(y_j - \beta_j)^2 + \lambda |\beta_j| = (y_j - \beta_j)^2 + \lambda \beta_j$$

We take the derivative with respect to β_j and set the derivative equal to zero:

$$\frac{\partial}{\partial \beta_j} (y_j - \beta_j)^2 + \lambda \beta_j = -2(y_j - \beta_j) + \lambda = 0$$

$$y_j - \beta_j = \frac{\lambda}{2}$$

$$-\beta_j = -y_j + \frac{\lambda}{2}$$

$$\hat{\beta}_j = y_j - \frac{\lambda}{2}$$

Thus, if the true parameter $\beta_j > 0$, then the estimated parameter must be $\hat{\beta}_j = y_j - \frac{\lambda}{2}$. In order for the constraint, $\beta_j > 0$ to be satisfied, $y_j > \lambda/2$.

Now we consider the case where $\beta_j < 0$. In that case,

$$(y_j - \beta_j)^2 + \lambda |\beta_j| = (y_j - \beta_j)^2 - \lambda \beta_j$$

since if β_j is negative, $+\lambda |\beta_j| = -\lambda \beta_j$. We take the derivative with respect to β_j and set the derivative equal to zero:

$$\frac{\partial}{\partial \beta_j} (y_j - \beta_j)^2 - \lambda \beta_j = -2(y_j - \beta_j) - \lambda = 0$$

$$y_j - \beta_j = -\frac{\lambda}{2}$$

$$-\beta_j = -y_j - \frac{\lambda}{2}$$

$$\hat{\beta}_j = y_j + \frac{\lambda}{2}$$

Thus, if the true parameter $\beta_j < 0$, then the estimated parameter must be $\hat{\beta}_j = y_j + \frac{\lambda}{2}$. In order for the constraint, $\beta_j < 0$ to be satisfied, $y_j < -\lambda/2$.

From the work we have done so far, we know that if $\beta_j \neq 0$, then

$$y_j < -\frac{\lambda}{2} \;\; or \;\; y_j > \frac{\lambda}{2}$$

Thus, we conclude that if the true parmeter $\beta_j = 0$, then

$$-\frac{\lambda}{2} < y < \frac{\lambda}{2}$$

or

$$|y_j| < \frac{\lambda}{2}$$

If we evaluate the original equation at $\beta_j = 0$, we get

$$\hat{\beta}_j = argmin_{\beta \in R^n} \sum_{i=1}^{n} (y_j - 0)^2 + \lambda \times 0$$

$$= argmin_{\beta \in R^n} \sum_{i=1}^{n} y_j^2$$

Of course, taking the derivative of y_j^2 with respect to β_j results in 0, so

$$\hat{\beta}_j = 0$$

if $|y_j| < \lambda/2$.

Thus, we conclude that

$$\hat{\beta}_j = y_j - \frac{\lambda}{2} \; if \; y_j > \lambda/2$$

$$\hat{\beta}_j = y_j + \frac{\lambda}{2} \; if \; y_j < -\lambda/2$$

$$\hat{\beta}_j = 0 \; if \; |y_j| < \lambda/2$$