Linear Model Selection and Regularization

Anuraag Girdhar

1.

Generate data set with p = 20 features, n = 1000, and response vector Y = XB + e, where some B = 0

```
library(glmnet)
library(caret)

set.seed(1234)
x <- matrix(rnorm(1000 * 20), 1000, 20)
b <- rnorm(20)
b[c(1, 3, 5, 7, 9, 11, 13, 15, 17, 19)] <- 0
error <- rnorm(1000)
y <- x %*% b + error
```

2.

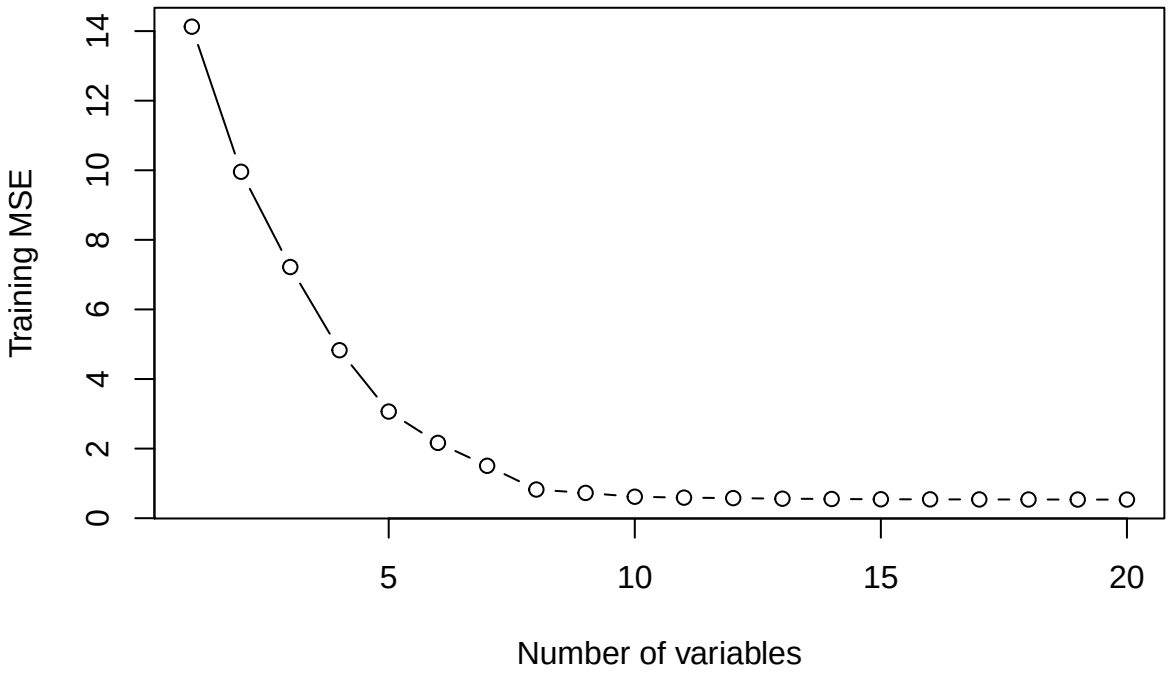
Split dataset into training set with 100 observations and test set with 900 observations

```
train <- sample(seq(1000), 100, replace = FALSE)
test <- -train
x.train <- x[train,]
x.test <- x[test,]
y.train <- y[train]
y.test <- y[test]
```

3.

Training set MSE for best model of each size

```
data.train <- data.frame(y = y.train, x = x.train)
regfit.full <- regsubsets(y ~ .,data.train, nvmax = 20)
train.mat <- model.matrix(y ~ ., data.train, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefficient <- coef(regfit.full, id = i)
    pred <- train.mat[, names(coefficient)] %*% coefficient
    val.errors[i] <- mean((pred - y.train)^2)
}
plot(val.errors, xlab = "Number of variables", ylab = "Training MSE", type="b")
```



The plot above indicates that the MSE significantly slows in its decline once the model reaches 10 variables However, even after 10 variables, the MSE continues to decline slightly, and reaches its true minimum at 20 variables.

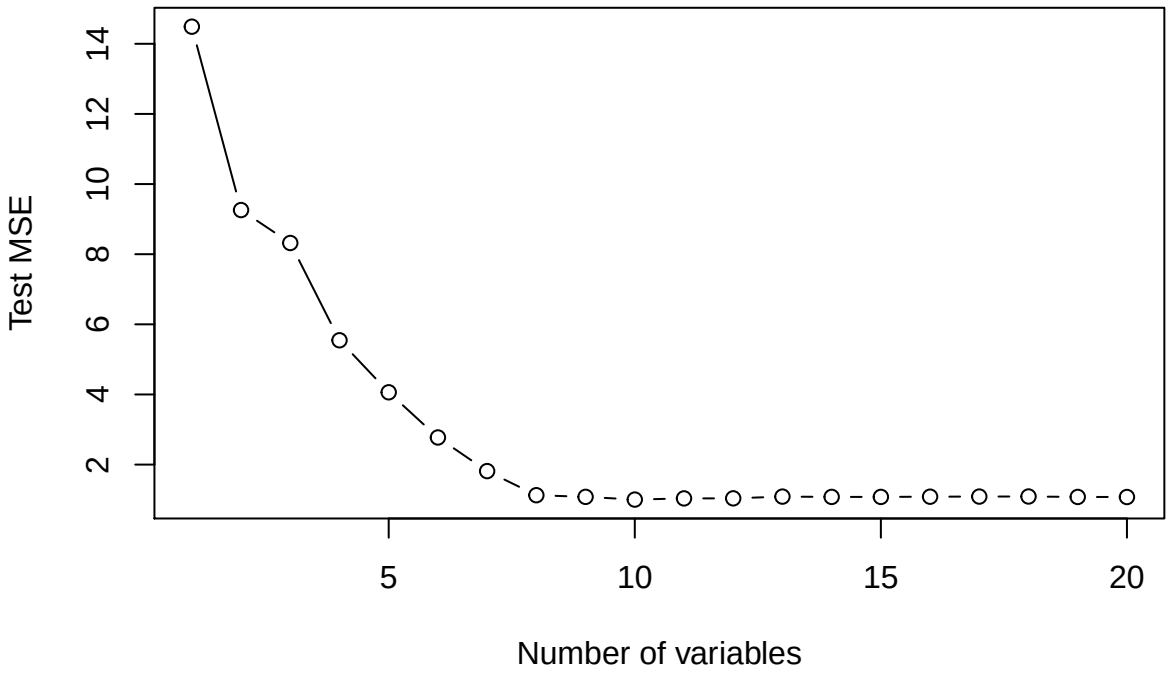
```
which.min(val.errors)
```

```
## [1] 20
```

4.

Testing set MSE for best model of each size

```
data.test <- data.frame(y = y.test, x = x.test)
test.mat <- model.matrix(y ~ ., data.test, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefficient <- coef(regfit.full, id = i)
    pred <- test.mat[, names(coefficient)] %*% coefficient
    val.errors[i] <- mean((pred - y.test)^2)
}
plot(val.errors, xlab = "Number of variables", ylab = "Test MSE", type = "b")
```



5.

For which model size does the test set MSE take on its minimum value?

```
test_set_MSE_min <- which.min(val.errors)
coef(regfit.full, which.min(val.errors))
```

```
## (Intercept)         x.2         x.4         x.6         x.8        x.10
## -0.05473098 -0.95433800  0.32137436 -1.58297915  1.70965364 -1.24856797
##        x.12        x.14        x.16        x.18        x.20
##  0.87474920  0.33889112  1.14728035  2.08250842  1.32044453
```

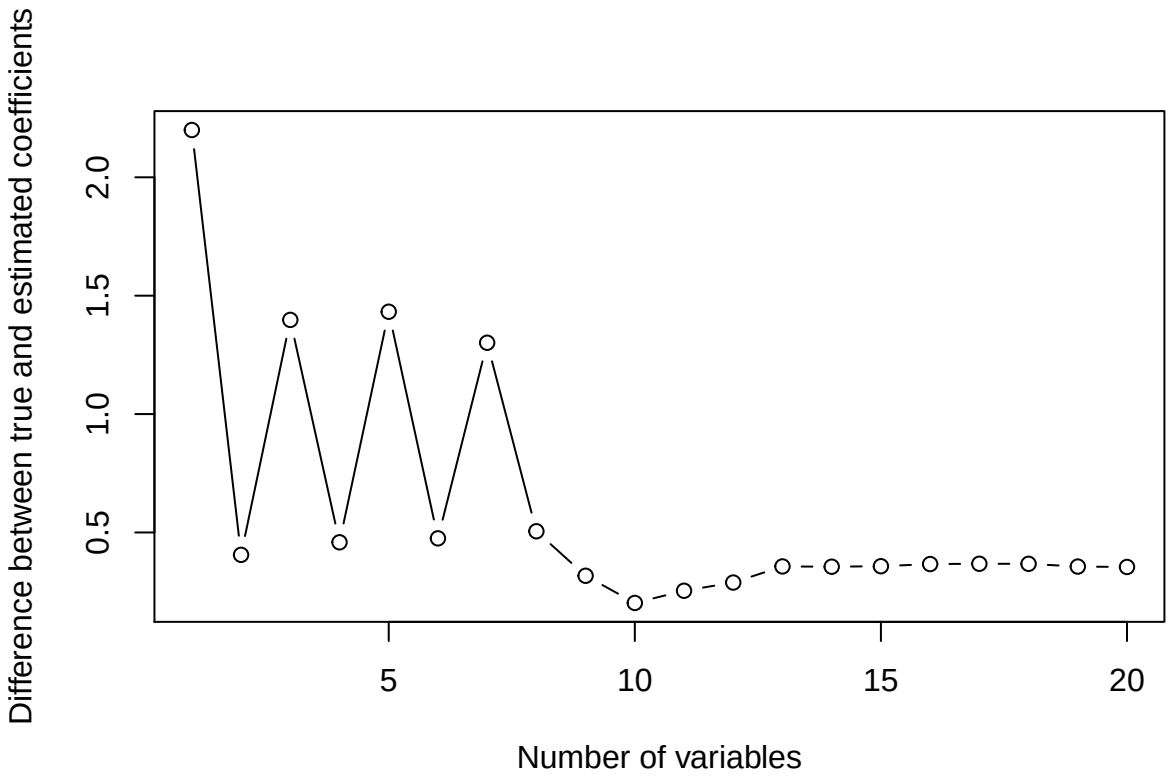
We find that the test set MSE takes on its minimum value at 10 variables.

6.

In the true model used to generate the data, I set half of the beta coefficient equal to 0. Therefore, I would expect that the test set MSE to be minimized near 50% of the number of variables, or 10. The minumum indeed occurs at 10 variables.

7.

```
val.errors <- rep(NA, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:20) {
    coefficient <- coef(regfit.full, id = i)
    val.errors[i] <- sqrt(sum((b[x_cols %in% names(coefficient)]
                               - coefficient[names(coefficient) %in% x_cols])^2)
                          + sum(b[!(x_cols %in% names(coefficient))])^2)
}
plot(val.errors, xlab = "Number of variables",
     ylab = "Difference between true and estimated coefficients", type = "b")
```



This plot displays the error between estimated and true coefficients, and this error remains fairly volatile until reaching 9 coefficients after it stabilizes and begins to increase. This is intuitively also approximately where the MSE begins to level out.

Application Exercises

1.

Finding the test MSE for least squares model

```
egal_test <- na.omit(read.csv("data/gss_test.csv"))
egal_train <- na.omit(read.csv("data/gss_train.csv"))
```

```
least_squares <- glm(egalit_scale ~ ., data=egal_train, family=gaussian)
ls_test_MSE <- mean((egal_test$egalit_scale - predict.lm(least_squares, egal_test)) ^ 2)
```

The test MSE for the least squares model is `63.2136296`.

2.

```
y_index <- grep("egalit_scale", colnames(egal_train))

ridge <- cv.glmnet(
  x = as.matrix(egal_train[, -y_index]),
  y = as.matrix(egal_train[, y_index]),
  alpha = 0
)

newx = as.matrix(egal_test[, -y_index])
ridge_predictions <- predict(ridge, s = ridge$lambda.min, newx = newx)
```

```
ridge_test_MSE <- mean((egal_test$egalit_scale - ridge_predictions) ^ 2)
```

The test MSE for ridge regression with 10-fold CV is 61.0378066.

3.

```
lasso <- cv.glmnet(
  x = as.matrix(egal_train[, -y_index]),
  y = as.matrix(egal_train[, y_index]),
  alpha = 1
)

lasso_predictions <- predict(lasso, s = lasso$lambda.min, newx = newx)
lasso_test_MSE <- mean((egal_test$egalit_scale - lasso_predictions) ^ 2)
```

The test MSE for lasso regression with 10-fold CV is 61.2700561, and there are 493 nonzero coefficient estimates.

4.

```
# Set the training control
trControl <- trainControl(method = "repeatedcv", number = 10, repeats = 5,
                          search = "random", verboseIter = TRUE)

# tuneLength parameter tests all possible combinations of alpha and lambda
elastic <- train(egalit_scale ~ ., data = egal_train, method = "glmnet",
                 preProcess = c("center", "scale"), tuneLength = 10, trControl = trControl)
elastic$bestTune

elastic_predictions <- predict(elastic, newx)
elastic_test_MSE <- mean((egal_test$egalit_scale - elastic_predictions) ^ 2)
```

For elasticnet regression, the lowest cross-validation MSE is found for alpha = 0.7283905 and lambda = 0.1171241. For that combination, the test MSE is 61.5209152, and there are 493 nonzero coefficient estimates.

5.

```
least_squares_var <- var(predict(least_squares))
ridge_var <- var(ridge_predictions)[1]
lasso_var <- var(lasso_predictions)[1]
elastic_var <- var(elastic_predictions)
```

The scale of mean squared error is on the order of the actual predicted variable squared. If we achieved a mean squared error of 0 for any of the test sets, that would mean that we miraculously predicted the test set with perfect accuracy. Compared to the simplest model (least squares), ridge, lasso ,and elasticnet perform quite similarly. Therefore, we are not able to improve much by using the penalized regression methods. Of course, since elasticnet is just a combination of ridge (L2 norm) and lasso (L1 Norm), the similar MSE for elasticnet intuitively follows.

Recall that MSE is equal to the sum of model variance, model bias, and uncorrelated noise. In our case, we know that least squares variance is 37.4124251, ridge regression variance is 29.0075183, lasso regression variance is 29.0910952, and elasticnet regression variance is 32.7190363. Given my MSE values for each of these approaches, I can say that the models all perform with moderate accuracy, but one cannot make any

strong or specific statements about causality on the basis of mean squared error alone. Based on MSE, the marginally best-performing model is ridge regression.