HUDM6026 Homework_10

Chenguang Pan

April 04, 2023

0.1 ISLR_Chapter 6.10

We have seen that as the number of features used in a model increases, the training error will necessarily decrease, but the test error may not. We will now explore this in a simulated data set.

0.1.1 (a)

Generate a data set with $p = 20$ features, $n = 1{,}000$ observations, and an associated quantitative response vector generated according to the model

MY SOLUTION:

Thinking multiple variate regression in matrix form provides a more efficient way to estimate or simulate. Since generating the multivariate normal distribution needs the connivance matrix and mean vector, like shown in Dr.Keller's in-class R syntax, we need to define them first.

```
> # import the packages
> library(mvtnorm)
> library(clusterGeneration)
>
> # ---------------------
> # STAGE 1 PREPARE
> # ---------------------
> # set the random seed
> set.seed(666)
> # Generate a random covariance matrix with package clusterGeneration
> cov1 <- genPositiveDefMat(dim=20, # create 20 covariates
+                           covMethod = "eigen")
> # Generate a random vector of 20 means from a normal distribution with N(0, 20)
> mns1 <- rnorm(20,0,sd=20)
> # Generate coefficients vector for the output for Y from norm(0, 1)
> coef1 <- rnorm(21, # Beta0 + Beta1 +...+ Beta20
+                0,1) # drawn from a normal distribution N(0,1)
> ### Set ten of them equal to zero
> coef1[sample(2:21, 10, replace = FALSE)] <- 0
>
> # --------------------------------------
> # STAGE 2 BUILD DATA GENERATING FUNCTION
> # --------------------------------------
> dataGen <- function(N){
+   # Generate the X matrix
+   X <- rmvnorm(n=N, mean = mns1,sigma = cov1$Sigma)
```

```
+   # augmenting the X matrix
+   X_aug <- cbind(1,X)
+   # create the output Y with error term vector following the N(0,1)
+   Y <- X_aug %*% coef1 + rnorm(N,0,1)
+   # adjust the output
+   dfOut <-  data.frame(cbind(X,Y))
+   names(dfOut) <- c(paste0("X",1:20), "Y")
+   return(dfOut)
+ }
>
> # ----------------------
> # STAGE 3 GENERATE DATA
> # ----------------------
> df <- dataGen(1000)
> head(df)
          X1        X2       X3        X4        X5        X6        X7
1 -15.654375 -23.63880 23.30226 -9.037281  1.333409 -33.06151 -26.56916
2 -13.886764 -21.96735 27.40210 -8.405507 -1.067594 -30.14542 -21.91764
3 -16.544472 -24.79274 25.22519 -5.824548  2.412756 -32.54058 -24.17569
4 -12.779982 -20.92270 21.60266 -6.448326 -2.055508 -28.52098 -20.26761
5  -9.483628 -21.20340 27.07318 -8.032887  1.118195 -26.48337 -18.48465
6  -9.034796 -26.44232 24.74899 -6.218634  1.071316 -32.66566 -23.47574
         X8        X9       X10      X11       X12      X13       X14      X15
1 -38.65066 -22.00314 -26.93117 16.61306 -14.18094 26.11095 -14.44719 5.456897
2 -34.84637 -20.95672 -29.48340 12.95538 -13.58884 29.53441 -13.16991 3.997159
3 -36.27806 -22.09635 -24.52454 16.02539 -12.39565 28.99292 -11.51038 5.682973
4 -32.40778 -21.28272 -24.59206 17.65221 -13.14092 30.28706 -12.10753 7.614404
5 -34.09399 -20.28110 -24.05303 14.96859 -11.37569 33.35054 -11.67114 6.937878
6 -37.49877 -20.05385 -25.27227 18.48872 -14.45867 28.59119 -10.64753 3.655086
       X16      X17       X18      X19      X20         Y
1 9.945809 7.827704 -27.34912 4.186049 39.31443 -60.59024
2 6.217472 8.767564 -30.60701 4.419715 43.42606 -65.44828
3 7.294496 9.247503 -28.08444 7.811671 45.24822 -62.32994
4 7.385289 8.772892 -30.73892 2.614271 40.79408 -65.53384
5 3.612012 9.087225 -26.15211 5.705587 42.05414 -66.31407
6 4.453720 9.404593 -29.21027 5.087559 42.82547 -63.94680
> dim(df)
[1] 1000   21
```

The data looks good.

0.1.2 (b)

Split your dataset into a training set containing 100 observations and a test set containing 900 observations.

MY SOLUTION:

```
> # generate a random index from 1:1000
> set.seed(666)
> index_rdm <- sample(c(1:1000),100)
> # separate the dataset into train and test dataset
> df_train <- df[index_rdm,]
> df_test <- df[-index_rdm,]
```

```
> dim(df_train)
[1] 100  21
> dim(df_test)
[1] 900  21
```

The randomly-subseted train- and test-dataset look good.

0.1.3 (c)

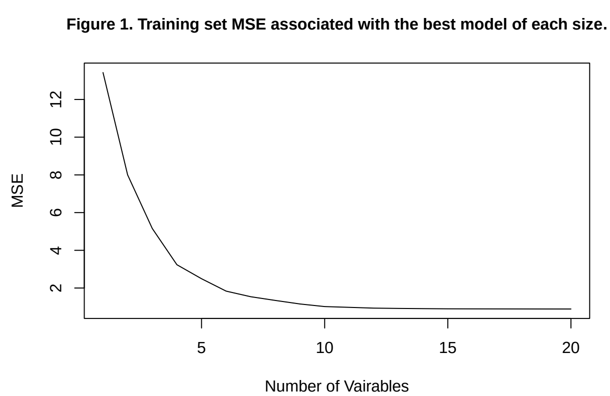
Perform best subset selection on the training set, and plot the training set MSE associated with the best model of each size.

MY SOLUTION:

```
> library(bestglm)
> bss_out <- regsubsets(Y ~., data = df_train, nvmax=20)
> bss_out_summary <- summary(bss_out)
```

Note, by default, the `regsubsets()` only returns the first 8 models.
Based on the definition, we can get easily get the MSE since $MSE = \frac{RSS}{n}$.

```
> plot(bss_out_summary$rss/nrow(df_train),
+      xlab = "Number of Vairables",
+      ylab = "MSE",
+      type = "l")
> title(main = "Figure 1. Training set MSE associated with the best model of each size.",
+       cex.main = 1)
```



0.1.4 (d)

Plot the test set MSE associated with the best model of each size.

MY SOLUTION:
Since there is no `predict()` function built in the `regsubsets()`, I need to write by hand.

```
> # note the test dataset is in data.frame format need to be change to matrix
> df_test_matrix <- as.matrix(df_test)
> names(coef(bss_out, id=2))[-1]
[1] "X19" "X20"
>
> test_mse <- c()
> for (i in 1:20) {
+   # to extract the coefficient vector for each best model
+   coefi <- coef(bss_out, id=i)
+   # select the corresponding variables and times the coefficients
+   X_temp <- df_test_matrix[,names(coefi)[-1]]
+   X_temp_aug <- cbind(1, X_temp)
+   pred <- X_temp_aug %*% coefi
+   # use the estimated vector of outcome to get the MSE
+   mse <- mean((df_test[,"Y"]-pred)^2)
+   test_mse[i] <- mse
+ }
> test_mse
 [1] 14.587205  7.695202  4.852660  3.615263  2.790152  1.987071  1.975713
 [8]  1.442175  1.207953  1.073172  1.151072  1.257745  1.295256  1.316578
[15]  1.350052  1.343764  1.340008  1.357865  1.362500  1.356349
```

The result looks good. Next, I plot the MSE.

```
> plot(test_mse,
+      xlab = "Number of Vairables",
+      ylab = "MSE",
+      type = "l")
> title(main = "Figure 2. Test-set MSE associated with the best model of each size.",
+       cex.main = 1)
> min_index <- which.min(test_mse)
> points(min_index,test_mse[min_index],col="red",cex=2, pch=20)
> abline(h=min(test_mse), col="red",lty="longdash")
```

Figure 2. Test−set MSE associated with the best model of each size.