Mingtao_Gao_HW3

Mingtao Gao

2/9/2020

```
# Packages used in this assignment
library(leaps)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(lattice)
library(ggplot2)
library(glmnet)
```

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

```
## Loaded glmnet 3.0-2
```

Conceptual exercises Training/test error for subset selection 1. Generate dataset

```
# Generate dataset with p=20 and n=1000
set.seed(222)
x <- matrix(rnorm(1000 * 20), 1000, 20)
beta <- rnorm(20)
eps <- rnorm(1000)

# Set up 5 elements in beta as 0
to_zero <- sample(1:10, 5)
for (i in to_zero) {beta[i] <- 0}

# Generate response vector y
y <- x %*% beta + eps
```

2. Split dataset into training set and testing set

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

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

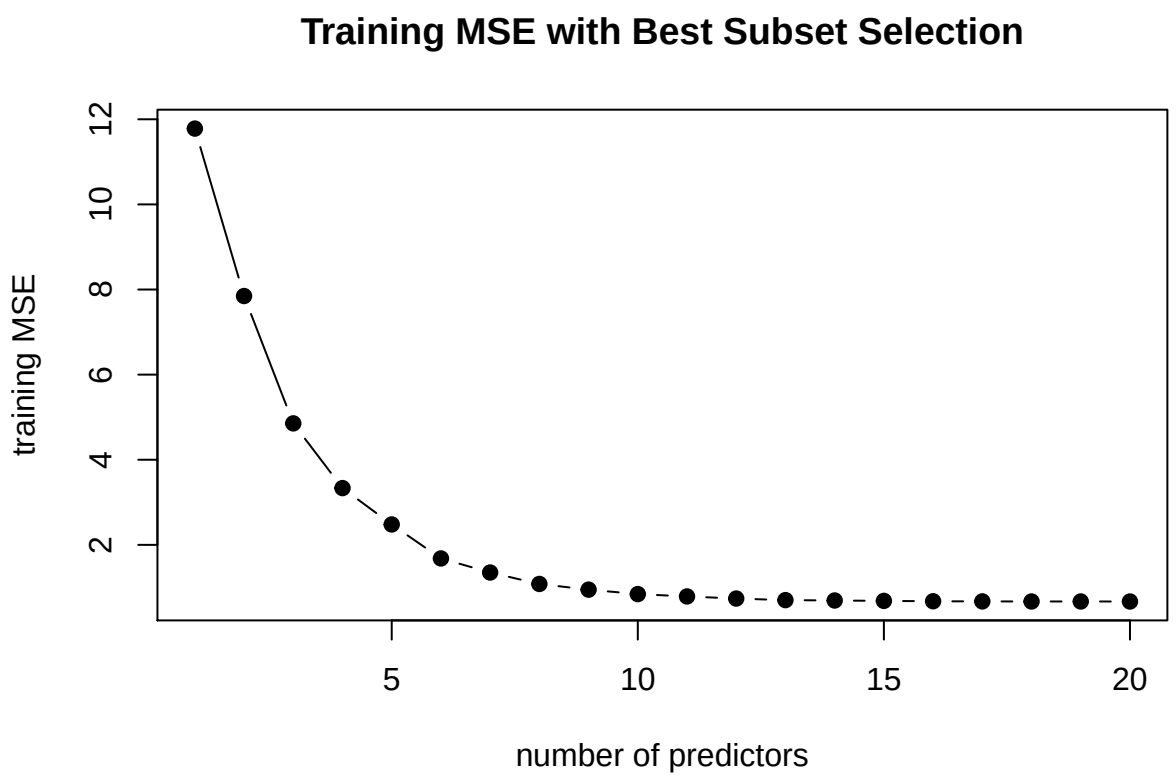
```
# Generate train and test dataframe and matrix for quick access to the data
df.train <- data.frame(y=y.train, x=x.train)
matrix.train <- model.matrix(y~ ., data=df.train, nvmax=20)
df.test <- data.frame(y=y.test, x=x.test)
```

```r
matrix.test <- model.matrix(y~ ., data = df.test, nvmax=20)

# Perform subset selection on training set
regfit.full <- regsubsets(y~ ., data=df.train, nvmax=20)

# Get the MSE of each subset model's prediction on training set
train.mse <- rep(NA, 20)
for (i in 1:20) {
    i_coef <- coef(regfit.full, id=i)
    pred <- matrix.train[, names(i_coef)] %*% i_coef
    train.mse[i] <- mean((pred - y.train) ^ 2)
}

# Plot the training MSE with changing number of predictors
plot(train.mse,
     main="Training MSE with Best Subset Selection",
     xlab="number of predictors",
     ylab="training MSE",
     pch=19,
     type="b")
```




```r
# Check which model size takes the smallest training MSE
which.min(train.mse)
```

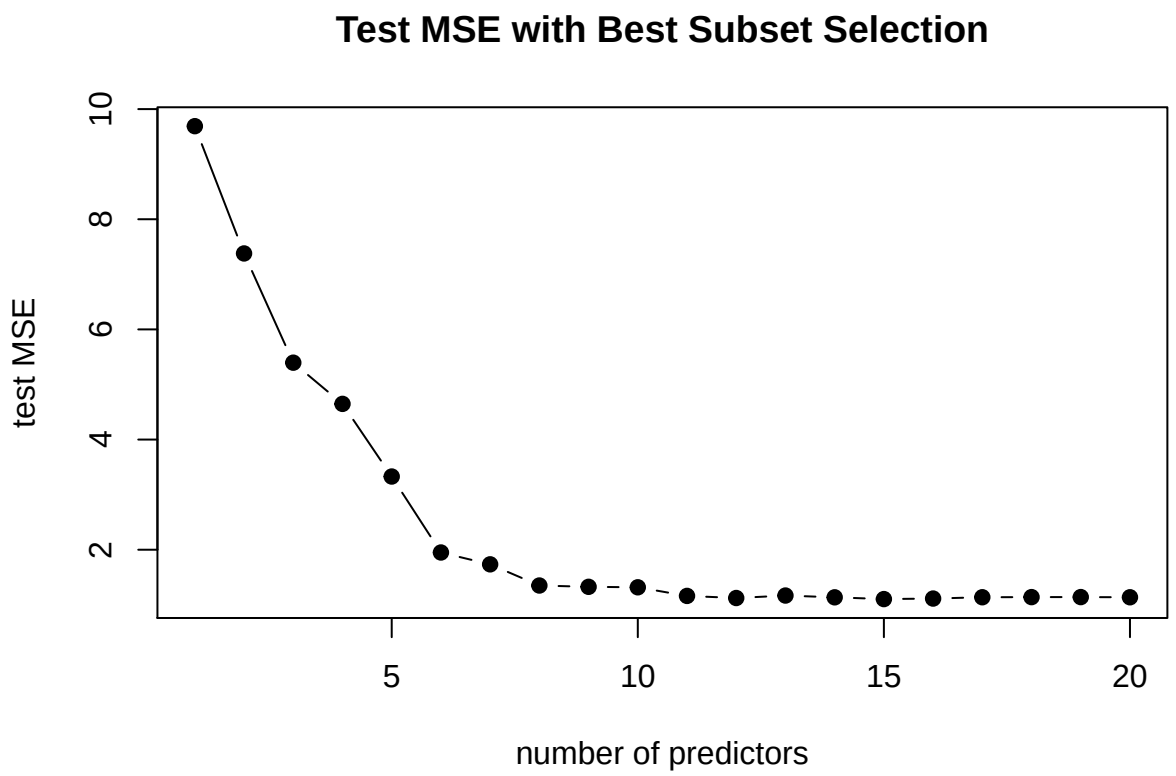
```
## [1] 20
```

Based on the graph generated, we can see as the number of predictors passed into the model, the training MSE decreases. The smallest MSE is when we use all 20 predictors to train the model.

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

```r
# Get the MSE of each subset model's prediction on training set
test.mse <- rep(NA, 20)
for (i in 1:20) {
    i_coef <- coef(regfit.full, id = i)
    pred <- matrix.test[, names(i_coef)] %*% i_coef
    test.mse[i] <- mean((pred - y.test) ^ 2)
}

# Plot the testing MSE with changing number of predictors
plot(test.mse,
     main="Test MSE with Best Subset Selection",
     xlab="number of predictors",
     ylab="test MSE",
     pch=19,
     type="b")
```



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

```r
# Check which model size takes the smallest testing MSE
which.min(test.mse)
```

```
## [1] 15
```

Based on the graph, we found that as the number of predictors increases, the test MSE generally decreases too. However, the smallest test MSE appears when we use 15 predictors to predict response y using testing dataset.

6. How does the model at which the test set MSE is minimized compare to the true model used to generate the data?

```r
# Display the coefficients generated by best performed model (smallest test mse)
coef(regfit.full, which.min(test.mse))
```

```
## (Intercept)         x.2         x.4         x.6         x.7         x.8
##  -0.0113512   0.3828615   1.2877547   1.6025290  -0.9282209   0.6293771
##         x.11        x.12        x.13        x.14        x.15        x.16
##   0.9570763  -0.3438500  -1.9746126   0.6653037   0.2133242   0.1131151
##         x.17        x.18        x.19        x.20
##  -0.2985081  -0.1892843  -0.1238602  -1.6162431
```

We found the coefficient size for the best subset model is 15 predictors with intercept.

```
# Compare the model coefficients with the true beta coefficients
test.coef <- coef(regfit.full, which.min(test.mse))
for (i in seq(1, 20, 1)){
  x.str <- sprintf("x.%d", i)
  cat(x.str, sprintf("%f ",beta[i]), test.coef[x.str], "\n")
  }
```

```
## x.1 0.000000  NA
## x.2 0.298250  0.3828615
## x.3 0.000000  NA
## x.4 1.304504  1.287755
## x.5 0.000000  NA
## x.6 1.549350  1.602529
## x.7 -0.951414  -0.9282209
## x.8 0.517860  0.6293771
## x.9 0.000000  NA
## x.10 0.000000  NA
## x.11 1.073758  0.9570763
## x.12 -0.195219  -0.34385
## x.13 -1.864361  -1.974613
## x.14 0.593225  0.6653037
## x.15 0.029510  0.2133242
## x.16 0.103921  0.1131151
## x.17 -0.368687  -0.2985081
## x.18 -0.161306  -0.1892843
## x.19 -0.234420  -0.1238602
## x.20 -1.405331  -1.616243
```

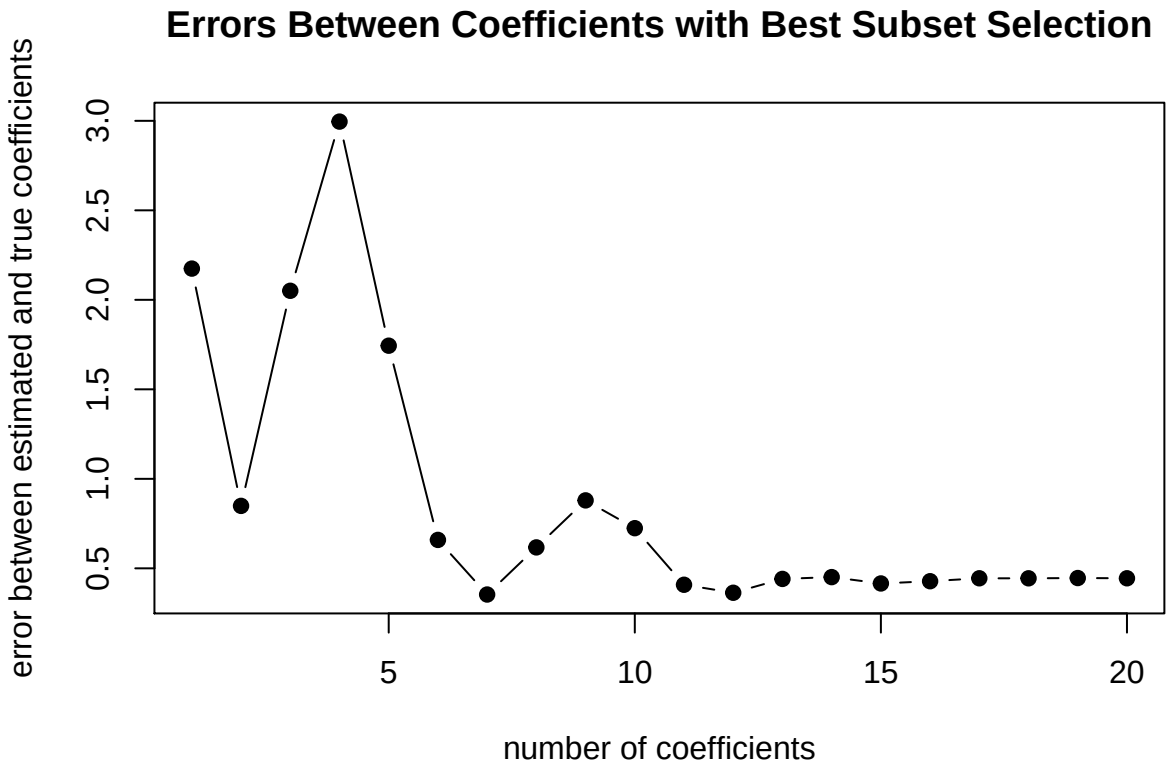
We found that the best subset model (smallest test mse) eliminated all 0 elements in beta we set up in the beginning. From the table, x.1, x.3, x.5, x.9, and x.10 was initially set up as 0 in beta and the 15-predictor model eliminate all of them.

7. Create a plot displaying errors between estimated and true coefficients

```
# Extract columns from x and prefix to better access coefficients from model
x_cols <- colnames(x, do.NULL=F, prefix="x.")

coef.errors <- rep(NA, 20)
for (i in 1:20) {
    i_coef <- coef(regfit.full, id=i)
    # Calculate errors between estimated and true coefficients using the formula
    coef.errors[i] <- sqrt(sum(
      (beta[x_cols %in% names(i_coef)] - i_coef[names(i_coef) %in% x_cols]) ^ 2) +
        sum(beta[!(x_cols %in% names(i_coef))]) ^ 2)
}
```

```r
# Plot errors between coefficients with changing number of predictors
plot(coef.errors,
     main="Errors Between Coefficients with Best Subset Selection",
     xlab="number of coefficients",
     ylab="error between estimated and true coefficients",
     pch=19,
     type="b")
```




```r
which.min(coef.errors)
```

```
## [1] 7
```

We can see that the two graphs take different trends. The errors between true and estimated coefficients do not decreases as the number of coefficients increases. At 7 predictors, the error calculated minimizes, but the test MSE minimizes when number of predictors is equal to 15. Thus, a better fit of true coefficients does not neccessarily mean smallest test MSE.

Application exercises 0. Read dataset

```r
# Load the dataset from file
test <- read.csv('data/gss_test.csv')
train <- read.csv('data/gss_train.csv')
```

1. Fit a least squares linear model on the training set.

```r
model.lm <- lm(egalit_scale~ ., data=train)
pred.lm <- predict(model.lm, newdata=test)
mean((test$egalit_scale - pred.lm) ^ 2)
```

```
## [1] 63.21363
```

The test MSE using linear model is 63.21363.

2. Fit a ridge regression model on the training set

```
# Separate dataset into predictors and response variable
xtrain <- model.matrix (egalit_scale~., train)[,-1]
ytrain <- train$egalit_scale
xtest <- model.matrix (egalit_scale~., test)[,-1]
ytest <- test$egalit_scale
```

```
# Perform 10-fold cross validation to find the best lambda
cv.out <- cv.glmnet (xtrain,ytrain, alpha =0, type.measure="mse")
bestlam <- cv.out$lambda.min

# Use Ridge regression with best lambda and calculate test MSE
model.ridge <- glmnet(xtrain,ytrain, alpha=0, lambda=bestlam)
pred.ridge <- predict(model.ridge, s=bestlam, newx=xtest)
MSE.ridge <- mean((pred.ridge - ytest) ^ 2)
print(MSE.ridge)
```

```
## [1] 60.96776
```

The test MSE using ridge regression is 60.96776.

3. Fit a lasso regression on the training set

```
# Perform 10-fold cross validation to find the best lambda
cv.out <- cv.glmnet (xtrain, ytrain, alpha=1, type.measure="mse")
bestlam <- cv.out$lambda.min

# Use Ridge regression with best lambda and calculate test MSE
model.lasso <- glmnet(xtrain,ytrain,alpha=1,lambda=bestlam)
pred.lasso <- predict(model.lasso, s=bestlam, newx=xtest)
MSE.lasso <- mean((pred.lasso - ytest) ^ 2)
print(MSE.lasso)
```

```
## [1] 61.269
```

The test MSE using ridge regression is 61.22212.

```
# Print lasso model's coefficients
lasscoef <- predict(model.lasso,
                    type="coefficients",
                    s=bestlam)[1:length(model.lasso$beta),]
lasscoef[lasscoef!=0]
```

```
##         (Intercept)                 age               black              childs
##         30.58527416         -0.03979472          0.94109267          0.16003046
##             colhomo               grass               happy            income06
##          0.08072144         -1.28294232          0.21835366         -0.11137199
##              owngun            polviews               pres08        science_quiz
##          0.71695643         -1.55518887         -4.31120985         -0.05630268
##                 sex                sibs           tolerance             tvhours
##          0.96543510          0.09764766         -0.19834190          0.17916102
##            vetyears  degree_Junior.Coll degree_Bachelor.deg     marital_Widowed
##         -0.07150823         -0.38227955         -1.66026565         -0.07861622
##          news_NEVER       partyid_3_Ind       partyid_3_Rep      relig_HINDUISM
##          0.10220439         -0.92011865         -2.50177021         -1.49863982
##          spend3_Mod      spend3_Liberal        zodiac_VIRGO      zodiac_SCORPIO
##          0.07551828          1.24670553          0.34170206         -0.17387974
```

```
##     zodiac_AQUARIUS
##          0.16537497
```

```
length(lasscoef[lasscoef!=0]) - 1
```

```
## [1] 28
```

There are 29 non-zero coefficient estimates using the lasso regression model.

4. Fit an elastic net regression model on the training set

```
# Create a list to record each alpha and lambda used in the model
best_combo <- list()
for (i in seq(0, 1, .1)) {
  cv.out=cv.glmnet(xtrain, ytrain, alpha=i, type.measure="mse")
  best_combo[[i * 10 + 1]] <- cv.out$lambda.min
  # Calculate the MSE of best lambda and print out the result
  mse.min <- cv.out$cvm[cv.out$lambda == cv.out$lambda.min]
  cat(i, mse.min, "\n")
}
```

```
## 0 61.11413
## 0.1 60.45036
## 0.2 59.68561
## 0.3 60.11249
## 0.4 60.07209
## 0.5 59.97989
## 0.6 59.60213
## 0.7 59.68769
## 0.8 59.80775
## 0.9 59.94661
## 1 59.70469
```

We found the lowest cross-validation MSE is when alpha equals to 1.

```
# Use Elastic Net regression with best lambda and alpha and calculate test MSE
model.elasnet <- glmnet(xtrain, ytrain, alpha=1, lambda=best_combo[[11]])
pred.elasnet <- predict(model.elasnet, s=bestlam, newx=xtest)
MSE.elasnet <- mean((pred.elasnet - ytest) ^ 2)
print(MSE.elasnet)
```

```
## [1] 61.269
```

The test MSE using ridge regression is 61.22212. Because when alpha=1, the elastic net model perform lasso regression, thus the number of non-zero coefficients is the same as lasso regression, 29.

5. All models generate test MSE that is above 60, which means the predictions of individual's egalitarianism is not very accurate. There is not a big difference among all the regression models. Comparing each model's test MSE, the ridge regression model actually performs the best, with the lowest test MSE value.