Chen Zhaoyang HW2

Zhaoyang Chen

2/3/2020

```
library(tidyverse)
library(glmnet)
```

Conceptual exercises

Training/test error for subset selection

Step one

```
set.seed(1234)
eps = rnorm(1000, 0, 1) # generate epsilon
X = data.frame(matrix(0, 1000, 20)) # generate zero matrix to store random values
i = 1
while (i <= 20){
  mu = runif(1, -10, 10)
  theta = runif(1, 1, 10)
  X[,i] = rnorm(1000, mu, theta)
  i = i + 1
} # generate random x and store
beta = runif(20, -10, 10) # generate beta from U[-1, 1]
beta[abs(beta) <= 2] = 0 # set some values to zero
beta = matrix(beta, ncol = 1)
X = as.matrix(X)
X_beta = X %*% beta
Y = X_beta + eps # calculate y
```

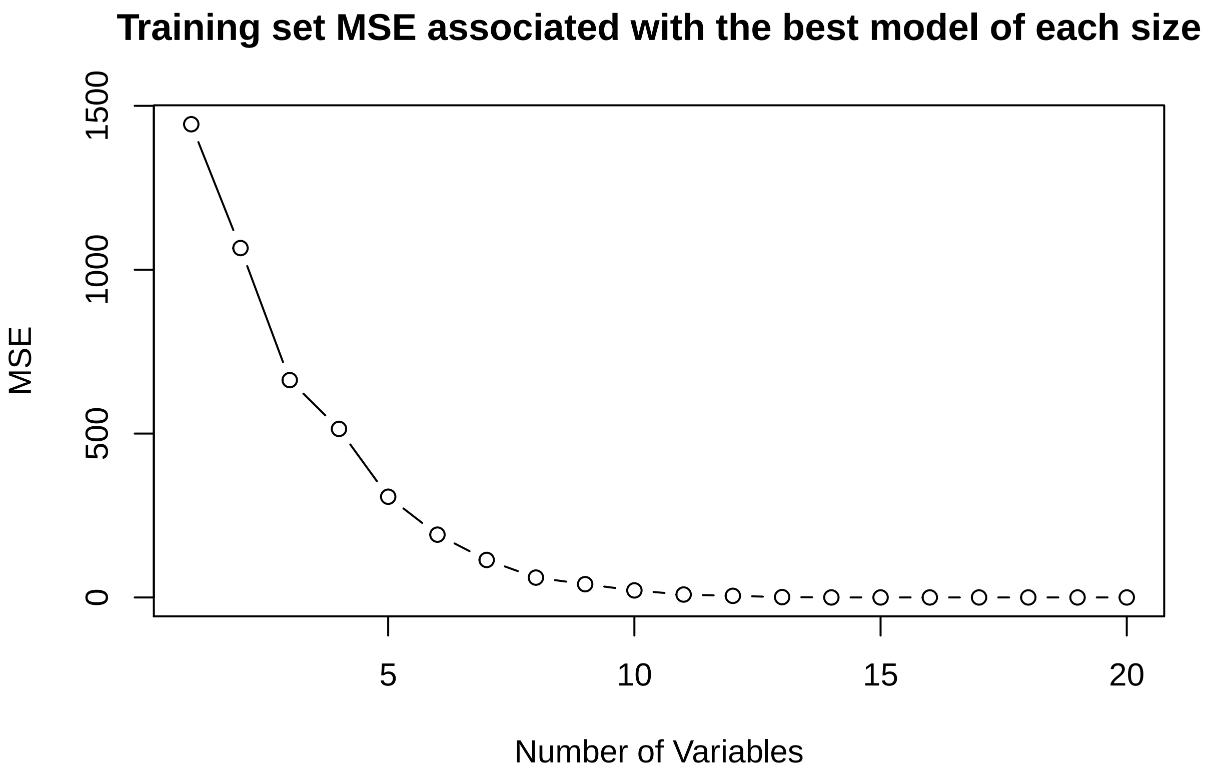
Step two

```
data = data.frame(X, Y) # combine X and Y
data$id = 1:1000
train = data %>% sample_frac(.1) # training set
test = anti_join(data, train, by = 'id') # testing set
```

Step three

```
#install.packages('leaps')
library(leaps)
model = regsubsets(Y ~ ., train[,-22], nvmax = 20)
model_sum = summary(model)
results_best <- tibble(
  `adj_r2` = which.max(model_sum$adjr2), # Adjusted r-squared
  BIC = which.min(model_sum$bic), # Schwartz's information criterion
  `c_p` = which.min(model_sum$cp) # Mallows' Cp
) %>%
  gather(statistic, best)

mse = model_sum$rss / 1000
plot(mse, xlab = "Number of Variables", ylab = "MSE", type = "b", main = 'Training set MSE associated wi
```



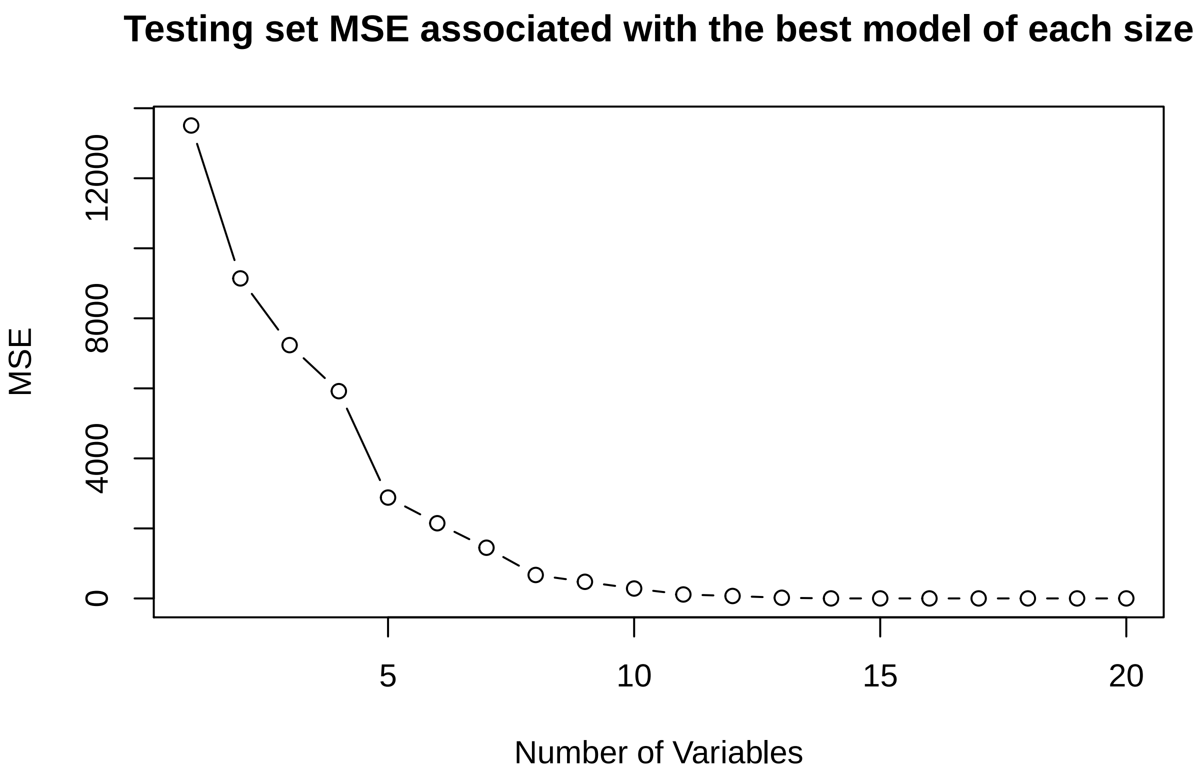
```
mse
```

```
##  [1] 1.443855e+03 1.066047e+03 6.631459e+02 5.142475e+02 3.073048e+02
##  [6] 1.916146e+02 1.144689e+02 6.064888e+01 4.050051e+01 2.157106e+01
## [11] 8.882488e+00 4.901343e+00 1.113475e+00 7.615075e-02 7.253144e-02
## [16] 7.227960e-02 7.208854e-02 7.185557e-02 7.180644e-02 7.180629e-02
```

When the number of variables is 20, we have the smallest MSE.

Step four

```
mse = c()
i = 1
while (i <= 20){
  tb = as.data.frame(coef(model, i))
  name = row.names(tb)
  temp = test[,name[-1]]
  intercept = rep(1, 900)
  temp = cbind(intercept, temp)
  pred = as.matrix(temp) %*% as.matrix(tb[,1])
  actual = test['Y']
  mse = c(mse, sum((as.data.frame(pred) - actual)^2) / 900)
  i = i+1
}
plot(mse, xlab = "Number of Variables", ylab = "MSE", type = "b", main = 'Testing set MSE associated wi
```



```
mse
```

```
##  [1] 13505.936586  9139.687491  7234.795815  5919.617417  2881.470205
##  [6]  2147.973000  1448.539780   670.636064   475.538661   283.980773
## [11]   114.073894    70.709761    19.865692     1.348150     1.364781
## [16]     1.388273     1.383200     1.388587     1.392319     1.392560
```

Step five

From the plot, we can hardly tell which point is the lowest. However, from the table of mse I find that test mse come to the minimum when p = 14. This is because the model overfit once p is too large and its generalization ability becomes terrible. As a result, the test mse will first decrease then increase.

Step six

```
beta_pre = as.data.frame(coef(model, 14))
beta; beta_pre
```

```
##            [,1]
##  [1,]  0.000000
##  [2,]  5.475396
##  [3,]  2.354963
##  [4,] -3.029862
##  [5,]  2.848496
##  [6,]  0.000000
##  [7,]  0.000000
##  [8,]  5.896982
##  [9,] -9.676453
## [10,]  4.369937
## [11,]  0.000000
## [12,]  2.003314
## [13,]  0.000000
## [14,] -5.454568
## [15,]  2.666605
## [16,]  7.837760
## [17,]  9.256451
## [18,]  8.474316
## [19,]  0.000000
## [20,] -9.018156
```

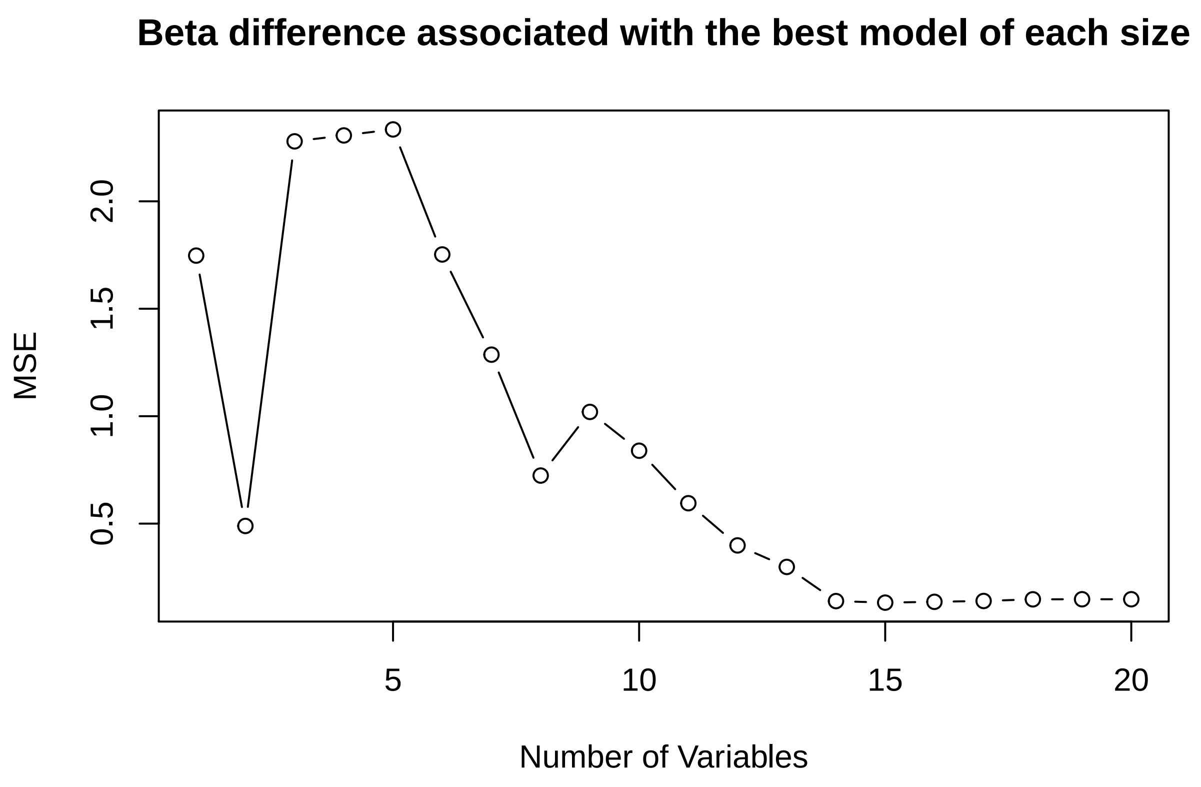
```
##             coef(model, 14)
## (Intercept)      -0.7556694
## X2                5.4707913
## X3                2.3840446
## X4               -3.0558830
## X5                2.8534154
## X8                5.8962440
## X9               -9.6675532
## X10               4.4537709
## X12               1.9244380
## X14              -5.4218397
## X15               2.6440655
## X16               7.8371359
## X17               9.2208654
## X18               8.5161570
## X20              -9.0190482
```

For the true beta, we have 6 parameters whose value equals to 0 and the predicted beta also shows that those betas are omitted. Thus, I can conclude that the best model is accurate and close to the real model.

Step seven

```
beta = as.matrix(beta)
rownames(beta) = colnames(data)[1:20]

smse = c()
i = 1
while (i <= 20){
  tb = as.data.frame(coef(model, i))
  name = row.names(tb)[-1]
  beta_hat = tb[,1][-1]
  beta_act = beta[name,]
  smse = c(smse, sqrt(sum((beta_hat - beta_act)^2)))
  i = i+1
}
plot(smse, xlab = "Number of Variables", ylab = "MSE", type = "b", main = 'Beta difference associated wi
```



The value is smallest when p = 14, which is exactly the best model. In other words, the predicted beta is most close to true beta when p = 14.

Application exercises

Question one

```r
train = read_csv('data/gss_train.csv')
test = read_csv('data/gss_test.csv')

mse = function(actual, prediction){
  return(sum((actual - prediction)^2)/length(actual))
}

model_ols = lm(egalit_scale ~ ., data = train)
mse(test$egalit_scale, predict(model_ols, newdata = test[,-14]))
```

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

Question two

```r
x = as.matrix(train[,-14])
y = as.matrix(train[,14])
model_ridge = cv.glmnet(x, y, alpha = 0, nfolds = 10)
mse(test$egalit_scale, predict(model_ridge, newx = as.matrix(test[,-14])))
```

```
## [1] 61.66853
```

Question three

```r
model_lasso = cv.glmnet(x, y, alpha = 1, nfolds = 10)
mse(test$egalit_scale, predict(model_lasso, newx = as.matrix(test[,-14])))
```

```
## [1] 62.36956
```

Question four

```r
alpha_seq = seq(0, 1, 0.1)
lambda_seq = seq(0, 12, length.out = 100)
error = 0
parameter = c(0, 0)

for (alpha in alpha_seq) {
  model = cv.glmnet(x, y, alpha = alpha, lambda = lambda_seq)
  if (error == 0){
    error = min(model$cvm)
    parameter[1] = alpha
    parameter[2] = model$lambda.min
  } else{
    if (min(model$cvm) < error){
      error = min(model$cvm)
      parameter[1] = alpha
      parameter[2] = model$lambda.min
    }
  }
}

model_elastic = glmnet(x, y, alpha = parameter[1], lambda = parameter[2])
mse(test$egalit_scale, predict(model_elastic, newx = as.matrix(test[,-14])))
```

```
## [1] 61.07957
```

```r
model_elastic$beta@Dim[1] - length(model_elastic$beta@x)
```

```
## [1] 47
```

Based on the four models I build above, the best model is the elastic model (alpha = 0.9; lambda = 0.242). The mean squared error is 61.07957, which is not very accurate. However, the performance of the other four models are very close to the elastic model so I tend to say that I have not improved the prediction accuracy through the adjustment of models.