HW3 Responses

Yawei LI

2/4/2020

Subset Selection

Generate Data

```
set.seed(1234)
beta <- runif(20,-1,1)
beta[c(3,8,17)] <- 0
predictors <- matrix(data = runif(200000,0,1),nrow = 1000, ncol = 20)
y <- predictors %*% beta + rnorm(1000,0,1)
data <- data.frame(predictors,y)
```

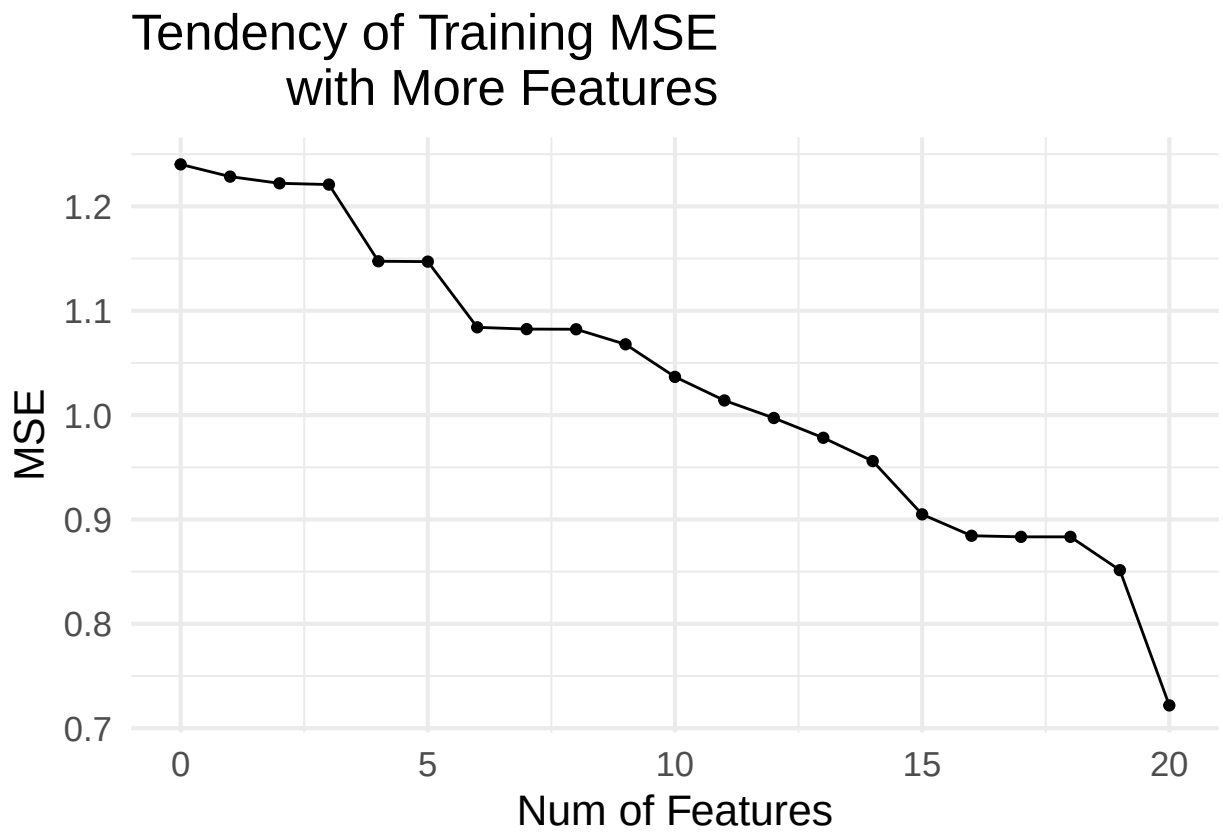
Split Data

```
set.seed(2234)
split <- initial_split(data,0.1)
train <- training(split)
test <- testing(split)
```

Subset Selection

As can be seen, training MSE drops consistently with the increase in the number of features included in our model.

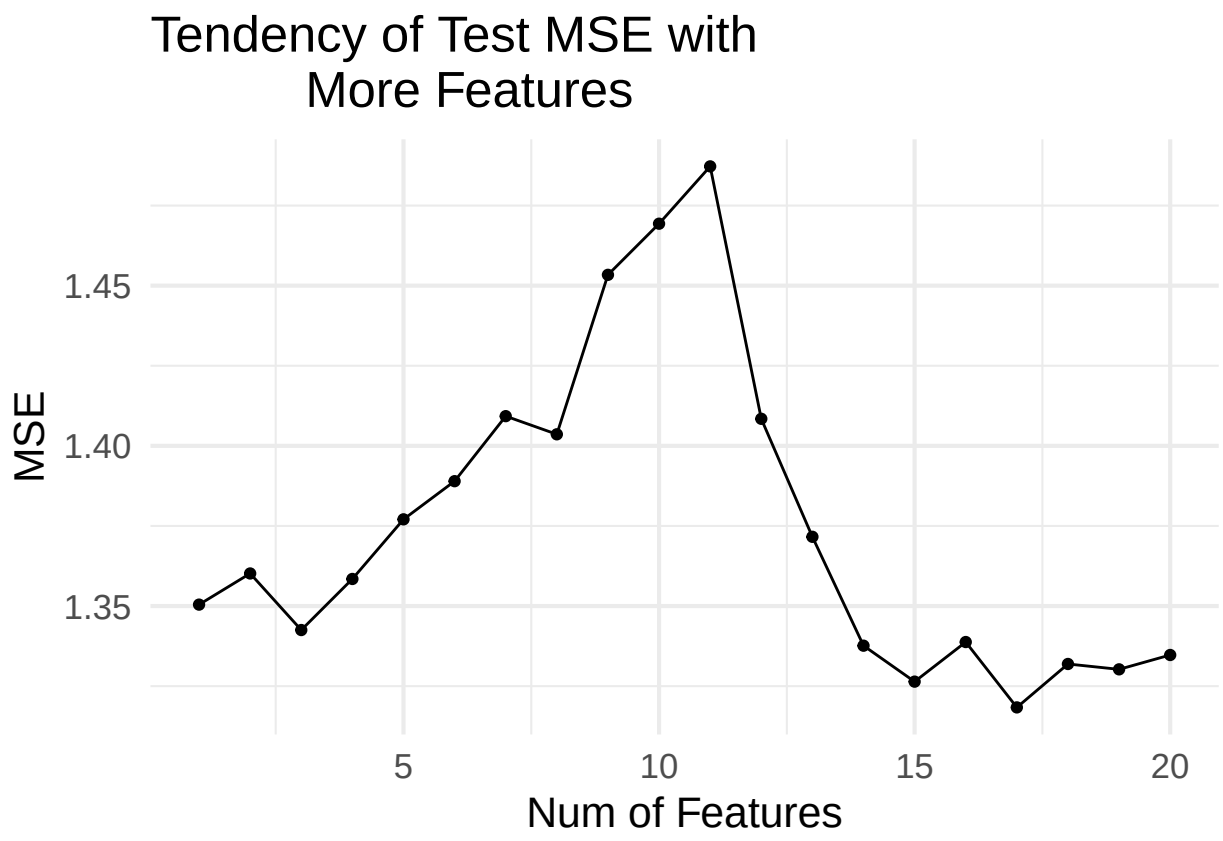
```
best_subsets <- regsubsets(y ~ .,
                           data = train,
                           nvmax = 20)
mse <- best_subsets$rss/100
p1 <- ggplot() +
      geom_point(aes(x=0:20,y=mse)) +
      geom_line(aes(x=0:20,y=mse)) +
      labs(x = 'Num of Features', y = 'MSE', title = 'Tendency of Training MSE
           with More Features')
p1
```



Test Set MSEs

The least test MSE is produced by the model with 17 features, yet the improvement compared with the 10-feature model is relatively small compared to added complexity.

```
sum = summary(best_subsets)
mse = rep(NA,20)
test2 = data.frame(rep(1,900),test)
colnames(test2)[1] <- '(Intercept)'
for (i in 1:20) {
    coefi = coef(best_subsets, id = i)
    pred = as.matrix(test2[, names(coefi)]) %*% coefi
    mse[i] = mean((test$y - pred)^2)
}
p2 <- ggplot() +
      geom_point(aes(x=1:20,y=mse)) +
      geom_line(aes(x=1:20,y=mse)) +
      labs(x = 'Num of Features', y = 'MSE', title = 'Tendency of Test MSE with
           More Features')
p2
```



Comparing Model to Generation Process

As can be seen from the table below, our best model did a mediocre job. Two out of three zero coefficients are correctly placed, but value of some coefficients deviates a lot from true value.

```
df <- data.frame(coef(best_subsets,id = 17))
df <- InsertRow(data = df,NewRow = 0,RowNum = 4)
df <- InsertRow(data = df,NewRow = 0,RowNum = 5)
df <- InsertRow(data = df,NewRow = 0,RowNum = 9)
df <- data.frame(df,c(0,beta))
rownames(df)[2:21]<- colnames(test)[1:20]
colnames(df)<- c('Model','Generating Mechanism')
df
```

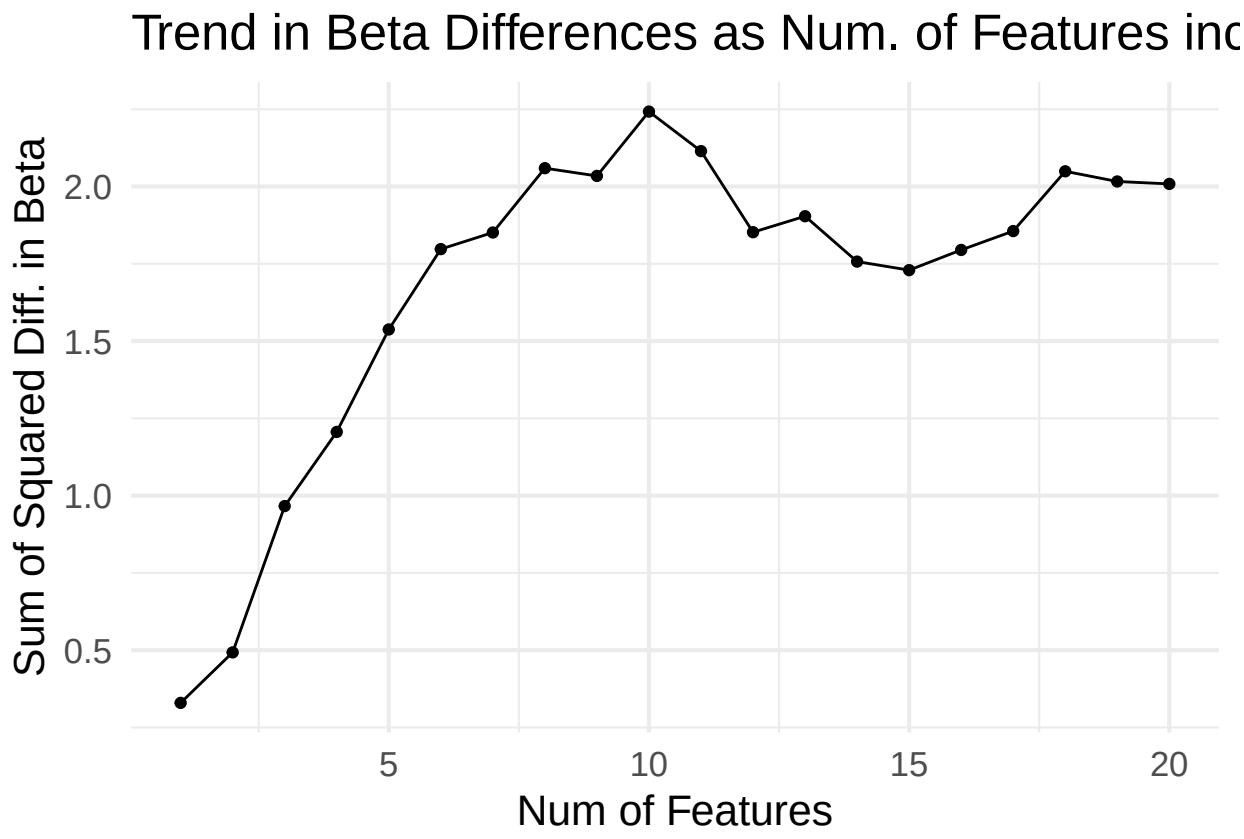
```
##                  Model Generating Mechanism
## (Intercept)  0.8767718           0.00000000
## X1          -1.4077904          -0.77259318
## X2          -0.5625972           0.24459881
## X3           0.0000000           0.00000000
## X4           0.0000000           0.24675888
## X5          -0.7435244           0.72183077
## X6           0.6662480           0.28062121
## X7           0.6918066          -0.98100849
## X8           0.0000000           0.00000000
## X9           0.2729665           0.33216752
## X10         -0.7882646           0.02850228
```

```
## X11          0.6230339          0.38718258
## X12          0.5904758          0.08994967
## X13          0.4149137         -0.43453283
## X14         -0.1981172          0.84686697
## X15         -0.6514985         -0.41536832
## X16          0.6931301          0.67459126
## X17         -0.7222973          0.00000000
## X18         -0.5361876         -0.46635844
## X19         -0.1797529         -0.62655442
## X20         -0.4134409         -0.53554818
```

Plot on Summed Square of Coefficient Differences

Plot created below.

```
df_vector = rep(0,20)
for (i in 1:20) {
    coefi <- coef(best_subsets, id = i)
    true_b <- df[names(coefi),2]
    sum_dif <- 0
    for (j in 1:i){
      sum_dif = sum_dif + (coefi[j] - true_b[j])^2
    }
    df_vector[i] <- sqrt(sum_dif)
}
p3 <- ggplot() +
      geom_point(aes(x = 1:20,y = df_vector))+
      geom_line(aes(x = 1:20,y = df_vector))+
      labs(x = 'Num of Features', y = 'Sum of Squared Diff. in Beta',
           title = 'Trend in Beta Differences as Num. of Features increase')
p3
```



Application in GSS

Data and preparation

```
gss_train <- read.csv('gss_train.csv')
gss_test <- read.csv('gss_test.csv')

mean_sum_squares <- function(x1,x2)
{
  sum = numeric()
  for (i in 1:length((x1))) {
    sum[i] = (x1[i]-x2[i])^2
  }
  return(mean(sum))
}
```

Least Squares Linear

Test MSE for Least Squares Linear is 63.2

```
ls <- lm(egalit_scale ~ .,data = gss_train)
beta_ls <- coef(ls)
p_ls <- predict(ls,newdata = gss_test)
```

```
test_mse <- mean_sum_squares(p_ls,gss_test$egalit_scale)
paste('Test MSE is',test_mse)
```

```
## [1] "Test MSE is 63.2136296230151"
```

Ridge

Test MSE for Ridge is 61.9.

```
gss_train_x <- model.matrix(egalit_scale~ ., gss_train)[, -1]
gss_train_y <- gss_train$egalit_scale

gss_test_x <- model.matrix(egalit_scale ~ ., gss_test)[, -1]
gss_test_y <- gss_test$egalit_scale

gss_ridge <- cv.glmnet(y = gss_train_y,x = gss_train_x, alpha = 0) #nfold = 10 by default
p_ridge <- predict(gss_ridge,newdata = gss_test,newx = gss_test_x)
test_mse <- mean_sum_squares(p_ridge,gss_test$egalit_scale)
paste('Test MSE is',test_mse)
```

```
## [1] "Test MSE is 61.9285675411698"
```

LASSO

Test MSE for LASSO is 61.9. Coefficients reported in printed table.

```
gss_lasso <- cv.glmnet(y = gss_train_y,x = gss_train_x, alpha = 1)
p_lasso <- predict(gss_lasso,newdata = gss_test,newx = gss_test_x)
test_mse <- mean_sum_squares(p_lasso,gss_test$egalit_scale)
paste('Test MSE is',test_mse)
```

```
## [1] "Test MSE is 61.8918977175881"
```

```
df_lasso = data.frame(coef(gss_lasso)[which(coef(gss_lasso)>0)])
rownames(df_lasso) <- colnames(gss_test)[which(coef(gss_lasso)>0)]
colnames(df_lasso) <- c('Coef')
print(df_lasso)
```

```
##                       Coef
## age             28.36339796
## born             1.10706476
## owngun           0.56832607
## sex              0.66618374
## sibs             0.04592067
## tvhours          0.10859795
## spend3_Liberal   0.97543443
```

Elastic Net

Test MSE is 62.1 with alpha = 1 and lambda = 0.23. Other results are printed below.

```
set.seed(1234)

fold_id <- sample(1:10, size = length(gss_train_y), replace = TRUE)
```

```
tuning_grid <- tibble::tibble(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
  lambda_min = NA
)

for(i in seq_along(tuning_grid$alpha)) {
  fit <- cv.glmnet(gss_train_x,
                   gss_train_y,
                   alpha = tuning_grid$alpha[i],
                   foldid = fold_id)

    tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
    tuning_grid$lambda_min[i] <- fit$lambda.min
}
  min_mse = min(tuning_grid$mse_min)
  paste('The lowest MSE is',min_mse,'with alpha =',
        tuning_grid$alpha[which(tuning_grid$mse_min == min_mse)],
        'and lambda =',
        tuning_grid$lambda_min[which(tuning_grid$mse_min == min_mse)])
```

```
## [1] "The lowest MSE is 59.5626796259866 with alpha = 1 and lambda = 0.212784533609135"
```

```
  # The last fit object turns out to be the best model,
  # so we use it directly below.

  df_net = data.frame(coef(fit)[which(coef(fit)>0)])
  rownames(df_net) <- colnames(gss_test)[which(coef(fit)>0)]
  colnames(df_net) <- c('Coef')
  print(df_net)
```

```
##                       Coef
## age             28.04802615
## born             1.07687848
## owngun           0.53706499
## sex              0.57934250
## sibs             0.03383409
## tvhours          0.09458254
## spend3_Liberal   0.91985899
```

```
  p_net = predict(fit,newdata = gss_test, newx = gss_test_x)
  test_mse <- mean_sum_squares(p_net,gss_test$egalit_scale)
  paste('Test MSE is',test_mse)
```

```
## [1] "Test MSE is 62.0982758128179"
```

Comments

It is confusing to me why Elastic Net did not yield better results than LASSO and ridge. This may suggest something went wrong in my analysis yet I failed to find. If I am correct in the process, the value of our MSEs are quite disappointing, meaning that we could not predict an individual's egalitarianism level reliably and precisely, given all the predictors we collect. Besides, none of the approach yields a major improvement and they do not differ a lot in test error.