Takuyo_Ozaki_HW3

Takuyo Ozaki

2/7/2020

```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
knitr::opts_chunk$set(fig.width = 7, fig.height = 4, fig.align = 'center')

# load necessary package
library(tidyverse)
library(broom)
library(rsample)
library(caret)
library(glmnet)
library(tibble)
library(rcfss)
library(knitr)
library(leaps)

# Set the theme as minimal
theme_set(theme_minimal())
```

Conceptual exercises

Question 1

```
set.seed(0) #Ensure the reproducibility

# Given p = 20 features, n = 1000 observations
p <- 20
n <- 1000
x <- matrix(rnorm(n * p), n, p)
beta <- rnorm(p)
epsilon <- rnorm(p)

# Beta has some elements that are exactly equal to zero
beta[4] <- 0
beta[6] <- 0
beta[9] <- 0
beta[15] <- 0

# Create a data set from the model
y <- x %*% beta + epsilon
```

Question 2

```
# Split x and y into training 100, test 900 observations
train <- sample(seq(1000), 100, replace = FALSE)
y_train <- y[train, ]
y_test <- y[-train, ]
x_train <- x[train, ]
x_test <- x[-train, ]
```

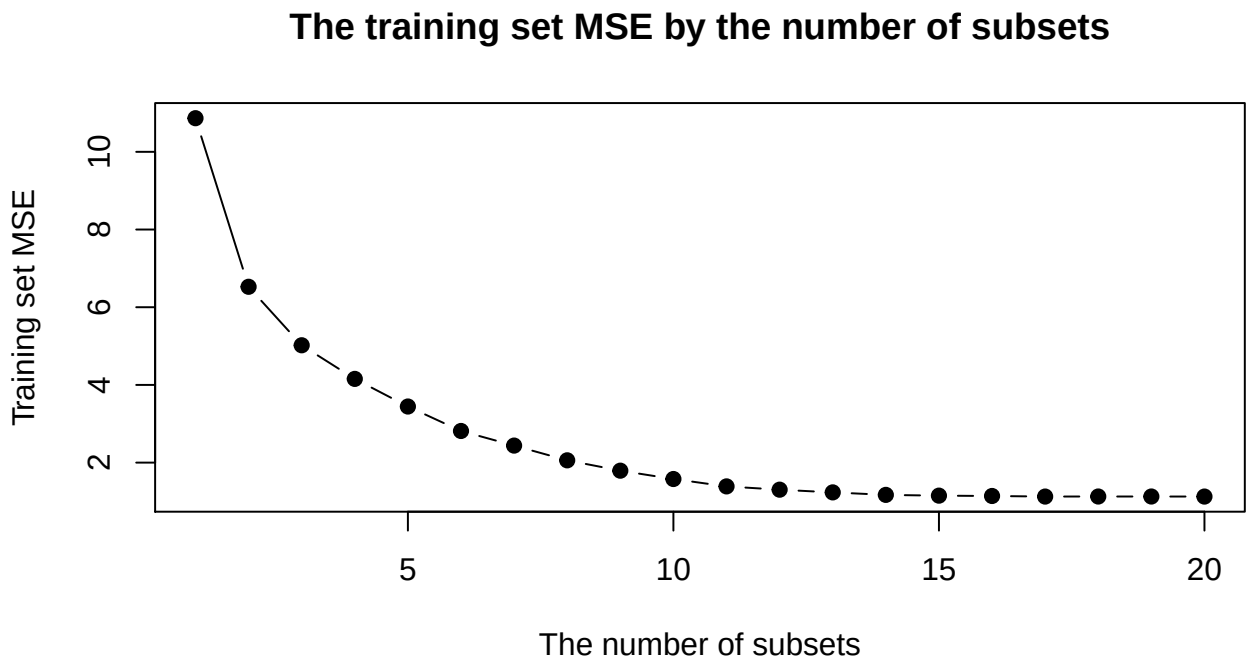
Question 3

```
# Perform the best subset selection on the training set
model_q1 <-
  regsubsets(y ~ ., data = data.frame(x = x_train, y = y_train), nvmax = p)

# Prepare a vector and x column names for loop
train_mse <-  rep(NA, p)
x_cols <- colnames(x, do.NULL = FALSE, prefix = "x.")

# Calculate the train MSE by each number of the best subset
for (i in 1:p) {
    coefficient <- coef(model_q1, id = i)
    prediction <- as.matrix(x_train[, x_cols %in% names(coefficient)]) %*%
                            coefficient[names(coefficient) %in% x_cols]
    train_mse[i] <- mean((y_train - prediction)^2)
}

# Plot the training set MSE associated with the best model of each size
plot(train_mse, main = "The training set MSE by the number of subsets",
     xlab = "The number of subsets",
     ylab = "Training set MSE", pch = 19, type = "b")
```




```
# Calculate the number of subsets when the training set MSE is minimum
which.min(train_mse)
```

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

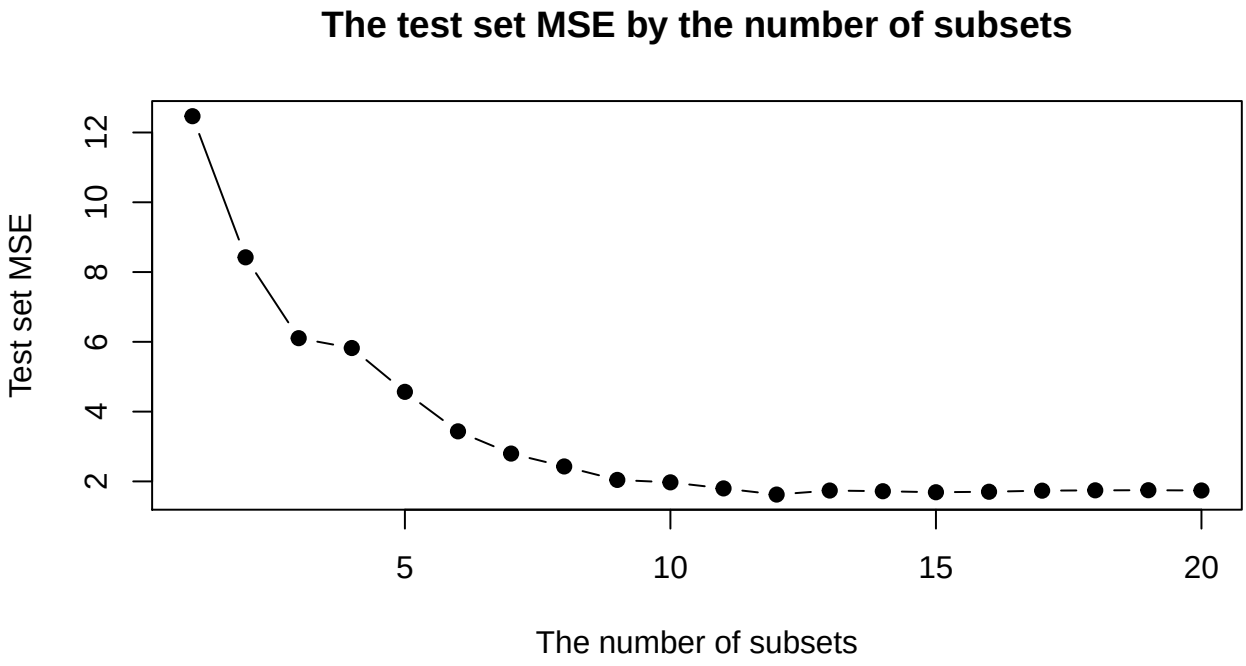
For the model with 20 subsets, the training set MSE take on its minimum value. We can expect this result because when the model has more subsets, it has a better fit in general, which could leads to less training MSE (by overfitting).

Question 4

```
# Prepare a vector for loop
test_mse <- rep(NA, p)

# Calculate the test MSE by each number of the best subset
for (i in 1:p) {
    coefficient <- coef(model_q1, id = i)
    prediction <- as.matrix(x_test[, x_cols %in% names(coefficient)]) %*%
                            coefficient[names(coefficient) %in% x_cols]
    test_mse[i] <- mean((y_test - prediction)^2)
}

# Plot the test set MSE associated with the best model of each size
plot(test_mse, main = "The test set MSE by the number of subsets",
     xlab = "The number of subsets",
     ylab = "Test set MSE", pch = 19, type = "b")
```



Question 5

```
# Calculate the number of subsets when the test set MSE is minimum
which.min(test_mse)
```

```
## [1] 12
```

For the model with 12 subsets, the test set MSE take on its minimum value. We can expect this result because test MSE has U-shape curve in general because when the model has more subsets, it has more variance as well as a better fit. In this case, the models with more than 13 subsets might have the possibility of overfitting.

Question 6

```
# Show the subsets when the test MSE is minimum
coef(model_q1, id = 12)
```

```
## (Intercept)         x.1         x.5         x.7         x.8        x.10
##   0.4837526  -0.5883935   0.9545127   1.0860392  -0.9030662  -0.5958637
##         x.11        x.12        x.13        x.14        x.17        x.18
##   0.3453905   1.9851346  -2.6073612  -0.8837738  -1.3666687   0.6482974
##         x.20
##  -0.4625209
```

```
# Generate the data frame of the minimum test MSE coefficients
minMSE_model <- data.frame(coef(model_q1, id = 12))
colnames(minMSE_model) <- "minMSE_model"
```

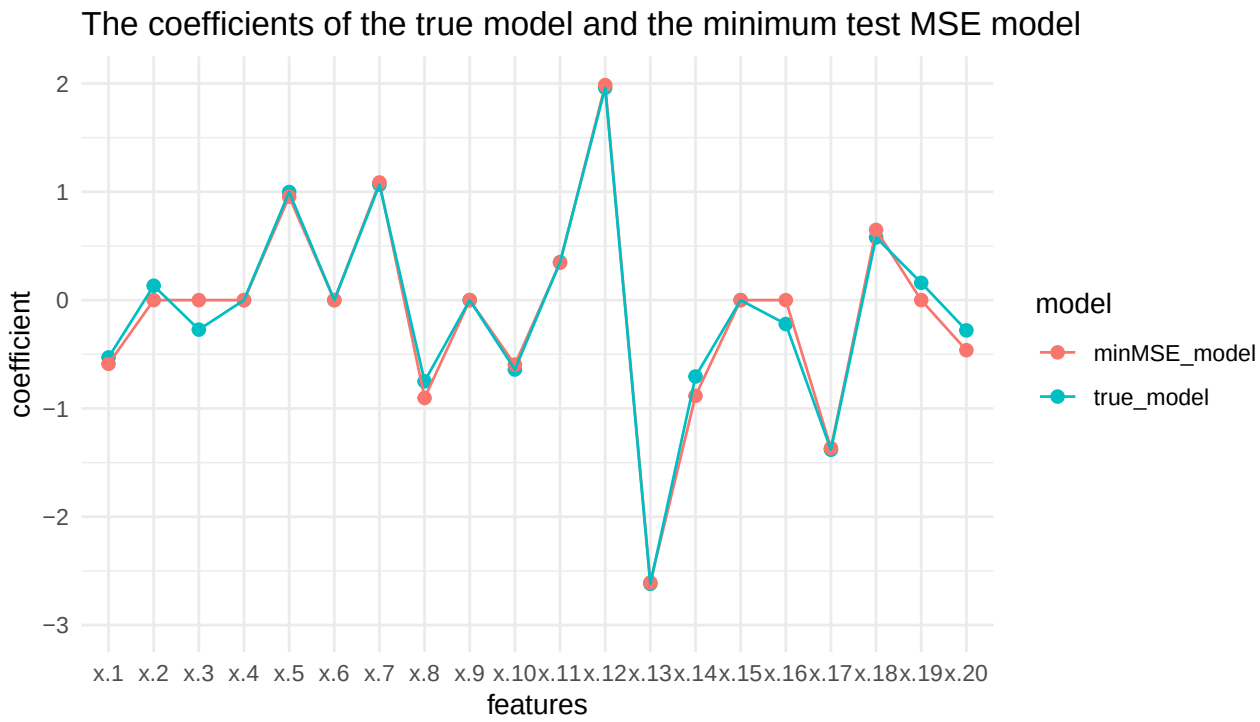
```r
minMSE_model$features <- rownames(minMSE_model)

# Generate the data frame of the true coefficients
true_model <- data.frame(features = x_cols, true_model = beta)

# Compare the true model and the minimum test MSE model
df_q6 <- left_join(true_model, minMSE_model) %>%
  gather(model, coefficient, -features) %>%
  mutate(coefficient = ifelse(is.na(coefficient), 0, coefficient))

# Change the order of features to show the x-axis properly
df_q6 <- transform(df_q6, features = factor(features, levels = x_cols))

# Plot the coefficients of the true model and the minimum test MSE model
df_q6 %>%
  ggplot(aes(x = features, y = coefficient, color = model, group = model)) +
  geom_point(size = 2) +
  geom_line() +
  ylim(-3, 2) +
  labs(title = "The coefficients of the true model and the minimum test MSE model")
```



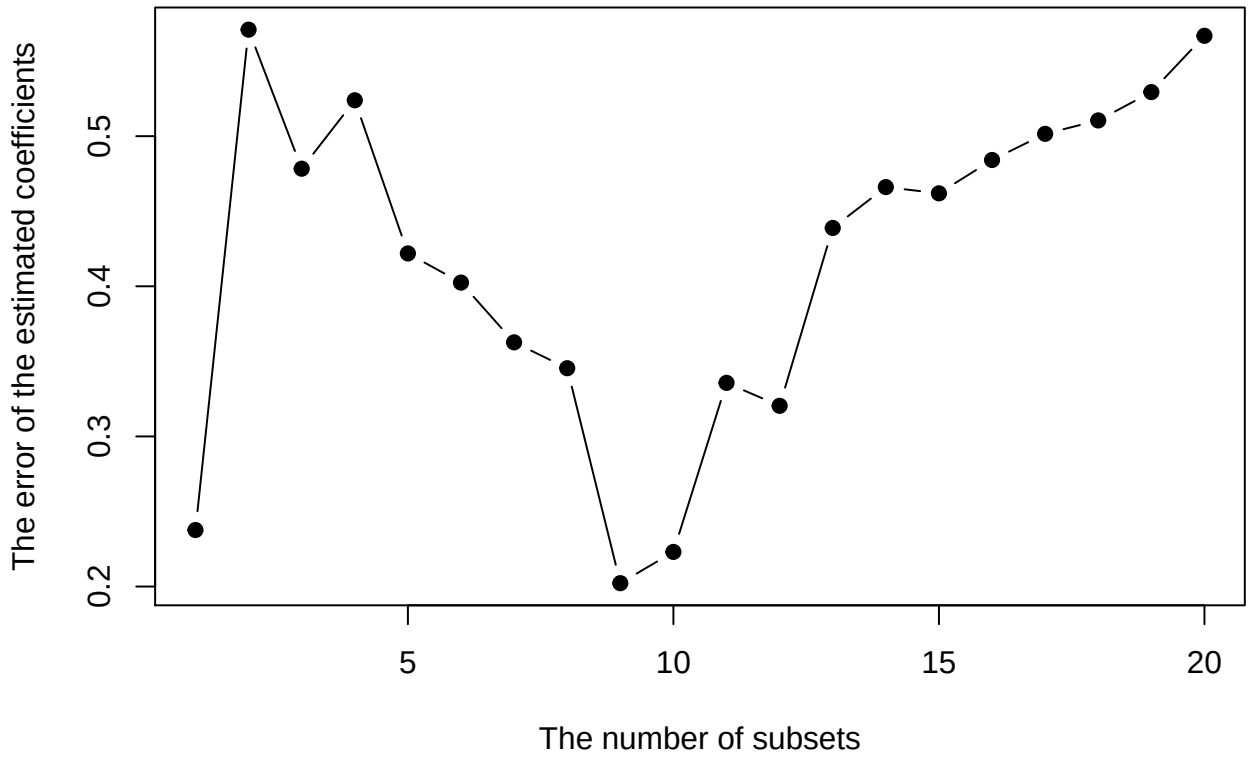
- In the minimum test MSE model, the 8 subsets are excluded (the coefficients are zero): x.2, x.3, x.4, x.6, x.9, x.15, x.16, x.19.
- In the true model, the coefficients (beta) of the 4 subsets are zero: x.4, x.6, x.9, x.15, which are all excluded from the minimum test MSE model.
- As for the coefficient sizes, the graph above shows that the two models are almost the same. The coefficient of the four subsets: x.2, x.3, x.16, x.19 are not zero, but close to zero in the true model. Thus, we can guess that these subsets are excluded from the minimum test MSE model.

Question 7

```
set.seed(0)

# Calculate the error between estimated and true coefficients
subset <- rep(NA, p)
value <- rep(NA, p)
for (i in 1:p) {
    coefficient <- coef(model_q1, id = i)
    subset[i] <- i
    value[i] <- sqrt(sum((beta[x_cols %in% names(coefficient)] -
                          coefficient[names(coefficient) %in% x_cols])^2))
}

# Plot the error between estimated and true coefficients
plot(x = subset, y = value, xlab = "The number of subsets",
     ylab = "The error of the estimated coefficients",
     pch = 19, type = "b")
```




```
# The number of subset with minimum estimate error
which.min(value)
```

```
## [1] 9
```

- The value given in the question shows the the square root of the sum of the squared difference between true and estimated coefficient, which means the error of the estimated coefficients compared to the true coefficient (continue to the next page).

- The model with 9 subsets minimizes the error of the estimated coefficients while the test error is minimized in the model with 12 subset. Thus, even if the estimated coefficients has a better fit of the true coefficients, it doesn't always mean a lower test MSE in the model.

Application exercises

```
# Import the data
gss_test <- read_csv("data/gss_test.csv")
gss_train <- read_csv("data/gss_train.csv")

# Create training and testing feature model matrices and response vectors
# Use model.matrix(...)[, -1] to discard the intercept
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
```

Question 1

```
# Fit a least squares linear model on the training set
linear_model <- lm(egalit_scale ~ ., data = gss_train)

# Report the test MSE
yhat_linear <- predict(linear_model, newdata = gss_test)
(linear_MSE <- mean((gss_test_y - yhat_linear)^2))
```

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

The test MSE of the least squares linear model is 63.21363.

Question 2

```
set.seed(1)

# Find lambda: apply 10-fold CV Ridge regression to gss data
gss_ridge <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0
)

# Calcurate the MSE of the ridge model
coef_ridge <- coefficients(gss_ridge, s = "lambda.min")
yhat_ridge <- gss_test_x %*% coef_ridge[-1,] + coef_ridge[1, ]
(ridge_MSE <- mean((gss_test_y - yhat_ridge)^2))
```

```
## [1] 61.03781
```

The test MSE of the ridge model is 61.03781.

Question 3

```
set.seed(1)

# Find lambda: apply 10-fold CV Lasso regression to gss data
gss_lasso <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 1
)

# Calcurate the test MSE of the lasso model
coef_lasso <- coefficients(gss_lasso, s = "lambda.min")
yhat_lasso <- gss_test_x %*% coef_lasso[-1,] + coef_lasso[1,]
(lasso_MSE <- mean((gss_test_y - yhat_lasso)^2))
```

```
## [1] 61.19366
```

```
# Calcurate the number of non-zero coefficient estimates
# "length(coef_lasso@x)" include intercept, so need minus one
(lasso_nonzero <- length(coef_lasso@x) - 1)
```

```
## [1] 32
```

The test MSE of the lasso model is 61.19366 and the number of non-zero coefficient estimates is 32.

Question 4

```
set.seed(0)

# Search across a range of alphas
tuning_grid <- tibble(
  alpha      = seq(0, 1, by = .1),
  mse_min    = NA,
  lambda_min = NA
)

# Fit CV model for each alpha and lambda
for (i in seq_along(tuning_grid$alpha)) {
  fit <- cv.glmnet(gss_train_x,
                   gss_train_y,
                   alpha = tuning_grid$alpha[i],
                   lambda = seq(0, 1, by = .1))

  # extract MSE and lambda values
  tuning_grid$mse_min[i]    <- fit$cvm[fit$lambda == fit$lambda.min]
  tuning_grid$lambda_min[i] <- fit$lambda.min
}

# Show the combination of alpha and lambda
tuning_grid %>% filter(mse_min == min(mse_min))
```

```
## # A tibble: 1 x 3
##   alpha mse_min lambda_min
##   <dbl>   <dbl>      <dbl>
## 1   0.5    59.5        0.4
```

```
# Calculate the test MSE of the elastic net model alpha = 0.5, lambda = 0.4
gss_elastic <- cv.glmnet(
  x = gss_train_x,
  y = gss_train_y,
  alpha = 0.5,
  lambda = seq(0, 1, by = .1)
)
coef_elastic <- coefficients(gss_elastic, s = "lambda.min")
yhat_elastic <- gss_test_x %*% coef_elastic[-1,] + coef_elastic[1,]
(elastic_MSE <- mean((gss_test_y - yhat_elastic)^2))
```

```
## [1] 61.12175
```

```
# Calcurate the number of non-zero coefficient estimates
# "length(coef_elastic@x)" include intercept, so need minus one
(elastic_nonzero <- length(coef_elastic@x) - 1)
```

```
## [1] 29
```

The combination of *alpha* and λ with the lowest cross-validation MSE is $alpha = 0.5, \lambda = 0.4$. The test MSE of the elastic net regression model is 61.12175 and the number of non-zero coefficient estimates is 29.

Question 5

```
# Summarize the results obtained.
model <- c("Least squares linear model",
           "Ridge regression model",
           "Lasso regression model",
           "Elastic net regression model")
test_MSE <- c(linear_MSE, ridge_MSE, lasso_MSE, elastic_MSE)
nonzero_coefficients <- c("all", "all", lasso_nonzero, elastic_nonzero)
kable(data.frame(model, test_MSE, nonzero_coefficients), align = "c")
```

model	test_MSE	nonzero_coefficients
Least squares linear model	63.21363	all
Ridge regression model	61.03781	all
Lasso regression model	61.19366	32
Elastic net regression model	61.12175	29

- As for the test MSE, Least squares linear model is a little bit worse than the other three models, but the other three models has almost the same MSE.
- As for the non-zero coefficients estimates, elastic net regression model is little less than lasso regression model.

- As for accuracy, since $MSE = \frac{1}{n}\sum(y_i - \hat{y_i})^2$ and the MSE of the four models are about 61-63, the difference between the prediction and truth of individual's egalitarianism $(y_i - \hat{y_i})$ is about 8 $(\sqrt{63})$ on average. Since egalitarianism scale is from 1 to 35, we can not say these models are "accurate" prediction.
- There are not so much differences among the test errors resulting from these approaches, althought there is a little difference from Least squares linear model and the other three models in terms of the test errors.