MLaE: Assignment #1

Boyu, Chen R11323006

2023-03-19

Q1

The square of Euclidean norm represented by matrix form is

$$
\begin{aligned}
\|\beta\|_2^2 &= \sum_j |\beta_j|^2 = \sum_i \beta_j^2 = \beta^T\beta \\
\lambda\|\beta\|_2^2 &= \lambda\sum_i |\beta_j|^2 = \lambda\sum_i \beta_j^2 = \beta^T\lambda I\beta
\end{aligned}
$$

Let Q denote the objective function, $Q = (Y - X\beta)^T(Y - X\beta) + \lambda\|\beta\|_2^2$

$$
\begin{aligned}
\hat{\beta} &= \arg\min_{\beta} Q \\
&= \arg\min_{\beta} (Y - X\beta)^T(Y - X\beta) + \beta^T\lambda I\beta \\
&= \arg\min_{\beta} Y^TY - 2Y^TX\beta + \beta^TX^TX\beta + \beta^T\lambda I\beta
\end{aligned}
$$

By the First-order condition w.r.t.β, we have

$$
F.O.C.
$$

$$
\begin{aligned}
&\frac{\partial Q}{\partial \beta} = -2X^TY + 2X^TX\hat{\beta} + 2\lambda I\hat{\beta} = 0 \\
&\Rightarrow -X^TY + (X^TX + I)\hat{\beta} = 0 \\
&\Rightarrow \hat{\beta}_{Ridge} = (X^TX + \lambda I)^{-1}X^TY
\end{aligned}
$$

Q2

Set Parameters

```r
beta_vector <- rep(1, 21)
beta <- setNames(beta_vector, paste0("b", 0:20))
n <- 500; p <- 50; tra.idx <- 1:400; R <- 2000
```

The program for a. and b.

```r
# main1: The variable naming rule is "beta_method_dgp_model"
set.seed(1234)
container <- matrix(nrow=R, ncol=24)
params <- expand.grid(b_hat=c(1, 21),
                      met=c("OLS", "Ridge", "LASSO"), dt=1:2, md=1:2)
colnames(container) <- apply(params, 1, paste0, collapse="_")
container[] <- mapply(function(b_hat, met, dt, md) {
    sapply(1:R, function(...)
        draw_and_get(method=met, beta=b_hat, model=md, dgp=dt))
}, params$b_hat,params$met,params$dt,params$md)
```

The program for c.

```r
# main2: The variable naming rule is "method_dgp_model"
set.seed(1234)
params2 <- expand.grid(met=c("OLS", "Ridge", "LASSO"), dt=1:2, md=1:2)
matrix_list <- lapply(1:12, function(x) matrix(nrow = 100, ncol = R))
matrix_list <- lapply(seq_along(matrix_list), function(i) {
    sapply(1:R, function(r){
        draw_and_pred_minus_test(met=params2[i,"met"],
                                 model=params2[i,"md"],
                                 dgp=params2[i,"dt"])
    })
})
names(matrix_list) <- apply(params2, 1, paste0, collapse="_")
```

Result for a.

In this sub-question, the naming rule is "beta_method_dgp_model", for example, `1_OLS_1_1` stands for estimate β_1 by "OLS" and "DGP1" and "Model 1".

```r
##a
mean.for.b1 <- apply(container[,seq(1,24,2)], 2, mean)
sd.for.b1 <- apply(container[,seq(1,24,2)], 2, var) %>% sqrt()
rmses.for.b1 <- apply(container[,seq(1,24,2)], 2,
                      function(col) sqrt((1/length(col)) * sum((col - 1)^2)))
a_result <- cbind(mean.for.b1, sd.for.b1, rmses.for.b1)
a_result ## row names are "beta_method_dgp_model"
```

```
##              mean.for.b1  sd.for.b1 rmses.for.b1
##   1_OLS_1_1     0.9999728 0.05268816   0.05267500
##   1_Ridge_1_1   0.9415650 0.04429994   0.07332222
##   1_LASSO_1_1   0.9344095 0.04816695   0.08136958
##   1_OLS_2_1     0.9983061 0.05187496   0.05188964
##   1_Ridge_2_1   0.9691564 0.04530778   0.05480050
##   1_LASSO_2_1   0.9883570 0.04642489   0.04785136
##   1_OLS_1_2     0.9990179 0.05503857   0.05503357
##   1_Ridge_1_2   0.9291313 0.04753122   0.08532562
```

```
## 1_LASSO_1_2   0.9217798 0.04821886   0.09188193
## 1_OLS_2_2     0.9996304 0.05314407   0.05313207
## 1_Ridge_2_2   0.9680075 0.04452884   0.05482103
## 1_LASSO_2_2   0.9714882 0.04616072   0.05424640
```

Result for b.

In this sub-question, the naming rule is "beta_method_dgp_model", for example, `21_OLS_1_1` stands for estimate β_{21} by "OLS" and "DGP1" and "Model 1".

```
##b
mean.for.b21 <- apply(container[,seq(2,24,2)], 2, mean)
sd.for.b21 <- apply(container[,seq(2,24,2)], 2, var) %>% sqrt()
rmses.for.b21 <- apply(container[,seq(2,24,2)], 2,
                       function(col) sqrt((1/length(col)) * sum((col - 0)^2)))
b_result <- cbind(mean.for.b21, sd.for.b21, rmses.for.b21)
b_result ## row names are "beta_method_dgp_model"
```

```
##                mean.for.b21  sd.for.b21 rmses.for.b21
## 21_OLS_1_1     1.015533e-03 0.052591154   0.052587811
## 21_Ridge_1_1 -7.451429e-06 0.043614923   0.043604019
## 21_LASSO_1_1  4.092341e-04 0.013579568   0.013582339
## 21_OLS_2_1   -1.016475e-03 0.051920706   0.051917675
## 21_Ridge_2_1 -4.305459e-04 0.044431411   0.044422388
## 21_LASSO_2_1 -1.409655e-03 0.038489342   0.038505530
## 21_OLS_1_2    3.971180e-04 0.054216876   0.054204775
## 21_Ridge_1_2 -4.284384e-04 0.043843080   0.043834212
## 21_LASSO_1_2  3.010986e-04 0.009071752   0.009074481
## 21_OLS_2_2   -7.508395e-04 0.053228337   0.053220325
## 21_Ridge_2_2 -1.178877e-03 0.045566448   0.045570306
## 21_LASSO_2_2 -6.788195e-04 0.028943779   0.028944503
```

Result for c.

In this sub-question, the naming rule is "method_dgp_model", for example, `OLS_1_1` stands for the MSPE calculated by "OLS" and "DGP1" and "Model 1".

```
##c
MSPE_method_dgp_model <- numeric(12)
for (i in 1:12){
    sum <- 0
    for (col in 1:R){
        sum <- sum + 0.01*t(matrix_list[[i]][,col]) %*% matrix_list[[i]][,col]
    }
    result <- R^-1 * sum
    MSPE_method_dgp_model[i] <- result
}
names(MSPE_method_dgp_model) <- apply(params2, 1, paste0, collapse="_")
MSPE_method_dgp_model ## "method_dgp_model"
```

```
##   OLS_1_1 Ridge_1_1 LASSO_1_1   OLS_2_1 Ridge_2_1 LASSO_2_1   OLS_1_2 Ridge_1_2
##  1.080565  1.071341  1.023735  1.064871  1.091881  1.066986  1.149662  1.136247
```

```
## LASSO_1_2   OLS_2_2 Ridge_2_2 LASSO_2_2
##  1.033675  1.142114  1.160296  1.114068
```

Appendix: My Functions

```
get_coef <- function(method, beta, model, dgp){
    if (model == 1) X <- X[,1:26]
    if (dgp == 1) y <- y_dgp1 else y <- y_dgp2
    if(method == "OLS"){
        return(lm(y[tra.idx,] ~ X[tra.idx,])$coefficients[beta+1])
    }
    else if (method == "Ridge"){
        ridge_model <- cv.glmnet(X[tra.idx, ],y[tra.idx, ],
                                  alpha = 0,
                                  nfolds =10)
        ridge_best <- glmnet(X, y, alpha = 0,lambda = ridge_model$lambda.min)
        return(coef(ridge_best)[beta+1])
    }
    else if(method == "LASSO"){
        lasso_model <- cv.glmnet(X[tra.idx, ],y[tra.idx, ],
                                  alpha = 1,
                                  nfolds =10)
        lasso_best <- glmnet(X, y, alpha = 1,lambda = lasso_model$lambda.min)
        return(coef(lasso_best)[beta+1])
    }
}

get_pred_minus_test <- function(method, model, dgp){
    if (model == 1) X <- X[,1:26]
    if (dgp == 1) y <- y_dgp1 else y <- y_dgp2
    if (method == "OLS"){
        train_df <- cbind(data.frame(Y=y[tra.idx,]), data.frame(X[tra.idx,]))
        new <- data.frame(X[-tra.idx,])
        my_lm <- lm(Y~., data=train_df)
        return(predict(my_lm, new) - y[-tra.idx,])

    }else if (method == "Ridge"){
        ridge_model <- cv.glmnet(X[tra.idx, ],y[tra.idx, ],
                           alpha = 0,
                           nfolds =10)

        ridge_best <- glmnet(X[tra.idx,], y[tra.idx,], alpha = 0,
                             lambda = ridge_model$lambda.min)

        return(predict.glmnet(ridge_best, X[-tra.idx,]) - y[-tra.idx,])

    }else if(method == "LASSO"){
        lasso_model <- cv.glmnet(X[tra.idx, ],y[tra.idx, ],
                           alpha = 1,
                           nfolds =10)
```

```r
        lasso_best <- glmnet(X[tra.idx,], y[tra.idx,], alpha = 1,
                             lambda = lasso_model$lambda.min)

        return(predict.glmnet(lasso_best, X[-tra.idx,]) - y[-tra.idx,])
    }
}

drawn <- function(...){
    X <<- matrix(rnorm(n * p), nrow = n, ncol = p)
    X <<- cbind(1, X)
    u <<- rnorm(n)
    y_dgp1 <<- X[,1:3] %*% beta[1:3]+u
    y_dgp2 <<- X[,1:21] %*% beta + u
    X <<- X[,2:51]
}

draw_and_get <- function(method, beta, model, dgp){
    drawn()
    get_coef(method, beta, model, dgp)
}

draw_and_pred_minus_test <- function(method, model, dgp){
    drawn()
    get_pred_minus_test(method, model, dgp)
}
```