

STATS 790

Assignment #2

Dean Hansen - 400027416

13 February, 2023

Contents

Question #1	2
Question #2	6
Question 3 (ESL 3.6)	8
Question 4 (ESL 3.19 - Ridge and Lasso only)	9
Question 5 (ESL 3.28)	10
Question 6 (ESL 3.30)	11

Question #1

a) Below we show how to compute the linear regression coefficients for each method.

Naive Method

Our model is $Y = X\beta + \epsilon$, so we can the least squares estimator for β as follows.

$$\text{RSS} = \epsilon' \epsilon \tag{1}$$

$$= (Y - X\beta)'(Y - X\beta) \tag{2}$$

$$= Y'Y - 2Y'X\beta + \beta'X'X\beta \tag{3}$$

$$\tag{4}$$

We differentiate with respect to β and set this quantity to 0.

$$\frac{d\text{RSS}}{d\beta} = -2X'y + 2X'X\beta \tag{5}$$

$$0 = -X'y + X'X\beta \tag{6}$$

$$\hat{\beta} = (X'X)^{-1}X'y \tag{7}$$

$$\tag{8}$$

Thus, the naive method gives coefficients for β as $\hat{\beta} = (X'X)^{-1}X'y$.

QR Decomposition

The QR decomposition assumes the design matrix X can be written in the form $X = QR$ where Q is an orthogonal matrix and R is upper triangular. Since the derivative of the RSS does not change based on the form of X , we can input the QR decomposition directly into the above to find the coefficients.

$$\hat{\beta} = (X'X)^{-1}X'y \quad (9)$$

$$= ((QR)'QR)^{-1}(QR)'y \quad (10)$$

$$= ((R'Q'QR)^{-1}R'Q'y \quad (11)$$

$$= ((R'R)^{-1}R'Qy \quad (12)$$

$$(13)$$

Thus, the QR decomposition gives coefficients for β as $\hat{\beta} = (R'R)^{-1}R'Qy$.

Singular Value Decomposition

The SVD assumes the design matrix X can be written in the form $X = UDV'$ where U and V are orthonormal and D is diagonal. Once again, we can input the SVD directly into the previously derived expression to find the coefficients.

$$\hat{\beta} = (X'X)^{-1}X'y \quad (14)$$

$$= ((UDV')'UDV')^{-1}(UDV')'y \quad (15)$$

$$= (VD'U'UDV')^{-1}VD'U'y \quad (16)$$

$$= (VD'DV')^{-1}VD'U'y \quad (17)$$

$$= (V')^{-1}D^{-1}U'y \quad (18)$$

$$(19)$$

Thus, the QR decomposition gives coefficients for β as $\hat{\beta} = (V')^{-1}D^{-1}U'y$.

Cholesky Decomposition

The Cholesky decomposition assumes the design matrix X can be written in the form $X = LL'$ where L is upper triangular. Once again, we can input the Cholesky decomposition directly into the previously derived expression to find the coefficients.

$$\hat{\beta} = (X'X)^{-1}X'y \quad (20)$$

$$= ((LL')'LL')^{-1}(LL')'y \quad (21)$$

$$= (LL'LL')^{-1}LL'y \quad (22)$$

$$= (LL')^{-1}y \quad (23)$$

$$(24)$$

Thus, the Cholesky decomposition gives coefficients for β as $\hat{\beta} = (LL')^{-1}y$.

b) Below we define and use a function which implements the above for general x and y.

```
#load required packages
library(microbenchmark)
library(rbenchmark)

fit_model <- function(x, y, type = c("naive", "svd", "qr")) {
  type = match.arg(type)
  if (type == "naive") {
    fit$hat <- x %*% solve(t(x) %*% x) %*% t(x)
    fit$coef <- fit$hat %*% y
  }
  if (type == "svd") {
    svd_x <- svd(x)
    fit$hat <- svd_x$u %*% t(svd_x$u)
    fit$coef <- fit$hat %*% y
  }
  if (type == "qr") {
    svd_x <- svd(x)
    fit$hat <- svd_x$u %*% t(svd_x$u)
    fit$coef <- fit$hat %*% y
  }
}
```

```
}
```

Plotting on log-log scale the scaling behaviour of each method.

Question #2

Below we implement ridge regression via data augmentation and using the native *glmnet* implementation of ridge regression.

```
# load required packages
library(glmnet)
library(Matrix)
library(tidyverse)

# write function to compute ridge
df <- read.table("https://hastie.su.domains/ElemStatLearn/datasets/prostate.data")
df_x_scaled <- scale(df[,1:8],TRUE,TRUE) %>% as_tibble()
df_scaled <- cbind(df_x_scaled,
                   outcome = df[,9],
                   train = as.numeric(df[,10])) %>% as_tibble()

df_train <- df_scaled %>% filter(train == 1) %>% select(-train)
df_test <- df_scaled %>% filter(train == 0) %>% select(-train)

train_x <- df_train %>% select(-outcome) %>% as.matrix()
train_y <- df_train %>% select(outcome) %>% as.matrix()

test_x <- df_test %>% select(-outcome) %>% as.matrix()
test_y <- df_test %>% select(outcome) %>% as.matrix()

# data augmentation defined fit
my_ridge <- function(x, y, alpha, niter = 1000) {
  y_prime <- append(y, seq(1, 1, length.out = length(y)))
}
```

```
# glmnet implementation
```

```
native_ridge <- glmnet(x = train_x, y = train_y, alpha = 0)
```

Question 3 (ESL 3.6)

The ridge estimator is given by

$$\hat{\beta} = (X'X + \lambda I)^{-1} X'y$$

. Assuming that $\beta \sim N(0, \tau I)$ and $y \sim N(X\beta, \sigma^2 I)$, we have:

$$\hat{\beta} = \operatorname{argmax}_{\beta} Pr(y|X, \beta) Pr(\beta) \quad (25)$$

$$= \operatorname{argmax}_{\beta} e^{-\frac{1}{2\tau^2} \beta' \beta} e^{-\frac{1}{2\sigma^2} (y - X\beta)'(y - X\beta)} \quad (26)$$

$$= \operatorname{argmax}_{\beta} e^{-\frac{1}{2\tau^2} \beta' \beta - \frac{1}{2\sigma^2} (y - X\beta)'(y - X\beta)} \quad (27)$$

Next, we take the logarithm of both sides and turn it into a minimization problem.

$$\log(\hat{\beta}) = \operatorname{argmin}_{\beta} \frac{1}{2\tau^2} \beta' \beta + \frac{1}{2\sigma^2} (y - X\beta)'(y - X\beta) \quad (28)$$

$$= \operatorname{argmin}_{\beta} \frac{1}{2\tau^2} \|\beta\|_2^2 + \frac{1}{2\sigma^2} (y - X\beta)'(y - X\beta) \quad (29)$$

$$= \operatorname{argmin}_{\beta} \frac{\sigma^2}{\tau^2} \|\beta\|_2^2 + (y - X\beta)'(y - X\beta) \quad (30)$$

This is same quantity that is minimized for Ridge Regression where $\lambda = \frac{\sigma^2}{\tau^2}$. Further, our prior and likelihood are both Gaussian, so their product is Gaussian (mean = mode). Thus, the MLE for $\hat{\beta}$ is the mean of the posterior distribution.

Question 4 (ESL 3.19 - Ridge and Lasso only)

Consider the minimization problem below for elastic-net, where $\alpha = 1$ is lasso and $\alpha = 2$ is ridge.

$$\min_{\beta} RSS + \lambda \sum_{j=1}^p |\beta|_j^{\alpha} \tag{31}$$

Question 5 (ESL 3.28)

Question 6 (ESL 3.30)