STATS 790

Assignment #2

Dean Hansen - 400027416

14 February, 2023

Contents

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.

$$
\begin{aligned}
\text{RSS} &= \epsilon'\epsilon && (1)\\
&= (Y - X\beta)'(Y - X\beta) && (2)\\
&= Y'Y - 2Y'X\beta + \beta'X'X\beta && (3)\\
& && (4)
\end{aligned}
$$

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

$$
\begin{aligned}
\tfrac{dRSS}{d\beta} &= -2X'y + 2X'X\beta && (5)\\
0 &= -X'y + X'X\beta && (6)\\
\hat{\beta} &= (X'X)^{-1}X'y && (7)\\
& && (8)
\end{aligned}
$$

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.

$$
\begin{aligned}
\hat{\beta} &= (X'X)^{-1}X'y && (9)\\
&= ((QR)'QR)^{-1}(QR)'y && (10)\\
&= ((R'Q'QR)^{-1}R'Q'y && (11)\\
&= ((R'R)^{-1}R'Qy && (12)\\
& && (13)
\end{aligned}
$$

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.

$$
\begin{aligned}
\hat{\beta} &= (X'X)^{-1}X'y && (14)\\
&= ((UDV')'UDV')^{-1}(UDV')'y && (15)\\
&= (VD'U'UDV')^{-1}VD'U'y && (16)\\
&= (VD'DV')^{-1}VD'U'y && (17)\\
&= (V')^{-1}D^{-1}U'y && (18)\\
& && (19)
\end{aligned}
$$

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.

$$
\begin{aligned}
\hat{\beta} &= (X'X)^{-1}X'y && (20)\\
&= ((LL')'LL')^{-1}(LL')'y && (21)\\
&= (LL'LL')^{-1}LL'y && (22)\\
&= (LL')^{-1}y && (23)\\
& && (24)
\end{aligned}
$$

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_naive <- function(x, y) {
  x %*% solve(t(x) %*% x) %*% t(x) %*% y
}

fit_svd <- function(x, y) {
  svd(x)$u %*% t(svd(x)$u) %*% y
}

fit_qr <- function(x, y) {
  t(t(solve.qr(qr(x), y)))
}

#different magnitudes
simfun <- function(n, p) {
    y <- rnorm(n)
    X <- matrix(rnorm(p*n), ncol = p)
```

```r
    list(X = X, y = y)
}

s <- simfun(1000, 10)
nvec <- round(10^seq(2, 5, by = 0.25))


## set aside some storage for the results ...
n <- 1000

for (i in seq_along(n)) {
    s <- simfun(nvec[i], p = 10)
    m <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_naive(s$X, s$y))
results <- summary(m)

m_qr <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_qr(s$X, s$y))
results_qr <- summary(m_qr)

m_svd <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_svd(s$X, s$y))
results_svd <- summary(m_svd)
}


## set aside some storage for the results ...
n <- 10000
```

```r
for (i in seq_along(n)) {
    s <- simfun(nvec[i], p = 100)
    m <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_naive(s$X, s$y))
results <- summary(m)

m_qr <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_qr(s$X, s$y))
results_qr <- summary(m_qr)

m_svd <- microbenchmark(
    lm.fit(s$X, s$y),
    fit_svd(s$X, s$y))
results_svd <- summary(m_svd)
}
```

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.

```r
# 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_scaled_x <- df_scaled %>% select(-train, -outcome) %>% as.matrix()
df_scaled_y <- df_scaled %>% select(outcome) %>% as.matrix()



# data augmentation defined fit
my_ridge <- function(x, y) {
  y_prime <- append(y, seq(1, 1, length.out = length(y)))
  x <- rbind(x, diag(x = 1, nrow = dim(x)))
  hat <- qr.solve(x, y_prime)
}

# glmnet implementation
native_ridge <- glmnet(x = df_scaled_x, y = df_scaled_y, alpha = 0)
cv_native_ridge <- cv.glmnet(x = df_scaled_x, df_scaled_y, alpha = 0)
optimal_lambda <- cv_native_ridge$lambda.min
optimal_lambda
```

```
## [1] 0.08434274
```

Question 3 (ESL 3.6)

The ridge estimator is given by

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

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

$$\hat{\beta} = \text{argmax } Pr(y|X, \beta)Pr(\beta) \tag{25}$$

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

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

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

$$log(\hat{\beta}) = \text{argmin } \tfrac{1}{2\tau^2}\beta'\beta + \tfrac{1}{2\sigma^2}(y - X\beta)'(y - X\beta) \tag{28}$$

$$= \text{argmin } \tfrac{1}{2\tau^2}\|\beta\|_2^2 + \tfrac{1}{2\sigma^2}(y - X\beta)'(y - X\beta) \tag{29}$$

$$= \text{argmin } \tfrac{\sigma^2}{\tau^2}\|\beta\|_2^2 + (y - X\beta)'(y - X\beta) \tag{30}$$

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)

From the notes, we have that the Ridge weights are of the form $\frac{d_j^2}{d_j^2+\lambda}$. Thus, when $\lambda \to 0$, this quantity gets bigger and thus $\|\hat{\beta}^{ridge}\|$ increases.

For Lasso, the same will hold, since as we relax the penalty, the coefficients should increase towards the usual least squares estimate.

Question 5 (ESL 3.28)

See file titled ‘Q5.pdf’ in github folder.

Question 6 (ESL 3.30)

See file titled ‘Q6.pdf’ in github folder.