STATS 790
Assignment 2

Doudou Jin – 400174871

2/13/23

Q1

The multivariate linear regression model is: $Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_k X_k + \epsilon$

In matrix format, with n observations and k predictors, $\vec{Y} = \mathbf{X}\vec{\beta} + \vec{\epsilon}$, where

$$\vec{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix}$$

design matrix $\mathbf{X}$

$$\mathbf{X} = \begin{pmatrix} 1 & X_{11} & \cdots & X_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & \cdots & X_{nk} \end{pmatrix}$$

$$\vec{\beta} = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_k \end{pmatrix}$$

$$\vec{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

Compute $\vec{\beta}$ using naive linear algebra

using least squares estimation of $\vec{\beta}$:

$$L = \sum_{i=1}^{n} \epsilon_i^2 = \vec{\epsilon}^\top \vec{\epsilon} = (\vec{Y} - \mathbf{X}\vec{\beta})^\top (\vec{Y} - \mathbf{X}\vec{\beta})$$

To minimize L, we differentiate and set it to 0:

$$\frac{\partial(\vec{Y}-\mathbf{X}\vec{\beta})^\top(\vec{Y}-\mathbf{X}\vec{\beta})}{\partial\vec{Y}} = -2\mathbf{X}^\top\vec{Y}+2\mathbf{X}^\top\mathbf{X}\vec{\beta}$$
$$= 0$$

Then we get the normal equation:

$$\Rightarrow \mathbf{X}^\top\mathbf{X}\vec{\beta} = \mathbf{X}^\top\vec{Y}$$

And thus,

$$\hat{\vec{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\vec{Y}$$

Compute $\vec{\beta}$ using QR decomposition

In QR decomposition, $\mathbf{X}$ can be decomposed as $\mathbf{X} = \mathbf{QR}$, where $\mathbf{Q}$ is a nx(k+1) orthonormal matrix and $\mathbf{R}$ is a (k+1)x(k+1) upper triangular matrix. By the property of orthogonality, $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}$, then using the normal equation, we have

$$\mathbf{X}^\top\mathbf{X}\vec{\beta} = \mathbf{X}^\top\vec{Y}$$
$$\mathbf{QR}^\top\mathbf{QR}\vec{\beta} = \mathbf{QR}^\top\vec{Y}$$
$$\mathbf{R}^\top\mathbf{Q}^\top\mathbf{QR}\vec{\beta} = \mathbf{R}^\top\mathbf{Q}^\top\vec{Y}$$
$$\mathbf{R}^\top\mathbf{R}\vec{\beta} = \mathbf{R}^\top\mathbf{Q}^\top\vec{Y}$$
$$\mathbf{R}\vec{\beta} = \mathbf{Q}^\top\vec{Y}$$

Thus,

$$\hat{\vec{\beta}} = \mathbf{R}^{-1}\mathbf{Q}^\top\vec{Y}$$

Compute $\vec{\beta}$ using SVD

In SVD, $\mathbf{X}$ can be decomposed as $\mathbf{X} = \mathbf{USV}^\top$, where $\mathbf{U}$ is a nxn matrix, $\mathbf{S}$ is a nx(k+1) matrix and $\mathbf{V}$ is a (k+1)x(k+1) matrix. Both $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, and so $\mathbf{U}^\top\mathbf{U} = \mathbf{V}^\top\mathbf{V} = \mathbf{I}$.

$$\mathbf{X}^\top \mathbf{X}\vec{\beta} = \mathbf{X}^\top \vec{Y}$$
$$(\mathbf{USV}^\top)^\top(\mathbf{USV}^\top)\vec{\beta} = (\mathbf{USV}^\top)^\top \vec{Y}$$
$$\mathbf{VS}^\top\mathbf{U}^\top\mathbf{USV}^\top\vec{\beta} = \mathbf{VS}^\top\mathbf{U}^\top\vec{Y}$$
$$\mathbf{VS}^\top\mathbf{SV}^\top\vec{\beta} = \mathbf{VS}^\top\mathbf{U}^\top\vec{Y}$$
$$\mathbf{SV}^\top\vec{\beta} = \mathbf{U}^\top\vec{Y}$$

Thus,

$$\hat{\vec{\beta}} = \mathbf{VS}^{-1}\mathbf{U}^\top\vec{Y}$$

Compute $\vec{\beta}$ using Cholesky decomposition

In Cholesky decomposition, if $\mathbf{X}^\top\mathbf{X}$ is positive definite matrix, it can be decomposed as $\mathbf{X}^\top\mathbf{X} = \mathbf{LL}^\top$, where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{L}^\top$ is the conjugate transpose of $\mathbf{L}$.

Using the normal equation,

$$\mathbf{LL}^\top\vec{\beta} = \mathbf{X}^\top\vec{Y}$$

Let $\mathbf{L}^\top\vec{\beta} = Z$ and $\mathbf{L}Z = \mathbf{X}^\top\vec{Y}$. Then we can solve for Z since $\mathbf{L}$, $\mathbf{X}^\top$, $\vec{Y}$ are known. After solving Z, we can solve for $\vec{\beta}$ using $\mathbf{L}^\top\vec{\beta} = Z$, where $\mathbf{L}^\top$ and Z are known.

So,

$$\hat{\vec{\beta}} = (\mathbf{L}^\top)^{-1}Z$$

Compare running time of Naive Linear Algebra, QR Decomposition and SVD

```
library(microbenchmark)
library(ggplot2)
library(dplyr)
library(pracma)
```

```r
#simulate data
test_matrix <- function(n,p){
  X <- matrix(rnorm(n*p), n, p)
  y <- X %*% rnorm(p) + rnorm(p)
  list(X = X, y = y)
}

set.seed(1)
matrix1 <- test_matrix(1000,100)
```

```r
# simply lm()
test_model <- function(X,y){
  b <- data.matrix(lm(y ~ X+0)$coef)
  b <- as.data.frame(b, row.names = 1:nrow(b))
  b
}

#naive linear algebra
linear_alg <- function(X,y){
  b <- data.matrix(solve(t(X) %*% X, t(X) %*% y))
  b <- as.data.frame(b, row.names = 1:nrow(b))
  b
}

#QR decomposition
qr_decomp <- function(X,y){
  q_r <- qr(X)
  b <- data.matrix(backsolve(qr.R(q_r), t(qr.Q(q_r)) %*% y))
  b <- as.data.frame(b, row.names = 1:nrow(b))
  b
}
```

```r
#SVD
sv_decomp <- function(X,y){
  s_v <- svd(X)
  b <- data.matrix(solve(diag(s_v$d) %*% t(s_v$v), t(s_v$u) %*% y))
  b <- as.data.frame(b, row.names = 1:nrow(b))
  b
}
```

```r
#check
stopifnot(all.equal(test_model(matrix1$X,matrix1$y),
                    linear_alg(matrix1$X,matrix1$y)))
stopifnot(all.equal(test_model(matrix1$X,matrix1$y),
                    qr_decomp(matrix1$X,matrix1$y)))
stopifnot(all.equal(test_model(matrix1$X,matrix1$y),
                    sv_decomp(matrix1$X,matrix1$y)))
```

```r
# benchmarking
bm <- microbenchmark(test_model(matrix1$X,matrix1$y)
                    , linear_alg(matrix1$X,matrix1$y)
                    , qr_decomp(matrix1$X,matrix1$y)
                    , sv_decomp(matrix1$X,matrix1$y))
result1 <- summary(bm)
```

```r
#try more n
nvec <- round(10^seq(2, 5, by = 0.5))
results <- data.frame()

for (i in seq_along(nvec)) {
    s <- test_matrix(nvec[i], 100)
    mbm <- microbenchmark(test_model(matrix1$X,matrix1$y)
                    , linear_alg(matrix1$X,matrix1$y)
```

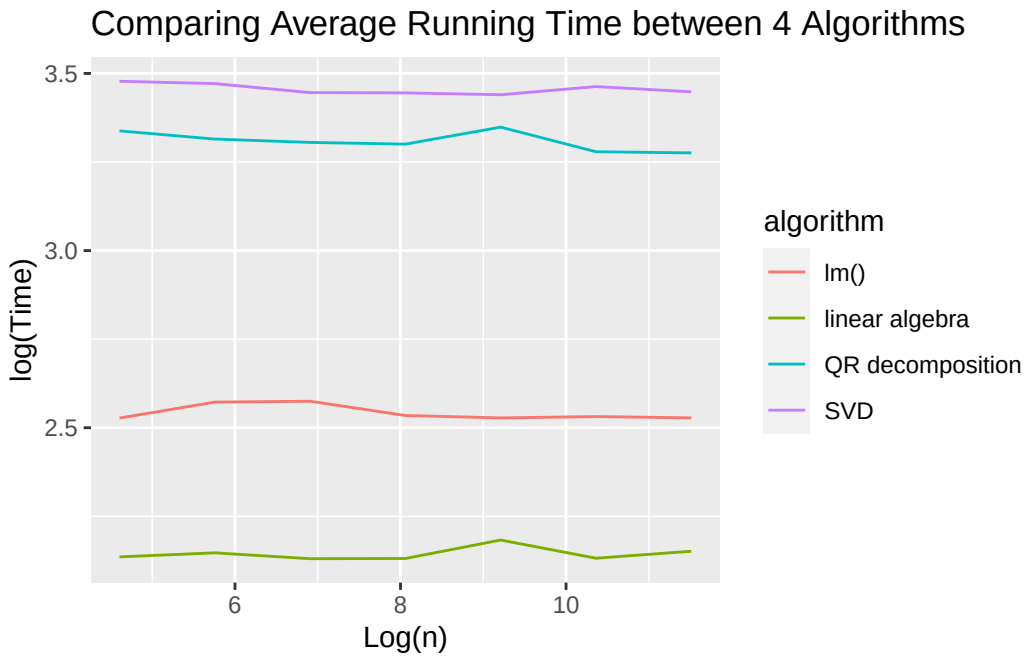
```
                    , qr_decomp(matrix1$X,matrix1$y)
                    , sv_decomp(matrix1$X,matrix1$y))
    results <- rbind(results, summary(mbm))
}
```

```
nvec <- rep(nvec, each = 4)
results <- cbind(results, nvec)
```

```
results_new <- results %>%
                    select(expr, mean, nvec) %>%
                    mutate(logmean = log(mean), lognvec=log(nvec))
levels(results_new$expr) <- c("lm()", "linear algebra",
                              "QR decomposition", "SVD")
```

Plot

```
#compare using log-log plot
ggplot(data = results_new, aes(x=lognvec, y=logmean, group=expr)) +
  geom_line(aes(color=expr)) +
  labs(color='algorithm', x="Log(n)", y="log(Time)") +
  ggtitle("Comparing Average Running Time between 4 Algorithms")
```



The plot shows that native linear algebra runs fastest among the four algorithms, even faster than the native implementation of linear regression. In contrast, the SVD algorithm runs the slowest of the three other algorithms.

Model

```
#fit log-log model
logmodel <- lm(logmean~lognvec, data = results_new)
summary(logmodel)
```

```
Call:
lm(formula = logmean ~ lognvec, data = results_new)

Residuals:
     Min       1Q   Median       3Q      Max
-0.73762 -0.42841  0.06583  0.51162  0.60687

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept)  2.886811   0.386208   7.475 6.17e-08 ***
lognvec     -0.002978   0.046079  -0.065    0.949
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5615 on 26 degrees of freedom
Multiple R-squared:  0.0001606, Adjusted R-squared:  -0.03829
F-statistic: 0.004177 on 1 and 26 DF,  p-value: 0.949
```

The log-log model is log(mean) = 2.9133 - 0.0019*log(n)

Q2

```
library(glmnet)
```

```
#read table
prostate <- read.delim("prostate.data", row.names=1)

#split to training and test sets
prostate_train <- filter(prostate, train == "TRUE")
prostate_test <- filter(prostate, train == "FALSE")

#remove last column "train"
prostate_train <- prostate_train[1:9]

prostate_train_matrix <- as.matrix(subset(prostate_train, select = -lpsa))
prostate_test_x <- as.matrix(subset(prostate_test, select = -c(train, lpsa)))
prostate_test1 <- subset(prostate_test, select = -train)
```

```
#native implementation of ridge regression
ridge <- function(x,y){
  set.seed(1)
  cv <- cv.glmnet(x, y, alpha = 0, standardize=TRUE)
  lambda <- cv$lambda.min
  rid <- glmnet(x, y, alpha = 0, standardize=TRUE, lambda = lambda)
  list("model" = rid, "opt_lambda" = lambda)
}
```

```
RR <- ridge(prostate_train_matrix,prostate_train$lpsa)
RR
```

```
$model

Call:  glmnet(x = x, y = y, alpha = 0, lambda = lambda, standardize = TRUE)
```

```
  Df  %Dev  Lambda
1  8 68.87 0.08789

$opt_lambda
[1] 0.08788804
```

```r
# ridge regression by data augmentation
# augment data by adding gaussian noise
ridge_aug <- function(x){
  set.seed(1)
  len <- dim(x)[1] * dim(x)[2]
  noise <- matrix(rnorm(len, mean=0, sd = 1), dim(x)[1])
  aug <- as.matrix(x) + noise
  aug <- rbind(aug, as.matrix(x))
  aug <- as.data.frame(aug)
  osl <- lm(lpsa~., data = aug)
}
```

```r
RR_aug <- ridge_aug(prostate_train)
RR_aug
```

```
Call:
lm(formula = lpsa ~ ., data = aug)

Coefficients:
(Intercept)       lcavol      lweight          age         lbph          svi
  -0.589414     0.431696     0.123882     0.009317     0.186757     0.354539
        lcp      gleason        pgg45
   0.141671     0.205649    -0.001791
```

```r
#R^2 from ridge regression by data augmentation
pred <- predict(RR_aug, newdata = prostate_test[,-9])
sst <- sum((prostate_test$lpsa - mean(prostate_test$lpsa))^2)
ssr <- sum((prostate_test$lpsa - pred)^2)
rsq <- 1 - ssr/sst
rsq
```

```
[1] 0.514647
```

```r
#R^2 from ridge regression
RRpred <- predict(RR$model, s=RR$opt_lambda, newx = prostate_test_x)
RRsst <- sum((prostate_test$lpsa - mean(prostate_test$lpsa))^2)
RRssr <- sum((prostate_test$lpsa - RRpred)^2)
RRrsq <- 1 - RRssr/RRsst
RRrsq
```

```
[1] 0.5289966
```

```r
set.seed(1)
bm_ridge <- microbenchmark(ridge(prostate_train_matrix,prostate_train$lpsa)
                     , ridge_aug(prostate_train))
result_ridge <- summary(bm_ridge)
```

```r
data.frame(result_ridge$expr, result_ridge$mean)
```

```
                                result_ridge.expr result_ridge.mean
1 ridge(prostate_train_matrix, prostate_train$lpsa)         53.168495
2                          ridge_aug(prostate_train)          1.454887
```

The R^2 of ridge regression without data augmentation is 52.90%, while the R^2 of ridge regression by data augmentation is 51.46%. This indicates that the model using ridge regression without data augmentation explains more variations compared to the model using ridge regression with data augmentation. However, the algorithm of ridge regression by data augmentation is much faster than the native ridge regression.

Q3

By ESL Equation 3.44, we know that

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

By the definition of posterior distribution,

$$p(\beta|\mathbf{y}) \propto p(\mathbf{y}|\beta)p(\beta)$$

where $p(\mathbf{y}|\beta)$ is the likelihood, with $\mathbf{y} \sim N(\mathbf{X}\beta, \sigma^2\mathbf{I})$ and $p(\beta)$ is the prior distribution with $\beta \sim N(0, \tau\mathbf{I})$. Since the product of two gaussians is also a gaussian, the mean of the posterior distribution is the same value as it's mode.

Let β be the mode and thus the negative log-posterior density of β is

$$\begin{aligned}
\hat{\beta} &= \underset{\beta}{argmax} \quad log(p(\mathbf{y}|\beta)) + log(p(\beta)) \\
&= \underset{\beta}{argmin} \quad C + \frac{(\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta)}{2\sigma^2} + \frac{\beta^\top \beta}{2\tau} \\
&= \underset{\beta}{argmin} \quad (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) + \frac{\sigma^2}{\tau}\beta^\top \beta
\end{aligned}$$

By ESL Equation 3.43, both equation is equavelent when $\lambda = \frac{\sigma^2}{\tau}$, so

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X} + \frac{\sigma^2}{\tau}\mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

$\hat{\beta}$ is the estimated mean and mode of posterior distribution, and thus, $\hat{\beta} = \hat{\beta}^{ridge}$ when $\lambda = \frac{\sigma^2}{\tau}$.

Therefore, the ridge regression estimate is the mean and mode of the posterior distribution, under a Gaussian prior $\beta \sim N(0, \tau\mathbf{I})$, and Gaussian sampling model $\mathbf{y} \sim N(\mathbf{X}\beta, \sigma^2\mathbf{I})$, when $\lambda = \frac{\sigma^2}{\tau}$.

Q4

Using SVD, let $\mathbf{X} = \mathbf{UDV}^\top$, by ESL Equation 3.47,

$$\hat{\beta}^{ridge} = \mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y} = \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2 + \lambda} u_j^\top \mathbf{y}$$

where u_j are the column of $\mathbf{U}$ and d_j are the diagonal entries of diagonal matrix $\mathbf{D}$.

So, $\hat{\beta}^{ridge}$ is

$$\begin{aligned}
\hat{\beta}^{ridge} &= \mathbf{X}^{-1}\mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y} \\
&= \mathbf{VD}^{-1}\mathbf{U}^{-1}\mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y} \\
&= \mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y}
\end{aligned}$$

Thus,

$$\begin{aligned}
\|\hat{\beta}^{ridge}\| &= (\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y})^\top\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y} \\
&= \mathbf{y}^\top\mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{V}^\top\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{DU}^\top\mathbf{y} \\
&= \mathbf{y}^\top\mathbf{UD}(\mathbf{D}^2 + \lambda\mathbf{I})^{-2}\mathbf{U}^\top\mathbf{y} \\
&= \sum_{j=1}^{p} \mathbf{y} u_j \frac{d_j^2}{(d_j^2 + \lambda)^2} u_j^\top \mathbf{y}
\end{aligned}$$

So, as $\lambda \to 0$, $\frac{d_j^2}{(d_j^2+\lambda)^2}$ increases, and so $\|\hat{\beta}^{ridge}\|$ increases.

Also, by ESL 3.41 and 4.42 the penality term is $\lambda \sum_{i=1}^{p} \beta_j^2$. To keep the formula constant, $\sum_{i=1}^{p} \beta_j^2$ should increase as $\lambda \to 0$. So the value of t increases as λ decreases.

In lasso, the penality is replaced by $\sum_{i=1}^{p} |\beta_j|$, but the relationship between t and λ still holds. So in lasso, as $\lambda \to 0$, $\|\hat{\beta}^{lasso}\|$ increases, as well.

Q5

By ESL Equation 3.51, we have

$$\hat{\beta}^{lasso} = \underset{\beta}{argmin} \quad \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{m=1}^{p} x_{im}\beta_m)^2$$
$$\text{subject to} \quad \sum_{m=1}^{p} |\beta_m| \leq t$$

Then suppose we augment $\mathbf{X}$ with an identical copy $X_j^* = X_j$. Let $\beta_j = \tilde{\beta}_j + \tilde{\beta}_j^*$, where $\tilde{\beta}_j$ is the coefficient of $\mathbf{X_j}$ and $\tilde{\beta}_j^*$ is the coefficient of $\mathbf{X_j^*}$ in new problem.

The new lasso problem becomes

$$\hat{\beta}^{lassoNew} = \underset{\beta}{argmin} \quad \sum_{i=1}^{N}(y_i - \beta_0 - \sum_{m\neq j}^{p} x_{im}\beta_m - x_{ij}\beta_j)^2$$
$$\text{subject to} \quad \sum_{m\neq j}^{p} |\beta_m| + |\beta_j| \leq t$$

The penalty term has been adjusted to include the penalty on β_j. To let this becomes the original lasso problem, we need to find the optimal values of $\tilde{\beta}_j$ and $\tilde{\beta}_j^*$ so that

$$\sum_{m\neq j}^{p} |\beta_m| + |\beta_j| = \sum_{m\neq j}^{p} |\beta_m| + |\tilde{\beta}_j| + |\tilde{\beta}_j^*| \leq t$$

By the property of absolute value, $|\beta_j| = |\tilde{\beta}_j + \tilde{\beta}_j^*| \leq |\tilde{\beta}_j| + |\tilde{\beta}_j^*|$.

So only when $\tilde{\beta}_j$ and $\tilde{\beta}_j^*$ have the same sign i.e. $\tilde{\beta}_j\tilde{\beta}_j^* > 0$, the condition holds.

Since the value of t remains the same, $\beta_j = \hat{\beta}_j = \tilde{\beta}_j + \tilde{\beta}_j^* = a$

Therefore, when the exact collinearity happens, the sum of coefficients of two identical predictors will be the original coefficient of one predictor. Moreover, the coefficients of two identical predictors must have the same sign.

Q6

Let augment the matrix $\mathbf{X}$ with p additional rows and augment $\mathbf{y}$ with p zeros. Then

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X} \\ \gamma \mathbf{I}_p \end{pmatrix}$$

$$\tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y} \\ 0_p \end{pmatrix}$$

Then the residual sum of squares is

$$\begin{aligned} RSS(\tilde{\beta}) &= \|\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\beta\|_2^2 \\ &= \| \begin{pmatrix} \mathbf{y} - \mathbf{X}\beta \\ \gamma\beta \end{pmatrix} \|_2^2 \\ &= \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \gamma^2\|\beta\|_2^2 \end{aligned}$$

Then turn this into a lasso problem, we add the penalty term and minimize it,

$$\Rightarrow \underset{\beta}{argmin} \quad \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \gamma^2\|\beta\|_2^2 + \tilde{\lambda}\|\beta\|_1$$

Thus, when $\gamma^2 = \lambda\alpha$ and $\tilde{\lambda} = \lambda(1-\alpha)$, the elastic-net optimization problem

$$\underset{\beta}{argmin} \quad \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda[\alpha\|\beta\|_2^2 + (1-\alpha)\|\beta\|_1]$$

turns into a lasso problem.

Reference

Borchers, Hans W. 2022. *Pracma: Practical Numerical Math Functions*. https://CRAN.R-project.org/package=pracma.

Mersmann, Olaf. 2021. *Microbenchmark: Accurate Timing Functions*. https://CRAN.R-project.org/package=microbenchmark.

Simon, Noah, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2011. "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software* 39 (5): 1–13. https://doi.org/10.18637/jss.v039.i05.

Tibshirani, R., T. Hastie, and J. H. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction : With 200 Full-Color Illustrations*. Springer Series in Statistics. Springer. https://books.google.ca/books?id=SECjnQAACAAJ.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Romain François, Lionel Henry, and Kirill Müller. 2022. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.