stat790 a2

2023-02-12

```
library(matlib)
```

```
## Warning in rgl.init(initValue, onlyNULL): RGL: unable to open X11 display
## Warning: 'rgl.init' failed, running with 'rgl.useNULL = TRUE'.
```

```
library(microbenchmark)
```

1a)

Naive linear algebra:

Suppose X is a full column rank matrix and y is the vector contains all response variable values, we have:

$\vec{\beta} = (X'X)^{-1}X'\vec{y}$

QR decomposition:

suppose X could be decomposed to an orthogonal matrix Q and an upper triangular matrix R,

$\vec{\beta} = R^{-1}Q'\vec{y}$

SVD:

Suppose X could be decomposed as the product of U, Σ, and V', where U and V are orthogonal matriics andd Σ is diagonal matrix with the singular values of X

$\vec{\beta} = V\Sigma^{-1}U'\vec{y}$

Cholesky decomposition:

Suppose X could be decomposed as the product of L and L', L is a lower triangular matrix.

$\vec{\beta} = (L')^{-1}L^{-1}X'\vec{y}$

1b

```
naive1=function(n,p){
  s.star=1
  z1= c(rep(1, n/2), rep(0, n/2))
  Z = matrix(rnorm(n * (p - 2)), nrow = n, ncol = (p - 2))
  X = cbind(1, z1, Z)
  b.star = p^(-1) * rnorm(p)#randomly select
  y = X %*% b.star + s.star * rnorm(n)
  inv(t(X) %*% X) %*% t(X) %*% y
  }
```

```
set.seed(1)
QR=function(n,p){
  s.star=1
  z1= c(rep(1, n/2), rep(0, n/2))
  Z = matrix(rnorm(n * (p - 2)), nrow = n, ncol = (p - 2))
  X = cbind(1, z1, Z)
  out = qr(x = X)
  b.star = p^(-1) * rnorm(p)#randomly select
  y = X %*% b.star + s.star * rnorm(n)
  qr.coef(qr(x = X), y = y)
}
```

```
set.seed(1)
svd1=function(n,p){
  s.star=1
  z1 = c(rep(1, n/2), rep(0, n/2))
  Z = matrix(rnorm(n * (p - 2)), nrow = n, ncol = (p - 2))
  X = cbind(1, z1, Z)
  b.star = p^(-1) * rnorm(p)#randomly select
  y = X %*% b.star + s.star * rnorm(n)
  svd(X)$v %*% diag(1 / svd(X)$d) %*% t(svd(X)$u) %*% y
}
```

```
summary(microbenchmark(naive1(100000,10)))$mean
```

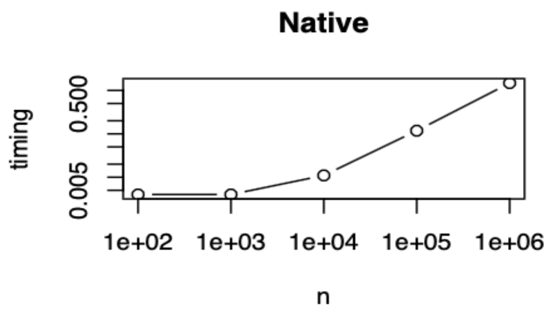
```
## [1] 105.8302
```

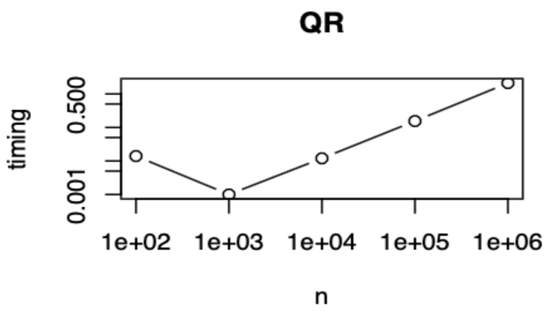
```
nvec=c(10^(seq(2,6)))
timing=c(rep(NA,length(nvec)))
for (i in nvec){
    a=match(i,nvec)
    timing[a]=system.time(naive1(i,10))["elapsed"]
  }
```

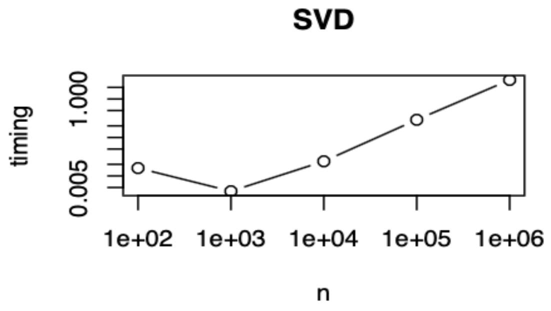
```
timing1=c(rep(NA,length(nvec)))
for (i in nvec){
    a=match(i,nvec)
    timing1[a]=system.time(QR(i,10))["elapsed"]
  }
```

```
timing2=c(rep(NA,length(nvec)))
for (i in nvec){
    a=match(i,nvec)
    timing2[a]=system.time(svd1(i,10))["elapsed"]
  }
```

```
par(mfrow=c(2,2))
plot(nvec,timing,log="xy",type="b",xlab="n",ylab="timing",main="Native")
plot(nvec,timing1,log="xy",type="b",xlab="n",ylab="timing",main="QR")
plot(nvec,timing2,log="xy",type="b",xlab="n",ylab="timing",main="SVD")
```







```
lm(log(timing)~log(nvec))
```

```
##
## Call:
## lm(formula = log(timing) ~ log(nvec))
##
## Coefficients:
## (Intercept)    log(nvec)
##     -9.5433       0.6605
```

```
lm(log(timing1)~log(nvec))
```

```
##
## Call:
## lm(formula = log(timing1) ~ log(nvec))
##
## Coefficients:
## (Intercept)    log(nvec)
##     -9.3731       0.6547
```

```
lm(log(timing2)~log(nvec))
```

```
##
## Call:
## lm(formula = log(timing2) ~ log(nvec))
##
## Coefficients:
## (Intercept)    log(nvec)
##     -8.6283       0.6438
```

2

```
data_link <- "https://hastie.su.domains/ElemStatLearn/datasets/prostate.data"
prostate <- read.table(data_link, header = TRUE)


set.seed(1)
traindata <- prostate[which(prostate$train=="TRUE"), ]
testdata <- prostate[which(prostate$train=="FALSE"), ]


ridge.augment <- function(x, y, lambda) {
  n <- nrow(x)
  p <- ncol(x)
  x=cbind(1,x)
  X_t <- rbind(x, sqrt(lambda) * diag(c(0,rep(1,p))))
  y_t <- c(y, rep(0, p+1))

  beta_t <- solve(t(X_t) %*% X_t, t(X_t) %*% y_t)
  beta_t[1:p]

}


x_train <- as.matrix(traindata[, 1:8])
y_train <- traindata[, 9]
#lambdas <- seq(0, 1, length.out = 10)
lambdas=0.1
start_time1 <- Sys.time()
betas <- matrix(NA, nrow = length(lambdas), ncol = ncol(x_train))
for (i in seq_along(lambdas)) {
  betas[i, ] <- ridge.augment(x_train, y_train, lambdas[i])
}
end_time1 <- Sys.time()
total_time1 <- end_time1 - start_time1

# Fit a native ridge regression model using the glmnet package
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
start_time2 <- Sys.time()
glmnet_fit <- glmnet(x_train, y_train, alpha = 0, lambda = lambdas)
end_time2 <- Sys.time()
total_time2 <- end_time2 - start_time2
t(betas)
```

```
##             [,1]
## [1,]  0.44807782
## [2,]  0.57664742
## [3,]  0.60980133
## [4,] -0.01890718
## [5,]  0.14499264
## [6,]  0.72575413
## [7,] -0.20366883
## [8,] -0.03053894
```

```
coef(glmnet_fit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  0.067417167
## lcavol       0.483882238
## lweight      0.598968504
## age         -0.014335661
## lbph         0.137022687
## svi          0.673005902
## lcp         -0.108028074
## gleason      0.021020364
## pgg45        0.006871759
```

```
c(total_time1,total_time2)
```

```
## Time differences in secs
## [1] 0.01718736 0.01391530
```

We could find that ridge regression by data augmentation is similar to the native implementation, but has more shrinkage. also, the running time of native implementation of ridge regression is shorter than ridge regression by data augmentation.

ex3.6

In ridge regression, we want to minimize $||\vec{y} - X\vec{\beta}||_2^2 + \lambda||\vec{\beta}||_2^2$

The posterior distribution of $\vec{\beta}$ given $\vec{y}$ is proportional to the product of the likelihood and the prior:

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

$p(\vec{y}|\vec{\beta}) \propto exp(-\frac{1}{2\sigma^2}||\vec{y} - X\vec{\beta}||_2^2)exp(-\frac{1}{2\tau}||\vec{\beta}||_2^2)$

$-log(p(\vec{y}|\vec{\beta})) = \frac{1}{2\sigma^2}||\vec{y} - X\vec{\beta}||_2^2 + \frac{1}{2\tau}||\vec{\beta}||_2^2$

The ridge regression objective function is the negative log posterior. Therefore, minimizing the ridge regression objective function is equivalent to maximizing the posterior distribution, which means we want to find the mode of the posterior distribution. By the properties of the Gaussian distribution, the mode and mean of the posterior distribution are the same, so the ridge regression estimate is also the mean of the posterior distribution.

From the above, we could see that $\hat{\vec{\beta}} = (X'X + \frac{\sigma^2}{\tau}I)^{-1}X'\vec{y}$, therefore, $\lambda = \frac{\sigma^2}{\tau}$.

ex3.19

As we know, $\hat{\beta}^{ridge} = (X'X + \lambda I)^{-1}X'\vec{y}$, it could also be written as $\frac{X'\vec{y}}{X'X+\lambda I}$; therefore, we could find that as λ decrease, $\hat{\beta}^{ridge}$ will increase, which means $\lambda \rightarrow 0$, $\hat{\beta}^{ridge}$ will increase.

For lasso, $\hat{\beta}^{lasso} = argmin_{\beta}\frac{1}{2n}||\vec{y} - X\vec{\beta}||_2^2 + \alpha||\beta||_1$, where α is the tuning parameter, the size of the updates to the coefficients depends on the value of α, with larger values of α leading to larger updates, we could see that when α decrease, $\hat{\beta}^{lasso}$ will also decrease. So, the same property does not hold for lasso.

ex3.28

suppose we have a copy of X named X^*, the new predictor would be $X^N = [X, X^*]$.

then the new coefficient would be $\beta^N = (\beta', \beta^{*\prime})$, $X^N\beta^N = X\beta + X^*\beta^* = X(\beta+\beta^*)$

$L = ||\vec{y} - X^N\beta^N||^2 + \lambda(\sum_{j=1}^p |\beta_j| + \sum_{j=1}^p |\beta_j^*|) = ||\vec{y} - X(\beta+\beta^*)||^2 + \lambda(\sum_{j=1}^p |\beta_j + \beta_j^*|) - \lambda(\sum_{j=1}^p |\beta_j + \beta_j^*|) + \lambda(\sum_{j=1}^p |\beta_j| + \sum_{j=1}^p |\beta_j^*|)$

the first two term is the same problem for the lossa, and the last two term would be non negative since |a+b|≤ |a|+|b|

By 3.52, we could know that the solution of the first two term would be $\beta + \beta^* = \hat{\beta}$

So, $\beta_j + \beta_j^* = \hat{\beta}_j = \alpha$

and in the same time, we want the last two term to be zero, so β and β^* should be both positive or negative.

ex3.30

suppose we have a new $\bar{X} = (X, \gamma I)'$ and $\bar{y} = (y, 0)'$ where I is identity matrix and $\gamma$is a constant number

$||\bar{y} - \bar{X}\beta||_2^2 = ||(y - X\beta, -\gamma\beta)||_2^2 = ||y - X\beta||_2^2 + \gamma^2||\beta||_2^2$

$\hat{\beta} = argmin_\beta(||\bar{y} - \bar{X}\beta||_2^2 + \bar{\lambda}||\beta||_1) = argmin_\beta(||y - X\beta||_2^2 + \gamma^2||\beta||_2^2 + \bar{\lambda}||\beta||_1)$

from 3.91, we know that we want $min_\beta||y - X\beta||^2 + \lambda(\alpha||\beta||_2^2 + (1-\alpha)||\beta||_1)$

we could see that$\alpha\lambda = \gamma^2, (1-\alpha)\lambda = \bar{\lambda}$

so $\gamma = \sqrt{\alpha\lambda}, \bar{\lambda} = (1-\alpha)\lambda$