Homework 3

Xining Yuan

10/30/2020

```
library(bis557homework3)
```

1. This is the first question of the homework 3

Suppose $X = U\Sigma V^T$, then We have that $X^TX = V\Sigma^2V^T$

Suppose $\Sigma^2 = I$, then $X^TX = VIV^T$

We also know that $X^TDX = V\Sigma^TU^TDU\Sigma V^T$, i.e. $X^TDX = VU^TD(VU)^T$

In this condition, we know that whether the condition number is good or not, it depends on D.

Therefore, let $X = \begin{bmatrix} 0.96 & -0.28 \\ 0.28 & 0.96 \end{bmatrix}$ and $\beta = \begin{bmatrix} 280 \\ 950 \end{bmatrix}$.

Based on β and X we have above, we can get that $p_1 = \frac{1}{2}$ and $p_2 = 0.999$.

Therefore, $D = \begin{bmatrix} p_1(1-p_1) & 0 \\ 0 & p_2(1-p_2) \end{bmatrix} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.000999 \end{bmatrix}$

Based on all matrices we have, we can calculate condition numbers of X^TX and X^TDX.

The condition number of X^TX is 1, which is good.

The condition number of X^TDX is 354.164, which is bad.

2. This is the second question of the homework 3

```
my_glm_gd <-
  function(X, y, mu_fun, gamma, maxit=25, tol=1e-10) {
    beta <- rep(0,ncol(X))
    for(j in seq_len(maxit)) {
      b_old <- beta
      eta <- X %*% beta
      mu <- mu_fun(eta)
      delta <- t(X) %*% (y - mu)
      beta <- beta + gamma * delta
    }
    beta
  }
```

```
my_glm_nag <-
  function(X, y, mu_fun, gamma, maxit=25, tol=1e-10) {
    alpha <- 0.5
    beta <- rep(0,ncol(X))
    v <- rep(0,ncol(X))
```

```r
    for(j in seq_len(maxit)) {
      b_old <- beta
      eta <- X %*% (beta - alpha * v)
      mu <- mu_fun(eta)
      delta <- t(X) %*% (y - mu)
      v <- alpha * v - gamma * delta
      beta <- beta - v
    }
    beta
  }
```

```r
n <- 5000; p <- 3
beta <- c(-1, 0.2, 0.1)
X <- cbind(1, matrix(rnorm(n * (p- 1)), ncol = p - 1))
eta <- X %*% beta
lambda <- exp(eta)
y <- rpois(n, lambda = lambda)
beta_hat_gd <- my_glm_gd(X, y, mu_fun = function(eta) exp(eta), gamma=0.0005, maxit=200)
beta_hat_nag <- my_glm_nag(X, y, mu_fun = function(eta) exp(eta), gamma=0.0005)
beta_glm <- coef(glm(y ~ X[,-1], family = "poisson"))
cbind(beta, beta_hat_gd, beta_hat_nag , as.numeric(beta_glm))
```

```
##      beta
## [1,] -1.0 -0.9634515 -0.9634504 -0.9634515
## [2,]  0.2  0.1863403  0.1863414  0.1863403
## [3,]  0.1  0.1462146  0.1462152  0.1462146
```

Description:

For the GLM maximum likelihood, I use $L(y) = \sum_{i=1}^{n} X_i^t \beta \cdot y_i - A(X_i^t \beta) + \log(h(y_i))$

We want to get that $\hat{\beta} = \arg\max L(y, \beta)$. Therefore, we use $\nabla L(\beta) = X^T(y - \mu(X^T\beta))$, where $\mu = Ey_i$

According to the gradient descent, we have that $\beta_{i+1} = \beta_i + \gamma \nabla L(\beta)$.

Based on those formulas, we can build up the my__glm__gd function shown above.

Then I choose to use Nesterov as a standard adaptive of step size.

We have $v_t = \alpha v_{t-1} - \gamma \nabla_\theta L(\beta - \alpha v_{t-1})$ and $\beta = \beta - v_t$

Then according to this formula, we build up the my__glm__nag function shown above.

Explanation:

I build up the random data putting in to my__glm__gd function and my__glm__nag function to do the comparison. The comparison among beta, beta__hat__gd and beta__hat__nag, we can get they are similar. Therefore, it means that both of methods work.

3. This is the third question of the homework 3

```r
my_glm_nr_logistic <-
  function(X, y, maxit=25L, tol=1e-10)
  {
    beta <- rep(0,ncol(X))
    for(j in seq(1L, maxit)) {
```

```
      b_old <- beta
      p <- 1 / (1 + exp(- X %*% beta))
      W <- as.numeric(p * (1 - p))
      XtX <- crossprod(X, diag(W) %*% X)
      score <- t(X) %*% (y - p)
      delta <- solve(XtX, score)
      beta <- beta + delta
      if(sqrt(crossprod(beta - b_old)) < tol) break
    }
    beta
  }
```

```
library(palmerpenguins)
library(usethis)
library(missForest)
```

```
## Loading required package: randomForest

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

## Loading required package: foreach

## Loading required package: itertools

## Loading required package: iterators
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
penguinsi <- penguins %>%
  as.data.frame()%>%
  missForest() %>%
  `$`(ximp)%>%
  as_tibble()
```

```
##   missForest iteration 1 in progress...done!
##   missForest iteration 2 in progress...done!
##   missForest iteration 3 in progress...done!
```

```
library(usethis)
library(testthat)
```

```
##
## Attaching package: 'testthat'
```

```
## The following object is masked from 'package:dplyr':
##
##     matches
```

```
K = 3
P = 6

library(dplyr)
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.0.3
## ------------------------------------------------------------------------------
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
## ------------------------------------------------------------------------------
##
## Attaching package: 'plyr'
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(ramify)
```

```
## Warning: package 'ramify' was built under R version 4.0.3
##
## Attaching package: 'ramify'
## The following object is masked from 'package:graphics':
##
##     clip
```

```
d <- penguinsi %>%
    select(species, island, bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g, sex) %>%
    mutate(sex = if_else(sex == "male", 1, 0))
d$species <- revalue(d$species, c(Adelie=0, Gentoo=1, Chinstrap=2))
d$island <- revalue(d$island, c(Torgersen=0, Biscoe=1, Dream=2))

n_train <- 300
n_test <- 44
X_train <- d[1:n_train,2:7]
y_train <- d[1:n_train,1]
X_test <- d[(n_train+1):344,2:7]
y_test <- d[(n_train+1):344,1]

X_train <- data.matrix(X_train)
y_train <- data.matrix(y_train)
X_test <- data.matrix(X_test)
y_test <- data.matrix(y_test)
# coefficients of all K models
beta_hat <- matrix(rep(0, len=K*(P+1)), nrow=P+1)
# training coefficients with training data
```

```r
X_k <- cbind(1, X_train)
for (k in seq(1L, K))
{
  y_k <- as.numeric(y_train == k)
  beta_tmp <- my_glm_nr_logistic(X_k, y_k)
  beta_hat[1:(P+1),k] <- beta_tmp[1:(P+1)]
}
# predicting
prob <- matrix(rep(0, len=n_test*K), nrow=n_test)
X_k <- cbind(1, X_test)
for (k in seq(1L, K))
{
  prob_tmp <- 1 / ( 1 + exp(-X_k %*% beta_hat[1:(P+1),k]))
  prob[1:n_test,k] <- prob_tmp[1:n_test]
}
y_pred <- argmax(prob, rows = TRUE)

bias <- abs(y_test-y_pred)
bias1 <- sum(bias)


expect_lt(bias1, 5)
```

Description:

Use one-vs-all approach fits K binary models, one for each class. In each time, we set the class k as one. The rest classes are coded as zero. Then we train the regression model and get the coefficients β_k. During the testing, we run the prediction models k times. Each time, we predict with β_k and get the probability that testing sample belongs to class k. For the final result, we find the class that the testing sample has the largest probability.