Homework 2 on Newton's methods

Xinyi Lin xl2836

Due: 03/13/2019, Wednesday, by 1pm

```r
library(matrixcalc)
```

Problem 1

Design an optmization algorithm to find the minimum of the continuously differentiable function $f(x) = -e^{-1}\sin(x)$ on the closed interval $[0, 1.5]$. Write out your algorithm and implement it into **R**.

Answer:

We use Golden section search algorithm to find minimum of this function on the closed interval $[0, 1.5]$.

```r
fx = function(x){
  return(-sin(x)/exp(1))
}
```

```r
a=0
b=1.5
w = 0.618
theta0 = a+(b-a)*(1-w)
theta1 = theta0 + (b-a)*(1-w)*w
tol = 1e-10

rlist = c("a", "b", "theta0", "theta1")
while(abs(b-a)>tol){

if(fx(theta1) < fx(theta0)){
  a=theta0;
  theta0 = theta1
  theta1 = theta0 + (b-a)*(1-w)*w
}
  else{
    b=theta1;
    theta0 = a+(b-a)*(1-w)
    theta1 = theta0 + (b-a)*(1-w)*w
  }

  rlist = rbind(rlist, c(a, b, theta0, theta1))
}

tail(rlist)
```

```
##  [,1]               [,2]  [,3]               [,4]
##  "1.49999999904592" "1.5" "1.49999999941038" "1.49999999963561"
##  "1.49999999941038" "1.5" "1.49999999963561" "1.49999999977481"
##  "1.49999999963561" "1.5" "1.49999999977481" "1.49999999986083"
```

```
## "1.49999999977481" "1.5" "1.49999999986083" "1.49999999991399"
## "1.49999999986083" "1.5" "1.49999999991399" "1.49999999994685"
## "1.49999999991399" "1.5" "1.49999999994685" "1.49999999996715"
```

According to the result we get from R process, we can know that the minmum of function $f(x) = -e^{-1}\sin(x)$ on the closed interval $[0, 1.5]$ is approximately equals to -0.3669579.

Problem 2

The Poisson distribution is often used to model "count" data — e.g., the number of events in a given time period.
The Poisson regression model states that

$$Y_i \sim \text{Poisson}(\lambda_i),$$

where

$$\log \lambda_i = \alpha + \beta x_i$$

for some explanatory variable x_i. The question is how to estimate α and β given a set of independent data $(x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)$.

1. Modify the Newton-Raphson function from the class notes to include a step-halving step.
2. Further modify this function to ensure that the direction of the step is an ascent direction. (If it is not, the program should take appropriate action.)
3. Write code to apply the resulting modified Newton-Raphson function to compute maximum likelihood estimates for α and β in the Poisson regression setting.

The Poisson distribution is given by

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

for $\lambda > 0$.

Answer:

Question 1

```r
logisticstuff <- function(dat, betavec) {
  u <- betavec[1] + betavec[2] * dat$x
  expu <- exp(u)
  loglik <- sum(dat$y * u - log(1 + expu))
# Log-likelihood at betavec
  p <- expu / (1 + expu)
# P(Y_i=1|x_i)
  grad <- c(sum(dat$y - p), sum(dat$x * (dat$y - p)))
# gradient at betavec
  Hess <- -matrix(c(sum(p * (1 - p)),
                    rep(sum(dat$x * p * (1-p)),2),
                   sum(dat$x^2 * p * (1 - p))), ncol=2)
# Hessian at betavec
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}
```

```r
# generate some data
set.seed(123)
n <- 40
truebeta <- c(1, -2)
x <- rnorm(n)
expu <- exp(truebeta[1] + truebeta[2] * x)
y <- runif(n) < expu / (1 + expu)
```

Modified Newton-Raphson algorithm with step halving function is as follow:

```r
NewtonRaphson <- function(dat, func, start, tol=1e-10,
                               maxiter = 200) {
  i <- 0
  cur <- start
  stuff <- func(dat, cur)
  res <- c(0, stuff$loglik, cur)
   prevloglik <- -Inf      # To make sure it iterates
  while(i < maxiter && abs(stuff$loglik - prevloglik) > tol)
 {
    i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    lambda = 1
    #cur <- prev - lambda*solve(stuff$Hess) %*% stuff$grad
    #stuff <- func(dat, cur)        # log-lik, gradient, Hessian
    while(prevloglik >= stuff$loglik){
      cur <- prev - lambda*solve(stuff$Hess) %*% stuff$grad
      stuff <- func(dat, cur)        # log-lik, gradient, Hessian
      lambda = lambda/2
    }
    res <- rbind(res, c(i, stuff$loglik, cur))
    # Add current values to results matrix
}
  return(res)
}

ans1 <- NewtonRaphson(list(x=x,y=y),logisticstuff,c(1,-2))
ans1
```

```
##     [,1]      [,2]      [,3]      [,4]
## res    0 -19.85680 1.0000000 -2.000000
##        1 -19.73914 0.8046966 -1.706660
##        2 -19.73692 0.8253681 -1.741493
##        3 -19.73692 0.8256830 -1.742070
##        4 -19.73692 0.8256831 -1.742070
```

Question 2

As shown in lecture, Hessian matrix of logistic regression is negative definite. But if it is not, we need to replace it with a negative definite matrix that is similar to $\nabla^2 f(\theta_{i-1})$, like$\nabla^2 f(\theta_{i-1}) - \gamma I$.

Modified Newton-Raphson algorithm is as follow:

```r
NewtonRaphson2 <- function(dat, func, start, tol=1e-10,
                               maxiter = 200) {
```

```r
  i <- 0
  cur <- start
  stuff <- func(dat, cur)
  len = length(start)
  res <- c(0, stuff$loglik, cur)
  prevloglik <- -Inf      # To make sure it iterates
  while(i < maxiter && abs(stuff$loglik - prevloglik) > tol)
 {
    i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    lambda = 1
    #cur <- prev - lambda*solve(stuff$Hess) %*% stuff$grad
    #stuff <- func(dat, cur)         # log-lik, gradient, Hessian
    while(prevloglik >= stuff$loglik){
      gamma = 0
      mod_Hess = stuff$Hess - gamma*diag(len)
      while (is.negative.definite(mod_Hess) == FALSE) {
        gamma = gamma+1
        mod_Hess = stuff$Hess - gamma*diag(len)
      }
      cur <- prev - lambda*solve(mod_Hess) %*% stuff$grad
      stuff <- func(dat, cur)         # log-lik, gradient, Hessian
      lambda = lambda/2
    }
    res <- rbind(res, c(i, stuff$loglik, cur))
    # Add current values to results matrix
}
  return(res)
}

ans2 <- NewtonRaphson2(list(x=x,y=y),logisticstuff,c(1,-2))
ans2
```

```
##     [,1]      [,2]      [,3]      [,4]
## res    0 -19.85680 1.0000000 -2.000000
##        1 -19.73914 0.8046966 -1.706660
##        2 -19.73692 0.8253681 -1.741493
##        3 -19.73692 0.8256830 -1.742070
##        4 -19.73692 0.8256831 -1.742070
```

Question 3

As

$$Y_i \sim Poisson(\lambda_i)$$

where

$$\log \lambda_i = \alpha + \beta x_i$$

and

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

For data (x_1, Y_1), (x_2, Y_2), (x_3, Y_3),..., (x_n, Y_n), the likelihood function is given by

$$L(\alpha, \beta; x) = \prod_{i=1}^{n} [\frac{\lambda_i^{y_i} e^{-\lambda_i}}{y_i!}]$$

and the log likelihood function is given by

$$l(\alpha, \beta; x) = \sum_{i=1}^{n} [y_i \log \lambda_i - \lambda_i - log(y_i!)] = \sum_{i=1}^{n} [y_i(\alpha + \beta x_i) - e^{\alpha + \beta x_i} - log(y_i!)]$$

So the gradient of this function is

$$\nabla f(\alpha, \beta) = \begin{pmatrix} \sum_{i=1}^{n} y_i - e^{\alpha+\beta x_i} \\ \sum_{i=1}^{n} x_i \times (y_i - e^{\alpha+\beta x_i}) \end{pmatrix}$$

And the Hessian matrix of this function is

$$\nabla^2 f(\alpha, \beta) = -\begin{pmatrix} \sum_{i=1}^{n} e^{\alpha+\beta x_i} & \sum_{i=1}^{n} x_i \times e^{\alpha+\beta x_i} \\ \sum_{i=1}^{n} x_i \times e^{\alpha+\beta x_i} & \sum_{i=1}^{n} x_i^2 \times e^{\alpha+\beta x_i} \end{pmatrix}$$

Calculate loglikelihood, gradient and Hessian matrix of Y by using following function.

```r
posslogstuff <- function(dat, betavec) {
  u <- betavec[1] + betavec[2] * dat$x
  expu <- exp(u)
  # Log-likelihood at betavec
  loglik <- sum(dat$y * u - expu - log(gamma(y+1)))
  # gradient at betavec
  grad <- c(sum(dat$y - expu), sum(dat$x * (dat$y - expu)))
  Hess <- -matrix(c(sum(expu),
                   rep(sum(dat$x * expu),2),
                  sum(dat$x^2 * expu)), ncol=2)
# Hessian at betavec
  return(list(loglik = loglik, grad = grad, Hess = Hess))
}
```

Generate data and test function:

```r
set.seed(123)
n <- 40
truebeta <- c(1, -2)
x <- rnorm(n)
expu <- exp(truebeta[1] + truebeta[2] * x)
y = vector(mode = "numeric", 40)
for (i in 1:40) {
  y[i] = rpois(1,expu[i])
}
ans3 <- NewtonRaphson2(list(x=x,y=y),posslogstuff,c(1,-2))
ans3
```

```
##     [,1]      [,2]      [,3]      [,4]
## res    0 -74.00364 1.0000000 -2.000000
##        1 -73.02032 0.9274011 -2.084497
##        2 -73.01907 0.9264554 -2.083507
##        3 -73.01907 0.9264553 -2.083505
##        4 -73.01907 0.9264553 -2.083505
```

Estimated $\alpha = 0.296, \beta = -2.084$ is very closed to Ture value($\alpha = 1, \beta = -2$).

problem 3

Consider the ABO blood type data, where you have $N_{\text{obs}} = (N_A, N_B, N_O, N_{AB}) = (26, 27, 42, 7)$.

- design an EM algorithm to estimate the allele frequencies, P_A, P_B and P_O; and
- Implement your algorithms in R, and present your results..

Answer: your answer starts here…

```
#R codes:
```