question 2

xinyi Lin

2/28/2019

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(matrixcalc)
```

```
cancer_data = read_csv("./breast-cancer-1.csv")
```

```
## Warning: Missing column names filled in: 'X33' [33]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   id = col_integer(),
##   diagnosis = col_character(),
##   X33 = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 569 parsing failures.
## row # A tibble: 5 x 5 col     row col   expected   actual     file                    expected   <int
## ... ................. ... ........................................................... ........ .....
## See problems(...) for more details.
```

classical Newton Raphson

```
logisticstuff <- function(x, y, betavec) {
  u <- x %*% betavec
  expu <- exp(u)
  loglik = vector(mode = "numeric", 569)
  for(i in 1:569)
    loglik[i] = y[i]*u[i] - log(1 + expu[i])
  loglik_value = sum(loglik)
  # Log-likelihood at betavec
  p <- expu / (1 + expu)
  # P(Y_i=1|x_i)
  grad = vector(mode = "numeric", 13)
```

```
  #grad[1] = sum(y - p)
  for(i in 1:13)
    grad[i] = sum(t(x[,i])%*%(y - p))
  #Hess <- -t(x)%*%p%*%t(1-p)%*%x
  Hess = hess_cal(x, p)
  return(list(loglik = loglik_value, grad = grad, Hess = Hess))
}
```

```
hess_cal = function(x,p){
  len = length(p)
  hess = matrix(0, ncol(x), ncol(x))
  for (i in 1:len) {
    x_t = t(x[i,])
    unit = t(x_t)%*%x_t*p[i]*(1-p[i])
    #unit = t(x[i,])%*%x[i,]*p[i]*(1-p[i])
    hess = hess + unit
  }
  return(-hess)
}
```

Newton-Raphson process

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

Using data to get answer

Variables we used: mean radius, mean texture, mean perimeter, mean smoothness,mean compactness, mean concavity, mean concave points, mean fractal dimension, mean symmetry, standard error of radius, perimeter, and symmetry.

```
intercept = rep(1, 569)
central = function(x){
  x = (x-mean(x))/sd(x)
  return(x)
}
x = cancer_data %>%
```

```
  dplyr::select(radius_mean:fractal_dimension_mean, radius_se, perimeter_worst, symmetry_worst) %>%
  dplyr::select(-area_mean) %>%
  apply(2, central) %>%
  cbind(intercept, .) %>%
  as.matrix()
#colnames(x) = NULL
y = as.vector(ifelse(cancer_data$diagnosis=="M",1,0))  # response variables
beta = rep(0.001,13)
ans1 = NewtonRaphson(x, y, logisticstuff, beta)
ans1
```

```
##     [,1]        [,2]      [,3]      [,4]       [,5]      [,6]        [,7]
## res    0  0.00100000  0.001000 0.0010000   0.001000 0.0010000  0.00100000
##        1 -0.50964380  5.147502 0.3226138  -5.843089 0.1406309  0.03548446
##        2 -0.60601139  6.335673 0.5835811  -7.714058 0.2901484 -0.15319664
##        3 -0.55866759  7.237710 0.8874514  -9.659794 0.5265803 -0.42718660
##        4 -0.37848818  8.878794 1.1989997 -12.562417 0.8220036 -0.75893316
##        5 -0.05485482 10.407713 1.5220255 -15.558813 1.0914868 -1.31063637
##        6  0.28006582 13.174284 1.8707722 -20.007399 1.3153466 -1.92358397
##        7  0.46701881 16.324378 2.1329513 -24.333715 1.4690367 -2.29314642
##        8  0.51210991 17.433986 2.2147109 -25.774287 1.5161136 -2.39330462
##        9  0.51461258 17.508973 2.2200802 -25.869490 1.5190832 -2.39935133
##       10  0.51462157 17.509265 2.2201010 -25.869858 1.5190939 -2.39937354
##       11  0.51462157 17.509265 2.2201010 -25.869858 1.5190939 -2.39937354
##          [,8]      [,9]      [,10]       [,11]       [,12]     [,13]
## res 0.0010000 0.0010000  0.0010000  0.00100000 0.001000000  0.001000
##     0.2101853 0.9089571 -0.1567598 -0.09712642 0.002966365  1.021052
##     0.4422269 1.4662525 -0.2322546 -0.21072539 0.007516439  2.231985
##     0.7342960 1.8673460 -0.3091729 -0.33732476 0.126226228  4.105397
##     1.0238375 2.3016066 -0.4173747 -0.43523793 0.437832941  6.395949
##     1.2080870 3.0666062 -0.5701351 -0.40947301 0.914325507  9.055346
##     1.2831384 4.0425448 -0.7010318 -0.25156970 1.309641478 11.961355
##     1.2880069 4.7265840 -0.7526548 -0.10312549 1.518338833 13.955242
##     1.2800853 4.9247466 -0.7563426 -0.05136739 1.570744151 14.531279
##     1.2790946 4.9374292 -0.7560138 -0.04767842 1.573794274 14.567568
##     1.2790888 4.9374789 -0.7560109 -0.04766296 1.573805521 14.567707
##     1.2790888 4.9374789 -0.7560109 -0.04766296 1.573805522 14.567707
##         [,14]
## res 0.0010000
##     0.3985837
##     0.6060434
##     0.7693548
##     0.9607607
##     1.2204323
##     1.4844603
##     1.6505765
##     1.6924346
##     1.6946372
##     1.6946440
##     1.6946440
```

```
glm_x = x[,2:13]
cancer_model <- glm(y ~ glm_x, family = binomial(link = "logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(cancer_model)
```

```
##
## Call:
## glm(formula = y ~ glm_x, family = binomial(link = "logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.93772  -0.04566  -0.00439   0.00019   2.95456
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                    0.51462    0.45946   1.120  0.26269
## glm_xradius_mean              17.50927   15.50880   1.129  0.25890
## glm_xtexture_mean              2.22010    0.49540   4.481 7.42e-06 ***
## glm_xperimeter_mean          -25.86986   17.19151  -1.505  0.13237
## glm_xsmoothness_mean           1.51909    0.85720   1.772  0.07637 .
## glm_xcompactness_mean         -2.39937    1.47304  -1.629  0.10334
## glm_xconcavity_mean            1.27909    0.98943   1.293  0.19610
## glm_xconcave points_mean       4.93748    1.93550   2.551  0.01074 *
## glm_xsymmetry_mean            -0.75601    0.67359  -1.122  0.26171
## glm_xfractal_dimension_mean   -0.04766    0.92303  -0.052  0.95882
## glm_xradius_se                 1.57381    0.68948   2.283  0.02245 *
## glm_xperimeter_worst          14.56771    3.46115   4.209 2.57e-05 ***
## glm_xsymmetry_worst            1.69464    0.64415   2.631  0.00852 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 751.44  on 568  degrees of freedom
## Residual deviance:  74.57  on 556  degrees of freedom
## AIC: 100.57
##
## Number of Fisher Scoring iterations: 10
```

modified Hessian

```r
modified <- function(x, y, logisticstuff, start, tol=1e-5, maxiter = 200){
  i <- 0
  cur <- start
  beta_len <- length(start)
  stuff <- logisticstuff(x, y, cur)
  res = c(0, cur)
  #res <- c(0, stuff$loglik,cur)
  prevloglik <- -Inf # To make sure it iterates
  while(i <= maxiter && abs(stuff$loglik - prevloglik) > tol)
  #while(i <= maxiter && abs(stuff$loglik - prevloglik) > tol && stuff$loglik > -Inf)
    { i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    lambda = 0
```

```
    while (is.negative.definite(stuff$Hess-lambda*diag(beta_len)) == FALSE) {
      lambda = lambda + 1
    }
    cur <- prev - solve(stuff$Hess-lambda*diag(beta_len)) %*% stuff$grad
    #cur <- prev + (diag(beta_len)/10)%*%(stuff$grad)
    #cur = prev + t(stuff$grad)%*%(stuff$grad)
    stuff <- logisticstuff(x, y, cur) # log-lik, gradient, Hessian
    res = rbind(res, c(i, cur))
    #res <- rbind(res, c(i, stuff$loglik, cur))
    }
  return(round(res,2))
}
#ans2 <- modified(x, y, logisticstuff, beta, maxiter = 1000)
#ans2
```