Aula 13: Otimização numérica - Aspectos Computacionais

Prof. Dr. Eder Angelo Milani

19/06/2023

A seguir é apresentado algumas funções interessantes no R que podem ajudar na difícil tarefa de otimização de funções.

Funções hessian e grad no R

O método de Newton-Raphson, que além do gradiente, usa o hessiano e é apropriado para calcular estimadores de máxima verossimilhança por meio de uma aproximação quadrática da verossimilhança ao redor de valores iniciais dos parâmetros, apresenta a dificuldade dos cálculos teóricos.

Uma alternativa é a obtenção dos valores necessários por meio de aproximações numéricas. Veja o exemplo a seguir.

Exemplo 1

Voltando ao Exemplo 2 da Aula 10, foi gerado uma amostra da distribuição Weibull(α, γ), na sequência foi obtido as funções de verossimilhança, log-verossimilhança, vetor escore e matriz hessiana.

Podemos utilizar funções no R que calculam, de forma aproximada, o vetor escore e a matriz hessiana. Portanto, facilitando o processo como um todo.

```
set.seed(2023)
n <- 1000
alpha <- 2
gama <- 0.5

x<- rweibull(n, shape =gama , scale = alpha)

summary(x)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
##   0.0000   0.1802   1.1010   3.9462   4.1123 125.5789
```

```
# o valor esperado e
alpha*gamma(1+1/gama)
```

```
## [1] 4
```

```
# construir a funcao objeto

log_vero <- function(p0){
  alpha <- p0[1]
  gama <- p0[2]
  n <- length(x)
  aux <- n*log(gama) - n*gama*log(alpha) + (gama-1)*sum(log(x)) - sum((x/alpha)^gama)
  return(aux)
}

# teste da funcao
log_vero(p0=c(1,0.5))
```

```
## [1] -1938.947
```

```
# pacote a ser carregado
library(numDeriv)

# o vetor escore
grad(log_vero, x=c(1,0.5))
```

```
## [1]  217.3866 -545.4308
```

```
# a matriz hessian
hessian(log_vero, x=c(1,0.5))
```

```
##          [,1]       [,2]
## [1,] -576.080   1518.516
## [2,] 1518.516 -10737.908
```

```
# metodo de newton-raphson

n.max <- 100
theta <- matrix(NA, nrow=2, ncol=n.max)
dif_ <- 1
epsilon <- 0.001
cont <- 1
theta[, 1] <- c(1, 2)

while(abs(dif_)>= epsilon & cont<=n.max){
  cat("cont+1=", cont+1, "\n")
  theta[ ,cont+1] <- theta[ ,cont] - solve(hessian(log_vero, theta[ ,cont]))%*%grad(log_vero, theta[ ,co
  cat("theta[", cont+1, "]=", theta[ ,cont+1], "\n")
  dif_ <- sqrt(sum((theta[ ,cont+1]-theta[ ,cont])^2))
  cat("dif=", dif_, "\n")
  cont <- cont+1
}
```

```
## cont+1= 2
## theta[ 2 ]= 0.8781681 1.677699
## dif= 0.3445592
```

```
## cont+1= 3
## theta[ 3 ]= 0.7362294 1.334386
## dif= 0.3714973
## cont+1= 4
## theta[ 4 ]= 0.5547017 0.9573204
## dif= 0.4184863
## cont+1= 5
## theta[ 5 ]= 0.3563464 0.5738512
## dif= 0.4317331
## cont+1= 6
## theta[ 6 ]= 0.3312675 0.381048
## dif= 0.1944274
## cont+1= 7
## theta[ 7 ]= 0.5641077 0.4181372
## dif= 0.2357757
## cont+1= 8
## theta[ 8 ]= 0.9305036 0.4671185
## dif= 0.3696554
## cont+1= 9
## theta[ 9 ]= 1.378278 0.501388
## dif= 0.4490838
## cont+1= 10
## theta[ 10 ]= 1.790903 0.5153879
## dif= 0.412862
## cont+1= 11
## theta[ 11 ]= 2.041855 0.5178614
## dif= 0.2509642
## cont+1= 12
## theta[ 12 ]= 2.112259 0.5178338
## dif= 0.07040396
## cont+1= 13
## theta[ 13 ]= 2.116738 0.5178167
## dif= 0.004479602
## cont+1= 14
## theta[ 14 ]= 2.116755 0.5178166
## dif= 1.699994e-05
```

```r
cont
```

```
## [1] 14
```

```r
theta[, 1:cont]
```

```
##      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,]    1 0.8781681 0.7362294 0.5547017 0.3563464 0.3312675 0.5641077 0.9305036
## [2,]    2 1.6776988 1.3343860 0.9573204 0.5738512 0.3810480 0.4181372 0.4671185
##          [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## [1,] 1.378278 1.7909026 2.0418546 2.1122585 2.1167381 2.1167551
## [2,] 0.501388 0.5153879 0.5178614 0.5178338 0.5178167 0.5178166
```

Compare os resultados obtidos aqui com os resultados da Aula 10.

Funções optim e MaxLik

Para obter máximos ou mínimos de funções pode-se usar a função `optim()` do pacote `stats`, que inclui vários métodos de otimização, dentre os quais destacamos:

i) o método default que é o Método Simplex de Nelder e Mead (1965), e corresponde a um método de busca não gradiente,

ii) o método BFGS,

iii) o método do gradiente conjugado (conjugate gradient - CG),

iv) o método L-BFGS-B, que permite restrições limitadas,

v) o método SANN, que é método não gradiente e pode ser encarado como uma variante do método da têmpera simulada (simulated annealing).

Outra opção é o pacote `optimization`, que também inclui vários métodos, como o método Nelder-Mead e da têmpera simulada. O pacote `maxLik` também é uma opção, contendo quase todas as funções do `stats`, além do algoritmo de `Newton-Rapson` (função `maxNR()`). Esse pacote é apropriado para a maximização de verossimilhanças, daí o rótulo maxLik.

Exemplo 2

Considere a função $f(x, y) = \exp(-(x + y))$, que tem um máximo no ponto (0, 0) e valor máximo igual a 1. A seguir é apresentado a função `maxNR()` para obter o ponto de máximo.

```
#install.packages("maxLik")
library("maxLik")
```

```
## Warning: package 'maxLik' was built under R version 4.2.3
```

```
## Carregando pacotes exigidos: miscTools
```

```
## Warning: package 'miscTools' was built under R version 4.2.3
```

```
##
## Please cite the 'maxLik' package as:
## Henningsen, Arne and Toomet, Ott (2011). maxLik: A package for maximum likelihood estimation in R. C
##
## If you have questions, suggestions, or comments regarding the 'maxLik' package, please use a forum o
## https://r-forge.r-project.org/projects/maxlik/
```

```
##
## Attaching package: 'maxLik'
```

```
## The following object is masked from 'package:numDeriv':
##
##     hessian
```

```r
f1 <- function(theta){
x <- theta[1]
y <- theta[2]
z <- exp(-(x^2+y^2))
return(z)
}

result <- maxNR(f1, start=c(1,1))

print(summary(result))
```

```
## --------------------------------------------
## Newton-Raphson maximisation
## Number of iterations: 3
## Return code: 1
## gradient close to zero (gradtol)
## Function value: 1
## Estimates:
##           estimate gradient
## [1,] -7.221252e-12        0
## [2,] -7.221475e-12        0
## --------------------------------------------
```

Exemplo 3

Uma função comumente usada como teste em otimização é a função de Rosenbrock, ou função banana

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2,$$

Essa função tem um mínimo global em $(1, 1)$. Os resultados da aplicação da função `optim()` para determinar o mínimo dessa função, usando diferentes métodos são apresentados a seguir.

```r
f1 <- function(theta){
x <- theta[1]
y <- theta[2]
#cat("x=", x, "y=", y, "\n")
z <- (1-x)^2 + 100*(y-x^2)^2
return(z)
}

chute <- c(-1.2, 1)

# BFGS

optim(chute, f1, method="BFGS")
```

```
## $par
## [1] 0.9998044 0.9996084
##
## $value
## [1] 3.827383e-08
```

```
##
## $counts
## function gradient
##      118       38
##
## $convergence
## [1] 0
##
## $message
## NULL
```

```r
# L-BFGS-B

optim(chute, f1, method="L-BFGS-B")
```

```
## $par
## [1] 0.9998000 0.9996001
##
## $value
## [1] 3.998487e-08
##
## $counts
## function gradient
##       49       49
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

```r
# CG

optim(chute, f1, method="CG")
```

```
## $par
## [1] -0.7648079  0.5927148
##
## $value
## [1] 3.106475
##
## $counts
## function gradient
##      402      101
##
## $convergence
## [1] 1
##
## $message
## NULL
```

```r
# Nelder-Mead

optim(chute, f1, method="Nelder-Mead")
```

```
## $par
## [1] 1.000260 1.000506
##
## $value
## [1] 8.825241e-08
##
## $counts
## function gradient
##      195       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Exercícios:

(Entregar na sala de aula) (1) Gerar 100 valores da distribuição normal($\mu = 3, \sigma^2 = 4$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha as estimativas de máxima verossimilhança dos parâmetros μ e σ^2. Obtenha também a estimativa adotando a função *optim* e compare os resultados.

(2) Gerar 100 valores da distribuição Poisson($\lambda = 3$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha a estimativa de máxima verossimilhança do parâmetro λ. Obtenha também a estimativa adotando a função *optim* e compare os resultados.

(3) Gerar 300 valores da distribuição Bernoulli($p = 0,3$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha a estimativa de máxima verossimilhança do parâmetro p. Obtenha também a estimativa adotando a função *optim* e compare os resultados.

(4) Gerar 100 valores da distribuição Exp($\lambda = 0.3$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha a estimativa de máxima verossimilhança do parâmetro λ. Obtenha também a estimativa adotando a função *optim* e compare os resultados.

(5) Gerar 100 valores da distribuição gama($\alpha = 10, \beta = 10$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha as estimativas de máxima verossimilhança dos parâmetros α e β. Obtenha também a estimativa adotando a função *optim* e compare os resultados.

(6) Gerar 100 valores da distribuição beta($\alpha = 10, \beta = 10$). Utilizando o método de Newton-Rapson, como feito anteriormente, obtenha as estimativas de máxima verossimilhança dos parâmetros α e β. Obtenha também a estimativa adotando a função *optim* e compare os resultados.