Lista 2

Técnicas Computacionais em Estatística

Tailine J. S. Nonato

April 13, 2025

Lista 2 - Otimização e máxima verossimilhança

Exercício 1

Seja T uma variável aleatória seguindo uma distribuição Birnbaum-Saunders. Então, T é definido por:

$$T = \beta \left[\frac{\alpha}{2} Z + \sqrt{\left[\frac{\alpha}{2}\right]^2 Z + 1} \right]^2,$$

em que $\alpha > 0$ e $\beta > 0$ são parâmetros de forma e escala, respectivamente, e Z é uma variável aleatória com distribuição normal padrão. Temos a notação $T \sim BS(\alpha, \beta)$. O parâmetro β é também um parâmetro de localização, pois ele é a mediana da distribuição de T.

Note que se $T \sim BS(\alpha, \beta)$, então:

$$Z = \frac{1}{\alpha} \left[\sqrt{\frac{T}{\beta}} - \sqrt{\frac{\beta}{T}} \right] \sim N(0, 1).$$

A função densidade de probabilidade de T é dada por:

$$f_T(t) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2\alpha^2} \left[\frac{t}{\beta} + \frac{\beta}{t} - 2 \right] \right) \frac{t^{-\frac{3}{2}}[t+\beta]}{2\alpha\sqrt{\beta}}, \quad t > 0.$$

Uma forma alternativa de expressar a densidade dada na equação anterior é:

$$f_T(t) = \phi(a_t) A_t, \quad t > 0, \alpha > 0, \beta > 0,$$

em que A_t é a derivada de a_t com respeito a t, e:

$$a_t(\alpha,\beta) = a_t = \frac{1}{\alpha}\left[\sqrt{\frac{t}{\beta}} - \sqrt{\frac{\beta}{t}}\right],$$

que é expresso como:

$$A_t = \frac{d}{dt}a_t = \frac{t^{-\frac{3}{2}}[t+\beta]}{2\alpha\sqrt{\beta}}.$$

Seja $T_1, T_2, \ldots, T_n$ uma amostra aleatória de tamanho n de $T \sim BS(\alpha,\beta)$, e seja $t_1, t_2, \ldots, t_n$ as correspondentes observações. A função de log-verossimilhança para $\theta = (\alpha,\beta)^\top$ é dada por:

$$\ell(\theta) = c_1 + \frac{n}{\alpha^2} - \frac{1}{2\alpha^2}\sum_{i=1}^{n}\left(\frac{t_i}{\beta} + \frac{\beta}{t_i}\right) - n\log(\alpha) - \frac{n}{2}\log(\beta) + \sum_{i=1}^{n}\log(t_i+\beta),$$

em que c_1 é uma constante que não depende de θ.

Para maximizar a função de log-verossimilhança $\ell(\theta) = \ell(\alpha,\beta)$, precisamos das primeiras derivadas em relação a α e β, formando o vetor escore definido por $\dot{\ell} = (\dot{\ell}_\alpha, \dot{\ell}_\beta)^\top$, cujos elementos são dados por:

$$\dot{\ell}_\alpha = \frac{\partial\ell(\alpha,\beta)}{\partial\alpha} = -\frac{2n}{\alpha^3} + \frac{1}{\alpha^3}\sum_{i=1}^{n}\left(\frac{t_i}{\beta} + \frac{\beta}{t_i}\right) - \frac{n}{\alpha},$$

$$\dot{\ell}_\beta = \frac{\partial\ell(\alpha,\beta)}{\partial\beta} = \frac{1}{2\alpha^2}\sum_{i=1}^{n}\left(\frac{t_i}{\beta^2} - \frac{1}{t_i}\right) - \frac{n}{2\beta} + \sum_{i=1}^{n}\frac{1}{t_i+\beta}.$$

A matriz Hessiana é dada por:

$$\ddot{\ell} = \left(\frac{\partial^2\ell(\theta)}{\partial\theta_i\partial\theta_j}\right) = \begin{pmatrix} \ddot{\ell}_{\alpha\alpha} & \ddot{\ell}_{\alpha\beta} \\ \ddot{\ell}_{\beta\alpha} & \ddot{\ell}_{\beta\beta} \end{pmatrix}, \quad i,j = 1,2,$$

em que:

$$\ddot{\ell}_{\alpha\alpha} = \frac{\partial^2\ell(\alpha,\beta)}{\partial\alpha^2} = \frac{6n}{\alpha^4} - \frac{3}{\alpha^4}\sum_{i=1}^{n}\left(\frac{t_i}{\beta} + \frac{\beta}{t_i}\right) + \frac{n}{\alpha^2},$$

$$\ddot{\ell}_{\alpha\beta} = \ddot{\ell}_{\beta\alpha} = \frac{\partial^2\ell(\alpha,\beta)}{\partial\alpha\partial\beta} = \frac{1}{\alpha^3}\sum_{i=1}^{n}\left(\frac{1}{t_i} - \frac{t_i}{\beta^2}\right),$$

$$\ddot{\ell}_{\beta\beta} = \frac{\partial^2 \ell(\alpha, \beta)}{\partial \beta^2} = -\frac{1}{\alpha^2 \beta^3} \sum_{i=1}^{n} t_i + \frac{n}{2\beta^2} - \sum_{i=1}^{n} \frac{1}{(t_i + \beta)^2}.$$

Gere uma amostra simulada de tamanho $n = 100$ da distribuição Birnbaum-Saunders com $\alpha = 0.5$ e $\beta = 2.0$. Estime os parâmetros α e β através do método da máxima verossimilhança (usando o método de Newton) baseado na amostra simulada (pode usar como valores iniciais $\alpha_0 = 0.1$ e $\beta_0 = 1.0$).

Resolução

Conforme descrito no enunciado, a ideia é seguir os passos abaixo:

1. Gerar a amostra aleatória de tamanho $n = 100$ da distribuição Birnbaum-Saunders com $\alpha = 0.5$ e $\beta = 2.0$.
2. Definir a função de log-verossimilhança $\ell(\theta)$.
3. Definir as derivadas da função de log-verossimilhança em relação a α e β.
4. Definir a matriz Hessiana.
5. Definir a função de Newton-Raphson para encontrar os parâmetros α e β.

```r
# dados definidos
n <- 100
alpha <- 0.5
beta <- 2

set.seed(345)

# 1. gerar amostra aleatória

rs_BS <- function(n, alpha, beta) {
  Z <- rnorm(n)
  T <- beta * ((alpha / 2) * Z + sqrt((alpha / 2)^2 * Z + 1))^2
  return(T)}

t <- rs_BS(n, alpha, beta)

# 2. função de log-verossimilhança
loglikelihood <- function(alpha, beta, t) {
  c1 <- 0
  resultado <- c1 + (n / alpha^2) - (1 / (2 * alpha^2)) *
```

```r
  sum((t / beta) + (beta / t)) -
  n * log(alpha) - (n / 2) * log(beta) + sum(log(t + beta))
  return(resultado)}

# 3. derivadas da função de log-verossimilhança
d_loglikelihood <- function(alpha, beta, t) {
  d_alpha <- (-2 * n / alpha^3) + (1 / alpha^3) *
  sum((t / beta) + (beta / t)) - (n / alpha)
  d_beta <- (1 / (2 * alpha^2)) * sum((t / beta^2) - (1 / t)) -
  (n / (2 * beta)) + sum(1 / (t + beta))
  return(c(d_alpha, d_beta))}

# 4. definir a matriz Hessiana
hessiana <- function(alpha, beta, t) {
  h_alpha <- (6 * n / alpha^4) - (3 / alpha^4) *
  sum((t / beta) + (beta / t)) + (n / alpha^2)
  h_alpha_beta <- (1 / alpha^3) * sum((1 / t) -
  (t / beta^2))
  h_beta <- (-1 / (alpha^2 * beta^3)) * sum(t) +
  (n / (2 * beta^2)) - sum(1 / ((t + beta)^2))
  return(matrix(c(h_alpha, h_alpha_beta,
   h_alpha_beta, h_beta), nrow = 2))}

# 5. função de Newton-Raphson
my_newton <- function(t, alpha0, beta0, tol, max.iter) {
  x <- c(alpha0, beta0)
  iterations <- 0
  made.changes <- TRUE

  while (made.changes & (iterations < max.iter)) {
    x.old <- x
    iterations <- iterations + 1
    made.changes <- FALSE

    # gradiente e hessiana nas iterações atuais
    grad <- d_loglikelihood(x[1], x[2], t)
    hess <- hessiana(x[1], x[2], t)

    # passo de Newton: x - H^{-1} * grad
    x.new <- x - solve(hess, grad)
```

```
    # mudança relativa (norma euclidiana relativa)
    relative.change <- sqrt(sum((x.new - x.old)^2)) /
    sqrt(sum(x.old^2))
    made.changes <- (relative.change > tol)

    x <- x.new
  }

  if (made.changes) {
    warning("Newton's method terminated before convergence")
  }

  return(list(value = x,
              logLik = loglikelihood(x[1], x[2], t),
              iterations = iterations,
              converged = !made.changes))
}

# resultado
result <- my_newton(t, alpha0 = 0.1, beta0 = 1.0,
                    tol = 1e-3, max.iter = 50)
```

Parâmetros estimados:

alpha = 0.6566975

beta = 1.691785

Log-verossimilhança = 93.88825

de iterações = 11

Convergência = TRUE