STATISTICA 1, metodi matematici e statistici

Introduzione al linguaggio R

Esercitazione 7: 3-05-2004

Andrea Tancredi

Università di Roma “La Sapienza”, Rome, Italy

andrea.tancredi@uniroma1.it
http://3w.eco.uniroma1.it/utenti/tancredi

Approssimazione normale della verosimiglianza

Se la verosimiglianza $L(\theta)$ ammette un punto di massimo $\hat{\theta}$ interno a θ possiamo sviluppare $\log L(\theta)$ con la formula di Taylor intorno a $\hat{\theta}$:

$$\log L(\theta) \approx \log L(\hat{\theta}) + \frac{1}{2}(\theta - \hat{\theta})^2 \left[\frac{d^2 \log L(\hat{\theta})}{d\theta^2} \right]_{\theta=\hat{\theta}}$$

da cui ricaviamo che

$$L(\theta) \approx c \cdot \exp \left\{ -\frac{1}{2}(\theta - \hat{\theta})^2 j(\hat{\theta}) \right\}$$

dove $j(\hat{\theta}) = - \left[\frac{d^2 \log L(\hat{\theta})}{d\theta^2} \right]_{\theta=\hat{\theta}}$ viene detta informazione (di Fisher) osservata.

La funzione di verosimiglianza è quindi approssimativamente proporzionale ad una densità del tipo $\mathcal{N}\left(\hat{\theta}, j(\hat{\theta})^{-1}\right)$

R offre la possibilità di calcolare derivate simboliche di funzioni attraverso i comandi `D,deriv,deriv3`. La fuzione che vogliamo derivare può essere passata a questi comandi attraverso oggetti di tipo `expression`. Cerchiamo di capire cosa sono questi oggetti per R

```
> a <- expression(2 + 2)
> a

expression(2 + 2)

> mode(a)

[1] "expression"

> eval(a)

[1] 4
```

con l'ultimo comando diciamo ad R di eseguire il contenuto dell'esspressione che abbiamo chiamato `a`.

Vediamo ora cosa succede se il contenuto di expression ùn oggetto che ancora non abbiamo definito

```
> b <- expression(w + 1)
```

proviamo a dare il comando `eval(b)`.

Consideriamo ora i seguenti esempi

```
> ex1 <- expression(1 + 0:9)
> ex1

expression(1 + 0:9)

> eval(ex1)

 [1]  1  2  3  4  5  6  7  8  9 10
```

```
> length(ex3 <- expression("carla", "paolo", 1 + 0:9))
[1] 3
> eval(ex3)
 [1]  1  2  3  4  5  6  7  8  9 10
> eval(ex3[1])
[1] "carla"
> eval(ex3[3])
 [1]  1  2  3  4  5  6  7  8  9 10
```

Notiamo che possiamo mettere più oggetti nella nostra espressione, ma che attraverso `eval` vediamo solo il risultato realtivo all'ultimo oggetto.

Vediamo allora come ottenere la derivata di x^2. Definiamo prima la seguente espressione

```
> f <- expression(x^2)
> x <- 5
> eval(f)

[1] 25
```

Osserviamo che `f` non è una funzione e che se `x` nonè stato definito non possiamo valuatare il valore di x^2 (provate a eseguire f(1) e poi a cancellare x e poi a rivalutare f).

La sintassi di `deriv` è

```
 deriv(expr, namevec, function.arg, tag = ".expr", hessian =
FALSE)
```

In particolare `expr` vuole un espressione contenente la funzione che vogliamo derivare mentre `namevec` richiede una stringa contenente la variabile rispetto a cui vogliamo effettuare la derivata.

```
> f1 <- deriv(f, "x")
> f1

expression({
    .value <- x^2
    .grad <- array(0, c(length(.value), 1), list(NULL, c("x")))
    .grad[, "x"] <- 2 * x
    attr(.value, "gradient") <- .grad
    .value
})
```

`f1` è quindi un espressione e come tale può essere valutata.

```
> x <- 1:5
> eval(f1)

[1]  1  4  9 16 25
attr(,"gradient")
      x
[1,]  2
[2,]  4
[3,]  6
[4,]  8
[5,] 10

> .grad

      x
[1,]  2
[2,]  4
[3,]  6
[4,]  8
[5,] 10
```

```
> y <- rpois(20, 3)
> n <- length(y)
> s <- sum(y)
> hat.theta <- mean(y)
> LP <- expression(log(exp(-theta * n) * theta^s))
> theta <- hat.theta
> LP1 <- deriv(LP, "theta", hessian = T)
> eval(LP1)

[1] 12.79888
attr(,"gradient")
     theta
[1,]     0
attr(,"hessian")
, , theta

          theta
[1,] -6.060606
```

La derivata seconda della logverosimiglianza cambiata di segno (informazione osservata) calcolata nel punto di massima verosimiglianza è quindi

```
> i.o <- -.hessian
```

Vediamo allora se è vero che possiamo approssimare la verosimiglianza del nostro campione estratto da una Poisson con la densità di una normale con media la s.m.v e varianza l'inverso dell'informazione osservata calcolata sempre nella s.m.v.

```
> m <- dnorm(hat.theta, hat.theta, sd = (sqrt(-.hessian))^(-1))
> curve(dnorm(x, hat.theta, sd = (sqrt(-.hessian))^(-1))/m,
+   from = 0, to = 6, col = 2)
> LP2 <- function(theta) exp(-theta * n) * theta^s
> curve(LP2(x)/LP2(hat.theta), from = 0, to = 6, add = T, col =
```

Test di Neyman e Pearson

Dato il seguente campione

```
> y <- c(0.4, 1.76, 0.65, 0.52, 0.27, 0.53, 1.22,
+    1.25, 0.17, 1.61)
```

estratto da una v.a. esponenziale verificare l'ipotesi $H_0\ \theta = 1$ contro l'ipotesi alternativa $H_1 \theta = 2$ attraverso il test di Neyman e Pearson imponendo un errore di prima specie pari a 0.05 e calcolare la potenza del test.

Per prima cosa ci costruiamo una funzione che restituisca il rapporto delle verosimiglianze

```
> rapp.ver <- function(data, theta0, theta1) {
+     theta1^(length(data)) * exp(-theta1 * sum(data))/
+     theta0^(length(data)) * exp(-theta0 * sum(data))
+ }
```

e ci calcoliamo il valore di tale rapporto per il nostro campione

```
> t <- rapp.ver(data = y, theta0 = 1, theta1 = 2)
```

Per vedere se `t` cade nella zona di accettazione o in quella di rifiuto ci dobbiamo calcolare il quantile di livello 0.95 della distribuzione del test del rapporto delle verosimiglianze sotto l'hp nulla (ricordiamoci che valori elevati del rapp.ver portano a rifiutare H_0)

```
> n <- 10
> m <- 10000
> theta.vero = 1
> a <- rexp(n * m, theta.vero)
> campioni <- matrix(a, nrow = m)
> lrt <- apply(campioni, FUN = rapp.ver, MAR = 1,
+         theta0 = 1, theta1 = 2)
> lim <- quantile(lrt, 0.95)
```

Accettiamo l'ipotesi se il nostro valore `t` è più piccolo di `lim`

```
> if (t < lim) print(c("accettiamo H_0"))

[1] "accettiamo H_0"
```

Per calcolarci la potenza del test dobbiamo calcolare la probabilita di avere valori maggiori di `lim` sotto l'ipotesi alternativa

```
> n <- 10
> m <- 10000
> theta.vero = 2
> a <- rexp(n * m, theta.vero)
> campioni <- matrix(a, nrow = m)
> lrt2 <- apply(campioni, FUN = rapp.ver, MAR = 1, theta0 = 1,
+     theta1 = 2)
> pot <- sum(lrt2 > lim)/m
```

Il seguente grafico ci aiuta a capire meglio la situazione

```
> plot(density(log(lrt)), col = 2, xlim = c(-70, 70),
+  ylim = c(0,0.1))
> lines(density(log(lrt2)), col = 3)
> abline(v = log(lim), lty = 2)
> abline(v = log(t), lty = 3)
```