Laboratorio di ST1 - Lezione 3

Antonietta di Salvatore

Dipartimento di Matematica
Universitá degli Studi Roma Tre

Outline

- funzione di verosomiglianza
- stima di massima verosimiglianza
- Proprietá degli stimatori

Funzione di verosimiglianza

Funzione di Verosomiglianza

Costruiamo la funzione di verosimiglianza per il modello normale

in input:

- mu : valore medio
- sigma2 : varianza
- campione : campione osservato

in output:

- verosimNormale

```
verosimNormale <- function(mu, sigma2, campione){
n <- length(campione)
fattore1 <- (2 * pi * sigma2)^  (-(n/2))
fattore2 <- exp(-1/(2*sigma2) * sum((campione - mu)^  2))
fattore1 * fattore2
}
```

creata la funzione, per usarla usiamo la seguente sintassi

```
verosimNormale(mu, sigma2, campione)
```

CASO 1: σ noto e μ incognito

caso 1: supponiamo che la varianza sia nota e pari a 12
campione di 10 osservazioni estratto da una variabile casuale normale

```
campioneOsservato <- c(28, 19, 30, 25, 27, 27, 28, 20, 26, 28)
```

per valutare la funzione di verosimiglianza su differenti valori del parametro é necessario vettorizzare la funzione usando la funzione `Vectorize`, si usano come argomenti di vettorizzazione sia mu che sigma2 in modo da utilizzare la funzione anche per gli altri casi (di media nota e varianza incognita e di media incognita e varianza incognita)

```
verosimNormaleVec <-
Vectorize(verosimNormale,vectorize.args=c('mu','sigma2'))
verosimNormaleVec(c(22,25), 12, campioneOsservato)
```

con i comandi

```
verosimNormale(22, 12, campioneOsservato)
verosimNormale(25, 12, campioneOsservato)
```

otteniamo gli stessi risultati

costruisco un vettore di valori per mu su cui valutare la funzione di verosimiglianza

```
ascisseMu <- seq(18, 30, by=0.1)
```

calcolo la funzione di verosimiglianza per il campione osservato ipotizzando di conoscere il valore di sigma2

```
ordinate <- verosimNormaleVec(ascisseMu, 12, campioneOsservato)
```

normalizzo la funzione di verosimiglianza

```
ordinate <- ordinate / max(ordinate)
```

e la rappresento graficamente

```
plot(ascisseMu, ordinate, type='l',xlab=expression(mu),
ylab=expression(paste('L(',mu,')')))
```

costruisco un secondo campione con la stessa media del campione precedente e con un numero di osservazioni pari a 50 ottenute replicando per 50 volte la media del campione osservato

```
campione2 <- rep(mean(campioneOsservato), 50)
```

calcolo e normalizzo la funzione di verosimiglianza relativa al secondo campione

```
ordinate2 <- verosimNormaleVec(ascisseMu, 12, campione2)
ordinate2 <- ordinate2 / max(ordinate2)
```

e la rappresento sul grafico usando una linea di colore rosso

```
lines(ascisseMu, ordinate2, col='red')
```

aggiungo una legenda al grafico

```
legend('topright',c('n = 4','n = 50'),col=c(1,2),pch=15)
```

CASO 2: μ noto e σ incognito

caso 2: supponiamo che la media sia nota e pari a 26
costruisco un vettore di valori per sigma2 su cui valutare la funzione di verosimiglianza

```
ascisseSig <- seq(1, 50, by=0.1)
```

calcolo e normalizzo la funzione di verosimiglianza sul campione osservato ipotizzando di conoscere il valore di mu

```
ordinate <- verosimNormaleVec(26, ascisseSig, campioneOsservato)
ordinate <- ordinate / max(ordinate)
```

e la rappresento graficamente

```
plot(ascisseSig, ordinate, type='l', xlab=expression(sigma),
ylab=expression(paste('L(',sigma,')')))
```

posso osservare che il picco del grafico si ha in corrispondenza del valore sigma=2

```
x=rep(12,length=ordinate)
lines(x,ordinate,type='l',col='red')
```

secondo campione di 50 osservazioni

```
 campione3 <- c(31.38608, 23.26010, 21.64751, 31.26172,
29.12355, 26.74486, 31.25434, 28.44542, 28.99208, 17.38735,
29.03265, 26.15551, 27.09402, 24.49145, 22.47440, 26.43422,
28.48345, 22.86583, 24.51925, 25.93945, 23.86060, 23.07029,
27.70992, 31.68761, 25.37425, 23.74364, 23.04960, 23.57411,
25.24512, 26.35674, 23.77628, 27.78201, 22.23127, 26.71575,
25.61334, 29.77525, 20.40043, 31.26144, 21.81445, 30.65199,
24.65500, 23.56402, 24.01116, 22.38173, 21.52463, 24.55481,
19.40000, 21.79867, 28.40000, 29.64880)
```

Calcoliamo la varianza. Possiamo usare il comando `var()` oppure consideriamo la seguente funzione

```
n=length(campione3)
m=mean(campione3)
sommascarti2=0
for (i in 1:n) {
x =campione3[i]
sommascarti2 = sommascarti2+ (x-m)^  2
}
varianza=sommascarti2/n

var(campione3)
varianza*50/49
```

il comando `var()` fornisce la varianza CORRETTA!!!

calcolo e normalizzo la funzione di verosimiglianza relativa al secondo campione di 50 osservazioni

```
ordinate2 <- verosimNormaleVec(26, ascisseSig, campione3)
ordinate2 <- ordinate2 / max(ordinate2)
```

e la rappresento sul grafico usando una linea di colore rosso

```
lines(ascisseSig, ordinate2, col='red')
```

aggiungo una legenda al grafico

```
 legend('topright', c('n = 4','n = 50'), col=c(1,2), pch=15)
```

CASO 3: μ e σ entrambi incogniti

se voglio passare in input due vettori, uno per il parametro mu e l'altro per sigma2, non posso usare la chiamata tradizionale (come risulta dal messaggio di warning prodotto)

```
verosimNormaleVec(ascisseMu, ascisseSig, campioneOsservato)
```

la funzione `outer` permette di costruire una griglia bidimensionale attraverso il prodotto cartesiano dei due vettori passati in input come primi due argomenti; il terzo argomento ad outer indica la funzione che deve essere valutata su ogni cella di tale griglia

```
ordinate3d <- outer(ascisseMu, ascisseSig,
verosimNormaleVec,campioneOsservato)
```

un plot 3d puó essere ottenuto usando la funzione `persp`

```
persp(ascisseMu, ascisseSig, ordinate3d)
```

la funzione `contour` permette di ottenere il grafico delle curve di livello

```
contour(ascisseMu, ascisseSig, ordinate3d)
```

Calcoliamo la funzione di verosomoglianza congiunta per μ e σ sulla base dei dati contenuti in `campione3`

```
ascisseSig = seq(5, 30, by=0.1)

ascisseMu = seq(22, 30, by=0.1)

ordinate3d = outer(ascisseMu, ascisseSig,
verosimNormaleVec,campione3)

persp(ascisseMu, ascisseSig, ordinate3d)

contour(ascisseMu, ascisseSig, ordinate3d)
```

Stima di massima verosimiglianza

La stima di massima verosimiglianza puó essere invece realizzata mediante il comando `fitdistr`, disponibile nella libreria MASS: il comando usa un algoritmo iterativo per la stima dei parametri e richiede l'inserimento di valori iniziali.

```
 library(MASS)
y.fit<-fitdistr(campione3,'normal')
y.fit
```

Selezioniamo gli errori standard delle stime trovate:

```
y.fit$sd
```

Facciamo un grafico

```
 hist(campione3, freq=F)
curve(dnorm(x, y.fit$estimate[1], y.fit$estimate[2]),add=TRUE)
curve(dnorm(x, 0, 1),col='red', add=TRUE)
```

Proprietá degli stimatori

Per una distribuzione normale, lo stimatore di massima verosimiglianza della media é corretto e consistente

```
n =10
M = c()
varM = c()
x=rnorm(10000,0,1)
for (i in 1:1000) {
x1 =x[1:n]
x1.fit = fitdistr(x1,'normal')
M=append(M,x1.fit$estimate[1])
varM=append(varM, x1.fit$estimate[2]/n)
n = n+10
}

plot(M)
plot(varM)
```

lo stimatore di massima verosimiglianza della varianza é asintoticamente corretto e consistente, la distorsione e la varianza diminuiscono all'aumentare della numerositá campionaria

```
n =10
var = c()
varc = c()
x=rnorm(10000,0,1)
for (i in 1:1000) {
x1 =x[1:n]
x1.fit = fitdistr(x1,'normal')
var=append(var,x1.fit$estimate[2])
vc= x1.fit$estimate[2]*n/(n-1)
varc=append(varc,vc)
n = n+10
}
```

Facciamo il grafico

```
 inf1=min(var)
inf2=min(varc)
inf=min(inf1, inf2)
sup1=max(var)
sup2=max(varc)
sup=max(sup1, sup2)
plot(var, ylim=c(inf,sup),type='l')
points(varc, col=2, type='l')
```

guardiamo la prima parte del grafico

```
 plot(var, ylim=c(inf,sup),type='l',xlim=c(0,100))
points(varc, col=2, type='l',xlim=c(0,100))
legend('topright',c('stimatore distorto','stimatore
corretto'),pch=c('o','o'),col=c(1,2))
```

verifica della correttezza dello stimatore `n =10`

```
var = c()
varc = c()
n=10
for (i in 1:100) {
x=rnorm(n,0,1)
x.fit = fitdistr(x,'normal')
var=append(var,x.fit$estimate[2])
vc= x.fit$estimate[2]*n/(n-1)
varc=append(varc,vc)
}
mean(var)
mean(varc)
```

calcoliamo la distorsione

```
 1-mean(var)
1-mean(varc)
```