Il modello di regressione logistica: Stima iterativa con metodo di Fisher scoring – Applicazione in R

- Definiamo in R i dati x e y su cui stimare i parametri della retta di regressione.

```
x<-c(8,14,-7,6,5,6,-5,1,0,-17)
y<-c(1,1,0,0,1,0,1,0,0,0)
n<-length(y)
```

- Costruiamo la matrice del modello lineare.

```
X<-matrix(c(rep(1,n),x),ncol=2,byrow=FALSE)
X
      [,1] [,2]
 [1,]    1    8
 [2,]    1   14
 [3,]    1   -7
 [4,]    1    6
 [5,]    1    5
 [6,]    1    6
 [7,]    1   -5
 [8,]    1    1
 [9,]    1    0
[10,]    1  -17
```

- Identifichiamo con j il numero di variabili x considerate ($j = 1$; una sola variabile indipendente x) e scriviamo un vettore $(j + 1)$ di partenza (*initial trial*) per i parametri ignoti:

$$\boldsymbol{\varphi}^{(t=0)} = \begin{bmatrix}\beta_0 \\ \beta_1\end{bmatrix} = \begin{bmatrix}0 \\ 0\end{bmatrix}$$

- Alla partenza dell'algoritmo di ottimizzazione, Le probabilità previste dal modello non lineare logistico saranno così tutte uguali a 0.5,

$$p(y_i = 1|x_i, \beta_j) = \frac{exp\{0 + \beta 0\}}{1 + exp\{0 + 0x_i\}} = \frac{1}{2}$$

```
# Numero di variabili indipendenti (colonne matrice X - 1)
j=1
# Stime iniziali dei parametri ignoti b0 e b1
phit0<-matrix(c(0,0))
```

- Scriviamo il vettore $\boldsymbol{u}$ delle derivate prime, di dimensione $(j + 1)\times 1$, formato dalla $Eq(54)$,

$$\boldsymbol{u} = \mathbf{X}'(\boldsymbol{y} - \boldsymbol{p})$$

```
B<-phit0[1:(j+1)]
p<-exp(X%*%B)/(1+exp(X%*%B))
u<-t(X)%*%(matrix(y)-p)

u
     [,1]
[1,] -1.0
[2,] 16.5
```

- Scriviamo la matrice Hessiana delle derivate seconde, nella quale inseriamo i risultati della $Eq(55)$

$$E(\boldsymbol{H}) = \mathbf{X}'\mathbf{W}\mathbf{X},$$

- Dove $\mathbf{W}$ denota una matrice diagonale $n\times n$ con elementi:

$$p(y_i = 1|x_i, \beta_i)[1 - p(y_i = 1|x_i, \beta_i)] = \frac{exp\{\alpha + \beta x_i\}}{1 + exp\{\alpha + \beta x_i\}}\left[1 - \frac{exp\{\alpha + \beta x_i\}}{1 + exp\{\alpha + \beta x_i\}}\right]$$

sulla diagonale principale,

```
W<-diag( c(-p*(1-p)) )
H<-t(X)%*%W%*%X

W
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
 [1,] -0.25  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [2,]  0.00 -0.25  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [3,]  0.00  0.00 -0.25  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 [4,]  0.00  0.00  0.00 -0.25  0.00  0.00  0.00  0.00  0.00  0.00
 [5,]  0.00  0.00  0.00  0.00 -0.25  0.00  0.00  0.00  0.00  0.00
 [6,]  0.00  0.00  0.00  0.00  0.00 -0.25  0.00  0.00  0.00  0.00
 [7,]  0.00  0.00  0.00  0.00  0.00  0.00 -0.25  0.00  0.00  0.00
 [8,]  0.00  0.00  0.00  0.00  0.00  0.00  0.00 -0.25  0.00  0.00
 [9,]  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00 -0.25  0.00
[10,]  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00 -0.25

H
      [,1]    [,2]
[1,] -2.50   -2.75
[2,] -2.75 -180.25
```

- Aggiorniamo le stime iniziali con una prima iterazione Fisher scoring:

$$\boldsymbol{\varphi}^{(t=1)} = \boldsymbol{\varphi}^{(t=0)} - \left(E\left(\boldsymbol{H}^{(t=0)}\right)\right)^{-1}\boldsymbol{u}^{(t=0)}$$

```
phit1<- phit0 - solve(H)%*%u
phit1
            [,1]
[1,] -0.50923967
[2,]  0.09930879
```

Automatizziamo in R:

```
j=1                     # Numero di variabili indipendenti (colonne matrice X - 1)
i=0                     # Numero progressivo di iterazioni
phit0<-matrix(c(0,0))   # Stime iniziali dei parametri ignoti b0 e b1
n<-length(y)            # Grandezza del campione
FisherScoring<-c()      # data set vuoto per i risultati:
```

Notiamo che, un vettore di condizioni logiche TRUE/FALSE può essere verificato contemporaneamente mediante il comando `all()`, il cui complemento è `any()`:

```
c(1,2,10)>5
[1] FALSE FALSE  TRUE
all(c(1,2,10)>5)        # Tutte le condizioni VERE?
[1] FALSE
any(c(1,2,10)>5)        # Ce n'è qualcuna VERA?
[1] TRUE
```

- Questo ci sarà utile per confrontare <u>contemporaneamente</u> la condizione:

$$\boldsymbol{\varphi}^{(t+1)} - \boldsymbol{\varphi}^{(t)} < .00001$$

```
phit1
            [,1]
[1,] -0.50923967
[2,]  0.09930879

phit0
     [,1]
[1,]    0
[2,]    0

phit1-phit0
            [,1]
[1,] -0.50923967
[2,]  0.09930879

abs(phit1-phit0) < .00001
      [,1]
[1,] FALSE
[2,] FALSE

all(abs(phit1-phit0) < .00001)
[1] FALSE

###### Preamboli
j=1                     # Numero di variabili indipendenti (colonne matrice X - 1)
i=0                     # Numero progressivo di iterazioni
phit0<-matrix(c(0,0))   # Stime iniziali dei parametri ignoti b0 e b1
n<-length(y)            # Grandezza del campione
FisherScoring<-c()      # data set vuoto per i risultati:

###### Fisher Scoring
repeat{

          i=i+1 # prima iterazione (la prima sarà nulla, quindi registreremo i-1)

          B<-phit0[1:(j+1)]
          p<-exp(X%*%B)/(1+exp(X%*%B))
          u<-t(X)%*%(matrix(y)-p)

          W<-diag( c(-p*(1-p)) )
          H<-t(X)%*%W%*%X

          phit1<- phit0 - solve(H)%*%u

          #registriamo i risultati nel data set FisherScoring:
          Result_iter<-c(phit1[1:(j+1)])
          FisherScoring<-cbind(FisherScoring, Result_iter)

          #condizione logica per interrompere il ciclo:
          if( all(abs(phit1 - phit0) < .00001) ){
                              rownames(FisherScoring)<-c(paste("b",seq(0,j)))
                              colnames(FisherScoring)<-seq(0,(i-1))
                              print(FisherScoring)
                              break
                                        }
          phit0<-phit1
}
```

Risultato: <u>4 iterazioni</u> :

```
              0          1          2          3          4
b 0 -0.50923967 -0.6860552 -0.7215888 -0.7227523 -0.7227534
b 1  0.09930879  0.1328327  0.1394152  0.1396279  0.1396281
```

- R dispone di diverse funzioni di ottimizzazione numerica, che non necessitano cioè di vettori di derivate prime e matrici Hessiane. La convergenza generalmente richiede molti cicli e sono utili nel caso di funzioni molto complesse e non derivabili.

- Iniziamo scrivendo la funzione di verosimiglianza, <u>che moltiplicheremo per -1, dal momento che questi algoritmi ricercano un minimo</u> (e non un massimo come nel nostro caso!): infatti log(1) = Max log-Likelihood = 0; log(0) = Min log-Likelihood = -Inf; moltiplicando per -1 rovesciamo la funzione che da concava diventa convessa, con minimo a 0 e massimi a +Inf.

```
x<-c(8,14,-7,6,5,6,-5,1,0,-17)
y<-c(1,1,0,0,1,0,1,0,0,0)
n<-length(y)
j=1 # Numero di variabili indipendenti

#Log Likelihood function con argomento «phi» = vettore di parametri ignoti.
# Versione 1: Eq. 48
Log_Lik<-function(phi){
                        B<-phi[1:(j+1)]
                        ##p<-exp(X%*%B)/(1+exp(X%*%B)## Non necessario...
                        score=sum( y*X%*%B - log(1+exp(X%*%B)) )
                        (-1)*score # algoritmi di minimizzazione!
                       }
```

- La funzione che useremo `optim()` richiede tre argomenti: *par* = vettore iniziale di parametri, *fn* = la funzione da minimizzare, *method* = il metodo usato.

```
phit0<-c(0,0) #vettore di stime iniziali

optim(par=phit0, fn=Log_Lik, method="Nelder-Mead")
optim(par=phit0, fn=Log_Lik, method="BFGS")
optim(par=phit0, fn=Log_Lik, method="CG")
optim(par=phit0, fn=Log_Lik, method="L-BFGS-B")
optim(par=phit0, fn=Log_Lik, method="SANN")
```

Ad esempio:

```
optim(par=phit0, fn=Log_Lik, method="L-BFGS-B")

$par
[1] -0.7227566  0.1396287     # Parametri stimati

$value                        # Valore minimo della funzione Log_Lik
[1] 5.669806

$counts
function gradient
      14       14             # Numero di iterazioni (> di Fisher scoring!)


optim(par=phit0, fn=Log_Lik, method="CG")

$par
[1] -0.7227491  0.1396278

$value
[1] 5.669806

$counts
function gradient
     381       83
```

Il modello lineare generalizzato - GLM

- La stima per massima verosimiglianza del modello di regressione logistica si può ottenere in R mediante la funzione `glm()`

```
summary(                                        # per ottenere il sommario
        glm(y~x,family=binomial(link=logit))    # funzione glm() per dati binomiali
       )

Call:
glm(formula = y ~ x, family = binomial(link = logit))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.2266  -0.9287  -0.4383   0.9403   1.8095

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -0.7228     0.8133  -0.889    0.374
x             0.1396     0.1155   1.209    0.227

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 13.46  on 9  degrees of freedom
Residual deviance: 11.34  on 8  degrees of freedom
AIC: 15.34

Number of Fisher Scoring iterations: 4
```

- Vediamo di «ritrovare» i risultati all'interno della nostra procedura iterativa di stima:

A) Coefficienti B:

```
               0          1          2          3          4
b 0 -0.50923967 -0.6860552 -0.7215888 -0.7227523 -0.7227534
b 1  0.09930879  0.1328327  0.1394152  0.1396279  0.1396281
```

B) Errori standard dei coefficienti di regressione.

- Negativo dell'inversa della matrice Hessiana, aggiornata con la stima finale MLE.

```
# È noto che le stime delle varianze mediante il metodo MLE siano distorte, poiché
non corrette per i gradi di libertà:

-1*solve(H)
##             [,1]        [,2]
## [1,]  0.66149431 -0.04421518   # Varianza distorta di (Intercetta) B_0
## [2,] -0.04421518  0.01333360   # Varianza distorta di B_1

vcov.B<-(-1*solve(H))             # var/cov (distorte) dei coefficienti B
diag(vcov.B)                      # diagonale delle var (distorte) coefficienti B
[1] 0.6614943 0.0133336
sqrt(diag(vcov.B))                # errori standard dei coefficienti B
[1] 0.8133230 0.1154712
```

C) «Deviance» → -2*(Funzione Log_Lik alla convergenza)

```
phit1                 # Stime MLE alla convergenza dell'algoritmo di ottimizzazione
           [,1]
[1,] -0.7227534
[2,]  0.1396281
(-1)*Log_Lik(phit1)   # Log_lik veniva moltiplicata per -1 per usare optim()!!
[1] -5.669806
-2*Log_Lik(phit1)
[1] 11.33961          # Approssima una distribuzione Chi-quadrato!!
```

Il modello lineare generalizzato – Diagnosi sui residui

D) «Deviance residuals».

Il sommario che si trova in cima all'output si basa sui valori (addendi) della funzione di log-verosimiglianza (*Eq. 48*), moltiplicati per (-2) come nel calcolo della Deviance e opportunamente trasformati (radice quadrata):

$$-2\times\sum_{i}^{n}[y_i\mathbf{XB} - ln(1+exp\{\mathbf{XB}\})] = \sum_{i}^{n}-2[y_i\mathbf{XB} - ln(1+exp\{\mathbf{XB}\})] \approx \sum_{i}^{n} z^2 \approx \chi^2_{n-(j+1)}$$

```
sum(-2*( y*X%*%B - log(1+exp(X%*%B)) ))      # La sommatoria = Residual deviance
[1] 11.33961

-2*(y*X%*%B - log(1+exp(X%*%B)))             # Gli addendi, distribuiti come z^2

            [,1]
 [1,] 1.03063661
 [2,] 0.51191426
 [3,] 0.33552645
 [4,] 1.50461486
 [5,] 1.41105851
 [6,] 1.50461486
 [7,] 3.27442194
 [8,] 0.88700072
 [9,] 0.79138762
[10,] 0.08843653

# TRASFORMAZIONE:
# (y-p) # residui grezzi(1,0) - probabilità logistiche
# sign() resituisce il segno algebrico dei residui, veniva perso con il quadrato.
d_res<-sign(y-p)*sqrt(  -2*(y*X%*%B - log(1+exp(X%*%B)))  )
summary(d_res)
                  V1
 Min.    :-1.22663
 1st Qu.:-0.92876
 Median :-0.43831
 Mean    :-0.04332
 3rd Qu.: 0.94027
 Max.    : 1.80954

residuals(glm(y~x,family=binomial),type="deviance")
         1          2          3          4          5          6
 1.0152025  0.7154813 -0.5792457 -1.2266275  1.1878799 -1.2266275
         7          8          9         10
 1.8095374 -0.9418068 -0.8895993 -0.2973821
```

- In generale, il motivo per cui potremmo essere interessati a questa sintesi è verificare quanto bene il nostro modello si adatti ai dati.

- I residui sono le differenze tra ciò che osserviamo e ciò che il nostro modello prevede. Sarebbe auspicabile che i residui fossero distribuiti in modo uniforme, vorremmo che i valori del primo e del terzo quantile, nonché quelli minimo e massimo, fossero all'incirca uguali in valore assoluto, e che la mediana fosse vicina a 0.

- Inoltre, vorremmo che i valori minimo e massimo fossero < 3, in valore assoluto. Questo perché i residui di devianza possono essere approssimati grossolanamente con una distribuzione normale standard quando il modello è valido (Agresti, 2002). I residui superiori al valore assoluto di 3 si trovano nelle code di una distribuzione normale standard e di solito indicano una distorsione nel modello.

Il modello lineare generalizzato - Diagnosi sui residui

- Da un modello di *regressione logistica* binaria è possibile calcolare diversi residui. Il più elementare è il residuo grezzo, che abbiamo chiamato ε_i. Si tratta semplicemente della differenza tra il valore osservato (0-1) e la *probabilità prevista* p(y=1|X,B). Il suo valore varia tra -1 e 1.

$$\varepsilon_i = y_i - p(y_i = 1|x_i, \beta_j) = \begin{cases} 1 - p(y_i = 1|x_i, \beta_j) & \text{quando } y_i = 1 \\ 0 - p(y_i = 1|x_i, \beta_j) & \text{quando } y_i = 0 \end{cases}$$

con distribuzione di Bernoulli:

$$P\left(\varepsilon_i = 1 - p(y_i = 1|x_i, \beta_j)\right) = \frac{exp\{\alpha + \beta x_i\}}{1 + exp\{\alpha + \beta x_i\}};$$

$$P\left(\varepsilon_i = 0 - p(y_i = 1|x_i, \beta_j)\right) = 1 - \frac{exp\{\alpha + \beta x_i\}}{1 + exp\{\alpha + \beta x_i\}}.$$

```
residuals(glm(y~x,family=binomial),type="response")
          1           2           3           4           5           6           7
 0.40268944  0.22582453 -0.15444564 -0.52872217  0.50615289 -0.52872217  0.80547854
          8           9          10
-0.35821378 -0.32678695 -0.04325469
# manualmente
y-p
             [,1]
 [1,]  0.40268956
 [2,]  0.22582484
 [3,] -0.15444598
 [4,] -0.52872214
 [5,]  0.50615287
 [6,] -0.52872214
 [7,]  0.80547819
 [8,] -0.35821400
 [9,] -0.32678720
[10,] -0.04325489
```

- Possiamo standardizzare gli errori (Bernoulliani) grezzi mediante i risultati delle equazioni Eq.42 e 43, ottenendo i residui di Pearson:

$$z_{\varepsilon_i} = \frac{\varepsilon_i - E(\varepsilon_i)}{\sqrt{Var(\varepsilon_i)}} = \frac{\varepsilon_i - 0}{\sqrt{p_i(1 - p_i)}}$$

```
# Residui di Pearson
((y-p)-0)/sqrt(p*(1-p))
            [,1]
 [1,]  0.8210794
 [2,]  0.5400901
 [3,] -0.4273834
 [4,] -1.0591933
 [5,]  1.0123824
 [6,] -1.0591933
 [7,]  2.0348984
 [8,] -0.7470956
 [9,] -0.6967169
[10,] -0.2126275
# manualmente
residuals(glm(y~x,family=binomial),type="pearson")
         1          2          3          4          5          6          7
 0.8210791  0.5400896 -0.4273828 -1.0591934  1.0123824 -1.0591934  2.0349007
         8          9         10
-0.7470952 -0.6967165 -0.2126270
```

Proviamo a stimare un modello più complesso: Per casa!

X1	X2	Y
8	2	1
14	14	1
-7	6	0
6	4	0
5	4	1
6	4	0
-5	-7	1
1	2	0
0	10	0
-17	-19	0

A partire dai dati tabulati:

❖ Stimare i coefficienti del modello di regressione logistica mediante algoritmo di Fisher scoring e replicare i risultati con la funzione glm().

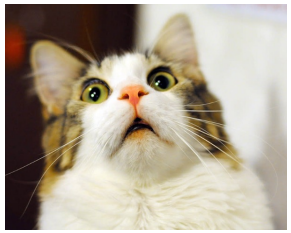
❖ Rispetto al sommario glm(), individuare (o ricalcolare se necessario) nei risultati dell'algoritmo iterativo:

- i coefficienti B,
- gli errori standard dei coefficienti stimati,
- la Deviance del modello (Residual Deviance)
- i tre tipi di residui che abbiamo visto (Deviance residuals, grezzi e residui di Pearson)

Cosa succede qui?

```
x1 = c(0.5, 2, 10, 6, 8, 1, 8, 4, 0, 7, 5, 1.1)
x2 = c(7, 9.5, 6, 0, 8.1, 9, 10, 0, 8, 5, 3, 4)
y = c(0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0)
```

❖ Stimare i coefficienti del modello di regressione logistica mediante algoritmo di Fisher scoring e replicare i risultati con la funzione glm().



❖ Quanti cicli sono necessari? Gli errori standard dei coefficienti hanno dei valori insoliti? La matrice Hessiana come diventa? Ne parliamo la prossima volta!