Modelo de regresión de Poisson

Contents

1 Modelo

Durante la guerra del Vietnam, la armada de los Estados Unidos empleó diferentes tipos de bombarderos en misiones para destruir puentes, carreteras, y otros objetivos de transporte. Entre ellos destacaban el A-4 Skyhawk y el A-6 Intruder. El archivo *Aircraft_Damage.csv* contiene la información de 30 misiones donde participaron estos aviones:

```
d = read.csv("datos/Aircraft_Damage.csv")
str(d)
```

```
## 'data.frame':    30 obs. of  4 variables:
##  $ damage    : int  0 1 0 0 0 0 1 0 0 2 ...
##  $ bomber    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ load      : int  4 4 4 5 5 5 6 6 6 7 ...
##  $ experience: num  91.5 84 76.5 69 61.5 80 72.5 65 57.5 50 ...
```

- damage: numero de zonas donde el bombardero presentaba daños debido a las defensas antibombarderos;
- bomber: variable cualitativa, vale 0 para el A-4 y 1 para el A-6;
- load: carga de bombas (en toneladas);
- experience: número de meses de experiencia en vuelo de la tripulación.

El objetivo es utilizar un modelo que relacione el número de zonas con daño y el resto de variables. En este caso la variable respuesta no es normal; y tampoco es binomial. Cuando la variable respuesta sea del tipo "número de veces que ocurre algo" se utiliza una variable de Poisson.

El modelo de Poisson se define como

$$P(Y_i = y_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}, \quad y_i = 0, 1, 2, 3, \dots$$

donde:

$$\lambda_i = exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki}) = exp(x_i^T \beta)$$

Esta última expresión se puede reescribir como:

$$\lambda_i = exp(x_i^T \beta)$$

ya que

$$\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} = \begin{bmatrix} 1 & x_{1i} & x_{2i} & \cdots & x_{ki} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \cdots \\ \beta_k \end{bmatrix} = x_i^T \beta$$

Otra forma de expresar el modelo es:

$$Y_i = f(x_{1i}, x_{2i}, \cdots, x_{ki}) + U_i$$

donde $Y_i = 0, 1, 2, 3, \cdots$. Se admite que $E[u_i] = 0$, por lo que:

$$E[Y_i] = f(x_{1i}, x_{2i}, \cdots, x_{ki})$$

Las variables de Poisson cumplen que $E[Y_i] = \lambda_i$, por lo que:

$$f(x_{1i}, x_{2i}, \cdots, x_{ki}) = exp(x_i^T \beta)$$

2 Estimación de los parámetros del modelo: máxima verosimilitud

2.1 La función de verosimilitud

La función de verosimilitud es la probabilidad de obtener la muestra dada. Por tanto, dada la muestra $\{y_1, y_2, \cdots, y_n\}$, la probabilidad de obtener dicha muestra es:

$$P(Y_1 = y_1, Y_2 = y_2, \cdots, Y_n = y_n) = \prod_{i=1}^{n} P(Y_i = y_i) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}$$

Se denomina función de verosimilitud a la probabilidad de obtener la muestra:

$$L(\beta) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!}$$

El logaritmo de la función de verosimilitud es:

$$logL(\beta) = log \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{y_i}}{y_i!} = \sum_{i=1}^{n} (-\lambda_i + y_i log(\lambda_i) - log(y_i!))$$

2.2 El máximo de la función de verosimilitud

Derivando e igualando a cero:

$$\frac{\partial logL(\beta)}{\partial \beta} = \begin{bmatrix} \frac{\partial logL(\beta)}{\partial \beta_0} \\ \frac{\partial logL(\beta)}{\partial \beta_1} \\ \cdots \\ \frac{\partial logL(\beta)}{\partial \beta_k} \end{bmatrix} = X^T(y - \lambda) = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

donde X:

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{k1} \\ 1 & x_{12} & \cdots & x_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{1n} & \cdots & x_{kn} \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, \quad \lambda = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \cdots \\ \lambda_n \end{bmatrix}$$

El máximo de la función log-verosimilitud se tiene que hacer numéricamente.

En los siguientes apartados se va a necesitar la matriz de derivadas segundas o matriz hessiana. Su valor es:

$$\frac{\partial logL(\beta)}{\partial \beta \partial \beta^T} = \begin{bmatrix} \frac{\partial^2 logL(\beta)}{\partial \beta_0^2} & \frac{\partial^2 logL(\beta)}{\partial \beta_0 \partial \beta_1} & \cdots & \frac{\partial^2 logL(\beta)}{\partial \beta_0 \partial \beta_k} \\ \frac{\partial^2 logL(\beta)}{\partial \beta_1 \partial \beta_0} & \frac{\partial^2 logL(\beta)}{\partial \beta_1^2} & \cdots & \frac{\partial^2 logL(\beta)}{\partial \beta_1 \partial \beta_k} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 logL(\beta)}{\partial \beta_k \partial \beta_0} & \frac{\partial^2 logL(\beta)}{\partial \beta_k \partial \beta_1} & \cdots & \frac{\partial^2 logL(\beta)}{\partial \beta_k^2} \end{bmatrix} = -X^T W X$$

donde W es una matriz diagonal con

$$W_{ii} = \lambda_i$$

2.3 Funciones en R

Las funciones que calculan el logaritmo de la verosimilitud, el gradiente y el hessiano se han incluido en el archivo poisson_funciones.R.

```
source("poisson_funciones.R")
beta = c(-0.5,0.5,0.5,-0.05)
y = d$damage
X = cbind(rep(1,nrow(d)), d[,2:4])
logL(beta,y,X)
```

```
## [1] -121.8364
```

```
grad_logL(beta,y,X)
```

```
##                        [,1]
## rep(1, nrow(d))   -107.8044
## bomber            -111.6042
## load             -1484.6930
## experience       -6768.9506
```

```
hess_logL(beta, X)
```

```
##                  rep(1, nrow(d))     bomber        load  experience
## rep(1, nrow(d))       -153.8044  -147.6042   -1961.693  -10250.551
## bomber                -147.6042  -147.6042   -1919.973   -9834.587
## load                 -1961.6930 -1919.9729  -25677.110 -130926.748
## experience          -10250.5506 -9834.5867 -130926.748 -706790.727
```

```
nlme::fdHess(beta,logL, y = d$damage, X)
```

```
## $mean
## [1] -121.8364
##
## $gradient
## [1]  -107.8044  -111.6042 -1484.6930 -6768.9506
##
## $Hessian
##              [,1]       [,2]        [,3]        [,4]
## [1,]   -153.8105  -147.6050   -1961.733  -10250.658
## [2,]   -147.6050  -147.6112   -1920.013   -9834.694
## [3,]  -1961.7332 -1920.0127  -25677.108 -130930.699
## [4,] -10250.6578 -9834.6936 -130930.699 -706792.179
```

2.4 Cálculo del máximo mediante el algoritmo BFGS

La función optim() solo minimiza. Pero como calcular el máximo de logL equivale a minimizar -logL, se define la función logL_optim en la que simplemente se le cambia el signo a la verosimilitud.

Como punto de partida del algoritmo podemos utilizar por ejemplo la solución de mínimos cuadrados:

```
m = lm(damage ~ bomber + load + experience, data = d)
beta_i = coef(m)
X = model.matrix(m)
mle = optim(par = beta_i, fn = logL_optim, y = d$damage, X = X, gr = NULL, method = "BFGS", hessian = T
```

```
## initial  value 237.034129
## iter   2 value 102.962769
## iter   3 value 87.328274
## iter   4 value 71.799473
## iter   5 value 66.921488
## iter   6 value 53.751169
## iter   7 value 42.182454
## iter   8 value 40.762415
## iter   9 value 40.547879
## iter  10 value 39.831923
## iter  11 value 39.831042
## iter  12 value 39.830381
## iter  13 value 39.826114
## iter  14 value 39.825354
## iter  15 value 39.825144
## iter  16 value 39.825143
## iter  16 value 39.825143
## iter  17 value 39.825124
## iter  18 value 39.825010
## iter  18 value 39.825010
## iter  19 value 39.824999
## iter  19 value 39.824999
## iter  19 value 39.824999
```

```
## final  value 39.824999
## converged
```

```r
mle$par
```

```
## (Intercept)      bomber        load  experience
## -0.38275001  0.57195138  0.16453262 -0.01373949
```

2.5 Estimacion con R

```r
d$bomber = factor(d$bomber, labels = c("A4","A6"))
m2 = glm(damage ~ bomber + load + experience, data = d, family = poisson)
summary(m2)
```

```
##
## Call:
## glm(formula = damage ~ bomber + load + experience, family = poisson,
##     data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6418  -1.0064  -0.0180   0.5581   1.9094
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.406023   0.877489  -0.463   0.6436
## bomberA6     0.568772   0.504372   1.128   0.2595
## load         0.165425   0.067541   2.449   0.0143 *
## experience  -0.013522   0.008281  -1.633   0.1025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 53.883  on 29  degrees of freedom
## Residual deviance: 25.953  on 26  degrees of freedom
## AIC: 87.649
##
## Number of Fisher Scoring iterations: 5
```