Modelo de regresión logística con k regresores

Contents

1 Modelo

El archivo *MichelinNY.csv* contiene linformación de 164 restaurantes franceses incluidos en la guía *Zagat Survey 2006: New York City Restaurants*.

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

```
## 'data.frame':    164 obs. of  6 variables:
##  $ InMichelin     : int  0 0 0 1 0 0 1 1 1 0 ...
##  $ Restaurant.Name: chr  "14 Wall Street" "212" "26 Seats" "44" ...
##  $ Food           : int  19 17 23 19 23 18 24 23 27 20 ...
##  $ Decor          : int  20 17 17 23 12 17 21 22 27 17 ...
##  $ Service        : int  19 16 21 16 19 17 22 21 27 19 ...
##  $ Price          : int  50 43 35 52 24 36 51 61 179 42 ...
```

El objetivo es utilizar un modelo que relacione una serie de regresores con una variable respuesta binaria:

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

donde en este caso $y_i = \{0, 1\}$. Para ello se definen las siguientes probabilidades:

- $P(y_i = 1) = \pi_i$
- $P(y_i = 0) = 1 - \pi_i$.

donde:

$$\pi_i = \frac{exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki})}{1 + exp(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki})}$$

Se puede escribir que

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

Es decir

$$\pi_i = \frac{exp(x_i^T\beta)}{1+exp(x_i^T\beta)}$$

Como se admite que $E[u_i] = 0$:

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

Como y_i toma valores 1 y 0 con probabilidades π_i y $1-\pi_i$ se tiene que:

$$E[y_i] = 1 * \pi_i + 0 * (1-\pi_i) = \pi_i$$

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

2 Estimación de los parámetros del modelo: máxima verosimilitud

Para estimar los parámetros del modelo (β_0 y β_1) se utiliza el método de máxima verosimilitud, que consiste en:

- Definir la función logaritmo de la verosimilitud;
- La estimación de los parámetros son aquellos que maximizan la funcion log-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:

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

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

$$L(\beta) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

El logaritmo de la función de verosimilitud es:

$$logL(\beta) = \prod_{i=1}^{n} \pi_i^{y_i}(1-\pi_i)^{1-y_i} = \sum_{i=1}^{n}(y_i log\pi_i + (1-y_i)log(1-\pi_i))$$

$$= \sum_{i=1}^{n}\left(y_i log\left(\frac{exp(x_i^T\beta)}{1+exp(x_i^T\beta)}\right) + (1-y_i)log\left(1-\frac{exp(x_i^T\beta)}{1+exp(x_i^T\beta)}\right)\right)$$

$$= \sum_{i=1}^{n}\left(y_i log\left(\frac{exp(x_i^T\beta)}{1+exp(x_i^T\beta)}\right) + (1-y_i)log\left(\frac{1}{1+exp(x_i^T\beta)}\right)\right)$$

$$= \sum_{i=1}^{n} (y_i log(exp(x_i^T \beta) - y_i log(1 + exp(x_i^T \beta)) - (1 - y_i) log(1 + exp(x_i^T \beta)))$$

$$= \sum_{i=1}^{n} (y_i (x_i^T \beta) - log(1 + exp(x_i^T \beta)))$$

En R, la función de verosimilitud la podemos calcular así:

```
logL = function(beta,y,X){
  # asumimos que beta es un vector
  # beta = [beta0 beta1 .. betak]

  n = length(y)
  suma = 0
  for (i in 1:n){
    suma = suma + y[i]*sum(X[i,]*beta) -
      log(1 + exp( sum(t(X[i,])*beta) ))
  }
  return(suma)
}
```

Por ejemplo, para $\beta_0 = -2$ y $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0.5$, la función de verosimilitud vale:

```
beta = c(-2,0.05,0.05,0.05,0.05)
X = cbind(rep(1,nrow(d)), d[,3:6])
logL(beta,d$InMichelin,X)
```

```
## [1] -261.6999
```

2.2 El máximo de la función de verosimilitud

Derivando e igualando a cero:

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

donde X:

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{k1} \\ 1 & x_{12} & \cdots & x_{k2} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{1n} & \cdots & x_{kn} \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}, \quad \pi = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \cdots \\ \pi_n \end{bmatrix}$$

Sin embargo no es posible despejar las incóginitas del vector β de las ecuaciones anteriores. 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

$$W_{ii} = \pi_i(1 - \pi_i)$$

En R:

```
grad_logL = function(beta,y,X){
  X = as.matrix(X)
  n = length(y)
  y = matrix(y, nrow = n, ncol = 1)
  pi = matrix(0, nrow = n, ncol = 1)
  for (i in 1:n){
    pi[i,1] = exp(sum(X[i,]*beta))/(1 + exp(sum(X[i,]*beta)))
  }
  grad = t(X) %*% (y - pi)
  return(grad)
}
```

Comprobacion:

```
beta = c(-2,0.05,0.05,0.05,0.05)
X = cbind(rep(1,nrow(d)), d[,3:6])
grad_logL(beta, d$InMichelin, X)
```

```
##                          [,1]
## rep(1, nrow(d))   -82.17658
## Food            -1638.18822
## Decor           -1425.64297
## Service         -1512.89102
## Price           -3366.03967
```

```
hess_logL = function(beta,X){
  X = as.matrix(X)
  n = nrow(X)
  W = matrix(0, nrow = n, ncol = n)
  for (i in 1:n){
    pi = exp(sum(X[i,]*beta))/(1 + exp(sum(X[i,]*beta)))
    W[i,i] = pi*(1-pi)
  }
  hess = - t(X) %*% W %*% X
  return(hess)
}
```

```
beta = c(-2,0.05,0.05,0.05,0.05)
X = cbind(rep(1,nrow(d)), d[,3:6])
hess_logL(beta, X)
```

```
##                 rep(1, nrow(d))       Food     Decor   Service      Price
## rep(1, nrow(d))       -7.216586  -144.6739  -122.702  -128.918   -279.872
```

```
## Food                  -144.673934 -2935.8604 -2466.220 -2601.383  -5614.096
## Decor                 -122.702048 -2466.2196 -2144.581 -2206.913  -4883.881
## Service               -128.917957 -2601.3826 -2206.913 -2345.857  -5104.190
## Price                 -279.872024 -5614.0957 -4883.881 -5104.190 -11574.359
```

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

```
## $mean
## [1] -261.6999
##
## $gradient
## [1]   -82.17658 -1638.18822 -1425.64297 -1512.89102 -3366.03967
##
## $Hessian
##             [,1]       [,2]      [,3]       [,4]        [,5]
## [1,]   -7.214614  -144.6798  -122.729  -129.0228   -279.9345
## [2,] -144.679824 -2932.3514 -2466.672 -2601.8495  -5615.4312
## [3,] -122.729044 -2466.6724 -2142.991 -2210.5800  -4887.4590
## [4,] -129.022841 -2601.8495 -2210.580 -2342.6567  -5108.8271
## [5,] -279.934455 -5615.4312 -4887.459 -5108.8271 -11571.9056
```

2.3 Algoritmo de Newton-Raphson

El algoritmo de Newton-Raphson para la función log-verosimilitud es:

$$\beta_{k+1} = \beta_k - \alpha H_k^{-1} G_k$$

donde $\beta = [\beta_0 \; \beta_1 \; \cdots \beta_k]^T$. Este algoritmo para la función log-verosimilitud se puede implementar en R de manera sencilla:

```
Newton_logL = function(beta_i, y, X, max_iter = 100, tol = 10^(-6), alfa = 0.1){

  # punto de partida
  beta = beta_i

  iter = 1
  tol1 = Inf
  while ((iter <= max_iter) & (tol1 > tol)){
    f = logL(beta,y,X)
    grad = grad_logL(beta,y,X)
    hess = hess_logL(beta,X)
    beta = beta - alfa*solve(hess) %*% grad
    f1 = logL(beta,y,X)
    tol1 = abs((f1-f)/f)
    print(paste("Iteracion ",iter," log-verosimilitud ",f1))
    iter = iter + 1
  }
  return(beta)
}
```

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

```
m = lm(InMichelin ~ Food + Decor + Service + Price, data = d)
beta_i = coef(m)
X = cbind(rep(1,nrow(d)), d[,3:6])
Newton_logL(beta_i,d$InMichelin,X)
```

```
## [1] "Iteracion  1  log-verosimilitud  -103.697764191695"
## [1] "Iteracion  2  log-verosimilitud  -99.4843006321041"
## [1] "Iteracion  3  log-verosimilitud  -95.9467514594378"
## [1] "Iteracion  4  log-verosimilitud  -92.9381705910831"
## [1] "Iteracion  5  log-verosimilitud  -90.3511527629657"
## [1] "Iteracion  6  log-verosimilitud  -88.1070168072994"
## [1] "Iteracion  7  log-verosimilitud  -86.1493000858726"
## [1] "Iteracion  8  log-verosimilitud  -84.4384134102019"
## [1] "Iteracion  9  log-verosimilitud  -82.945869738486"
## [1] "Iteracion  10  log-verosimilitud  -81.6488396319209"
## [1] "Iteracion  11  log-verosimilitud  -80.5266972236002"
## [1] "Iteracion  12  log-verosimilitud  -79.5598372440519"
## [1] "Iteracion  13  log-verosimilitud  -78.7297318616064"
## [1] "Iteracion  14  log-verosimilitud  -78.0192580762245"
## [1] "Iteracion  15  log-verosimilitud  -77.4129016978384"
## [1] "Iteracion  16  log-verosimilitud  -76.8967965069963"
## [1] "Iteracion  17  log-verosimilitud  -76.4586572635604"
## [1] "Iteracion  18  log-verosimilitud  -76.087662365363"
## [1] "Iteracion  19  log-verosimilitud  -75.7743198740009"
## [1] "Iteracion  20  log-verosimilitud  -75.5103336404772"
## [1] "Iteracion  21  log-verosimilitud  -75.2884766341898"
## [1] "Iteracion  22  log-verosimilitud  -75.1024738677407"
## [1] "Iteracion  23  log-verosimilitud  -74.9468952611842"
## [1] "Iteracion  24  log-verosimilitud  -74.8170580259202"
## [1] "Iteracion  25  log-verosimilitud  -74.7089379461753"
## [1] "Iteracion  26  log-verosimilitud  -74.6190889452876"
## [1] "Iteracion  27  log-verosimilitud  -74.5445703893246"
## [1] "Iteracion  28  log-verosimilitud  -74.4828816408913"
## [1] "Iteracion  29  log-verosimilitud  -74.4319034137963"
## [1] "Iteracion  30  log-verosimilitud  -74.3898454952845"
## [1] "Iteracion  31  log-verosimilitud  -74.3552004038294"
## [1] "Iteracion  32  log-verosimilitud  -74.326702544819"
## [1] "Iteracion  33  log-verosimilitud  -74.3032924202697"
## [1] "Iteracion  34  log-verosimilitud  -74.2840854462995"
## [1] "Iteracion  35  log-verosimilitud  -74.2683449358644"
## [1] "Iteracion  36  log-verosimilitud  -74.2554588149883"
## [1] "Iteracion  37  log-verosimilitud  -74.2449196580623"
## [1] "Iteracion  38  log-verosimilitud  -74.2363076507231"
## [1] "Iteracion  39  log-verosimilitud  -74.2292761159564"
## [1] "Iteracion  40  log-verosimilitud  -74.2235392689608"
## [1] "Iteracion  41  log-verosimilitud  -74.218861897558"
## [1] "Iteracion  42  log-verosimilitud  -74.2150506963734"
## [1] "Iteracion  43  log-verosimilitud  -74.2119470136567"
## [1] "Iteracion  44  log-verosimilitud  -74.2094207987802"
## [1] "Iteracion  45  log-verosimilitud  -74.2073655656118"
## [1] "Iteracion  46  log-verosimilitud  -74.2056942118484"
## [1] "Iteracion  47  log-verosimilitud  -74.2043355568448"
## [1] "Iteracion  48  log-verosimilitud  -74.2032314804836"
## [1] "Iteracion  49  log-verosimilitud  -74.2023345632666"
## [1] "Iteracion  50  log-verosimilitud  -74.2016061432063"
## [1] "Iteracion  51  log-verosimilitud  -74.2010147184303"
## [1] "Iteracion  52  log-verosimilitud  -74.2005346358697"
## [1] "Iteracion  53  log-verosimilitud  -74.2001450161937"
## [1] "Iteracion  54  log-verosimilitud  -74.1998288734636"
```

```
## [1] "Iteracion  55  log-verosimilitud  -74.1995723950056"
## [1] "Iteracion  56  log-verosimilitud  -74.1993643529103"
## [1] "Iteracion  57  log-verosimilitud  -74.1991956235178"
## [1] "Iteracion  58  log-verosimilitud  -74.1990587953824"
## [1] "Iteracion  59  log-verosimilitud  -74.1989478496529"
## [1] "Iteracion  60  log-verosimilitud  -74.1988578996574"
## [1] "Iteracion  61  log-verosimilitud  -74.1987849788507"
```

```
##                          [,1]
## rep(1, nrow(d)) -11.15839984
## Food              0.40328290
## Decor             0.10005790
## Service          -0.19161023
## Price             0.09123707
```

2.4 Algoritmo BFGS

La función que vamos a minimizar es:

```
logLa = function(beta,y,X){
  logL = logL(beta,y,X)
  return(-logL)
}
```

Utilizando el mismo punto de partida que para el algoritmo Newton:

```
mle = optim(par = beta_i, fn = logLa, y = d$InMichelin, X = X, gr = NULL, method = "BFGS", hessian = TRU
```

```
## initial  value 108.793234
## iter   2 value 108.179208
## iter   3 value 95.583943
## iter   4 value 94.546080
## iter   5 value 93.925129
## iter   6 value 76.862843
## iter   7 value 74.881601
## iter   8 value 74.297605
## iter   9 value 74.231461
## iter  10 value 74.210848
## iter  11 value 74.202494
## iter  12 value 74.199408
## iter  13 value 74.199353
## iter  14 value 74.199254
## iter  15 value 74.199238
## iter  16 value 74.199235
## iter  17 value 74.198614
## iter  18 value 74.198479
## iter  19 value 74.198474
## iter  19 value 74.198474
## iter  19 value 74.198474
## final  value 74.198474
## converged
```

```
mle$par
```

```
##  (Intercept)         Food        Decor      Service        Price
## -11.19660070   0.40483799   0.09992470  -0.19249028   0.09175322
```

2.5 Estimacion con R

```r
m2 = glm(InMichelin ~ Food + Decor + Service + Price, data = d, family = binomial)
summary(m2)
```

```
##
## Call:
## glm(formula = InMichelin ~ Food + Decor + Service + Price, family = binomial,
##     data = d)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3923  -0.6723  -0.3810   0.7169   1.9694
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.19745    2.30896  -4.850 1.24e-06 ***
## Food          0.40485    0.13146   3.080  0.00207 **
## Decor         0.09997    0.08919   1.121  0.26235
## Service      -0.19242    0.12357  -1.557  0.11942
## Price         0.09172    0.03175   2.889  0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.79  on 163  degrees of freedom
## Residual deviance: 148.40  on 159  degrees of freedom
## AIC: 158.4
##
## Number of Fisher Scoring iterations: 6
```