Logistic Regression Estimation & Inference

Gustavo

10/14/2022

The Logistic Regression handout provides an overview of maximum likelihood in the logistic regression model. Here we present code related to the concepts introduced in that handout.

In a logistic regression we model the logarithm of the odds, log(p/(1-p)), as a linear regression on covariates.

Specifically, let yi be a 0/1 Bernoulli random variable and $\mathbf{x}_i$ a vector of covariates for the ith individual (the first entry would typically be a 1, to accomodate an *intercept*).

Using this, we model the log-odds using a linear function:

$log(pi/(1-pi)) = \mathbf{x}_i'\mathbf{b} = \eta_i$,

where $\mathbf{b}$ is a vector of regression coefficients (typically the first entry is the *intercept*), and $\eta_i = \mathbf{x}_i'\mathbf{b}$ is the linear predictor. Solving for the success probability, this yields $p_i = exp(\eta_i)/(1 + exp(\eta_i))$.

The log-likelihood

Assuming conditional independence, the log-likelihood function is

$l(\mathbf{b}|\mathbf{X}, \mathbf{y}) = \Sigma_{i=1}^n y_i log(p_i) + (1 - y_i) * log(1 - p_i)$ (see handout for an step-by-step derivation).

The following function evaluates the likelihood function.

```r
negLogLik=function(y,X,b){
  eta=X%*%b  # linear predictor
  theta=exp(eta)/(1+exp(eta)) # success probability
  logLik=sum(y*log(theta),(1-y)*log(1-theta))
      return(-logLik)
}
```

Alternatively, we can compute the log-likelihood using `dbinom`.

```r
negLogLik2=function(y,X,b){
  eta=X%*%b  # linear predictor
  theta=exp(eta)/(1+exp(eta)) # success probability
  logLik=sum( dbinom(size=1,prob=theta,x=y,log=TRUE)  )
      return(-logLik)
}
```

Estimation

We are going to use the above functions and `optim()` to obtain Maximum Likelihood estimates of the coefficients in the logistic regression model. However, before we do that, let's fit a logistic regression model using `glm()`; we will then use the results of `glm()` to compare with the ML estimates derived using `optim()`.

```r
DATA=read.table('https://raw.githubusercontent.com/gdlc/STAT_COMP/master/DATA/goutData.txt',header=TRUE
## Recall that for logistic regression glm likes y coded as 0/1
DATA$y=ifelse(is.na(DATA$gout),NA,ifelse(DATA$gout=='Y',1,0))
fm=glm(y~age+sex+race,data=DATA,family=binomial)
summary(fm)
```

```
##
## Call:
## glm(formula = y ~ age + sex + race, family = binomial, data = DATA)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8294  -0.4256  -0.3537  -0.2713   2.6115
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.08409    2.35356  -3.435 0.000593 ***
## age          0.09239    0.03644   2.536 0.011227 *
## sexM         0.44915    0.38821   1.157 0.247288
## raceW       -0.74329    0.41914  -1.773 0.076168 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 213.11  on 399  degrees of freedom
## Residual deviance: 203.05  on 396  degrees of freedom
## AIC: 211.05
##
## Number of Fisher Scoring iterations: 5
```

I) Obtaining ML estimates using optim

We first create the incidence matrix of effects (X), then generate initial values for the coefficients, and finally obtain estimats using `optim()`.

```r
X=model.matrix(~age+sex+race,data=DATA) # incidence matrix

# initial values
b=rep(0,ncol(X))
b[1]=log(mean(DATA$y)/(1-mean(DATA$y)))

fm2=optim(fn=negLogLik,X=X,y=DATA$y,par=b)
```

Before proceding to extract the estimates, we check if the algorithm converged

```r
 fm2$convergence
```

```
## [1] 0
```

Zero here indicates that the algorithm converged (try `help(optim)`) to obtain information about what other codes mean.

Now that we know that the algorithm converged, we extract the estimates and compare with those obtained using `glm()`.

Note: for ease of convergence, it may be convinient to center all columns (except the 1st one, corresponding to the intercept); however, for this simple model that is not needed.

```
fm2$par # these are the estimates
```

```
## [1] -8.08392093  0.09239889  0.44784750 -0.74305194
```

```
cbind(coef(fm),fm2$par)
```

```
##                   [,1]        [,2]
## (Intercept) -8.08408808 -8.08392093
## age          0.09239102  0.09239889
## sexM         0.44914774  0.44784750
## raceW       -0.74328787 -0.74305194
```

II) Inferences

In this section we will show how to:

- Obtain the variance-covariance matrix of estimated regression coefficients,
- Derive SEs for each of the coefficients, and
- Derive the corresponding t-statistics and p-values.

These pvalues can be used to test for the significance of individual coefficients.

Then we will show how to perform test involving more than one restriction (i.e., more than one degree of freedom) using Likelihood-ratio test (LRT) and Wald's test.

The math behind the computations presented here are presented in the Logistic Regression handout.

The variance-covariance matrix of estimates

In large samples, the sampling (co)variance of estimates can be approximated using the inverse of the matrix of second derivatives of the negative log-likelhiood (aka the Hessian) evaluated the ML estimates.

Using glm you can extract the (co)variance matrix of estimates using the following:

```
 V=vcov(fm)
```

Now using optim. First I re-fit the model and now specify `hessian=T`.

```
fm2=optim(fn=negLogLik,X=X,y=DATA$y,par=b,hessian=TRUE) #hessian=TRUE returns the hessian
H=fm2$hessian
V2=solve(H)
V
```

```
##             (Intercept)           age          sexM        raceW
## (Intercept)  5.53925465 -0.0846288167 -0.0474192397  0.014111132
## age         -0.08462882  0.0013277321 -0.0002919836 -0.001914644
## sexM        -0.04741924 -0.0002919836  0.1507094604 -0.018281305
## raceW        0.01411113 -0.0019146440 -0.0182813048  0.175677880
```

```
V2
```

```
##             [,1]          [,2]          [,3]         [,4]
## [1,]  5.54935638 -0.0847657856 -0.0480918398  0.015498286
## [2,] -0.08476579  0.0013295353 -0.0002807722 -0.001935971
## [3,] -0.04809184 -0.0002807722  0.1506926164 -0.018286371
## [4,]  0.01549829 -0.0019359711 -0.0182863705  0.175729964
```

SEs, t-stat, and pvalues

Now that we have estimates and the variance covariance matrix of estimates, we can obtain the SEs, t-statistics, and p-values (once you have estiamtes and V, these steps are the same as in the linear model).

```
EST=matrix(nrow=length(fm2$par),ncol=4)
colnames(EST)=c('Estimate','SE','tStat','pvalue')
rownames(EST)=colnames(X)

EST[,1]=fm2$par
EST[,2]=sqrt(diag(V2))
EST[,3]=EST[,1]/EST[,2]
EST[,4]=2*pt(abs(EST[,3]),df=nrow(DATA)-nrow(EST),lower.tail=FALSE)

EST
```

```
##                Estimate         SE     tStat       pvalue
## (Intercept) -8.08392093 2.35570719 -3.431632 0.0006631481
## age          0.09239889 0.03646279  2.534060 0.0116595118
## sexM         0.44784750 0.38819147  1.153677 0.2493285123
## raceW       -0.74305194 0.41920158 -1.772541 0.0770735588
```

```
summary(fm)$coef
```

```
##                Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -8.08408808 2.35356212 -3.434831 0.0005929235
## age          0.09239102 0.03643806  2.535564 0.0112266468
## sexM         0.44914774 0.38821316  1.156962 0.2472880297
## raceW       -0.74328787 0.41913945 -1.773367 0.0761679920
```

To obtain (approximate) 95% CIs, you can simply use $estimate + / - 1.96 \times SE$.

Test involving 2 or more DF

We cover two tests: Likelihood Ratio Test and Wald's test.

Likelihood Ratio Test

For any model fitted via Max. Likelihood (under regularity conditions) we can test hypothesis involving any arbitrary numbers of restrictions using a likelihood ratio test. This test is analogous to the F-test that we use in linear models. However, unlike F-test (which can only be used for linear models), LRT can be used with any model fitted via ML.

To implement a LRT we need to (see handout for further details):

- Fit the model under H0 and under HA
- Extract the log-likleihood of each of these model
- Compute the chi-square statistic as $-2(logLik(H0) - logLik(HA))$
- Obtain the pvalue using a chi-square distribution with DF equal to the difference in the number of parameters between HA and H0.

Suppose we want to test whether sex and race have an effect on the probability of developing gout after accounting for age. Here the null is `gout~age` and the alternative is `gout~age+sex+race`. The test is implemented as follows.

```
H0=glm(y~age,data=DATA,family='binomial')
HA=glm(y~age+sex+race,data=DATA,family='binomial')

CHISQ.LRT=-2*(logLik(H0)-logLik(HA))
DF=length(coef(HA))-length(coef(H0))
```

```
pValue.LRT=pchisq(df=DF,q=CHISQ.LRT,lower.tail=FALSE)

LRT=c('df'=DF,'CHISQ'=CHISQ.LRT,'pvalue'=pValue.LRT)
LRT
```

```
##       df    CHISQ   pvalue
## 2.000000 3.950004 0.138761
```

```
anova(H0,HA)
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ age
## Model 2: y ~ age + sex + race
##   Resid. Df Resid. Dev Df Deviance
## 1       398     207.00
## 2       396     203.05  2     3.95
```

Wald Test

To implement Wald's test we need to (see handout for further details):

- Express the null hypothes in linear form $\mathbf{Tb} = \mathbf{a}$ where $\mathbf{T}$ is a matrix of contrasts (as many contrasts as rows in $\mathbf{T}$), $\mathbf{b}$ is the vector of regression coefficients, and $\mathbf{a}$ is the value assumed for the contrasts under the null (typically $\mathbf{0}$).
- Extract from the fitted model the (co)cariance matrix of the estimated effects $Var(b) = \mathbf{V}$,
- Compute the (co)cariance matrix of the contrasts ($Cov(Tb) = \mathbf{TVT}'$)
- Compute the chi-square statistic $CHISQ = (\hat{\mathbf{b}} - \mathbf{a})'\mathbf{T}'(\mathbf{TVT})^{-\mathbf{1}}\mathbf{T}(\hat{\mathbf{b}} - \mathbf{a})$
- Compute the pvalue using `pchisq()` with df eaual to the number of rows of T

To test whether sex, or sex, or both have an effect, after acounting for age, the contrast matrix of Wald's test is

```
T=rbind('sex'=c(0,0,1,0),'race'=c(0,0,0,1))
colnames(T)=names(coef(fm))
T
```

```
##      (Intercept) age sexM raceW
## sex            0   0    1     0
## race           0   0    0     1
```

Now we peoceed with the steps delinated above

```
V=vcov(fm)
bHat=coef(fm)
d=T%*%bHat

V_D=T%*%V%*%t(T) # the variance covariance matrix of the contrasts
CHISQ.WALD=t(d)%*%solve(V_D)%*%d

## compare with the Chi Square from the likleihood-ratio test
CHISQ.WALD
```

```
##          [,1]
## [1,] 4.073785
```

```
CHISQ.LRT
```

```
## 'log Lik.' 3.950004 (df=2)
```

```
pValWald=pchisq(CHISQ.WALD,df=nrow(T),lower.tail=FALSE)

WALD=c('df'=nrow(T),'CHISQ'=CHISQ.WALD,'pValue'=pValWald)

cbind(WALD,LRT)
```

```
##             WALD      LRT
## df     2.0000000 2.000000
## CHISQ  4.0737853 3.950004
## pValue 0.1304334 0.138761
```

Predictions

We focus now on how to predict the success probabilities for special cases and how to obtain CIs for the predicted probabilities (see handout for more details).

Recall that the linear predictor takes the form

$log(p_i/(1-p_i) = \eta_i = \mathbf{x_i'b}$

where $p_i = Prob(Y_i = 1)$ and $log(p_i/(1-p_i)$ is the logarithm of the odds (or logit).

For any observation we can obtain predictions in the scale of the logit using $\mathbf{x}_i'\hat{\mathbf{b}}$. Furthermore, we can obtain the variance of the prediction using $Var(\hat{\eta}_i) = \mathbf{x}_i'\mathbf{V}\mathbf{x}_i$ where $\mathbf{V}$ is the covariance matrix of the estimated effects (see previous code). Using this we can obtain a 95% CI for the prediction in the scale of the logit using $CI_{95} = \hat{\eta}_i + / - 1.96\sqrt{\mathbf{x}_i'\mathbf{V}\mathbf{x}_i}$.

The following code shows how to obtain these predictions, SEs, and 95% CIs using glm for a 50 years old female white.

```
TMP=data.frame(sex='F',race='W',age=50)
pred=predict(fm,newdata=TMP,se.fit=TRUE)
pred
```

```
## $fit
##         1
## -4.207825
##
## $se.fit
## [1] 0.6388575
##
## $residual.scale
## [1] 1
```

```
## alternatively
fit=sum(c(1,50,0,1)*coef(fm)) # here c(1,50,0,1) is 'x_i'
var.fit=c(1,50,0,1)%*%V%*%c(1,50,0,1)
se.fit=sqrt(var.fit)
fit
```

```
## [1] -4.207825
```

```
se.fit
```

```
##           [,1]
## [1,] 0.6388575
```

```
CI=fit+c(-1.96,1.96)*se.fit
```

```
## Warning in c(-1.96, 1.96) * se.fit: Recycling array of length 1 in vector-array arithmetic is depreca
##   Use c() or as.vector() instead.
```

Solving the regression equation $(log(p_i/(1-p_i)) = \eta_i$ for the success probability we get

$p_i = exp(\eta_i)/(1+exp(\eta_i))$

We can use this to map from predictions (and CIs) on the scale of the logit to predictions on the scale of the response (i.e., success probability).

```
predProb=exp(fit)/(1+exp(fit))
CI.FIT=exp(CI)/(1+exp(CI))
predProb
```

```
## [1] 0.01466057
```

```
CI.FIT
```

```
## [1] 0.0042356 0.0494695
```

Or, using `predict()`, specifying `type=response`.

```
tmp= predict(fm,newdata=TMP,type='response',se.fit=TRUE)
tmp
```

```
## $fit
##          1
## 0.01466057
##
## $se.fit
##           1
## 0.009228703
##
## $residual.scale
## [1] 1
```