Compulsory exercise 1

TMA4315 Generalized Linear Models F2019

Julie Berg and Liv Elise Herstad

30 September, 2019

For the binary regression model with grouped data and the logit choice of link function, show that the log likelihood, the score function and the Fisher information is given by the expression on p. 283 in Fahrmeir.

The response model is $\pi_i = h(\eta_i) = \frac{1}{1+e^{-\eta_i}}$ where $\eta_i = x_i^T\beta$ is the linear predictor with $x_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik})^T$ and $\beta = (\beta_0, \beta_1, \ldots, \beta_k)^T$.

We use the link function $g(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \eta_i$. Since we are working with grouped data, the summarizations are from Group 1 to Group G.

When $y_i \sim Bin(n_i, \pi_i)$, the log-likelihood is:

$$
\begin{aligned}
L(\beta) &= \prod_{i=1}^{G} \binom{n_i}{y_i} \pi_i^{y_i}(1-\pi_i)^{n_i-y_i} \\
l(\beta) &= \ln L(\beta) = \sum_{i=1}^{G} \ln\left[\binom{n_i}{y_i} + \pi_i^{y_i}(1-\pi_i)^{n_i-y_i}\right] \\
&= \sum_{i=1}^{G}\left[\ln\binom{n_i}{y_i} + y_i \ln\pi_i + (n_i - y_i)\ln(1-\pi_i)\right] \\
&= \sum_{i=1}^{G}\left[\ln\binom{n_i}{y_i} + y_i \ln\pi_i - y_i\ln(1-\pi_i) + n_i\ln(1-\pi_i)\right]
\end{aligned}
\tag{1}
$$

The term $\ln\binom{n_i}{y_i}$ is missing in Fahrmeir et. al. (2013). This does not affect the score function.

The score function is found by taking the first derivative of the log likelihood w.r.t. β.

$$
\begin{aligned}
s(\beta) &= \frac{\partial}{\partial\beta}\sum_{i=1}^{G}(y_i\ln(\pi_i) - y_i\ln(1-\pi_i) + n_i\ln(1-\pi_i)) \\
&= \sum_{i=1}^{G}\frac{\partial}{\partial\beta}(y_i\ln(\pi_i) - y_i\ln(1-\pi_i) + n_i\ln(1-\pi_i)) \\
&= \sum_{i=1}^{G}(y_i\frac{\partial}{\partial\beta}\ln(\pi_i) - y_i\frac{\partial}{\partial\beta}\ln(1-\pi_i) + n_i\frac{\partial}{\partial\beta}\ln(1-\pi_i)) \\
&= \sum_{i=1}^{G}(y_i x_i(1-\pi_i) - y_i(-\pi_i x_i) + n_i(-\pi_i x_i)) \\
&= \sum_{i=1}^{G}(y_i x_i - y_i x_i\pi_i + y_i x_i\pi_i - n_i x_i\pi_i) \\
&= \sum_{i=1}^{G}(y_i x_i - n_i x_i\pi_i) = \sum_{i=1}^{G} x_i(y_i - n_i\pi_i)
\end{aligned}
\tag{2}
$$

The expected Fisher Information is the variance of the score function, found by taking expectation of $s(\beta)s(\beta)^T$.

$$
\begin{aligned}
F(\beta) = E(s(\beta)s(\beta)^T) &= E(\sum_{i=1}^{G} n_i^2 x_i x_i^T (\bar{y_i} - \pi_i)^2) \\
&= \sum_{i=1}^{G} E(n_i^2 x_i x_i^T (\bar{y_i} - \pi_i)^2) = \sum_{i=1}^{G} n_i^2 x_i x_i^T E(\bar{y_i} - \pi_i)^2 \\
&= \sum_{i=1}^{G} n_i^2 x_i x_i^T Var(\bar{y_i}) = \sum_{i=1}^{G} n_i^2 x_i x_i^T Var(\frac{y_i}{n_i}) \\
&= \sum_{i=1}^{G} x_i x_i^T Var(y_i) = \sum_{i=1}^{G} x_i x_i^T n_i \pi_i (1 - \pi_i)
\end{aligned}
\tag{3}
$$

Write a function myglm in R that implements the Fisher scoring algorithm for this model.

```
library(investr)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
myglm <- function(formula, data) {
  #here we use Fisher's algorithm to calculate the estimated parameters
  X <- model.matrix(formula, data = data)
  beta <- c(0,0)
  for (i in 1:10) {
    pi <- 1 / (1 + exp(-X %*% beta))
    score <- c(0,0)
    fisher <- matrix(0,2,2)
    score <- t(X) %*% (data$y-data$n*pi)
    for (i in 1:8) {
      fisher <- fisher + X[i,] %*% t(X[i,]) * data$n[i] * pi[i] *(1 - pi[i])
    }
    h <- solve(fisher)
    beta <- beta + h %*% score
    print(beta) #prints the parameter values for each iteration
  }

  #we are now going to find the deviance:

  ybar = data$y/data$n
  pihat <- 1 / (1 + exp(-X %*% beta))

  deviance <- 2*sum(log(ybar^data$y*(1-ybar)^(data$n-data$y))-
                      log(pihat^data$y*(1-pihat)^(data$n-data$y)))

  return(list(beta, deviance))
}
```

Then, we test our function with the given call function and call it `testmyglm`. We iterate through the for-loop

10 times just to be sure, but we see that it converges after the 5th iteration. We chose not to use `apply` function as suggested because the data is not that big, therefore we did not prioritize to spend time on it.

```
library(investr)
testmyglm <- myglm(formula= ~ldose, data=beetle)
```

```
##                  [,1]
## (Intercept) -37.85638
## ldose        21.33743
##                  [,1]
## (Intercept) -53.85319
## ldose        30.38351
##                  [,1]
## (Intercept) -59.96521
## ldose        33.84419
##                  [,1]
## (Intercept) -60.70778
## ldose        34.26485
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27032
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
```

```
testmyglm
```

```
## [[1]]
##                  [,1]
## (Intercept) -60.71745
## ldose        34.27033
##
## [[2]]
## [1] 11.23223
```

Then we compare it to the

```
summary(glm(cbind(y,n-y) ~ ldose, binomial, data=beetle))
```

```
##
## Call:
## glm(formula = cbind(y, n - y) ~ ldose, family = binomial, data = beetle)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.5941  -0.3944   0.8329   1.2592   1.5940
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -60.717      5.181  -11.72   <2e-16 ***
## ldose         34.270      2.912   11.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 284.202  on 7  degrees of freedom
## Residual deviance:  11.232  on 6  degrees of freedom
## AIC: 41.43
##
## Number of Fisher Scoring iterations: 4
```

We see from the printed functions that we get the same values for `Intercept`, `ldose` and `Residual deviance`, which are:

`Intercept`= -60.717, `ldose`= 34.270, and `Residual deviance`= 11.232.

Based on the observed deviance, does the model fit the data reasonably well?

If our model is good, it should not deviate far from the saturated model. To test this, we formulate the hypothesis test

$$H_0 : \text{our model fits the data well vs.} H_1 : \text{our model does not fit the data well.}$$

and reject the null hypothesis if

$$D > \chi^2_{n-p_0}$$

where D is the deviance of the candidate model, n is the number of groups and p_0 is the number of parameters. In the beetle case, we have $n = 8$ and $p_0 = 2$.

We made a Goodness-of-Fit test which can be seen below.

```
goodnessOfFit <- function(deviance, alpha, n, p0) {
  #p0 is number of parameters in fitted models
  chi <- qchisq(1-alpha, df=n-p0)
  print(deviance)
  print(chi)
  return(deviance > chi) #gets TRUE if reject H0

}

goodnessOfFit(testmyglm[2], 0.05, 8, 2) #alpha=0.05, n=8, p0=2
```

```
## [[1]]
## [1] 11.23223
##
## [1] 12.59159
```

```
## [1] FALSE
```

The deviance is not larger than the chi-squared at the 0.05 level, meaning we do not reject H_0.

```
goodnessOfFit(testmyglm[2], 0.1, 8, 2) #alpha=0.1, n=8, p0=2
```

```
## [[1]]
## [1] 11.23223
##
## [1] 10.64464
```

```
## [1] TRUE
```

The deviance is larger than the chi-squared at level 0.1, meaning we reject H_0.

We calculated the p-value for the model, to see which level we reject and keep the null hypothesis.

```
D <- as.numeric(testmyglm[2])
1-pchisq(D,6)
```

```
## [1] 0.08145881
```

The p-value is 0.08145881, which corresponds to our previous chi-squared test where we kept the null hypothesis at 0.05 level, but rejected at 0.1. Showing the p-value makes it more general, and one can decide at which level to reject.

For level 0.05, the model is reasonably well fitted.