Homework 9

Jared Andreatta

2025-03-27

```r
# setwd("~/Dropbox/Home/Teaching/MATH531/MATH-531/MATH531S2024/Homework")
suppressMessages(library(fields))
```

```
## Warning: package 'fields' was built under R version 4.4.2

## Warning: package 'spam' was built under R version 4.4.2
```

```r
knitr::opts_chunk$set(echo = TRUE)
```

A note on this Rmd file

I added some include latex files in the heading so that some macros are available. E.g. For example a quick macro way for some boldface symbols:

```
  $\bc^2 = \ba^2 + \bb^2$
```

and the rendered version in pdf:

$\mathbf{c}^2 = \mathbf{a}^2 + \mathbf{b}^2$ For the me the rendering is touchy and a simple latex mistake can result in very strange errors. So be aware of that issue.

Some Background on Logistic Regression.

The linear and Gaussian model is

$$\mathbf{y}_i = x_i^T \beta + \mathbf{e}$$

with $\mathbf{e} \sim MN(0, \sigma^2 I)$ and $\beta, x_i \in \Re^p$ and n observations. X is an $n \times p$ matrix with each row being filled by the vector x_i. This might seem fuss but in taking partials in the following problems I find it useful to think of the rows as vectors.

For 0,1 count data we can adapt this linear model to handle a binary response – known as logistic regression.

Suppose that $\mathbf{y}_i$ are independent Bernoulli random variables with $P(\mathbf{y}_i = 1) = \phi(x_i^T \beta)$ and ϕ is the logistic transformation:

$$\phi(u) = \frac{e^u}{1 + e^u}$$

So

$$P(\mathbf{y}_i = 1) = \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}}$$

and course

$$P(\mathbf{y}_i = 0) = 1 - P(\mathbf{y}_i = 1) = \frac{1}{1 + e^{x_i^T \beta}}$$

The logistic transformation is a clever way to take a linear model whose range is $\Re$ and maps this into the range $[0, 1]$. There may be other theoretical justifications for this choice but this mapping makes it a practical model for binary responses. The space where the linear model lives I call the *logit space* and the transformation to probabilities are the fitted values. Note that for the true parameter vector

$$E(y_i) = \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}}$$

But when β is found by the MLE both $\hat{\beta}$ and this predicted value will *not* be unbiased.

Problem 1

Let $P_i = \frac{e^{x_i^T\beta}}{1+e^{x_i^T\beta}}$ to streamline notation. (Dependence on β is hidden.) Assume that $\mathbf{y}$ is a vector of biary repsonses observed according these probabilities.

The log likelihood for these data is:

$$\ell(\mathbf{y}, \beta) = \sum_{i=1}^{n} log(P_i)\mathbf{y}_i + log(1 - P_i)(1 - \mathbf{y}_i)$$

where I have used a standard notational trick to handle the [0,1] response.

Show that the score vector is

$$S(\beta) = \sum_{i=1}^{n} (\mathbf{y}_i - P_i)\mathbf{x}_i$$

or in matrix notation

$$S(\beta) = X^T(y - P)$$

Note in your work that simple algebra gives $log(P_i/(1 - P_i)) = x_i^T\beta$ (aka as the log odds ratio).

Proof. We know that the score vector is the partial derivative of the log-likelihood function with respect to β.

$$S(\beta) = \frac{\partial \ell}{\partial \beta}$$

Therefore, the score vector is

$$S(\beta) = \frac{\partial}{\partial \beta}\left[\sum_{i=1}^{n} \log(P_i)y_i + \log(1 - P_i)(1 - y_i)\right]$$

In order to compute the partial derivative, we first algebraically simplify the expression inside the summation term. We start by expanding the summation term by distributing. As noted, we also know that $log(P_i/(1 - P_i)) = x_i^T\beta$, which we can substitute into the expression.

$$\begin{aligned}\sum_{i=1}^{n} \log(P_i)y_i + \log(1 - P_i)(1 - y_i) &= \sum_{i=1}^{n} \log(P_i)y_i - \log(1 - P_i)y_i + \log(1 - P_i)\\ &= \sum_{i=1}^{n} \log(\frac{P_i}{1 - P_i})y_i + \log(1 - P_i)\\ &= \sum_{i=1}^{n} x_i^T\beta y_i + \log(1 - P_i)\end{aligned}$$

Again, we can use basic algebraic properties of logarithms to simplify this expression further. We know that $1 - P_i = \frac{1}{1+e^{x_i^T\beta}}$, so we have

$$\begin{aligned}\sum_{i=1}^{n} x_i^T \beta y_i + \log(1-P_i) &= \sum_{i=1}^{n} x_i^T \beta y_i + \log\left(\frac{1}{1+e^{x_i^T\beta}}\right)\\ &= \sum_{i=1}^{n} x_i^T \beta y_i + \log(1) - \log(1+e^{x_i^T\beta})\\ &= \sum_{i=1}^{n} x_i^T \beta y_i - \log(1+e^{x_i^T\beta})\end{aligned}$$

Now that the expression has been simplified, we can easily differentiate to derive the score function for β.

$$\begin{aligned}S(\beta) = \frac{\partial \ell}{\partial \beta} &= \frac{\partial}{\partial \beta}\sum_{i=1}^{n} x_i^T \beta y_i - \log(1+e^{x_i^T\beta})\\ &= \sum_{i=1}^{n} x_i^T y_i - \left(\frac{1}{1+e^{x_i^T\beta}}\right) x_i e^{x_i^T\beta}\\ &= \sum_{i=1}^{n} x_i^T y_i - \left(\frac{e^{x_i^T\beta}}{1+e^{x_i^T\beta}}\right) x_i\\ &= \sum_{i=1}^{n} x_i^T y_i - P_i x_i\\ &= \sum_{i=1}^{n} (y_i - P_i) x_i\\ &= X^T(y-P)\end{aligned}$$

$\square$.

Problem 2

- Show that the Fisher information has the (amazingly) simple expression (after some algebra):

$$I(\beta) = X^T V(\beta) X$$

where $V(\beta)$ is a diagonal matrix with $V_{i,i} = P_i(1-P_i)$.

Proof. From (1), we know that the score vector $S(\beta) = X^T(y-P)$. In order to derive the Fisher information matrix, we first take the partial derivative of $S(\beta)$ with respect to β, that is

$$\frac{\partial S}{\partial \beta} = \frac{\partial}{\partial \beta}\sum_{i=1}^{n}(y_i - P_i)x_i$$

First, we can start by expanding the equation.

$$\frac{\partial}{\partial \beta}\sum_{i=1}^{n}(y_i - P_i)x_i = \frac{\partial}{\partial \beta}\sum_{i=1}^{n} x_i^T y_i - P_i x_i$$

Note that y_i is not dependent on β, so the derivative of $x_i^T y_i$ is simply 0. We apply the derivative and the expression becomes

$$\frac{\partial}{\partial\beta}\sum_{i=1}^{n} x_i^T y_i - P_i x_i = \sum_{i=1}^{n} -\frac{\partial}{\partial\beta} P_i x_i$$

We can expand this further by substituting for P_i.

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

We can use u substitution and the quotient rule of differentiation. The expression becomes

$$\begin{aligned}
\sum_{i=1}^{n} -\left(\frac{(1+e^u)u'e^u - e^u u' e^u}{(1+e^u)^2}\right) x_i \\
= \sum_{i=1}^{n} -\left(\frac{u'e^u}{(1+e^u)^2}\right) x_i \\
= \sum_{i=1}^{n} -x_i^T\left(\frac{e^{x_i^T\beta}}{(1+e^{x_i^T\beta})^2}\right) x_i
\end{aligned}$$

With some simple algebra, we can rewrite this expression as

$$\begin{aligned}
\sum_{i=1}^{n} -x_i^T\left(\frac{e^{x_i^T\beta}}{(1+e^{x_i^T\beta})}\right)\left(\frac{1}{(1+e^{x_i^T\beta})}\right) x_i \\
\rightarrow \frac{\partial S}{\partial\beta} = \sum_{i=1}^{n} -x_i^T P_i\,(1-P_i)\,x_i
\end{aligned}$$

In matrix notation, we can rewrite this as

$$\frac{\partial S}{\partial\beta} = -X^T V(\beta) X$$

where $V(\beta)$ is a diagonal matrix with $V_{ii} = P_i(1-P_i)$, as noted above. Lastly, to derive the Fisher information matrix, we take the negative expected value of the score vector.

$$I(B) = -\mathbb{E}\left[\frac{\partial S}{\partial\beta}\right] = -\mathbb{E}\left[-X^T V(\beta) X\right] = X^T V(\beta) X$$

$\square$.

- In the standard linear model there is also σ^2. Why is this parameter not part of a logistic regression model?

  This is because in a standard linear regression model with Gaussian random variables, the response is continuous over $\mathbb{R}$ and the error term is assumed to be distributed normally with mean 0 and variance σ^2. On the other hand, the logistic regression output is a Bernoulli random variable that maps onto the closed interval [0,1] as a probability of y_i being 1 or 0. The formal variance (per Wikipedia) is $\text{Var}[y_i] = P_i(1-P_i)$, so it is determined by the probabilities

Problem 3 Key Indicators of Heart Disease

This *kaggle* dataset:

https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease contains 18 variables (9 booleans, 5 strings and 4 decimals). These data are taken from the

2020 annual CDC survey data of 400k adults related to their health status

According to the CDC, heart disease is one of the leading causes of death for people of most races in the US. About half of all Americans (47%) have at least 1 of 3 key risk factors for heart disease: high blood pressure, high cholesterol, and smoking. Other key indicator include diabetic status, obesity (high BMI), not getting enough physical activity or drinking too much alcohol.

Read in the data and some wrangling.

For a simpler analysis reduce the dataset by approxiamtely 50% to just female respondents and look at *Smoking* and *BMI*.

```
load( "heart2020.rda")
cardioF<- heart2020[heart2020$Sex=="Female",]
y<- ifelse(cardioF$HeartDisease =="Yes",1,0 )
Smoke<- ifelse(cardioF$Smoking =="Yes",1,0 )
BMI <- cardioF$BMI
int <- 1
X<- cbind( int, BMI, Smoke)
colnames(X) <- c("int", "BMI", "Smoke")
```

Fitting the logisitic model using `glm`

Fitting these data with the covariates Smoking and BMI.

```
fitCardio<- glm( y ~ X-1, family=binomial())
betaHat<- fitCardio$coefficients
summary(fitCardio)$coefficients
```

```
##          Estimate  Std. Error   z value      Pr(>|z|)
## Xint   -3.6108047 0.040382724 -89.41459  0.000000e+00
## XBMI    0.0239840 0.001302271  18.41705  9.588241e-76
## XSmoke  0.6520763 0.019635279  33.20942 7.870535e-242
```

Here the estimates for β are the MLEs and the standard errors reported are based on the Fisher information matrix evaluated at the MLEs.

- Transform the fitted linear model into predicted probabilities using X and $\hat{\beta}$ and verify they are the same as `fitCardio$fitted.values`

  ```
  yhat <- X %*% betaHat
  probabilities <- exp(yhat) / (1+ exp(yhat)) # logistic transformation

  print(probabilities[1:5])
  ```

```
## [1] 0.07171798 0.04216964 0.04608214 0.04555785 0.09395181
```

```
print(fitCardio$fitted.values[1:5])
```

```
##          1          2          3          4          5
## 0.07171798 0.04216964 0.04608214 0.04555785 0.09395181
```

```
if(all(probabilities == fitCardio$fitted.values)){
  print("Predicted probabilities match fitted values.")
}
```

```
## [1] "Predicted probabilities match fitted values."
```

- Based on your Fisher information derived from Problem 2 verify that you get the same standard errors reported by the `summary` function for the `glm` fit.

To compute your Fisher information do not create a large diagonal matrix as you will run out of memory! Instead use the tricksy short cut:

```
U<- sqrt((V)*X)
Info<- t(U)%*%U
```

where `V` is a vector of the diagonal elements

```
fitCardio<- glm( y ~ X-1, family=binomial())
betaHat<- fitCardio$coefficients
summary(fitCardio)$coefficients
```

```
##          Estimate  Std. Error   z value      Pr(>|z|)
## Xint   -3.6108047 0.040382724 -89.41459  0.000000e+00
## XBMI    0.0239840 0.001302271  18.41705  9.588241e-76
## XSmoke  0.6520763 0.019635279  33.20942 7.870535e-242
```

```
yhat <- X %*% betaHat
probabilities <- exp(yhat) / (1+ exp(yhat)) # logistic transformation

V <- diag(probabilities * (1 - probabilities))  # Diagonal V matrix
U <- sqrt(V) * X
Info <- t(U) %*% U

print(Info)
```

```
##               int       BMI      Smoke
## int    11171.536  314617.7   4162.838
## BMI   314617.671 9383368.1 118287.089
## Smoke   4162.838  118287.1   4162.838
```

```
InfoInv <- solve(Info) # I^-1(beta)
print(InfoInv)
```

```
##                  int           BMI         Smoke
## int    0.0016443610 -5.360770e-05 -1.210976e-04
## BMI   -0.0000536077  1.913712e-06 -7.704532e-07
## Smoke -0.0001210976 -7.704532e-07  3.832108e-04
```

```
SEs <- sqrt(diag(InfoInv)) # Sqrt of diagonal to derive standard errors

### The estimates for the standard error via the Fisher Info matrix are close to the glm fit, but not e
summary(fitCardio)$coefficients
```

```
##          Estimate  Std. Error   z value      Pr(>|z|)
## Xint   -3.6108047 0.040382724 -89.41459  0.000000e+00
## XBMI    0.0239840 0.001302271  18.41705  9.588241e-76
## XSmoke  0.6520763 0.019635279  33.20942 7.870535e-242
```

```
print(SEs)
```

```
##        int        BMI      Smoke
## 0.04055072 0.00138337 0.01957577
```

Problem 4 Fisher method of scoring.

A numerical iterative method to find the MLEs in logistic regression is *Fisher method of scoring* – a variation on Newton's method for root finding.

Here is the algorithm where below L indexes the iteration number.

0 β^0 Initial estimate for β

1 Update Score Vector: $S(\beta^L)$

2 Update Information Matrix: $I(\beta^L)$

3 Update β: $\beta^{L+1} = \beta^L + I(\beta^L)^{-1}S(\beta^L)$

Repeat last three steps until "convergence", e.g. $\|\beta^{L+1} - \beta^L\| \leq \epsilon$ and ϵ is say 1e-6.

- Using a for loop apply the Fisher method of scoring to the Cardio data and verify you get the same MLEs as from the glm function.

```r
betaL <- as.vector(c(0,0,0)) # Init beta

epsilon <- 1e-6 # Error tolerance

for (i in 1:100000) {
  probabilities <- exp(X%*%betaL) / (1+ exp(X%*%betaL)) # Update logistic transformation

  s_beta <- t(X)%*%(y-probabilities) # Update score vector

  V <- diag(probabilities * (1 - probabilities))  # Diagonal V matrix
  U <- sqrt(V) * X
  Info <- t(U) %*% U # Update Info matrix
  InfoInv <- solve(Info) # Invert

  beta_last <- betaL # betaL
  betaL <- betaL + solve(Info) %*% s_beta # betaL+1

  if (sqrt(sum((betaL - beta_last)^2)) <= epsilon ) { #Tolerance condition
    break
  }

}

### These are estimating the same beta ###
print(betaL)
```

```
##               [,1]
## int   -3.61080464
## BMI    0.02398399
## Smoke  0.65207625
```

```r
print(betaHat)
```

```
##       Xint       XBMI     XSmoke
## -3.6108047  0.0239840  0.6520763
```

- In the scoring algorithm multiply by X:

$$X\beta^{L+1} = X\beta^L + XI(\beta^L)^{-1}S(\beta^L)$$

Using this form show that Fisher scoring is the same as:

1 Compute predicted values $P_i = \phi(x_i^T\beta^L)$

2 Compute weights $W_i = P_i(1 - P_i)$

3 Compute pseudo data $z_i = X_i^T\beta^L + (1/W_i)(\mathbf{y}_i - P_i)$

4 β^{L+1} is obtained by weighted least squares applied to **z** and with weights, **W**

that is

`fitL<- lm( z ~ X - 1, weights=W)`

```r
betaL2 <- as.vector(c(0,0,0)) # Init beta



for (i in 1:100000) {
  P <- exp(X%*%betaL2) / (1+ exp(X%*%betaL2)) # Update P vector

  W <- P * (1-P) # Update weight vector

  z <- X%*%betaL2 + (y-P) / W # Psuedo data

  fitL <- lm( z ~ X - 1, weights=W) # Fitting WLS

  beta_last <- betaL2 # Old beta
  betaL2 <- coef(fitL) # Coefficients of WLS estimate

  if (sqrt(sum((betaL2 - beta_last)^2)) <= epsilon ) { # Tolerance condition
        break
  }
}

### This method is the same as the Fisher scoring algorithm ###
print(betaL2)
```

```
##       Xint       XBMI     XSmoke
## -3.6108048  0.0239840  0.6520763
```

```r
print(betaL)
```

```
##              [,1]
## int   -3.61080464
## BMI    0.02398399
## Smoke  0.65207625
```