STAT0030 Lab 8: Numerical methods for GLMs

In the previous workshop, you looked at numerical methods for minimising (or equivalently maximising) a function. In statistics, this problem arises most often in the context of likelihood-based inference. You looked at some general-purpose optimisation methods, and saw that the R function `nlm` can be used to maximise likelihoods and to approximate the Fisher information matrix.

The disadvantage of general-purpose algorithms is that they may be inefficient in comparison with schemes that are designed specifically for a particular problem, or class of problems. In this workshop we will look at likelihood-based inference for *Generalized Linear Models* (GLMs).

1 GLMs — revision

You have studied GLMs in your course *STAT0028 Statistical Models & Data Analysis* or *STAT0032 Introduction to Statistical Data Science*. They are used when we wish to study the relationship between some variable of interest, Y say, and some collection of explanatory variables $x_1, \ldots, x_p$. The data consist of pairs $(Y_1, \mathbf{x}_1), \ldots, (Y_n, \mathbf{x}_n)$ say, where Y_i is the response for the ith case in our dataset and $\mathbf{x}_i = (x_{i,1} \ \ldots \ x_{i,p})^T$ is the corresponding vector of explanatory variables. The important things to remember are:

- A GLM says that given $\mathbf{x}_i$, Y_i has some probability distribution with mean μ_i, such that $g(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta} = \sum_{j=1}^p \beta_j x_{i,j}$ $(= \eta_i$, say) for some coefficient vector $\boldsymbol{\beta} = (\beta_1 \ \ldots \ \beta_p)^T$ and monotonic, differentiable function $g(\cdot)$ (the LINK FUNCTION). η_i is the LINEAR PREDICTOR for Y_i.

- The distributions of the Ys are in the EXPONENTIAL FAMILY: Y_i has density of the form

$$f(y_i; \theta_i, \phi) = \exp\left[\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi)\right] , \tag{1}$$

  for some parameters θ_i and ϕ, and functions $b(.)$ and $c(.,.)$. This family contains many standard distributions. The mean is $\mu_i = \partial b/\partial\theta_{|\theta=\theta_i}$ and the variance is $V_i = \phi\partial^2 b/\partial\theta^2_{|\theta=\theta_i}$ $= \phi\partial\mu/\partial\theta_{|\theta=\theta_i}$. ϕ is the DISPERSION PARAMETER, and is the same for all i. In your STAT0028/STAT0032 course, you had $a_i(\phi)$ instead of ϕ; however, the as are not necessary for our purposes.

- If $g(\cdot)$ is the identity, and we specify a normal distribution for Y_i, we end up with a standard linear regression model — hence (1) gives us a generalised linear model. If we write $N(\mu, \sigma^2)$ in 'exponential family' form, we find $\theta = \mu$ and $\phi = \sigma^2$ — so in this case, the dispersion parameter is just the variance and the assumption of a common ϕ is just the usual assumption of constant variance.

2 Maximum likelihood estimation for GLMs

When fitting a GLM to data, the aim is to estimate the coefficient vector $\boldsymbol{\beta}$ and, if necessary, the dispersion parameter ϕ. In the context of exponential families, $\boldsymbol{\beta}$ can be regarded as a

function of θ (and vice versa), since both are related to the mean of the distribution. The log-likelihood for $\boldsymbol{\beta}$ and ϕ is therefore

$$\ell\left(\boldsymbol{\beta}, \phi|\mathbf{y}\right) = \sum_{i=1}^{n}\left[\frac{y_i\theta_i - b(\theta_i)}{\phi} + c(y_i, \phi)\right] \ . \tag{2}$$

Differentiating with respect to β_j $(j = 1, \ldots, p)$ and setting to zero yields the p SCORE EQUATIONS

$$U_j \text{ (say)} \ = \frac{\partial \ell}{\partial \beta_j} = \frac{1}{\phi}\sum_{i=1}^{n}\frac{\partial}{\partial \beta_j}\left[y_i\theta_i - b(\theta_i)\right] \ = 0 \qquad (j = 1, \ldots, p) \ . \tag{3}$$

Write $V_i = V(\mu_i)\phi$, where $V(\mu_i)$ is the variance function. It can be shown (STAT0028 notes) that

$$U_j = \sum_{i=1}^{n}\left[\frac{y_i - \mu_i}{V(\mu_i)}\left(\frac{\partial \mu_i}{\partial \eta_i}\right)x_{i,j}\right] \ = \ \sum_{i=1}^{n}\left[\frac{y_i - \mu_i}{V(\mu_i)g'\left(\mu_i\right)}x_{i,j}\right] \ . \tag{4}$$

The solution of the score equations (which, from (3), clearly does not depend on ϕ) yields the maximum likelihood estimate of $\boldsymbol{\beta}$. However, the solution must be obtained numerically in all but the simplest cases. Assembling all p equations into vector form, we seek the solution of

$$\mathbf{U}\left(\boldsymbol{\beta}\right) = \mathbf{0} \tag{5}$$

where $\mathbf{U}\left(\boldsymbol{\beta}\right) = \left(U_1, \ldots, U_p\right)^T$ is the SCORE VECTOR of log-likelihood derivatives.

The Newton-Raphson algorithm may be used to solve these equations: start with an initial guess at the solution, $\boldsymbol{\beta}^{(0)}$ say, and then successively calculate

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \left[\frac{\partial \mathbf{U}}{\partial \boldsymbol{\beta}}_{|\boldsymbol{\beta}^{(t)}}\right]^{-1}\mathbf{U}\left(\boldsymbol{\beta}^{(t)}\right) \tag{6}$$

until convergence is achieved. $\partial\mathbf{U}/\partial\boldsymbol{\beta}$ here is the $p \times p$ matrix of second derivatives of the log-likelihood with respect to $\boldsymbol{\beta}$. This is actually Newton-Raphson in p dimensions. In Lab 7 you looked at a 1-dimensional example and derived the formula

$$x_{i+1} = x_i - \left[h''(x_i)\right]^{-1}h'(x_i)$$

which you can recognise (hopefully!) as a simplified version of (6), although the notation is different.

Conventionally, when fitting GLMs we replace the matrix of second derivatives in (6) by its expected value $-\mathbf{I}\left(\boldsymbol{\beta}\right)$, where $\mathbf{I}\left(\boldsymbol{\beta}\right)$ is the FISHER INFORMATION MATRIX. The resulting algorithm is called the METHOD OF SCORING. It can be shown that it is a perfectly legitimate alternative to (6), so long the initial value $\boldsymbol{\beta}^{(0)}$ is chosen sensibly. In practice, you have to choose *extremely* poor starting values for the scoring method to fail! Moreover, if the link function $g(\cdot)$ is such that $g(\mu_i) = \theta_i$ in the exponential family, it can be shown that the scoring method is exactly the same as (6). Such a link function is called a CANONICAL LINK.

The scoring algorithm is particularly convenient because the information matrix has an alternative representation as the covariance matrix of the score vector (you may have covered

this in course STAT0008 *Statistical Inference*). Together with (4), this can be used to show (see STAT0028 notes) that the information matrix is

$$\mathbf{I}(\boldsymbol{\beta}) = \mathbf{X}^T \mathbf{W} \mathbf{X} \ , \tag{7}$$

where $\mathbf{X}$ is an $n \times p$ matrix whose (i,j)th entry is $x_{i,j}$, and $\mathbf{W}$ is a diagonal $n \times n$ matrix with elements $w_{ii} = (\partial \mu_i / \partial \eta_i)^2 / V(\mu_i) = [g'(\mu_i)]^{-2} / V(\mu_i)$ (which depend on $\boldsymbol{\beta}$). The iterative scheme for the scoring method is therefore

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \left[\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X} \right]^{-1} \mathbf{U}\left(\boldsymbol{\beta}^{(t)}\right) \ . \tag{8}$$

Multiplying both sides by $\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X}$ gives

$$\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X} \boldsymbol{\beta}^{(t+1)} = \mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X} \boldsymbol{\beta}^{(t)} + \mathbf{U}\left(\boldsymbol{\beta}^{(t)}\right) \ . \tag{9}$$

Noting that $\mathbf{X}\boldsymbol{\beta}^{(t)} = \boldsymbol{\eta}^{(t)}$, the vector of linear predictors at iteration t, and that $\mathbf{U}\left(\boldsymbol{\beta}^{(t)}\right)$ can itself be written as a vector product involving the matrix $\mathbf{X}^T \mathbf{W}^{(t)}$ (from (4)), the scoring algorithm can finally be written in matrix form as

$$\left[\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X}\right] \boldsymbol{\beta}^{(t+1)} = \mathbf{X}^T \mathbf{W}^{(t)} \mathbf{z}^{(t)} \quad \Rightarrow \quad \boldsymbol{\beta}^{(t+1)} = \left[\mathbf{X}^T \mathbf{W}^{(t)} \mathbf{X}\right]^{-1} \mathbf{X}^T \mathbf{W}^{(t)} \mathbf{z}^{(t)} \ , \tag{10}$$

where $\mathbf{z}^{(t)}$ is an $n \times 1$ vector whose ith element is

$$z_i^{(t)} = \eta_i^{(t)} + \left(y_i - \mu_i^{(t)}\right) \left(\frac{\partial \eta}{\partial \mu}_{|\mu_i^{(t)}} \right) = \eta_i^{(t)} + \left(y_i - \mu_i^{(t)}\right) g'\left(\mu_i^{(t)}\right) \ .$$

Equation (10) in fact gives the solution of the weighted least-squares regression of $\mathbf{z}^{(t)}$ upon $\mathbf{X}$, with weights contained in the diagonal elements of $\mathbf{W}^{(t)}$. For this reason, $\mathbf{z}^{(t)}$ is sometimes called the ADJUSTED DEPENDENT VARIATE. The need for iteration arises because $\mathbf{z}$ and $\mathbf{W}$ both depend, in general, upon $\boldsymbol{\beta}$. Expressed in this form, the algorithm for fitting GLMs is referred to as ITERATIVE WEIGHTED LEAST SQUARES (IWLS). A side effect of the algorithm is that for large samples, the covariance matrix of the parameter estimates can be calculated easily from the information matrix (7); this enables us to derive standard errors for the estimates. Compare this with general-purpose routines such as `nlm`, in which the Hessian is an *approximation* to the information matrix.

3 Summary of the algorithm

The exposition above is perhaps not very helpful if you're about to sit down and write a program to fit a GLM. So here are the steps:

1. Assemble your explanatory variables into an $n \times p$ matrix $\mathbf{X}$.
2. Choose a sensible starting value $\boldsymbol{\beta}^{(0)}$.
3. For the current estimate of $\boldsymbol{\beta}$, calculate the vector $\boldsymbol{\eta}$ of linear predictors, the vector $\boldsymbol{\mu}$ of means and the vector $\mathbf{V}$ of variances (each of these vectors has an element for each case in the dataset). If the model involves an unknown dispersion parameter ϕ, set it to 1 for the moment (since we saw earlier that the MLE of $\boldsymbol{\beta}$ doesn't depend on the value of ϕ).

4. Calculate the diagonal elements of $\mathbf{W}$. Since all the off-diagonal elements are zero, you can save memory by storing it as a vector with n elements, rather than as a matrix with n^2. The ith element is $\left[g'\left(\mu_i\right)\right]^{-2}/V_i$.

5. Calculate the vector $\mathbf{z}$, with ith element $\eta_i + \left(y_i - \mu_i\right) g'\left(\mu_i\right)$.

6. Calculate the $p \times n$ matrix $\mathbf{X}^T\mathbf{W}$. To do this, multiply each row of $\mathbf{X}$ by the corresponding diagonal element of $\mathbf{W}$, and transpose the result. You can't use matrix arithmetic here, because you've stored $\mathbf{W}$ as a vector. However, you *can* use element-wise arithmetic (`t(W * X)`), because the elements of $\mathbf{W}$ get duplicated as many times as necessary, and the elements of $\mathbf{X}$ are stored in column order.

7. Calculate the $p \times p$ matrix $\mathbf{X}^T\mathbf{W}\mathbf{X}$, and invert it to obtain $\left[\mathbf{X}^T\mathbf{W}\mathbf{X}\right]^{-1}$. Also calculate the $p \times 1$ vectors $\mathbf{X}^T\mathbf{W}\mathbf{z}$ and $\mathbf{U} = \mathbf{X}^T\mathbf{W}\left(\mathbf{z} - \boldsymbol{\eta}\right)$. Here you *can* use matrix arithmetic.

8. If all elements of $\mathbf{U}$ are sufficiently close to zero, go to step 9 ($\mathbf{U}$ is the score vector, so we've converged when it reaches zero). Otherwise, recalculate $\boldsymbol{\beta}$ as $\left[\mathbf{X}^T\mathbf{W}\mathbf{X}\right]^{-1}\mathbf{X}^T\mathbf{W}\mathbf{z}$ and go back to step 3.

9. If the model involves a dispersion parameter, estimate it (this is usually done via a method of moments — we won't go into details here). Denote the estimate by $\hat{\phi}$.

10. Extract the diagonal elements of $\left[\mathbf{X}^T\mathbf{W}\mathbf{X}\right]^{-1}$, multiply by $\hat{\phi}$ and take the square roots, to obtain standard errors for the parameter estimates.

11. Output the results.

Most software packages implement this algorithm with a small improvement, which gets round the need to choose a starting value $\boldsymbol{\beta}^{(0)}$. This sounds too good to be true; notice, however, that in the first iteration $\boldsymbol{\beta}^{(0)}$ is only used to calculate $\boldsymbol{\eta}$ in step 3. Therefore, instead of specifying $\boldsymbol{\beta}_0$, for this first iteration we could use the data $\mathbf{y}$ as an estimate of $\boldsymbol{\mu}$, calculate the corresponding vector $\boldsymbol{\eta}$ and use this throughout the remainder of steps 4–8 (in practice, it may be necessary to make small adjustments to 'awkward' elements of $\boldsymbol{\mu}$ — such as zeroes for a Poisson distribution, for example). The first explicit value for $\boldsymbol{\beta}$ will then be calculated in step 8. Note that the score vector $\mathbf{U}$ will be calculated incorrectly as zero in step 7 of this first iteration; hence the algorithm should be forced always to return to step 3 at the end of this iteration. You may be glad to know that we will *not* attempt anything so sophisticated as this, however!

4 Example

We will illustrate the algorithm by applying it to a very simple problem whose answer is known: the estimation of a Poisson mean. The data for this exercise can be downloaded from the STAT0030 web page on Moodle. The file `nstorms.dat` contains annual numbers of tropical storms in the North-West Pacific from 1959 to 2000. The first column is `Year`; the remaining three columns are storm counts, for different strengths of storm. The file `IWLS.r` contains an R function to implement the IWLS algorithm for this problem. Download both files, save them to your STAT0030 working directory, and use R (or RStudio).

4.1 Preliminary analysis

We'll focus on the `Storms` counts throughout this exercise. To get a feel for the data, let's start by producing some useful plots and summaries. Use `read.table` to read the data into R, and assign it to a data frame called `storm.data`. Produce a plot showing the number of storms in each year (i.e. a line graph with `Year` on the x-axis and `Storms` on the y-axis). Calculate the mean and variance of `Storms`.

4.2 A simple GLM formulation

We will treat the `Storms` counts as i.i.d. observations from a Poisson distribution with mean μ (**Exercise:** is this reasonable?). Write $\beta = \ln \mu$, and consider the estimation of β from the observed Ys. The MLE of β is clearly $\ln \overline{Y}$ (**Exercise:** verify that this is equal to 3.309848). However, we can view this as a very simple GLM:

- There is only one explanatory variable x_1, which always takes the value 1. Hence, in the notation of the previous section, $p = 1$ and $\mathbf{X}$ is an $n \times 1$ matrix of 1s.
- The link function $g(\cdot)$ is the logarithm: $\ln \mu = \beta$.
- The Poisson distribution is in the exponential family. It has 'density' $f(y) = e^{-\mu}\mu^y/y! = \exp[y \ln \mu - \mu - \ln y!]$, so that $\theta = \ln \mu$ and $\phi = 1$. Note that $g(\mu) = \theta$ so that we have a canonical link function. The variance of the distribution is $V = \mu$.

4.3 The log-likelihood, score and information functions

The log-likelihood for β is

$$\sum_{i=1}^{n} [y_i \ln \mu - \mu - \ln y_i!] = \sum_{i=1}^{n} \left[y_i\beta - e^{\beta} - \ln y_i!\right]$$

The first and second derivatives with respect to β yield the score and information functions:

$$U(\beta) = \sum_{i=1}^{n} \left[y_i - e^{\beta}\right] \qquad \text{and} \qquad I(\beta) = -E\left[\frac{\partial U}{\partial \beta}\right] = E\left[\sum_{i=1}^{n} e^{\beta}\right] = ne^{\beta} \ .$$

Questions: *(i)* why does the score function $U(\beta)$ depend on the data only through $\sum_{i=1}^{n} y_i$? *(ii)* what feature of this problem ensures that, aside from n, the second derivative of the log-likelihood doesn't involve the data at all? If unsure, ask.

Use the code below to produce a plot of the log-likelihood, score and information functions for β in the range $(3, 3.6)$ (you can omit the comments!). Since any normal human will make some typing errors at the first attempt, I suggest you type into an editor and paste the code into R as necessary (you may also care to save it when you've finished). There are a couple of points where you have to fill in the appropriate bits yourself.

```
#
# Define a grid of values for beta, and allocate some storage
# for the log-likelihood
#
```

```
beta <- seq(3,3.6,0.01)
logl.beta <- vector("numeric",length(beta))
#
# Evaluate the log-likelihood, score and information functions
# for each value of beta. We can't easily avoid a loop for the
# log-likelihood, but the other 2 functions can each be
# evaluated in a single vector operation (which is more efficient)
#
for (i in 1:length(beta)) {
  logl.beta[i] <- sum(log(dpois(storm.data$Storms,exp(beta[i]))))
}
n <- length(storm.data$Storms)
ysum <- sum(storm.data$Storms)
u.beta <- DEFINE THE SCORE FUNCTION HERE (FORMULA IS GIVEN ABOVE)
i.beta <- DEFINE THE INFORMATION FUNCTION HERE (       "       )
#
# Set up a graphics screen with space for 3 plots, and plot.
# Prepare to be impressed with the axis labels!
#
par(mfrow=c(3,1))
plot(beta,logl.beta,type="l",xlab=expression(beta),
     ylab=expression(paste("ln L",(beta))),
     main="Log-likelihood for tropical storm data")
ADD YOUR OWN CODE TO PLOT THE SCORE (u.beta), ALSO SHOWING THE LINE U=0
(abline(0,0) WILL DO THIS). THEN PLOT THE INFORMATION (i.beta).
```

Notice that the score function looks roughly linear over this range of β values, and the information function varies relatively slowly (from about 900 to about 1500). This implies that the log-likelihood is roughly quadratic. Hence, since the Newton-Raphson and scoring algorithms are based on a quadratic approximation to the log-likelihood (or equivalently, a linear approximation to the score function), they should perform well in this problem.

4.4 Implementation of the IWLS algorithm

Now we'll implement the IWLS algorithm to find the MLE of β. You have downloaded the file `IWLS.r`, which contains a function `IWLS` to do this. Use the `source` command to read this function into R, and read through the code. Make sure that you understand how each line relates to the algorithm in Section 3. It may be helpful to note the following:

- Since there's only one coefficient β and $\mathbf{X}$ is a column of 1s, all the linear predictors are equal to β.
- For the Poisson distribution, the variance is equal to the mean so $V_i = \mu_i$.
- We're working with a log link: $g(\mu) = \ln \mu$ so that $g'(\mu) = \mu^{-1}$.
- For a Poisson distribution, the dispersion parameter ϕ is equal to 1 so it doesn't need to be estimated.

It may also be helpful to remember that `%*%` is the symbol for matrix multiplication, and that `t(X)` reads as 'the transpose of `X`', if `X` is a matrix object.

When you're satisfied that you understand the code, test the function using different starting values, to see if it gives the right answer. Begin your tests with sensible starting values, and try to relate the behaviour of the algorithm to your plot of the score function. Then try some silly starting values. Are there any starting values for which the algorithm fails?

4.5 Deviance

You may recall that in a GLM, models can be compared using either likelihood ratio or deviance tests. The deviance for a model is the equivalent of the residual sum of squares in a linear model, and is defined as

$$D = 2\phi \left[\ell \left(\mathbf{y}; \mathbf{y} \right) - \ell \left(\boldsymbol{\mu}; \mathbf{y} \right) \right] \ , \tag{11}$$

where:

- $\mathbf{y}$ is the vector of observed responses;
- $\boldsymbol{\mu}$ is the corresponding vector of modelled means; and
- $\ell \left(\mathbf{m}; \mathbf{y} \right)$ is the log-likelihood for $\mathbf{y}$ when the mean vector is $\mathbf{m}$.

D is also sometimes called the *residual deviance*. If you have 2 models with deviances D_1 and D_2 respectively, and model 2 is an extension of model 1 containing p extra parameters, then a formal test of model 1 against model 2 can be carried out by comparing the quantity

$$\frac{(D_1 - D_2)/p}{D_2/\nu}$$

to an $F_{p,\nu}$ distribution, where ν is the residual degrees of freedom for model 2. This is the exact equivalent of the F-test for comparing nested linear models — and hence we can construct an 'Analysis of deviance' table in the same way as we would construct an ANOVA table for linear models. The need for an F-test arises from the unknown dispersion parameter. If the dispersion parameter is known (as in the Poisson case, where we know that $\phi = 1$), an alternative is to base tests on the χ^2 distribution for the likelihood ratio statistic $\Lambda = 2(\ell_2 - \ell_1) = \phi^{-1} (D_1 - D_2)$, where ℓ_1 and ℓ_2 are the log-likelihoods for the two models. If D is the deviance for a model, $\phi^{-1} D$ is sometimes called the *scaled deviance*. If you're confused, you're in good company — it's wrong in Chapter 5 of Dobson (textbook for your G1 course), for example. Most of the time you'll only deal with GLMs whose dispersion parameter is known (e.g. in loglinear modelling and logistic regression), so you don't need to worry.

Let's calculate the deviance for the Poisson storm model. The Poisson log-likelihood for data $\mathbf{y}$ with fitted values $\mathbf{m}$ is

$$\ell(\mathbf{m}; \mathbf{y}) = \sum_{i=1}^{n} \left[y_i \ln m_i - m_i - \ln y_i! \right] \ .$$

Substitute $\ell(\mathbf{y}; \mathbf{y})$ and $\ell(\boldsymbol{\mu}; \mathbf{y})$ into equation 11, to get an expression for D (you should be able to simplify this expression). Then use R to evaluate the deviance for the model you've fitted to the storm counts.

4.6 An easier way ...

Of course, we don't want to have to write our own functions every time we want to fit a GLM. The point of the exercise above was to develop an understanding of how the IWLS algorithm works. In practice, you will be relieved to know that R has a built-in `glm` command, which works in very much the same way as the `lm` command for linear models. Try

```
summary(glm(Storms ~ 1, family = poisson(link="log"),data=storm.data))
```

Compare the output with your previous results, and check that your deviance calculation was correct.

In the next workshop, we'll explore the `glm` command in more detail. In the meantime, if you have any spare time today you should repeat the analyses above, but using a Poisson model in which $\beta = \mu$ i.e. in which the link function is the identity: $g(\mu) = \mu$. You will need to modify the `IWLS` function slightly, but note that the *only* changes are those involving $g(\mu)$ and $g'(\mu)$ — everything else remains exactly the same. How does the use of a different link affect the log-likelihood, score and information functions?