Week 4 Notes: GLMs in Practice

This week, we'll mainly be doing some open-ended applications of generalized linear models on real datasets. Some of them may involve doing some independent work on your own in figuring out how to fit the models. For each question, I would like you to consider the following issues:

- What modeling strategy is appropriate for the data?

For the purpose of downloading and cleaning the data, we will use the `tidyverse` package:

```
library(tidyverse)
```

1 Problem 1: Estimating a Generalized Linear Model from Scratch

The following question is adapted from James Scott's SDS 383D course notes (all mistakes my own).

The regression coefficients β in a GLM are typically fit using some variation on likelihood-based inference. In class, we saw that the score function is of the form

$$s(\beta, \phi) = \sum_i \frac{\omega_i (Y_i - \mu_i)}{\phi V(\mu_i)\, g'(\mu_i)} X_i.$$

which, under the canonical link, simplifies to

$$s(\beta) = \sum_i \frac{\omega_i (Y_i - \mu_i)}{\phi} X_i.$$

Let's take the specific case of logistic regression for a binomial outcome, where $Z_i \sim \text{Binomial}(N_i, \mu_i)$ for known sample size N_i and $Y_i = Z_i / N_i$ is the observed success fraction, where the canonical link is used.

A common approach to fitting GLMs via maximum likelihood (or, really, optimizing *any* function) is to use *gradient descent*, i.e., iteratively update

$$\beta \leftarrow \beta + \gamma\, s(\beta)$$

where $s(\cdot)$ is the score function and γ is a "learning rate" that is, ideally, selected automatically. Read up on the method of steepest descent[1] , i.e. gradient descent and write your own function that will fit a logistic regression by gradient descent. For extra coding brownie points, try to maintain some level of generality to your code so that it could also work with different GLMs, assuming you wrote different sub-routines.

After doing this, grab the data `wdbc.csv` from the course website, or obtain some other real data that interests you, and test out your routine (comparing it to the output of, say, `glm` in `R` to sanity check your results). The WDBC file has information on 569 breast-cancer patients from a study done in Wisconsin. The first column is a patient ID, the second column is a classification of a breast cell (Malignant or Benign), and the next 30 columns are measurements computed from a digitized image of the cell nucleus. These are things like radius, smoothness, etc. For this problem, use the first 10 features for X, i.e. columns 3-12 of the file.

Some notes:

- We're trying to maximize the log-likelihood but the convention in the optimization literature is to minimize things. No big deal; what we're doing is the same as minimizing the negative of the log-likelihood.
- You need to add an intercept term, and the simplest way is to add a column of 1's as the first column of the feature matrix $\mathbf{X}$. (If you've never seen this trick before, convince yourself why it makes sense.)
- Ensure that, at every iteration of gradient descent, you compute and store the current value of the log-likelihood so that you can track and plot the convergence of the algorithm.
- How did you select the learning rate γ? You can be as clever (or as non-clever) as you want here. Generally, smaller step sizes will be more robust, but converge slower, while larger step sizes may overshoot the solution entirely. One idea, called "line search" is to find the stepsize that itself minimizes the log-likelihood at each iteration, but this involves extra work (both in terms of coding and computations).
- How do you determine (automatically) that you have reached a reasonable stopping point? One possibility is to stop when $s(\beta)$ is "small enough" but you will need to be precise about what this means.

2 Problem 2: Estimating a Generalized Linear Model from Scratch: 2nd Order Methods

Gradient descent is known as a *first order* method for optimizing a function: to implement it, we only need to evaluate the function itself and its partial derivatives. A more powerful (but less generally applicable) approach is to use a *second order* method, which also makes use of the second derivatives, i.e., the Hessian matrix of the log-likelihood (or similar). Equivalently, the Hessian is the *negative* of the observed Fisher information, which we derived in class.

a. Consider a point $b \in \mathbb{R}^P$ that serves as an intermediate guess for β. Show that, for any GLM, the second-order Taylor approximation of the log-likelihood around b can be expressed in the form

$$q(\beta, b) = -\frac{1}{2}(\widehat{\mathbf{Y}} - \mathbf{X}\beta)^\top W(\widehat{\mathbf{Y}} - \mathbf{X}\beta) + c,$$

where W is a diagonal matrix of "working weights" and $\widehat{\mathbf{Y}}$ is a vector of "working responses", and c is a constant that doesn't involve β. What is the matrix W given by, and what are its diagonal entries?

b. Read up on Newton's method for optimizing smooth functions (e.g., in Nocedal and Wright, Chapter 2). In the above case, notice that Newton's method just corresponds to optimizing $q(\beta, \beta_t)$ with respect to β in order to obtain the next value of β_{t+1}, and that this optimization is essentially a weighted least-squares problem. Implement it for the logistic regression model and test it out on the same data set you just used to test out gradient descent. Note: while you could do linear search, there is a "natural" step size of 1 in Newton's method. Verify that your numerical solution replicates the β estimate you get when using a package solver, e.g., the `glm` function in `R`, up to minor numerical differences.

Note: This algorithm is referred to as *iteratively reweighted least squares* (IRLS). A neat trick that makes IRLS very nice is that, in addition to avoiding specifying a step size, there is also a very simple way to initialize the algorithm: just set $\widehat{\mathbf{Y}} = \mathbf{Y}$.

3 Problem 3: The Crabs Data

The following two datasets are from the textbook *Categorical Data Analysis* by Alan Agresti. The first dataset looks at the mating behavior of crabs during spawning season. Roughly speaking, during spawning season some number of "satellite" males cluster around females and try to fertilize their eggs. The number of satellites a female attracts depends on a number of other traits of the crab including her size, health, and color. The dataset can be loaded by running the following code:

```
crabs_file <- str_c("https://raw.githubusercontent.com/alanagresti/",
                    "categorical-data/master/Crabs.dat")
crabs <- read_table(crabs_file) %>%
  mutate(spine = factor(spine), color = factor(color))
```

```
-- Column specification --------------------------------------------------------
cols(
  crab = col_double(),
  sat = col_double(),
  y = col_double(),
  weight = col_double(),
  width = col_double(),
  color = col_double(),
  spine = col_double()
)
```

```
print(head(crabs))
```

```
# A tibble: 6 x 7
   crab   sat     y weight width color spine
  <dbl> <dbl> <dbl>  <dbl> <dbl> <fct> <fct>
1     1     8     1   3.05  28.3 2     3
2     2     0     0   1.55  22.5 3     3
3     3     9     1   2.3   26   1     1
4     4     0     0   2.1   24.8 3     3
5     5     4     1   2.6   26   3     3
6     6     0     0   2.1   23.8 2     3
```

The dataset contains the following columns:

- `crab`: an index for each crab (not really useful for us).
- `sat`: the number of satellites of each crab.
- `y`: 1 if the crab has a satellite, 0 otherwise.
- `width`: the width of the crab carapice in centimeters.
- `weight`: the weight in kilograms of the crab.
- `color`: A factor with levels 1 = light medium, 2 = medium, 3 = dark medium, 4 = dark.
- `spine`: A factor with levels 1 = both good, 2 = one worn or broken, 3 = both worn or broken.

a. For each of the variables width, weight, color, and spine, make a prediction about how these features relate to the number of satellites within the population of all crabs.

b. Fit a Poisson loglinear model (could be as simple or complex as you want in terms of, e.g., interactions, but be prepared to defend any choices you make). Then, interpret the estimated coefficients in terms of what they say about the relationship between each predictor and the number of satellites.

c. Use the loglinear model to assess each of the predictions you made, both in terms of whether the data supports (or fails to support) the predictions and the strength of the evidence in terms of uncertainty quantification.

d. A possible limitation of the Poisson loglinear model is that count data is often *overdispersed* with respect to the Poisson distribution, i.e., we tend to see $\operatorname{Var}(Y_i \mid X_i = x) > \mathbb{E}(Y_i \mid X_i = x)$. The main consequence of overdispersion is that the model will tend to give a false sense of precision in the inferences/predictions it makes.

   Try to come up with a simple procedure for assessing whether or not the loglinear model you fit is overdispersed relative to the Poisson distribution. Then, use this to determine if there is evidence of overdispersion for the crab dataset. *Hint:* there are a lot of possibilities here, but one thing to observe is that if the Poisson model is correct that we should have $\mathbb{E}\{(Y_i - \mu_i)^2/\mu_i\} = 1$.

4 Problem 4: Injury Dataset

From Agresti this dataset contains

> … results of accidents in the state of Maine for 68,694 passengers in autos and light trucks. The table classifies passengers by gender, location of accident, seat-belt use, and injury.

The columns `yes` and `no` state whether the passenger was injured in the accident.

```r
injury_file <- str_c("https://raw.githubusercontent.com/alanagresti/",
                     "categorical-data/master/Injury_binom.dat")
injury <- read_table(injury_file)[,-6]
```

```
Warning: Missing column names filled in: 'X6' [6]
```

```
-- Column specification --------------------------------------------------------
cols(
  gender = col_character(),
  location = col_character(),
  seatbelt = col_character(),
  no = col_double(),
  yes = col_double(),
  X6 = col_logical()
)
```

```r
head(injury)
```

```
# A tibble: 6 x 5
  gender location seatbelt    no   yes
  <chr>  <chr>    <chr>    <dbl> <dbl>
1 female urban    no        7287   996
2 female urban    yes      11587   759
3 female rural    no        3246   973
4 female rural    yes       6134   757
5 male   urban    no       10381   812
6 male   urban    yes      10969   380
```

In cases like this, where the data has columns with counts of "yes" and "no" rather than columns of 0s and 1s, a standard way to fit a logistic regression model is to have a response be a matrix with the success counts in the first column and failure counts in the second. For example:

```r
seatbelt_glm <- glm(cbind(yes, no) ~ seatbelt * gender,
                    data = injury,
                    family = binomial)
```

fits a logistic regression model the models logit $p = \beta_0 + \beta_1 \times \texttt{seatbelt\_yes} + \beta_2 \times \texttt{gender\_male} + \beta_3 \times \texttt{seatbelt\_yes} \times \texttt{gender\_male}$.

Use a logistic regression model(s) to attempt to answer the following questions:

a. Within the subpopulations defined by each (gender, location) combination, is there evidence that use of a seatbelt results in reduced incidents of injury? If so, can you give an estimate or this effect and an estimate about how precisely the effect is measured?

b. Is there evidence that, within each subpopulation defiend by an individual's location, the effect of using a seatbelt is different for males and females? If so, can you give an estimate of this effect (e.g., seatbelt use reduces the odds of an injury for women by a factor of XXX more/less relative to men) and an estimate of how precisely this is measured?