# Unifying Theory of GLMs (Part IV)

Rebecca C. Steorts

# Computing set up

```
library(tidyverse)
library(tidymodels)
library(knitr)
library(viridis)

knitr::opts_chunk$set(fig.width = 8,
                      fig.asp = 0.618,
                      fig.retina = 3,
                      dpt = 300,
                      out.width = "70%",
                      fig.align = "center")

ggplot2::theme_set(ggplot2::theme_bw(base_size = 16))

colors <- tibble::tibble(green = "#B5BA72")
```

# Poisson Regression

We extend the Poisson model to incorporate covariates using a generalized linear model:

$$\log(\underbrace{E(Y \mid X)}_{\lambda}) = X^T \beta,$$

where we assume the outcome is Poisson and the canonical link is the logarithm.

**Question:** Can we differentiate the log-likelihood, set it equal to zero, and solve for the MLEs of $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ as before?

# Poisson Regression Log-Likelihood

The log-likelihood is:

$$\log L = \sum_{i=1}^{n} \Big( y_i \log \lambda - \lambda - \log(y_i!) \Big) \tag{1}$$

$$= \sum_{i=1}^{n} \Big( y_i X_i^T \beta - e^{X_i^T \beta} - \log(y_i!) \Big). \tag{2}$$

In general, setting the score equations

$$\frac{\partial \log L}{\partial \beta_j} = 0,$$

leads to what we call "transcendental equations," that typically have no closed-form solution.

# Newton Raphson Method (1-D)

Newton-Raphson for root finding is based on a second-order Taylor approximation:

1. Start with an initial guess $\theta^{(0)}$.
2. Iterate:
$$\theta^{(t+1)} = \theta^{(t)} - \frac{f'(\theta^{(t)})}{f''(\theta^{(t)})}.$$
3. Stop when a convergence criterion is satisfied.

There are some necessary conditions for convergence, however, this is beyond the scope of this class. Many likelihood functions you are likely to encounter (e.g., GLMs with canonical link) will in fact converge from any starting value.

# Newton Raphon (general)

Define the score vector and Hessian for $logL(\theta \mid X)$ with parameter vector $\theta = (\theta_1, \ldots, \theta_p)$ as follows:

$$\nabla \log L = \begin{pmatrix} \frac{\partial \log L}{\partial \theta_1} \\ \vdots \\ \frac{\partial \log L}{\partial \theta_p} \end{pmatrix}, \quad \nabla^2 \log L = \begin{pmatrix} \frac{\partial^2 \log L}{\partial \theta_1^2} & \cdots & \frac{\partial^2 \log L}{\partial \theta_1 \theta_p} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \log L}{\partial \theta_p \theta_1} & \cdots & \frac{\partial^2 \log L}{\partial \theta_p^2} \end{pmatrix}.$$

Then the update is:

$$\theta^{(t+1)} = \theta^{(t)} - \left[\nabla^2 \log L(\theta^{(t)} \mid X)\right]^{-1} \nabla \log L(\theta^{(t)} \mid X).$$

# Newton Raphon (general)

We can modify the Newton-Raphson algorithms for higher dimensions as follows:

1. Start with an initial guess $\theta^{(0)}$.
2. Iterate

$$\theta^{(t+1)} = \theta^{(t)} - \left[\nabla^2 \log L(\theta^{(t)} \mid X)\right]^{-1} \nabla \log L(\theta^{(t)} \mid X).$$

3. Stop when convergence criterion is satisfied.

# Newton Raphson for Poisson Regression

For Poisson regression:

$$\log L = \sum_{i=1}^{n} \Big( y_i X_i \beta - e^{X_i \beta} - \log(y_i!) \Big),$$

1. What are the score vector and Hessian corresponding to the log-likelihood?
2. What would the Newton-Raphson steps be?

# Newton Raphson for Poisson Regression

For Poisson regression:

$$\log L = \sum_{i=1}^{n} \Big( y_i X_i \beta - e^{X_i \beta} - \log(y_i!) \Big),$$

The score vector is:

$$\nabla \log L = \sum_{i=1}^{n} \Big( y_i X_i - e^{X_i \beta} X_i \Big) \tag{3}$$

$$= \sum_{i=1}^{n} X_i \Big( y_i - e^{X_i \beta} \Big) \tag{4}$$

$$= \sum_{i=1}^{n} \Big( y_i - e^{X_i \beta} \Big) X_i^T \tag{5}$$

# Newton Raphson for Poisson Regression

Recall the score function is

$$\nabla \log L = \sum_{i=1}^{n} \left( y_i - e^{X_i \beta} \right) X_i^T .$$

The Hessian is:

$$\nabla^2 \log L = -\sum_{i=1}^{n} e^{X_i \beta} X_i X_i^T .$$

Thus, the Newton-Raphson update becomes:

$$\beta^{(t+1)} = \beta^{(t)} - \left[ -\sum_{i=1}^{n} e^{X_i^T \beta} X_i X_i^T \right]^{-1} \sum_{i=1}^{n} \left( y_i - e^{X_i^T \beta} \right) X_i^T .$$

# Newton Raphson for Poisson Regression

Recall:

$$\beta^{(t+1)} = \beta^{(t)} - \left[-\sum_{i=1}^{n} e^{X_i^T \beta^{(t)}} X_i X_i^T\right]^{-1} \sum_{i=1}^{n} \left(y_i - e^{X_i^T \beta}\right) X_i^T.$$

Let $W = e^{X\beta}$, which is an $n \times n$ matrix with $e^{x_i^T \beta}$ on each diagonal. We can re-write this more compactly in the following way:

$$\beta^{(t+1)} = \beta^{(t)} + (X^T W_{(t)} X)^{-1} X^T (y - \exp(X\beta^{(t)}))$$

# Example

We use a data set from the book "An Introduction to Generalized Linear Models (3rd)" on page 67. We make a design matrix, added a constant column for the intercept. Please refer to the book itself regarding the data set.

```
y <- c(2,3,6,7,8,9,10,12,15)
x <- matrix(c(1,1,1,1,1,1,1,1,1, -1, -1, 0, 0, 0, 0, 1, 1, 1), nrow=9, ncol=2)
data_4.3 <- data.frame(y, x1 = c(-1, -1, 0, 0, 0, 0, 1, 1, 1))
```

# Newton Raphson

```r
poisson_Newton_Raphson<-function(x, y, b.init, tol=1e-8){
  change <- Inf
  b.old <- b.init
  while(change > tol) {
    eta <- x %*% b.old  # linear predictor
    w <-diag(as.vector(exp(eta)),
             nrow=nrow(x), ncol=nrow(x))
    b.new<-b.old+ solve(t(x) %*% w %*% x) %*% t(x) %*% (y-exp(eta))
    change <- sqrt(sum((b.new - b.old)^2))
    b.old <- b.new
  }
  b.new
}
poisson_Newton_Raphson(x,y,b.init = c(2,4))
```

```
##              [,1]
## [1,] 1.8892720
## [2,] 0.6697856
```

# Iteratively Re-weighted Least Squares

Newton Raphson can be re-written as what is known as the Iteratively Re-weighted Least Squares (IRLS) algorithm, which is can be faster and/or more stable, and thus, is often preferred over the more simplistic approach.

Derivations can be found in more advanced classes, and it may be useful to know this fact for practical situations.

# Newton Raphson for Poisson Regression

Recall:

$$\beta^{(t+1)} = \beta^{(t)} - \left[-\sum_{i=1}^{n} e^{X_i^T \beta^{(t)}} X_i X_i^T\right]^{-1} \sum_{i=1}^{n} \left(y_i - e^{X_i^T \beta}\right) X_i^T.$$

Let $W = e^{X\beta}$, which is an $n \times n$ matrix with $e^{x_i^T \beta}$ on each diagonal. We can re-write this more compactly in the following way:

$$\beta^{(t+1)} = \beta^{(t)} + (X^T W_{(t)} X)^{-1} X^T (y - \exp(X\beta^{(t)}))$$

# Iteratively Re-weighted Least Squares

Let

$$\beta^{(t+1)} = \beta^{(t)} + (X^T W_{(t)} X)^{-1} X^T (y - \exp(X\beta^{(t)}))$$

and let $p = \exp(X\beta^{(t)})$. Let $z = W^{-1}(y - p) \implies (y - p) = Wz$.

Then it follows that

$$\begin{aligned} \beta^{(t+1)} &= \beta^{(t)} + (X^T W_{(t)} X^T)^{-1} X^T (y - p) && (6) \\ &= \beta^{(t)} + (X^T W_{(t)} X^T)^{-1} X^T W z && (7) \end{aligned}$$

# Iteratively Re-weighted Least Squares

1. z is viewed as the response variable the goal is to minimize $(z - X\beta)^T W(x - X\beta)$ with respect to $\beta$
2. Linear regression by least square solves $(z - X\beta)^T W(x - X\beta)$ where $z$ is referred to as the adjusted response.
3. The algorithm is referred to as iteratively reweighted least squares or IRLS.

# IRLS Code

```
poisson_IRLS <- function(x, y, b.init, tol=1e-8){
  change <- Inf
  b.old <- b.init
  while(change > tol) {
    eta <- x %*% b.old
    w <- diag(as.vector(exp(eta)), nrow=nrow(x), ncol=nrow(x))
    z <- solve(w) %*% (y - exp(eta))
    weight <- exp(eta)
    b.new <- b.old + lm(z ~ x - 1, weights = weight)$coef
    change <- sqrt(sum((b.new - b.old)^2))
    b.old <- b.new
  }
  b.new
}

poisson_IRLS(x, y, rep(1,2))
```

```
##        x1        x2
## 1.8892720 0.6697856
```

# Comparison to glm function

```r
m1<-glm(y ~ x1, family=poisson(link="log"), data=data_4.3)
summary(m1)
```

```
##
## Call:
## glm(formula = y ~ x1, family = poisson(link = "log"), data = data_4.3)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.8893     0.1421  13.294  < 2e-16 ***
## x1            0.6698     0.1787   3.748 0.000178 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 18.4206  on 8  degrees of freedom
## Residual deviance:  2.9387  on 7  degrees of freedom
## AIC: 41.052
##
## Number of Fisher Scoring iterations: 4
```

# Exercises

Repeat the same exercises above for logistic regression.