Untitled

Adam Lauretig

11/8/2018

Calculating an MLE

https://onlinecourses.science.psu.edu/stat504/node/28

Proofs of most Distributions http://www.math.uah.edu/stat/point/Likelihood.html

Binomial Distribution

http://pages.uoregon.edu/aarong/teaching/G4075_Outline/node13.html

1. Suppose that X is an observation from a binomial distribution $X \sim \text{Bin}(n, p)$ where n is known and p is to be estimated. The likelihood function is

$$L(p; x) = \frac{n!}{x!(n-x)!} p^x (1-p)^{n-x}$$

2. which, except for the factor $\frac{n!}{x!(n-x)!}$, is identical to the likelihood from n independent Bernoulli trials with $x = \sum_{i=1}^{n} x_i$. But since the likelihood function is regarded as a function only of the parameter p, the factor $\frac{n!}{x!(n-x)!}$ is a fixed constant and does not affect the MLE. Thus the MLE is again $\hat{p} = \frac{x}{n}$ the sample proportion of successes.

3. You get the same value by maximizing the *binomial loglikelihood function*

$$l(p; x) = k + x \log p + (n-x) \log(1-p)$$

4. We take the derivative of this function with respect to p.

$$\frac{\partial \log L(p)}{\partial p} = \frac{x}{p} - \frac{n-x}{1-p}$$

5. Additionally, the second derivative can be written as (note: θ and p are interchangeable in this context):

$$l''(\theta) = -\frac{x}{\theta^2} - \frac{n-x}{(1-\theta)^2}$$

When this function (the first derivative) equals zero, we will have either a minimum or a maximum. So solve for p.

$$\begin{aligned}
0 &= \frac{x}{p} - \frac{n-x}{1-p} \\
\frac{n-x}{1-p} &= \frac{x}{p} \\
p(n-x) &= x(1-p) \\
pn - px &= x - px \\
pn &= x \\
p &= \frac{x}{n}
\end{aligned}$$

Exponential Applied Example

Given iid data $x_1, \cdots, x_n$ from an exponential distribution $\theta e^{-\theta x}$

The Likelihood Funtion

$$f(x_i \cdots x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta e^{-\theta x_i}) = L(\theta|\mathbf{x})$$

θ is the parameter/set of parameters we want to find the maximum likelihood estimator for, given sample of values $\mathbf{x}$. f is the PDF.

Log Likelihood

We now take lnL to transform the $\prod$ into a $\sum$, which will make the math we will have to do later easier.\

Starting with taking the log of the original likelihood function:

$$= ln(\theta e^{-\theta x_i})$$

Break it in to more manageable sections (in accordance with logarithm properties), the n in front of the θ is because θ will be summed n times :

$$= nln(\theta) + ln(e^{-\theta \sum_{i=1}^{n} x_i})$$

\ Now, the $ln(e)$ will cancel (logarithm properties):

$$= nln(\theta) - \theta \sum_{i=1}^{n} x_i$$

\ leaving us with our log-likelihood.

\subsection{Score Function - First Derivative of the Log Likelihood}

We now take the first derivative of the log likelihood with respect to θ, in order to find maximum likelihood:\

$$ \frac{\partial{n ln(\theta) - {\theta} {{\sum_{i=1}^n}x_{i}}}}{\partial{\theta}} $$\\

Expanded out (but prior to simplification) this looks like:

$$= \left(\frac{\partial n}{\partial \theta} ln(\theta)\right) + \left(n\left(\frac{1}{\theta}\right)\right) - \left[\left(\left(\frac{\partial \theta}{\partial \theta}\right)\sum_{i=1}^{n} x_i\right) + \left(\left(\frac{\partial \sum_{i=1}^{n} x_i}{\partial \theta}\right)\theta\right)\right]$$

\ Which simplifies to:

$$\left(\frac{n}{\theta}\right) - \sum_{i=1}^{n} x_i$$

\

The MLE

Now we solve for $\hat{\theta}$, beginning with setting the score function $= 0$

$$0 = \left(\frac{n}{\hat{\theta}}\right) - \sum_{i=1}^{n} x_i \tag{1}$$

\ We want to isolate $\hat{\theta}$

$$\hat{\theta} = \frac{n}{\sum_{i=1}^{n} x_i} \tag{2}$$

\ Thus, as an estimator:

$$\hat{\theta} = \frac{n}{\sum_{i=1}^{n} x_i} \tag{3}$$

\

This is the generic estimator for an exponential distribution, though one could replace $\hat{\theta}$ with $\hat{\lambda}$ if you so desired.

Observed Fisher Information

The Observed Fisher Information is the curvature of at the MLE, which tells us our certainty in out MLE for a given distribution. It is the inverse of the second derivative of the log-likelihood:

$$I(\hat{\theta}) = -\frac{\partial^2}{\partial\theta^2} lnL(\hat{\theta})$$

\

So, for the exponential distribution, we take the derivative of $\hat{\theta}$ we found above (only before we manipulated it to solve for $\hat{\theta}$, starting with:

$$= \frac{\partial\left(\frac{n}{\hat{\theta}}\right) - \sum_{i=1}^{n} x_i}{\partial\hat{\theta}}$$

\

which results in:

$$= \left(n\frac{\partial\hat{\theta}^{-1}}{\partial\hat{\theta}}\right) + \left(\hat{\theta}\frac{\partial n}{\partial\hat{\theta}}\right)$$

\

which simplifies to:

$$0 = -\frac{n}{\theta^2}$$

\ and then we take the inverse:

$$\frac{n}{\hat{\theta}^2}$$

\ which is the generic Observed Fisher Information for an exponential distribution.

Hand-Rolling a Poisson MLE in R

```
# generate our data
set.seed(216)
n <- 100
b0 <- 2
b1 <- -1.9
x <- rnorm(n)
lp <- exp(b0 + x*b1)
y <- rpois(n, lambda =  lp)

loglik_poisson <- function(X, par,y)
{
  beta <- par
  # the deterministic part of the model:
  lambda <- exp(X%*%beta)
  # and here comes the negative log-likelihood of the whole dataset, given the
  # model:
  LL <- -sum(dpois(y, lambda, log = TRUE))
  return(LL)
}
par <- c(rep(rnorm(1), 2))
X <- cbind(1, x)
pois_out<- optim(par = par, fn = loglik_poisson, y = y, X = X, hessian = TRUE)
pois_out
```

```
## $par
## [1]  2.023447 -1.887521
##
## $value
## [1] 223.7191
##
## $counts
## function gradient
##       77       NA
##
## $convergence
## [1] 0
##
## $message
## NULL
##
## $hessian
##           [,1]      [,2]
## [1,]  4542.573 -7893.217
## [2,] -7893.217 16119.692
```

Bayesian Estimation

```
library(rstanarm) # bayesian regression models in R
```

```
## Loading required package: Rcpp

## rstanarm (Version 2.17.4, packaged: 2018-04-13 01:51:52 UTC)

## - Do not expect the default priors to remain the same in future rstanarm versions.

## Thus, R scripts should specify priors explicitly, even if they are just the defaults.

## - For execution on a local, multicore CPU with excess RAM we recommend calling

## options(mc.cores = parallel::detectCores())

## - Plotting theme set to bayesplot::theme_default().
```

```
# generate our data
set.seed(216)
n <- 1000
b0 <- 2
b1 <- -1.9
eps <- rnorm(n, 0, 1)
x <- rnorm(n)
lp <- exp(b0 + x*b1 + eps)
y <- rpois(n, lambda =  lp)

df <- data.frame(y = y, x = x)

formula_to_use <- as.formula("y~x")

m1 <- stan_glm(formula_to_use, family = poisson(), data = df, prior = normal(0, 10), algorithm = "sampl
```

```
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 1).
##
## Gradient evaluation took 0.000249 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 2.49 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 4.35814 seconds (Warm-up)
##                5.06591 seconds (Sampling)
```

```
##                9.42405 seconds (Total)
##
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 2).
##
## Gradient evaluation took 0.000156 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.56 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 6.0318 seconds (Warm-up)
##                5.49658 seconds (Sampling)
##                11.5284 seconds (Total)
##
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 3).
##
## Gradient evaluation took 0.000182 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.82 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 3.74404 seconds (Warm-up)
##                4.20687 seconds (Sampling)
##                7.95091 seconds (Total)
##
##
## SAMPLING FOR MODEL 'count' NOW (CHAIN 4).
```

```
##
## Gradient evaluation took 0.000158 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 1.58 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 4.0027 seconds (Warm-up)
##                4.87304 seconds (Sampling)
##                8.87574 seconds (Total)
```

```r
summary(m1)
```

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       poisson [log]
##  formula:      y ~ x
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 1000
##  predictors:   2
##
## Estimates:
##                 mean     sd       2.5%     25%      50%      75%
## (Intercept)        2.1      0.0      2.1      2.1      2.1      2.1
## x                 -2.2      0.0     -2.2     -2.2     -2.2     -2.2
## mean_PPD          87.0      0.4     86.2     86.8     87.0     87.3
## log-posterior -55715.6      1.0 -55718.3 -55715.9 -55715.3 -55714.9
##                 97.5%
## (Intercept)        2.1
## x                 -2.2
## mean_PPD          87.9
## log-posterior -55714.6
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0   801
## x             0.0  1.0   879
## mean_PPD      0.0  1.0  3128
## log-posterior 0.0  1.0   916
```

```
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```r
# compare to
m2 <- glm(formula_to_use, family = poisson(), data = df)
summary(m2)
```

```
##
## Call:
## glm(formula = formula_to_use, family = poisson(), data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -101.975    -2.561    -0.572     1.752   166.240
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2.093271   0.008910   234.9   <2e-16 ***
## x           -2.211200   0.003999  -552.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 487642  on 999  degrees of freedom
## Residual deviance: 107713  on 998  degrees of freedom
## AIC: 111425
##
## Number of Fisher Scoring iterations: 5
```