Fitting linear models in R (1)

Yue Jiang

Duke University

Estimating bike crashes in NC counties

Data

```
## # A tibble: 100 x 2
##    county    crashes
##    <chr>       <dbl>
##  1 Alamance       77
##  2 Alexander       1
##  3 Alleghany       1
##  4 Anson           7
##  5 Ashe            4
##  6 Avery           5
##  7 Beaufort       37
##  8 Bertie         10
##  9 Bladen          9
## 10 Brunswick      88
## # ... with 90 more rows
```

Suppose we thought these crashes came from a Poisson distribution.

How might we estimate the parameter of that Poisson distribution, given our observed data?

Maximum likelihood estimation

We can maximize the likelihood function. Assuming the observations are i.i.d., in general we have:

$$\begin{aligned}\mathcal{L}(\lambda|Y) &= f(y_1, y_2, \cdots, y_n|\lambda) \\ &= f(y_1|\lambda)f(y_2|\lambda)\cdots f(y_n|\lambda) \\ &= \prod_{i=1}^{n} f(y_i|\lambda).\end{aligned}$$

The likelihood function is the probability of "seeing our observed data," **given** a value of λ. Do not get $f(y_i|\lambda)$ confused with $f(\lambda|y_i)$!

Maximum likelihood estimation

For our Poisson example, we thus have:

$$
\begin{aligned}
\mathcal{L}(\lambda|Y) &= \prod_{i=1}^{n} f(y_i|\lambda) \\
&= \prod_{i=1}^{n} \frac{\lambda_i^{y} e^{-\lambda}}{y_i!} \\
\log \mathcal{L}(\lambda|Y) &= \sum_{i=1}^{n} (y_i \log \lambda - \lambda - \log y_i!) \\
&= \log \lambda \sum_{i=1}^{n} y_i - n\lambda - \sum_{i=1}^{n} \log y_i!
\end{aligned}
$$

Why do we maximize the log-likelihood function here?

Maximum likelihood estimation

Setting the score function equal to 0:

$$\begin{aligned}
\frac{\partial}{\partial \lambda} \log \mathcal{L}(\lambda|Y) = \frac{1}{\lambda}\sum_{i=1}^{n} y_i - n \overset{set}{=} 0 \\
\implies \hat{\lambda} = \frac{1}{n}\sum_{i=1}^{n} y_i,
\end{aligned}$$

as expected. Next, let's verify that $\hat{\lambda}$ is indeed a maximum:

$$\begin{aligned}
\frac{\partial^2}{\partial \lambda^2} \log \mathcal{L}(\lambda|Y) = -\frac{1}{\lambda^2}\sum_{i=1}^{n} y_i - n \\
< 0.
\end{aligned}$$

Can we do better?

```
## # A tibble: 100 x 6
##    county         pop med_hh_income traffic_vol pct_rural crashes
##    <chr>        <dbl>         <dbl>       <dbl>     <dbl>   <dbl>
##  1 Alamance    166436          50.5         182        29      77
##  2 Alexander    37353          49.1          13        73       1
##  3 Alleghany    11161          39.7          28       100       1
##  4 Anson        24877          38            79        79       7
##  5 Ashe         27109          41.9          18        85       4
##  6 Avery        17505          41.7          35        89       5
##  7 Beaufort     47079          46.4          53        66      37
##  8 Bertie       19026          35.4          24        83      10
##  9 Bladen       33190          37            19        91       9
## 10 Brunswick   136744          60.2          43        43      88
## # ... with 90 more rows
```

We might expect that more populous, more urban counties might have more crashes. There might also be a relationship with traffic volume.

Can we incorporate this additional information while accounting for potential confounding?

Poisson regression

$$\log(\underbrace{E(Y|\mathbf{X})}_{\lambda}) = \beta_0 + \mathbf{X}^T\boldsymbol{\beta}$$

Generalized linear model often used for count (or rate) data

- Assumes outcome has Poisson distribution
- Canonical link: log of conditional expectation of response has linear relationship with predictors

Can we differentiate the (log) likelihood function, set it equal to zero, and solve for the MLEs for $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_p)$ as before?

Poisson regression

$$\begin{aligned}\log \mathcal{L} &= \sum_{i=1}^{n} (y_i \log \lambda - \lambda - \log y_i!) \\ &= \sum_{i=1}^{n} y_i \mathbf{X}_i \boldsymbol{\beta} - e^{\mathbf{X}_i \boldsymbol{\beta}} - \log y_i!\end{aligned}$$

We would like to solve the equations

$$\left(\frac{\partial \log \mathcal{L}}{\partial \beta_j} \right) \overset{set}{=} \mathbf{0},$$

but there is no closed-form solution, as this is a transcendental equation in the parameters of interest.

How might we solve these equations numerically?

A one-dimensional problem

Suppose you're trying to find the maximum of the following function:

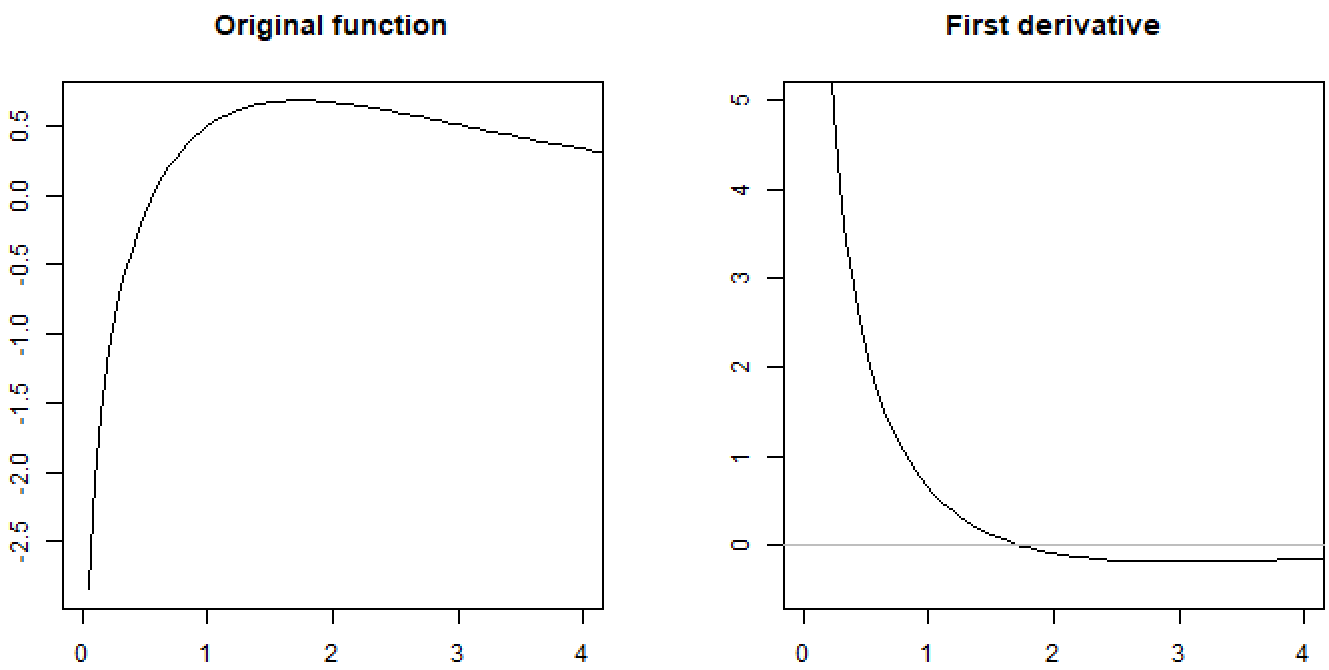
$$f(x) = \frac{x + \log(x)}{2^x}$$

Let's try differentiating, setting equal to 0, and solving:

$$\frac{d}{dx} f(x) = 2^{-x} \left(1 + \frac{1}{x} - \log(2)(x + \log(x)) \right).$$

We run into a similar problem: we cannot algebraically solve for the root of this equation.

A one-dimensional problem



It looks lke the maximum is a bit shy of 2 (trust me on this one, it's a global maximum). How might we find where it is?

A one-dimensional problem

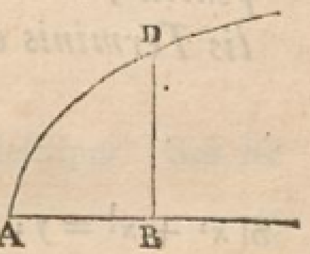
DE ANALYSI

Per Æquationes Numero Terminorum

INFINITAS.

Methodum generalem, quam de Curvarum quantitate per Infinitam terminorum Seriem mensurandam, olim excogitaveram, in sequentibus breviter explicatam potius quam accuratè demonstratam habes.

BASI AB Curvæ alicujus AD, sit Applicata BD perpendicularis: Et vocetur $AB = x$, $BD = y$, & sint a, b, c, &c. Quantitates datæ, & m, n, Numeri Integri. Deinde,



Curvarum Simplicium Quadratura.

A one-dimensional problem

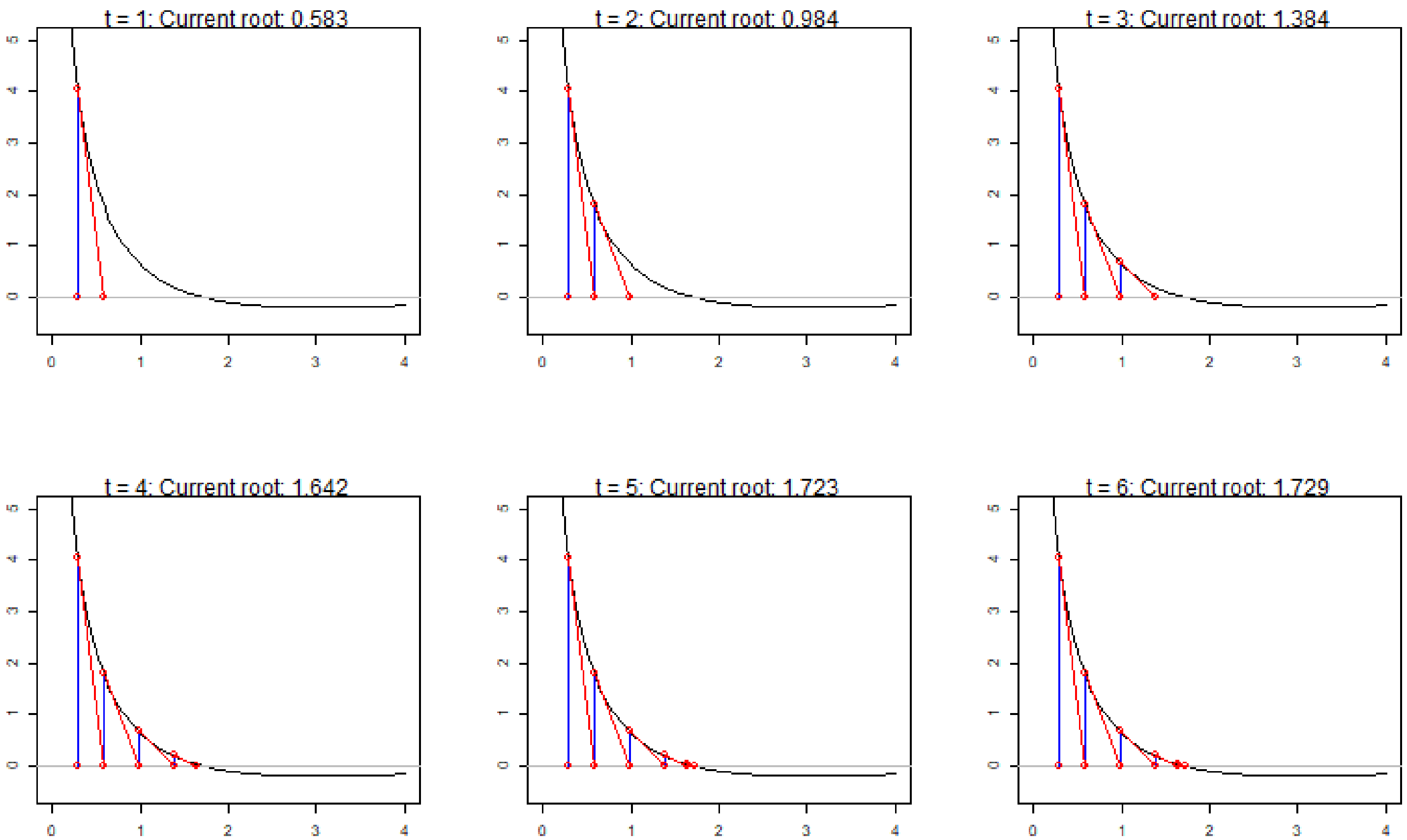
Newton-Raphson algorithm for root finding is based on second-order Taylor approximation around true root:

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

Although it requires explicit forms of first two derivatives, the convergence speed is quite fast.

There are some necessary conditions for convergence, but this is beyond the scope of STA 440. Many likelihood functions you are likely to encounter (e.g., GLMs with canonical link) will in fact converge from any starting value.

A one-dimensional problem



A one-dimensional problem

$$f(x) = \frac{x + \log(x)}{2^x}$$
$$\frac{d}{dx} f(x) = 2^{-x} \left(1 + \frac{1}{x} - \log(2)(x + \log(x))\right).$$

```
testing <- function(x){
  2^(-1 * x) * (1 + 1/x - log(2) * (x + log(x)))
}

testing(1.729)
```

```
## [1] 0.0001174952
```

That's pretty good (only six steps from starting guess of 0.3)!

Newton-Raphson in higher dimensions

Score vector and Hessian for $\log \mathcal{L}(\boldsymbol{\theta}|\mathbf{X})$ with $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_p)^T$:

$$\nabla \log \mathcal{L} = \begin{pmatrix} \frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\theta}_1} \\ \vdots \\ \frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\theta}_p} \end{pmatrix}$$

$$\nabla^2 \log \mathcal{L} = \begin{pmatrix} \frac{\partial^2 \log \mathcal{L}}{\partial \theta_1^2} & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_1 \theta_2} & \cdots & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_1 \theta_p} \\ \frac{\partial^2 \log \mathcal{L}}{\partial \theta_2 \theta_1} & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_2^2} & \cdots & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_2 \theta_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 \log \mathcal{L}}{\partial \theta_p \theta_1} & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_p \theta_2} & \cdots & \frac{\partial^2 \log \mathcal{L}}{\partial \theta_p^2} \end{pmatrix}$$

Newton-Raphson in higher dimensions

We can modify the Newton-Raphson algorithm for higher dimensions:

- Start with initial guess $\boldsymbol{\theta}^{(0)}$
- Iterate $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \left(\nabla^2 \log \mathcal{L}(\boldsymbol{\theta}^{(t)}|\mathbf{X})\right)^{-1} \left(\nabla \log \mathcal{L}(\boldsymbol{\theta}^{(t)}|\mathbf{X})\right)$
- Stop when convergence criterion is satisfied

Under certain conditions, a global maximum exists; this again is guaranteed for many common applications.

Computing the Hessian can be computationally demanding (and annoying), but there are ways around it in practice.

Poisson regression

$$\log \mathcal{L} = \sum_{i=1}^{n} y_i \mathbf{X}_i \boldsymbol{\beta} - e^{\mathbf{X}_i \boldsymbol{\beta}} - \log y_i!$$

$$\nabla \log \mathcal{L} = \sum_{i=1}^{n} \left(y_i - e^{\mathbf{X}_i \boldsymbol{\beta}} \right) \mathbf{X}_i^T$$

$$\nabla^2 \log \mathcal{L} = - \sum_{i=1}^{n} e^{\mathbf{X}_i \boldsymbol{\beta}} \mathbf{X}_i \mathbf{X}_i^T$$

Newton-Raphson update steps for Poisson regression:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \left(- \sum_{i=1}^{n} e^{\mathbf{X}_i \boldsymbol{\beta}} \mathbf{X}_i \mathbf{X}_i^T \right)^{-1} \left(\sum_{i=1}^{n} \left(y_i - e^{\mathbf{X}_i \boldsymbol{\beta}} \right) \mathbf{X}_i^T \right)$$