Newton's Method

Ishtiaq Mahmud Fahim

Table of contents

How Newton–Raphson Works

The **Newton–Raphson method** is an iterative root-finding algorithm. Suppose we want to solve for the root of an equation $g(\theta) = 0$.

- Start with an initial guess $\theta^{(0)}$.
- Update iteratively using:

$$\theta^{(t+1)} = \theta^{(t)} - \frac{g(\theta^{(t)})}{g'(\theta^{(t)})}$$

That is, we use the slope of the function to "jump" closer to the root. Graphically, you take the tangent line at the current point, and where it crosses the x-axis becomes the new guess.

In **statistics and estimation**, we usually apply Newton–Raphson to the **score equations** (derivatives of the log-likelihood).

Newton–Raphson vs MLE

- **MLE**: Maximum Likelihood Estimation finds parameter values that maximize the likelihood (or log-likelihood). Formally, we solve:

  $$\frac{\partial \ell(\theta)}{\partial \theta} = 0$$

  for the parameter(s) θ. This gives exact closed-form estimates **if the equation can be solved analytically**.

Newton–Raphson: If solving the likelihood equations is difficult or impossible in closed form, Newton–Raphson provides a numerical way to approximate the MLE by iteratively solving:

$$\theta^{(t+1)} = \theta^{(t)} - \left[\frac{\partial^2 \ell(\theta)}{\partial \theta^2}\right]^{-1} \cdot \frac{\partial \ell(\theta)}{\partial \theta}$$

This works because:

- The score equation $U(\theta) = \partial \ell(\theta)/\partial \theta = 0$ is the root we want.
- Newton–Raphson finds that root numerically.

Why Newton–Raphson Helps

1. **No closed form needed** – Many models (logistic regression, survival models, mixed models) don't have neat closed-form MLE solutions. Newton–Raphson lets us approximate them.
2. **Faster convergence** – Compared to simple methods like gradient ascent, Newton–Raphson converges very quickly (quadratic convergence near the solution).
3. **Directly uses curvature** – It uses both the first and second derivative of the log-likelihood, making jumps more efficient.
4. **Flexible** – Can handle multiple parameters at once (multivariate case uses Hessian matrix).

Problem 1

Data and parameter

- $n = 15$
- Unknown parameter: $\theta = \sigma^2$ (variance)
- Statistic: $Q = \sum_{i=1}^{n} x_i^2 = 124.88$ (given)

So now:

- θ = what we want to estimate
- Q = fixed number from data

Log-likelihood

$$\ell(\theta) = -\frac{n}{2}\ln(\theta) - \frac{Q}{2\theta}.$$

First derivative (score):

$$\ell'(\theta) = -\frac{n}{2\theta} + \frac{Q}{2\theta^2} = \frac{Q - n\theta}{2\theta^2}.$$

Second derivative (Hessian):

$$\ell''(\theta) = \frac{n}{2\theta^2} - \frac{Q}{\theta^3} = \frac{n\theta - 2Q}{2\theta^3}.$$

Newton–Raphson update

$$\theta_{k+1} = \theta_k - \frac{\ell'(\theta_k)}{\ell''(\theta_k)} = \theta_k - \frac{(Q - n\theta_k)\,\theta_k}{n\theta_k - 2Q}.$$

```r
# Problem data
n <- 15
Q <- 124.88      # sum of squares
eps <- 5         # iter
theta0 <- 2.5    # initial guess for sigma^2
result <- c()    # for storing

while (eps > 0.0001) {

  l1 <- (Q - n*theta0) / (2*theta0^2)     # Score
  l2 <- (n*theta0 - 2*Q) / (2*theta0^3)   # Hessian

  # Newton update

  theta1 <- theta0 - l1/l2
  eps <- abs(theta0 - theta1)

  # Save

  result <- rbind(result,
                  c(theta0 = theta0, theta1 = theta1, l1 = l1, eps = eps ))
  theta0 <- theta1
}

library(knitr)
kable(result)
```

theta0	theta1	l1	eps
2.500000	3.529162	6.9904000	1.0291623
3.529162	4.819141	2.8881028	1.2899791
4.819141	6.247261	1.1322898	1.4281197
6.247261	7.495147	0.3993398	1.2478856
7.495147	8.174777	0.1108349	0.6796305
8.174777	8.319985	0.0168969	0.1452075
8.319985	8.325326	0.0005795	0.0053417
8.325326	8.325333	0.0000007	0.0000069

The iteration table illustrates that, $\sigma^2 = 8.325$

Problem 2

Log-likelihood function

The log-likelihood for the normal distribution is

$$\ell(\mu, \sigma) = -\frac{n}{2} \log(2\pi) - n \log \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2.$$

Score functions (first derivatives)

Derivative with respect to μ:

$$\frac{\partial \ell}{\partial \mu} = \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu)$$

Derivative with respect to σ:

$$\frac{\partial \ell}{\partial \sigma} = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n} (x_i - \mu)^2$$

So the **score vector** is

$$s(\mu, \sigma) = \begin{pmatrix} \frac{1}{\sigma^2} \sum_{i=1}^{n} (x_i - \mu) \\ -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n} (x_i - \mu)^2 \end{pmatrix}.$$

Observed information (Hessian matrix)

Compute the **second derivatives**:

Second derivative w.r.t. μ:

$$\frac{\partial^2 \ell}{\partial \mu^2} = -\frac{n}{\sigma^2}$$

Mixed derivative:

$$\frac{\partial^2 \ell}{\partial \mu \, \partial \sigma} = -\frac{2}{\sigma^3} \sum_{i=1}^{n} (x_i - \mu)$$

Second derivative w.r.t. σ :

$$\frac{\partial^2 \ell}{\partial \sigma^2} = \frac{n}{\sigma^2} - \frac{3}{\sigma^4} \sum_{i=1}^{n} (x_i - \mu)^2$$

Thus the **Hessian matrix** is

$$H(\mu, \sigma) = \begin{pmatrix} -\dfrac{n}{\sigma^2} & -\dfrac{2}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu) \\ -\dfrac{2}{\sigma^3}\sum_{i=1}^{n}(x_i - \mu) & \dfrac{n}{\sigma^2} - \dfrac{3}{\sigma^4}\sum_{i=1}^{n}(x_i - \mu)^2 \end{pmatrix}.$$

Newton–Raphson update

The Newton–Raphson iteration is:

$$\begin{pmatrix} \mu \\ \sigma \end{pmatrix}_{\text{new}} = \begin{pmatrix} \mu \\ \sigma \end{pmatrix}_{\text{old}} - H(\mu, \sigma)^{-1} s(\mu, \sigma)$$

Given,

- $n = 15$
- $\sum_{i=1}^{n} x_i = 871.67$
- $\sum_{i=1}^{n} x_i^2 = 30736.31$

```r
# Given summary statistics
n <- 15
sum_x <- 871.67
sum_x2 <- 30736.31

# Initial guesses
mu0 <- 30
sigma0 <- sqrt(2.5)   # nb: problem gave sigma^2=2.5, so sigma = sqrt(2.5)

theta <- matrix(
  c(
    mu0,
    sigma0
  ),
  nrow = 2
)

eps1 <- 5
eps2 <- 5
tol <- 0.00001

result <- c()

while (eps1 > tol && eps2 > tol) {

  # Compute useful sums
  sum_xmu  <- sum_x - n * mu0
  sum_xmu2 <- sum_x2 - 2 * mu0 * sum_x + n * mu0^2

  # Score vector
  s1 <- (1/sigma0^2) * sum_xmu
  s2 <- -n/sigma0 + (1/sigma0^3) * sum_xmu2
  s  <- matrix(
    c(
      s1,
      s2
    ),
    nrow = 2
  )

  # Hessian
  h11 <- -n/sigma0^2
```

```r
  h12 <- -2/sigma0^3 * sum_xmu
  h22 <- n/sigma0^2 - 3/sigma0^4 * sum_xmu2
  H   <- matrix(c(h11, h12, h12, h22), nrow = 2, byrow = TRUE)

  # Newton step
  delta <- theta - solve(H) %*% s

  mu1    <- delta[1]
  sigma1 <- delta[2]

  eps1 <- abs(mu1 - mu0)
  eps2 <- abs(sigma1 - sigma0)

  result <- rbind(result,
                  c(
                    mu = mu0,
                    step_mu = mu1,
                    eps1 = eps1,
                    sigma = sigma0,
                    step_sigma = sigma1,
                    eps2 = eps2
                  ))

  mu0 <- mu1
  sigma0 <- sigma1
}

library(knitr)
kable(result[1:4,])
kable(result[5:8,])
kable(result[9:12,])
```

mu	step_mu	eps1	sigma	step_sigma	eps2
30.00000	33.14936	3.1493572	1.581139	2.283139	0.7020007
33.14936	33.78625	0.6368973	2.283139	2.549554	0.2664148
33.78625	33.87371	0.0874598	2.549554	2.652911	0.1033563
33.87371	33.88255	0.0088396	2.652911	2.695113	0.0422025

mu	step_mu	eps1	sigma	step_sigma	eps2
33.88255	33.88238	0.0001712	2.695113	2.712765	0.0176519
33.88238	33.88177	0.0006097	2.712765	2.720212	0.0074474
33.88177	33.88144	0.0003322	2.720212	2.723364	0.0031512
33.88144	33.88129	0.0001510	2.723364	2.724698	0.0013347

mu	step_mu	eps1	sigma	step_sigma	eps2
33.88129	33.88122	6.54e-05	2.724698	2.725264	0.0005655
33.88122	33.88120	2.79e-05	2.725264	2.725503	0.0002396
33.88120	33.88118	1.19e-05	2.725503	2.725605	0.0001015
33.88118	33.88118	5.00e-06	2.725605	2.725648	0.0000430

So our desired estimates are $\mu = 33.881$ and $\sigma = 2.726$