Optimization

2012-9-13

Yijuan Hu

Maximum Likelihood Estimation

MLE $\hat{\theta} = \arg\max_\theta l(\theta)$, where θ is a p-vector parameter

Review

Log likelihood	$l(\theta) = \log \Pr(X\|\theta)$
Score function	$\dot{l}(\theta) = (\partial l/\partial\theta_1, \ldots, \partial l/\partial\theta_p)'$
Hessian matrix	$\ddot{l}(\theta) = \{\partial^2 l/\partial\theta_i\partial\theta_j\}_{i,j=1,\ldots,p}$
Fisher information	$I(\theta) = -\mathrm{E}\ddot{l}(\theta) = \mathrm{E}\dot{l}(\theta)\{\dot{l}(\theta)\}'$
Observed information	$-\ddot{l}(\hat{\theta})$

Approach Find $\hat{\theta}$ such that $\dot{l}(\hat{\theta}) = 0$. When $\hat{\theta}$ is a local maximum of l, $\dot{l}(\hat{\theta}) = 0$ and $\ddot{l}(\hat{\theta})$ is negative definite.

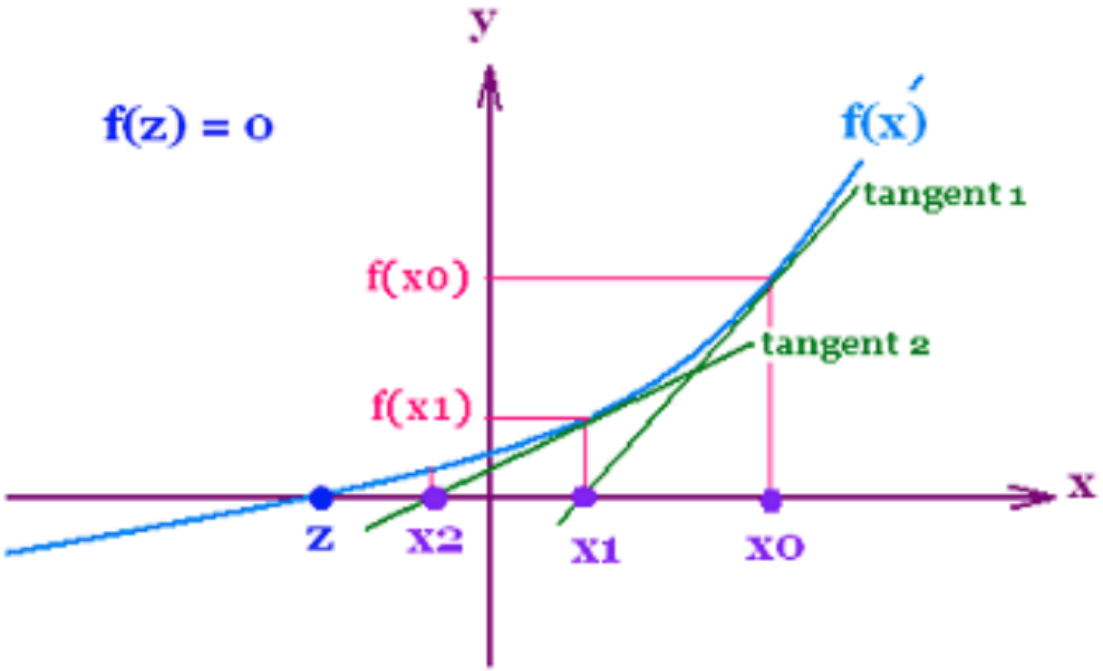
Maximization vs. minimization: Maximizing $f(.)$ minimizes $-f(.)$. Switch from minimization to maximization by standing on your head.

Newton-Raphson method

$$\theta^{(k+1)} = \theta^{(k)} - \dot{l}(\theta^{(k)})/\ddot{l}(\theta^{(k)}).$$

By Taylor expansion at $\theta^{(k)}$, $\dot{l}(\theta) \approx \dot{l}(\theta^{(k)}) + \ddot{l}(\theta^{(k)})(\theta - \theta^{(k)})$. Setting $\dot{l}(\theta) = 0$, we obtain $\theta^{(k+1)}$ as above.

Notice the change in the figure: $f(z) = -\dot{l}(\theta)$



Quadratic convergence

$$\lim_{k\to\infty} \frac{|\theta^{(k+1)} - \hat{\theta}|}{|\theta^{(k)} - \hat{\theta}|^2} = c \qquad (\text{rate} = c > 0, \text{order} = 2)$$

The # of significant digits nearly doubles at each step (in the neighborhood of $\hat{\theta}$).

Proof: Let $f(\theta) = \dot{l}(\theta)$ and $\hat{\theta}$ is the root. By Taylor expansion at $\theta^{(k)}$,

$$0 = f(\hat{\theta}) = f(\theta^{(k)}) + f'(\theta^{(k)})(\hat{\theta} - \theta^{(k)}) + \frac{1}{2} f''(\xi^{(k)})(\hat{\theta} - \theta^{(k)})^2, \quad \xi^{(k)} \in [\hat{\theta}, \theta^{(k)}]$$

Dividing the equation by $f'(\theta^{(k)})$ gives

$$-f(\theta^{(k)})/f'(\theta^{(k)}) - (\hat{\theta} - \theta^{(k)}) = \frac{f''(\xi^{(k)})}{2f'(\theta^{(k)})}(\hat{\theta} - \theta^{(k)})^2.$$

The definition of $\theta^{(k+1)} = \theta^{(k)} - f(\theta^{(k)})/f'(\theta^{(k)})$ gives

$$\theta^{(k+1)} - \hat{\theta} = \frac{f''(\xi^{(k)})}{2f'(\theta^{(k)})}(\hat{\theta} - \theta^{(k)})^2.$$

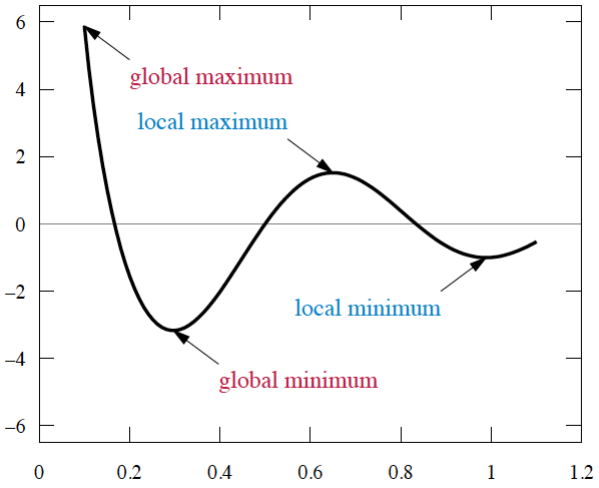
What conditions are needed?

- $f'(\theta^{(k)}) \neq 0$ in the neighborhood of $\hat{\theta}$
- $f''(\xi^{(k)})$ is bounded
- Starting point is sufficiently close to the root $\hat{\theta}$

What can go wrong?

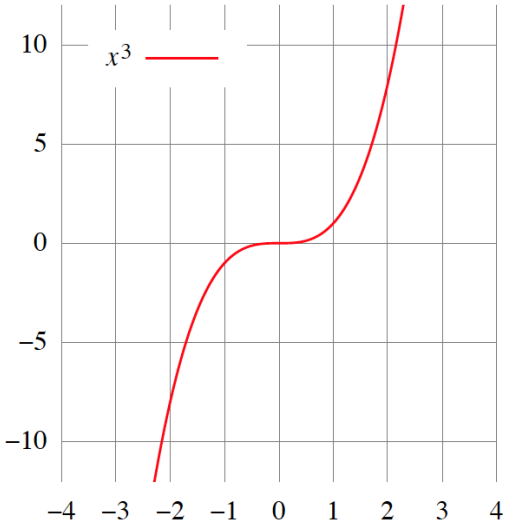
- Bad starting point
- May not converge to the global maximum
- Saddle point: $\dot{l}(\hat{\theta}) = 0$, but $\ddot{l}(\hat{\theta})$ is neither negative definite nor positive definite (stationary point but not a local extremum; can be used to check the likelihood)

starting point & local extremum



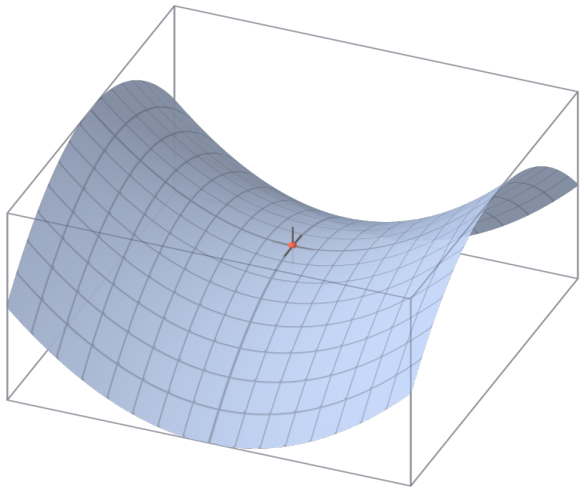
saddle point

$l(\theta) = \theta^3$



saddle point

$l(\theta_1, \theta_2) = \theta_1^2 - \theta_2^2$



Generalization to higher dimensions

General Algorithm

1. **(Starting point)** Pick a starting point $\theta^{(0)}$ and let $k = 0$
2. **(Iteration)** Determine the direction $d^{(k)}$ (a p-vector) and the step size $\alpha^{(k)}$ (a scaler) and calculate

$$\theta^{(k+1)} = \theta^{(k)} + \alpha^{(k)} d^{(k)},$$

such that

$$l(\theta^{(k+1)}) > l(\theta^{(k)})$$

3. **(Stop criteria)** Stop iteration if

$$|l(\theta^{(k+1)}) - l(\theta^{(k)})|/(|l(\theta^{(k)})| + \epsilon_1) < \epsilon_2$$

or

$$|\theta_{k+1,j} - \theta_{k,j}|/(|\theta_{k,j}| + \epsilon_1) < \epsilon_2, \quad j = 1, \ldots, p$$

for precisions such as $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-6}$. Otherwise go to 2.

Key: Determine the direction and the step size

Determining the direction (general framework, details later)

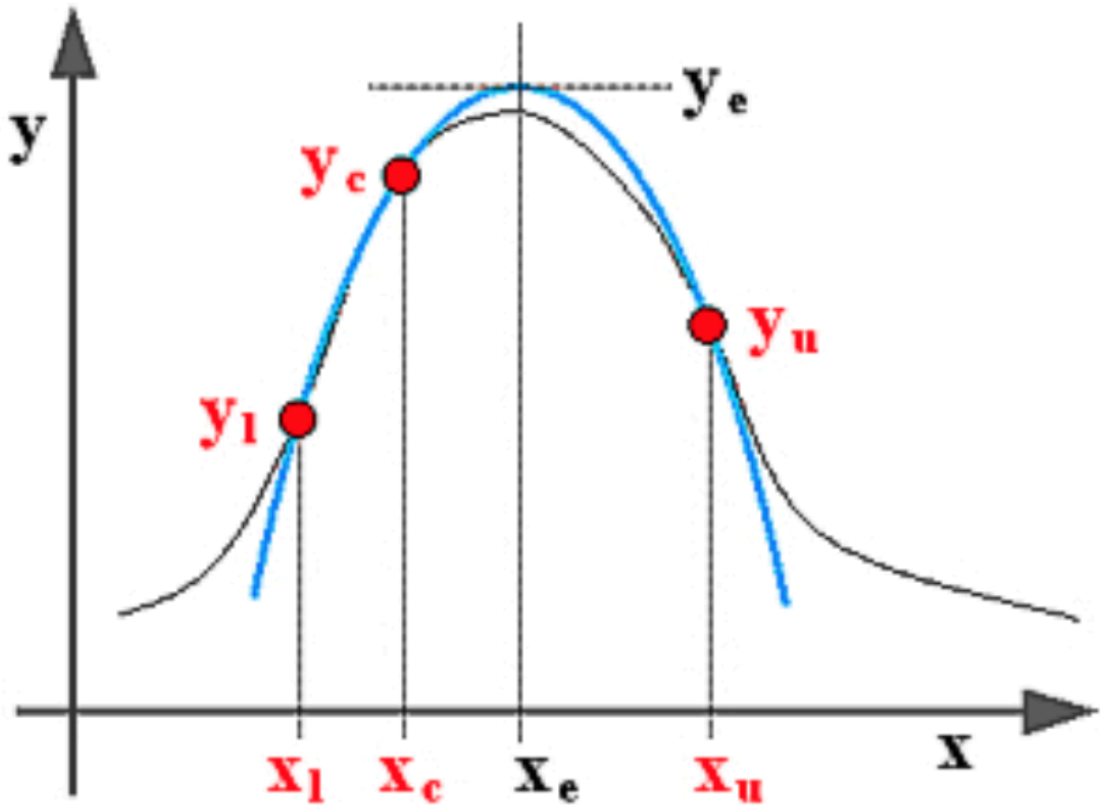
We generally pick $d^{(k)} = R^{-1}\dot{l}(\theta^{(k)})$, where R is a positive definite matrix.

Choosing a step size (given the direction)

- Step halving
  - To find $\alpha^{(k)}$ such that $l(\theta^{(k+1)}) > l(\theta^{(k)})$
  - Start at a large value of $\alpha^{(k)}$. Halve $\alpha^{(k)}$ until $l(\theta^{(k+1)}) > l(\theta^{(k)})$
  - Simple, robust, but relatively slow
- Linear search
  - To find $\alpha^{(k)} = \arg\max_\alpha l(\theta^{(k)} + \alpha d^{(k)})$
  - Approximate $l(\theta^{(k)} + \alpha d^{(k)})$ by doing a polynomial interpolation and find $\alpha^{(k)}$ maximizing the polynomial
  - Fast

Polynomial interpolation

Given a set of $p+1$ data points from the function $f(\alpha) \equiv l(\theta^{(k)} + \alpha d^{(k)})$, we can find a unique polynomial with degree p that goes through the $p+1$ data points. (For a quadratic approximation, we only need 3 data points.)



Survey of basic methods

1. Steepest ascent: $R = I$ = identity matrix

$$d^{(k)} = \dot{l}(\theta^{(k)})$$
$$\alpha^{(k)} = \arg\max_{\alpha} l(\theta^{(k)} + \alpha \dot{l}(\theta^{(k)})) \text{ or a small fixed number}$$
$$\theta^{(k+1)} = \theta^{(k)} + \alpha^{(k)} \dot{l}(\theta^{(k)})$$

Why $\dot{l}(\theta^{(k)})$ is the steepest ascent direction?

By Taylor expansion at $\theta^{(k)}$,

$$l(\theta^{(k)} + \Delta) - l(\theta^{(k)}) = \Delta' \dot{l}(\theta^{(k)}) + o(||\Delta||).$$

By Cauchy-Schwarz inequality,

$$\Delta' \dot{l}(\theta^{(k)}) \leq ||\Delta|| \cdot ||\dot{l}(\theta^{(k)})||,$$

and equality holds at $\Delta = \alpha \dot{l}(\theta^{(k)})$. It means when $\Delta = \alpha \dot{l}(\theta^{(k)})$, $l(\theta^{(k)} + \Delta)$ increases the most.

- Easy to implement; only require the first derivative/gradient/score
- Guarantee an increase at each step no matter where you start
- Converge slowly. The directions of two consecutive steps are nearly orthogonal, so the algorithm 'zigzags' to the maximum point.

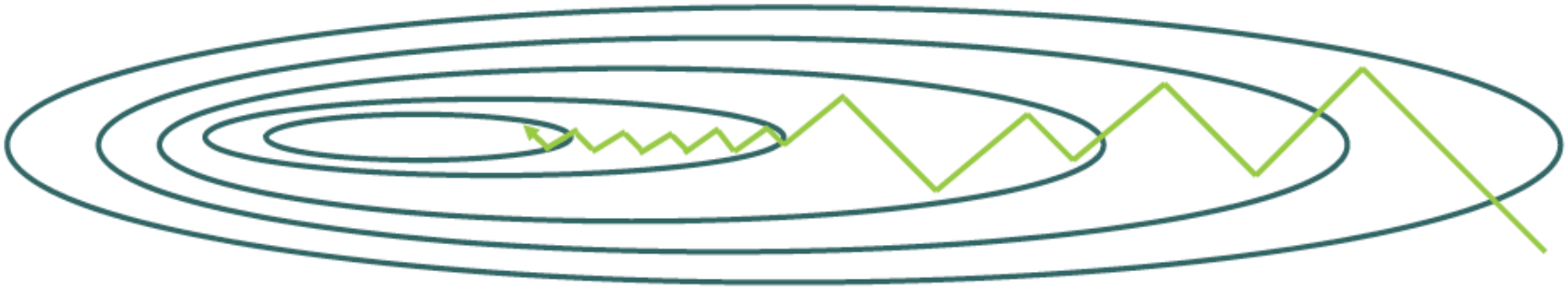
Steepest ascent (continued)

When $\alpha^{(k)}$ is chosen as $\arg\max_\alpha l(\theta^{(k)} + \alpha \dot{l}(\theta^{(k)}))$, the directions of two consecutive steps are orthogonal, i.e.,

$$[\dot{l}(\theta^{(k)})]' \dot{l}(\theta^{(k+1)}) = 0.$$

Proof: By the definition of $\alpha^{(k)}$ and $\theta^{(k+1)}$

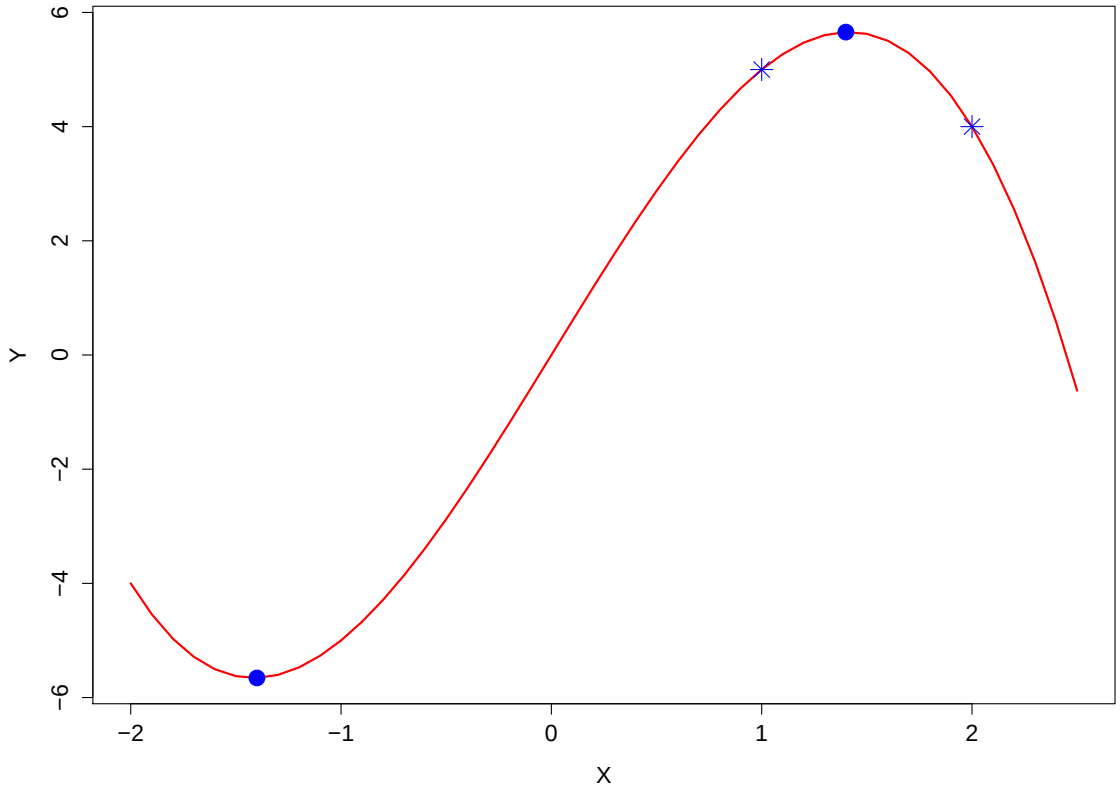
$$0 = \frac{\partial l(\theta^{(k)} + \alpha \dot{l}(\theta^{(k)}))}{\partial \alpha}\Big|_{\alpha=\alpha^{(k)}} = \dot{l}(\theta^{(k)} + \alpha^{(k)} \dot{l}(\theta^{(k)}))' \dot{l}(\theta^{(k)}) = \dot{l}(\theta^{(k+1)})' \dot{l}(\theta^{(k)}).$$



Example

Maximize the function

$$f(x) = 6x - x^3$$



Example: Steepest Ascent? Steepest Descent?


```
fun0 <- function(x) return(- x^3 + 6*x)   # target function
grd0 <- function(x) return(- 3*x^2 + 6)   # gradient

# Steepest Ascent Algorithm
Steepest_Ascent <- function(x, fun=fun0, grd=grd0, step=0.01, kmax=1000, tol1=1e-6, tol2=1e-4)
{
        diff <- 2*x  # use a large value to get into the following "while" loop
        k <- 0       # count iteration

        while ( all(abs(diff) > tol1*(abs(x)+tol2) )  & k <= kmax) # stop criteria
        {
                g_x <- grd(x)          # calculate gradient using x
                diff <- step * g_x     # calculate the difference used in the stop criteria
                x <- x + diff          # update x
                k <- k + 1             # update iteration
        }

        f_x = fun(x)

        return(list(iteration=k, x=x, f_x=f_x, g_x=g_x))
}
```

Example: Steepest Ascent? Steepest Descent? (cont.)

```
> Steepest_Ascent(x=2, step=0.01)

$iteration
[1] 117
$x
[1] 1.414228
$f_x
[1] 5.656854
$g_x
[1] -0.0001380379

> Steepest_Ascent(x=1, step=-0.01)

$iteration
[1] 159
$x
[1] -1.414199
$f_x
[1] -5.656854
$g_x
[1] 0.0001370128
```

2. Newton-Raphson: $R = -\ddot{l}(\theta^{(k)}) =$ observed information

$$d^{(k)} = [-\ddot{l}(\theta^{(k)})]^{-1}\dot{l}(\theta^{(k)})$$
$$\theta^{(k+1)} = \theta^{(k)} + [-\ddot{l}(\theta^{(k)})]^{-1}\dot{l}(\theta^{(k)})$$
$$\alpha^{(k)} = 1 \text{ for all } k$$

- Fast, quadratic convergence
- Need very good starting points
- Hessian may not be negative definite; the algorithm is perfectly happy to go down rather than up

Theorem: If R is positive definite, the equation set $Rd^{(k)} = \dot{l}(\theta^{(k)})$ has a unique solution for the direction $d^{(k)}$, and the direction ensures ascent of $l(\theta)$.

Proof: When R is positive definite, it is invertible. So we have a unique solution $d^{(k)} = R^{-1}\dot{l}(\theta^{(k)})$. Let

$$\theta^{(k+1)} = \theta^{(k)} + \alpha d^{(k)} = \theta^{(k)} + \alpha R^{-1}\dot{l}(\theta^{(k)}).$$

By Taylor expansion,

$$l(\theta^{(k+1)}) \approx l(\theta^{(k)}) + \alpha[\dot{l}(\theta^{(k)})]' R^{-1}\dot{l}(\theta^{(k)}).$$

The positive definite matrix R ensures that $l(\theta^{(k+1)}) > l(\theta^{(k)})$ for sufficiently small positive α.

Example: Newton Raphson


```
fun0 <- function(x) return(- x^3 + 6*x)   # target function
grd0 <- function(x) return(- 3*x^2 + 6)   # gradient
hes0 <- function(x) return(- 6*x)         # Hessian

# Newton-Raphson Algorithm
Newton_Raphson <- function(x, fun=fun0, grd=grd0, hes=hes0, kmax=1000, tol1=1e-6, tol2=1e-4)
{
        diff <- 2*x
        k <- 0

        while ( all(abs(diff) > tol1*(abs(x)+tol2) )  & k <= kmax)
        {
                g_x <- grd(x)
                h_x <- hes(x)         # calculate the second derivative (Hessian)
                diff <- -g_x/h_x      # calculate the difference used by the stop criteria
                x <- x + diff
                k <- k + 1
        }

        f_x = fun(x)

        return(list(iteration=k, x=x, f_x=f_x, g_x=g_x, h_x=h_x))
}
```

Example: Newton Raphson


```
> Newton_Raphson(x=2)

$iteration
[1] 5
$x
[1] 1.414214
$f_x
[1] 5.656854
$g_x
[1] -1.353229e-11
$h_x
[1] -8.485281

> Newton_Raphson(x=1)

$iteration
[1] 5
$x
[1] 1.414214
$f_x
[1] 5.656854
$g_x
[1] -1.353229e-11
$h_x
[1] -8.485281
```

4. Modification of Newton-Raphson

- **Fisher scoring**: replace $-\ddot{l}(\theta)$ with $-\mathrm{E}\ddot{l}(\theta)$
  - $-\mathrm{E}\ddot{l}(\theta) = \mathrm{E}\dot{l}(\theta)\dot{l}(\theta)'$ is always positive and stabilize the algorithm
  - $-\mathrm{E}\ddot{l}(\theta)$ can have a simpler form than $-\ddot{l}(\theta)$
  - Newton-Raphson and Fisher score are equivalent for parameter estimation in GLM with canonical exponential families.

- **Quasi-Newton**: aka "variable metric methods" or "secant methods". Approximate $\ddot{l}(\theta)$ in a way that
  - avoids calculating Hessian and its inverse
  - has convergence properties similar to Newton

Fisher Scoring: Example

In the Poisson regression model of n subjects,

- The responses $Y_i \sim \text{Poisson}(\lambda_i) = (Y_i!)^{-1}\lambda_i^{Y_i}e^{-\lambda_i}$. We know that $\lambda_i = \text{E}(Y_i|X_i)$.
- We relate the mean of Y_i to X_i by $g(\lambda_i) = X_i\beta$. Taking derivative on both sides,

$$g'(\lambda_i)\frac{\partial \lambda_i}{\partial \beta} = X_i \quad \Rightarrow \quad \frac{\partial \lambda_i}{\partial \beta} = \frac{X_i}{g'(\lambda_i)}$$

- Log likelihood: $l(\beta) = \sum_{i=1}^{n}(Y_i \log \lambda_i - \lambda_i)$, where λ_i's are such that $g(\lambda_i) = X_i\beta$.
- Maximum likelihood estimation: $\hat{\beta} = \arg\max_\beta l(\beta)$

Newton-Raphson needs

$$\begin{aligned}
\dot{l}(\beta) &= \sum_i \left(\frac{Y_i}{\lambda_i} - 1\right)\frac{\partial \lambda_i}{\partial \beta} = \sum_i \left(\frac{Y_i}{\lambda_i} - 1\right)\frac{1}{g'(\lambda_i)}X_i \\
\ddot{l}(\beta) &= -\sum_i \frac{Y_i}{\lambda_i^2}\frac{\partial \lambda_i}{\partial \beta}\frac{1}{g'(\lambda_i)}X_i - \sum_i \left(\frac{Y_i}{\lambda_i} - 1\right)\frac{g''(\lambda_i)}{g'(\lambda_i)^2}\frac{\partial \lambda_i}{\partial \beta}X_i \\
&= -\sum_i \frac{1}{\lambda_i}\frac{1}{g'(\lambda_i)^2}X_i^2 - \sum_i \left(\frac{Y_i}{\lambda_i} - 1\right)\frac{1}{\lambda_i}\frac{1}{g'(\lambda_i)^2}X_i^2 - \sum_i \left(\frac{Y_i}{\lambda_i} - 1\right)\frac{g''(\lambda_i)}{g'(\lambda_i)^3}X_i^2
\end{aligned}$$

Fisher Scoring: Example (continued)

Fisher scoring needs $\dot{l}(\beta)$ and

$$\mathrm{E}\left[\ddot{l}(\beta)\right] = -\sum_i \frac{1}{\lambda_i} \frac{1}{g'(\lambda_i)^2} X_i^2$$

which is $\ddot{l}(\beta)$ without the extra terms.

With the canonical link for Poisson regression:

$$g(\lambda_i) = \log \lambda_i,$$

we have

$$g'(\lambda_i) = \lambda_i^{-1} \quad \text{and} \quad g''(\lambda_i) = -\lambda_i^{-2}.$$

So that the extra terms equal to zero and we conclude that Newton-Raphson and Fisher scoring are equivalent.

1. Davidson-Fletcher-Powell QNR algorithm

Let $\Delta \dot{l}^{(k)} = \dot{l}(\theta^{(k)}) - \dot{l}(\theta^{(k-1)})$ and $\Delta\theta^{(k)} = \theta^{(k)} - \theta^{(k-1)}$. Approximate negative Hessian by

$$G^{(k+1)} = G^{(k)} + \frac{\Delta\theta^{(k)}(\Delta\theta^{(k)})'}{(\Delta\theta^{(k)})'\Delta\theta^{(k)}} - \frac{G^{(k)}\Delta\dot{l}^{(k)}(\Delta\dot{l}^{(k)})'G^{(k)}}{(\Delta\dot{l}^{(k)})'G^{(k)}\Delta\dot{l}^{(k)}}.$$

Use the starting matrix $G^{(0)} = I$.

Theorem: If the starting matrix $G^{(0)}$ is symmetric positive definite, the above formula ensures that every $G^{(k)}$ during the iteration is positive definite.

Data: (x_i, y_i) for $i = 1, \ldots, n$

Notation and assumptions

- Model: $y_i = h(x_i, \beta) + \epsilon_i$, where $\epsilon_i \overset{i.i.d}{\sim} N(0, \sigma^2)$ and $h(,)$ is known
- Residual: $e_i(\beta) = y_i - h(x_i, \beta)$
- Jacobian: $\{J(\beta)\}_{ij} = \frac{\partial h(x_i,\beta)}{\partial \beta_j} = -\frac{\partial e_i(\beta)}{\partial \beta_j}$, a $n \times p$ matrix

Goal: to obtain MLE $\hat{\beta} = \arg\min_\beta S(\beta)$, where $S(\beta) = \sum_i \{y_i - h(x_i, \beta)\}^2 = [e(\beta)]' e(\beta)$

We could use the previously-discussed **Newton-Raphson algorithm**.

- Gradient: $g_j(\beta) = \frac{\partial S(\beta)}{\partial \beta_j} = 2 \sum_i e_i(\beta) \frac{\partial e_i(\beta)}{\partial \beta_j}$, i.e., $g(\beta) = -2J(\beta)' e(\beta)$
- Hessian: $H_{jr}(\beta) = \frac{\partial^2 S(\beta)}{\partial \beta_j \partial \beta_r} = 2 \sum_i \{e_i(\beta) \frac{\partial^2 e_i(\beta)}{\partial \beta_j \partial \beta_r} + \frac{\partial e_i(\beta)}{\partial \beta_j} \frac{\partial e_i(\beta)}{\partial \beta_r}\}$

Towards Gauss-Newton algorithm

Recall in linear regression models, we minimize

$$S(\beta) = \sum_i \{y_i - x_i'\beta\}^2$$

Because $S(\beta)$ is a quadratic function, it is easy to get MLE

$$\hat{\beta} = (\sum_i x_i x_i')^{-1}(\sum_i x_i y_i)$$

Now in the nonlinear regression models, we want to minimize

$$S(\beta) = \sum_i \{y_i - h(x_i, \beta)\}^2$$

Idea: Approximate $h(x_i, \beta)$ by a linear function, iteratively at $\beta^{(k)}$

Given $\beta^{(k)}$ and by Taylor expansion of $h(x_i, \beta)$ at $\beta^{(k)}$, $S(\beta)$ becomes

$$S(\beta) \approx \sum_i \left\{ y_i - h(x_i, \beta^{(k)}) - (\beta - \beta^{(k)})' \frac{\partial h(x_i, \beta^{(k)})}{\partial \beta} \right\}^2 .$$

Gauss-Newton algorithm

1. Find a good starting point $\beta^{(0)}$

2. At step $k+1$,

   (a) Form $e(\beta^{(k)})$ and $J(\beta^{(k)})$

   (b) Use a standard linear regression routine to obtain
   $\delta^{(k)} = [J(\beta^{(k)})' J(\beta^{(k)})]^{-1} J(\beta^{(k)})' e(\beta^{(k)})$

   (c) Obtain the new estimate $\beta^{(k+1)} = \beta^{(k)} + \delta^{(k)}$

- Need good starting values
- Require $J(\beta^{(k)})' J(\beta^{(k)})$ to be invertible.

Generalized linear models

Data: (y_i, x_i) for $i = 1, \ldots, n$

Notation and assumptions

- Mean: $\mathrm{E}(y|x) = \mu$
- Link g: $g(\mu) = x'\beta$
- Variance function V: $\mathrm{Var}(y|x) = \phi V(\mu)$
- Log likelihood (exponential family): $l(\theta, \phi; y) = \{y\theta - b(\theta)\}/a(\phi) + c(y, \phi)$

We obtain

- Score function: $\dot{l} = \{y - b'(\theta)\}/a(\phi)$
- Observed information: $-\ddot{l} = b''(\theta)/a(\phi)$
- Mean (θ): $\mathrm{E}(y|x) = a(\phi)\mathrm{E}(\dot{l}) + b'(\theta) = b'(\theta)$
- Variance (θ, ϕ): $\mathrm{Var}(y|x) = \mathrm{E}(y - b'(\theta))^2 = a(\phi)^2\mathrm{E}(\dot{l}\dot{l}') = a(\phi)^2\mathrm{E}(-\ddot{l}) = b''(\theta)a(\phi)$

Canonical link: g such that $g(\mu) = \theta$, i.e. $g^{-1} = b'$

Generally we have $a(\phi) = \phi/w$, in which case ϕ will drop out of the following.

Model	Normal	Poisson	Binomial	Gamma
ϕ	σ^2	1	$1/m$	$1/\nu$
$b(\theta)$	$\theta^2/2$	$\exp(\theta)$	$\log(1+e^\theta)$	$-\log(-\theta)$
μ	θ	$\exp(\theta)$	$e^\theta/(1+e^\theta)$	$-1/\theta$
Canonical link g	identity	$\log$	logit	reciprocal
Variance function V	1	μ	$\mu(1-\mu)$	μ^2

Towards Iteratively reweighted least squares

In linear regression models, $\mathrm{E}(y_i|x_i) = x_i'\beta$, so we minimize

$$S(\beta) = \sum_i \{y_i - x_i'\beta\}^2$$

Because $S(\beta)$ is a quadratic function, it is easy to get MLE

$$\hat{\beta} = (\sum_i x_i x_i')^{-1}(\sum_i x_i y_i)$$

In generalized linear models, consider construct a similar quadratic function $S(\beta)$.

Question? Can we use

$$S(\beta) = \sum_i \{g(y_i) - x_i'\beta\}^2$$

Answer: No, because

$$\mathrm{E}\{g(y_i)|x_i\} \neq x_i'\beta$$

Idea: Approximate $g(y_i)$ by a linear function with expectation $x_i'\beta^{(k)}$, interactively at $\beta^{(k)}$

Iteratively reweighted least squares

Linearize $g(y_i)$ around $\hat{\mu}_i^{(k)} = g^{-1}(x_i'\beta^{(k)})$

$$\tilde{y}_i^{(k)} \equiv g(\hat{\mu}_i^{(k)}) + (y_i - \hat{\mu}_i^{(k)})g'(\hat{\mu}_i^{(k)})$$

Check the variances of $\tilde{y}_i^{(k)}$ and use them as weights

$$W_i^{(k)} = \left\{\mathrm{Var}(\tilde{y}_i^{(k)})\right\}^{-1} = \left[\{g'(\hat{\mu}_i^{(k)})\}^2 V(\hat{\mu}_i^{(k)})\right]^{-1}$$

Given $\beta^{(k)}$, we consider minimize

$$S(\beta) = \sum_i W_i^{(k)} \left\{\tilde{y}_i^{(k)} - x_i'\beta\right\}^2$$

IRLS algorithm:

1. Start with initial estimates, generally $\hat{\mu}_i^{(0)} = y_i$
2. Form $\tilde{y}_i^{(k)}$ and $W_i^{(k)}$
3. Estimate $\beta^{(k+1)}$ by regressing $\tilde{y}_i^{(k)}$ on x_i with weights $W_i^{(k)}$
4. Form $\hat{\mu}_i^{(k+1)} = g^{-1}(x_i'\beta^{(k+1)})$ and return to step 2.

Model	Poisson	Binomial	Gamma
$\mu = g^{-1}(\eta)$	e^{η}	$e^{\eta}/(1+e^{\eta})$	$1/\eta$
$g'(\mu)$	$1/\mu$	$1/[\mu(1-\mu)]$	$-1/\mu^2$
$V(\mu)$	μ	$\mu(1-\mu)$	μ^2

- McCullagh and Nelder (1983) justified IRLS by showing that IRLS is equivalent to Fisher scoring.
- In the case of the canonical link, IRLS is also equivalent to Newton-Raphson.
- IRLS is attractive because no special optimization algorithm is required, just a subroutine that computes weighted least square estimates.

Miscellaneous things

Dispersion parameter: When we do not take $\phi = 1$, the usual estimate is via the method of moments:

$$\hat{\phi} = \frac{1}{n-p} \sum_i \frac{(y_i - \hat{\mu}_i)^2}{V(\hat{\mu}_i)}$$

Standard errors:

$$\widehat{\mathrm{Var}}(\hat{\beta}) = \hat{\phi}(X' \widehat{W} X)^{-1}$$

Quasi likelihood: Pick a link and a variance function, and IRLS can proceed without worrying about the model. In other words, IRLS is a good thing!

IRLS example for Poisson regression


```
poisreg <- function(y, x, tol=1e-8) {

  ## get init value from lm
  lmfit <- lm(log(y+1)~x)
  b <- coef(lmfit);
  xb <- fitted(lmfit)

  ## iterate
  diff <- 1;
  maxiter <- 50;
  iter <- 0

  while(diff > tol & iter < maxiter) {

    ## form adjusted response
    mu <- exp(xb);
    ty <- xb + (y-mu)/mu

    ## create weights - weights are mu
    w <- mu
```

```
    ## weighted regression
    fit <- lm(ty~x, weights=w)

    ## get updated beta and fitted values
    b.old <- b;
    b <- coef(fit);
    xb <- fitted(fit)

    ## check for convergence
    diff <- sum((b-b.old)^2)
    iter <- iter+1
    cat("iter", iter, ": b=",b, ", diff=", diff, "\n")
  }

  ## calculate variance/covariance matrix
  X1 <- cbind(int=rep(1, length(x)), x)
  v <- solve(t(X1) %*% diag(w) %*% X1)

  ## return
  list(b=b, v=v, niter=iter)
}
```

IRLS example for Poisson regression (continued)

```
> ### simulation
> n=100;  beta=2;
> X=rnorm(n, mean=1, sd=.3)
> Y=rpois(n, exp(beta*X))

> result=poisreg(Y,X)
iter 1 : b= 0.07046616 1.990243 , diff= 0.02322995
iter 2 : b= 0.06573701 1.992766 , diff= 2.873326e-05
iter 3 : b= 0.06581331 1.992703 , diff= 9.809174e-09

> result
$b
(Intercept)           x
 0.06581331  1.99270302

$v
              int            x
int  0.003326508 -0.002614283
x   -0.002614283  0.002122114

$niter
[1] 3
```