STAT3003: 统计算法基础

优化问题（1）

曾靖

中国科学技术大学管理学院

大纲

大纲

1. 二分法
2. 牛顿法
3. 最大似然估计与 Fisher 得分法

本节内容来自于《统计计算使用 *R*》第 *11* 章和《统计计算》第六章。

1 引言

1.1 优化问题

- 在统计计算中广泛用到求最小（最大）值点。如
  - 极大似然估计 (MLE)
  - 最小二乘估计 (OLS)
- 最大值问题 $\max_{x\in\mathbb{R}^d} f(x)$ 和最小值问题 $\min_{x\in\mathbb{R}^d} f(x)$ 统称为优化问题。
- 我们只讨论最小值问题，因为最大值问题总能转化为最小值问题，即

$$\max_{x\in\mathbb{R}^d} f(x) = -\min_{x\in\mathbb{R}^d}\{-f(x)\}.$$

- 一般而言，优化问题没有显式解，这时候需要通过迭代算法 *(iterative algorithm)* 来进行数值求解：

$$\cdots \xrightarrow{\text{iteration}} x_t \xrightarrow{\text{iteration}} x_{t+1} \xrightarrow{\text{iteration}} x_{t+2} \xrightarrow{\text{iteration}} \cdots.$$

- 由于是迭代算法，实际中需要找到合适的停止法则 *(stopping criteria)* 来停止算法的迭代。令 $\varepsilon > 0$ 为一个很小的常数，通常称为 tolerance，常用的停止法则如下：
  - 绝对收敛：$\|x_{t+1} - x_t\| < \varepsilon$;
  - 相对收敛：$\|x_{t+1} - x_t\| / \|x_t\| < \varepsilon$;
  - 其他收敛：$\|f'(x)\| < \varepsilon$.

- ε 并不是越小越好。过小的 ε 可能导致算法在真实解附近反复跳跃而不能收敛。
- 为了保险起见，我们设置一个最大迭代次数，防止算法失败导致无法停止。

1.2 收敛速率

- 记优化问题第 t 步的变量为 x_t, 设

$$\lim_{t\to\infty} x_t = x^*,$$

我们通常用 *Q (Quotient)*-收敛和 *R(Root)*-收敛来刻画算法的收敛速率。

1.2 收敛速率

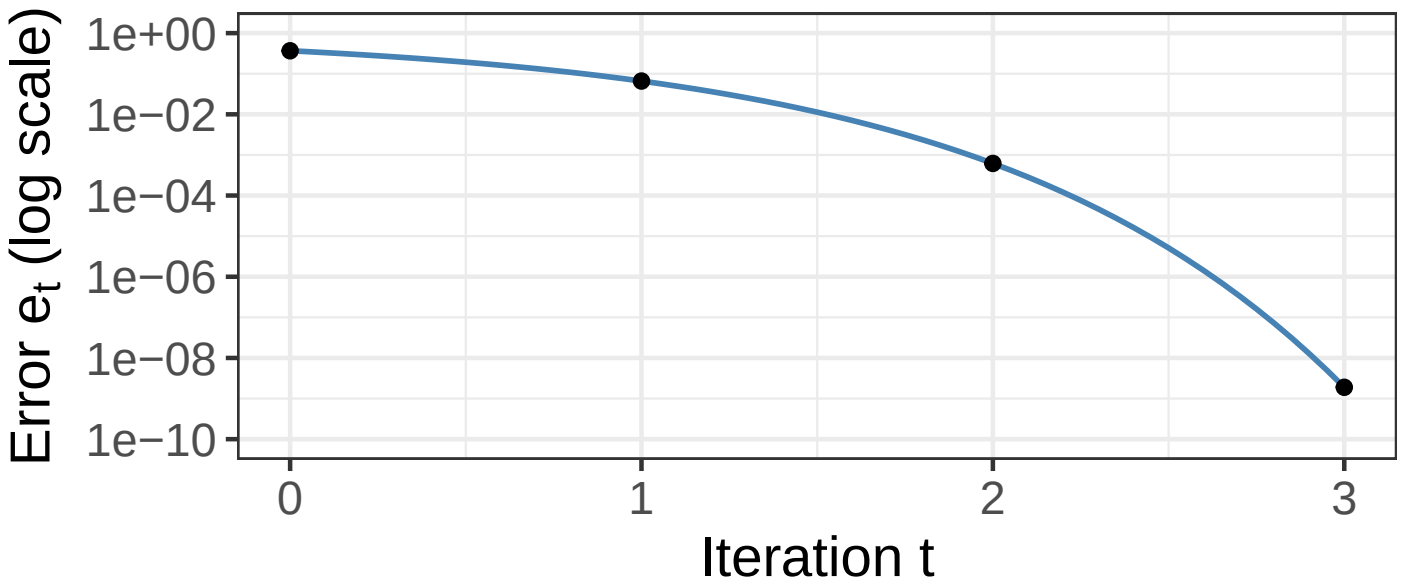
Q-收敛: 记

$$\boxed{Q = \overline{\lim}_{t\to\infty} \frac{\|x_{t+1} - x^*\|}{\|x_t - x^*\|},}$$

记误差 $e_t = \|x_t - x^*\|$。下面分别介绍超线性、线性和次线性收敛。

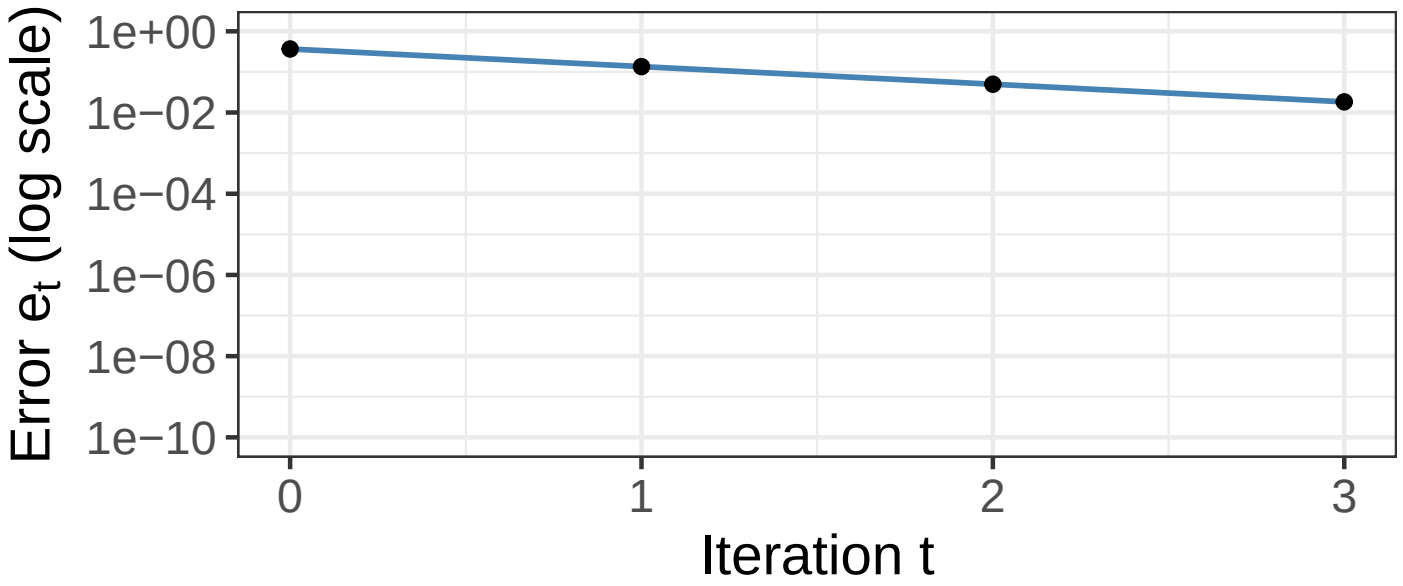
1.2 收敛速率：超线性收敛

- 超线性收敛：$Q = 0$。
  - 例如 $e_t = \exp(-\exp(t))$，则 $e_{t+1}/e_t = \exp(-(e-1)\exp(t)) \to 0$。
  - 取对数得 $\log e_t = -\exp(t)$，随 t 下降得越来越快，对数纵轴下曲线越来越陡。



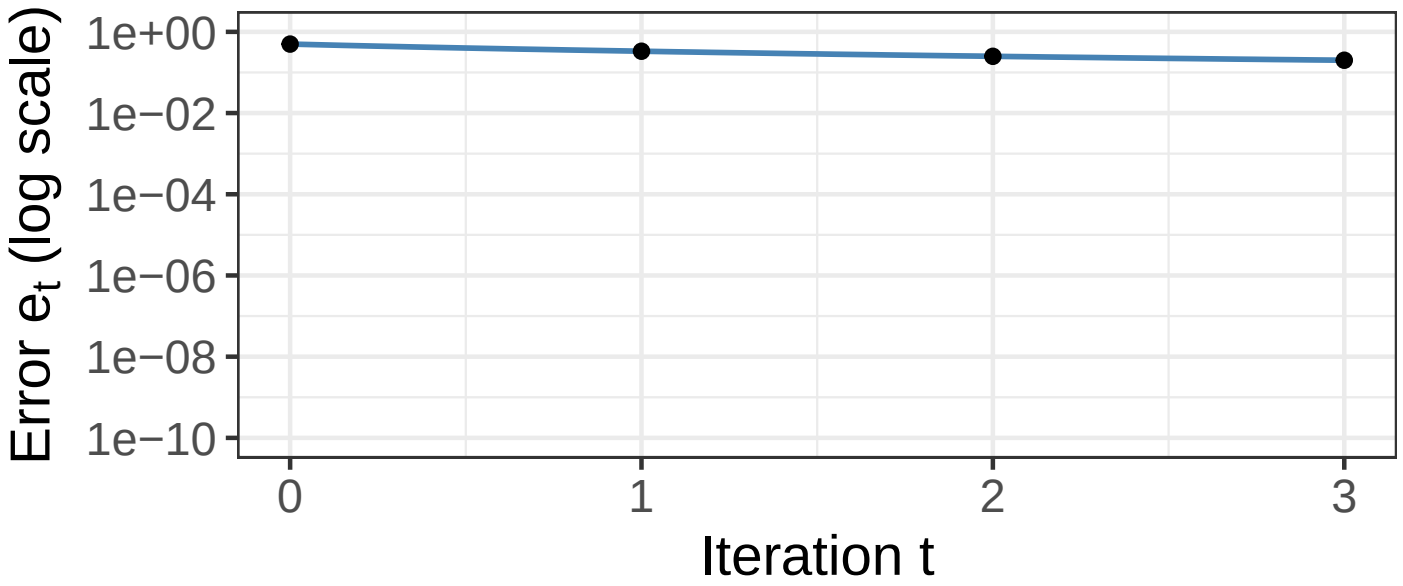
1.2 收敛速率：线性收敛

- 线性收敛：$0 < Q < 1$。
  - 例如 $e_t = \exp(-(t+1))$，则 $e_{t+1}/e_t = \exp(-1)$。
  - 取对数得 $\log e_t = -(t+1)$ 是 t 的线性函数，对数纵轴下呈直线。



1.2 收敛速率：次线性收敛

- 次线性收敛：$Q = 1$。
  - 例如 $e_t = 1/(t+2)$，则 $e_{t+1}/e_t = (t+2)/(t+3) \to 1$。
  - 取对数得 $\log e_t = -\log(t+2)$，随 t 下降得越来越慢，对数纵轴下曲线越来越平缓。



1.2 收敛速率：收敛阶

- 在超线性收敛的基础上，如果存在 $p > 1$ 使得

$$\overline{\lim}_{t\to\infty} \frac{\|x_{t+1} - x^*\|}{\|x_t - x^*\|^p} = C < \infty,$$

则称算法依 p 阶收敛到 x^*。

- 二阶收敛的例子：设误差序列为

$$e_t = 2^{-2^t}.$$

因而

$$\frac{e_{t+1}}{e_t^2} = 1,$$

故 x_t 二阶收敛到 x^*。

- 到达一定精度 $\varepsilon > 0$ 需要的迭代次数来经常被用来刻画收敛速度：
  - 超线性收敛: $O(\log(\log(1/\varepsilon)))$;
  - 线性收敛: $O(\log(1/\varepsilon))$;
  - 次线性收敛: $O(1/\varepsilon)$.

1.2 收敛速率

R-收敛: 如果 Q 不存在，我们利用另一个收敛序列来刻画其收敛性，称为 R 收敛。

- 假设 $\|x_t - x^*\| \leq y_t$, 如果序列 $\{y_t\}$ 是线性收敛，则称 $\{x_t\}$ 是 R-线性收敛。相应地有 R-超线性收敛，R-次线性收敛。
- 例如，若 $\|x_t - x^*\| \leq y_t = q^t$，$0 < q < 1$，则 $\{x_t\}$ 是 *R*-线性收敛的。

2 一维问题中的优化算法

2.1 二分法求根

- 优化问题与与求根问题密切相关，例如，若一元目标函数 $f(x)$ 连续可微，则极值点一定是 $g(x) = f'(x) = 0$ 的根。
- 理论根据：介值定理
  - 设函数 $g(x)$ 在闭区间 $[a, b]$ 上连续并且 $g(a)g(b) < 0$，则由介值定理可知至少存在一个点 $x^* \in [a, b]$ 使得 $g(x^*) = 0$，即 x^* 为 $g(x)$ 的根。如果 $g(x)$ 在 $[a, b]$ 上是严格单调函数，则根 x^* 存在且唯一。

算法：

- Input: 函数 $g(x)$，左右端点 a, b。令 $g_a = g(a), g_b = g(b)$。
- Loop: 重复一下步骤直到满足某停止法则为止
  1. (计算中点函数值) 令 $c = \frac{1}{2}(a+b), g_c = g(c)$。
  2. (更新区间) 如果 $g_a g_c \leq 0$, 则令 $b = c$；否则，令 $a = c$。
- Output：c

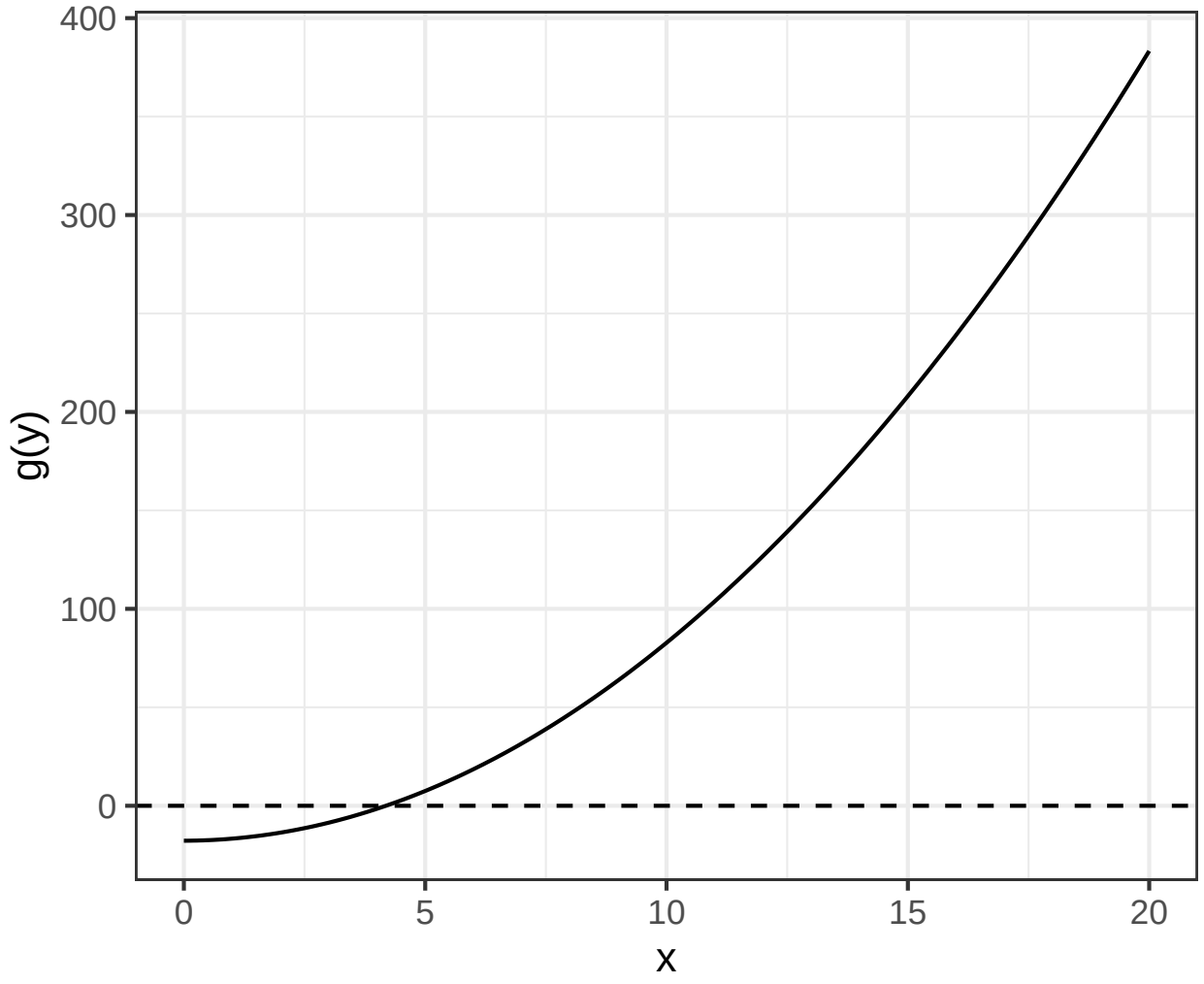
例 2.1

求解方程

$$a^2 + y^2 + \frac{2ay}{n-1} = n - 2.$$

- 令 $g(y) = a^2 + y^2 + \frac{2ay}{n-1} - (n-2)$.
- 交互实验：逐步观察二分法更新端点

```r
g0 <- function(y, a = 0.5, n = 20) {
    a^2 + y^2 + 2 * a * y/(n - 1) - (n - 2)
}
g0_data <- data.frame(x = seq(0, 20,
    length.out = 301))
ggplot(g0_data, aes(x = x, y = g0(x))) +
    geom_line() + geom_hline(yintercept = 0,
    linetype = 2) + labs(x = "x", y = "g(y)") +
    theme_bw()
```


400
300
200
100
0
g(y)
0
5
10
15
20
x

```r
# Definition of function g
g <- function(y, a, n) {
    a^2 + y^2 + 2 * a * y/(n - 1) - (n - 2)
}

a <- 0.5
n <- 20
b0 <- 0  # left endpoint
b1 <- 5 * n  # right endpoint
it <- 0  # iteration
eps <- .Machine$double.eps^0.25  # tolerance
```

```r
# solve using bisection
r <- seq(b0, b1, length = 3)
y <- c(g(r[1], a, n), g(r[2], a, n), g(r[3], a,
    n))
if (y[1] * y[3] > 0) {
    stop("Endpoints must have opposite signs")
}
track <- c(r[2], y[2])
while (it < 1000 && abs(y[2]) > eps) {
    it <- it + 1
    if (y[1] * y[2] < 0) {
        r[3] <- r[2]
        y[3] <- y[2]
    } else {
        r[1] <- r[2]
        y[1] <- y[2]
    }
    r[2] <- (r[1] + r[3])/2
    y[2] <- g(r[2], a = a, n = n)
    track <- rbind(track, c(r[2], y[2]))
}
```

```
r[2]  # root
```

```
## [1] 4.186845
```

```
y[2]  # value at root
```

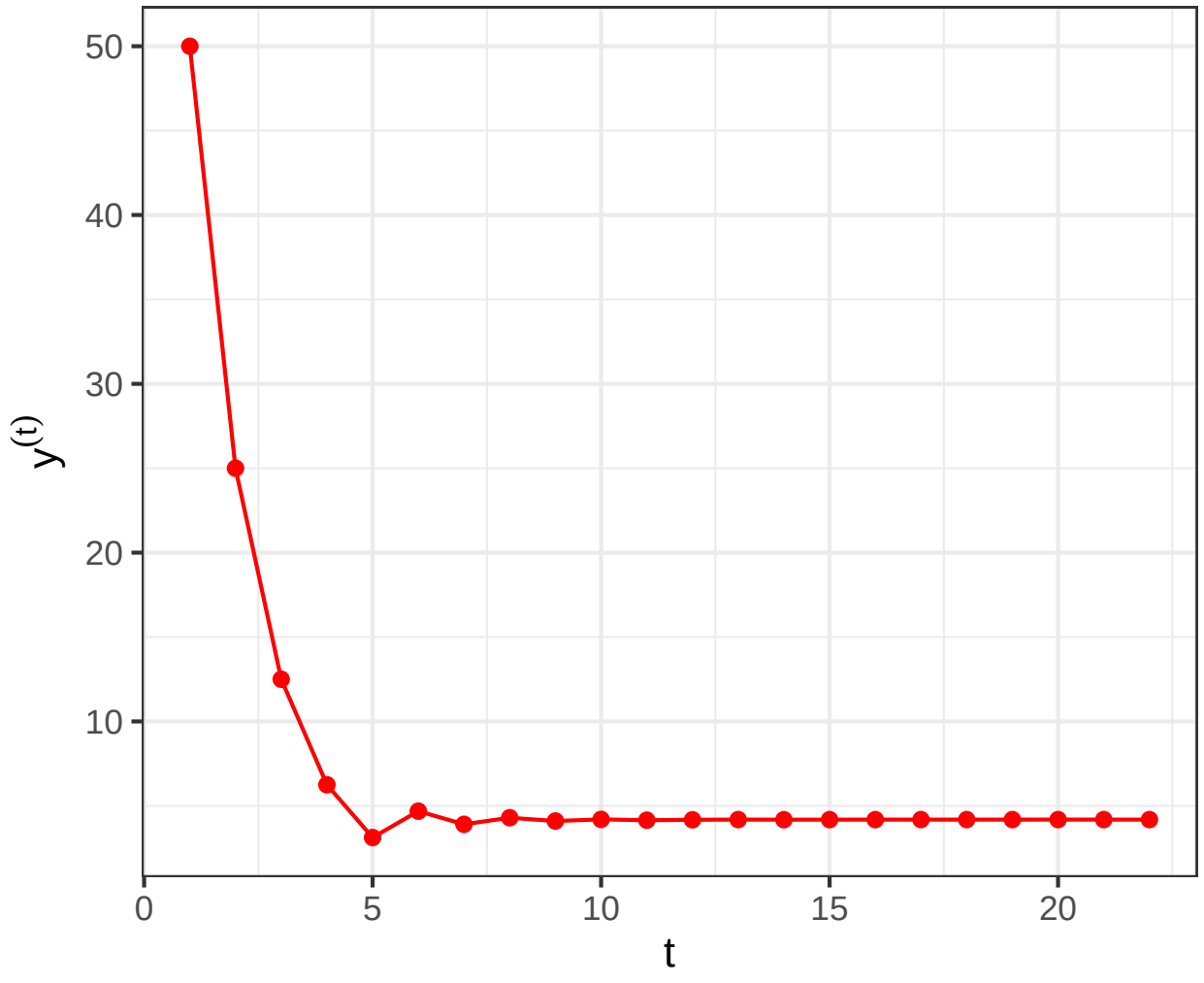
```
## [1] 2.984885e-05
```

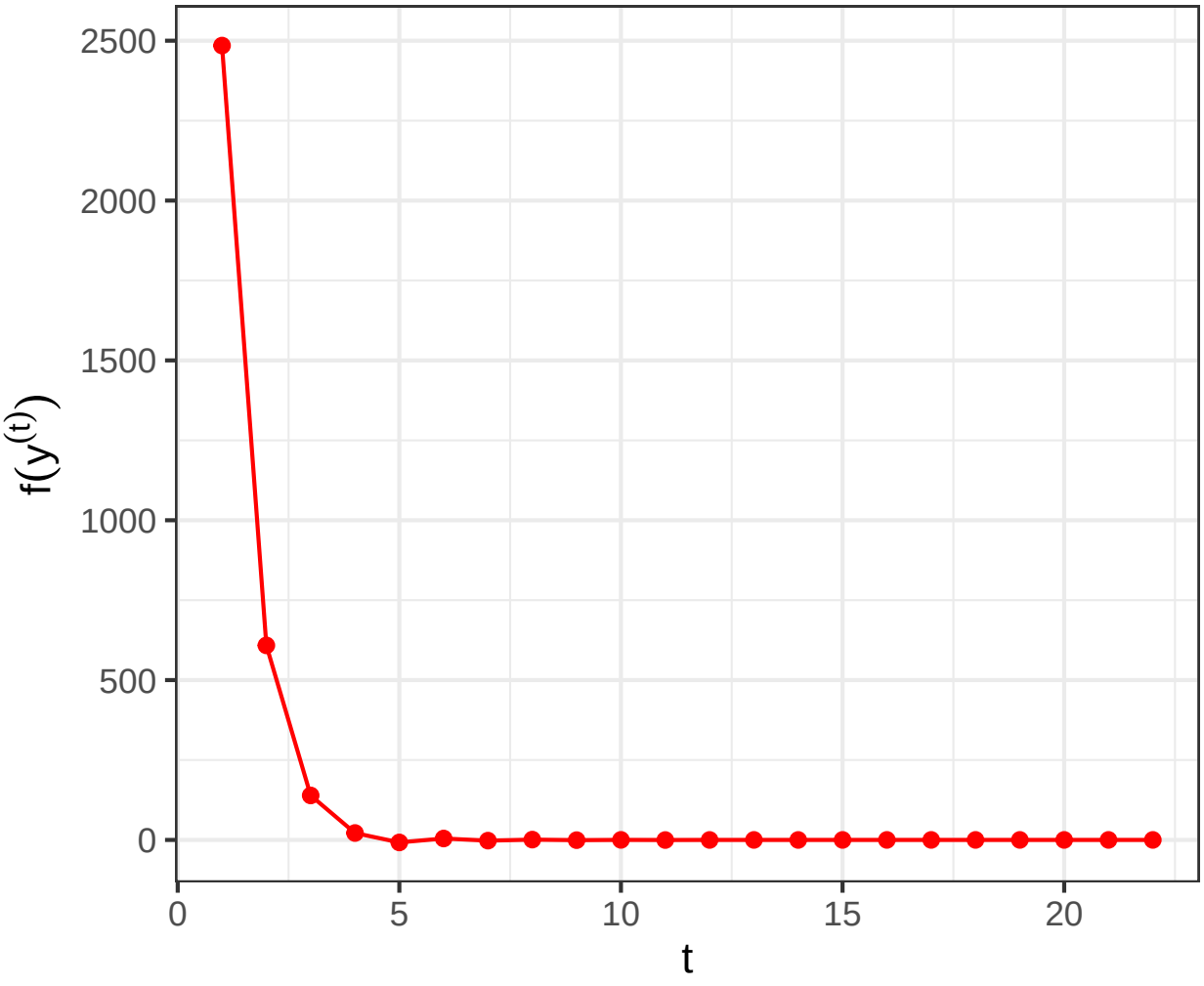
```
it  # the number of iterations
```

```
## [1] 21
```

```r
library(latex2exp)
ggplot_series(track[, 1], xlab = TeX("$t$"),
    ylab = TeX("$y^{(t)}$"), color = "red")
```

```r
ggplot_series(track[, 2], xlab = TeX("$t$"),
    ylab = TeX("$f(y^{(t)})$"), color = "red")
```


50
40
30
20
10
$y^{(t)}$
0
5
10
15
20
t


2500
2000
1500
1000
500
0
f(y(t))
0
5
10
15
20
t

2.2 一维问题中的牛顿法

牛顿法 (Newton's method or Newton–Raphson method) 是一种求根方法

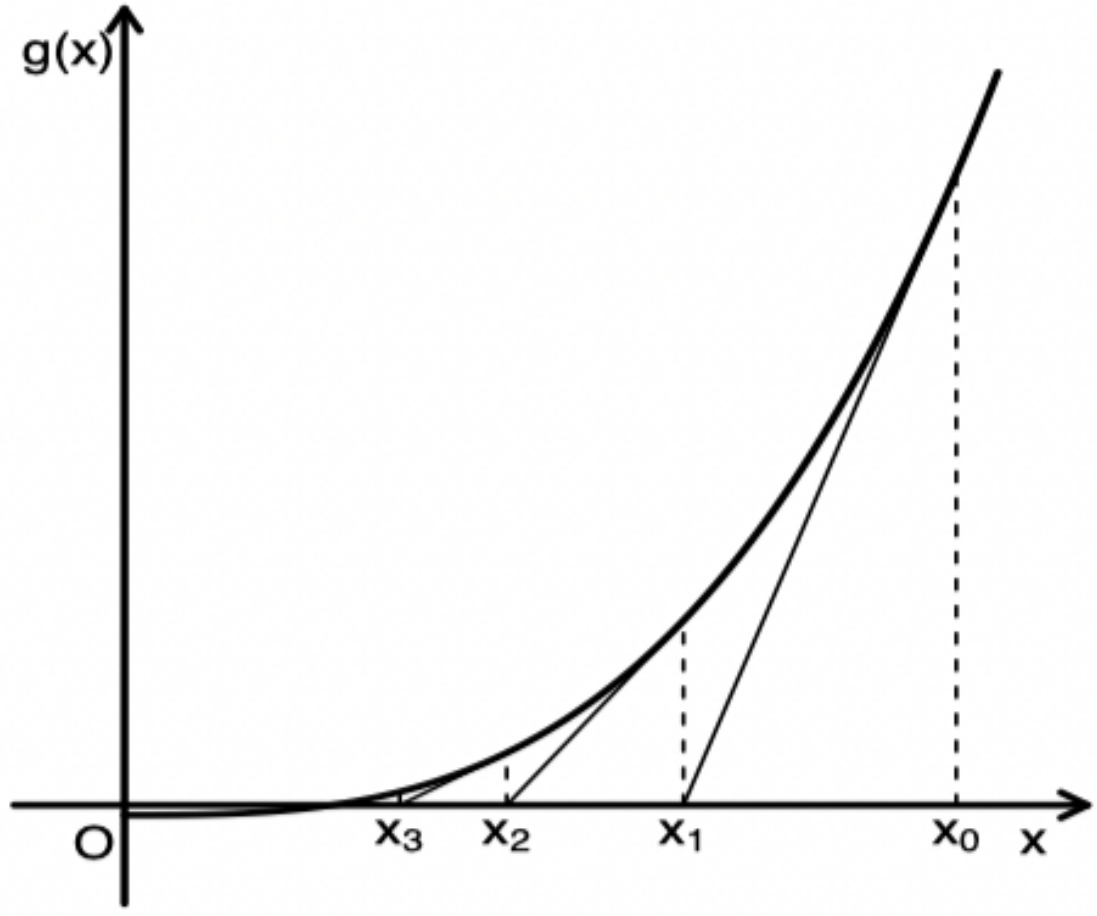
- 设目标函数 $g(x)$ 有连续一阶导数，我们希望得到 $g(x)$ 的根。
- 记根为 x^*，设 x_0 是 x^* 附近的一个点，在 x_0 附近对 $g(x)$ 作一阶 Taylor 展开近似得

$$g(x) \approx g(x_0) + g'(x_0)(x - x_0).$$

- 令 $g(x) = 0$，则近似解为 $x_1 = x_0 - g(x_0)/g'(x_0)$.
- 我们迭代地更新每一步解:

$$\boxed{x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)}, \quad t = 0, 1, 2, \ldots.}$$

交互实验：逐步观察牛顿法切线求根



- 如果 $g'(x)$ 很难推导，可以利用**割线法**。简而言之，用 $(x_{t-1}, g(x_{t-1}))$ 和 $(x_t, g(x_t))$ 连线的斜率代替 $g'(x_t)$。迭代公式为

$$x_{t+1} = x_t - \frac{x_t - x_{t-1}}{g(x_t) - g(x_{t-1})} g(x_t), \quad t = 0, 1, 2, \ldots.$$

- 如果问题为**求解连续二阶可导函数** $f(x)$ 的最小值点，则问题等价于求解方程的根：

$$g(x) = f'(x) = 0.$$

因此，Newton 法求解 $f(x)$ 的最小值点的迭代公式为

$$\boxed{x_{t+1} = x_t - \frac{f'(x_t)}{f''(x_t)}, \quad t = 0, 1, 2, \ldots.}$$

3 多维问题中的优化算法

3.1 牛顿法

- 考虑多维空间 $\mathbb{R}^p$ 中的最小值问题:

$$x^* = \arg\min_{x \in \mathbb{R}^p} f(x)$$

- 类似于一维问题，对于多维问题中的求解最小值问题，假设目标函数 f 二阶连续可导，则迭代公式为

$$\boxed{x_{t+1} = x_t - (\nabla^2 f(x_t))^{-1} \nabla f(x_t),}$$

这里 $\nabla^2 f(x_t) \in \mathbb{R}^{p \times p}$ 为 f 在 x_t 的 Hessian 阵。

- 在一维问题中，牛顿法是梯度下降法的特例。

优点

在一些正则化条件下，如果初始值接近于最小值点，则牛顿法有二阶收敛速率，即

$$\overline{\lim}_{t\to\infty} \frac{\|x_{t+1} - x^*\|}{\|x_t - x^*\|^2} = C < \infty.$$

缺点

- 每次迭代中需要计算 Hessian 阵的逆，使得牛顿法计算量非常大；
- 牛顿法的性能十分依赖 Hessian 矩阵 $\nabla^2 f(x_t)$，如果 $\nabla^2 f(x_t)$ 不满秩，则其逆不存在，算法无法进行。

牛顿法的改进：正则化

- 当 Hessian 矩阵不可逆，通过给 $\nabla^2 f(x_t)$ 加一个正定阵来保证正定型，即 $\nabla^2 f(x_t) \leftarrow \nabla^2 f(x_t) + \lambda I$，这里 $\lambda > 0$ 为某个常数。（*思考：与 ridge regression 有相似之处*）

例 3.1

考虑具有共线性的回归模型：

$$y = x_1 + 2x_2 + 3(x_1 + x_2) + \varepsilon.$$

- 设计矩阵和响应向量为

$$X_n = \begin{pmatrix} x_{11} & x_{12} & x_{11} + x_{12} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{n1} + x_{n2} \end{pmatrix} \in \mathbb{R}^{n \times 3}.$$

  其中 $Y_n = (y_1, \dots, y_n)^\top \in \mathbb{R}^n$。

- 我们用调整过 Hessian 矩阵的 Newton's method 来计算 OLS 估计量。

$$\hat{\beta} \in \arg\min_{\beta \in \mathbb{R}^3} \frac{1}{2n} \|Y_n - X_n \beta\|_2^2.$$

- 目标函数为

$$f(\beta) = \frac{1}{2n}\|Y_n - X_n\beta\|_2^2.$$

- 梯度和 Hessian 分别为

$$\nabla f(\beta) = \frac{1}{n}X_n^\top(X_n\beta - Y_n), \qquad \nabla^2 f(\beta) = \frac{1}{n}X_n^\top X_n.$$

- 因为三个变量之间存在共线性，所以目标函数 f 的 Hessian 矩阵不可逆。用

$$H_\lambda = \frac{1}{n}X_n^\top X_n + \lambda I_3, \qquad \lambda > 0$$

替代 Hessian，使其正定且可逆；I_3 为三阶单位矩阵。

例 3.1：正则化牛顿迭代

- 从初值 $\beta^{(0)}$ 出发，每次计算梯度，并用修正后的 Hessian 更新：

$$\begin{aligned}\beta^{(t+1)} &= \beta^{(t)} - H_\lambda^{-1}\nabla f(\beta^{(t)}) \\ &= \beta^{(t)} - (X_n^\top X_n + n\lambda I_3)^{-1} X_n^\top (X_n\beta^{(t)} - Y_n).\end{aligned}$$

- 以下代码取 $\beta^{(0)} = 0$、$\lambda = 10^{-5}$，当梯度足够小或达到最大迭代次数时停止。

例 3.1：代码

```
x <- matrix(rnorm(200), 100, 2)
x <- cbind(x, x[, 1] + x[, 2])
svd(x)$d
```

```
## [1] 1.603105e+01 9.277932e+00 1.448787e-15
```

```r
y <- x %*% 1:3 + rnorm(100)
iter <- 0
b <- rep(0, 3)

n <- nrow(x)
grad <- crossprod(x, x %*% b - y)/n
H <- crossprod(x)/n + diag(1e-05, 3)
while (norm(grad) > 0.001 & iter < 100) {
    iter <- iter + 1
    b <- b - solve(H, grad)
    grad <- crossprod(x, x %*% b - y)/n
    print(b)
}
```

```
##             [,1]
## [1,] 1.034506
## [2,] 1.955485
## [3,] 2.989991
```

牛顿法的改进（二）：拟牛顿法

在牛顿法的每一步迭代 $x_{t+1} = x_t - (\nabla^2 f(x_t))^{-1} \nabla f(x_t)$ 中，将 Hessian 阵 $\nabla^2 f(x_t)$ 换成一个正定的矩阵 $\mathbf{H}_t$.

- $\mathbf{H}_t$ 是迭代得到的。$\mathbf{H}_{t+1} = \mathbf{H}_t + \Delta_t$, 其中 Δ_t 称为修正矩阵；
- 并且在迭代中也考虑步长，即求得

$$s_t = \text{argmin}_{s>0} f(x_t - s\mathbf{H}_t^{-1} \nabla f(x_t)).$$

- 这样的方法称为拟牛顿法。

- 算法如下：
  1. 寻找最优步长: $s_t = \text{argmin}_{s>0} f(x_t - s\mathbf{H}_t^{-1}\nabla f(x_t))$;
  2. 更新 x_t: $x_{t+1} = x_t - s_t\mathbf{H}_t^{-1}\nabla f(x_t)$;
  3. 更新修正矩阵: Δ_t;
  4. 更新矩阵 $\mathbf{H}_t$: $\mathbf{H}_{t+1} = \mathbf{H}_t + \Delta_t$.

$$(x_t, \mathbf{H}_t) \xrightarrow{\text{选择步长}} s_t \xrightarrow{\text{更新位置}} x_{t+1} \xrightarrow{\text{计算修正}} \Delta_t \xrightarrow{\text{更新矩阵}} \mathbf{H}_{t+1}$$

- 迭代地构造每一步的矩阵 $\mathbf{H}_t$, 并使得其满足如下条件：
  - $\mathbf{H}_t$ 是迭代得到的。$\mathbf{H}_{t+1} = \mathbf{H}_t + \Delta_t$, 其中 Δ_t 称为修正矩阵；
  - $\mathbf{H}_t$ 是正定矩阵（思考：为什么?）
  - $\mathbf{H}_{t+1}$ 满足 Newton 条件：

$$\mathbf{H}_{t+1}\delta_t = \zeta_t,$$

这里 $\delta_t = x_{t+1} - x_t, \zeta_t = \nabla f(x_{t+1}) - \nabla f(x_t).$

拟牛顿法：Newton 条件的由来

- 在新位置 x_{t+1} 处，对梯度作一阶 Taylor 展开：

$$\nabla f(x_t) \approx \nabla f(x_{t+1}) + \nabla^2 f(x_{t+1})(x_t - x_{t+1}).$$

整理得

$$\nabla^2 f(x_{t+1})\delta_t \approx \zeta_t.$$

- 用 H_{t+1} 近似新位置的 Hessian，并要求它精确匹配本次位移与梯度变化，得到

$$\boxed{H_{t+1}\delta_t = \zeta_t.}$$

拟牛顿算法之一：*BFGS(Broyden-Fletcher-Goldfarb-Shanno)* 算法

- 修正矩阵为

$$\Delta_t = \frac{\zeta_t \zeta_t^\top}{\zeta_t^\top \delta_t} - \frac{(\mathbf{H}_t \delta_t)(\mathbf{H}_t \delta_t)^\top}{\delta_t^\top \mathbf{H}_t \delta_t},$$

  这里 $\delta_t = x_{t+1} - x_t, \zeta_t = \nabla f(x_{t+1}) - \nabla f(x_t)$.

- 在拟牛顿法中，我们可以直接迭代更新 $\mathbf{H}_t^{-1}$，无需每一步对 $\mathbf{H}_t$ 求逆。例如，在 BFGS 算法中，$\mathbf{H}_{t+1}^{-1}$ 可由 $\mathbf{H}_t^{-1}, \zeta_t, \delta_t$ 直接更新得到。记 $\mathbf{V}_t = \mathbf{H}_t^{-1}$，则

$$\mathbf{V}_{t+1} = \mathbf{V}_t + \frac{(\zeta_t^\top \delta_t + \zeta_t^\top \mathbf{V}_t \zeta_t)(\delta_t \delta_t^\top)}{(\zeta_t^\top \delta_t)^2} - \frac{(\mathbf{V}_t \zeta_t \delta_t^\top) + (\mathbf{V}_t \zeta_t \delta_t^\top)^\top}{\zeta_t^\top \delta_t}$$

- BFGS 算法为：
  1. 寻找最优步长: $s_t = \text{argmin}_{s>0} f(x_t - s\mathbf{V}_t \nabla f(x_t))$;
  2. 更新 x_{t+1}: $x_{t+1} = x_t - s_t \mathbf{V}_t \nabla f(x_t)$;
  3. 计算 ζ_t 和 δ_t;
  4. 更新矩阵 $\mathbf{V}_{t+1}$

$$(x_t, \mathbf{V}_t) \xrightarrow{\text{选择步长}} s_t \xrightarrow{\text{更新位置}} x_{t+1} \xrightarrow{\text{计算修正量}} (\zeta_t, \delta_t) \xrightarrow{\text{更新矩阵}} \mathbf{V}_{t+1}$$

牛顿法的改进：阻尼牛顿法

- 如果在拟牛顿法中保持 $\mathbf{H}_t = \nabla^2 f(x_t)$，则称该方法为阻尼牛顿法。

- 算法如下：
  1. 寻找最优步长：
     $s_t = \text{argmin}_{s>0} f(x_t - s(\nabla^2 f(x_t))^{-1}\nabla f(x_t))$;
  2. 更新：$x_{t+1} = x_t - s_t(\nabla^2 f(x_t))^{-1}\nabla f(x_t)$.

3.2 Fisher 得分法

最大似然估计（*MLE*）

- 在参数估计中，一类重要的方法是最大似然估计 (MLE)。记随机变量 X 的密度函数为 $f(x \mid \theta)$, 其中未知参数为 $\theta \in \mathbb{R}^p$。当给定样本 $x_1, \ldots, x_n$, 似然函数为

$$L(\theta) = \prod_{i=1}^{n} f(x_i \mid \theta).$$

- 为了计算方便，我们通常考虑对数似然函数:

$$\ell(\theta) = \sum_{i=1}^{n} \ln f(x_i \mid \theta).$$

- MLE 通过求解最大化对数似然函数来估计参数 θ:

$$\hat{\theta} = \arg\max_{\theta \in \mathbb{R}^p} \ell(\theta).$$

- MLE 问题等价于如下的最小值问题:

$$\hat{\theta} = \arg\min_{\theta} -\ell(\theta).$$

- 事实上，*MLE* 问题只是优化问题中的一个特例。由于似然函数的特殊性（具有统计解释），我们可以采取一些特殊的方法去处理 MLE 问题。

3.2 Fisher 得分法

- Fisher 得分法是 Newton 法在 MLE 问题中的应用。
- 回顾：在牛顿法中，迭代公式为

$$\theta_{t+1} = \theta^{(t)} - (\nabla^2 f(\theta^{(t)}))^{-1} \nabla f(\theta^{(t)}).$$

- MLE 问题中的牛顿法：

$$\theta_{t+1} = \theta^{(t)} - (\nabla^2 \ell(\theta^{(t)}))^{-1} \nabla \ell(\theta^{(t)}).$$

- 因此，每次迭代中都需要计算一个 $p \times p$ 矩阵的逆

$$-(\nabla^2 \ell(\theta^{(t)}))^{-1}.$$

这往往是牛顿法最耗费算力的地方。

Fisher 信息矩阵

$$
\begin{aligned}
I_n(\theta) &= \mathrm{Cov}\left(\frac{\partial \ell(\theta)}{\partial \theta}\right) \\
&= E\left\{\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)^{\top}\right\} \\
&\quad - \left\{E\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)\right\}\left\{E\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)\right\}^{\top}.
\end{aligned}
$$

- Fisher 信息阵可以用来度量随机变量 X 所携带的关于参数 θ 的信息。Fisher 信息阵越小，意味着似然函数关于 θ 越平整，包含 θ 的信息也就越少。

- 在一些正则条件下，可得

$$E\left(\frac{\partial \ell(\theta)}{\partial \theta}\right) = 0.$$

因此，

$$I_n(\theta) = E\left\{\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)\left(\frac{\partial \ell(\theta)}{\partial \theta}\right)^{\top}\right\}.$$

- 另外，在一些正则条件下，

$$\boxed{I_n(\theta) = -E\left\{\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^{\top}}\right\}.}$$

- 当样本量足够大且 $\theta^{(t)}$ 接近真值时，我们有近似结果

$$I_n(\theta^{(t)}) \approx -\left(\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^\top} \Big|_{\theta=\theta^{(t)}} \right).$$

  注意，左边是一个期望，右边是在给定 n 个样本下的值。

- 因此，在 MLE 问题中，我们可以利用 Fisher 信息阵来近似替代 Newton 法中的 Hessian 阵。
  - 在一些模型中，取期望后表达式会简化，可以直接写出解析形式，从而简化计算。

- Fisher 得分法解 MLE 问题的迭代公式:

$$\boxed{\theta_{t+1} = \theta^{(t)} + (I_n(\theta^{(t)}))^{-1} \nabla \ell(\theta^{(t)}).}$$

- Fisher 得分法广泛应用于广义线性模型（GLM）下的 MLE 问题。
  - 在 GLM 中，Fisher 得分迭代可等价地写为迭代加权最小二乘（IRLS，Iteratively Reweighted Least Squares）算法，即每一步求解一个加权最小二乘问题。

例 3.2: MLE in logistic regression

考虑如下 *logistic* 回归模型，$i = 1, \dots, n$:

$$Y_i \mid X_i \sim \text{Bernoulli}(p_i),$$
$$\text{logit}(p_i) = \log \frac{p_i}{1 - p_i} = \beta^\top X_i.$$

其中 $X_i \in \mathbb{R}^p$，$\beta \in \mathbb{R}^p$ 为未知参数。等价地，

$$p_i = \Pr(Y_i = 1 \mid X_i) = \frac{\exp(\beta^\top X_i)}{1 + \exp(\beta^\top X_i)}.$$

给定 *i.i.d.* 样本 (X_i, Y_i)，$i = 1, \dots, n$，求参数 β 的 *MLE*。

似然函数

- 给定协变量 $x_1, \ldots, x_n$，似然函数为

$$L(\beta) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1 - y_i}.$$

- 对数似然函数为

$$\begin{aligned} \ell(\beta) &= \sum_{i=1}^{n} \{ y_i \log p_i + (1 - y_i) \log(1 - p_i) \} \\ &= \sum_{i=1}^{n} \left\{ y_i \beta^\top x_i - \log \left(1 + \exp(\beta^\top x_i) \right) \right\}. \end{aligned}$$

- 梯度和 Hessian 矩阵分别为

$$\nabla \ell(\beta) = \sum_{i=1}^{n} (y_i - p_i) x_i, \quad \nabla^2 \ell(\beta) = -\sum_{i=1}^{n} p_i (1 - p_i) x_i x_i^\top,$$

- 由于 Hessian 矩阵不含随机成分 (x_i 样本固定)，因此牛顿法和 Fisher 得分法是相同的。

IRLS 迭代

记 $X = (x_1, \dots, x_n)^\top$，$y = (y_1, \dots, y_n)^\top$。

- 在第 t 步，计算概率与权重

$$p_{it} = \frac{\exp(x_i^\top \beta^{(t)})}{1 + \exp(x_i^\top \beta^{(t)})}, \qquad w_{it} = p_{it}(1 - p_{it}).$$

  记 $p_t = (p_{1t}, \dots, p_{nt})^\top$，$W_t = \mathrm{diag}(w_{1t}, \dots, w_{nt})$。

- 定义工作响应

$$z_t = X\beta^{(t)} + W_t^{-1}(y - p_t).$$

- Newton ／ Fisher 更新可写成加权最小二乘的形式：

$$\boxed{\begin{aligned}\beta^{(t+1)} &= \beta^{(t)} + (X^\top W_t X)^{-1} X^\top (y - p_t) \\ &= (X^\top W_t X)^{-1} X^\top W_t z_t\end{aligned}}$$

- 每次更新等价于求解加权最小二乘问题：

$$\beta^{(t+1)} = \arg\min_{\beta} \ \frac{1}{2} \sum_{i=1}^{n} w_{it}(z_{it} - x_i^\top \beta)^2.$$

```r
# Data generation
n <- 1000
x <- rnorm(n)
beta <- 0.3
prob <- exp(x * beta)/(1 + exp(x * beta))
y <- rbinom(n, 1, prob = prob)

## Initialization
iter <- 0
beta <- 0.3
g <- 1
```

```r
while (abs(g) > sqrt(.Machine$double.eps) &&
    iter < 100) {
    iter <- iter + 1
    p <- exp(x * beta)/(1 + exp(x * beta))
    h <- -sum(p * (1 - p) * x^2)
    g <- sum((y - p) * x)
    beta <- beta - h^-1 * g

    cat(iter, beta, abs(g), "\n")
}
```

```
## 1 0.2900838 2.576295
## 2 0.2901061 0.005821046
## 3 0.2901061 2.898313e-08
## 4 0.2901061 1.69309e-14
```

交互实验 Lab 05：Fisher 得分法估计 logistic 回归参数