第8回　母数の推定
最尤解
（関数の最適化）

データの同時分布

$Y_i, i = 1, 2, \ldots, n$ を独立な観測値とする。$f(y_i|\theta)$ をデータの確率関数（密度関数）とする。データの同時分布の確率関数は

$$f(y_i, y_2, \ldots, y_n|\theta) = \prod_{i=1}^{n} f(y_i|\theta) \tag{1}$$

である。これは、母数の値が θ の時に、$y_i, y_2, \ldots, y_n$ というデータを観測する確率（密度）であり、その値が高いほど、それらのデータ系列（パタン）が出現する確率が高い。

尤度関数 Likelihood Function

この同時確率関数を母数 θ の関数と見たものが尤度関数である。

$$L(\theta) = f(y_i, y_2, \ldots, y_n|\theta) \ = \prod_{i=1}^{n} f(y_i|\theta) \tag{2}$$

これは、データ系列を与えられた時に、母数の値がどの程度尤もらしいかを表す値と解釈することが出来る。ただし、θ の確率を与えるものではない。（θ は確率変数ではないので確率は付与できない。）

なお、パラメタを含まない部分は、尤度関数の定義から落としても良い。その意味で、

$$L(\theta) \propto f(y_i, y_2, \ldots, y_n|\theta) \tag{3}$$

と書く場合がある。

尤度関数の対数を対数尤度関数という。

$$lnL(\theta) = \log L(\theta) = \sum_{i=1}^{n} \log f(y_i|\theta) \tag{4}$$

多くの場合、対数尤度関数の方が尤度関数よりも簡単な形をしている。

ロジスティク回帰

データ

$$(x_i, y_i), i = 1, 2, \ldots, n \quad \text{ただし } y_i \text{ は } 0 \text{ か } 1$$

これを x_i の異なる値ごとに度数分布の形にしたもの

$$(x_i, f_i, r_i), i = 1, 2, \ldots, N \quad \text{ただし } f_i \text{ は試行数 } r_i \text{ は成功数}$$

モデル

確率は $(0,)$ の区間に入らなければならないので、

$$\Pr(Y_i = 1|x_i) = \frac{1}{1 + \exp(-(\alpha x_i + \beta))} = P(x_i|\alpha, \beta) = \pi_i$$

とする。これは

$$\log \frac{\Pr(Y_i = 1|x_i)}{1 - \Pr(Y_i = 1|x_i)} = \alpha x_i + \beta$$

と同値である。なお、モデルのパラメタは $\boldsymbol{\xi} = \{\alpha, \beta\}$ であり、

$$\begin{aligned} \boldsymbol{\mathcal{E}}(Y_i|x_i) &= P(x_i|\alpha, \beta) \ = \ \pi_i \\ var(Y_i|x_i) &= \pi_i(1 - \pi_i) \end{aligned}$$

となる。また、(f_i, r_i) を試行数と度数として、

$$\begin{aligned} \boldsymbol{\mathcal{E}}(R_i|x_i) &= f_i \times P(x_i|\alpha, \beta) \ = \ f_i\pi_i \\ var(R_i|x_i) &= f_i\pi_i(1 - \pi_i) \end{aligned}$$

尤度関数

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} P(x_i|\alpha,\beta)^{y_i}(1 - P(x_i|\alpha,\beta))^{1-y_i}$$

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} P(x_i|\alpha,\beta)^{r_i}(1 - P(x_i|\alpha,\beta))^{f_i-r_i}$$

対数尤度関数

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} y_i \log P(x_i|\alpha,\beta) + (1 - y_i)\log(1 - P(x_i|\alpha,\beta))$$

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} r_i \log P(x_i|\alpha,\beta) + (f_i - r_i)\log(1 - P(x_i|\alpha,\beta))$$

これを最大にするように x_i を求める。すなわち、連立方程式

$$\frac{\partial\, lnL}{\partial\, \alpha} = 0, \quad \frac{\partial\, lnL}{\partial\, \beta} = 0,$$

を解く。答えは解析的には解けない。

数値例

Data1

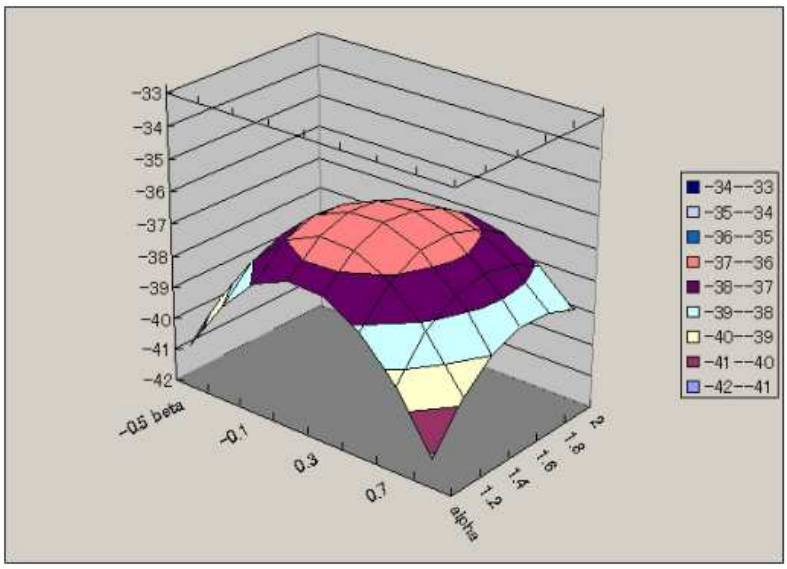
Obs	X	F	R	P	phat
1	-3.0	1	0	0.000	0.015
2	-2.6	2	0	0.000	0.026
3	-2.3	3	0	0.000	0.040
4	-1.9	4	0	0.000	0.071
5	-1.5	5	1	0.200	0.120
6	-1.1	6	1	0.166	0.198
7	-0.8	7	2	0.285	0.277
8	-0.4	8	1	0.125	0.407
9	0.0	9	7	0.777	0.552
10	0.4	8	7	0.875	0.689
11	0.8	7	5	0.714	0.799
12	1.1	6	6	1.000	0.860
13	1.5	5	4	0.800	0.917
14	1.9	4	4	1.000	0.952
15	2.3	3	2	0.666	0.972
16	2.6	2	2	1.000	0.982
17	3.0	1	1	1.000	0.990

最尤解の探索

ALPHA	BETA	LL
1.0	-0.5	-40.6408
1.0	-0.3	-38.9813
1.0	-0.1	-37.8991
1.0	0.1	-37.3991
1.0	0.3	-37.4813
1.0	0.5	-38.1408
1.0	0.7	-39.3681
1.0	0.9	-41.1493
1.2	-0.5	-39.6113
1.2	-0.3	-38.0549
1.2	-0.1	-37.0249
1.2	0.1	-36.5249
1.2	0.3	-36.5549
1.2	0.5	-37.1113
1.2	0.7	-38.1873
1.2	0.9	-39.7725
1.4	-0.5	-39.1354
1.4	-0.3	-37.6707
1.4	-0.1	-36.6870
1.4	0.1	-36.1870
1.4	0.3	-36.1707 *
1.4	0.5	-36.6354
1.4	0.7	-37.5759
1.4	0.9	-38.9847
1.6	-0.5	-39.0962
1.6	-0.3	-37.7122
1.6	-0.1	-36.7692
1.6	0.1	-36.2692
1.6	0.3	-36.2122
1.6	0.5	-36.5962
1.6	0.7	-37.4173
1.6	0.9	-38.6698
1.8	-0.5	-39.4032
1.8	-0.3	-38.0898
1.8	-0.1	-37.1823
1.8	0.1	-36.6823
1.8	0.3	-36.5898
1.8	0.5	-36.9032
1.8	0.7	-37.6195
1.8	0.9	-38.7345
2.0	-0.5	-39.9865
2.0	-0.3	-38.7348
2.0	-0.1	-37.8585
2.0	0.1	-37.3585

最尤解

α	1.4642 (0.861)
β	0.2127 (-0.14)



最尤解

α	1.4642 (0.861)
β	0.2127 (-0.14)

関数の最適化

一般に，ある関数 $r(\boldsymbol{\xi})$ の極値（極小値または極大値のことであるが，本節では極小値とする）を与えるパラメタ $\boldsymbol{\xi}$ の値を数値的に求めることを関数の最適化と呼ぶが，その方法は，何らかの方法で求めた $\boldsymbol{\xi}$ の初期値から，$r(\boldsymbol{\xi})$ が減少する方向 $\boldsymbol{d}$ へある分量 α だけ修正する，

$$\boldsymbol{\xi}_{(\ell+1)} = \boldsymbol{\xi}_{(\ell)} + \alpha \boldsymbol{d}_{(\ell)} \tag{54}$$

という形の繰り返し計算でパラメタの値を更新していくという形をとる．なお、この関数 $r(\boldsymbol{\xi})$ は目的関数 objective function と呼ばれる。

数値的最適化の方法には様々なものがあるが，それらの主な違いは，更新の方向 $\boldsymbol{d}_{(\ell)}$ をどの様にして定めるかによる．一般的には、以下に示すように、$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$ を $r(\boldsymbol{\xi})$ のパラメタの各要素に関する一次偏微分係数ベクトル (gradient) を $\boldsymbol{\xi} = \boldsymbol{\xi}_{(\ell)}$ で評価したものとして、

$$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})' \boldsymbol{d}_{(\ell)} < 0 \tag{55}$$

であれば

$$r(\boldsymbol{\xi}_{(\ell+1)}) \leq r(\boldsymbol{\xi}_{(\ell)}) \tag{56}$$

を満たすステップサイズ α が存在する。

- とりあえず R でやってみる。

R の nlminb 関数

- 目的関数 $r(\xi)$ を与えるとその関数の最小値を与える母数 ξ を Newton-Raphson 法で探してくれる。
- 引数は

  パラメタの初期値ベクトル、目的関数

  必要に応じて、

  目的関数の補助的パラメタ、gradient、hessian

  制御用のパラメタリスト　（rel.tol, iter.max, trace 等）

```
resnlm <- nlminb( param, func )
resnlm <- nlminb( param, func, control=list(trace=1) )
resnlm <- nlminb( param, func, data=data )
resnlm <- nlminb( param, func, gradient=g )
resnlm <- nlminb( param, func, gradient=g, hessian=H )
```

- 結果はリスト

  $par, $objective, $convergence(=0), $iterations

- その他の関数： optimize, optim, nlm , optimx

ロジスティク回帰

尤度関数

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} \pi_i(\boldsymbol{\xi})^{y_i}(1-\pi_i(\boldsymbol{\xi}))^{1-y_i} \qquad (69)$$

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} \pi_i(\boldsymbol{\xi})^{r_i}(1-\pi_i(\boldsymbol{\xi}))^{f_i-r_i} \qquad (70)$$

対数尤度関数

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} y_i \log \pi_i(\boldsymbol{\xi}) + (1-y_i)\log(1-\pi_i(\boldsymbol{\xi})) \qquad (71)$$

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} r_i \log \pi_i(\boldsymbol{\xi}) + (f_i-r_i)\log(1-\pi_i(\boldsymbol{\xi})) \qquad (72)$$

これを最大にするように $\boldsymbol{\xi}$ を求める。

R による計算

param が $\boldsymbol{\xi}$ に相当する。

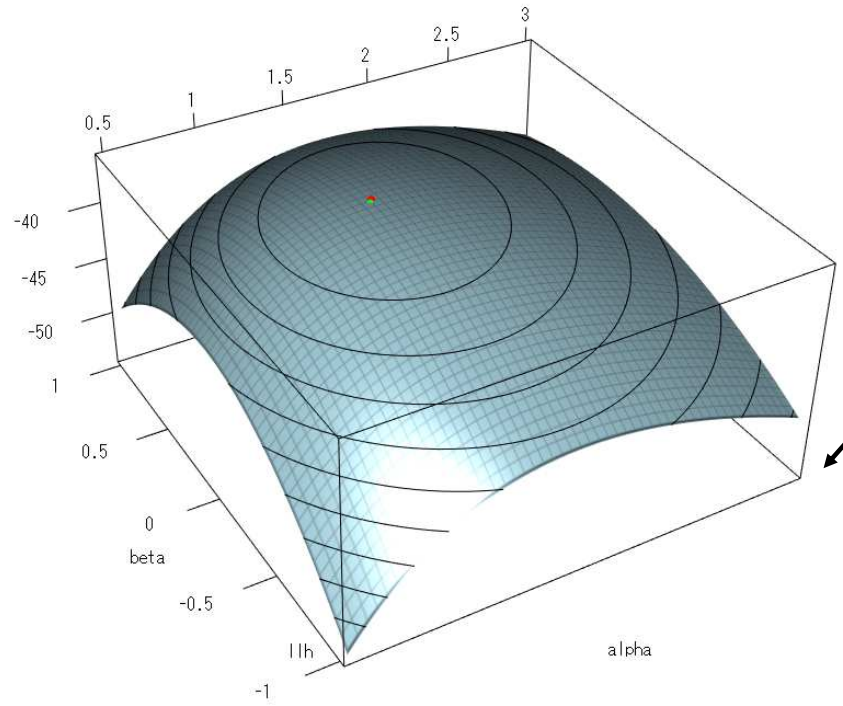
```
data <- data.frame( x, r, f)

# probability (logistic function with slope and intercept, no 1.7)
P <- function( x, param ){
 a=param[1]; b=param[2]
 z=a*x+b
 P=1/(1+exp(-z))
 return( P )
} # end of P

# minus log likelihood
mllh <- function( param, data ){
 x=data$x; f=data$f; r=data$r
 P=P(x,param)
 llh=sum( r*log(P)+(f-r)*log(1-P) )
 return( -llh )
} # end of mllh

resnlm <- nlminb( c(3, -1), mllh, control=list(trace=1) )
```

ロジスティク回帰の尤度関数



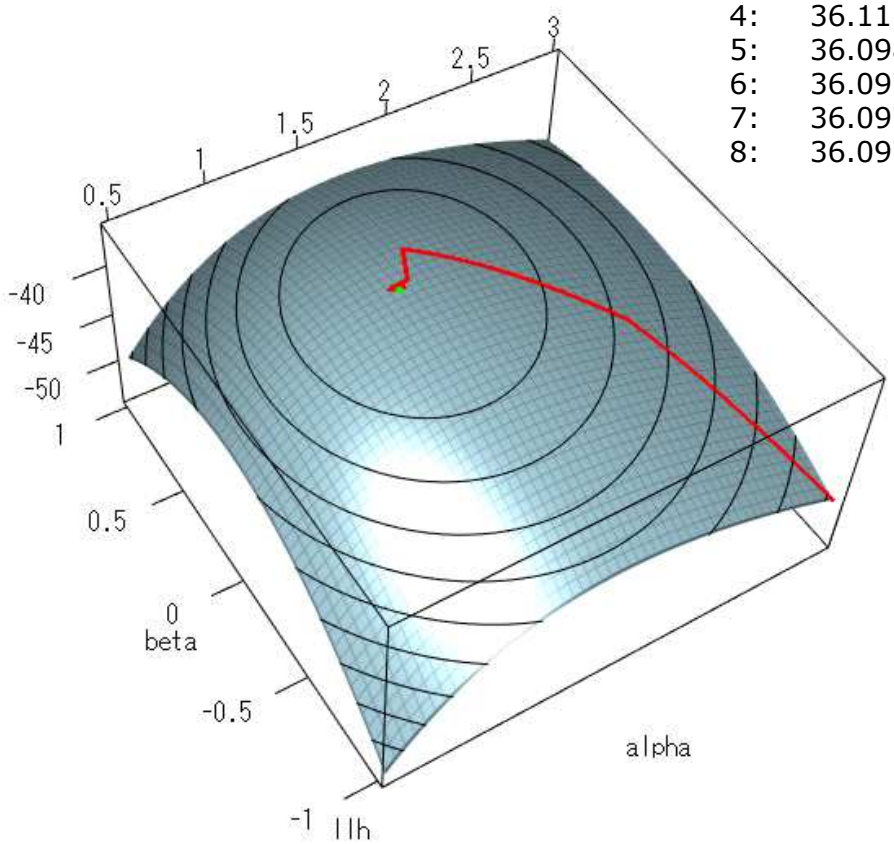
param=c(3,-1) から始める。

最尤解

α	1.4642 (0.861)
β	0.2127 (-0.14)

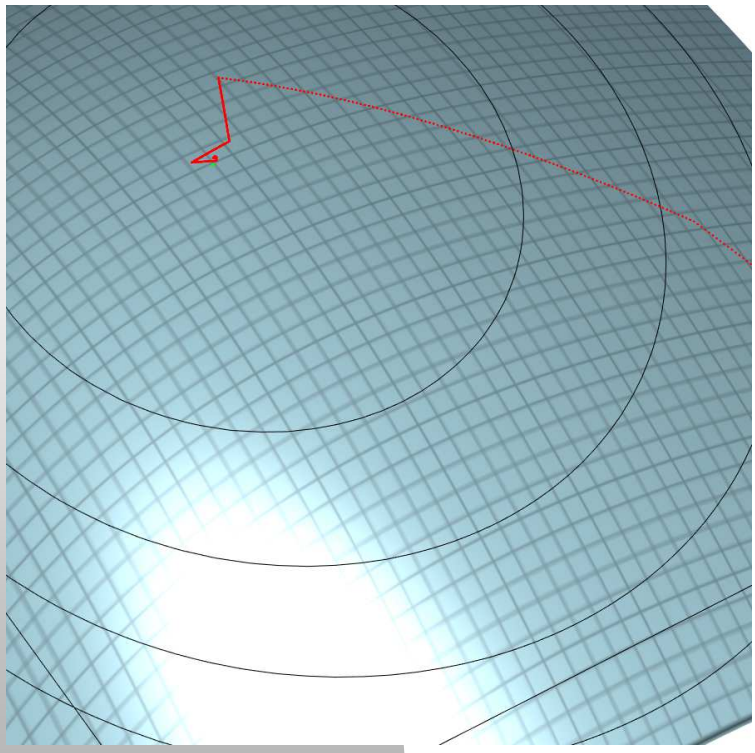
nlminb の経路

	-log likelihood	alpha	beta
0:	49.294500:	3.00000	-1.00000
1:	40.270717:	2.43365	-0.175834
2:	36.331303:	1.60680	0.386591
3:	36.113120:	1.51818	0.234972
4:	36.111554:	1.41473	0.224860
5:	36.098074:	1.46310	0.205857
6:	36.097812:	1.46518	0.214235
7:	36.097797:	1.46435	0.212764
8:	36.097797:	1.46434	0.212761



最尤解

α	1.4642 (0.861)
β	0.2127 (-0.14)



	-log likelihood	alpha	beta
0:	49.294500:	3.00000	-1.00000
1:	40.270717:	2.43365	-0.175834
2:	36.331303:	1.60680	0.386591
3:	36.113120:	1.51818	0.234972
4:	36.111554:	1.41473	0.224860
5:	36.098074:	1.46310	0.205857
6:	36.097812:	1.46518	0.214235
7:	36.097797:	1.46435	0.212764
8:	36.097797:	1.46434	0.212761

最尤解

α	1.4642	(0.861)
β	0.2127	(-0.14)

- 微分の話, gradient, Hessian, Jacobian

微分の話

1　微分、偏微分、数値微分

1.1　定義等

スカラー値を持つスカラー量 x の関数

$$y = f(x) \tag{1}$$

の x に関する微分 (derivative) を差分商 (difference quotient)

$$q(x) = \frac{f(x+h) - f(x)}{h} \tag{2}$$

の極限

$$f'(x) = \frac{d\,y}{d\,x} = \frac{d\,f(x)}{d\,x} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{3}$$

と定義される。関数 $f(x)$ が x で微分可能とは、h が上方ならびに下方から 0 に近づく極限が存在し、それらが一致する場合である。

$f(\boldsymbol{x})$ が $p \times 1$ のベクトル $\boldsymbol{x}$ の関数である場合、$f(\boldsymbol{x})$ の x_j に関する偏微分 (partial derivative) を

$$\frac{\partial\, f(\boldsymbol{x})}{\partial\, x_j} = \lim_{h \to 0} \frac{f(x_1, x_2, \ldots, x_{j-1}, x_j + h, x_{j+1}, \ldots, x_p) - f(\boldsymbol{x})}{h} \tag{4}$$

で定義する。また、これをまとめて $p \times 1$ のベクトルの形に書いたものを

$$\frac{\partial\, f(\boldsymbol{x})}{\partial\, \boldsymbol{x}} = \begin{pmatrix} \dfrac{\partial\, f(\boldsymbol{x})}{\partial\, x_1} \\ \dfrac{\partial\, f(\boldsymbol{x})}{\partial\, x_2} \\ \vdots \\ \dfrac{\partial\, f(\boldsymbol{x})}{\partial\, x_j} \\ \vdots \\ \dfrac{\partial\, f(\boldsymbol{x})}{\partial\, x_p} \end{pmatrix} \tag{5}$$

と記す。

gradient vector

1.2 Gradient, Hessian, Jacobian

$r(\boldsymbol{\xi})$ を $p \times 1$ のパラメタベクトル $\boldsymbol{\xi}$

$$\boldsymbol{\xi} = \begin{pmatrix} \xi_1 \\ \xi_2 \\ \vdots \\ \xi_j \\ \vdots \\ \xi_p \end{pmatrix} \tag{6}$$

を持つスカラー値の関数とする。r の $\boldsymbol{\xi}$ の各要素 ξ_j に関する一次微分を $\boldsymbol{\xi}$ と同じ $p \times 1$ の形に並べたものを r の gradient vector（勾配ベクトル）と呼び、$\nabla r(\boldsymbol{\xi})$ または $\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}}$ と記す。

$$\nabla r(\boldsymbol{\xi}) = \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} = \begin{pmatrix} \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_1} \\ \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_2} \\ \vdots \\ \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_j} \\ \vdots \\ \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_p} \end{pmatrix} \tag{7}$$

Hessian matrix

また、$\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}}$ の第 j 番目の要素 $\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_j}$ を ξ_i で微分したものを ij 要素として持つ $p \times p$ の行列を Hessian 行列と喚び $H(\boldsymbol{\xi})$ または $\frac{\partial^2 r(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}\ \partial \boldsymbol{\xi}'}$ と記す。

$$H(\boldsymbol{\xi}) = \frac{\partial^2 r(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}\ \partial \boldsymbol{\xi}'} = \left\{ \frac{\partial^2 r(\boldsymbol{\xi})}{\partial \xi_i\ \partial \xi_j} \right\} \tag{8}$$

この表記は $\frac{\partial^2 r(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}\ \partial \boldsymbol{\xi}'}$ が、$p \times 1$ のベクトル $\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}}$ の転置である $1 \times p$ のベクトルを $\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}'}$ と記し、その第 j 要素、すなわち第 j 列である $\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_j}$ を $\boldsymbol{\xi}$ で微分した結果である $p \times 1$ のベクトル $\frac{\partial \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \xi_j}}{\partial\, \boldsymbol{\xi}}$ を横に並べたものであることを示している。なお、$H(\boldsymbol{\xi})$ は対称行列となり、その第 j 番目の対角要素には $\frac{\partial^2 r(\boldsymbol{\xi})}{\partial \xi_j^2}$ が入る。

Jacobian matrix

一般的に、n 次元のベクトル値を持つ関数 $\boldsymbol{r}(\boldsymbol{\xi})$

$$\boldsymbol{r}(\boldsymbol{\xi}) = \begin{pmatrix} r_1(\boldsymbol{\xi}) \\ r_2(\boldsymbol{\xi}) \\ \vdots \\ r_i(\boldsymbol{\xi}) \\ \vdots \\ r_n(\boldsymbol{\xi}) \end{pmatrix} \tag{9}$$

一般的に、n 次元のベクトル値を持つ関数 $\boldsymbol{r}(\boldsymbol{\xi})$

をそのパラメタである $p \times 1$ のベクトル $\boldsymbol{\xi}$ の第 j 要素 ξ_j で微分したもの $\dfrac{\partial\, r_i(\boldsymbol{\xi})}{\partial\, \xi_j}$ を $n \times p$ の行列の形に並べたものを $\nabla \boldsymbol{r}$ もしくは $\dfrac{\partial\, \boldsymbol{r}}{\partial\, \boldsymbol{\xi}}$ と記しヤコビアン行列 (Jacobia matrix) と呼ぶ。

$$\nabla \boldsymbol{r} = \frac{\partial\, \boldsymbol{r}(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} = \left\{ \frac{\partial\, r_i(\boldsymbol{\xi})}{\partial\, \xi_j} \right\} = \begin{pmatrix} \dfrac{\partial\, r_1(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}'} \\ \dfrac{\partial\, r_2(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}'} \\ \vdots \\ \dfrac{\partial\, r_i(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}'} \\ \vdots \\ \dfrac{\partial\, r_n(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}'} \end{pmatrix} \tag{10}$$

伝統的に、Jacobian の場合はその列にパラメタ $\boldsymbol{\xi}$ を配置するため、Jacobian の第 i 行には $r_i(\boldsymbol{\xi})$ の gradient の転置が入っていることになる。すなわち、スカラー値を返す関数 $r(\boldsymbol{\xi})$ の Jacobian 行列は gradient の転置である。同様に、 $r(\boldsymbol{\xi})$ の Hessian は Jacobian の表記を使えば

$$H(\boldsymbol{\xi}) = \left(\nabla \frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} \right)' \tag{11}$$

となる。

微分の連鎖律 chain rule 合成関数の微分

$q \times 1$ のベクトル $\boldsymbol{y}$ が実は $p \times 1$ のベクトル $\boldsymbol{x}$ の関数として

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}) = \begin{pmatrix} y_1(\boldsymbol{x}) \\ y_2(\boldsymbol{x}) \\ y_q(\boldsymbol{x}) \end{pmatrix} \tag{12}$$

として表現できる場合、スカラー値を取る $\boldsymbol{y}$ の関数 $r(\boldsymbol{y})$ を x_j で偏微分したものは

$$\frac{\partial\, r(\boldsymbol{y})}{\partial\, x_j} = \sum_{i=1}^{q} \frac{\partial\, r(\boldsymbol{y})}{\partial\, y_i} \frac{\partial\, y_i}{\partial\, x_j} = \sum_{i=1}^{q} \frac{\partial\, r(\boldsymbol{y})}{\partial\, y_i} \frac{\partial\, y_i(\boldsymbol{x})}{\partial\, x_j} \tag{13}$$

である。したがって、これをまとめて $p \times 1$ の gradient の形で表現すれば

$$\frac{\partial\, r(\boldsymbol{y})}{\partial\, \boldsymbol{x}} = \left(\left(\frac{\partial\, r(\boldsymbol{y})}{\partial\, \boldsymbol{y}} \right)' \frac{\partial\, \boldsymbol{y}(\boldsymbol{x})}{\partial\, \boldsymbol{x}} \right)' \tag{14}$$

$$= \left(\frac{\partial\, \boldsymbol{y}}{\partial\, \boldsymbol{x}} \right)' \frac{\partial\, r(\boldsymbol{y})}{\partial\, \boldsymbol{y}} \tag{15}$$

となる。ただし、$\frac{\partial\, \boldsymbol{y}(\boldsymbol{x})}{\partial\, \boldsymbol{x}}$ は $q \times p$ の Jacobian 行列、$\frac{\partial\, r(\boldsymbol{y})}{\partial\, \boldsymbol{y}}$ は $q \times 1$ の $r(\boldsymbol{y})$ の gradient である。

また、$\boldsymbol{r}(\boldsymbol{y})$ を n 次元の関数とすればその $\boldsymbol{x}$ に関する $n \times p$ の Jacobian 行列は

$$\frac{\partial\, \boldsymbol{r}(\boldsymbol{y})}{\partial\, \boldsymbol{x}} = \frac{\partial\, \boldsymbol{r}(\boldsymbol{y})}{\partial\, \boldsymbol{y}} \frac{\partial\, \boldsymbol{y}}{\partial\, \boldsymbol{x}} \tag{16}$$

と書ける。ただし、$\frac{\partial\, \boldsymbol{r}(\boldsymbol{y})}{\partial\, \boldsymbol{y}}$ は $n \times q$ の Jacobian 行列である。

数値微分

1.4 数値微分

$f(x)$ の x に関する微分のを $x = a$ で評価したものは、たとえば、ごく小さな h の値を用いて

$$\left.\frac{d\, f(x)}{d\, x}\right|_{x=a} = f'(a) \approx \frac{f(a+h) - f(a-h)}{2h} \tag{17}$$

で近似的に求めることが出来るが、これを数値微分 (numerical differentiation) と呼ぶ。同様に、$f(\boldsymbol{x})$ の x_j に関する偏微分の値を $\boldsymbol{x} = \boldsymbol{a}$ で評価したものは

$$\left.\frac{\partial\, f(\boldsymbol{x})}{\partial\, x_j}\right|_{\boldsymbol{x}=\boldsymbol{a}} \approx \frac{f(a_1, a_2, \ldots, a_{j-1}, a_j + h, a_{j+1}, \ldots, a_p) - f(a_1, a_2, \ldots, a_{j-1}, a_j - h, a_{j+1}, \ldots, a_p)}{2h} \tag{18}$$

である。

R で数値微分

```
JacobianMat <- function( a, f, h=1e-06, ... ){
 # Function to calculate the difference-quotient approx Jacobian matrix
 # of f(x) at x=a
 #
 # Args:
 #   a    parameter vector of size p x 1
 #   f    the function which returns a vector of size n x 1
 #   h    eps for difference quotient
 #   ...  additional parameters to f
 #
 # Values:
 #   J      n x p Jacobian matrix consisting of
 #             part d f / part d x
 #

 p=length(a)
 n=length(f(a, ...))

 epsmat=diag(p)*h
 J=matrix(0, n, p)
 for(j in 1:p)
  J[, j]=( f(a+epsmat[, j], ...) - f(a-epsmat[, j], ...) )/(2*h)

 return( J )

} # end of JacobianMat
```

Numerical gradient and Hessian

```
# objective function
r <- function( x, additional_param ){
 # calculation of the function value at x
 return( function_value )
}

# gradient vector of r at x
grad <- function( x, additional_param ){
 g <- JacobianMat( x, r, additional_param as ... )
 return( g )
}
# Hessian matrix of r at x
Hess <- function( x, additional_param ){
 H <- JacobianMat( x, grad, additional_param as ... )
 return( H )
}

# gradient at x=a
a <- the value of x
gradient <- g( a, additional_param )
Hessian <- Hess( a, additional_param )
```


- スカラー値を取る多変数関数の最適化

関数の最適化

一般に，ある関数 $r(\boldsymbol{\xi})$ の極値（極小値または極大値のことであるが，本節では極小値とする）を与えるパラメタ $\boldsymbol{\xi}$ の値を数値的に求めることを関数の最適化と呼ぶが，その方法は，何らかの方法で求めた $\boldsymbol{\xi}$ の初期値から，$r(\boldsymbol{\xi})$ が減少する方向 $\boldsymbol{d}$ へある分量 α だけ修正する，

$$\boldsymbol{\xi}_{(\ell+1)} = \boldsymbol{\xi}_{(\ell)} + \alpha \boldsymbol{d}_{(\ell)} \tag{54}$$

という形の繰り返し計算でパラメタの値を更新していくという形をとる．なお、この関数 $r(\boldsymbol{\xi})$ は目的関数 objective function と呼ばれる。

数値的最適化の方法には様々なものがあるが，それらの主な違いは，更新の方向 $\boldsymbol{d}_{(\ell)}$ をどの様にして定めるかによる．一般的には、以下に示すように、$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$ を $r(\boldsymbol{\xi})$ のパラメタの各要素に関する一次偏微分係数ベクトル (gradient) を $\boldsymbol{\xi} = \boldsymbol{\xi}_{(\ell)}$ で評価したものとして、

$$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})' \boldsymbol{d}_{(\ell)} < 0 \tag{55}$$

であれば

$$r(\boldsymbol{\xi}_{(\ell+1)}) \leq r(\boldsymbol{\xi}_{(\ell)}) \tag{56}$$

を満たすステップサイズ α が存在する。

なぜなら、いま、$r(\boldsymbol{\xi}_{(\ell+1)})$ を $\boldsymbol{\xi}_{(\ell)}$ の周りで１次のテイラー展開をすると

$$r(\boldsymbol{\xi}_{(\ell+1)}) \approx r(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'(\boldsymbol{\xi}_{(\ell+1)} - \boldsymbol{\xi}_{(\ell)})$$

であるが、これに $\boldsymbol{\xi}_{(\ell)}$ の更新の方向を示すベクトル $\boldsymbol{d}$ を用いて

$$\boldsymbol{\xi}_{(\ell+1)} = \boldsymbol{\xi}_{(\ell)} + \boldsymbol{d}$$

を代入すると

$$r(\boldsymbol{\xi}_{(\ell)} + \boldsymbol{\delta}) \approx r(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})' \boldsymbol{d}$$

を得る。したがって、$r(\boldsymbol{\xi}_{(\ell)} + \boldsymbol{d})$ を $r(\boldsymbol{\xi}_{(\ell)})$ より減少させるためには

$$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})' \boldsymbol{d} < 0$$

である必要があるが、ベクトル間の内積はベクトル間のコサインに比例するため、それが負であるということは $\boldsymbol{d}$ と $-\boldsymbol{\xi}_{(\ell)}$ の間の角度が 90 度以内であることを示している。すなわち、gradient が示す方向と反対の方向から 90 度以内の方向へ $\boldsymbol{\xi}_{(\ell)}$ を更新すれば、$r(\boldsymbol{\xi}_{(\ell+1)})$ の値は減少すると期待される。

最急降下法 steepest descent

最も単純なものは最急降下法と呼ばれる方法であり，修正の方向を $r(\boldsymbol{\xi})$ の gradient $\boldsymbol{g}$ (gradient) の逆方向

$$\boldsymbol{d}_{(\ell)} = -\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$$

に取るものである．ただし、

$$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)}) = \left.\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}}\right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_{(\ell)}} \tag{57}$$

は $r(\boldsymbol{\xi})$ の gradient を $\boldsymbol{\xi} = \boldsymbol{\xi}_{(\ell)}$ で評価したものである。すなわち最急降下法の更新式は

$$\boldsymbol{\xi}_{(\ell+1)} = \boldsymbol{\xi}_{(\ell)} - \alpha \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)}) \tag{58}$$

である。

Newton-Raphson 法

$r(\boldsymbol{\xi})$ を $\boldsymbol{\xi}_{(\ell)}$ のまわりで２次の項まで Taylor 展開すると

$$r(\boldsymbol{\xi}) \approx r(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)}) + \tfrac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})'\boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})$$

$$\boldsymbol{H}(\boldsymbol{\xi}_{(\ell)}) = \left[\left.\frac{\partial^2 r(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}\ \partial\boldsymbol{\xi}'}\right|_{\boldsymbol{\xi}=\boldsymbol{\xi}_{(\ell)}}\right]$$

$$r(\boldsymbol{\xi}) \approx r(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)}) + \tfrac{1}{2}(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})'\boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})$$

したがって，極値の条件は

$$\frac{\partial\, r(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} \approx \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)}) = 0$$

となり，これより

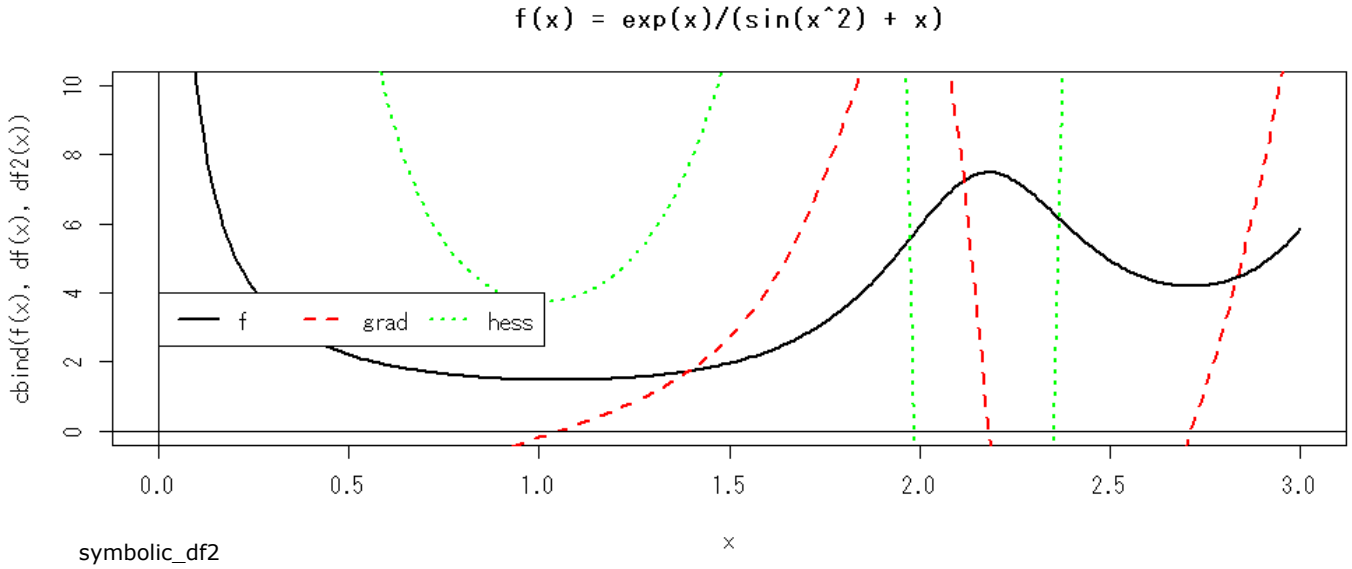
$$\boldsymbol{\xi} = \boldsymbol{\xi}_{(\ell)} - \boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})^{-1}\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$$

となる．したがって，Eq.(54) において

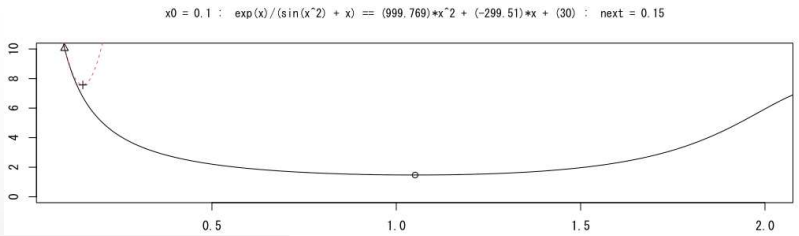
$$\boldsymbol{d}_{(\ell)} = -\boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})^{-1}\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$$

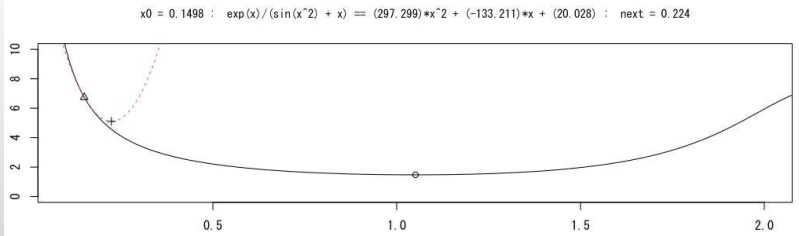
$$\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{d}_{(\ell)} = -\,\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'\,\boldsymbol{H}(\boldsymbol{\xi}_{(\ell)})^{-1}\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)}) \;<0 \quad \text{if } \boldsymbol{H} \text{ is p.d.}$$

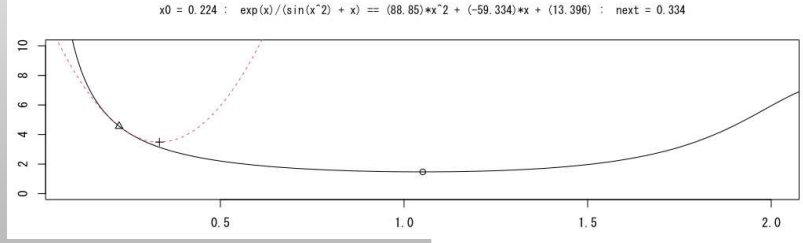
- NR 一変数の場合の例

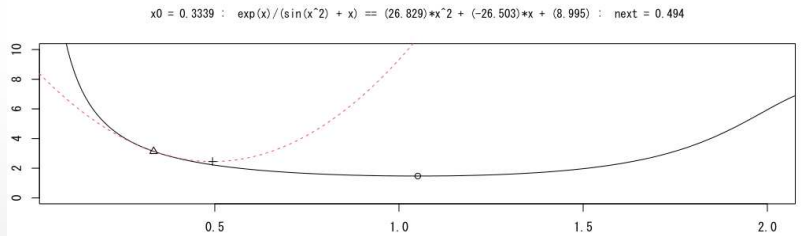


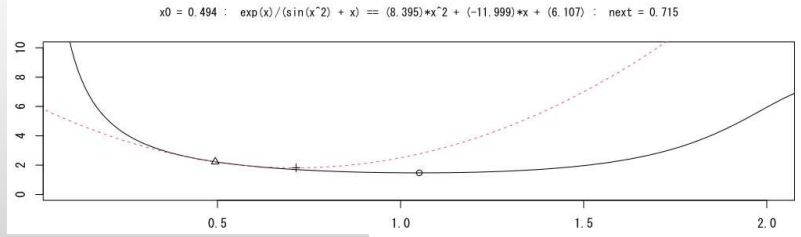

```
symbolic_df2
function (x)
{
    .expr1 <- exp(x);     .expr2 <- x^2;     .expr3 <- sin(.expr2);
    .expr4 <- .expr3 + x;     .expr5 <- .expr1/.expr4
    .expr6 <- cos(.expr2);     .expr7 <- 2 * x;     .expr9 <- .expr6 * .expr7 + 1
    .expr10 <- .expr1 * .expr9;     .expr11 <- .expr4^2;     .expr13 <- .expr5 - .expr10/.expr11
    .value <- .expr5
    .grad <- array(0, c(length(.value), 1L), list(NULL, c("x")))
    .hessian <- array(0, c(length(.value), 1L, 1L), list(NULL, c("x"), c("x")))
    .grad[, "x"] <- .expr13
    .hessian[, "x", "x"] <- .expr13 - ((.expr10 +
        .expr1 * (.expr6 * 2 - .expr3 * .expr7 * .expr7))/.expr11 -
        .expr10 * (2 * (.expr9 * .expr4))/.expr11^2)
    attr(.value, "gradient") <- .grad
    attr(.value, "hessian") <- .hessian
    .value
}
```

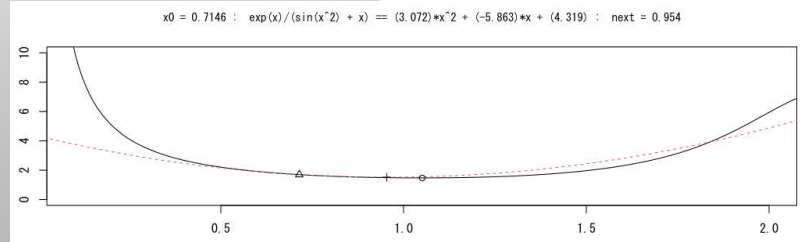



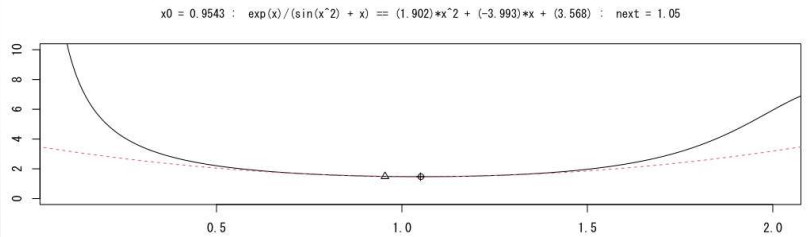


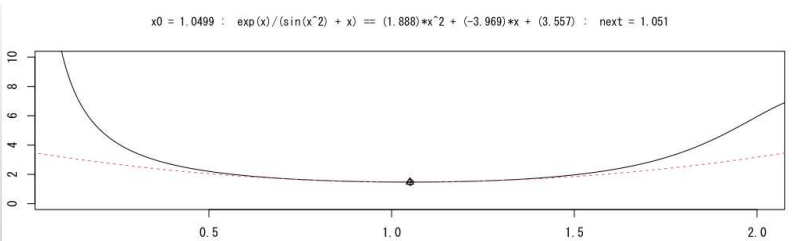



x0 = 0.3339 : exp(x)/(sin(x^2) + x) == (26.829)*x^2 + (-26.503)*x + (8.995) : next = 0.494
0 2 4 6 8 10
0.5 1.0 1.5 2.0


x0 = 0.494 : exp(x)/(sin(x^2) + x) == (8.395)*x^2 + (-11.999)*x + (6.107) : next = 0.715
0 2 4 6 8 10
0.5 1.0 1.5 2.0


x0 = 0.7146 : exp(x)/(sin(x^2) + x) == (3.072)*x^2 + (-5.863)*x + (4.319) : next = 0.954
0 2 4 6 8 10
0.5 1.0 1.5 2.0


x0 = 0.9543 : exp(x)/(sin(x^2) + x) == (1.902)*x^2 + (-3.993)*x + (3.568) : next = 1.05
0 2 4 6 8 10
0.5 1.0 1.5 2.0


x0 = 1.0499 : exp(x)/(sin(x^2) + x) == (1.888)*x^2 + (-3.969)*x + (3.557) : next = 1.051
0 2 4 6 8 10
0.5 1.0 1.5 2.0

- ロジスティク回帰

ロジスティク回帰

$$\Pr\left(Y_i = 1 | x_i\right) = \frac{1}{1 + \exp\left(-(\alpha x_i + \beta)\right)} = P(x_i|\alpha, \beta) = \pi_i(\boldsymbol{\xi}) \tag{66}$$

モデルのパラメタは $\boldsymbol{\xi} = \{\alpha, \beta\}$
(f_i, r_i) を試行数と度数として、

$$\boldsymbol{\mathcal{E}}(R_i|x_i) = f_i \times P(x_i|\alpha, \beta) \ = \ f_i \pi_i(\boldsymbol{\xi}) \tag{67}$$

$$var\left(R_i|x_i\right) = f_i \pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi})) \tag{68}$$

である。

尤度関数

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} \pi_i(\boldsymbol{\xi})^{y_i}(1-\pi_i(\boldsymbol{\xi}))^{1-y_i} \tag{69}$$

$$L(\boldsymbol{\xi}) = \prod_{i=1}^{n} \pi_i(\boldsymbol{\xi})^{r_i}(1-\pi_i(\boldsymbol{\xi}))^{f_i-r_i} \tag{70}$$

対数尤度関数

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} y_i \log \pi_i(\boldsymbol{\xi}) + (1-y_i)\log(1-\pi_i(\boldsymbol{\xi})) \tag{71}$$

$$lnL(\boldsymbol{\xi}) = \sum_{i=1}^{n} r_i \log \pi_i(\boldsymbol{\xi}) + (f_i-r_i)\log(1-\pi_i(\boldsymbol{\xi})) \tag{72}$$

これを最大にするように $\boldsymbol{\xi}$ を求める。

Gradient と Hessian

gradient は

$$\frac{\partial\, lnL}{\partial\, \alpha} = \sum_{i=1}^{n}(r_i - f_i\pi_i(\boldsymbol{\xi}))x_i \tag{73}$$

$$\frac{\partial\, lnL}{\partial\, \beta} = \sum_{i=1}^{n}(r_i - f_i\pi_i(\boldsymbol{\xi})) \tag{74}$$

Hessian は

$$\frac{\partial^2 lnL(\boldsymbol{\xi})}{\partial \alpha^2} = -\sum_{i=1}^{n} f_i\, x_i^2\, \pi_i(\boldsymbol{\xi})(1-\pi_i(\boldsymbol{\xi}) \tag{75}$$

$$\frac{\partial^2 lnL(\boldsymbol{\xi})}{\partial \beta\ \partial \alpha} = \frac{\partial^2 lnL(\boldsymbol{\xi})}{\partial \alpha\ \partial \beta} = -\sum_{i=1}^{n} f_i\, x_i\, \pi_i(\boldsymbol{\xi})(1-\pi_i(\boldsymbol{\xi}) \tag{76}$$

$$\frac{\partial^2 lnL(\boldsymbol{\xi})}{\partial \beta^2} = -\sum_{i=1}^{n} f_i\, \pi_i(\boldsymbol{\xi})(1-\pi_i(\boldsymbol{\xi}) \tag{77}$$

導出

導出

$$\frac{\partial\, lnL(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} = \sum_{i=1}^{n} \left(r_i \frac{1}{\pi_i(\boldsymbol{\xi})} \frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} - (f_i - r_i) \frac{1}{1 - \pi_i(\boldsymbol{\xi})} \frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \boldsymbol{x_i}} \right)$$

$$= \sum_{i=1}^{n} \left(\frac{r_i(1 - \pi_i(\boldsymbol{\xi})) - (f_i - r_i)\pi_i(\boldsymbol{\xi})}{\pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi}))} \frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} \right) \tag{78}$$

$$= \sum_{i=1}^{n} \left(\frac{r_i - f_i \pi_i(\boldsymbol{\xi})}{\pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi}))} \frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \boldsymbol{\xi}} \right) \tag{79}$$

$$\frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, A} = \frac{-1}{(1 + \exp(-(\alpha x_i + \beta)))^2} \times \exp(-(\alpha x_i + \beta))$$

$$\times \frac{\partial\ -(\alpha x_i + \beta)}{\partial\, A} \tag{80}$$

$$= \pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi})) \times \frac{\partial\, \alpha x_i + \beta}{\partial\, A} \tag{81}$$

$$\frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \alpha} = \pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi}))x_i \tag{82}$$

$$\frac{\partial\, \pi_i(\boldsymbol{\xi})}{\partial\, \beta} = \pi_i(\boldsymbol{\xi})(1 - \pi_i(\boldsymbol{\xi})) \tag{83}$$

R による計算

```
data <- data.frame( x, r, f)

# probability (logistic function with slope and intercept, no 1.7)
P <- function( x, param ){
 a=param[1]; b=param[2]
 z=a*x+b
 P=1/(1+exp(-z))
 return( P )
} # end of P

# minus log likelihood
mllh <- function( param, data ){
 x=data$x; f=data$f; r=data$r
 P=P(x,param)
 llh=sum( r*log(P)+(f-r)*log(1-P) )
 return( -llh )
} # end of mllh
```


解析的な gradient と Hessian

```
# analytic first derivative of mllh
dmllha <- function( param, data ){
 x=data$x; f=data$f; r=data$r
 P=P(x,param)
 temp=(r-f*P)
 dab=c( sum(temp*x),sum(temp) )
 cat(".da.")
 return( -dab )
} # end of dmllha
```

← これが gradient, ***g***

これが Hessian, ***H*** →

```
# analytic second derivative matrix
# of mllh
d2mllhaa <- function( param, data ){
 x=data$x; f=data$f; r=data$r
 P=P(x,param)
 PQ=P*(1-P)
 cat(".d2aa.")
 H=matrix(0,2,2)
 H[1,1]=-sum(f*x*x*PQ)
 H[1,2]=-sum(f*x*PQ)
 H[2,1]=H[1,2]
 H[2,2]=-sum(f*PQ)
 return( -H )
} # end of d2mllhaa
```

数値微分による gradient と Hessian

```
# probability (logistic function with slope and intercept, no 1.7)
P <- function( X, param ){
 a=param[1]; b=param[2]
 z=a*X+b
 P=1/(1+exp(-z))
 return( P )
} # end of P

# minus log likelihood
mllh <- function( param, data ){
 x=data$x; f=data$f; r=data$r
 P=P(x,param)
 llh=sum( r*log(P)+(f-r)*log(1-P) )
 return( -llh )
} # end of mllh

# numeric first derivative of mllh
dmllhn <- function( param, data ){
 # mllh comes from the parent environment
 res=c( JacobianMat( param, mllh, data=data ) )
 cat(".dn.")
 return( res )
} # end of dmllhn

# second derivative matrix using numeric first derivative of mllh
d2mllhnn <- function( param, data ){
 # dmllhan and mllh come from the parent environment
 res=JacobianMat( param, dmllhn, data=data )
 cat(".d2nn.")
 return( res )
} # end of d2mllhnn
```


Newton-Raphson function

```
for( llll in 1:maxiter ){

  # gradient and hessian
  g=gradient(xp, ...)
  H=hessian(xp, ...)

  # direction
  d=solve(H)%*%g

  # update with step-size halving
  alpha=1
  for( lll in 1:20 ){
   x=xp-alpha*d
   fval=f(x, ...)
   if( fval < fvalp ) break
   alpha=alpha/2
  }

  # new function value
  fval=f(x, ...)

  # check convergence
  if( abs(fvalp-fval) <= eps ) break

  # update previous values
  xp=x
  fvalp=fval

 } # end of llll loop
```

方法の比較

```
# nlminb default
res0=nlminb( param_init, mllh, data=data, control=list(trace=1) )
  0:     49.294500:  3.00000 -1.00000
  1:     40.270717:  2.43365 -0.175834
  2:     36.331303:  1.60680 0.386591
  3:     36.113120:  1.51818 0.234972
  4:     36.111554:  1.41473 0.224860
  5:     36.098074:  1.46310 0.205857
  6:     36.097812:  1.46518 0.214235
  7:     36.097797:  1.46435 0.212764
  8:     36.097797:  1.46434 0.212761

# with analytic gradient
res1=nlminb( param_init, mllh, data=data, control=list(trace=1)
        , gradient=dmllha )
.da.  0:     49.294500:  3.00000 -1.00000
.da.  1:     40.270717:  2.43365 -0.175834
.da.  2:     36.331299:  1.60680 0.386590
.da.  3:     36.113121:  1.51818 0.234971
.da.  4:     36.111556:  1.41473 0.224862
.da.  5:     36.098074:  1.46310 0.205858
.da.  6:     36.097812:  1.46518 0.214236
.da.  7:     36.097797:  1.46435 0.212763
.da.  8:     36.097797:  1.46434 0.212762


# with analytic gradient and hessian
res2=nlminb( param_init, mllh, data=data, control=list(trace=1)
        , gradient=dmllha, hessian=d2mllhaa )
.da..d2aa.  0:     49.294500:  3.00000 -1.00000
.da..d2aa.  1:     40.321942:  2.33143 -0.312787
.da..d2aa.  2:     37.035600:  1.05938 0.161357
.da..d2aa.  3:     36.153368:  1.37589 0.152402
.da..d2aa.  4:     36.099561:  1.45078 0.199298
.da..d2aa.  5:     36.097856:  1.46157 0.210525
.da..d2aa.  6:     36.097799:  1.46388 0.212286
.da..d2aa.  7:     36.097797:  1.46424 0.212685
.da..d2aa.  8:     36.097797:  1.46432 0.212745
.da..d2aa.  9:     36.097797:  1.46433 0.212759
```

```
# with numeric gradient and hessian
res3=nlminb( param_init, mllh, data=data, control=list(trace=1)
           , gradient=dmllhn, hessian=d2mllhnn )
..dn..d2nn.  0:     49.294500:  3.00000 -1.00000
..dn..d2nn.  1:     40.647963:  2.36131 -0.344719
..dn..d2nn.  2:     36.225279:  1.60935 0.150543
..dn..d2nn.  3:     36.098968:  1.44977 0.216337
..dn..d2nn.  4:     36.097797:  1.46420 0.212780
..dn..d2nn.  5:     36.097797:  1.46434 0.212762
             6:     36.097797:  1.46434 0.212762
```

	nhalv	x1	x2	g1	g2	d1	d2	f(x)
0	0	3.0000	-1.0000	6.3045	-9.1746	NA	NA	49.2945
1	1	1.5628	-0.7273	6.3045	-9.1746	2.8743	-0.5455	41.2021
2	0	1.2694	0.1514	-0.2562	-10.6094	0.2934	-0.8787	36.3078
3	0	1.4405	0.1828	-2.0706	-0.5555	-0.1710	-0.0314	36.1051
4	0	1.4634	0.2086	-0.2041	-0.3286	-0.0229	-0.0258	36.0979
5	0	1.4643	0.2126	-0.0045	-0.0474	-0.0009	-0.0040	36.0978
6	0	1.4643	0.2128	0.0000	-0.0020	0.0000	-0.0002	36.0978
7	0	1.4643	0.2128	0.0000	-0.0001	0.0000	0.0000	36.0978
8	0	1.4643	0.2128	0.0000	-0.0001	0.0000	0.0000	36.0978