第9回　母数の推定
最尤解（一対比較法）

最小２乗解

尤度関数 Likelihood Function

$Y_i, i = 1, 2, \ldots, N$ を独立な観測値とする。$f(y_i|\xi)$ をデータの確率関数（密度関数）とする。

データの同時確率関数を母数 ξ の関数と見たものが尤度関数である。

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

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、ξ の確率を与えるものではない。（ξ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、

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

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。

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

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。

Bayes Modal

パラメタ $\boldsymbol{\xi}$ の事前分布が $h(\boldsymbol{\xi})$ という密度関数の形で与えられている場合、
$\boldsymbol{\xi}$ の事後分布は尤度関数に事前分布の密度関数を掛け合わせた形に比例する。

$$h(\boldsymbol{\xi}|\boldsymbol{y}) \propto L(\boldsymbol{\xi}) \times h(\boldsymbol{\xi}) = \prod_{i=1}^{N} f(y_i|\boldsymbol{\xi}) \times h(\boldsymbol{\xi})$$

（比例乗数部分には $\boldsymbol{\xi}$ は含まれないことに注意。）

事後分布はそのモードを中心に広がっていると考えられるが
Newton-Raphson 法を用いることによりモードを求めることが出来る。
また、事後分布の平均は EAP 推定値と呼ばれ、
その精度は事後平均を求めることにより評価できるが、
$\boldsymbol{\xi}$ が多変量数の場合は難しい。 $\rightarrow$ MCMC

- 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。

Newton-Raphson 法

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

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

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

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

- 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

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

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

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

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

R による計算

param が ξ 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) )
```

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

- 最尤解の例　　一対比較法

一対比較法：比較判断の法則

- 計量心理的なモデルとしてとても大切
- Thurstone のモデル 1927

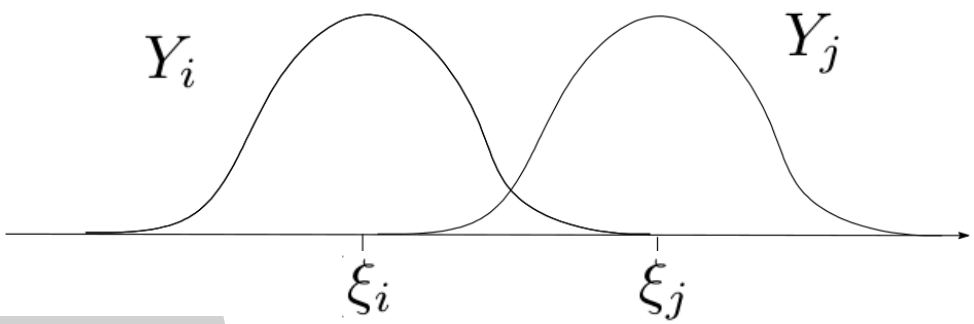
  Thurstone, L.L. (1927).
  A law of comparative judgement. Psychological Review, 34, 278-286.

  ◆様々な統計・数理モデルの基礎となるもの。

- 刺激は１次元の心理学的連続体の上に位置づけられているが、その値は揺れている。

一対比較法

$$o_{ij} = \begin{cases} 1 & S_i \text{ が } S_j \text{ より好まれる} \\ 0 & S_j \text{ が } S_i \text{ より好まれる} \end{cases}$$



心理学的連続体
psychological
continuum

$$Y_i = \xi_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma_i^2)$$

$$Y_i \geq Y_j \rightarrow S_i \succ S_j$$

$$\Pr(S_i \succ S_j) = \Pr(Y_i \geq Y_i)$$

$$\begin{aligned} \Pr(S_i \succ S_j) &= \Pr(Y_i \geq Y_i) \\ &= \Pr(\xi_i - \xi_j + \varepsilon_i - \varepsilon_j \geq 0) \\ &= \Phi\left(\frac{\xi_i - \xi_j}{\sqrt{\sigma_{ij}^2}}\right) \end{aligned}$$

$$Y_i = \xi_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma_i^2)$$

$$\varepsilon_i - \varepsilon_j \sim N(0, \sigma_{ij}^2), \quad \sigma_{ij}^2 = \sigma_i^2 + \sigma_j^2 - 2cov(\varepsilon_i, \varepsilon_j)$$

Case V

$$\sum_{i=1}^{p} \xi_i = 0 \qquad \sigma_i^2 = \sigma_j^2 = \ldots = \sigma^2 \qquad 2\sigma^2 = 1$$

$$\Pr(S_i \succ S_j) \;=\; \Pr(Y_i \geq Y_j) \;=\; \Phi\left(\frac{\xi_i - \xi_j}{\sqrt{2}}\right)$$

データ

$$o_{ij} = \begin{cases} 1 & S_i \text{ が } S_j \text{ より好まれる} \\ 0 & S_j \text{ が } S_i \text{ より好まれる} \end{cases}$$

これを、各刺激対に関して f_{ij} 回繰り返したもの

$$o_{ij}^{(a)},\ a = 1, 2, \ldots, f_{ij}$$

$$r_{ij} = \sum_{a=1}^{f_{ij}} o_{ij}^{(a)} \qquad p_{ij} \;=\; \frac{r_{ij}}{f_{ij}}$$

S_i と S_j との f_{ij} 回の比較において S_i が S_j よりも好まれた度数

Thurstone の最小２乗解

$$p_{ij} \approx \pi_{ij} = \Phi\left(\frac{\xi_i - \xi_j}{\sqrt{2\sigma^2}}\right)$$

$$\Phi^{-1}(p_{ij}) \approx \Phi^{-1}(\pi_{ij}) = \frac{\xi_i - \xi_j}{\sqrt{2\sigma^2}}$$

$$z_{ij} \approx \widehat{z_{ij}} = \frac{\xi_i - \xi_j}{\sqrt{2\sigma^2}}$$

$$\boldsymbol{\xi}_i = \frac{1}{p}\sum_{j=1}^{p} z_{ij}$$

$$\sum_{i=1}^{p} \xi_i = 0, \quad \text{and} \quad 2\sigma^2 = 1$$

$$d_{k1} = i, \quad \text{and} \quad d_{k2} = j,$$

$$G_{k,d_{k1}} = 1 \quad \text{and} \quad G_{k,d_{k2}} = -1$$

$$z_{ij} \approx \widehat{z_{ij}} = \frac{\xi_i - \xi_j}{\sqrt{2\sigma^2}}$$

$$\boldsymbol{z} \approx \widehat{\boldsymbol{z}} = \frac{1}{\sqrt{2\sigma^2}}\,\boldsymbol{G}\,\boldsymbol{\xi}$$

$$\widehat{\boldsymbol{\xi}} = (\boldsymbol{G}'\boldsymbol{G})^{-}\boldsymbol{G}'\boldsymbol{z}$$

$\boldsymbol{G}$				$\boldsymbol{D}$
1	2	3	4	
-1	1	0	0	2-1
-1	0	1	0	3-1
-1	0	0	1	4-1
0	-1	1	0	3-2
0	-1	0	1	4-2
0	0	-1	1	4-3

最尤解

データのモデル

$$r_{ij} \sim Bin(f_{ij}, \pi_{ij}) \qquad \pi_{ij} = \Phi\left(\frac{\xi_i - \xi_j}{\sqrt{2\sigma^2}}\right)$$

尤度関数

$$\begin{aligned} L &= \prod_{i>j} \prod_{a=1}^{f_{ij}} {\pi_{ij}}^{{o_{ij}}^{(a)}} (1-\pi_{ij})^{(1-{o_{ij}}^{(a)})} \\ &= \prod_{i>j} {\pi_{ij}}^{r_{ij}} (1-\pi_{ij})^{(f_{ij}-r_{ij})} \end{aligned}$$

対数尤度

$$\ln L = \sum_{i>j} r_{ij} \ln \pi_{ij} + (f_{ij} - r_{ij}) \ln (1-\pi_{ij})$$

$$\sum_{i=1}^{p} \xi_i = 0, \qquad \text{and} \qquad 2\sigma^2 = 1$$

R pgm

```
#
# Here, (f, r, IJ) is given as data.
#
# design matrix from index

n=max(IJ)-min(IJ)+1

G=matrix(0,npair,n)

for( k in 1:npair ){
 G[k,IJ[k,1]]=1; G[k,IJ[k,2]]=-1
}
```

IJ ← ***D***

R pgm

```
# given x, calculate minus llh
mllh <- function( x, f=0, r=0, G=0 ){
 p=pnorm(G%*%x/sqrt(2))
 llh=sum(r*log(p)+(f-r)*log(1-p))
 return( -llh )
} # end of mllh


# resNR=NR( xLS, mllh, maxiter=2, print=2, f=f, r=r, G=G )

res=nlminb( xLS, objective=mllh, control=list(trace=1), f=f, r=r, G=G )
xML=res$par
# centring
xML=xML-mean(xML)
# final gradient
grad=JacobianMat( xML, mllh, f=f, r=r, G=G )
Print(x0,xLS,xML, grad)
```


- 非線形モデルの最小２乗解

データの期待値のモデル化

$$\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i|\boldsymbol{\mathcal{B}}) + \boldsymbol{\varepsilon}_i, i = 1, 2, \ldots, N$$

$$\boldsymbol{y_i} = \hat{\boldsymbol{y}}_i + \boldsymbol{\varepsilon_i} \quad \text{and} \quad \boldsymbol{\mathcal{E}}(\boldsymbol{y_i} \mid \boldsymbol{x_i}) = \hat{\boldsymbol{y}}_i$$
$$\hat{\boldsymbol{y}}_i = \boldsymbol{\mathcal{E}}(\boldsymbol{y_i}|\boldsymbol{x_i}) = \boldsymbol{f}(\boldsymbol{x_i}|\boldsymbol{\mathcal{B}})$$

$$\boldsymbol{\mathcal{E}}(\boldsymbol{\varepsilon_i}|\boldsymbol{x_i}) = \boldsymbol{\mathcal{E}}(\boldsymbol{\varepsilon_i}) = \boldsymbol{0}$$

例
因子分析
多次元尺度法
ニューラルネットワーク

残差の２乗和 == データへのモデルの当てはまりの悪さ

$$\mathrm{RSS} = \sum_{i=1}^{N}(y_i - \widehat{y_i})^2 = (\boldsymbol{y} - \widehat{\boldsymbol{y}})'(\boldsymbol{y} - \widehat{\boldsymbol{y}}) = \|\boldsymbol{y} - \widehat{\boldsymbol{y}}\|^2$$

最小２乗法

$\mathrm{RSS}(\boldsymbol{\beta})$ を最小とする様なパラメタ $\boldsymbol{\beta}$ の値を求める方法

$$\frac{\partial\,\mathrm{RSS}}{\partial\,\boldsymbol{\beta}} = \boldsymbol{0}$$

Gauss-Newton 法

$$\boldsymbol{y}_i = \boldsymbol{f}(\boldsymbol{x}_i|\boldsymbol{\mathcal{B}}) + \boldsymbol{\varepsilon}_i, i = 1, 2, \ldots, N$$

まとめて *N x 1* のベクトルとして書くと

$$\boldsymbol{y} = \boldsymbol{\theta}(\boldsymbol{\xi}) + \boldsymbol{e}$$

$\widehat{\boldsymbol{y}}$ を $\boldsymbol{\theta}(\boldsymbol{\xi})$ としている。

$\mathrm{RSS} = \|\boldsymbol{y} - \boldsymbol{\theta}(\boldsymbol{\xi})\|^2 = (\boldsymbol{y} - \boldsymbol{\theta}(\boldsymbol{\xi}))'(\boldsymbol{y} - \boldsymbol{\theta}(\boldsymbol{\xi}))$ を最小にする $\boldsymbol{\xi}$ を求める

$\boldsymbol{\theta}(\boldsymbol{\xi})$を $\boldsymbol{\xi}_{(\ell)}$ のまわりで 1 次の項まで Taylor 展開したもの

$$\boldsymbol{\theta}(\boldsymbol{\xi}) \approx \boldsymbol{\theta}(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})$$

$$\boldsymbol{F}(\boldsymbol{\xi}) = \frac{\partial\,\boldsymbol{\theta}(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}} = \left[\frac{\partial\,\theta_i(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}_k}\right]$$ ヤコビアン行列

1 次近似を RSS に代入すれば,

$$\begin{aligned}\mathrm{RSS}(\boldsymbol{\xi}) &\approx \left\|\boldsymbol{v} - \left(\boldsymbol{\theta}(\boldsymbol{\xi}_{(\ell)}) + \boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})\right)\right\|^2 \\ &= \left\|\boldsymbol{v} - \boldsymbol{\theta}(\boldsymbol{\xi}_{(\ell)}) - \boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)})\right\|^2\end{aligned}$$

$$\frac{\partial\,\mathrm{RSS}(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}} \approx -2\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'(\boldsymbol{v} - \boldsymbol{\theta}(\boldsymbol{\xi}_{(\ell)})) + 2\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})(\boldsymbol{\xi} - \boldsymbol{\xi}_{(\ell)}) \;=\; 0$$

$$\boldsymbol{\xi} = \boldsymbol{\xi}_{(\ell)} + \left(\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})\right)^{-1}\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'(\boldsymbol{v} - \boldsymbol{\theta}(\boldsymbol{\xi}_{(\ell)}))$$

$$\boldsymbol{g} = \frac{\partial\,\mathrm{RSS}(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}} = -2\boldsymbol{F}(\boldsymbol{\xi})'(\boldsymbol{v} - \boldsymbol{\theta}(\boldsymbol{\xi}))$$

$$\boldsymbol{d}_{(\ell)} = -\tfrac{1}{2}\left(\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})\right)^{-1}\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)}) \qquad \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{d}_{(\ell)} \le 0$$

基準変数が多変量の場合は、

$$\mathrm{RSS} = \|\boldsymbol{Y} - \boldsymbol{\Theta}(\boldsymbol{\xi})\|^2 \;=\; \sum_{j=1}^{p}\left\|\boldsymbol{y}_j - \boldsymbol{\theta}_{(j)}(\boldsymbol{\xi})\right\|^2$$

と書けるため、変数ごとのヤコビアン行列を

$$\boldsymbol{F}_j(\boldsymbol{\xi}) = \frac{\partial\,\boldsymbol{\theta}_{(j)}(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}} = \left[\frac{\partial\,\theta_{ij}(\xi)}{\partial\,\boldsymbol{\xi}_k}\right], \quad i = 1, 2, \ldots, N,\ k = 1, 2, \ldots, Q$$

と定義すれば、グレイディエント並びに更新の方向は

$$\boldsymbol{g} = \frac{\partial\,\mathrm{RSS}(\boldsymbol{\xi})}{\partial\,\boldsymbol{\xi}} \;=\; -2\sum_{j=1}^{p}\boldsymbol{F}_{(j)}(\boldsymbol{\xi})'(\boldsymbol{y}_j - \boldsymbol{\theta}_{(j)}(\boldsymbol{\xi}))$$

$$\boldsymbol{d}_{(\ell)} = -\tfrac{1}{2}\left(\textstyle\sum_{j=1}^{p}\boldsymbol{F}_{(j)}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}_{(j)}(\boldsymbol{\xi}_{(\ell)})\right)^{-1}\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$$

で与えられる。

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

$$\boldsymbol{d} = -\boldsymbol{g}$$

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

$$\boldsymbol{d}_{(\ell)} = -\tfrac{1}{2}\left(\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})\right)^{-1} \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$$

Gauss-Newton 法

ξ_1 に初期値を設定する。
for $\ell = 1$ to maxiter until($\mathrm{RSS}(\boldsymbol{\xi}_{(\ell+1)})$ - $\mathrm{RSS}(\boldsymbol{\xi}_{(\ell)})$ が十分小さい)
  $\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})$, $\boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$ を計算する。
  $\boldsymbol{d}_{(\ell)} = -\frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})'\boldsymbol{F}(\boldsymbol{\xi}_{(\ell)})\right)^{-1} \boldsymbol{g}(\boldsymbol{\xi}_{(\ell)})$ を計算する。
  α=1
  for iters = 1 to maxhalf
    $\boldsymbol{\xi}_{(\ell+1)} = \boldsymbol{\xi}_{(\ell)} + \alpha\, \boldsymbol{d}_{(\ell)}$
    if $\mathrm{RSS}(\boldsymbol{\xi}_{(\ell+1)}) \leq \mathrm{RSS}(\boldsymbol{\xi}_{(\ell)})$ then break
    α=$\alpha/2$
  end
end

R の nlfb 関数　in　nlsr　package

- データとモデルのずれを与えると RSS の最小値を与える母数 ξ を Gauss-Newton 法で探してくれる。
- 引数は

  パラメタの初期値ベクトル、

  パラメタを与えたときに、データとモデルのずれを与える関数 resfn

  必要に応じて、

  目的関数の補助的パラメタ、Jacobian を与える関数 jacfn

  trace=TRUE, lower=, upper=

  制御用のパラメタリスト　（ femax 等）

```
resnlfb <- nlfb( param, resfn )
resnlfb <- nlfb( param, resfn, trace=1 )
resnlfb <- nlfb( param, resfn, jacfn=Jac )
```

- 結果はリスト

  $coefficients, $ssquares, $feval, $resid, and many others

- その他の関数： nls

ロジスティク回帰的なモデル

説明変数 $\boldsymbol{x}_i$ を与えたときの正反応の割合 p_i をロジスティク曲線（曲面）で予想する。

$$p_i \approx \theta(\boldsymbol{x}_i|\beta) + \boldsymbol{e}_i$$

$$\theta(\boldsymbol{x}_i|\beta) = \frac{1}{1+\exp(-\boldsymbol{x}_i'\beta)}$$

$p_i,\ i = 1, 2, \ldots, N$ がデータ（観測された比率）で $\boldsymbol{y}$

回帰係数 $\boldsymbol{\beta}$ がパラメタ $\boldsymbol{\xi}$ である。

R pgm

```
# probability (logistic function with matrix
X, no 1.7)
P <- function( X, beta ){
      if( is.matrix(X) | is.data.frame(X) )
z=X%*%beta
      else z=X*beta
      P=1/(1+exp(-z))
      return( P )
} # end of P
```

```
# 残差ベクトル　データ – モデル
ymyhat <- function( beta, X=NULL, y=NULL ){
 # returns residuals as y-yhat
 resid=y-P(X,beta)
 return( resid )
} # end of ymyhat

# 残差ベクトル nlfb 用は モデル – データ
yhatmy <- function( beta, X=NULL, y=NULL ){
 # returns residuals as yhat-y
 return( -ymyhat( beta, X=X, y=y ) )
} # end of yhatmy
```

```
resnlsr=nlfb( beta0, yhatmy, X=X, y=y
           , control=list(femax=500) )
betaLS=resnlsr$coeff
```


その他の例