Lasso 回帰，Ridge 回帰，Elastic Net でのパラメータ λ 列検討

2017. 05. 17 株式会社 応用数理研究所 佐々木 俊久

１．Lasso 回帰，Ridge 回帰，Elastic Net

３つのモデルは，どれも線形回帰で，パラメータ β_0, β は，以下の式によって推定されます。（R の library glmnet に関する文献[1]から引用）

データ数 N　　データ (x_i, y_i)

$Y \in R$　：目的変数（１次元）

$X \in R^p$　：説明変数（p 次元）

x_i を標準化(standarlize)

$$\sum_{i=1}^{N} x_{ij} = 0, \quad \frac{1}{N}\sum_{i=1}^{N} x_{ij}^2 = 1 \qquad j = 1, \ldots, p$$

$$\min_{(\beta_0,\beta)\in R^{p+1}} \left[\frac{1}{2N}\sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda P_\alpha(\beta) \right]$$

ここで

$$P_\alpha(\beta) = (1-\alpha)\frac{1}{2}\| \beta \|_{l2}^2 + \alpha \| \beta \|_{l1}$$

$$= \sum_{j=1}^{N} \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha | \beta_j | \right]$$

$\alpha = 0$ とした場合，Ridge 回帰となる。

$\alpha = 1$ とした場合，Lasso 回帰となる。

$0 < \alpha < 0$ とした場合，Elastic Net となる。

λ は，complexity Parameter と呼ばれ，λ が 0 だと，通常の最小二乗法に一致する。

２．R での関数と λ

library(glmnet)には，glmnet 関数と cv.glmnet 関数があります。

cv.glmnet 関数は，交差確認法（cross validation）用です。

いずれの関数もパラメータ λ は，次の２通りのどちらかの方法で指定します。

(1)　直接 λ の値群を指定する。交差確認法では，この各 λ での MSE(Mean Squared Error)により最適な λ を決めます。

---

[1] Regularization Paths for Generalized Linear Models via Coordinate Descent , http://www.stanford.edu/~hastie/Papers/glmnet.pdf

(2)　自動設定（関数にお任せ）方法　但し λ_{max} と λ_{min} の比(lambda.min.ratio)と個数(nlambda)を指定して，λ 列を作成する。交差推定は，この λ 列で行います。

それぞれの引数の省略時の値は，

lambda.min.ratio = ifelse(nobs<nvars,0.01,0.0001)

nlambda = 100

です。　nobs は，データ数 N で，nvars は，説明変数数 p である。

$\lambda_{min} = \lambda_{max}$ * lambda.nin.ratio から，λ_{max} を決めれば，log スケールで nlambda 個を分割すれば，λ 列を作成出来ます。この時の等比級列の値(alf)は，

alf**(nlambda-1) = lambda.nin.ratio から，

alf = lambda.nin.ratio**(1/(nlambda-1)) となります。

nobs=100, nvars=20, nlambda=100 とすると，lambda.nin.ratio=0.0001 で

alf = 0.0001**(1/99)= 0.9111628

この値は, lambda.min.ratio と nlambda から決まるので, データが異なっても同じです。（パラメータを変えても，alf は，同じです）

問題は，どのようにして λ_{max} の値を決定するかです。これは，次章で検討します。

この alf 値について実際に R の関数で検証してみます。

glmnet 関数の help の Examples での例 (先頭の５つの比)

glmnet 関数の結果の　lambda ベクトルが，自動設定された λ 列（降順）です。

先頭の５つの比をみてみます。

```
> x=matrix(rnorm(100*20),100,20)
> y=rnorm(100)
> fit1.1=glmnet(x,y,alpha=1)      #   α=1
> fit1.2=glmnet(x,y,alpha=0)      #   α=0
> fit1.3=glmnet(x,y,alpha=0.2)    #   α=0.2

> fit1.1$lambda[2:6]/fit1.1$lambda[1:5]
[1] 0.9111628 0.9111628 0.9111628 0.9111628 0.9111628
> fit1.2$lambda[2:6]/fit1.2$lambda[1:5]
[1] 0.9111628 0.9111628 0.9111628 0.9111628 0.9111628
> fit1.3$lambda[2:6]/fit1.3$lambda[1:5]
[1] 0.9111628 0.9111628 0.9111628 0.9111628 0.9111628
```

すべて同じ値です。

３．λ_{max}の決め方

λ_{max}は，λを huge-value（この時はパラメータβがすべてゼロ）の時と同等なλの最小値であります。このことの記述が glmnet Package source 内[2]にあります。

lambda.max is not given, but easily computed from the input x and y; it is the smallest value for lambda such that all the coefficients are zero. For alpha=0 (ridge) lambda.max would be infty; hence for this case we pick a value corresponding to a small value for alpha close to zero.)

また，１章で参照した PDF 内の 2.5 Pathwise Coordinate Descent 内には，

$$\frac{1}{N}|\langle x_j, y\rangle| < \lambda\alpha \qquad j = 1,\ldots,p$$

ここで　$\langle x_j, y\rangle = \sum_{i=1}^{N} x_{ij} y_i$

これから

$$N\alpha\lambda_{max} = \max_i |\langle x_l, y\rangle|$$

αが小さい場合（特に$\alpha = 0$）には，$\alpha = 1.0e-3 = 0.001$にしているようです。

（glmnet Package source 内　glmnet¥src¥ glmnet5.f90 内で確認，1.0e-3 は固定値）

つまり

$$\lambda_{max} = \frac{\max_i |\langle x_l, y\rangle|}{N * \max(\alpha, 0.001)}$$

ここで　x_iは標準化された値

これは，関数に引数 family が”gaussian”（正規線形モデル）の場合です，他のタイプでは異なるようです。他のタイプに関しては，次章で検討します。

次に，glmnet 関数の help の Examples で，λ_{max}の式とλ_{max}と huge-value での同等の確認をおこないます。

３．１　glmnet 関数の help の Examples で，λ_{max}の式の確認

glmnet 関数の結果の　lambda ベクトルが，自動設定されたλ列で，１番目の値がλ_{max}です。

```
> set.seed(1)
> x=matrix(rnorm(100*20),100,20)        #   nrow=100, ncol=20
> y=rnorm(100)
> xx<-set.ms(x)                          #   標準化 set.ms 後述
> xxy <- matrix(y, nrow=100, ncol=20)*xx   #     x_ij y_i
```

[2] glmnet_2.0-10.tar.gz　　glmnet¥inst¥doc¥ glmnet_beta.pdf

```
#   α=1
> max(abs(apply(xxy,2,sum)))/(100*1)
[1] 0.1975946
> fit1.1=glmnet(x,y,alpha=1)
> fit1.1$lambda[1]
[1] 0.1975946
#   α=0  ---→ 0.001
> max(abs(apply(xxy,2,sum)))/(100*0.001)
[1] 197.5946
> fit1.2=glmnet(x,y,alpha=0)
> fit1.2$lambda[1]
[1] 197.5946
#   α=0.2
> max(abs(apply(xxy,2,sum)))/(100*0.2)
[1] 0.9879729
> fit1.3=glmnet(x,y,alpha=0.2)
> fit1.3$lambda[1]
[1] 0.9879729
> set.ms <- function (x)
{
        l <- dim(x)[1]
        m <- apply(x, 2, mean)
        s <- apply(x, 2, sd)*sqrt((l-1)/l)
        n <- dim(x)[2]
        xx <- x
        for(k in 1:n)
        xx[,k] <- (x[,k]-m[k])/s[k]
        xx
}
```

３．２　λ_{max} と huge-value での同等の確認

Huge-value は，lmnet Package source 内　glmnet¥src¥ glmnet5.f90 内では，99e35 の値を使用して計算しています。この値から上記計算式の λ_{max} までの λ は，すべて同等の結果です。　実際に lambda 引数に 99e35, 10000, λ_{max} の３値を指定して，回帰係数 β_0, β を glmnet 関数で計算して比較してみます。

$\alpha = 1$

```
> fit1.1a=glmnet(x, y, alpha=1, lambda=c(99e35, 10000, fit1.1$lambda[1]))
> fit1.1a$a0      # 係数 β0
          s0           s1           s2
-0.1417928 -0.1417928 -0.1417928
> fit1.1$b            # 係数 β
20 x 3 sparse Matrix of class "dgCMatrix"
      s0 s1 s2
V1     0  0  0
V2     .  .  .
.
.
V19    .  .  .
V20    .  .  .
```

$\alpha = 0$

```
> fit1.2a=glmnet(x ,y ,alpha=0, lambda=c(99e35, 10000, fit1.2$lambda[1]))
> fit1.2a$a0
          s0           s1           s2
-0.1417928 -0.1417930 -0.1418009
> fit1.2a$b
20 x 3 sparse Matrix of class "dgCMatrix"
                    s0                 s1                 s2
V1     1.134031e-38   1.122573e-05   5.649140e-04
V2     1.634864e-38   1.618346e-05   8.158905e-04
.
.
V19    6.534351e-39   6.468080e-06   3.249566e-04
V20 -2.074094e-39 -2.052037e-06 -1.005880e-04
```

$\alpha = 0.2$

```
> fit1.3a=glmnet(x ,y, alpha=0.2, lambda=c(99e35, 10000, fit1.3$lambda[1]))
> fit1.3a$a0
          s0           s1           s2
-0.1417928 -0.1417928 -0.1417928
> fit1.3a$b
20 x 3 sparse Matrix of class "dgCMatrix"
```

s0 s1 s2
V1 0 0 0
V2 . . .
.
.
V19 . . .
V20 . . .

$\alpha=1$，$\alpha=0.2$は，問題ないです。定数項の係数β_0(a0)は，yの平均値で，係数βはすべてゼロです。$\alpha=0$（Ridge 回帰）では，係数βがジャストゼロにならないのが特長のようで（係数の二乗和のみで，絶対値和がないからか？），ちょっと差が生じます。このことは，「リッジ回帰は、罰則を課す回帰手法の中では初期のものの 1 つです。リッジ手法は係数を 0 に収縮しないので、変数の選択は行いません。」という記述が JMP マニュアルにあるそうです。

また，glmnet 関数のλの自動設定時での結果の１番目（λ_{max}）の係数は，λを 99e35 として計算した係数です。（λ_{max}での係数ではありません)。λ_{max}値は，２番目のλ値を計算するのに使用しています。
実際には glmnet 関数の最後の処理で，２番目と３番目のλ値から１番目の値を計算して設定しています。

$$\lambda_1 = \exp(2\log(\lambda_2) - \log(\lambda_3))$$

４．gaussian 以外のタイプの検討

今回の参考文献内で，gaussian 以外のタイプのλ_{max}の記述は見つかりませんでしたが，以下は FORTRAN ソース(glmnet5.f90)を見ての推測です。

(1) multivariate Gaussian

目的関数Yが，l次元

データ　y_{ij}　　$i=1,\ldots,N$　　$j=1,\ldots,l$

$$\lambda_{\max} = \frac{\max_i \sqrt{\sum_{j=1}^{l} \langle y_j, x_i \rangle^2}}{N * \max(\alpha, 0.001)}$$

ここで　$\langle x_j, y \rangle = \sum_{i=1}^{N} x_{ij} y_i$

glmnet 関数の help の Examples
> y=matrix(rnorm(100*3), 100, 3)
> fit1m=glmnet(x, y, family="mgaussian", alpha=1)

[1] 0.2506854 0.2284152 0.2081234 0.1896343 0.1727877

```
> y1<-y[,1]
> y2<-y[,2]
> y3<-y[,3]
> xxy1 <- matrix(y1, nrow=100, ncol=20)*xx
> xxy2 <- matrix(y2, nrow=100, ncol=20)*xx
> xxy3 <- matrix(y3, nrow=100, ncol=20)*xx
> tmp1<-apply(xxy1, 2, sum)^2
> tmp2<-apply(xxy2, 2, sum)^2
> tmp3<-apply(xxy3, 2, sum)^2
> max(sqrt(tmp1+tmp2+tmp3))/(100*1)
[1] 0.2506854
```

(2) binomial　一般化線形モデル　ロジスティック回帰

目的関数Yが　[1,2] の 2 値のベクトル　データ　y_i

$$\lambda_{\max} = \max_i |\langle x_i, y-1\rangle| \Big/ N * \max(\alpha, 0.001)$$

ここで　$\langle x_j, y-1\rangle = \sum_{i=1}^{N} x_{ij}(y_i - 1)$

Y を[0 1] に変換して，gaussian の式を採用

glmnet 関数の help の Examples

```
> g2=sample(1:2,100,replace=TRUE)
> fit2=glmnet(x, g2, family="binomial", alpha=1)
> fit2$lambda[1:5]
[1] 0.1557289 0.1418944 0.1292889 0.1178032 0.1073379

> xg10 <- matrix(g2-1, nrow=100, ncol=20)*xx
> tmp10 <- apply(xg10, 2, sum)/(100*1)
> max(abs(tmp10))
[1] 0.1557289
```

＊別式

Y を行数(nrow)が 2 の行列 g_{il} に変換

$$y_i = 1 \rightarrow g_{i1} = 1, g_{i2} = 0$$
$$y_i = 2 \rightarrow g_{i1} = 0, g_{i2} = 1$$

$$\lambda_{\max} = \frac{\max_{i,l} |\langle x_l, g_l \rangle|}{N * \max(\alpha, 0.001)}$$

ここで $\langle x_j, g_l \rangle = \sum_{i=1}^{N} x_{ij} g_{il}$

```
> yg2 <- diag(2)[g2,]
> xg1 <- matrix(yg2[,1], nrow=100, ncol=20)*xx
> xg2 <- matrix(yg2[,2], nrow=100, ncol=20)*xx
> tmp1 <- apply(xg1, 2, sum)/(100*1)
> tmp2 <- apply(xg2, 2, sum)/(100*1)
> max(tmp1,tmp2)
[1] 0.1557289
```

(3) multinomial　　一般化線形モデル　多項ロジスティック回帰

目的関数 Y が　[1,2, …,k] のベクトル(k 種類の値)　データ　y_i

Binomial の別式のように，Y を行数(nrow)が k の行列 g_{il} に変換

$$y_i = m \rightarrow g_{im} = 1, g_{il} = 0 (l \neq m)$$

$$\lambda_{\max} = \frac{\max_{i,j} |\langle x_l, g_l \rangle|}{N * \max(\alpha, 0.001)}$$

と推定しましたが，結果が少し違います。

glmnet 関数の help の Examples

```
> g4=sample(1:4, 100, replace=TRUE)
> fit3=glmnet(x, g4, family="multinomial", alpha=1)
> fit3$lambda[1:5]
[1] 0.11285781 0.10283183 0.09369653 0.08537279 0.07778851

> yg4 <- diag(4)[g4,]
> xg1 <- matrix(yg4[,1], nrow=100, ncol=20)*xx
> xg2 <- matrix(yg4[,2], nrow=100, ncol=20)*xx
```

```
> xg3 <- matrix(yg4[,3], nrow=100, ncol=20)*xx
> xg4 <- matrix(yg4[,4], nrow=100, ncol=20)*xx
> tmp1 <- apply(xg1, 2, sum)/(100*1)
> tmp2 <- apply(xg2, 2, sum)/(100*1)
> tmp3 <- apply(xg3, 2, sum)/(100*1)
> tmp4 <- apply(xg4, 2, sum)/(100*1)
> max(tmp1,tmp2,tmp3,tmp4)
[1] 0.09579216
```

結果が違います。g4 を変えると一致する場合もあり，現在検討中。

(4) poisson　一般化線形モデル　　ポアソン回帰

目的関数Yは，ゼロ以上の整数値

Gaussian と同じ式

$$\lambda_{\max} = \frac{\max_i |\langle x_l, y\rangle|}{N * \max(\alpha, 0.001)}$$

ここで　$\langle x_j, y\rangle = \sum_{i=1}^{N} x_{ij} y_i$

glmnet 関数の help の Examples

```
> N=500; p=20
> nzc=5
> x=matrix(rnorm(N*p),N,p)
> beta=rnorm(nzc)
> f = x[,seq(nzc)]%*%beta
> mu=exp(f)
> y=rpois(N,mu)
> fit=glmnet(x ,y, family="poisson", alpha=1)
> fit$lambda[1:5]
[1] 9.034805 8.232178 7.500854 6.834499 6.227341

> xx<-set.ms(x)
> xxy <- matrix(y, nrow=500, ncol=20)*xx
> max(abs(apply(xxy, 2, sum)))/(500*1)
[1] 9.034805
```

以上