统计模型及其R实现笔记

Statistical Models with R

马学俊

副教授，苏州大学数学科学学院统计系，主要从事海量数据分析、高维数据分析、统计计算、非参数回归等统计模型及其应用等研究。个人主页https://xuejunma.github.io.

献给我的家人、恩师和所有在学术道路上帮助我的人

缩写与记号

- s.t. ：约束
- $\boldsymbol{Y}$：向量
- $\mathbf{X}$ ：矩阵（不包含向量）
- $\mathbb{R}^k$：k维欧式空间
- $\top$：转置

目　录

表　　格

第1章　变量选择

假设回归模型是

$$Y = X^\top \beta + \varepsilon \tag{1.1}$$

其中Y是一维随机变量，$X = (X_1, \ldots, X_p)$是p维随机变量，ε是一维随机变量。$\beta = (\beta_1, \ldots, \beta_p)^\top$ 是未知参数。假设

$$E(\varepsilon | X = x) = 0, \tag{1.2}$$

模型(1.1)可以表示为：

$$E(Y | X = x) = x^\top \beta. \tag{1.3}$$

上述模型是均值回归(Mean regression)，其参数可以通过下面得到:

$$\min E(Y - x\beta)^2 \tag{1.4}$$

备注 假条条件(1.2)不同，可以得到不同类别的统计模型，比如分位数回归(Quantile regression)和众数回归(Mode regression)。这里主要讨论均值回归。

假设$(\boldsymbol{x}_1, y_1)$,...,$(\boldsymbol{x}_n, y_n)$ 是一组样本，其中$\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$，表达式(1.4)的样本实现值为

$$\min_\beta \sum_{i=1}^n (y_i - \boldsymbol{x}_i^\top \beta)^2 \tag{1.5}$$

经过简单运算，$\hat{\beta}_{ols} = (\boldsymbol{x}^\top \boldsymbol{x})^{-1} \boldsymbol{x}^\top y$，其中$y = (y_1, \ldots, y_n)^\top$。$\boldsymbol{x} = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top)^\top$是$n \times p$的设计矩阵(Design matrix，矩阵是设计好的，换句话为是给定，直白点说就是已知的) 在条件(1.2)，$\hat{\beta}_{ols}$是无偏估计（注意没有其它假设条件，如果想证明其它的性质，如渐近性质或，需要其它条件。

现在，如果$\boldsymbol{x}^\top \boldsymbol{x}$不可逆，也就是说$\boldsymbol{x}$不是列满秩，那么$\hat{\beta}_{ols}$的不存在。上面这种线性成为完全共线性。这也是为什么研究变量选择的一个重要原因。下面我们来讨论另一个原因，假如研究儿童身高的影响因素，我们收集了性别、体重、父亲体重、母亲体重、家里花草的数量等几百个因素，目的是找到主要影响因素。大家注意，我们这里其实有一个假设，那就是儿童身高的影响因素

是很少的，也只有几个。换句统计的词汇就是“稀疏性假设”。“家里花草的数量”显然就是需要排除的因素。排除因素就是变量选择。除了上述原因外，还有

- 估计量的方差变大，预测的精度较低；
- 过拟合，保留大量的解释变量会降低模型的可解释性。

怎么进行变量选择或者消去共线性，我们学习了很多方法，比如最有子集方法(best subset method)，逐步回归和岭回归(Ridge regression)等。下面简单介绍这几种方法：

最优子集方法对p个变量的所有可能组合分别进行拟合，选择残差平方和(Residual square sum) 或者R^2最小的模型。最优子集的优点是简单直观，但效率太低，当p很大时，从一个巨大的搜索空间中得到的模型通常会有过拟合和系数估计方差高的问题；改进的子集选择还有逐步选择(向前、向后)，与全子集相比限制了搜索空间，提高了运算效率，但是无法保证找到的模型是2^p个模型中最优的。

岭回归是求带有约束的凸优化问题：

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \beta)^2 \tag{1.6}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} \beta_j^2 \leq t \tag{1.7}$$

其中s.t.是subject to的缩写，表示约束条件。$t > 0$。引入Lagrange乘子可以转化为，上面的优化问题可以转化为

$$\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Big) \tag{1.8}$$

岭回归得到的估计量是有偏的，但方差小了，得到的均方误差小，也就是其牺牲了无偏性，降低了方差。

1.1　LASSO及其拓展

LASSO(Least absolute shrinkage and selection operator) 是Tibshirani (1996)提出，其求下面目标函数最小值

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^{\top}\beta)^2 \tag{1.9}$$

$$\text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq t \tag{1.10}$$

上式等价于

$$\min_{\beta} \Big(\sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^{\top}\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \Big) \tag{1.11}$$

其中λ 是截断参数(Tuning parameter)。我们可以采用(Wang et al. (2007))方法，

$$\text{BIC}(\lambda_n) = \log \Big(\sum_{i=1}^{n} (Y_i - Z_i^{\top}\beta) \Big)^2 + \frac{\log n}{n} \times df$$

其中df 估计非零的参数的个数。$\lambda_{opt} = \arg\min_{\lambda_n} \text{BIC}(\lambda_{\text{n}})$.

相比岭回归，LASSO只是将约束条件修改为绝对值。这样做为什么可以选择变量？从图1.1，LASSO更有可能得到稀疏的解，即某一个解为0。这是由于解易出现菱角或者边缘。对于岭回归而言是约束域是圆，所以每一点的可能性相同，而矩阵有几个角，角的可能性更大些。

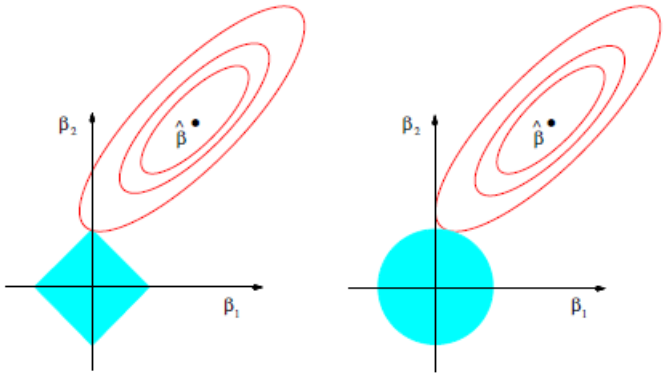

图 1.1: LAOO和岭回归的几何解释；左边是LASSO，右图是岭回归

LASSO被提出后，后面有很多文章提出了不同的方法，如SCAD (Fan and Li (2001))和Adaotive LASSO (Zou (2006)). 一般而言，后者更为简单，其为：

$$\min_{\beta}\Big(\sum_{i=1}^{n}(y_i-\boldsymbol{x}_i^{\top}\beta)^2+\lambda\sum_{j=1}^{p}w_j|\beta_j|\Big) \tag{1.12}$$

其中$w_j=\frac{1}{|\widetilde{\beta}_j|^{\kappa}}$, and $\kappa>0$. $\widetilde{\beta}$ 最小二乘的解Zou (2006)建议$\kappa=1$。

下面我们讨论几种变量选择的关系。假设$\boldsymbol{x}^{\top}\boldsymbol{x}=I$，其中$I$是单位矩阵。所以$\hat{\beta}_{ols}=\boldsymbol{x}^{\top}y,\hat{y}_{ols}=\boldsymbol{x}\boldsymbol{x}^{\top}y$，且$\boldsymbol{x}^{\top}(y-\hat{y}_{ols})=0$。我们考虑一般的变量选择的惩罚函数形式分析：

$$\frac{1}{2}\|y-\boldsymbol{x}\beta\|^2+\lambda\sum_{j=1}^{p}p_{\lambda}(|\beta_j|) \tag{1.13}$$

其中$p_{\lambda}(\cdot)$是罚函数。经过计算，我们可以得到：

$$\begin{aligned}&\frac{1}{2}\|y-\boldsymbol{x}\beta\|^2+\lambda\sum_{j=1}^{p}p_{\lambda}(|\beta_j|)\\&=\frac{1}{2}\|y-\hat{y}_{ols}|^2+\frac{1}{2}\sum_{j=1}^{p}\|\hat{\beta}_{ols,j}-\beta_j\|^2+\lambda\sum_{j=1}^{p}p_{\lambda}(|\beta_j|)\end{aligned}$$

这是因为:

$$\begin{aligned}&(y-\boldsymbol{x}\beta)^{\top}(y-\boldsymbol{x}\beta)\\&=(y-\hat{y}+\hat{y}-\boldsymbol{x}\beta)^{\top}(y-\hat{y}+\hat{y}-\boldsymbol{x}\beta)\\&=(y-\hat{y})^{\top}(y-\hat{y})+(\hat{y}-\boldsymbol{x}\beta)^{\top}(\hat{y}-\boldsymbol{x}\beta)+2(\hat{y}-\boldsymbol{x}\beta)^{\top}(y-\hat{y})\\&=\|y-\hat{y}\|^2+(\hat{\beta}_{ols}-\beta)^{\top}\boldsymbol{x}^{\top}\boldsymbol{x}(\hat{\beta}_{ols}-\beta)+2(\hat{y}-\boldsymbol{x}\beta)^{\top}(y-\hat{y})\end{aligned}$$

模型(1.13)可以转化为

$$\frac{1}{2}(\hat{\beta}_{ols,j}-\beta_j)^2+\lambda p_{\lambda}(|\beta_j|),$$

更为一般的形式为：

$$\frac{1}{2}(z-\theta)^2+\lambda p_{\lambda}(|\theta|),$$

$p_{\lambda}(\cdot)$取不同的形式对应不同的方法：

1. $p_{\lambda}(\theta)=\lambda^2-(|\theta|-\lambda)I(|\theta|<\lambda)|\theta|^2$是最优子集估计量。

2. $p_\lambda(\theta)=\lambda|\theta|^2$是岭回归估计量
3. $p_\lambda(\theta)=\lambda|\theta|$是LASSO估计量
4. $p'_\lambda(\theta)=\lambda\Big\{I(\theta<\lambda)+\frac{(a\lambda-\theta)_+}{(a-1)\lambda}I(\theta\geq\lambda)$ 是SCAD估计量

经过推断，我们可以得到如下结论：

1. 最优子集的估计量$\hat{\theta}=zI(|z|>\lambda)$
2. 岭回归估计量$\hat{\theta}=\frac{z}{1+2\lambda}$
3. LASSO估计量$\hat{\theta}=\mathrm{sgn}(z)(|z|-\lambda)_+$
4. SCAD估计量

$$\hat{\theta}=\begin{cases}\mathrm{sgn}(z)(|z|-\lambda)_+ & |z|\leq 2\lambda\\ \frac{(a-1)z-\mathrm{sgn}(z)a}{\lambda}a-2 & 2\lambda\leq|z|\leq a\lambda\text{x!=0}\\ z & |z|\geq a\lambda\end{cases}$$

下面我们介绍Fan and Li (2001)提出好的罚函数应该具备如下性质：

1. 无偏性(Unbiasedness)：对于较大的θ，$p'_\lambda(|\theta|)=0$，则$\hat{\theta}=z$。
2. 稀疏性(Sparsity)：$\min_{\theta\neq0}\{|\theta|+p'_\lambda(|\theta|)\}>0$，则解具有稀疏性，即当$|z|<\min_{\theta\neq0}\{|\theta|+p'_\lambda(|\theta|)\}$时,$\hat{\theta}=0$.
3. 连续性(Continuity)：$\min\{|\theta|+p'_\lambda(|\theta|)\}$

经过计算，我们可以得到

表 1.1: 几种罚函数的比较

方法	无偏性	稀疏性	连续性
最优子集	√	√	
岭回归			√
LASSO		√	
SCAD	√	√	√

1.2 算法

1.2.1 二次逼近算法

关于惩罚函数求解的方法有很多。我们主要介绍局部二次逼近算法(Local quadratic approximation,Fan and Li (2001))。该算法采用二次逼近罚函数。假设β^*比较接近最小值。

- 如果β_j^*非常接近0，我们设定$\hat{\beta}_j = 0$.
- 否则，我们使用二次逼近罚函数$p_\lambda(|\beta_j|)$

由于，当$\beta_j \neq 0$，且$\beta_j^* \approx \beta_j$

$$[p_\lambda(|\beta_j|)]' = p'_\lambda(|\beta_j|)\text{sgn}\beta_j \approx \frac{p'_\lambda(|\beta_j^*|)}{|\beta_j^*|}\beta_j.$$

换句话说：

$$p_\lambda(|\beta_j|) \approx p_\lambda(|\beta_j^*|) + \frac{1}{2}\frac{p'_\lambda(|\beta_j^*|)}{|\beta_j^*|}\left(\beta_j^2 - (\beta_j^*)^2\right) \tag{1.14}$$

上面的式子是$\beta_j^2 - (\beta_j^*)^2$(不考虑1/2)，不是$|\beta_j| - |\beta_j^*|$。根据Taloy展开式这一项应该是绝对值的差，但作者使用了平方，因为平方可导。这也是称为二次逼近的原因。能不能用绝对值？当然可以。它的名字是局部线性逼近(Local linear approximation, Zou and Li (2008))。二次逼近可导，线性逼近不可导；二次逼近的解不稀疏，而线性逼近的解稀疏。关于二次逼近的讨论，详见Lee, Kwon and Kin (2016)。

将上面逼近代入一般的惩罚函数的表达式为：

$$\min\left\{(y - \boldsymbol{x}\beta)^\top(y - \boldsymbol{x}\beta) + \sum_{j=1}^{p}\left[p_\lambda(|\beta_j^*|) + \frac{1}{2}\frac{p'_\lambda(|\beta_j^*|)}{|\beta_j^*|}\left(\beta_j^2 - (\beta_j^*)^2\right)\right]\right\}$$

由于β_j^*是给定的，所以上面的式子进一步转化为：

$$\min\left\{(y - \boldsymbol{x}\beta)^\top(y - \boldsymbol{x}\beta) + \sum_{j=1}^{p}\frac{1}{2}\frac{p'_\lambda(|\beta_j^*|)}{|\beta_j^*|}\beta_j^2\right\} \tag{1.15}$$

为了记号方便，令$u_i(\beta_j^*) = \frac{1}{2}\frac{p'_\lambda(|\beta_j^*|)}{|\beta_j^*|}$，则

$$\beta = (\boldsymbol{x}^\top\boldsymbol{x} + U(\beta^*))^{-1}\boldsymbol{x}^\top y \tag{1.16}$$

其中$U(\beta^*) = \text{diag}\{u_1(\beta_j^*), \ldots, u_p(\beta_j^*)\}$.

具体算法如下：

1. 第一步：给定初始值$\beta^{(0)}$,
2. 第二步：令$\beta^{(m)} = \beta^{(0)}$ 根据(1.16)更新$\beta^{(m+1)}$
3. 第三步：重复第二步，直到其收敛。

对于Adaptive LASSO，而言，$u_i(\beta_j^*) = \lambda \frac{1}{2|\hat{\beta}_{ols,j}|} \frac{1}{|\beta_j^*|}$。我们进行下面模型模型看看算法效果如何。

在进行分析前，我们需要解决常数项的问题，变量惩罚不会对其对其进行惩罚。下面我们简单的说明一下。回归模型为

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$$

其经验的回归方程是：

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \cdots + \hat{\beta}_1 x_1 \tag{1.17}$$

经过中心化$(\overline{x}_1, \ldots, \overline{x}_p, \overline{y})$的变量为$(\widetilde{x}_1, \ldots, \widetilde{x}_p, \widetilde{y})$，则令

$$\widetilde{y} = \hat{\beta}_1^\star \widetilde{x}_1 + \cdots + \hat{\beta}_p^\star \widetilde{x}_p$$

进行得到：

$$y - \overline{y} = \hat{\beta}_1^\star (x_1 - \overline{x}_1) + \cdots + \hat{\beta}_p^\star (x_p - \overline{x}_p)$$

从而得到

$$y = (\overline{y} - \hat{\beta}_1^\star \overline{x}_1 - \cdots - \hat{\beta}_p^\star \overline{x}_p) + \hat{\beta}_1^\star x_1 + \cdots + \hat{\beta}_p^\star x_p \tag{1.18}$$

比较(1.17)和(1.18)，我们可以得到:

$$\begin{aligned} \beta_0 &= \overline{y} - \hat{\beta}_1^\star \overline{x}_1 - \cdots - \hat{\beta}_p^\star \overline{x}_p \\ \hat{\beta}_j &= \hat{\beta}_j^\star, \ \ j = 1, \ldots, p \end{aligned}$$

下面是二次逼近的代码，我们可以修改阈值(Threshold)看看变量选择结果，默认`thre=1e-2`，如果估计参数值的绝对值小于0.01设置为0。从输出的结果来看，算法的效果还可以。

Adaptive LASSO二次逼近代码

```
rm(list=ls())
library(MASS)

data1 <- function(n, p, sig0=0.25, rho=0.5){
  sigm <- sig0^abs(outer(1:(p),1:(p),"-"))
  muz <- rep(0, p)
  x <- mvrnorm(n = n, mu=muz, Sigma=sigm)
  e <- rnorm(n, 0, 1)
  beta <-c(3, 1.5, 0, 0, 2, rep(0, p-5))
  y <-  x %*% beta + rho * e
  return(list(y=y, x=x, beta=beta))
}

p <- 8
n <- 200
dat <- data1(n = n, p = p)
x <- dat$x
y <- dat$y
beta.ture <- dat$beta

#lam:lambda
#eps:
#itemax: maximum iteration time
#thre:  threshold
adalasso.my <- function(x, y, lam=0.1, eps=1e-5, itemax=1000, thre=1e-2, intercept=
  p <- dim(x)[2]
  n <- dim(x)[1]
  x_colname <- colnames(x)
  beta_ols <- coef(lm(y~x-1))
  BB <- beta_ols
  if(is.null(x_colname)==TRUE){x_colname <- paste("x", 1:p, sep = "")}
  #
  if(intercept==TRUE){
    ##################################inclued intercept begin
    x_mean <- apply(x, 2, mean)
```

```
36    y_mean <- mean(y)
37    x_c <- scale(x=x, center=TRUE, scale=FALSE)
38    y_c <- y - mean(y)
39    txx <- t(x_c) %*% x_c
40    txy <- t(x_c) %*% y_c
41    iter <- 0
42    juli <- 1
43    ############## loop begin
44    while( juli > eps ){##eps <- 0.1 stop
45      u_beta <- lam * 1 / (abs(beta_ols) * abs(BB))
46      U_beta <- diag(u_beta)
47      BB_new <- solve( txx + U_beta) %*% txy
48      cha  <-  BB_new  -  BB
49      juli  <- ( t(cha) %*% cha ) ^ 0.5
50      iter <- iter + 1
51      BB   <- BB_new
52      if(iter > itemax) break;
53    }
54    #############loop end
55    beta0 <- c(y_mean - x_mean %*% BB)
56    index_zero <- which(BB <= thre)
57    BB_threshold <- BB
58    BB_threshold[index_zero] <- 0
59
60    beta_full <- c(beta0, BB_threshold)
61    names(beta_full) <- c("Intercept", x_colname)
62    res <- y - (beta0 + x %*% BB_threshold)
63    BIC_my <- log(sum(res^2)) + log(n) / n * sum(BB_threshold!=0)
64    return(list(beta=beta_full, BIC=BIC_my, iter=iter))
65    ##################################inclued intercept end
66  }else{
67    ################################not inclued intercept begin
68    txx <- t(x) %*% x
69    txy <- t(x) %*% y
70    iter <- 0
71    juli <- 1
```

```
72      ##########loop begin
73      while( juli > eps ){##eps <- 0.1 stop
74        u_beta <- lam * 1 / (abs(beta_ols) * abs(BB))
75        U_beta <- diag(u_beta)
76        BB_new <- solve( txx + U_beta) %*% txy
77        cha  <-  BB_new  -  BB
78        juli  <- ( t(cha) %*% cha ) ^ 0.5
79        iter <- iter + 1
80        BB   <- BB_new
81        if(iter > itemax) break;
82      }
83      ########loop end
84      index_zero <- which(BB <= thre)
85      BB_threshold <- BB
86      BB_threshold[index_zero] <- 0
87      beta_full <- c(BB_threshold)
88      names(beta_full) <- x_colname
89      res <- y -  x %*% BB_threshold
90      BIC_my <- log(sum(res^2)) + log(n) / n * sum(BB_threshold!=0)
91      return(list(beta=beta_full, BIC=BIC_my, iter=iter))
92      ##################################not inclued intercept end
93    }
94 }
95
96 adalasso.my(x=x, y=y, intercept=TRUE)
97 adalasso.my(x=x, y=y, intercept=FALSE)
98
99 library(doParallel)
100 library(foreach)
101
102 cl <- makeCluster(8)
103 registerDoParallel(cl)
104
105 nlam <- 100
106 lam_min <- 0.05
107 lam_max <- 4
```

```
lam_v   <- seq(from=lam_min, to=lam_max, length=nlam)
BIC_lam <- foreach(lam=lam_v, .combine="rbind") %dopar%
  {adalasso.my(y=y, x=x, lam=lam, intercept=FALSE)$BIC}

##plot
plot(lam_v, BIC_lam)
index_optlam <- min(which(BIC_lam==min(BIC_lam)))
lam_opt <- lam_v[index_optlam]
print(lam_opt)
fit <- adalasso.my(y=y, x=x, lam=lam_opt)
##compared
cbind(fit$beta[-1], beta.ture)
stopCluster(cl)

```

输出的结果

```
> cbind(fit$beta[-1], beta.ture)
              beta.ture
x1 3.01404086       3.0
x2 1.50588042       1.5
x3 0.01028767       0.0
x4 0.00000000       0.0
x5 2.00791256       2.0
x6 0.00000000       0.0
x7 0.00000000       0.0
x8 0.00000000       0.0
```

1.2.2　坐标下降法

上面的算法不适合$p > n$的情况，下面我介绍一种常见的算法坐标下降法(Coordinate descent algorithm,Wu and Lang (2008))，该算法一个分量一个分量计算。

给定$\beta_1^{(k)}, \ldots, \beta_{j-1}^{(k)}, \beta_{j+1}^{(k)}, \ldots, \beta_p^{(k)}$，我们求解

$$\beta_j^{(k+1)} = \arg\min_{b_j} \left\{ Q(\beta_1^{(k)}, \ldots, \beta_{j-1}^{(k)}, \beta_j, \beta_{j+1}^{(k)}, \beta_p^{(k)}) + \lambda_n \frac{1}{2|\beta_j^{(k)}||\hat{\beta}_{ols,j}|} \beta_j^2 \right\} \quad (1.19)$$

其中$Q(\beta)$是$\|y-\boldsymbol{x}\beta\|^2$。算法的具体步骤如下：**大家可以尝试编写一下代码试试**)

- 第一步：初始值$\beta^{(0)}$
- 第二步：$k \geq 0$,给定$\beta^{(k)}$

2.1 对于$j \in 1,\ldots,p$, 利用(1.19)更新$b_j^{(k+1)}$

2.2 重复上面不步骤直到$b_j^{(k+1)}$ 收敛，从得到$\beta^{(k+1)}$

- 第三步：重复第二步直到$\beta^{(k)}$收敛。

1.3 组变量选择

顾名思义，组变量是一组变量。如分类变量具有3个水平，其需要转化为2个虚拟变量(Dummy variable)。这2个虚拟变量是一组。我们进行变量选择，不能只选择其中的一个变量保留另一个变量。为了解决这个问题，Yuan & Lin. (2006)提出了组变量LASSO，Huang, Breheny & Ma (2012)详细总结了组变量选择方法LASSO、SCAD和MCP，并且简单介绍了其在可加模型和变系数模型的应用。假设$(X_1,\ldots,X_p)$可以分成K组，其中每一组的自变量个数为d_k，则一般表达式为：

$$\frac{1}{2n}\|y-\sum_{k=1}^{K}X_k\beta_k\|_2^2+\sum_{k=1}^{K}p_\lambda(\|\beta_k\|_{R_k})$$

其中$\|v\|_R^2 = v^\top Rv$.通常R是一个单位矩阵。`grpreg`包中的函数`grpreg`可以实现LASS，SCAD和MCP的组变量选。

```
grpreg(X, y, group=1:ncol(X), penalty=c("grLasso", "grMCP", "grSCAD"),
            family=c("gaussian", "binomial", "poisson"), .....
```

下面是利用CV准则选择λ的命令。

组变量选择代码

```
1 rm(list=ls())
2 library(grpreg)
3 data(Birthwt)
4 summary(Birthwt)
5 X <- Birthwt$X
```

```
y <- Birthwt$bwt
group <- Birthwt$group

cvfit <- cv.grpreg(X, y, group)
plot(cvfit)
summary(cvfit)
coef(cvfit) ## Beta at minimum CVE
```

参考文献

Fan J. and Li R. (2001) Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American Statistical Association, 96(456): 1348–1360.

Huang, J., Breheny, P., & Ma, S. (2012). A selective review of group selection in high-dimensional models. Statistical science: a review journal of the Institute of Mathematical Statistics, 27(4).

Lee, S., Kwon, S. and Kim, Y. (2016). A modified local quadratic approximation algorithm for penalized optimization problems. Computational Statistics & Data Analysis, 94, 275-286.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 68(1), 49-67.

Wang H., Li R. and Tsai. C. (2007) Tuning parameter selectors for the smoothly clipped absolute deviation method. Biometrika, 94(3): 553–568, 2007.

Wang H. Li B. and Leng C. (2009) Shrinkage tuning parameter selection with a diverging number of parameters, Journal of the Royal Statistical Society. Series B (Statistical Methodology), 71(3): 671–683.

Tibshirani R. (1996) Regression shrinkage and selection via the LASSO. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 58(1): 267–288.

Wu, T. and Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. The Annals of Applied Statistics, 2(1), 224-244.

Zhang C. and Huang J.(2008) The sparsity and bias of the LASSO selection in highdimensional linear regression, The Annals of Statistics, 36(4): 1567–1594.

Zou H.(2006) The adaptive LASSO and its oracle property. Journal of the American Statistical Association, 101(476): 1418–1429.

Zou H. and Li R. One-step sparse estimates in nonconcave penalized likelihood models. Annals of statistics, 36(4):1509–1533.