线性回归模型的变量选择方法

王一鑫

2025 年 5 月 28 日

摘要

变量选择是构建高效线性回归模型的重要环节，对于提高模型解释性和预测精度具有重要意义. 本文系统介绍了几种常用的变量选择方法，包括**岭回归**（Ridge Regression）、**Lasso 回归**、**SCAD**（平滑剪切绝对偏差）方法以及**自适应 Lasso**（Adaptive Lasso）. 在阐述各方法数学原理的基础上，利用 R 语言对波士顿房价数据集进行实证分析，绘制了系数路径图与交叉验证误差图，直观展示了调节参数对模型的影响. 结果表明，不同方法在变量筛选能力、模型稀疏性和估计偏差方面各具特点，适应不同实际需求. 本文对比分析了这些方法的优劣，为实际数据建模中的变量选择提供了理论基础和实用参考.

关键词：变量选择；岭回归；Lasso；SCAD；自适应 Lasso

目录

1 变量选择方法

1.1 问题背景

“回归”的概念是 1886 年由英国统计学家 Galton 在研究父代身高与子代身高之间的关系时提出的. 目前回归分析已成为统计学应用最为广泛的方法之一，主要用于探索和检验协变量与相应变量之间的相关关系.

我们在对实际问题构建线性回归模型的过程中，可能会把对响应变量 Y 可能产生影响的协变量 $X_1, X_2, \cdots X_p$ 全部引入回归方程，这会导致模型中包含了一些对 Y 影响很小的变量，出现过拟合的现象，降低了模型的预测精度，因此对协变量做变量选择是非常必要的.

除了如 AIC 和 BIC 等经典的 L_0 惩罚变量选择方法之外，一些压缩估计方法也是重要的统计方法，例如岭回归、桥回归、Lasso、SCAD 和自适应 Lasso.

考虑多元线性回归模型，假设对 $Y, X_1, \ldots, X_p$ 进行了 n 次独立的试验，得到 n 组独立的观测值，即 $\{(y_i, x_{i1}, \ldots, x_{ip}), i = 1, \ldots, n\}$. 简便起见，对协变量数据进行标准化处理，使得

$$\sum_{i=1}^{n} x_{ij} = 0, \quad \sum_{i=1}^{n} x_{ij}^2 = 1. \tag{1}$$

则标准化后的数据满足

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i, \quad i = 1, \ldots, n. \tag{2}$$

引进矩阵记号：

$$\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{X} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix},$$

其中 $\boldsymbol{Y}$ 为 $n \times 1$ 的响应变量的观测向量，$\boldsymbol{X}$ 为 $n \times p$ 的已知设计矩阵，$\boldsymbol{\beta}$ 为 p 维的未知参数向量，$\boldsymbol{\varepsilon}$ 为 $n \times 1$ 的随机模型误差向量。模型 (2)写成如下矩阵形式：

$$\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}. \tag{3}$$

1.2 岭回归

1.2.1 模型原理

岭回归（Ridge Regression），也称为 Tikhonov 正则化（Tikhonov Regularization），是一种专门用于处理多重共线性（特征之间高度相关）问题的线性回归改进算法. 在多重共线性的情况下，数据矩阵可能不是满秩的，这意味着矩阵不可逆，因此不能直接使用普通最小二乘法（Ordinary Least Squares，OLS）来估计模型参数. 岭回归通过在损失函数中添加一个正则化项（惩罚项）来解决这个问题.

岭回归的损失函数是残差平方和（RSS）与正则化项的和. 通过极小化下面的惩罚最小二乘目标函数，可得到未知回归系数 $\boldsymbol{\beta}$ 的岭回归估计 $\hat{\boldsymbol{\beta}}^R = (\hat{\beta}_1, \ldots, \hat{\beta}_p)'$：

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2, \tag{4}$$

正则化项的作用是惩罚模型参数的大小。当 λ 增大时，正则化项的影响增大，参数 $\boldsymbol{\beta}$ 趋向于较小的值，这有助于减少模型的复杂度和过拟合的风险. 正则化项也可以使模型在面对多重共线性时更加稳定. 在没有正则化（即 $\lambda = 0$）的情况下，岭回归退化为普通最小二乘回归. 式 (4) 中，$\sum_{j=1}^{p}\beta_j^2$ 是惩罚项，针对回归系数 $\beta_1, \ldots, \beta_p$，作用是把接近于 0 的回归系数在 0 的方向进行压缩。

极小化惩罚最小二乘目标函数 (4)，求得岭回归估计为：

$$\hat{\boldsymbol{\beta}}^R = (\hat{\beta}_1, \ldots, \hat{\beta}_p)' = (\boldsymbol{X}'\boldsymbol{X} + n\lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}'\boldsymbol{Y}, \tag{5}$$

其中 $\boldsymbol{I}_p$ 是 p 维单位矩阵。由式 (5) 定义的岭回归估计可知，当 $\lambda = 0$ 时，岭回归估计就是最小二乘估计，即惩罚项不起任何作用. 随着 $\lambda \to \infty$，惩罚项的作用增强，岭回归估计也会随着 λ 增大越来越接近于 0.

岭回归的优势是平衡了偏差和方差，随着 λ 的增加，岭回归拟合的光滑度降低，尽管方差变小，但是偏差变大. 通过极小化下面的广义交叉验证（generalized cross-validation, GCV）目标函数，获得最优的调节参数 λ，即

$$\hat{\lambda}_{\text{gcv}} = \arg\min_{\lambda}\text{GCV}(\lambda) = \arg\min_{\lambda}\frac{\frac{1}{n}\|(\boldsymbol{I}_n - \boldsymbol{H}(\lambda))\boldsymbol{Y}\|_2^2}{[n^{-1}\text{tr}(\boldsymbol{I}_n - \boldsymbol{H}(\lambda))]^2}, \tag{6}$$

其中

$$\boldsymbol{H}(\lambda) = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X} + n\lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}'. \tag{7}$$

1.2.2 代码实践

在 **R** 语言中，程序包 `ridge` 中的函数 `linearRidge()`、程序包 `MASS` 中的函数 `lm.ridge()` 和程序包 `glmnet` 中的函数 `glmnet()` 都可以实现岭回归，其中在函数 `glmnet()` 中，`alpha=0` 拟合岭回归模型；`alpha=1` 拟合 Lasso 模型.

例 1.1. *波士顿房价预测是一个典型的回归问题，在机器学习领域中经常作为基准测试. 它利用了波士顿地区的房屋特征和相应的中位数价格数据，目标是建立一个预测模型来根据房屋的特征估算其价格. 这个问题的数据集通常来源于 1978 年，包含 506 个样本，每个样本有 13 个属性特征，如犯罪率、房产税率、学生-教师比例等，以及一个目标值，即房屋的中位数价格.*

首先导入必要的包，并加载 Boston 房价数据集.

```r
library(MASS)
library(glmnet)
library(ggplot2)
library(broom)
library(dplyr)

data("Boston")
x <- as.matrix(Boston[, -14])
y <- Boston$medv
```

岭回归，并整理数据，以便作图:

```r
fit_ridge <- glmnet(x, y, alpha = 0, nlambda = 100)
ridge_tidy <- tidy(fit_ridge) %>%
filter(term != "(Intercept)") %>%
mutate(log_lambda = log(lambda))
```

使用 `ggplot2` 自定义路径图，见图(1).

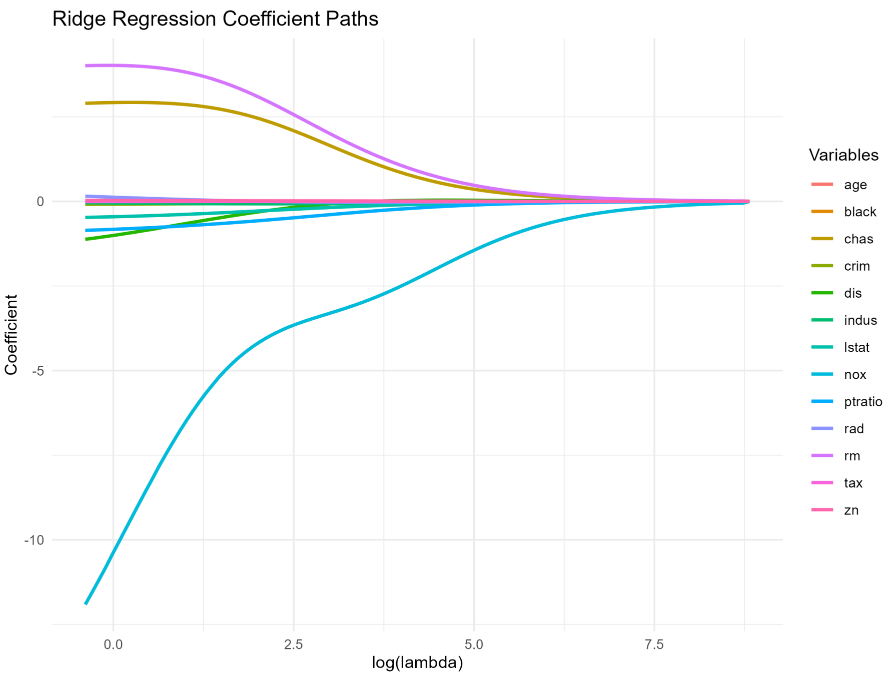

图 1: 岭回归估计随着 λ 变化的路径图.

```r
ridge_plot <- ggplot(ridge_tidy, aes(x = log_lambda, y =
    estimate, color = term)) +
geom_line(linewidth = 1) +
labs(
title = "Ridge Regression Coefficient Paths",
x = "log(lambda)", y = "Coefficient"
) +
theme_minimal() +
theme(legend.position = "right") +
guides(color = guide_legend(title = "Variables"))
```

从图(1)中可以看出，随着调节参数 λ 的增大，岭回归估计向原点收缩，但并不会使任何回归系数严格等于零.

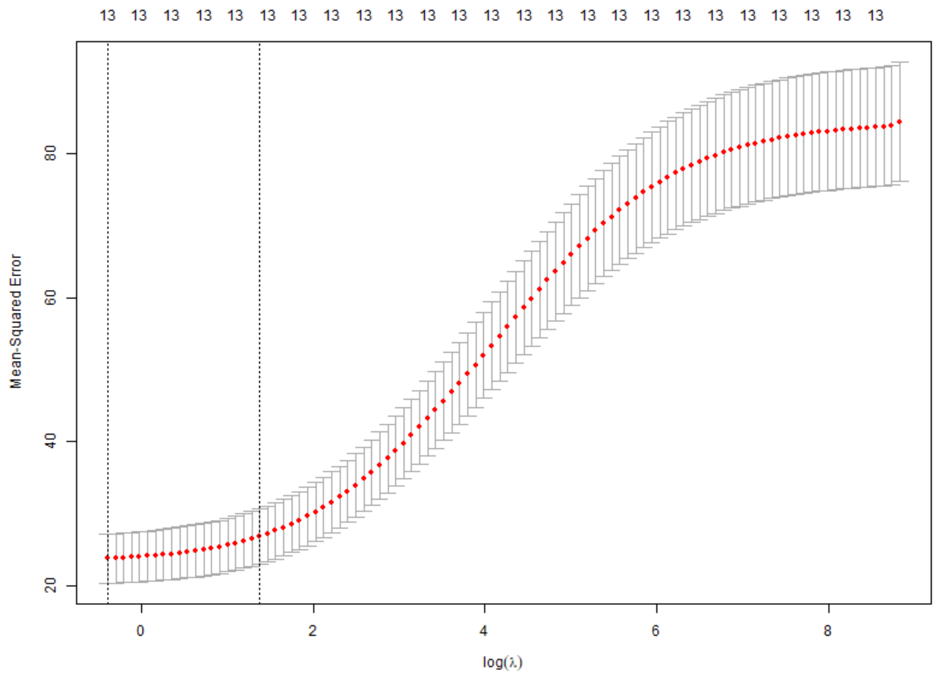

图 2: 岭回归的交叉验证误差图.

使用程序包 `glmnet` 中的函数 `cv.glmnet()` 选择最优的 λ，并绘制交叉验证误差图，见图(2).

```
set.seed(2025)
cv.ridge <- cv.glmnet(x, y, alpha = 0)
plot(cv.ridge, xlab = expression(log(lambda)))
```

对于图(2)，横轴为 $\log(\lambda)$，而纵轴为交叉验证误差，同时图上还显示了交叉验证误差的正、负标准差. 左边的垂直虚线表示能使得交叉验证误差最小的 $\log(\hat{\lambda})$ 取值. 右边的垂直虚线表示比 $\hat{\lambda}$ 更大，且与 $\mathrm{CV}(\hat{\lambda})$ 相距一个标准差的调节参数的取值，记为 $\log(\tilde{\lambda})$. 用 `cv.ridge$lambda.min` 获得 $\hat{\lambda} = 0.6777654$；用 `cv.ridge$lambda.1se` 获得 $\tilde{\lambda} = 3.969686$.

我们可以提取岭回归系数，结果见附录.

```
cv.ridge$lambda.min
cv.ridge$lambda.1se
coef(cv.ridge, s = "lambda.min")
coef(cv.ridge, s = "lambda.1se")
```

1.3 Lasso 方法

1.3.1 模型原理

岭回归的一个劣势是不能产生稀疏模型，即岭回归方法产生的最终模型还是包含 p 个预测变量，惩罚项 $\lambda\sum_{j=1}^{p}\beta_j^2$ 随着 λ 的增大可以把回归系数往 0 的方向推进压缩，但不会把任一个变量的回归系数压缩到 0（除非 $\lambda=\infty$）.

为了进行变量选择，考虑下面的惩罚最小二乘目标函数：

$$Q(\boldsymbol{\beta})=\frac{1}{2n}\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\|_2^2+\sum_{j=1}^{p}p_\lambda(|\beta_j|) \tag{8}$$

其中 $p_\lambda(\cdot)$ 是惩罚函数，λ 是调节参数或截断参数，是用来控制模型的复杂度，可以采用交叉验证（CV）方法、广义交叉验证（GCV）方法或 BIC 等数据驱动的准则进行选取. 一个好的惩罚函数应具有无偏性、稀疏性和连续性的性质.

Tibshirani 把 L_1 惩罚函数施加于回归模型的一般最小二乘和似然函数，提出了 Lasso 变量选择方法.

针对多元线性回归模型(2)，极小化下面的 L_1 惩罚最小二乘目标函数，可得回归系数 $\boldsymbol{\beta}$ 的 Lasso 估计 $\hat{\boldsymbol{\beta}}^L$：

$$\frac{1}{2n}\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|. \tag{9}$$

Lasso 估计 $\hat{\boldsymbol{\beta}}^L$ 也等价于求解下面的约束优化问题：

$$\begin{cases}\min\limits_{\boldsymbol{\beta}} & \dfrac{1}{2n}\displaystyle\sum_{i=1}^{n}\left(y_i-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2,\\ \text{s.t.} & \displaystyle\sum_{j=1}^{p}|\beta_j|\le c,\end{cases} \tag{10}$$

其中非负参数 c 的作用相同于调节参数 λ，c 的大小控制着 $\sum_{j=1}^{p}|\beta_j|$ 的大小. 随着 c 的变小，约束条件 $\sum_{j=1}^{p}|\beta_j|\le c$ 的作用会变强，这时会把回归系数接近于 0 的参数压缩到 0，从而产生稀疏模型. 寻找 Lasso 估计，就是寻找最优的调节参数 λ 或控制最大 L_1 范数的 c，找使得 RSS 最小的回归系数估计. 而对于调节参数 λ，可以通过一些数据驱动的方法进行选取，如 CV、GCV 或 BIC 准则进行选取.

岭回归只能满足连续性，而 Lasso 方法能够满足稀疏性和连续性.

Hastie 和 Tibshirani (1990) 定义了有效的自由度，它表示模型的复杂度. 为了定义 Lasso 的自由度，首先给出下面的 Stein 引理.

引理 1.2. 假设 $\hat{\boldsymbol{\mu}}: \mathbb{R}^n \to \mathbb{R}^n$ 是几乎处处可微的，且令

$$\nabla \cdot \hat{\boldsymbol{\mu}} = \sum_{i=1}^{n} \frac{\partial \hat{\mu}_i}{\partial y_i}.$$

如果 $\boldsymbol{Y} \sim N_n(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_n)$，则有

$$\sum_{i=1}^{n} \mathrm{Cov}(\hat{\mu}_i, y_i)/\sigma^2 = \mathbb{E}[\nabla \cdot \hat{\boldsymbol{\mu}}].$$

假设一个同方差模型：$\boldsymbol{Y} \sim N(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_n)$. 对任何估计：$\hat{\boldsymbol{\mu}} = \delta(\boldsymbol{Y})$，引理(1.2) 定义了 $\hat{\boldsymbol{\mu}}$ 的自由度：

$$df(\hat{\boldsymbol{\mu}}) = \sum_{i=1}^{n} \mathrm{Cov}(\hat{\mu}_i, y_i)/\sigma^2. \tag{11}$$

自由度表示模型的复杂度，即回归系数估计的非零系数的个数. 对给定的调节参数 λ，令 $\hat{\boldsymbol{\beta}}(\lambda)$ 表示回归系数向量 $\boldsymbol{\beta}$ 的 Lasso 估计，则 Lasso 的自由度定义为：

$$df(\lambda) = \#\{j : \hat{\beta}_j(\lambda) \neq 0\}, \tag{12}$$

其中 # 表示集合中元素的个数. 可知 $df(\lambda) = \mathbb{E}[\hat{df}(\lambda)]$，则 $df(\lambda)$ 是 Lasso 自由度的无偏估计.

有了自由度的定义，在实际应用中，可以使用 GCV 方法和 BIC 方法选取调节参数 λ.

GCV 方法 可以极小化下面的 GCV 准则选择调节参数 λ，即

$$\hat{\lambda}_{\mathrm{gcv}} = \arg\min_{\lambda} \mathrm{GCV}(\lambda) = \arg\min_{\lambda} \frac{\|\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}_\lambda\|_2^2}{n(1 - \hat{df}(\lambda)/n)^2},$$

其中 $\hat{\boldsymbol{\beta}}_\lambda$ 表示给定 λ 时极小化 L_1 惩罚最小二乘目标函数 (9)的 Lasso 估计，且 $\hat{df}(\lambda)$ 表示给定 λ 的自由度.

BIC 方法 可以通过极小化下面的 BIC 准则进行选择调节参数 λ，即

$$\mathrm{BIC}(\lambda) = \log \hat{\sigma}_\lambda^2 + \frac{\hat{df}(\lambda)\log(n)}{n},$$

其中 $\hat{\sigma}_\lambda^2$ 是基于任一 λ 所得模型的残差平方和除以 n. 极小化上面的 BIC 准则目标函数 $\mathrm{BIC}(\lambda)$，可以选择最优的调节参数 λ，记为 $\hat{\lambda}_{\mathrm{bic}}$.

1.3.2 代码实践

例 1.3. 用程序包 *glmnet* 中的函数 *glmnet()* 对波士顿房价数据进行 *Lasso* 回归分析.

Lasso 回归只需将岭回归中 `glmnet()` 函数参数 `alpha` 设置为 1.

```
fit_lasso <- glmnet(x, y, alpha = 1, nlambda = 100)
lasso_tidy <- tidy(fit_lasso) %>%
filter(term != "(Intercept)") %>%
mutate(log_lambda = log(lambda))
lasso_plot <- ggplot(lasso_tidy, aes(x = log_lambda, y =
    estimate, color = term)) +
geom_line(linewidth = 1) +
labs(
title = "lasso Regression Coefficient Paths",
x = "log(lambda)", y = "Coefficient"
) +
theme_minimal() +
theme(legend.position = "right") +
guides(color = guide_legend(title = "Variables"))
```

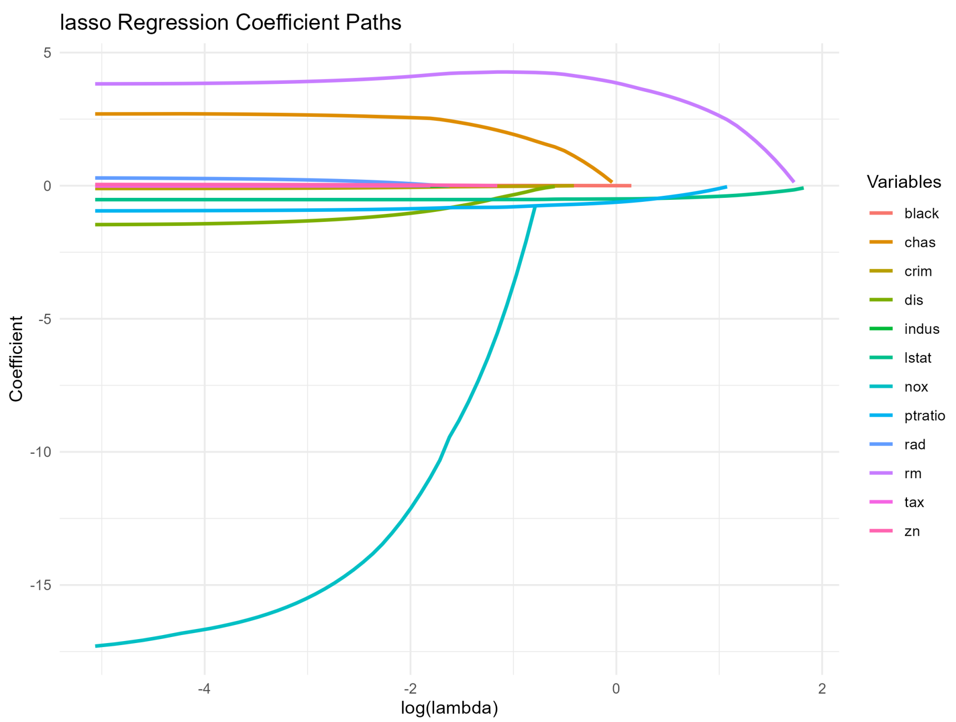

图 3: Lasso 回归估计随着 λ 变化的路径图.

从图(3)中可以看出，λ 足够大时，Lasso 回归估计得到一个零模型，所有回归系数

的 Lasso 估计均为 0. 因此，根据不同 λ 的取值，可以得到包含不同变量的模型，说明了 Lasso 方法筛选变量的功能.

同样，我们可以绘制交叉验证误差图，见图(4).

```
set.seed(2025)
cv.lasso <- cv.glmnet(x, y, alpha = 1)
plot(cv.lasso, xlab = expression(log(lambda)))
```

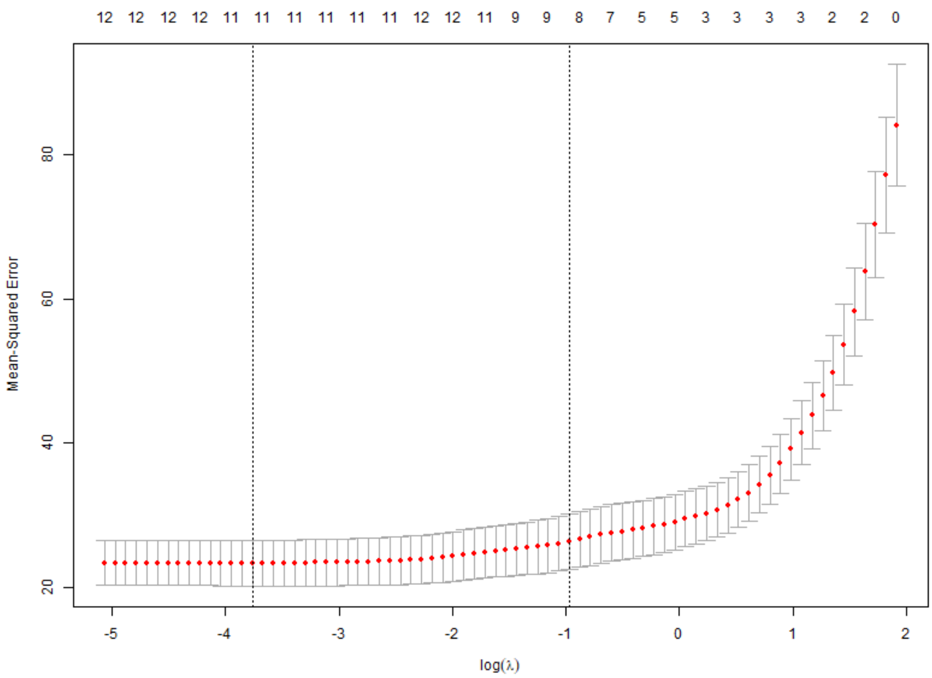

图 4: Lasso 回归的交叉验证误差图.

从(4)中可见使得 $\mathrm{CV}(\hat{\lambda})$ 最小化的 λ 为 $\hat{\lambda} = 0.02325053$, 而利用“一个标准差”准则选取的 λ 为 $\tilde{\lambda} = 0.3789258$. 我们可以用函数 `coef()` 分别提取 $\hat{\lambda}$ 和 $\tilde{\lambda}$ 对应的回归系数的 Lasso 估计.

```
cv.lasso$lambda.min
cv.lasso$lambda.1se
coef(cv.lasso, s = "lambda.min")
coef(cv.lasso, s = "lambda.1se")
```

从结果看来，部分回归系数为 0，即得到了稀疏解，使得模型更为简单，不易导致过拟合.

1.4 SCAD 回归

1.4.1 模型建立

先前提到的惩罚函数不能同时满足无偏性、稀疏性和连续性。为了解决这个问题，Fan (1997) 提出了一个连续可微的惩罚函数，称为 SCAD 惩罚函数：

$$p'_\lambda(|\theta|) = \lambda\left\{I(|\theta| \le \lambda) + \frac{(a\lambda - |\theta|)_+}{(a-1)\lambda} I(|\theta| > \lambda)\right\}, \tag{13}$$

针对多元线性模型(2)，考虑下面的 SCAD 惩罚最小二乘目标函数：

$$Q(\boldsymbol{\beta}) = \frac{1}{2n}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \sum_{j=1}^{p} p_\lambda(|\beta_j|), \tag{14}$$

其中 $p_\lambda(\cdot)$ 是由式(13)定义的 SCAD 惩罚函数，λ 是调节参数或截断参数. 极小化 $Q(\boldsymbol{\beta})$ 可得回归系数向量的一个 SCAD 估计 $\hat{\boldsymbol{\beta}}$. 为求解，可利用 LQA 算法.

1.4.2 代码实践

例 1.4. *用程序包 ncverge 中的函数 ncvreg() 对波士顿房价数据进行 SCAD 回归分析.*

进行 SCAD 回归.

```
library(MASS)
library(ncvreg)
library(ggplot2)
library(reshape2)
data("Boston")
x <- as.matrix(Boston[, -14])
y <- Boston$medv
fit_SCAD <- ncvreg(x, y, family = "gaussian", penalty = "SCAD",
    nlambda = 100)
```

可以绘制 SCAD 估计随着 λ 变化的路径图以及交叉验证误差图.

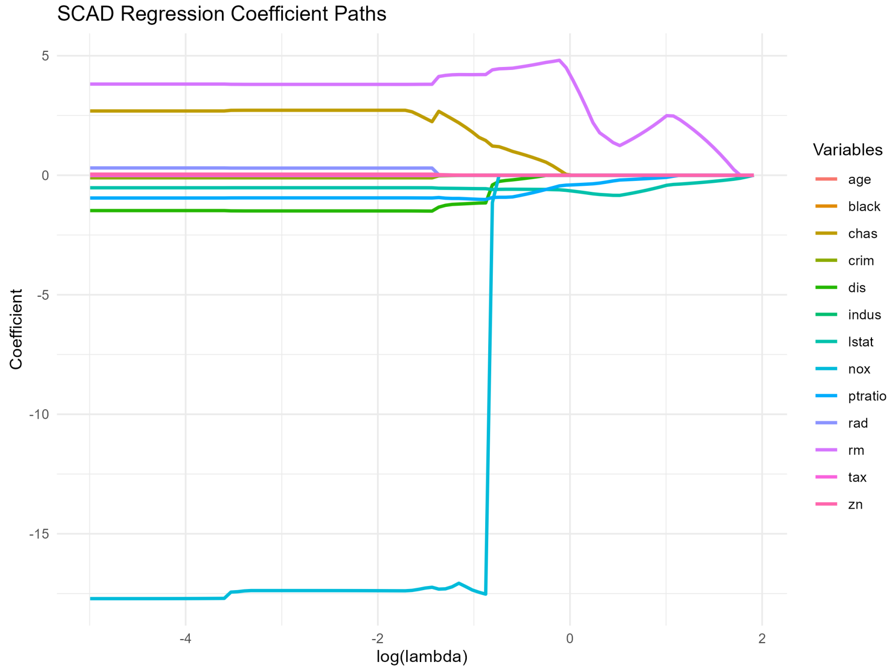

图 5: SCAD 回归估计随着 λ 变化的路径图.

图(5)显示了 SCAD 估计的路径图. 类似于 Lasso 方法，随着 λ 的取值变化，可以得到包含不同变量的模型，具有筛选变量的作用.

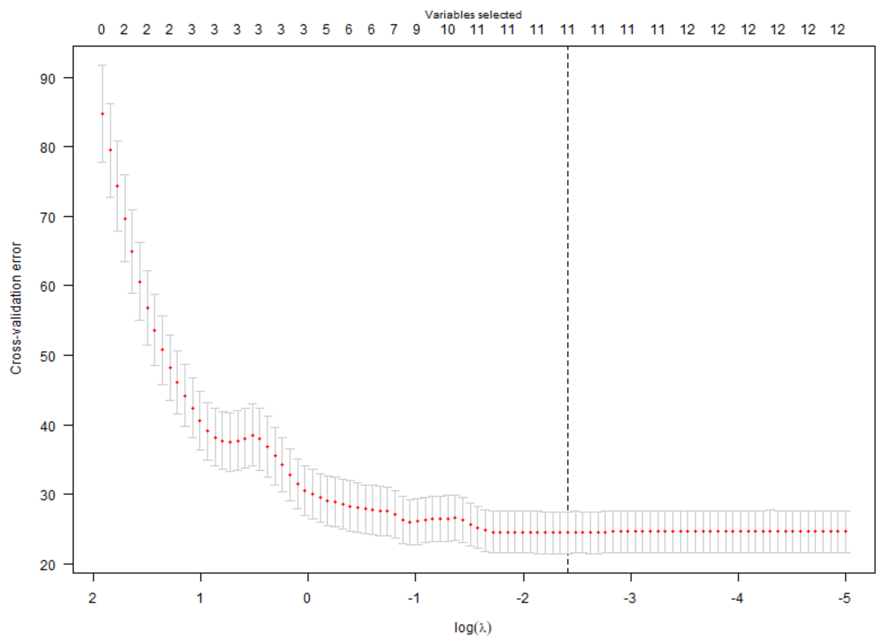

图 6: SCAD 回归的交叉验证误差图.

图(6)展示了交叉验证误差图.

1.5 自适应 Lasso

1.5.1 模型构建

我们先前得到了 Lasso 估计：

$$\hat{\boldsymbol{\beta}}^L = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2n} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right\}, \tag{15}$$

其中 $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^p |\beta_j|$。令 $\mathcal{A} = \{j : \beta_{0j} \neq 0\}$ 表示活动模型，$\hat{\mathcal{A}}_n = \{j : \hat{\beta}_j^L \neq 0\}$ 表示选择模型。变量选择的目的就是希望能正确识别正确的模型，即理论上需要证明下面变量选择相合性，即

$$\lim_{n\to\infty} \Pr(\hat{A}_n = A) = 1.$$

而 Lasso 估计不满足，因此引出自适应 Lasso 估计. 令 $\hat{\boldsymbol{\beta}}$ 是 $\boldsymbol{\beta}$ 的一个 $\sqrt{n}$-相合估计，如最小二乘估计. 定义权向量：$\hat{\boldsymbol{w}} = 1/|\hat{\boldsymbol{\beta}}|^\gamma$，$\gamma > 0$。自适应 Lasso 估计 $\hat{\boldsymbol{\beta}}^{\text{alasso}}$ 定义为：

$$\hat{\boldsymbol{\beta}}^{\text{alasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2n} \|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^p \hat{w}_j |\beta_j| \right\}. \tag{16}$$

从自适应 Lasso 估计式 (16) 可知，自适应 Lasso 的基本思想是，对于最小二乘估计大的回归系数，不进行惩罚，而对于接近于 0 的回归系数给予尽量大的惩罚，并压缩到 0. 如果选取合适的 γ，自适应 Lasso 的解将满足无偏性、稀疏性和连续性. 求解自适应 Lasso 可以利用 LARS 算法.

1.5.2 代码实践

例 1.5. *用程序包 msgps() 对波士顿房价数据进行自适应 Lasso 回归分析.*

可以直接通过msgps()进行自适应 Lasso 分析：

```r
library(msgps)
library(MASS)

# 加载 Boston 数据
data("Boston")
x <- as.matrix(Boston[, -14])
y <- Boston$medv

# 自适应Lasso回归
alasso_fit <- msgps(x, y, penalty = "alasso", gamma = 1, lambda
    = 0)
```

```
# 绘图
png("alasso_plot.png", width = 800, height = 400)
par(mfrow = c(1, 2))
plot(alasso_fit , criterion = "gcv", xvar = "t", main = "GCV")
plot(alasso_fit , criterion = "bic", xvar = "t", main = "BIC")
dev.off()
```

图(7)展示了分别采用 GCV 准则和 BIC 准则选取的调节参数.

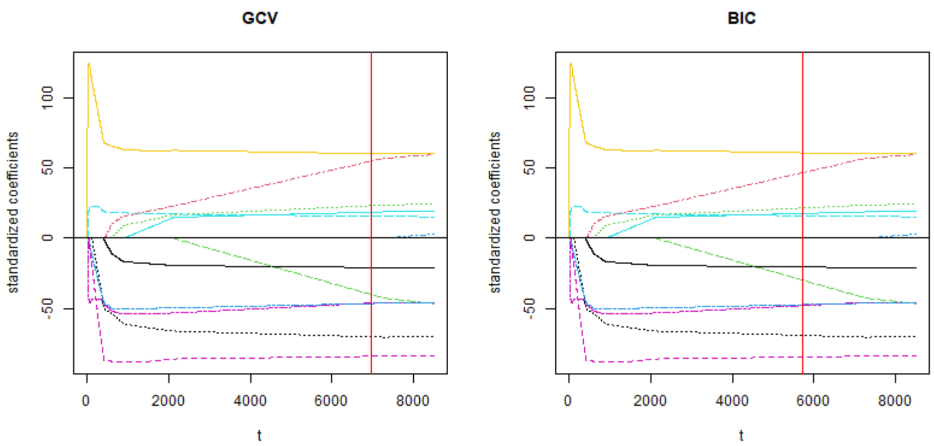

图 7: SCAD 回归估计随着 λ 变化的路径图.

使用 `summary()` 可以对输出结果进行汇总，结果见附录.

参考文献

[1] 李高荣, 吴密霞编著. 多元统计分析 [M]. 北京: 科学出版社, 2021.

A 回归系数表

表 1: Ridge 回归在 $\hat{\lambda}$ 与 $\tilde{\lambda}$ 下的非零系数

(a) $\hat{\lambda}$

	Coef	Term
(Intercept)	28.0015	(Intercept)
crim	-0.0876	crim
zn	0.0327	zn
indus	-0.0380	indus
chas	2.8998	chas
nox	-11.9134	nox
rm	4.0113	rm
age	-0.0037	age
dis	-1.1189	dis
rad	0.1537	rad
tax	-0.0058	tax
ptratio	-0.8550	ptratio
black	0.0091	black
lstat	-0.4724	lstat

(b) $\tilde{\lambda}$

	Coef	Term
(Intercept)	20.8162	(Intercept)
crim	-0.0679	crim
zn	0.0204	zn
indus	-0.0687	indus
chas	2.7637	chas
nox	-5.4131	nox
rm	3.6171	rm
age	-0.0077	age
dis	-0.5190	dis
rad	0.0327	rad
tax	-0.0029	tax
ptratio	-0.6703	ptratio
black	0.0076	black
lstat	-0.3479	lstat

表 2: Lasso 回归在 $\hat{\lambda}$ 与 $\tilde{\lambda}$ 下的非零系数

(a) $\hat{\lambda}$

	Coef	Term
(Intercept)	34.7411	(Intercept)
crim	-0.1000	crim
zn	0.0422	zn
indus	0.0000	indus
chas	2.6914	chas
nox	-16.4841	nox
rm	3.8564	rm
age	0.0000	age
dis	-1.4121	dis
rad	0.2603	rad
tax	-0.0101	tax
ptratio	-0.9327	ptratio
black	0.0091	black
lstat	-0.5225	lstat

(b) $\tilde{\lambda}$

	Coef	Term
(Intercept)	17.4715	(Intercept)
crim	-0.0218	crim
zn	0.0000	zn
indus	0.0000	indus
chas	1.8977	chas
nox	-3.3413	nox
rm	4.2663	rm
age	0.0000	age
dis	-0.3171	dis
rad	0.0000	rad
tax	0.0000	tax
ptratio	-0.7873	ptratio
black	0.0066	black
lstat	-0.5181	lstat

Listing 1: 自适应 Lasso 回归的结果

```
Call: msgps(X = x, y = y, penalty = "alasso", gamma = 1,
    lambda = 0)

Penalty: "alasso"

gamma: 1

lambda: 0

df:
tuning      df
[1,]   0.0000  0.0000
[2,]   0.3814  0.2484
[3,]   0.7630  0.4969
[4,]   1.1444  0.7453
[5,]   4.0198  1.2183
[6,]   7.4965  2.1132
[7,]   9.2763  2.7975
[8,]   9.4515  3.3418
[9,]  10.0666  3.7518
[10,] 10.6814  4.0942
[11,] 11.2660  4.6070
[12,] 11.8551  4.9074
[13,] 12.7180  5.4213
[14,] 14.0164  6.6353
[15,] 15.5274  7.6835
[16,] 16.6140  8.7381
[17,] 17.2934  9.7463
[18,] 17.7604 10.2261
[19,] 18.2293 10.6565
[20,] 23.6064 12.7498

tuning.max: 23.61

ms.coef:
Cp      AICC      GCV      BIC
(Intercept)  36.286119  36.286119  36.286119  36.195154
```

```
crim        -0.107663   -0.107663   -0.107663   -0.105748
zn           0.044418    0.044418    0.044418    0.041215
indus        0.000000    0.000000    0.000000    0.000000
chas         2.750745    2.750745    2.750745    2.819947
nox        -17.690035  -17.690035  -17.690035  -18.410529
rm           3.819632    3.819632    3.819632    3.858720
age          0.000000    0.000000    0.000000    0.000000
dis         -1.482663   -1.482663   -1.482663   -1.461273
rad          0.281093    0.281093    0.281093    0.239207
tax         -0.010572   -0.010572   -0.010572   -0.007861
ptratio     -0.953955   -0.953955   -0.953955   -0.971207
black        0.009085    0.009085    0.009085    0.008628
lstat       -0.523807   -0.523807   -0.523807   -0.526730

ms.tuning:
Cp    AICC    GCV    BIC
[1,]  18.44  18.44  18.44  18.11

ms.df:
Cp    AICC    GCV    BIC
[1,]  10.85  10.85  10.85  10.54
```