# 第3章 回归分析[①]

**本章学习目标**

- 理解回归分析的基本逻辑
- 理解岭回归和拉索回归函数形式
- 了解岭回归、拉索回归和弹力网等算法的应用场景、原理及优势
- 掌握相关算法在 Python 中的分析

## 3.1 算法基础

在一般意义上，回归分析(Regression Analysis) 作为一种统计分析方法，旨在获取两个或两个以上变量间相互依赖的定量关系。在机器学习中，回归(Regression)特指一类算法，这类算法一般被用来预测与对象关联的连续型变量取值。在统计学发展初期，“回归”由英国统计学家高尔顿提出，描述子代身高向族群平均身高靠拢的现象。在当下主流的社科定量研究方法中，“回归”泛指一类剥离并观察自变量与因变量之间净效应的技术。鉴于回归概念本身的复杂性，需要澄清的是，本章仅从机器学习角度理解回归，并将介绍其中主要的几种线性回归(Linear Regression)算法，包括最小二乘法(Ordinary Least Squares，OLS)，以及改进了最小二乘法的岭回归(Ridge Regression)、拉索回归(LASSO)和弹性网(Elastic Net)。

与本书其他章节介绍的“酷炫”技巧相比，线性回归可能会显得有些平淡无奇，但了解它仍有相当的必要性。从应用角度看，线性回归的适用范围广泛，且与其他复杂算法相比，实际效果甚至相差无几。例如，著名的计算社会科学家萨尔加尼克等人[②]曾召集 160 个顶尖

① 感谢范晓光对本章的完成给予的大力帮助，同时感谢毕向阳通读初稿并提出建设性意见.

② Salganik M J. et al. Measuring the Predictability of Life Outcomes with a Scientific Mass Collaboration. Proceedings of the National Academy of Sciences，2020，117(15)：8398-8403.

团队，让他们各自利用机器学习对某一社会变量进行预测。结果发现，表现最好的算法尽管非常复杂且利用了数千个预测变量，但预测精度也就比使用四个预测变量的线性回归高一点点。而从学习路径看，线性回归不仅与社会科学家们熟悉的传统方法思路最为相近，且以岭回归为代表的正则化(regularization)方法也最能体现机器学习与传统计量在功能取向上的不同，以及机器学习方法在预测方面的优势[①]。

### 3.1.1 线性回归的基本原理

有一连续型变量 $Y$，以及 $p$ 个可以用来预测的变量 $X=(X_1,X_2,\cdots,X_p)$。通常我们会建构方程 $Y=f(X)+\epsilon$，其中，$f$ 是未知的关于 $X$ 的方程，$\epsilon$ 是随机误差项。研究者需要做的，是根据现有数据 $Y$ 和 $X$ 得到 $\hat{f}$，即对 $f$ 的形式进行估计。估计 $f$ 有两类目的。传统定量研究的目的主要是推断，即探究部分自变量对因变量的净效应。机器学习更关心预测，即给定 $X$，要用 $\hat{Y}=\hat{f}(X)$来预测那些难以获取的 $Y$。我们不再对 $\hat{f}$ 本身感兴趣，而是将其视为一个黑箱。本章涉及的线性回归旨在找到一个具有较好预测效力的 $\hat{f}(X)$。其中，“线性”指假设方程 $f(X)$的形式为预测变量的线性组合：

$$f(X)=\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_pX_p$$

其中，$\beta_1,\cdots,\beta_p$ 被称为权重(weight)或系数，$\beta_0$ 被称为偏置(bias)或截距。在线性方程中，估计 $f$ 的问题，就转换为估计 $p+1$ 个参数 $\beta_0,\beta_1,\cdots,\beta_p$ 的问题。进一步说，就是通过确定这组参数，来尽可能让 $Y\approx\hat{f}(X)$，确保提出的方程能拟合当前的数据。那么，当数据体现的模式在其他对象中同样存在时，就可以依据此方程进行预测。问题是，该如何确定一组 $\beta$ 呢？换句话说，如何才能让 $Y\approx\hat{f}(X)$呢？这需要用到最小二乘法。最小二乘法试图找到一组 $\beta$ 来让模型产生的残差平方和(Residual Sum of Squares，RSS)最小。RSS 被定义为

$$\begin{aligned}\mathrm{RSS}&=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2\\&=\sum_{i=1}^{n}(y_i-\hat{\beta}_0-\hat{\beta}_1x_{i1}-\hat{\beta}_2x_{i2}-\cdots-\hat{\beta}_px_{ip})^2\end{aligned}\tag{3.1}$$

其中，$y_i$ 为第 $i$ 个观测对象目标变量的真实值，$\hat{y}_i$ 为模型对第 $i$ 个观测对象的预测值。为了表达上的简洁，可以将上述表述转化为矩阵的形式。设 $\boldsymbol{X}$ 为一个 $n\times(p+1)$的矩阵，其中 $n$ 为观测对象数，$p$ 为特征数。设$\boldsymbol{\beta}$ 为一个 $p+1$ 维的向量，有$\boldsymbol{\beta}=(\beta_0,\beta_1,\cdots,\beta_p)^{\mathrm{T}}$。设 $\boldsymbol{y}$ 为目标变量真实值所组成的 $n$ 维向量，那么有：

$$\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ \|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2\tag{3.2}$$

针对正规方程求解，可以直接得到解析解$\hat{\boldsymbol{\beta}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$。当特征数 $p$ 比较大时(例如上万个特征)，求逆过程的运算量也会特别大，得出结果会比较慢，这时梯度下降(Gradient Descent)求解法可能是更好的选择。在实操中，计算机根据指定算法去求解最优$\boldsymbol{\beta}$ 的过程被称为训练。训练完成后，就得到可以用来执行预测任务的模型。向模型输入新的观测对象，模型会给出对新数据的预测。

① 陈云松，吴晓刚，胡安宁，等. 社会预测：基于机器学习的研究新范式. 社会学研究，2020，(3)：94-117.

需要注意的是，训练所得模型中的 $\hat{\beta}_1,\cdots,\hat{\beta}_p$，不能直接解读为现实中各预测变量对目标变量的(平均)效应，原因主要有三点。一是从理论角度看，机器学习模型的主要目的是预测，而非推断，因此变量设置没有经过严格的理论论证。二是从统计学角度看，由于无法获得无穷多的样本，权重估计必然存在不确定性，传统计量中需要计算置信区间来将这部分不确定性纳入考量。最后，模型设置本身也具有不确定性，不同预测变量的组合，实际上会导致权重的估计值发生较大改变，甚至正负号都会发生变化。①

### 3.1.2 线性回归模型的评估指标

评估得到的模型 $\hat{f}(X)$，需要测量它在预测方面的准确性。常用的指标之一为均方误差(Mean Squared Error，MSE)。具体来说，那些用来训练模型的观测对象组成的集合被称为训练集(training set)，而我们需要找到一些训练集之外的观测对象$(x_0,y_0)$，构成测试集(test set)。测试集上的均方误差为(Ave 表示取均值)：

$$\mathrm{MSE}=\mathrm{Ave}(y_0-\hat{f}(x_0))^2 \tag{3.3}$$

模型在测试集上的误差为测试误差(Test Error)。该误差被视为模型在全新样本中误差(泛化误差，Generalization Error)的近似。泛化误差越低，说明模型的准确性越高。除 MSE 外，线性回归中另一常用指标是决定系数(Coefficient of Determination)，也叫 $R$ 方，之后统一表示为 $R^2$。$R^2$ 的定义为：

$$R^2=1-\frac{\sum_i(\hat{y}_i-y_i)^2}{\sum_i(y_i-\bar{y})^2} \tag{3.4}$$

其中，$\sum_i(y_i-\bar{y})^2$ 为数据集中目标变量自身固有的变异性，或者说所携带的信息量。$\sum_i(\hat{y}_i-y_i)^2$ 是模型所能预测，或者说解释了的变异性。$R^2$ 的取值为负无穷到 1。② $R^2$ 取值越大，说明模型的泛化能力越好。0 是一个重要的分界线。当 $R^2\leqslant 0$，说明与均值模型相比，模型表现地一样甚至更糟，所以模型没有任何价值。

### 3.1.3 方差与偏差的权衡

在机器学习领域中，方差与偏差的权衡(bias-variance trade-off)是一个十分重要的概念。本节将对此概念进行简要解释。一般来说，使用模型对新对象$(x_0,y_0)$进行预测时，泛化误差可以被分解为三个部分：

$$E[(y_0-\hat{f}(x_0))^2]=\mathrm{Var}(\hat{f}(x_0))+[\mathrm{Bias}(\hat{f}(x_0))]^2+\mathrm{Var}(\epsilon) \tag{3.5}$$

其中，$\mathrm{Var}(\hat{f}(x_0))=E[(E[\hat{f}(\boldsymbol{x}_0)]-\hat{f}(\boldsymbol{x}_0))^2]$，被称为 $\hat{f}(x_0)$ 的方差(variance)，$[\mathrm{Bias}(\hat{f}(x_0))]^2=E[(f(\boldsymbol{x}_0)-E[\hat{f}(\boldsymbol{x}_0)])^2]$，被称为 $\hat{f}(x_0)$ 的偏差(bias)，$\mathrm{Var}(\varepsilon)=E[(y_0-f(\boldsymbol{x}_0))^2]$，是随机扰动项方差(Irreducible Error)。$\mathrm{Var}(\epsilon)$是无法控制与调整的，因此，提高预测精准度，或者说减少泛化误差，需要从削减方差和误差入手。

---

① Young C，Chen X. Patients as Consumers in the Market for Medicine：The Halo Effect of Hospitality. Social Forces，2020，99(2)：504-531.

② 一般来说，$R^2$ 小于 0 的情况很少见.

不难看出，$\hat{f}$ 的方差是指当使用不同训练集来训练时，$\hat{f}$ 的变化程度。理想情况下，估计出的 $\hat{f}$ 不会因为使用不同训练集而剧烈变化。$\hat{f}$ 的偏差是指因为用一个简单的模型去逼近一个复杂现实生活问题(Real-life Problem)而引入的误差(Error)。例如，线性回归假设 $\hat{Y}$ 是 $X$ 的简单线性组合，但事实上的关系可能是非线性的。

例如，图 3.1 左侧的散点图显示，数据呈现出非常明显的非线性模式。右侧我们训练出三个不同的模型来对数据模式进行拟合。“长短虚线交替”为基于普通最小二乘的线性回归。“虚线”为三次多项式回归，“实线”为高次多项式回归。当训练集发生变化时，可以想见，训练得到的高次多项式模型会发生很大变化，而三次多项式和线性回归则变化不会很大，换句话说，二者的方差都比较低。但是，线性回归没有能够充分反映数据所展现的模式，其对应偏差会比较大。因此案例中只有三次多项式回归做到了方差与偏差的权衡。

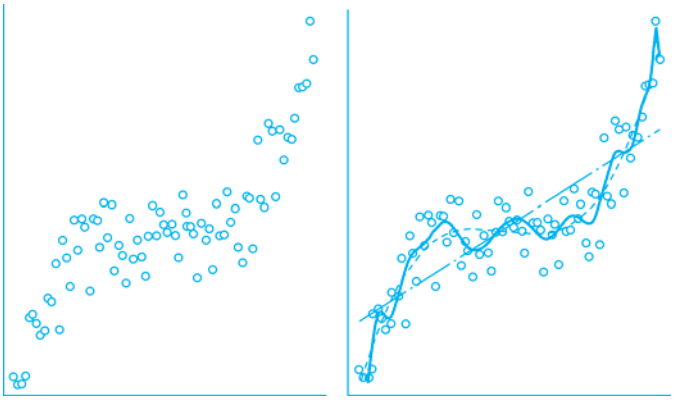

图 3.1　三个不同模型对数据的拟合

### 3.1.4　过拟合与欠拟合

#### 1. 概念界定

过拟合与欠拟合两个概念与上节所介绍的方差和偏差概念密切相关。

当模型设定过于复杂时，模型很可能对训练集拟合得“太好”。这意味着它将样本的一些随机误差都拟合进去了。由于过多方差的存在，预测结果会很不准确，这被称为过拟合(Overfitting)。当模型因太过“简单”时，它很可能并未充分挖掘样本所能提供的有关数据模式的信息，由于过多偏差的存在，其预测效力同样很差，这被称为欠拟合(Underfitting)。理想情况下，我们在方差与偏差之间找到一个平衡，避免过拟合和欠拟合问题的发生，提升模型的泛化能力。

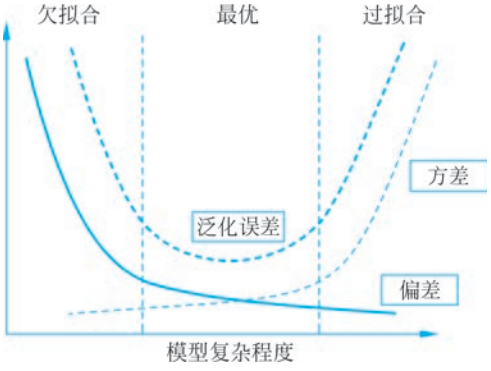


图 3.2　方差和偏差的平衡图示

随着模型复杂度的提升(例如预测变量和参数的增多)，偏差会不断下降，但方差会逐渐上升，模型由欠拟合状态逐渐走向过拟合状态(见图 3.2)。这意味着存在一个最优的模型复杂度设置，可以规避欠拟合和过拟合问题，这就是研究者旨在寻找的最优设置。一般来说，研究者在建模过程中总倾向于尽量解释训练

集中数据的变异性。因此，有大量不重要甚至无关的特征会被添加到线性回归模型中，所以过拟合通常比欠拟合更容易发生。本章将介绍的岭回归和拉索回归，相较于普通最小二乘算法，其原理就是通过引入少量偏差来减少方差，降低预测时的误差，一定程度上避免过拟合，从而提升模型的预测效果，或者说泛化能力。这反映了机器学习与传统定量方法的差异。

2. 判断是否出现过拟合

那么，研究者该如何判断模型有没有过拟合呢？最简单的方法是找到训练集之外的数据进行预测，计算 MSE 或 $R^2$ 等评估指标。如果指标表明模型在训练集上的表现明显好于测试集，那么就有理由认为建模过程导致了过拟合。一般来说，研究者会将手头的数据分为训练集和测试集。训练集用来进行模型的训练，测试集用来评估建模过程的合理性。这种方法被称为交叉验证。交叉验证的方法有很多，这里介绍“留出法”(Hold-out)和“K 折法”(K-fold)两种。

留出法指将数据按指定比例划分为两个互斥的集合，分别作为训练集和测试集。研究者在训练集上得到模型后，在测试集评估其表现，将计算得到的指标作为模型泛化误差的近似。需要注意的是，给定一个样本，理论上有很多种方法对样本进行分割。每一种分割方式都会产生一个评估量。因此，单次的留出法的结果可能并不可靠。

K 折法指将数据分为 $k$ 个互斥且大小相同的子集。用 $k$-1 个子集作为训练集，剩下的一个子集作为测试集，计算相应的指标。循环这一过程 $k$ 次，直至每一个子集都做过一次测试集，将求得的 $k$ 个指标求平均值，作为对模型泛化误差的近似。图 3.3 展示了 5 折法交叉验证的过程。当然，有时也会计算 $R^2$ 或其他指标并最终取平均。

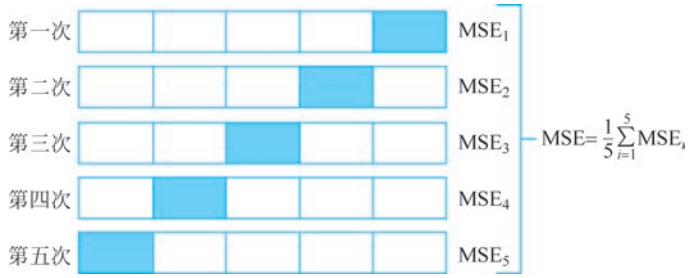


图 3.3　5 折法交叉验证图示

K 折法存在一种极端情况，那就是每次只将 1 个观测对象作为测试集，其他观测对象作为训练集。循环执行的次数等于样本量，直至每一观测对象都作为过测试集。这种方法也被称为留一验证(Leave-one-out)。当样本量比较少的时候，K 折法获得的训练集和测试集都会比较小。一方面较少的测试集测不准确，另一方面较少训练集意味着训练过程对异常值、标签错误比较敏感，方差很大。此时对泛化误差的估计会很不准确。由于留一法每次训练都能充分利用数据，相比之下模型方差要小得多，不会倾向高估泛化误差。另外，针对特定数据集，留一验证的方式有且只有一种，因而有时无须像 K 折那样重复多次。留一法的缺陷(在大多数时候)是运算量大，需要消耗较多算力。

3. 过拟合的处理简述

如之前所述，模型泛化误差的期望，可以被分解为三个成分：方差、偏差和随机误差项的方差：

$$E[y_0-\hat{f}(x_0)]^2=\underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{Variance}}+\underbrace{[\mathrm{Bias}(\hat{f}(x_0))]^2}_{\text{Bias}^2}+\underbrace{\mathrm{Var}(\epsilon)}_{\text{Irreducible Error}} \tag{3.6}$$

设训练集有 $n$ 个观测对象，$p$ 个观测变量。如果线性假设成立，基于 OLS 训练线性回归模型时，可能存在三种情形。

(1) 当 $n\gg p$，偏差小，方差一般也小，模型预测精度很可能不错。

(2) 当 $n>p$ 但是 $n\gg p$ 不成立时，误差小，但方差可能会很大，模型预测精度堪忧。

(3) 当 $p>n$，OLS 不存在唯一解，方差无穷大，模型根本无法使用。当发现线性回归模型存在过拟合时，一般是因为上述的(2)、(3)两种情形。

此时，有四类方法可以缓解过拟合问题。一是增加观测对象的个数，这相当于直接增加模型训练时所能获取的信息量。但鉴于训练模型前，研究者应该已经尽力搜集了所有能搜集到的信息，所以增加观测对象很困难。二是进行特征选择(Feature Selection)，也就是从 $p$ 个预测变量中选出我们相信对预测目标变量效果最好的一部分。特征选择的方法有很多，没有标准化的方法，且一般流程比较烦琐。三是进行特征提取，或者说降维。以 PCA (Principal Component Analysis)为代表，这类方法将 $p$ 个预测变量投影到 $M$ 维的空间内，且 $M<p$。然后用这新生成的 $M$ 个特征作为新的预测变量。一般来说，特征提取会不可避免的损失掉一些信息，最为关键的是，线性回归优势在于模型的可解释性，但提取的特征不再具有明确的现实意义，相当于让线性回归“自废武功”。四是本章将要介绍的正则化(Regularization)方法。正则化方法以岭回归和拉索回归为代表，它们会利用所有的预测变量，但相比最小二乘方法，整体上的系数会向 0 收缩，这起到了引入偏差但减少方差的作用，从而提升模型的泛化能力。另外，拉索回归可以使部分变量的系数收缩到 0，从而起到了特征选择的作用，帮助得到更加简单且易于解释的模型。

正则化方法不仅可以在线性回归中使用，在多项式回归、广义线性模型等建模过程和算法中也有使用。例如，Logistic 回归中也可以使用正则化方法来避免过拟合。

## 3.2 岭 回 归

### 3.2.1 对岭回归的理解

在机器学习中，经常通过使某一函数最小化来对模型参数进行训练，该函数被称为损失函数(Loss Function)。回忆一下，OLS 定义的损失函数为模型在训练集上残差的平方和：

$$\mathrm{RSS}=\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2 \tag{3.7}$$

上式表达成矩阵的形式为：$\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2$。正如前文所述，岭回归得到的同样是包含 $p$ 个预测变量的线性模型，但损失函数的形式略有不同：

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}\beta_j^2=\mathrm{RSS}+\lambda\sum_{j=1}^{p}\beta_j^2 \tag{3.8}$$

上式表达成矩阵的形式为：$\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2+\|\boldsymbol{\beta}^{p}\|_2^2$。[①] 可以看到，相比

① 偏置一般不参与正则化，所以这里用$\boldsymbol{\beta}^{p}$表示去掉偏置后的 $p$ 维向量.

OLS,岭回归损失函数增加了一项,该项为各权重平方求和后再乘以“超参数”(Hyperparameter)$\lambda$,且有$\lambda \geqslant 0$。$\lambda \sum_{j=1}^{p} \beta_j^2$ 整体称为收缩惩罚(Shrinkage Penalty),$\sum_{j=1}^{p} \beta_j^2$ 求平方根后被称为 L2 范数(L2 norm)。参数 $\lambda$ 越大,收缩惩罚就越大,损失函数也就越大。当$\lambda=0$,岭回归与 OLS 一样。当$\lambda>0$,与 OLS 相比,岭回归算法下的 $\beta_j$ 会呈现向 0 收缩的趋势。我们知道,对于线性回归而言,OLS 算法会找到对训练集拟合程度最佳的模型,但岭回归中惩罚项的出现会让其找到次佳的模型,这无疑引入了少量偏差,但由于权重的收缩效应,参数因不同训练集而发生的变化程度也被压缩,方差减少,从而提升了预测效力,或者说模型的泛化能力。

具体来说,岭回归主要克服的是 OLS 的线性回归中的多重共线(Multicollinearity)问题。多重共线问题包含两类情况——完全相关和高度相关,二者都是针对特征矩阵 $\boldsymbol{X}$ 而言的。完全相关是指 $\boldsymbol{X}$ 中各特征向量是线性相关的,例如一个特征和另一个特征成比例。高度相关是指各 $\boldsymbol{X}$ 中的特征间有高度的相关关系,表现为方差膨胀因子(Variance Inflation Factor,VIF)过高。当特征高度相关时,虽然 OLS 的估计仍然具有无偏性,但估计的标准误差很大。换句话说,不同训练集上拟合的模型可能会非常不同,使模型泛化误差中的方差成分很高。

进一步说,OLS 方法的解析解为$\hat{\boldsymbol{\beta}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$。当完全相关发生时,$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不可逆,一些统计软件甚至会直接报错。[①] 当高度相关发生时,$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 虽然可逆,但求逆的结果会相当不稳定,直接影响对$\hat{\boldsymbol{\beta}}$ 的求解。总的来看,面临多重共线时,OLS 的功能便受到极大限制。相比之下,岭回归的解析解为$\hat{\boldsymbol{\beta}}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}^{p})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$[②]。当完全相关发生时,虽然 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 不可逆,但 $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}+\lambda\boldsymbol{I}^{p}$ 在绝大多数情况下都是可逆,因而可以求解。当高度相关发生时,随着超参数 $\lambda$ 的增加,模型将越不容易受到多重共线的影响。

### 3.2.2 调参

需要注意的是,当选取不同 $\lambda$ 值时,岭回归都会输出一组不同的$\hat{\boldsymbol{\beta}}$,得到不同的模型,也就必然给出不同的预测。然而,$\hat{\boldsymbol{\beta}}$ 是基于算法在训练过程中得到的,但 $\lambda$ 却无法通过算法本身得到,却在很大程度上影响模型表现。所以$\boldsymbol{\beta}$ 被称为参数,而 $\lambda$ 被称为超参数。研究者通过种种方式来选择、判断超参数该取何值更好的过程,被称为调参(Parameter Tuning)。

一种比较知名的 $\lambda$ 确定方法来自岭回归的提出者 Hoerl 和 Kennard。[③] 他们发明了岭迹图(Ridge Trace)(见图 3.4)。图像的横轴为 $\lambda$,纵轴为对应的权重,即不同预测变量的权重被表现为 $\lambda$ 值的函数。一般来说,随着 $\lambda$ 的增大,权重先是变化很大,然后会趋于稳定,而研究者要做的,是选择权重平稳时的 $\lambda$,但要让 $\lambda$ 尽可能小,即在削减方差的同时尽可能少

① 此时,R 语言中的 lm 函数会自动删除精确共线的特征,返回的结果中对应权重显示为 NA,也就是缺失值。Python 语言 sklearn.linear_model 的 LinearRegression 虽然不会返回 NA,但会在多个解中选择一个返回,本质上仍是没有唯一解,此时的方差无穷大.

② 由于偏置不参与正则化,$\boldsymbol{I}^{p}$ 并不等于单位矩阵。区别在于 $\boldsymbol{I}^{p}$ 主对角线上第一个元素是 0,其他 $p$ 个元素则都跟单位矩阵一样是 1.

③ Hoerl A E,Kennard R W. Ridge Regression: Applications to Nonorthogonal Problems. Technometrics,1970,12(1):69-82.

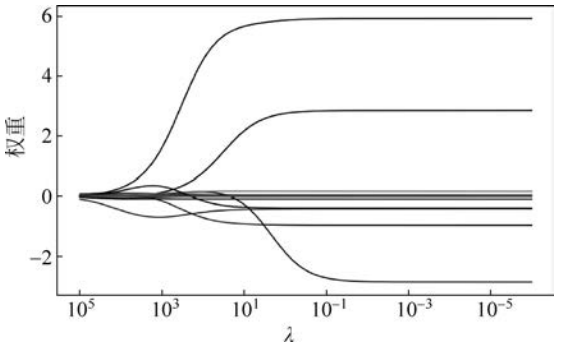


图 3.4 岭迹图

地引入偏差。

岭迹图本质上是一种定量与定性相结合的方法，判别标准不能具体的量化，有一定的主观色彩。在机器学习中，更常用的是通过交叉验证来确定 λ[①]，这种方法最大程度增加了找到一个好模型的可能性。之前提到过，交叉验证可以评估模型的泛化能力，其原理是将数据分为训练集和测试集，前者训练模型，后者评估模型表现。在调参过程中，交叉验证的流程基本一致，只不过它是将数据分为训练集和验证集(Validation Set)。例如，在岭回归中，先假设最佳的 λ 可能是 200 个 1～200 整数中的一个。然后用 200 个不同的 λ 值在训练集上训练 200 个对应模型，再用验证集评估这 200 个模型的表现，表现最好的模型所对应 λ 值，就是应该选择的参数。之后便可以使用该参数，在整个数据集中训练模型。

知晓如何调参后，另一个问题是，如何对包含调参的建模过程所产生模型的泛化能力进行评估？评估需要用到三个互斥集合：训练集、验证集和测试集。流程主要分为四步：(1)基于训练集训练出使用不同 λ 的 $K$ 个模型；(2)然后用验证集评估这 $K$ 个模型，提取表现最佳的模型所使用的 λ；(3)将训练集和验证集合并，用选出的 λ 进行模型训练；(4)用测试集评估模型的泛化能力。

### 3.2.3 案例 1：如何预测居民的幸福度

#### 1. 背景

假如你是某民生研究院的研究员，现在需要你基于一份综合调查数据，通过机器学习建模来预测居民的幸福度，幸福度的区间为 1～5，分数越高，幸福度越高。所用数据是 2005 年的“中国综合社会调查”(CGSS2005)。[②]

所涉变量包括居民的主观幸福感、职业阶层、教育水平、家庭人均月收入对数、对社会经济地位相对于过去高低的判断、对社会经济地位相对同龄人高低的判断、性别、民族、年龄、户籍、是否为党员、自评健康、婚姻状况和与亲朋联系的密切程度等。

#### 2. 代码与实操

我们使用 python 语言完成全部建模过程。实操第一步是导入必要的模块(module)：

① 前提是不考虑计算成本.

② 本数据是洪岩璧(2017)的文章所用资料。感谢 CGSS 调查项目组授权允许我们使用此数据.该数据集为官方发布数据的一部分，包含部分受访者和再编码后的部分变量，只作案例演示之用.获得完整数据请查看 CGSS 官网 http://cgss.ruc.edu.cn/.另外，严格来说幸福度在问卷中属于定序变量，这里为演示方便将其作为连续变量进行处理。参见：洪岩璧.再分配与幸福感阶层差异的变迁(2005—2013).社会，2017，37(2)：106-132.

```
import pandas as pd                    # 用于表格形式展示、处理和加工数据
import numpy as np                     # 处理数学运算
```

以上两库是使用 Python 从事数据科学时常用的，所以先行导入。对于其他模块，我们则到使用时再导入，以便更为清晰地讲解和呈现代码。

下面读取数据：

```
X = pd.read_csv("data/data105025/CGSS2005_predictors.csv")
y = pd.read_csv("data/data105025/CGSS2005_target .csv")
```

除可以使用 data.head()查看前 5 行信息外，还可以通过 data.info()查看各变量的名称、缺失值个数以及类型等信息；通过 data.describe()查看各变量的描述统计信息，包括均值、四分位数等；通过 data.hist()绘制各变量的直方图。对数据基本情况进行了解和描述的方法有很多，这里不一一介绍了。

正如之前提到了，我们需要将手头的数据分为训练集和测试集。这要用到 sklearn 模块中的 train_test_split 函数。向其传入特征矩阵和目标变量，然后设定测试集占总数据量的比例为 30%。

```
#  使用 sklearn 中的 train_test_split 函数拆分训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = .3, random_state = 728)
```

拆分完成后，就可以在训练集上进行岭回归的训练了：

```
from sklearn.linear_model import Ridge
ridge_temp = Ridge(alpha = 20)                    # 设定 lambda = 20
ridge_temp.fit(X_train, y_train)                  # 训练模型
```

模型训练完成后，可以通过 ridge_temp.coef_来查看各预测变量对应的权重，通过 ridge_temp.intercept_来查看偏置。但正如之前所述，人为设定 λ 值不仅没有根据，而且很不可靠，所以应该使用交叉验证的方法。下面我们基于训练集，使用 5 折交叉验证，来计算不同 λ 所得岭回归模型的表现(用 MSE 衡量)：

```
# 绘制所有 lambda 对应的 mse(仅针对训练集)
#导入模块
from sklearn.linear_model import Ridge
from sklearn.model_selection import cross_val_score, KFold
# 待搜索的参数 lambda
ridgeAlphas = np.logspace(-5, 2, 100)
# 定义 K 折方法(5 折交叉验证)
fold5 = KFold(n_splits = 5, random_state = 728, shuffle = True)
# 遍历 lambda 值
ridges = []
for alpha in ridgeAlphas:
    ridge_ = Ridge(alpha = alpha)
    ridgeScore = -1 * cross_val_score(ridge_, X_train, y_train, cv = fold5, scoring =
    "neg_mean_squared_error").mean()
    ridges.append(ridgeScore)
# 绘图
import matplotlib.pyplot as plt
plt.plot(ridgeAlphas, ridges, c = "black", label = "Ridge")
plt.legend()
```

图 3.5 表明交叉验证显示最佳的 λ 值为 20～40。在实际操作中，我们不必执行上面繁复的操作去调参，sklearn 模块已经写好了自带交叉验证调参的 RidgeCV。

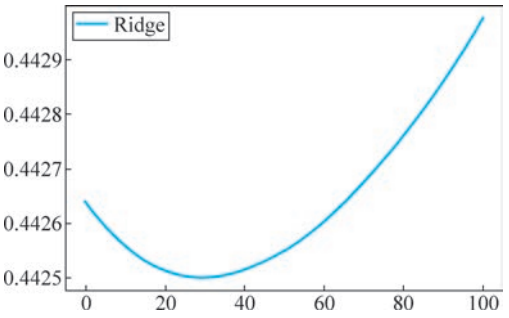


图 3.5 不同 λ 所对应岭回归模型的 MSE 表现

```
#定义一个自带 cv 调试 lambda 功能的岭回归算法
from sklearn.linear_model import RidgeCV
# 定义算法
ridgeReg = RidgeCV(alphas = ridgeAlphas,
cv = fold5, scoring = "neg_mean_squared_error")
# 训练
ridgeReg.fit(X_train, y_train)
# 查看找到的 alpha
ridgeReg.alpha_
```

ridgeReg.alpha_可以查看计算机为我们找到的最佳参数，结果为 31.99。而且 RidgeCV 已经用该选值为我们在训练集上训练好了模型。为了便于展示岭回归收缩系数的效果，图 3.6 用线性回归系数和岭回归得到的系数进行了对比。

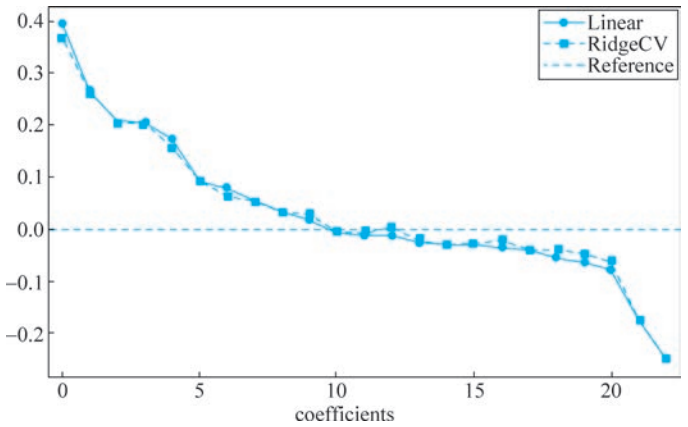


图 3.6 线性回归系数和岭回归系数对比

可以看到，线性回归中那些绝对值较大的系数，在岭回归大都向 0 收缩。但需要注意两点，一是不是所有系数都向 0 靠拢，二是岭回归的系数永远不可能收缩到 0。当然，由于数据本身的特点（观测对象数远大于预测变量数），岭回归对系数的惩罚效果不是特别明显。

目前，我们已经在训练集上完成了岭回归的建模和调参。下面就可以用训练好的模型在测试集上进行测试：

```
39.    # 基于训练好的模型进行预测
40.    testPred = ridgeReg.predict(X_test)
41.    # 查看模型在测试集上的表现
42.    from sklearn.metrics import mean_squared_error
43.    mean_squared_error(y_test, testPred)
```

mean_squared_error 会根据根据我们提供的真实值和预测值计算 MSE。结果显示，测试集上的 MSE 为 0.4263。另外，基于 OLS 的线性回归建模显示，测试集 MSE 为 0.4265。这说明相较 OLS，岭回归对模型泛化能力的提升并不明显。事实上，提升不明显的情况在社科类数据中是比较常见的。

## 3.3 拉索回归

### 3.3.1 对拉索回归的理解

同样是包含 $p$ 个预测变量的线性模型，拉索(LASSO)回归与岭回归的损失函数形式略有不同，表达如下：

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j|=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j| \tag{3.9}$$

上式写为矩阵形式为 $\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2+\|\boldsymbol{\beta}^p\|_1$。可以看出，拉索回归和岭回归损失函数的唯一的区别，在于 $\beta_j^2$ 被替换为了 $|\beta_j|$。$\sum_{j=1}^{p}|\beta_j|$ 被称为 L1 范数(L1 norm)。和岭回归一样，拉索回归对训练集的拟合程度作出了牺牲，但可以换来方差的减少，做到方差与偏差的权衡。当然，拉索回归同样会使系数向 0 收缩，且 $\lambda$ 越大，收缩越明显。区别在于，岭回归的系数不会等于 0，拉索回归的系数却可能等于 0，尤其是当 $\lambda$ 比较大的时候。换句话说，拉索回归会得到一个稀疏模型(Sparse Model)，最后只有部分预测变量被用来预测。这体现了拉索回归在特征选择方面的作用。那么，为什么拉索回归中系数可能等于 0，岭回归却不会呢？

可以证明，岭回归和拉索回归的损失函数最小化问题，分别等价于两个条件极值问题：(1)岭回归为 $\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2$ s.t. $\sum_{j=1}^{p}\beta_j^2\leqslant s$；(2) 拉索回归为 $\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\|\boldsymbol{X\beta}-\boldsymbol{y}\|_2^2$ s.t. $\sum_{j=1}^{p}|\beta_j|\leqslant s$。当只对两个权重进行估计时，上面的估计过程可以表示为图 3.7 的形式。

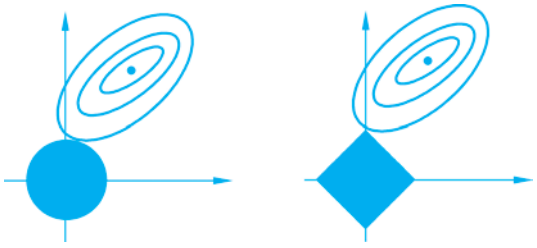

图 3.7 岭回归和拉索回归损失函数比较

其中，左侧为岭回归，右侧为拉索回归。$x$ 轴和 $y$ 轴分别代表预测变量的对应系数 $\hat{\beta}_1$

和 $\hat{\beta}_2$。椭圆线是 OLS 损失函数的等高线图，黑色的圆和菱形则分别是岭回归和拉索回归的约束条件。在 OLS 中，系数应该是图中的实心点，因为此时有 RSS 最小。而带有惩罚项时，应该找等高线与代表约束条件的图形的切点。可以看到，拉索回归更容易让切点落在坐标轴上，也就是其中一个系数会等于 0。

很难说岭回归和拉索回归哪个更好。如果认为每一个预测变量都必不可少，并起到了大致相当的预测作用，岭回归可能是个更好的选择，因为它会综合利用所有变量进行预测。如果认为预测变量中只有一少部分对目标变量有预测效力，并且想获得一个系数更少，更好解释的模型，那么最好选择拉索回归，它可以自动帮你完成变量选择的过程。变量选择能力让拉索有一些更灵活的使用方式。例如，由于社科数据中缺失值比较普遍，一些学者在进行简单的均值插补后，会通过拉索筛选出一个更小的预测变量集合，从而便于执行更为复杂的插补，最终来预测儿童福祉状况。[①] 需要注意的是，拉索以预测而非统计推断为目的，所以传统社会计量对拉索正则化的应用比较少，但也有例外。比如，对数线性模型（Log-Linear Model）常被用来对列联表数据进行建模，由于该模型的主要目的是对定类变量间关系模式进行描述，而不是因果推断，因此可以使用拉索正则化来进行模型选择。[②]

虽然拉索具有特征选择的良好性质，但也有一些缺陷。例如，当两个预测变量呈现高度相关时，它们对目标变量的影响会非常相似。在这种情况下，拉索只会随机选择一个变量，而另一个变量的系数很可能被压缩到 0。尽管这一般不会影响预测精度，但当基于回归结果对不同变量效应进行解释，或导出某些假设时，一些重要变量很可能因此被忽视。[③]

### 3.3.2 贝叶斯角度理解岭回归与拉索回归

设参数向量 $\boldsymbol{\beta}=(\beta_0,\beta_1,\cdots,\beta_p)^{\mathrm{T}}$，$\boldsymbol{X}$ 为预测变量构成的矩阵。$\boldsymbol{\beta}$ 的先验分布为 $p(\boldsymbol{\beta})$，那么当前目标变量 $\boldsymbol{Y}$ 的似然函数可以写成 $f(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\beta})$。需要假设 $\boldsymbol{Y}$ 的分布为 $\boldsymbol{Y}\sim N(\boldsymbol{X\beta},\sigma^2\boldsymbol{I})$，其实就是说每一观测对象的目标变量彼此独立，且服从方差相同的正态分布。

根据贝叶斯原理，我们知道 $\boldsymbol{\beta}$ 参数的后验分布有 $p(\boldsymbol{\beta}|\boldsymbol{X},\boldsymbol{Y})\propto f(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\beta})p(\boldsymbol{\beta})$。为了估计 $\boldsymbol{\beta}$ 的值，需要计算最大化参数后验概率（MAP），即

$$\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmax}}\, f(\boldsymbol{Y}\mid\boldsymbol{X},\boldsymbol{\beta})\cdot p(\boldsymbol{\beta}) \tag{3.10}$$

我们进一步假设 $p(\boldsymbol{\beta})=\prod_{j=1}^{p}g(\beta_j)$，$g$ 是某分布 $\boldsymbol{S}$ 的概率密度函数。那么存在以下结论：如果 $\boldsymbol{S}$ 为正态分布，其均值为零且标准差是与 $\lambda$ 有关的一个函数，则贝叶斯公式给出的最大后验估计，就是岭回归给出的估计。如果 $\boldsymbol{S}$ 是拉普拉斯分布，其均值为零且尺度参数是与 $\lambda$ 有关的一个函数，则贝叶斯公式给出的最大后验估计，就是拉索回归给出的估计。

---

① Stanescu D, Wang E. Using LASSO to Assist Imputation and Predict Child Well-Being. Socius: Sociological Research for a Dynamic World, 2019, 5: 1-21.

② Bucca M, Urbina D R. Lasso Regularization for Selection of Log-Linear Models: An Application to Educational Assortative Mating. Sociological Methods & Research, 2021, 50(4), 1763-1800.

③ Wang H, Lengerich B J, Aragam B, et al. Precision Lasso: Accounting for Correlations and Linear Dependencies in High-dimensional Genomic Data. Bioinformatics, 2019, 35(7): 1181-1187.

### 3.3.3 案例 2：如何预测住院群体的医疗支出

#### 1. 背景

假如你是某医院的研究员，现在需要你基于一份住院群体的数据，通过机器学习建模来预测有住院需求的群体在一年内的住院总支出。所用数据是 2011 年和 2013 年 CHARLS 全国基线调查数据中有住院情况的样本。[①]

所涉变量包括：个人全年住院的医疗总支出、是否加入新农合、家庭财富、慢性疾病数量、15 岁时的健康状况、健康冲击、抽烟、喝酒、健身开支、已婚、年龄、小孩数、老人数、性别、村医疗机构数量、村人均纯收入、村企业数、受教育水平、所在省份、地区类型和年份。

#### 2. 代码与实操

首先读取数据并拆分训练集和测试集：

```
X = pd.read_csv("data/data104952/CNCMS_predictors.csv")
y = pd.read_csv("data/data104952/CNCMS_target.csv")
#  使用 sklearn 中的 train_test_split 函数拆分训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_
state = 728)
```

拆分完成后，就可以在训练集上进行拉索回归的训练了：

```
from sklearn.linear_model import Lasso
lasso_temp = Lasso(alpha = 10)                    # 设定 lambda = 10
lasso_temp.fit(X_train, y_train)                  # 训练模型
```

刚才我们手动设定 $\lambda=10$。可以看一下此时有多少个系数收缩到 0：

```
np.sum(lasso_temp.coef_ == 0)
```

结果显示，55 个预测变量中有 7 个系数都收缩为 0。人为设定 $\lambda$ 值不仅没有根据，而且很不可靠，所以应该使用交叉验证的方法。下面我们基于训练集，使用 5 折交叉验证，来计算不同 $\lambda$ 所得拉索回归模型的表现（用 MSE 衡量），见图 3.8：

```
# 绘制所有 lambda 对应的 mse(仅针对训练集)
# 导入模块
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score, KFold
# 待搜索的参数 lambda(这里是 0.01 到 1000 的等比数列)
lassoAlphas = np.logspace(-2, 3, 100)
# 定义 K 折方法
fold5 = KFold(n_splits = 5, shuffle = True, random_state = 728)
# 遍历 lambda 值
lassos = []
for alpha in lassoAlphas:
    lasso_ = Lasso(alpha = alpha)
    lassoScore = -1 * cross_val_score(lasso_, X_train, y_train,
cv = fold5, scoring = "neg_mean_squared_error").mean()
```

① 本数据是章丹等(2019)的文章所用资料. 感谢 CHARLS 调查项目组授权，允许我们使用此数据. 该数据集为官方发布数据的一部分，包含部分受访者和再编码后的部分变量，只作案例演示之用. 获得完整数据请查看 CHARLS 官网 http://charls.pku.edu.cn. 参见：章丹，徐志刚，陈品. 新农合“病有所医”有无增进农村居民健康？对住院患者医疗服务利用、健康和收入影响的再审视. 社会，2019，39(02)：58-84.

```
        lassos.append(lassoScore)
    # 绘图
    import matplotlib.pyplot as plt
    plt.plot(lassoAlphas, lassos, c = "black", label = "Lasso")
    plt.legend()
```

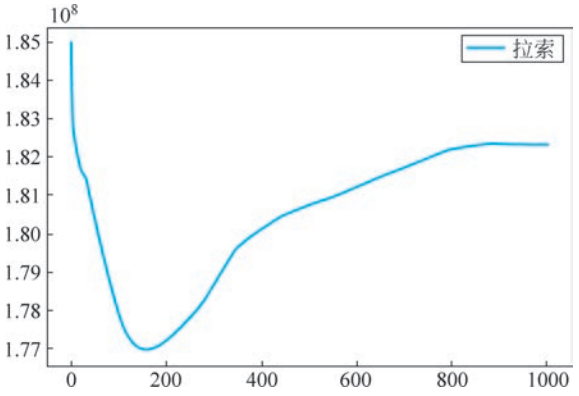


图 3.8 训练集拉索模型不同 λ 值对应的 MSE 表现

可以看到，交叉验证显示最佳的 λ 值为 0～200，且特别靠近 200。在实际操作中，我们也不必执行上面繁复的操作去调参，sklearn 模块已经写好了自带交叉验证调参的 LassoCV：

```
    #定义一个自带 cv 调试 lambda 功能的拉索回归算法
    from sklearn.linear_model import LassoCV
    #待搜索的参数 lambda
    lassoAlphas = np.logspace(-2, 3, 100)
    #定义算法
    lassoReg = LassoCV(alphas = lassoAlphas, cv = fold5)
    #训练
    lassoReg.fit(X_train, y_train)
    #查看找到的 alpha
    lassoReg.alpha_
```

lassoReg.alpha_可以查看计算机为我们找到的最佳超参数，结果为 155.57。而且 LassoCV 已经用该选值为我们在训练集上训练好了模型。np.sum(lassoReg.coef_==0)的结果说明，有 35 个变量的系数被收缩到 0，也就是说，我们得到了一个更易于解释的模型！为了便于展示拉索回归收缩系数的效果，图 3.9 用线性回归系数、岭回归系数和拉索回归的系数进行了对比。

可以看到，线性回归中那些绝对值较大的系数，在拉索回归大都向 0 收缩。拉索回归的系数是可以收缩到 0 的。目前，我们已经在训练集上完成了拉索回归的建模和调参，下面就可以用训练好的模型在测试集上进行测试：

```
    #基于训练好的模型进行预测
    testPred = lassoReg.predict(X_test)
    #查看模型在测试集上的表现
    from sklearn.metrics import mean_squared_error
    mean_squared_error(y_test, testPred)
```

结果表明，测试集上的 MSE 为 203382628。如果采用 OLS 线性回归，测试集 MSE 为

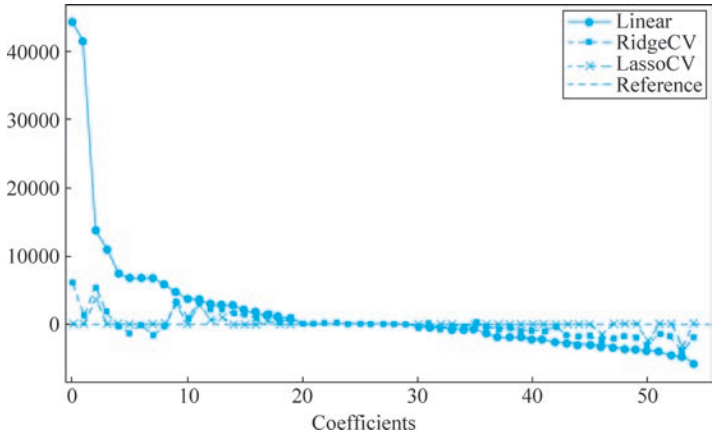


图 3.9 线性回归系数、岭回归系数与拉索回归系统对比

210170904。相较 OLS,拉索回归在一定程度上提升了线性回归的泛化能力,而且使用的变量更少,模型更容易解释,不过预测误差仍然比较大。这说明面对影响因素很多且机制复杂的问题时,线性回归起到的预测作用会比较有限。

## 3.4 弹 力 网

### 3.4.1 对弹力网的理解

如前所述,选择岭回归还是拉索回归取决于研究者的先验知识。如果先验知识是所有预测变量都对预测有所贡献,就应该选择岭回归。反之,如果只有部分变量对预测有贡献,就应该选择拉索回归,因为它可以帮助你完成特征选择。但是,当先验把握并不大时,可以尝试用弹力网(elastic net),它会在岭回归和拉索回归之间进行权衡。弹力网的损失函数形式:

$$\mathrm{RSS}+\lambda\left((1-\alpha)\sum_{j=1}^{p}\beta_j^2+\alpha\sum_{j=1}^{p}\mid\beta_j\mid\right)$$

上式表达成矩阵的形式为:$\hat{\boldsymbol{\beta}}=\underset{\boldsymbol{\beta}}{\operatorname{argmin}}\parallel\boldsymbol{X\beta}-\boldsymbol{y}\parallel_2^2+\lambda(1-\alpha)\parallel\boldsymbol{\beta}^p\parallel_2^2+\lambda\alpha\parallel\boldsymbol{\beta}^p\parallel_1$。可以看到损失函数中多了一个超参数 $\alpha$。当 $\alpha=0$,实际执行的便是岭回归;当 $\alpha=1$,实际执行的是拉索回归;当 $0<\alpha<1$,实际执行的是岭回归和拉索回归的混合,$\alpha$ 越大,拉索回归所占“比例”就越大。弹力网可以像拉索回归一样起到特征选择的作用。另外,当一组预测变量相关时,拉索回归可能会随机选择其中一个,而弹力网却可能保留更多。

### 3.4.2 案例 3: 如何预测小麦产量

#### 1. 背景

假如你受雇于美国爱荷华州的农业生产部门,现在需要你基于一份过往的小麦产量数据,通过机器学习建模来预测未来的小麦产量。数据给出了美国爱荷华州在 1930—1962 年农业关键月份的降水量、平均温度及最终小麦产量,所涉变量基本情况见表 3.1。①

① 可以在 R 语言中输入?lasso2::Iowa 查看数据详情或下载本数据,也可以在百度飞桨 AI studio 平台数据集中搜索 lanzhou_Iowa 下载本数据.本数据及案例 Rhys(2020)有参考.

表 3.1 数据所含变量

| 变 量 名 | 含 义 |
|---|---|
| Year | 年份(代表品种改进) |
| Rain0 | 季前降水 |
| Temp1 | 第一个生长季的平均气温(华氏度,下同) |
| Rain1 | 第一个生长季的平均降水(英寸,下同) |
| Temp2 | 第二生长季的平均气温 |
| Rain2 | 第二个生长季的平均降水 |
| Temp3 | 第三生长季的平均气温 |
| Rain3 | 第三个生长季的平均降水 |
| Temp4 | 收获季的平均气温 |
| Yield | 爱荷华州某年的小麦产量(蒲式耳每英亩) |

### 2. 代码与实操

我们读取数据,并显示数据的前 5 行信息,如图 3.10 所示。

```
data = pd.read_csv("data/data101272/Iowa.csv")          # 读取数据
data.head()                                             # 显示前 5 行
```

| | Year | Rain0 | Temp1 | Rain1 | Temp2 | Rain2 | Temp3 | Rain3 | Temp4 | Yield |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1930 | 17.75 | 60.2 | 5.83 | 69.0 | 1.49 | 77.9 | 2.42 | 74.4 | 34.0 |
| 1 | 1931 | 14.76 | 57.5 | 3.83 | 75.0 | 2.72 | 77.2 | 3.30 | 72.6 | 32.9 |
| 2 | 1932 | 27.99 | 62.3 | 5.17 | 72.0 | 3.12 | 75.8 | 7.10 | 72.2 | 43.0 |
| 3 | 1933 | 16.76 | 60.5 | 1.64 | 77.8 | 3.45 | 76.4 | 3.01 | 70.5 | 40.0 |
| 4 | 1934 | 11.36 | 69.5 | 3.49 | 77.2 | 3.85 | 79.7 | 2.84 | 73.4 | 23.0 |

图 3.10 显示前 5 行信息

需要将手头的数据分为训练集和测试集:

```
#  提取特征矩阵 X 和目标变量 y
X = data.drop(['Yield'], axis = 1)
y = data.Yield
# 使用 sklearn 中的 train_test_split 函数拆分训练集和测试集
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = .3, random_state = 728)
```

拆分完成后,就可以在训练集上进行弹力网的训练了:

```
from sklearn.linear_model import ElasticNet
# 设定 lambda = 10; alpha = 0.5
elastic_temp = ElasticNet(alpha = 10, l1_ratio = .5)
elastic_temp.fit(X_train, y_train)                      # 训练模型
```

可以看一下此时有多少系数收缩到 0:

```
np.sum(elastic_temp.coef_ == 0)
```

结果显示,9 个预测变量中有 4 个收缩为 0。如果使用拉索回归拟合该数据,相同 $\lambda$ 值

下，9 个预测变量的系数中会有 7 个收缩为 0，这体现了弹性网对岭回归和拉索回归的权衡。人为设定 λ 和 α 的值没有根据，所以应该使用交叉验证的方法。和之前两种算法的区别是，弹性网需要同时调试两个超参数。sklearn 模块已经写好了自带交叉验证调参的 ElasticNetCV：

```
#定义一个自带 cv 调试 lambda 功能的弹力网算法
from sklearn.linear_model import ElasticNetCV
# 定义交叉验证的方法(3 折交叉验证,重复 5 次取平均)
from sklearn.model_selection import RepeatedKFold
r3f = RepeatedKFold(n_splits = 3, n_repeats = 5, random_state = 728)
# 定义算法
elasticReg = ElasticNetCV(alphas = np.logspace( - 1, 1, 40),
                          l1_ratio = np.linspace(0.001, 1, 40),
cv = r3f, max_iter = 5000)
# 训练
elasticReg.fit(X_train, y_train)
```

elasticReg.alpha_ 可以查看计算机为我们找到的最佳参数 λ，结果为 2.42。而 elasticReg.l1_ratio_ 可以查看计算机为我们找到的最佳参数 α，结果为 1。也就是说，弹性网认为在这一案例的训练集中，直接执行拉索回归的效果更好。当然，在很多案例中，交叉验证的结果并不一定正好等于 0 或者 1。这一结果也说明，如果没有非常有把握的先验证据或理论，可以直接使用弹性网，而不是岭回归和拉索回归。

## 本章小结

本章，我们学习了线性回归的基本思想，并对其在机器学习和传统计量中的不同定位有了基本了解。传统计量重视假设检验，机器学习重视预测效力。从这种区别出发，我们接着介绍了偏差与方差、过拟合与欠拟合这两对重要的概念。对这两组概念的讨论，有助于我们理解为何机器学习需要岭回归和拉索回归，这是因为它们都缓解过拟合带来的高泛化误差问题。

接着，我们讨论了岭回归和拉索回归对损失函数的不同处理，以及对应的不同正则化项。这些设置降低了模型对训练集的拟合度，当然，也降低了模型的方差。在论述中，我们着重对这两种方法的优缺点进行了探讨和对比。同时，使用两个与社会科学相关的例子，展示了如何在 Python 中进行实践操作。案例所展示的过程，涵盖了数据读取、数据拆分、建模、预测和调参等多个步骤。

最后，我们对综合了岭回归和拉索回归的弹力网算法进行了简要介绍。当研究者先验知识不足时，可以直接使用该方法进行操作，而不必逐一试验基于 OLS 的线性回归、岭回归和拉索回归，从而让建模过程更加简便。在 3.4.2 节爱荷华州小麦数据的案例中，我们展示了如何在样本量较小时，基于重复 K 折方法对弹力网所涉及的超参数进行调参。

## 习 题 3

1. 在机器学习中，理解一个概念的最好方法就是用代码计算它。给定波士顿房价数据的训练集和测试集，你能对基于 OLS 的线性回归算法的平均方差和平均偏差进行估计吗？导入数据的代码如下：

```
# 读取数据；拆分训练集和测试集
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
X = load_boston().data
y = load_boston().target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = .3, random_
state = 822)
```

可以参考下列过程[①]：

```
import numpy as np
num_random = 200                                    # 设定随机抽样的次数
all_pred = np.zeros((num_random, y_test.shape[0]), dtype = np.float64)
                                                    # 存储预测结果的矩阵

# 定义算法 estimator(读者可以自行改变算法,对比结果)
from sklearn.linear_model import LinearRegression
estimator = LinearRegression()

# 随机数种子
rng = np.random.RandomState(822)

# 使用 boostrop 获取多组预测值
for i in range(num_random):
    # 放回抽样,抽取一个训练集
    sample_ind = np.arange(X_train.shape[0])
    bootstrap_ind = rng.choice(sample_ind,
                               size = sample_ind.shape[0],
                               replace = True)
    X_boot, y_boot = X_train[bootstrap_ind], y_train[bootstrap_ind]
    # 拟合模型,并对测试集进行预测
    pred = estimator.fit(X_boot, y_boot).predict(X_test)
    all_pred[i] = pred                              # 存储预测的结果

# 计算的 MSE
avg_MSE = ((y_test - all_pred) ** 2).mean()

# 计算 bias 和 variance
# 按列求均值,得到每个个体的 200 个预测值的平均值
main_predictions = np.mean(all_pred, axis = 0)

# 模型的偏差
avg_bias = np.sum((main_predictions - y_test) ** 2) / y_test.size
# 模型的方差
avg_variance = np.sum((main_predictions - all_pred) ** 2) / all_pred.size

print(avg_MSE, avg_bias, avg_variance)
```

2. 当岭回归和拉索回归的泛化误差特别接近时，从模型易解释性考虑，你应该选择哪

① 真实的偏差和方差是不可能计算的，因为不知道 $f(X)$ 的具体形式，所以只能估算。此处代码参考：https://github.com/rasbt/mlxtend/blob/master/mlxtend/evaluate/bias_variance_decomp.py.

个模型？为什么？

3. 在 sklearn 模块中，程序给出的均方误差往往以负值的形式出现（即负均方误差，neg_mean_squared_error）。猜测一下，这是为什么呢？

4. 在 3.2.2 节中，我们提到了如何对包含调参过程的算法所产生模型的泛化能力进行评估。该流程被称为嵌套交叉验证（Nested Cross-validation）。你能基于爱荷华州小麦数据集，对包含调参过程的岭回归、拉索回归和弹力网的泛化误差进行估计吗？（提示：考虑 cross_validate()函数。）