第七章　模型选择

7.1　简介

7.1 简 介

- 当我们考虑用模型来拟合数据, 刻画数据的结构预测变量的时候, 需要考虑两个基本问题: 一是采用什么样的模型; 二是采用哪些变量, 即判断哪些变量是重要的, 哪些变量是不重要的. 由于统计推断和预测是在选定模型之后进行的, 模型选择的好坏将直接影响到统计推断和预测的效果, 因此模型选择问题得到了极大的重视和广泛的研究.
- 回归模型包含的预测变量并不是越多越好. 首先, 收集更多变量的数据会浪费时间和精力; 其次, 使用全部的预测变量会出现多重共线性的问题; 最后, 有些预测变量对响应变量并无影响, 增加不必要的变量会带来估计噪声.因此要选择简单的且 “足够好” 的模型.
- 先介绍一个概念: 方差–偏差平衡. 当模型越来越复杂的时候, 模型的预测能力也会越好. 但问题是, 模型只是在训练集上表现不错, 而在测试集上表现比较差. 具体来说: 模型的偏差虽然小了, 但是方差很大, 即模型对 “新” 数据表现出 “不稳定性”. 这就是过拟合 (over-fitting) 了. 模型过于复杂, 连 “噪声” 也不放过, 导致训练误差 (training error) 非常小, 测试误差 (test error) 非常大.

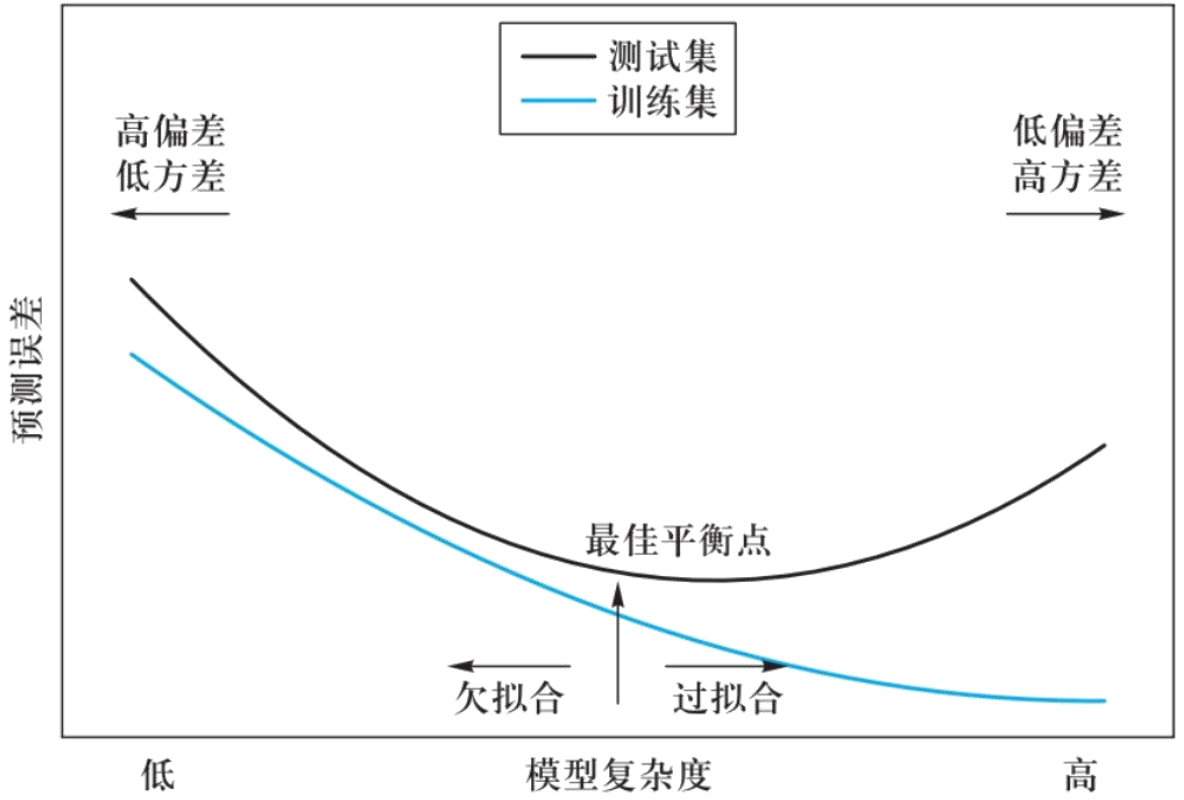
7.1 简 介

- 如果一个模型对未知数据 (通常是测试集的数据) 的预测能力很好, 就称它的泛化能力 (generalization ability) 很好, 显然过拟合意味着模型的泛化能力很差.
- 如图 7.1 所示, 当模型复杂度 (degree of complexity) 太低, 训练误差和测试误差都很高时, 模型是欠拟合 (under-fitting) 的. 随着模型复杂度的提升,测试误差先减小后增大, 但是训练误差不断地减小, 最后过拟合. 因此, 控制模型复杂度可以实现方差–偏差的平衡. 而控制模型复杂度可以转为选择最优模型. 模型选择 (model selection) 就是从多个候选模型中, 基于某评价准则和训练数据选出最优的模型. 它的主要思想是通过训练数据来估计期望的测试误差, 用数学语言描述模型选择就是: 给定数据集 D, 依据某个模型评价准则, 从候选模型集合 S_m 中选出最优模型 S^* , 即

$$S^* = \arg\min_m \left(\mathrm{crit}\left(S_m; D\right)\right),$$

  - 其中 crit 表示模型评价准则. 不同的评价准则对应不同的模型选择方法.

7.1 简 介



- 接下来介绍两种常用的模型选择方法: **子集选择法**和**压缩估计法**. 在子集选择法中我们将介绍基于检验的模型选择方法和基于准则的模型选择方法, 在压缩估计法中介绍 Lasso、非凸惩罚函数、群组变量选择方法和双层变量选择方法.

7.2　基于准则的方法

7.2 简 介

- 基于准则的方法将介绍 C_p 准则、AIC 准则、BIC 准则与交叉验证.

7.2.1 各种准则

- 通常训练集的均方误差比测试集的均方误差要低, 而我们在选择模型时是希望得到一个具有最小测试误差的模型. 并且训练误差随着更多变量加入模型中将会逐渐降低. 因此, 残差平方和 SSE 并不适用于对包含不同预测变量的模型进行模型选择.
- 为了基于测试误差选择最优模型, 就需要去估计测试误差, 通常有两种方法估计测试误差:
  - (1) 间接估计: 根据过拟合导致的偏差对训练误差进行调整.
  - (2) 直接估计: 通过交叉验证方法直接估计测试误差.
- 接下来就介绍几种适用于对包含不同预测变量的模型进行模型选择的方法, 是间接估计测试误差的方法, 包括 C_p 准则、AIC 准则、BIC 准则.

1. C_p 统计量

- C_p 统计量的思想是通过在训练集 SSE 的基础上增加惩罚项, 来调整训练误差倾向于低估测试误差这一现象.

7.2.1 各种准则

- 具体来说就是训练集 RSS 随着模型中预测变量的增加而降低, 但是此时测试集的 SSE 要比训练集的高, 所以就在训练集 SSE 的基础上增加一项, 使得该增加项的大小随着模型中预测变量个数的增加而增加. 采用最小二乘法拟合一个包括 k 个预测变量的模型, C_p 值计算如下:

$$C_p = \frac{1}{n}\left(\mathrm{SSE} + 2k\hat{\sigma}^2\right),$$

  - 其中, $\hat{\sigma}$ 是响应变量观测误差的方差的估计值, k 是模型中变量的个数. C_p 统计量在训练集 SSE 的基础上增加惩罚项 $2k\hat{\sigma}^2$, 这可以看作是在考虑拟合残差同时, 依变量个数施加的惩罚. 显然, 惩罚项随模型中变量个数的增加而增大, 用于调整由于变量个数增加而不断降低的训练集 SSE. 如果 $\hat{\sigma}$ 是观测误差方差的无偏估计, 那么 Cp 是测试均方误差的无偏估计. Cp 值越小, 表示模型的准确性越高.

2. AIC 准则

- AIC 准则是衡量统计模型拟合优良性的一种准则, 由日本统计学家赤池弘次[81] 在 1974 年提出, 因而又叫赤池信息准则.

7.2.1 各种准则

- 它是权衡估计模型复杂度和拟合数据优良性的准则. AIC 准则的定义公式为

$$\mathrm{AIC} = 2k - 2\ln(L),$$

  - 其中 k 是模型参数个数, L 是最大似然函数. 由于似然函数的值越大得到的估计量就越好, 因此 $-2\ln(L)$ 越小越好, 该项刻画了模型的精度或拟合程度,又考虑到模型包含的预测变量的个数不能太多, 故加入 $2k$ 对模型参数个数进行惩罚. 即一方面要使似然函数值较大, 另一方面通过惩罚部分限制模型的复杂度. 因此, AIC 是寻找可以最好地解释数据但包含最少自由参数的模型.

- 假设模型的随机误差服从独立正态分布. 此时 $-2\ln(L) = n\ln\left(\frac{\mathrm{SSE}}{n}\right) +$ 常数, 那么 AIC 变为

$$\mathrm{AIC} = 2k + n\ln\left(\frac{\mathrm{SSE}}{n}\right),$$

  - 其中 n 为样本量, k 为参数个数, SSE 为残差平方和.

7.2.1 各种准则

3. BIC 准则

- Schwarz[82] 为了能够克服在大样本数据情形下 AIC 准则容易失效的不足, 提出了贝叶斯信息准则. 其定义公式为

$$\mathrm{BIC} = 2k - 2\ln(L),$$

- 假设条件是模型的误差服从独立正态分布, BIC 准则的公式为

$$\mathrm{BIC} = k\ln(n) + n\ln\left(\frac{\mathrm{SSE}}{n}\right).$$

  - 其中, n 为样本量. 当样本量很大时, AIC 准则中的残差平方和的所在项会受到样本量影响而放大, 而参数个数的惩罚因子却未随样本量的变化而变化. 因此当样本量很大时, 使用 AIC 准则选择的模型不收敛到真实的最优模型. 事实上, 它通常比真实模型所含的未知参数个数要多. 为引入对预测变量过多的惩罚, BIC准则把 AIC 中的 2 换成了 $\ln(n)$. 所以, 当 $n > e^2$ 时, BIC 统计量相比于 AIC统计量给包含多个变量的模型施以较重的惩罚, 所以与 AIC 统计量相比, BIC 得到的模型规模更小. 如果真实模型是有限维参数, BIC 准则会表现得更好.

7.2.1 各种准则

4. AIC 和 BIC 比较

- AIC 和 BIC 的原理是不同的, AIC 是从预测角度, 选择一个好的模型来预测, BIC 是从拟合角度, 选择一个对现有数据拟合最好的模型, 从贝叶斯因子的解释来讲, 就是边际似然最大的那个模型.
- **相同点:** 构造这些统计量所遵循的统计思想是一致的, 就是在考虑拟合残差的同时, 依预测变量个数施加“惩罚”.
- **不同点:** BIC 的惩罚项比 AIC 大, 考虑了样本个数, 可以防止模型精度过高造成的模型复杂度过高. AIC 和 BIC 前半部分是惩罚项, 当 $n \geqslant 8$ 时,$k\ln(n) \geqslant 2k$, 所以, BIC 相比 AIC 在大数据量时对模型参数惩罚得更多, 导致 BIC 更倾向于选择参数少的简单模型.

7.2.2 交叉验证

- 交叉验证 (cross-validation) 是一种没有任何前提假定直接估计泛化误差的模型选择方法. 由于没有任何假定, 它可以应用于各种模型选择中. 它的基本思想是将数据分为两部分, 一部分数据用来进行模型的训练, 通常叫做训练集, 另一部分数据用来测试训练生成模型的误差, 叫做测试集. 由于两部分数据的不同, 泛化误差的估计是在新的数据上进行, 这样的泛化误差的估计可以更接近真实的泛化误差. 在数据足够的情况下, 可以很好估计出真实的泛化误差, 但是在实际应用中, 往往只有有限的数据可用, 就必须对数据进行重用, 即对数据进行多次切分来得到好的估计, 自从交叉验证提出以后, 人们提出了不同的数据切分方式, 因此产生了多种形式的交叉验证方法, 下面介绍一些主要的交叉验证方法.

1. 验证集方法

- 用给定的观测数据集估计使用某种模型拟合所产生的测试误差, 一种最简单、直接的方法是验证集方法. 首先, 将给定的样本观测数据集随机地分为不重复的两部分, 然后用训练集来训练模型, 在测试集上验证模型及参数. 接着, 再把样本打乱, 重新选择训练集和测试集, 继续训练数据和检验模型. 最后选择损失函数评估最优的模型和参数. 这就是验证集方法的基本原理. 下面以5×2交叉验证为例介绍训练集和测试集的选择方法.

7.2.2 交叉验证

- 5×2 交叉验证法的主要思想是将数据集 V 平均分为两部分 $V_1^{(1)}$ 和 $V_1^{(2)}$,首先用 $V_1^{(1)}$ 作为训练集, $V_1^{(2)}$ 作为验证集, 然后互换角色, 用 $V_1^{(2)}$ 作为训练集, $V_1^{(1)}$作为验证集, 这样就得到了第一折, 即第一次对折. 为了得到第二折,将数据集重新打乱并划分为新的两个等份 $V_2^{(1)}$ 和 $V_2^{(2)}$. 将 $V_2^{(1)}$ 作为训练集,$V_2^{(2)}$ 作为验证集, 然后对调两者的角色, 得到第二折, 重复以上作法五次, 会得到十个训练集和验证集. 当然可以进行超过五次的对折得到更多的训练集和验证集, 但在五次对折之后, 各个集合共享了许多样本, 使得计算出来的泛化误差估计变得相互依赖无法增加新的信息.
- 可以看出验证集方法的**优点**: 处理简单, 只需随机把原始数据分为两组即可; 并且测试集和训练集是分开的, 避免了过拟合的现象. 但是也有**缺点**:
  - (1) 由于是随机地将原始数据分组, 所以最后验证集的分类准确率与原始数据的划分有很大的关系, 最终模型的确定将强烈依赖于训练集和验证集的划分方式, 测试误差会根据划分方式的不同而变化.
  - (2) 由于只使用部分数据进行训练, 得到的结果并不具有说服性. 在实际应用中, 用于模型训练的观测数据越多, 模型的效果就越理想. 而验证集方法无法充分利用所有的观测数据, 对模型的效果会产生影响.

7.2.2 交叉验证

- 接下来介绍的交叉验证法, 针对验证集方法的上述问题进行了改进.

2. 留一交叉验证法

- 留一法 (leave-one-outcross-validation, LOOCV) 的基本思想是每次从个数为 n 的样本集中取出一个样本作为验证集, 剩下的 $n-1$ 个样本作为训练集, 重复进行多次, 依次取遍所有 n 个数据作为验证集. 也就是每次只留下一个样本作测试集, 其他样本作训练集. 操作步骤如下:
  - (1) 设原始数据有 n 个样本, 即 $\{(\boldsymbol{X}_1, Y_1), \cdots, (\boldsymbol{X}_n, Y_n)\}$, 以每个样本单独作为验证集, 其余的 $n-1$ 个样本作为训练集进行训练, 则会得到 n 个模型.
  - (2) 对于每个在训练集上拟合的模型而言, 考虑其相应的验证集进而得到该模型的均方误差.

回归问题: $\mathrm{MSE}_i = \left(Y_i - \hat{Y}_i\right)^2$;

分类问题: $\mathrm{Err}_i = I\left(Y_i \neq \hat{Y}_i\right)$.

7.2.2 交叉验证

- (3) 将 n 个模型的 n 个均方误差取均值后, 得到测试均方误差的 LOOCV估计:

  回归问题: $\mathrm{CV}_{(n)}=\frac{1}{n}\sum_{i=1}^{n}\mathrm{MSE}_i$;

  分类问题: $\mathrm{CV}_{(n)}=\frac{1}{n}\sum_{i=1}^{n}\mathrm{Err}_i$.

- LOOCV 是交叉验证方法中最常见的一种方法, 相比于验证集方法, 它有如下特点:
  - **优点:** 每一回合中几乎所有的样本都用于训练模型, 这样评估所得的结果比较可靠; 试验中没有随机因素, 整个过程是可重复的.
  - **缺点:** 计算成本高, 当 n 非常大时, 计算耗时.
- 于是就有了一种折中的方法——K 折交叉验证法.

7.2.2 交叉验证

3. K 折交叉验证法

- K 折交叉验证法 (K-fold cross validation, K-CV) 的原理:
  - (1) 将原始数据分成 K组 (一般是均分), 或者说折 (fold); 然后, 让每个子集数据分别做一次验证集, 其对应的剩余$K-1$组子集数据作为训练集, 这样会得到 K 个模型.
  - (2) 同 LOOCV 方法一样, 基于 K-CV 的试验建立的 K 个模型, 分别在训练集上拟合模型, 再将拟合的模型用于保留的验证集上计算均方误差.

    回归问题: $\mathrm{MSE}_i = \dfrac{1}{|G_k|}\sum_{i\in G_k}\left(Y_i - \hat{Y}_i\right)^2;$

    分类问题: $\mathrm{Err}_i = I\left(Y_i \neq \hat{Y}_i\right).$
  - (3) 用这 K 个模型最终的验证集的均方误差的平均数作为此 K-CV 的估计.

    回归问题: $\mathrm{CV}_{(K)} = \dfrac{1}{K}\sum_{K=1}^{K}\mathrm{MSE}_K;$

    分类问题: $\mathrm{CV}_{(K)} = \dfrac{1}{K}\sum_{K=1}^{K}\mathrm{Err}_K.$

7.2.2 交叉验证

- 在实际操作上, K 要够大才能使各回合中的训练样本数够多. 根据经验一般选取 $K = 5$ 或 $K = 10$, 这两个取值会使测试误差的估计不会有过大的偏差或方差, $K = 10$ 是相当足够了.
- K 折交叉验证法有如下特点:
  - **与 LOOCV 方法的关系:** LOOCV 方法就是 K 折交叉验证法当 $K = n$ 时的一个特例. 由于 $K < n$, K 折交叉验证法拟合模型次数 K 便小于LOOCV 方法, 这样就降低了计算成本.
  - **优点:** K-CV 使得每一个样本数据都既被用作训练数据, 也被用作测试数据, 可以有效避免过拟合以及欠拟合的发生, 最后得到的结果也比较具有说服性.
  - **缺点:** K 折交叉验证法的缺点在于 K 的确定. 一方面, K 越大, 样本划分的组多, 训练集包含的观测数据多, 拟合的偏差越小. 特例是当 $K = n$ 时也就是 LOOCV, 我们能够得到一个近似无偏的测试误差估计. 另一方面, K越大就意味着每次用于拟合模型的训练集的观测数据重合度高, 模型也就更相似. 还是考虑特殊情况 LOOCV, 每一次训练的观测数据几乎是相同的, 拟合的结果之间是高度 (正) 相关的, 由于许多高度相关的量的均值要比相关性相对较小的量的均值具有更高的波动性, 因此方差也将更大
- 与 C_p 准则、AIC 准则、BIC 准则相比, 这些方法的优势在于直接给出了测试误差的直接估计, 符合我们最初选择测试误差最小的模型的目的.

7.3　基于检验的方法

7.3 基于检验的方法

- 在基于检验的方法中介绍两种最常见的方法: 最优子集法和逐步选择法.

7.3.1 最优子集法

- 对于含有p个预测变量组成的集合来说, 最优子集选择即对p个预测变量的所有可能组合分别使用最小二乘回归进行拟合, 即对含有一个预测变量的模型, 拟合 p 个模型; 对含有两个预测变量的模型, 拟合 $p(p-1)/2$ 个模型,依次类推最后一共拟合了 $C_p^0+C_p^1+\cdots+C_p^p$ 即 2^p 个模型, 再根据指标从所有可能模型中选取一个最优的模型. 最优子集法具体过程可以概括为
  - 1. 记不包含预测变量的零模型为 M_0.
  - 2. 对 $k=1,2,\cdots,p$,
    (1) 从 p 个预测变量中任意选择 k 个, 拟合 C_p^k 个模型.
    (2) 在 C_p^k 个模型中选择最优的一个 (SSE 最小或决定系数最大), 记为 M_k.
  - 3. 根据交叉验证预测误差、Cp、AIC、BIC 或者修正的决定系数 $\overline{R}^2$ 等指标, 从 $M_0,\cdots,M_p$ 个模型中选择一个最优模型.
- 步骤 2 先在相同数量预测变量的模型中选择一个最优模型 (内循环); 步骤 3 在不同数量预测变量的模型中进行模型选择 (外循环). 通过外循环选择最优模型, 需要改变准则.

7.3.1 最优子集法

- 因为随着模型中预测变量数目增加, 这 $p+1$ 个模型的 SSE 会下降. 如果我们仅依据 SSE 进行模型选择的话, 最后选出来的最优模型将包含所有变量. 因此对于变量个数相同的模型, 可通过 SSE 来选择最优模型; 对于变量个数不同的模型, 可使用交叉验证预测误差、C_p、AIC、BIC 或者 $\overline{R}^2$ 等指标进行模型选择. 例如, 根据修正决定系数 $\overline{R}^2$ 的定义:

$$\overline{R}^2 = 1 - \frac{\mathrm{SSE}/(n-k-1)}{\mathrm{SST}/(n-1)}. \tag{7.3.1}$$

  - 我们知道, $\overline{R}^2$ 越大, 模型拟合程度越好. 这里引入了预测变量数量 k 从而对预测变量增加进行了惩罚, 即模型中包含的预测变量 k 增加时会抑制 $\overline{R}^2$ 的增加, 使选出来的模型具有更小的测试误差. 通俗来讲, 修正决定系数并不会“偏爱”预测变量较多的模型.

- 显然, 最优子集法运用了穷举的思想. 其优势很明显: 简单直接, 遍历所有可能的情况, 最后的选择一定是最优的; 劣势也很明显: 当 p 越大时, 可选模型数量在迅速增加, 计算量明显增大, 降低计算效率. 因此, 最优子集法只适用于 p 较小的情况. 一般来说, 当 $p<10$ 时, 可以考虑最优子集法.

7.3.2 逐步选择法

- 最优子集法在 p 很大时运算效率低, 并且随着搜索空间的增大, 找到的模型泛化能力差. 因此, 本章引入逐步选择法避免上述缺点.
- 在逐步选择法中, 模型会在原有变量的基础上一次增加一个显著的预测变量或剔除一个不显著的预测变量, 直到既没有显著的预测变量选入回归方程, 或也没有不显著的预测变量从回归方程中剔除为止. 最优子集法与逐步选择法的区别在于, 最优子集法可选择任意 k 个变量进行建模, 而逐步选择法只能在之前所选的 k 个变量的基础上建模. 常用的逐步选择法包括向前逐步选择、向后逐步选择、双向选择.

1. 向前逐步选择

- 向前逐步回归的思想是由少到多, 逐个引入预测变量. 首先考虑一个不包含任何预测变量的零模型, 从零模型开始, 依次引入一个预测变量, 直至没有可引入的变量为止. 所谓没有可引入的变量是指要添加的任何新变量都不会使模型有所改进.

7.3.2 逐步选择法

- 具体来说就是从零模型 M_0 开始, 这个模型只有截距项而没有任何预测变量. 然后, 将 p 个预测变量依次加入模型中 (每次只加入一个预测变量, 考虑 p 次, 得到 $p+1$ 个模型), 保留 SSE 最小或决定系数最大的那个预测变量, 此时模型含有一个预测变量, 记为 M_1. 然后在此基础上, 将剩余的 $p-1$个预测变量依次分别加入, 仍然保留 SSE 最小或决定系数最大的那个预测变量, 此时得到的模型含有 2 个预测变量, 记为 M_2. 这样重复操作, 直至包含 p个预测变量的模型 M_p. 最后根据交叉验证预测误差、C_p、AIC、BIC 或者修正的 $\overline{R}^2$等指标, 从 $p+1$ 个最优模型中选择一个最优模型.

- 向前逐步回归的算法过程如下:
  - (1) 记不包含任何预测变量的零模型为 M_0.
  - (2) 对 k=0,1,2,$\cdots$, $p-1$,

    ①基于上一步选取的最优模型 M_k 的 k 个预测变量, 将剩余的 $p-k$ 个预测变量分别加入, 这样就得到 $p-k$ 个模型;

    ② 在上述 $p-k$ 个模型中选择 SSE 最小或决定系数最大的模型作为最优模型, 记为 M_{k+1}.

7.3.2 逐步选择法

- (3) 进一步使用交叉验证误差、C_p、AIC、BIC 或者修正的决定系数$\overline{R}^2$等指标, 从 $p+1$ 个最优模型中选择一个最优模型.

- 与最优子集选择法要对 2^p 个模型进行拟合不同, 向前回归只需要对零模型以及第 k 次迭代所包含的 $p-k$ 个模型进行拟合, 其中 $k=0, 1, 2, \cdots, p-1$.相当于拟合 $\sum_{k=0}^{p-1}(p-k)=p(p+1)/2$ 个模型, 特别地, 当 $p=30$ 时, 最优子集需要拟合 1073741824 个模型, 而向前逐步回归只需要拟合 465 个模型.

- 但是需要注意的是向前逐步回归无法保证最后得到的模型是 2^p 个模型中最优的. 例如在给定的包含三个变量 X_1、X_2、X_3 的数据集中, 最优的单变量模型只包含 X_2, 最优的双变量模型只包含 X_1、X_3, 且通过一些指标得出最优双变量模型在 $23=8$ 个模型中是最优的. 则通过向前逐步回归无法得到该最优模型, 因为 M_1 包含变量 X_2, 而 M_2 只能包含 X_2、X_3 或者 X_2、X_1,而无法得到包含 X_1、X_3 的双变量模型.

- 下面用一个例子说明向前回归的具体过程.

7.3.2 逐步选择法

- 假设 $\{X_1, X_2, \cdots, X_p\}$ 为待选择的预测变量的集合, 首先可以将每一个单一变量看作是一个子集, 构成 p 个模型, 在这 p 个模型中选择 SSE 最小或决定系数最大的为最优模型, 在此不妨假设 $\{X_1\}$ 是最优的单一变量模型; 接下来在上一步的基础上添加一个变量, 得到 $p-1$ 个包含两个预测变量的模型, 再基于上述准则选出最优模型, 不妨假设 $\{X_1, X_2\}$ 为最优, 且优于 $\{X_1\}$,则第二步选择结束; 依次类推, 直到第 $k+1$ 轮选择的最优模型拟合效果不如第 k 轮选出最优模型, 则整个子集选择过程结束, 并将第 k 轮选出的模型作为最后的最优模型.

2. 向后逐步选择

- 与向前逐步回归法的选择方向相反, 向后逐步回归法以一个包含全部预测变量的全模型为起点, 逐次迭代, 每次剔除一个对模型效果最不利的冗余变量, 直到继续剔除变量却不能使模型质量有所改进为止.具体来说就是从包含全部 p 个预测变量的模型 M_p 开始, 一个个地移除 p 个预测变量, 保留 SSE 最小或决定系数最大的那个模型, 保留下来的模型是包含 $p-1$ 个预测变量的模型, 记为 M_{p-1},基于该模型继续移除剩余的 $p-1$ 个预测变量.

7.3.2 逐步选择法

- 这样重复操作, 直至包含 0 个预测变量的模型 M_0. 然后根据交叉验证预测误差、C_p、AIC、BIC 或者修正的决定系数 $\overline{R}^2$ 等指标, 从 $p+1$ 个最优模型中选择一个最优模型.
- 向后逐步回归的算法过程如下:
  - (1) 记包含所有预测变量的全模型为 M_p.
  - (2) 对 $k=p, p-1, \cdots, 1,$

    ①基于上一步选取的最优模型的 k 个预测变量, 依次逐个剔除, 这样就可以构建 k 个模型;

    ② 在上述 k 个模型中选择最优模型, 记为 M_{k-1}.
  - (3) 进一步使用交叉验证预测误差、C_p、AIC、BIC 或者修正的决定系数 $\overline{R}^2$ 等指标, 从 $p+1$ 个最优模型中选择一个最优模型, 为最后得到的最优模型.
- 同样, 向后逐步回归共需对 $1+p(p+1)/2$ 个模型进行搜索, 比最优子集选择更高效. 但是, 向后逐步回归无法保证得到的模型是包含 p 个预测变量子集的最优模型.

7.3.2 逐步选择法

- 这是因为向后逐步回归同样都依赖于上一步的迭代结果.此处需要注意, 因为向后逐步回归法是从全模型开始的, 如果使用最小二乘法进行拟合, 需要满足 $n > p$ 的条件. 而向前逐步回归法则不需要满足这一条件. 当预测变量向量的维度 p 比较高时, 应优先选择向前逐步回归法.
- 向前逐步回归和向后逐步回归都属于贪心算法, 能够局部达到最优, 但是从全局来看不一定是最优的. 向前逐步回归虽然每一次都能选取最显著的一个预测变量, 但在实际情况下, 很可能有的预测变量在开始时是显著的, 但是在其余预测变量添加进去之后, 它就变得不显著了, 但向前逐步回归此时不会剔除该变量. 而向后逐步回归则很有可能会遗漏一些很重要的变量, 比如刚开始变量 X_1 不显著, 但是在剔除 m 个变量后其变得显著, 此时向后逐步回归并不会重新加入这个变量. 总的来说这些问题是由于向前逐步回归不会剔除已经加入进来的变量, 向后逐步回归不会加入已经剔除的变量导致的,下面介绍的双向选择会解决这些问题.

7.3.2 逐步选择法

3. 双向选择

- 双向选择是向前和向后逐步选择法的结合. 每次引入一个变量, 但与此同时也会剔除对模型没有贡献的变量. 引入一个变量后, 我们首先通过 F 检验, 验证新变量是否会使得模型发生显著性变化. 当原有变量由于新变量的加入变得不再显著时, 剔除此变量. 若模型依旧显著, 再对所有变量进行 t 检验, 剔除不显著变量. 直到既没有新的显著的预测变量加入回归方程, 也没有原有的不显著的预测变量从回归方程中剔除为止, 最终得到一个最优模型. 在这个过程中, 某一特定预测变量可能会被反复引入和删除. 该方法在试图达到最优子集选择效果的同时也保留了向前和向后逐步回归在计算效率上的优势.

7.4　正则化方法

7.4 正则化方法

- 子集选择方法通过保留预测变量的一个子集, 并舍弃其他变量, 产生一个简单的具有一定解释性的模型. 然而子集选择法也有缺点, 主要体现在两个方面: 首先, 在进行最优变量子集筛选的过程时离散, 通常具有很高的变异性, 不够稳定; 其次, 忽略了变量选择过程中的随机误差.
- 模型选择的另一个典型方法是正则化, 正则化就是通过对模型参数进行调整 (数量和大小), 降低模型的复杂度, 以达到可以避免过拟合的效果. 具体的解决方法是在模型的损失函数中加入正则项. 如果不加入正则项, 我们的目标是最小化损失函数; 加入正则项以后, 变成最小化损失同时复杂度不要太高, 这一方法称为结构风险最小化. 通过惩罚函数约束模型的回归系数, 同步实现变量选择和系数估计, 模型估计是一个连续的过程, 更正了子集选择法的缺陷.
- 在详细介绍这一方法之前, 先介绍目标函数和范数的概念.

7.4 正则化方法

1. 目标函数

- 求解优化问题的关键就是最小化目标函数. 假设样本矩阵为$\mathbf{X}=(X_1,\cdots,X_n)^{\mathrm{T}}$是$n\times p$矩阵, 因变量记为$\boldsymbol{Y}=(Y_1,\cdots,Y_n)^{\mathrm{T}}$. 我们考虑每组样本中的预测变量为$\boldsymbol{X}_i$, 响应变量为$\boldsymbol{Y}_i$, p维参数为β. 目标函数的一般形式为

$$\min_{\beta} L(\boldsymbol{\beta})=\min_{\beta}\left\{Q(\boldsymbol{\beta}|\boldsymbol{Y},\boldsymbol{X})+\sum_{j=1}^{p} p_{\lambda}\left(\left|\beta_j\right|\right)\right\},$$

  - 其中$Q(\beta|\boldsymbol{Y},\boldsymbol{X})$是损失函数, 不同模型的损失函数形式不同, 线性模型的损失函数为最小二乘函数、逻辑斯谛回归的损失函数为极大似然函数的负向变换.

- $p_{\lambda}(|\beta_j|)$为正则化项, 所谓正则化项, 也叫惩罚项, 描述的其实是模型的复杂度, 模型的复杂度越高, 过拟合的风险也就越大. 惩罚项的惩罚力度越大时, 模型的复杂度越低. 例如前面章节讲到的岭回归, 是在极小化损失函数的基础上通过对系数添加L_2范数惩罚来进行系数的连续收缩的, 它是 Hoerl 和Kennard[83] 于 1970 年提出的, 是统计学著名的系数收缩方法之一.

7.4 正则化方法

- 岭回归的目标函数是

$$\min_{\beta} L(\boldsymbol{\beta}) = \frac{1}{2n}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{\beta},$$

  - 其中第一项是残差平方和 SSE (线性模型的损失函数), 第二项是 L_2 正则化项, λ 为调节参数, $\lambda \geqslant 0$, 为非负数, 其作用是控制损失函数和正则化项对回归系数估计的相对影响程度, $\lambda = 0$ 时, 岭回归估计值与最小二乘估计值相同,λ 越大, 则为了使目标函数最小, 回归系数 β 的估计值就越接近于 0.

2. 范数

- 我们都知道, 函数与几何图形往往是有对应的关系, 这个很好想象, 特别是在三维以下的空间内, 函数是几何图像的数学概括, 而几何图像是函数的高度形象化. 但当函数与几何超出三维空间时, 就难以获得较好的想象, 于是就有了映射的概念, 进而引入范数的概念.
- 当有了范数的概念后, 就可以引出两个向量的距离的定义, 这个向量可以是任意维数的. 通过距离的定义, 进而可以讨论逼近程度, 从而讨论收敛性、求极限. 设 $\boldsymbol{a} = (a_1, \cdots, a_n)^{\mathrm{T}}$

7.4 正则化方法

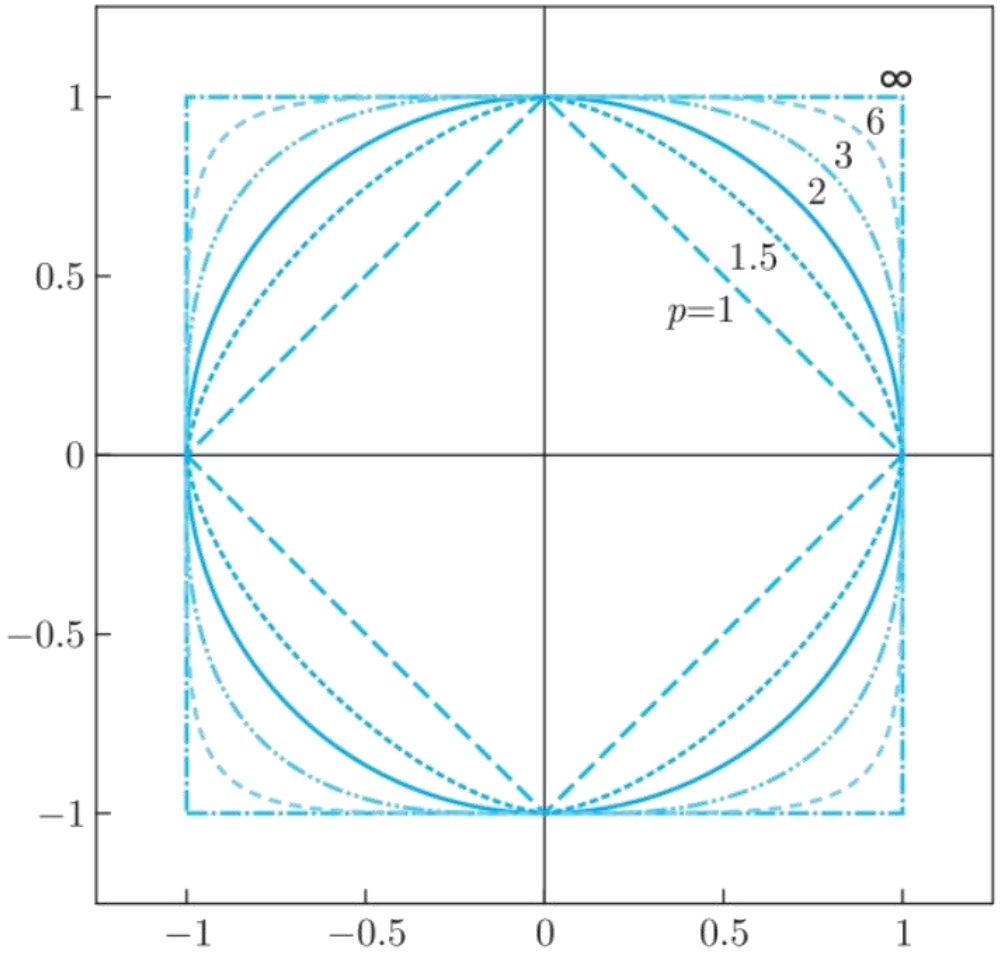
- L_0 范数: 即向量中非零元素的个数. 公式为

$$\|\boldsymbol{a}\|_0 = \sum_{i=1}^{n} I(a_i \neq 0).$$

- L_p 范数: 即向量元素绝对值的 p 次方和的 $1/p$ 次幂:

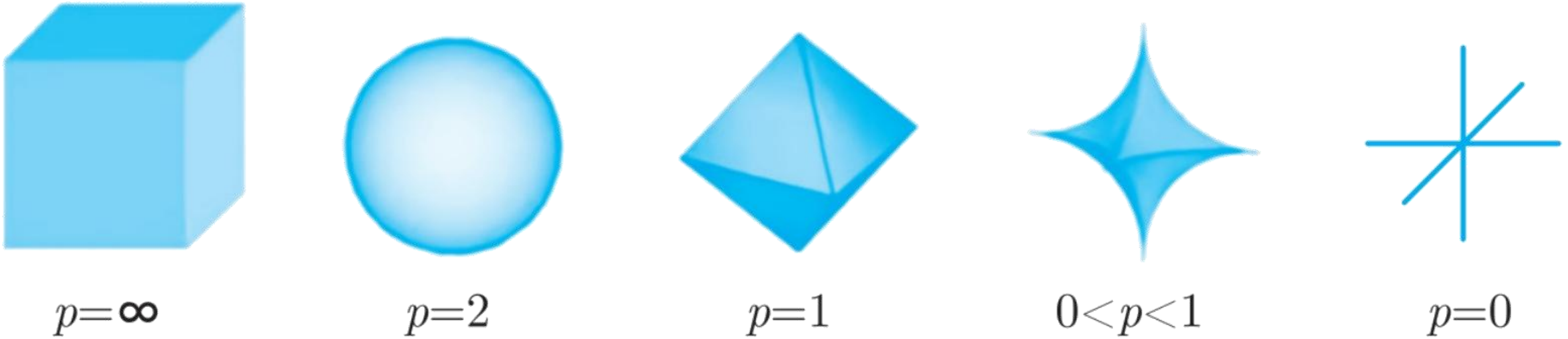
$$\|\boldsymbol{a}\|_p = \left(|a_1|^p + |a_2|^p + \cdots + |a_n|^p\right)^{1/p}.$$

- 图 7.2 展示了直角坐标系中 p 取不同的值时各个范数下单位向量终点的轨迹:



7.4 正则化方法

- 图 7.3 表示了p从无穷到0变化时, 三维空间中到原点的距离 (范数) 为1的点构成的图形的变化情况, 以 $p = 2$ 为例, 此时的范数也即欧氏距离, 空间中到原点的欧氏距离为 1 的点构成了一个球面:



- 其中, L_1 范数正则化、L_2 范数正则化都有助于降低过拟合风险, 我们在此主要运用 L_1 和 L_2 范数正则化, 或者二者相结合的方法. 对于线性回归模型, 使用L_1 正则化的模型叫做 Lasso 回归, 使用 L_2 正则化的模型叫做岭回归, 两者相结合的方法叫做弹性网.

7.4.1 Lasso 回归

- 与岭回归类似, Lasso 回归 (Least absolute shrinkage and selection opera_x0002_tor) 是在极小化残差平方和的基础上通过对系数添加 L_1 范数惩罚来收缩系数的, 1996 年由 Robert Tibshirani[130] 首次提出. 该方法是一种压缩估计, 当λ 充分大时, 可以把某些待估系数精确地收缩到零, 因而特别适用于参数个数的缩减与参数的选择.
- 该方法以岭回归为基础, 增加了稳定性. Lasso 的核心是稀疏性, 稀疏性的优势在于它可以解释拟合模型, 并且计算简单.
- Lasso 回归的目标函数是

$$\min_{\beta} L(\boldsymbol{\beta}) = \frac{1}{2n}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1,$$

- 岭回归和 Lasso 回归的主要区别就是在正则化项, 岭回归用的是 L_2 正则化, 而 Lasso 回归用的是 L_1 正则化. L_2 正则能够有效防止模型过拟合, 解决非满秩下求逆困难的问题; L_1 正则化最大的特点是能稀疏矩阵, 进行庞大预测变量数量下的预测变量选择.

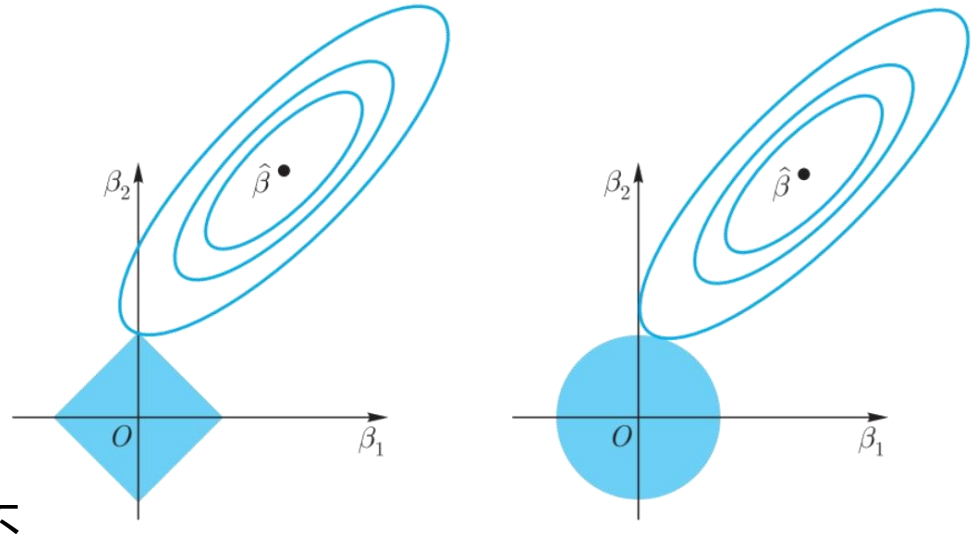
7.4.1 Lasso 回归

- 如果把二者目标函数最小化看成是有约束的极值问题, 那么岭回归可以表达为

$$\min_{\beta} \frac{1}{2n}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}), \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_2^2 \leq t.$$

- Lasso 回归可以表达为

$$\min_{\beta} \frac{1}{2n}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}), \quad \text{s.t.} \quad \|\boldsymbol{\beta}\|_1 \leq t.$$



- 因此 Lasso 回归和岭回归可以看作是两个同样目标函数但在不同的约束区域下求解的问题.

- 如图 7.4, 二维空间为例, 不添加正则项的目标函数可以用一圈圈等值线表示, 约束区域对应图中的阴影区域. 等值线和约束域的切点就是目标函数的最优解. 岭方法对应的约束域是圆, 其切点只会存在于圆周上, 不会与坐标轴相切, 则在任一维度上的取值都不为零, 因此没有稀疏; 对于 Lasso 方法, 其约束域是正方形, 会存在与坐标轴的切点, 使得部分回归系数变为零, 因此很容易产生稀疏的结果.

7.4.1 Lasso 回归

- 所以, Lasso 方法可以达到变量选择的效果, 将不显著的变量系数压缩至零, 在估计参数的同时实现了变量选择; 而岭方法虽然也对原本的系数进行了一定程度的压缩, 但是任一系数都不会压缩至零, 只有压缩功能, 没有选择功能, 最终模型保留了所有的变量. Lasso 回归的计算量将远远小于岭回归,但无论对于岭回归还是 Lasso 回归, 它们的本质都是通过调节 λ 来实现模型偏差和方差的平衡调整.

7.4.2 非凸惩罚函数回归——SCAD 和 MCP

1. SCAD

- Lasso 总是有偏估计, 这是因为 Lasso 的调节参数 λ 对所有参数一视同仁, 导致大的系数被过分压缩, 带来较大的估计偏差, 而且 Lasso 的结果具有不稳定性. 由 Fan 和 Li[84] 在 2001 年提出了 SCAD (smoothly clippedabsolute deviation penalty) 较好地弥补了 Lasso 的不足, 更进一步地, 该方法具有 Oracle 性质, 使预测效果与真实模型别无二致.

- Fan 和 Li 指出, 一个好的惩罚函数应该使得最后得到的估计量满足三个性质: **(1) 无偏性**. 即当未知参数的真实值很大时, 估计值应当近乎无偏.**(2) 稀疏性**. 即可以自动地将很小的估计系数压缩为零, 降低模型的复杂度.**(3) 连续性**. 即估计的系数应当是连续的, 保证模型预测的稳定性.

- 该方法也是在损失函数的基础上施加惩罚项, 例如基于二次损失的惩罚最小二乘目标函数可以表示为

$$L(\beta)=\frac{1}{2n}\|\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\|^2+\sum_{j=1}^{p}p_\lambda(|\beta_j|) \tag{7.4.1}$$

7.4.2 非凸惩罚函数回归——SCAD 和 MCP

- SCAD 的惩罚函数为

$$p_\lambda\left(|\beta_j|;\alpha\right)=\begin{cases}\lambda|\beta_j|, & 0\le|\beta_j|<\lambda,\\ -\dfrac{|\beta_j|^2-2\alpha\lambda|\beta_j|+\lambda^2}{2(\alpha-1)}, & \lambda\le|\beta_j|<\alpha\lambda,\\ (\alpha-1)\lambda^2/2, & \text{其他,}\end{cases}$$

  - 其中, α 和 λ 是两个调节参数.

- 考虑 $\beta_j\in[0,+\infty)$ 的情况, SCAD 惩罚函数的导数为

$$p'_\lambda\left(\beta_j;\alpha\right)=\begin{cases}\lambda, & \beta_j<\lambda,\\ \dfrac{\alpha\lambda-\beta_j}{\alpha-1}, & \lambda<\beta_j<\alpha\lambda,\\ 0, & \beta_j>\alpha\lambda.\end{cases}$$

7.4.2 非凸惩罚函数回归——SCAD 和 MCP

- 可以看出 SCAD 惩罚函数的导数在原点附近与 Lasso 相同, 离原点越远导数值越小, 直至为零, 也就是说, SCAD 方法对较大的参数施加较少的惩罚,当参数大于 $\alpha\lambda$, 不施加惩罚. 因此显著的变量更容易被选入模型.

2. MCP

- 2010 年, Zhang[85] 提出的 MCP 方法保留了 SCAD 的渐近无偏优点, 其惩罚函数同样是分段函数:

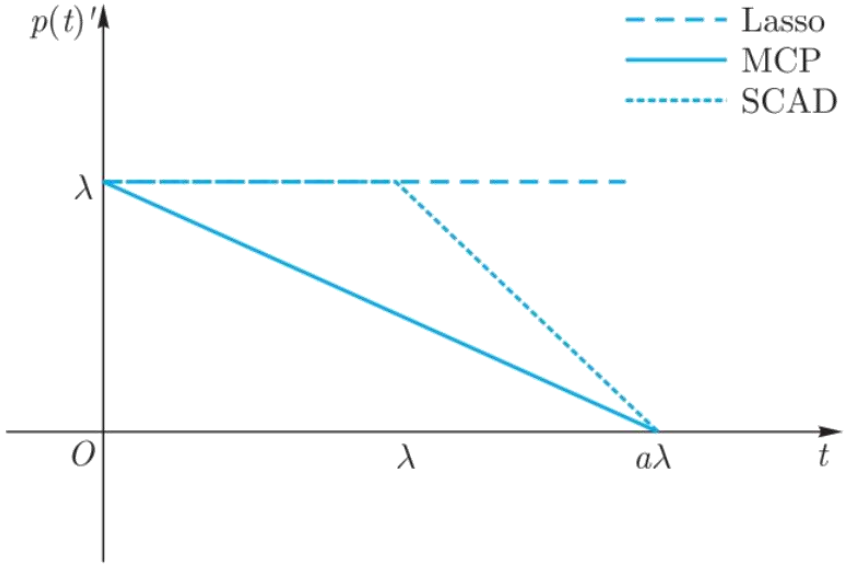
$$p_\lambda\left(|\beta_j|;\alpha\right)=\begin{cases}\lambda|\beta_j|-\dfrac{|\beta_j|^2}{2\alpha}, & |\beta_j|\le\alpha\lambda,\\ \dfrac{\alpha\lambda^2}{2}, & |\beta_j|>\alpha\lambda.\end{cases}$$

- 2010 年, Zhang[85] 提出的 MCP 方法保留了 SCAD 的渐近无偏优点, 其惩罚函数同样是分段函数:

$$p'_\lambda\left(|\beta_j|;\lambda,\alpha\right)=\begin{cases}\lambda-\dfrac{|\beta_j|}{\alpha}, & |\beta_j|\le\alpha\lambda,\\ 0, & |\beta_j|>\alpha\lambda.\end{cases}$$

7.4.2 非凸惩罚函数回归——SCAD 和 MCP

- 可以看出, 当 β_j 近似于零时, MCP 和 Lasso 的惩罚力度一致, 而随着 β_j从零开始增大, MCP 的惩罚力度逐渐缩减为零. 当 $|\beta_j| > \alpha\lambda$ 时, MCP 对大系数不施加惩罚, 因此 MCP 同 SCAD 类似, 也实现了回归系数的有差别惩罚, 实现了更精确的估计.
- 图 7.5 展示了回归系数变化时 Lasso、SCAD 和 MCP 的惩罚力度的变化情况.



7.4.3 群组变量选择方法

- 随着科技发展, 数据的种类与维数不断增加, 导致了回归建模时, 预测变量的群组结构成为了一种普遍现象. 比如, 变量间的相关性无法忽略时, 建模时应当处理共线性问题; 或者在医学上研究基因对疾病的影响时, 一般把同一基因下的变量作为一组变量来进行处理; 或者在处理多分类变量时, 经常使用的虚拟变量也是一类组变量, 在进行组变量选择时它们要当成一个整体去对待. 在这种情况下, 相关的选择问题就变成选择组而不仅仅是单个变量,群组变量选择模型及其算法成为当下高维数据建模的主要研究方向, 在此我们介绍两种代表性方法.

1. 弹性网 (elastic net)

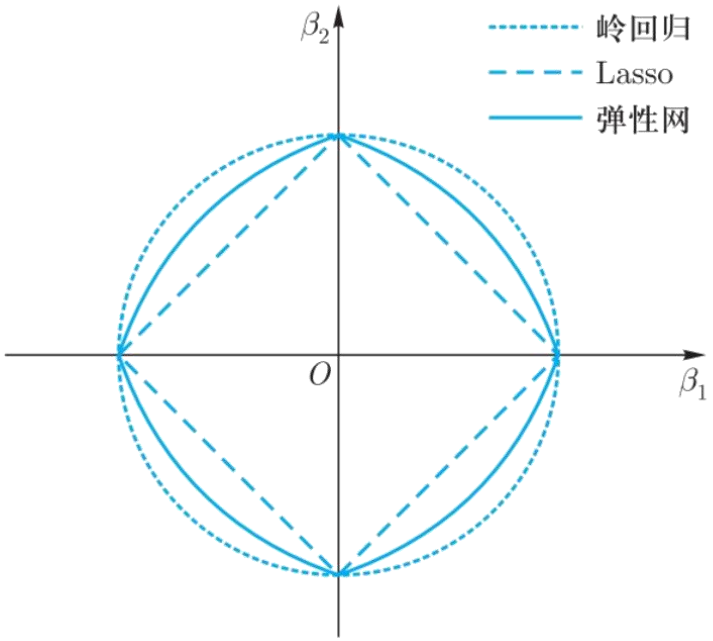
- 岭回归结果表明, 岭回归虽然一定程度上可以拟合模型, 但容易导致回归结果失真; Lasso 回归虽然能刻画模型代表的现实情况, 但是模型过于简单,不符合实际, 当多个变量和另一个变量高度相关的时候, Lasso 倾向于随机选择其中一个. 但如果这几个变量都是对模型重要的变量, 那么 Lasso 只能从多个相关强预测变量中选一个.

7.4.3 群组变量选择方法

- Zou 和 Hastie[86] 在 2005 年提出的弹性网结合了岭回归和 Lasso 算法,既能做到岭回归不能做的预测变量提取, 又能实现 Lasso 不能做的预测变量分组, 是最早解决共线性问题的变量选择方法, 能够将显著的强相关变量同时选入模型. 其目标函数为

$$\min_{\beta} \frac{1}{2n}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})^{\mathrm{T}}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})+\lambda\left[\alpha\|\boldsymbol{\beta}\|_1+(1-\alpha)\|\boldsymbol{\beta}\|_2^2\right]$$

- 显然, 这种方法是利用 $L_1 + L_2$ 范数的凸组合来实现约束的, 其中 L_1 正则化用于变量选择, L_2 正则化用于解决共线性问题. 除了加入调节参数 λ, 我们还引入参数 α 来调节 L_1 范数和 L_2 范数的平衡. 当 $\alpha = 0$ 时, 弹性网回归即为岭回归; 当 $\alpha = 1$ 时, 弹性网回归即为 Lasso 回归. 当 $\alpha = 0.5$ 时, 弹性网的几何结构如图 7.6 所示:



- 由图形可知, 弹性网回归在坐标轴上有尖角, 在各个象限内部呈弧形, 因此既具备 Lasso 的变量选择功能, 又具备岭回归的系数收缩功能.

7.4.3 群组变量选择方法

2. Group Lasso

- Yuan 和 Lin[87] 在 2006 年将 Lasso 方法推广到组群结构上面, 诞生了Group Lasso. 我们可以将所有变量分组, 然后在目标函数中加总每个组系数的 L_2 范数, 这样达到的效果就是可以将一整组的系数同时消成零, 即抹掉一整组的变量, 这种方法叫做 Group Lasso 算法. 其目标函数为

$$\min_{\beta} \frac{1}{2n}\left(\left\| \boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} \right\|_2^2 + \lambda \sum_{g=1}^{G} \left\| \sqrt{p_g}\, \beta_{I_g} \right\|_2 \right),$$

  - 其中, I_g 是 g 组的预测变量下标, $g = 1, 2, \cdots, G$, G 代表组的个数, λ 是调整参数, $\sqrt{p_g}$ 是每一组的加权, p_g 代表每一组变量个数. 其惩罚函数中, 在组内,是只具有压缩功能而没有选择功能的岭惩罚; 在组间是具有变量选择功能的 Bridge 惩罚, 因而整组变量会同时被选入或删除. 我们在此提到的 Bridge 惩罚, 即 $0 < p < 1$ 的 L_p 惩罚函数, 从惩罚函数形式来看, Lasso 对系数 β_j 的惩罚强度是固定的, 而 Bridge 对 β_j 的惩罚强度随 β_j 的增大而减小. 当 β_j 趋于 0 时, Bridge 的惩罚强度趋于无穷大, 使得其稀疏性较 Lasso 更加显著, 而当 β_j 很大时, Bridge 的惩罚强度变得很小, 使得其参数估计近似无偏.

7.4.3 群组变量选择方法

- 由于 Group Lasso 是一种通过对惩罚项的修改将 Lasso 方法在群组结构下的拓展, 因此与 Lasso 具备相同的缺点, 惩罚率不随组系数的大小而变化,因此会对较大的系数过分压缩, 导致估计偏差.

7.4.4 双层变量选择方法

- Group Lasso 方法由于只进行组变量的选择, 不能选择组内重要的单变量, 使得同组变量全被选择或全被删除, 因此有些显著单变量在进行变量选择时会随着组变量被淘汰掉, 从而导致模型误差过大, 精确度降低. 为了既考虑组的选择又考虑组内重要变量选择, 引入双层变量选择方法.

1. Group Lasso

- Huang et al.[88] 提出的 Group Brige 是一种可以满足组间、组内选择的双层变量选择方法, 是最早实现双层变量选择的方法. 其惩罚函数的形式为

$$P(\boldsymbol{\beta};\lambda,\gamma)=\lambda\sum_{j=1}^{W}p_j^{\gamma}\left\|\boldsymbol{\beta}^{(j)}\right\|_1^{\lambda}.$$

  - 其中 $0<\gamma<1$, p_j 是第 j 组变量所包含个数, $\boldsymbol{\beta}^{(j)}=\left(\beta_1^{(j)},\cdots,\beta_{pj}^{(j)}\right)$ 是第 j组预测变量的回归系数, 其中回归系数共分为 W 个组. 其惩罚函数由组内惩罚和组间惩罚函数复合而成, 其中, 组内选择的是 Lasso 惩罚函数, 组间选择的是 Bridge 惩罚函数, 从而实现了组间变量与组内变量的选择.

7.4.4 双层变量选择方法

2. 复合 MCP

- 复合 MCP 是 Huang et al. 2009 年提出的另一种复合惩罚类的变量选择方法[89], 其惩罚函数形式为

$$P(\boldsymbol{\beta};\lambda,\gamma)=\sum_{j=1}^{W}P_{\mathrm{MCP}}\left(\sum_{m=1}^{p_j}P_{\mathrm{MCP}}\left(\left|\beta_j^{(m)}\right|;\lambda,a\right);\lambda,b\right).$$

  - 其组内和组间均使用 MCP 惩罚, 均具备单变量选择的功能.

3. SGL

- 在双层变量选择方法中也存在另一种形式, 它的惩罚函数不是由两个具有单变量选择的功能惩罚函数复合而成的, 而是由单个变量惩罚和仅选择组变量惩罚的线性组合构成的函数. 这种惩罚方法叫做稀疏组惩罚.
- Simon et al. 2013 年提出的 SGL[90] (sparse group penalty) 就属于此类方法, 其惩罚函数形式为

$$P_{\mathrm{SGL}}(\boldsymbol{\beta};\lambda,\gamma)=\lambda_1\sum_{j=1}^{W}\left\|\boldsymbol{\beta}^{(j)}\right\|_2+\lambda_2\|\boldsymbol{\beta}\|_1.$$

7.4.4 双层变量选择方法

- 其中第一项是选择重要组变量, 选择的是 Group Lasso 方法的惩罚函数; 第二个惩罚是选择单个变量, 用的是 Lasso 方法的惩罚函数, 其中 λ_1、λ_2 分别为作用在单个变量和组变量上的调整参数.

4. Adaptive SGL

- SGL 相对复合函数型方法计算简单, 但它对于所有系数及组系数的惩罚力度相同, 过度压缩大系数, 引起估计偏差. 基于此 Fang et al. 2014 年提出了更一般化的 Adaptive SGL[91], 其惩罚函数形式为

$$P_{\text{adSGL}}\left(\boldsymbol{\beta};\lambda_1,\lambda_2\right)=\lambda_1\sum_{j=1}^{W}\omega_j\left\|\boldsymbol{\beta}^{(j)}\right\|_2+\lambda_2\sum_{j=1}^{W}\xi^{(j)\mathrm{T}}\left|\boldsymbol{\beta}^{(j)}\right|.$$

- 它通过引入权重 $\boldsymbol{\xi}=(\xi^{(1)\mathrm{T}},\cdots,\xi^{(W)\mathrm{T}})$ 和 $\boldsymbol{\omega}=\left(\omega_1,\cdots,\omega_W\right)$ 分别对单个系数和组系数施加不同程度的惩罚, 权重依据样本数据而定, 系数的真实值越大, 给予权重越小, 惩罚力度越小, 增进了预测精度.

7.5　保形预测实践

实践代码