机器学习（L06）：正则化线性回归

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

1 岭回归（Ridge Regression）

我们在之前的线性回归问题中，使用正规方程得到参数的值。

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} \tag{1}$$

当 $(\mathbf{X}^\top \mathbf{X})$ 不可逆时，便无法使用正规方程法。一种经典的解决方法是在对角线上添加一个的正数。

$$\widehat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y} \tag{2}$$

其中 $\lambda > 0$。我们可以证明对称阵 $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ 是正定矩阵。根据正定矩阵的性质，这个矩阵满秩，因此也是可逆矩阵。

证明. 对于任意非零向量 $\mathbf{v}$，

$$\begin{aligned} &\mathbf{v}^\top(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\mathbf{v} \\ =&\mathbf{v}^\top \mathbf{X}^\top \mathbf{X}\mathbf{v} + \lambda \mathbf{v}^\top \mathbf{I}\mathbf{v} \\ =&\|\mathbf{X}\mathbf{v}\|_2^2 + \lambda \|\mathbf{v}\|_2^2 \\ >&0 \end{aligned} \tag{3}$$

所以矩阵 $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ 为正定矩阵。

我们可以发现，$\widehat{\mathbf{w}}$ 这个形式正好是下面这个最优化问题的解

$$\widehat{\mathbf{w}}^{\text{ridge}} = \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^\top \mathbf{w} \tag{4}$$

通过导数等于零可以解出最小值，类似正规方程的方法。使用拉格朗日乘数法，可以把这个最优化问题可以等价为

$$\begin{aligned} &\widehat{\mathbf{w}}^{\text{ridge}} = \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\mathbf{w})^\top (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &\text{subject to} \sum_{j=1}^{p} \mathbf{w}_j^2 \le s^2 \end{aligned} \tag{5}$$

通过添加这个限制条件，可以防止过拟合。我们把这种最小二乘法加上 L_2 范数正则化的方法称之为岭回归。从几何角度看，画出岭回归的示意图，约束条件使参数向量始终在圆内，参数 s 便是圆的半径，然后再寻找使目标函数最小的值。原先的全局最优解在 $\hat{\theta}$ 处，而正则化后取到的最优解在红叉位置，这样就可以在一定程度上防止过拟合。

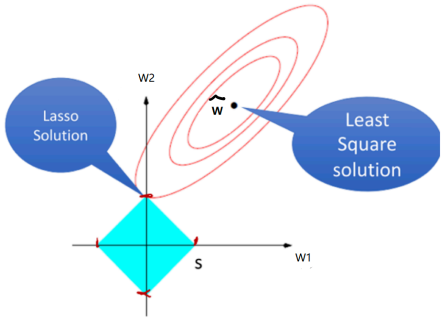

Figure 1: 岭回归示意图

我们对比一下使用最小二乘法得到的系数 $\widehat{\mathbf{w}}^{\text{OLS}}$ 和用岭回归得到的系数 $\widehat{\mathbf{w}}^{\text{ridge}}$，可以发现 $\widehat{\mathbf{w}}^{\text{ridge}}$ 一般都会比 $\widehat{\mathbf{w}}^{\text{OLS}}$ 小，因此岭回归可以达到参数收缩的效果。例如，当 $\mathbf{X}^\top \mathbf{X} = \mathbf{I}$ 时，最小二乘法得到的参数 $\widehat{\mathbf{w}}^{\text{OLS}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y} = \mathbf{X}^\top \mathbf{y}$，岭回归得到的参数 $\widehat{\mathbf{w}}^{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{y} = \frac{1}{1+\lambda}\mathbf{X}^\top \mathbf{y} = \frac{1}{1+\lambda}\widehat{\mathbf{w}}^{\text{OLS}}$，不难看出，$\widehat{\mathbf{w}}^{\text{ridge}}$ 是 $\widehat{\mathbf{w}}^{\text{OLS}}$ 的 $\frac{1}{1+\lambda}$ 倍。

之前讲到的式4和式5，是岭回归完全等价的两种形式，那么参数 λ 和 s 之间必然存在着某种联系。还是从特殊例子的角度看，使用拉格朗日乘数法求得岭回归的最优解，满足 $\lambda(\sum_{j=1}^{p}(\widehat{\mathbf{w}}^{\text{ridge}})_j^2 - s^2) = 0$。因为 $\lambda > 0$，所以有 $s^2 = (\sum_{j=1}^{p}\widehat{\mathbf{w}}^{\text{ridge}})_j^2$。当 $\mathbf{X}^\top\mathbf{X} = \mathbf{I}$ 时，根据之前结论，可以得到 $s^2 = \frac{1}{(1+\lambda)^2}\sum_{j=1}^{p}(\widehat{\mathbf{w}}^{\text{OLS}})_j^2$，因此我们可以推出 s 和 λ 存在如下关系

$$s^2 \propto \frac{1}{(1+\lambda)^2} \tag{6}$$

得到 s 和 λ 的关系后，可以轻易发现 λ 与我们所求参数 $\mathbf{w}$ 的关系

- 当 $\lambda = 0$ 时，我们得到的就是最小二乘法求得的解 $\widehat{\mathbf{w}}^{\text{OLS}}$
- 当 $\lambda > 0$ 时，$\widehat{\mathbf{w}}^{\text{ridge}}$ 是 $\widehat{\mathbf{w}}^{\text{OLS}}$ 的收缩
- 当 $\lambda \to +\infty$ 时，$\widehat{\mathbf{w}}^{\text{ridge}} \to 0$

从几何角度看，当 $\lambda \to +\infty$ 时，圆半径 $s \to 0$，所以 $\widehat{\mathbf{w}}^{\text{ridge}}$ 也会趋近与 0；

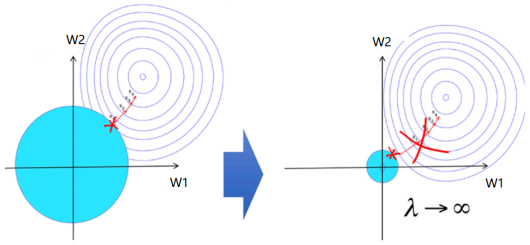

Figure 2: 当 $\lambda \to +\infty$ 时，对岭回归的影响

当 $\lambda \to 0$ 时，$s \to +\infty$，因此最小二乘法的最优解也会被包含在约束条件内，那么得到的最优解便是 $\widehat{\mathbf{w}}^{\text{OLS}}$。

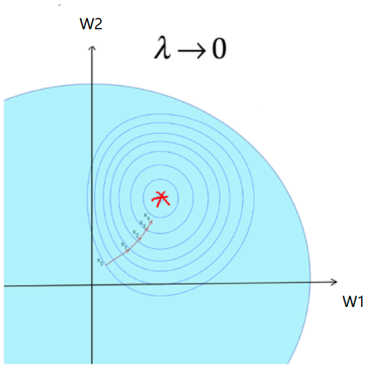

Figure 3: 当 $\lambda \to 0$ 时，对岭回归的影响

2 套索回归（Lasso Regression）

套索回归是类似于岭回归的收缩方法，但是以非线性方式作用于结果 $\mathbf{y}$。岭回归是最小二乘法加上 L_2 范数的约束，而套索回归是最小二乘法加上 L_1 范数的约束。它的定义式为：

$$\begin{aligned} &\widehat{\mathbf{w}}^{\text{lasso}} = \underset{\mathbf{w}}{\text{argmin}}(\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) \\ &\text{subject to} \sum_{j=1}^{p}|\mathbf{w}_j| \le s \end{aligned} \tag{7}$$

另一种形式为

$$\widehat{\mathbf{w}}^{\text{lasso}} = (\mathbf{y}-\mathbf{X}\mathbf{w})^\top(\mathbf{y}-\mathbf{X}\mathbf{w}) + \lambda\sum_{j=1}^{p}|\mathbf{w}_j| \tag{8}$$

套索回归的示意图如图4所示。套索回归的约束区域变为一个正方形，其方程为 $|\mathbf{w}_1| + |\mathbf{w}_2| = s$。在这个示意图中，注意到套索回归的解中 $\mathbf{w}_1 = 0$，从而消除了 $\mathbf{x}_1$ 的影响。这种使部分系数等于 0 的效果也就是稀疏性。

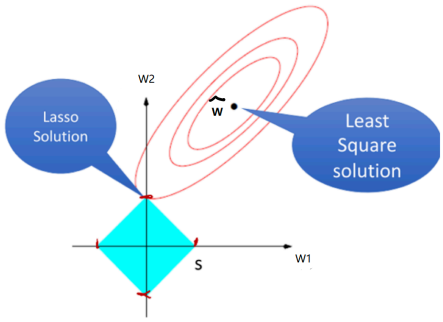

Figure 4: 套索回归示意图

由于套索回归的这种性质，可以通过调整参数来使部分系数等于 0，因此可以用于隐式特征选择。即当特征数 p 过大时，若使用岭回归性能会很低，而套索回归可以使一部分特征的系数等于 0，从而消除这些特征的影响，减小运算量。

因为套索回归的目标函数中有 L_1 范数，因此梯度下降法等需要目标函数处处可微的优化方法无法使用。我们这里介绍一种可以解决这种问题的方法——近端梯度法。在介绍近端梯度法前，首先引入近端算子的概念

定义：近端算子

对于凸函数 $f:\mathbb{R}^n \mapsto R\cup\{+\infty\}$，其近端算子为

$$\mathbf{prox}_{\lambda f}(\mathbf{v}) = \underset{\mathbf{x}}{\operatorname{argmin}}(f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x}-\mathbf{v}\|_2^2) \tag{9}$$

其中 $\lambda > 0$。

函数 f 可以是不可微的。近端算子可以理解为对于给定的点 $\mathbf{v}$，找到其最优的点 $\mathbf{x} = \mathbf{prox}_{\lambda f}(\mathbf{v})$，使得 $f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x}-\mathbf{v}\|_2^2$ 最小，这个解是肯定存在的。也就是说我们去求一个点 $\mathbf{x}$ 不仅使得不可微函数 $f(\mathbf{x})$ 的函数值足够小，而且还接近原不可微点 $\mathbf{v}$。

接下来开始介绍近端梯度法。近端梯度法一般用于解决此类优化问题：

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}) + g(\mathbf{x}) \tag{10}$$

其中 f 是光滑的凸函数，g 是不可微的凸函数。它的迭代公式为

$$\mathbf{x}^{k+1} = \mathbf{prox}_{\lambda^k g}(\mathbf{x}^k - \lambda^k \nabla f(\mathbf{x}^k)) \tag{11}$$

步长 λ 有两种设置方式，一种是设为常数 $\lambda = \frac{1}{L}$，L 为 ∇f 的利普希茨连续常数，但是这个常数通常不知道；另一种是用线性搜索，即通过迭代 $\lambda \leftarrow \beta\lambda$，记

$$\mathbf{z} = \mathbf{prox}_{\lambda^k g}(\mathbf{x}^k - \lambda \nabla f(\mathbf{x}^k)) \tag{12}$$

则迭代的终止条件为

$$f(\mathbf{z}) \le \widehat{f}_\lambda(\mathbf{z}, \mathbf{x}^k) \tag{13}$$

其中 $0 < \beta < 1$，参数 β 通常设为 $\frac{1}{2}$，函数 $\widehat{f}_\lambda(\mathbf{x},\mathbf{y})$ 为

$$\widehat{f}_\lambda(\mathbf{x},\mathbf{y}) = f(\mathbf{y}) + \nabla f(\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) + \frac{1}{2\lambda}\|\mathbf{x}-\mathbf{y}\|_2^2 \tag{14}$$

当使用近端梯度法解套索回归时，有 $f(\mathbf{w}) = \frac{1}{2}\|\mathbf{y}-\mathbf{X}\mathbf{w}\|_2^2$，$g(\mathbf{w}) = \gamma\|\mathbf{w}\|_1$，那么 $\nabla f(\mathbf{w}) = \mathbf{X}^\top(\mathbf{X}\mathbf{w}-\mathbf{y})$，$g(\mathbf{w})$ 的近端算子为 $\mathbf{prox}_{\lambda g}(\mathbf{w}) = S_{\gamma\lambda}(\mathbf{w})$，其中 S 为软阈值函数

$$[S_{\gamma\lambda}(\mathbf{x})]_i = \begin{cases} \mathbf{x_i} - \gamma\lambda, & \mathbf{x_i} > \gamma\lambda \\ 0, & |\mathbf{x_i}| \le \gamma\lambda \\ \mathbf{x_i} + \gamma\lambda, & \mathbf{x_i} < -\gamma\lambda \end{cases} \tag{15}$$

那么迭代公式便为

$$\mathbf{w}^{k+1} = \mathbf{prox}_{\lambda^k g}(\mathbf{w}^k - \lambda^k \mathbf{X}^\top(\mathbf{X}\mathbf{w}^k - \mathbf{y})) \tag{16}$$

Algorithm 1: 近端梯度法解套索回归

输入：样本矩阵 $\mathbf{X}$，样本对应标签 $\mathbf{y}$，步长 λ，正则化参数 γ，线性搜索参数 β，最大迭代次数 T，初始系数向量 $\mathbf{w}$;
for $i \leftarrow 1$ *to* T **do**
  while *True* **do**
    $z = \mathbf{prox}_{\lambda g}(\mathbf{w} - \lambda \mathbf{X}^{\top}(\mathbf{X}\mathbf{w} - \mathbf{y}))$;
    if $f(\mathbf{z}) \leq \widehat{f}_{\lambda}(\mathbf{z}, \mathbf{w})$ **then**
      break;
    end
    $\lambda = \beta\lambda$;
  end
  $\mathbf{w} = \mathbf{z}$;
end
输出：预测系数 $\mathbf{w}$

编程实现

```python
import numpy as np

# 求MSE
def mse(X, Y, x):
  prediction = X @ x
  error = np.linalg.norm(prediction - Y)
  MSE = error**2 / Y.size
  return MSE

n = 100 #训练样本个数
m = 50  #测试样本个数
p = 8   #特征数
X_train = np.random.uniform(size=(n, p))
X_test = np.random.uniform(size=(m, p))
theta0 = np.array([1, -4, 0.3, -20, 0.001, 7, -0.04, 57])
Y_train = X_train @ theta0 + 0.1*np.random.normal(size=(n, ))
Y_test = X_test @ theta0 + 0.1 * np.random.normal(size=(m, ))

# 岭回归
alpha = 1 #正则化参数
theta_ridge = np.linalg.inv(X_train.T @ X_train + alpha * np.identity(p)) @ X_train.T @ Y_train
print(theta_ridge)
mse_ridge = mse(X_test, Y_test, theta_ridge)
print("MSE of Ridge regression: ", mse_ridge)

# Output:
# [  0.8602228   -3.88557529   1.26550861 -16.91530975
#    0.07884014   7.74470326   0.26014671  51.56092402]
# MSE of Ridge regression:  4.367606710236091


# 近端梯度解套索回归
def f(A, b, x):
  return 0.5 * np.linalg.norm(A @ x - b)**2

def gradient_f(A, b, x):
  return A.T @ (A @ x - b)

# L1范数的近端算子，也就是软阈值函数
def s(l, x):
  return np.sign(x) * np.maximum(0, np.abs(x) - l)

theta_lasso = np.random.uniform(size=8)
t = 1 #步长
gamma = 1 #正则化参数
beta = 0.5 #线性搜索参数
max_iters = 10000 #最大迭代次数

for i in range(max_iters):
  ptheta = theta_lasso
  # 线性搜索求步长
  while True:
    z = s(gamma * t, ptheta - t * gradient_f(X_train, Y_train, ptheta))
```

```
    if f(X_train, Y_train, z)<=f(X_train, Y_train, ptheta) \
      + gradient_f(X_train, Y_train, ptheta) @ (z - ptheta) \
      + np.linalg.norm(z - ptheta)**2/2/t:
      break;
    t = t * beta

  theta_lasso = z

print(theta_lasso)
mse_lasso = mse(X_test, Y_test, theta_lasso)
print("MSE of Lasso regression: ", mse_lasso)
# Output:
# [  0.8629727   -3.91535179   0.28165264 -19.84917905   0.
#    6.91438783  -0.          56.93411617]
# MSE of Lasso regression:  0.016039052956306046
```

3 弹性网络（Elastic Net）

Lasso 能够很好地完成变量选择的要求，然而在一些情况下，Lasso 仍然有局限性。当变量数量大于样本数量（$p > n$）时，Lasso 最多只能从 p 个变量中选出 n 个变量。也就是说能选择出的变量数量会受到样本数量的限制。另一方面，在实际情况中可能会出现一组变量之间相关性很高，从而导致这一组对应的系数基本相同。Lasso 只会独立地关注每一个变量，因此无法获得这种“群组效应”。为了解决这些问题，弹性网络同时使用了 ℓ_1 和 ℓ_2 范数。弹性网络模型的形式为

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2 \tag{17}$$

其中 ℓ_1 范数的作用和在 Lasso 中一样，是为了使得系数变得稀疏。弹性网络模型在 Lasso 的基础上添加了一个 ℓ_2 范数的平方。这个多添加的一项使得弹性网络的变量选择不再受到样本数量影响，同时还能获得群组效应。

虽然公式17中有两个参数 λ_1 和 λ_2，但是如果我们假设 $\alpha = \dfrac{\lambda_1}{\lambda_1 + \lambda_2}$，弹性网络的形式就变为

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \alpha||\mathbf{w}||_1 + (1-\alpha)||\mathbf{w}||_2^2 \tag{18}$$

α 的取值是在 ℓ_1 范数和 ℓ_2 范数之间的权衡。特别地，当 $\alpha = 1$ 时，弹性网络退化为套索回归，而 $\alpha = 0$ 时退化为岭回归。图5展示当 $\alpha = 0.5$ 时，弹性网络正则项对应的范数球（norm ball）。弹性网络兼具岭回归和套索回归的特性。

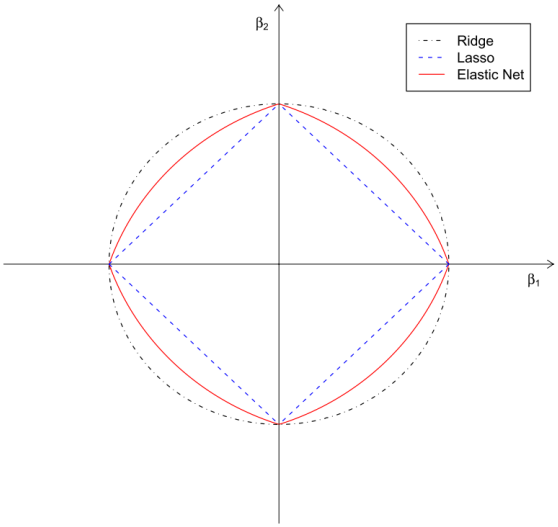

Figure 5: 弹性网络正则项范数球示意

下面我们来具体分析一下弹性网络的相关性质。

3.1 群组效应

群组效应指的是一个回归模型预测的系数 $\mathbf{w}$，对于两个高度相关的变量 $i, j \in \{1, 2, \cdots, p\}$，对应系数会非常接近。在极端的情况下，如果两个变量完全相同，对应的系数会完全相等。一个包含有正则项的线性回归模型的通用形式为

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \ ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda J(\mathbf{w}) \tag{19}$$

正则函数应该满足对于 $\boldsymbol{\theta} \neq 0$ 有 $J(\boldsymbol{\theta}) > 0$。对于这样一个通用的公式19，我们有以下的定理：

定理 3.1. 假设 $i, j \in \{1, 2, \cdots, p\}$ 表示两个变量，则

(1) 如果 $J(\cdot)$ 是一个强凸函数，那么 $\widehat{\boldsymbol{\theta}}_i = \widehat{\boldsymbol{\theta}}_j,\ \forall \lambda > 0.$

(2) 如果 $J(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1$，那么 $\widehat{\boldsymbol{\theta}}_i\widehat{\boldsymbol{\theta}}_j \leq 0$ 并且 $J(\boldsymbol{\theta})$ 还有另外一个最优解 $\widehat{\boldsymbol{\theta}}^*$。

我们先来解释什么是强凸函数。

定义：强凸函数

对于一个函数 $f(\cdot)$ 和 $\mu > 0$，如果满足

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top(\mathbf{y} - \mathbf{x}) + \frac{\mu}{2}||\mathbf{y} - \mathbf{x}||_2^2 \tag{20}$$

那么我们就称函数 f 为强凸函数。

在一些情况下，如果函数本身并不是强凸的，那么可以通过加上一个二次项 $||\cdot||_2^2$ 使函数获得强凸性。弹性网络就是如此，通过加上一个二次项 $||\mathbf{w}||_2^2$，弹性网络的正则项就变成了强凸函数。所以弹性网络的正则项正好满足定理3.1的第一条。反观套索回归（定理3.1第二条），它并不能获得群组效应。套索回归的解甚至都不是唯一的。图6是套索回归和弹性网络对于 $p = 6$ 的预测系数的对比，显然弹性网络能够获得群组效应而套索回归并不具备这种能力。

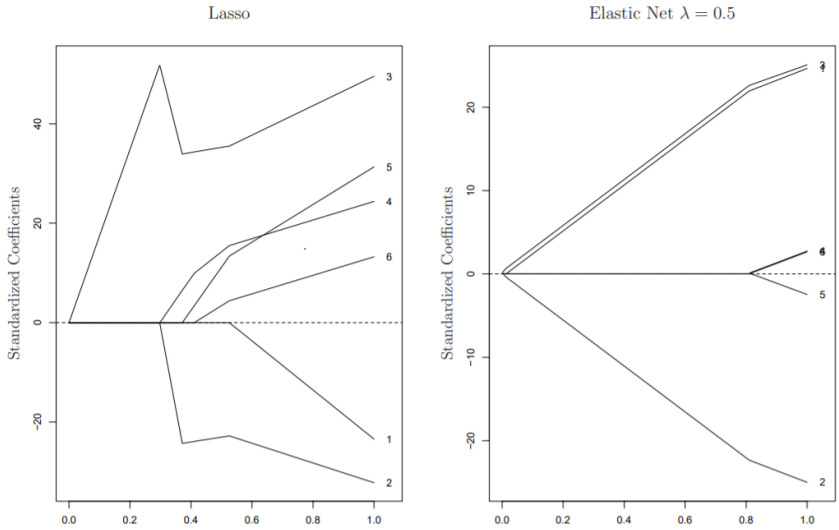

Figure 6: 套索回归与弹性网络求的系数的对比

3.2 编程实现

弹性网络的求解比岭回归和套索回归都要复杂，不在本课的讨论范围之内。在 Python 中，scikit-learn 库提供了基本的机器学习算法，我们可以直接用来解决弹性网络。

```python
import numpy as np
from sklearn.linear_model import ElasticNet
from sklearn.metrics import mean_squared_error

# 生成稀疏数据
p = 10
n = 120
X = np.random.uniform(size = (n, p))
theta = np.random.random(size = (p,)) * 10
theta[[0, 2, 5, 8]] = 0
Y = X @ theta
# 将数据划分为训练集和测试集
train_X = X[:100, :]
train_Y = Y[:100]
test_X = X[100:, :]
test_Y = Y[100:]

# 用弹性网络预测系数
model = ElasticNet(random_state=0)
model.fit(train_X, train_Y)

# 计算MSE
train_pred = model.predict(train_X)
train_MSE = mean_squared_error(train_pred, train_Y)
print('训练集上的MSE为：{}'.format(train_MSE))
```

```
test_pred = model.predict(test_X)
test_MSE = mean_squared_error(test_pred, test_Y)
print('测试集上的MSE为：{}'.format(test_MSE))

# Output:
# 训练集上的MSE为：19.367750668410633
# 测试集上的MSE为：27.4472340302309951
```

4 收缩估计器（Shrinkage Estimator）

岭回归（Ridge Regression）和套索回归（Lasso Regression）都是以下通用形式的两个特例

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \lambda \sum_{j=1}^{p} ||\mathbf{w}_j||_q^q \tag{21}$$

根据 q 选值的不同，$\sum_{j=1}^{p} ||\mathbf{w}_j||_q^q$ 的轮廓如图7所示，当 $q \geq 1$ 时，正则项是凸函数。特别地，$q = 1$ 时收缩估计器就是套索回归，$q = 2$ 时收缩估计器就是岭回归。一般来说，我们都会选择 $q \geq 1$ 从而优化凸函数。除此之外，虽然 q 可以选任意值，但是在实际使用中我们通常都会选择 $q = 1$ 或 $q = 2$。因为过大的 q 会对计算造成很大负担并且对预测结果不会有很大提升。从图8中我们可以感受到，$q < 2$ 的范数趋向于获得稀疏的系数，而 $q > 2$ 的范数倾向于使预测系数对于每个变量相同。

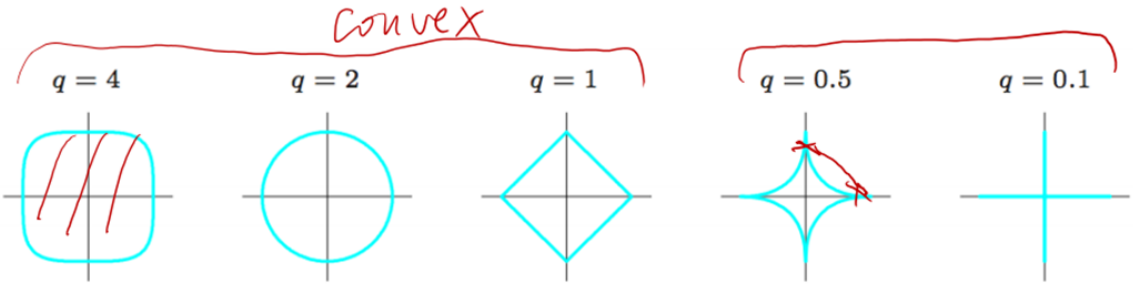

Figure 7: 不同 q 值对应的单位范数球轮廓

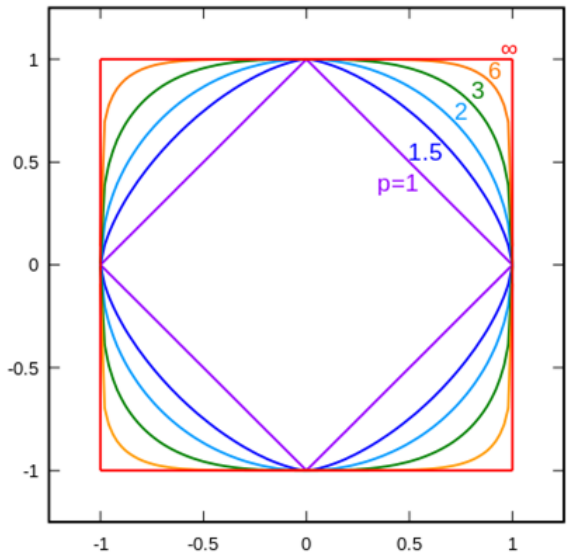

Figure 8: 不同范数对应的单位范数球轮廓

5 参数选择

我们之前介绍的三种带有正则项的方法中都具有需要调整的参数，比如在岭回归中的参数 λ。显然，这些参数的值会影响模型的效果。因此，在实际使用这些模型的时候，我们需要谨慎选择这些参数的值。我们可以使用之前介绍的方法，比如 K-层交叉检验，来对参数值进行选择。

图9和图10分别表示在岭回归和套索回归中使用不同的参数 λ 所对应的预测系数的不同。从图中我们可以发现，随着 λ 的增加，岭回归中系数的值虽然在一直收缩，但是在过程中没有收敛到 0，因此岭回归并不能很好地获得稀疏性。反观套索回归，随着 λ 增加，会有一些系数的值变为 0，也就是说这些系数被套索回归认为是“无用”的而剔除了。

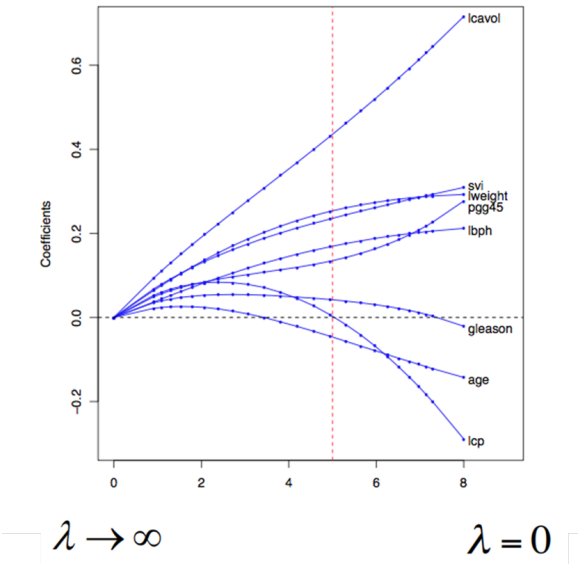

Figure 9: 岭回归中选择不同 λ 获得的系数

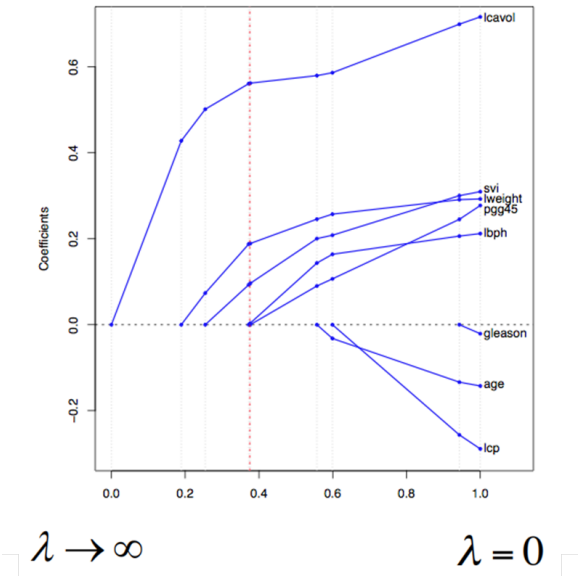

Figure 10: 套索回归中选择不同 λ 获得的系数

6 相关系数（Correlation Coefficient）

在统计上，我们通常用相关系数来衡量两个变量变化的**线性相关性**。相关系数有很多种不同的计算方法，但无论用什么方法相关系数都要满足几个要求：（1）相关系数值在 -1.0 至 1.0 之间；（2）相关系数值小于 0 时表示两个变量负相关，而大于 0 时表示两个变量正相关。假设我们用 ρ 来表示相关系数，图11表示了对于不同的相关系数值，两个变量之间的线性相关性。

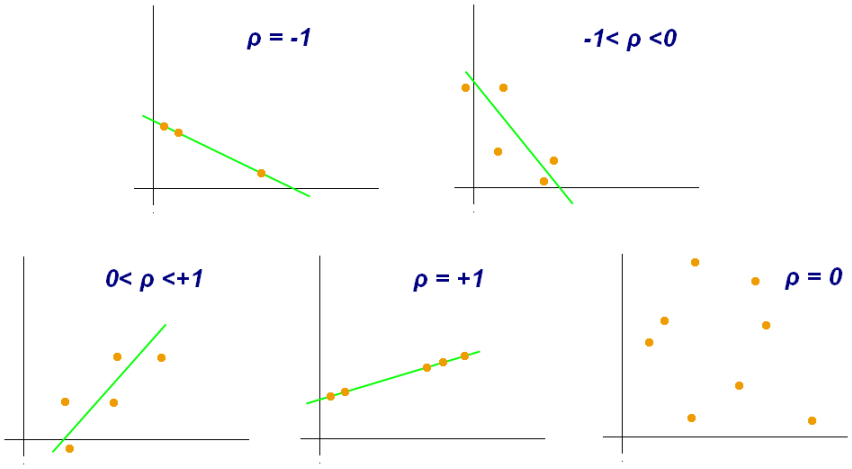

Figure 11: 不同相关系数值及对应变量间的线性相关性

下面我们介绍几个不同的计算相关系数的方法。

6.1 皮尔森相关系数

定义：皮尔森相关系数

对于两个随机变量 X, Y，他们之间的皮尔森相关系数计算方法为

$$\rho(\mathbf{X}, \mathbf{Y}) = \frac{\mathrm{cov}(\mathbf{X}, \mathbf{Y})}{\sigma_{\mathbf{X}} \sigma_{\mathbf{Y}}} \tag{22}$$

其中 cov 是协方差，而 $\sigma_{\mathbf{X}}$ 和 $\sigma_{\mathbf{Y}}$ 分别表示 $\mathbf{X}$ 和 $\mathbf{Y}$ 的标准差。

6.2 斯皮尔曼等级相关系数

与皮尔森相关系数不同，斯皮尔曼等级相关系数先将变量值转化为“等级”，然后利用等级来计算相关系数。一般来说，等级指的就是顺序。假设我们有两个变量 $\mathbf{X} = \{1, 2, 3, 4, 5\}$ 和 $\mathbf{Y} = \{3, 6, 10, 2, 5\}$。如果我们定义 $R_{\mathbf{X}}$ 和 $R_{\mathbf{Y}}$ 分别表示 $\mathbf{X}$ 和 $\mathbf{Y}$ 中每个值的顺序，那么可以得到 $R_{\mathbf{X}} = \{1, 2, 3, 4, 5\}$ 与 $R_{\mathbf{Y}} = \{2, 4, 5, 1, 3\}$。

定义：斯皮尔曼等级相关系数

对于两个变量 $\mathbf{X}, \mathbf{Y}$，他们之间的斯皮尔曼等级相关系数的计算方法为

$$\rho(\mathbf{X}, \mathbf{Y}) = \frac{\mathrm{cov}(R_{\mathbf{X}}, R_{\mathbf{Y}})}{\sigma_{R_{\mathbf{X}}} \sigma_{R_{\mathbf{Y}}}} \tag{23}$$

其中 cov 是协方差，而 $\sigma_{R_{\mathbf{X}}}$ 和 $\sigma_{R_{\mathbf{Y}}}$ 分别表示 $R_{\mathbf{X}}$ 和 $R_{\mathbf{Y}}$ 的标准差。

斯皮尔曼等级相关系数为 1 表示两个变量之间是单调相关的。如图12所示，$\mathbf{X}, \mathbf{Y}$ 之间是单调相关的而不是线性相关，使用皮尔森相关系数计算得到的 $\rho(\mathbf{X}, \mathbf{Y}) = 0.8$ 而用斯皮尔曼等级相关系数计算得到 $\rho(\mathbf{X}, \mathbf{Y}) = 1$。所以斯皮尔曼等级相关系数更适合用来分析单调相关而非严格线性相关的变量。

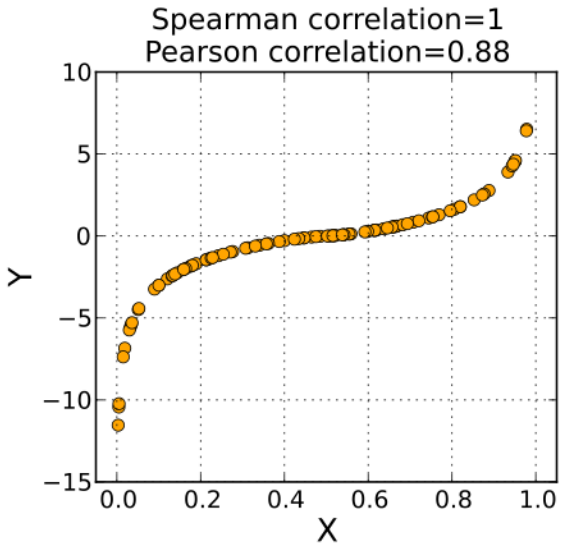

Figure 12: 皮尔森相关系数和斯皮尔曼等级相关系数对于单调相关的两个变量的分析结果

除此之外，斯皮尔曼等级相关系数对于离群点不敏感，因此比较适合分析噪声较大的数据。如图13所示，由于各种噪声的存在，数据中会存在一些少量的离群点（图中右侧的几个点）。斯皮尔曼相关系数会将变量值转化为等级，因此离群值对于数据分析的影响会被弱化。

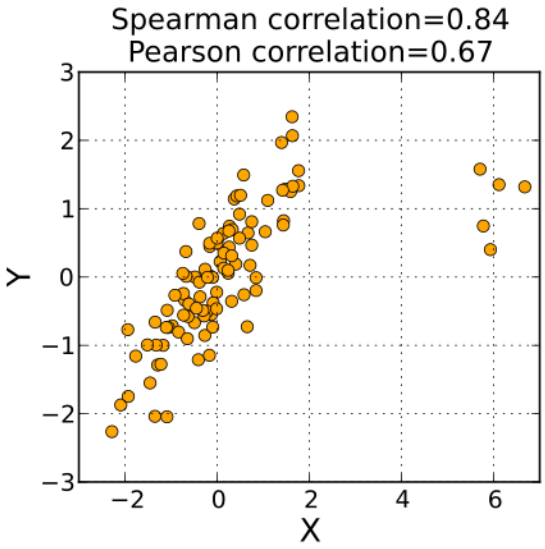

Figure 13: 皮尔森相关系数和斯皮尔曼等级相关系数对于存在离群值的两个变量的分析结果

6.3 编程实现

现在我们用 Python 来实现这两种相关系数的计算：

```python
import numpy as np
import matplotlib.pyplot as plt

# Pearson Correlation Coefficient
def pearsonCoeff(x, y):
    return np.cov(x,y)[0,1] / (np.std(x) * np.std(y))

# Spearman's Rank Correlation Coefficient
def spearmanCoeff(x, y):
    order_x = x.argsort()
    order_y = y.argsort()
    rank_x = order_x.argsort()
    rank_y = order_y.argsort()
    return np.cov(rank_x, rank_y)[0,1] / (np.std(rank_x) * np.std(rank_y))

# Tip: Notice that in numpy, calculating np.cov(a,b) will obtain a 2 x 2 matrix:
#     cov(a,a)   cov(a,b)
#     cov(b,a)   cov(b,b)
#Therefore, the np.cov(a,b)[0,1] is what we want.
```

现在我们生成一些数据并且来计算两种相关系数：

```python
# Generate 100 data points
X = np.random.uniform(low = 0.0, high = 2.0, size = (200,))
Y = 2 * X**2 + np.random.random((200,)) # Y = 2x⋯^2 + noise
pc = pearsonCoeff(X, Y)
sc = spearmanCoeff(X, Y)

plt.title('Pearson: %.3f \n Spearman: %.3f' % (pc, sc), fontsize = 20)
plt.scatter(X, Y)
plt.show()
```

计算结果如图14所示。

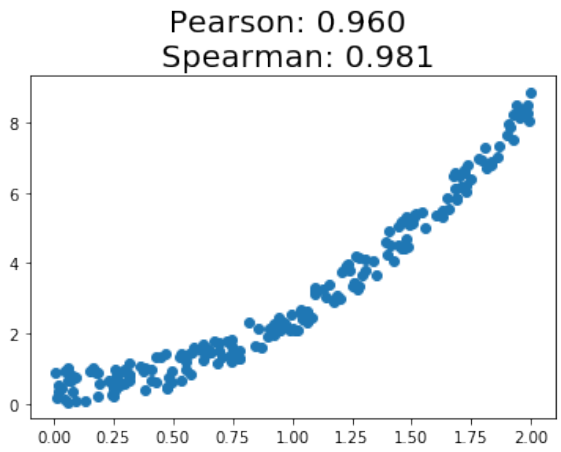

Figure 14: 相关系数计算结果

引用

[1] Zou, Hui, and Trevor Hastie. "Regularization and variable selection via the elastic net." Journal of the royal statistical society: series B (statistical methodology) 67.2 (2005): 301-320.

[2] Pearson Correlation Coefficient: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[3] Spearman's Rank Correlation Coefficient: https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

[4] Ridge Regression: https://en.wikipedia.org/wiki/Tikhonov_regularization

[5] Lasso Regression: https://en.wikipedia.org/wiki/Lasso_(statistics)