系统工程导论作业四——黑箱建模 2

彭程 2020011075

1. 试说明：病态线性回归问题中，显著性检验是否需要？

结论：

病态线性回归中需要显著性检验，且在自变量降维去线性之后进行检验。

需要进行检验：

显著性检验就是事先对总体（随机变量）的参数或总体分布形式做出一个假设，然后利用样本信息来判断这个假设（备择假设）是否合理，即判断总体的真实情况与原假设是否有显著性差异。在线性回归中体现为验证线性回归模型的有效性。病态线性回归问题也是利用样本信息来判断总体的参数，所以也需要显著性检验来确定总体分布形式假设是否合理，判断总体的真实情况与原假设是否有显著性差异，因此病态线性回归问题中也需要进行显著性检验。

降维之后检验：

病态回归问题中，由于自变量成线性相关或近似线性相关关系，所以自变量降维去线性之前，参数估计的误差可能会被严重放大，线性相关的变量的系数是不稳定的，因此即便是估计出来参数，意义也并不大，更没有必要进行显著性检验。而在自变量降维去线性之后，线性回归误差较小，因此应该在此时进行显著性检验。

2. 编程实现多元线性回归。

2.1 算法思路

1. 样本数据规范化，消除单位影响。

$$\bar{x}_i(t) = \frac{x_i(t) - e\left(x_i\right)}{\sqrt{\delta^2\left(x_i\right)}} \quad \forall i, t$$

其中, $e\left(x_i\right) = \frac{1}{N}\sum_{t=1}^{N} x_i(t)$ 为样本均值, $\delta^2\left(x_i\right) = \frac{1}{N-1}\sum_{t=1}^{N}\left(x_i(t) - e\left(x_i\right)\right)^2$ 为样本方差

$$\bar{y}(t) = \frac{y(t) - e(y)}{\sqrt{\delta^2(y)}} \quad \forall t$$

2. 根据阈值确定最优维度 m

对归一化的 XX^T 进行特征值分解, 得到 $XX^T = Q\Lambda Q^T$。按照特征值从大到小选取特征向量，直到相对逼近误差（末选取的特征值占特征值的比例）小于选取的阈值。

3. 求出降维后的矩阵。

将特征向量组成矩阵 $Q_m = [q(1)q(2)\cdots q(m)]$，从而降维后的矩阵为 $Z = Q_m^T X$。

4. 计算降维后的回归系数 d，从而确定降维前的回归系数 c, 恢复得到归一化前的回归系数

$$\hat{d} = \left(ZZ^T\right)^{-1} ZY^T$$
$$\hat{c} = Q_m\hat{d}$$

5. 进行显著性检验

对计算得到的回归系数进行显著性检验，注意其中 n 为降维之后的维度 m：

$$F = \frac{(N-n-1)ESS}{nRSS}$$

7. 求取置信区间

给定显著性水平 α, 对某一 x_0 , 相应的 y_0 将以 $1-\alpha$ 的概率落在置信区间:

$$\left(\hat{y}_0 - Z_{\alpha/2}S_\delta, \hat{y}_0 + Z_{\alpha/2}S_\delta\right)$$

其中 $Z_{\alpha/2}$ 是标准正态分布上 $\alpha/2$ 百分位点的值, 剩余均方差 $S_\delta = \sqrt{\frac{RSS}{N-n-1}}$。

2.2 实验结果

```
特征值从大到小依次为: [21.06874612 11.2273649   7.68243347  0.02145552]
降维后的维度为: 3
降维后的系数为: [ 0.66742279 -0.00376926 -0.25422719]
规范化后的系数为: [0.48232919 0.21752131 0.00671087 0.47968412]
原始系数为: [0.0729661  0.59856205 0.00187169 0.10548166]
原始偏移为: -9.1515
F检验结果为: F = 195.1165 > F_alpha = 4.3468  说明x与y存在线性关系
置信区间为: (y - 1.1557, y + 1.1557)
回归方程为:  y = -9.1515 + 0.0730X1 + 0.5986X2 + 0.0019X3 + 0.1055X4
```

回归方程: $y = -9.1515 + 0.0730x_1 + 0.5986x_2 + 0.0019x_3 + 0.1055x_4$

F 检验: $F = 195.1165 > 4.3468$, 因此 x, y 存在线性关系。

置信区间: $[y - 1.1557, y + 1.1557]$

2.3 代码

```python
import numpy as np
from scipy import stats

def linear_regression1(Y, X, alpha=0.05, error=0.01):
    """
    输入：N×N的矩阵X,1×N的矩阵Y
    输出：N×1的回归系数THETA
    功能：实现Y=c^T*X+b的多元线性回归，自适应病态线性回归
    """
    n, N = X.shape

    # 数据规范化
    x_mean = np.mean(X, axis=1, keepdims=True)
    delta_x = np.sqrt(
        np.sum(np.multiply(X - np.mean(X, axis=1, keepdims=True), X - np.mean(X, axis=1, keepdims=True)),
               axis=1,
               keepdims=True) / (N - 1))
    X_normal = np.divide(X - x_mean, delta_x)
```

```
    y_mean = np.mean(Y)
    delta_y = np.sqrt(np.sum(np.multiply(Y - y_mean, Y - y_mean)) / (N - 1))
    Y_normal = (Y - y_mean) / delta_y

    # 根据阈值确定最优维度M
    eigenvalue, eigenvector = np.linalg.eig(np.dot(X_normal, X_normal.T))
    eigen_index = np.argsort(-eigenvalue) # 返回排序的下标
    eigen_sum = np.sum(eigenvalue)
    for m in range(0, n):
        eigen_error = 0
        for i in range(0, m):
            eigen_error += eigenvalue[eigen_index[n - 1 - i]]
        eigen_error = eigen_error / eigen_sum
        if eigen_error > error:
            break
    m = n - m + 1 # 则前M项是要保留的

    # 计算降维后的回归系数D, 从而确定降维前的回归系数C_N,恢复得到归一化前的回归系数C
    Q = eigenvector[:, eigen_index[0: m]]
    Z = Q.T.dot(X_normal)
    d = np.linalg.inv(Z.dot(Z.T)).dot(Z.dot(Y_normal.T))
    c_n = Q.dot(d)
    c = (c_n.T / np.squeeze(delta_x) * delta_y).T
    b = y_mean - np.sum(c * np.squeeze(x_mean))
    print("特征值从大到小依次为: ", eigenvalue[eigen_index[:]])
    print("降维后的维度为: ", m)
    print("降维后的系数为: ", d)
    print("规范化后的系数为: ", c_n)
    print("原始系数为: ", c)
    print("原始偏移为: {:.4f}".format(b))

    # F检验操作
    Y_estimate = np.dot(c.T, X) + b
    ESS = np.dot((Y_estimate - y_mean), (Y_estimate - y_mean).T)
    RSS = np.dot((Y - Y_estimate), (Y - Y_estimate).T)
    F = ((N - m - 1) * ESS) / (m * RSS) # ESS自由度为 (N-M-1) , RSS自由度为M
    F_alpha = stats.f.isf(alpha, m, N - m - 1)
    if F > F_alpha:
        print("F检验结果为: F = {:.4f} > F_alpha = {:.4f}".format(F, F_alpha), " 说明x与y存在线性关系")
    elif F <= F_alpha:
        print("F检验结果为: ")
        print("F-value = {} <= F_alpha = {}".format(F, F_alpha))
        print("x与y不存在线性关系")
        return 0

    # 置信区间
    S_delta = np.sqrt(RSS / (N - m - 1))
    Z_alpha_div2 = stats.norm.isf(alpha / 2, 0, 1)
    interval = Z_alpha_div2 * S_delta
    print("置信区间为: (y - {:.4f}, y + {:.4f})".format(interval, interval))

    # 打印回归方程
    equation = "y = %.4f" % b
    for i in range(n):
        equation += " + %.4fX%d" % (c[i], i + 1)
    print("回归方程为: ", equation)
```

```
74
75      return
76
77
78  if __name__ == '__main__':
79      f = open('data.txt', 'r', encoding='utf-8')
80      # RAW_DATA = NP.ARRAY([LIST(MAP(FLOAT, I[:-1].SPLIT(' '))) FOR I IN F.READLINES()[1:]])
81      raw_data = np.array([i[:-1].split(' ') for i in f.readlines()[1:]], dtype=float)
82      x = raw_data[:, 1: 5]
83      y = raw_data[:, -1]
84      linear_regression1(y.T, x.T, 0.05, 0.01)
```