МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра теоретической и прикладной информатики

Лабораторная работа №5
по дисциплине «Статистические методы анализа данных»

Студенты ИВАНОВ ВЛАДИСЛАВ (92)

ОБЕРШТ ЕЛЕНА (93)

Вариант 5

Преподаватель ПОПОВ АЛЕКСАНДР АЛЕКСАНДРОВИЧ

Новосибирск, 2022

1 Постановка задачи

Сгенерировать экспериментальные данные, в которых в явном виде присутствует эффект мультиколлинеарности, используя регрессию на 8 факторах. Эффект мультиколлинеарности создают две пары факторов. Имеется разброс в масштабах факторов.

Рассчитать ряд показателей, характеризующих эффект мультиколлинеарности. Определить факторы, ответственные за возникновение эффекта мультиколлинеарности.

Построить ридж-оценки параметров при различных значениях параметра регуляризации. Выбрать оптимальное значение параметра регуляризации. Построить графики изменения квадрата евклидовой нормы оценок параметров и остаточной суммы квадратов от параметра регуляризации.

Провести оценивание модели регрессии по методу главных компонентов. Перейти к описанию в исходном пространстве факторов. Сравнить решение с ридж-оцениванием по смещению оценок и точности предсказания отклика.

2 Моделирование процесса

$$f(x) = (1, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)^T$$
$$\theta = (1, 1, 1, 1, 1, 1, 1, 1)^T$$
$$x_1 \in [-1, 1]$$
$$x_2 = 2x_1 + \tilde{\epsilon}$$
$$x_3 \in [-1, 1]$$
$$x_4 = 2x_3 + \tilde{\epsilon}$$
$$x_5 \in [-1, 1]$$
$$x_6 \in [-1, 1]$$
$$x_7 \in [-1, 1]$$
$$x_8 \in [-1, 1]$$
$$\epsilon \in N(0, \sigma^2)$$
$$\tilde{\epsilon} \in N(0, 0.01)$$
$$\sigma_i^2 = 0.05$$
$$n = 3000$$

3 Показатели, характеризующие мультиколлинеарность

3.1 Определитель информационной матрицы

$$det(X^T X) = 2.576795e + 20$$

3.2 Минимальное собственное число информационной матрицы

$$\lambda_{min}(X^TX) = 5.786125e - 02$$

Матрица обладает плохой обусловленностью.

3.3 Мера обусловленности матрицы по Нейману-Голдстейну

$$\lambda_{max}(X^TX)/\lambda_{min}(X^TX) = 8.920608e + 04$$

3.4 Максимальная парная сопряженность

Построим матрицу сопряженности:

$$\begin{pmatrix} 1 & r_{12} & \dots & r_{1m} \\ r_{21} & 1 & \dots & r_{2m} \\ \dots & \dots & \dots & \dots \\ r_{m1} & r_{m2} & \dots & 1 \end{pmatrix}$$

$$r_{ij} = cos(x_i, x_j)$$

```
[[ 1.       0.99996 -0.03563 -0.03547 -0.00511 -0.02919  0.03185 -0.04593]
 [ 0.99996  1.      -0.0355  -0.03534 -0.00515 -0.02906  0.03207 -0.04562]
 [-0.03563 -0.0355   1.       0.99996 -0.01341 -0.03165  0.01802  0.02747]
 [-0.03547 -0.03534  0.99996  1.      -0.01349 -0.03159  0.01807  0.02752]
 [-0.00511 -0.00515 -0.01341 -0.01349  1.      -0.00674 -0.00088  0.00429]
 [-0.02919 -0.02906 -0.03165 -0.03159 -0.00674  1.       0.00709 -0.00794]
 [ 0.03185  0.03207  0.01802  0.01807 -0.00088  0.00709  1.      -0.01119]
 [-0.04593 -0.04562  0.02747  0.02752  0.00429 -0.00794 -0.01119  1.     ]]
```

Показателем мультиколлинеарности является величина:

$$\max_{i,j} |r_{ij}| = 0.99996$$

3.5 Максимальная сопряженность

$$R_i^2 = 1 - \frac{1}{R_{ii}^{-1}}$$

```
R2 = [0.99992  0.99992  0.99992  0.99992  0.000382 0.002393 0.002196 0.004205]
```

Показателем мультиколлинеарности является величина:

$$\max_i |R_i| = 0.99992$$

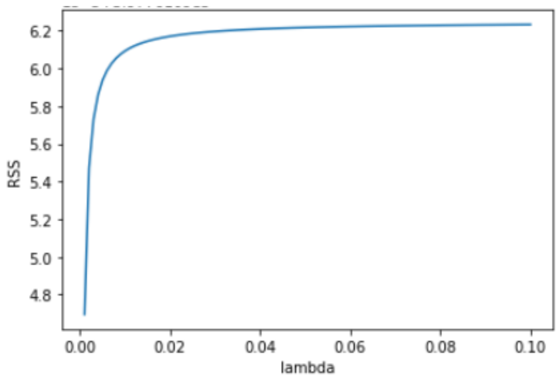
4 Ридж-оценка

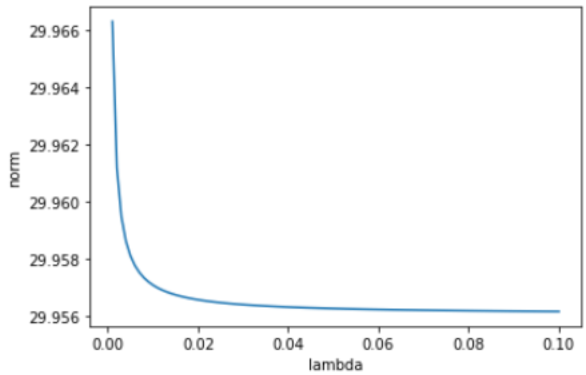
Все критерии показали наличие эффекта мультиколлинеарности. В этом случае вектор параметров целесообразно оценивать с использованием слагаемого L2-регуляризации. Построим ридж-оценку:

$$\Lambda = \lambda diag(X^TX)$$

$$\hat{\theta} = (X^T X + \Lambda)^{-1} X^T y$$

Параметр регуляризации будем выбирать исходя из значений $RSS(\lambda)$ и квадрата евклидовой нормы оценки параметров:





Для λ=1 имеем оценку:

$$\hat{\theta} = (1.0006, 9.4701, -3.2333, 9.6000, -3.2982, 0.9999, 1.0018, 0.9998, 0.9999)^T$$

5 Метод главных компонент

Перейдем к центрированным переменным:

$$y^* = y - \bar{y}$$

$$X_t^* = X_t - \bar{X}_t$$

Построим ковариационную матрицу, найдем для неё собственные значения и матрицу собственных векторов V матрицы $X^{*T} X^*$. Выразим главные компоненты в матричном виде:

$$Z = X^* V$$

Вектор собственных значений имеет вид:

$$(3.5508, 1.2528, 0.6048, 0.3891, 0.1910, 0.0878, 0.0152, 0.0001, 0.0001)^T$$

Исключим из матриц Z и V столбцы, соответствующие последним трём собственным значениям. Получим новую оценку вектора параметров:

$$b = (Z_R^T Z_R)^{-1} Z_R^T y^*$$

$$\hat{\theta} = V_R b = (0.1654, 2.1050, 0.4163, 1.8192, 0.4824, 1.2093, 1.2831, 0.8561, 1.2035)^T$$

$$||\hat{\theta}||^2 = 13.5$$

Оценка, полученная с помощью МГК получилась более точной, чем ридж-оценка.

6 Код программы

```python
import pandas as pd
import numpy as np
import random
import scipy.stats
from matplotlib import pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
np.set_printoptions(suppress=True)
random.seed(42)

n = 3000
m = 8
x1j, x2j, x3j, x4j, x5j, x6j, x7j, x8j, yj = [], [], [], [], [], [], [], [], []
factors_list = [x1j, x2j, x3j, x4j, x5j, x6j, x7j, x8j]

def u(x1,x2,x3,x4,x5,x6,x7,x8):
    return 1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8

for i in range(n):
    x1 = random.uniform(-1, 1)
    x2 = 2*x1 + np.random.normal(0, 0.01)
    x3 = random.uniform(-1, 1)
    x4 = 2*x3 + np.random.normal(0, 0.01)
    x5 = random.uniform(-1, 1)
    x6 = random.uniform(-1, 1)
    x7 = random.uniform(-1, 1)
    x8 = random.uniform(-1, 1)

    sigma2 = 0.05
    e = np.random.normal(0, sigma2)
    y = u(x1,x2,x3,x4,x5,x6,x7,x8) + e
    x1j.append(x1), x2j.append(x2), x3j.append(x3), x4j.append(x4),
    x5j.append(x5), x6j.append(x6), x7j.append(x7), x8j.append(x8), yj.append(y)

X = np.array([np.ones(n),
              np.reshape(x1j, (n, )),
              np.reshape(x2j, (n, )),
              np.reshape(x3j, (n, )),
              np.reshape(x4j, (n, )),
              np.reshape(x5j, (n, )),
              np.reshape(x6j, (n, )),
              np.reshape(x7j, (n, )),
              np.reshape(x8j, (n, )),]).T
y = np.reshape(yj, (n, ))

print('{:e}'.format(np.linalg.det(np.dot(X.T, X))))
```

```python
eig_max = np.amax(np.linalg.eigvals(np.dot(X.T, X)))
eig_min = np.amin(np.linalg.eigvals(np.dot(X.T, X)))
print('{:e}'.format(eig_min))
print('{:e}'.format(eig_max / eig_min))

R, max = np.empty((m, m)), 0
for i in range(m):
    for j in range(m):
        R[i][j] = round(1 - scipy.spatial.distance.cosine(factors_list[i], factors_list[j]), 5)
        if (R[i][j] > max) and (i != j):
            max = R[i][j]
print(R)

R2 = 1 - 1/np.diagonal(np.linalg.inv(R))
print('R2 =', np.around(R2, 6))
print(round(np.amax(abs(R2)), 6))

lambda_list, norm_list, rss_list = [], [], []

for lambda_param in np.linspace(0.01, 1, 100):
    L = lambda_param * np.diagonal(np.dot(X.T, X))
    theta_hat = np.dot(np.linalg.inv(np.dot(X.T, X) + L), np.dot(X.T, y))
    norm = np.dot(theta_hat.T, theta_hat)
    y_hat = np.dot(X, theta_hat)
    RSS = sum((y - y_hat)**2)
    lambda_list.append(lambda_param), norm_list.append(norm), rss_list.append(RSS)

plt.plot(lambda_list, rss_list)
plt.xlabel('lambda')
plt.ylabel('RSS')
plt.show()

plt.plot(lambda_list, norm_list)
plt.xlabel('lambda')
plt.ylabel('norm')
plt.show()

print(theta_hat)

df = pd.DataFrame(list(zip(lambda_list, norm_list, rss_list)), columns=['lambda', '||theta_new||^2', 'RSS(lambda)'])
df.index += 1
print(df)

X_star = np.array([X[0] - np.mean(X[0]),
```

```
                   X[1] - np.mean(X[1]),
                   X[2] - np.mean(X[2]),
                   X[3] - np.mean(X[3]),
                   X[4] - np.mean(X[4]),
                   X[5] - np.mean(X[5]),
                   X[6] - np.mean(X[6]),
                   X[7] - np.mean(X[7]),
                   X[8] - np.mean(X[8])])
y_star = np.reshape(np.array([y - np.mean(y)]), (n, ))

covmat = np.cov(X_star)
eigenvalues, eigenvectors = np.linalg.eig(covmat)
Z = np.dot(X, eigenvectors)
print(eigenvalues)

Z_new = np.delete(Z, (6, 7, 8), axis=1)
eigenvectors_new = np.delete(eigenvectors, (6, 7, 8), axis=1)
b = np.dot(np.linalg.inv(np.dot(Z_new.T, Z_new)), np.dot(Z_new.T,
 ↪  y_star))
theta_hat = np.dot(eigenvectors_new, b)
print(theta_hat)
norm = np.dot(theta_hat.T, theta_hat)
print(norm)
```