|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
|  | | |
| Лабораторная работа № 5 | | |
| по дисциплине «Статистические методы анализа данных» | | |
|  | | |
|  | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-02 |
| Вариант: | 6 |
| Студент: | Сидоров Даниил, |
|  | Дюков Богдан |
| Преподаватель: | Попов Александр Александрович. |
|  |  |
|
|  |  |
| Новосибирск | | |
| 2023 | | |

1. **Постановка задачи**
2. В соответствии с вариантом задания сгенерировать экспериментальные данные, в которых в явном виде присутствует эффект мультиколлинеарности.
3. Рассчитать ряд показателей, характеризующих эффект мультиколлинеарности. Определить факторы, ответственные за возникновение эффекта мультиколлинеарности.
4. 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.
5. 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.

Регрессия на 6 факторах. Эффект мультиколлинеарности создают 2 фактора. Имеется разброс в масштабах факторов.

1. **Ход работы**

**Генерация экспериментальных данных**

;

**Показатели, характеризующие мультиколлинеарность**

1. Определитель информационной матрицы.

Определитель информационной матрицы может быть большим числом, даже если присутствует мультиколлинеарность. Это связано с тем, что определитель матрицы зависит не только от наличия линейной зависимости между столбцами, но и от масштаба данных.

1. Минимальное собственное число информационной матрицы.

Число близко к нулю, это может указывать на наличие мультиколлинеарности.

Найдем также максимальное собственное число информационной матрицы:

1. Мера обусловленности матрицы по Нейману-Голдстрейну.

Это число значительно больше 1, что может указывать на наличие мультиколлинеарности.

1. Максимальная парная сопряженность. Построим матрицу сопряженности:

,

Косинус угла между векторами можно интерпретировать как косинусное сходство между векторами. Построенная матрица сопряженности:



В качестве показателя мультиколлинеарности выступает величина:

*;*

1. Максимальная сопряженность. В качество меры мультиколлинеарности можно взять . Вычислить можно следующим способом:

братный к сопряженной.



Максимальная сопряженность: За возникновение эффекта мультиколлинеарности отвечают факторы: , .

**Построение ридж-оценок параметров**

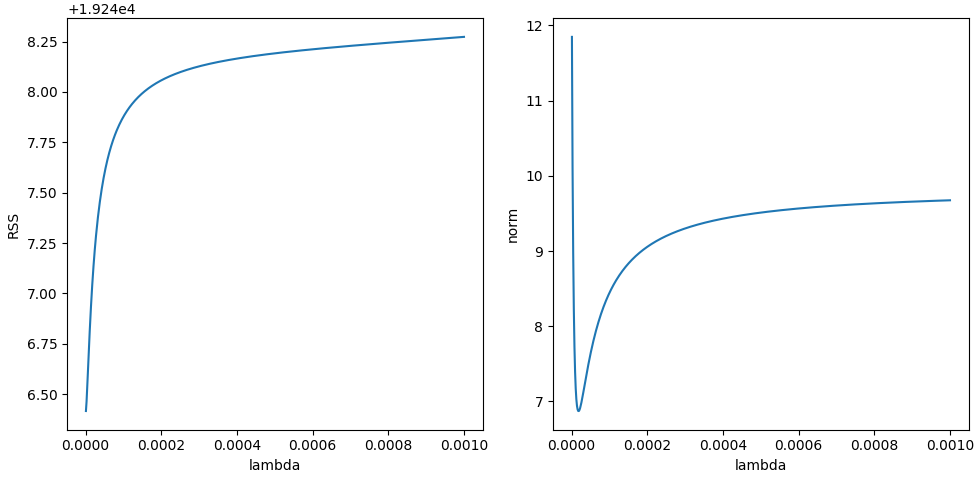
Для оценок параметров модели в условиях мультиколлинеарности используется следующая формула:

,

где матрицу можно задать как диагональную матрицу с элементами

, где - параметр регуляризации.

Параметр регуляризации выбираем как компромисс между неизбежным увеличением остаточной суммы квадратов и желаемым уменьшением евклидовой нормы оценок параметров.



Исходя из графиков имеем оптимальный и соответствующую ему ридж-оценку:

**Метод главных компонент**

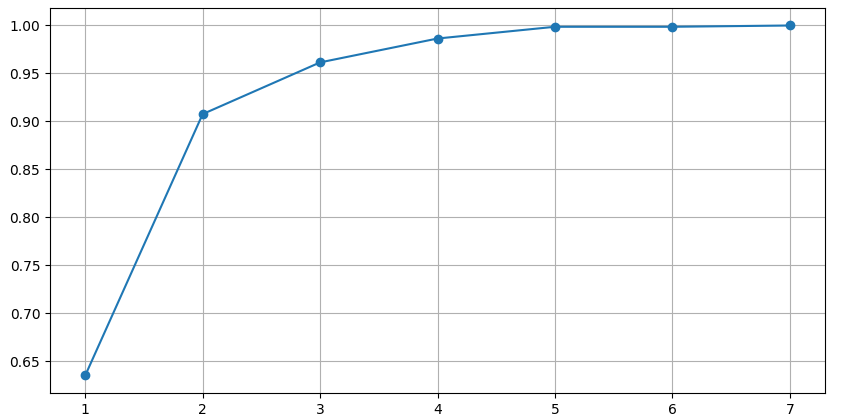
Перейдем к центрированным переменным:

Построим матрицу ковариаций , найдем для неё собственные значения и собственные вектора. Выразим главные компоненты в матричном виде:

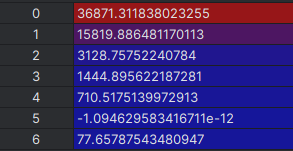
где –матрица, состоящая из всех собственных векторов матрицы .

Исключим из матриц и столбцы, соответствующие собственным значениям с незначительными вкладами. Для этого построим график зависимости критерия информативности от номера собственного значения.

Критерий информативности определяется как отношение кумулятивной суммы собственных значений к общей сумме собственных значений.



Собственные значения ковариационной матрицы:



Видим, что незначительный вклад дает собственное значения , соответствующие ему столбцы и исключаем. Обозначим полученные матрицы как и . Найдем вектор с помощью него найдем искомые оценки параметров: .

Метод главных компонент оказался хуже метода ридж-оценок как по смещению оценок, так и по точности предсказания отклика.

1. **Код программы**

import numpy as np

import matplotlib.pyplot as plt

np.random.seed(10)

# Генерация комбинаций факторов

def generate\_random(n):

x1\_list = [np.random.uniform(x1\_interval[0], x1\_interval[1]) for \_ in range(n)]

x2\_list = [np.random.uniform(x2\_interval[0], x2\_interval[1]) for \_ in range(n)]

x3\_list = [np.random.uniform(x3\_interval[0], x3\_interval[1]) for \_ in range(n)]

x4\_list = [np.random.uniform(x4\_interval[0], x4\_interval[1]) for \_ in range(n)]

x5\_list = [np.random.uniform(x5\_interval[0], x5\_interval[1]) for \_ in range(n)]

x6\_list = np.array(x5\_list) / 3 + np.random.normal(0, 0.01, size=n)

return map(np.array, [x1\_list, x2\_list, x3\_list, x4\_list, x5\_list, x6\_list])

# Получение дисперсии шума

def get\_sigma\_squared(u):

# Вычисление мощности сигнала

omega\_squared = np.dot(u - np.mean(u), u - np.mean(u)) / (len(u) - 1)

# Доля от мощности сигнала

rho = 0.05

# Вычисление дисперсии шума

return rho \* omega\_squared

# Получение ошибки

def get\_noise(u, sigma\_squared):

return np.random.normal(0, sigma\_squared, len(u))

n=1200

# Определение параметров

theta = np.array([1, 1, 1, 1, 1, 1, 1])

x1\_interval = (-1, 1)

x2\_interval = (-7, 7)

x3\_interval = (-3, 3)

x4\_interval = (-2, 2)

x5\_interval = (-10, 10)

#---------- Задание №1. Генерация данных ----------#

# Генерация комбинаций факторов

x1, x2, x3, x4, x5, x6 = generate\_random(n)

# Вычисление истинного отклика без шума

u = theta[0] + theta[1]\*x1 + theta[2]\*x2 + theta[3]\*x3 \

+ theta[4]\*x4 + theta[5]\*x5 + theta[6]\*x6

sigma2 = get\_sigma\_squared(u)

e = get\_noise(u, sigma2)

y = u + e

X = np.column\_stack((np.ones(len(x1)), x1, x2, x3, x4, x5, x6))

#---------- Задание №2. Расчет показателей ----------#

# Вычисление информационной матрицы и её определителя

info\_matrix = np.dot(X.T, X)

det\_info\_matrix = np.linalg.det(info\_matrix)

print("Определитель информационной матрицы: ", det\_info\_matrix)

# Вычисление собственных чисел информационной матрицы и сохранение max и min собственного числа

eigvals = np.linalg.eigvals(info\_matrix)

min\_eigval = min(eigvals)

max\_eigval = max(eigvals)

print("Минимальное собственное число информационной матрицы: {:.5e}".format(min\_eigval))

print("Максимальное собственное число информационной матрицы: {:.5e}".format(max\_eigval))

# Вычисление меры обусловленности матрицы по Нейману-Голдстейну

neuman\_goldstein\_measure = max\_eigval / min\_eigval

print("Мера обусловленности матрицы по Нейману-Голдстейну: {:.5e}".format(neuman\_goldstein\_measure))

# Инициализация матрицы сопряженности

R = np.zeros((X.shape[1] - 1, X.shape[1] - 1))

# Вычисление матрицы сопряженности без первого столбца (свободный член)

# Косинус угла между векторами можно интерпретировать как косинусное сходство между векторами

for i in range(1, X.shape[1]):

for j in range(i, X.shape[1]):

R[i - 1, j - 1] = (np.dot(X[:, i], X[:, j]) / (np.linalg.norm(X[:, i]) \* np.linalg.norm(X[:, j]))).round(8)

R[j - 1, i - 1] = R[i - 1, j - 1] # Отражение верхнего треугольника в нижний

print("Матрица сопряженности")

print(R)

np.fill\_diagonal(R, 0)

# Вычисление максимальной парной сопряженности

max\_pairwise\_conjugacy = np.max(np.abs(R))

print("Максимальная парная сопряженность: ", max\_pairwise\_conjugacy)

np.fill\_diagonal(R, 1)

# Вычисление обратной матрицы к матрице сопряженности

R\_inv = np.linalg.inv(R)

# Вычисление R\_2 для каждого i и нахождение максимального значения

R\_2 = [(1 - 1 / R\_inv[i, i]).round(8) for i in range(0, R.shape[0])]

max\_conjugacy = max(R\_2)

print("Вектор R\_2: ", R\_2)

print("Максимальная сопряженность: ", max\_conjugacy)

#---------- Задание №3. Ридж-оценки ----------#

# Инициализация списков для хранения евклидовых норм и остаточных сумм квадратов для графиков

euclidean\_norms = []

rss\_values = []

# Инициализация списка для хранения значений параметра регуляризации

lambdas = np.linspace(1e-06, 0.001, 600)

# Вычисление ридж-оценок для каждого значения параметра регуляризации

for lambda\_ in lambdas:

# Создание диагональной матрицы Lambda

Lambda = np.diag(lambda\_ \* np.diag(info\_matrix))

# Вычисление ридж-оценки

theta\_ridge = np.linalg.inv(X.T @ X + Lambda) @ X.T @ y

# Вычисление евклидовой нормы между оценкой и искомыми параметрами

euclidean\_norms.append(theta\_ridge.T @ theta\_ridge)

# Вычисление остаточной суммы квадратов

rss\_values.append((y - X @ theta\_ridge).T @ (y - X @ theta\_ridge))

# Построение графиков параметра регуляризации от RSS и евклидовой нормы

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1) # первый график

plt.plot(lambdas, rss\_values)

plt.xlabel('lambda')

plt.ylabel('RSS')

plt.subplot(1, 2, 2) # второй график

plt.plot(lambdas, euclidean\_norms)

plt.xlabel('lambda')

plt.ylabel('norm')

# Автоматическое выравнивание элементов на рисунке

plt.tight\_layout()

plt.show()

# Создание диагональной матрицы при оптимальном lambda = 3.6e-05

optimal\_diagonal\_lambda = np.diag(3.6e-05 \* np.diag(info\_matrix))

# Получаем ридж-оценку, соответствующую оптимальному значению lambda

optimal\_theta\_ridge = np.linalg.inv(X.T @ X + optimal\_diagonal\_lambda) @ X.T @ y

# Вычисление евклидовой нормы между оценкой и искомыми параметрами

optimal\_norm = np.sqrt(optimal\_theta\_ridge.T @ optimal\_theta\_ridge)

# Вычисление остаточной суммы квадратов

optimal\_rss = (y - X @ optimal\_theta\_ridge).T @ (y - X @ optimal\_theta\_ridge)

print("Ридж-оценка при lambda примерно равном 3.6e-05: ", optimal\_theta\_ridge)

print("RSS и Евклидова норма оценок параметров при lambda примерно равном 3.6e-05: {:.5e}".format(optimal\_rss), optimal\_norm)

#---------- Задание №4. Метод главных компонент ----------#

# Переход к центрированным оценкам

y\_centered = y - np.mean(y)

X\_centered = X - np.mean(X, axis=1, keepdims=True)

# Вычисление матрицы ковариации

cov\_matrix = X\_centered.T @ X\_centered

# Вычисление собственных значений и собственных векторов

eigen\_values, eigen\_vectors = np.linalg.eig(cov\_matrix)

print("Собственные числа ковариационной матрицы: ", eigen\_values)

# Матрица V состоит из всех собственных векторов

V = eigen\_vectors

# Главные компоненты Z получаются умножением X\* на V

Z = X\_centered @ V

# Построение графика зависимости критерия информативности от номера собственного значения

# Вычисляем кумулятивную сумму собственных значений

cumulative\_eigen\_values = np.cumsum(eigen\_values)

# Вычисление общей суммы собственных значений

total\_eigen\_values = np.sum(eigen\_values)

# Вычисление критерия информативности

ratios = cumulative\_eigen\_values / total\_eigen\_values

plt.figure(figsize=(10, 5))

plt.plot(range(1, len(eigen\_values) + 1), ratios, marker='o')

plt.grid(True)

plt.show()

# Исключение из Z и V столбцов соответствующих собственным значениям с незначительным вкладом

Z\_new = np.delete(Z, -2, axis=1)

V\_new = np.delete(V, -2, axis=1)

# Посчет b=(Z\_R^T \* Z\_R)^(−1) \* Z\_R^(T) \* y^\*

b = np.linalg.inv(Z\_new.T @ Z\_new) @ Z\_new.T @ y\_centered

# Выражение оценок V\_R \* b

theta\_main = V\_new @ b

# Вычисление остаточной суммы квадратов

rss\_main = (y - X @ theta\_main).T @ (y - X @ theta\_main)

# Вычисление евклидовой нормы между оценкой и искомыми параметрами

norm\_main = np.sqrt(theta\_main.T @ theta\_main)

print("Оценки параметров по методу главных компонентов: ", theta\_main)

print("RSS и Евклидова норма оценок параметров: {:.5e}".format(rss\_main), norm\_main))