Министерство науки и высшего образования Российской Федерации

Новосибирский государственный технический университет

Кафедра теоретической и прикладной информатики

Планирование и анализ эксперимента

Лабораторная работа №1

Факультет:	ФПМИ
Группа:	ПМ-63
Студенты:	Кожекин М.В.
	Майер В.А.
	Назарова Т.А.
	Утюганов Д.С.
Вариант:	9(1)

Новосибирск

2020

1. Цель работы

Изучить понятие оптимального плана эксперимента и критерии оптимальности планов.

2. Задание

1. Изучить понятия непрерывного плана эксперимента и информационной матрицы, а также критерии оптимальности, связанные с точностью оценивания параметров модели и точностью оценивания математического ожидания функции отклика.

2. Разработать программу по обработке различных планов эксперимента для регрессионных моделей. Обработка заключается в вычислении различных характеристик плана, связанных с тем или иным критерием оптимальности.

3. Для каждого из планов вычислить значения функционалов от информационной (дисперсионной) матриц, связанных с такими критериями, как: $D-$, $A-$, $E-$, Φ_2-, $\Lambda-$, $MV-$, $G-$ оптимальности. Проранжировать планы, указанные в варианте, с позиций различных критериев. Выбрать план, наиболее предпочтительный по совокупности критериев. Список планов приведен в табл. 1.

4. В качестве спектра плана выбрать один из приведенных в табл. 1 для соответствующей модели. Веса точек выразить в виде зависимости от одного параметра, как в примере аналитического построения оптимального плана. Для этого параметра определить допустимые интервалы значений, руководствуясь тем, что веса точек должны быть неотрицательные, а число таких точек с ненулевыми весами должно быть не меньше числа параметров в модели. Построить графики изменения критерия оптимальности плана, указанного в варианте, в зависимости от этого скалярного параметра; определить по графику оптимальные значения параметра и критерия. Сравнить полученный результат с результатами из п. 3.

5. Оформить отчет, включающий в себя постановку задачи, полученные результаты и текст программы.

6. Защитить лабораторную работу.

3. Анализ

В рамках работы исследуется общая линейная модель:

$$y = f^T(x)\Theta + \varepsilon = \sum_{l=1}^{m} f_l(x)\Theta_l + \varepsilon$$

$$y = \Theta_0 + \Theta_1 \cdot x + \Theta_2 \cdot x^2$$

Исходя из варианта задания (1) эксперимент проводится в трёх точках -1, 0, 1

Непрерывным нормированным планом эксперимента называется совокупность величин вида

$$\varepsilon^* = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ p_1 & p_2 & \cdots & p_n \end{bmatrix} \text{где } \sum_{i=1}^{n} p_i = 1, p_i \geq 0$$

Информационной матрицей дискретного непрерывного плана называется величина

$$M = \sum_{j=1}^{n} p_j f(x_j) f^T(x_j) = \frac{X^T X}{N}$$

Матрица, обратная к информационной, называется **дисперсионной**

$$D(\varepsilon) = M^{-1}(\varepsilon) D(\varepsilon_N) = M^{-1}(\varepsilon_N)$$

Критерии оптимальности плана эксперимента:

$$D - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} |D(\varepsilon)|$$

$$A - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} tr D(\varepsilon)$$

$$E - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} \max_{i} \lambda_i [D(\varepsilon)]$$

$$\Phi_2 - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} \sqrt{m^{-1} tr D^2(\varepsilon)}$$

$$\Lambda - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} \sum_{i=1}^{n} [\lambda_j(D(\varepsilon)) - \overline{\lambda}(D(\varepsilon))]^2$$

$$MV - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} \max_{i} D_{ii}(\varepsilon)$$

$$G - \text{оптимальности } \varepsilon^* = Arg \min_{\varepsilon} \max_{x \in \hat{X}} d(x, \varepsilon)$$

$$d(x, \varepsilon) = f^T(x) M^{-1} f(x)$$

4. Выбор оптимального плана

В каждом из 4 вариантов эксперимент проводится в трёх точках: -1, 0, 1

$y = \Theta_0 + \Theta_1 \cdot x + \Theta_2 \cdot x^2$

№	p_1	p_2	p_3
1	0.2	0.6	0.2
2	0.25	0.5	0.25
3	0.1884	0.6233	0.1884
4	0.333	0.333	0.333

Найдём информационные и дисперсионные матрицы для каждого из вариантов:

Матрицы M:

variant 1			variant 2			variant 3			variant 4		
1.0	0.0	0.4	1.0	0.0	0.5	1.00	0.00	0.38	1.00	0.00	0.67
0.0	0.4	0.0	0.0	0.5	0.0	0.00	0.38	0.00	0.00	0.67	0.00
0.4	0.0	0.4	0.5	0.0	0.5	0.38	0.00	0.38	0.67	0.00	0.67

Матрицы D:

variant 1			variant 2			variant 3			variant 4		
1.67	0.0	-1.67	2.0	0.0	-2.0	1.6	0.00	-1.60	3.0	0.0	-3.0
0.00	2.5	0.00	0.0	2.0	0.0	0.0	2.65	0.00	0.0	1.5	0.0
-1.67	0.0	4.17	-2.0	0.0	4.0	-1.6	0.00	4.26	-3.0	0.0	4.5

D	A		E		Phi_2		Lambda		MV		G		
i	value	i	value	i	value	i	value	i	value	i	value	i	value
4	6.77	2	8.00	1	5.00	2	2.83	3	8.72	2	4.00	2	4.00
2	8.00	1	8.33	3	5.01	1	2.97	1	8.80	1	4.17	1	4.17
1	10.42	3	8.52	2	5.24	3	3.04	2	10.67	3	4.26	3	4.26
3	11.30	4	9.01	4	6.85	4	3.24	4	22.55	4	4.50	4	4.50

Исходя из совокупности критериев оптимальным является план из варианта 2.

$$\varepsilon^* = \begin{bmatrix} -1 & 0 & 1 \\ 0.25 & 0.5 & 0.25 \end{bmatrix}$$

5. Выбор оптимального параметра

Под планом понимается модель из варианта 4 вида:

$$\varepsilon^* = \begin{bmatrix} -1 & 0 & 1 \\ q & 1 - 2 \cdot q & q \end{bmatrix} q \in (0, 0.5)$$

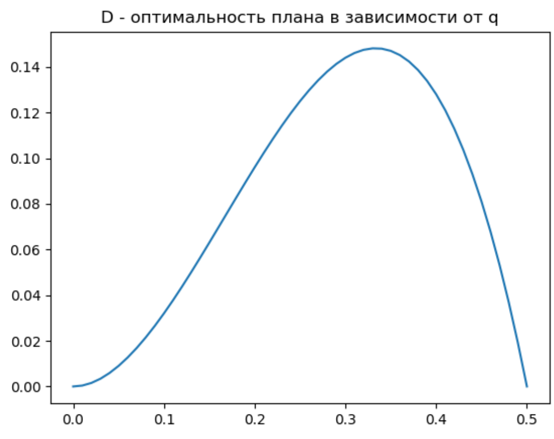

Рис. 1: Оптимальное значение параметра q = 0.333

Проверим решение аналитически:

$$M = q \cdot \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} + (1 - 2 \cdot q) \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + q \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$M = q \cdot \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + (1 - 2 \cdot q) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + q \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$det(M) = \begin{bmatrix} 1 & 0 & 2q \\ 0 & 2q & 0 \\ 2q & 0 & 2q \end{bmatrix} = 4q^2 + 0 + 0 - 8q^3 - 0 - 0 = 4q^2 - 8q^3$$

$$\frac{d(|M|)}{dq} = 4 \cdot 2q - 8 \cdot 3q^2 = 0$$

$$\frac{d(|M|)}{dq} = 3q^2 - q = 0$$

Аналитическое значение параметра $q = \dfrac{1}{3}$

6. Исходный код программы

lab1.py

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import re
from numpy.linalg import inv, det, eigvals, norm


pd.set_option('precision', 2)
n = 3
m = 3
points_count = 51

def f(theta, x):
    return theta[0] + theta[1]*x + theta[2]*x**2

def f_vector(x):
    return np.array([ [1], [x], [x**2] ])

def f_vector_T(x):
    return np.array([ 1, x, x**2 ])


#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
class Lab1():
    '''
    Класс для 1 лабораторной работы
    Для стандартизации все критерии ищут минимальное значение
    '''
    def __init__(self):
        ''' Выделение памяти под массивы '''
        self.x = np.ndarray(n)
```

```
        self.p = np.ndarray(n)
        self.M = np.ndarray((m, m))

    def read_plan_from_file(self, filename):
        ''' Считываем данные из файла '''
        data = np.loadtxt(filename)
        self.x = data[0]
        self.p = data[1]

    def create_plan(self, q):
        ''' Создание плана '''
        self.p = np.array([q, 1−2*q, q])

    def build_matrix_M(self):
        ''' Построение информационной матрицы M '''
        self.M = np.zeros((m,m))
        for i in range(n):
            self.M += self.p[i] * f_vector(self.x[i]) * f_vector_T(self.x[i])

    def build_matrix_D(self):
        ''' Построение дисперсионной матрицы D '''
        self.D = inv(self.M)

    def find_optimal_q(self):
        ''' Поиск оптимального значения q '''
        self.x_axis   = np.linspace(0, 0.5, points_count)
        self.D_from_E = np.ndarray(points_count)
        i = 0
        for q in self.x_axis:
            self.create_plan(q)
            self.build_matrix_M()
            # self.build_matrix_D() # тк.. матрица сингулярная
            self.D_from_E[i] = self.calc_M()
            i+=1

        i = np.where(self.D_from_E == max(self.D_from_E))
        print('Оптимальное значение параметра q: ' + str(self.x_axis[i][0]))

    def compare_plans(self, filename):
        ''' Сравнение планов различными критериями '''
        plan_files = [
            '1.txt',
            '2.txt',
            '3.txt',
            '4.txt'
            ]
        with open('report/matricies_M.tex', 'w') as f:
            f.write('\\begin{tabular}{|c|c|c|c|}\n')
            f.write('\hline\n')
            f.write('\tvariant 1 & variant 2 & variant 3 & variant 4 \\\ \n')
            f.write('\hline\n')
            s = ''
            for plan_file in plan_files:
                s += '&'
                self.read_plan_from_file('plans/' + plan_file)
                self.build_matrix_M()
                self.build_matrix_D()
                df = pd.DataFrame(data = self.M)
                s += df.to_latex(index=False, header = False, column_format='ccc
    ')
```

```
            s = re.sub(r'\\[a-z]+rule\n', '', s)# удаляем \toprule и \bottomrule
            s = re.sub(r'\n&', ' &\n', s)        # переносим &
            f.write(s[1:-1] + ' \\\\ \n')
            f.write('\hline\n\end{tabular}')

        with open('report/matricies_D.tex', 'w') as f:
            f.write('\\begin{tabular}{|c|c|c|c|}\n')
            f.write('\hline\n')
            f.write('\tvariant 1 & variant 2 & variant 3 & variant 4 \\\ \n')
            f.write('\hline\n')
            s = ''
            for plan_file in plan_files:
                s += '&'
                self.read_plan_from_file('plans/' + plan_file)
                self.build_matrix_M()
                self.build_matrix_D()
                df = pd.DataFrame(data = self.D)
                s += df.to_latex(index=False, header = False, column_format='ccc')
            s = re.sub(r'\\[a-z]+rule\n', '', s)# удаляем \toprule и \bottomrule
            s = re.sub(r'\n&', ' &\n', s)        # переносим &
            f.write(s[1:-1] + ' \\\\ \n')
            f.write('\hline\n\end{tabular}')



        criterion_files = [
            'D.txt',
            'A.txt',
            'E.txt',
            'Phi_2.txt',
            'Lambda.txt',
            'MV.txt',
            'G.txt'
            ]
        criterion_functions = {
            'D.txt': self.calc_D,
            'A.txt': self.calc_A,
            'E.txt': self.calc_E,
            'Phi_2.txt': self.calc_Phi_2,
            'Lambda.txt': self.calc_Lambda,
            'MV.txt': self.calc_MV,
            'G.txt': self.calc_G
        }

        i_vec = np.ndarray(4)
        cr_vec = np.ndarray(4)
        table = pd.DataFrame()

        # для каждого критерия
        for criterion_file in reversed(criterion_files):
            criterion_name = criterion_file[:-4]
            # рассмотрим 4 плана
            for plan_file in plan_files:
                self.read_plan_from_file('plans/' + plan_file)
                self.build_matrix_M()
                self.build_matrix_D()
                result = criterion_functions[criterion_file]()
                i = int(plan_file[:-4]) - 1
                i_vec[i] = i + 1
```

```python
                cr_vec[i] = result
            i_vec = np.argsort(cr_vec) + 1
            cr_vec = np.sort(cr_vec)
            d = {(criterion_name, 'i'): i_vec,
                 (criterion_name, 'value'): cr_vec}
            df = pd.DataFrame(data = d)
            table = df.join(table, lsuffix='_caller', rsuffix='_other')
        with open(filename, 'w') as f:
            f.writelines(table.to_latex(index=False))

    def draw_plot(self):
        ''' Отрисовка зависимости D(E) от q '''
        plt.title('D — оптимальность плана в зависимости от q')
        plt.plot(self.x_axis, self.D_from_E)
        plt.savefig('report/opt_q.png')

    def calc_M(self):
        '''
        Критерий D — оптимальности. (D — determinant)
        Эллипсоид рассеивания имеет минимальный объём
        '''
        return det(self.M)

    def calc_D(self):
        '''
        Критерий D — оптимальности. (D — determinant)
        Эллипсоид рассеивания имеет минимальный объём
        '''
        return det(self.D)

    def calc_A(self):
        '''
        Критерий A — оптимальности. (A — average variance)
        Эллипсоид рассеивания с наименьшей суммой квадратов длин осей
        '''
        return np.trace(self.D)

    def calc_E(self):
        '''
        Критерий E — оптимальности. (E — eigenvalue)
        Минимизация максимального собственного значения дисперсионной матрицы
        '''
        return np.max(eigvals(self.D))

    def calc_Phi_2(self):
        '''
        Критерий \Phi_2 — оптимальности. (\Phi_2 — функционал класса \Phi_p, p in (0, inf))
        Минимизация максимального собственного значения дисперсионной матрицы
        '''
        return np.sqrt(np.trace(self.D**2) / m)

    def calc_Lambda(self):
        '''
        Критерий \Lambda — оптимальности. (Lambda — собственное значение)
        Минимизация дисперсии собственных значений
        '''
        eig_vec = eigvals(self.D)
        avg_eig = np.mean(eig_vec)
        return np.sum((eig_vec[:]-avg_eig)**2)
```

```python

    def calc_MV(self):
        '''
        Критерий MV — оптимальности. (MV — maximum variation)
        Минимизация максимального диагонального значения дисперсионной матрицы
        '''
        return max(np.diag(self.D))

    def calc_G(self):
        '''
        Критерий G — оптимальности. (G — general varience)
        Минимизация максимального значения общей дисперсии
        '''
        return np.max([f_vector_T(x) * self.D * f_vector(x) for x in self.x])



#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------

l1 = Lab1()
# выбор оптимального плана
l1.compare_plans('report/plans_comparison.tex')
# выбор оптимального значения q
l1.read_plan_from_file('plans/4.txt')
l1.find_optimal_q()
l1.draw_plot()
```