

Министерство науки и высшего образования Российской Федерации
Новосибирский государственный технический университет

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:

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

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

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

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

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

Критерии оптимальности плана эксперимента:

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

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

$$E - \text{оптимальности } \varepsilon^* = \text{Arg min}_{\varepsilon} \max_i \lambda_i[D(\varepsilon)]$$

$$\Phi_2 - \text{оптимальности } \varepsilon^* = \text{Arg min}_{\varepsilon} \sqrt{\text{tr} D^2(\varepsilon)}$$

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

$$MV - \text{оптимальности } \varepsilon^* = \text{Arg min}_{\varepsilon} \max_i D_{ii}(\varepsilon)$$

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

4. Выбор оптимального плана

D			A			E			Phi_2			Lambda			MV			G			
i		value	i		value	i		value	i		value	i		value	i		value	i		value	
0	4	6.770	2	8.000	1	5.000	2	2.828	3	8.721	2	4.000	2	4.000	1	2	8.000	1	5.013	1	2.966
1	2	8.000	1	8.333	3	5.013	1	2.966	1	8.796	1	4.167	1	4.167	2	1	10.417	3	8.517	2	5.236
2	1	10.417	3	8.517	2	5.236	3	3.041	2	10.667	3	4.258	3	4.258	3	3	11.300	4	9.009	4	6.849
3	3	11.300	4	9.009	4	6.849	4	3.244	4	22.545	4	4.505	4	4.505							

Исходя из совокупности критериев оптимальным является план из варианта 2.

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 $q = 0.333$

6. Исходный код программы

lab1.py

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 import pandas as pd
4 from numpy.linalg import inv, det, eigvals, norm
5
6 n = 3
7 m = 3
8 points_count = 21
9
10 def f(theta, x):
11     return theta[0] + theta[1]*x + theta[2]*x**2
12
13 def f_vector(x):
14     return np.array([ 1, [x], [x**2] ])
15
16 def f_vector_T(x):
17     return np.array([ 1, x, x**2 ])
18
19
20 #
21 #
22 #
23 class Lab1():
24     '''
25     Класс для 1 лабораторной работы
26     Для стандартизации все критерии ущют минимальное значение
27     '''
28     def __init__(self):
29         ''' Выделение памяти под массивы '''
30         self.x = np.ndarray(n)
31         self.p = np.ndarray(n)

```

```

32     self.M = np.ndarray((m, m))
33
34 def read_plan_from_file(self, filename):
35     ''' Считываем данные из файла '''
36     data = np.loadtxt(filename)
37     self.x = data[0]
38     self.p = data[1]
39
40 def create_plan(self, q):
41     ''' Создание плана '''
42     self.p = np.array([q, 1-2*q, q])
43
44 def build_matrix_M(self):
45     ''' Построение информационной матрицы M '''
46     self.M = np.zeros((m,m))
47     for i in range(n):
48         self.M += self.p[i] * f_vector(self.x[i]) * f_vector_T(self.x[i])
49
50 def build_matrix_D(self):
51     ''' Построение дисперсионной матрицы D '''
52     self.D = inv(self.M)
53
54 def find_optimal_q(self):
55     ''' Поиск оптимального значения q '''
56     self.x_axis = np.linspace(0, 0.5, points_count)
57     self.D_from_E = np.ndarray(points_count)
58     i = 0
59     for q in self.x_axis:
60         self.create_plan(q)
61         self.build_matrix_M()
62         # self.build_matrix_D() # тк.. матрица сингулярная
63         self.D_from_E[i] = self.calc_M()
64         i+=1
65
66     i = np.where(self.D_from_E == max(self.D_from_E))
67     print('Оптимальное значение параметра q: ' + str(self.x_axis[i][0]))
68
69 def compare_plans(self, filename):
70     ''' Сравнение планов различными критериями '''
71     plan_files = [
72         '1.txt',
73         '2.txt',
74         '3.txt',
75         '4.txt'
76     ]
77     criterion_files = [
78         'D.txt',
79         'A.txt',
80         'E.txt',
81         'Phi_2.txt',
82         'Lambda.txt',
83         'MV.txt',
84         'G.txt'
85     ]
86     criterion_functions = {
87         'D.txt': self.calc_D,
88         'A.txt': self.calc_A,
89         'E.txt': self.calc_E,
90         'Phi_2.txt': self.calc_Phi_2,
91         'Lambda.txt': self.calc_Lambda,

```

```

92         'MV.txt': self.calc_MV,
93         'G.txt': self.calc_G
94     }
95
96     i_vec = np.ndarray(4)
97     cr_vec = np.ndarray(4)
98     pd.set_option('precision', 3)
99     table = pd.DataFrame()
100
101     # для каждого критерия
102     for criterion_file in reversed(criterion_files):
103         criterion_name = criterion_file[:−4]
104         # рассмотрим 4 плана
105         for plan_file in plan_files:
106             self.read_plan_from_file('plans/' + plan_file)
107             self.build_matrix_M()
108             self.build_matrix_D()
109             result = criterion_functions[criterion_file]()
110             i = int(plan_file[:−4]) − 1
111             i_vec[i] = i + 1
112             cr_vec[i] = result
113         i_vec = np.argsort(cr_vec) + 1
114         cr_vec = np.sort(cr_vec)
115         d = {(criterion_name, 'i'): i_vec,
116              (criterion_name, 'value'): cr_vec}
117         df = pd.DataFrame(data = d)
118         table = df.join(table, lsuffix='_caller', rsuffix='_other')
119     with open(filename, 'w') as f:
120         f.writelines(table.to_latex())
121
122     def draw_plot(self):
123         ''' Отрисовка зависимости D(E) от q '''
124         plt.title('D − оптимальность плана в зависимости от q')
125         plt.plot(self.x_axis, self.D_from_E)
126         plt.savefig('report/opt_q.png')
127
128     def calc_M(self):
129         '''
130         Критерий D − оптимальности. (D − determinant)
131         Эллипсоид рассеивания имеет минимальный объём
132         '''
133         return det(self.M)
134
135     def calc_D(self):
136         '''
137         Критерий D − оптимальности. (D − determinant)
138         Эллипсоид рассеивания имеет минимальный объём
139         '''
140         return det(self.D)
141
142     def calc_A(self):
143         '''
144         Критерий A − оптимальности. (A − average variance)
145         Эллипсоид рассеивания с наименьшей суммой квадратов длин осей
146         '''
147         return np.trace(self.D)
148
149     def calc_E(self):
150         '''
151         Критерий E − оптимальности. (E − eigenvalue)

```

```

152     Минимизация максимального собственного значения дисперсионной матрицы
153     '''
154     return np.max(eigvals(self.D))
155
156     def calc_Phi_2(self):
157         '''
158         Критерий \Phi_2 – оптимальности. (\Phi_2 – функционал класса \Phi_p, p in
159         (0, inf))
160         Минимизация максимального собственного значения дисперсионной матрицы
161         '''
162         return np.sqrt(np.trace(self.D**2) / m)
163
164     def calc_Lambda(self):
165         '''
166         Критерий \Lambda – оптимальности. (\Lambda – собственное значение)
167         Минимизация дисперсии собственных значений
168         '''
169         eig_vec = eigvals(self.D)
170         avg_eig = np.mean(eig_vec)
171         return np.sum((eig_vec[:] - avg_eig)**2)
172
173     def calc_MV(self):
174         '''
175         Критерий MV – оптимальности. (MV – maximum variation)
176         Минимизация максимального диагонального значения дисперсионной матрицы
177         '''
178         return max(np.diag(self.D))
179
180     def calc_G(self):
181         '''
182         Критерий G – оптимальности. (G – general varience)
183         Минимизация максимального значения общей дисперсии
184         '''
185         return np.max([f_vector_T(x) * self.D * f_vector(x) for x in self.x])
186
187
188 #
189 #
190 #
191
192 l1 = Lab1()
193 # выбор оптимального плана
194 l1.compare_plans('report/plans_comparison.tex')
195 # выбор оптимального значения q
196 l1.read_plan_from_file('plans/4.txt')
197 l1.find_optimal_q()
198 l1.draw_plot()

```