Министерство науки и высшего образования Российской Федерации
Новосибирский государственный технический университет
Кафедра теоретической и прикладной информатики

Планирование и анализ эксперимента

Лабораторная работа №4

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 методы оптимального планирования эксперимента при нелинейной параметризации функции отклика.

2. Задание

1. Изучить понятия локально-оптимального планирования и информационной матрицы при нелинейной параметризации функции отклика, ознакомиться с видом производственной функции Кобба-Дугласа.

2. По заданному типу технологии сформировать имитационную модель в виде производственной функции Кобба-Дугласа. При этом задать истинные значения для параметров, нелинейно входящих в модель. Выход модели зашумить, уровень шума установить в пределах 15...20 % от мощности полезного сигнала.

3. Выбрать план для затравочног оэксперимента, состоящий из небольшого числа наблюдений, и смоделировать на его основе экспериментальные данные.

4. Оценить параметры модели по полученным экспериментальным данным. Для этого необходимо перейти к линейной модели, воспользовавшись логарифмическим представлением уравнения модели наблюдения. Параметры преобразованной модели тогда можно оценить обычным «линейным» МНК.

5. Построить локально-оптимальный план эксперимента для исходной нелинейной модели, воспользовавшись разработанной ранее программой синтеза дискретных оптимальных планов и полученными оценками параметров модели. Число наблюдений должно в 4...5 раз превышать число параметров модели.

6. По сформированной ранее (п. 2) имитационной модели провести имитационный эксперимент в точках полученного локально-оптимального плана. Провести оценку параметров и вычислить норму отклонения оценок от их истинны хзначений. Вычислительный эксперимент повторить не менее 100 раз, каждый раз с новой реализацией помехи. Вычислить среднее значение нормы отклонения оценок. Процедуру повторить, используя в качестве плана эксперимента случайно расположенные точки в факторном пространстве. В серии вычислительных экспериментов случайный план фиксируется (выбирается один раз). Сделайте вывод об эффективности оптимального планирования эксперимента для идентификации заданной нелинейной модели.

7. Оформить отчет, включающий в себя постановку задачи, оценки параметров по затравочному эксперименту, полученный локально-оптимальный план, результаты проведенного в п. 6 исследования и текст программы.

8. Защитить лабораторную работу.

3. Анализ

Технология Кобба-Дугласа. Ресурсов два, изменяются в пределах [1, 10]. Постоянная отдача от масштаба. Локально-D-оптимальное планирование.

Модель наблюдения за объектом представляет собой уравнение вида:

$$y = \eta(x, \Theta) + e, \quad \text{где:}$$

y - значение отклика;

η - нелинейная функция вектора параметров Θ;

Θ - вектор неизвестных параметров;

e - ошибка наблюдения

Функция Кобба-Дугласа $\eta(x, \Theta)$ имеет вид:

$$y = \Theta_0 \cdot X_1^{\Theta_1} \cdot X_2^{\Theta_2}$$

Её логарифмическое представление:

$$y = \log(\Theta_0) + \Theta_1 \cdot \log(X_1) + \Theta_2 \cdot \log(X_2)$$

При постоянной отдаче от масштаба параметры удовлетворяют ограничению

$$\sum_{i=1}^{k} \Theta_i = 1$$

Информационная матрица Фишера для нелинейной модели зависит от $\hat{\Theta}$. Приближенное значение нормированной информационной матрицы дискретного плана можно вычислить по формуле

$$M\left(\varepsilon_N, \hat{\Theta}\right) \approx M\left(\varepsilon_N, \Theta_{true}\right) = \sum_{j=1}^{n} p_j f\left(x_j, \Theta_{true}\right) f^T\left(x_j, \Theta_{true}\right), \quad \text{где}$$

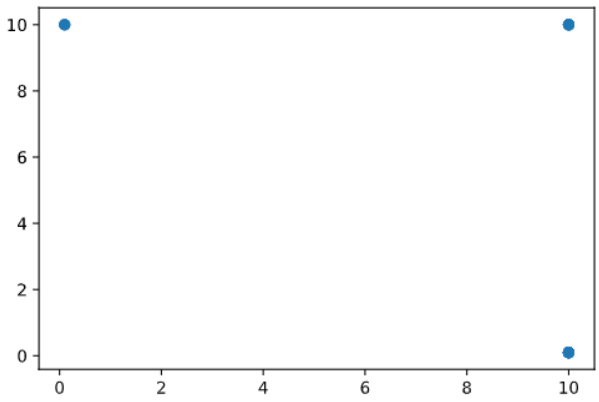
$$f\left(x, \hat{\Theta}\right) = \left.\frac{\partial \eta(x, \Theta)}{\partial \Theta}\right|_{\Theta=\hat{\Theta}} = \left(X_1^{\Theta_1} \cdot X_2^{\Theta_2}, \Theta_0 \cdot \Theta_1 \cdot X_1^{\Theta_1 - 1} \cdot X_2^{\Theta_2}, \Theta_0 \cdot X_1^{\Theta_1} \cdot \Theta_2 \cdot X_2^{\Theta_2 - 1}\right)^T$$

Дисперсионная матрица определяется как обратная к информационной, т.е.

$$D\left(\varepsilon_N, \hat{\Theta}\right) = M^{-1}\left(\varepsilon_N, \hat{\Theta}\right)$$

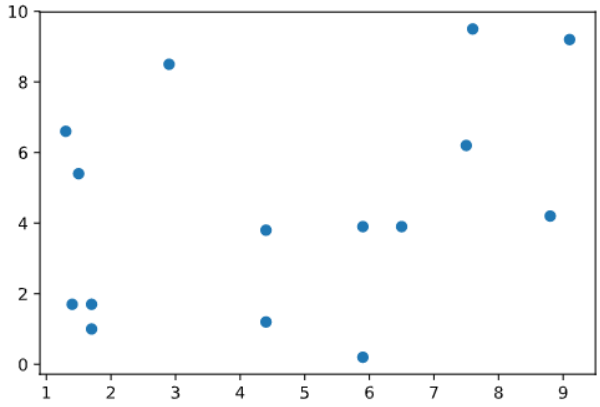
Пусть Θ_{true} = [0.2, 0.4, 0.4], шум ρ = 20%, N = 3x5 = 15 точек, ε = 1e-4.

Оптимальный план



RSS: 50.6925
отклонение нормы оценок: 0.0026

Случайный план



RSS: 50.7152
отклонение нормы оценок: 0.0027

Вывод:

Построение оптимального дискретного плана повышает точность оценки параметров и, соответственно, точность модели.

4. Исходный код программы

PlanOfExp4.py

```
import numpy as np
import random
import matplotlib.pyplot as plt

#Функция Ф, которая как производная от \Eta.
def function(x):
    return np.array([
        [x[0] ** theta[1] * x[1] ** theta[2]],
        [theta[0] * theta[1] * x[0] ** (theta[1] - 1) * x[1] ** theta[2]],
        [theta[0] * theta[2] * x[0] ** theta[1] * x[1] ** (theta[2] - 1)]
    ])

#Построение массива откликов с зашумлением
def makeY(plan, p = 0.2):
    Y = list(map(lambda x: theta[0] * x[0]**theta[1] * x[1]**theta[2], plan))
    Y = list(map(lambda y: y + random.normalvariate(0, p * y), Y))
    return Y

#Построение сдучайного плана эксперимента на сетке
def makePlan(grid = np.linspace(1, 10, 1001), m = 9):
    return list(map(lambda x: [random.choice(grid), random.choice(grid)], range(
    m)))


```

```
#Построение матрицы Икс по плану
def makeX(plan):
    return np.array(list(map(lambda x: [1.0, np.math.log(x[0]), np.math.log(x[1])], plan)))

N = 15
grid = np.linspace(0.1, 10, 100)
plan = makePlan(grid = grid)
theta = [0.2, 0.4, 0.4]
Y = makeY(plan)
Y = np.log(Y)
X = makeX(plan)

#функция вычисления информационной матрицы
def InfMatrix(plan):
  m = len(function(plan[0]))
  M = np.zeros((m,m))
  for j in range(m):
    M += (1 / N) * function(plan[j]) * function(plan[j]).T
  return M

#функция вычисления дисперсионной матрицы
def DispMatrix(a):
  return np.linalg.inv(a)

M = InfMatrix(plan)
D = DispMatrix(M)

#дисперсия модели
def DisM(x, M):
    return np.dot(np.dot(function(x).T, DispMatrix(M)), function(x))

def DisM_(x,x_j,M):
    return np.dot(np.dot(function(x).T, DispMatrix(M)), function(x_j))

def Delta(x,x_j,M):
    return ((1/N) * (DisM(x,M) - DisM(x_j,M))) - ((1/(N**2)) * (DisM(x,M) * DisM(x_j,M) - DisM_(x,x_j,M)**2))

#нахождения максимального значения дельты на для одной точки из плана, но для всейй сетки.
def findMaxforOneX(x, M):
    maxdot = [grid[0], grid[0]]
    maxvalue = Delta(maxdot,x, M)
    for x1 in grid:
        for x2 in grid:
            value = Delta([x1, x2], x, M)
            if value > maxvalue:
                maxvalue = value
                maxdot = [x1, x2]
    return [maxvalue, maxdot]

#функция нахождения максимумов для всех точек плана и выбора из него наибольшего
def findMaxforAll(X, M):
    listofmax = [findMaxforOneX(x,M) for x in X]
    return [*max(listofmax),listofmax.index(max(listofmax))]
```

```
#Построение оптимального плана эксперимента по некоторому плану plan
def makeOptimalPlan(plan):
    eps = 0.0001
    iteration = 0
    while True:
        M = InfMatrix(plan)
        print("det", np.linalg.det(M))
        delta = findMaxforAll(plan, M)
        if delta[0] > eps:
            plan[delta[2]] = delta[1]
        else:
            break
        iteration += 1
    return plan

plan = makePlan(grid = grid, m = N)
first_plan = plan
plt.scatter([first_plan[i][0] for i in range(len(first_plan))],[first_plan[i][1]
    for i in range(len(first_plan))],)
plt.show()
optim_plan = makeOptimalPlan(plan)

plt.clf()

plt.scatter([optim_plan[i][0] for i in range(len(optim_plan))],[optim_plan[i][1]
    for i in range(len(optim_plan))],)
plt.show()

def MNK(x,y):
    exp_theta =  np.dot(np.dot(np.linalg.inv(np.dot(x.T,x)),x.T),y)
    exp_theta[0] = np.math.e ** exp_theta[0]
    return exp_theta

def RSS(x,y,theta_):
    y_exp = np.dot(x,theta_)
    return np.dot(y - y_exp, y - y_exp)

def Experiment(plan):
    rss = 0
    dif_norma = 0
    for i in range(100):
        X = makeX(plan)
        Y = makeY(plan)
        Y = np.log(Y)
        exp_theta = MNK(X,Y)
        rss += RSS(X,Y,exp_theta)
        dif_norma += np.dot(exp_theta - theta,exp_theta - theta)
    rss /= 100
    dif_norma /= 100
    return rss, dif_norma

opt_rss, opt_norma = Experiment(optim_plan)
print('Оптимальный план')
print('RSS: ', opt_rss)
print('отклонение нормы оценок: ', opt_norma)

rand_rss, rand_norma = Experiment(first_plan)
print('Случайный план')
print('RSS: ', rand_rss)
print('отклонение нормы оценок: ', rand_norma)
```