## **Постановка задачи**

Исследовать зависимость между стоимостью грузовой автомобильной перевозки Y тысяч рублей, массой груза X1 тонн и расстоянием X2 тысяч километров по 20 транспортным компаниям, используя данные наблюдений, приведенные в таблице 1.

Таблица 1. Исходные данные

|  |  |  |
| --- | --- | --- |
| **X1** | **X2** | **Y** |
| 1,9 | 1,8 | 9 |
| 2 | 2,2 | 9,6 |
| 2,6 | 0,9 | 7,44 |
| 3 | 1,8 | 6,96 |
| 3,4 | 1 | 6,48 |
| 3,7 | 3 | 16,56 |
| 4,5 | 1,1 | 8,52 |
| 8 | 2,3 | 30,6 |
| 9 | 0,5 | 7,2 |
| 14 | 1,5 | 18,96 |
| 15 | 2,4 | 26,4 |
| 17 | 1,1 | 12,8 |
| 18 | 0,6 | 6,32 |
| 19 | 2,6 | 59,2 |
| 21 | 0,6 | 9,2 |
| 22 | 1,3 | 20,8 |
| 25 | 2,5 | 44,8 |
| 27 | 2,3 | 41,6 |
| 34 | 1,6 | 20,8 |
| 36 | 2 | 40,8 |

## **Алгоритм построения модели**

1. Специфицировать модель, как линейную зависимость объясняемых переменных Y от объясняющих переменных X.
2. По результатам наблюдений сформировать матрицы X= (n, k-1), F=(n,k), Y= (n,1).
3. При помощи алгоритма случайного поиска необходимо минимизировать функцию и найти оптимальное решение.

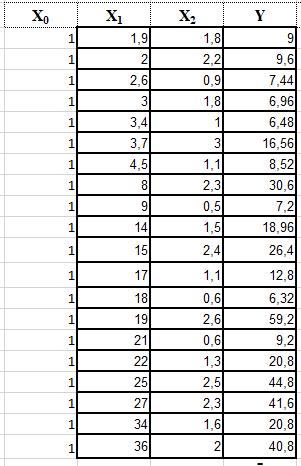
## **Спецификация модели**

Располагаем пространственной выборкой объема n=20 c числом объясняющих переменных k=2. Модель в виде линейной зависимости:

 (1)

По результатам наблюдений формируем матрицы F=[20x3] и Y=[20x1]. Фиктивная переменная X0 равна единице для всех наблюдений i.

Таблица 2. Матрицы F и Y



## **Алгоритм случайного поиска**

*Алгоритм случайного поиска* относится к алгоритмам нелинейного математического программирования. Такие алгоритмы снискали себе широкую популярность при решении практических инженерных задач.

Простейший алгоритм – локальный неадаптивный алгоритм случайного поиска следующий (рис. 1).

1. Задаем начальную точку, представленную вектором X0, объявляем ее текущей и вычисляем в ней значений целевой функции.
2. Текущей точке придаем приращение в виде случайного вектора дельта X и вычисляется значение целевой функции.
3. Если значение целевой функции улучшилось, то данную точку делаем текущей.
4. Проверить условие остановки. Если оно выполняется, то переходим на шаг 5, в противном случае на шаг 2.
5. Остановка.

Для реализации используем python и библиотеки numpy, pandas.

Импортируем необходимые библиотеки

import numpy as np

import pandas as pd

import random

from google.colab import files

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

Загрузим наши данные в формате csv, затем используя библиотеку pandas создадим новый DataFrame и инициализируем данные.

#Инициализация данных

data = pd.read\_csv(io.BytesIO(uploaded['Data.csv']), sep=";", dtype={'x1': np.float64, 'x2': np.float64})

print(data)

X = data.iloc[:, 0:2]

w = np.ones(20)

W = pd.DataFrame(w)

X['Ones'] = W

print(X)

print('   ')

Y = data.iloc[:, 2]

print(Y[:5])

При помощи библиотеки PyPlot визуализируем данные.

fig = plt.figure(1)

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X.iloc[:, 0], X.iloc[:, 1], Y)

ax.set\_xlabel('X1')

ax.set\_ylabel('X2')

ax.set\_zlabel('Y')

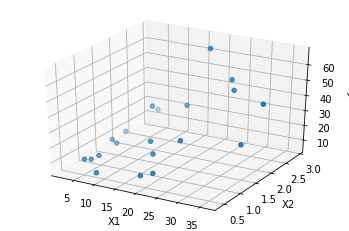


Рисунок 1. Визуализация данных.

Функция для подсчёта ошибки:

def cost\_func(x\_1, x\_2, w0, w1, w2, \_y):  # Подсчёт ошибки

    total\_error = 0

    count = x\_1.size

    for i in range(count):

      total\_error += (\_y[i] - (w0 + w1 \* x\_1[i] + w2 \* x\_2[i])) \*\* 2

    return total\_error / count

Создадим функцию алгоритма случайного поиска:

def update\_wights(x\_1, x\_2, \_y, max\_iter, function):  # Нахождение коэффецентов

      MAXSIZE = 3  # Количество искомых коэффецентов

      w = np.empty(MAXSIZE, dtype=np.float32)

      new\_w = np.empty(MAXSIZE, dtype=np.float32)

Создадим границы поиска:

   l = [-20, -10, 0]  # Нижние границы

    h = [-10, 0, 10]  # Верхние границы

Создадим массивы случайных коэффициентов:

# Случайная инициализация кооэфицентов

    w[0] = random.uniform(l[0], h[0])

    w[1] = random.uniform(l[1], h[1])

    w[2] = random.uniform(l[2], h[2])

    new\_w[0] = w[0]

    new\_w[1] = w[1]

    new\_w[2] = w[2]

Далее напишем реализацию алгоритма:

for \_ in range(max\_iter):

        # Обновление кооэфицентов на случайный сдвиг

        for t in range(MAXSIZE):

            new\_w[t] = l[t] + random.random() \* (h[t] - l[t])

        # Проверка на минимизацию

        if cost\_func(x\_1, x\_2, new\_w[0], new\_w[1], new\_w[2], \_y) < cost\_func(x\_1, x\_2, w[0], w[1], w[2], \_y):

            # Обновление коэффецентов и границ

            for t in range(MAXSIZE):

                w[t] = new\_w[t]

                h[t] = w[t] + 5

                l[t] = w[t] - 5

    print("Невязка:", cost\_func(x\_1, x\_2,w[0], w[1], w[2], \_y))  # Вывод ошибки

    print("Коэфиценты:", w[0], w[1], w[2])  # Вывод весов

    return w[0], w[1], w[2], cost\_func(x\_1, x\_2, w[0], w[1], w[2], \_y)

Нам понадобится вектор предсказанных y:

def predicted(x\_1, x\_2, y, a, b, c):

  pr = np.empty(19, dtype=np.float32)

  for i in range(19):

    pr[i] = (a + b \* x\_1[i] + c \* x\_2[i])

  return pr

Вызовем функцию алгоритма случайного поиска:

a,b,c,cost = update\_wights(X.iloc[:,0],X.iloc[:,1],Y,10000,your\_function)

На выходе получаем массив содержащий три решения, которые и будут являться искомыми коэффициентами.

Невязка: 78.65404252309818

Коэффициенты: -14.278664 0.8955865 13.991849

Визуализируем полученные результаты, отобразив невязку между результатами ожидаемыми данными, используя библиотеку matplotlib.pyplot.

predictedY = np.dot(X,[b,c,a])

#  Подготовка данных для визуализации

xx, yy, zz = np.meshgrid(X.iloc[:, 0], X.iloc[:, 1], X.iloc[:, 2])

combinedArrays = np.vstack((xx.flatten(), yy.flatten(), zz.flatten())).T

Z = combinedArrays.dot([b,c,a])

# Визуализация предсказанных данных и реальных.

fig = plt.figure(2)

ax = fig.add\_subplot(111, projection='3d')

ax.scatter(X.iloc[:, 0], X.iloc[:, 1], Y, color='r', label='Actual')

ax.scatter(X.iloc[:, 0], X.iloc[:, 1], predictedY, color='g', label='Predicted')

ax.plot\_trisurf(combinedArrays[:, 0], combinedArrays[:, 1], Z, alpha=0.5)

ax.set\_xlabel('X1')

ax.set\_ylabel('X2')

ax.set\_zlabel('Y')

ax.legend()

plt.show()

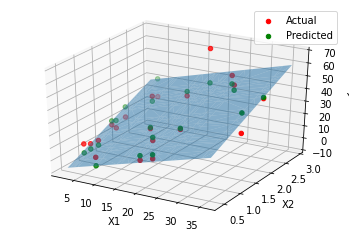


Рисунок 2. После нахождения коэффициентов