Метод наименьших квадратов

Общие положения

Пусть две величины x и y связаны табличной зависимостью:

x	x_1	x_2	x_3	…	x_{n-1}	x_n
y	y_1	y_2	y_3	…	y_{n-1}	y_n

Требуется установить функциональную зависимость $y = f(x)$ между переменными. В общем случае искомая функция $y = f(x)$ будет зависеть не только от x, но и от некоторого количества параметров $y = f(x,a,b,\ldots)$.

Постановка задачи:

Найти аппроксимирующую функцию $y = f(x,a,b,\ldots)$, чтобы в точках $x = x_i$ она принимала значения максимально близкие к табличным. Вид искомой функции может быть известен из теоретических соображений или определяться характером расположения исходных данных на графике.

Суть метода наименьших квадратов: между искомой функцией и табличными значениями в точках x_i наблюдаются отклонения. Обозначим их $\Delta y_i = f(x_i,a,b,\ldots) - y_i$, где $i = 1,2,3,\ldots n$. Выбираем значения коэффициентов $a,b,\ldots$ таким образом, чтобы сумма квадратом отклонений принимала минимальное значение:

$$S(a,b,\ldots) = \sum_{i=1}^{n} (\Delta y_i)^2 = \sum_{i=1}^{n} \left(f(x_i,a,b,\ldots) - y_i \right)^2 \to \min .$$

Сумма $S(a,b,\ldots)$ является функцией нескольких переменных. Необходимым условием экстремума функции нескольких переменных является обращение в ноль частных производных:

$$S_a' = 0, \quad S_b' = 0, \ldots \qquad (1)$$

Алгоритм решения:

1. Выбираем функцию $y = f(x,a,b,\ldots)$.
2. Составляем систему уравнений (1).
3. Решая систему уравнений, находим коэффициенты $a,b,\ldots$.
4. Подставляем коэффициенты $a,b,\ldots$ в искомую функцию.
5. По достаточному признаку экстремума функции нескольких переменных следует убедиться в постоянстве значка дифференциала второго порядка: $d^2S > 0$ при любых приращениях аргументов $da,\ db,\ldots$. Такая проверка рассматривается в теоретической части метода и на практике не повторяется.
6. Обычно рассматривают несколько видов функций и выбирают ту, для которой суммарная погрешность $\sum_{i=1}^{n} \left(f(x_i,a,b,\ldots) - y_i \right)^2$ будет наименьшей.

Для линейной функции $y = ax + b$ метод наименьших квадратов будет выглядеть следующим образом:

1. $S(a,b) = \sum_{i=1}^{n} \left(ax_i + b - y_i \right)^2 \to \min$
2. $\begin{cases} S'_a(a,b) = \sum_{i=1}^{n} 2\left(ax_i + b - y_i \right) \cdot x_i = 0 \\ S'_b(a,b) = \sum_{i=1}^{n} 2\left(ax_i + b - y_i \right) \cdot 1 = 0 \end{cases}$
3. $\begin{cases} \left(\sum_{i=1}^{n} x_i^2 \right) \cdot a + \left(\sum_{i=1}^{n} x_i \right) \cdot b = \sum_{i=1}^{n} x_i \cdot y_i \\ \left(\sum_{i=1}^{n} x_i \right) \cdot a + n \cdot b = \sum_{i=1}^{n} y_i \end{cases}$
4. Получается система линейных уравнений с двумя неизвестными a и b. Коэффициенты при неизвестных a и b находятся из исходных данных и являются постоянными для данной выборки. При различных x_i главный определитель системы не равен нулю:

$$\Delta = \begin{vmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{vmatrix} = n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(x_i - x_j \right)^2 \neq 0$$

5. Полученная система уравнений имеет единственное решение, которое находится по формулам Крамера:

$$a = \frac{\Delta_a}{\Delta} = \frac{n \cdot \sum_{i=1}^{n} x_i \cdot y_i - \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} y_i}{n \cdot \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2}$$

$$b = \frac{\Delta_b}{\Delta} = \frac{1}{n} \cdot \sum_{i=1}^{n} y_i - a \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

Подставим найденные значения в исходное уравнение и получим искомую линейную функцию, описывающую табличные данные.

Ниже представлен пример реализации метода наименьших квадратов в программе Mathcad (архив с файлом в приложении к лекции).

1. x и y – исходные табличные данные.
2. N – вспомогательная переменная для цикла, определяет количество элементов в столбце.
3. S(a,b) – функция квадрата отклонений.
4. Модуль Given – Find – задаёт и решает систему уравнений, представляющих собой частные производные от S(a,b).
5. a, b – искомые коэффициенты линейного уравнения.
6. F(x) – искомая функция.
7. График исходных данных и обобщающая функция.

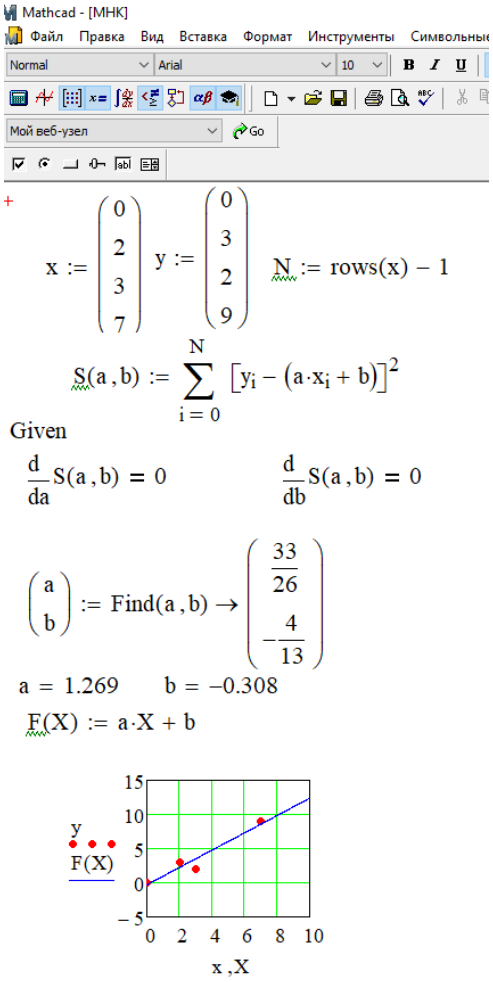

Рисунок 1 Реализация метода наименьших квадратов в Mathcad (архив МНК.zip в приложении к лекции)

Рассмотрим реализацию метода наименьших квадратов в Python (https://habr.com/ru/articles/672540/):

1. Функция **datasets_make_regression** – создаёт массив данных, на основе которых будет строиться обобщающая прямая.
2. Функция **coefficient_reg_inv** – находит коэффициенты a и b с помощью обратной матрицы для данных сгенерированных функцией **datasets_make_regression**.
3. Далее строится график с исходными данными и обобщающей прямой.

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

def datasets_make_regression(coef, data_size, noise_sigma, random_state):
    x = np.arange(0, data_size, 1.)
    mu = 0.0
    random.seed(random_state)
    noise = np.empty((data_size, 1))
    y = np.empty((data_size, 1))

    for i in range(data_size):
        noise[i] = random.gauss(mu, noise_sigma)
        y[i] = coef[0] + coef[1]*x[i] + noise[i]
```

```
    return x, y

coef_true = [34.2, 2.] # весовые коэффициенты
data_size = 200 # размер генерируемого набора данных
noise_sigma = 10 # СКО шума в данных
random_state = 42
x_scale, y_estimate = datasets_make_regression(coef_true, data_size,
noise_sigma, random_state)

plt.plot(x_scale, y_estimate, 'o')
plt.xlabel('x (порядковый номер измерения)', fontsize=14)
plt.ylabel('y (оценка температуры)', fontsize=14)



def coefficient_reg_inv(x, y):
    size = len(x)
    # формируем и заполняем матрицу размерностью 2x2
    A = np.empty((2, 2))
    A[[0], [0]] = sum((x[i]) ** 2 for i in range(0, size))
    A[[0], [1]] = sum(x)
    A[[1], [0]] = sum(x)
    A[[1], [1]] = size
    # находим обратную матрицу
    A = np.linalg.inv(A)
    # формируем и заполняем матрицу размерностью 2x1
    C = np.empty((2, 1))
    C[0] = sum((x[i] * y[i]) for i in range(0, size))
    C[1] = sum((y[i]) for i in range(0, size))

    # умножаем матрицу на вектор
    ww = np.dot(A, C)
    return ww[1], ww[0]


[w0_1, w1_1] = coefficient_reg_inv(x_scale, y_estimate)
print(w0_1, w1_1)

def predict(w0, w1, x_scale):
    y_pred = [w0 + val*w1 for val in x_scale]
    return y_pred

y_predict = predict(w0_1, w1_1, x_scale)

plt.plot(x_scale, y_estimate, 'o', label = 'Истинные значения')
plt.plot(x_scale, y_predict, '*', label = 'Расчетные значения')
plt.legend(loc = 'best', fontsize=12)
plt.xlabel('x (порядковый номер измерения)', fontsize=14)
plt.ylabel('y (оценка температуры)', fontsize=14)
plt.show()
```

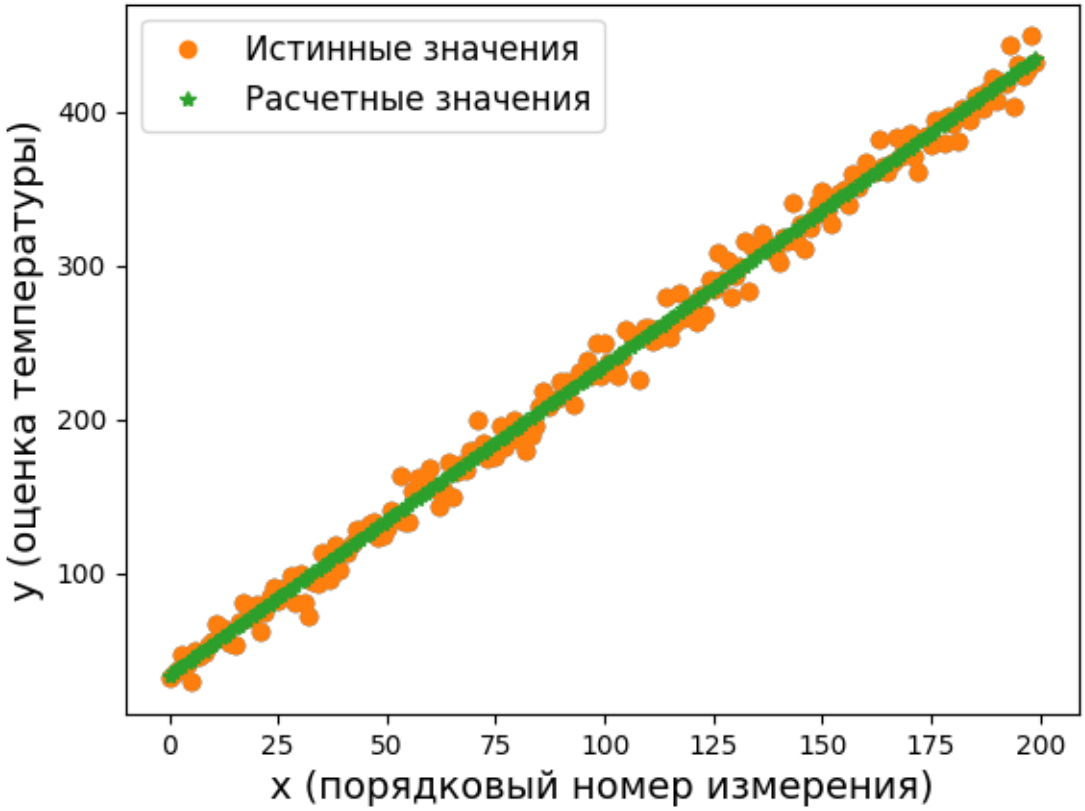

Рисунок 2 Пример реализации МНК в python