Отчёт ЛР 4 методы оптимизации

Чураков А А P3231, В-19

17 апреля 2025 г.

1 Постановка задачи

Найти минимум функции

$$f(x_1, x_2) = 2x_1^2 + 4x_2^2 - 5x_1x_2 + 11x_1 + 8x_2 - 3$$

методами:

- покоординатного спуска,
- градиентного спуска с фиксированным шагом,
- наискорейшего спуска (с поиском оптимального шага).

Точность остановки: $\varepsilon = 10^{-4}$. Начальное приближение:

$$x^{(0)} = (1, 1).$$

2 Метод покоординатного спуска

2.1 Ручные вычисления

Итерация 1. $M^0 = (1, 1)$.

Шаг 1: минимизация по x_1, $x_2 = 1$.

$$f(x_1, 1) = 2x_1^2 + 4 \cdot 1^2 - 5x_1 \cdot 1 + 11x_1 + 8 \cdot 1 - 3 = 2x_1^2 + 6x_1 + 9.$$

$$\frac{\partial f}{\partial x_1} = 4x_1 + 6 = 0 \quad \Longrightarrow \quad x_1^{(1)} = -\frac{6}{4} = -1.5.$$

Получаем $M^{1/2} = (-1.5, 1)$.

Шаг 2: минимизация по x_2, $x_1 = -1.5$.

$$\begin{aligned} f(-1.5, x_2) &= 2 \cdot (-1.5)^2 + 4x_2^2 - 5(-1.5)x_2 + 11(-1.5) + 8x_2 - 3 \\ &= 4.5 + 4x_2^2 + 7.5x_2 - 16.5 + 8x_2 - 3 = 4x_2^2 + 15.5x_2 - 15. \end{aligned}$$

$$\frac{\partial f}{\partial x_2} = 8x_2 + 15.5 = 0 \quad \Longrightarrow \quad x_2^{(1)} = -\frac{15.5}{8} = -1.9375.$$

Итак, после первой итерации:

$$M^1 = \big(-1.5,\ -1.9375\big).$$

Итерация 2. $M^1 = (-1.5, -1.9375)$.

Шаг 1: минимизация по x_1, $x_2 = -1.9375$.

$$\begin{aligned} f(x_1, -1.9375) &= 2x_1^2 + 4 \cdot (-1.9375)^2 - 5x_1(-1.9375) + 11x_1 + 8(-1.9375) - 3 \\ &= 2x_1^2 + 15.0156 + 9.6875x_1 + 11x_1 - 15.5 - 3 \\ &= 2x_1^2 + 20.6875x_1 - 3.4844. \end{aligned}$$

$$\frac{\partial f}{\partial x_1} = 4x_1 + 20.6875 = 0 \quad \Longrightarrow \quad x_1^{(2)} = -\frac{20.6875}{4} = -5.171875.$$

Получаем $M^{2/2} = (-5.171875, -1.9375)$.

Шаг 2: минимизация по x_2, $x_1 = -5.171875$.

$$\begin{aligned} f(-5.171875, x_2) &= 2 \cdot (-5.171875)^2 + 4x_2^2 - 5(-5.171875)x_2 + 11(-5.171875) + 8x_2 - 3 \\ &= 53.5156 + 4x_2^2 + 25.8594x_2 - 56.8906 + 8x_2 - 3 \\ &= 4x_2^2 + 33.8594x_2 - 6.3750. \end{aligned}$$

$$\frac{\partial f}{\partial x_2} = 8x_2 + 33.8594 = 0 \quad \Longrightarrow \quad x_2^{(2)} = -\frac{33.8594}{8} = -4.23242.$$

Итак,

$$M^2 = \big(-5.171875,\ -4.23242\big).$$

Итерация 3. $M^2 = (-5.171875, -4.23242)$.

Шаг 1: минимизация по x_1, $x_2 = -4.23242$.

$$f(x_1, -4.23242) = 2x_1^2 + (9.1 \ldots x_1\text{–term}) + \text{const},$$

$$\frac{\partial f}{\partial x_1} = 4x_1 + 32.1621 = 0 \quad \Longrightarrow \quad x_1^{(3)} \approx -8.04053.$$

Шаг 2: минимизация по x_2, $x_1 \approx -8.04053$.

$$\frac{\partial f}{\partial x_2} = 8x_2 + 48.2027 = 0 \quad \Longrightarrow \quad x_2^{(3)} \approx -6.02534.$$

Поэтому

$$M^3 \approx (-8.04053,\ -6.02534).$$

Вывод. После трёх полных итераций:

$$x^* \approx (-8.0405,\ -6.0253).$$

3 Метод градиентного спуска с фиксированным шагом

3.1 Ручные вычисления

Итерация 1. $M^0 = (1, 1)$.

$$\nabla f(1,1) = \begin{pmatrix} 4 \cdot 1 - 5 \cdot 1 + 11 \\ 8 \cdot 1 - 5 \cdot 1 + 8 \end{pmatrix} = \begin{pmatrix} 10 \\ 11 \end{pmatrix},$$

$$x^{(1)} = (1,1) - 0.1\,(10, 11) = (0,\ -0.1).$$

Итерация 2. $M^1 = (0, -0.1)$.

$$\nabla f(0, -0.1) = \begin{pmatrix} 4 \cdot 0 - 5(-0.1) + 11 \\ 8(-0.1) - 5 \cdot 0 + 8 \end{pmatrix} = \begin{pmatrix} 11.5 \\ 7.2 \end{pmatrix},$$

$$x^{(2)} = (0, -0.1) - 0.1\,(11.5, 7.2) \approx (-1.15,\ -0.82).$$

Итерация 3. $M^2 \approx (-1.15, -0.82)$.

$$\nabla f(-1.15, -0.82) \approx \begin{pmatrix} 4(-1.15) - 5(-0.82) + 11 \\ 8(-0.82) - 5(-1.15) + 8 \end{pmatrix} \approx \begin{pmatrix} 10.5 \\ 7.19 \end{pmatrix},$$

$$x^{(3)} \approx (-1.15, -0.82) - 0.1\,(10.5, 7.19) \approx (-2.20,\ -1.539).$$

Вывод. После трёх итераций:

$$x^* \approx (-2.20,\ -1.54).$$

4 Метод наискорейшего спуска

4.1 Ручные вычисления

Итерация 1. $M^0 = (1, 1)$.

$$\nabla f(1,1) = (10, 11), \quad d^{(0)} = -\nabla f(1,1) = (-10, -11).$$
$$\varphi(\lambda) = f\big((1,1) + \lambda\, d^{(0)}\big) = f(1 - 10\lambda,\ 1 - 11\lambda).$$

$$\begin{aligned}\varphi(\lambda) &= 2(1-10\lambda)^2 + 4(1-11\lambda)^2 - 5(1-10\lambda)(1-11\lambda)\\ &\quad + 11(1-10\lambda) + 8(1-11\lambda) - 3\\ &= 134\lambda^2 - 221\lambda + 17.\end{aligned}$$

$$\varphi'(\lambda) = 268\lambda - 221 = 0 \quad\Longrightarrow\quad \lambda_0 = \frac{221}{268} \approx 0.825.$$
$$x^{(1)} = (1,1) + 0.825\,(-10, -11) \approx (-7.25,\ -8.08).$$

Итерация 2. $M^1 \approx (-7.25, -8.08)$.

$$\nabla f(-7.25, -8.08) \approx (-2.65,\ 3.89), \quad d^{(1)} = (2.65,\ -3.89).$$
$$\varphi(\lambda) = f\big(M^1 + \lambda\, d^{(1)}\big) \approx \alpha\lambda^2 + \beta\lambda + \gamma \quad\Longrightarrow\quad \lambda_1 \approx 0.074.$$
$$x^{(2)} \approx (-7.25, -8.08) + 0.074\,(2.65, -3.89) \approx (-7.05,\ -8.37).$$

Итерация 3. Градиент в M^2 уже мал, дальнейшие шаги незначительны, принимаем

$$x^* \approx (-7.05,\ -8.37).$$

5 Программная реализация на Python

```
import numpy as np
from scipy.optimize import minimize_scalar

def f(x):
    """ : f(x1, x2) = 2x1^2 + 4x2^2 - 5x1x2 + 11x1 + 8x2 - 3"""
    return 2 * x[0]**2 + 4 * x[1]**2 - 5 * x[0] * x[1] + 11 * x[0] + 8 * x[1] - 3

def grad_f(x):
    """  """
    return np.array([4 * x[0] - 5 * x[1] + 11, 8 * x[1] - 5 * x[0] + 8])

def coord_descent(x0, epsilon=1e-4, max_iter=1000):
    """
          f.

    Args:
        x0 (numpy.ndarray):   .
        epsilon (float): .
        max_iter (int):    .

    Returns:
        tuple:       .
    """

    x = x0.copy()
    f_values = [f(x)]
    x_history = [x.copy()] # Store the history of x

    for _ in range(max_iter):
        x_prev = x.copy()
```

```
        #   x1
        res_x1 = minimize_scalar(lambda x1: f([x1, x[1]]))
        x[0] = res_x1.x

        #   x2
        res_x2 = minimize_scalar(lambda x2: f([x[0], x2]))
        x[1] = res_x2.x

        x_history.append(x.copy())
        f_values.append(f(x))

        if np.abs(f(x) - f(x_prev)) < epsilon:
            break

    return x, f(x), x_history, f_values


def gradient_descent(x0, epsilon=1e-4, learning_rate=0.01, max_iter=1000):
    """
          f.

    Args:
        x0 (numpy.ndarray):  .
        epsilon (float): .
        learning_rate (float):   .
        max_iter (int):   .

    Returns:
        tuple:       .
    """

    x = x0.copy()
    f_values = [f(x)]
    x_history = [x.copy()] # Store the history of x

    for _ in range(max_iter):
        x_prev = x.copy()
        grad = grad_f(x)
        x = x - learning_rate * grad

        x_history.append(x.copy())
        f_values.append(f(x))

        if np.linalg.norm(x - x_prev) < epsilon or np.abs(f(x) - f(x_prev)) < epsilon:
            break

    return x, f(x), x_history, f_values

def steepest_descent(x0, epsilon=1e-4, max_iter=1000):
    """
          f.

    Args:
        x0 (numpy.ndarray):  .
        epsilon (float): .
        max_iter (int):   .

    Returns:
        tuple:       .
    """

    x = x0.copy()
    f_values = [f(x)]
    x_history = [x.copy()]

    for _ in range(max_iter):
```

```
        x_prev = x.copy()
        grad = grad_f(x)
        direction = -grad

        #
        res_alpha = minimize_scalar(lambda alpha: f(x + alpha * direction))
        alpha = res_alpha.x

        x = x + alpha * direction

        x_history.append(x.copy())
        f_values.append(f(x))


        if np.linalg.norm(x - x_prev) < epsilon or np.abs(f(x) - f(x_prev)) < epsilon:
            break

    return x, f(x), x_history, f_values

#
x0 = np.array([0.0, 0.0])

#
x_min_coord_descent, f_min_coord_descent, coord_descent_history, coord_descent_f_values =
    coord_descent(x0)
x_min_grad_descent, f_min_grad_descent, grad_descent_history, grad_descent_f_values =
    gradient_descent(x0, learning_rate=0.01) # You might need to tune learning_rate
x_min_steepest_descent, f_min_steepest_descent, steepest_descent_history,
    steepest_descent_f_values = steepest_descent(x0)

#
print("  :")
print(" :", x_min_coord_descent)
print("  :", f_min_coord_descent)

print("\n  :")
print(" :", x_min_grad_descent)
print("  :", f_min_grad_descent)

print("\n  :")
print(" :", x_min_steepest_descent)
print("  :", f_min_steepest_descent)
```