Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК4 Программное обеспечение ЭВМ, информационные технологии

ПРАКТИЧЕСКАЯ РАБОТА

«Методы минимизации»

по дисциплине: «Методы машинного обучения»

Выполнил: студент группы ИУК4-72Б _____________ (Подпись) Губин Е.В. (И.О. Фамилия)

Проверил: _____________ (Подпись) Семененко М.Г. (И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:
- Оценка:

Калуга, 2025

Цель: изучить и реализовать метод половинного деления для нахождения минимума функции одной переменной и сравнить его эффективность с встроенными методами оптимизации.

Задачи:

- Реализовать алгоритм метода половинного деления и метода градиентного спуска для поиска минимума функции.
- Применить методы к заданной функции
- Построить график функции с отмеченной точкой минимума и сравнить результат с встроенной функцией Python.

Вариант №4

Формулировка задания №1:

1. Написать блок-схему алгоритма нахождения минимума функции методом половинного деления.
2. Методом половинного деления найти минимум функции J(u) на отрезке [-10; 10] (по вариантам). Построить график функции и показать на нем точку минимума. Функция: $u^2 + \alpha e^{\beta u}$; a = 4; b = -0.25.
3. Сравнить результат с результатом использования встроенной функции.

Блок схема метода дихотомии:

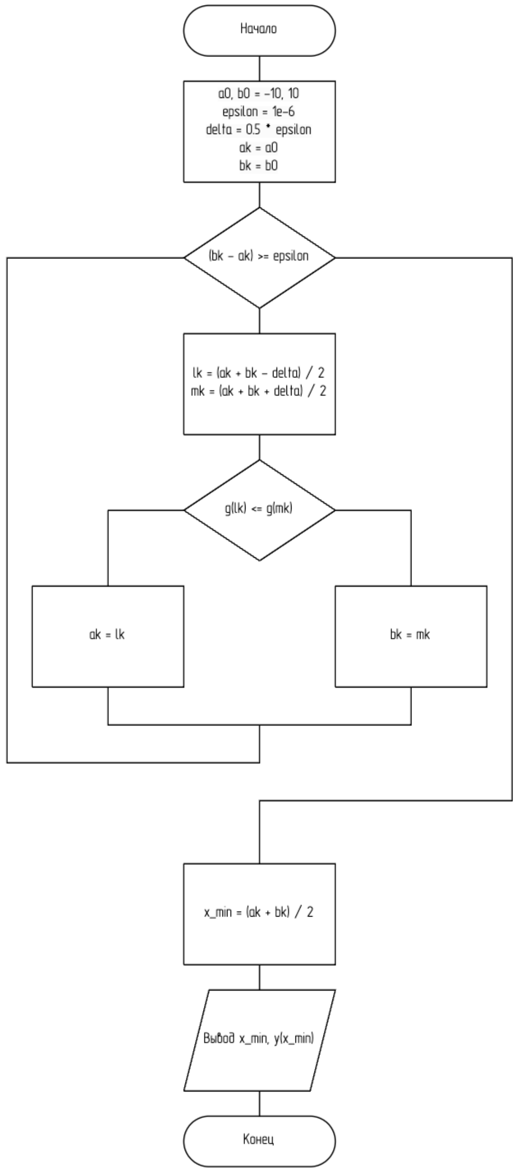

Рисунок 1 Блок схема метода дихотомии

Результаты выполнения программы для метода дихотомии:

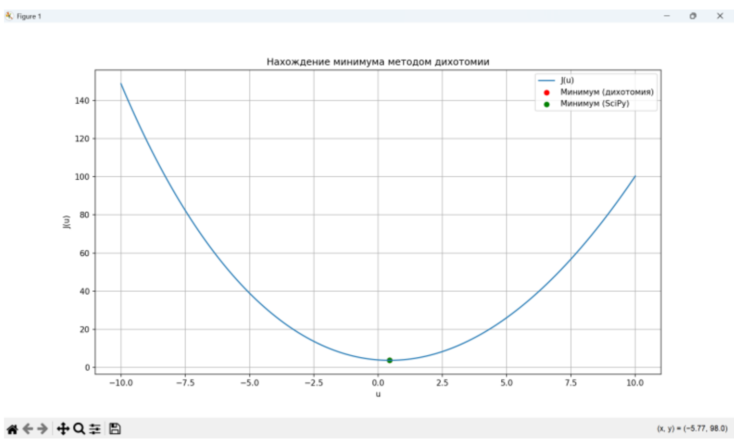

Рисунок 2 Нахождение минимума методом дихотомии

На Рисунках 2 и 3 видно, что минимум на отрезке, найденный по методу дихотомии приблизительно равен минимуму, найденному с помощью встроенной функции Python scipy.optimize.minimize_scalar.

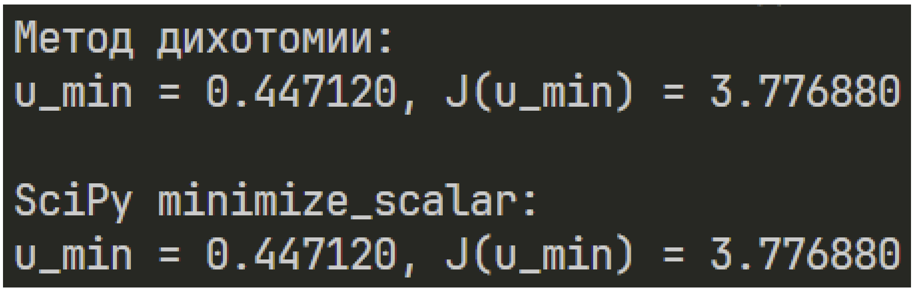

Рисунок 3 Сравнение результата нахождения минимума по методу дихотомии с встроенной функцией поиска минимума

Формулировка задания №2:

1. Написать блок-схему алгоритма нахождения минимума функции двух переменных методом градиентного спуска.
2. Допустим, что задана решающая функция линейного классификатора в упрощенном виде (по вариантам).
   Найти координаты и значение функции в точке минимума методом градиентного спуска.
3. Сравнить результат с результатом использования встроенной функции (показать график).
   Решающая функция:
   $$f(x_1, x_2) = x_1^3 + x_2^2 - x_1 \cdot x_2 - 2x_1 + 3x_2 - 4$$

Блок-схема алгоритма метода градиентного спуска:

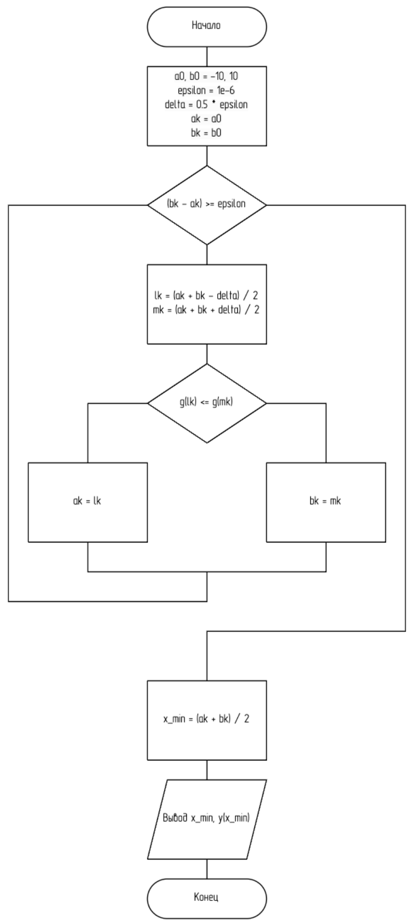

Рисунок 4 Блок-схема алгоритма минимизации функции по методу градиентного спуска

Результаты выполнения программы для метода градиентного спуска:

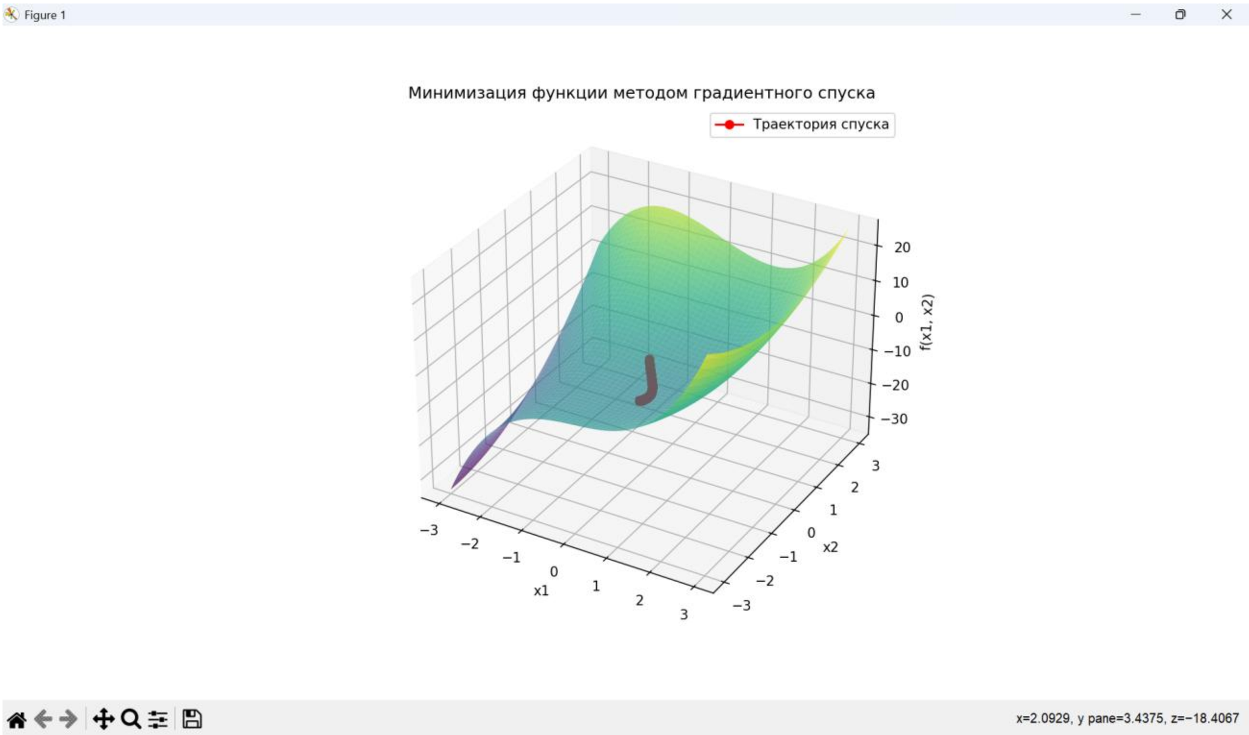

Рисунок 5 Нахождение минимума методом градиентного спуска

На Рисунках 5 и 6 видно, что минимум на отрезке, найденный по методу дихотомии приблизительно равен минимуму, найденному с помощью встроенной функции Python scipy.optimize.minimize_scalar.

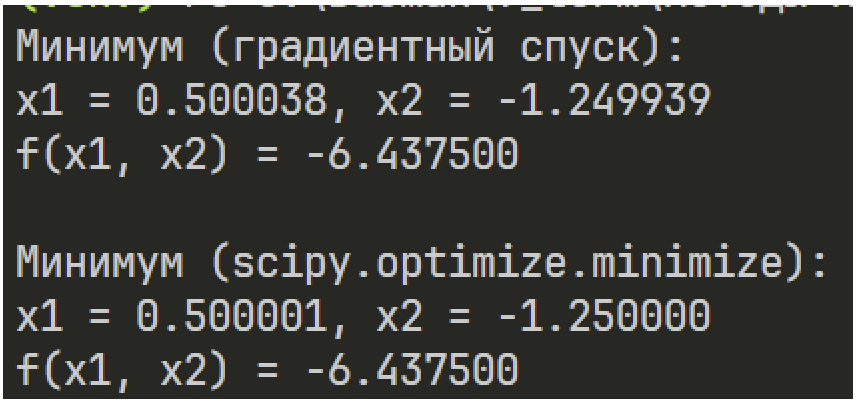

Рисунок 6 Сравнение результата нахождения минимума по методу градиентного спуска с встроенной функцией поиска минимума

Листинг программы метода дихотомии:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize_scalar

a = 4
b = -0.25

def J(u):
    return u**2 + a * np.exp(b * u)
```

```
def dichotomy_method(func, left, right, eps=1e-5, delta=1e-4,
max_iter=10000):
    iteration = 0
    while abs(right - left) > eps and iteration < max_iter:
        x1 = (left + right - delta) / 2
        x2 = (left + right + delta) / 2
        f1, f2 = func(x1), func(x2)

        if f1 < f2:
            right = x2
        else:
            left = x1

        iteration += 1

    return (left + right) / 2

u_min = dichotomy_method(J, -10, 10)
J_min = J(u_min)

res = minimize_scalar(J, bounds=(-10, 10), method='bounded')

print("Метод дихотомии:")
print(f"u_min = {u_min:.6f}, J(u_min) = {J_min:.6f}")
print("\nSciPy minimize_scalar:")
print(f"u_min = {res.x:.6f}, J(u_min) = {res.fun:.6f}")

u = np.linspace(-10, 10, 400)
plt.plot(u, J(u), label='J(u)')
plt.scatter(u_min, J_min, color='red', label='Минимум (дихотомия)')
plt.scatter(res.x, res.fun, color='green', label='Минимум (SciPy)')
plt.title("Нахождение минимума методом дихотомии")
plt.xlabel("u")
plt.ylabel("J(u)")
plt.legend()
plt.grid(True)
plt.show()
```

Листинг программы по методу градиентного спуска:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import minimize

def f(x):
    x1, x2 = x
    return x1**3 + x2**2 - x1*x2 - 2*x1 + 3*x2 - 4

def grad_f(x):
    x1, x2 = x
    df_dx1 = 3*x1**2 - x2 - 2
    df_dx2 = 2*x2 - x1 + 3
    return np.array([df_dx1, df_dx2])

def gradient_descent(start, learning_rate=0.01, eps=1e-6, max_iter=10000):
    x = np.array(start, dtype=float)
    path = [x.copy()]

    for i in range(max_iter):
        grad = grad_f(x)
        x_new = x - learning_rate * grad
```

```
        if np.linalg.norm(x_new - x) < eps:
            break
        x = x_new
        path.append(x.copy())

    return x, f(x), np.array(path)

start = [0, 0]
learning_rate = 0.01

x_min, f_min, path = gradient_descent(start, learning_rate)

print("Минимум (градиентный спуск):")
print(f"x1 = {x_min[0]:.6f}, x2 = {x_min[1]:.6f}")
print(f"f(x1, x2) = {f_min:.6f}")

res = minimize(f, start)
print("\nМинимум (scipy.optimize.minimize):")
print(f"x1 = {res.x[0]:.6f}, x2 = {res.x[1]:.6f}")
print(f"f(x1, x2) = {res.fun:.6f}")

x1_vals = np.linspace(-3, 3, 100)
x2_vals = np.linspace(-3, 3, 100)
X1, X2 = np.meshgrid(x1_vals, x2_vals)
Z = f([X1, X2])

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X1, X2, Z, cmap='viridis', alpha=0.7)
ax.plot(path[:,0], path[:,1], [f(p) for p in path], color='red', marker='o',
label='Траектория спуска')

ax.set_title('Минимизация функции методом градиентного спуска')
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('f(x1, x2)')
ax.legend()
plt.show()
```

Вывод: в ходе лабораторной работы был реализован алгоритм поиска минимума функции методом дихотомии (с заданным максимальным количеством итераций) и произведено сравнение с результатом поиска минимума встроенной библиотечной функцией.