Методы нулевого и первого порядка

Лабораторная работа №1

Иванов Владимир, M3235,
Шепелев Матвей, M3235
Зубарев Денис, M3235
Команда `new m3235::bald[3]`

2025-03-29

Содержание

1 Постановка задачи

В данной работе исследуются методы оптимизации первого порядка, в частности, различные варианты градиентного спуска.

1.1 Градиентный спуск

Градиентный спуск - способ нахождения минимума функции с помощью движения вдоль ее градиента.

$$x_{k+1} = x_k - h(k)\nabla f(x_k)$$

Задача лабораторной работы - реализовать несколько видов градиентного спуска и исследовать различия в их эффективности при работе на функциях двух переменных.

2 Реализованные методы

2.1 Градиентный спуск с различными стратегиями выбора шага

Нами был самостоятельно реализован спуск с возможностью изменения стратегии выбора шага (где стратегия выбора шага это $h : \mathbb{N} \to \mathbb{R}$) и вместе с этим сами стратегии:

1. Константная (в тестовой реализации $h = 0.1$)
2. Экспоненциальная ($h = e^{-\lambda k}$, в тестовой реализации $\lambda = 0.5, \lambda = 0.3$)

Вместе с этим реализованы два критерия остановки:

1. Относительная по координате ($\|x_{k+1} - x_k\| < \varepsilon(\|x_{k+1}\| + 1)$) (при измерении результатов пользуемся именно им)
2. Относительная по значению функции ($\|\nabla f(x_k)\|^2 < \varepsilon \ \|\nabla f(x_0)\|^2$)

Дополнительно есть предохранитель: ограничение на количество операций (10000 в тестовой реализации).

2.2 Градиентный спуск на основе одномерного поиска

Также мы самостоятельно написали наискорейший (steepest) градиентный спуск.

Принцип его работы - ищем h^*, такой что значение $f(x_k - h^*\nabla f(x_k))$ минимально. Этот h^* ищется с помощью одномерного поиска в направлении антиградиента.

Мы реализовали два вида одномерного поиска:

1. Поиск методом дихотомии (тернарный). Реализован итеративно.
2. Неточный поиск по правилу Армихо (метод backtracking). c_1, q выбираются случайно, α - точка пересечения прямой, заданной c_1, и оси OX.

Градиент квадратичной функции вычисляется аналитически, для не квадратичной используем приближенную симметричную производную.

$$f_i'(x) = \frac{f_i(x+\varepsilon) - f_i(x-\varepsilon)}{2\varepsilon}, i \in \{0, 1\}$$

Если градиент равен 0, то используется случайное смещение на ε.

Также для сравнения мы взяли поиск на основе сильного условия Вульфе из библиотеки Scipy (`scipy.optimize.line_search`). У него есть одна особенность - если поиск не сходится, то он возвращает `None`. В таком случае мы запускаем наш поиск методом backtracking.

3 Результаты работы методов

Эффективность методов исследуем на следующих функциях:

3.1 Квадратичная функция q_1

$$q_1 = \left(A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, C = 1.4 \right)$$

метод	x0	итераций	вычислений функции	вычислений градиента	погрешность
LR const(0.1)	[0,0]	73	0	73	5.77e-15
LR exp(0.5)	[0,0]	34	0	34	1.07e-4
LR exp(0.3)	[0,0]	44	0	44	1.09e-7
LR const(0.1)	[1,4]	10002	0	10002	1.33e-15
LR exp(0.5)	[1,4]	38	0	38	4.82e-3
LR exp(0.3)	[1,4]	51	0	51	4.89e-6
LR const(0.1)	[1,1]	10002	0	10002	8.88e-16
LR exp(0.5)	[1,1]	36	0	36	9.64e-4
LR exp(0.3)	[1,1]	48	0	48	9.79e-7
Armijo GD	[0,0]	20	1614	20	7.99e-15
Armijo GD	[1,4]	19	20835	19	1.78e-15
Armijo GD	[1,1]	18	21332	18	4.44e-15
Dichotomy GD	[0,0]	14	843	14	2.22e-16
Dichotomy GD	[1,4]	250	15031	250	1.33e-15
Dichotomy GD	[1,1]	344	20663	344	4.44e-16
Scipy Wolfe GD	[0,0]	2	20017	5	0.0
Scipy Wolfe GD	[1,4]	2	20017	5	0.0
Scipy Wolfe GD	[1,1]	2	20017	5	0.0

Максимально стандартная, хорошо обусловленная функция с единственным глобальным минимум в точке (0, 0). Результаты не очень интересные: почти все алгоритмы управились примерно за одно количество итераций (за исключением градиентного спуска с

константным шагом). Спуску на основе одномерного поиска потребовалось заметно больше вычислений функций, но в то же время в среднем он был точнее поиска с планированием шага.

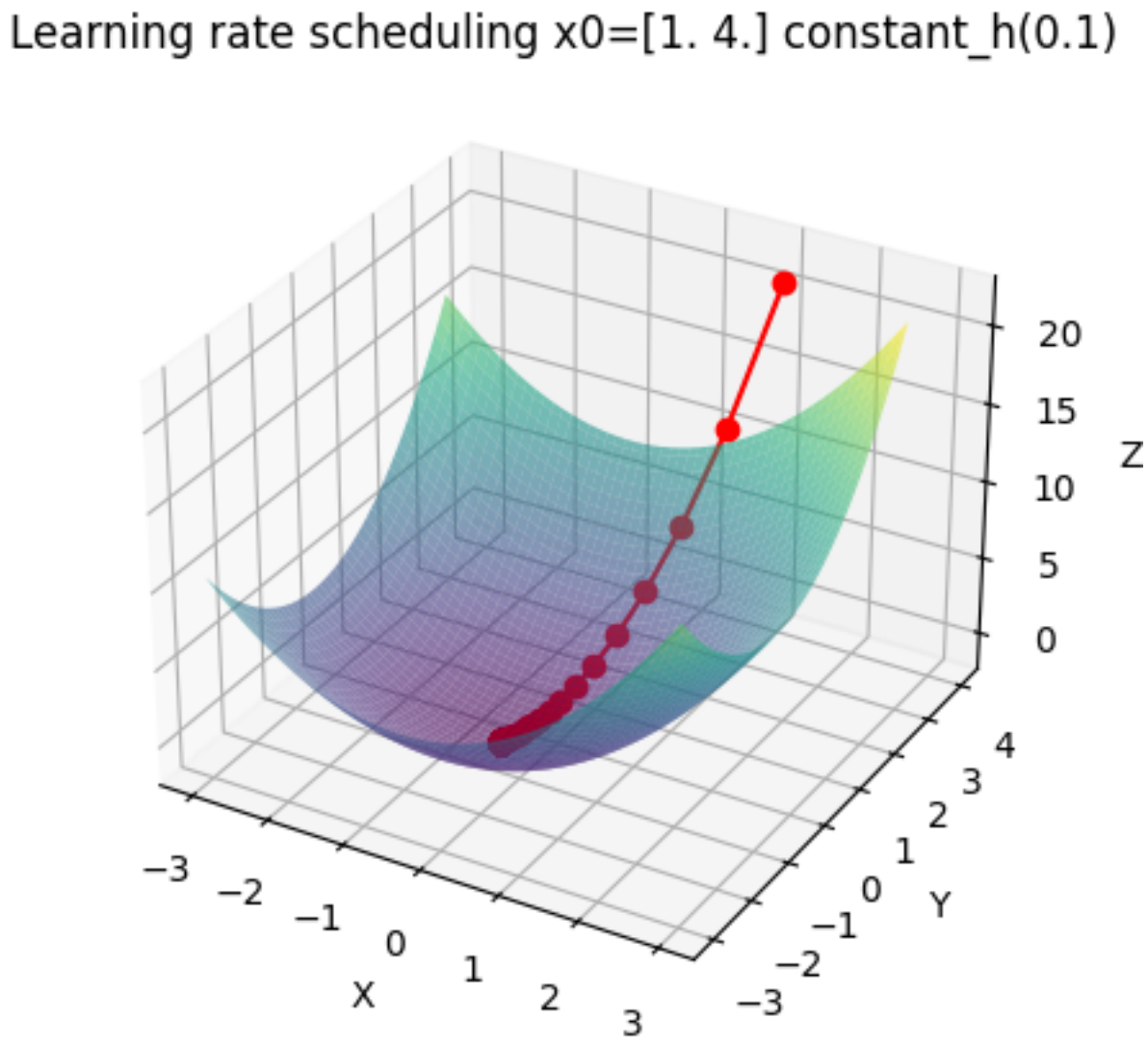

Рис. 1: ГС с константным шагом

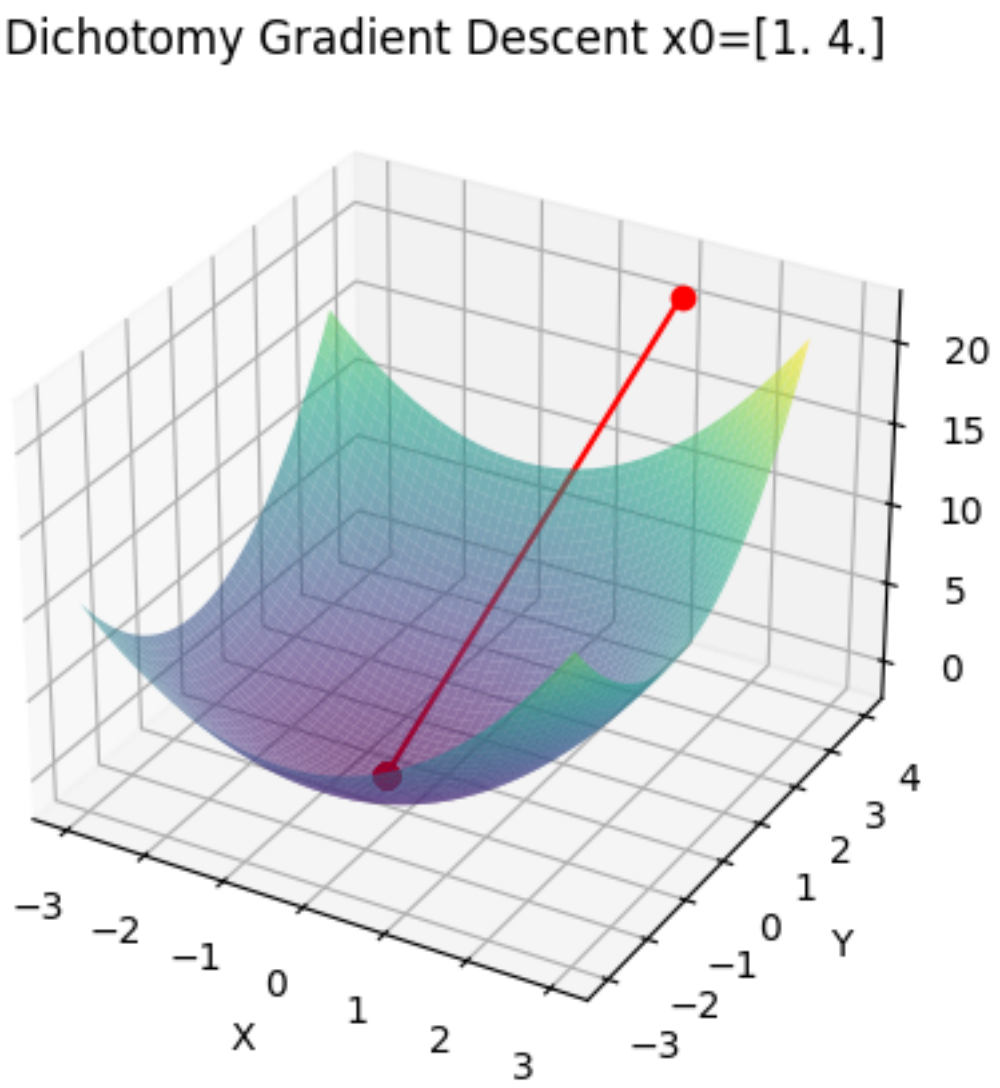

Рис. 2: ГС на основе дихотомии

3.2 Квадратичная функция q_2:

$$q_2 = \left(A = \begin{pmatrix} 0.1 & 0 \\ 0 & 3 \end{pmatrix}, B = (0 \;\; 0), C = 0.0 \right)$$

метод	x0	итераций	вычислений функции	вычислений градиента	погрешность
LR const(0.1)	[1,4]	721	0	721	2.89e-14
LR exp(0.5)	[1,4]	45	0	45	2.39
LR exp(0.3)	[1,4]	62	0	62	1.97e-2
LR const(0.1)	[1,1]	719	0	719	2.63e-14
LR exp(0.5)	[1,1]	43	0	43	1.81e-1
LR exp(0.3)	[1,1]	61	0	61	1.94e-2
Armijo GD	[1,4]	1315	4407	1315	1.95e-12
Armijo GD	[1,1]	1289	24477	1289	2.04e-12
Dichotomy GD	[1,4]	18	1004	18	1.12e-14
Dichotomy GD	[1,1]	14	799	14	3.05e-15
Scipy Wolfe GD	[1,4]	123	20328	369	5.91e-15
Scipy Wolfe GD	[1,1]	194	20587	583	2.26e-14

Унимодальная функция с плохой обусловленностью (condition number = 30). Разброс в результатах тут побольше.

Плохо показал себя спуск на основе backtracking: из-за неточности поиска по правилу Армихо (и, возможно, неудачно выбранных c_1, q) ему потребовалось достаточно много итераций.

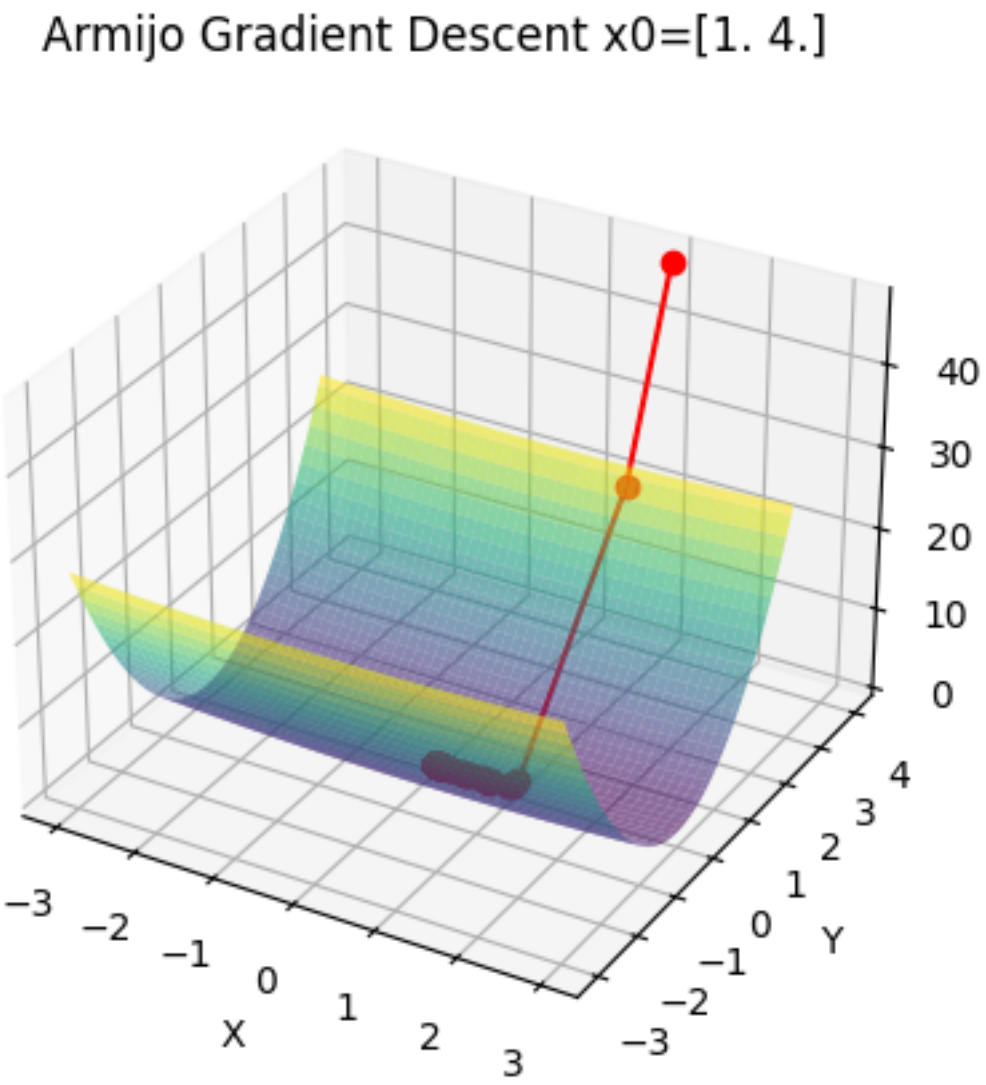

Рис. 3: ГС с поиском на основе backtracking

С другой стороны, особенно сильно тут засиял ГС на основе дихотомии: он смог максимально быстро найти точный минимум, что неудивительно, ведь эта функция - фактически идеальный юзкейс для тернарного поиска.

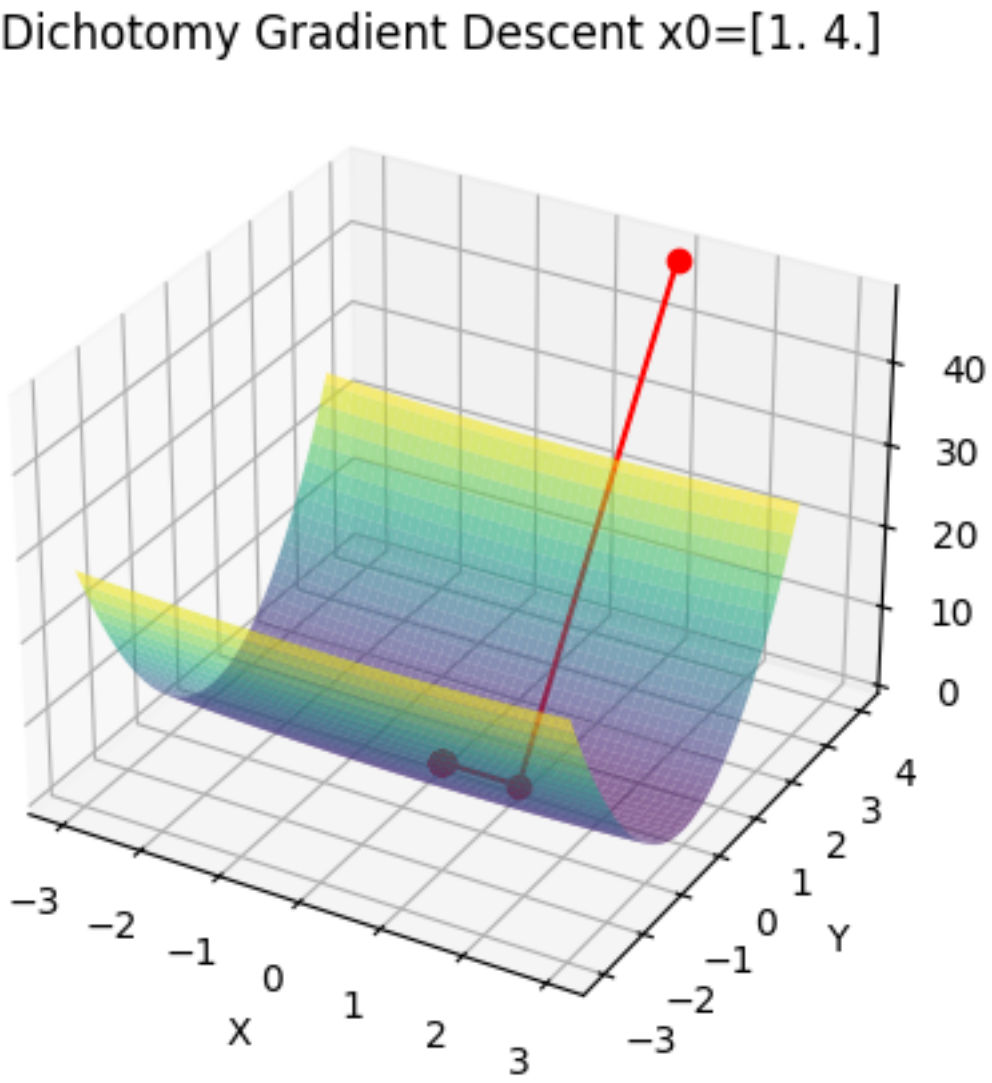

Рис. 4: график ГС на основе дихотомии

Сильно оплошал градиентный спуск с экспоненциальным шагом ($\lambda = 0.5$). Он не сумел найти минимум и в целом пошел куда-то не туда.

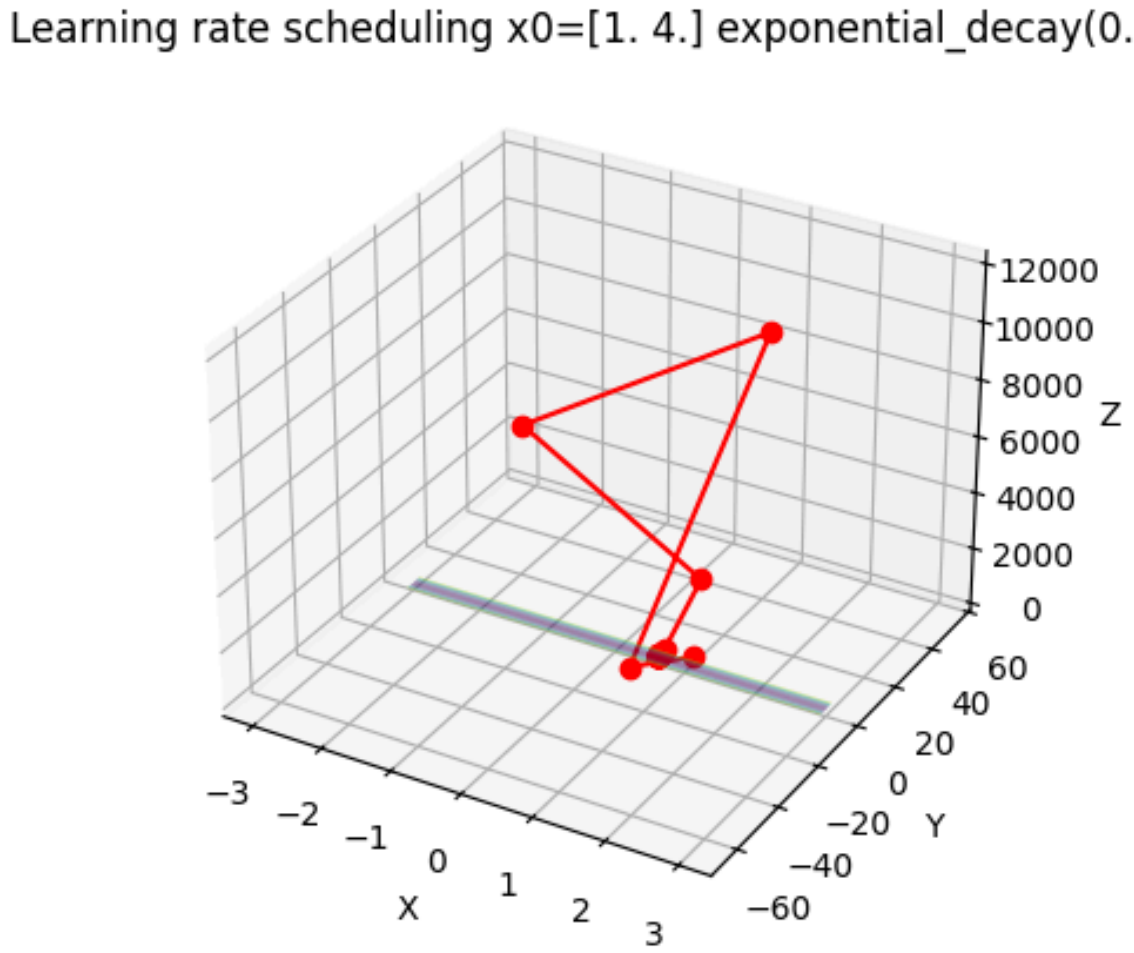

Рис. 5: ГС на основе экспонецниального выбора шага

3.3 Мультимодальная функция f_4

$$f_4 = \left(x^2 - 1\right)^2 + y^2 + 0.5x$$

метод	x0	итераций	вычислений функции	вычислений градиента	погрешность
LR const(0.1)	[0,0]	1295	0	1295	9.07e-13
LR exp(0.5)	[0,0]	8	0	8	2.80e+117
LR exp(0.3)	[0,0]	7	0	7	2.14e+118
LR const(0.1)	[1,4]	1980	0	1980	9.98e-1
LR exp(0.5)	[1,4]	38	0	38	1.00e+0
LR exp(0.3)	[1,4]	50	0	50	3.48e-6
LR const(0.1)	[1,1]	7036	0	7036	9.98e-1
LR exp(0.5)	[1,1]	36	0	36	9.99e-1
LR exp(0.3)	[1,1]	45	0	45	2.19e-7
Armijo GD	[0,0]	11	20609	11	3.11e-12
Armijo GD	[1,4]	36	21746	36	1.37e-11
Armijo GD	[1,1]	45	23365	45	5.13e-13
Dichotomy GD	[0,0]	2	113	2	1.11e-16
Dichotomy GD	[1,4]	10	569	10	9.98e-1
Dichotomy GD	[1,1]	13	745	13	9.98e-1
Scipy Wolfe GD	[0,0]	4	20027	13	5.83e-13
Scipy Wolfe GD	[1,4]	28	20134	83	1.11e-16
Scipy Wolfe GD	[1,1]	25	20134	73	4.23e-13

Мультимодальная функция с глобальным минимумом в точке

$$x_0 \approx (-1.058, 0),\ \ f(x_0) \approx -0.515$$

и еще одним локальным в точке

$$x_1 \approx (0.930, 0),\ \ f(x_1) \approx 0.483$$

Градиентный спуск на основе дихотомии в двух из трех случаев нашел неверный минимум (неудивительно, ведь тернарный поиск работает корректно только с унимодальными функциями).

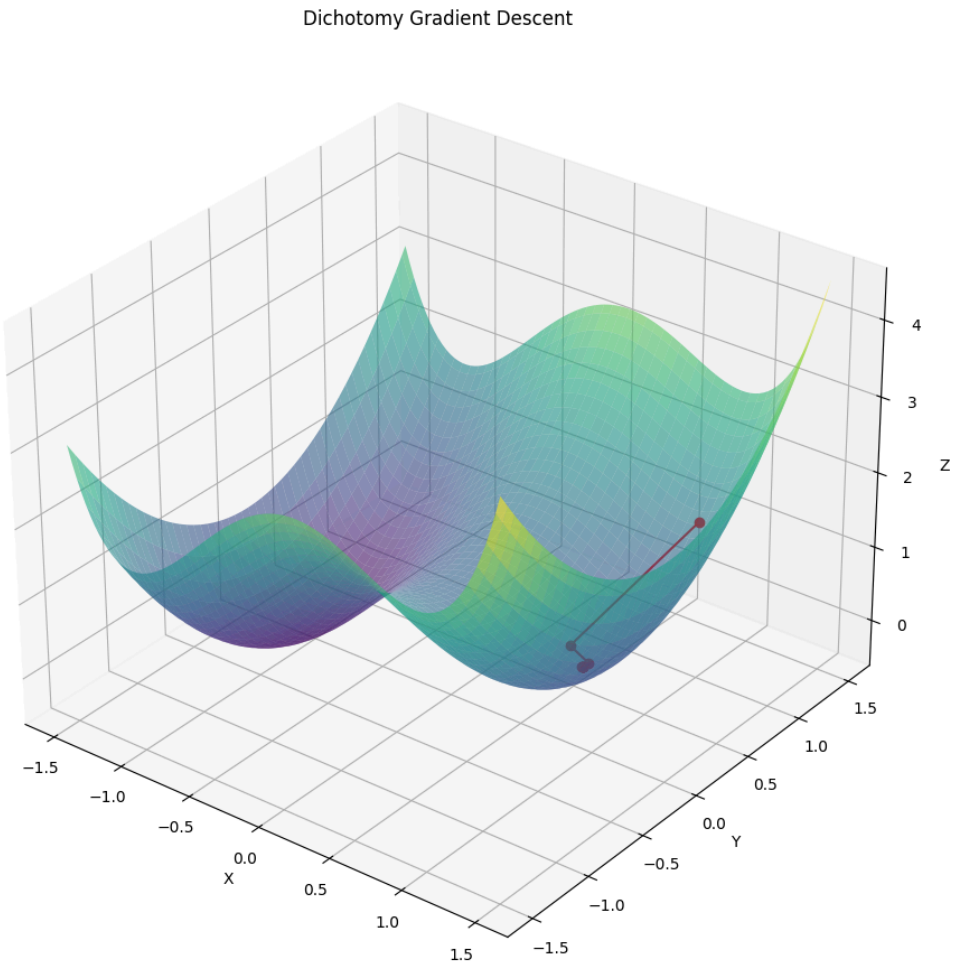

Рис. 6: мимо :(

И поиск на основе правила Армихо, и на основе условий Вульфа смогли найти глобальный минимум с достаточно малой погрешностью, однако для этого им потребовалось достаточно много значений функции.

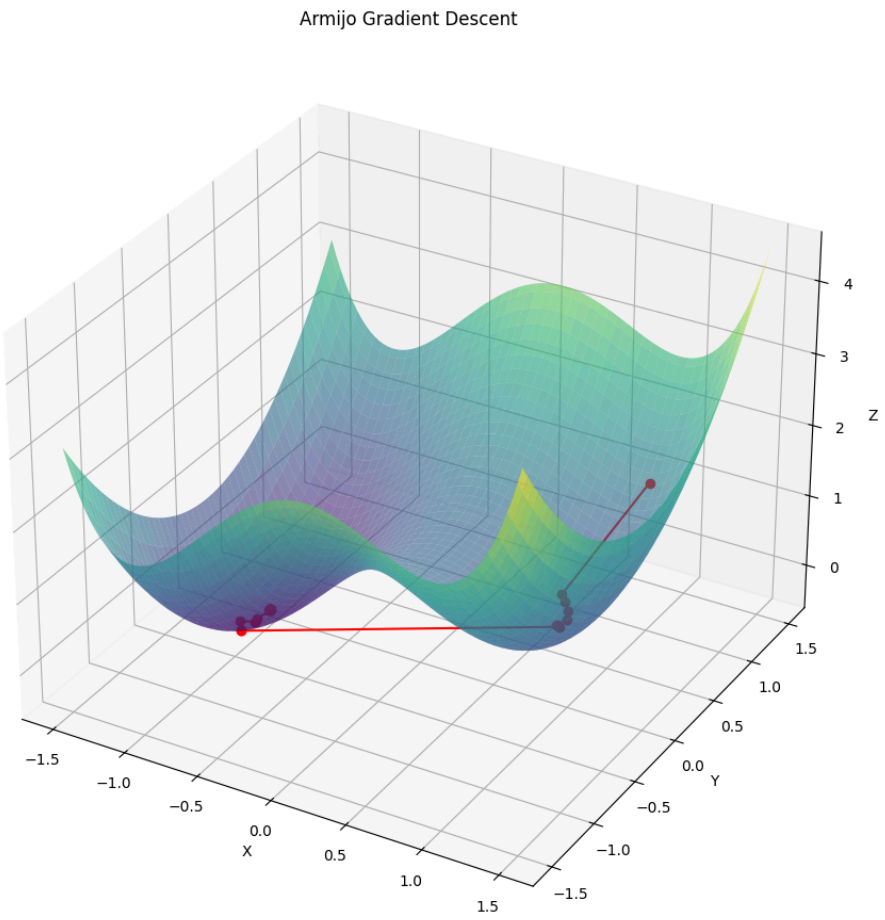

Рис. 7: в точку :)

Среди ГС со стратегией выбора шага с задачей не справился ни один.

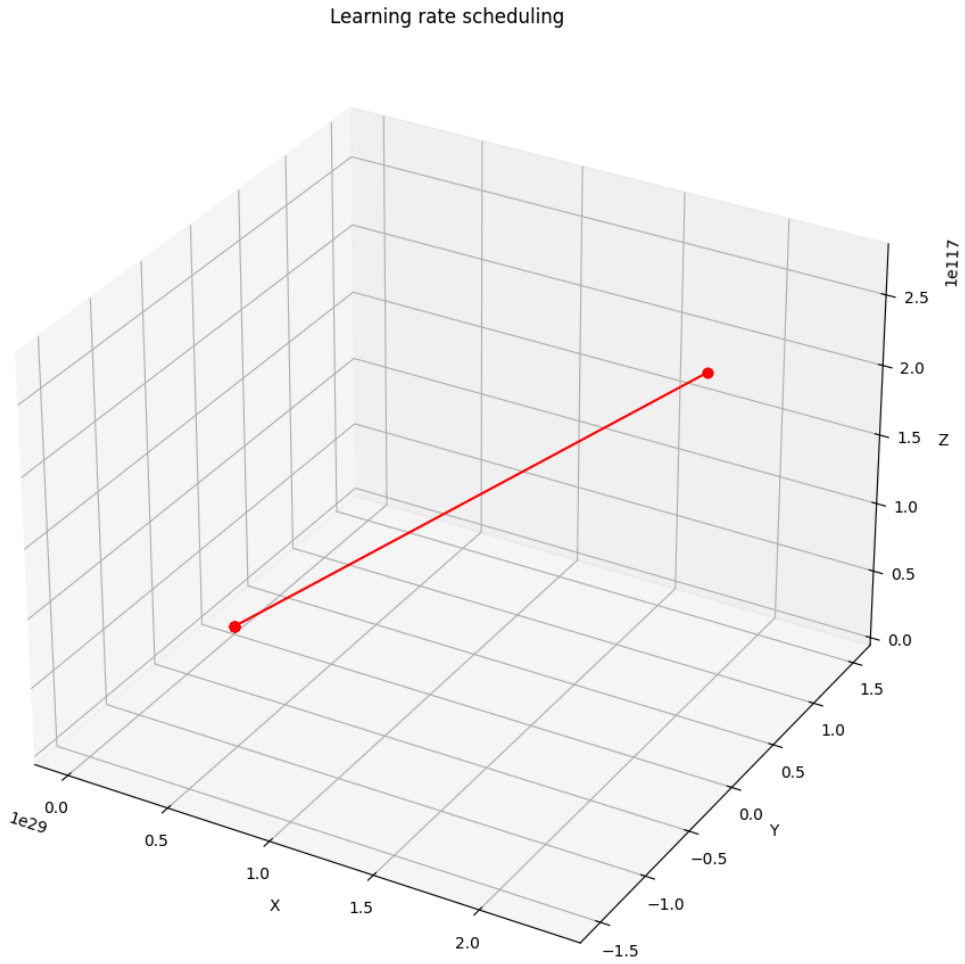

Рис. 8: ???????????? (экспоненциальная стратегия)

3.4 Зашумленная f_4.

Для зашумления существующих функций использовался следующий класс с кастомизируемым фактором шума (0.3 в тестовой реализации):

```
class NoisyWrapper:
    f: BiFunc
    factor: float

    def __init__(self, f: BiFunc, factor: float = 1.0):
        self.f = f
        self.factor = factor

    def __call__(self, x: np.ndarray):
        return self.f.__call__(x) + random.random() * self.factor

    def gradient(self, x: np.ndarray):
        return self.f.gradient(x)
```

метод	x0	итераций	вычислений функции	вычислений градиента	погрешность
LR const(0.1)	[0,0]	1238	0	1238	6.92e-2
LR exp(0.5)	[0,0]	8	0	8	2.80e+117
LR exp(0.3)	[0,0]	7	0	7	2.14e+118
LR const(0.1)	[1,4]	83	0	83	1.06
LR exp(0.5)	[1,4]	38	0	38	1.09
LR exp(0.3)	[1,4]	50	0	50	2.31e-1
LR const(0.1)	[1,1]	5300	0	5300	1.28
LR exp(0.5)	[1,1]	36	0	36	1.07
LR exp(0.3)	[1,1]	45	0	45	2.39e-1
Armijo GD	[0,0]	895	14935	895	1.93e-1
Armijo GD	[1,4]	4081	67705	4081	3.28e-1
Armijo GD	[1,1]	314	5234	314	1.27e-1
Dichotomy GD	[0,0]	10002	608011	10002	8.17e-2
Dichotomy GD	[1,4]	10002	608011	10002	1.76e-1
Dichotomy GD	[1,1]	10002	608359	10002	2.62e-1
Scipy Wolfe GD	[0,0]	1880	37471	6543	1.66e-1
Scipy Wolfe GD	[1,4]	2148	43464	7490	7.39e-2
Scipy Wolfe GD	[1,1]	7222	146795	25245	1.92e-1

Все ГС с выбором шага не сумели найти глобальный минимум и/или улетели очень далеко. ГС на основе дихотомии не смогла сойтись и вышла по предохранителю (опять же, неудивительно), но все же приблизилась к минимуму. ГС на основе Армихо/Вульфа смогли достаточно близко приблизиться к глобальному минимуму, однако Scipy Wolfe потребовалось больше итераций и вычислений значений функций при старте из $x_0 = (1, 1)$

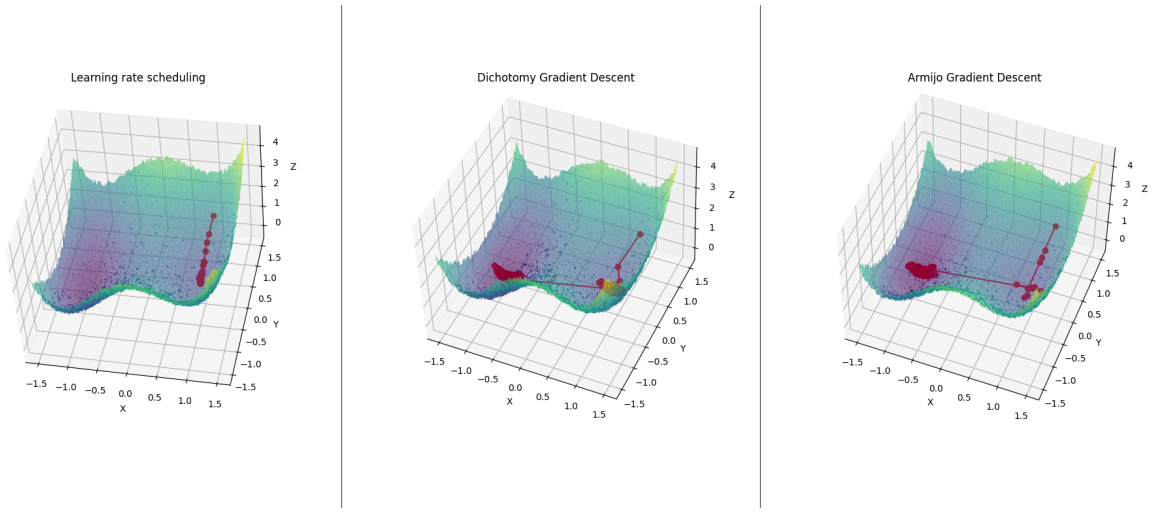

Рис. 9: зашумленная функция

4 Вывод

В результате лабораторной работы мы реализовали несколько методов градиентного спуска и проанализировали их эффективность на нескольких функциях.

Наименее эффективен градиентный спуск с постоянным шагом, т.к он требует наибольшее число итераций. ГС с экспоненциальным шагом нередко не только не находит минимум, но и выходит за пределы допустимой области, так что часто он просто не работает.

ГС на основе дихотомии прекрасно работает на унимодальных функциях, но неэффективен на мультимодальных. ГС на основе правила Вульфа из Scipy - самый точный из всех исследуемых, но в некоторых случаях требует больше вычислений.

Наиболее эффективным и универсальным оказался поиск на основе правила Армихо, обеспечивающий максимальный баланс между точностью и скоростью работы.

При этом важно заметить, что эффективность работы методов сильно зависит от функции: так, на хорошо обусловленной унимодальной функции все методы отработали примерно одинаково, но на других уже стали проявляться различия. Наиболее устойчивыми оказались ГС на основе правила Армихо и условий Вульфа – во всех случаях они смогли найти минимум с достаточной точностью.

5 Приложения

Исходный код можно найти на: *https://github.com/postmodernist1848/metopt-lab1*