МО26 · ДЗ1 · Вариант №63

Памятка к защите

Для Николаева Степана. Здесь всё объяснено с нуля; на защите достаточно говорить простыми словами.

Оглавление

1. Что вообще требуется 1
2. Что означает аналитическая часть 1
3. Наискорейший спуск 2
4. Имитация отжига 2
5. Разбор ноутбука по ячейкам 2
6. Какие результаты надо знать 3
7. Что могут спросить 4
8. Как рассказывать на защите 4

1. Что вообще требуется

У нас есть функция от трёх переменных: подставляешь x_1, x_2, x_3 — получаешь число. Надо подобрать такие три числа, чтобы результат был наименьшим.

Аналитически — решить обычную систему уравнений с производными. **Численно** — заставить Python двигаться из заданной далёкой точки к минимуму.

> Старт: (−97.47; −64.17; −10.99). Настоящий минимум: (1.263265; −1.416327; 0.445918).

Стартовое значение функции очень большое (примерно 1 833 499), а минимум около −3.584. Алгоритмы должны его уменьшить.

2. Что означает аналитическая часть

Градиент — это три первые производные в одном векторе. Он показывает направление самого быстрого роста функции. Если взять градиент с минусом, пойдём в сторону уменьшения.

Стационарная точка — там, где все первые производные равны нулю. Поэтому в отчёте мы решаем систему из трёх уравнений $\nabla f = 0$.

Матрица Гессе H — таблица вторых производных. У нас она постоянная, потому что функция квадратичная.

Критерий Сильвестра — проверяем три числа (угловые миноры): 132, 1172, 19600. Все положительные, значит H положительно определена.

Что это даёт: график функции имеет форму единственной «чаши». Поэтому стационарная точка — единственный глобальный минимум.

Почему минимум вообще существует? Квадратичная часть с положительно определённой H растёт при удалении от центра. Получается $f(x) \to +\infty$ при $||x|| \to \infty$; непрерывная функция достигает минимума.

3. Наискорейший спуск

Алгоритм смотрит, куда растёт функция (градиент g), идёт в противоположную сторону и выбирает лучший размер шага α. Так повторяется много раз.

$$x^{k+1} = x^k - \alpha_k \cdot g_k, \quad g_k = \nabla f(x^k).$$

Шаг для нашей квадратичной функции можно найти без перебора: подставить $x - \alpha g$ в f, взять производную по α и приравнять к нулю.

$$\alpha = (g \cdot g) / (g \cdot H \cdot g).$$

Символ @ в Python — умножение матриц/векторов: g @ g — квадрат длины градиента, g @ H @ g — число, которое получается через Гессиан.

Когда остановимся: самая большая по модулю из трёх производных станет меньше 10^{-6}. Это означает, что уклон в каждой координате почти нулевой.

Почему не берём просто фиксированный шаг: здесь лучший α вычисляется точно и метод так называется — наискорейший спуск.

4. Имитация отжига

Это случайный метод: рядом с текущей точкой берётся новая случайная. Если там функция меньше — переходим. Если больше — иногда тоже переходим: так метод не боится временного ухудшения.

$$y = x + \delta, \quad \text{каждый } \delta \text{ от } -0.1 \text{ до } 0.1; \quad T_k = 100/k.$$

$$\Delta f = f(y) - f(x); \quad \text{если } \Delta f \le 0 \text{ — принимаем; иначе вероятность } p = \exp(-\Delta f/T_k).$$

Температура сначала высокая — вероятность принять ухудшение выше. Затем T уменьшается — алгоритм становится более осторожным. Для случайных чисел используется генератор NumPy.

seed=42 фиксирует последовательность случайных чисел. Поэтому запуск у тебя и у проверяющего даёт одинаковый результат.

Важный нюанс условия ДЗ. Там сказано завершать метод, если соседние значения f отличаются меньше 10^{-6}. Если сравнивать вообще все попытки и отклонённую точку считать тем же x, то после первого отказа разность окажется ровно нулевой и алгоритм остановится почти сразу. Поэтому в нашем коде соседними приближениями считаются **принятые** точки: после отказа генерируем новую точку на той же итерации. Если спросят — именно так и объясни.

Из-за случайности метод отжига обычно менее точный и может потребовать больше проб. Число Iterations здесь означает число переходов к очередной принятой точке, а не общее число случайных попыток.

5. Разбор ноутбука по ячейкам

Ячейка 1 — import numpy as np. Подключаем библиотеку для векторов и случайных чисел. Других библиотек здесь не нужно.

Ячейка 2 — данные. f — сама функция; x0 — старт; epsilon = 10^{-6} — точность; H — Гессиан; x_star и f_star — точный ответ из аналитической части, чтобы сравнить с

численным.

Ячейка 3 — функции. function_value(point) считает f; gradient(point) считает три производные; error(value) считает абсолютную погрешность.

steepest_descent(start, eps). В цикле вычисляет g, проверяет остановку, считает α, делает новый шаг. В конце возвращает точку, градиент, значение, число шагов и ошибку.

simulated_annealing(start, eps, seed=42). Создаёт случайную точку, считает изменение функции, случайно принимает/отклоняет, уменьшает температуру, проверяет изменение двух принятых значений. Возвращает точку, температуру, значение, шаги, ошибку.

Ячейка 4. Запускаем спуск и подписанно выводим Point, Gradient, Value, Iterations, Error.

Ячейка 5. Запускаем отжиг и выводим Point, Temperature, Value, Iterations, Error.

6. Какие результаты надо знать

Точный ответ из аналитического решения:

$x^* = (619/490;\ -347/245;\ 437/980),\quad f(x^*) = -1405/392 \approx -3.584184.$

Численные результаты конкретного запуска ноутбука:

Method: Steepest Descent

Point: [1.263265 -1.41632616 0.44591824]

Gradient: [-9.99053309e-07 5.44755707e-08 -5.39751632e-07]

Value: -3.5841836734691745

Iterations: 1668

Error: 2.1316282072803006e-13

Method: Simulated Annealing

Point: [1.2270456 -1.37742334 0.44544054]

Temperature: 0.009250693802035153

Value: -3.577685319198001

Iterations: 10810

Error: 0.006498354271386564

Численная погрешность — разница между найденным значением f и точным f_{min}; у разных методов она не обязана быть одинаковой.

7. Что могут спросить

Вопрос	Короткий ответ
Что мы ищем?	Три координаты, при которых функция принимает наименьшее значение.
Что такое глобальный минимум?	Меньше значения функции не будет нигде на всей R^3.
Зачем градиент?	Показывает рост; идём в противоположную сторону.
Что такое Гессиан?	Матрица вторых производных функции.
Зачем Сильвестр?	Убедиться, что Гессиан положительно определён и минимум один.
Почему решаем $\nabla f=0$?	В точке минимума без ограничений первые производные равны нулю.
Откуда α?	Минимизируем $f(x - \alpha g)$ по одному числу α.
Почему остановился спуск?	Максимум модулей трёх производных стал меньше ε.
Что такое температура?	Параметр вероятности принять худшую точку; он уменьшается.
Зачем отжиг принимает плохой шаг?	Чтобы случайно пройти через временное ухудшение.
Почему seed?	Чтобы случайный результат можно было повторить.
Что такое Δ?	Абсолютная ошибка по значению функции относительно точного минимума.
Почему разные результаты?	Спуск выбирает точный шаг вдоль градиента, а отжиг случайный.
Можно было np.linalg.solve?	Для аналитической проверки да, но сама численная оптимизация написана вручную.

8. Как рассказывать на защите

«У меня квадратичная функция трёх переменных. Сначала я выписал градиент и Гессиан. По Сильвестру все угловые миноры положительные, поэтому функция строго выпуклая и бесконечно растущая. Затем решил систему из трёх производных и нашёл точный минимум.

В программе реализовал два метода. Наискорейший спуск идёт против градиента и на каждом шаге выбирает оптимальную длину шага. Отжиг проверяет случайные соседние точки и иногда допускает ухудшение с вероятностью, зависящей от температуры. В конце сравниваю найденное значение функции с аналитическим».