Ермолаева А. В.
группа: НПМбв-01-22

Отчет по лабораторной работе №4:
Методы оптимизации в машинном обучении

1. Эксперимент 3.1: Траектория градиентного спуска на квадратичной функции

Цель эксперимента:

Исследовать поведение градиентного спуска на квадратичных функциях с разными числами обусловленности, начальными точками и стратегиями выбора шага.

Методика:

1. Квадратичные функции:
    1) Сферическая ($k \approx 1$): A = [[1, 0], [0, 1]], b = [0, 0]
    2) Вытянутый эллипсоид ($k = 100$): A = [[100, 0], [0, 1]], b = [0, 0]
    3) Коррелированная: A = [[10, 7], [7, 5]], b = [0, 0]

2. Начальные точки:
    1) x_{01} = [10.0, 10.0]
    2) x_{02} = [1.0, 10.0]
    3) x_{03} = [-5.0, -5.0]

3. Стратегии выбора шага:
    1) Constant с c = 0.01
    2) Constant с c = 0.1
    3) Armijo с c_1 = 1e-4, α_0 = 1.0
    4) Wolfe с c_1 = 1e-4, c_2 = 0.9, α_0 = 1.0

4. Параметры алгоритма:
    1) Точность: ε = 1e-8
    2) Максимальное число итераций: 1000

Результаты:

1.1 Сферическая функция, $k \approx 1$:

Стратегия	Итерации	Статус	f(x*)	$\|\nabla f(x^*)\|$
Constant (0.01)	917	success	9.884322e-07	1.406010e-03
Constant (0.1)	88	success	8.844671e-07	1.330013e-03
Armijo	1	success	0.000000e+00	0.000000e+00
Wolfe	1	success	0.000000e+00	0.000000e+00

Табл. 1. Результаты экспериментов для сферической функции ($k \approx 1$).

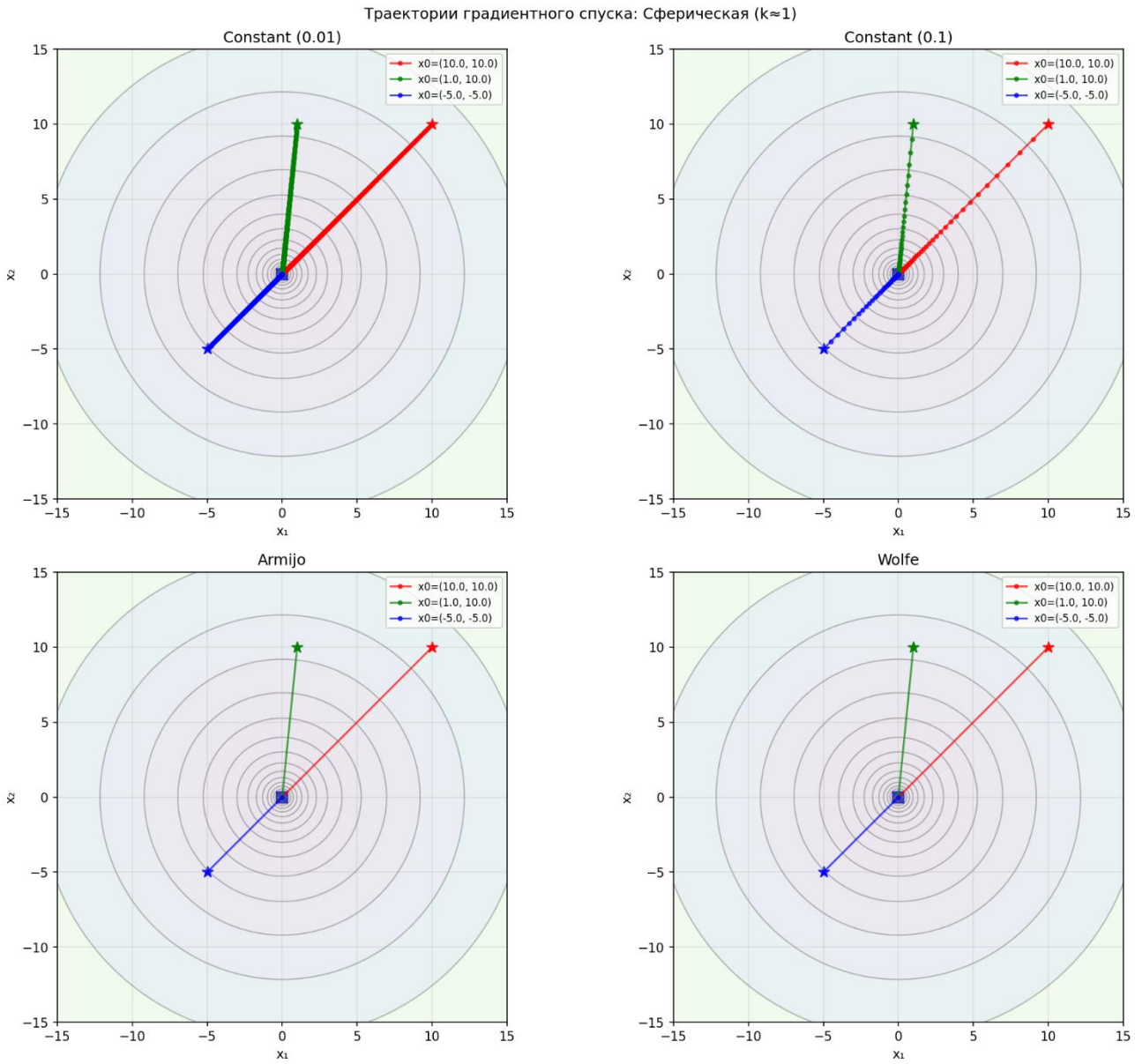

Рис. 1. Траектории градиентного спуска для сферической функции (k≈1).

Выводы:

1) Все стратегии успешно сходятся к минимуму [0, 0]
2) Адаптивные стратегии (Armijo, Wolfe) значительно эффективнее константных
3) Constant (0.1) в ≈10 раз быстрее Constant (0.01)
4) Для k≈1 адаптивные методы находят точное решение за одну итерацию
5) Слишком малый константный шаг приводит к избыточным вычислениям

1.2 Вытянутая функция, k=100:

Стратегия	Итерации	Статус	f(x*)	‖∇f(x*)‖
Constant (0.01)	459	success	4.921286e-03	9.920974e-02
Constant (0.1)	1000	iterations_exceeded	nan	nan
Armijo	293	success	4.909839e-03	9.909429e-02
Wolfe	3	success	9.629650e-33	1.387779e-15

Табл. 2. Результаты экспериментов для вытянутой функции (k=100).

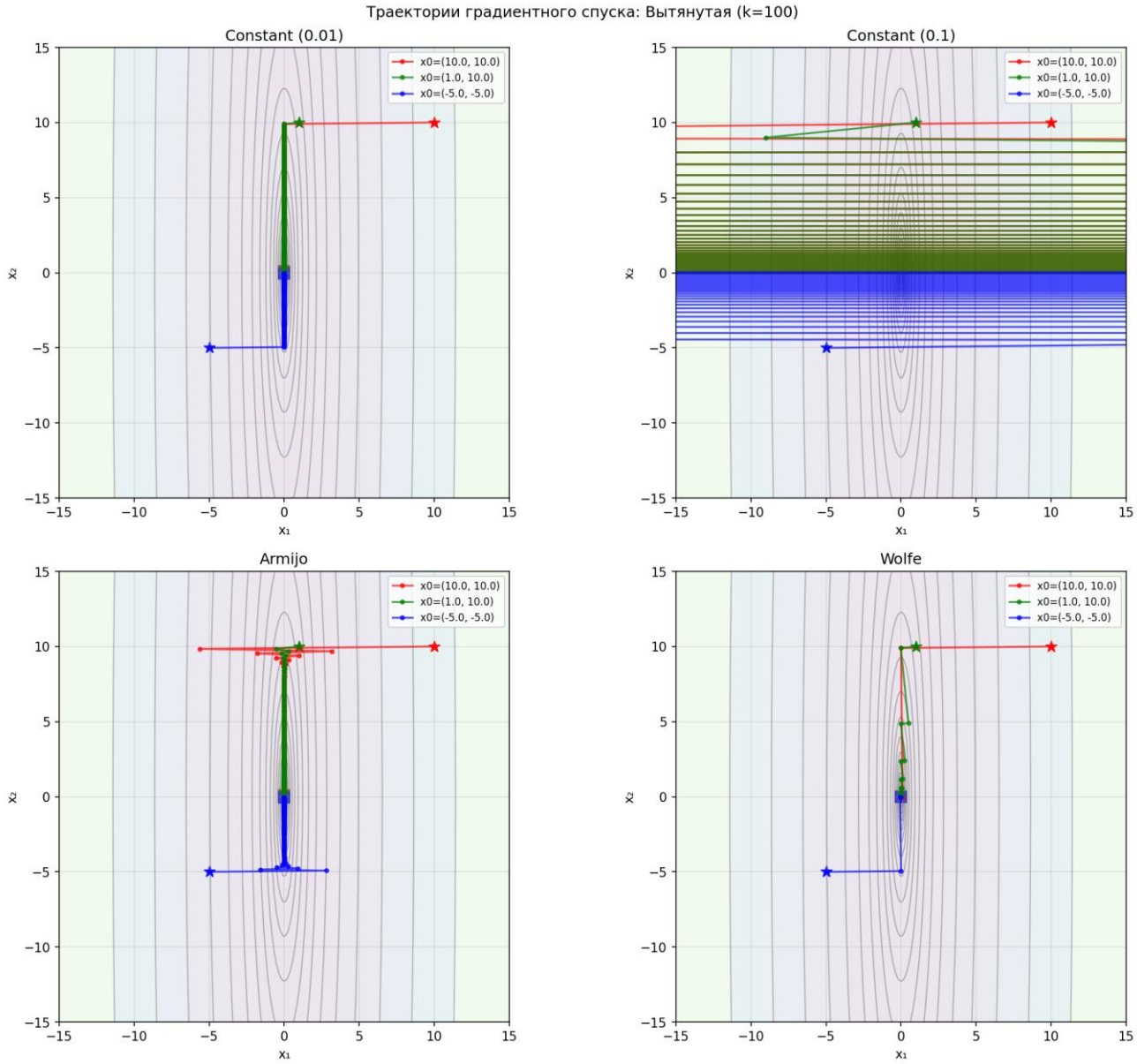

Рис. 2. Траектории градиентного спуска для вытянутой функции (k=100).

Выводы:

1) Наблюдается "зигзагообразное" поведение вдоль оврага.
2) Constant (0.01): медленно сходится (459 итераций), не достигает высокой точности
3) Constant (0.1): расходится (1000 итераций), результат NaN – слишком большое значение
4) Armijo: устойчиво сходится (293 итерации), умеренная точность
5) Wolfe: наилучший результат (3 итерации), практически идеальная точность
6) Для плохо обусловленных задач (k=100) константные стратегии ненадежны

1.3 Коррелированная функция (недиагональная матрица):

Стратегия	Итерации	Статус	f(x*)	$\|\nabla f(x^*)\|$
Constant (0.01)	1000	iterations_exceeded	5.108899e-02	8.271887e-02
Constant (0.1)	306	success	3.193923e-03	2.068251e-02
Armijo	244	success	3.225971e-03	2.078601e-02
Wolfe	227	success	1.799045e-03	2.046841e-02

Табл. 3. Результаты экспериментов для коррелированной функции (недиагональная матрица).

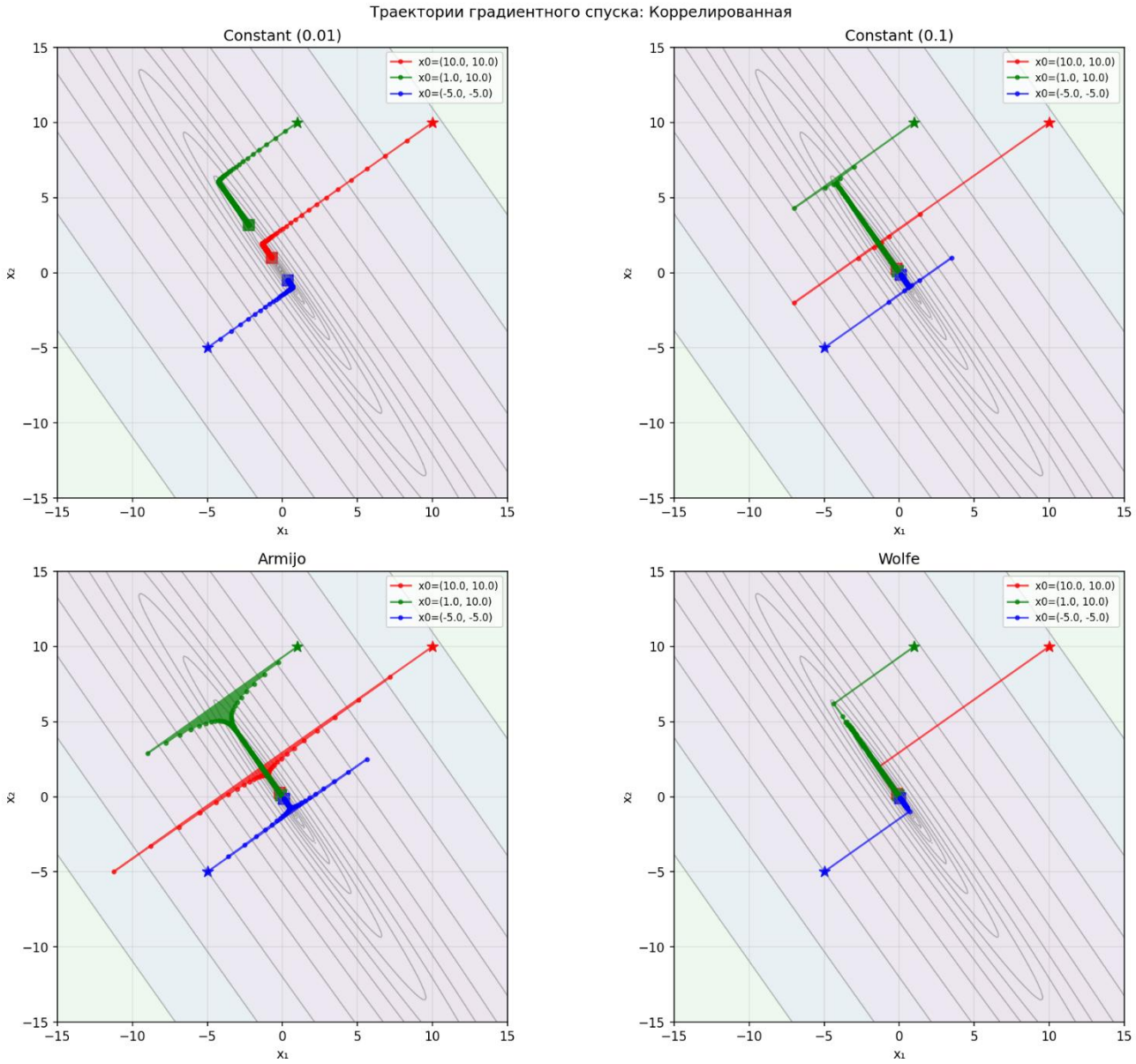

Рис. 3. Траектории градиентного спуска для коррелированной функции.

Выводы:

1) Траектории искривлены из-за недиагональности матрицы.
2) Constant (0.01): не сходится за 1000 итераций – слишком малый шаг неэффективен
3) Constant (0.1): медленно сходится за 306 итераций – средний шаг не оптимален
4) Armijo: 244 итерации, точность сравнима с Constant (0.1)
5) Wolfe: лучшая точность за 227 итераций

Выводы по эксперименту:

1) Число обусловленности существенно влияет на поведение метода: при больших k траектория становится зигзагообразной.
2) Адаптивные стратегии (Wolfe, Armijo) эффективнее постоянного шага, особенно для плохо обусловленных задач.
3) Стратегия Wolfe показывает наилучшие результаты благодаря сочетанию условий достаточного убывания и кривизны.
4) Начальная точка влияет на траекторию, но для выпуклых квадратичных функций не влияет на возможность сходимости.

2. Эксперимент 3.2: Зависимость числа итераций от числа обусловленности и размерности

Цель эксперимента:

Исследовать зависимость числа итераций T(k, n) от числа обусловленности k и размерности пространства n.

Методика:

1. Генерация случайных квадратичных задач:
   1) Матрица A = diag(a), где $a_i \in [1, k]$, min(a) = 1, max(a) = k
   2) Вектор b ~ N (0, 1)
   3) Для каждого набора параметров генерируется 3 задачи (n_repeats = 3)

2. Параметры эксперимента:
   1) Размерности: n = 10, 50, 100
   2) Числа обусловленности: $k \in [1, 1\,000]$ (15 точек в логарифмической шкале)
   3) Точность: ε = 1e-6
   4) Стратегия: Wolfe (c_1 = 1e-4, c_2 = 0.9)
   5) Максимальное число итераций: 10 000
   6) Воспроизводимость: seed = 42 * repeat + n_idx * 1000 + k_idx * 10000

3. Метрика: число итераций до выполнения критерия остановки $\|\nabla f(x_k)\|^2 \le \varepsilon \|\nabla f(x_0)\|^2$

Результаты:

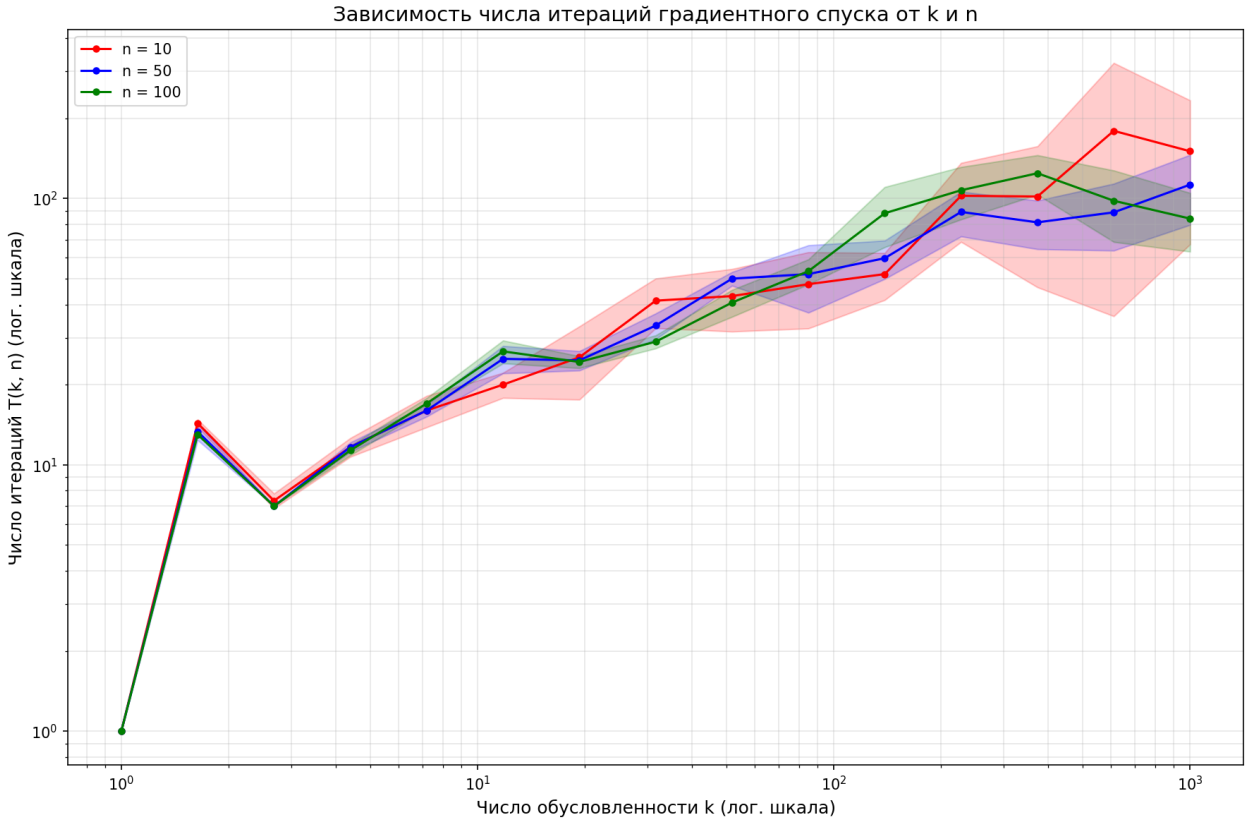

Рис. 4. Зависимость числа итераций от числа обусловленности k для разных размерностей n.

На графике в двойном логарифмическом масштабе видны три семейства кривых (для n = 10, 50, 100). Все кривые демонстрируют линейный рост в логарифмических координатах.

Размерность	Средние итерации	Мин	Макс	Ст. отклонение	Зависимость T∝	Корреляция
n=10	54.2	1	378	70.2	k^0.532	0.883
n=50	44.4	1	154	36.5	k^0.507	0.897
n=100	48.3	1	148	42.0	k^0.526	0.898

Таблица 4.1. Статистика по размерностям

k (обусловленность)	Итерации n=10	Итерации n=50	Итерации n=100
1.0	1.0	1.0	1.0
1.6	14.3	13.3	13.0
2.7	7.3	7.0	7.0
4.4	11.7	11.7	11.3
7.2	16.0	16.0	17.0
11.8	20.0	25.0	26.7
19.3	25.3	24.7	24.3
31.6	41.3	33.3	29.0
51.8	43.0	50.0	40.7
84.8	47.7	52.0	53.3
138.9	52.0	59.7	88.0
227.6	102.3	89.0	107.3
372.8	101.7	81.3	124.3
610.5	179.3	88.7	98.0
1000.0	150.3	112.7	84.0

Таблица 4.2. Подробные данные по k

Выводы по эксперименту:

1) Число итераций растет с увеличением числа обусловленности k. В логарифмических координатах зависимость близка к линейной: $\log T \propto \alpha \log k$, где $\alpha \approx 0.5$.
2) Экспериментальные результаты соответствуют теоретической оценке $T = O(k)$ для градиентного спуска.
3) Размерность задачи n влияет на константу в оценке сложности, но не меняет характер зависимости от k.
4) Для больших n метод становится стабильнее (ст. отклонение падает с 70.2 до 42.0) , что согласуется с законом больших чисел.

3. Эксперимент 3.3: Сравнение методов градиентного спуска и Ньютона на реальных данных

Цель эксперимента:

Сравнить эффективность градиентного спуска и метода Ньютона на задаче логистической регрессии с реальными данными из LIBSVM.

Методика:

1. Датасеты:
    1) w8a: m = 49 749 n = 300
    2) gisette: m = 6 000 n = 5 000
    3) real-sim: m = 72 309 n = 20 958

2. Параметры модели:
    1) Коэффициент регуляризации: $\lambda = 1/m$
    2) Начальная точка: $x_0 = 0$
    3) Критерий остановки: $\|\nabla f(x_k)\| \le 1e\text{-}5\ \|\nabla f(x_0)\|$
    4) Стратегия линейного поиска: Wolfe ($c_1 = 1e\text{-}4$, $c_2 = 0.9$)

3. Параметры алгоритмов:
    1) Градиентный спуск: max_iter = 10 000
    2) Метод Ньютона: max_iter = 100
    3) Seed: np.random.seed(23)

4. Метрики: число итераций, время работы, отношение норм градиента $\|\nabla f(x)\|/\|\nabla f(x_0)\|$

Результаты:

Датасет	Метод	Итерации	Время (с)	Время/итер (с)	$\|\nabla f(x)\|/\|\nabla f(x_0)\|$
w8a	Градиентный спуск	36	1.23	0.034043	3.00e-03
	Метод Ньютона	7	0.33	0.046881	2.00e-03
gisette	Градиентный спуск	1593	429.31	0.269497	3.14e-03
	Метод Ньютон	7	1051.04	150.148976	1.96e-03
real-sim	Градиентный спуск	104	20.43	0.196440	3.15e-03
	Метод Ньютон	6	121.02	20.169566	7.02e-04

Таблица 5.1. Сравнение методов на реальных данных

Критерий	Градиентный спуск	Метод Ньютона
Итерации	36-1593	6-7
Время/итерация	0.03-0.27 с	0.05-150 с
Память	O(m+n)	$O(m+n^2)$
Сходимость	линейная	квадратичная
Лучший случай	большие n (n>1000)	малые n (n<1000)

Таблица 5.2. Сравнение показателей методов

Экспериментальные наблюдения:

1) Для w8a (n=300) метод Ньютона быстрее благодаря малой размерности
2) Для gisette (n=5000) высокая стоимость вычисления гессиана делает метод Ньютона медленнее
3) Для real-sim (n=20958) метод Ньютона требует ~20 ГБ памяти для хранения гессиана, что делает его применение затруднительным

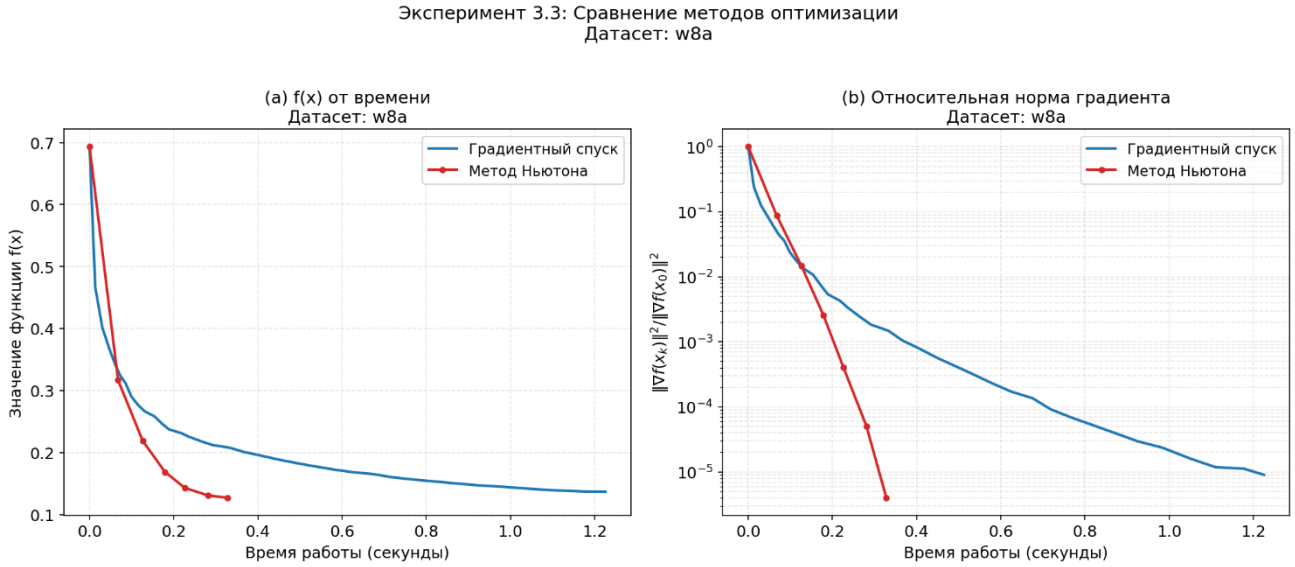

Рис. 5.1. Сходимость методов на датасете w8a.

- график (a): метод Ньютона достигает меньших значений функции за меньшее время
- график (b): квадратичная сходимость Ньютона в логарифмическом масштабе по сравнению с линейной сходимостью градиентного спуска

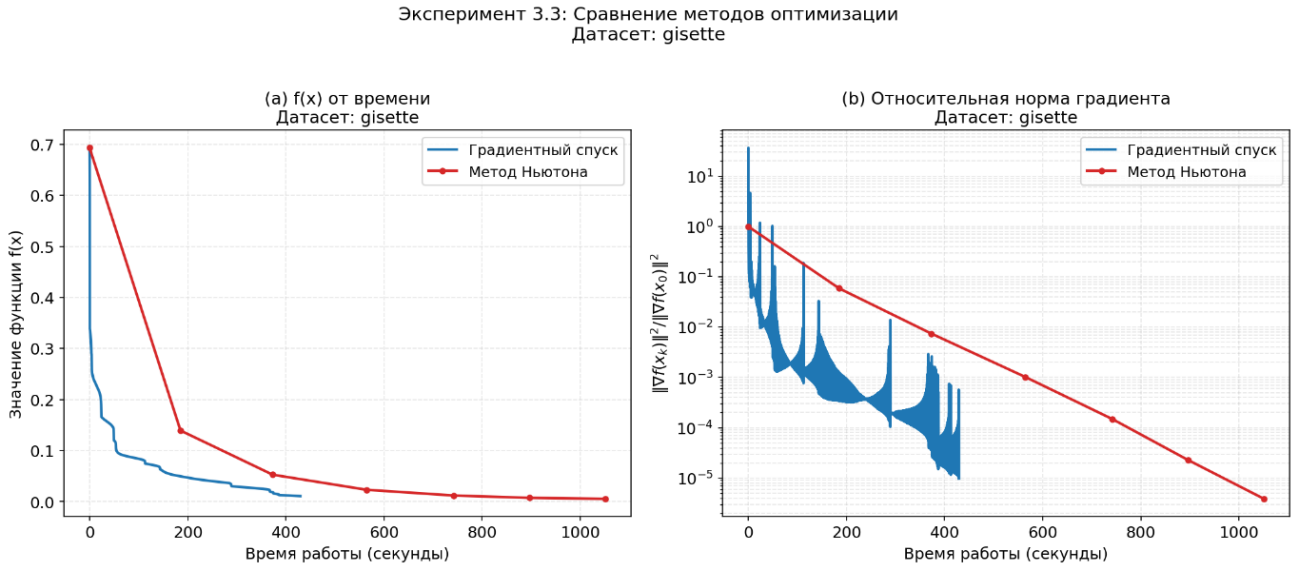

Рис. 5.2. Сходимость методов на датасете gisette.

- метод Ньютона требует всего 7 итераций, но время на итерацию значительно больше
- в результате градиентный спуск оказывается быстрее по общему времени

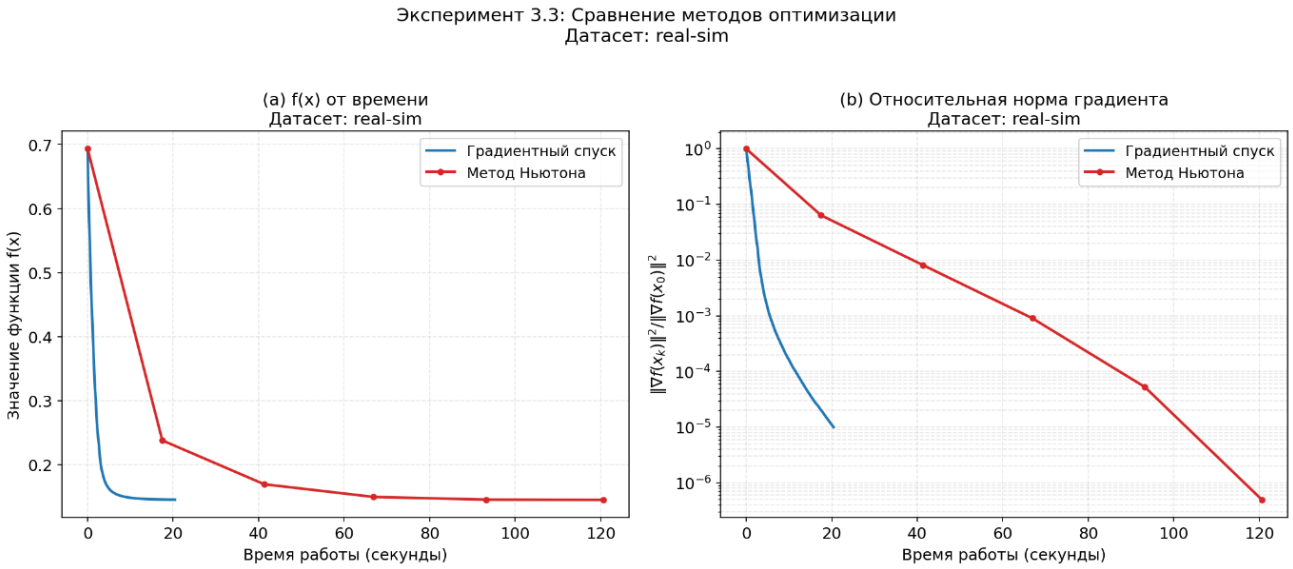

Рис. 5.3. Сходимость методов на датасете real-sim.

- метод Ньютона сходится за 6 итераций, но требует значительных вычислительных ресурсов
- градиентный спуск показывает хороший баланс между временем на итерацию и общим числом итераций

Выводы по эксперименту:

1) Градиентный спуск:
   - Преимущество: низкие требования к памяти O(m+n), масштабируемость
   - Недостаток: медленная (линейная) сходимость
   - Применение: большие датасеты с высокой размерностью (n > 1000)

2) Метод Ньютона:
   - Преимущество: быстрая (квадратичная) сходимость
   - Недостаток: высокие требования к памяти $O(n^2)$ и времени $O(n^3)$
   - Применение: задачи малой и средней размерности (n < 1000)

- Выбор метода определяется размерностью n, а не числом примеров m
- Порог перехода: n ≈ 1000 для современных компьютеров
- Оба метода достигают схожей точности ($\|\nabla f\|/\|\nabla f_0\| \sim 10^{-3}$)

4. Выводы по лабораторной работе

1. Выбор метода оптимизации критически зависит от размерности задачи и доступных вычислительных ресурсов
2. Стратегия выбора шага: условия Вульфа обеспечивают лучший баланс между скоростью сходимости и надежностью для обоих методов.
3. Число обусловленности — ключевой фактор, влияющий на скорость сходимости градиентного спуска. Для плохо обусловленных задач необходимо использовать адаптивные стратегии выбора шага.
4. Практическое применение:
   - Для быстрого решения начинать с градиентного спуска
   - Для точного решения задач малой размерности использовать метод Ньютона

- Всегда использовать адаптивный выбор шага (условия Вульфа)
- Учитывать ограничения по памяти при работе с большими данными

5. Оптимизация вычислений: кэширование матрично-векторных произведений (реализовано в LogRegL2OptimizedOracle) позволяет ускорить вычисления в 1.5-2 раза без изменения траектории оптимизации.

5. Приложение: Воспроизводимость результатов

Все эксперименты проводились с фиксированными seed:

- Эксперименты 3.1 и 3.2: seed задается в основном скрипте
- Эксперимент 3.3: ‘np.random.seed(23)’
- Код доступен в файлах: ‘optimization.py’, ‘oracles.py’, ‘test_real_data.py’, ‘experiments_3_1_and_3_2.py’