# Практическое задание 1 по курсу "Методы оптимизации в машинном обучении". Методы безусловной оптимизации.

Выполнила студентка группы 191,
Косовская Арина

# Содержание

# 1 Введение

Данное задание посвящено задаче гладкой безусловной оптимизации: $\min\limits_{x \in \mathbb{R}^n} f(x)$. Для ее решения мною были реализованы следующие методы:

- метод градиентного спуска;
- процедура линейного поиска;
- метод Ньютона;
- подсчет разностных производных.

В качестве критерия остановки используется $||\nabla f(x_k)||_2^2 \leq \varepsilon ||\nabla f(x_0)||_2^2$, где $\varepsilon \in (0; 1)$ — заданная относительная точность. В нашей реализации для выбора длины шага по умолчанию используется метод Вульфа с константами $c_1 = 1e-4$ и $c_2 = 0.9$.

Также были выведены формулы градиента и гессиана функции логистической регрессии, реализован ее оракул. Наконец, были проведены соответствующие эксперименты. В них в качестве $f(x)$ мы рассматривали квадратичную функцию $f(x) = \frac{1}{2} x^T A x - b^T x$ и двухклассовую логистическую регрессию $f(x) = \frac{1}{m} \sum\limits_{i=1}^{m} \ln(1 + \exp(-b_i < a_i, x >)) + \frac{\lambda}{2} ||x||_2^2$, $b(a) = \text{sign } (< a, x >)$, $\lambda$ — параметр регуляризации.

# 2 Вывод формул в матрично-векторной форме

## 2.1 Формула логистической регрессии

Необходимо представить $f(x) = \frac{1}{m} \sum\limits_{i=1}^{m} \ln(1 + \exp(-b_i < a_i, x >)) + \frac{\lambda}{2} ||x||_2^2$ в матрично-векторной форме.

Пусть матрица $A \in \mathbb{R}^{m \times n}$ — это матрица, в которой по строкам записаны векторы $a_i$, вектор $B \in \mathbb{R}^m$ содержит $b_i$, $i \in \{1, ..., m\}$. Обозначим за $1_{m \times 1}$ вектор-столбец из 1 размера $m$. Тогда:

$$f(x) = \frac{1}{m} \sum\limits_{i=1}^{m} \ln(1+\exp(-b_i < a_i, x >)) + \frac{\lambda}{2} ||x||_2^2 = \frac{1_{m \times 1}}{2} \ln(1_m + \exp(-B \bigodot Ax)) + \frac{\lambda}{2} ||x||_2^2$$

## 2.2 Формула градиента логистической регрессии

$$df = \frac{1}{m}\sum_{i=1}^{m} d\left(\ln\left(1+\exp\left(-b_i < a_i, x >\right)\right) + \frac{\lambda}{2} d(\langle x, x\rangle\right) =$$

$$= \frac{1}{m}\sum_{i=1}^{m} d\left(\ln\left(1+\exp\left(-b_i < a_i, x >\right)\right)\right) + \frac{\lambda}{2} < x, x >^0 < x, dx >= =$$

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{d\left(1+\exp\left(-b_i < a_i x >\right)\right)}{1+\exp\left(-b_i < a_i, x >\right)} + \lambda < x, dx >=$$

$$= \frac{1}{m}\sum_{i=1}^{m} -\frac{d\left(b_i < a_i, x >\right)\exp\left(-b_i < a_i, x >\right)}{1+\exp\left(-b_i < a_i, x >\right)} + \lambda < x, dx >=$$

$$= \frac{1}{m}\sum_{i=1}^{m} -\frac{b_i < a_i, dx > \exp\left(-b_i < a_i, x >\right)}{1+\exp\left(-b_i < a_i, x >\right)} + \lambda < x, dx >$$

Заметим, что $\dfrac{\exp(-b_i < a_i, x >)}{1+\exp(-b_i < a_i, x >)} = \dfrac{1}{\exp(b_i < a_i, x >)\cdot(1+\exp(-b_i < a_i, x >))} =$

$$= \frac{1}{1+\exp(b_i < a_i, x >)}$$

Тогда $df = \dfrac{1}{m}\sum_{i=1}^{m} -\dfrac{b_i < a_i, dx >}{1+\exp\left(b_i < a_i, x >\right)} = \dfrac{1}{m}\sum_{i=1}^{m} -\dfrac{b_i < a_i, dx >}{1+\exp\left(b_i < a_i, x >\right)} +$

$$+\lambda < x, dx > = \left\langle \frac{1}{m}\sum_{i=1}^{m} -\frac{b_i a_i}{1+\exp\left(b_i \langle a_i, x\rangle\right)} + \lambda x, dx \right\rangle$$

$$\nabla f = \frac{1}{m}\sum_{i=1}^{m} \frac{-b_i a_i}{1+\exp\left(b_i \langle a_i, x\rangle\right)} + \lambda x$$

Представим в матричном виде.

Введем $\text{sigmoid}(a) = \dfrac{1}{1+\exp(-a)}$, тогда $\dfrac{1}{1+\exp\left(b_i \langle a_i, x\rangle\right)} = \text{sigmoid}(-b_i \langle a_i, x\rangle) =$

$= \text{sigmoid}(-b_i a_i^T x) = \text{sigmoid}(\text{-}B \bigodot Ax)$.

Тогда $\boxed{\nabla f = \frac{1}{m} A^T(-B \bigodot \text{sigmoid}\ (-B \bigodot Ax)) + \lambda x.}$

## 2.3 Формула гессиана логистической регрессии

$$d^2 f = \left\langle \frac{1}{m}\sum_{i=1}^{m} d\left(\frac{-b_i a_i}{1+\exp\left(b_i < a_i, x >\right)} + \lambda x\right), dx_1 \right\rangle =$$

$$= -\frac{1}{m}\sum_{i=1}^{m} d\left(\frac{b_i a_i}{1+\exp\left(b_i < a_i, x >\right)} + \lambda x\right)^\top dx_1 =$$

$$= -\frac{1}{m}\sum_{i=1}^{m} d\left(\frac{b_i a_i^\top}{1+\exp\left(b_i < a_i, x >\right.} + \lambda x^\top\right) dx_1 =$$

$$= \frac{1}{m}\sum_{i=1}^{m} \frac{b_i a_i^\top dx_1 b_i a_i^\top \exp\left(b_i < a_i, x >\right) dx_2}{\left(1+\exp\left(b_i < a_{i,} x >\right)\right)^2} + \lambda \left(dx_2\right)^\top dx_1 =$$

$$= \frac{1}{m} < \sum_{i=1}^{m} \frac{a_i a_i^\top b_i^2 \exp\left(b_i \left\langle a_i, x\right\rangle\right) dx_1}{\left(1+\exp\left(b_i < a_i, x >\right)\right)^2}, dx_2 \Bigg\rangle + \left\langle \lambda dx_1, dx_2\right\rangle =$$

$$= \left\langle \left(\sum_{i=1}^{m} \frac{b_i^2 \exp\left(b_i \left\langle a_i, x\right\rangle\right) a_i a_i^\top}{m\left(1+\exp\left(b_i \left\langle a_i, x\right\rangle\right)^2\right.} + \lambda I_n\right) dx_1, dx_2 \right\rangle$$

$$\nabla^2 f(x) = \sum_{i=1}^{m} \frac{b_i^2 \exp\left(b_i < a_i, x >\right) a_i a_i^T}{m\left(1+\exp\left(b_i < a_{i,} x >\right)\right)^2} + \ \lambda I_n$$

$$\nabla^2 f(x) = \frac{1}{m}\sum_{i=1}^{m} b_i^2 \cdot a_i a_i^\top \exp\left(b_i \left\langle a_i, x\right\rangle\right) \frac{1}{1+\exp\left(b_i \left\langle a_i, x\right\rangle\right))^2} + \lambda I_n$$

$$\nabla^2 f(x) = \frac{1}{m}\sum_{i=1}^{m} \left(b_i^2 a_i a_i^\top \left(1 - \text{sigma}\left(-b_i < a_i, x >\right)\right) \text{sigma}\left(-b_i < a_i, x >\right)\right) + \lambda I_n$$

$$\boxed{\nabla^2 f(x) = \frac{1}{m} A^\top \operatorname{diag}[B \circ B \circ (1 - \text{sigma}(-B \circ Ax)) \circ \text{sigma}(-B \circ Ax)] A + \lambda I_n}$$

# 3 Эксперимент 1. Траектория градиентного спуска на квадратичной функции.

В данном эксперименте необходимо проанализировать траекторию градиентного спуска для нескольких квадратичных функций в зависимости от числа обусловленности функции, выбора начальной точки и стратегии выбора шага. Для этого были сгенерированы три квадратичные двумерные функции, на которых работа метода отличается:

1) $f_1(x) = \frac{1}{2} < \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x, x >.$

2) $f_2(x) = \frac{1}{2} < \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix} x, x > .$

3) $f_3(x) = \frac{1}{2} < \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix} x, x > .$

Заметим, что у данных функций числа обусловленности равны 1, 10 и 33.97 соответственно. Для всех функций точка минимума — $(0, 0)$. Мы будем пользоваться уже готовой реализацией оракула квадратичной функции из модуля oracles.

## 3.1 Зависимость траектории от числа обусловленности функции

В качестве точки старта градиентного спуска выберем $(-4; 1)$ (на выбор точки никак не влияли свойства сгенерированных функций). Для каждой функции $f_1, f_2, f_3$ сделаем запуск градиентного спуска, в качестве стратегии выбора шага будем использовать метод Вульфа.

Из графиков 3.1 видно, что чем больше число обусловленности, тем более вытянутые линии уровня. В случаях, когда линии уровня вытянутые, градиентный спуск работает хуже, траектория становится "зигзагообразной и методу требуется большее количество итераций до сходимости.

**Вывод:** с ростом числа обусловленности траектория становится более сложной, зигзагообразной, градиентному спуску требуется бОльшее число итераций до сходимости.

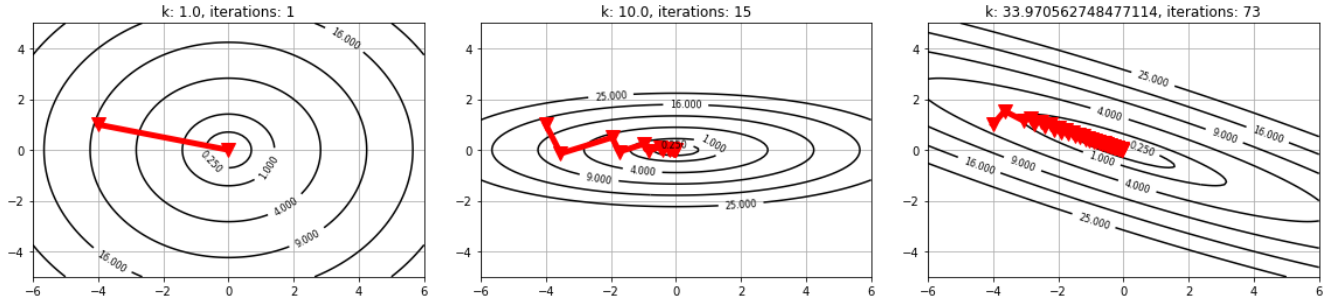



Рис. 3.1: Зависимость траектории от числа обусловленности.
Точка старта: (-4; 1)

## 3.2 Зависимость траектории от начальной точки

Исследуем, зависит ли траектория метода от выбора начальной точки. Для этого в качестве начальных точек возьмем (-4; 1), (4, 4) и (1, 3) и запустим из них градиентный спуск с параметрами по умолчанию.

Траектории для случая, когда в качестве точки старта была выбрана точка (-4; 1), изображена на рисунке 3.1.

Траектории для случая, когда в качестве точки старта была выбраны точки (4, 4) и (1, 3), изображены на рисунке 3.2.

Как мы видим, в случае, когда число обусловленности равно 1, выбор точки не влияет на траекторию, сходимость происходит за 1 шаг.

Однако в случаях, когда число обусловленности больше 1, траектория градиентного спуска зависит от выбора начальной точки. Так как если линии уровня сильно вытянуты в каком-то направлении (а при больших числах обусловленности это и происходит), то вдоль направления функция будет меняться медленнее, чем по нормали к нему. [1] Тогда если мы выберем начальную точку, которая находится в вытянутой части (как это произошло в случае с 3.1), то мы будем двигаться не в ту сторону, направление антиградиента будет все время меняться, и траектория будет зигзагообразной. В таком случае метод будет сходиться медленно. Если выбрать точку, где линии уровня менее вытянуты, то сходимость будет значительно быстрее, что мы и наблюдаем на 3.2.

**Вывод:** в случае, если линии уровня вытянутые (это происходит при достаточно большом числе обусловленности), начальная точка влияет на траекторию. Иначе явной зависимости не наблюдается.

[1] Для осознания этого я воспользовалась материалами из этого документа: http://www.apmath.spbu.ru/ru/staff/grigorieva/mfk.pdf

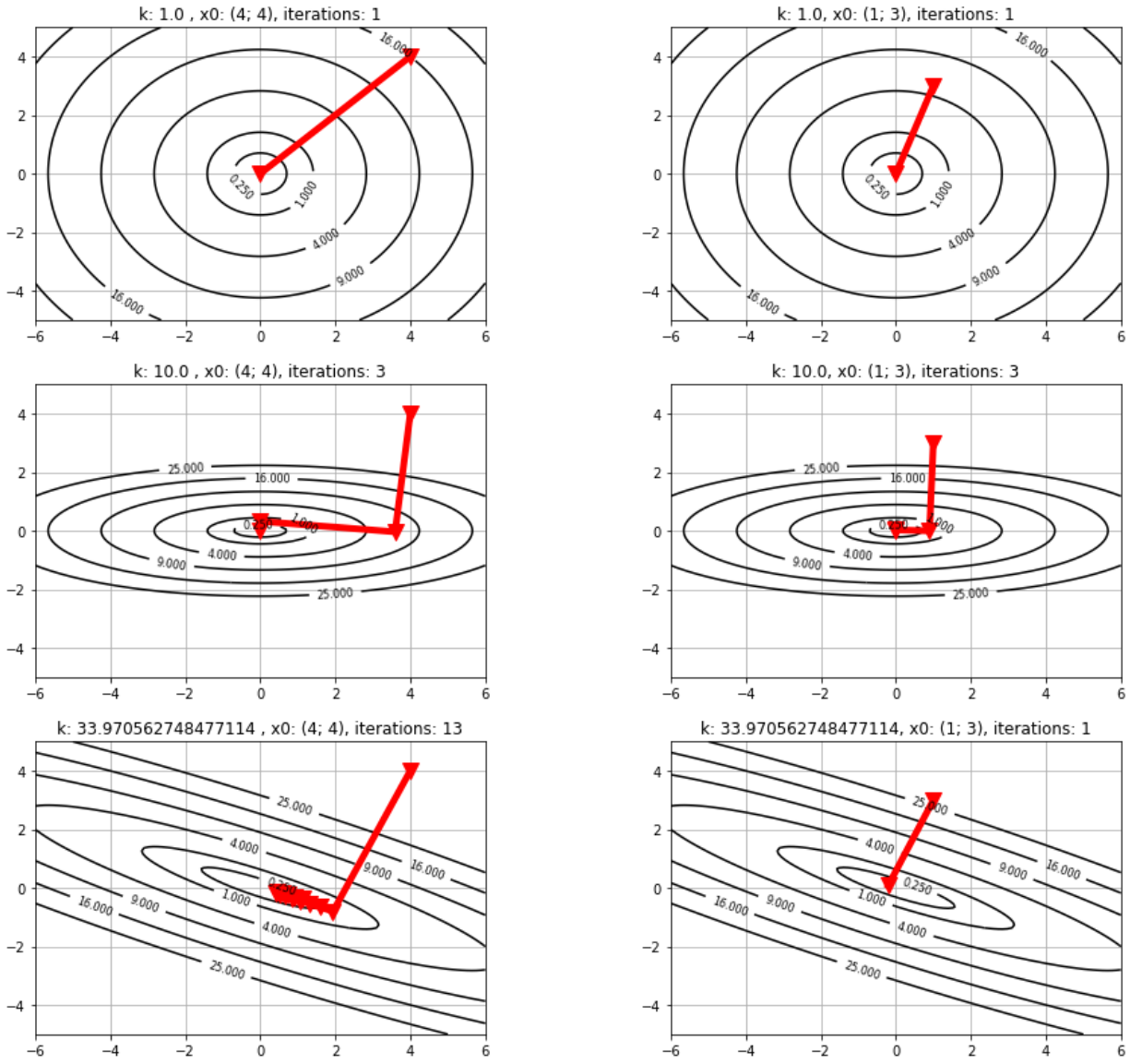



Рис. 3.2: Зависимость траектории от начальной точки.

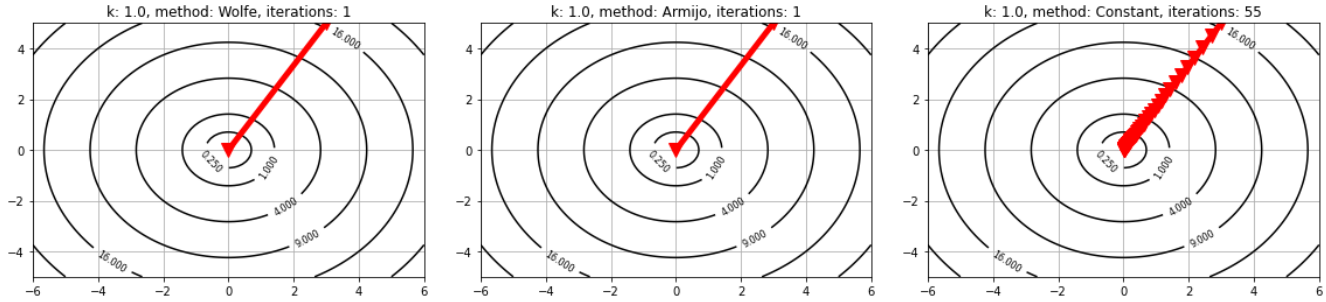
## 3.3 Зависимость траектории от выбора стратегии шага

Наконец, исследуем, зависит ли траектория метода от выбора стратегии шага. В качестве точки старта выберем (3; 5) (опять же, на выбор точки никак не влияли свойства сгенерированных функций).

Сделаем запуски градиентного спуска для всех трех функции, используя для выбора шага условия Вульфа, Армихо или константный метод. В случае, когда используется константный метод, $c = 0.1$ (иначе метод не сходится). При использовании других стратегий все параметры выбираются по умолчанию.

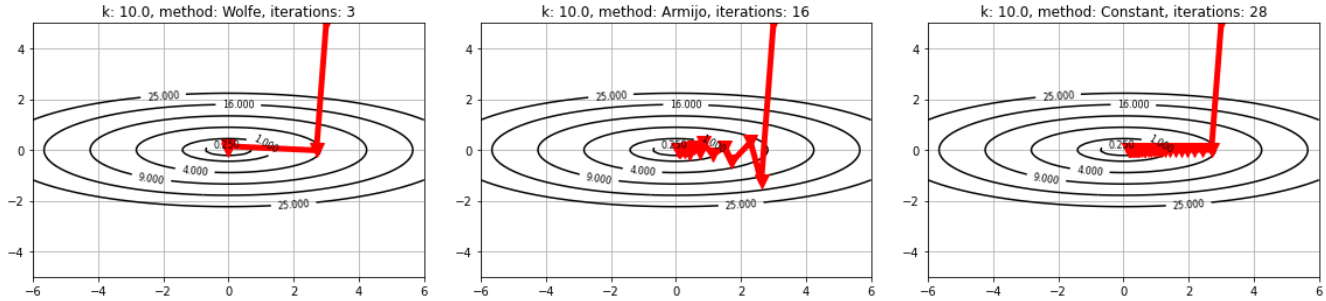
Получившиеся траектории для функций с числом обусловленности 1, 10 и 33.97 изображены на рисунках 3.3, 3.4 и 3.5 соответственно.

Для того, чтобы констатный метод сошелся, требуется подбирать параметр $c$. Даже при подборе параметра констактному методу все равно потребовалось




(a) Wolfe (b) Armijo (c) Constant

Рис. 3.3: Траектории метода для числа обусловленности 1



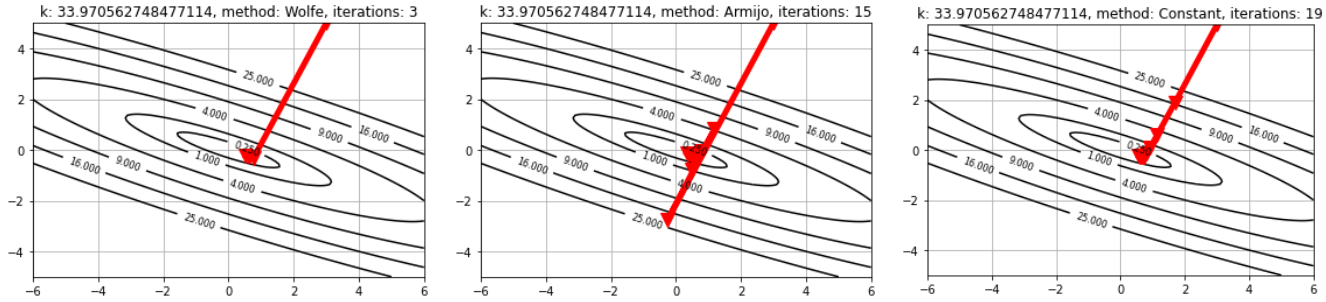

(a) Wolfe (b) Armijo (c) Constant

Рис. 3.4: Траектории метода для числа обусловленности 10

большее число итераций, чем условиям Армихо и Вульфа. Меньше всего требуется итераций в случае, когда мы используем условию Вульфа. При использовании условия Армихо траектория немного зигзагообразная.

**Вывод:** в случае, если мы используем константный метод, необходимо аккуратно подбирать $c$, иначе метод не сойдется. Быстрее всего градиентный спуск сходится, если использовать условие Вульфа.

**Вывод из первого эксперимента:** если число обусловленности небольшое, то градиентный спуск будет быстро сходиться вне зависимости от выбора начальной точки. Если же число обусловленности большое, то скорость сходимости будет зависеть от выбора начальной точки, вытянуты ли в ней линии уровня. Быстрее всего градиентный спуск сходится при использовании условия Армихо, медленнее всего — при константном методе.




(a) Wolfe (b) Armijo (c) Constant

Рис. 3.5: Траектории метода для числа обусловленности 33.97

# 4 Эксперимент 2. Зависимость числа итераций градиентного спуска от числа обусловленности и размерности пространства.

В данном эксперименте мы исследуем зависимость числа итераций, необходимых для сходимости градиентного спуска, от числа обусловленности $k \geq 1$ оптимизируемой функции и размерности пространства $n$ оптимизируемой переменной. В качестве оптимизируемой функции мы рассматриваем квадратичную задачу размера $n$ и числа обусловленности $k$.

Случайным образом мы будем генерировать квадратичную задачу размера $n$ и числа обусловленности $k$. Для этого сначала мы сгенерируем разреженную диагональную матрицу $A \in S^n_{++}$, на диагонали которой будут стоять числа из промежутка от 1 до $k$ (причем среди данных чисел одно будет равняться 1, а другое — $k$, остальные значения выбираются случайно из равномерного распределения). Элементы вектора $b \in \mathbb{R}^n$ будут генерироваться случайно из нормального распределения $N(0, 5)$.

Далее мы будем запускать на сгенерированной квадратичной задаче градиентный спуск с фиксированной точностью (1e-5). В качестве $x_0$ будем брать массив из случайных чисел от 0 до 50. На каждом запуске будем замерять число итераций $T(n, k)$, необходимых для сходимости (то есть когда был выполнен критерий остановки). Остальные параметры будут выбраны по умолчанию.

Фиксируя значение $n$, будем случайно генерировать квадратичную функцию и перебирать $k$ (от 1 до 500 с шагом 10). Повторим данный эксперимент 4 раза. У нас получится семейство кривых зависимости $T(n, k)$ от $k$, построим график, нарисовав кривые одним цветом. Изменив значение $n$, повторим описанную процедуру: сгенерируем квадратичную функцию, переберем $k$ и повторим 4 раза.

Полученные кривые зависимости $T(n, k)$ от $k$ изобразим уже другим цветом, и так далее. Значения $n$ будем перебирать по логарифмической сетке от 10 до $10^6$ с шагом 10.

Проведем данный эксперимент.

На графике 4.1 изображен график зависимости $T(n, k)$ против $k$.

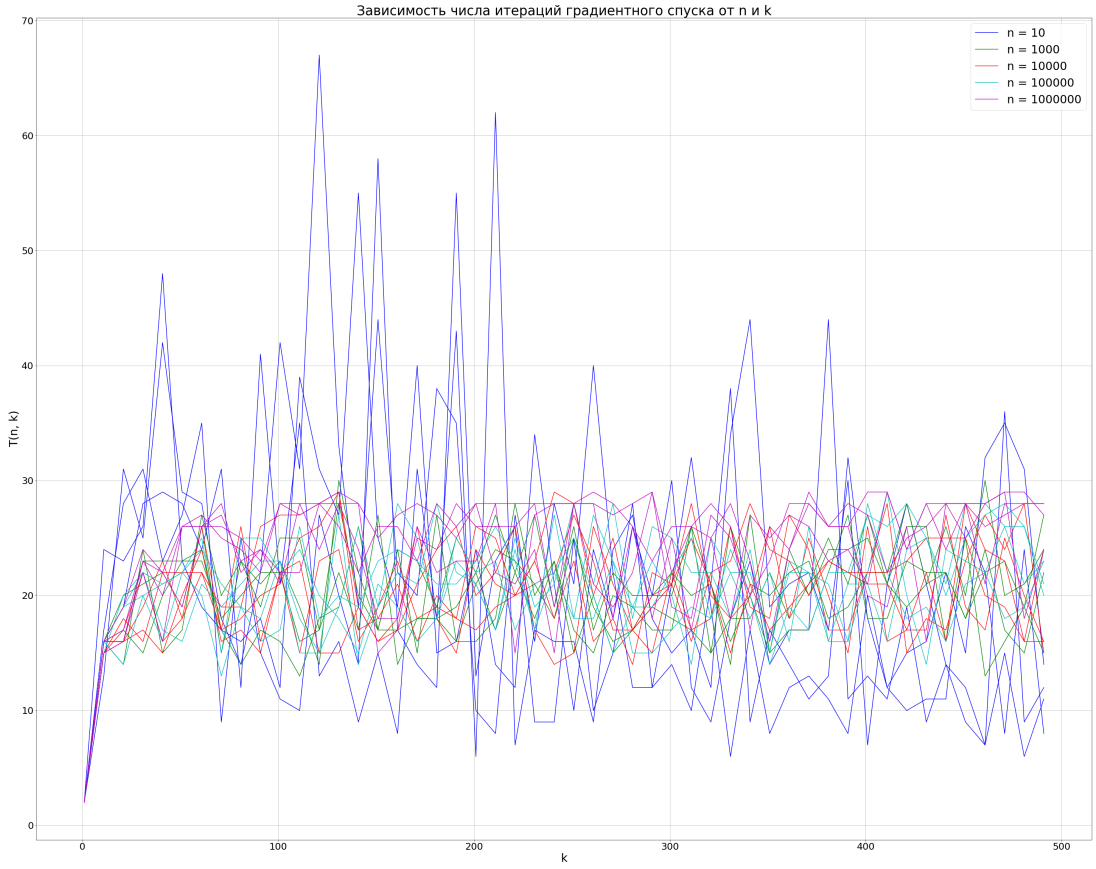



Рис. 4.1: Зависимость числа итераций градиентного спуска от числа обусловленности и размерности пространства

**Выводы:**

Заметим, что при увеличении числа обусловленности увеличивается и число итераций до сходимости, и эта зависимость близка к линейной.

При увеличении размерности пространства уменьшаются разброс и дисперсия числа итераций до сходимости.

Отметим, что получившиеся результаты согласовываются с тем, что мы получили в предыдущем эксперименте. Действительно, мы видели, что при увеличении числа обусловленности увеличивалось и число итераций до сходимости, но

сейчас получилось понять больше про характер зависимости.

# 5 Эксперимент 3. Какая точность оптимизации нужна в реальных задачах для метода Ньютона?

В данном эксперименте мы исследуем, как влияет точность оптимизации целевой функции на процент ошибок при классификации на тестовой выборке в случае задачи бинарной классификации и модели логистической регрессии с $l^2$-регуляризатором и использовании метода Ньютона.

Для этого мы выбрали данные с сайта LIBSVM: w8a и gisette. Коэффициент регуляризации $\lambda$ и начальную точку $x_0$ мы взяли стандартными: $\frac{1}{m}$ и 0 соответственно, где $m$ — количество наблюдений в данных. Параметр точности оптимизации $\varepsilon$ перебирается по массива [1, 1e-1, 1e-2, ..., 1e-8]. Остальные параметры выбираются по умолчанию.

Для каждого $\varepsilon$ мы запускаем метод Ньютона и сравниваем качество прогноза логистической регрессии $\hat{b}_{\text{test}} = \text{sign}(A_{\text{test}}\hat{x}) \in -1, 1)$ ($\hat{x}$ — найденная с помощью метода Ньютона точка оптиума) с истинным значением $b_{\text{test}}$. Для сравнения используется среднее число позиций, в которых векторы $\hat{b}_{\text{test}}$ и $b_{\text{test}}$ отличаются: $\frac{1}{m} \cdot \sum_{i=1}^{m} \text{Ind}[\hat{b}_{\text{test}} \neq b_{\text{test}}$, где $Ind(x)$ — индикаторная функция.

Рассмотрим полученный график 5.1 зависимости процента ошибок против точности оптимизации в логарифмической шкале:

Для анализа графика вспомним, что в качестве критерия остановки используется $||\nabla f(x_k)||_2^2 \leq \varepsilon ||\nabla f(x_0)||_2^2$. Тогда при $\varepsilon = 1$ метод будет возвращать $x_0 = 0$, так как критерий остановки будет выполнен сразу же, до начала оптимизации. Тогда мы будем ошибаться на всех метках, ведь $\hat{b}_{\text{text}}$ не может получать нулевое значение из его определения.

С уменьшением $\varepsilon$ до $10e-4$ начинает сильно убывать и процент ошибки (так как чем большую точность мы требуем, тем ближе найденная методом точка должна быть к истинной точке оптиума). При $10e-8 < \varepsilon < 10e-4$ процент ошибок перестает меняться значительно.

Таким образом, при маленькой точности оптимизации процент ошибок стремится к 1. С увеличением точности начинает падать процент ошибок. При очень большой точности (больше $10e-4$) процент ошибок перестает значительно меняться.

**Вывод:** при $\varepsilon = 1$ доля ошибок всегда будет равна 1, так как в качестве начальной точки мы выбрали ноль. При уменьшении $\varepsilon$ до $10e-4$ резко падает доля

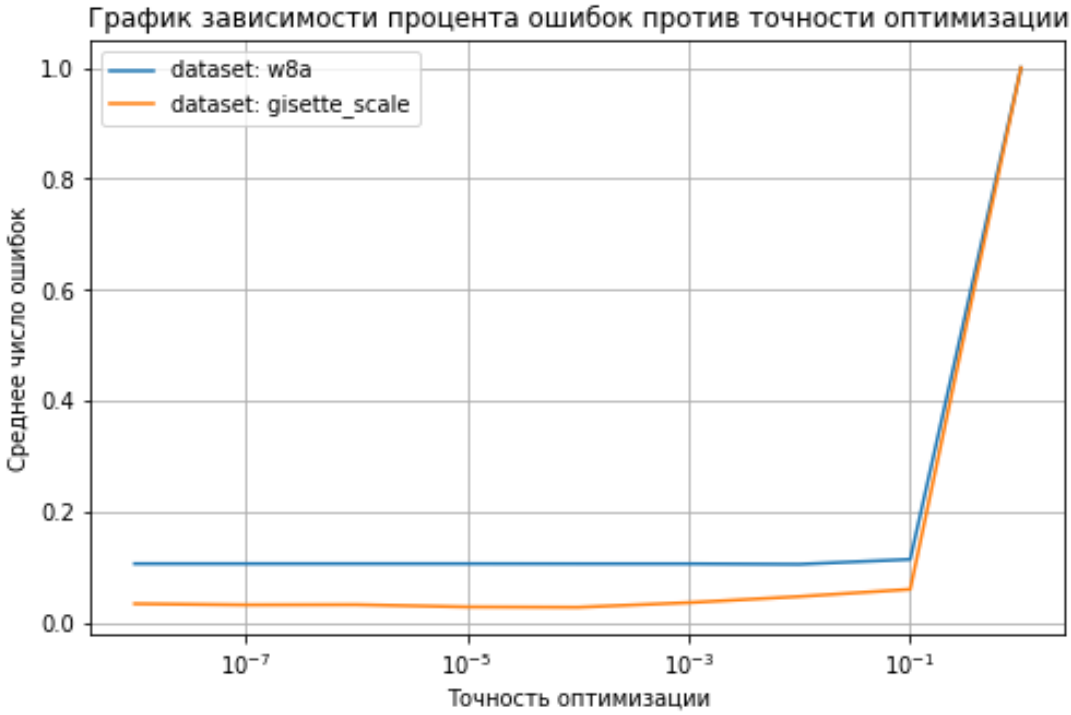



Рис. 5.1: Зависимость процента ошибок против точности оптимизации в логарифмической шкале

ошибок, и при дальнейшем уменьшении перестает меняться значительно. Таким образом, необязательно в качестве требуемой точности брать очень маленькие значения, чтобы получить высокое итоговое качество. Можно выбрать, например, $\varepsilon = 10e - 4$ (при меньших $\varepsilon$ процент ошибок будет убывать незначительно, при больших $\varepsilon$ процент ошибок начинает сильно расти).

# 6 Эксперимент 4. Сравнение методов на реальной задаче логистической регрессии

В данном эксперименте мы будем сравнивать метод Ньютона и градиентный спуск на задаче логистической регрессии. В качестве данных будут использоваться наборы данных с сайта LIBSVM: w8a, gisette, real-sim, news20.binary, rcv1.binary. В качестве коэффициента регуляризации $\lambda$ и начальной точки $x_0$ взяты $\frac{1}{m}$ и 0 соответственно. В качестве стратегии выбора шага были рассмотрены методы Вульфа, Армихо и константный метод. Остальные параметры выбраны по умолчанию. Для считывания данных я воспользовалась кодом, который прислал Владимир Романов в общий чат курса.

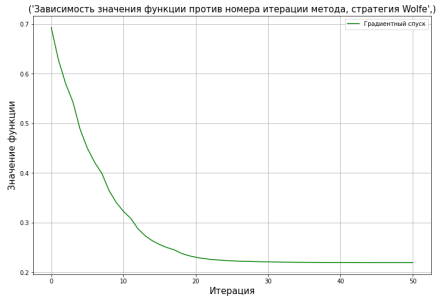
Рассмотрим размерности наборов, с которых нам предстоит работать:

Таблица 6.1: Размерности наборов данных

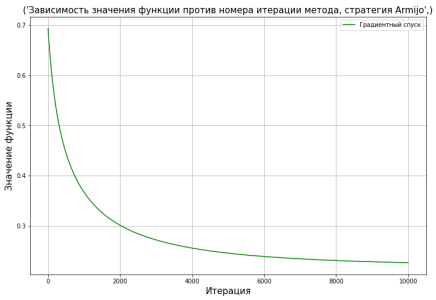
| Название набора | Количество объектов ($q$) | Количество признаков ($n$) |
|---|---|---|
| w8a | 49749 | 300 |
| gisette | 6000 | 5000 |
| real-sim | 72309 | 20958 |
| news20.binary | 19996 | 1355191 |
| rcv1_train.binary | 20242 | 47236 |

При запуске метода Ньютона при использовании наборов данных real-sim, news20.binary и rcv1_train.binary происходило переполнение памяти. Это объясняется тем, что на лекции обсуждалось, что метод Ньютона требует $O(n^2)$ памяти, так как требуется вычислять гессиан. Также одна итерация стоит $O(n^3)$, поэтому метод Ньютона не запускается и не эффективен по времени в случае большого количества признаков. Градиентному спуску же нужно $O(n)$ памяти, одна итерация стоит $O(n)$, поэтому для него подобных проблем не возникает.

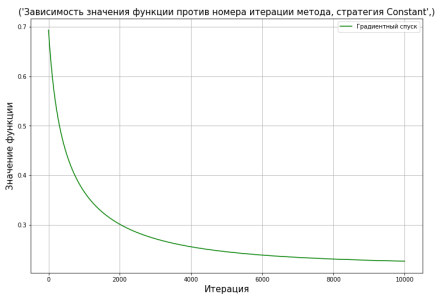
Для полноты все же рассмотрим, как зависят значения функции против номера итерации и реального времени работы, и как зависит относительный квадрат нормы градиента против реального времени работы градиентного спуска (рисунки 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8, 6.9)




(a) стратегия: Wolfe

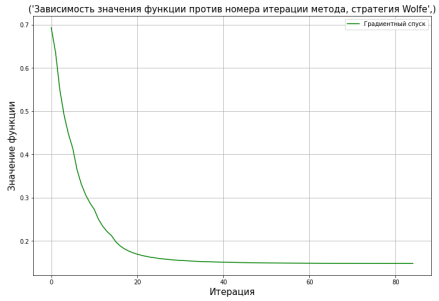



(b) стратегия: Armijo

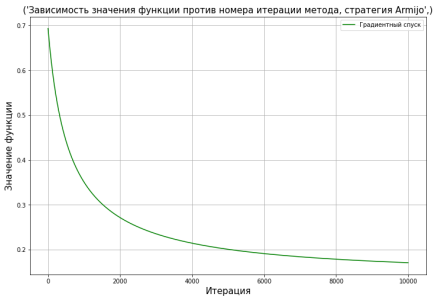



(c) стратегия: Constant

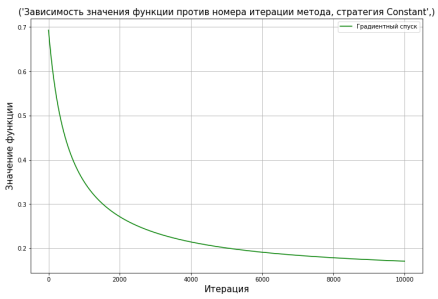
Рис. 6.1: Зависимость значения функции против номера итерации, данные: real-sim




(a) стратегия: Wolfe

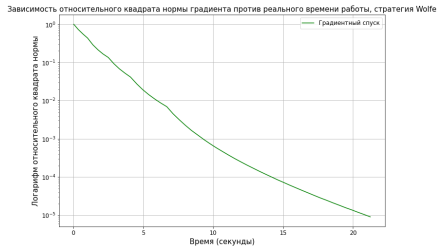



(b) стратегия: Armijo

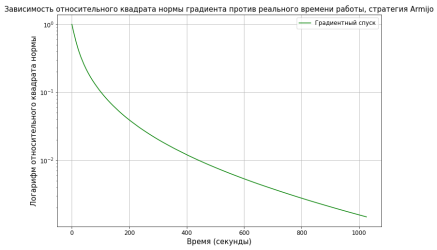



(c) стратегия: Constant

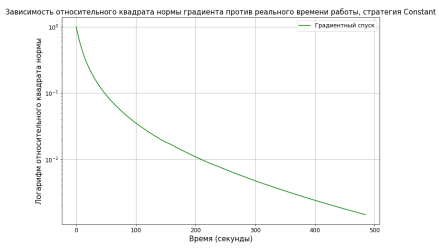
Рис. 6.2: Зависимость значения функции против реального времени работы, данные: real-sim




(a) стратегия: Wolfe

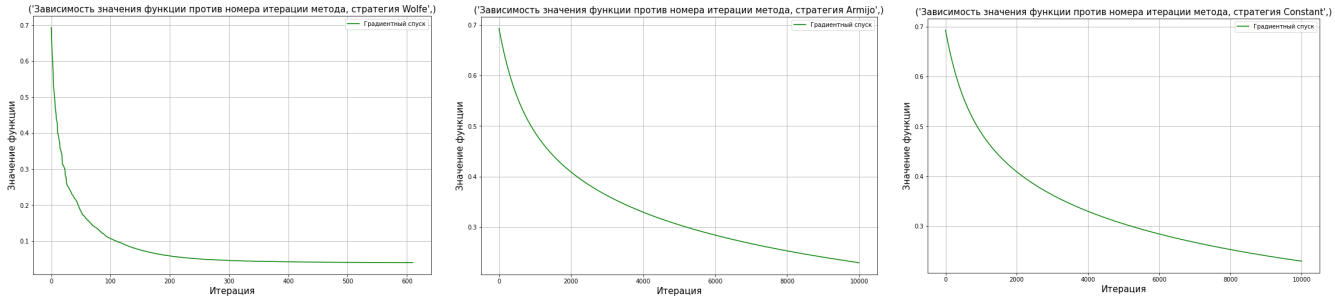



(b) стратегия: Armijo



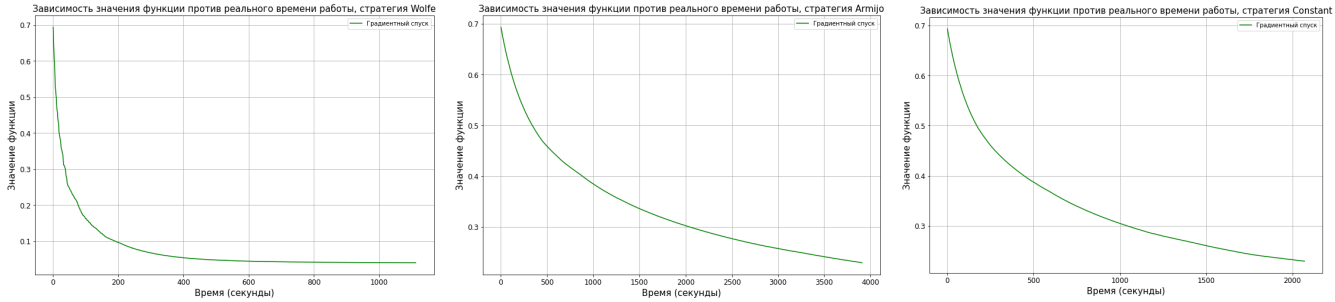

(c) стратегия: Constant

Рис. 6.3: Зависимость относительного квадрата нормы градиента против реального времени работы, лог. шкала, данные real-sim




(a) стратегия: Wolfe (b) стратегия: Armijo (c) стратегия: Constant

Рис. 6.4: Зависимость значения функции против номера итерации, данные: news20.binary



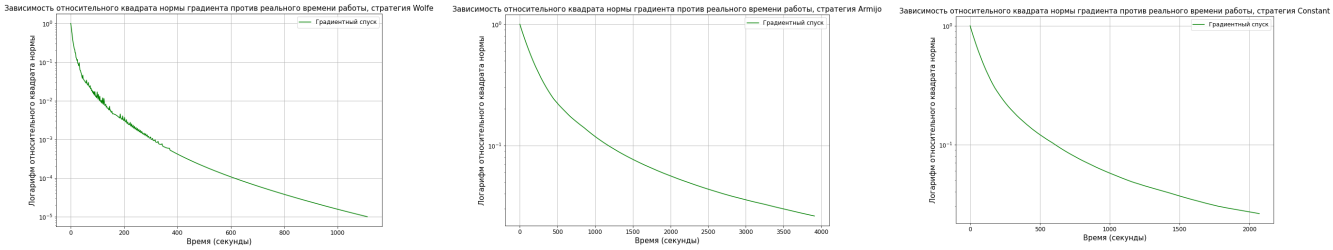

(a) стратегия: Wolfe (b) стратегия: Armijo (c) стратегия: Constant

Рис. 6.5: Зависимость значения функции против реального времени работы, данные: news20.binary

Видим, что градиентному спуску требуется большое количество итераций до сходимости. Несмотря на то, что каждая итерация занимает $O(n)$ времени, за счет числа итераций метод все равно работает долго. Тем не менее, градиентный спуск обеспечил сходимость на всех наборах данных.

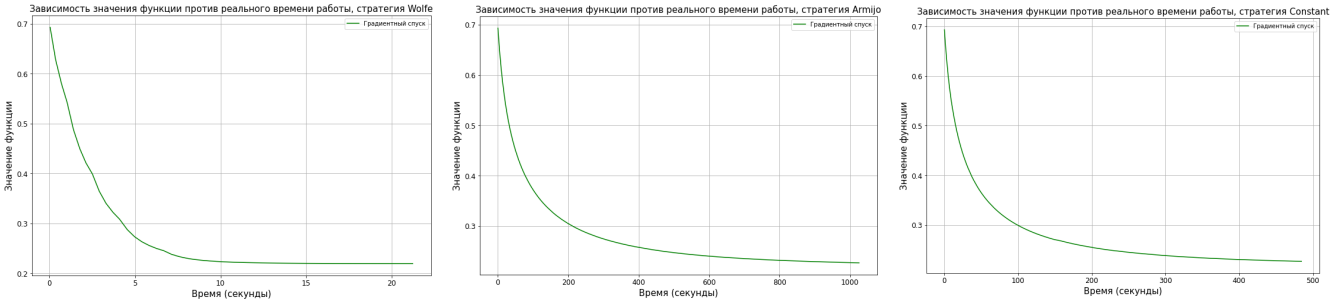
Перейдем к сравнению методов Ньютона и градиентного спуска на наборах данных w8a и gisette. Рассмотрим следующие графики:

- графики зависимости значения функции против номера итерации метода (6.10, 6.11);
- график зависимости значения функции против реального времени работы (6.12, 6.13);
- график зависимости относительного квадрата нормы градиента $||\nabla f(x_k)||_2^2/||\nabla f(x_0)||_2^2$ в логарифмической шкале против реального времени работы (6.14, 6.15).




(a) стратегия: Wolfe (b) стратегия: Armijo (c) стратегия: Constant

Рис. 6.6: Зависимость относительного квадрата нормы градиента против реального времени работы, лог. шкала, данные: news20.binary

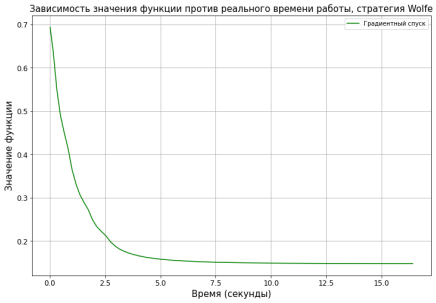



(a) стратегия: Wolfe (b) стратегия: Armijo (c) стратегия: Constant

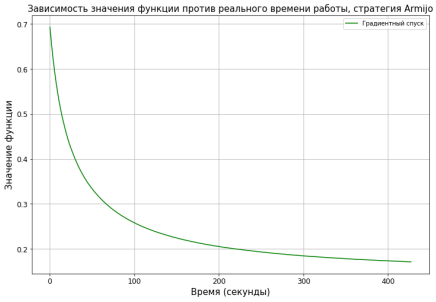
Рис. 6.7: Зависимость значения функции против номера итерации, данные: rcv1_train.binary

Как мы заметили в случае работы с наборами данных с большим количеством признаков, для сходимости градиентного спуска требуется много итераций (для любой стратегии выбора шага). Из-за этого, несмотря на скорость выполнения одной итерации, целиком метод работает долго. Относительный квадрат нормы градиента в логарифмической шкале уменьшается достаточно медленно в сравнении с методом Ньютона.

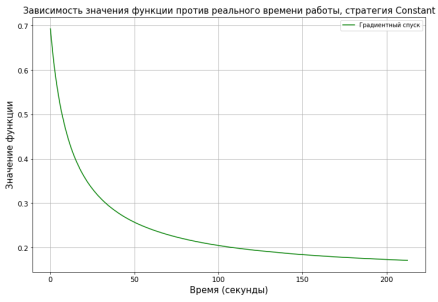
Для сходимости метода Ньютона требовалось значительно меньшее число итераций, чем для сходимости градиентного спуска. При этом одна итерация работает долго, она требует $O(n^3)$ времени (что противоположно градиентному спуску, где наоборот много быстрых итераций). То есть с увеличением $n$ каждая итерация будет выполняться дольше, но при этом количество итераций все еще будет меньше, чем у градиентного спуска. Поэтому метод Ньютона стоит использовать, если число признаков небольшое (для набора данных w8a метод Ньютона работает значительно быстрее градиентного спуска, а для gisette время работы приблизительно одинаковое). Относительный квадрат нормы градиента убывает значительно быстрее, чем в случае градиентного спуска.




(a) стратегия: Wolfe

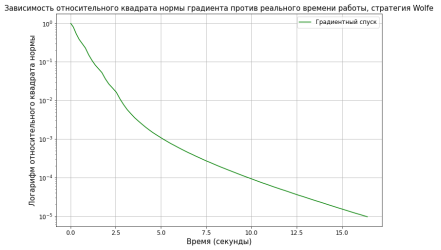



(b) стратегия: Armijo

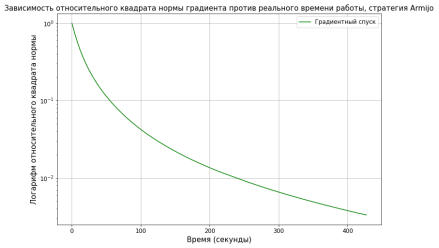



(c) стратегия: Constant

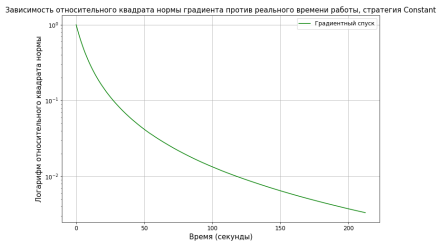
Рис. 6.8: Зависимость значения функции против реального времени работы, данные: rcv1_train.binary




(a) стратегия: Wolfe




(b) стратегия: Armijo




(c) стратегия: Constant

Рис. 6.9: Зависимость относительного квадрата нормы градиента против реального времени работы, лог. шкала, данные: rcv1_train.binary

**Вывод.**

Метод Ньютона запускается и эффективен по времени только для пространств малой размерности. В случае, если признаков много, стоит воспользоваться градиентным спуском (так как он значительно эффективнее по памяти).

В случае, когда признаков достаточно мало, вне зависимости от стратегии выбора шага метод Ньютона работает быстрее за счет малого количества итераций.

# 7 (Бонусная часть) Эксперимент: Оптимизация вычислений в градиентном спуске

В данном эксперименте требуется сравнить градиентный спуск на логистической регрессии для обычного и оптимизированного оракула. Оптимизированный оракул отличается от обычного тем, что он запоминает последние матрично-векторные произведения.

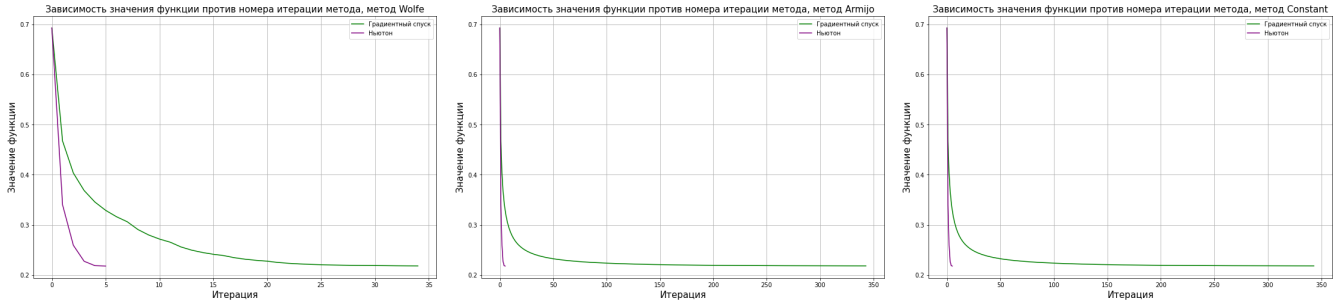



Рис. 6.10: Зависимость значения функции против номера итерации метода, набор данных: w8a

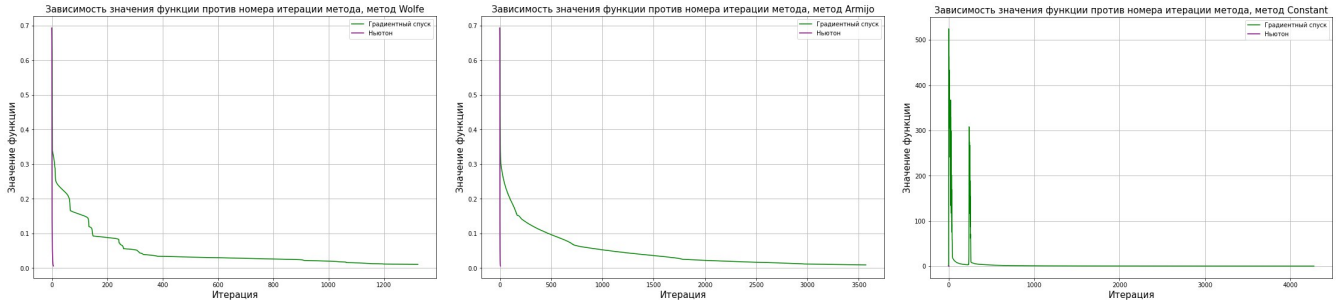



Рис. 6.11: Зависимость значения функции против номера итерации метода, набор данных: gisette

В качестве данных используется модельная выборка с размерами $(m,\ n)$, где $m = 10000,\ n = 8000$. $A \in \mathbb{R}^{m \times n}$ — выборка размера $(m, n)$ из стандартного нормального распределения. Для получения $b \in \mathbb{R}^m$ генерируется выборка из стандартного нормального распределения размера $m$, а затем от каждого элемента берется знак. Таким образом, $b_i \in \{1; -1\}\ \forall i \in 1...m$. Коэффициент регуляризации $\lambda = \frac{1}{m}$, начальная точка $x_0 = 0$. Остальные параметры берутся по умолчанию.

Как в предыдущем эксперименте, требуется построить:

∘ графики зависимости значения функции против номера итерации градиентного спуска (7.1a);

∘ графики зависимости значения функции против реального времени работы градиентного спуска (7.1b);

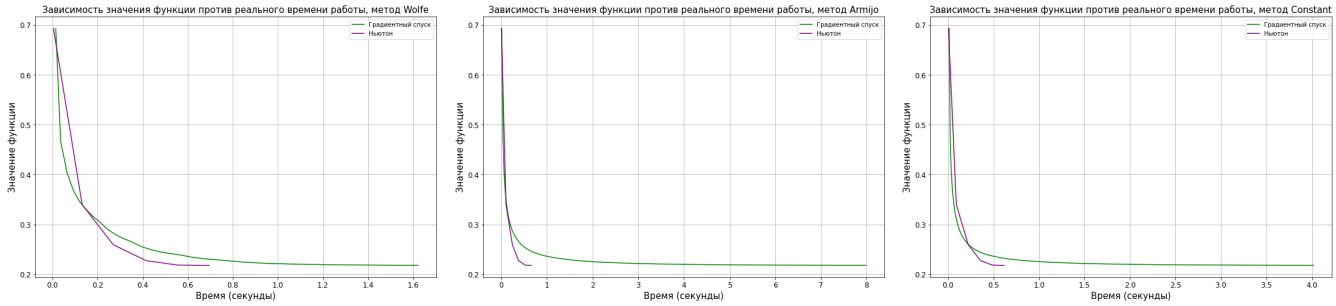



Рис. 6.12: Зависимость значения функции против реального времени работы, набор данных: w8a

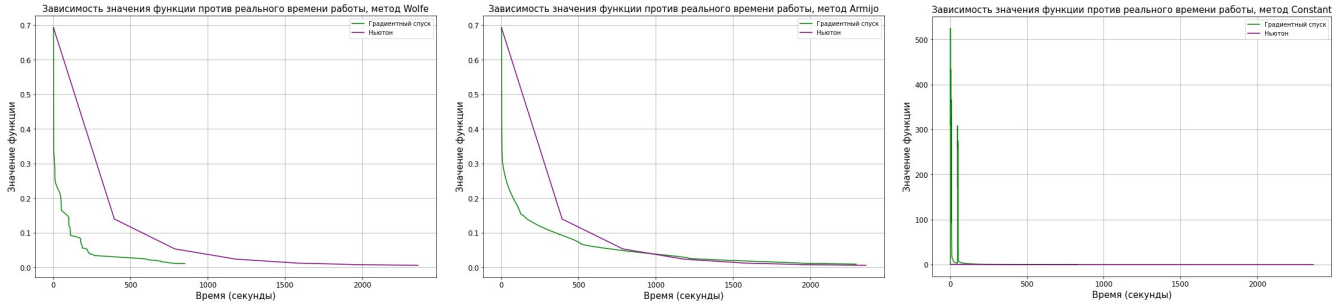



Рис. 6.13: Зависимость значения функции против реального времени работы, набор данных: gisette

∘ графики зависимости относительного квадрата нормы градиента $||\nabla f(x_k)||_2^2/||\nabla f(x_0)||_2^2$ в логарифмической шкале против реального времени работы (7.1c).

Заметим, что траектория метода при использовании оптимизированного оракула не будет отличаться от случая, когда мы используем обычный оракул (рис. 7.1a). В оптимизированном оракуле мы улучшили вычисления, запоминая значения матриц. Это позволяет не считать матрицы $Ax$ и $Ad$, которые нужны для вычисления градиента, гессиана, значений функций и производных по направлению, повторно. Но количество итераций градиентного спуска и их порядок не зависят от того, каким способом мы вычисляли градиент или производную по направлению, поэтому траектории на графике 7.1a совпадают.

По графику 7.1b видим, что при использовании оптимизированного оракула скорость сходимости сильно улучшилась. Этого следовало ожидать, ведь мы перестали делать лишние вычисления, которые занимали время.

**Вывод:** использование оптимизированного оракула не влияет на траекторию градиентного спуска, но сильно улучшает время работы метода.

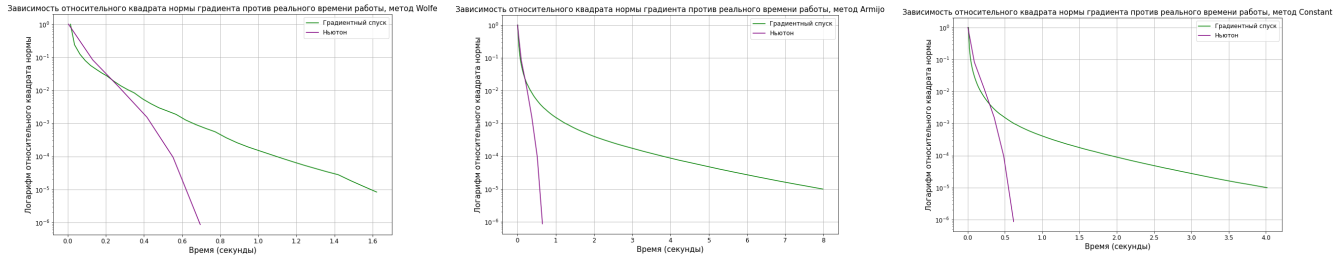



Рис. 6.14: Зависимость относительного квадрата нормы градиента в логарифмической шкале против реального времени работы, набор данных: w8a

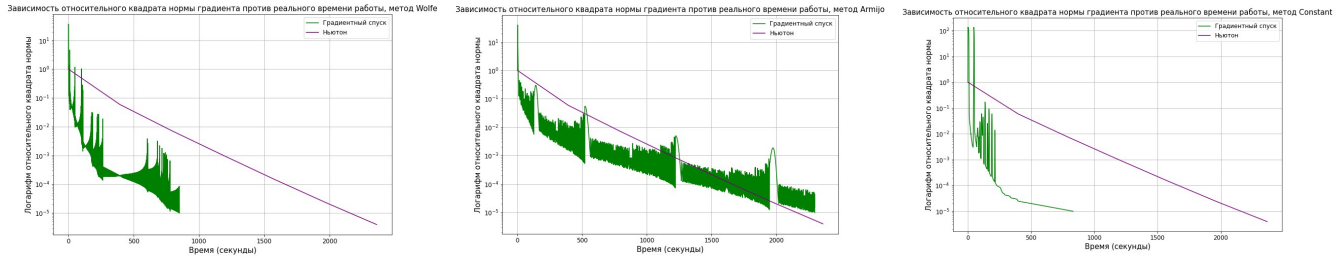
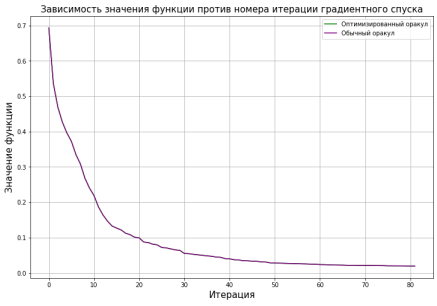



Рис. 6.15: Зависимость относительного квадрата нормы градиента в логарифмической шкале против реального времени работы, набор данных: gisette

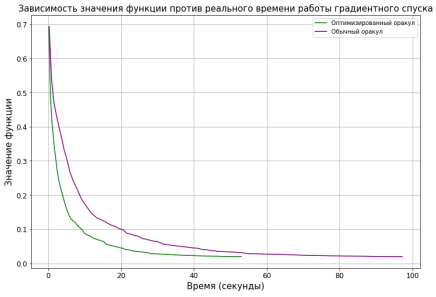
# 8 (Бонусная часть) Эксперимент: Стратегия выбора длины шага в градиентном спуске

В данном эксперименте мы исследуем, как зависит поведение градиентного спуска от стратегии подбора шага. Для этого мы рассматриваем логистическую регрессию и квадратичную функцию и с модельными данными (выбираются из случайного нормального распределения) и запускаем градиентный спуск с разными стратегиями выбора шага и из одной начальной точки. Для каждой стратегии мы подбираем соответствующие параметры, при которых количество итераций наименьшее. Для константного шага подбираем $c$ по массиву, состоящему из 3 чисел из диапозона от 0 до 1, для бэктрэкинга — аналогично подбираем $c$, для условия Вульфа — $c_2$ по массиву, состоящему из 3 чисел из диапозона от 0.5 до 1. Начальную точку $x_0$ перебираем по массиву, элементы которого сгенерированы как массивы из нормального распределения с нулевым матожиданием и дисперсией 3. Остальные параметры были выбраны по умолчанию.

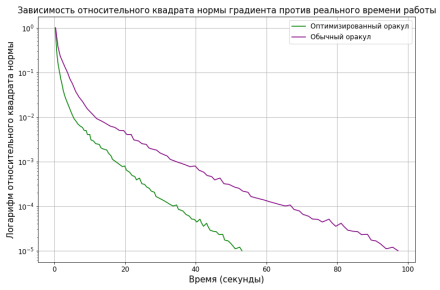
В предыдущем эксперименте мы увидели, что использование оптимизированного оракула влияет только на скорость сходимости, а не траекторию, поэтому при рассмотрении задачи логистической регрессии будем использовать его.




(a) Зависимость значения функции против номера итерации




(b) Зависимость значения функции против реального времени работы




(c) Зависимость относительного квадрата нормы градиента против реального времени работы, лог.шкала

Рис. 7.1: Сравнение градиентного спуска на логистической регрессии для обычного и оптимизированного оракула