Национальный исследовательский университет
«Высшая школа экономики»
Санкт-Петербургская школа физико-математических и компьютерных наук

Отчет к лабораторной работе № 2
«Продвинутые методы безусловной оптимизации»

Выполнил:
студент группы МОАД
Сморчков Данил Дмитриевич

Санкт-Петербург
2024 год

Оглавление

I. Зависимость числа итераций метода сопряженных градиентов от числа обусловленности и размерности пространства

В данном эксперименте необходимо исследовать, как зависит число итераций, необходимое методу сопряженных градиентов для сходимости, от следующих двух параметров: 1) числа обусловленности $\kappa \geq 1$ оптимизируемой функции и 2) размерности пространства n оптимизируемых переменных. По сути, весь эксперимент является копией эксперимента из прошлой лабораторной работы для градиентного спуска за исключением метода оптимизации.

Эксперимент проводился следующим образом:

Для пространств размерности $n \in \{2, 10, 100, 1000, 10000\}$, генерируется матрица с числом обусловленности k. Для этого сначала определяем диагональ как:

$$a_{11} = 1, \qquad a_{nn} = k, \qquad a_{ii} = randint(1, k),$$

затем создаем диагональную матрицу с данной диагональю и делим её на k для нормировки. Вектор b задаем как случайный вектор из стандартного нормального многомерного распределения размерности n. Ну и начальную точку x_0 задаем как случайный вектор из многомерного нормального распределения с вектором средних значений $\mu = \begin{pmatrix} 13 \\ \vdots \\ 13 \end{pmatrix}$ и матрицей ковариаций $\Sigma = \begin{pmatrix} 3^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 3^2 \end{pmatrix}$.

Из-за того, что матрица и начальная точка генерируются случайным образом, для каждого n и k эксперимент проводится несколько раз и усредняется, заливкой обозначена зона, отдаленная на стандартное отклонение.

Точность в данном эксперименте $\varepsilon = 10^{-9}$.

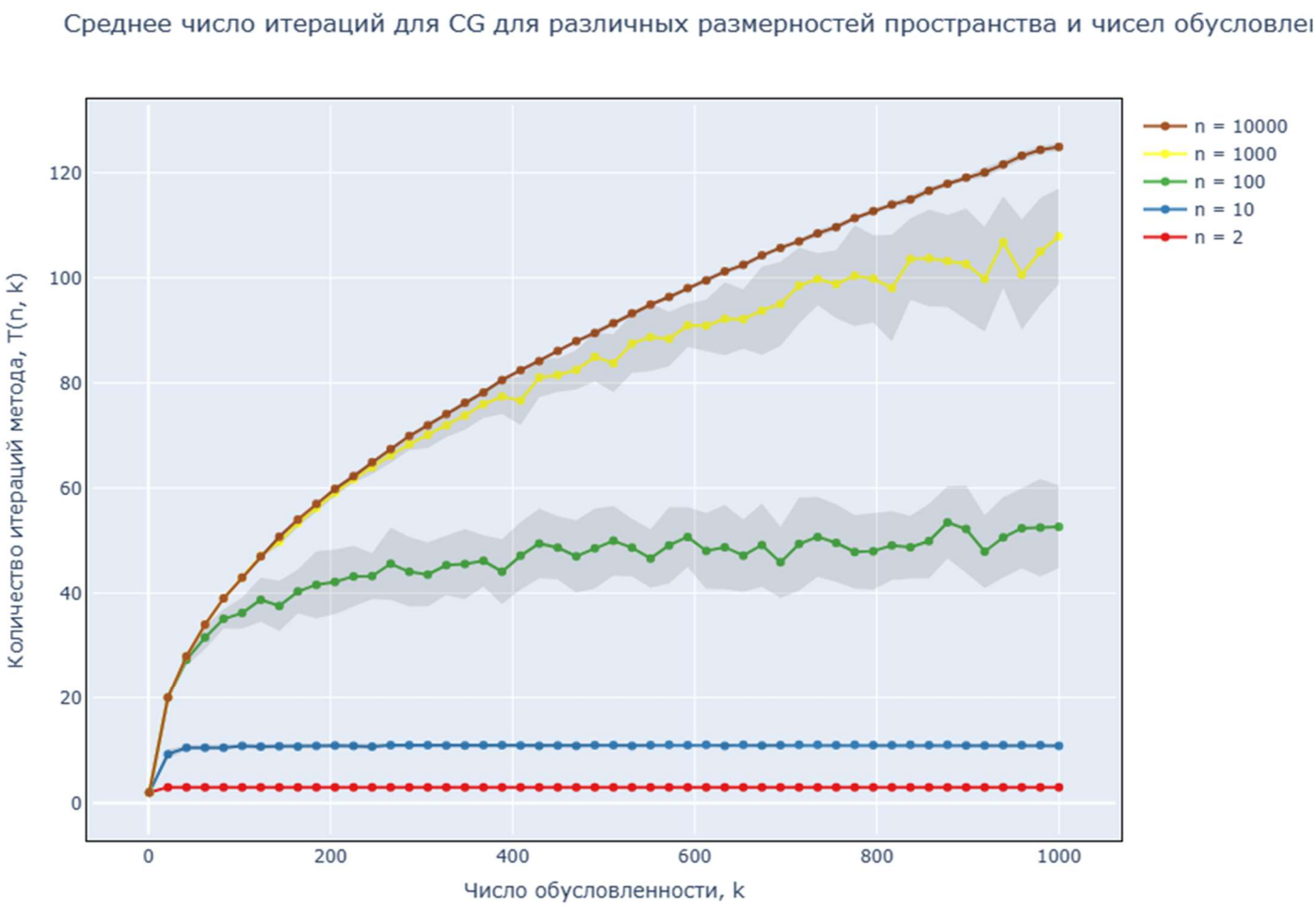

Рис 1. Число итераций CD для различных размерностей пространства и чисел обусловленности

Из рисунка выше четко прослеживается взаимосвязь между количеством итераций и числом обусловленности задачи: чем больше число обусловленности, тем больше итераций понадобится методу, чтобы сойтись к нужной точности, причем зависимость сублинейная.

Аналогично и с размерностью задачи: чем больше размерность задачи, тем больше итераций, однако здесь сложно сказать про вид зависимости.

Если сравнивать с градиентным спуском, то можно увидеть несколько отличий:

- Во-первых, сходится метод сопряженных градиентов гораздо быстрее – в случае использования условий Вульфа GD сходился более чем за 1000 итераций почти для всех размерностей при высокой обусловленности; метод CD же максимум выполняет 126 итераций.
- Во-вторых, в случае с CD, как уже упоминалось выше, есть явная зависимость между размерностью задачи и количеством итераций метода, а при использовании GD – сложно что-то сказать.
- В-третьих, CD ведет себя более стабильно в терминах количества итераций по сравнению с GD (график хоть и имеет изломы, но не такие крутые, как в эксперименте с градиентным спуском).

II. Выбор размера истории в методе L-BFGS

В данном эксперименте исследуется влияние размера истории в методе L-BFGS на поведение метода.

Сначала оценим сложность итерации и размер памяти метода L-BFGS:

Память. Во время работы метода нужно хранить вспомогательные вещи: x_k, x_{k-1}, $grad_k$, $grad_{k-1}$, однако самое затратное – именно история $\mathcal{H}_k := \left((s_{k-i}, y_{k-i})\right)_{i=1}^{m}$, что суммарно дает $O(n \cdot m)$.

Сложность. Приведу псевдокод из лабораторной работы, описывающий поиск направления в методе L-BFGS:

Алгоритм 1 Рекурсивное умножение L-BFGS матрицы на вектор

function BFGS_MULTIPLY(v, $\mathcal{H}$, γ_0)
  if $\mathcal{H} = \emptyset$ **then**
    return $\gamma_0 v$
  end if
  $(s, y) \leftarrow$ последняя пара из $\mathcal{H}$.
  $\mathcal{H}' \leftarrow \mathcal{H}$ без последней пары.
  $v' \leftarrow v - \frac{\langle s,v \rangle}{\langle y,s \rangle} y$
  $z \leftarrow$ BFGS_MULTIPLY($v', \mathcal{H}', \gamma_0$)
  return $z + \frac{\langle s,v \rangle - \langle y,z \rangle}{\langle y,s \rangle} s$.
end function

Алгоритм 2 Вычисление направления поиска d_k в методе L-BFGS

function LBFGS_DIRECTION
  $(s, y) \leftarrow$ последняя пара из $\mathcal{H}_k$
  $\gamma_0 \leftarrow \frac{\langle y,s \rangle}{\langle y,y \rangle}$
  return BFGS_MULTIPLY($-\nabla f(x_k)$, $\mathcal{H}_k$, γ_0)
end function

Из Алгоритма 1 видно, что сложность одного вызова функции BFGS_MULTIPLY равна $O(n)$, так как происходят скалярные произведение и суммирования векторов. Однако количество вызовов данной функции максимум равно количеству памяти да плюс один. Итого $O(n \cdot m)$. Конечно, еще расходы на вычисления α_k и другие небольшие операции, которые входят в $O(n \cdot m)$.

В данном эксперименте рассматривается несколько вариантов выбора размера истории, а именно $l \in \{0, 1, 5, 10, 50, 100\}$. Оценка производится по графикам:

– Зависимость относительного квадрата нормы градиента $\log_{10}\left(\frac{||\nabla f(x_k)||_2^2}{||\nabla f(x_0)||_2^2}\right)$ (в логарифмической шкале) против номера итерации.

– Зависимость относительного квадрата нормы градиента $\log_{10}\left(\frac{\|\nabla f(x_k)\|_2^2}{\|\nabla f(x_0)\|_2^2}\right)$ (в логарифмической шкале) против реального.

В качестве тестовой функции взята логистическая регрессия с l_2-регуляризатором. Данные *gesette* с сайта LIBSVM.

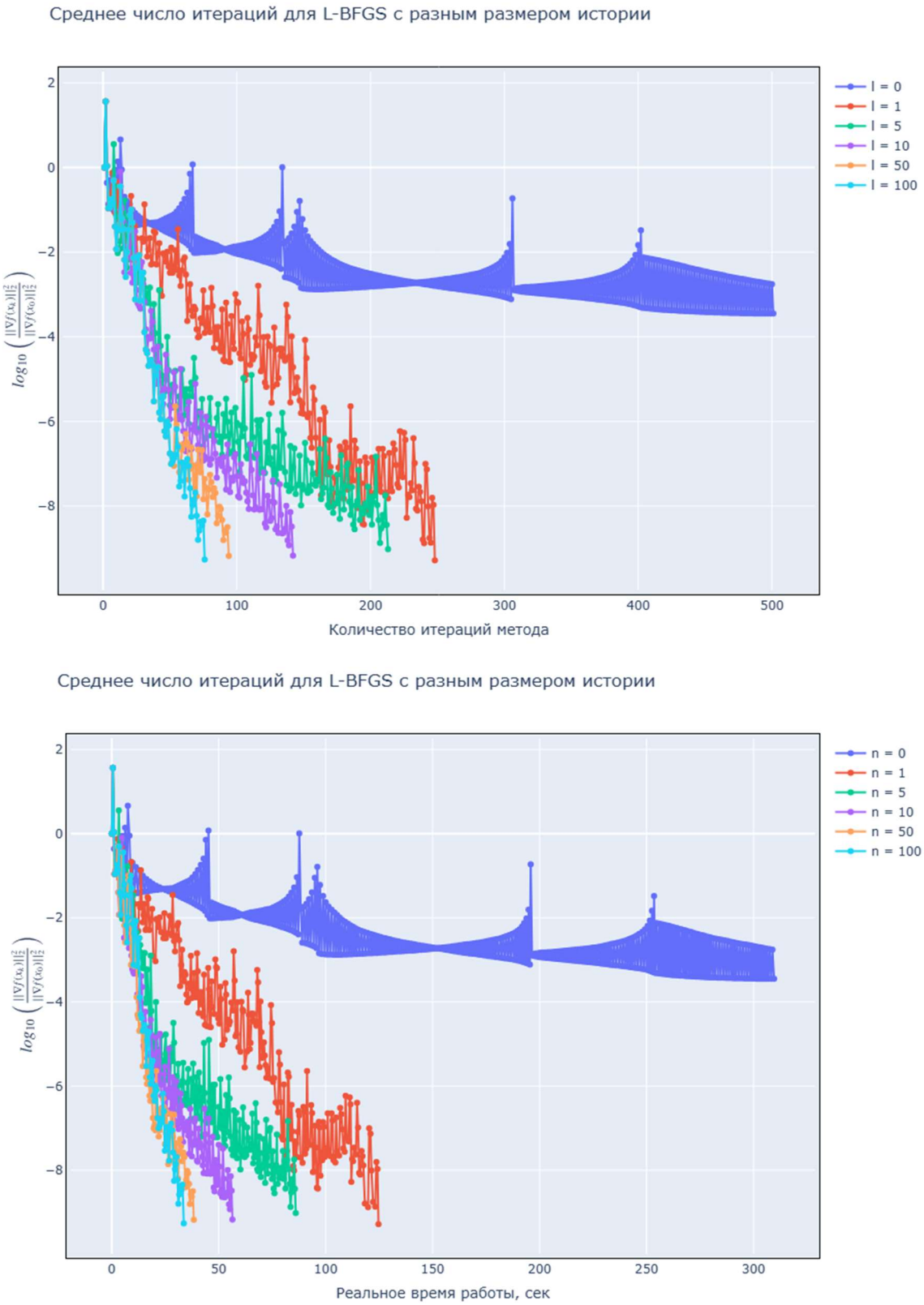

Рис 2. Графики относительной невязки в виде градиента в зависимости от размера истории

На графиках видно, что при отсутствии истории поведение метода очень отличается. Оно и понятно: если посмотреть на Алгоритмы 1 и 2, подставить $\mathcal{H} = \emptyset$ и $\gamma_0 = 1$, то получится обычный градиентный спуск. Неудивительно, что работает он гораздо медленнее.

Если рассмотреть методы с историей, то можно заметить, что итераций лучше всего сходится L-BFGS с большей историей. Однако нужно понимать, что при этом мы расходуем в разы больше памяти (хранить 10 векторов или 100…), поэтому иногда может быть более рационально использовать меньше памяти, но медленнее сходиться. Еще хочется обратить внимание на то, что чем больше памяти мы используем во время итераций, тем медленнее работает программа при прочих равных, то есть если бы мы взяли слишком большой размер истории, то помимо времени для совершения большего числа итераций для нахождения d_k, программа тратила бы больше времени просто потому, что оперативная память была бы больше забита.

III. Сравнение методов на реальной задаче логистической регрессии

В данном эксперименте необходимо сравнить усеченный метод Ньютона, L-BFGS и градиентный спуск на задаче обучения логистической регрессии на реальных данных. В качестве реальных данных используется следующие пять наборов с сайта LIBSVM3: w8a, gisette, real-sim, news20.binary и rcv1.binary.

Перейдем сразу к экспериментам:

I. Датасет w8a. Размер датасета: (49749, 300). Для градиентного спуска точность $\varepsilon = 10^{-5}$, для остальных методов – $\varepsilon = 10^{-9}$. Для хорошего масштабирования. В качестве стратегии линейного поиска использовалось условие Вульфа с параметрами по умолчанию.

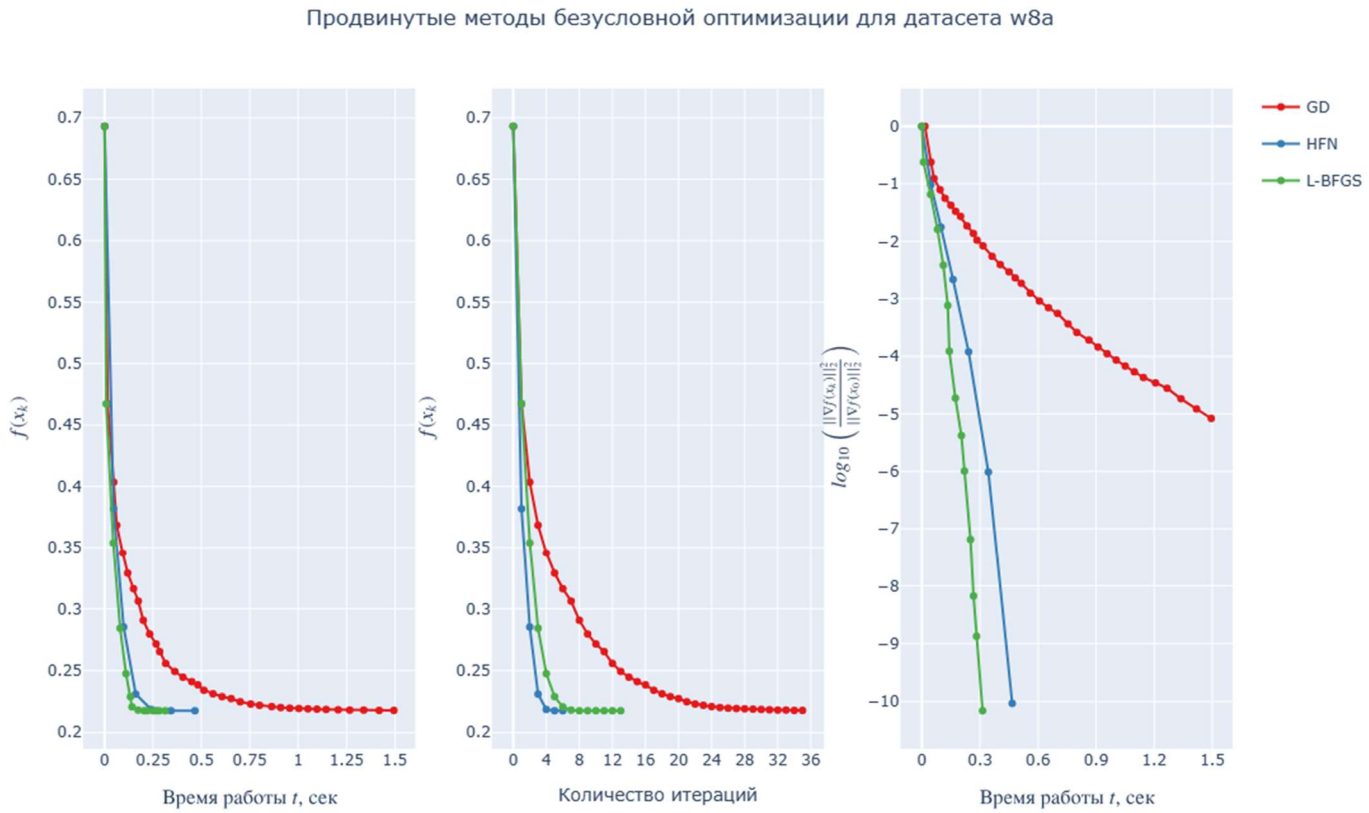

Рис 3. Графики относительной невязки в виде градиента и значений функции для различных методов оптимизации для датасета w8a

II. Датасет gissete. Размер датасета: (6000, 5000). Для градиентного спуска точность $\varepsilon = 10^{-4}$, для остальных методов – $\varepsilon = 10^{-9}$. В качестве стратегии линейного поиска использовалось условие Вульфа с параметрами по умолчанию.

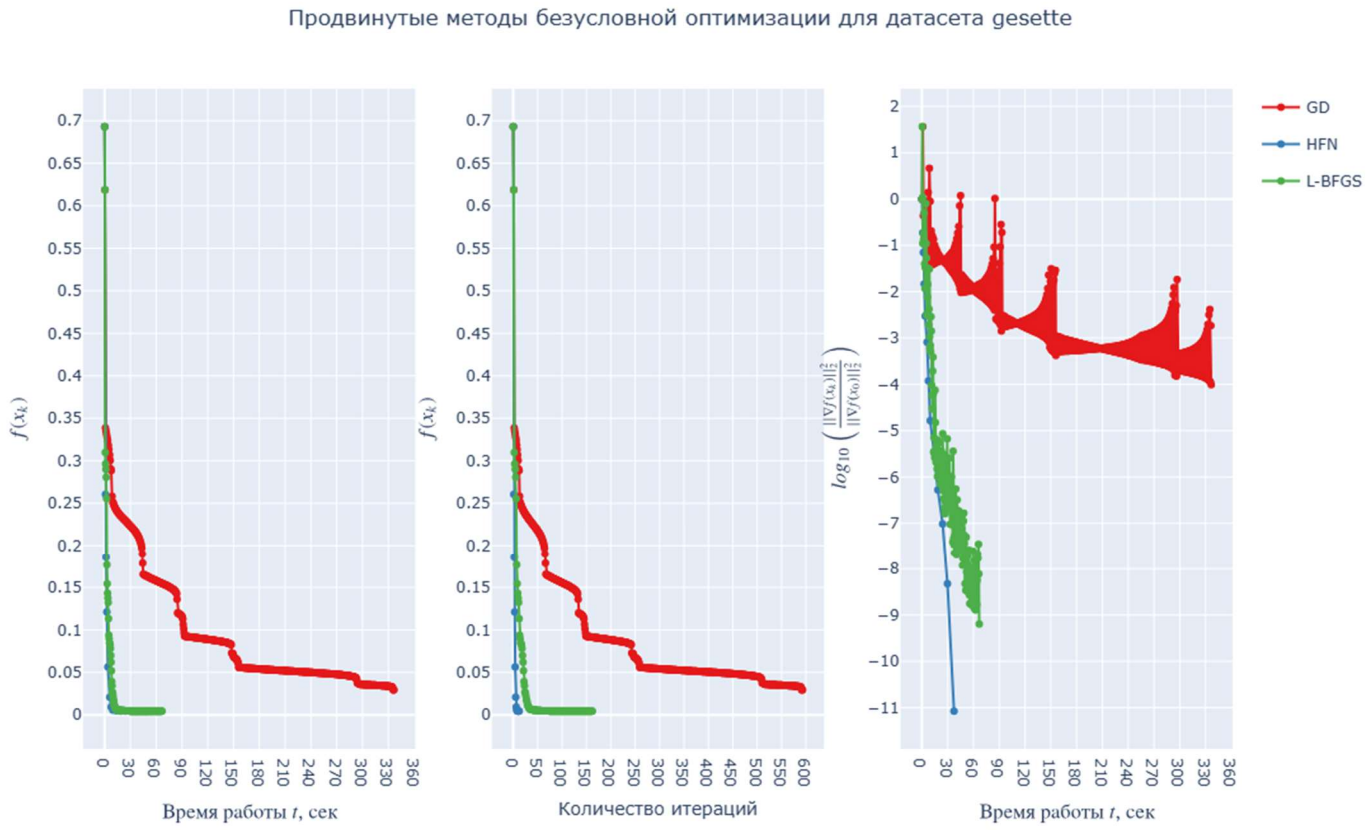

Рис 4. Графики относительной невязки в виде градиента и значений функции для различных методов оптимизации для датасета gisette

III. Датасет real-sim. Размер датасета: (72309, 20958). Данные разрежены. Для градиентного спуска точность $\varepsilon = 10^{-4}$, для остальных – $\varepsilon = 10^{-9}$. В качестве стратегии линейного поиска использовалось условие Вульфа с параметрами по умолчанию.

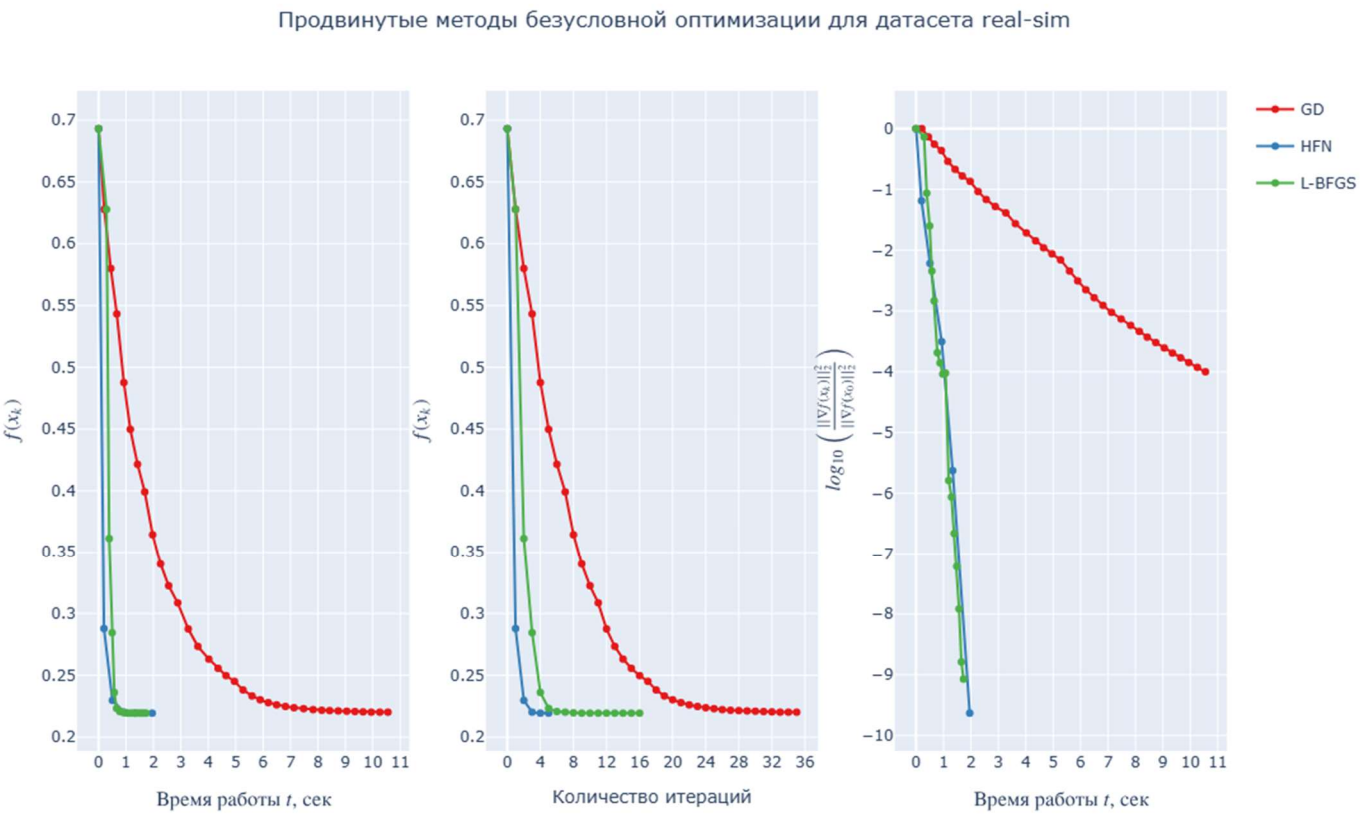

Рис 5. Графики относительной невязки в виде градиента и значений функции для различных для датасета real-sim

IV. Датасет news20.binary. Размер датасета: (19996, 1355191). Для градиентного спуска точность $\varepsilon = 10^{-4}$, для остальных – $\varepsilon = 10^{-9}$. В качестве стратегии линейного поиска использовалось условие Вульфа с параметрами по умолчанию.

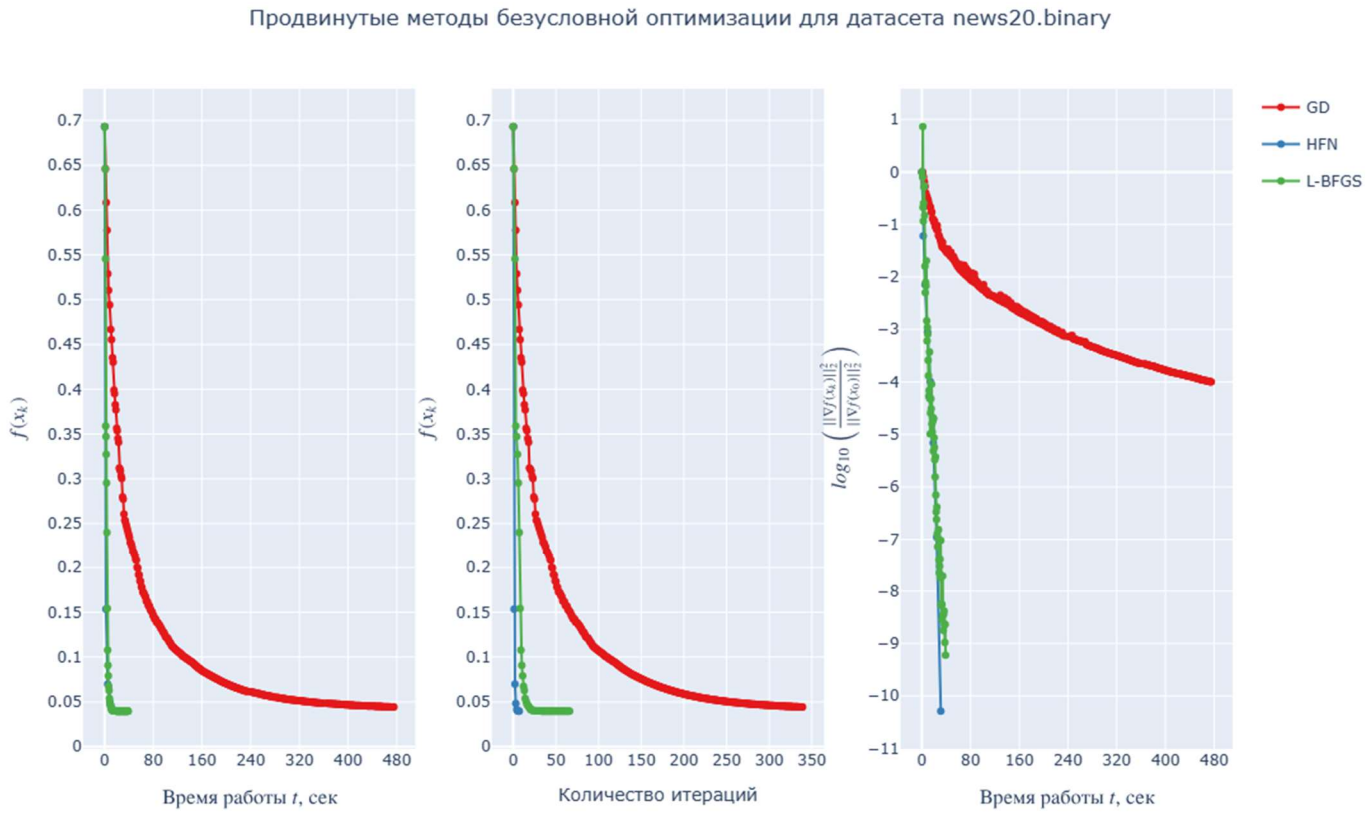

Рис 6. Графики относительной невязки в виде градиента и значений функции для различных для датасета news20.binary

V. Датасет rcv1.binary. Размер датасета: (20242, 47236). Для градиентного спуска точность $\varepsilon = 10^{-4}$, для остальных – $\varepsilon = 10^{-9}$. В качестве стратегии линейного поиска использовалось условие Вульфа с параметрами по умолчанию.

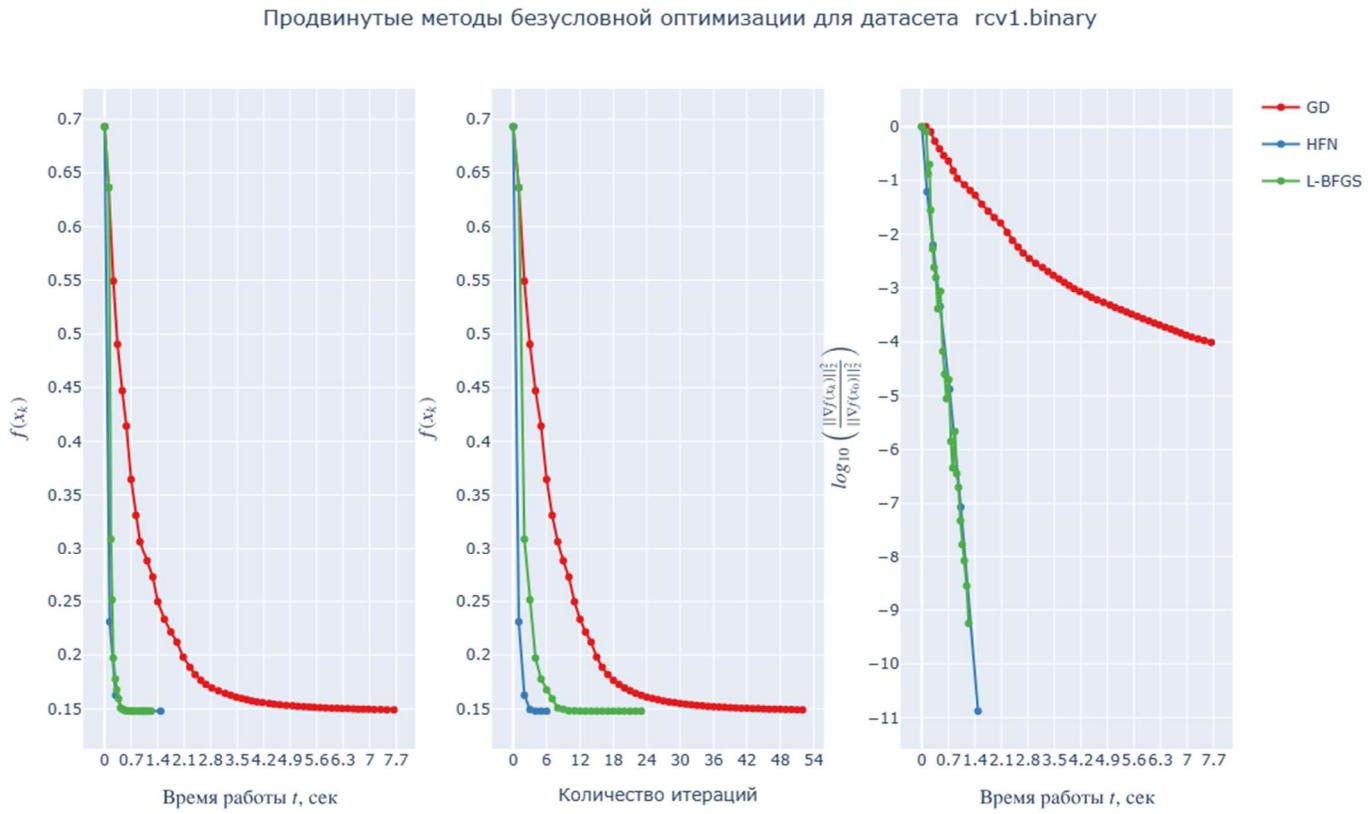

Рис 6. Графики относительной невязки в виде градиента и значений функции для различных для датасета rcv1.binary

Для всех датасетов ситуация похожая: градиентный спуск сходится сублинейно (для первого и третьего похоже на линейную скорость, однако кажется, что излом появляется). Продвинутые методы сходятся суперлинейно для всех датасетов, притом когда-то лучше работает L-BFGS, когда-то HFN, но для 3 датасетов из 5 по времени работы они справляются одинаково. Однако нужно понимать, что для HFN необходимо хранить гессиан, точнее произведение гессиана и вектора, однако суть в том, что в нашей реализации матрица хранится в памяти. В L-BFGS же храним $m = 10$ пар векторов, в наших задачах это линейные затраты на память (стоит просто взглянуть на размеры датасетов).

Я был очень приятно удивлен, когда увидел результаты этого эксперимента – мы действительно получили методы, которые сходятся сублинейно, как метод Ньютона, но позволяют использовать меньше памяти. При этом по сравнению со скоростью в прошлой лабораторной, работают эти методы намного быстрее.

IV. Сравнение метода сопряженных градиентов и L-BFGS на квадратичной функции

В данном эксперименте сравниваются методы сопряженных градиентов и L-BFGS на квадратичной строго выпуклой функции:

$$f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle, \quad x \in \mathbb{R}^n,$$

где $A \in S_{++}$ и $b \in \mathbb{R}^n$.

Эксперимент проводился следующим образом:

Генерировалась случайная положительно определенная матрица с обусловленностью, семплированной из $\mathbb{U}[20, 200]$; вектор b просто из стандартного нормального распределения, x_0 – вектор-нуль. Далее идет создание оракула и запуск методов CD и L-BFGS. Этот алгоритм повторяется 100 раз, затем вычисляются среднее и стандартное отклонение относительно всех запусков на каждой итерации методов для того, чтобы получить более уверенный результат с точки зрения статистики.

Для того, чтобы эксперимент был честный в методе L-BFGS используется точный поиск оптимальной длины шага, так как в методе сопряженных градиентов α_k по определению получается точным. С этой целью выведем α_k:

$$\alpha_k = argmin_{\alpha>0} f(x_k + \alpha d_k)$$

$$\nabla_\alpha f(x_k + \alpha d_k) = \nabla_{x_k+\alpha d_k} f(x_k + \alpha d_k) \cdot d_k = (A(x_k + \alpha d_k) - b)^T d_k = 0$$

$$\boxed{\alpha = \frac{(b - Ax_k)^T}{d_k^T A d_k} d_k = -\frac{\nabla_{x_k} f(x_k)^T d_k}{d_k^T A d_k}}$$

Таким образом, на каждом шаге метода L-BFGS оптимальный шаг линейного поиска будет вычисляться по формуле выше.

Первый раз запускаем эксперимент с $l = 10$ (Рис. 7). Темной областью отмечена зона, отходящая на одно стандартное отклонение от среднего.

Возможно, на графике не очень хорошо видно, но оба метода полностью совпадают как по средним значениям, так и по дисперсии.

Далее заменили l со значения 10 на 1 (Рис. 8) Однако результат остался таким же, как и в прошлом случае.

Совершенно очевидно, когда мы вовсе убрали память и тем самым включили градиентный спуск вместо L-BFGS, на графике появились две отдельные кривые, одна из которых имеет сублинейную скорость сходимости (градиентный спуск), а другая, по-видимому, линейную (CD).

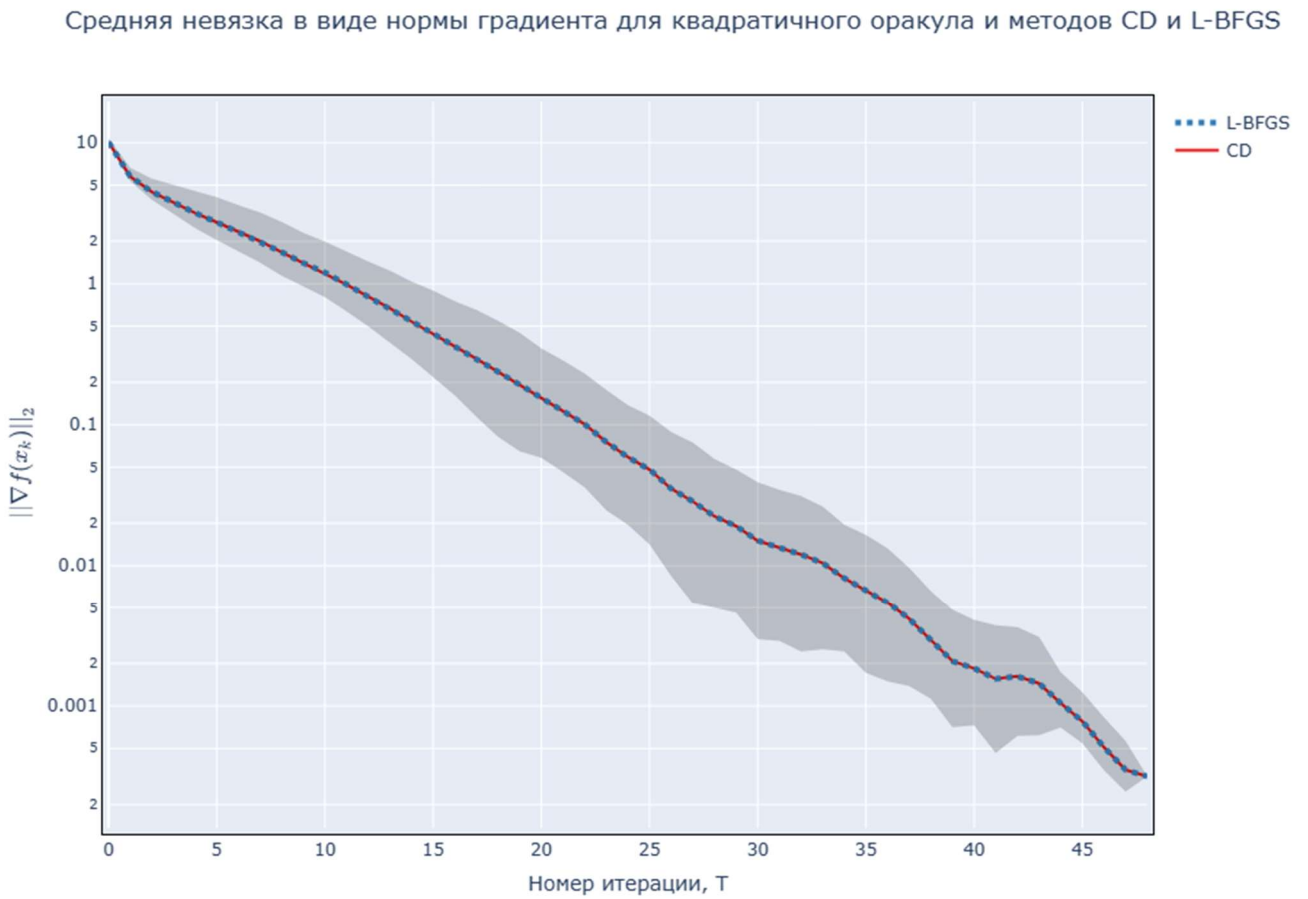

Рис 7. Средняя невязка для методов CD и L-BFGS с памятью размера 10 для квадратичного оракула

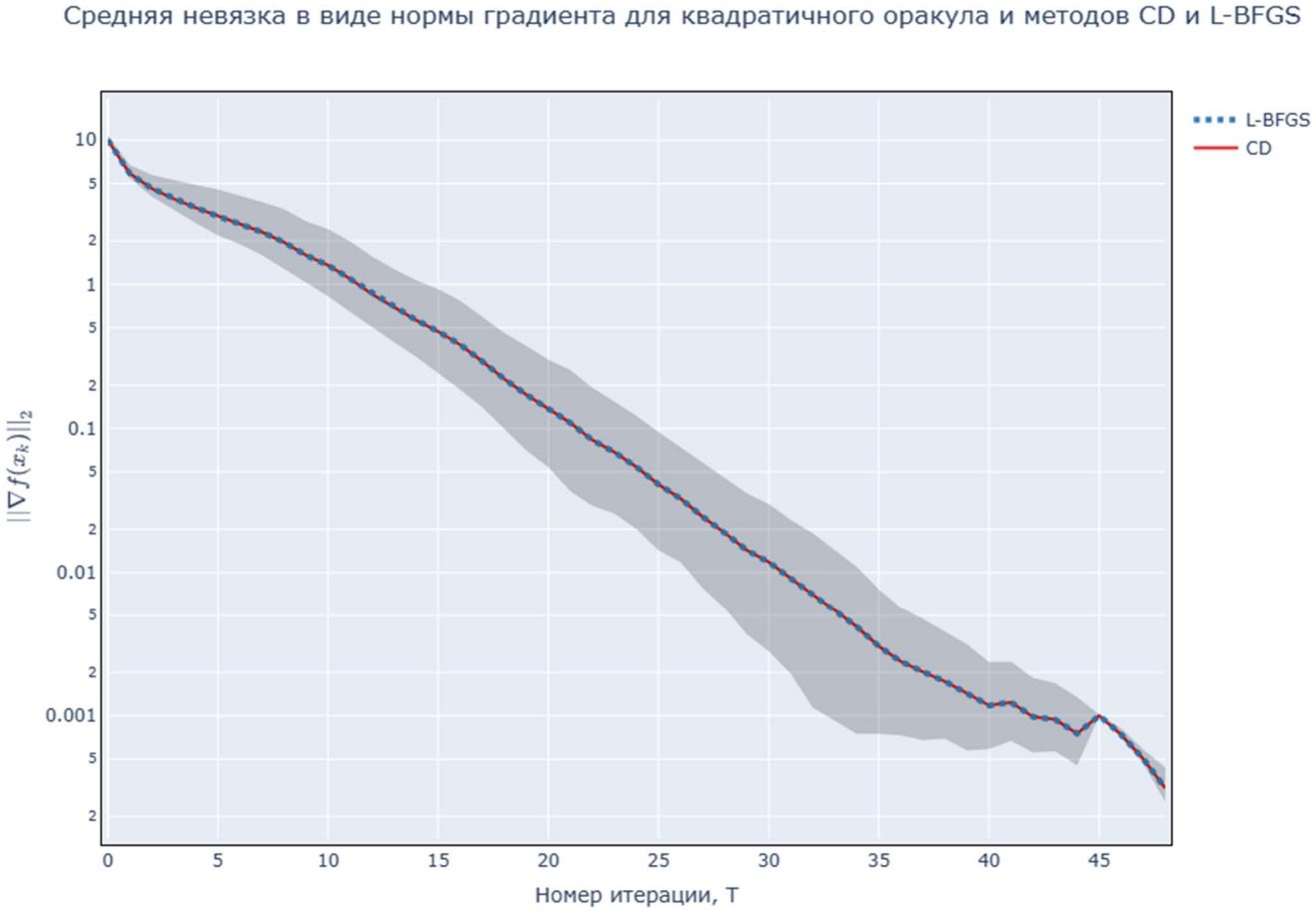

Рис 8. Средняя невязка для методов CD и L-BFGS с памятью размера 1 для квадратичного оракула

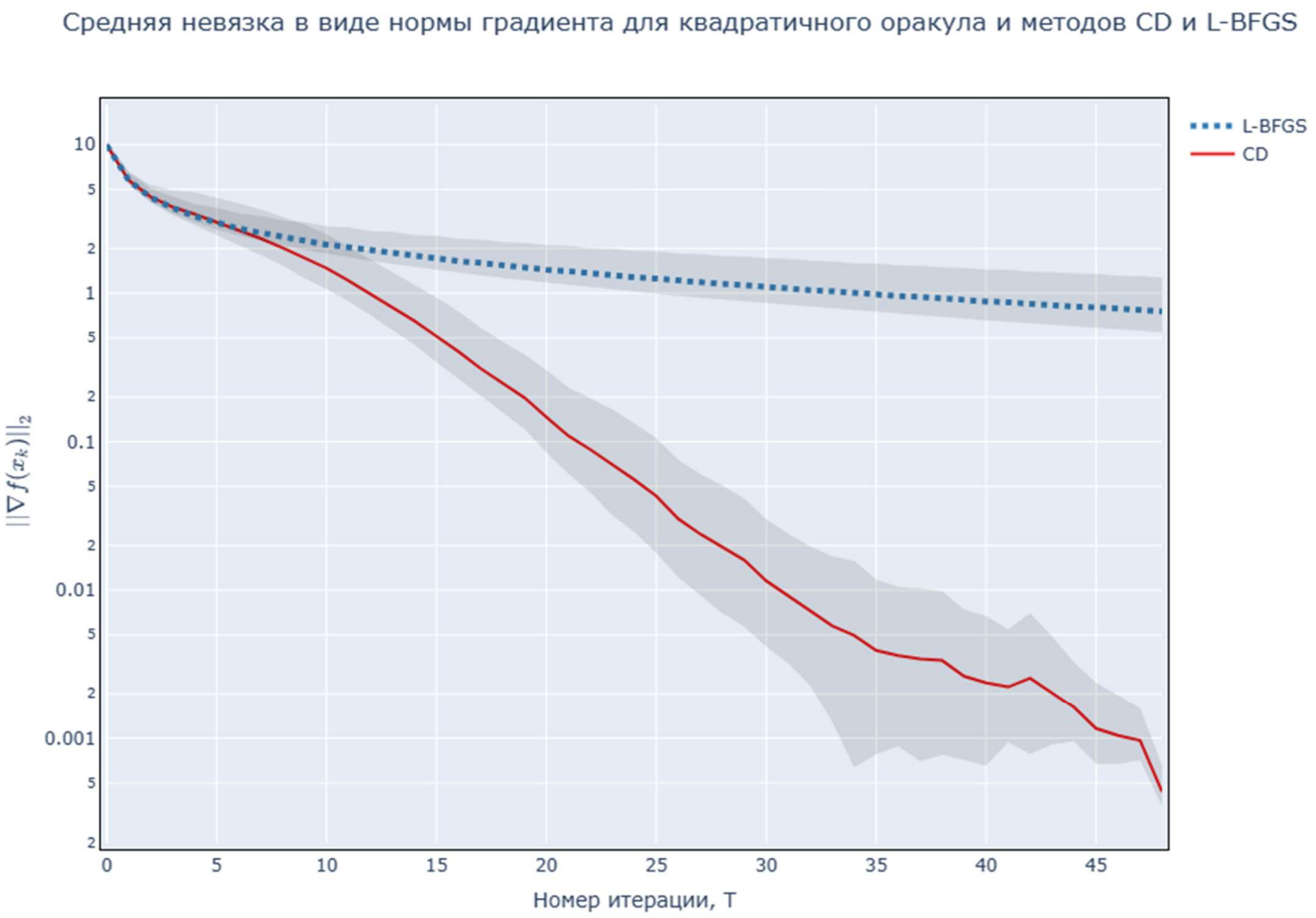

Рис 9. Средняя невязка для методов CD и L-BFGS с памятью размера 0 для квадратичного оракула

Таким образом, мы получили, что если в методе L-BFGS есть память (что заведомо должно быть), то при решении задачи минимизации квадратичной функции метод ведет себя аналогично методу сопряженных градиентов.

V. Какая точность оптимизации нужна в реальных задачах?

В данном эксперименте исследуется вопрос влияния точности оптимизации целевой функции на итоговое качество решения исходной задачи. В качестве задачи берется бинарная классификация, а целевой функции – функция потерь логистической регрессии с l_2-регуляризатором.

Для проведения эксперимента был взят метод L-BFGS. Для различных датасетов (w8a, real-sim, news20.binary) были созданы обучающая и тестовая выборки (в отношении 0.8:0.2 во всех случаях). Далее на обучающей выборке был запущен метод оптимизации с различными ε. Для каждого ε была взята итоговая точка $\hat{x}$, которую вернул метод и сравнивалось качество прогноза логистической регрессии $\hat{b}_{test} = sign(A_{test}\hat{x}_{test})$ с истинными значениями метод b_{test}. Ошибка вычислялась как среднее число позиций, в которых векторы b_{test} и $\hat{b}_{test}$ не совпадают.

Коэффициент регуляризации и начальную точку возьмите стандартными: $\lambda = \frac{1}{m}$ и $x_0 = 0$.

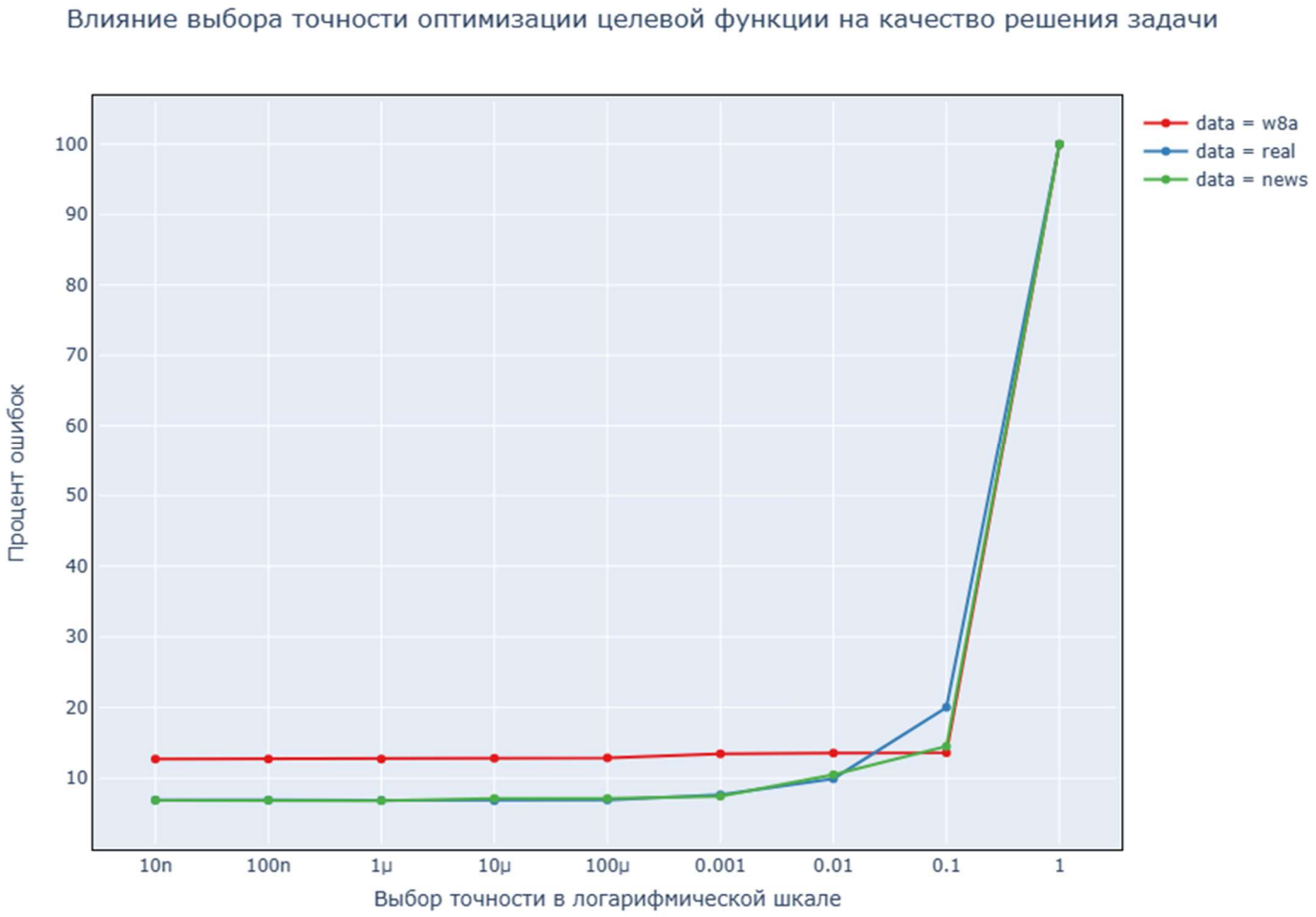

Рис 10. Зависимость процента ошибок от выбора точности для разных данных

Мы получили, что чем выше точность модели, тем меньше процент ошибок. Это и логично: в нашей модели присутствует регуляризация, поэтому модель не переобучается во время оптимизации, однако, возможно, если бы её не было, то графики выглядели бы по-другому.

Также можно сделать вывод, что иногда повышение точности не приводит к сильным изменениям качества решения задачи, например, на графиках при $\varepsilon \leq 0.0001$ процент ошибок практически не изменяется. И тут нужно сделать выбор, что нам важнее: точность или время? Для каждой задачи ответ будет своим.

Вывод

В данной работе были проведены различные эксперименты для исследования продвинутых методов безусловной оптимизации: CD, HFN, L-BFGS.

Была проведена оценка скорости сходимости всех методов на реальных датасетах.

Были исследованы влияние размерности задачи и обусловленности функции для метода сопряженных градиентов.

Также рассматривалась зависимость работы метода L-BFGS от размера хранимой истории.

В одном из экспериментов обнаружена взаимосвязь методов CD и L-BFGS для задачи минимизации квадратичной функции.

В конце было рассмотрено взаимосвязь оптимизации с реальным качеством задачи.

Мне подсказал Андрей Широбоков, что у меня неправильно реализован L-BFGS, за что я ему очень благодарен. (А именно, он подсказал использовать параметр maxlen в deque; ошибка была в том, что я добавлял новую пару (s, y) в начало, а нужно было в конец…)