Лабораторная работа 2: Продвинутые методы безусловной оптимизации

Белова Юлия

30 мая 2023 г.

Содержание

1 Эксперимент 1: Зависимость числа итераций метода сопряженных градиентов от числа обусловленности и размерности пространства

В данном эксперименте рассматривается квадратичная задача. Исследуем, сколько итераций необходимо совершить методу сопряженных градиентов до сходимости в зависимости от числа обусловленности матрицы и количества оптимизируемых переменных. Для проведения эксперимента были выбраны следующие параметры:

1. Размерность пространства n: $[10, 10^2, 10^3, 10^4, 10^5]$

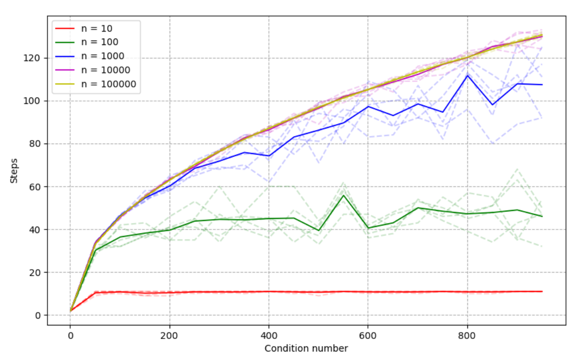
2. Число обусловленности матрицы k: числа в интервале $[1, 1000]$ с шагом 50

Для каждой размерности пространства n задача генерировалась случайным образом 5 раз. В градиентном спуске использовались параметры по умолчанию:

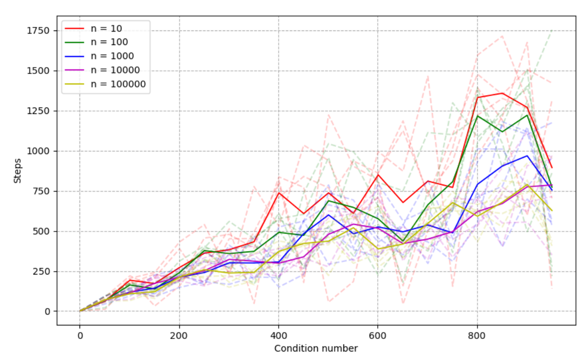
- Начальная точка $x_0 = 0$

- Метод подбора шага - алгоритм Вульфа с параметрами $c_1 = 1\text{e}{-4}$, $c_2 = 0.9$

На рис. 1 отображены результаты эксперимента: ярким цветом выделены средние результаты (по случайным генерациям) для каждого n.



(a) Метод сопряженных градиентов



(b) Градиентный спуск

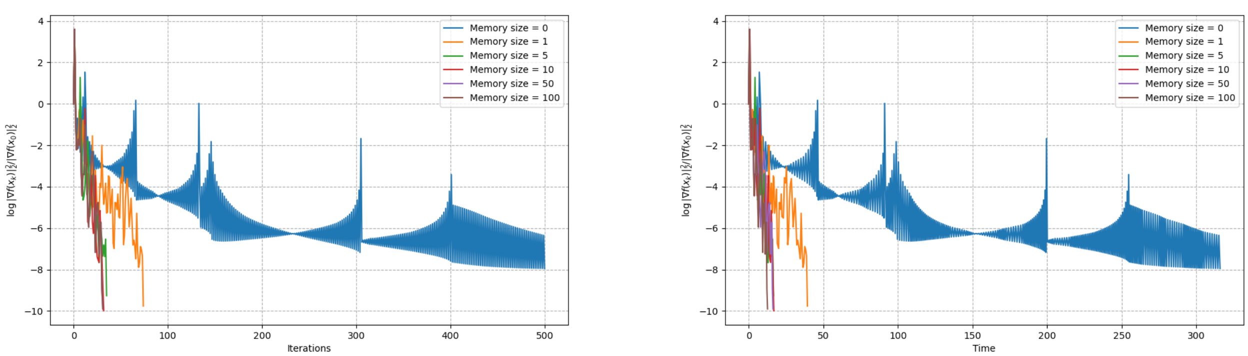
Рис. 1: Зависимость числа итераций от числа обусловленности и размерности пространства

По результатам эксперимента можно сделать следующий вывод: по сравнению с градиентным спуском метод хорошо работает в случае больших чисел обусловленности. Число итераций не превосходит размерности пространства даже при высоких значениях числа обусловленности. Зависимость числа итераций от числа обусловленности - сублинейная, в то время как в градиентном спуске зависимость линейная. Также видно, что количество итераций практически не зависит от размерности пространства.

2 Эксперимент 2: Выбор размера истории в методе L-BFGS

Исследуем, как влияет размер истории в методе L-BFGS на поведение метода. Пусть l - размер истории, n - размерность пространства. Методу L-BFGS требуется $\mathcal{O}(l \cdot n)$ памяти и $\mathcal{O}(l \cdot n)$ времени на одну итерацию. Такая память, потому что храним l векторов s, y, вектор градиента, текущую точку. На итерацию нужно $\mathcal{O}(l \cdot n)$ времени, потому что совершаем две проходки, внутри которых совершаем l переходов, каждый из которых занимает линейное время - умножение векторов. Данный метод, по сравнению с ранее изученными, в теории выигрывает как по памяти, так и по времени на одну итерацию. Для проведения эксперимента были выбраны следующие параметры:

1. Данные *gisette*, где размерность пространства $n = 5000$, число наблюдений $m = 6000$.
2. Размер истории: $[0, 1, 5, 10, 50, 100]$
3. $x_0 = 0$
4. $\lambda = \frac{1}{m}$



(a) Зависимость относительного квадрата нормы градиента против номера итерации

(b) Зависимость относительного квадрата нормы градиента против реального времени работы

Рис. 2: Выбор размера истории

При размере истории, равном нулю, метод $L - BFGS$ вырождается в градиентный спуск. Из рис. 2 видно, что в таком случае метод сходится достаточно долго (ему не хватило 500 итераций). Если сделать размер истории равным единице, то метод сойдется всего за 75 итераций, причем с высокой точностью. Для размеров памяти $10, 50, 100$ метод сошелся за 32 итерации и за одно и то же время - 12 секунд. Можно сделать вывод, что оптимальнее всего хранить 10 предыдущих значений.

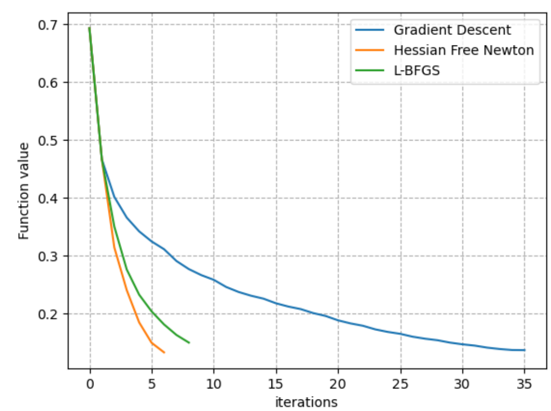
3 Эксперимент 3: Сравнение методов на реальной задаче логистической регрессии

Для сравнения усеченного метода Ньютона, метода L-BFGS и градиентного спуска на задаче обучения логистической регрессии были использованы пять датасетов: *w8a*, *gisette*, *real-sim*, *news20.binary*, *rvc1.binary*.

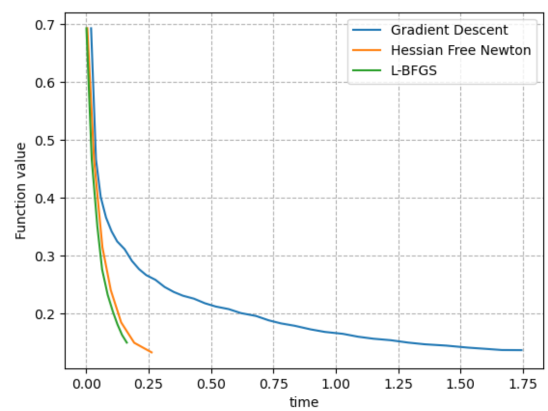
Параметры методов взяты по умолчанию, начальная точка $x_0 = 0$. Пусть d - размерность пространства, N - число наблюдений, тогда методу HFN на совершение одной итерации (а конкретно нахождению направления с помощью CG и эвристикой в умножении гессиана на вектор) потребуется $\mathcal{O}(Nd^2)$ времени, а в случае разреженных матриц все работает сильно быстрее. Методу L-BFGS требуется, как уже упоминалось выше, $O(m \cdot d)$ времени на одну итерацию, где m - размер памяти.

3.1 w8a

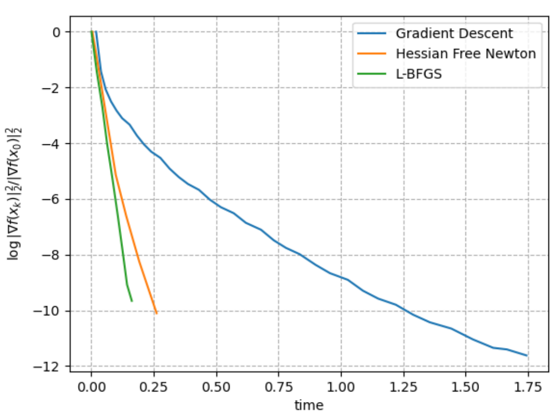
Количество наблюдений в датасете - 49 740, количество признаков - 300.



(a) Зависимость значения функции от номера итерации метода



(b) Зависимость значения функции от реального времени работы метода



(c) Зависимость относительного квадрата нормы градиента от реального времени работы

1. HFN - 7 итераций, 0.024 сек./итер.

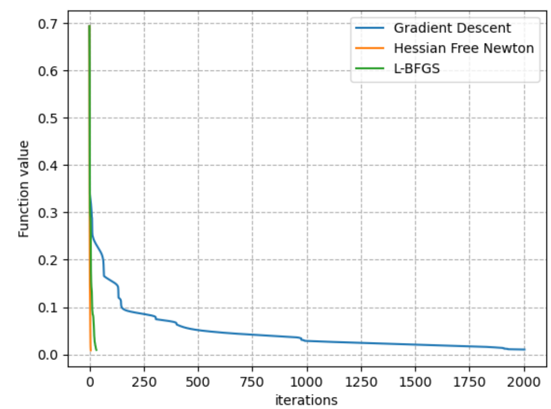
2. L-BFGS - 9 итераций, 0.012 сек./итер.

3. GD - 36 итераций, 0.034 сек./итер.

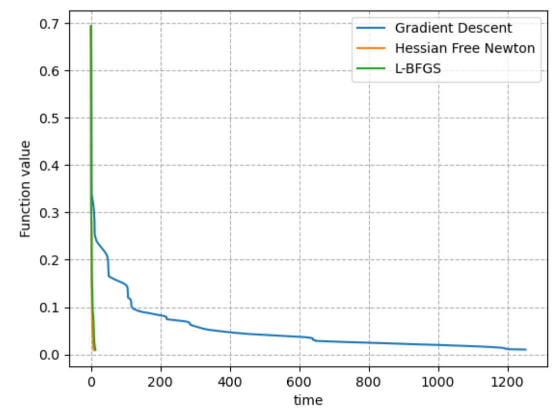
Данные имеют невысокую размерность, быстрее всего отработал метод L-BFGS, больше всего итераций потребовалось градиентному спуску.

3.2 gisette

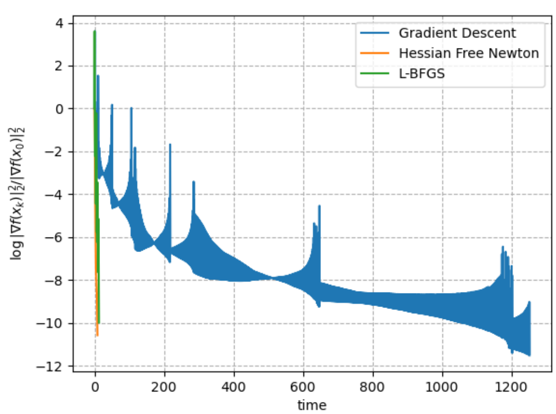
Количество наблюдений - 6 000, количество признаков - 5 000. Данный датасет хранится в формате `scipy.sparse.csr_matrix`, однако 99% данных имеют ненулевые значения.



(a) Зависимость значения функции от номера итерации метода



(b) Зависимость значения функции от реального времени работы метода



(c) Зависимость относительного квадрата нормы градиента от реального времени работы

1. HFN - 7 итераций, 1.147 сек./итер.

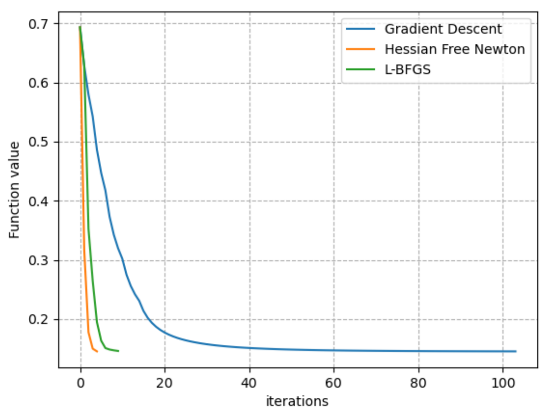
2. L-BFGS - 33 итерации, 0.377 сек./итер.

3. GD - 2004 итерации, 0.483 сек./итер.

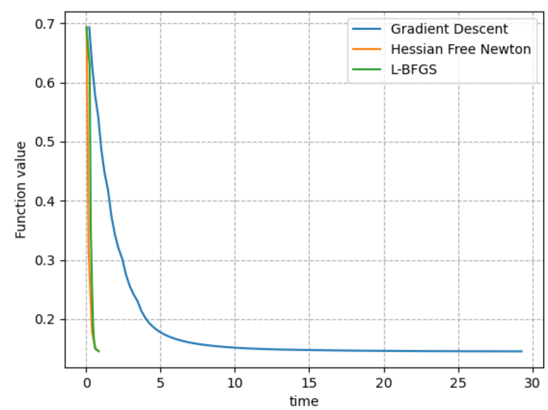
Здесь уже заметна большая разница между методами (HFN, L-BFGS) и градиентным спуском. Датасет имеет неразреженные данные высокой размерности, градиентному спуску требуется сильно больше итераций чтобы сойтись, хотя по стоимости итерация примерно такая же (вспомним метод Ньютона - там итерация стоила 500 секунд). Наши новые методы HFN и L-BFGS сходятся на порядок быстрее, поэтому их предпочтительнее использовать в случае неразреженных данных.

3.3 real-sim

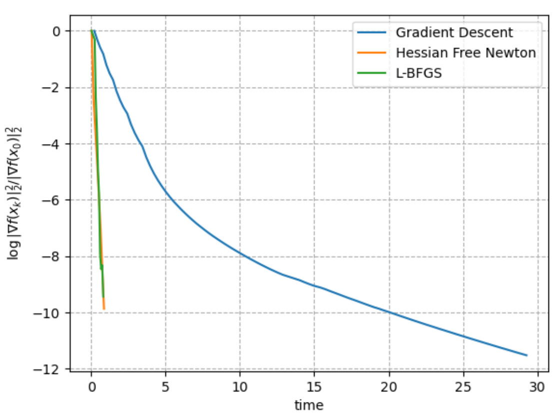
Количество наблюдений - 72 309, количество признаков - 20 958. Данный датасет хранится в формате `scipy.sparse.csr_matrix`, процент ненулевых значений в матрице с данными 0.002%.



(a) Зависимость значения функции от номера итерации метода



(b) Зависимость значения функции от реального времени работы метода



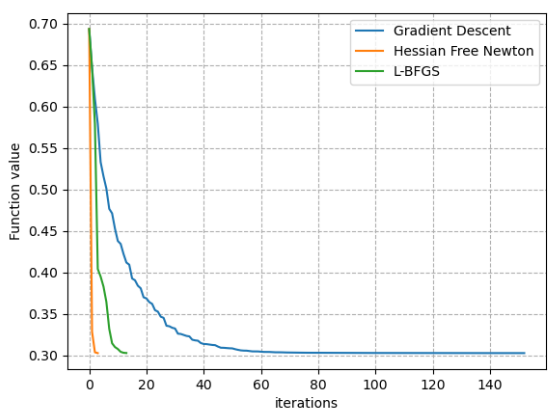
(c) Зависимость относительного квадрата нормы градиента от реального времени работы

1. HFN - 5 итераций, 0.184 сек./итер.
2. L-BFGS - 10 итераций, 0.081 сек./итер.
3. GD - 104 итерации, 0.296 сек./итер.

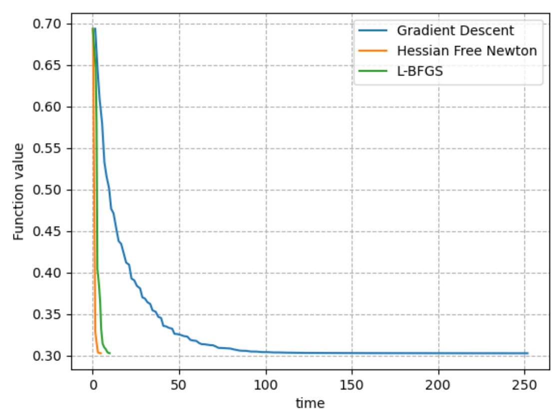
Данные сильно разреженные, здесь градиентный спуск сходится достаточно быстро, но все равно проигрывает методам HFN и L-BFGS. Видна закономерность: методу L-BFGS требуется больше итераций, чем методу HFN, однако итерация стоит дешевле. Подтверждаются наши предыдущие наблюдения - количество итераций не зависит от размерности пространства. На данных размерности 300 ($w8a$) методу HFN потребовалось 7 итераций, в то время как на текущем датасете с данными размерности около 21 тысячи число итераций даже стало меньше - 5!

3.4 news20.binary

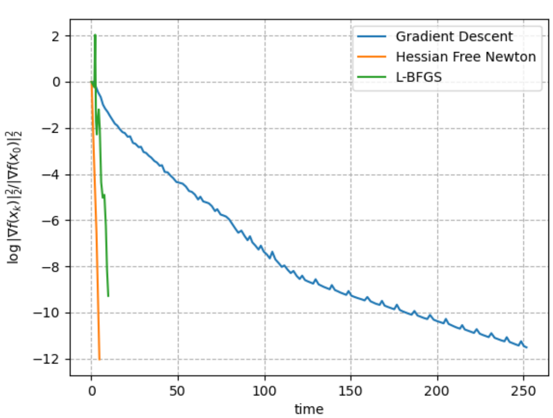
Количество наблюдений - 19 996, количество признаков - 1 335 191. Хранится в формате `scipy.sparse.csr_matrix`, процент ненулевых значений в матрице с данными 0.0003%.



(a) Зависимость значения функции от номера итерации метода



(b) Зависимость значения функции от реального времени работы метода



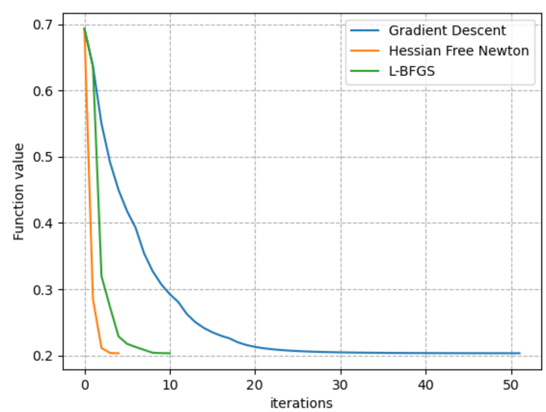
(c) Зависимость относительного квадрата нормы градиента от реального времени работы

1. HFN - 4 итераций, 0.999 сек./итер.
2. L-BFGS - 14 итераций, 0.483 сек./итер.
3. GD - 153 итераций, 1.35 сек./итер.

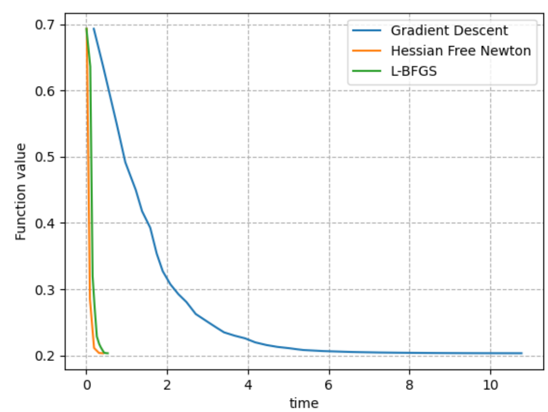
Все те же самые закономерности: HFN быстрее сходится, чем L-BFGS, но стоимость итерации выше. Здесь уже метод L-BFGS оказался более чувствительным к размерности пространства и ему потребовалось больше итераций, чем в прошлых случаях, в отличие от HFN.

3.5 rcv1.binary

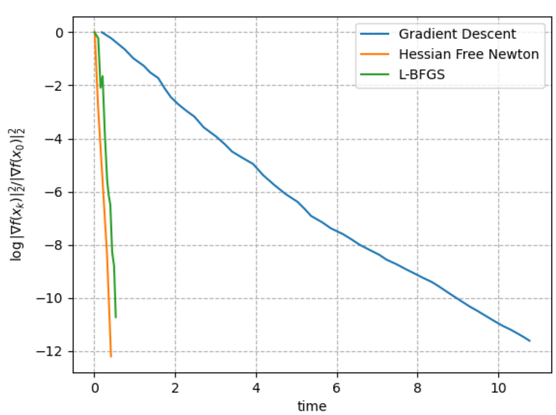
Количество наблюдений - 20 242, количество признаков - 47 236. Хранится в формате `scipy.sparse.csr_matrix`, процент ненулевых значений в матрице с данными 0.002%.



(a) Зависимость значения функции от номера итерации метода



(b) Зависимость значения функции от реального времени работы метода



(c) Зависимость относительного квадрата нормы градиента от реального времени работы

1. HFN - 5 итераций, 0.07 сек./итер.
2. L-BFGS - 11 итераций, 0.033 сек./итер.
3. GD - 52 итерации, 0.118 сек./итер.

Выводы:

1. Градиентный спуск - самый долгий метод по сравнению с остальными для любого из рассматриваемых датасетов. Хуже всего градиентный спуск работает на неразреженных данных.

2. Метод HFN сходится быстрее (по количеству итераций), чем L-BFGS, однако по времени итерация стоит чуть дороже. В целом, по совокупному времени на нахождение оптимального решения они работают примерно одинаково. Сходимость обоих методов не зависит (точнее почти не зависит) от размерности задачи.