Федеральное государственное автономное образовательное учреждение
высшего образования «Национальный исследовательский университет
ИТМО»

Отчёт

Лабораторная работа №3
по методам оптимизации

Выполнили:
Абрамов Илья Дмитриевич M32341
Кузнецов Илья Дмитриевич M32341

Санкт-Петербург 2023

Лабораторная работа # 3
Методы высокого порядка

Предполагаемый язык выполнения лабораторных работ Python 3. Лабораторные работы выполняются студентами индивидуально или в группах по 2-3 человека (по желанию). По результатам выполнения лабораторной работы необходимо подготовить отчет. Отчет должен содержать описание реализованных вами алгоритмов, ссылку на реализацию, необходимые тесты и таблицы.

Постановка задачи

1. Реализовать методы Gauss-Newton и Powell Dog Leg для решения нелинейной регрессии. Сравнить эффективность с методами реализованными в предыдущих работах.

2. Реализовать метод BFGS и исследовать его сходимость при минимизации различных функций. Сравнить с другими реализованными методами.

Дополнительное задание

Реализовать и исследовать метод L-BFGS

Критерии оценивания

1. Работоспособность и качество кода.

2. Полнота отчета: наличие постановки задачи, описания методов, промежуточных выводов, результатов, а также графиков и таблиц, которые их демонстрируют.

3. Знание теории, которая лежит в основе применяемых методов.

4. Анализ результатов, преимуществ и ограничений методов.

5. Дополнительное задание.

Каждый критерий оценивается максимально в 5 баллов. Итого максимальный балл за лабораторную работу: 25 баллов.

Состав команды: Абрамов Илья M32341, Кузнецов Илья M32341

Gauss-Newton

Алгоритм Гаусса Ньютона используется для решения задачи наименьших квадратов

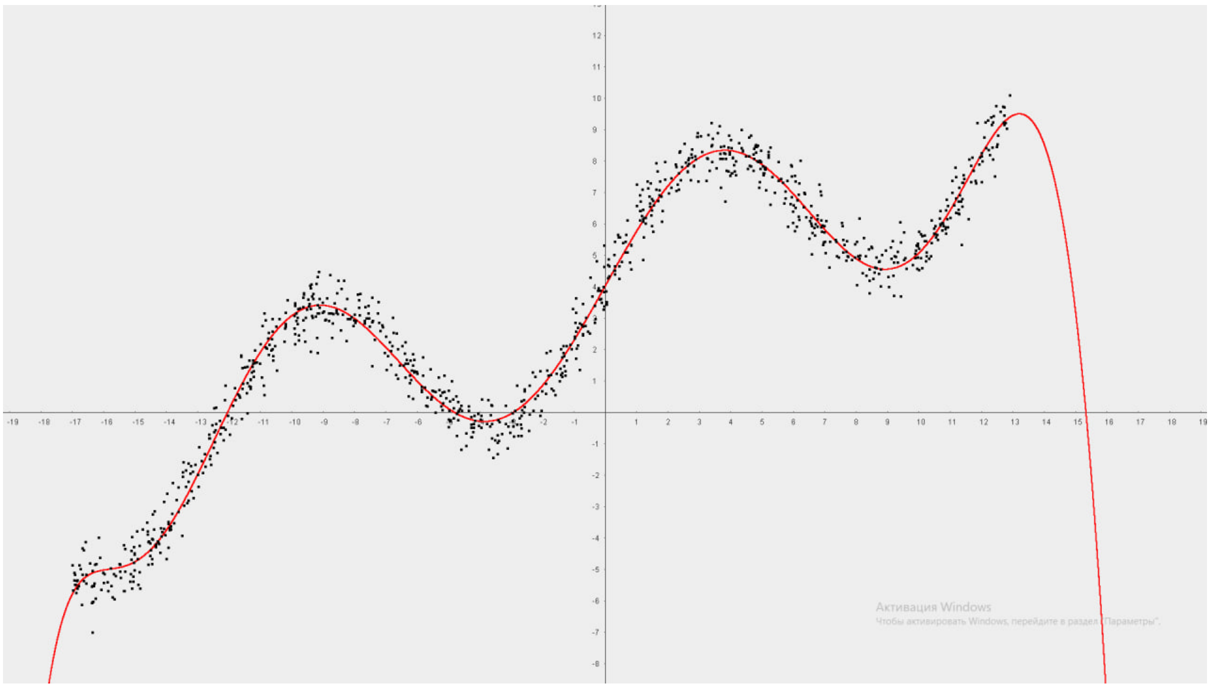
Пусть задано m функций $f = (f_1, f_2, \dots, f_m)$ от n переменных $x = (x_1, x_2, \dots, x_n)$. Задача найти:

$$\arg\min_x \sum_{i=1}^{m} f_i^2(x)$$

Алгоритм работает следующим образом: пусть у нас есть некоторое приближение x_0, тогда следующее приближение будет считаться по формуле:

$x_{i+1} = x_i - (J^T J)^{-1} J^T f(x_i)$, где J - матрица Якоби $J_{i,j} = \dfrac{df_i}{dx_j}$

Решим с помощью этого алгоритма задачу полиномиальной регрессии.



Как видно на рисунке, алгоритм работает.

Powell Dog Leg

Этот алгоритм решает ту же задачу, что и алгоритм Гаусса Ньютона. Алгоритм работает следующим образом: пусть у нас есть некоторое число Δ - радиус, внутри которого будут производиться все шаги, некоторое приближение x_0, x_i - текущее приближение,

$\delta_{gn} = (J^T J)^{-1} J^T f(x_i)$ - шаг в алгоритме Гаусса Ньютона.

$\delta_{sd} = J^T f(x_i)$ - шаг в алгоритме наискорейшего спуска

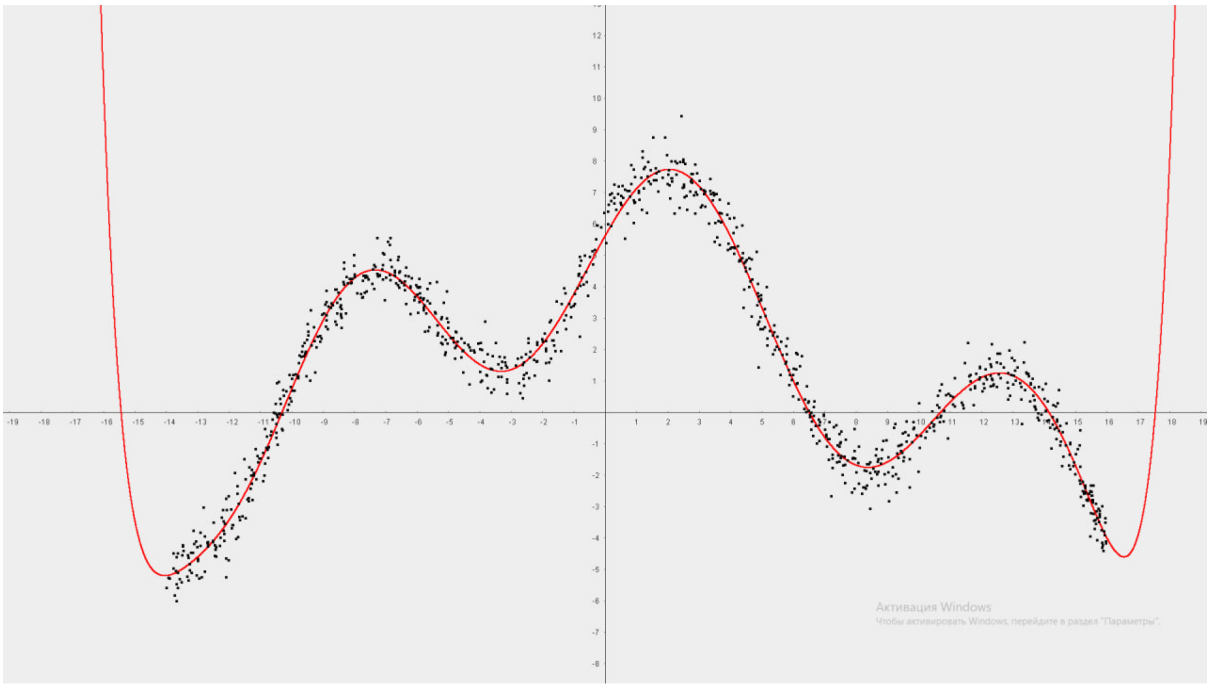
$t = \frac{||\delta_{sd}||^2}{||J\delta_{sd}||^2}$, тогда

1) $\delta = \delta_{gn}$, если $||\delta_{gn}|| \leq \Delta$
2) $\delta = \frac{\Delta}{||\delta_{sd}||}\delta_{sd}$, если $||\delta_{gn}|| > \Delta$ и $t||\delta_{sd}|| > \Delta$
3) $\delta = t\delta_{sd} + s(\delta_{gn} - t\delta_{sd})$, s - такое, что $||\delta|| = \Delta$, иначе

Следующее приближение будет считаться по формуле:

$x_{i+1} = x_i - \delta$

Решим с помощью этого алгоритма задачу полиномиальной регрессии.



Как видно на рисунке, алгоритм также работает.

BFGS

Пусть решается задача поиска $argmin f(x),\ x \in R^n$
Алгоритм стартует в некотором начальном приближении x_0.
Каждое следующее приближение находится по следующему принципу
$x_{k+1} = x_k + \alpha_k * p_k$, где α_k удовлетворяет условиям Вольфе.
$p_k = -B_k^{-1} * \nabla f(x_k)$, где B_k - приближенное значение гессиана функции в данной точке. В силу дороговизны вычисления обратной матрицы, мы на каждом шаге просто обновляем значения уже имеющейся:

$$B_{k+1}^{-1} = \left(I - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) B_k^{-1} \left(I - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$$

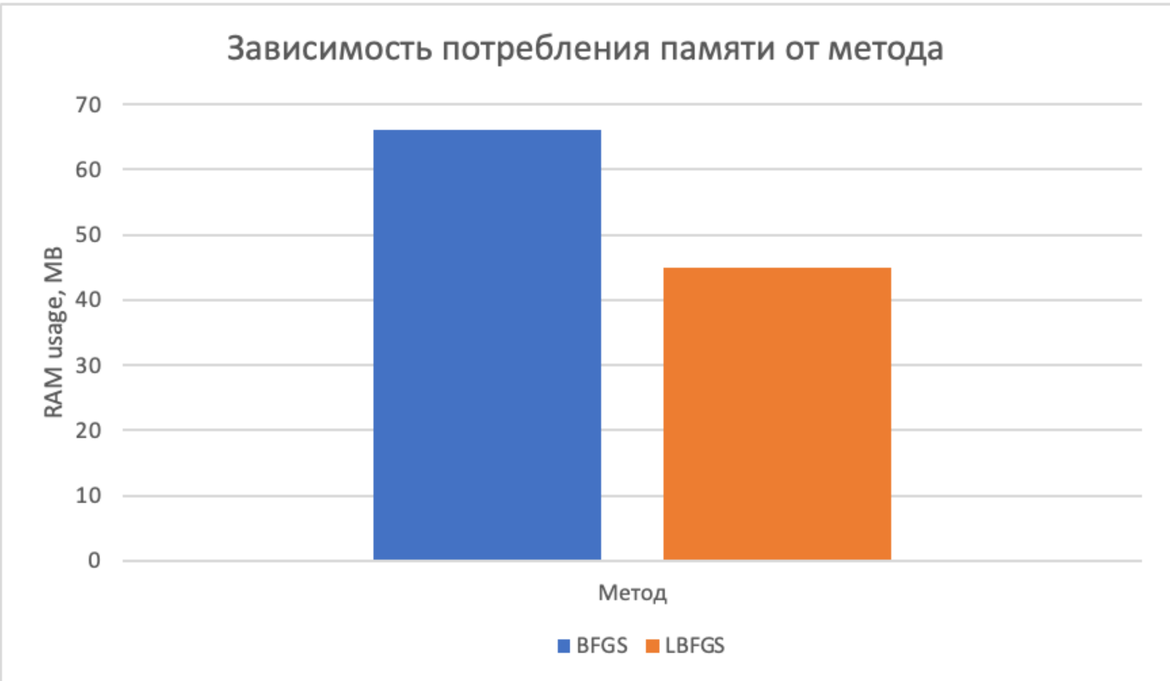
$$s_k = x_{k+1} - x_k,\ y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

L-BFGS

Проблема BFGS заключалась в том, что нам приходится хранить матрицу, что делает использование памяти в методе квадратичным. Вместо этого будем хранить значение s_k, y_k, α_k, полученные на последних m итерациях метода. Этого будет достаточно, чтобы с необходимой точностью получить следующее приближение x_k.

Рассмотрим потребление памяти методами BFGS и LBFGS на примере следующей функции

$$f = \sum_{i=0}^{49} (x_i - 30)^2 + \sum_{i=50}^{99} (2x_i - 30)^2 + \sum_{i=100}^{500} (3x_i - 30)^2$$

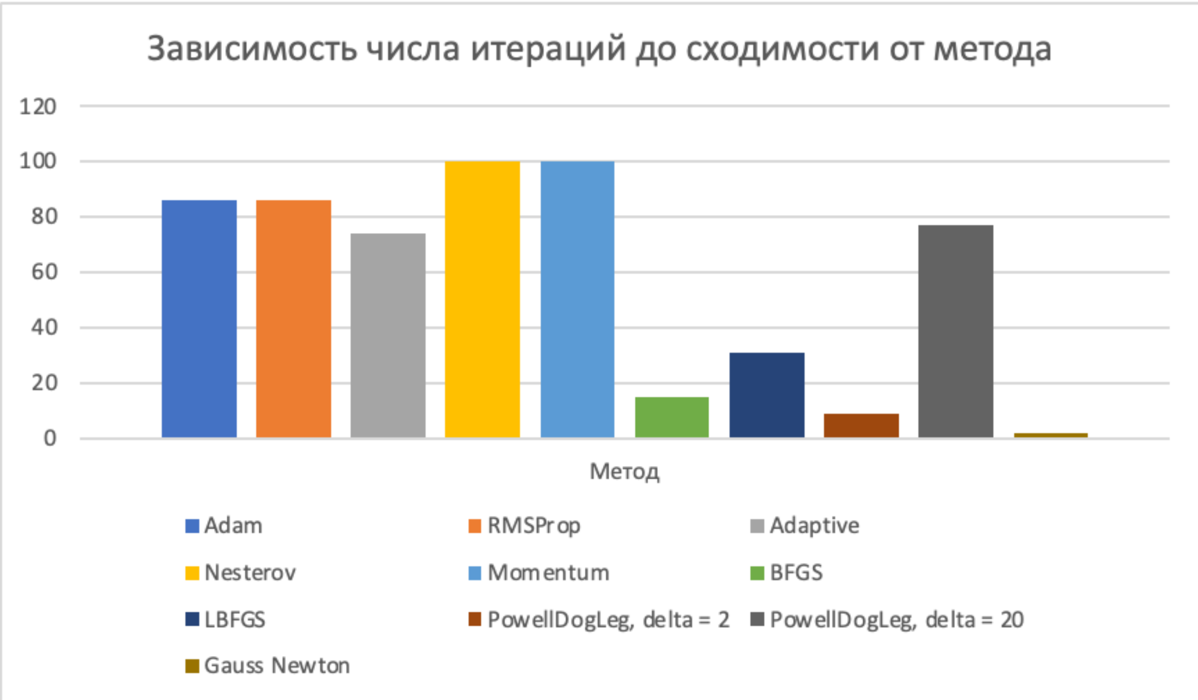


Видно, что при использовании LBFGS мы тратим меньше памяти.

Сравнение методов.

Сравним методы Gauss-Newton, Powell Dog Leg, BFGS и LBFGS с реализованными ранее методами на примере следующей функции

$$f = \sum_{i=0}^{49}(x_i - 30)^2 + \sum_{i=50}^{99}(2x_i - 30)^2 + \sum_{i=100}^{149}(3x_i - 30)^2$$



Видно, что метод Gauss-Newton сходится на порядок быстрее других методов, а скорость сходимости метода Powell Dog Leg напрямую зависит от радиуса trust region: при большом радиусе он будет сходится почти так же быстро, как и метод Gauss-Newton, а при маленьком радиусе он будет сходится довольно долго.
Также видно, что BFGS и LBFGS сходятся быстрее всех ранее реализованных методов, и медленнее методов Gauss-Newton и Powell Dog Leg.