Национальный исследовательский университет ИТМО
Факультет информационных технологий и программирования
Прикладная математика и информатика

Методы оптимизации

Отчёт по лабораторной работе №3

Работу выполнили:

Салахов Камиль M33381

Мухтаров Айнур M33371

Преподаватель:

Свинцов Михаил Викторович

Санкт-Петербург, 2023

1 Задачи:

Код:https://colab.research.google.com/drive/1BocKkr_umSkC2nOyFgUGAvElYBU42LEx?usp=sharing

1. Реализовать методы Gauss-Newton и Powell Dog Leg для решения нелинейной регрессии. Сравнить эффективность с методами реализованными в предыдущих работах.
2. Реализовать BFGS и исследовать его сходимость при минимизации различных функций. Сравнить с другими реализованными методами.
3. Реализовать и исследовать метод L-BFGS.

2 Нелинейная регрессия

Имеется m точек (x_i, y_i) - значения функции $y(x)$ $(y(x_i) = y_i)$, которую мы хотим приблизить и функция регрессии $F_w(x)$ зависящая от n параметров $w = (w_1, ..., w_n), n < m$ которые мы будем подбирать, стараясь приблизиться к исходной функции. Для решение задачи нелинейной регрессии сведём её к задаче Least squares, где

$r = (r_1, r_2, ...r_m)$ - регрессионные остатки (невязки) $r_i(w) = F(w, x_i) - y_i$, сумму квадратов которых мы будем минимизировать. Задача Least squares:

Рассмотрим функцию $f(x)$ от $x = (x_1, x_2, ..., x_n)$, где $f(x) = \frac{1}{2}\sum_{k=1}^{m} r_i^2(x)$ - минимизируемая функция

Введем обозначение якобиана: $J(x) = \begin{bmatrix} \frac{\partial r_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial r_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial r_m(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla r_1(x)^T & \cdots & \nabla r_m(x)^T \end{bmatrix}^T$

Тогда градиент f можно выразить через якобиан и вектор ошибок: $\nabla f(x) = J(x)^T r(x)$

Аналогично выразим Гессиан: $\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) * \nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) = J(x)^T J(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x)$. $r(x)$ - нелинейная функция, следовательно применим соответствующие методы.

2.1 Метод Gauss-Newton

Воспользуемся методом Ньютона для одномерного случая: $\nabla^2 f(x_k)p = -\nabla f(x_k)$ и подставим в него соответсвующие значения из нашей задаче: $J_k^T J_k p_k + \sum_{j=1}^{m} r_j \nabla^2 r_j p_k = -J_k^T r_k$. Так как невязки в окрестности минимума становятся близки к линейным, второе слагаемое в полученном выражении для $\nabla^2 f(x)$ близко к нулю и в общем случае много меньше первого. Тогда решая уравнение $J_k^T J_k p_k = -J_k^T r_k$. Получаем, что $p_k = -(J_k^T J_k)^{-1} J_k^T r_k$.

2.2 Метод Powell Dog Leg

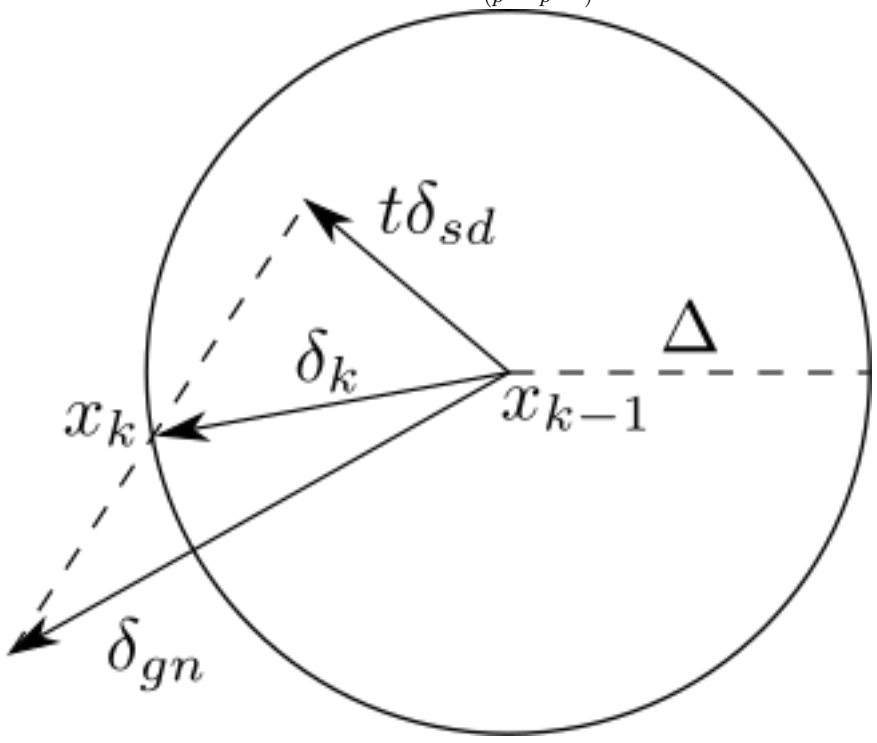
Построим квадратичную модель $m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$ и ищем минимум $m_k(p)$ при $\|p\| < \Delta_k$, т.е. шаг в шаре с радиусом дельта (регион), в котором действует наша модель. Возьмём в качестве g_k градиент, а B_k гессиан функции и следовательно направление можно выразить как $p = -B_k^{-1} g_k$.

Коофициент точности модели: $\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$, где p_k - напревление следующего шага по методу Ньютона.

В зависимости от коофициента мы расширяем регион если модель хорошо описывает нашу функцию, либо сужаем его, либо оставлем прежним.

Если коээфициент больше заданного гиперпараметра, то алгоритм переходит в новую точку $x_{k+1} = x_k + p_k$, иначе остаётся в прежней и пытается изменить регион снова.

Если p_k лежит в регионе то используем его, иначе построим линейную модель - $m(p) = f + g^T p$. Для неё мы можем найти минимум способом аналогичным в градиентном спуске. Пусть $p^N(\delta_{gn})$ шаг по квадратичной модели, $p^{GD}(\delta_{sd})$ - по линйеной, тогда пусть $\widetilde{p}^{GD}$ минимальный шаг по квадратичной модели по направлению антиградиента равный $\widetilde{p}^{GD} = -\frac{g^t g}{g^t B g} g$. Если он за пределами региона - берём $p, \|p\| = \Delta_k$ по этому направлению, иначе минимум будет находиться на дуге соединяющей p^N и $\widetilde{p}^{GD}$. Найдём минимум для $\widetilde{p}(\tau) = \widetilde{p}^{GD} + (\tau - 1)(p^N - \widetilde{p}^{GD}), 1 \leq \tau \leq 2$, получается $\tau = -\frac{(\widetilde{p}^{GD} - \Delta)(p^N - \widetilde{p}^{GD})}{(p^N - \widetilde{p}^{GD})^2}$



2.3 Примеры работы методов

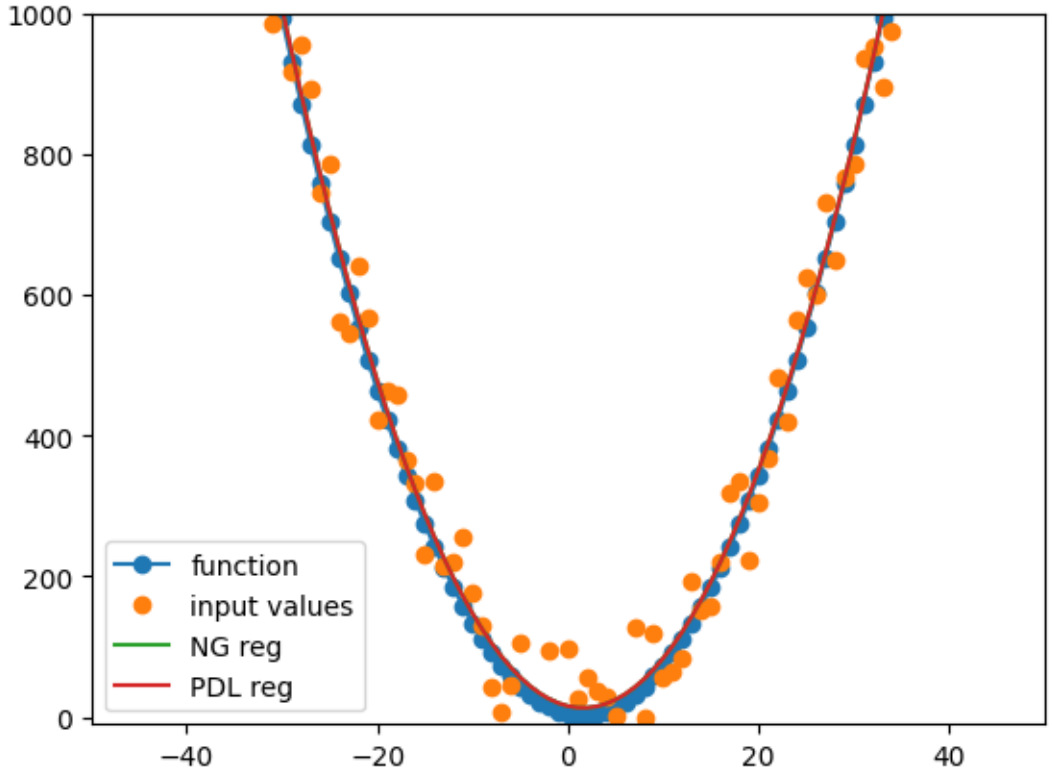
Функция: $y = x^2 - 3x + 4$

Регрессия: полиноминальная (2 степень)

Значения: целые точки от -40 до 40 с шагом 1

Шум: Каждое реальное значение функции отличается от изменённого не более чем на 200 (прибавление константы к функции)

Начальные значения: $\vec{1}$



Метод Gauss-Newton отработал 1 эпоху со значениями $[9.90492265, -3.13679557, 0.991662]$

Метод Powell Dog Leg отработал 20 эпох со значениями $[9.90492264, -3.13679557, 0.991662]$

Оба алгоритма нашли подходящую регрессию - $y(x) = 0.9x^2 - 3.1x + 10$ (константа не совпадает из за шума), но PDL потребовалось больше эпох, т.к. необходимо было увеличить доверительный регион пока мы не начнём делать шаги.

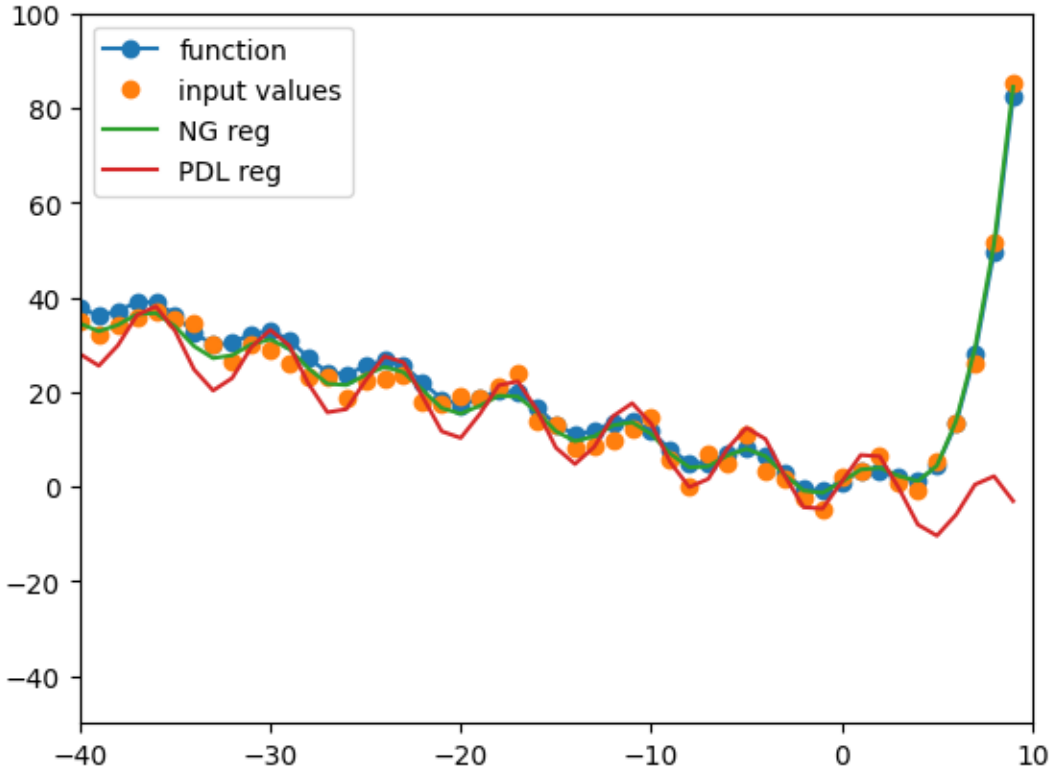
Функция: $y = e^{\frac{x}{2}} + 3sin(x) - x$

Регрессия: $f(w, x) = e^{w_1 \frac{x}{2}} + w_2 sin(x) + w_3 x$

Значения: целые точки от -40 до 10 с шагом 1

Шум: Каждое реальное значение функции отличается от изменённого не более чем на 10 (прибавление константы к функции)

Начальные значения: $\vec{0}.7$



Метод Gauss-Newton отработал 5 эпох со значениями $[0.50162497, 3.40040692, -0.92491769]$

Метод Powell Dog Leg отработал 300 эпох (лимит по эпохам) со значениями $[0.02936598, 7.68060405, -0.$

Из-за экспоненциального роста функции метод PDL нашёл регрессию с гораздо худшей точностью т.к. модель необходимо часто увеличивать и перестраивать

3 BFGS метод

Зпишем снова квадратичную модель для функции $m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p$ если g и B - градиент и гессиан соответственно, то получим метод Ньютона. Т.к. точное вычисление гессиана является дорогой операцией попробуем подобрать итеративно такую матрицу B_k, которая будет хорошо приблежать гессиан. Расммотрим квадратичную модель: $m_k(p) = f_k + \nabla f_k^T p + \frac{1}{2} p^T B_k p$.

Аналогично алгоритмам выше: $p_k = -B_k^{-1} \nabla f_k$. $x_{k+1} = x_k + \alpha_k p_k$

Построим B_k так, чтобы градиент модели и функции совпадал на предыдушем шаге (на текущем шаге он совпадает по определению).

Т.е. : $\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_k p_k = \nabla f_k$. Следовательно $B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k$. Обозначим $s_k = x_{k+1} - x_k = \alpha_k p_k$ и $y_k = \nabla f_{k+1} - \nabla f_k$.

Необходимо решить: $B_{k+1} s_k = y_k$,при этом мы ищем α в соответствие с условиями Вольфе, следовательно $s_k^T B_k s_k > 0, s_k^T y_k > 0$.

Его решение: $B_{k+1} = argmin \|B - B_k\|$, где $B = B^T$ и $B s_k = y_k$.

Обозначим $H_k = B_k^{-1}$,$\rho_k = \frac{1}{y_k^T s_k}$.

В новых обозначениях: $H_{k+1} = argmin \|H - H_k\| = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$, где $H = H^T$ и $H y_k = s_k$.

Направление будет: $p_k = -H_k \nabla f_k$. Ищем α в соотв. с условием Вольфе.

3.1 L-BFGS

Модификацией BFGS являеться алгоритм L-BFGS позволяющий в явном виде не хранить матрицу H что позволяет работать с функциями больших размерностей.

Храним m последних изменений для приближённого гессиана и применяем их.

Обозначим $V_k = I - \rho_k y_k s_k^T$. Тогда вместо $H_{k+1} = V_k^T H_K V_k + \rho s_k s_k^T$, используем m изменений:

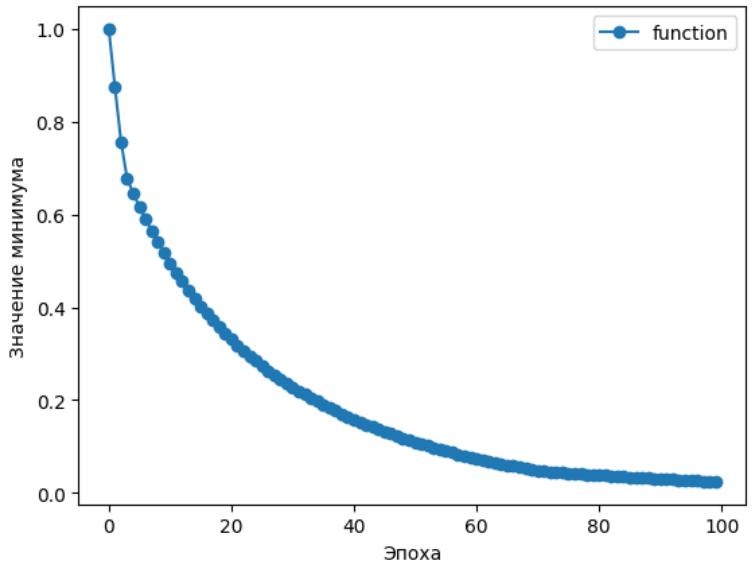
$H_k = (V_{k-1}^T ... V_{k-m}^T) H_k^0 (V_{k-m} ... V_{k-1}) + \rho_{k-m} (V_{k-1}^T ... V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} ... V_{k-1}) + ... + \rho_{k-1} s_{k-1} s_{k-1}^T$,
где H_k^0 начальное приближение гессиана на каждой итерации: $H_k^0 = \gamma_k I$, где $\gamma_k = \frac{s_{k-1}^t y_{k-1}}{y_{k-1}^t y_{k-1}}$.

Это позволяет уменьшить затраты по памяти с n^2 до nm.

3.2 Примеры работы методов

Функция: $f(x, y) = (1 - x)^2 + 10(y - x^2)^2$

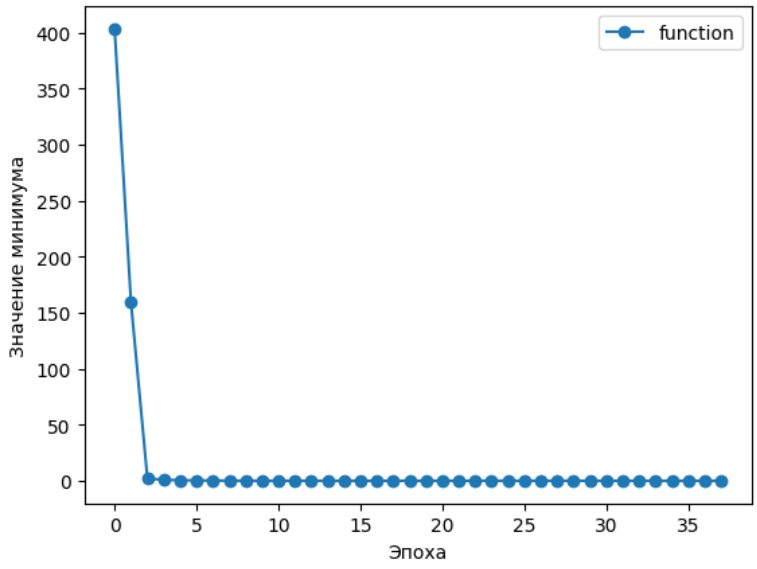
Минимум: в точке (1,1) равный 0



Метод BFGS отработал 100 эпох (лимит по эпохам) с координатами минимума [0.85452311, 0.7139605] и минимумом равным 0.023803904445279057

Функция: $f(x, y, z, t) = (x - y)^2 + sin(2z + t)^2 + t^2 + (e^{y+2z})^2$

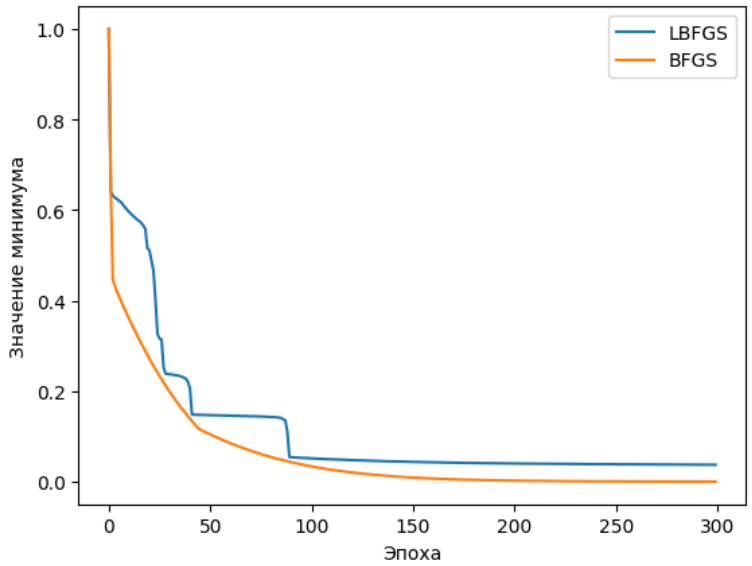
Минимум: в точке (1,1,,0) равный 0



Метод BFGS отработал 38 эпох (лимит по эпохам) с координатами минимума [-5.92617333, -5.92620277, -26.58937648, -12.79468824] и минимумом равным 8.868156478661523e-10 Так как мы используем численное дифференцирование то точность всех методов будет несколько хуже и они будут медленнее сходиться к ответу, однако это компенсируется скоростью BFGS т.к. нам не нужно вычислять гессиан.

Функция: $f(x, y) = (1 - x)^2 + 20(y - x^2)^2$

Минимум: в точке (1,1) равный 0

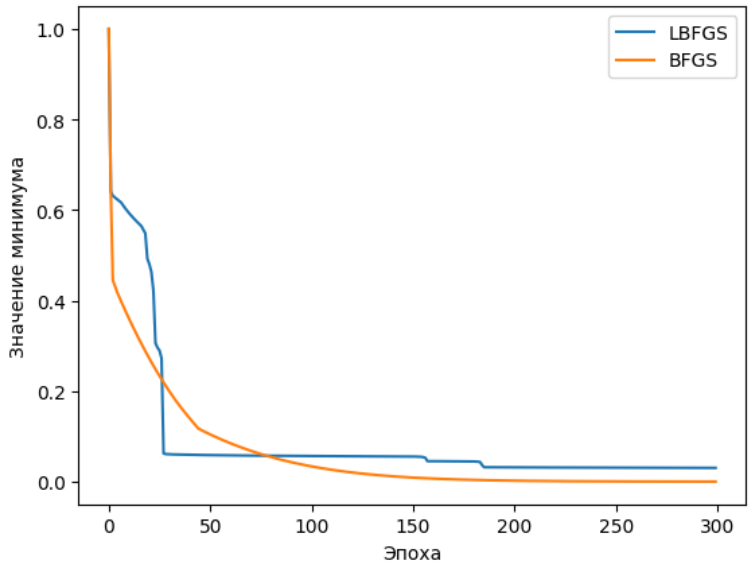


Метод BFGS отработал 300 эпох (лимит по эпохам) с координатами минимума [0.99020038, 0.9801906] и минимумом равным 9.79076618030884e-05

Метод LBFGS с m=3 отработал 300 эпох (лимит по эпохам) с координатами минимума [0.80706889, 0.65135542] и минимумом равным 0.037222414039960676

Функция: $f(x, y) = (1 - x)^2 + 20(y - x^2)^2$

Минимум: в точке (1,1) равный 0



Метод BFGS отработал 300 эпох (лимит по эпохам) с координатами минимума [0.99020038 0.9801906] и минимумом равным 9.79076618030884e-05

Метод LBFGS с m=30 отработал 300 эпох (лимит по эпохам) с координатами минимума [0.8277842 0.67949133] и минимумом равным 0.030316166491942714

По графикам можно предположить что методу BFGS в среднем требуется меньше ите-

раций и нахождение минимума является более гладким, в свою очередь LBFGS выполняет итерации быстрее и иногда получает существенную убавку к минимуму т.к. одно из нерассмотренных преобразований на очередном шаге "замедляло"поиск минимума.