МО, ЛР2

1 Реализованные методы, Градиентный спуск (присутствует с первой работы)

compute gradient - вычисляет градиент предоставленной функции в точке. Результат работы - вектор.

solve - решает задачу оптимизации по переданной функции и заданной точке. Гиперпараметры указываются до вызова данного метода, либо вручную, либо с помощью оптимизационного метода на основе optuna. Вектор-направление, в котором происходит шаг оптимизации, вычисляется либо по градиентному спуску, лиюо с помощью метода Ньютона. Результат работы - словарь с полями Результат, Значение Функции, Количество Итераций, Количество Вычислений Градиента, Количество Вычисленйи Функции.

plot descent - создает график на основе точек сохраненных во время решения оптимизационной задачи (хранятся в history array). Результат - таблица созданная pyplot на основе массива точек, где последняя точка - результат решения оптимизационной задачи.

armijo - ищет оптимальный шаг для очередной итерации поиска минимума на основе armijo method, используемое условие остановки $f(x + ad) \leqslant f(x) + c * a * \Delta f(x) * d$, максимум итераций - 20. Результат - размер шага.

exp decay - ищет оптимальный шаг на основе экспоненциального увядания. Результат - размер шага.

dec time - вычисляет шаг, уменьшая его каждую итерацию. Результат - размер шага.

golden ratio - метод одномерного поиска шага на основе золотого сечения. (np.sqrt(5) - 1) / 2 - константа золотого сечения. Изначальный интервал поиска длины шага - [0, 1]. Точность сужения интервала - 1e-5. Результат - длина шага.

dichotomy - метод одномерного поиска длины шага на основе дихотомии. Изначальный интервал поиска длины шага - [0, 1]. Точность сужения интервала - 1e-5. Результат - длина шага.

2 Реализованные методы, Ньютоновские методы

compute hessian - вычисляет Гессиан предоставленной функции в точке. За счет более дорогих вычислений, а именно Гессиана, достигается более быстрая сходимость - квадратичная. Результат - таблица Гессиана (ndarray).

solve newton system - решает систему Ньютона из метода Ньютона. $H(x_k)d_k = -\Delta f(x_k)$. Результат - решение системы (ndarray)

[newton-cg, bfgs, l-bfgs-b] - ньютоновский и 2 квазиньютоновских метода на основе scipy.optimize. Квазиньютоновские методы отличаются тем, что не вычисляют Гессиан явно, вместо этого они вычисляют аппроксимацию Гессиана на основе уже проведенных ранее шагов. BFGS — один из наиболее широко применяемых квазиньютоновских методов.

Рассмотрим задачу оптимизации функционала:

$$\arg\min_x f(x).$$

Методы второго порядка решают эту задачу итеративно, используя разложение функции в квадратичный полином (аппроксимацию второй степени):

$$f(x_k + p) = f(x_k) + \nabla f^T(x_k)p + \frac{1}{2}p^T H(x_k)p,$$

где H — гессиан (матрица вторых производных) функции f в точке x.

Поскольку вычисление гессиана может быть сложным, BFGS-алгоритм использует его приближенное значение B_k, а затем находит минимум квадратичной задачи:

$$p_k = -B_k^{-1}\nabla f(x_k).$$

После этого обычно выполняется линейный поиск вдоль направления p_k.

Инициализация и обновление матрицы B_k Начальное приближение гессиана B_0 может быть любой невырожденной и хорошо обусловленной матрицей (часто берут единичную матрицу I).

На каждой итерации матрица B_k обновляется по формуле:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

где: - $s_k = x_{k+1} - x_k$ — шаг алгоритма, - $y_k = \nabla f_{k+1} - \nabla f_k$ — изменение градиента.

Работа с обратной матрицей $C_k = B_k^{-1}$ Поскольку вычисление обратной матрицы требует больших вычислений, BFGS напрямую обновляет C_k:

$$C_{k+1} = (I - \rho_k s_k y_k^T) C_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T,$$

где $\rho_k = \frac{1}{y_k^T s_k}$.

Это позволяет избежать трудоемких операций обращения матрицы на каждом шаге.

3 Результаты. Запуски и вычисления

Далее будет приведен пример работы оптимизатора гиперпараметров для конкретной задачи.
В двнном случае $3(x-3)^2+y^2$ Далее будем упускать данный момент работы кода, предоставляя лишь варианты с лучшими результатами на уже оптимизированных гиперпараметрах.

Tuning hyperparameters for Gradient Descent...

[I 2025-06-12 19:09:16,991] A new study created in memory with name: no-name-3e4963d2-6b31-40d5-9873-a7cd8fe539eb

[I 2025-06-12 19:09:20,200] Trial 0 finished with value: 3.067337180655583e-13 and parameters: 'learning rate': 0.00897393473622532, 'lr method': 'armijo', 'lr method const': 0.00044610224397846415. Best is trial 0 with value: 3.067337180655583e-13.

[I 2025-06-12 19:09:20,239] Trial 1 finished with value: 2.321173434083764e-17 and parameters: 'learning rate': 0.04518475204357972, 'lr method': 'dichotomy'. Best is trial 1 with value: 2.321173434083764e-17.

[I 2025-06-12 19:09:20,278] Trial 2 finished with value: 2.321173434083764e-17 and parameters: 'learning rate': 0.0036586450905467197, 'lr method': 'dichotomy'. Best is trial 1 with value: 2.321173434083764e-17.

...

[I 2025-06-12 19:10:06,186] Trial 97 finished with value: 2.3162182759827013e-17 and parameters: 'learning rate': 0.5534748601316051, 'lr method': 'golden ratio'. Best is trial 5 with value: 2.3162182759827013e-17.

[I 2025-06-12 19:10:06,243] Trial 98 finished with value: 2.3162182759827013e-17 and parameters: 'learning rate': 0.04727707680793929, 'lr method': 'golden ratio'. Best is trial 5 with value: 2.3162182759827013e-17.

[I 2025-06-12 19:10:06,298] Trial 99 finished with value: 2.3162182759827013e-17 and parameters: 'learning rate': 9.050704041710368e-05, 'lr method': 'golden ratio'. Best is trial 5 with value: 2.3162182759827013e-17.

———————————————-

Best hyperparameters for gradient:
Learning rate: 0.2660847164945598
LR method: golden ratio
Best value: 2.3162182759827013e-17

———————————————-

Tuning hyperparameters for Newton's Method...

[I 2025-06-12 19:10:06,624] Trial 0 finished with value: 6.083548274804782e-15 and parameters: 'learning rate': 0.19037921513443637, 'lr method': 'exp decay', 'lr method const': 2.0310668080105438e-05. Best is trial 0 with value: 6.083548274804782e-15.

[I 2025-06-12 19:10:10,015] Trial 1 finished with value: 123.66978557669759 and parameters: 'learning rate': 1.2014807586641177e-05, 'lr method': 'dec time', 'lr method const': 0.031133647560298936. Best is trial 0 with value: 6.083548274804782e-15.

[I 2025-06-12 19:10:10,027] Trial 2 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.11423594712894017, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:10:15,913] Trial 3 finished with value: 35.055960375857225 and parameters: 'learning rate': 0.0006314684963273133, 'lr method': 'armijo', 'lr method const': 0.5993958999678658. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:10:16,154] Trial 4 finished with value: 3.0704529786150986e-15 and parameters: 'learning rate': 0.29659734653309644, 'lr method': 'exp decay', 'lr method const': 0.004019007708897. Best is trial 2 with value: 4.678713636805452e-20.

...

[I 2025-06-12 19:11:20,269] Trial 92 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.007906641983110918, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,285] Trial 93 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.005378404133135358, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,776] Trial 94 finished with value: 1.6876286886600635e-14 and parameters: 'learning rate': 0.1066319483368602, 'lr method': 'fixed'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,791] Trial 95 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.009878553078243778, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,808] Trial 96 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.046474643870840236, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,833] Trial 97 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.018007005332610168, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

[I 2025-06-12 19:11:20,855] Trial 98 finished with value: 4.678713636805452e-20 and parameters: 'learning rate': 0.0036785238377348662, 'lr method': 'dichotomy'. Best is trial 2 with value: 4.678713636805452e-20.

———————————————-

Best hyperparameters for newton:

Learning rate: 0.11423594712894017

LR method: dichotomy

Best value: 4.678713636805452e-20

———————————————-

Результаты для $3(x-3)^2 + y^2$:

Gradient Descent требует значительно больше вычислений функции (так как сходится медленнее).

Newton's Method показывает наилучшую точность за наименьшее число итераций, но требует больше вычислений функции на каждой итерации.

Newton-CG использует больше расчетов градиента, чем итераций. Судя по всему

BFGS и L-BFGS-B демонстрируют хороший баланс между точностью и вычислительными затратами.

Результаты для $(x^2 + y - 11)^2 + (x + y^2 - 7)^2$

Gradient Descent сходится к решению, но требует больше итераций и вычислений.

Newton's Method дал неверный результат (0, 0) — возможно, из-за неудачного начального приближения или особенностей гессиана.

Newton-CG показывает хорошую точность, но требует больше расчетов градиента (Либо криво написан(((.

BFGS демонстрирует наилучшую точность ($f_{value} = 1.37e - 22$) при умеренных вычислительных затратах.

L-BFGS-B немного менее точен, чем BFGS, но работает быстрее.

Результаты для $(1-x)^2 + 100 * (y - x^2)^2$

Gradient Descent - Очень медленная сходимость (1000 итераций). Низкая точность ($f_{value} 3 * 10^{-7}$). Огромные вычислительные затраты (36 000 расчетов функции).

Newton's Method - Самый точный ($f_{value} 4 * 10^{-28}$). Быстрая сходимость (27 итераций). Но требует больше вычислений градиента и функции на каждой итерации.

Newton-CG - Хорошая точность (10^{-13}), но медленнее BFGS.

BFGS - Оптимальный баланс между точностью ($5 * 10^{-22}$) и скоростью. Эффективнее Newton-CG (меньше grad calcs при схожей точности).

L-BFGS-B - Быстрее BFGS (36 итераций против 57). Чуть менее точен ($5 * 10^{-17}$), но требует меньше вычислений.

4 Выводы

ф

Лучшая точность → Метод Ньютона, но он может быть вычислительно затратным.

Лучший компромисс → BFGS (высокая точность + умеренные вычисления).

Быстрая оптимизация → L-BFGS-B (хорошая точность при малых затратах).

Градиентный спуск - стабильный метод, но медленный и не всегда точный.