Домашняя работа №2
по дисциплине "Методы оптимизации"

Винницкая Дина Сергеевна

Группа: Б9122-02.03.01сцт

Задание

Введение

Метод градиентного спуска — это численный метод оптимизации, используемый для нахождения минимума функции. Этот метод особенно полезен для многомерных функций, где аналитическое нахождение минимума сложно или невозможно. Основная идея градиентного спуска заключается в движении от начальной точки в направлении отрицательного градиента функции, чтобы достичь точки минимума. Градиент указывает направление наибольшего возрастания функции, поэтому, двигаясь в противоположную сторону, мы приближаемся к точке минимума.

Градиентный спуск можно применять для минимизации квадратичной функции вида:

$$f(x) = \frac{1}{2}x^T Hx + c \cdot x,$$

где H — положительно определённая матрица, c — вектор, а x — вектор переменных.

Основные понятия

- **Градиент**: Вектор, который указывает направление наибольшего возрастания функции. Для функции $f(x)$, градиент определяется как $\nabla f(x)$.
- **Шаг градиентного спуска**: Параметр, определяющий длину шага, который мы делаем в направлении отрицательного градиента. Если шаг слишком мал, сходимость может быть медленной, если слишком велик — может возникнуть нестабильность.
- **Точность (точность остановки)**: Критерий, при достижении которого градиентный спуск прекращает работу, если изменение значений функции на очередном шаге меньше определенного порога.
- Градиентный спуск — это численный метод оптимизации, используемый для нахождения минимума функции.

  Этот метод особенно эффективен в многомерных задачах, где нахождение аналитического решения может быть затруднительным или невозможным.

  Основная идея градиентного спуска заключается в движении от начальной точки в направлении, противоположном градиенту функции, чтобы постепенно приближаться к точке минимума.

  Градиент функции $f(x)$ обозначается как $\nabla f(x)$ и представляет собой вектор, который указывает направление наибольшего возрастания функции.

  Путем движения в направлении, противоположном градиенту, мы уменьшаем значение функции.

  Это можно выразить следующей формулой обновления:

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

  где x_k — текущее значение переменной, α — коэффициент шага (или скорость обучения), который определяет, насколько сильно мы корректируем значение x на каждом шаге, и $\nabla f(x_k)$ — градиент функции в точке x_k. Основные компоненты градиентного спуска

  - **Градиент** $\nabla f(x)$: Вектор частных производных функции, который указывает направление наибольшего возрастания функции. Двигаясь в противоположном направлении, мы приближаемся к минимуму.

– **Шаг (коэффициент шага)** α: Параметр, который определяет, насколько сильно изменяется x на каждом шаге. Слишком малый шаг делает алгоритм медленным, а слишком большой может привести к нестабильности и неудаче в нахождении минимума.

– **Точность (или критерий остановки)**: Алгоритм может завершиться, когда изменение значений функции или нормы градиента становится достаточно малым, указывая на то, что достигнута точка, близкая к минимуму.

Применение
Градиентный спуск широко используется для минимизации функций в различных областях, таких как машинное обучение, нейронные сети, регрессия и другие задачи оптимизации.
Метод может быть адаптирован для работы с большими наборами данных и высокой размерностью, что делает его особенно популярным в задачах обучения моделей.
Таким образом, градиентный спуск — это универсальный и гибкий метод, который позволяет эффективно находить минимумы сложных функций.

Постановка задачи

Необходимо найти минимум функции в пространстве $\mathbb{R}^n$ методом градиентного спуска, применяя его к квадратичной функции с положительно определенной матрицей H размером 3×3 и вектором c. Необходимо:

1. Сгенерировать случайную положительно определённую матрицу H и вектор c.
2. Реализовать метод градиентного спуска для поиска минимума функции.
3. Исследовать влияние различных значений шага на количество итераций и результат сходимости.

Код реализации

Реализованный код на Python приведен ниже:

```python
import numpy as np

def generate_positive_definite_matrix(size):
    M = np.random.rand(size, size)
    H = np.dot(M, M.T)
    return H

dimension = 3
H = generate_positive_definite_matrix(dimension)
c = np.random.rand(dimension)
x_start = np.zeros(dimension)

initial_step = 0.1
precision = 1e-6
max_steps = 1000
gradient_limit = 1e10

def gradient_descent(H, c, x_start,
initial_step, precision, max_steps, gradient_limit):
    x = x_start
    step_size = initial_step
    iterations = 0
    path = [x]

    for _ in range(max_steps):
        grad = H @ x + c
        grad_norm = np.linalg.norm(grad)
        if grad_norm > gradient_limit:
            step_size *= 0.5

        x_new = x - step_size * grad
        path.append(x_new)

        if np.linalg.norm(x_new - x) < precision:
            break

        x = x_new
        iterations += 1

    return x, iterations, path
```

Пояснение

- `generate_positive_definite_matrix` генерирует случайную положительно определённую матрицу H, перемножая случайную матрицу на её транспонированную версию, чтобы получить симметричную положительно определённую матрицу.
- В коде инициализируется начальная точка x_start, шаг градиентного спуска, требуемая точность и максимальное количество итераций.
- Функция `gradient_descent` реализует алгоритм градиентного спуска. На каждом шаге вычисляется градиент, и обновляется значение x в направлении отрицательного градиента.
- При слишком большом значении нормы градиента происходит уменьшение шага вдвое, чтобы избежать численных ошибок и переполнения.

Результаты и выводы

Программа выводит сгенерированную матрицу H и вектор c, а затем находит минимум функции методом градиентного спуска. В результате выполнения наблюдается, как различные значения шага (α) влияют на количество итераций и точку минимума:

- Маленькие значения шага приводят к большему количеству итераций, но обеспечивают стабильную сходимость.
- При оптимальном значении шага сходимость достигается быстро и стабильно.
- Слишком большие значения шага вызывают нестабильность и численные ошибки, требующие корректировки.