МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра математического анализа

Численные методы решения экстремальных задач для выпуклых функций

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы

направление 02.03.01 — Математика и компьютерные науки

механико-математического факультета

Челибанова Максима Романовича

Научный руководитель
доцент. к.ф. -м. н. М.А.Осипцев

Зав. кафедрой
и.о. зав. кафедрой, д.ф.-м.н. П.А.Терехин

Саратов 2024

Введение. В области математики и оптимизации экстремумов функций важную роль играют численные, а также градиентные методы. Они позволяют находить экстремумы выпуклых функций, которые в основном используются в задачах оптимизации и машинном обучении. Численные методы являются эффективными алгоритмами, способными находить экстремумы функций быстро и точно. В целях этой дипломной работы мы будем исследовать численные и градиентные методы их применение в поиске экстремумов выпуклых функций. Мы рассмотрим основные алгоритмы градиентного спуска и стохастического градиентного спуска, а также проведем анализ их преимуществ и недостатков. В дополнение к этому, мы также рассмотрим более современные методы градиентного оптимизации, такие как методы сопряженного градиента, метод Адама и Ньютона, которые могут использоваться в повышении скорости поиска экстремумов.

Целью данной работы является изучение темы градиентные методы поиска экстремумов выпуклых функций:

-Изучить и узнать все градиентные методы поиска экстремумов.

-Научиться применять данные методы для выпуклых функций.

-Написать и изучить программу на языке python.

Основное содержание работы. Изначально нам надо ввести понятие выпуклости функции, с чем нам предстоит работать, а также экстремума. В дальнейшем рассмотрим различные численные методы для нахождения экстремума выпуклой функции.

Выпуклая функция - это функция, для которой выполнено условие, что отрезок, соединяющий любые две точки на графике функции, находится полностью выше самого графика функции. Другими словами, функция выпуклая, если ее касательные лежат выше самой функции. Например, функция $f(x) = x^2$ является выпуклой, тогда как функция $f(x) = x^3$ не является. Выпуклые функции широко используются в экономике, оптимизации, дифференциальной геометрии и других областях математики.

Определение 1.1. Непрерывная функция $y = f(x)$ называется выпуклой вверх на отрезке $[a, b]$, если для любых точек x_1 и x_2 отрезка $[a, b]$ выполняется неравенство: $f(\frac{x_1+x_2}{2}) \geq \frac{f(x_1)+f(x_2)}{2}$.

Дадим геометрическую интерпретацию понятия выпуклости (рис.1). Пусть $M1, M2, M0$ — точки графика функции $y = f(x)$, абсциссы которых равны соответственно $x_1, x_2, x_0 = \frac{x_1+x_2}{2}$. Тогда $\frac{f(x_1)+f(x_2)}{2}$ есть ордината точки K — середины отрезка M_1, M_2, а $f(\frac{x_1+x_2}{2}) = f(x_0)$— ордината точки M_0 графика с абсциссой, равной абсциссе точки K.

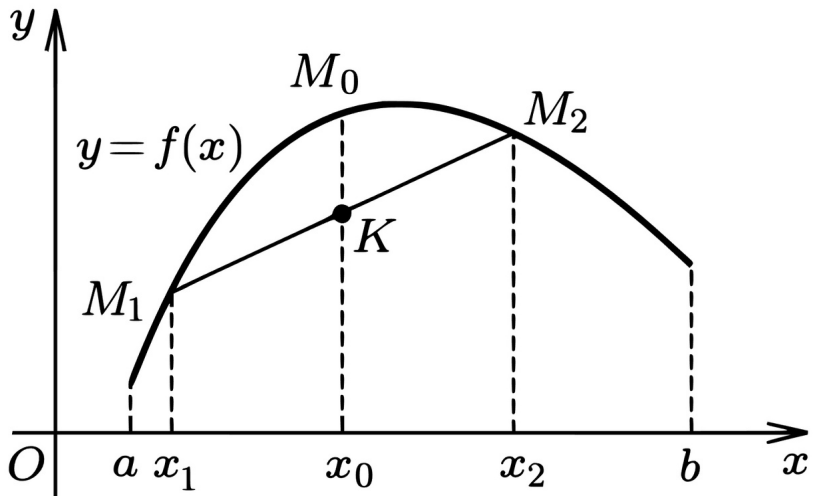

Рис. 1

Экстремумами (максимумами и минимумами) функции называются значения функции в точках максимума и минимума.

Пусть дана функция $f : M \subset \mathbb{R} \to \mathbb{R}$, $x_0 \in M^0$ - внутренняя точка области определения f.

Тогда x_0 называется точкой локального максимума функции f, если существует проколотая окрестность $U(x_0)$ такая, что
$\forall x \in U(x_0)\ f(x) \leq f(x_0)$;

x_0 называется точкой локального минимума функции f, если существует проколотая окрестность $U(x_0)$ такая, что
$\forall x \in U(x_0)\ f(x) \geq f(x_0)$;

x_0 называется точкой глобального (абсолютного) максимума, если
$\forall x \in M\ f(x) \leq f(x_0)$;

x_0 называется точкой глобального (абсолютного) минимума, если
$\forall x \in M\ f(x) \geq f(x_0)$;

Если неравенства выше строгие, то x_0 называется точкой строгого локального или глобального максимума или минимума соответственно.

Значение функции $f(x_0)$ называют соответственно (строгим) локальным или глобальным максимумом или минимумом. Точки, являющиеся точками (локального) максимума или минимума, называются точками (локального) экстремума.

Функция $f(X)$, заданная на выпуклом множестве, содержащем граничные точки, называется выпуклой, если для любых двух точек X_1 и X_2 из этого множества и любого λ такого, что $0 \leq \lambda \leq 1$, справедливо неравенство: $f[\lambda x_2 + (1-\lambda)x_1] \leq \lambda f(x_2) + (1-\lambda)f(x_1)$.

Функция, для которой знак неравенства изменен на обратный, называется вогнутой. Если $f(X)$ - выпуклая функция, то - $a(X)$ - вогнутая функция, и наоборот. Функции, для которых справедливы строгие неравенства вида (1.2), называются строго выпуклыми или строго вогнутыми.

Гиперповерхность $z = f(X)$ является выпуклой, если отрезок, соединяющий любые две ее точки, лежит на самой поверхности или выше ее. Выпуклая функция одной переменной изображена на рис. 2. Геометрическое построение, представленное на этом рисунке, иллюстрирует неравенство (1.2).

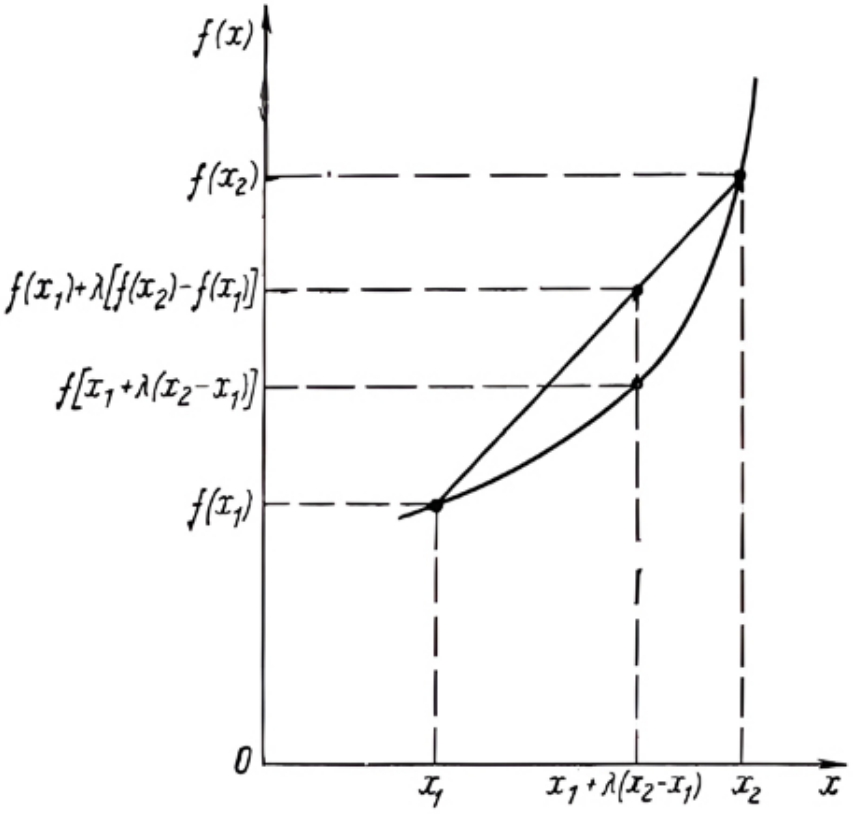

Рис. 2

Численные методы выпуклой оптимизации - это класс методов, предназначенных для решения задач оптимизации, в которых целевая функция (функция, которую нужно минимизировать или максимизировать) является выпуклой. Выпуклая оптимизация имеет широкий спектр приложений, включая машинное обучение, управление системами, финансовое моделирование и другие области.

Градиент функции многих переменных $F(z)$ - вектор, компонентами которого являются частные производные этой функции по координатам: $grad_z[F(z)] = (\frac{\partial F}{\partial z_1}, \frac{\partial F}{\partial z_2}, ..., \frac{\partial F}{\partial z_n})^T$

В градиентных методах предполагается пошаговое движение в направлении антиградиента функции $F(z)$. Координаты z на $k+1$-ом шаге вычисляются по формуле: $z^{k+1} = z^k - h^k grad_z[F(z)]$

Субградиентные методы - это алгоритмы, которые используются для решения задач выпуклой оптимизации, когда функция, которую мы пытаемся оптимизировать, не обязательно дифференцируема в каждой точке.

Градиентный спуск - это метод оптимизации функций, который применяется для поиска локального минимума или максимума функции. Он часто используется в машинном обучении для настройки параметров модели.

Метод условного градиента (также известный как алгоритм Франк-Вульфа) представляет собой итеративный алгоритм, используемый в оптимизации для решения задач выпуклой оптимизации на множествах с простой структурой. Основная идея метода заключается в том, чтобы аппроксимировать сложную оптимизационную задачу, рассмотрим задачу с ограничениями: $\min f(x)$, где $x \in Q$.

Теорема 2.7.1. Пусть - f выпуклая функция, градиент которой на Q удовлетворяет условию Липшица с константой L по отношению к некоторой норме $||\cdot||$: для всех $x, y \in Q$, выполняется

$$||\nabla f(y) - \nabla f(x)|| \leq L||y - x||,$$

$R = sup||x - y||, \gamma_k = \frac{2}{k+1}$ для $k \geq 1$. Тогда для любого $k \geq 2$ выполняется

$$f(x_k) - f(x^*) \leq \frac{2LR^2}{k+2}$$

Основные результаты для выпуклых функций, хочу рассмотреть и показать теорему для ускоренного мета алгоритма.

Рассмотрим следующую задачу (x^* - решение задачи):
$\min\{F(x) := f(x) + g(x)\}$, где f и g - выпуклые функции.

Теорема 2.8.1 Пусть y_k - выход ускоренного мета-алгоритма (x_0, f, g, p, H, k) после k итераций при $p \geq 1$ и $H \geq (p+1)L_{p,f}$.

Тогда:

$F(y_k) - F(x^*) \leq \frac{c_p H R^{p+1}}{k^{\frac{3p+1}{2}}}$

где $c_p = 2^{p-1}(p+1)^{\frac{3p+1}{2}}/p!$, $R = ||x_0 - x^*||$.

Более того, при $p \geq 2$ для достижения точности ε:

$F(y_k) - F(x_*) \leq \varepsilon$

На каждой итерации придётся решать (процедурной типа бинарного поиска) для подбора пары (λ_{k+1}, y_{k+1}) не более чем $O(ln(\varepsilon^{-1}))$ раз.

Опишем рестартованный ускоренный мета-алгоритм: Введём: r-равномерно выпуклая функция $F = f + g : \mathbb{R}^n \to \mathbb{R}$ с const σ_r.

Обозначим: $z_0 = x_0$, $R_k = R_0 \cdot 2^{-k}$.

$$N_k = \max\left\{\left(\frac{r c_p H 2^r}{\sigma_r}\right)^{\frac{2}{3p+1}}, 1\right\}$$

Определим: $z_{k+1} := y_{N_k}$, где y_{N_k} - вывод.

Теорема 2.8.2. Пусть y_k - вывод алгоритма из верхнего уравнения, после k итераций. Тогда если $H \geq (p+1)L_{p,f}$, $\sigma_r > 0$, то общее число вычислений (2.24) для достижения

$$F(y_k) - F(x^*) \leq \varepsilon$$

будет

$$N = \tilde{O}\left(\left(\frac{H R^{p+1-r}}{\sigma_r}\right)^{\frac{2}{3p+1}}\right), \tag{1}$$

где $\tilde{O}(\quad)$ - означает тоже самое, что $O(\quad)$ с точностью до множителя $\ln(\varepsilon^{-1})$.

Несмотря на то, что градиентный спуск имеет преимущество при оптимизации выпуклых функций, он может страдать от проблем, таких как локальные минимумы и седловые точки, которые могут замедлить сходимость алгоритма.

Из соотношения: $z^{k+1} = z^k - h^k grad_z[F(z)]$

следует, что

$\frac{z^{k+1}-z^k}{h^k} = grad_z[F(z)]$. При $h \to 0$ получается: $\frac{dz}{dh} = grad_z(F(z))$

Последнее уравнение описывает градиентный метод в чистом виде, считается, что текущий шаг бесконечно мал и в каждой точке направление движения совпадает с направлением антиградиента (рис.3). Практическая реализация метода градиентного спуска требует очень большого числа шагов.

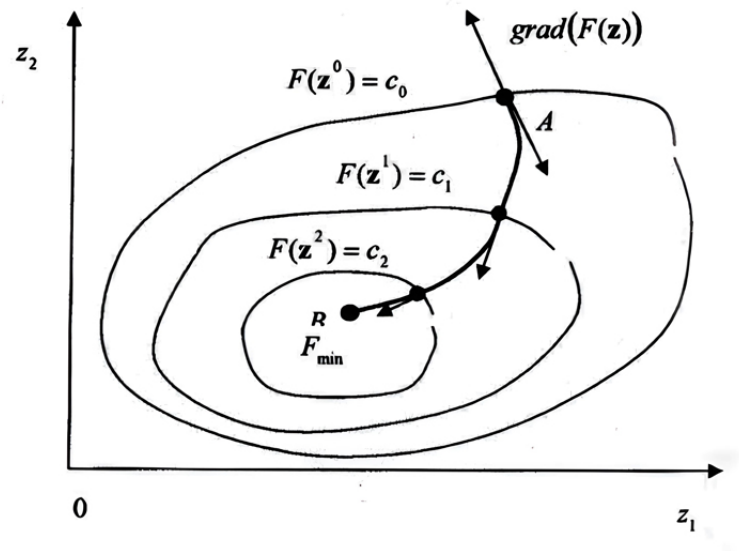

Рис.3

Простой градиентный метод с постоянным шагом, означает что: $h^{k+1} = h^k = const$

Для достижения заданной точности в конце поиска шаг должен быть с самого начала небольшим, что ведет к большим вычислительным затратам кроме того, этот метод неэффективен для функций, имеющих «овражную» структуру.

Градиентный метод с переменным шагом, говорит о том что на $K+1$-ом шаге вычисляется $F(z^{k+1})$ с шагом h^k. Если $F(z^{k+1}) \leq F(z^k)$, то шаг увеличивается, например, $h^{k+1} = 2h^k$, вычисляется целевая функция и шаг

принимается равным h^k или $2h^k$ в зависимости от того, какое значение ш ага обеспечивает наименьшее значение $F(z^{k+1})$. Если $F(z^{k+1}) > F(z^k)$, то шаг уменьшается, например, $h^{k+1} = \frac{h^k}{2}$. Если это не приводит к уменьшению целевой функции, шаг уменьшается до тех пор, пока не будет найдено меньшее значение $F(z^{k+1})$, чем на предыдущем шаге. По мере приближения к минимуму шаг будет уменьшаться, а точность увеличиваться. Недостатком такой схемы расчета является замедление процесса сходимости в окрестности минимума.

Метод наискорейшего спуска - это величина текущего шага, которая оптимизируется из условия минимума целевой функции (рис.4) в направлении антиградиента, определенном для данной точки:

$h^k = arg_h min F(z^{k+1}) = arg_h min F(z^k - h^k grad(F(z^k)))$

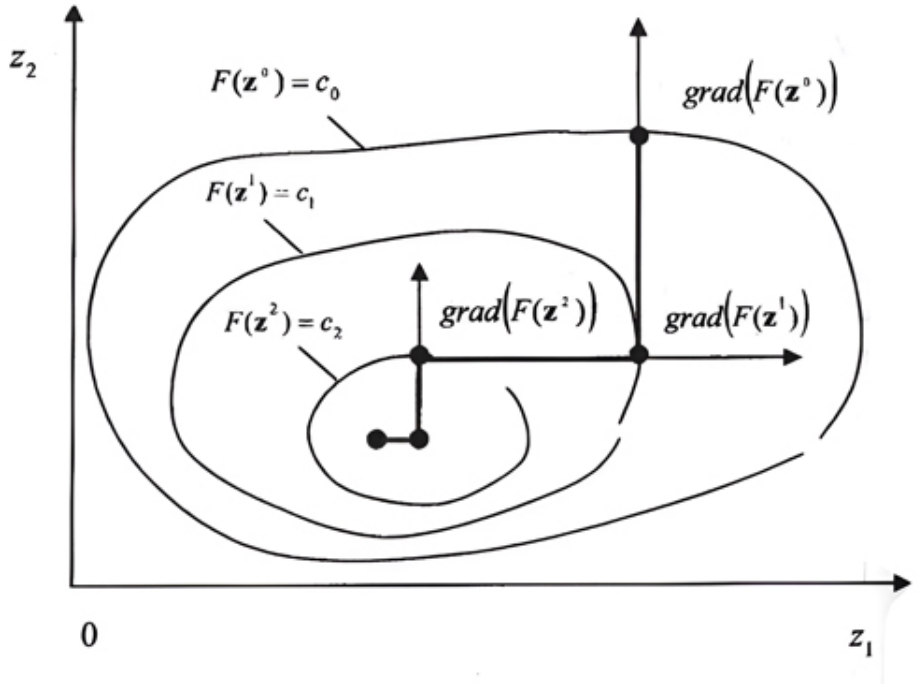

Рис.4

Нам нужно создать программу на языке программирования python, в которой будут использоваться градиентные методы, а именно для нахождения экстремума выпуклой функции.

```
import numpy as np

def gradient_descent(f, grad_f, x_0, alpha=0.1, eps=1e-5,
    max_iter=1000):
    x_prev = x_0
    grad_prev = grad_f(x_prev)
    for i in range(max_iter):
        x_next = x_prev - alpha * grad_prev
        grad_next = grad_f(x_next)
         if np.linalg.norm(grad_next) <= eps:
             break
         x_prev, grad_prev = x_next, grad_next
    return x_next

def f(x):
    return x[0]**2 + 2*x[1]**2 + np.exp(x[0]+x[1])

def grad_f(x):
    return np.array([2*x[0] + np.exp(x[0]+x[1]), 4*x[1]
    + np.exp(x[0]+x[1])])

# Пример использования:
x_0 = np.array([1., 1.])
x_opt = gradient_descent(f, grad_f, x_0)
print("Минимум функции:", f(x_opt), "в точке", x_opt)
```

Вывод:

```
Минимум функции: 0.7722682277380957 в точке [-0.312763]
```

Комментарии к программе: здесь, я буду обозначать, через x. строку, который буду давать комментарий(x - номер строки).

1. Здесь я портировал библиотеку numpy, NumPy это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

3-4. Здесь я завожу функцию, которую обзываю , внутри скобок завожу переменные, к которым мы будем обращаться внутри функции, часть из них будет вызвана вне функции, часть переменных мы ввели внутри функции.

5. Здесь заводим новую переменную, в который мы берем значение, за x_0.

6. Затем опять создаем новую переменную, градиента, который будет равен градиенту, от новой переменной.

7. Далее идет цикл, который будет проходить от 0 до 1000.

8. Где внутри самого цикла, берется следующее значение x, которое изменяется, от предыдущего x, минус альфа, умноженное на предыдущий градиент.

9. Здесь я присваиваю следующий градиент, от градиента, по x, который мы получили, на прошлой строке.

10-11. Здесь условие, в котором переменная np.linalg.norm - это функция, которая представляет математическую норму. По сути, нормой вектора является его длина. От переменной с прошлой строки. Если она будет меньше или равна переменной eps, которую мы заводили внутри функции, то останавливаем цикл из 7 строки.

12. Здесь мы присваиваем предыдущие переменные, за следующие.

13. Делаем возврат переменной в функцию, а именно x, который изменялся на протяжение цикла.

15-18. Здесь просто заводим, функции, возвращаем в них значения, к которым мы будем обращаться, как раз в большой функции 3 строки.

24. Заводим переменную, которая обратится к функции, точнее будет равна самой функции.

25. Выводим результат в консоль.

В программе я ввел сам градиентный метод, который пробегает по итерации, потом идет вызов самой функции и градиента, а в дальнейшем вывод минимума функции, и в какой точке он находится.

В данном примере мы имеем двумерную функцию $f(x)$ и ее градиент $grad_f(x)$. Используя градиентный метод оптимизации, мы ищем минимум $f(x)$ из начального приближения x_0. В данном примере мы задали свойственные параметры, такие как $alpha$ - скорость обучения и eps - заданная точность результата. Для уменьшения числа итераций мы ограничили максимальное число итераций до $maxiter$.

Заключение. В ходе данной работы мы повторили понятие выпуклых функций, экстремумов функции, экстремумы выпуклых функций, изучили градиентные методы, а именно "Градиентный спуск рассмотрели алгоритм Франк-Вульфа, ускоренный мета-алгоритм и другие. Написали программу на python и оптимизировали процесс.