



Homo homini lupus est.

ПМИ ФКН ВШЭ
Непрерывная оптимизация. Экзамен.
171 группа

Июнь 2020

Содержание

1	Теоретический минимум	3
1.1	Определение линейной/сублинейной/суперлинейной скорости сходимости. Пример последовательности для каждого класса. @mercivboku	3
1.2	Условия Армихо и Вульфа для неточной одномерной оптимизации. Процедура бэктрекинга. @mercivboku	3
1.3	Общее определение производной функции (через линейное отображение). Основные свойства производной (правила суммы, произведения и композиции). Определения второй производной, градиента и гессиана. @mercivboku	4
1.4	Классы липшицевых функций, гладких выпуклых и сильно выпуклых функций. Соответствующие глобальные верхние и нижние оценки на функцию. Примеры функций для каждого класса. @mercivboku	5
1.5	Выпуклые множества и функции. Операции, сохраняющие выпуклость. Простейшие свойства выпуклых функций (неравенство Йенсена, выпуклость надграфика, выпуклость множеств подуровней). Дифференциальные критерии выпуклости. @mercivboku	7
1.6	Определения субградиента и субдифференциала. Субдифференциальное исчисление (умножение на скаляр, композиция с аффинным преобразованием, теорема Моро–Рокафеллара, максимум конечного числа функций и теорема Данскина). @poly_nomial	9
1.7	Сопряженная функция Фенхеля. Неравенство Фенхеля–Юнга. Теорема Фенхеля–Моро. Связь сопряженной функции с субдифференциалом. @poly_nomial	12
1.8	Сопряженная норма. Неравенство Гельдера. Основные примеры. @poly_nomial	14
1.9	Проксимальный оператор. Связь с евклидовой проекцией. @bigbluebutterfly	14
1.10	Основные матричные разложения, их стоимость вычисления. @bigbluebutterfly	15
1.11	Формулировка теоремы Каруша–Куна–Таккера для общей задачи нелинейного программирования. @Bitchert	16
1.12	Каноническая форма задачи линейного программирования. Пример сведения общей задачи ЛП к задаче в канонической форме. @Bitchert	17
1.13	Двойственная задача Лагранжа и ее основные свойства. Пример построения двойственной задачи. @Bitchert	17
1.14	Общая схема какого-то метода оптимизации (итерация, выбор длины шага, критерий останова), например, градиентного спуска, проксимального градиентного метода, метода SAG, метода логарифмических барьеров и проч.	18
2	Билеты	19
2.1	Матрично-векторное дифференцирование в терминах дифференциалов. Примеры вычисления градиентов и гессианов для матричных функций.	19
2.2	Решение задачи одномерной оптимизации (метод золотого сечения, парабол, Брента). Условия для неточной одномерной оптимизации. @sofloud	19
2.3	Метод градиентного спуска. Стратегии выбора длины шага. Скорость сходимости метода для сильно выпуклых функций, примеры быстрой и медленной работы метода. @julasemavina	24
2.4	Метод Ньютона, его локальная и глобальная скорость сходимости. Модификации метода Ньютона для невыпуклых задач оптимизации. Примеры быстрой и медленной работы метода. @Bitchert	27
2.5	Метод сопряженных градиентов для решения системы линейных уравнений. Доказательство сходимости метода за конечное число итераций (в предположении, что сопряженные направления известны). Модификации метода для минимизации произвольных функций. @bigbluebutterfly	30
2.6	Безгессианный (неточный) метод Ньютона (HFN), его скорость сходимости. Способы оценивания произведения гессиана на вектор с помощью разностного дифференцирования. @kirili4ik	33
2.7	Квазиньютоновские методы. Схемы BFGS и L-BFGS. @isadrtinov	36
2.8	Стандартные классы выпуклых задач: линейное программирование (LP), коническое квадратичное программирование (CQP/SOCP), полуопределенное программирование (SDP). Примеры задач для каждого из классов. Сводимость. L/CQ/SD-представимые множества и функции.	39
2.9	Симплекс-метод для решения задачи линейного программирования (в версии revised simplex method). Примеры сведения задачи линейного программирования к канонической форме. @elvarid	39
2.10	Метод Ньютона и логарифмических барьеров для выпуклых задач условной оптимизации с ограничениями вида равенств и неравенств. @aginaruck	42
2.11	Субградиентный метод для негладких задач оптимизации. Различные способы выбора длины шага. Субдифференциальное исчисление, примеры вычисления субдифференциалов. @sergevkim	45
2.12	Проксимальный градиентный метод для задачи композитной оптимизации. Примеры вычисления проксимальных операторов. @poly_nomial	46

2.13	Стохастический градиентный спуск, его скорость сходимости. Различные стратегии выбора длины шага. Стохастические методы оптимизации с линейной скоростью сходимости (SAG и SVRG). @Bitchert	46
2.14	Риманова оптимизация, её основные компоненты (риманово многообразие, касательное пространство, римановый градиент, операции ретракции и транспортировки вектора). Римановый градиентный спуск, примеры применения. Примеры римановых многообразий. @isadrtdinov . . .	50

1 Теоретический минимум

1.1 Определение линейной/сублинейной/суперлинейной скорости сходимости. Пример последовательности для каждого класса. @mercivboku

Невязку можно определять по-разному, в дальнейшем будут использоваться все виды определений:

1. $r_k = \|x_k - x_{opt}\|$
2. $r_k = |f(x) - f_{opt}|$
3. $r_k = \|\nabla f(x_k)\|$

Теперь рассмотрим различные виды сходимости.

Определения из лекции:

$\{r_k\}$ сходится **Q-сублинейно**, если

$$\overline{\lim}_{k \rightarrow \infty} \frac{r_{k+1}}{r_k} = 1.$$

$\{r_k\}$ сходится **Q-линейно**, если $\exists c \in (0; 1)$:

$$\overline{\lim}_{k \rightarrow \infty} \frac{r_{k+1}}{r_k} = c.$$

$\{r_k\}$ сходится **Q-суперлинейно**, если

$$\overline{\lim}_{k \rightarrow \infty} \frac{r_{k+1}}{r_k} = 0.$$

$\{r_k\}$ сходится **Q-квадратично**, если $\exists M > 0$:

$$r_{k+1} \leq M r_k^2 \quad \forall k \geq k_0$$

$\{r_k\}$ сходится **R-линейно**, если \exists монотонная последовательность $\{\eta_k\}$: $r_{k+1} \leq \eta_k$ и $\{\eta_k\}$ сходится Q-линейно.

Определения из конспекта:

$\{r_k\}$ сходится **q-линейно** с параметром $q \in (0; 1)$, если существует такое $C > 0$, что

$$r_k \leq C q^k \quad \forall k \geq m.$$

При этом если существует хотя бы одно такое q , то последовательность сходится **линейно**.

Пример: $r_k = \frac{1}{3^k}$ и $r_k = \frac{4}{3^k}$ сходятся линейно с любым параметром $q \in \left[\frac{1}{3}; 1\right)$. При этом константа

линейной сходимости (точная нижняя грань множества из всех q) для обеих последовательностей равна $\frac{1}{3}$.

$\{r_k\}$ сходится **сублинейно**, если $\{r_k\}$ сходится к 0, но не обладает линейной сходимостью. Если же, наоборот, $\{r_k\}$ обладает линейной сходимостью, и константа линейной сходимости равна 0, говорят, что $\{r_k\}$ сходится **суперлинейно** или **сверхлинейно**.

Пример: Последовательность $r_k = \frac{1}{k}$ не обладает линейной сходимостью, однако сходится к 0, значит, обладает сублинейной сходимостью.

Последовательность $r_k = \frac{1}{3^{k^2}}$ сходится линейно, при этом параметр сходимости приближается к 0 сколь угодно близко, значит, константа линейной сходимости равна 0, значит, данная последовательность сходится также суперлинейно.

1.2 Условия Армихо и Вульфа для неточной одномерной оптимизации. Процедура бэктрекинга. @mercivboku

Введем некоторые обозначения:

$$x_{k+1} = x_k + \alpha_k d_k, \quad x_k, d_k \in \mathbb{R}^n$$

$$\varphi(\alpha) = f(x_k + \alpha_k d_k), \text{ также есть примерное приближение: } \varphi(\alpha) \approx f(x_k) + \alpha \nabla f(x_k)^T d_k$$

$$\varphi(\alpha) \rightarrow \min_{\alpha}$$

$$\nabla f(x_k)^T d_k = \varphi'(0) < 0$$

Условие Армихо:

- $\varphi(\alpha_k) \leq \varphi(0) + c_1 \alpha_k \varphi'(0), \quad c_1 \in (0; 1)$

Нужно для этого: $\varphi(0) - \varphi(\alpha_k) = f(x_k) - f(x_k + \alpha_k d_k) \geq -\alpha_k(c_1 \varphi'(0)) > 0$

Хотим делать большие шаги, пока функция тоже меняется сильно. c_1 должно быть близким к 0, чем ближе, тем больше шаги мы разрешаем.

Условия Вульфа:

- $\varphi'(\alpha_k) \geq c_2 \varphi'(0)$ – слабое условие (помним, что $\varphi'(0) < 0$)
- $|\varphi'(\alpha_k)| \leq c_2 |\varphi'(0)|$, $c_2 \in (0, 1)$ – сильное условие
- $\varphi(\alpha_k) \leq \varphi(0) + c_1 \alpha_k \varphi'(0)$, $c_1 \in (0, 1)$ (условие Армихо, иногда он включал его в Вульфа, иногда нет, но всегда, когда рассматриваем Вульфа, разумно, чтобы Армихо тоже было выполнено)

Чем ближе c_2 к 0, тем ближе мы к локальному минимуму (тем больше берем шаг), чем ближе c_2 к 1, тем гибче (можем взять маленький шаг).

Утверждение. Если $\varphi'(0) < 0$, $\varphi(\alpha) > -\infty \forall \alpha$, $0 < c_1 \leq c_2 < 1$, то $\exists \alpha_*$, которое удовлетворяет условиям Армихо и Вульфа (в любой форме).

Алгоритм Backtracking:

```

1: procedure BACKTRACKING( $\alpha_{start}$ ,  $\beta \in [0.1, 0.8]$ ,  $\varphi_k : \mathbb{R}_+ \rightarrow \mathbb{R}$ )
2:    $\alpha \leftarrow \alpha_{start}$ 
3:   while  $\varphi_k(\alpha) > \varphi(0) + c_1 \alpha \varphi'_k(0)$  do
4:      $\alpha \leftarrow \beta \alpha$ 
5:   return  $\alpha$ 

```

1.3 Общее определение производной функции (через линейное отображение). Основные свойства производной (правила суммы, произведения и композиции). Определения второй производной, градиента и гессиана. @mercivboku

Для начала вспомним самое начало: Для функции одной переменной $f : \mathbb{R} \rightarrow \mathbb{R}$ ее производная в точке x обозначается $f'(x)$ и определяется из равенства:

$$f(x+h) = f(x) + f'(x)h + o(h) \text{ для всех достаточно малых } h.$$

То есть при некоторой зафиксированной точке x мы хотим приблизить изменение функции $f(x+h) - f(x)$ в окрестности нашего x с помощью линейной функции по h , и $f'(x)h$ – наилучший способ это сделать.

Определение. Пусть $x \in X$ – внутренняя точка множества X и пусть $L : U \rightarrow V$ – линейный оператор. Будем говорить, что функция f дифференцируема в точке x с производной L , если для всех достаточно малых $h \in U$ справедливо следующее разложение:

$$f(x+h) = f(x) + L[h] + o(\|h\|).$$

Будем обозначать производную символом $df(x)$, также валидны символы $Df(x)[h]$ или $df(x)[h]$, где $h = \Delta x$ – приращение.

Определение. Производная функции в точке x – это линейный оператор $df(x)$, который лучше всего аппроксимирует приращение функции:

$$f(x+h) - f(x) \approx Df(x)[h].$$

Свойства производной:

Пусть U и V – векторные пространства, X – подмножество U , $x \in X$ – внутренняя точка X . Справедливы следующие свойства:

- (Линейность) Пусть $f : X \rightarrow V$ и $g : X \rightarrow V$ – функции. Если f и g дифференцируемы в точке x , а $c_1, c_2 \in \mathbb{R}$ – числа, то функция $(c_1 f + c_2 g)$ также дифференцируема в точке x и ее производная:

$$d(c_1 f + c_2 g)(x) = c_1 df(x) + c_2 dg(x).$$

- (Правило произведения) Пусть $\alpha : X \rightarrow \mathbb{R}$ и $f : X \rightarrow V$ – функции. Если α и f дифференцируемы в точке x , то функция αf также дифференцируема в точке x и

$$D(\alpha f)(x)[h] = (D\alpha(x)[h])f(x) + \alpha(x)Df(x)[h].$$

- (Правило композиции) Пусть Y – подмножество V , $f : X \rightarrow Y$ – функция. Также пусть W – векторное пространство, $g : Y \rightarrow W$ – функция. Если f дифференцируема в точке x , и g дифференцируема в точке $f(x)$, то их композиция $(g \circ f) : X \rightarrow W$ (определяем как $g(f(x))$) дифференцируема в точке x , и

$$D(g \circ f)(x) = Dg(f(x))[df(x)]$$

Определение. В случае $U = \mathbb{R}^n$ линейную функцию $Df(x)[h]$ можно представить с помощью скалярного произведения с некоторым вектором:

$$Df(x)[h] = \langle a_x, h \rangle, \quad \text{где } a_x \in \mathbb{R}^n \text{ – разный для каждого } x.$$

Вектор a_x называется *градиентом* функции f в точке x и обозначается $\nabla f(x)$.

Отметим, что в стандартном базисе он представляется в привычном нам виде вектора (столбца) частных производных:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x) \right) \in \mathbb{R}^n.$$

Определение. Пусть функция $f : X \rightarrow V$ дифференцируема в каждой точке $x \in X \subseteq U$.

Рассмотрим производную функции f при фиксированном приращении $h_1 \in U$ как функцию от x :

$$g(x) = Df(x)[h_1].$$

Если для функции g в некоторой точке x существует производная, то она называется *второй производной* функции f в точке x :

$$D^2 f(x)[h_1, h_2] := Dg(x)[h_2].$$

Определение. Вторую производную для функций $f : \mathbb{R}^n \rightarrow \mathbb{R}$ также можно представить в виде матрицы:

$$D^2 f(x)[h_1, h_2] = \langle H_x h_1, h_2 \rangle, \quad H_x \in \mathbb{R}^{n \times n}.$$

Матрица H_x называется *гессианом* функции f в точке x и обозначается $\nabla^2 f(x)$. В стандартном базисе это матрица частных производных:

$$\nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{i,j=1}^n.$$

Для дважды непрерывно дифференцируемой функции гессиан симметричен.

1.4 Классы липшицевых функций, гладких выпуклых и сильно выпуклых функций. Соответствующие глобальные верхние и нижние оценки на функцию. Примеры функций для каждого класса. @mercivboku

Липшицевость:

Обозначение: $f \in C_L^{k,m}$. Пусть $m \leq k$, $L > 0$ – константа Липшица, тогда:

$$f \in C_L^{k,m} \Leftrightarrow \left[\begin{array}{l} f \in C^k, \\ \|\nabla^m f(y) - \nabla^m f(x)\| \leq L \|y - x\| \quad \forall x, y \end{array} \right]$$

В частности, если $f \in C_L^{1,1}$, то верно

$$\|\nabla f(y) - \nabla f(x)\| \leq L \|y - x\| \Leftrightarrow \begin{aligned} f(y) &\leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2 \\ f(y) &\geq f(x) + \nabla f(x)^T (y - x) - \frac{L}{2} \|y - x\|^2 \end{aligned}$$

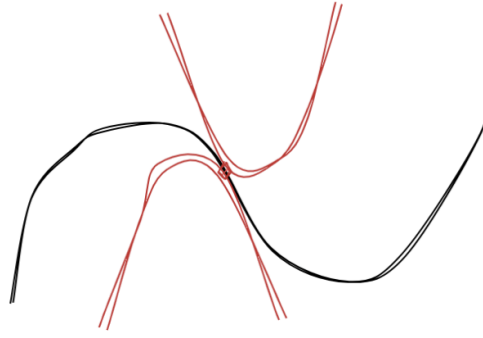


Рис. 1: Липшицевость $f \in C_L^{1,1}$ говорит о том, что в любой точке функцию сверху и снизу можно ограничить параболой, она не будет возрастать или убывать слишком быстро

А также можно сказать важные вещи про гессиан:

$$f \in C_L^{2,1} \Leftrightarrow \nabla^2 f(x) \preceq LI \Leftrightarrow \lambda_{\max}(\nabla^2 f(x)) \leq L,$$

то есть максимальное собственное значение гессиана ограничено сверху константой Липшица. В частности липшицевость означает, что не может быть вертикальных асимптот у функции.

Примеры:

- $f(x) = \sin(x)$ – функция с липшицевым градиентом, так как $f \in C_L^{1,1}$ и $|f'(y) - f'(x)| = |\cos(y) - \cos(x)| \leq L|x - y|$ с константой $L = 1$
- $f(x) = \exp(x)$ не является липшицевой, так как любые ее производные всегда будут возрастать экспоненциально.

Выпуклость:

Определение. Функция называется *выпуклой*, когда $\text{Dom}(f)$ выпукла и $\forall x, y \in \text{Dom}(f) \forall \alpha \in [0; 1]$ верно, что

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y).$$

В случае строгой выпуклости знак также строгий.

Для них верно следующее:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \forall y, x$$

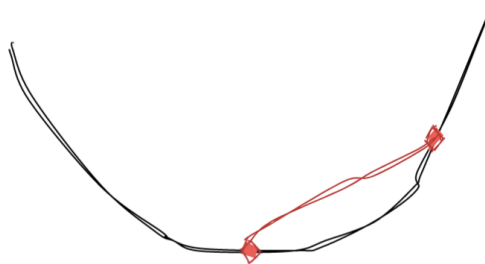


Рис. 2: Выпуклость означает, что если между любыми двумя точками провести отрезок, функция окажется снизу

Примеры:

- $f(x) = x^2$ – выпуклая
- $f(x) = \exp(x)$ – выпуклая
- $f(x) = \log(x)$ не является выпуклой

Сильная выпуклость.

Определение. Функция называется *сильно выпуклой* с параметром $\mu > 0$, если $f(x) - \frac{\mu}{2}\|x\|^2$ – выпуклая.

Также справедливо:

$$f(x) - \frac{\mu}{2}\|x\|^2 \text{ – выпуклая} \Leftrightarrow f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\mu}{2}\|y-x\|^2 \quad \forall x, y$$

И еще:

$$f \in C^2, f - \mu\text{-сильно выпукла} \Leftrightarrow \lambda_{\min}(\nabla^2 f(x)) \geq \mu$$

Таким образом, если есть и липшицевость, и сильная выпуклость, максимальное и минимальное собственное значение гессиана ограничено, это круто.

Примеры:

- $f(x) = x^2$ – сильно выпуклая с $\mu = 2$
- $f(x) = \exp(x)$ – не является сильно выпуклой, т.к. вторая производная может быть сколь угодно близкой к 0

1.5 Выпуклые множества и функции. Операции, сохраняющие выпуклость. Простейшие свойства выпуклых функций (неравенство Йенсена, выпуклость надграфика, выпуклость множеств подуровней). Дифференциальные критерии выпуклости. @mercivboku

Выпуклые множества. Пусть U – вещественное векторное пространство, Q – подмножество U . Множество Q называется *выпуклым*, если для любых двух точек $x, y \in Q$ и любого $\lambda \in [0; 1]$ точка $\lambda x + (1 - \lambda)y$ также принадлежит множеству Q .

Примеры: Все пространство U , множество из одного элемента, выпуклое множество, множество решений системы линейных неравенств:

$$Q := \{x \in U : \langle a_i, x \rangle \leq b_i \forall i \in I\}.$$

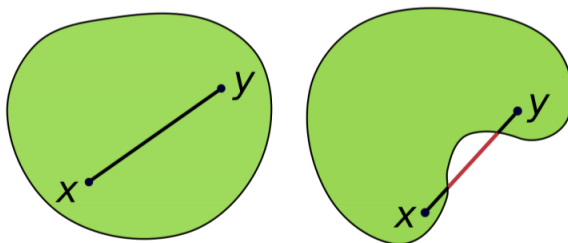


Рис. 3: Слева – выпуклое, справа – не выпуклое.

Свойства выпуклых функций

1. (Неравенство Йенсена) Пусть U – вещественное векторное пространство, Q – непустое выпуклое множество в U , и $f: Q \rightarrow \mathbb{R}$ – выпуклая функция. Пусть также x_1, \dots, x_k – точки в множестве Q и $\lambda_1, \dots, \lambda_k$ – неотрицательные коэффициенты, суммирующиеся в единицу: $\lambda_i \geq 0$ и $\sum_{i=1}^k \lambda_i = 1$. Тогда справедливо следующее неравенство:

$$f\left(\sum_{i=1}^k \lambda_i x_i\right) \leq \sum_{i=1}^k \lambda_i f(x_i),$$

причем равенство достигается тогда и только тогда, когда функция f является аффинной или когда все точки совпадают: $x_1 = \dots = x_k$.

Другими словами, неравенство Йенсена говорит о том, для выпуклой функции значение функции от выпуклой комбинации точек не превосходит соответствующей выпуклой комбинации значений функции.

2. (Надграфик) Пусть U – вещественное векторное пространство, Q – непустое множество в U . Надграфиком функции $f : Q \rightarrow \mathbb{R}$ называется множество

$$\text{Epi}(f) := \{(x, t) \in Q \times \mathbb{R} : f(x) \leq t\}.$$

Надграфик – это множество из пространства $U \times \mathbb{R}$.

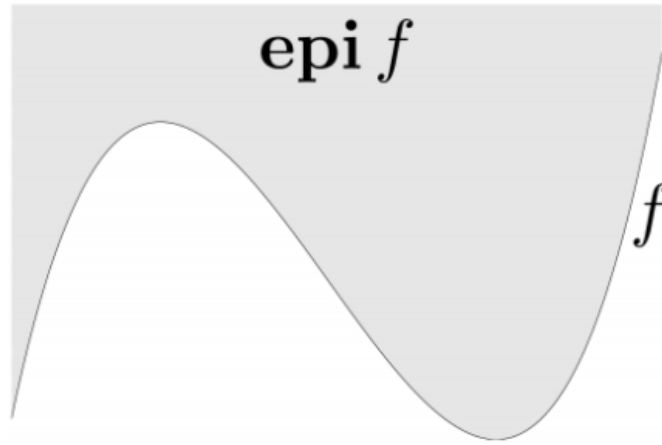


Рис. 4: Все серое – надграфик функции $f(x)$.

Определение. (Выпуклости через надграфик). Пусть U – вещественное векторное пространство, Q – непустое выпуклое множество в U . Функция $f : Q \rightarrow \mathbb{R}$ является выпуклой тогда и только тогда, когда ее надграфик $\text{Epi}(f)$ является выпуклым множеством в пространстве $U \times \mathbb{R}$.

Утверждение. (Выпуклость множества линий уровня). Пусть U – вещественное векторное пространство, Q – непустое множество в U , и пусть $f : Q \rightarrow \mathbb{R}$ – выпуклая функция. Тогда для любого $\alpha \in \mathbb{R}$ соответствующее множество линий уровня

$$\text{Lev}_f(\alpha) := \{x \in Q : f(x) \leq \alpha\}$$

является выпуклым.

3. (Дифференциальные критерии выпуклости).

Утверждение. (Условие выпуклости первого порядка). Пусть $\text{Dom}(f)$ является открытым множеством, и функция f дифференцируема всюду на $\text{Dom}(f)$. Функция f является выпуклой тогда и только тогда, когда $\text{Dom}(f)$ является выпуклым множеством и

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$$

для всех $x, y \in \text{Dom}(f)$.

Утверждение. (Дифференциальное условие оптимальности для выпуклой функции). Пусть $f : U \rightarrow \mathbb{R} \cup \{+\infty\}$ – выпуклая функция, $\text{Dom}(f)$ является открытым множеством, и пусть $x^* \in \text{Dom}(f)$. Тогда x^* является глобальным минимумом функции f , если и только если $\nabla f(x^*) = 0$. Другими словами, любая стационарная точка автоматически является глобальным минимумом функции f .

Утверждение. (Условие выпуклости второго порядка). Пусть $\text{Dom}(f)$ является открытым множеством, и функция f дважды дифференцируема на $\text{Dom}(f)$. Функция f является выпуклой тогда и только тогда, когда $\text{Dom}(f)$ является выпуклым множеством и

$$D^2 f(x)[h, h] := \langle \nabla^2 f(x)h, h \rangle \geq 0$$

для всех $x \in \text{Dom}(f)$ и всех $h \in U$. Если $U = \mathbb{R}^n$, то это эквивалентно положительной полуопределенности гессиана:

$$\nabla^2 f(x) \succcurlyeq 0$$

для всех $x \in \text{Dom}(f)$.

1.6 Определения субградиента и субдифференциала. Субдифференциальное исчисление (умножение на скаляр, композиция с аффинным преобразованием, теорема Моро–Рокафеллара, максимум конечного числа функций и теорема Данскина). @poly_nomial

Субдифференциал функции:

$$\partial F(x) = \{z | F(y) \geq F(x) + z^T(y - x) \forall y\}$$

Фактически это множество касательных к функции в точке x . Если функция гладкая и дифференцируемая в каждой точке, то в этом множестве будет всего одна касательная - градиент к функции в каждой её точке. Т.е.

$$\text{Если } F(x) \in C^1 \rightarrow \partial F(x) = \{\nabla F(x)\}$$

Если же функция просто гладкая, но производная есть не в каждой её точке - тогда и будет получено данное мн-во.

Пример:

$$F(x) = |x| \rightarrow \partial F(x) \begin{cases} x > 0, \{1\} \\ x < 0, \{-1\} \\ x = 0, [-1, 1] \end{cases}$$

Здесь в качестве z получены коэф-ты к касательным в каждой точке функции. Последний случай также очень легко проверяется:

$$F(y) \geq F(0) + z^T(y - 0) \forall y \rightarrow |y| \geq z * y \rightarrow \begin{cases} 1) y > 0 \rightarrow z \leq 1 \\ 2) y < 0 \rightarrow z \geq -1 \end{cases}$$

Также субдифференциал это выпуклое мн-во, проверяется легко через его определение:

$$z_1, z_2 \in \partial F(x) \rightarrow \begin{cases} F(y) \geq F(x) + z_1^T(y - x) \\ F(y) \geq F(x) + z_2^T(y - x) \end{cases} \forall y \rightarrow$$

$$\begin{aligned} \alpha F(y) + (1 - \alpha)F(y) &\geq \alpha(F(x) + z_1^T(y - x)) + (1 - \alpha)(F(x) + z_2^T(y - x)) \rightarrow \\ F(y) &\geq F(x) + (\alpha z_1 + (1 - \alpha)z_2)^T(y - x) \rightarrow \alpha z_1 + (1 - \alpha)z_2 \end{aligned}$$

Субградиент - эл-т из мн-ва субдифференциала: $z \in \partial F(x)$.

Пусть $f : E \rightarrow \mathbb{R}$ - функция, определенная на множестве E в евклидовом пр-ве V , и пусть $x_0 \in E$. Вектор $s \in V$ называется **субградиентом** функции f в точке x_0 , если:

$$f(x) \geq f(x_0) + \langle s, x - x_0 \rangle \quad \forall x \in E$$

Это определение субградиента функции в точке x_0 в отрыве от определения субдифференциала. Из этих определений следует, что субград/субдиф зависят от введенного в пр-ве V скалярного произведения. Поэтому если в рамках задачи/теории не оговорено иное, то под скалярным произведением в рассматриваемом V мы берем стандартное.

При использовании субграда можно перейти к терминологии, что это часть двойственного пр-ва ($s \in V^*$), а $\langle s, x \rangle$ - вычисление лин функционала s на аргументе x (это позволят рассматривать такие V , где скалярное произведение явно ввести невозможно), но это все сложно и для упрощения теории далее не рассматривается.

Также важно упомянуть, что из субдифференцируемости функции (т.е. в каждой её точке есть субдифференциал как непустое мн-во) следует выпуклость этой функции (проверяется через опр выпуклости и субдиф-ла в точках x, y). Но обратное утверждение неверно. Пример:

$$\text{Пусть } f : [0, +\infty) \rightarrow \mathbb{R}; f(x) = -\sqrt{x}. \text{ Тогда } \partial f(0) = \emptyset$$

Тут также распишете субдиф в нуле по определению и получите, что $-\sqrt{x} \geq z^T(x - 0) \rightarrow zx \leq -\sqrt{x} \rightarrow z \leq -\frac{1}{\sqrt{x}} \rightarrow z \leq -\infty$, что невозможно. Поэтому в обратную сторону эта тема не работает.

Не всякая выпуклая функция является субдифференцируемой. Но чаще всего, отсутствие субградиентов у выпуклой функции — это довольно экзотическая ситуация, которая происходит только в граничных точках области определения. Один из основных результатов теории субдифференциалов состоит в том, что в любой внутренней точке области определения выпуклая функция гарантированно имеет субградиенты, причем субдифференциал является не просто выпуклым и замкнутым множеством, но также ограниченным (и, тем самым, компактным, согласно теореме Гейне–Бореля).

Из этого дальше вытекает такая теорема. Пусть $f : E \rightarrow \mathbb{R}$ - выпуклая функция, определенная на множестве E в конечно мерно евклидовом пр-ве V , и пусть $x_0 \in \text{int}(E)$. Тогда множество $\partial f(x_0)$ - не пусто и является выпуклым и компактом.

Дальше есть что-то про связь субдиффов и сопряженных норм, но это просят в рамках другого пункта теор-мина, поэтому дальше к субдиффу исчислениям.

Умножение на полож. скаляр. Пусть $f : E \rightarrow \mathbb{R}$ - функция, определенная на мн-ве E в евклидовом пр-ве, $c > 0$, и $x_0 \in E$. Покажем, что

$$\partial(cf)(x_0) = c\partial f(x_0).$$

Это вроде легко показывается через определение субдифференциала и рассмотрение субградиента из него в общем виде. (поправьте, если это не так :)

Одним из самых первых результатов, появившихся в теории субдифференциального исчисления, является правило суммы, которое в лит-ре известно под названием теоремы Моро-Рокафеллара.

Теорема Моро-Рокафеллара. Пусть $f : R \rightarrow \mathbb{R}$ и $g : G \rightarrow \mathbb{R}$ - функции, определенные на мн-вах E и G в одном и том же евклидовом пр-ве, и пусть $x_0 \in E \cap G$. Тогда

$$\partial(f+g)(x_0) \supseteq \partial f(x_0) + \partial g(x_0)$$

Если дополнительно функции f и g выпуклые, и при этом $E \cap \text{int}(G) \neq \emptyset$, то

$$\partial(f+g)(x_0) = \partial f(x_0) + \partial g(x_0).$$

Для док-ва вложения (показано выше) используют опр. субдифференциала через нер-ва. Для док-ва обратного вложения подключают теорему об отделимости выпуклых множеств.

Пример про важность $E \cap \text{int}(G) \neq \emptyset$:

Упражнение. Пусть $f : [0, +\infty) \rightarrow \mathbb{R}$ - функция $f(x) := -\sqrt{x}$, и пусть $g : (-\infty, 0] \rightarrow \mathbb{R}$ - функция $g(x) := -\sqrt{-x}$. Покажите, что $\partial(f+g)(0) = \mathbb{R}$, в то время как $\partial f(0) = \partial g(0) = \emptyset$.

Используя индукцию и свойство о том, что внутренность конеч. пересечения множеств равна пересечению внутренностей, получаем обобщение теоремы Моро-Рокафеллара для конечных сумм:

Следствие, теорема Моро-Рокафеллара II. Пусть $m \geq 2$ - целое, $f_1 : E_1 \rightarrow \mathbb{R}, \dots, f_m : E_m \rightarrow \mathbb{R}$ - функции, определенные на множествах E_1, \dots, E_m в одном и том же евклидовом пр-ве, и пусть $x_0 \in \bigcap_{i=1}^m E_i$. Тогда

$$\partial \left(\sum_{i=1}^m f_i \right) (x_0) \supseteq \sum_{i=1}^m \partial f_i(x_0).$$

Если дополнительно функции f_i - выпуклые, и при этом $\bigcap_{i=1}^{m-1} \text{int}(E_i) \cap E_m \neq \emptyset$, то

$$\partial \left(\sum_{i=1}^m f_i \right) (x_0) = \sum_{i=1}^m \partial f_i(x_0).$$

Композиция с аффинным преобразованием:

Пусть V и W - евклидовы пр-ва, $T : V \rightarrow W$ - аффинное преобр. $T(x) := Ax + b$, где $A : V \rightarrow W$ - линейное преобр., $b \in W$. Пусть $g : G \rightarrow \mathbb{R}$ - функция, определенная на множестве G в пр-ве W и $x_0 \in T^{-1}(G)$. Тогда

$$\partial(g \circ T)(x_0) \supseteq A^* \partial g(T(x_0)),$$

где $A^* : W \rightarrow V$ - сопряженный оператор для A . Если дополнительно функция g выпуклая, и при этом $T(V) \cap \text{int}(G) \neq \emptyset$, то

$$\partial(g \circ T)(x_0) = A^* \partial g(T(x_0)).$$

Пример. Пусть $f : \mathbb{R}^n \rightarrow \mathbb{R}$ - функция $f(x) := | \langle a, x \rangle |$, где $a \in \mathbb{R}^n$. Пусть $A : \mathbb{R}^n \rightarrow \mathbb{R}$ - линейное преобр. $A(x) := \langle a, x \rangle$. В этом случае $A^*t = ta$ для всех $t \in \mathbb{R}$ (опять тут от автора "почему?"). Поскольку $A(\mathbb{R}^n) \cap \text{int}(\mathbb{R}) \neq \emptyset$ (например, это мн-во содержит ноль), то

$$\partial f(x) = \partial | \langle a, x \rangle | a = \begin{cases} [-1, 1]a, & \text{если } \langle a, x \rangle = 0, \\ \text{sign}(\langle a, x \rangle)a, & \text{иначе} \end{cases} = \begin{cases} [-a, a], & \text{если } \langle a, x \rangle = 0, \\ \text{sign}(\langle a, x \rangle)a, & \text{иначе} \end{cases}$$

(Второе равенство вытекает из формулы для субдиф-ла модуля, пример выше)

функция g это просто модуль, а скалярное произведение внутри g и есть аффинное преобр, с которым функция

комбинируется. Выпуклость обеих функций очевидна, плюс пересечение результат лин опратора и выпуклого мн-ва от \mathbb{R} также понятны. Поэтому далее субдиф для функции g вычисляется как от композиции функции и аффин. преобр. или как сопряж лин. оператор на субдиф от функции $g(T(x_0))$.

Для поиска субдифференциала от функций максимума используют следующих два крутых правила:

Теорема, максимум конечного числа функций. Пусть $f_1 : E_1 \rightarrow \mathbb{R}, \dots, f_m : E_m \rightarrow \mathbb{R}$ - функции, опр. на мн-вах E_1, \dots, E_m в одном и том же конечномерном евклидовом пр-ве, пусть $f : \cap_{i=1}^m E_i \rightarrow \mathbb{R}$ - функция

$$f(x) := \max\{f_1(x), \dots, f_m(x)\},$$

и пусть $x_0 \in \cap_{i=1}^m E_i$. Тогда

$$\partial f(x_0) \supseteq \overline{\text{Conv}(\cup_{i \in I(x_0)} \partial f_i(x_0))},$$

где $I(x_0)$ - мн-во индексов, на которых достигается максимум в точке x_0 (гарантированно непустое):

$$I(x_0) := \{1 \leq i \leq m : f_i(x_0) = f(x_0)\}.$$

Если дополнительно функции f_1, \dots, f_m - выпуклые, и при этом $x_0 \in \text{int}(\cap_{i=1}^m E_i)$ (или, эквивалентно, $x_0 \in \cap_{i=1}^m \text{int}(E_i)$), то

$$\partial f(x_0) = \text{Conv}(\cup_{i \in I(x_0)} \partial f_i(x_0)).$$

Теорема местами носит название ДУбовицкого-Милютина.

Пример. Пусть $f : \mathbb{R} \rightarrow \mathbb{R}$ - функция $f(x) := |x| = \max\{-x, x\}$, и пусть $x_0 \in \mathbb{R}$. Применяя теорему выше и идею о том, что субдифференцируемая функция в каждой точке - выпуклая, получим:

$$\partial f(x_0) = \begin{cases} \text{Conv}(\{-1, 1\}), & \text{если } x_0 = 0, \\ \text{Conv}(\text{sign}(x_0)), & \text{если } x_0 \neq 0 \end{cases} = \begin{cases} [-1, 1], & \text{если } x_0 = 0, \\ \text{sign}(x_0), & \text{если } x_0 \neq 0 \end{cases}$$

Что совпадает с результатом посчитанного субдиф-ла для модуля ранее.

Для того, чтобы распространить предыдущее правило на беск семейство функций, необходимы дополнительные топологические условия: компактность и полунепрерывность.

Теорема Данскина. Пусть V - конечномерное евклидово пространство, I - произвольное множество (не обязательно конечное и не обязательно счетное), и пусть для каждого $i \in I$ задано множество E_i в V и функция $f_i : E_i \rightarrow \mathbb{R}$. Пусть $f : E \rightarrow \mathbb{R}$ - функция

$$f(x) := \sup_{i \in I} f_i(x),$$

определенная на мн-ве $E := \{x \in \cap_{i \in I} E_i : \sup_{i \in I} f_i(x) < +\infty\}$, и пусть $x_0 \in E$. Тогда

$$\partial f(x_0) \supseteq \overline{\text{Conv}(\cup_{i \in I(x_0)} \partial f_i(x_0))},$$

где $I(x_0)$ - множество индексов, на которых достигается супремум в точке x_0 (возможно, пустое):

$$I(x_0) := \{i \in I : f_i(x) = f(x)\}.$$

Если дополнительно функция f_i выпукла для каждого $i \in I$, мн-во I является компактом (в некотором топологическом пространстве), и при этом для каждого $x \in \cap_{i \in I} E_i$ функция $i \rightarrow f_i(x)$ является полунепрерывной сверху на I , то $E = \cap_{i \in I} E_i$, мн-во $I(x_0)$ не пусто, и для $x_0 \in \text{int}(E)$ имеет место

$$\partial f(x_0) = \text{Conv}(\cup_{i \in I(x_0)} \partial f_i(x_0)).$$

В этой теореме помимо выпуклости требуются компактность индексного мн-ва I , а также полунепрерывность сверху функции $i \rightarrow f_i(x)$ на I для каждого $x \in E$. Данные ограничения выглядят достаточно логично, если мы хотим, чтобы в каждой точке $x \in E$ супремум по $i \in I$ достигался хотя бы для одного из индексов, это будет следовать из того факта, что полунепрерывная сверху функция, опр на компакте, будет достигать своего наибольшего значения на нём. Особый интерес может представлять тот факт, что никаких других дополнительных ограничений при этом накладывать не надо: имеющихся условий будет достаточно для вычисления дифференциала.

Упражнение. Пусть $I = \mathbb{R}_{++}$ и пусть $f_i(x) = -i \log x$. Тут предлагают проверить, выполняются ли условия теоремы Данскина. Как выглядит ф-ия $f(x) = \sup_{i>0} f_i(x)$? И стоит посмотреть $\partial f(1)$.

Пример. Снова попытаемся вывести субдифференциал для нормы, но уже через теорему Донскина. Пусть V - конечномерное евклидово пространство, $\|\cdot\|$ - произвольная норма в V , и пусть $x_0 \in V$. Используя сопряж норму $\|\cdot\|_*$ и тот факт, что сопряжение является инволюцией (боже, это просто досвидание!!!), можно записать след. вариационное представление:

$$\|x\| = \max_{\|s\|_* = 1} \langle s, x \rangle$$

справедливое для любого $x \in V$. В этом случае индексное мн-во $I := \{s \in V : \|s\|_* = 1\}$ представляет собой единичную сферу и является компактом (в пр-ве V). Поскольку $x_0 \in \text{int}(V)$, и для каждого $x \in V$ функция $s \rightarrow \langle s, x \rangle$ является линейной (а, значит, непрерывной, и тем более, полунепрерывной сверху), то, согласно теореме Донскина, получаем

$$I(x_0) = \{s \in V : \|s\|_* = 1; \langle s, x_0 \rangle = \|x_0\|\},$$

и

$$\partial \|\cdot\|(x_0) = \text{Conv}(\{s \in V : \|s\|_* = 1; \langle s, x_0 \rangle = \|x_0\|\}).$$

Здесь была использована формула $\partial \langle s, \cdot \rangle(x_0) = s$.

Если $x_0 = 0$, то условие $\langle s, x_0 \rangle = \|x_0\|$ выполняется бессодержательно, и

$$\partial \|\cdot\|(0) = \text{Conv}(\{s \in V : \|s\|_* = 1\}) = \overline{B}_{\|\cdot\|_*}(0,1),$$

(получили снова какой-то индикаторный оператор для шара). Если же $x_0 \neq 0$, то множество

$$\{s \in V : \|s\|_* = 1 : \langle s, x_0 \rangle = \|x_0\|\}$$

является выпуклым (снова почему? - думаю тут можно по определению выпуклости показать). Тогда операцию взятия выпуклой оболочки можно убрать, и в итоге получаем:

$$\partial \|\cdot\|(x_0) = \begin{cases} \overline{B}_{\|\cdot\|_*}(0,1), & \text{если } x_0 = 0, \\ \{s \in V : \|s\|_* = 1 : \langle s, x_0 \rangle = \|x_0\|\}, & \text{иначе.} \end{cases}$$

В общем из всего, что тут есть - надо знать определения субградиента/субдифференциала и всех теоремок из их исчисления. Примеры даны для понимания применения этих теоремок, может задачу вам на что нибудь такое дадут (правда тогда это вообще поминки будут).

1.7 Сопряженная функция Фенхеля. Неравенство Фенхеля–Юнга. Теорема Фенхеля–Моро. Связь сопряженной функции с субдифференциалом. @poly _nomial

Сопряженная функция Фенхеля. Пусть $f : E \rightarrow \mathbb{R}$ - функция, заданная на множестве E в евклидовом пр-ве V . Сопряженной функцией для функции f наз-ся функция $f^* : E_* \rightarrow \mathbb{R}$, определенная как

$$f^*(s) := \sup_{x \in E} \{\langle s, x \rangle - f(x)\},$$

где $E_* := \{s \in V : \sup_{x \in E} \{\langle s, x \rangle - f(x)\} < +\infty\}$. Её и называют сопряж. функцией Фенхеля, а преобразование $f \rightarrow f^*$ зовут преобр. Фенхеля/Лежандра.

Сопряж ф-ия также зависит от скалярного произведения в пр-ве V , про него стоит дополнительно пояснить. Везде дальше в курсе по дефолту - стандартное.

Неравенство Фенхеля–Юнга. Пусть $f : E \rightarrow \mathbb{R}$ - функция, заданная на множестве E в евклидовом пр-ве, и пусть $f^* : E_* \rightarrow \mathbb{R}$ - сопряженная функция. Тогда для всех $x \in E$ и всех $s \in E_*$ выполнено:

$$\langle s, x \rangle \leq f^*(s) + f(x).$$

Несмотря на свою простоту, это утв. имеет многочисленные приложения. Нер-во Фенхеля–Юнга дает систематичный способ построения оценок вида

$$\langle s, x \rangle \leq f^*(s) + f(x), \quad (1)$$

для всех s и x , где f и g - некоторые функции. Необходимость сопряженной функции можно связать с таким вопросом: какие пары функций f, g нужно брать в нер-ве (1), чтобы оно было "максимально точным"? Если зафиксировать конкретным значением функцию f , то другая функция g , дающая максимальную "точную" оценку в нер-ве (1) уже определяется однозначно - в точности сопряженная функция f^* .

Пример с поиском сопряженной функции: пусть V - евклидово пр-во, и пусть $f : V \rightarrow \mathbb{R}$ - функция $f(x) := \langle a, x \rangle$, где $a \in V$. Поскольку

$$\sup_{x \in V} \{\langle s, x \rangle - \langle a, x \rangle\} = \sup_{x \in V} \langle s - a, x \rangle = \begin{cases} 0, & \text{если } s = a, \\ +\infty, & \text{иначе} \end{cases}$$

для всех $s \in V$, то сопряженная функция f^* - это индикаторная функция $\delta_{\{a\}}$ одноэлементного мн-ва $\{a\}$. И пример наоборот:

Пусть V - евклидово пр-во, $a \in V$, и пусть $\delta_{\{a\}} : \{a\} \rightarrow \mathbb{R}$ - индикаторная функция одноэлементного мн-ва $\{a\}$. Поскольку для $s \in V$ выполнено

$$\sup_{x \in \{a\}} \{ \langle s, x \rangle - \delta_{\{a\}}(x) \} = \langle s, a \rangle,$$

то сопряженная функция $\delta_{\{a\}}^*$ равна линейной форме $x \rightarrow \langle a, x \rangle$, рассмотренной в предыдущем примере.

В след теореме будут несколько примеров, которые показывают, что преобразование Фенхеля это инволюция, т.е. сопряженная функция f^{**} к функции f^* - это исходная функция f . На основе этой идеи существует теорема Фенхеля-Моро.

Теорема Фенхеля-Моро. Пусть f - функция, заданная на непустом множестве в евклидовом пространстве. Тогда $f = f^{**}$, если и только если f является выпуклой и замкнутой.

В одну сторону теорема очевидна: если $f = f^{**}$, то f обязана быть выпуклой и замкнутой, т.к. сопряженная сопряж. функция к любой функции (в том числе к f^*) является выпуклой и замкнутой вне зависимости от того, обладала ли исходная функция этими свойствами или нет. Нетривиальной частью теоремы Фенхеля-Моро является обратное утв., что замкнутость f является не только необходимым условием, но и достаточным. (хз как доказывать в эту сторону, наверно уточним у Кротова, надо ли это делать).

Список кучи полезных свойств с сопряженными функциями. Пусть $f : E \rightarrow \mathbb{R}$ и $g : G \rightarrow \mathbb{R}$ - функции, заданные на множествах E и G в евклидовых пр-вах V и W соответственно, и пусть $f^* : E_* \rightarrow \mathbb{R}$ и $g^* : G_* \rightarrow \mathbb{R}$ - соответствующие сопряженные функции.

1. (Умножение на полож. скаляр) Если $c > 0$, то $(cf)^* : cE_* \rightarrow \mathbb{R}$ - это функция $(cf)^*(s) = cf^*(s/c)$ для всех $s \in E_*$.
2. (Сдвиг аргумента) Пусть $a \in V$, и пусть $h : E - a \rightarrow \mathbb{R}$ - $h(x) := f(x + a)$. Тогда $h^* : E_* \rightarrow \mathbb{R}$ - это функция $h^*(s) = f^*(s) - \langle s, a \rangle$ для всех $s \in E_*$.
3. (Прибавление аффинной функции) Пусть $a \in V$, $b \in \mathbb{R}$ и пусть $h : E \rightarrow \mathbb{R}$ - функция $h(x) := f(x) + \langle a, x \rangle + b$. Тогда $h^* : E_* + a \rightarrow \mathbb{R}$ - это функция $h^*(s) = f^*(s - a) - b$ для всех $s \in E_* + a$.
4. (Композиция с обратимым лин. преобразованием) Пусть $A : V \rightarrow V$ - обратимое линейное преобразование. Тогда $(f \circ A)^* : A^*(E_*) \rightarrow \mathbb{R}$ - это функция $(f \circ A)^*(s) = f^*(A^{-1}s)$ для всех $s \in A^*(E_*)$.
5. (Сепарабельная сумма). Пусть $\phi : E \times G \rightarrow \mathbb{R}$ - функция $\phi(x, y) := f(x) + g(y)$. Тогда $\phi^* : E_* \times G_* \rightarrow \mathbb{R}$ - это функция $\phi^*(s, u) = f^*(s) + g^*(u)$ для всех $s \in E_*$ и всех $u \in G_*$.

На применение этих исчислений могут спокойно дать задачу какую-нибудь.

Перед выводом связи меж сопряж ф-ей и субдифф-ом стоит рассмотреть след пример. Пусть V - конечномерное евклидово пр-во, $x_0 \in V$. Пусть $\| \cdot \|$ - произвольная норма в V (необязательно индуцированная скалярным произведением), и $\| \cdot \|_*$ - соотв. сопряженная норма. Тогда:

$$\partial \| \cdot \| (x_0) = \{ s \in V : \| s \|_* \leq 1; \langle s, x_0 \rangle = \| x_0 \| \} = \begin{cases} \overline{B_{\| \cdot \|}}(0, 1), & \text{если } x_0 = 0 \\ \{ s \in V : \| s \|_* = 1; \langle s, x_0 \rangle = \| x_0 \| \}, & \text{иначе} \end{cases} \quad (1)$$

Где $\overline{B_{\| \cdot \|}}(0, 1) := \{ s \in V : \| s \|_* \leq 1 \}$ - замкнутый единичный шар с центром в нуле относительно сопряженной нормы. Другими словами, вектор $s \in V, \| s \|_* = 1$, является субградиентом нормы $\| \cdot \|$ в точке $x_0 \neq 0$, если и только если нер-во Гельдера (след пункта теоремы) $\langle s, x_0 \rangle \leq \| x_0 \|$ переходит в равенство.

Док-во. Пусть $s \in V$. По опр субградиента: $s \in \partial \| \cdot \| (x_0)$, если и только если

$$\langle s, x \rangle - \| x \| \leq \langle s, x_0 \rangle - \| x_0 \|$$

, для всех $x \in V$, или, эквивалентно

$$\sup_{x \in V} \{ \langle s, x \rangle - \| x \| \} \leq \langle s, x_0 \rangle - \| x_0 \| \quad (*)$$

По определению супремума, последнее равносильно (тут даже автор конспекта чет сомневается)

$$\sup_{x \in V} \{ \langle s, x \rangle - \| x \| \} = \langle s, x_0 \rangle - \| x_0 \|$$

Заметим, что выражение в левой части есть супремум из определения сопряженной функции Фенхеля для нормы, который, как мы уже знаем, равен:

$$\sup_{x \in V} \{ \langle s, x \rangle - \|x\| \} = \begin{cases} 0, & \text{если } \|s\|_* \leq 1, \\ +\infty, & \text{иначе} \end{cases}$$

Таким образом, (*) эквивалентно

$$\|s\|_* \leq 1 \quad \text{и} \quad \langle s, x_0 \rangle = \|x_0\|.$$

В итоге

$$\partial \| \cdot \| (x_0) = \{ s \in V : \|s\|_* \leq 1; \langle s, x_0 \rangle = \|x_0\| \}.$$

Остается заметить, что для $x_0 \neq 0$ нер-во $\|s\|_* \leq 1$ обязано переходить в равенство, поскольку при $\|s\|_* < 1$, из нер-ва Гельдера следует $\langle s, x_0 \rangle < \|s\|_* \|x_0\| < \|x_0\|$. (одним словом - ужас).

На этом примере с сопряженной нормой можно понять, что аналогичным образом для произвольной функции f (т.е. не только для нормы) ее субдиф можно описать в терминах двойственного объекта - сопряженной функции Фенхеля.

Тогда в сухом остатке эта формулировка звучит так: Пусть $f : E \rightarrow \mathbb{R}$ - функция, определенная на мн-ве E в евклидовом пр-ве. Пусть $x_0 \in E$, и пусть $f^* : E \rightarrow \mathbb{R}$ - сопряженная функция. Тогда можно показать, что

$$\partial f(x_0) = \{ s \in E_* : \langle s, x_0 \rangle = f^*(s) + f(x_0) \} \quad (2)$$

Словами: вектор $s \in E_*$ является субградиентом функции f в точке x_0 , если и только если неравенство Фенхеля-Юнга $\langle s, x_0 \rangle \leq f^*(s) + f(x_0)$ переходит в равенство.

В случае $f = \| \cdot \|$, получаем $f^* = \delta_{\overline{B_{\| \cdot \|}}, (0,1)}$, т.е. сопряженная функция равна индикаторной функции шара $\overline{B_{\| \cdot \|}}, (0,1)$ и формула выше (2) переходит в формулу из примера (1).

Следствие из этих рассуждений - критерии равенства в нер-ве Фенхеля-Юнга. Пусть $f : R \rightarrow \mathbb{R}$ - выпуклая замкнутая функция, $f^* : E_* \rightarrow \mathbb{R}$ - сопряженная функция, и пусть $x \in E$, $s \in E_*$. Тогда след. утверждения эквивалентны:

1. $\langle s, x \rangle = f^*(s) + f(x)$
2. $s \in \partial f(x)$.
3. $x \in \partial f^*(s)$.

Док-во. Согласно выводу по связи сопряж функции и субдиф-ов, условие $\langle s, x \rangle = f^*(s) + f(x)$ равносильно вложению $s \in \partial f(x)$. С другой стороны, поскольку f выпуклая и замкнутая, то, по теореме Фенхеля-Моро, $f^{**} = f$. Применяя вывод по связи выше к функции f^* , получаем, что равенство $\langle s, x \rangle = f^*(s) + f(x)$ эквивалентно вложению $x \in \partial f^*(s)$.

Открытый вопрос: надо ли в рамках этого пункта что-то про двойственность фенхеля говорить. Ответ узнаем завтра на консе.

1.8 Сопряженная норма. Неравенство Гельдера. Основные примеры. @poly_nomial

Сопряженная норма. Пусть V - конечномерное евклидово пр-во, пусть $\| \cdot \|$ - произвольная норма в V (необязательно индуцированная скалярным произведением). Тогда *сопряженной нормой* для $\| \cdot \|$ называется норма $\| \cdot \|_*$, определенная как

$$\|s\|_* := \max_{\|v\|=1} \langle s, v \rangle.$$

Заметим, что это определение корректное, поскольку максимум достигается по теореме Вейерштрасса в силу компактности единичной сферы.

Теперь примеры по работе с этой нормой:

Пусть $p \in [1, +\infty]$, и пусть $\| \cdot \|_p : \mathbb{R}^n \rightarrow \mathbb{R}$ - l^p -норма. Тогда из нер-ва Гельдера следует, что сопряженная норма $(\| \cdot \|_p)_*$ - это l^q -норма $\| \cdot \|_q$, где $q \in [1, +\infty]$ определяется из неравенства $1/p + 1/q = 1$. В частности l^1 - норма является сопряженной к l^∞ - норме; в то же время, l^∞ - норма является сопряженной к l^1 - норме, а l^2 норма является самосопряженной.

Беда.(около консы тут будут нужные примеры и нормальный вариант нер-ва гельдера).

1.9 Проксимальный оператор. Связь с евклидовой проекцией. @bigbluebutterfly

Пусть $f : E \rightarrow \mathbb{R}$ - функция, заданная на множестве E в евклидовом пространстве V . Определим проксимальный оператор функции f как отображение $\text{Prox}_f : V \rightarrow E$, заданное следующим образом:

$$\text{Prox}_f(x) := \arg \min_{y \in E} \left\{ f(y) + \frac{1}{2} \|y - x\|^2 \right\}$$

Если функция f выпуклая и замкнутая, то в каждой точке множество значений Prox_f состоит из одного элемента как множество минимумов сильно выпуклой и замкнутой функции.

Рассмотрим функцию $g : E \rightarrow \mathbb{R}$, определенную как

$$g(x) = \delta_C(x)$$

т.е. g - индикаторная функция непустого множества $C \subseteq E$. Запишем проксимальное отображение для данной функции:

$$\text{Prox}_g(x) = \arg \min_{y \in E} \left\{ \delta_C(y) + \frac{1}{2} \|y - x\|^2 \right\} = \arg \min_{y \in C} \|y - x\|^2 = P_C(x)$$

Таким образом, проксимальный оператор для индикаторной функции данного множества есть оператор ортогональной проекции на данное множество. Если C - замкнутое и выпуклое (а также непустое), то проектирующее отображение существует и единственно в каждой точке.

1.10 Основные матричные разложения, их стоимость вычисления. @bigbluebutterfly

LU-разложение матрицы A :

$$A = LU$$

где L - нижнетреугольная матрица, U - верхнетреугольная матрица. Собственно, разложение определено только для обратимых матриц с невырожденными главными минорами. В общем случае разложение определено неоднозначно - для однозначности можем потребовать унитарность матрицы L или U . Используется для нахождения определителя, решения СЛАУ и обращения матриц. Сложность $\approx \frac{2}{3}n^3$.

Разложение Холецкого:

$$A = LL^T$$

A - симметричная положительно определенная матрица, L - нижнетреугольная матрица (можно также записать как $U^T U$ через верхнетреугольную матрицу). Данное разложение единственно для любой удовлетворяющей условиям матрицы. Применяется для решения СЛАУ - в отличие от LU , здесь требуется примерно в 2 раза меньше итераций ($\frac{1}{3}n^3$), при этом имеет место большая численная устойчивость. Часто применяется для решения задачи наименьших квадратов. Также используется в квази-ньютоновских методах.

Разложение можно немного модифицировать:

$$A = LDL^T$$

D - диагональная матрица, диагональ которой является квадратом диагонали матрицы L в исходном разложении - соответственно, у матрицы L теперь будет единичная диагональ. С таким методом мы избавляемся от вычисления корней диагональных элементов в процессе факторизации. Кроме того, для некоторых неопределенных матриц существует LDL -разложение (в отличие от разложения Холецкого). Сложность сохраняется.

QR-разложение:

$$A = QR$$

где Q - унитарная матрица ($Q^T Q = Q Q^T = E$), R - верхнетреугольная матрица. Разложение существует для любой матрицы (более того, существует даже для прямоугольной размера $m \times n$ ($m > n$) - тогда получаем Q размера $m \times m$ и верхнетреугольную R размера $m \times n$). Разложение чаще всего применяется для решения задачи наименьших квадратов, также используется в одном из алгоритмов для поиска собственных чисел и векторов. Стоимость вычисления $\approx \frac{2}{3}n^3$.

Сингулярное разложение:

$$M = U \Sigma V^T$$

где M - произвольная матрица размера $m \times n$, Σ - диагональная матрица размера $m \times n$, составленная из сингулярных чисел M , матрицы U ($m \times m$) и V ($n \times n$) являются унитарными и составлены из левых и правых сингулярных векторов соответственно. Применяется для нахождения псевдообратной матрицы (используется в МНК) или низкоранговых приближений. Сложность - $O(m^2 n + m n^2 + n^3)$, в квадратном случае справедлива оценка $4n^3$.

Спектральное разложение:

$$A = V \Lambda V^{-1}$$

где A - диагонализуемая квадратная матрица размера $n \times n$, имеющая n линейно независимых собственных векторов, V - матрица из собственных векторов-столбцов для A , Λ - диагональная матрица с соответствующими собственными числами на диагонали. Разложение можно использовать для обращения матриц, решения СЛАУ, нахождения определителя и вычисления функций от матриц (к примеру, матричная экспонента). Сложность $\approx 4n^3$.

1.11 Формулировка теоремы Каруша-Куна-Таккера для общей задачи нелинейного программирования. @Bitchert

Задача оптимизации:

$$\begin{cases} f(x) \rightarrow \min_{x \in D \subset \mathbb{R}^n}, \\ g_i(x) \leq 0, \quad i \in \{1, \dots, m\}, \\ h_j(x) = 0, \quad j \in \{1, \dots, p\}. \end{cases}$$

Доменное множество (совместная область определения, где определены все функции):

$$D = \text{dom } f \cap \left(\bigcap_{i=1}^m \text{dom } g_i \right) \cap \left(\bigcap_{j=1}^p \text{dom } h_j \right)$$

Допустимое множество (все точки из доменного множества, которые удовлетворяют ограничениям g и h):

$$F = \{x \in D \mid g_i(x) \leq 0 \forall i, h_j(x) = 0 \forall j\}$$

Активное множество (множество ограничений, все ограничения которые имеют вид равенства в точке):

$$\text{Active}(x) = \{1, \dots, p\} \cup \{i \mid g_i(x) = 0\}$$

Функция Лагранжа:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x)$$

Условия регулярности:

- **LCQ** все функции g_i, h_j являются невырожденными аффинными
- **LICQ** вектора $\nabla g_i(x), \nabla h_j(x)$ для $i, j \in \text{Active}(x)$ являются линейно независимыми
- **MFCQ** вектора $\nabla g_i(x), \nabla h_j(x)$ для $i, j \in \text{Active}(x)$ являются линейно независимыми с положительными коэффициентами
- **условия Слейтера** для выпуклых задач оптимизации найдется \tilde{x} такое, что $h_j(\tilde{x}) = 0 \forall j$ и $g_i(\tilde{x}) < 0 \forall i$
- **ослабленные условия Слейтера** для выпуклых задач оптимизации найдется \tilde{x} такое, что $h_j(\tilde{x}) = 0 \forall j$ и $g_i(\tilde{x}) \leq 0$ для всех аффинных функций g_i и $g_i(\tilde{x}) < 0$ для остальных

Теорема ККТ:

Пусть в задаче оптимизации $f, g_i, h_j \in C^1$, x_* - точка локального экстремума и выполнены условия регулярности для $\{g_i(x_*), h_j(x_*) \mid i, j \in \text{Active}(x_*)\}$.

Тогда найдутся λ_*, μ_* такие, что:

- $\nabla_x L(x_*, \lambda_*, \mu_*) = \nabla f(x_*) + \sum_{i=1}^m \lambda_i \nabla g_{*,i}(x_*) + \sum_{j=1}^p \mu_{*,j} \nabla h_j(x_*) = 0$ - стационарность функции Лагранжа.
- $x_* \in F$ - прямая допустимость
- $\lambda_{*,i} \geq 0 \forall i$ - двойственная допустимость
- $\lambda_{*,i} g_i(x_*) = 0 \forall i$ - условия дополняющей нежесткости

Она утверждает, что для задач для которых справедливо некоторое условие регулярности, если точка x_* — локальный минимум, то мы можем выписать уравнения стационарности и дополняющей нежесткости которые обязательно разрешимы с некоторыми, неизвестными нам двойственными переменными.

Опциональный материал:

Введем два множества - $\mathcal{S}(x_0)$ множество всех гладких траекторий выходящих из x_0 и остающихся в множестве F ; $\mathcal{T}_F(x_0)$ касательное пространство к множеству F в точке x_0 :

$$\begin{aligned} \mathcal{S}(x_0) &= \{x(t) \mid x(t) \in F \forall t \geq 0; x(0) = x_0; x(t) \in C^1\} \\ \mathcal{T}_F(x_0) &= \left\{ d \mid \exists x(t) \in \mathcal{S}(x_0), d = \left. \frac{\partial x}{\partial t} \right|_{t=0} \right\} \end{aligned}$$

Теорема ККТ второго порядка:

Пусть в задаче оптимизации $f, g_i, h_j \in C^2$, x_* - точка локального минимума и выполнены условия регулярности для $\{g_i(x_*), h_j(x_*) \mid i, j \in \text{Active}(x_*)\}$.

Тогда найдутся λ_*, μ_* такие, что:

- $\nabla_x L(x_*, \lambda_*, \mu_*) = \nabla f(x_*) + \sum_{i=1}^m \lambda_i \nabla g_{*,i}(x_*) + \sum_{j=1}^p \mu_{*,j} \nabla h_j(x_*) = 0$ - стационарность функции Лагранжа.
- $x_* \in F$ - прямая допустимость
- $\lambda_{*,i} \geq 0 \forall i$ - двойственная допустимость
- $\lambda_{*,i} g_i(x_*) = 0 \forall i$ - условия дополняющей нежесткости
- $d^T \nabla_{xx}^2 L(x_*, \lambda_*, \mu_*) d \geq 0 \forall d \in \mathcal{T}_F(x_*)$

Если выполнены условия регулярности, то можно записать $\mathcal{T}_F(x_0)$ в виде:

$$\mathcal{T}_F(x_0) = \{d \mid \nabla g_i(x_0)^T d \leq 0, \nabla h_j(x_0)^T d \leq 0 \forall i, j \in \text{Active}(x_0)\}$$

1.12 Каноническая форма задачи линейного программирования. Пример сведения общей задачи ЛП к задаче в канонической форме. @Bitchert

Задача линейного программирования в общем виде:

$$\begin{cases} c^T x \rightarrow \min_x \\ Ax = b \\ Ex \leq d \end{cases}$$

Задача линейного программирования в канонической форме:

$$\begin{cases} c^T x \rightarrow \min_{x \in \mathbb{R}^n} \\ Ax = b, \quad A \in \mathbb{R}^{p \times n}, p < n, \text{Rk}(A) = p \\ x \geq 0 \end{cases}$$

Эквивалентные преобразования для сведения задачи к канонической форме:

- $Ex \leq d \Leftrightarrow \begin{cases} Ex + f = d \\ f \geq 0 \end{cases}$
- $Ax = b \Leftrightarrow \begin{cases} A(x^+ - x^-) = b \\ x = x^+ - x^-, \quad x^+ = \max(x, 0), \quad x^- = \max(-x, 0) \\ x^+, x^- \geq 0 \end{cases}$

Пример сведения:

$$\begin{cases} c^T x \rightarrow \min_x \\ Ax = b \\ Ex \leq d \end{cases} \Leftrightarrow \begin{cases} c^T(x^+ - x^-) \rightarrow \min_{x^+, x^-} \\ A(x^+ - x^-) = b \\ E(x^+ - x^-) + f = d \\ x^+, x^-, f \geq 0 \end{cases}$$

1.13 Двойственная задача Лагранжа и ее основные свойства. Пример построения двойственной задачи. @Bitchert

Задача условной оптимизации:

$$(P) \begin{cases} f(x) \rightarrow \min_x \\ g_i(x) \leq 0, i \in \{1, \dots, m\} \\ h_i(x) = 0, j \in \{1, \dots, p\} \end{cases} \quad f, g_i, h_j \in C^1$$

Функция Лагранжа:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^p \mu_j h_j(x)$$

Двойственная функция Лагранжа:

$$q(\lambda, \mu) := \inf_x L(x, \lambda, \mu)$$

Свойства двойственной функции:

- q - вогнутая функция
- $q(\lambda, \mu) \leq f_{opt}$, для $\lambda \geq 0, \mu$

Двойственная задача оптимизации:

$$(D) \begin{cases} q(\lambda, \mu) \rightarrow \max_{\lambda, \mu} \\ \lambda \geq 0 \end{cases}$$

Двойственность:

- **Слабая** двойственность: $q_{opt} \leq f_{opt}$
- **Сильная** двойственность: $q_{opt} = f_{opt}$

Если выполнено условие регулярности и (P) является выпуклой задачей оптимизации, то выполняется сильная двойственность. Пример построения двойственной задачи:

Исходная задача:

$$\begin{cases} c^T x \rightarrow \min_{x \in \mathbb{R}^n} \\ Ax \leq b, \quad A \in \mathbb{R}^{m \times n}, \text{Rk}(A) = \min(m, n) \end{cases}$$

Лагранжиан:

$$L(x, \lambda) = c^T x + \lambda^T (Ax - b) = (c + A^T \lambda)^T x - \lambda^T b$$

Двойственная функция:

$$q(\lambda) = \inf_{x \in \mathbb{R}^n} L(x, \lambda)$$

Двойственная задача:

- $c + A^T \lambda = 0$:
$$\begin{cases} q(\lambda) = -\lambda^T b \rightarrow \max_{y \in \mathbb{R}^m} \\ c + A^T \lambda = 0 \\ \lambda \geq 0 \end{cases}$$
- $c + A^T \lambda \neq 0$: $q(\lambda) = -\infty$

1.14 Общая схема какого-то метода оптимизации (итерация, выбор длины шага, критерий останова), например, градиентного спуска, проксимального градиентного метода, метода SAG, метода логарифмических барьеров и проч.

2 Билеты

2.1 Матрично-векторное дифференцирование в терминах дифференциалов. Примеры вычисления градиентов и гессианов для матричных функций.

Все написано [здесь](#).

2.2 Решение задачи одномерной оптимизации (метод золотого сечения, парабол, Брента). Условия для неточной одномерной оптимизации. @sofload

Рассмотрим задачу одномерной оптимизации вида

$$f(x) \rightarrow \min_x, x \in \mathbb{R}, \quad (1)$$

где функция $f: \mathbb{R} \rightarrow \mathbb{R}$ является непрерывной и унимодальной. Пусть x_{\min} - это решение задачи 1 и нам известен интервал $[a, b]$, в котором содержится x_{\min} . Предположим также, что у нас есть **оракул** нулевого порядка, т.е. в произвольной точке x возможно вычисление значения функции $f(x)$ и невозможно вычисление любых производных.

Пусть значение функции измерено в двух точках x, y внутри интервала $[a, b]$ (см. рисунок 2.2, слева). Тогда предположение об унимодальности функции позволяет сократить интервал поиска x_{\min} путем сравнения значений $f(x)$ и $f(y)$.

Если $f(y) \geq f(x)$, тогда $[a, b] \rightarrow [a, y]$, иначе $[a, b] \rightarrow [x, b]$.

По умолчанию в методах оптимизации предполагается, что обращение к оракулу является дорогостоящей операцией. Поэтому при сокращении интервала поиска, например, до $[a, y]$ желательно использовать точку x в качестве одной из двух точек внутри интервала $[a, y]$ для дальнейшего прогресса итерационного процесса оптимизации. В этом случае на каждой итерации требуется только одно обращение к оракулу. Кроме того, на каждой итерации поддерживается тройка точек таких, что значение функции в средней точке меньше, чем значение функции в крайних точках.

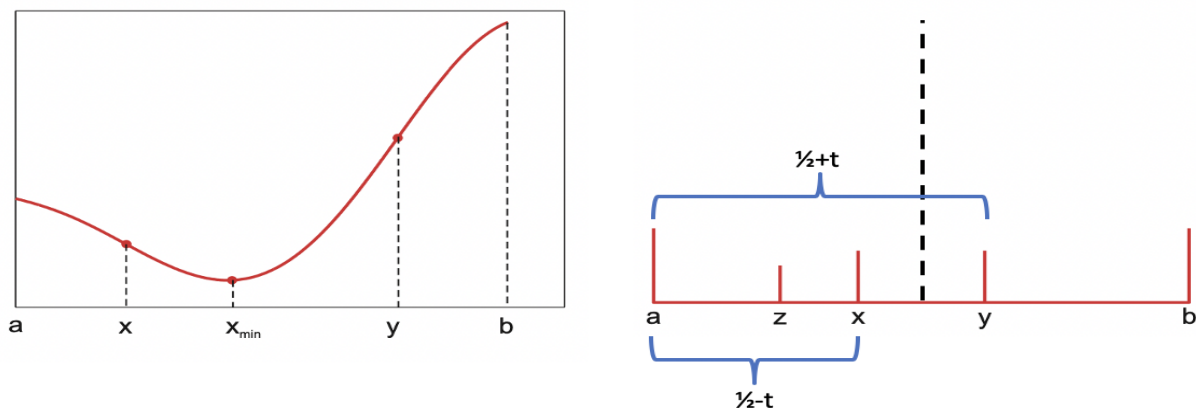


Рис. 5: Слева: унимодальная функция, справа: точки, выбираемые по золотому сечению

2.2.1 Метод золотого сечения

В методе золотого сечения точки внутри интервалов поиска решения выбираются исходя из следующих предложений:

- на текущей итерации оба потенциальных интервала сокращения равны между собой;
- на каждой итерации интервал сокращается на одну и ту же величину.

Эти два предложения позволяют однозначно определить процесс генерации новых точек. Предположим, что текущий интервал поиска $[a, b]$ имеет единичную длину и в нём выбраны две точки x, y (см. рисунок 2.2, справа). Равенство потенциальных интервалов сокращения означает, что длина $[a, x]$ равна $1/2 - t$, а длина $[a, y] - 1/2 + t$, где t - некоторая неизвестная величина.

Пусть на текущей итерации выбирается интервал $[a, y]$ и на следующей итерации к точке x добавляется точка z . Тогда предложение о сокращении интервалов на фиксированную величину означает, что:

$$\frac{l[a, y]}{l[a, b]} = \frac{l[a, x]}{l[a, y]} = K,$$

где через $l[a, y]$ обозначена длина интервала $[a, y]$. В результате получаем:

$$\frac{1/2 + t}{1} = \frac{1/2 - t}{1/2 + t} \Rightarrow t = \frac{\sqrt{5} - 2}{2} \Rightarrow K = \frac{\sqrt{5} - 1}{2}$$

Схема метода золотого сечения представлена в Алгоритме 1 (см. рисунок 2.2.1). В этой схеме границы

Алгоритм 1: Схема метода золотого сечения

Вход: Интервал $[x_{L,0}, x_{U,0}]$, точность ε ;
Выход: Точка и значение минимума x_{min}, f_{min} ;
 Инициализация $K = (\sqrt{5} - 1)/2$, $I_0 = x_{U,0} - x_{L,0}$;
 $I_1 = KI_0$, $x_{b,1} = x_{L,0} + I_1$, $x_{a,1} = x_{U,0} - I_1$;
 Вычислить $f_{a,1} = f(x_{a,1})$, $f_{b,1} = f(x_{b,1})$;
для $k = 1, \dots, \#iter$
 $I_{k+1} = KI_k$;
 если $f_{a,k} \geq f_{b,k}$ **то**
 $x_{L,k+1} = x_{a,k}$, $x_{U,k+1} = x_{U,k}$, $x_{a,k+1} = x_{b,k}$, $x_{b,k+1} = x_{L,k+1} + I_{k+1}$;
 $f_{a,k+1} = f_{b,k}$, $f_{b,k+1} = f(x_{b,k+1})$;
 иначе
 $x_{L,k+1} = x_{L,k}$, $x_{U,k+1} = x_{b,k}$, $x_{a,k+1} = x_{U,k+1} - I_{k+1}$, $x_{b,k+1} = x_{a,k}$;
 $f_{a,k+1} = f(x_{a,k+1})$, $f_{b,k+1} = f_{a,k}$;
 если $I_{k+1} < \varepsilon$ **то**
 если $f_{a,k+1} < f_{b,k+1}$ **то**
 $f_{min} = f_{a,k+1}$, $x_{min} = x_{a,k+1}$;
 иначе
 $f_{min} = f_{b,k+1}$, $x_{min} = x_{b,k+1}$;
 Выход из цикла;

Рис. 6: Алгоритм 1: Схема метода золотого сечения

интервала на итерации k обозначены как $x_{L,k}$ и $x_{U,k}$, длина интервала как I_k , а две внутренние точки интервала как $x_{a,k}$ и $x_{b,k}$. Очевидно, что длина интервала $I_k = KI_{k-1} = K^2 I_{k-2} = \dots = K^k I_0$, где I_0 – длина начального интервала, заданного пользователем. Поэтому **скорость сходимости** метода золотого сечения является **линейной** с константой K . Кроме того, из условия $K^k I_0 < \varepsilon$ можно определить количество итераций, которые потребуются методу для достижения точности ε . Заметим также, что в методе золотого сечения оптимизируемая функция f **не вычисляется в крайних точках** $x_{L,0}$ и $x_{U,0}$. Это удобно, например, для оптимизации функций с вертикальными асимптотами.

2.2.2 Метод парабол

В методе парабол предлагается аппроксимировать оптимизируемую функцию $f(x)$ с помощью квадратичной функции:

$$p(x) = ax^2 + bx + c.$$

Пусть имеются три точки $x_1 < x_2 < x_3$ такие, что интервал $[x_1, x_3]$ содержит точку минимума функции f . Тогда коэффициенты аппроксимирующей параболы a, b, c могут быть найдены путем решения системы линейных уравнений: $ax_i^2 + bx_i + c = f_i = f(x_i)$, $i = 1, 2, 3$. Минимум такой параболы: $u = -\frac{b}{2a}$.

Если $f(x_2) < f(x_1)$ и $f(x_2) < f(x_3)$, то точка u гарантированно попадает в интервал $[x_1, x_3]$. (см. рисунок 2.2.2, слева).

Если $f(u) \leq f(x_2)$, то $[x_1, x_3] \rightarrow [x_1, x_2]$, иначе $[x_1, x_3] \rightarrow [u, x_3]$.

В отличие от метода золотого сечения, метод парабол обладает **суперлинейной скоростью сходимости**. Однако, такая высокая скорость сходимости гарантируется только **в малой окрестности точки минимума** x_{min} . На начальных стадиях процесса оптимизации метод парабол может делать очень маленькие шаги или, наоборот, слишком большие шаги, приводящие к неустойчивым биениям (см. рисунок 2.2.2, справа). Также следует отметить, что на первой итерации метод парабол **требует измерения значений функции f в крайних точках** интервала оптимизации.

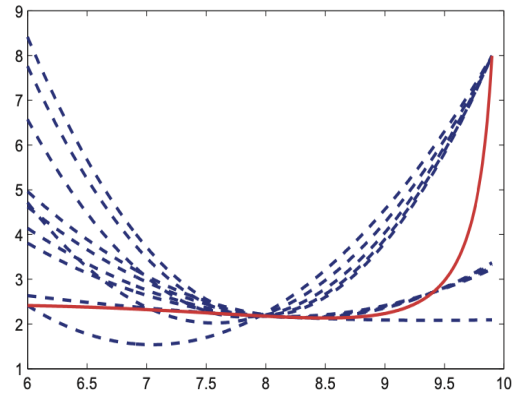
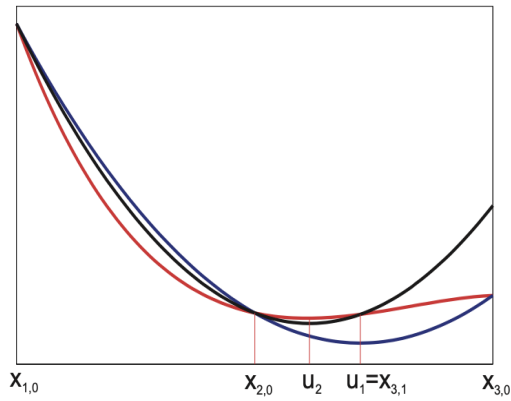


Рис. 7: Иллюстрация работы метода парабол. Слева: первые две итерации метода, красная кривая – оптимизируемая функция, синяя кривая – квадратичное приближение на первой итерации, черная кривая – квадратичное приближение на второй итерации. Справа: пример плохой сходимости метода парабол, красная кривая – оптимизируемая функция, синие кривые – итерационные квадратичные приближения.

2.2.3 Комбинированный метод Брента

Метод золотого сечения представляет собой надежный способ оптимизации, который сходится за гарантированное число итераций, но обладает лишь линейной скоростью сходимости. Метод парабол работает быстрее в малой окрестности оптимального решения, но может работать долго и неустойчиво на начальных стадиях итерационного процесса. Поэтому на практике для решения задачи одномерной оптимизации используется метод Брента, который эффективно комбинирует эти две стратегии.

В данном методе на каждой итерации отслеживаются значения в шести точках (не обязательно различных): a, c, x, w, v, u .

Точки a, c задают текущий интервал поиска решения,
 x – точка, соответствующая наименьшему значению функции,
 w – точка, соответствующая второму снизу значению функции,
 v – предыдущее значение w .

В отличие от метода парабол, в методе Брента аппроксимирующая парабола строится с помощью трех наилучших точек x, w, v (в случае, если эти три точки различны и значения в них также различны). При этом минимум аппроксимирующей параболы u принимается в качестве следующей точки оптимизационного процесса, если:

- u попадает внутрь интервала $[a, c]$;
- u отстоит от точки x не более, чем на половину от длины предыдущего шага.

Если точка u отвергается, то следующая точка находится с помощью золотого сечения большего из интервалов $[a, x]$ и $[x, c]$. Итоговая схема Брента представлена в Алгоритме 2. (см. рисунок 2.2.3)

Приведем некоторые аргументы в пользу обозначенных условий приема минимума параболы u . Так как парабола на текущей итерации проводится через точки x, w, v , для которых не гарантируются соотношения $v < x < w$ или $w < x < v$, то минимум параболы может оказаться вне интервала $[a, c]$. Ограничение на максимальную удаленность u от x позволяет избежать слишком больших шагов в оптимизации, которые могут соответствовать биениям в методе парабол. Использование в данном ограничении длины предыдущего шага, а не предыдущего, является эвристикой, эффективность которой подтверждается в экспериментах на больших базах задач оптимизации. Эта эвристика предлагает не штрафовать метод за текущий не слишком удачный маленький шаг в надежде на успешные шаги метода на следующих итерациях.

2.2.4 Неточная одномерная оптимизация

Во многих методах многомерной оптимизации на итерации k имеется текущая точка $x_k \in \mathbb{R}^n$ и некоторое направление минимизации $d_k \in \mathbb{R}^n$. Тогда следующая точка итерационного процесса x_{k+1} находится путем решения следующей задачи одномерной оптимизации:

$$\phi(\alpha) = f(x_k + \alpha d_k) \rightarrow \min_{\alpha \geq 0}. \quad (2)$$

Вход: Интервал оптимизации (a, c) , точность ε ;
Выход: Точка и значение минимума x_{min}, f_{min} ;
Инициализация $K = (3 - \sqrt{5})/2$, $x = w = v = a + K(c - a)$, $f_x = f_w = f_v = f(x)$;
Инициализация длины текущего и предыдущего шага $d = e = c - a$;
пока Итерации до сходимости
 $g = e, e = d$;
 $tol = \varepsilon|x| + \frac{\varepsilon}{10}$;
если Выполнен критерий останова: $|x - \frac{a+c}{2}| + \frac{c-a}{2} \leq 2tol$ **то**
Выход из цикла;
если Точки x, w, v и значения f_x, f_w, f_v – разные **то**
Параболическая аппроксимация, находим u – минимум параболы;
если $u \in [a, c]$ и $|u - x| < g/2$ **то**
Принимаем u ;
если $u - a < 2tol$ или $c - u < 2tol$ **то**
 $u = x - \text{sign}(x - (a + c)/2)tol$;
если Парабола не принята **то**
если $x < (a + c)/2$ **то**
 $u = x + K(c - x)$; // Золотое сечение $[x, c]$;
 $e = c - x$;
иначе
 $u = x - K(x - a)$; // Золотое сечение $[a, x]$;
 $e = x - a$;
если $|u - x| < tol$ **то**
 $u = x + \text{sign}(u - x)tol$; // Задаём минимальную близость между u и x
 $d = |u - x|$;
Вычисляем $f_u = f(u)$;
если $f_u \leq f_x$ **то**
если $u \geq x$ **то**
 $a = x$;
иначе
 $c = x$;
 $v = w, w = x, x = u, f_v = f_w, f_w = f_x, f_x = f_u$;
иначе
если $u \geq x$ **то**
 $c = u$;
иначе
 $a = u$;
если $f_u \leq f_w$ или $w = x$ **то**
 $v = w, w = u, f_v = f_w, f_w = f_u$;
иначе если $f_u \leq f_v$ или $v = x$ или $v = w$ **то**
 $v = u, f_v = f_u$;

Рис. 8: Алгоритм 2: Комбинированный метод Брента

Эту задачу можно решать с помощью методов, описанных выше. Однако, для сходимости общего процесса многомерной оптимизации задачу 2 зачастую не обязательно решать точно. На практике здесь оказывается достаточным лишь значимо уменьшить значение функции на текущей итерации. Отказ от точного решения 2 позволяет во многих случаях сократить количество обращений к оракулу.
Итак, дано:

$$\phi(\alpha) = f(x_k + \alpha d_k)$$

$$\phi'(0) = \nabla f(x_k)^T d_k < 0$$

Условия неточной одномерной оптимизации:

1. **Условие Армихо.** Гарантирует уменьшение значения функции на текущей итерации (условие, значимого уменьшения). См. рисунок 2.6 слева.

$$\phi(\alpha_k) \leq \phi(0) + c_1 \alpha_k \phi'(0), c_1 \in (0, 1)$$

2. **Условие Вольфа.** Есть два условия - слабое и сильное, мы используем одно из двух. См. рисунок 2.6 справа.

- а. (слабое условие) $\phi'(\alpha_k) \geq c_2 \phi'(0)$, $c_2 \in (0,1)$
- б. (сильное условие) $|\phi(\alpha_k)| \leq c_2 |\phi'(0)|$, $c_2 \in (0,1)$

Сильное условие Вольфа позволяет эффективно локализовывать минимум: если константу c_2 начинаем стремить к 0, мы все сильнее сжимаем интервал (на рис. между голубой и красной точками) и гарантируем то, что α_k , удовлетворяющая сильному условию, будет приближенным точным оптимумом при $c_2 \rightarrow 0$.

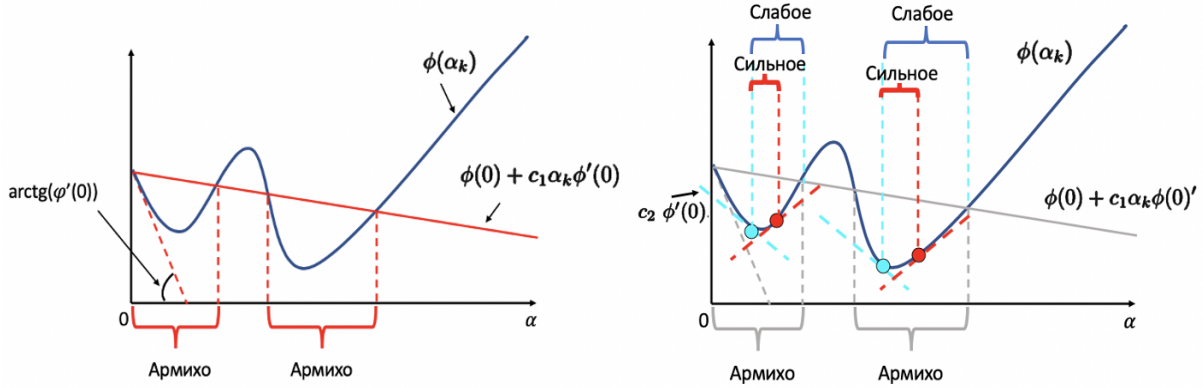


Рис. 9: Иллюстрация достаточных условий для неточной одномерной минимизации. Слева: условие Армихо, справа: слабое/сильное условие Вольфа.

На практике берут $c_1 = 10^{-4}$, $c_2 = 0.9$ (если интересует незначительное уменьшение функции) или $c_2 = 0.1$ (если важно по сильнее минимизировать функцию)

Теорема. Если $\phi(\alpha) \in C^1$, $\phi'(0) < 0$, $\phi(\alpha) > -\infty \forall \alpha$ и $0 < c_1 \leq c_2 < 1 \Rightarrow \exists \alpha_*$, удовлетворяющая условию Армихо и слабому или сильному условию Вольфа.

Доказательство.

$$\phi(\alpha) = \phi(0) + \phi'(0)\alpha + o(\alpha^2) \leq \phi(0) + c_1 \alpha \phi'(0)$$

$$\phi'(0) + o(\alpha) \leq c_1 \phi'(0)$$

$$\underbrace{(1 - c_1) \phi'(0)}_{>0} + \underbrace{o(\alpha)}_{<0} \leq 0$$

$$\Rightarrow \exists \alpha_* : \forall \alpha \in (0, \alpha_*) \text{ условие Армихо выполнено.}$$

$$\exists \hat{\alpha} : \phi(\hat{\alpha}) = \phi(0) + c_1 \hat{\alpha} \phi'(0)$$

$$\psi(\alpha) := \phi(\alpha) - \phi(0) - c_1 \alpha \phi'(0)$$

$$\psi(0) = 0; \psi(\hat{\alpha}) = 0 \text{ и т. к. } \alpha_* \text{ лежит между } 0 \text{ и } \hat{\alpha} \Rightarrow \exists \alpha_* : \psi'(\alpha_*) = 0$$

$$\psi'(\alpha_*) = \phi'(\alpha_*) - c_1 \phi'(0) = 0$$

$$\Rightarrow \phi'(\alpha_*) = c_1 \underbrace{\phi'(0)}_{<0} \geq c_2 \phi'(0)$$

$$\Rightarrow \alpha_* \text{ удовлетворяет слабому условию Вольфа.}$$

$$|\phi'(\alpha_*)| = |c_1 \phi'(0)| \leq c_2 |\phi'(0)|$$

$$\Rightarrow \alpha_* \text{ удовлетворяет сильному условию Вольфа.}$$

Алгоритм поиска α_* .

1. Расширение (поиск правой границы)

Пользователь задает начальную точку α_0 . Сначала нужно найти интервал, в котором находится искомая точка, т.е. берется точка α_0 и умножается на некоторую константу (сдвигается вправо) пока не будет выполнено одно из условий для правой границы.

2. Сужение

Теперь интервал нужно сужать:

- i. Внутри текущего интервала генерируем (через метод парабол или золотого сечения) какую-то точку
- ii. Проверяем для нее условия Армихо и Вольфа, если они не выполнены, то сдвигаем интервал налево или направо по условиям для правой и левой границ.
- iii. Сужаем интервал пока не найдем нужную точку.

Условия для правой границы:

1. Не выполнено условие Армихо
2. $\phi'(\alpha) > 0$ и не выполнено условие Вольфа
3. $\phi(\alpha) > \phi(\alpha_{prev})$

Условие для левой границы: в левой точке $\phi'(\alpha) < 0$

2.3 Метод градиентного спуска. Стратегии выбора длины шага. Скорость сходимости метода для сильно выпуклых функций, примеры быстрой и медленной работы метода. @juliasemavina

Обозначения:

- $C_L^{p,k}(Q)$ - класс функций со следующими свойствами:
 1. $f(x)$ k раз непрерывно дифференцируема на Q .
 2. p -ая производная функции f удовлетворяет условию Липшица на Q с константой L :

$$\forall x, y \in Q \quad \|f^{(p)}(x) - f^{(p)}(y)\| \leq L\|x - y\|$$

- $\mathcal{F}^1(Q)$ - класс функций, выпуклых на множестве Q .
- $\mathcal{F}_L^{p,k}(Q)$ - подкласс выпуклых на Q функций класса $C_L^{p,k}(Q)$.
- $\mathcal{F}_\mu^1(Q)$ - класс функций, сильно выпуклых на множестве Q с коэффициентом μ .
- $\mathcal{F}_{\mu,L}^{p,k}(Q)$ - подкласс сильно выпуклых с коэффициентом μ на Q функций класса $C_L^{p,k}(Q)$.

Будем решать задачу гладкой выпуклой безусловной минимизации:

$$\min_{x \in \mathbb{R}^n} f(x)$$

где $f \in C_L^{1,1}(\mathbb{R}^n)$, то есть f - непрерывно дифференцируемая с липшицевой первой производной.

2.3.1 Метод градиентного спуска

Алгоритм 1 Общая схема метода спуска

Вход: Начальная точка x_0 ; максимальное число итераций K .

- 1: **for** $k \leftarrow 0$ **to** K **do**
- 2: *(Вызов оракула)* Вычислить $f(x_k)$, $\nabla f(x_k)$ и пр.
- 3: *(Критерий остановки)* Если выполнен критерий остановки, то выход.
- 4: *(Вычисление направления)* Вычислить направление спуска d_k .
- 5: *(Линейный поиск)* Найти подходящую длину шага α_k .
- 6: *(Обновление)* $x_{k+1} \leftarrow x_k + \alpha_k d_k$.
- 7: **end for**

Выход: Последняя вычисленная точка x_k

Рис. 10: Метод спуска

В случае градиентного спуска достаточно двух оракулов $f(x_k)$ и $\nabla f(x_k)$, а в качестве направления выбирается направление наискорейшего спуска: $d_k = -\nabla f_k(x)$.

2.3.2 Стратегии выбора длины шага (α_k)

Мы решаем подзадачу нахождения $x_{k+1} = x_k - \alpha_k d_k$ в методе градиентного спуска. Нам бы хотелось найти такую точку x_{k+1} которая бы давала значительное уменьшение целевой функции. То есть наша задача состоит в минимизации функции

$$\phi_k(\alpha_k) = f(x_k + \alpha_k d_k)$$

Рассмотрим различные стратегии выбора длины шага:

1. Последовательность $\{\alpha_k\}$ задана заранее.

Например:

- $\alpha_k = \alpha > 0$ - константный шаг. Не гарантирует сходимость для всех гладких выпуклых функций. Сходимость гарантируется, если $\alpha \leq \frac{1}{L}$ и функция имеет липшицев градиент.
- $\alpha_k = \frac{\alpha}{\sqrt{k+1}}$
- $\{\alpha_k\}$ такова, что ряд $\sum \alpha_k$ расходится, а ряд $\sum \alpha_k^2$ сходится. (ШАД 2019 Введение в методы оптимизации). Такие последовательности обеспечивают сходимость метода градиентного спуска для любых выпуклых гладких функций.

Применение: Очень простой метод, чаще всего применяется для задач выпуклой оптимизации.

2. Полная релаксация - $\alpha_k = \arg \min_{\alpha > 0} f(x_k - \alpha \nabla f(x_k))$

В этом случае можно использовать любые методы одномерной оптимизации, однако, точно эта задача обычно не решается.

Применение: Имеет исключительно теоретический интерес, не применяется на практике.

3. Бэктрекинг - Метод дробления шага. Уменьшаем длину шага в фиксированное количество раз, пока не выполняется некоторое условие. В качестве условий часто применяются условия Армихо и Вульфа.

Условие Армихо: $\phi_k(\alpha_k) \leq \phi_k(0) + c_1 \alpha \phi'_k(0)$, $c_1 \in (0, 1)$

Условия Вульфа: $\phi_k(\alpha_k) \leq \phi_k(0) + c_1 \alpha \phi'_k(0)$, $\phi'_k(\alpha_k) \geq c_2 \alpha \phi'_k(0)$, $0 < c_1 < c_2 < 1$

Сильные условия Вульфа:

$$\phi_k(\alpha_k) \leq \phi_k(0) + c_1 \alpha \phi'_k(0), |\phi'_k(\alpha_k)| \leq c_2 \alpha |\phi'_k(0)|, 0 < c_1 < c_2 < 1$$

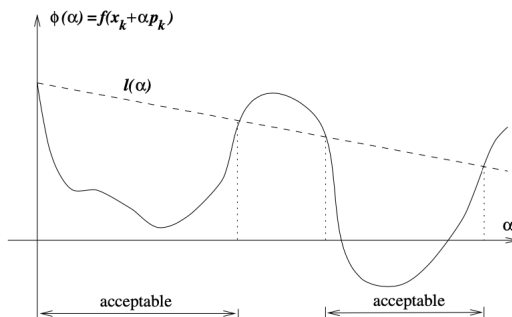


Рис. 11: Иллюстрация к условию Армихо: Накладывает условие достаточного уменьшения функции.

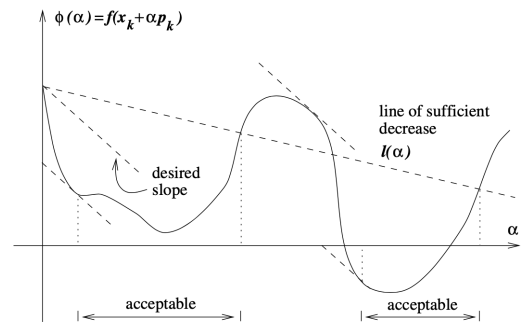


Рис. 12: Иллюстрация к условиям Вульфа: условие Армихо и условие кривизны функции: наклон не должен быть меньше начального, умноженного на константу.

Алгоритм 2 Метод дробления шага

Вход: Функция $\phi_k : \mathbb{R}_+ \rightarrow \mathbb{R}$. Начальная точка: $\alpha_k^{(0)}$.

1: $\alpha \leftarrow \alpha_k^{(0)}$.

2: **while** $\phi_k(\alpha) > \phi(0) + c_1 \alpha \phi'_k(0)$ **do**

3: $\alpha \leftarrow \alpha/2$.

4: **end while**

Выход: α

Рис. 13: Процедура бэктрекинга с условием Армихо

2.3.3 Скорость сходимости метода для сильно выпуклых функций

Лемма 1 Пусть $f \in \mathcal{F}_{\mu,L}^{1,1}(\mathbb{R}^n)$. Тогда $\forall x, y \in \mathbb{R}^n$:

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L} \|x - y\|^2 + \frac{1}{\mu + L} \|\nabla f(x) - \nabla f(y)\|^2$$

Лемма 2 Пусть $f \in \mathcal{F}_L^{1,1}(\mathbb{R}^n)$. Тогда $\forall x, y \in \mathbb{R}^n$:

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x \rangle \leq \frac{L}{2} \|x - y\|^2$$

Теорема (Скорость сходимости градиентного метода): Пусть $f \in \mathcal{F}_{\mu,L}^{1,1}(\mathbb{R}^n)$, $0 < h \leq \frac{2}{\mu+L}$. Тогда градиентный метод с константным шагом h образует такую последовательность $\{x_k\}$, что

$$\|x_k - x^*\|^2 \leq \left(1 - \frac{2h\mu L}{\mu + L}\right)^k \|x_0 - x^*\|^2$$

При этом, если $h = \frac{2}{\mu+L}$, обозначив $Q_f = \frac{L}{\mu}$:

$$\|x_k - x^*\| \leq \left(\frac{Q_f - 1}{Q_f + 1}\right)^k \|x_0 - x^*\|$$

$$f(x_k) - f^* \leq \frac{L}{2} \left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} \|x_0 - x^*\|^2$$

Доказательство:

Обозначим: $r_k = \|x_k - x^*\|$.

$$\begin{aligned} r_{k+1}^2 &= \|x_{k+1} - x^*\|^2 = \left\| x_{k+1} = x_k - h \nabla f(x_k) \right\|^2 \\ &= \|x_k - x^* - h \nabla f(x_k)\|^2 = \langle x_k - x^* - h \nabla f(x_k), x_k - x^* - h \nabla f(x_k) \rangle = \\ &= r_k^2 - 2h \langle x_k - x^*, \nabla f(x_k) \rangle + h^2 \|\nabla f(x_k)\|^2 \leq \left[\text{Лемма 1, } \nabla f(x^*) = 0 \right] \\ &\leq \left(1 - \frac{2h\mu L}{\mu + L}\right) r_k^2 + h \left(h - \frac{2}{\mu + L}\right) \|\nabla f(x_k)\|^2 \leq \left[h \leq \frac{2}{\mu + L} \right] \\ &\leq \left(1 - \frac{2h\mu L}{\mu + L}\right) r_k^2 \leq \left[\text{индукция} \right] \\ &\leq \left(1 - \frac{2h\mu L}{\mu + L}\right)^{k+1} \|x_0 - x^*\|^2 \end{aligned}$$

Чтобы получить результаты при $h = \frac{2}{\mu+L}$, необходимо подставить значение h в формулу, получить первое неравенство и из него при помощи леммы 2 получить второе.

- Отдельно отметим, что для класса $\mathcal{F}_L^{p,k}(Q)$ у градиентного метода с константным шагом $\alpha = \frac{1}{L}$ сходимость градиентного метода сублинейная.

2.3.4 Примеры быстрой и медленной работы метода.

Вспомним полученную нами скорость сходимости:

$$Q_f = \frac{L}{\mu}:$$

$$\|x_k - x^*\| \leq \left(\frac{Q_f - 1}{Q_f + 1} \right)^k \|x_0 - x^*\|$$

Это означает, что скорость сходимости линейна, но зависит от Q_f , а именно, при $Q_f \gg 1$ (Q_f сильно больше 1) сходимость метода очень медленная: при $Q_f = 100$: $\frac{Q_f - 1}{Q_f + 1} \approx 0.98$, $\left(\frac{Q_f - 1}{Q_f + 1} \right)^{30} \approx 0.55$.

При $Q_f \approx 1$ сходимость метода быстрая: при $Q_f = 2$: $\frac{Q_f - 1}{Q_f + 1} = 0.33$, $\left(\frac{Q_f - 1}{Q_f + 1} \right)^{10} \approx 0.000015$.

Кроме того, Q_f - оценка числа обусловленности гессиана f .

- Число обусловленности матрицы $\mu(A) = \|A\| \times \|A^{-1}\|$, $\|\cdot\|$ - любая операторная форма - то есть число обусловленности зависит от выбора нормы.

Пример операторной нормы: $\|A\| = \max\{\|Ax\| : \|x\| = 1\}$.

Получается, что чем ближе число обусловленности к 1, тем лучше работает метод. В геометрическом смысле: при числе обусловленности, близком к 1, метод работает хорошо, а при больших числах обусловленности требуется сильно больше итераций:

В примерах $f(x) = \frac{1}{2} \langle Ax, x \rangle - \langle b, x \rangle$

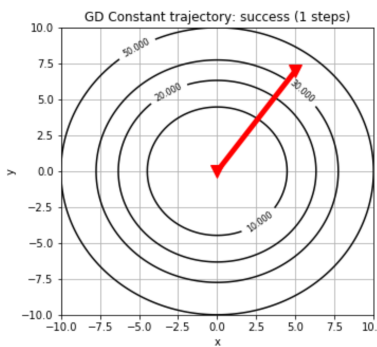


Рис. 14: Число обусловленности A : 1



Рис. 15: Число обусловленности A : 2

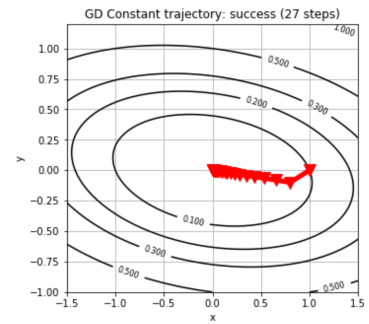


Рис. 16: Число обусловленности A : 6.4

2.4 Метод Ньютона, его локальная и глобальная скорость сходимости. Модификации метода Ньютона для невыпуклых задач оптимизации. Примеры быстрой и медленной работы метода. @Bitchert

2.4.1 Метод Ньютона

Рассмотрим нашу функцию в окрестности точки x_k , приблизим её некоторой квадратичной моделью $m_k(d_k)$:

$$f(x_k + d_k) \approx m_k(d_k) = f(x_k) + g_k^T d_k + \frac{1}{2} d_k^T B_k d_k \rightarrow \min_{d_k}$$

Для минимизации находим градиент модели и приравняем к нулю:

$$\nabla_{d_k} m_k(d_k) = g_k + B_k d_k = 0 \Rightarrow d_k = -B_k^{-1} g_k$$

Соответственно, если:

- $g_k = \nabla f(x_k)$, $B_k = \gamma_k I$ - получаем градиентный спуск
- $g_k = \nabla f(x_k)$, $B_k = \nabla^2 f(x_k)$ - получаем метод Ньютона

Для метода Ньютона соответственно $f \in C^2$, и было бы неплохо, если $\nabla^2 f(x) \succ 0$, чтобы мы вообще куда-то оптимизировали:

$$d_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \Rightarrow \frac{\partial}{\partial \alpha} f(x_k + \alpha d_k) \Big|_{\alpha=0} = \nabla f(x_k)^T d_k = -\nabla f(x_k)^T [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

Последнее выражение, очевидно, меньше нуля для любых точек если $\nabla^2 f(x) \succ 0$.

Далее мы покажем глобальную сходимость метода Ньютона, его локальную сходимость и локальную скорость сходимости.

2.4.2 Глобальная сходимость метода Ньютона

Запишем условия Вульфа:

$$\begin{cases} f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T d_k \\ \nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \nabla f_k^T d_k \end{cases} \quad \text{где } 0 < c_1 < c_2 < 1$$

Косинус:

$$\cos \Theta_k = \frac{-\nabla f_k^T d_k}{\|\nabla f_k\| \|d_k\|}$$

Положим, что мы совершаем на каждом шагу итерации вида $x_{k+1} := x_k + \alpha_k d_k$. Положим также, что f ограничена снизу на \mathbb{R}^n и $f \in C_L^{1,1}$. Тогда:

$$\sum_{k \geq 0} \cos^2 \Theta_k \|\nabla f_k\|^2 < \infty$$

Доказательство:

$$\begin{aligned} \nabla f(x_k + \alpha_k d_k)^T d_k - \nabla f_k^T d_k &\geq (c_2 - 1) \nabla f_k^T d_k \\ (\nabla f_{k+1} - \nabla f_k)^T d_k &\geq (c_2 - 1) \nabla f_k^T d_k \end{aligned}$$

Так как у f липшицевый градиент:

$$(\nabla f_{k+1} - \nabla f_k)^T d_k \leq \alpha_k L \|d_k\|^2$$

Тогда, совмещая два предыдущих неравенства:

$$\alpha_k \geq \frac{c_2 - 1}{L} \frac{\nabla f_k^T d_k}{\|d_k\|^2}$$

Подставляем в условие Вульфа (первое) и получаем:

$$f_{k+1} \leq f_k - c_1 \frac{1 - c_2}{L} \frac{[\nabla f_k^T d_k]^2}{\|d_k\|^2} = f_k - c_1 \frac{1 - c_2}{L} \cos^2 \Theta_k \|\nabla f_k\|^2$$

Теперь просуммируем по всем индексам до k :

$$f_{k+1} \leq f_0 - c_1 \frac{1 - c_2}{L} \sum_{j=0}^k \cos^2 \Theta_j \|\nabla f_j\|^2$$

Ну и так как функция ограничена снизу, то сумма ограничена сверху некой положительной константой. ■

Из предыдущей теоремы вытекает, что

$$\cos^2 \Theta_k \|\nabla f_k\|^2 \rightarrow 0.$$

Чтож, теперь остался маленький (!) трюк (!), который позволит доказать нам глобальную сходимость метода Ньютона. Для этого потребуем чтобы для матриц $B_k = [\nabla^2 f_k]^{-1}$ числа обусловленности были ограничены: $\mu(B_k) \leq M \forall k, M > 0$ и все матрицы B_k положительно определены. Теперь запишем косинус и попробуем его отделить от нуля:

$$\cos \Theta_k = \frac{\nabla f_k^T [\nabla^2 f_k]^{-1} \nabla f_k}{\|\nabla f_k\| \|[\nabla^2 f_k]^{-1} \nabla f_k\|}$$

Рассмотрим отдельно норму произведения обратной матрицы гессiana на градиент:

$$\|[\nabla^2 f_k]^{-1} \nabla f_k\| = \sqrt{\nabla f_k^T [\nabla^2 f_k]^{-2} \nabla f_k} \leq \sqrt{\lambda_{\max}([\nabla^2 f_k]^{-2})} \|\nabla f_k\| = \lambda_{\max}([\nabla^2 f_k]^{-1}) \|\nabla f_k\|$$

Тут мы используем микроскопический хитрый трюк: $\lambda_{\min}(A) \leq x^T A x \leq \lambda_{\max}(A)$ для $x^T x = 1$. Тогда:

$$\cos \Theta_k \geq \frac{\lambda_{\min}([\nabla^2 f_k]^{-1})}{\lambda_{\max}([\nabla^2 f_k]^{-1})} \geq \frac{1}{M} > 0$$

Таким образом, мы отделили косинус от нуля, значит последовательность норм градиентов стремится к нулю, что означает глобальную сходимость метода Ньютона для функций, у которых гессиан положительно определен во всех точках и имеет ограниченное сверху число обусловленности.

2.4.3 Локальная сходимость метода Ньютона

Пусть $f \in C_M^{2,2}$, x_* — точка локального минимума с $\nabla^2 f(x_*) \succeq \mu I, \mu > 0$. Мы стартуем из некоторой окрестности точки x_* :

$$x_0 : \|x_* - x_0\| < \bar{r} = \frac{2\mu}{3M}$$

Тогда метод Ньютона с $\alpha_k \equiv 1$ сходится $\|x_k - x_*\| < \bar{r} \forall k$ и имеет скорость сходимости квадратичную (суперлинейную):

$$\|x_{k+1} - x_*\| \leq \frac{M\|x_k - x_*\|^2}{2(\mu - M\|x_k - x_*\|)}$$

Доказательство:

Мы делаем итерации вида $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$. Рассмотрим разность между x_{k+1} и x_* :

$$\begin{aligned} x_{k+1} - x_* &= x_k - x_* - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) = x_k - x_* - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x_* + t(x_k - x_*)) (x_k - x_*) dt = \\ &= (x_k - x_*) [\nabla^2 f(x_k)]^{-1} \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))) dt \end{aligned}$$

Положим $r_k := \|x_k - x_*\|$. Тогда:

$$\left\| \int_0^1 (\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))) dt \right\| \leq \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))\| dt \leq M \frac{r_k}{2}$$

Так как для $f \in C_M^{2,2}$ выполнено для $\|y - x\| = r$:

$$\nabla^2 f(x) - MrI_n \preceq \nabla^2 f(y) \preceq \nabla^2 f(x) + MrI_n,$$

то:

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x_*) - Mr_k I_n \succeq (\mu - Mr_k) I_n$$

Тогда если $r_k < \frac{\mu}{M}$, то гессиан в точке x_k положительно определен и $\|[\nabla^2 f(x_k)]^{-1}\| \leq (\mu - Mr_k)^{-1}$. Значит имеем:

$$r_{k+1} \leq \frac{Mr_k^2}{2(\mu - Mr_k)}$$

Из этого следует, что для $x_0 : \|x_* - x_0\| < \bar{r} = \frac{2\mu}{3M}$ метод Ньютона сходится и сходится квадратично (суперлинейно). ■

2.4.4 Модификации метода Ньютона для невыпуклых задач оптимизации

Для невыпуклых задач не выполняется $\nabla^2 f \succ 0$. Хотим теперь использовать в качестве B_k какую-то матрицу, которую будем строить на основе гессиана, но чтобы она имела положительную определенность. Итерации нашего метода выглядят по-прежнему: $x_{k+1} = x_k - \alpha_k B_k^{-1} \nabla f(x_k)$.

Рассмотрим несколько методов коррекции гессиана для достижения положительной определенности:

- **Модификация собственных значений** Так как гессиан симметричная матрица, то можем представить его в виде:

$$\nabla^2 f(x_k) = Q\Lambda Q^T, \quad \Lambda = \text{Diag}(\lambda_1, \dots, \lambda_n), \quad Q^{-1} = Q^T$$

Тогда в качестве матрицы B_k можно взять

$$B_k = Q\bar{\Lambda}Q^T, \quad \text{где } \bar{\lambda}_i = \begin{cases} \lambda_i, & \lambda_i \geq \delta > 0, \\ \delta, & \text{иначе.} \end{cases}$$

После того, как мы нашли собственное разложение, уже можно не решать заново СЛАУ $B_k d_k = -\nabla f(x_k)$, а в явном виде вычислить d_k :

$$d_k = -Q\bar{\Lambda}^{-1}Q^T \nabla f(x_k)$$

- **Модификация диагонали** В качестве B_k берём следующую матрицу:

$$B_k = \nabla^2 f(x_k) + \tau_k I_n$$

```

1: procedure BkSEARCH( $f, x_k, \tau$ )
2:   get  $\nabla^2 f(x_k)$ 
3:   while  $\tau < \tau_{max}$  do
4:      $B_k \leftarrow \nabla^2 f(x_k) + \tau I_n$ 
5:     if CHOLESKYDECOMPOSITION( $B_k$ ) then break
6:      $\tau \leftarrow \min(\tau_{max}, \tau\nu), \nu > 1$ 
7:      $\tau \leftarrow \tau/\beta, \beta > 1$ 
8:   return  $B_k, \tau$ 

```

Соответственно, если всё удачно, то мы знаем разложение Холецкого для нашей матрицы B_k и для вычисления d_k нужно дважды решить СЛАУ с треугольной матрицей:

$$B_k = LL^T \Rightarrow d_k = -L^{-T}L^{-1}\nabla f(x_k)$$

Это дешевле чем первый способ, потому что разложение Холецкого на порядок дешевле чем собственное разложение и даже несколько раз его выполнить будет дешевле одного собственного разложения.

- **Модификация симметричной факторизацией** Делаем разложение вида:

$$P^T \nabla^2 f(x_k) P = LDL^T, \text{ где } L - \text{нижнетреугольная, } D - \text{блочно-диагональная с блоками } 1 \times 1 \text{ и } 2 \times 2$$

Тогда затем сделаем собственное разложение для D , это очень просто (блочно-диагональная матрица), потом сделаем модификацию для Λ как мы делали в модификации собственных значений:

$$B_k = PL\bar{D}L^T P^T \Rightarrow d_k = -PL^{-T}\bar{D}^{-1}L^{-1}P^T \nabla f(x_k)$$

2.4.5 Примеры медленной и быстрой работы метода

Идеальный вариант для работы метода Ньютона - квадратичная функция: $f(x) = \frac{1}{2}x^T A x + b^T x$, где $A \in \mathbb{S}_{++}^n, b \in \mathbb{R}^n$. Метод Ньютона сходится за одну итерацию из любой начальной точки.

В качестве функции, на который метод сходится линейно, можно назвать функцию x^4 .

Плохой функцией можно назвать функцию Розенброка, на ней метод Ньютона вообще не сходится.

Еще можно упомянуть про большую размерность, что гессиан не будет влезать в память.

2.5 Метод сопряженных градиентов для решения системы линейных уравнений. Доказательство сходимости метода за конечное число итераций (в предположении, что сопряженные направления известны). Модификации метода для минимизации произвольных функций. @bigbluebutterfly

Преамбула: здесь мы построим итеративный, относительно легковесный и при этом точный метод решения СЛАУ с хорошими свойствами сходимости, а потом модифицируем его для оптимизации произвольных функций.

Начнем с рассмотрения линейного метода сопряженных градиентов (далее CG). В сущности, это метод для решения СЛАУ вида

$$Ax = b$$

где A - симметричная положительно определенная матрица размера $n \times n$. Данную проблему, как известно, можно эквивалентно переформулировать в виде задачи оптимизации следующим образом:

$$\phi(x) = \frac{1}{2}\langle Ax, x \rangle - \langle b, x \rangle \rightarrow \min_x$$

Поскольку в обеих формулировках задача имеет одно и то же (притом единственное) решение, мы можем интерпретировать CG как метод решения СЛАУ или же как метод оптимизации выпуклых квадратичных функций. Отметим, что невязка СЛАУ

$$r(x) = Ax - b$$

эквивалентна градиенту оптимизируемой функции:

$$\nabla_x \phi(x) = Ax - b$$

В частности, на k -ом шаге:

$$r_k = Ax_k - b$$

Начнем немного издалека.

Определение. Будем называть набор ненулевых векторов $\{p_0, p_1, \dots, p_m\}$ **сопряженным** по отношению к симметричной положительно определенной матрице A , если

$$p_i^T A p_j = 0, \quad \forall i \neq j \quad (3)$$

Отметим, что такой набор векторов является линейно независимым. В самом деле, пусть

$$\exists \alpha : \sum_{i=0}^{n-1} \alpha_i p_i = 0$$

Домножим каждое слагаемое справа на $A p_j$ - все слагаемые кроме j -ого обнулятся (прямо как сроки Путина). При j -ом слагаемом придется поставить нулевой коэффициент. Повторим n раз, конец.

Важное свойство сопряженности: мы можем минимизировать $\phi(\cdot)$ за n шагов, последовательно минимизируя ее вдоль каждого направления в сопряженном наборе. Для этого рассмотрим метод *сопряженных направлений* (это пока не CG, к нему придем чуть позже).

Пусть нам данно начальное приближение $x_0 \in \mathbb{R}^n$ и набор сопряженных направлений $\{p_0, p_1, \dots, p_{n-1}\}$. Будем генерировать последовательность $\{x_k\}$ следующим образом:

$$x_{k+1} = x_k + \alpha_k p_k$$

при этом задавая α_k как минимум $f(\alpha) = \phi(x_k + \alpha p_k)$ - в силу квадратичности имеется closed-form выражение:

$$\alpha_k = -\frac{\langle r_k, p_k \rangle}{\langle A p_k, p_k \rangle}$$

Теорема. Оказывается, что для любого $x_0 \in \mathbb{R}^n$ последовательность $\{x_k\}$, сгенерированная методом сопряженных направлений, сходится к решению исходной СЛАУ x^* в худшем случае за n шагов.

Док-во. Во-первых отметим, что Кропотов умудрился сжать это в одну строчку. Гений, не иначе.

Поскольку набор сопряженных направлений линейно независим, он порождает пространство \mathbb{R}^n . Таким образом, мы можем записать разность между x_0 и x^* следующим образом:

$$x^* - x_0 = \sigma_0 p_0 + \dots + \sigma_{n-1} p_{n-1}$$

для некоторых множителей σ . Домножим на $p_k^T A$ слева и используем свойство сопряженности:

$$\sigma_k = \frac{p_k^T A (x^* - x_0)}{p_k^T A p_k}$$

Покажем, что σ_k совпадают с размерами шага α_k . Для сгенерированного алгоритмом x_k верно

$$x_k = x_0 + \alpha_0 p_0 + \dots + \alpha_{k-1} p_{k-1}$$

Домножим слева на $p_k^T A$ и используем свойство сопряженности:

$$p_k^T A (x_k - x_0) = 0$$

Отсюда получаем

$$p_k^T A (x^* - x_0) = p_k^T A (x^* - x_k) = p_k^T (b - A x_k) = -p_k^T r_k$$

Подставим это в выражение для σ_k и получим, что x^* является суммой x_0 и линейной комбинации p_i , что дает нам гарантию сходимости за n шагов. \square

Здесь стоит отметить еще одно важное свойство метода (видимо, без док-ва?): невязка r_k на k -ом шаге ортогональна всем предыдущим направлениям поиска p_0, \dots, p_{k-1} , то бишь

$$r_k^T p_i = 0, \quad \forall i = 0, 1, \dots, k-1$$

Ближе к делу.

Теперь перейдем наконец к CG. Самое важное, что стоит запомнить сейчас - его можно получить из метода сопряженных направлений с помощью *любого* алгоритма подбора сопряженных направлений. Здесь мы можем использовать к примеру собственные векторы или модифицировать процесс Грама-Шмидта для подбора сопряженных векторов. Тем не менее, это все вычислительно затратно. В идеале нам хотелось бы уметь пересчитывать текущее направление поиска p_k с использованием *только* предыдущего направления p_{k-1} так, чтобы p_k было автоматически сопряжено ко всем направлениям до этого (p_0, \dots, p_{k-2}).

Собственно, будем делать это следующим образом. Возьмем p_k как линейную комбинацию отрицательной невязки $-r_k$ (вспомним, что это антиградиент!), и предыдущего направления поиска p_{k-1} :

$$p_k = -r_k + \beta_k p_{k-1}$$

где β_k определяется требованием сопряженности направлений p_k и p_{k-1} . По факту, домножим на $p_{k-1}^T A$, вспомним про сопряженность и получим:

$$\beta_k = \frac{r_k^T A p_{k-1}}{p_{k-1}^T A p_{k-1}}$$

Теперь у нас есть все, чтобы сформулировать первую версию алгоритма!

```

1: procedure CG( $A, b, x_0$ )
2:    $r_0 \leftarrow Ax_0 - b$ 
3:    $p_0 \leftarrow -r_0$ 
4:    $k \leftarrow 0$ 
5:   while  $r_k \neq 0$  do
6:      $\alpha_k \leftarrow -\frac{r_k^T p_k}{p_k^T A p_k}$ 
7:      $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
8:      $r_{k+1} \leftarrow Ax_{k+1} - b$ 
9:      $\beta_{k+1} \leftarrow \frac{r_{k+1}^T A p_k}{p_k^T A p_k}$ 
10:     $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$ 
11:     $k \leftarrow k + 1$ 
12:   return  $x_{k+1}$ 

```

Однако, для использования метода на практике стоит внести некоторые коррективы. В первую очередь, формулу для обновления α_k можно переписать следующим образом:

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

в силу стр. 10 алгоритма и ортогональности текущей невязки всем предыдущим направлениям поиска.

Кроме того, поскольку разность невязок

$$r_{k+1} - r_k = \alpha_k A p_k$$

мы можем переформулировать обновление β_k следующим образом:

$$\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$$

Кроме того, часто может быть излишним ждать полной сходимости - из-за численных погрешностей она может даже и не наступить - поэтому разумно использовать критерий останова по норме невязки. Теперь попробуем записать финальную версию алгоритма:

```

1: procedure CG-FINALE( $A, b, x_0, \epsilon$ )
2:    $r_0 \leftarrow Ax_0 - b$ 
3:    $p_0 \leftarrow -r_0$ 
4:    $k \leftarrow 0$ 
5:   while  $\|r_k\| > \epsilon$  do
6:      $\alpha_k \leftarrow \frac{r_k^T r_k}{p_k^T A p_k}$ 
7:      $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
8:      $r_{k+1} \leftarrow r_k + \alpha_k A p_k$ 
9:      $\beta_{k+1} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ 
10:     $p_{k+1} \leftarrow -r_{k+1} + \beta_{k+1} p_k$ 
11:     $k \leftarrow k + 1$ 
12:   return  $x_{k+1}$ 

```

Отметим, что в любой момент алгоритма нам не нужно помнить векторы r, p, x более чем на две последних итерации, потому что в реализации мы можем спокойно перезаписывать старые значения. CG рекомендуется использовать только для больших задач, в малых больше подойдут основанные на факторизациях решения - в силу их меньшей чувствительности к погрешностям вычислений.

Упражнение: показать, что в методе сопряженных градиентов градиенты не являются сопряженными.

Теперь поговорим о сходимости метода. Доказывать здесь, думаю, ничего не надо. Мы уже обсудили, что за n итераций имеет место гарантированная сходимость. Однако, при определенных свойствах распределения собственных значений матрицы A сходимость может наступить гораздо раньше.

1. Если у матрицы есть только r различных собственных значений, метод сойдется к точному решению за не более чем r итераций.
2. Если у матрицы есть r различных кластеров собственных значений, метод сойдется примерно за r итераций.

Также имеют место следующие оценки:

$$\|x_{k+1} - x^*\|^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \|x_0 - x^*\|^2$$

где $\lambda_1 \leq \dots \leq \lambda_n$ - собственные значения A .

$$\|x_k - x^*\| \leq 2 \left(\frac{\sqrt{k(A)} - 1}{\sqrt{k(A)} + 1} \right)^k \|x_0 - x^*\|$$

где $k(A) = \frac{\lambda_n}{\lambda_1}$ - число обусловленности задачи.

В случае, если у задачи плохая обусловленность, мы можем попробовать использовать предобуславливание - то есть, перейти к новым переменным

$$\hat{x} = Cx$$

где C - невырожденная матрица, и решать новую систему

$$(C^{-T}AC^{-1})\hat{x} = C^{-T}b$$

основываясь на свойствах уже трансформированной матрицы. (стасян, привет!)

Нелинейная часть

СГ можно интерпретировать, как уже говорилось, в качестве метода решения СЛАУ или же в качестве метода оптимизации квадратичной функции. Однако, оказывается, что данный метод можно использовать также для минимизации произвольных нелинейных функций $f(x)$ - потребуется только дифференцируемость (в целом даже выпуклость необязательна). Для этого нам потребуется изменить следующие вещи:

1. Теперь r_k будет вычисляться как честный градиент $\nabla f(x)$ в точке x_k .
2. α_k теперь не будет иметь аналитического оптимального значения, поэтому будем искать его линпоиском. При этом, чтобы получить гарантированное направление спуска, нам потребуются условия Вульфа с константой $c_2 < 0.5$.

Это называется **алгоритмом Флетчера-Ривса**. Известно много вариантов данного алгоритма, в первую очередь различающихся методом обновления параметра β . Важной модификацией является **метод Полака-Рибье**, в котором β обновляется следующим образом:

$$\beta_{k+1} = \frac{r_{k+1}^T(r_{k+1} - r_k)}{r_k^T r_k}$$

Однако, в общем случае такая модификация опять может портить направление p_k . Чтобы от этого избавиться, будем брать $ReLU(\beta)$ - тогда сильное условие Вульфа гарантирует, что p_k будет направлением спуска. На практике, с неточной процедурой линпоиска метод PR демонстрирует большую работоспособность и стабильность.

2.6 Безгессианный (неточный) метод Ньютона (HFN), его скорость сходимости. Способы оценивания произведения гессиана на вектор с помощью разностного дифференцирования. @kirili4ik

Этот билет основан на нашей офлайн лекции 2020 и видеолекции 2016 ([ссылка](#))

$$f(x) \rightarrow \min_{x \in \mathbb{R}^n}$$

Данную задачу хорошо решает метод Ньютона. Но у него стоимость по памяти $O(n^2)$. Если $n > 5000$ уже просто не сможем хранить гессиан в памяти.

Первый способ обойти проблему - СГ. Но там скорость сходимости линейная. Попробуем сделать линейную стоимость по памяти и суперлинейную скорость сходимости. Начнем с обычного метода Ньютона:

$$x_{k+1} = x_k + \alpha_k d_k$$

$$H_k d_k = -\nabla f(x) \quad (1)$$

где H_k - гессиан на шаге k .

Ключевая идея HFN: решать СЛАУ(1) с помощью метода сопряженных градиентов.

Важные моменты:

- 1) Вместо хранения H используем процедуру " $H \cdot d$ "
- 2) Решаем СЛАУ неточно

1) будет обсуждаться далее (см. разностное дифференцирование). Интуиция для 2): вдалеке от точки оптимума можно решать систему не точно, а по мере приближения увеличивать точность.

Новый метод останова для CG:

$$r_k := H_k d_k + \nabla f_k$$

$$\|r_k\| \leq \eta_k \|\nabla f(x_k)\|$$

$\eta_k \in (0, 1)$ и называется *форсирующей последовательностью*.

— **(МОЖНО НЕ ДОКАЗЫВАТЬ) Локальная сходимость HFN.**

Доказательство с лекции 2020, более подробная версия в [12] раздел 7.1, теорема 7.1

Я не уверен, что его будут требовать, его так скомканно и давали:

$$f \in C_M^{2,2}, \alpha_k = 1, g_k = \nabla f(x_k)$$

$$x_{k+1} = x_k + \alpha_k d_k$$

$$r_k = H_k d_k + g_k \Rightarrow d_k = H_k^{-1}(r_k - g_k) \quad (2)$$

$$\|r_k\| \leq \eta_k \|g_k\| \quad (3)$$

Покажем сходимость:

$$\begin{aligned} \|x_{k+1} - x_{\text{opt}}\| &= \|x_k - x_{\text{opt}} + \underbrace{H_k^{-1}(r_k - g_k)}_{d_k \text{ по (2)}}\| \stackrel{\cdot H_k^{-1}}{\leq} \underbrace{\|H_k^{-1}\|}_{\leq \text{const}} \cdot \|H_k(x_k - x_{\text{opt}}) - g_k + r_k\| \leq \\ &\leq \text{const} \cdot \left(\underbrace{\|H_k(x_k - x_{\text{opt}}) - g_k\|}_{O(\|x_k - x_{\text{opt}}\|^2)} + \underbrace{\|r_k\|}_{\leq \eta_k \|g_k\| \text{ по (3)}} \right) \leq \text{const} \cdot (O(\|x_k - x_{\text{opt}}\|^2) + \eta_k \|g_k\|) \\ &\quad \left\{ \|g_k\| = \underbrace{\|\nabla f(x_k) - \nabla f(x_{\text{opt}})\|}_0 \stackrel{\text{липп. ?}}{=} O(\|x_k - x_{\text{opt}}\|) \right\} \\ &\leq \text{const} \cdot (O(\|x_k - x_{\text{opt}}\|^2) + \eta_k O(\|x_k - x_{\text{opt}}\|)) \quad \square \end{aligned}$$

(ДОКАЗАТЕЛЬСТВО ВЫШЕ МОЖНО НЕ ПРИВОДИТЬ)

— **Локальная скорость сходимости HFN.**

$$f \in C_M^{2,2}, \alpha_k = 1. \quad \|r_k\| \leq \eta_k \|\nabla f(x_k)\|$$

1) $\eta_k = \eta < 1 \rightarrow$ линейная скорость сходимости

2) $\eta_k \xrightarrow{k \rightarrow \infty} 0 \rightarrow$ суперлинейная скорость сходимости. Обычно берут: $\eta_k = \min\left(\frac{1}{2}, \sqrt{\|\nabla f(x_k)\|}\right)$

3) $\eta_k = O(\|\nabla f(x_k)\|) \rightarrow$ квадратичная скорость сходимости. Обычно берут: $\eta_k = \min\left(\frac{1}{2}, \|\nabla f(x_k)\|\right)$

Для 2) и 3) мы берем большую точность для CG, он делает больше шагов, но из-за этого получаем лучшую скорость сходимости метода Ньютона. В реализации используется 2) потому что суперлинейной скорости сходимости на практике достаточно!

Общая схема метода выглядит HFN так:

Краткое описание дальнейшей процедуры: внутренний цикл - CG, $\{z_j\}$ играют роль $\{x_j\}$ для внутреннего запуска CG:

```

1: procedure HFN( $f, \varepsilon, x_0$ )
2:   get  $\nabla f(x_0)$ 
3:   for  $k = 0, 1, 2, \dots$  do
4:      $\varepsilon_k \leftarrow \min\left(\frac{1}{2}, \sqrt{\|\nabla f_k\|}\right) \cdot \|\nabla f_k\|$ 
5:      $z_0 = 0, g_0 = H_k z_0 + \nabla f_0, d_0 = -g_0$ 
6:     for  $j = 0, 1, 2, \dots$  do
7:        $\alpha_j \leftarrow \frac{g_j^T g_j}{d_j^T H_j d_j}$ 
8:        $z_{j+1} \leftarrow z_j + \alpha_j d_j$ 
9:        $g_{j+1} \leftarrow g_j + \alpha_j H_k d_j$ 
10:      if  $\|g_{j+1}\| \leq \varepsilon_k$  then  $p_k \leftarrow z_{j+1}$ ; break
11:       $\beta_j \leftarrow \frac{g_{j+1}^T g_{j+1}}{g_j^T g_j}$ 
12:       $d_{j+1} \leftarrow -g_{j+1} + \beta_j d_j$ 
13:       $\alpha_k \leftarrow \text{BACKTRACKING}(\alpha_{start} = 1)$ 
14:       $x_{k+1} \leftarrow x_k + \alpha_k p_k$ 
15:      if  $\|\nabla f_{k+1}\| < \varepsilon$  then break
16:   return  $x_k$ 

```

Важно: В данной процедуре мы **не** храним в памяти H , а только используем функцию перемножения Hd это и позволяет уйти от стоимости по памяти $O(n^2)$. Как это делать разбираемся далее:

— Трюк с логистической регрессией (из дз)

Для логистической регрессии мы можем аналитически выписать гессиан, воспользуемся этим! (это было в дз)

$$f(w) = \frac{1}{N} \sum_{i=1}^N \log(1 + \exp(-y_i w^T x_i)) + \frac{\lambda}{2} \|w\|^2 \longrightarrow \min_w$$

$$\nabla^2 f(w) \stackrel{\text{аналитически}}{=} \frac{1}{N} X^T B X + \lambda I, \quad B = \text{diag}(b_1, \dots, b_n)$$

$$\nabla^2 f(w) \cdot d = \frac{1}{N} X^T B X \cdot d + \lambda \cdot d = u \Leftrightarrow \text{последовательным шагом:}$$

$$\begin{aligned} 1) \quad u_1 &= X d \\ 2) \quad u_2 &= B u_1 \\ 3) \quad u &= \frac{1}{N} X^T u_2 + \lambda d \end{aligned}$$

Итого получаем:

$$X \in \mathbb{R}^{N \times D} \Rightarrow$$

$$\text{Сложность } \nabla^2 f \cdot d = O(ND^2)$$

$$\text{Сложность } 1), 2), 3) = O(ND)$$

Это лучший способ решения логистической регрессии для хорошей точности.

Далее рассмотрим способ получить $H \cdot d$, имея только функцию для вычисления градиента.

— Разностное дифференцирование

Тут рассмотрены 3 метода и в конце их плюсы и минусы.

Хотим: $H \cdot d$, Есть: $\nabla f(x)$. Воспользуемся Тейлором!

ε – маленькое число, d – то на что надо умножить $\nabla^2 f(x)$

$$\nabla f(x + \varepsilon d) \stackrel{\text{Тейлор}}{=} \nabla f(x) + \nabla^2 f(x) \cdot \varepsilon d + O(\varepsilon^2)$$

$$1) \quad \nabla^2 f(x) d = \underbrace{\frac{\nabla f(x + \varepsilon d) - \nabla f(x)}{\varepsilon}}_{\Theta_1(\varepsilon)} + \delta_\varepsilon, \quad \delta_\varepsilon = O(\varepsilon)$$

В целом точность уменьшается линейно, но после какого-то момента погрешность станет больше, чем δ_ε (см. рис ниже). Из-за вычитания в числителе. Но какую точность брать?

$\text{fl}(\nabla f(x + \varepsilon d))$ - точность вычисления ∇f на компьютере, float .
 $\text{fl}(\nabla_i f(x + \varepsilon d)) = \nabla_i f(x + \varepsilon d)(1 + \varepsilon_m)$, ε_m - машинная точность.

Пусть $|\text{fl}(\nabla_i f(x + \varepsilon d)) - \nabla_i f(x + \varepsilon d)| \leq \underbrace{L_f}_{\text{const}} \varepsilon_m$

Ошибка 1) = $\frac{L_f \varepsilon_m + L_f \varepsilon_m}{\varepsilon} + \underbrace{L \cdot \varepsilon}_{O(\varepsilon)} \rightarrow \min_{\varepsilon} \Rightarrow$ производную:

$$L - \frac{2L_f \varepsilon_m}{\varepsilon^2} \Rightarrow \varepsilon^2 = \frac{2L_f \varepsilon_m}{L}, \varepsilon = \sqrt{\frac{2L_f}{L}} \sqrt{\varepsilon_m} \approx \varepsilon_m^{\frac{1}{2}}$$

"Для многих нормальных функций L_f и L одного порядка, поэтому имеем порядок корня из машинной точности (напр. 10^{-8} для $\varepsilon_m = 10^{-16}$)" Кротонов 2016.

Теперь подставим $\varepsilon = \sqrt{\varepsilon_m}$ в 1) и получим что наилучшая достижимая точность также порядка $\sqrt{\varepsilon_m}$ (см. рис ниже)

Улучшим:

$$2) \nabla^2 f(x)d = \underbrace{\frac{\nabla f(x + \varepsilon d) - \nabla f(x - \varepsilon d)}{2\varepsilon}}_{\Theta_2} + \delta_\varepsilon, \quad \delta_\varepsilon = O(\varepsilon^2)$$

$$\text{ошибка 2)} = \frac{L_f \varepsilon_m + L_f \varepsilon_m}{2\varepsilon} + L\varepsilon^2 \Rightarrow 2L\varepsilon - \frac{2L_f \varepsilon_m}{2\varepsilon^2} = 0 \Rightarrow \varepsilon \approx \varepsilon_m^{\frac{1}{3}}$$

Если подставить, то можем получить точность около $\varepsilon_m^{\frac{2}{3}}$

Еще улучшим!

Комплексное продолжение функции:

$$\begin{aligned} 3) \nabla f(x + \varepsilon id) &\stackrel{\text{Тейлор}}{=} \nabla f(x) + \nabla^2 f(x)i\varepsilon d + (-1) \cdot O(\varepsilon^2) + iO(\varepsilon^3) \\ \text{Im}[\nabla f(x + i\varepsilon d)] &= \nabla^2 f(x)\varepsilon d + O(\varepsilon^3) \\ \nabla^2 f(x)d &= \underbrace{\frac{\text{Im}[\nabla f(x + i\varepsilon d)]}{\varepsilon}}_{\Theta_3} + O(\varepsilon^2) \end{aligned}$$

Так как в числителе нет вычитания, то нет и потери точности! **НО!** Чтобы в ряд тейлора раскладывать комплексную функцию, требуется, чтобы функция была бесконечное число раз непрерывно дифференцируема. Но функции можно приводить к такому виду. Также из-за перехода к комплексным делаем несколько больше итераций по времени, но ничего страшного, этот метод из представленных самый лучший.

На графике $\Theta_* = \nabla^2 f(x)d$

Итого: 1) способ вычисляет всего 1 градиент, но может получить не такую большую точность. 2) способ считает 2 градиента, но получает лучше точность. 3) способ считает 1 градиент (но из-за перехода к комплексным на самом деле больше), но имеет лучшую сколь угодно низкую точность, применим не ко всем функциям.

2.7 Квазиньютоновские методы. Схемы BFGS и L-BFGS. @isadrtdinov

Как непосредственно следует из названия, квазиньютоновские методы пытаются вести себя как обычный метод Ньютона. Вспомним, что на каждой итерации метода Ньютона мы решаем систему линейных уравнений (по сути обращаем гессиан), что приводит к сложности итерации $O(n^3)$. Кроме того, от нас требуется подсчитывать сам гессиан, что может быть нетривиальной задачей для некоторых функций, например, для нейросетей или для логистической регрессии (привет, ТИ!). Все квазиньютоновские методы оперируют градиентом и с помощью него итеративно приближают гессиан функции. Мотивация использовать квазиньютоновские методы очень простая: мы снижаем затраты по времени, но при этом сохраняем суперлинейную скорость сходимости.

Разберем подход, который приводит к квазиньютоновским методам. Вспоминаем про многомерный ряд Тейлора и приближаем нашу гладкую функцию квадратичной моделью:

$$f(x_k + d_k) \approx \hat{f}_k(d_k) = f_k + \nabla f_k^T d_k + \frac{1}{2} d_k^T B_k d_k$$

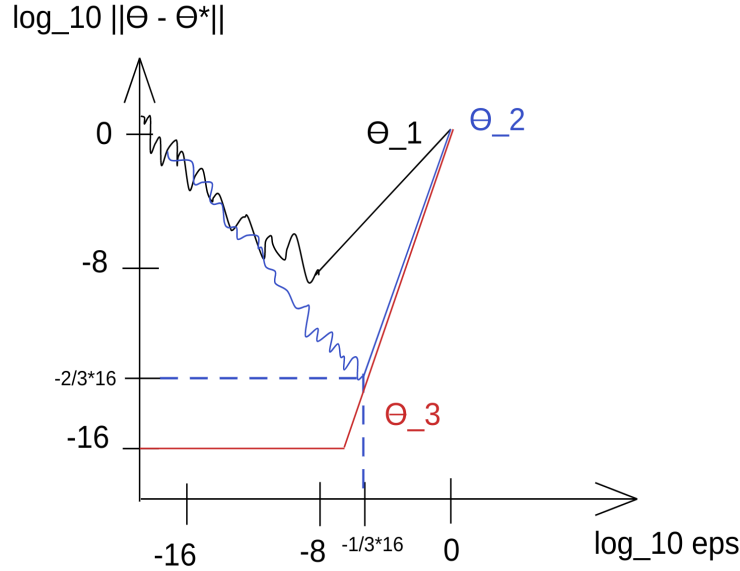


Рис. 17: Точность при разностном дифференцировании

Матрица B_k - симметричная (таков гессиан) и положительно-определенная (это необходимо для выпуклости квадратичной модели $\hat{f}_k(d_k)$, чтобы функция обладала единственным минимумом). В ванильном методе Ньютона мы используем $B_k = \nabla^2 f(x_k)$, но мы уже договорились, что такой подход нас не устраивает.

Мы хотим найти оптимальное направление, вдоль которого нужно сделать шаг. Приравняв к нулю градиент $\hat{f}_k(d_k)$, мы получим систему линейных уравнений:

$$B_k d_k = -\nabla f_k \Leftrightarrow d_k = -B_k^{-1} \nabla f_k$$

После этого с помощью линейного поиска мы находим длину α_k и делаем шаг спуска:

$$x_{k+1} = x_k + \alpha_k d_k$$

Получаем аппроксимацию функции в новой точке:

$$\hat{f}_{k+1}(d) = f_{k+1} + \nabla f_{k+1}^T d + \frac{1}{2} d^T B_{k+1} d$$

Нужно сочинить какие-то разумные требования для матрицы B_{k+1} , чтобы она была похожа на гессиан. Закажем, чтобы градиент функции \hat{f}_{k+1} совпадал с градиентом исходной функции f в точках x_k и x_{k+1} :

$$\nabla f_k = \nabla f(x_{k+1} - \alpha_k d_k) = \nabla \hat{f}_{k+1}(-\alpha_k d_k) = \nabla f_{k+1} - \alpha_k B_{k+1} d_k$$

$$\nabla f_{k+1} = \nabla \hat{f}_{k+1}(0) = \nabla f_{k+1}$$

Второе уравнение тривиально, а вот из первого можно вывести линейное ограничение на B_{k+1} :

$$B_{k+1} \alpha_k d_k = \nabla f_{k+1} - \nabla f_k$$

Приняв $s_k = x_{k+1} - x_k = \alpha_k d_k$ и $y_k = \nabla f_{k+1} - \nabla f_k$, получаем ограничение, именуемое **уравнением секущей**:

$$B_{k+1} s_k = y_k$$

Это ограничение используется во всех квазиньютоновских методах. При этом, искомая матрица B_{k+1} существует только при условии $s_k^T y_k > 0$. Если функция f сильно выпукла, то это неравенство выполнено для любых векторов $x \neq y$ (пользуемся дифференциальным критерием первого порядка):

$$\begin{cases} f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|^2 \\ f(x) \geq f(y) + \nabla f(y)^T (x - y) + \frac{\mu}{2} \|y - x\|^2 \end{cases} \Rightarrow$$

$$f(x) + f(y) \geq f(x) + f(y) + (\nabla f(x) - \nabla f(y))^T (y - x) + \mu \|y - x\|^2$$

$$(\nabla f(y) - \nabla f(x))^T (y - x) \geq \mu \|y - x\|^2 > 0$$

Если же функция не является сильно выпуклой, то выполнение неравенство обеспечит (слабое) условие Вульфа, поэтому именно его применяют для линейного поиска в квазиньютоновских алгоритмах.

$$\nabla f(x_{k+1})^T d_k \geq c_2 \nabla f(x_k) d_k$$

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T \alpha_k d_k \geq (c_2 - 1) \alpha_k \nabla f(x_k) d_k$$

$$(\nabla f(x_{k+1}) - \nabla f(x_k))^T (x_{k+1} - x_k) \geq (c_2 - 1) \alpha_k \nabla f(x_k) d_k = (< 0) \cdot (> 0) \cdot (< 0) > 0$$

Все квазиньютоновские методы обновляют матрицу B_k по правилу $B_{k+1} = B_k + U_k$, где U_k - матрица невысокого ранга. Это необходимо для эффективного вычисления обратной матрицы $H_{k+1} = B_{k+1}^{-1} = (B_k + U_k)^{-1}$. Конкретный вид матрицы U_k и отличает разные квазиньютоновские методы.

SR-1

SR-1 - это простейший квазиньютоновский метод, назван так за то, что обновляет гессиан симметричной матрицей ранга 1. Формула обновления B_k такая:

$$B_{k+1} = B_k + \sigma v v^T$$

Здесь $\sigma = +1$ или -1 , а σ и v подбираются так, чтобы выполнялось уравнение секущей. Итоговые формулы для метода такие:

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

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}$$

У схемы SR-1 есть ряд недостатков: получаемые матрицы могут не быть положительно определенными, а знаменатель порой оказывается очень маленьким, что выражается в численной неустойчивости. Однако, утверждается, что матрицы в SR-1 хорошо аппроксимируют настоящий гессиан. Со скоростью сходимости здесь сложно. Если методу удастся избежать проблем с положительной определенностью и знаменателем, то сходимость суперлинейная. Но зачастую метод ведет себя очень неустойчиво.

BFGS

Матрица H_{k+1} , используемая в методе BFGS, является решением следующей оптимизационной задачи (не уверен, что это нужно, но просто забавный факт):

$$\begin{cases} \|H - H_k\|_W \rightarrow \min_{H \in \mathbb{S}_{++}^n} \\ Hy_k = s_k \end{cases}$$

Здесь используется взвешенная норма Фробениуса: $\|A\|_W = \|W^{1/2} A W^{1/2}\|_F$, а в качестве матрицы весов берется любая, удовлетворяющая условию $W s_k = y_k$. Ограничение в системе - это уравнение секущей, записанное через матрицу H_{k+1} ($B_{k+1} s_k = y_k \Leftrightarrow H_{k+1} y_k = s_k$).

Эта оптимизационная задача имеет единственное решение, которое дает нам формулу для обновления матрицы H_k :

$$H_{k+1} = \left(I_n - \frac{s_k y_k^T}{\langle y_k, s_k \rangle} \right) H_k \left(I_n - \frac{y_k s_k^T}{\langle y_k, s_k \rangle} \right) + \frac{s_k s_k^T}{\langle y_k, s_k \rangle}$$

При этом формула для обновления B_k выглядит так (она не нужна в реализации алгоритма, но ее стоит знать):

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

С первого взгляда может показаться, что мы не облегчили себе задачу: ведь при пересчете H_k мы перемножаем матрицы, а сложность этой операции все так же $\mathcal{O}(n^3)$. Но если раскрыть скобки в формуле, то все операции в ней упростятся до сложности $\mathcal{O}(n^2)$ (это умножение матрицы на вектор, умножение вектора-столбца на вектор-строку и скалярное произведение):

$$H_{k+1} = H_k + \frac{(y_k^T s_k + y_k^T H_k y_k) s_k s_k^T}{\langle y_k, s_k \rangle^2} - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{\langle y_k, s_k \rangle}$$

Осталось договориться о выборе начального приближения H_0 . На этот случай нет универсального варианта, и обычно принимают либо $H_0 = I_n$, либо можно оценить гессиан в точке x_0 через конечные разности и обратить его.

Общая схема метода выглядит так:

```

1: procedure BFGS( $f, x_0, H_0, \varepsilon$ )
2:   for  $k = 0, 1, 2, \dots$  do
3:      $d_k \leftarrow -H_k \nabla f_k$ 
4:      $\alpha_k \leftarrow \text{LINESEARCH}(f, x_k, d_k)$ 
5:      $x_{k+1} \leftarrow x_k + \alpha_k d_k$ 
6:      $s_k \leftarrow x_{k+1} - x_k$ 
7:      $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$ 
8:      $H_{k+1} \leftarrow \text{UPDATEINVERSEDHESS}(H_k, s_k, y_k)$ 
9:     if  $\|\nabla f_{k+1}\| < \varepsilon$  then break
10:  return  $x_k$ 

```

К сожалению, ничто в этом мире не идеально, даже метод BFGS. Нам удалось сократить время итерации до $\mathcal{O}(n^2)$, но нам все еще нужно хранить плотную матрицу H_k , и это обеспечивает затраты на память $\mathcal{O}(n^2)$, что не всегда возможно. Тем не менее, метод BFGS имеет суперлинейную сходимость, что делает его популярным на практике.

L-BFGS

Метод L-BFGS - это модификация стандартного BFGS, которая позволяет сэкономить память и не хранить гессиан целиком. Все что от нас требуется - уметь умножать гессиан на произвольный вектор. Для этого метод поддерживает историю $\mathcal{H}_k = \{(s_{k-i}, y_{k-i})\}_{i=1}^l$ из последних l пар векторов. Вводится начальное приближение H_{k-l} :

$$H_{k-l} = \gamma_0^{(k)} I_n, \text{ где } \gamma_0^{(k)} = \frac{\langle y_{k-1}, s_{k-1} \rangle}{\langle y_{k-1}, y_{k-1} \rangle}$$

Новое приближение H_k получается из H_{k-l} путем l -кратного применения формулы из BFGS. Однако существует рекурсивная процедура, позволяющую подсчитывать произведение $d_k = -H_k \nabla f_k$ без формирования матриц в памяти. Выглядит она так:

```

1: procedure MULTIPLY( $v, \mathcal{H}, \gamma_0$ )
2:   if  $\mathcal{H} = \emptyset$  then return  $\gamma_0 v$ 
3:    $(s, y) \leftarrow$  последняя пара из  $\mathcal{H}$ 
4:    $\mathcal{H}' \leftarrow \mathcal{H}$  без последней пары
5:    $v' \leftarrow v - \frac{\langle s, v \rangle}{\langle y, s \rangle} y$ 
6:    $z \leftarrow \text{MULTIPLY}(v', \mathcal{H}', \gamma_0)$ 
7:   return  $z + \frac{\langle s, v \rangle - \langle y, z \rangle}{\langle y, s \rangle} s$ 
8:    $\gamma_0^{(k)} = \frac{\langle y_{k-1}, s_{k-1} \rangle}{\langle y_{k-1}, y_{k-1} \rangle}$ 
9:    $d_k = -\text{MULTIPLY}(\nabla f_k, \mathcal{H}_k, \gamma_0^{(k)})$ 

```

Таким образом, сложность итерации для вычисления направления d_k получается $\mathcal{O}(nl)$ (без учета сложности вычисления функции и градиента), а необходимая память для поддержания истории - $\mathcal{O}(nl)$. Типичное значение параметра размера истории $l = 10$ (при $k < l$ история \mathcal{H}_k состоит только из k пар). Заметим, что при $l = \infty$ мы в точности получаем метод BFGS, а при $l = 0$ - обычный градиентный спуск. На практике L-BFGS показывает себя лучше методов оптимизации первого порядка, хоть и не обладает суперлинейной сходимостью (гарантирована только линейная скорость сходимости).

2.8 Стандартные классы выпуклых задач: линейное программирование (LP), коническое квадратичное программирование (CQP/SOCP), полуопределенное программирование (SDP). Примеры задач для каждого из классов. Сводимость. L/CQ/SD-представимые множества и функции.

Все написано [здесь](#).

2.9 Симплекс-метод для решения задачи линейного программирования (в версии revised simplex method). Примеры сведения задачи линейного программирования к канонической форме. @elvarid

Чтобы понимать билет, вначале обратитесь к вопросам 1.12 и 1.13

2.9.1 Примеры сведения задачи линейного программирования к канонической форме

Смотреть пункт 1.12

2.9.2 Симплекс-метод для решения задачи линейного программирования (в версии revised simplex method)

Определение: x - угловая точка множества F , если $\nexists y, z \in F, y, z \neq x, \alpha \in (0, 1) : x = \alpha y + (1 - \alpha)z$

Утверждение: x - угловая точка множества $\{x | Ax = b, x \geq 0\} \Leftrightarrow \exists B, N : \{1, 2, \dots, n\} = B \cup N, |B| = p, |N| = n - p, x_N = 0, A_B = [A_{\cdot, i}]_{i \in B} \in \mathbb{R}^{p \times p}$ - невырожденная

Ремарка - B от слова базис

$Ax = b = A_B x_B + A_N x_N, x_B \in \mathbb{R}^p, x_N \in \mathbb{R}^{n-p}$, знаем, что $x_N = 0 \Rightarrow x_B = A_B^{-1} * b \geq 0$

На лекции было сказано что факт доказывается тривиально □

Перестановка одной координаты в множествах B и N соответствует переходу к соседней угловой точке.

Запишем вновь нашу задачу линейного программирования и условия ККТ чтобы пояснить как находить угловую точку.

Задача условной оптимизации:

$$\begin{cases} c^T x \rightarrow \min_x, x \in \mathbb{R}^n \\ Ax = b, \\ x \geq 0 \end{cases}$$

Функция Лагранжа:

$$L(x, \lambda, \mu) = c^T x - \lambda^T x + \mu^T (Ax - b)$$

Запишем условия ККТ:

$$\begin{cases} \nabla_x L(x, \lambda, \mu) = c - \lambda + A^T \mu = 0, \\ Ax = b, \\ \lambda \geq 0 \\ \lambda_i x_i = 0, i = 1, \dots, n \end{cases}$$

x - угловая точка, $x_N = 0, x_B = A_B^{-1} b \geq 0$ - то есть есть такое разбиение. Давайте восстановим значения λ и μ чтобы ККТ были выполнены.

$\lambda_B = 0$ - тогда произведение для всех индексов в ККТ будет равно нулю. Запишем следующее: $A^T \mu + c = \lambda$. Рассмотрим в этом уравнение подмножество, соответствующее B : $A_B^T \mu + c_B = \lambda_B = 0 \Rightarrow \mu = -A_B^{-T} c_B$ Таким образом мы восстановили μ . С этим знанием мы можем найти λ_N : $A_N^T \mu + c_N = \lambda_N \Rightarrow \lambda_N = c_N - A_N^T A_B^{-T} c_B = c_N - (A_B^{-1} A_N)^T c_B$. Таким образом мы удовлетворили всем условиям ККТ, кроме третьего условия, с которым еще надо повозиться.

Если уже $\lambda_N \geq 0 \Rightarrow x$ - решение.

Если это не выполнено, то ситуация следующая: $\exists q \in N : \lambda_q < 0$ и надо идти в другую угловую точку.

Теперь неформально можно описать шаг симплекс-метода : увеличиваем x_q до тех пор, пока какой-то из $(x_B)_i$ не станет равен нулю. Это и будет означать переходу в соседнюю угловую точку.

Распишем в виде формул.

Изначально есть $x : Ax = b, x_N = 0$. Вместо этой точки строим точку $x^+ : x_i^+ = 0, i \in N \setminus \{q\}$. Тогда получается следующее:

$$Ax^+ = b = A_B x_B^+ + A_q x_q^+ = A_B x_B$$

Поделим это выражение на невырожденную A^{-1} :

$$x_B^+ = x_B - A_B^{-1} A_q x_q^+$$

$A_B^{-1} A_q$ это какой-то вектор d размера p . x_q^+ это число. Тогда получается:

$$x_B^+ = x_B - d x_q^+ \geq 0$$

Если все компоненты вектора d меньше либо равны 0 то решений не будет. Если же найдутся $\exists j : d_j > 0$, то переход в ноль какой-то компоненте будет соответствовать следующему:

$$(x_B)_j - d_j x_q^+ = 0 \Rightarrow x_q^+ = \frac{(x_B)_j}{d_j}$$

Если же таких j несколько, то $x_q^+ = \min_{j: d_j > 0} \frac{(x_B)_j}{d_j}$ Осталось только перекинуть один индекс из множества B в N :

$$B = B \cup \{q\} \setminus \{j\}, N = N \setminus \{q\} \cup \{j\}$$

Вот так выглядит шаг симплекс-метода. Давайте докажем, что действительно на каждом шаге есть прогресс.

Утверждение: На каждом шаге симплекс-метода целевая функция уменьшается

Доказательство:

$$c^T x^+ = c_B^T X_B^+ + c_q x_q^+ = c_B^T (x_B - dx_q^+) + c_q x_q^+ = c_B^T x_B + x_q^+ (c_q - c_B^T d)$$

Рассмотрим $c_B^T d$. Это выражение равно $c_b^T A_B^{-1} A_q$, а ранее мы установили, что это равно $-\mu^T A_q = -(\lambda_q - c_q)$. Подставим этот результат в предыдущее выражение:

$$c_B^T x_B + x_q^+ (c_q - c_B^T d) = c^T x + x_q^+ (c_q + \lambda_q - c_q) = c^T x + \lambda_q x_q^+$$

Если мы посмотрим на знак последнего слагаемого, то он будет отрицательный, так как уже известно, что $\lambda_q < 0$, мы ее такой выбрали, а x_q^+ в свою очередь строго больше нуля (так как мы в ту сторону двигаем)

Значит $c^T x + \lambda_q x_q^+ < c^T x$

Таким образом значение целевой функции уменьшается на каждом шаге метода. □

Теперь формально опишем один шаг симплекс-метода.

Шаг симплекс-метода:

-
- 1: **procedure** REVISED SIMPLEX METHOD STEP($x : x_N = 0, x_B = A_B^{-1}b \geq 0$)
 - 2: Решить СЛАУ $A_B^T \mu = -c_B$
 - 3: $\lambda_N \leftarrow c_N + A_N^T \mu$
 - 4: Если $\lambda_N \geq 0$, то СТОП (найден оптимальное решение)
 - 5: Выбрать $q \in N, \lambda_q < 0$
 - 6: Решить СЛАУ $A_B d = A_q$
 - 7: Если $d \leq 0$, то СТОП (решения не существует)
 - 8: $x_q^+ \leftarrow \min_{i: d_i > 0} \frac{(x_B)_i}{d_i}$
 - 9: $j \leftarrow \operatorname{argmin}_{i: d_i > 0} \frac{(x_B)_i}{d_i}$
 - 10: $x_B^+ \leftarrow x_B - dx_q^+$
 - 11: $x_N^+ \leftarrow (0, 0, \dots, x_q^+, 0, \dots, 0)$
 - 12: $B \leftarrow B \cup \{q\} \setminus \{j\}$
 - 13: $N \leftarrow N \setminus \{q\} \cup \{j\}$
 - 14: **return** x
-

Прокомментируем еще раз алгоритм. Сначала по заданной точке мы находим μ и λ . Собственно изначально мы решаем СЛАУ относительно μ , и с помощью μ находим λ . Если все λ неотрицательные, то мы удовлетворили условиям ККТ и у нас есть решение, поэтому мы останавливаемся. Если же условие не выполнено, то мы выбираем какую-то компоненту λ_q меньше нуля и дальше начинаем двигать точку x_q на вектор d . Вектор d мы находим через решение СЛАУ. Если все компоненты этого вектора неположительные, то это означает что целевую функцию можно увести в минус бесконечность, тогда решений не существует. Если же хотя бы один d положительный, то тогда мы совершаем сдвиг согласно написанным формулам. И в конце перекидываем индексы между B и N .

По итогу что мы имеем метод, который сходится за конечное число шагов (благодаря доказанному выше утверждению про уменьшение целевой функции на каждом шаге). Однако, справедлива будет для общего случая следующая оценка на число шагов - C_n^p (число сочетаний) по построению метода. На практике эта сложность практически неактуальна (даже теоретический пример для такой сложности предъявили не сразу после создания метода), часто метод находит оптимальную точку за линейное время от p .

Стоит обсудить еще следующую особенность симплекс-метода. Как видно по алгоритму, здесь нет какого-то понятия вроде эпсилон-точности, как в предыдущих рассмотренных методах. Либо мы в итоге приходим в оптимум, либо нет. Это связано с особенностью хождения по угловым точкам. В какой-то момент мы можем пройти очень маленькое расстояние, так как две угловые точки находятся очень близко друг к другу, но это не значит, что оптимум близко, хотя изменение целевой функции будет минимальным. Из-за этой причины нет смысла в каком-то заранее задаваемом параметре точности.

Информацию далее для сдачи экзамена учить не надо, но если вы будете это знать, то это будет полезно (информация с консультации)

Возникающие проблемы при использовании.

- На каждом шаге у матрицы A_B меняется только один столбец, поэтому в некоторых реализациях хранится только LU разложение, в котором изменения происходят с помощью специальных алгоритмов.
- Момент связанный с выбором компоненты q . Изначально была стратегия $q^* = \operatorname{argmin}_q \lambda_q$. Данная стратегия обоснована анализом уменьшения функции на итерации метода. На практике в этом месте это

было бы слишком дорого, поэтому в этом моменте возникают различные эвристики которые хорошо выбирают q .

- Момент связанный с вырожденными точками. Угловая точка называется вырожденной, если какая-то или несколько из компонент из B равны нулю. В таком случае x_q^+ двигать за пределы множества нельзя, и эта компонента будет равна 0. Тогда изменение целевой функции не уменьшится, метод может даже заикнуться между несколькими вырожденными точками. В таком случае надо вставить какой-то АНТИЦИКЛИН (что-то вроде добавления нормального шума к вектору b из $Ax = b$).
- Момент связанный с выбором начальной угловой точки. Не всегда очевидно как выбрать начальную точку, один из возможных способов решения это так называемый метод "первой фазы". Он заключается в формировании какой-то специальной задачи, про которую известно что ее решение будет содержать какую-то угловую точку нашего изначального множества ($Ax = b$), и для этой задачи мы без проблем умеем выбирать начальную угловую точку. То есть первую фазу мы решаем с помощью симплекс-метода, получаем начальную точку и решаем уже с этой начальной точкой нашу задачу, так называемая вторая фаза.

2.10 Метод Ньютона и логарифмических барьеров для выпуклых задач условной оптимизации с ограничениями вида равенств и неравенств. @arinaruck

Что будет в билете: Сначала научимся решать задачу условной оптимизации с ограничениями только вида равенств, потом введем барьерную функцию для ограничений вида неравенств и приведем исходную задачу к эквивалентному виду только с равенствами в ограничениях

Задача с ограничениями только вида равенств:

$$\begin{cases} f(x) \rightarrow \min_x, x \in D \subseteq \mathbb{R}^n, f_{\text{вып}} \in C^2 \\ Ax = b, A \in \mathbb{R}^{p \times n}, p < n, \text{rank} A = p \end{cases}$$

$$F = \{x \in D : Ax = b\}$$

$x_{k+1} = x_k + \alpha_k d_k$, d_k направление спуска для f

Хотим сохранять допустимость, то есть чтобы все $x_i \in F : x_0 \in F \rightarrow x_1 \in F \rightarrow \dots$

$$f(x_k + d) \approx m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \rightarrow \min_d, B_k \in S_{++}^n, A(x_k + d) = b$$

Чтобы сохранить допустимость решаем задачу QP:

$$\begin{cases} f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d \rightarrow \min_d \\ Ad = 0 \end{cases}$$

Если $B_k = I$ получим метод проекции градиента

$$\begin{cases} m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T d = \frac{1}{2} \|d + \nabla f(x_k)\|^2 + \text{const} \rightarrow \min_d \\ Ad = 0 \end{cases} \implies d = \text{pr}_{Ad=0}(-\nabla f(x_k))$$

Если $B_k = \nabla^2 f(x_k)$

$$\begin{cases} m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d = \frac{1}{2} \|d + [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)\|_{\nabla^2 f(x_k)}^2 + \text{const} \rightarrow \min_d \\ Ad = 0 \end{cases}$$

$$\|x\|_B^2 = x^T B x, B \in S_{++}^n$$

Уже нельзя так просто взять проекцию, так как измененная норма

Запишем Лагранжиан для функции по d :

$$L(d, \mu) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d + \mu(Ad)$$

Выпишем ККТ:

$$\begin{cases} \nabla_d L(d, \mu) = \nabla f(x_k) + B_k d + A^T \mu = 0 \\ Ad = 0 \end{cases}$$

Получили СЛАУ относительно d, μ

Перепишем в блочном виде:

$$\begin{bmatrix} B_k & A^T \\ A & 0 \end{bmatrix} \cdot \begin{bmatrix} d \\ \mu \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) \\ 0 \end{bmatrix}, d \in \mathbb{R}^n, \mu \in \mathbb{R}^p$$

Обозначим

$$K = \begin{bmatrix} B_k & A^T \\ A & 0 \end{bmatrix}$$

$inertia(K) = (n_{+1}, n_{-1}, n_0)$ — количество положительных, отрицательных и нулевых собственных значений K

Если $B_k \succ 0$, $rank A = p \implies inertia(K) = (n, p, 0)$, так как нет нулевых собственных значений, матрица K невырожденная, тогда решение исходной системы единственно.

K симметричная и не положительно определенная, если есть хотя бы одно ограничение вида равенства. Можем решать систему, например, с помощью LDL (если не очень большая), или метода типа CG (Крыловского вида), если очень.

$$x_{k+1} = x_k + \alpha_k d_k$$

Для корректности нужно:

- 1) $x_{k+1} \in F$ — точка допустима
- 2) $\nabla f(x_k)^T d_k < 0$ — d_k направление спуска

Проверим:

- 1) $x_{k+1} \in F \Leftrightarrow x_{k+1} \in \{x \in D : Ax = b\}$
 $Ax_{k+1} = A(x_k + \alpha_k d_k) = Ax_k + \alpha_k Ad_k = b$

Должны прогарантировать, что возьмем α_k , которая не выведет нас из домена D , для этого можем, например, выполнить поиск следующим образом

```

1: procedure  $\alpha_{max}(\alpha_0 = 1)$ 
2:   for  $k = 0, 1, 2, \dots$  do
3:     if  $x_k + \alpha_k d_k \in D$  then break
4:      $\alpha_{k+1} \leftarrow \alpha_k / 2$ 
5:   return  $\alpha_k$ 

```

Из a_k будем начинать одномерный поиск для условий Армихо/Вульфа (для простых доменов можем также находить аналитически)

$$2) \nabla f(x_k)^T d_k = d_k^T (-B_k d - \mu_k A^T) = -d_k^T B_k d - \mu_k (Ad_k)^T = -d_k^T B_k d < 0$$

То есть d_k является направлением спуска

Критерий останова:

$$\|\nabla_x L(x_{k+1}, \mu)\| = \|\nabla f(x_{k+1}) + A^T \mu_{k+1}\| < \varepsilon$$

Скорость сходимости полностью наследует скорость сходимости для безусловного случая (т. е. зависит только от выбора B_k в приближении)

Теперь перейдем к случаю, когда ограничения могут быть не только вида равенств но и неравенств

$$\begin{cases} f(x) \rightarrow \min_x, x \in D \subseteq \mathbb{R}^n, f, g_i \text{ вып} \in C^2 \\ g_i(x) \leq 0, i = 1, \dots, m \\ Ax = b, A \in \mathbb{R}^{p \times n}, p < n, rank A = p \end{cases}$$

Пусть выполнено условие Слейтера, то есть $\exists \hat{x} : A\hat{x} = b, g_i(\hat{x}) < 0 \forall i$

Введем индикаторную функцию

$$I(u) = \begin{cases} 0, u \leq 0 \\ +\infty, \text{ иначе} \end{cases}$$

Тогда можем переписать исходную задачу в виде

$$\begin{cases} f(x) + \sum_{i=1}^m I(g_i(x)) \rightarrow \min_x \\ Ax = b, A \in \mathbb{R}^{p \times n}, p < n, rank A = p \end{cases}$$

Хотелось бы применить метод, который описали до этого, но индикатор не гладкий, хотим его сгладить

$$I_\tau(u) = -\frac{1}{\tau} \log(-u), \quad \tau > 0 \text{ — логарифмический барьер}$$

$$I_\tau(u) \xrightarrow{\tau \rightarrow +\infty} I(u)$$

Тогда перейдем к

$$\begin{cases} f(x) - \frac{1}{\tau} \sum_{i=1}^m \log(-g_i(x)) \rightarrow \min_x \\ Ax = b, A \in \mathbb{R}^{p \times n}, p < n, \text{rank } A = p \end{cases}$$

$D = \{x : g_i(x) < 0\}$ выпуклая гладкая задача оптимизации

Рассмотрим $\{\hat{x}(\tau), \hat{\mu}(\tau) | \tau > 0\}$ — центральный путь

Пример: (См. рисунок)

$$\begin{cases} c^T x \rightarrow \min_x \\ Ax \leq b \end{cases}$$

$$F_\tau(x) = c^T x - \frac{1}{\tau} \sum_{i=1}^m \log(-a_i^T x + b) \rightarrow \min_x$$

$$\frac{1}{\tau} \sum_{i=1}^m \log(-a_i^T x + b) = \phi_\tau(\hat{x}(\tau))$$

$$\nabla_x F_\tau = c + \nabla_x \phi_\tau(\hat{x}(\tau)) = 0 \implies \nabla_x \phi_\tau(\hat{x}(\tau)) = -c$$

Будем идти по центральному пути, начиная из центральной точки, постепенно увеличивая τ , повышая "вес" $c^T x$

Чем больше τ , тем ближе линия уровня к границе

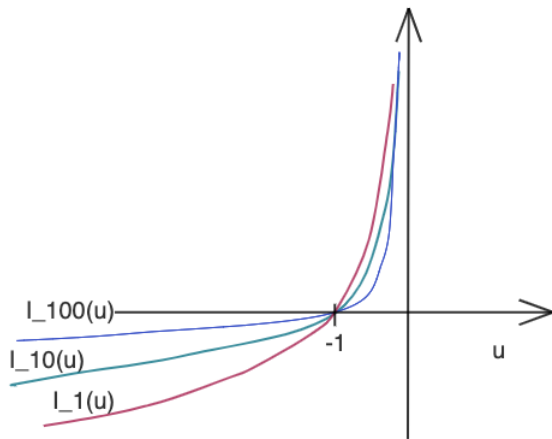


Рис. 18: $I_\tau(u)$ функции.

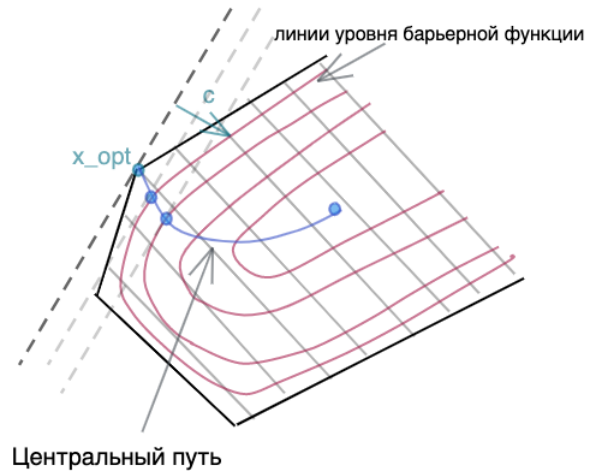


Рис. 19: Центральный путь для примера.

$$L_\tau(x, \mu) = f(x) - \frac{1}{\tau} \sum_{i=1}^m \log(-g_i(x)) + \mu^T (Ax - b)$$

$$\nabla_x L_\tau(\hat{x}, \hat{\mu}) = \nabla_x f(\hat{x}) + \frac{1}{\tau} \sum_{i=1}^m \frac{-\nabla_x g_i(\hat{x})}{g_i(\hat{x})} + A^T \hat{\mu} = 0$$

При этом для исходной задачи:

$$L_\tau(x, \mu, \lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \mu^T (Ax - b)$$

$$\nabla_x L_\tau(\hat{x}, \hat{\mu}, \hat{\lambda}) = \nabla_x f(\hat{x}) + \sum_{i=1}^m \nabla_x \hat{\lambda}_i g_i(\hat{x}) + A^T \hat{\mu} = 0$$

$$\text{Тогда введем обозначение: } \hat{\lambda}_i = \frac{-1}{\tau g_i(\hat{x})}$$

Для $(\hat{x}(\tau), \hat{\mu}(\tau), \hat{\lambda}(\tau))$:

$$\begin{cases} \nabla_x L_\tau(\hat{x}, \hat{\mu}, \hat{\lambda}) = 0 \\ A\hat{x} = b \\ \hat{\lambda}_i > 0 \quad \forall i = 1, \dots, m \\ \hat{\lambda}_i g_i(\hat{x}) = -\frac{1}{\tau} \quad \forall i = 1, \dots, m \end{cases}$$

При $\tau \rightarrow +\infty$ $(\hat{x}(\tau), \hat{\mu}(\tau), \hat{\lambda}(\tau))$ удовлетворяет ККТ для исходной задачи

Введем двойственную функцию:

$$g(\lambda, \mu) = \inf_x L(x, \mu, \lambda)$$

$$g(\lambda, \mu) \leq f_{opt} \quad \forall \mu, \forall \lambda > 0$$

$$g(\hat{\lambda}, \hat{\mu}) = L(\hat{x}, \hat{\mu}, \hat{\lambda}) \leq f_{opt}$$

$$f(\hat{x}) - f_{opt} \leq f(\hat{x}) - L(\hat{x}, \hat{\mu}, \hat{\lambda}) = f(\hat{x}) - f(\hat{x}) + \sum_{i=1}^m \hat{\lambda}_i g_i(\hat{x}) + \hat{\mu}^T (A\hat{x} - b) = \frac{m}{\tau}$$

$$(\hat{x} \text{ допустимое} \implies A\hat{x} = b)$$

$$\frac{m}{\tau} = \varepsilon, \quad \tau = \frac{m}{\varepsilon}$$

На практике постепенно увеличиваем τ

Приведем общую схему метода:

-
- 1: **procedure** *logarithmicbarrier*($x_0 \in D, Ax_0 = b, g_i(x_0) < 0 \quad \forall i, \varepsilon, \tau_0, \nu > 1$)
 - 2: **for** $k = 0, 1, 2, \dots$ **do**
 - 3: найти $\hat{x}(\tau_k)$ решение задачи

$$\begin{cases} f(x) - \frac{1}{\tau_k} \sum_i \log(-g_i(x)) \rightarrow \min_x \\ Ax = b \end{cases}$$

с помощью метода Ньютона из начального приближения x_k

- 4: $x_{k+1} \leftarrow \hat{x}(\tau_k)$
 - 5: **if** $\frac{m}{\tau_k} \leq \varepsilon$ **then break**
 - 6: $\tau_{k+1} \leftarrow \min\left(\frac{m}{\varepsilon}, \tau_k \cdot \nu\right)$
 - 7: **return** α_k
-

ν отвечает за трейд-офф внешних и внутренних итераций: чем больше ν , тем быстрее дойдем до $\tau_k = \frac{m}{\varepsilon}$, тем меньше будет внешних итераций, но тем хуже будет стартовая точка x_k , то есть тем больше внутренних итераций.

В качестве x_0 необходимо подать внутреннюю точку, иногда ее поиск довольно сложный, тогда нужно воспользоваться методом первой фазы (как в симплекс-методе)

2.11 Субградиентный метод для негладких задач оптимизации. Различные способы выбора длины шага. Субдифференциальное исчисление, примеры вычисления субдифференциалов. @sergevkim

Возможно, один из самых простых билетов

Хотим придумать что-то для функции, которая будет дифференцируемой не в каждой точке. Решение - коль скоро мы не можем подсчитать сам дифференциал, можно подсчитать в точке какой-нибудь его аналог. Спойлер: этот аналог мы будем называть субдифференциалом.

Definition 1

Пусть выполнено, что $\forall x \exists s : F(x) \geq F(x_0) + s^T(x - x_0)$

Тогда вектор s^T будем называть субградиентом функции F в точке x_0

Definition 2

$s : \forall x \exists s : F(x) \geq F(x_0) + s^T(x - x_0)$

Множество всех субградиентов в точке x_0 - субдифференциал, обозначение $\partial F(x_0)$

Утверждение 0. Если функция дифференцируема в точке x_0 , то субдифференциал будет состоять из единственного вектор - градиента в этой точке x_0

Утверждение 1. Субдифференциал - выпуклое множество

Доказывается по определению субдифференциала. Выписываем два определения субдифференциала и складываем неравенства с коэффициентами α и $(1 - \alpha)$

Утверждение 2. Необходимое условие минимума в точке x_0 аналогично дифференцируемому случаю, только теперь надо проверять, что $0 \in \partial F(x_0)$. Утверждение 1 позволяет решать задачу необходимого условия это геометрически

Сам шаг метода звучит просто: это градиентный метод, только вместо градиента в точке берём какой-нибудь субградиент из субдифференциала (при этом по Утверждению 0 этот "какой-нибудь" субградиент будет в точности совпадать с градиентом).

Проблема: мы теперь не можем использовать правило Армихо/Вульфа, т.к. существуют примеры того, что двигаясь вдоль антисубградиента мы будем увеличивать функцию (в видеосике он приводит пример того, что линии уровня из себя представляют сплюсненные ромбики, задача задаётся формулой $F(x_1, x_2) = |x_1| + \gamma * |x_2| \rightarrow \min$).

Альтернативные способ задания такой последовательности:

1) $\alpha_k = \alpha_0$ - если мы возьмём константную длину шага, метод сходиться не будет вообще

2) $\alpha_k = \frac{\alpha_0}{\|z_k\|}$ - делим на норму субградиента. Будем ближе к оптимуму, но сходимости всё ещё нет

3) $\{\alpha_k\}$ - выбрать такую последовательность, что $\sum_{i=0}^{\infty} \alpha_i = \infty$, однако $\sum_{i=0}^{\infty} \alpha_i^2 < \infty$

Пример: $\alpha_k = \alpha_0 / \sqrt{k+1}$ или такая последовательность, которая сходится к нулю, но не очень быстро

Критерий останова: его нет. Просто идём максимальное количество итераций. В памяти храним лучшую точку, так как метод не гарантирует сходимости к минимуму.

Скорость сходимости - $O(\frac{1}{\sqrt{k}})$

Примеры субдифференциального исчисления лучше всего смотреть в семинарской записи, там хорошо написано: [Примеры](#)

2.12 Проксимальный градиентный метод для задачи композитной оптимизации. Примеры вычисления проксимальных операторов. @poly_nomial

2.13 Стохастический градиентный спуск, его скорость сходимости. Различные стратегии выбора длины шага. Стохастические методы оптимизации с линейной скоростью сходимости (SAG и SVRG). @Bitchert

Рассматриваем задачу оптимизации суммы большого количества функций:

$$F(x) = \frac{1}{n} \sum_{i=1}^n f_i(x) \rightarrow \min_x, \quad f_i \in C^1, \quad n \gg 1$$

$$\nabla F(x) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x)$$

Рассмотрим стоимости вычисления различных величин:

Величина	Стоимость вычисления
$f_i(x)$	$O(q)$
$\nabla f_i(x)$	$O(q)$
$F(x)$	$O(nq)$
$\nabla F(x)$	$O(nq)$

Вычисляем функции $f_i(x)$ через граф вычислений, градиент это обратный проход по графу - стоимость та же. Видим, что стоимость вычисления оракулум функции F и её градиента линейно зависит от размера выборки. Хотим метод, у которого стоимость итерации не будет зависеть от размера выборки.

2.13.1 SGD (стохастический градиентный метод)

Посмотрим на одну итерацию метода:

- 1: **procedure** SGD
- 2: $i_k \leftarrow \text{UNIFORM}(1, 2, \dots, n) \quad (O(1))$
- 3: $g_k \leftarrow \nabla f_{i_k}(x_k) \quad (O(q))$
- 4: $x_{k+1} \leftarrow x_k - \alpha_k g_k \quad (O(d))$

Стоимость итерации $O(q)$. Видно, что g_k является несмещенной оценкой для $\nabla F(x)$:

$$\mathbb{E}_{i \sim \text{UNIFORM}} g_k = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) = \nabla F(x)$$

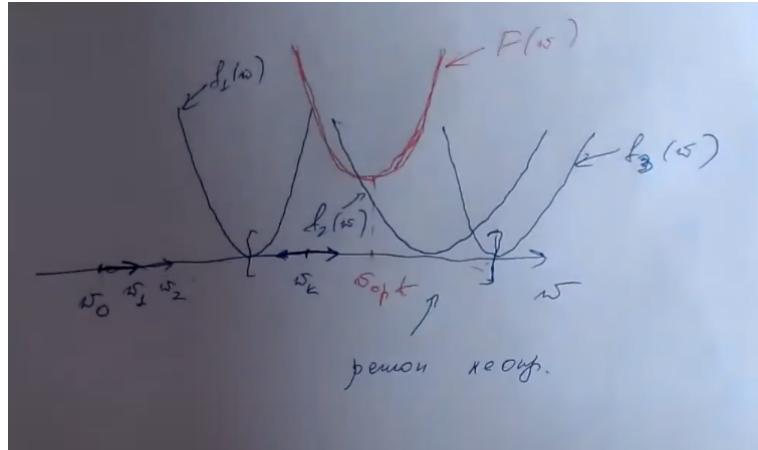
Также можно сэмплировать не один индекс, а какое-то множество индексов фиксированного размера:

- 1: **procedure** SGD + MINI-BATCHES
- 2: $I_k \subset \text{UNIFORM}(1, 2, \dots, n) \quad (O(|I_k|))$

- 3: $g_k \leftarrow \frac{1}{|I_k|} \sum_{i_k \in I_k} \nabla f_{i_k}(x_k) \quad (O(q|I_k|))$
 4: $x_{k+1} \leftarrow x_k - \alpha_k g_k \quad (O(d))$

Очевидно, что это тоже несмещенная оценка. Дисперсия стохастической оценки падает как $O(\frac{1}{\sqrt{k}})$, а увеличение стоимости происходит как $O(k)$. Таким образом выгодно использовать именно маленькие батчи. Хотим посмотреть как будет работать стохастический градиент, приведем пример одномерного случая:

$$F(w) = \frac{1}{n} \sum_{i=1}^n \underbrace{(y_i - wx_i)^2}_{f_i(w)} \rightarrow \min_{w \in \mathbb{R}}$$



Метод стохастического градиентного спуска вообще не является методом спуска, мы не гарантируем, что на каждой итерации значение функции уменьшится. Видно, что подходя к оптимуму мы попадаем в регион неопределенности, ведь даже в оптимальной точке наша оценка градиента g_k не равна нулю, как нам того хотелось бы. Метод застревает около оптимума и начинает бесконечные колебания. У нас нет возможности оценить значение функции F , таким образом мы можем просто запускать метод на какое-то фиксированное число итераций, либо по каким-то косвенным признакам отслеживать (допустим в машинном обучении смотреть качество на отложенной выборке). В дальнейшем мы рассмотрим решение этой проблемы с помощью выбора длины шага и вариаций оценки градиента. Пока же предлагается оценить сходимость метода SGD.

2.13.2 Скорость сходимости

$$x_{k+1} = x_k - \alpha_k g_k, \quad \mathbb{E} g_k = \nabla F(x_k)$$

Начинаем с невязки по точкам:

$$\|x_{k+1} - x_*\|^2 = \|x_k - \alpha_k g_k - x_*\|^2 = \|x_k - x_*\|^2 - 2\alpha_k g_k^T (x_k - x_*) + \alpha_k^2 \|g_k\|^2$$

Теперь берем матожидание по случайности на k -ой итерации:

$$\mathbb{E}_{i_k} : \quad \mathbb{E}_{i_k} \|x_{k+1} - x_*\|^2 = \|x_k - x_*\|^2 - 2\alpha_k \nabla F(x_k)^T (x_k - x_*) + \alpha_k^2 \mathbb{E}_{i_k} \|g_k\|^2$$

Предполагаем, что F - выпуклая функция, тогда:

$$\begin{aligned} F(x_*) &\geq F(x_k) + \nabla F(x_k)^T (x_* - x_k) \\ F(x_k) - F(x_*) &\leq \nabla F(x_k)^T (x_k - x_*) \end{aligned}$$

Теперь:

$$\begin{aligned} \alpha_k \nabla F(x_k)^T (x_k - x_*) &= \frac{\|x_k - x_*\|^2}{2} + \frac{\alpha_k^2 \mathbb{E}_{i_k} \|g_k\|^2}{2} - \frac{\mathbb{E}_{i_k} \|x_{k+1} - x_*\|^2}{2} \\ \alpha_k (F(x_k) - F(x_*)) &\leq \frac{\|x_k - x_*\|^2}{2} + \frac{\alpha_k^2 \mathbb{E}_{i_k} \|g_k\|^2}{2} - \frac{\mathbb{E}_{i_k} \|x_{k+1} - x_*\|^2}{2} \end{aligned}$$

Теперь давайте возьмем матожидание по всей случайности:

$$\mathbb{E} : \quad \alpha_k (\mathbb{E} F(x_k) - F(x_*)) \leq \frac{\mathbb{E} \|x_k - x_*\|^2}{2} + \frac{\alpha_k^2 \mathbb{E} \|g_k\|^2}{2} - \frac{\mathbb{E} \|x_{k+1} - x_*\|^2}{2}$$

Теперь просуммируем по индексам от 0 до k :

$$\sum_{i=0}^k \alpha_i (\mathbb{E}F(x_i) - F(x_*)) \leq \frac{\|x_0 - x_*\|^2}{2} + \frac{\sum_{i=0}^k \alpha_i^2 \mathbb{E}\|g_i\|^2}{2} - \frac{\mathbb{E}\|x_{k+1} - x_*\|^2}{2}$$

Вспоминаем, что F - выпуклая, записываем неравенство Йенсена:

$$\mathbb{E}F\left(\frac{\sum_{i=0}^k \alpha_i x_i}{\sum_{i=0}^k \alpha_i}\right) - F(x_*) \leq \frac{\sum_{i=0}^k \alpha_i (\mathbb{E}F(x_i) - F(x_*))}{\sum_{i=0}^k \alpha_i} \leq \frac{\|x_0 - x_*\|^2 + \sum_{i=0}^k \alpha_i^2 \mathbb{E}\|g_i\|^2}{2 \sum_{i=0}^k \alpha_i}$$

Теперь ограничим начальную невязку как R^2 , положим, что все $f_i \in C_G^{0,0}$, тогда можно ограничить норму квадрата градиента как G^2 :

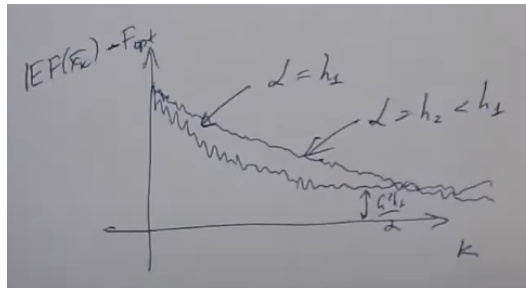
$$\mathbb{E}F(\bar{x}_k) - F(x_*) \leq \frac{R^2 + G^2 \sum_{i=0}^k \alpha_i^2}{2 \sum_{i=0}^k \alpha_i}$$

Таким образом с точки зрения теоремы мы можем гарантировать что-либо только для средней точки \bar{x}_k , однако "если провести численные эксперименты, мы увидим, что разница не существенная".
Теперь рассмотрим разные стратегии выбора длины шага α_k :

2.13.3 Стратегии выбора длины шага

- **Константный** $\alpha_k = h$:

$$\frac{R^2 + G^2 h^2 (k+1)}{2(k+1)h} = \frac{R^2}{2h(k+1)} + \frac{G^2 h}{2} \xrightarrow{k \rightarrow \infty} \frac{G^2 h}{2}$$



Видно, что при меньшем шаге мы сходимся медленнее, зато в меньшее значение невязки. Однако это всё про матожидание, и тут на помощь нам приходит неравенство Маркова:

$$\xi := F(\bar{x}_k) - F(x_*) \geq 0 \Rightarrow p(F(\bar{x}_k) - F(x_*) \geq a) \leq \frac{\mathbb{E}F(\bar{x}_k) - F(x_*)}{a}$$

- **Устремить к нулю** α

$$1. \sum_{i=0}^k \alpha_i = \infty, \quad \sum_{i=0}^k \alpha_i^2 < \infty$$

$$\alpha_i = \frac{h}{(i+1)^\tau}, \quad \tau \in \left(\frac{1}{2}, 1\right]$$

$$2. \sum_{i=0}^k \alpha_i = \infty, \quad \sum_{i=0}^k \alpha_i^2 = \infty, \quad \lim_{k \rightarrow \infty} \frac{\sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i} = 0$$

$$\alpha_i = \frac{h}{(i+1)^\tau}, \quad \tau \in \left(0, \frac{1}{2}\right]$$

$$\tau_{opt} = \frac{1}{2} \Rightarrow \text{скорость сходимости будет } O\left(\frac{\ln(k)}{\sqrt{k}}\right) = \tilde{O}\left(\frac{1}{\sqrt{k}}\right).$$

Составим табличку скоростей сходимости методов и их стохастических аналогов:

Метод	Функция	Скорость сходимости	
		Обычный	Стохастический
Градиентный	$f \in C_L^{1,1}$ и сильно выпуклая	$O(C^k)$	$O(1/k)$
Градиентный	$f \in C_L^{1,1}$ и выпуклая	$O(1/k)$	$O(1/\sqrt{k})$
Субградиентный	$f \in C_L^{0,0}$ и выпуклая	$O(1/\sqrt{k})$	$O(1/\sqrt{k})$

Видно, что с такой скоростью сходимости для требуемой точности необходимо огромное число итераций, хотим метод с линейной скоростью сходимости.

2.13.4 SAG и SVRG

SAG

$$\begin{aligned} \text{GD : } \quad x_{k+1} &= x_k - \alpha_k \frac{1}{n} \sum_{i=1}^n \nabla f_i(x_k) \\ \text{SAG : } \quad x_{k+1} &= x_k - \alpha_k \frac{1}{n} \sum_{i=1}^n \nabla f_i(v_i^k) \end{aligned}$$

Пытаемся сделать так, чтобы в пределе направление оптимизации приходило в полный градиент. На каждой итерации меняем одну точку v_i^k . Рассмотрим итерацию метода:

- 1: **procedure** SAG
- 2: $i_k \leftarrow \text{UNIFORM}(1, 2, \dots, n)$
- 3: $v_{i_k}^k \leftarrow x_k; v_j^k \leftarrow v_j^{k-1} \forall j \neq i_k$
- 4: $g_k \leftarrow \frac{1}{n} \sum_{i=1}^n \nabla f_i(v_i^k) = g_{k-1} + \frac{1}{n} (\nabla f_{i_k}(v_{i_k}^k) - \nabla f_{i_k}(v_{i_k}^{k-1}))$
- 5: $x_{k+1} \leftarrow x_k - \alpha_k g_k$

В методе возникает память, которая имеет размер, зависящий от количества объектов: $O(nd)$.

Для скорости сходимости метода SAG верно следующее утверждение (оно вида теоремы, но главный по оптам сказал можно не доказывать и даже не формулировать строго):

Пусть $F \in C_L^{1,1}$ и μ сильно выпуклая, длина шага $\alpha_k = \frac{1}{16L}$, тогда для невязки по функции для метода SAG верна следующая оценка:

$$\mathbb{E}F(x_k) - F(x_*) \leq \left(1 - \min\left(\frac{\mu}{16L}, \frac{1}{8n}\right)\right)^k C$$

Имеет обратить внимание на то, что k - это номер стохастической итерации, то есть он сходится быстрее градиентного спуска где-то в n раз. Также у него есть возможность адаптивного подбора константы Липшица. Однако он работает только для выпуклых и сильно выпуклых функций. И еще стоит обратить внимание в связи с этим методом, на то, что он оптимизирует именно сумму функций и направлен на это. То есть в более общем случае, когда мы оптимизируем функцию вида матожидание по распределению мы не можем использовать SAG:

$$F(x) = \mathbb{E}_{q(y)} f(x, y) \rightarrow \min_x$$

Ну и далее мы рассмотрим метод, похожий на SAG, однако который можно применять на более общие задачи вида выше.

SVRG Идея следующая, в какой-то точке потратить время и посчитать полный градиент функции, потом использовать его для построения оценок для g_k :

$$\begin{aligned} \tilde{x} : \quad \tilde{\mu} &= \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}) = \nabla F(\tilde{x}) \\ g_k &= \nabla f_{i_k}(x_k) - \nabla f_{i_k}(\tilde{x}) + \tilde{\mu} \\ \mathbb{E}g_k &= \mathbb{E}\nabla f_{i_k}(x_k) - \mathbb{E}\nabla f_{i_k}(\tilde{x}) + \mathbb{E}\tilde{\mu} = \nabla F(x_k) \end{aligned}$$

Таким образом мы в методе совершаем итерации вида: оцениваем полный градиент в точке \tilde{x}_0 , делаем какое-то количество стохастических итераций, оцениваем полный градиент в точке \tilde{x}_1 и так далее. Его можно применять для общих функций вида матожидание по распределению, у него есть теорема аналогичная как для SAG, что он сходится линейно с константным шагом обратно пропорциональным константе Липшица, у него есть критерий останова связанный с нормой $\tilde{\mu}$ и у него есть возможность адаптивного подбора константы Липшица, то есть его можно запускать что называется "из коробки".

2.14 Риманова оптимизация, её основные компоненты (риманово многообразие, касательное пространство, римановый градиент, операции ретракции и транспортировки вектора). Римановый градиентный спуск, примеры применения. Примеры римановых многообразий. @isadrtdinov

Прежде всего возникает вопрос: а зачем вообще нам нужны нетрадиционные методы оптимизации, раз мы уже разобрали градиентные спуски на любой вкус и цвет, методы Ньютона и прочая, и прочая, и прочая? Дело в том, что разобранные нами методы для условной оптимизации подходят не для всех множеств (например, мы можем захотеть оптимизировать функцию $f(x)$ по шару $x^T x = 1$, а метод барьеров не умеет работать с множествами без внутреннейности).

Рассмотрим еще один простой пример. Частая проблема в рекуррентных сетях - затухание градиента (когда норма весов $\|W\| < 1$) или взрыв градиента (соответственно, $\|W\| > 1$). Хорошей практикой здесь считается брать матрицу весов W ортогональной, что позволяет контролировать ее норму (она становится единичной). Прелесть римановской оптимизации в том, что она позволяет оптимизировать функции по очень сложным множествам (типа множества ортогональных матриц или шара без внутреннейности).

Центральным понятием в римановской оптимизации является риманов градиент. Рассмотрим ради простоты всю конструкцию на сфере S^{n-1} . Рассмотрим некоторую точку сферы x и ее окрестность. **Римановым градиентом** $\text{grad } f(x)$ называется направление кривой наибольшего возрастания в окрестности точки x . Он представляет собой вектор, лежащий в касательной плоскости к сфере в точке x .

Казалось бы все, наши проблемы решены! Берем и делаем шаг вдоль направления риманова антиградиента. Но есть проблема: такой шаг выведет нас за пределы нашего множества:

$$x_{k+1} = x_k - \alpha_k \text{grad } f(x_k) \notin S^{n-1}$$

Нам нужно как-то шагнуть в сторону уменьшения функции, но при этом остаться на сфере. Для этого вводится операция **ретракции** - это в некотором смысле проецирование риманова градиента на наше множество. По итогу, чтобы задать риманов градиентный спуск (RGD), нам нужно определить риманов градиент и операцию ретракцию.

$$x_{k+1} = R_{x_k}(-\alpha_k \text{grad } f(x_k))$$

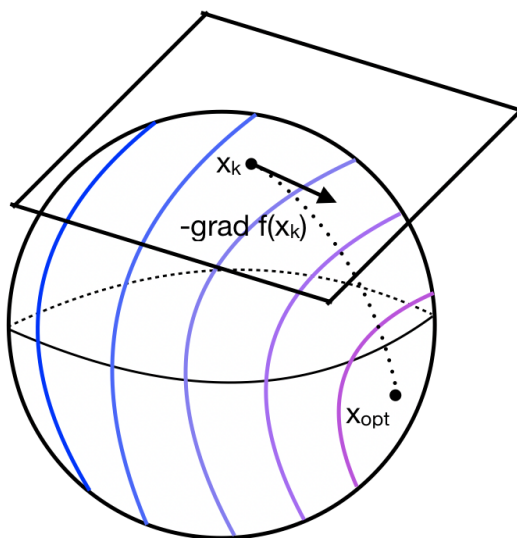


Рис. 20: Риманов антиградиент в точке x_k . Цветными кривыми показаны линии уровня функции $f(x)$.

Теперь перейдем к формализму (наверное, правильнее начинать рассказывать билет отсюда, а все, что выше, приведено для понимания общей логики происходящего).

Многообразием M размерности d называется множество, удовлетворяющее двум свойствам:

1. $\forall x \in M, U$ - окрестность $x \exists$ отображение $\phi : U \rightarrow \mathbb{R}^d$, биективно переводящее U в некоторое открытое множество. Пара (U, ϕ) называется картой множества M .
2. Для карт $(U, \phi), (W, \psi)$ в точке $x : U \cap W \neq \emptyset$ отображения $\phi \circ \psi^{-1}$ и $\psi \circ \phi^{-1}$ дифференцируемы бесконечное число раз как функции $\mathbb{R}^d \rightarrow \mathbb{R}^d$.

При этом $\phi(x) \in \mathbb{R}^d$ называются локальными координатами, а если $M \subset \mathbb{R}^n$, то $x \in \mathbb{R}^n$ называются глобальными координатами.

Примеры многообразий:

1. Окружность $S^1 = \{(x_1, x_2) : x_1^2 + x_2^2 = 1\}$. Тогда $t \in [0, 2\pi)$ - это локальные координаты, а в качестве отображения $\phi(x)$ можно взять подходящую обратную тригонометрическую функцию.
2. Все пространство \mathbb{R}^n . В качестве биекции берем тривиальную, а локальные координаты совпадают с глобальными.
3. Манифолд Штифеля $\text{St}(m, n) = \{X \in \mathbb{R}^{n \times m} | X^T X = I_m\}$ (в сущности, множество ортогональных матриц).

Касательным пространством в точке $x \in M$ называется множество $T_x M = \{\xi \in \mathbb{R}^n | \xi = \gamma'(0), \text{ где } \gamma(t) - \text{это кривая в } M \text{ и } \gamma(0) = x\}$. По сути, это множество всех направлений кривых на M в точке x .

Для примера найдем касательное пространство для шара $S^{n-1} = \{x \in \mathbb{R}^n | x^T x = 1\}$ в точке x_0 . Интересующее нас множество кривых выглядит так:

$$\begin{cases} x(t)^T x(t) = 1 \\ x(0) = x_0 \end{cases}$$

Продифференцируем первое уравнение по t и подставим $t = 0$:

$$x'(0)^T x(0) + x(0)^T x'(0) = 0 \Rightarrow x'(0)^T x_0 = 0$$

Обозначим $U_{x_0} = \{z | z^T x_0 = 0\}$. Мы только что доказали, что $T_{x_0} S^{n-1} \subseteq U_{x_0}$. Осталось показать обратное вложение. Пусть $x(t) = \frac{x_0 + tz}{\|x_0 + tz\|}$. Утверждение $x(0) = x_0$ тривиально. Нужно проверить, что $x'(0) = z$:

$$\begin{aligned} x^i(t) &= \frac{x_0^i + tz^i}{\|x_0 + tz\|} \\ x^i(t)' &= \frac{z^i \|x_0 + tz\| - (x_0^i + tz^i) \frac{2 \sum_{j=1}^n (x_0^j + tz^j) z^j}{2 \|x_0 + tz\|}}{\|x_0 + tz\|^2} = z^i \frac{1}{\|x_0 + tz\|} - (x_0^i + tz^i) \frac{z^T (x_0 + tz)}{\|x_0 + tz\|^3} \\ x^i(0)' &= z^i - x_0^i z^T x_0 = z^i \end{aligned}$$

Таким образом, $x'(0) = z$, а значит, $U_{x_0} \subseteq T_{x_0} S^{n-1}$, и поэтому $T_{x_0} S^{n-1} = U_{x_0}$. Мы получили, что касательное пространство к шару - это множество векторов, перпендикулярных радиусу, что логично. Тут нужно сделать еще два важных замечания:

1. $T_x M$ - линейное подпространство
2. $\dim T_x M = \dim M$

Теперь мы готовы ввести центральное понятие. **Римановым многообразием** называется многообразие M , снабженное скалярным произведением на каждом касательном пространстве $T_x M$, его принято обозначать $g_x(\cdot, \cdot) = \langle \cdot, \cdot \rangle_x$. Если $M \subset \mathbb{R}^n$, то в качестве скалярного произведения удобно брать стандартное евклидово.

Производной по направлению ξ в римановском случае называется

$$Df(x)[\xi] = \left. \frac{df(\gamma(t))}{dt} \right|_{t=0}, \text{ где } \gamma(t) - \text{кривая в } M : \gamma(0) = x, \gamma'(0) = \xi$$

Римановым градиентом называется вектор $\text{grad } f(x)$, для которого верно следующее:

$$\langle \text{grad } f(x), \xi \rangle_x = Df(x)[\xi], \forall \xi \in T_x M$$

Для градиента верно соотношение про максимальное направление роста функции на множестве. В частности, верно следующее (здесь нормы берутся по скалярному произведению из определения риманова многообразия):

$$\frac{\text{grad } f(x)}{\|\text{grad } f(x)\|} = \arg \max_{\xi \in T_x M: \|\xi\|=1} Df(x)[\xi]$$

При этом, если $M \subset \mathbb{R}^n$, то $\text{grad } f(x) = P_{T_x M}(\nabla f(x))$. Поскольку $T_x M$ - линейное подпространство в \mathbb{R}^n , то можно написать $\nabla f(x) = P_{T_x M}(\nabla f(x)) + P_{T_x M^\perp}(\nabla f(x))$. Тогда:

$$\langle P_{T_x M}(\nabla f(x)), \xi \rangle_x = \langle \nabla f(x) - P_{T_x M^\perp}(\nabla f(x)), \xi \rangle_x = \{\xi \in T_x M\} = \langle \nabla f(x), \xi \rangle_x = Df(x)[\xi]$$

В последнем равенстве мы получили производную по направлению в евклидовом смысле, но поскольку $\xi \in T_x M$, то она совпадает с римановской. Отсюда $\text{grad } f(x) = P_{T_x M}(\nabla f(x))$.

Обозначим $TM = \bigcup_{x \in M} (x, T_x M)$. **Ретракцией** называется отображение $R : TM \rightarrow M$ ($R_x : T_x M \rightarrow M$), удовлетворяющее двум свойствам:

1. $R_x(0) = x$
2. $\frac{dR_x(t\xi)}{dt} = \xi, \forall \xi \in T_x M$

Пример:

1. S^{n-1} : $R_x(\xi) = \frac{x+\xi}{\|x+\xi\|}$ (доказательство полностью аналогично примеру про касательное пространство).
2. $\text{St}(m, n)$: $R_X(\Xi) = (X + \Xi)(I_m + \Xi^T \Xi)^{-1/2}$

В общем случае поиск ретракции - это очень нетривиальная задача. В частности, ретракция определена не единственным образом. Периодически выходят статьи с предложениями разных операторов ретракции.

Теперь мы можем с чистой совестью определить RGD: $x_{k+1} = R_{x_k}(-\alpha_k \text{grad } f(x_k))$. Но мы пойдем дальше и выпишем формулы для RGD с моментумом. Получается система:

$$\begin{cases} d_{k+1} = \beta d_k + \alpha_k \text{grad } f(x_k) \\ x_{k+1} = R_{x_k}(-d_{k+1}) \end{cases}$$

Видите проблему? Нет? А она есть. Дело в том, что векторы d_k и $\text{grad } f(x_k)$ лежат в разных касательных пространствах ($d_k \in T_{x_{k-1}} M, \text{grad } f(x_k) \in T_{x_k} M$), потому складывать их некорректно. Чтобы избежать этой проблемы, вводится операция транспортировки векторов. **Транспортировкой** вектора ξ вдоль вектора η ($\xi, \eta \in T_x M$) называется отображение $\text{Transp} : TM \times TM \rightarrow TM$, обладающее следующими свойствами:

1. $\exists R_x : \text{Transp}_\eta(\xi) \in T_{R_x(\eta)} M$
2. $\text{Transp}_0(\xi) = \xi$
3. $\text{Transp}_\eta(a\xi + b\zeta) = a \text{Transp}_\eta(\xi) + b \text{Transp}_\eta(\zeta)$

Теперь более понятно о том, что только что произошло. Пусть у нас есть векторы ξ и η из касательного пространства точки x . Мы хотим сместиться из точки x в точку $R_x(\eta)$, но при этом "забрать с собой" вектор ξ , чтобы он из старого касательного пространства перешел в новое. Для этого и нужна операция транспортировки. Как правило, сначала фиксируют ретракцию, а транспортировку по ней строят так:

$$\text{Transp}_\eta(\xi) = \left. \frac{d}{dt} R_x(\eta + t\xi) \right|_{t=0}$$

Для шара S^{n-1} и ретракции $R_x(\xi) = \frac{x+\xi}{\|x+\xi\|}$ транспортировка выглядит так:

$$\text{Transp}_\eta(\xi) = \frac{1}{\|x + \eta\|} \left(I_n - \frac{(x + \eta)(x + \eta)^T}{\|x + \eta\|^2} \right) \xi$$

Теперь мы можем написать правильные формулы для RGD+momentum. Они выглядят так:

$$\begin{cases} d_{k+1} = \text{Transp}_{-d_k}(\beta d_k) + \alpha_k \text{grad } f(x_k) \\ x_{k+1} = R_{x_k}(-d_{k+1}) \end{cases}$$

Стоит отметить, что для римановской оптимизации существуют и алгоритмы линейного поиска, и методы второго порядка, но **к счастью**, это выходит за пределы нашего курса.