Непрерывная оптимизация. Экзамен

171 группа

Июнь 2020

Содержание

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.
Опреления из лекции:

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

$$\overline{\lim_{k\to\infty}} \frac{r_{k+1}}{r_k} = 1.$$

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

$$\overline{\lim_{k\to\infty}} \frac{r_{k+1}}{r_k} = c.$$

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

$$\overline{\lim_{k\to\infty}} \frac{r_{k+1}}{r_k} = 0.$$

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

$$r_{k+1} \le M r_k^2 \quad \forall k \ge k_0$$

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

$$r_k \le C q^k \quad \forall k \ge m.$$

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:
$x_{k+1} = x_k + \alpha_k d_k,\ x_k, d_k \in \mathbb{R}^n$
$\varphi(\alpha) = f(x_k + \alpha_k d_k)$, также есть примерное приближение: $\varphi(\alpha) \approx f(x_k) + \alpha \nabla f(x_k)^T d_k$
$\varphi(\alpha) \to \min\limits_{\alpha}$
$\nabla f(x_k)^T d_k = \varphi'(0) < 0$

Условие Армихо:

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

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(α_{start}, β, $\varphi_k : \mathbb{R}_+ \to \mathbb{R}$)
2: $\quad\alpha \leftarrow \alpha_{start}$
3: $\quad$**while** $\varphi_k(\alpha) > \varphi(0) + c_1\alpha\varphi_k'(0)$ **do**
4: $\quad\quad\alpha \leftarrow \beta\alpha$
5: $\quad$**return** α

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 $f : \mathbb{R} \to \mathbb{R}$ 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 $L : U \to V$ – линейный оператор. Будем говорить, что фукнция 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 $f : X \to V$ и $g : X \to V$ – 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 $\alpha : X \to \mathbb{R}$ и $f : X \to V$ – 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, $f : X \to Y$ – функция. Также пусть W – векторное пространство, $g : Y \to W$ – функция. Если f дифференцируема в точке x, и g дифференцируема в точке $f(x)$, то их композиция $(g \circ f) : X \to W$ (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 функци. $Df(x)[h]$ 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:

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

Определение. Пусть функция $f : X \to V$ 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 $f : \mathbb{R}^n \to \mathbb{R}$ также можно представить в виде матрицы:

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

Матрица H_x называется *гессианом* функции f в точке xи 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

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

$$f \in C_L^{k,m} \Leftrightarrow \left[\begin{array}{c} f \in C^k, \\ \|\nabla^m f(y) - \nabla^m f(x)\| \le L\|y - x\| \;\; \forall x, y \end{array} \right.$$

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

$$\|\nabla f(y) - \nabla f(x)\| \le L\|y - x\| \quad \Leftrightarrow \quad \begin{array}{l} f(y) \le f(x) + \nabla f(x)^T(y - x) + \dfrac{L}{2}\|y - x\|^2 \\ f(y) \ge f(x) + \nabla f(x)^T(y - x) - \dfrac{L}{2}\|y - x\|^2 \end{array}$$

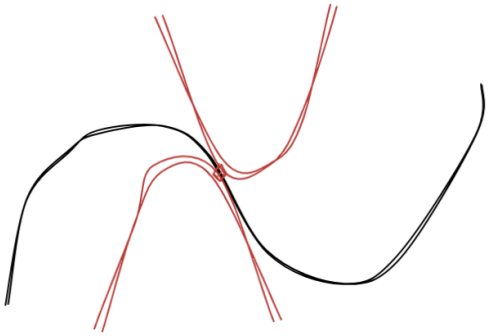

Рис. 1: Липшициевость $f \in C_L^{1,1}$ говорит о том, что в любой точке функцию сверху и снизу можно ограничить параболой, она не будет возрастать или убывать слишком быстро

А также можно сказать важные вещи про гессиан:

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

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

Примеры:

- $f(x) = \sin(x)$ – функция с липшициевым 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 липшициевой, так как любые ее производные всегда будут возрастать экспоненциально.

Выпуклость:

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

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

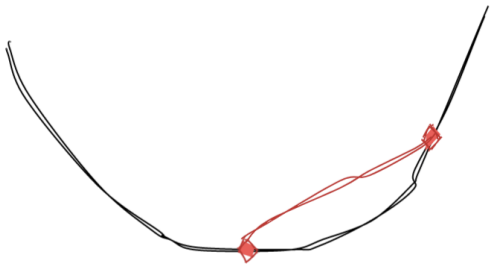

Рис. 2: Выпуклость означает, что если между любыми двумя точками провести отрезок, функция окажется снизу

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 липшициевость, 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. вторя 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 точке 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:

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

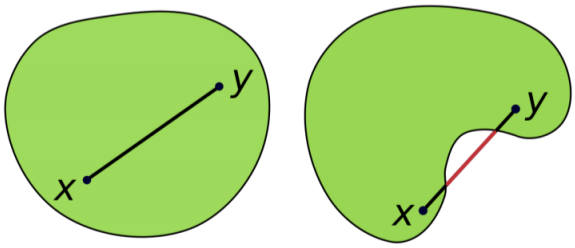

Рис. 3: Слева – выпуклое, справа – не выпуклое.

Свойства выпуклых функций

1. (Неравенство Йенсена) Пусть U – вещественное векторное пространство, Q – непустое выпуклое множество в U, и $f"Q \to \mathbb{R}$ – выпуклая функция. Пусть также $x_1, \ldots, x_k$ – точки в множестве Q и $\lambda_1, \ldots, \lambda_k$ – неотрицательные коэффициенты, сумммирующиеся в единицу: $\lambda_i \geq 0$ и $\sum_{i=1}^n \lambda_i = 1$. 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 афинной или когда все точки совпадают: $x_1 = \ldots = x_k$.

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 $f : Q \to \mathbb{R}$ называется множество

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

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

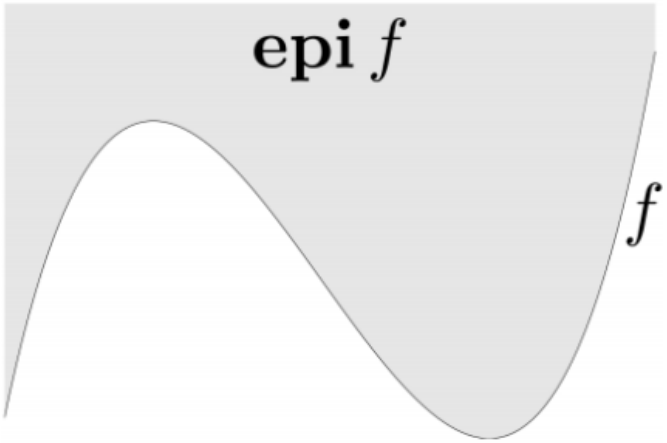

Рис. 4: Все серое – надграфик функции $f(x)$.

Определение. (Выпуклости через надграфик). Пусть U – вещественное векторное пространство, Q – непустое выпуклое множество в U. Функция $f : Q \to \mathbb{R}$ явдяется выпуклой тогда и только тогда, когда ее надграфик $\mathsf{Epi}(f)$ 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 $f : Q \to \mathbb{R}$ – выпуклая функция. Тогда для любого $\alpha \in \mathbb{R}$ соответствующее множество линий уровня

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

является выпуклым.

3. (Дифференциальные критерии выпуклости).
Утверждение.(Условие выпуклости первого порядка). Пусть $Dom(f)$ является открытым множеством, и функция f дифференцируема всюду на $Dom(f)$. Функция f является выпуклой тогда и только тогда, когда $Dom(f)$ является выпуклым множеством и

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

для всех x, $y \in Dom(f)$.

Утверждерние. (Дифференциальное условие оптимальности для выпуклой функции). Пусть $f : U \to \mathbb{R}\sup\{+\infty\}$ – выпуклая функция, $Dom(f)$ является открытым множеством, и пусть $x^* \in Dom(f)$. 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 $Dom(f)$ является открытым множеством, и функция f дважды дифференцируема на $Dom(f)$. Функция f является выпуклой тогда и только тогда, когда $Dom(f)$ является выпуклым множеством и

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

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

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

для всех $x \in Dom(f)$.

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.

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

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

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

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

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

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

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

$$\alpha F(y) + (1 - \alpha)F(y) \geq \alpha(F(x) + z_1^T(y - x)) + (1 - \alpha)(F(x) + z_2^T(y - x)) \to$$
$$F(y) \geq F(x) + (\alpha z_1 + (1 - \alpha)z_2)^T(y - x) \to \alpha z_1 + (1 - \alpha)z_2$$

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

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

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

1.9 Проксимальный оператор. Связь с евклидовой проекцией. @bigbluebutterfly

Пусть $f : E \to \mathbb{R}$ - функция, заданная на множестве E в евклидовом пространстве V. Определим проксимальный оператор функции f как отображение $\text{Prox}_f : V \to E$, 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 $g : E \to \mathbb{R}$, 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 проецирующее 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 унитреугольность 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^TU 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 ($Q^TQ = QQ^T = E$), 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 - $O(m^2n+mn^2+n^3)$, 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 $\approx O(n^3)$.

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

Задача оптимизации:

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

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

$$D = \operatorname{dom} f \bigcap (\cap_{i=1}^m \operatorname{dom} g_i) \bigcap \left(\cap_{j=1}^p \operatorname{dom} h_j\right)$$

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

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

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

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

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 афинными
- **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 $\widetilde{x}$ такое, что $h_j(\widetilde{x}) = 0 \, \forall j$ и $g_i(\widetilde{x}) < 0 \, \forall i$
- **ослабленные условия Слейтера** для выпуклых задач оптимизации найдется $\widetilde{x}$ такое, что $h_j(\widetilde{x}) = 0 \, \forall j$ и $g_i(\widetilde{x}) \leq 0$ для всех аффинных функций g_i и $g_i(\widetilde{x}) < 0$ 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 $\{g_i(x_*), h_j(x_*) \,|\, i, j \in \text{Active}(x_*)\}$.
Тогда найдутся λ_*, μ_* такие, что:

- $\nabla_x L(x_*, \lambda_*, \mu_*) = \nabla f(x_*) + \sum\limits_{i=1}^{m} \lambda_i \nabla g_{*,i}(x_*) + \sum\limits_{j=1}^{p} \mu_{*,j} \nabla h_j(x_*) = 0$ - станционарность 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 нежёсткости 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:

$$\mathcal{S}(x_0) = \left\{x(t) \,|\, x(t) \in F \, \forall t \geq 0; x(0) = x_0; x(t) \in C^1\right\}$$

$$\mathcal{T}_F(x_0) = \left\{d \,|\, \exists x(t) \in \mathcal{S}(x_0), \; d = \left.\frac{\partial x}{\partial t}\right|_{t=0}\right\}$$

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 $\{g_i(x_*), h_j(x_*) \,|\, i, j \in \text{Active}(x_*)\}$.
Тогда найдутся λ_*, μ_* такие, что:

- $\nabla_x L(x_*, \lambda_*, \mu_*) = \nabla f(x_*) + \sum\limits_{i=1}^{m} \lambda_i \nabla g_{*,i}(x_*) + \sum\limits_{j=1}^{p} \mu_{*,j} \nabla h_j(x_*) = 0$ - станционарность 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
- $d^T \nabla^2_{xx} L(x_*, \lambda_*, \mu_*) d \geq 0 \, \forall d \in \mathcal{T}_F(x_*)$

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

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

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

Задача линейного программирования в общем виде:

$$\begin{cases} c^T x \to \min\limits_{x} \\ Ax = b \\ Ex \le d \end{cases}$$

Задача линейного программирования в **канонической форме**:

$$\begin{cases} c^T x \to \min\limits_{x \in \mathbb{R}^n} \\ Ax = b, \quad A \in \mathbb{R}^{p \times n}, p < n, \mathrm{Rk}(A) = p \\ x \ge 0 \end{cases}$$

Эквивалентные преобразования для сведения задачи к канонической форме:

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

Пример сведения:

$$\begin{cases} c^T x \to \min\limits_{x} \\ Ax = b \\ Ex \le d \end{cases} \Leftrightarrow \begin{cases} c^T (x^+ - x^-) \to \min\limits_{x^+, x^-} \\ A(x^+ - x^-) = b \\ E(x^+ - x^-) + f = d \\ x^+, x^-, f \ge 0 \end{cases}$$

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

Задача условной оптимизации:

$$(P) \begin{cases} f(x) \to \min\limits_{x} \\ g_i(x) \le 0,\ i \in \{1, \dots, m\} \\ h_i(x) = 0,\ j \in \{1, \dots, p\} \end{cases} \qquad f, g_i, h_j \in C^1$$

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
- $q(\lambda, \mu) \le f_{opt}$, для $\lambda \ge 0, \mu$

Двойственная задача оптимизации:

$$(D) \begin{cases} q(\lambda, \mu) \to \max\limits_{\lambda, \mu} \\ \lambda \ge 0 \end{cases}$$

Двойственность:

- **Слабая** двойственность: $q_{opt} \le f_{opt}$
- **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:

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

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:

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

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. @sofloud

Рассмотрим задачу одномерной оптимизации вида

$$f(x) \to \min_x, x \in \mathbb{R}, \tag{1}$$

где функция $f : \mathbb{R} \to \mathbb{R}$ 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 $f(y)$.

$$\text{Если } f(y) \geq f(x), \text{ тогда } [a, b] \to [a, y], \text{ иначе } [a, b] \to [x, b].$$

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.

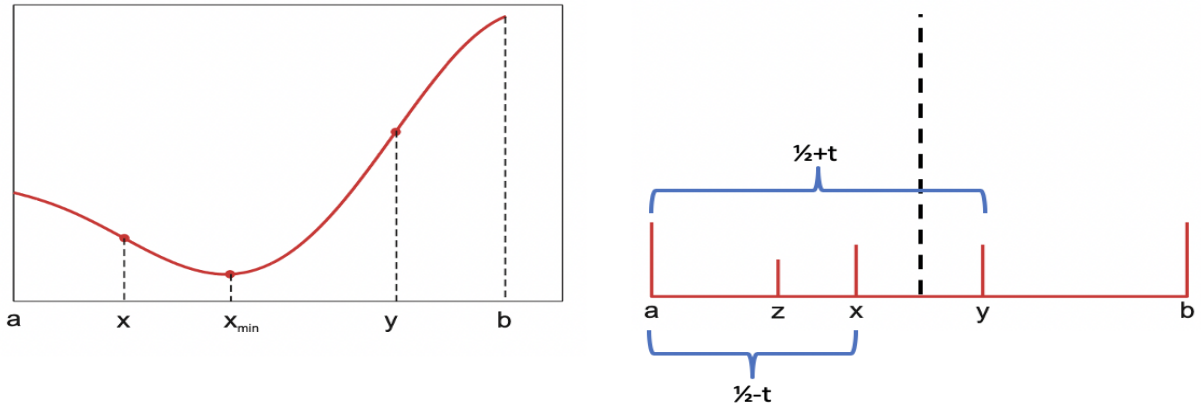

Рис. 5: Слева: унимодальная функция, справа: точки, выбираемые по золотому сечению

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 $k = 1, \ldots, \#iter$
  $I_{k+1} = KI_k$;
  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 $I_k = KI_{k-1} = K^2 I_{k-2} = \cdots = K^k I_0$, 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).

$$\text{Если } f(u) \leq f(x_2), \text{ то } [x_1,x_3] \to [x_1,x_2], \text{ иначе } [x_1,x_3] \to [u,x_3].$$

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.

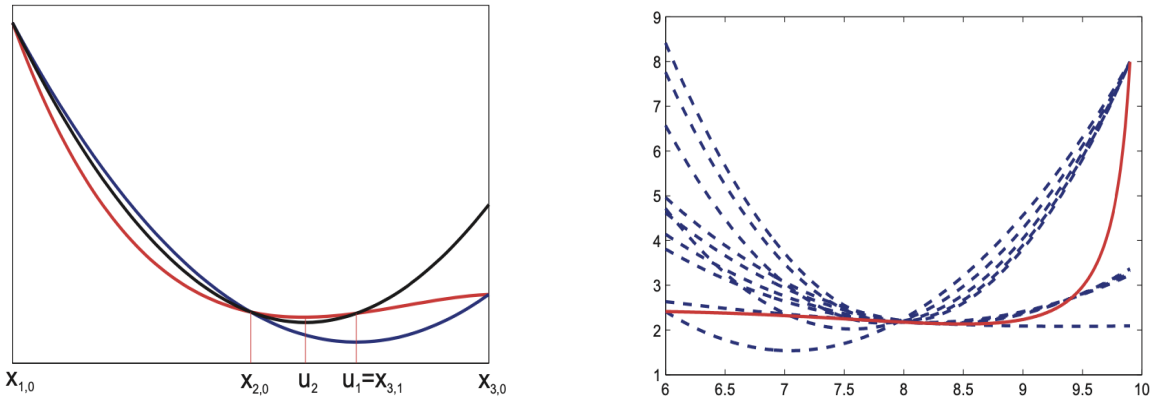

Рис. 7: Иллюстрация работы метода парабол. Слева: первые две итерации метода, красная кривая – оптимизируемая функция, синяя кривая – квадратичное приближение на первой итерации, чёрная кривая – квадратичное приближение на второй итерации. Справа: пример плохой сходимости метода парабол, красная кривая – 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 значениеw.

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 предпредыдущего 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 удалённость u от x позволяет избежать слишком больших шагов в оптимизации, которые могут соответствовать биениям в методе парабол. Использование в данном ограничении длины предпредыдущего 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:

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

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
$g = e, e = d$;
$\text{tol} = \varepsilon|x| + \frac{\varepsilon}{10}$;
если Выполнен критерий останова: $|x - \frac{a+c}{2}| + \frac{c-a}{2} \leq 2\text{tol}$ 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 $u - a < 2\text{tol}$ или $c - u < 2\text{tol}$ то
$u = x - \text{sign}(x - (a + c)/2)\text{tol}$;
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 $|u - x| < \text{tol}$ то
$u = x + \text{sign}(u - x)\text{tol}$; // 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.

   a. (слабое условие) $\phi'(\alpha_k) \geq c_2\ \phi'(0),\ c_2 \in (0{,}1)$
   b. (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 $c_2 \to 0$.

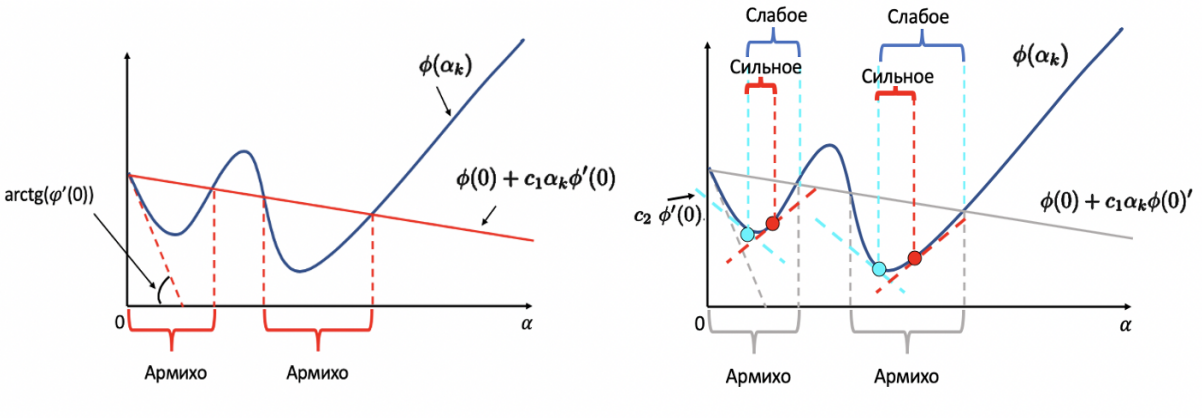

Рис. 9: Иллюстрация достаточных условий для неточной одномерной минимизации. Слева: условие Армихо, справа: слабое/сильное условие Вольфа.

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

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

$$\phi(\alpha) = \phi(0) + \phi'(0)\alpha + o(\alpha^2) \leq \phi(0) + c_1\alpha\phi'(0)$$
$$\phi'(0) + o(\alpha) \leq c_1\phi'(0)$$
$$\underbrace{\underbrace{(1-c_1)}_{>0}\underbrace{\phi'(0)}_{<0}}_{<0} + o(\alpha) \leq 0$$

$=> \exists\alpha_* : \forall\alpha \in (0, \alpha_*)$ условие Армихо выполнено.

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

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

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

$=>\ \alpha_*$ удовлетворяет слабому условию Вольфа.

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

$=>\ \alpha_*$ 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:
  $$\forall x, y \in Q \; ||f^{(p)}(x) - f^{(p)}(y)|| \leqslant L||x - y||$$
- $\mathscr{F}^1(Q)$ - класс функций, выпуклых на множестве Q.
- $\mathscr{F}_L^{p,k}(Q)$ - подкласс выпуклых на Q функций класса $C_L^{p,k}(Q)$.
- $\mathscr{T}_\mu^1(Q)$ - класс функций, сильно выпуклых на множестве Q с коэффициентом μ.
- $\mathscr{T}_{\mu,L}^{p,k}(Q)$ - 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 $\alpha \leqslant \frac{1}{L}$ 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: $\phi_k(\alpha_k) \leqslant \phi_k(0) + c_1 \alpha \phi_k'(0), c_1 \in (0, 1)$

   Условия Вульфа: $\phi_k(\alpha_k) \leqslant \phi_k(0) + c_1 \alpha \phi_k'(0), \phi_k'(\alpha_k) \geqslant c_2 \alpha \phi_k'(0), 0 < c_1 < c_2 < 1$

   Сильные условия Вульфа:

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

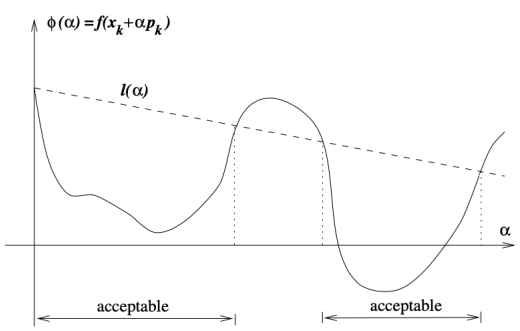

Рис. 11: Иллюстрация к условию Армихо: Накладывает условие достаточного уменьшения функции.

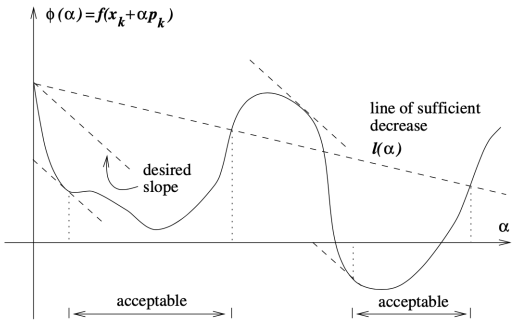

Рис. 12: Иллюстрация к условиям Вульфа: условие Армихо и условие кривизны функции: наклон не должен быть меньше начального, умноженного на константу.

Алгоритм 2 Метод дробления шага

Вход: Функция $\phi_k : \mathbb{R}_+ \to \mathbb{R}$. 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: $\quad \alpha \leftarrow \alpha/2$.
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 $f \in \mathscr{T}^{1,1}_{\mu,L}(\mathbb{R}^n)$. Тогда $\forall x, y \in \mathbb{R}^n$:

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

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

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

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

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

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

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

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

Доказательство:
Обозначим: $r_k = ||x_k - x^*||$.

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

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 $\mathscr{F}^{p,k}_{L}(Q)$ 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:
$Q_f = \frac{L}{\mu}$:

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

Это означает, что скорость сходимости линейна, но зависит от Q_f, а именно, при $Q_f >> 1$ (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 обучсловленности 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

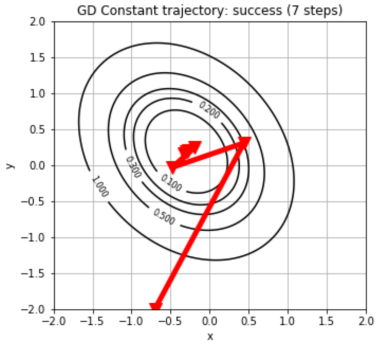

Рис. 15: Число обусловленности A: 2

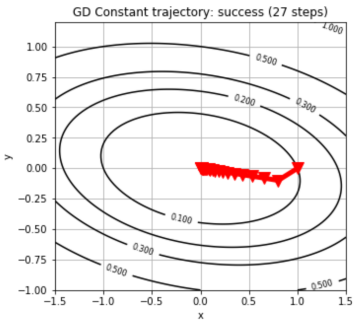

Рис. 16: Число обусловленности A: 6.4

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 $m_k(d_k)$:

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

Для минимизации находим градиент модели и приравниваем к нулю:

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

Соответсвенно, 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:

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

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:

$$\begin{cases} f(x_k + \alpha_k d_k) \le f(x_k) + c_1 \alpha_k \nabla f_k^T d_k \\ \nabla f(x_k + \alpha_k d_k)^T d_k \ge c_2 \nabla f_k^T d_k \end{cases} \qquad \text{где } 0 < c_1 < c2 < 1$$

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:

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

Доказательство:

$$\nabla f(x_k + \alpha_k d_k)^T d_k - \nabla f_k^T d_k \ge (c_2 - 1) \nabla f_k^T d_k$$
$$\left(\nabla f_{k+1} - \nabla f_k\right)^T d_k \ge (c_2 - 1) \nabla f_k^T d_k$$

Так как у f липшицевый градиент:

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

Тогда, совмещая два предыдущих неравенства:

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

Подставляем в условие Вульфа (первое) и получаем:

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

Теперь просуммируем по всем индексам до k:

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

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

Из предыдущей теоремы вытекает, что

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

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

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

Рассмотрим отдельно норму произведения обратной матрицы гессиана на градиент:

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

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

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

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_*:

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

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

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

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

$$x_{k+1} - x_* = x_k - x_* - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) = x_k - x_* - \left[\nabla^2 f(x_k)\right]^{-1} \int\limits_0^1 \nabla^2 f(x_* + t(x_k - x_*))(x_k - x_*)\, dt =$$

$$= (x_k - x_*) \left[\nabla^2 f(x_k)\right]^{-1} \int\limits_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))\right)\, dt$$

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

$$\left\| \int\limits_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))\right)\, dt \right\| \leq \int\limits_0^1 \left\|\nabla^2 f(x_k) - \nabla^2 f(x_* + t(x_k - x_*))\right\|\, dt \leq M\frac{r_k}{2}$$

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 $\left\|\left[\nabla^2 f(x_k)\right]^{-1}\right\| \leq (\mu - Mr_k)^{-1}$. Значит имеем:

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

Из этого следует, что для $x_0 : \quad \|x_* - x_0\| < \overline{r} = \frac{2\mu}{3M}$ 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:

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

  Тогда в качестве матрицы B_k можно взять

  $$B_k = Q\overline{\Lambda}Q^T, \text{ где } \overline{\lambda_i} = \begin{cases} \lambda_i, \ \lambda_i \geq \delta > 0, \\ \delta, \ \text{иначе.} \end{cases}$$

  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:

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

- **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, τ)
2:     get ∇²f(x_k)
3:     while τ < τ_max do
4:         B_k ← ∇²f(x_k) + τI_n
5:         if CholeskyDecomposition(B_k) then break
6:         τ ← min(τ_max, τν), ν > 1
7:     τ ← τ/β, β > 1
8:     return B_k, τ
```

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:

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

  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:

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

2.4.5 Примеры медленной и быстрой работы метода

Идеальный вариант для работы метода Ньютона - квадратичная функция: $f(x) = \frac{1}{2}x^TAx + b^Tx$, 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

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)

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

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 – CG. Но там скорость сходимости линейная. Попробуем сделать линейную стоимость по памяти и суперлинейную скорость сходимости. Начнем с обычного метода Ньютона:

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

$$H_k d_k = -\nabla f(x) \tag{1}$$

где H_k - гессиан на шаге k.

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

Важные моменты:

1) Вместо хранения H используем процедуру "$H \cdot d$"

2) Решаем СЛАУ неточно

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

Новый метод останова для CG:

$$r_k := H_k d_k + \nabla f_k$$
$$\|r_k\| \le \eta_k \|\nabla f(x_k)\|$$

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

— **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:

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

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

$$\|r_k\| \le \eta_k \|g_k\| \tag{3}$$

Покажем сходимость:

$$\|x_{k+1} - x_{\text{opt}}\| = \|x_k - x_{\text{opt}} + \underbrace{H_k^{-1}(r_k - g_k)}_{d_k \text{ по (2)}}\| \overset{\cdot H_k^{-1}}{\le} \underbrace{\|H_k^{-1}\|}_{\le \text{const}} \cdot \|H_k(x_k - x_{\text{opt}}) - g_k + r_k\| \le$$
$$\le \text{const} \cdot \Big(\underbrace{\|H_k(x_k - x_{\text{opt}}) - g_k\|}_{O(\|x_k - x_{\text{opt}}\|^2)} + \underbrace{\|r_k\|}_{\le \eta_k \|g_k\| \text{ по (3)}} \Big) \le \text{const} \cdot (O(\|x_k - x_{\text{opt}}\|^2) + \eta_k \|g_k\|)$$
$$\left\{ \|g_k\| = \|\nabla f(x_k) - \underbrace{\nabla f(x_{\text{opt}})}_{0}\| \overset{\text{липш. ?}}{=} O(\|x_k - x_{\text{opt}}\|) \right\}$$
$$\le \text{const} \cdot \big(O(\|x_k - x_{\text{opt}}\|^2) + \eta_k O(\|x_k - x_{\text{opt}}\|) \big) \quad \square$$

— **Локальная скорость сходимости HFN.**

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

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

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

3) $\eta_k = O(\|\nabla f(x_k)\|) \longrightarrow$ 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:

procedure HFN(f, ε, x_0)
  get $\nabla f(x_0)$
  for $k = 0, 1, 2, \ldots$ **do**
    $\varepsilon_k \leftarrow \min\left(\frac{1}{2}, \sqrt{\|\nabla f_k\|}\right) \cdot \|\nabla f_k\|$
    $z_0 = 0$, $g_0 = H_k z_0 + \nabla f_0$, $d_0 = -g_0$
    for $j = 0, 1, 2 \ldots$ **do**
      $\alpha_j \leftarrow \dfrac{g_j^T g_j}{d_j^T H_j d_j}$
      $z_{j+1} \leftarrow z_j + \alpha_j d_j$
      $g_{j+1} \leftarrow g_j + \alpha_j H_k d_j$
      if $\|g_{j+1}\| \le \varepsilon_k$ **then** $p_k \leftarrow z_{j+1}$; **break**
      $\beta_j \leftarrow \dfrac{g_{j+1}^T g_{j+1}}{g_j^T g_j}$
      $d_{j+1} \leftarrow -g_{j+1} + \beta_j d_j$
    $\alpha_k \leftarrow$ BACKTRACKING($\alpha_{start} = 1$)
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$
    if $\|\nabla f_{k+1}\| < \varepsilon$ **then break**
  return x_k

Важно: В данной процедуре мы **не** храним в памяти H, а только используем функцию перемножения Hd это и позволяет уйти от стоимость 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)

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

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

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

$$\begin{aligned} &1)\ u_1 = Xd \\ &2)\ u_2 = Bu_1 \\ &3)\ u = \frac{1}{N} X^T u_2 + \lambda d \end{aligned}$$

Итого получаем:
$X \in \mathbb{R}^{N \times D} \Rightarrow$
Сложность $\nabla^2 f \cdot d = O(ND^2)$
Сложность 1), 2), 3) $= O(ND)$
Это лучший способ решения логистической регрессии для хорошей точности.
Дальше 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 $\nabla^2 f(x)$

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

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

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

Ошибка 1) $= \dfrac{L_f\varepsilon_m + L_f\varepsilon_m}{\varepsilon} + \underbrace{L\cdot\varepsilon}_{O(\varepsilon)} \to \min\limits_{\varepsilon} \Rightarrow$ производную:

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

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

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

Улучшим:

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

ошибка 2) $= \dfrac{L_f\varepsilon_m + L_f\varepsilon_m}{2\varepsilon} + L\varepsilon^2 \Rightarrow 2L\varepsilon - \dfrac{2L_f\varepsilon_m}{2\varepsilon^2} = 0 \Rightarrow \varepsilon \approx \varepsilon_m^{\frac{1}{3}}$
Если подставить, то можем получить точность около $\varepsilon_m^{\frac{2}{3}}$

Еще улучшим!
Комплексное продолжение функции:

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

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 $\mathcal{O}(n^3)$. 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:

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

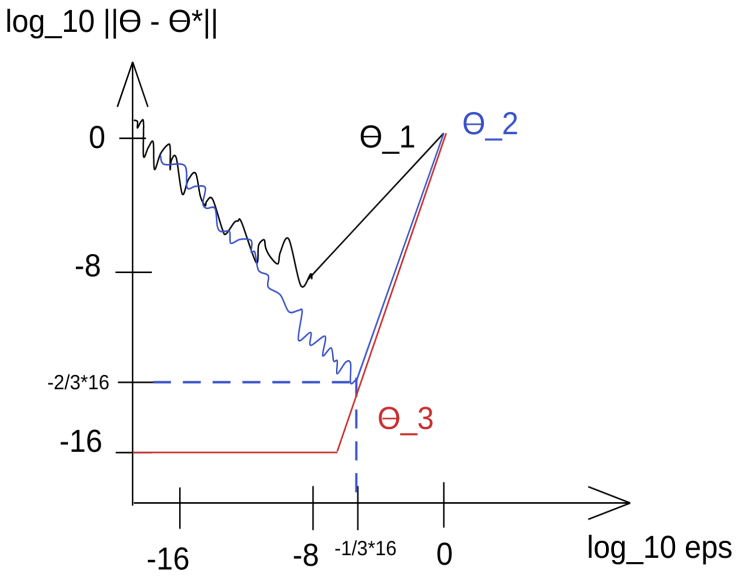

Рис. 17: Точность при разностном дифференцировании

Матрица B_k - симметричная (таков гессиан) и положительно-определенная (это необходимо для выпуклости квадратичной модели $\widehat{f}_k(d_k)$, 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 $\widehat{f}_k(d_k)$, мы получим систему линейных уравнений:

$$B_k d_k = -\nabla f_k \Leftrightarrow d_k = -B_k \nabla f_k$$

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:

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

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

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

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

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):

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

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

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

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

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

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

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

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 неусточивойсти. Однако, утверждается, что матрицы в SR-1 хорошо аппроксимируют настоящий гессиан. Со скоростью сходимости здесь сложно. Если методу удается 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):

$$\begin{cases} \|H - H_k\|_W \to \min\limits_{H \in \mathbb{S}^n_{++}} \\ H y_k = s_k \end{cases}$$

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 упростяться до сложности $\mathcal{O}(n^2)$ (это умножение матрицы на вектор, умножение вектора-столбца на вектор-строку и скалярное произведение):

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

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:

procedure BFGS(f, x_0, H_0, ε)
  for $k = 0, 1, 2, \ldots$ **do**
    $d_k \leftarrow -H_k \nabla f_k$
    $\alpha_k \leftarrow \text{LineSearch}(f, x_k, d_k)$
    $x_{k+1} \leftarrow x_k + \alpha_k d_k$
    $s_k \leftarrow x_{k+1} - x_k$
    $y_k \leftarrow \nabla f_{k+1} - \nabla f_k$
    $H_{k+1} \leftarrow \text{UpdateInversedHess}(H_k, s_k, y_k)$
    if $\|\nabla f_{k+1}\| < \varepsilon$ **then break**
  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:

procedure Multiply($v, \mathcal{H}, \gamma_0$)
  if $\mathcal{H} = \emptyset$ **then return** $\gamma_0 v$
  $(s, y) \leftarrow$ последняя пара из $\mathcal{H}$
  $\mathcal{H}' \leftarrow \mathcal{H}$ без последней пары
  $v' \leftarrow v - \frac{\langle s, v \rangle}{\langle y, s \rangle} y$
  $z \leftarrow \text{Multiply}(v', \mathcal{H}', \gamma_0)$
  return $z + \frac{\langle s, v \rangle - \langle y, z \rangle}{\langle y, s \rangle} s$
$\gamma_0^{(k)} = \frac{\langle y_{k-1}, s_{k-1} \rangle}{\langle y_{k-1}, y_{k-1} \rangle}$
$d_k = -\text{Multiply}(\nabla f_k, \mathcal{H}_k, \gamma_0^{(k)})$

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.

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

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

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

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

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 внутренности).

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 внутренности).

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** $\operatorname{grad} f(x)$ 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:

$$x_{k+1} = x_k - \alpha_k \operatorname{grad} f(x_k) \notin S^{n-1}$$

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 множества. По итогу, чтобы задать риманов градиентный спуск (RGD), нам нужно определить риманов градиент и операцию ретракцию.

$$x_{k+1} = R_{x_k}\big(-\alpha_k \operatorname{grad} f(x_k)\big)$$

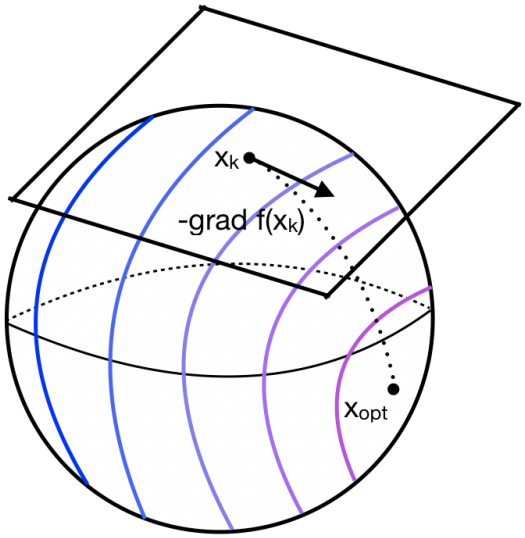

Рис. 18: Риманов антиградиент в точке x_k. Цветными кривыми показаны линии уровня функции $f(x)$.

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. $\forall x \in M, U$ - открестность x $\exists$ отображение $\phi : U \to \mathbb{R}^d$, биективно переводящее U в некоторое открытое множество. Пара (U, ϕ) называется картой множества M.
2. Для карт $(U, \phi), (W, \psi)$ в точке $x : U \cap W \neq \varnothing$ отображения $\phi \circ \psi^{-1}$ и $\psi \circ \phi^{-1}$ дифференцируемы бесконечное число раз как функции $\mathbb{R}^d \to \mathbb{R}^d$.

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 $\mathrm{St}(m, n) = \{X \in \mathbb{R}^{n \times m} \big| X^T X = I_m\}$ (в сущности, множество ортогональных матриц).

Касательным пространством в точке $x \in M$ называется множество $T_x M = \{\xi \in \mathbb{R}^n \big| \xi = \gamma'(0),$ где $\gamma(t) -$ это кривая в M и $\gamma(0) = x\}$. По сути, это множество всех направлений кривых на M в точке x.

Для примера найдем касательное пространство для шара $S^{n-1} = \{x \in \mathbb{R}^n \big| x^T x = 1\}$ 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 $x'(0) = z$:

$$x^i(t) = \frac{x_0^i + tz^i}{\|x_0 + tz\|}$$

$$x^i(t)' = \frac{z^i \|x_0 + tz\| - (x_0^i + tz^i) \frac{2 \sum_{j=1}^n (x_0^j + tz^j) z^j}{2 \|x_0 + tz\|}}{\|x_0 + tz\|^2} = z^i \frac{1}{\|x_0 + tz\|} - (x_0^i + tz^i) \frac{z^T (x_0 + tz)}{\|x_0 + tz\|^3}$$

$$x^i(0)' = z^i - x_0^i z^T x_0 = z^i$$

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 простанство к шару - это множество векторов, перпендикулярных радиусу, что логично. Тут нужно сделать еще два важных замечания:

1. T_xM - линейное подпространство
2. $\dim T_xM = \dim M$

Теперь мы готовы ввести центральное понятие. **Римановым многообразием** называется многообразие M, снабженное скалярным произведением на каждом касательном пространстве T_xM, его принято обозначать $g_x(\cdot,\cdot) = \langle\cdot,\cdot\rangle_x$. Если $M \subset \mathbb{R}^n$, то в качестве скалярного произведение 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 $\operatorname{grad} f(x)$, для которого верно следующее:

$$\big\langle \operatorname{grad} f(x), \xi \big\rangle_x = Df(x)[\xi], \forall \xi \in T_xM$$

Для градиента верно соображение про максимальное направление роста функции на множестве. В частности, верно следующее (здесь нормы берутся по скалярному произведению из определения риманова многообразия):

$$\frac{\operatorname{grad} f(x)}{\|\operatorname{grad} f(x)\|} = \underset{\xi \in T_xM : \|\xi\| = 1}{\arg\max} Df(x)[\xi]$$

При этом, если $M \subset \mathbb{R}^n$, то $\operatorname{grad} f(x) = P_{T_xM}(\nabla f(x))$. Поскольку T_xM - линейное подпространство в $\mathbb{R}^n$, то можно написать $\nabla f(x) = P_{T_xM}(\nabla f(x)) + P_{T_xM^\perp}(\nabla f(x))$. Тогда:

$$\big\langle P_{T_xM}(\nabla f(x)), \xi \big\rangle_x = \Big\langle \nabla f(x) - P_{T_xM^\perp}(\nabla f(x)), \xi \Big\rangle_x = \big\{\xi \in T_xM\big\} = \big\langle \nabla f(x), \xi \big\rangle_x = Df(x)[\xi]$$

В последнем равенстве мы получили производную по направлению в евклидовом смысле, но поскольку $\xi \in T_xM$, то она совпадает с римановской. Отсюда $\operatorname{grad} f(x) = P_{T_xM}(\nabla f(x))$.

Обозначим $TM = \bigcup_{x \in M}(x, T_xM)$. **Ретракцией** называется отображение $R : TM \to M$ ($R_x : T_xM \to M$), удовлетворяющее двум свойствам:

1. $R_x(0) = x$
2. $\dfrac{dR_x(t\xi)}{dt} = \xi, \forall \xi \in T_xM$

Пример:

1. S^{n-1}: $R_x(\xi) = \frac{x+\xi}{\|x+\xi\|}$ (доказательство полностью аналогично примеру про касательное пространство).
2. $\mathrm{St}(m,n)$: $R_X(\Xi) = (X + \Xi)(I_m + \Xi^T\Xi)^{-1/2}$

В общем случае поиск ретракции - это очень нетривиальная задача. В частоности, ретракция определена не единственным образом. Периодически выходят статьи с предложениями разных операторов ретракции.

Теперь мы можем с чистой совестью определить RGD: $x_{k+1} = R_{x_k}(-\alpha_k \operatorname{grad} f(x_k))$. Но мы пойдем дальше и выпишем формулы для RGD с моментумом. Получается система:

$$\begin{cases} d_{k+1} = \beta d_k + \alpha_k \operatorname{grad} f(x_k) \\ x_{k+1} = R_{x_k}(-d_{k+1}) \end{cases}$$

Видите проблему? Нет? А она есть. Дело в том, что векторы d_k и $\operatorname{grad} f(x_k)$ лежат в разных касательных пространствах ($d_k \in T_{x_{k-1}}M, \operatorname{grad} f(x_k) \in T_{x_k}M$), потому складывать их некорректно. Чтобы избежать этой проблемы, вводится операция транспортировки векторов. **Транспортировкой** вектора ξ вдоль вектора η ($\xi, \eta \in T_xM$) называется отображение $\mathrm{Transp} : TM \times TM \to TM$, обладающее следующими свойствами:

1. $\exists R_x : \mathrm{Transp}_\eta(\xi) \in T_{R_x(\eta)}M$
2. $\mathrm{Transp}_0(\xi) = \xi$
3. $\mathrm{Transp}_\eta(a\xi + b\zeta) = a\,\mathrm{Transp}_\eta(\xi) + b\,\mathrm{Transp}_\eta(\zeta)$

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:

$$\mathrm{Transp}_\eta(\xi) = \frac{d}{dt} R_x(\eta + t\xi)\Big|_{t=0}$$

Для шара S^{n-1} и ретракции $R_x(\xi) = \frac{x+\xi}{\|x+\xi\|}$ транспортировка выглядит так:

$$\mathrm{Transp}_\eta(\xi) = \frac{1}{\|x+\eta\|}\Big(I_n - \frac{(x+\eta)(x+\eta)^T}{\|x+\eta\|^2}\Big)\xi$$

Теперь мы можем написать правильные формулы для RGD+momentum. Они выглядят так:

$$\begin{cases} d_{k+1} = \mathrm{Transp}_{-d_k}\left(\beta d_k\right) + \alpha_k \operatorname{grad} f(x_k) \\ x_{k+1} = R_{x_k}(-d_{k+1}) \end{cases}$$

Стоит отметить, что для римановской оптимизации существуют и алгоритмы линейного поиска, и методы второго порядка, но **к счастью**, это выходит за пределы нашего курса.