Л1 Основные понятия

Л1.1 Задача оптимизации

Задачу оптимизации, которая будет рассматриваться в рамках данного пособия, можно сформулировать следующим образом:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \subseteq \mathbb{R}^d \\ & g_i(x) = 0, \quad i = \overline{1,l} \\ & h_j(x) \leq 0, \quad j = \overline{1,m}. \end{aligned} \tag{Л1.1}$$

Получается, что, варьируя вектор x, нам необходимо найти значение f^* — минимум функции f такой, что («s.t.» — «subject to») дополнительно выполнен ряд условий, ограничений. Взглянув на задачу (Л1.1), можно заметить, что с формальной точки зрения нам достаточно найти f^*, что и будет являться решением. Но наряду с f^* часто необходим и вектор x^*, на котором это значение достигается $f^* = f(x^*)$. Получается, что нам скорее интересна не постановка (Л1.1), а постановка вида argmin по поиску x^*. Уже более формально мы определим её в Параграфе Л2. Понятно, что в общем случае уникальность, да и вообще существование x^* и f^* никто не гарантирует (рассмотрите, например, задачу $\min_{x \in \mathbb{R}} x$). Отметим, что задача (Л1.1) имеет общий вид, и мы часто будем ее упрощать. Например, можно рассматривать задачу безусловной оптимизации, т.е. убрать все ограничения вида равенств и неравенств и положить $\mathcal{X} = \mathbb{R}^d$.
Более подробно остановимся на объектах из формулировки (Л1.1):

- $f : \mathcal{X} \to \mathbb{R}$ — некоторая функция, заданная на множестве $\mathcal{X}$. Данная функция является целевой для задачи оптимизации (Л1.1). Мы будем накладывать дополнительные свойства на f: липшицевость, гладкость, выпуклость. Связано это с тем, что уже в рамках этого параграфа станет понятно, что без предположений на f не получится построить оптимистичную теорию поиска решения (Л1.1).

- $\mathcal{X} \subseteq \mathbb{R}^d$ — подмножество d-мерного пространства. В общем случае $\mathcal{X}$ может быть любым множеством, но часто мы будем предполагать, что $\mathcal{X}$ является «простым» (суть «простоты» в данном случае неоднозначна и будет раскрыта далее, в момент изучения методов оптимизации на «простых» множествах). Например, в качестве $\mathcal{X}$ может выступать шар радиуса R с центром в точке a в p-норме

  $$\overline{B}_d^p(R, a) = \left\{ x \in \mathbb{R}^d \;\middle|\; \|x - a\|_p \leq R \right\},$$

  вероятностный симплекс

  $$\triangle_{d-1} = \left\{ x \in \mathbb{R}^d \;\middle|\; x_i \geq 0,\ i = \overline{1,d},\ \sum_{i=1}^{d} x_i = 1 \right\}$$

  или положительный ортант

  $$\perp_d = \left\{ x \in \mathbb{R}^d \;\middle|\; x_i \geq 0 \right\}.$$

- Также в задаче (Л1.1) присутствуют функциональные ограничения вида равенств и неравенств. $g_i : \mathcal{X} \to \mathbb{R}$, $i = \overline{1,l}$ и $h_j : \mathcal{X} \to \mathbb{R}$, $j = \overline{1,m}$ — функции, задающие ограничения. Мы также будем в дальнейшем предполагать, что данные функции являются достаточно «хорошими». Отметим, что формально все функциональные ограничения можно было просто спрятать в $\mathcal{X}$ (часто возможно и обратно — $\mathcal{X}$ записывается в виде набора функциональных ограничений), главной проблемой при таком действии может стать потеря «простоты» $\mathcal{X}$.

Замечание Л1.1. Часто используют краткую запись:

$$\min_{x\in\mathcal{X}} f(x) \quad \text{вместо} \quad \begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \subseteq \mathbb{R}^d \end{aligned}.$$

Л1.2 Примеры задач оптимизации

Л1.2.1 Оптимизация инвестиционного портфеля

Рассмотрим стандартную постановку задачи оптимизации портфеля, предложенную Г. Марковицем [15]. Цель задачи заключается в оптимальном распределении средств между доступными активами для минимизации совокупного инвестиционного риска при заданном уровне ожидаемой доходности.
Предположим на рынке имеется d акций, и мы владеем информацией об изменении цен на акции за n лет:

$$p_i^j \text{ — цена } i\text{-ого актива в } j \text{ год.}$$

Для начала мы можем рассчитать математическое ожидание μ_i доходности i-ой акции за n лет, применив формулу сложного процента:

$$\left(1+\mu_i\right)^n = \frac{p_i^n}{p_i^0} \implies \mu_i = \left(\frac{p_i^n}{p_i^0}\right)^{\frac{1}{n}} - 1.$$

Вектор ожидаемых доходностей всех активов обозначим как:

$$\mu = \left(\mu_1, \ldots, \mu_d\right)^\top.$$

Выпишем также выборочную ковариационную матрицу этих активов S, т.е. информацию о том, на сколько сильно варьируются цены на акции между собой:

$$S = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d1} & \ldots & \sigma_{dd} \end{bmatrix},$$

где σ_{ii} — выборочная дисперсия i-го актива, а σ_{ij} — выборочная ковариация i-го и j-го активов, которые считаются по формуле:

$$\sigma_{ij} = \frac{\sum\limits_{k=1}^{n} (\mu_i^k - \mu_i)(\mu_j^k - \mu_j)}{n},$$

где $\mu_i^k = \frac{p_i^k}{p_i^{k-1}} - 1$ — доходность i-го актива за k-ый год.
Наконец, мы хотим понять, какую долю портфеля стоит выделить на конкретную акцию $\omega = \left(\omega_1, \ldots, \omega_d\right)^\top$, где $\sum\limits_{i=1}^{d} \omega_i = 1$ и $\omega_i \geq 0$, $i = \overline{1,d}$, чтобы при фиксированной доходности $\left\langle \mu, \omega \right\rangle = r$ иметь наименьший риск. В рамках модели Марковица риск портфеля понимается как его *волатильность*, то есть степень разброса возможных значений доходности вокруг ожидаемого значения. Математически мерой такого разброса служит дисперсия. Тогда можно определить целевую функцию риска как

$$f(\omega) = \operatorname{Var}\left(\left\langle \omega, r \right\rangle\right) = \left\langle \omega, S\omega \right\rangle.$$

Итоговая задача оптимизации в таком случае принимает вид:

$$\begin{aligned} \min_{\omega \in \mathbb{R}^d} \quad & \langle \omega, S\omega \rangle \\ \text{s.t.} \quad & \langle \mu, \omega \rangle - r = 0 \\ & \sum_{i=1}^{d} \omega_i - 1 = 0 \\ & -\omega_i \leq 0, \quad i = \overline{1,d}. \end{aligned}$$

Л1.2.2 Оптимизация ценообразования

Предположим, что мы владеем небольшим бизнесом. В нашем магазине имеется d товаров. Для каждого товара i, мы знаем его себестоимость C_i, спрос Q_i^0 по начальной цене P_i^0 и функцию спроса $Q_i(P_i)$ от цены. В наших интересах — максимизировать оборот, не потеряв при этом текущую прибыль.
Определим целевую функцию как суммарный оборот нашего магазина:

$$f(P) = \sum_{i=1}^{d} P_i Q_i(P_i),$$

где $P = (P_1, \dots P_d)^\top$. Поскольку сильное изменение цен может привести к неожиданным последствиям, например, потере постоянных клиентов, дополнительно потребуем, чтобы изменения цен на товары, которые мы внесли, лежали в допустимых интервалах:

$$P_i \in [P_i^l, P_i^r], \quad i = \overline{1,d}.$$

Максимизируя оборот, мы не хотим потерять прибыль, поэтому нужно потребовать ограничения на неё:

$$\sum_{i=1}^{d} (P_i - C_i) Q_i(P_i) \geq \sum_{i=1}^{d} (P_i^0 - C_i) Q_i^0.$$

Наконец, сформулируем оптимизационную задачу:

$$\begin{aligned} \min_{P \in \mathbb{R}^d} \quad & -\sum_{i=1}^{d} P_i Q_i(P_i) \\ \text{s.t.} \quad & \sum_{i=1}^{d} (P_i - C_i) Q_i(P_i) \geq \sum_{i=1}^{d} (P_i^0 - C_i) Q_i^0 \\ & P_i \in [P_i^l, P_i^r], \quad i = \overline{1,d}. \end{aligned}$$

Л1.2.3 Оптимизация в статистике

Пусть есть параметрическое семейство вероятностных распределений $\{P(x|\theta)\}_{\theta \in \Theta}$, и дана независимая выборка $X = (x_1, \dots, x_n) \sim P(x|\theta)$ для некоторого $\theta \in \Theta$. Наша задача — оценить параметр θ. Например, $P(x|\theta)$ — нормальное распределение с математическим ожиданием $\theta \in \mathbb{R}$.
Существуют разные методы оценки параметров распределения по выборке, нас же интересует *метод максимальнoго правдоподобия* [21], суть которого заключается в поиске точки максимума функции совместного распределения наблюдаемых значений.

Предположим, что совместное распределение выборки задаётся функцией $p(X|\theta)$. Тогда функция правдоподобия:

$$L(\theta|X) = p(X|\theta).$$

Мы хотим максимизировать правдоподобие по θ, так как, чем оно больше, тем вероятнее, что наблюдаемые значения пришли из распределения $P(x|\theta)$.
В качестве целевой функции просто возьмем правдоподобие, а так как элементы выборки независимы, то его можно переписать:

$$L(\theta|X) = \prod_{i=1}^{n} p(x_i|\theta).$$

Зачастую удобно использовать логарифм правдоподобия:

$$\log L(\theta|X) = \sum_{i=1}^{n} \log p(x_i|\theta).$$

Рассмотрев минус логарифм правдоподобия, мы перейдём от задачи максимизации к задаче минимизации и получим следующую постановку:

$$\min_{\theta\in\Theta} \quad -\sum_{i=1}^{n} \log p(x_i|\theta).$$

Л1.2.4 Оптимизация обработки сигналов

Рассмотрим задачу идентификации неисправностей, произошедших в системе, на основе показаний датчиков. Пусть у нас есть d различных типов неисправностей, причём каждая из них происходит независимо с вероятностью p, также есть n ($n < d$) датчиков, которые получают данные системы с целью выявления неисправностей.
Пусть $x_i \in \{0, 1\}$, $i = \overline{1, d}$ — вектор индикаторов всех неисправностей, а показания датчиков задаются уравнением:

$$y = Ax + v,$$

где $A \in \mathbb{R}^{n\times d}$ — матрица неисправностей, в которой каждый столбец a_i равен показаниям датчиков при i-ой неисправности, а $v \sim \mathcal{N}(0, \sigma^2 I_n)$ — случайный вектор шума.
Цель состоит в том, чтобы выяснить, какой вид имеет вектор неисправностей x по вектору агрегированных показаний датчиков y.
Распределение датчиков имеет вид $y \sim \mathcal{N}(Ax, \sigma^2 I_n)$, где $x_i \sim Bern(p)$, $i = \overline{1, d}$. Воспользуемся предыдущим примером Л1.2.3. Найдём правдоподобие $L(x|y)$ с точностью до константы по формуле Байеса для апостериорной вероятности:

$$L(x|y) \propto p(y|x)p(x) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{\|Ax - y\|_2^2}{2\sigma^2}\right) p^{\sum_{i=1}^d x_i}(1-p)^{d-\sum_{i=1}^d x_i}.$$

Мы хотим максимизировать правдоподобие, поэтому зададим целевую функцию следующим образом:

$$\ell(x) = -\log L(x|y) + \text{const} = \frac{1}{2\sigma^2}\|Ax - y\|_2^2 + \log\left(\frac{1}{p} - 1\right)\sum_{i=1}^{d} x_i.$$

Таким образом, наша задача оптимизации примет следующий вид:

$$\begin{aligned} \min_{x\in\mathbb{R}^d} \quad & \frac{1}{2\sigma^2}\|Ax - y\|_2^2 + \log\left(\frac{1}{p} - 1\right)\sum_{i=1}^{d} x_i \\ \text{s.t.} \quad & x_i \in \{0, 1\}, \quad i = \overline{1, d}. \end{aligned}$$

Л1.2.5 Оптимизация в машинном обучении

Рассмотрим стандартную постановку задачи обучения с учителем. Пусть у нас есть обучающая выборка $\{x_i, y_i\}_{i=1}^n$, где $x_i \in \mathbb{R}^d$ — наши наблюдения, y_i — метки, их вид зависит от рассматриваемой задачи, например:

- $y_i \in \mathbb{R}$ для регрессии (предсказание цен на жильё, прогнозирование спроса, оценка стоимости бизнеса),
- $y_i \in \{0, 1\}$ для бинарной классификации (диагностика заболеваний, кредитный скоринг, фильтрация спама),
- $y_i \in \{c_1, \ldots, c_m\}$ для многоклассовой классификации (распознавание объектов на изображении, предсказание следующего слова, сентиментальный анализ текста).

Пусть у нас есть параметризованная модель $f(x|\theta)$, которая делает предсказания по наблюдениям x. Это может быть простая линейная функция: $f(x|\theta) = \langle \theta, x \rangle + b$ или нейросеть с несколькими миллиардами параметров. Задача ставится следующим образом — нужно найти оптимальные параметры модели θ, такие, чтобы предсказания были наиболее близки к истинным меткам.
Для этого вводится функция потерь $\ell\big(f(x_i|\theta), y_i\big)$, которая измеряет несоответствие предсказания модели и реального значения. Например, в случае регрессии это может быть среднеквадратичное отклонение:

$$MSE = \big(f(x_i|\theta) - y_i\big)^2,$$

а в случае бинарной или многоклассовой классификации — кросс-энтропия:

$$CE = -\sum_{c=1}^{m} y_{i,c} \log f_c(x_i|\theta),$$

где $y_{i,c}$ — индикатор принадлежности классу c, а $f_c(x_i|\theta)$ — предсказанная вероятность этого класса. Поскольку выборка конечная, то необходимо минимизировать *эмпирический риск*:

$$\frac{1}{n} \sum_{i=1}^{n} \ell\big(f(x_i|\theta), y_i\big).$$

Таким образом, задача нахождения оптимальных параметров модели формулируется в парадигме минимизации эмпирического риска:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i|\theta), y_i).$$

Л1.2.6 Оптимизация объёма эллипсоида

Предположим, есть какая-то выборка из $\mathbb{R}^d$, мы хотим использовать *MVE* (Minimum Volume Ellipsoid, см. Параграф 8.4 в [4]) — метод поиска эллипсоида минимального объема, содержащего заданный набор точек, для оценивания параметра многомерного распределения из которого пришли наблюдения. Зная параметры такого эллипсоида, можно найти сдвиг b и матрицу ковариаций Σ искомого распределения.

Положительно определенное линейное преобразование $A = \Sigma^{\frac{1}{2}}$ и сдвиг на вектор b, примененные к единичной сфере, задают эллипсоид с площадью, которая в $\det\left(A^{-1}\right)$ раз больше, чем площадь единичной сферы, поэтому мы параметризуем внутреннюю часть эллипсоида следующим образом:

$$S = \left\{ x \in \mathbb{R}^d \;\middle|\; \|Ax + b\|_2^2 \leq 1 \right\}.$$

Но есть проблема: функцию $\det\left(A^{-1}\right)$ сложно минимизировать напрямую. Однако, в силу $\det\left(A^{-1}\right) = \frac{1}{\det A}$, получаем, что $\log\det\left(A^{-1}\right) = -\log\det\left(A\right)$. А поскольку логарифм — монотонно возрастающая функция и сохраняет минимум, получим следующую задачу оптимизации:

$$\begin{aligned} \min_{\substack{A\in\mathbb{R}^{d\times d}\\ b\in\mathbb{R}^d}} \quad & -\log\det\left(A\right) \\ \text{s.t.} \quad & \left\|Ax_i + b\right\|_2 \leq 1, \quad i = \overline{1,n} \\ & A \succ 0. \end{aligned}$$

Л1.2.7 Оптимизационная задача в логистике

Рассмотрим следующую постановку: мы владельцы крупного бизнеса, у нас есть склады, каким-то образом соединенные дорогами. Мы хотим расположить новые склады так, чтобы транспортировочная стоимость была минимальной.
Задачу можно рассматривать как минимизацию функции на графе. Пусть $y_j \in \mathbb{R}^2$, $j = \overline{1,m}$ — наши склады, вершины графа, а $x_i \in \mathbb{R}^2$, $i = \overline{1,n}$ — склады, которые мы хотим поставить. Формализуем целевую функцию в виде:

$$f(x) = \sum_{i=1}^{n}\sum_{j=1}^{m} \omega_{ij} g\left(\left\|x_i - y_j\right\|\right),$$

где $w_{ij} \geq 0$ — веса, которые содержат информацию о том, насколько склады важны между собой, g — штрафная функция от расстояний. Также мы можем захотеть, чтобы некоторые склады находились в определенных областях. Это условие можно записать следующим образом:

$$x_i \in S_i, \quad i = \overline{1,n},$$

где S_i — прямоугольник, в котором должен находиться i-ый склад.
Тогда итоговая задача принимает вид:

$$\begin{aligned} \min_{x\in\mathbb{R}^d} \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} \omega_{ij} g\left(\left\|x_i - y_j\right\|\right) \\ \text{s.t.} \quad & x_i \in S_i, \quad i = \overline{1,n}. \end{aligned}$$

Л1.3 Общая схема методов оптимизации

Жизненный опыт подсказывает, что даже в одномерном случае задача оптимизации может не иметь аналитического решения (если вообще имеет). Если же решение существует, то на практике такую задачу обычно решают приближённо, итеративно находя лучшее решение, при этом допуская, что оно может не являться точным. Для этого применяются специальные алгоритмы, которые и называют *методами оптимизации*.
При создании методов оптимизации хочется добиться унифицированности. Потому что нет смысла искать лучший метод для решения конкретной задачи. Например, лучший алгоритм для решения задачи $\min_{x\in\mathbb{R}} x^2$ сходится за одну итерацию: этот метод просто всегда выдает ответ $x^* = 0$. Очевидно, что для других задач такой способ находить решение не пригоден. Метод должен подходить для целого класса однотипных или похожих задач.
Так как метод разрабатывается для целого класса задач, то он не может иметь с самого начала полной информации о задаче (Л1.1). Вместо этого используется модель задачи, например, формулировка задачи, описание функциональных компонент, множества, на которых происходит оптимизация и т.д. Предполагается, что численный метод может накапливать специфическую информацию о задаче при помощи некоторого

оракула. Под *оракулом* можно понимать некоторое устройство (программу, процедуру), которое отвечает на последовательные вопросы метода оптимизации о свойствах функции. В идеальном мире, наверное, можно было спросить у оракула всю информацию о решении задачи (Л1.1), но очевидно, что такого рода оракулы не интересны для практики. Мы будем работать с более приземленными вариантами оракулов, которые могут возвращать локальную информацию о целевой функции.

Определение Л1.1. Оракулы имеют разный порядок в зависимости от степени подробности, возвращаемой информации:

- *Оракул нулевого порядка* в запрашиваемой точке x возвращает значение целевой функции $f(x)$.
- *Оракул первого порядка* в запрашиваемой точке x возвращает значение целевой функции $f(x)$ и её градиент $\nabla f(x)$.
- *Оракул второго порядка* в запрашиваемой точке x возвращает значение целевой функции $f(x)$, её градиент $\nabla f(x)$ и её гессиан $\nabla^2 f(x)$.

Можно продолжать и до производных более высоких порядков.

Такое определение оракула имеет вполне прикладной смысл. В методах оптимизации обращение к оракулу часто реализуется посредством вызова отдельной от метода функции, программы или процедуры, возвращающей необходимую информацию, будь то значение функции, её градиента или её гессиана.
Методы оптимизации, которые мы будем рассматривать, описываются следующей схемой.

Алгоритм Л1.1 Общая итеративная схема метода оптимизации $\mathcal{M}$

Вход: начальная точка x^0, счётчик итераций $k = 0$, накапливаемая информационная модель решаемой задачи $I_{-1} = \varnothing$
while не выполнен критерий остановки $\mathcal{T}$ **do**
  Задать вопрос к оракулу $\mathcal{P}$ в точке x^k
  Пересчитать информационную модель: $I_k = I_{k-1} \cup (x^k, \mathcal{P}(x^k))$
  Применить правило метода $\mathcal{M}$ для получения новой точки x^{k+1} по модели I_k
  Положить $k = k + 1$
end while
Выход: $\hat{x}$

Алгоритм состоит из итераций, которые повторяются, пока не выполнен критерий остановки $\mathcal{T}$. В теле итерации мы обращаемся к оракулу, записываем его ответ в информационную модель, и получаем новую точку согласно правилу обновления.

Замечание Л1.2. В Алгоритме Л1.1 и далее мы будем использовать верхний индекс x^k для обозначения переменной x на k итерации.

Пример Л1.1. Рассмотрим задачу оптимизации

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

где функция $f(x)$ дифференцируема. Предположим, что в любой точке мы можем посчитать её градиент. Тогда можно воспользоваться следующим методом:

Алгоритм Л1.2 Градиентный спуск с постоянным размером шага

Вход: стартовая точка $x^0 \in \mathbb{R}^d$, размер шага $\gamma > 0$, количество итераций K
1: **for** $k = 0, 1, \ldots, K-1$ **do**
2: $\quad x^{k+1} = x^k - \gamma \nabla f\big(x^k\big)$
3: **end for**
Выход: x^K

С градиентным спуском мы будем подробнее знакомиться уже в следующих параграфах. Сейчас мы хотим лишь понять, как он ложится в общую модель. В данном случае оракул $\mathcal{P}$ — оракул первого порядка, который возвращает градиенты ∇f в запрашиваемых точках. При этом в информационную модель мы каждую итерацию кладём значения $\big(x^k, \mathcal{P}\big(x^k\big)\big) = \big(x^k, \nabla f\big(x^k\big)\big)$. Сам же метод градиентного спуска $\mathcal{M}$ использует только последнюю пару точек из информационной модели и получает новую точку по правилу: $x^{k+1} = x^k - \gamma \nabla f\big(x^k\big)$.

Замечание Л1.3. В общей итеративной схеме метода оптимизации (Алгоритм Л1.1) информационная модель растёт линейно с номером итерации k. Пример Л1.1 показывает, что для практических методов вовсе не обязательно хранить всю информационную модель — градиентному спуску нужна только текущая точка x^k и $\mathcal{P}\big(x^k\big) = \nabla f\big(x^k\big)$.

Замечание Л1.4. В Примере Л1.1 в качестве критерия остановки $\mathcal{T}$ используется ограничение на количество итераций, но может быть и требование на достижение точности решения.

Определение Л1.2. Под утверждением, что *задача решена с точностью ε (найдено ε-решение/выполнен критерий остановки)*, можно понимать:

- По аргументу: $\big\|x^k - x^*\big\|_2 \leq \varepsilon$.
- По значению функции: $f\big(x^k\big) - f^* \leq \varepsilon$.
- По норме градиента: $\big\|\nabla f\big(x^k\big)\big\|_2 \leq \varepsilon$.

Отметим, что первые два критерия являются теоретическими. Мы их будем использовать при доказательстве сходимости методов и оценке их скорости работы. На практике же сложно, например, оценить $\big\|x^k - x^*\big\|_2$, так как мы не знаем расположение точки x^*. При этом может оказаться полезным критерий сходимости вида: $\big\|x^{k+1} - x^k\big\|_2 \leq \varepsilon$. Из такого критерия не следуют какие-либо гарантии на $\big\|x^k - x^*\big\|_2$ — можно привести массу очевидно плохих методов, которые стоят на месте вдали от решения. Но верно обратное, а именно, если $\big\|x^{k+1} - x^*\big\|_2 \leq \big\|x^k - x^*\big\|_2 \leq \varepsilon/2$, то по неравенству треугольника:

$$\big\|x^{k+1} - x^k\big\|_2 \leq \big\|x^{k+1} - x^*\big\|_2 + \big\|x^k - x^*\big\|_2 \leq \varepsilon.$$

Поэтому при имеющейся теории или интуиции о том, что $\big\|x^k - x^*\big\|_2 \to 0$, критерий вида $\big\|x^{k+1} - x^k\big\|_2 \leq \varepsilon$ может быть более чем полезен.

Похожая ситуация с критерием на основе $f\big(x^k\big) - f^*$ с той лишь разницей, что для некоторых задач мы можем знать значение f^* (например, для $\min_{x \in \mathbb{R}^d} \big\|Ax - b\big\|_2^2$, где $b \in \operatorname{Im} A$, известно $f^* = 0$).

Следить за поведением $\big\|\nabla f\big(x^k\big)\big\|_2$ кажется более практичным. Это является правдой для определенного класса функций, пока мы рассматриваем безусловные варианты задачи (Л1.1) (без ограничений и с $\mathcal{X} = \mathbb{R}^d$). В общем же случае, такого рода критерий

может ничего не сказать — достаточно рассмотреть задачу $\min_{x\in[1,2]} x^2$. Существуют и другие критерии, с которыми познакомимся далее, которые подойдут и для $\mathcal{X} \neq \mathbb{R}^d$.

Л1.4 Сложность методов. Верхние и нижние оценки

Для понимания того, насколько эффективно/быстро/дёшево работают методы оптимизации, как их сравнивать между собой, необходимо ввести формальные понятия для описания *сложности* методов оптимизации:

Определение Л1.3.

- *Аналитическая/Оракульная сложность* — число обращений метода к оракулу, необходимое для решения задачи с точностью ε.
- *Арифметическая/Временная сложность* — общее число вычислений (включая работу оракула), необходимых для решения задачи с точностью ε.
- *Итерационная сложность* — общее число итераций метода, необходимых для решения задачи с точностью ε.

Арифметическая сложность даёт полное количество атомарных операций (сложений/умножений двух чисел), которое выполнил метод для решения задачи оптимизации. Время, которое затратит алгоритм, можно считать пропорциональным арифметической сложности.
Оракульная сложность учитывает только количество вызовов оракула, что пропорционально общему количеству атомарных операций, которые выполнил оракул в ходе работы метода. На самом деле, часто оракульной сложности бывает достаточно для оценки времени работы метода, так как вычисления оракулов являются наиболее дорогой операцией во всем методе. Рассмотрим, например, алгоритм из Примера Л1.1 для целевой функции f из Примера С2.3. Вычисления ∇f требуют умножения матрицы на вектор, а все остальные операции алгоритма производятся с вектором.
Итерационная сложность несёт меньше всего информации о времени работы метода, так как разные методы могут иметь абсолютно разную арифметическую сложность вызова оракула. Поэтому из того, что один метод достигает ε-решения за меньшее число итераций, вообще говоря, не следует, что он работает быстрее по времени.
Теперь попробуем понять, а какие гарантии на сложность мы вообще можем дать при поиске ε-решения (Л1.1). Для этого рассмотрим следующую, на первый взгляд, довольно простую задачу оптимизации:

$$\min_{x\in[0,1]^d} f(x), \tag{Л1.2}$$

где $[0,1]^d = \left\{ x \in \mathbb{R}^d \;\middle|\; 0 \leq x_i \leq 1,\ i = \overline{1,d} \right\}$ — кубик. При этом мы дополнительно предположим, что функция $f(x)$ является *M–липшицевой* на $[0,1]^d$ относительно ℓ_∞–нормы, т.е. для любых $x, y \in [0,1]^d$ справедливо

$$\left|f(x) - f(y)\right| \leq M\left\|x - y\right\|_\infty = M \max_{i=\overline{1,d}} |x_i - y_i|. \tag{Л1.3}$$

Замечание Л1.5. Множество $[0,1]^d$ является ограниченным и замкнутым, т.е. компактом, а из липшицевости функции f следует и её непрерывность. Поэтому задача (Л1.2) имеет решение, так как по теореме Вейерштрасса (см. Теорему 2 Параграфа 7 в [27]) непрерывная на компакте функция достигает своих минимального и максимального значений.

В поисках решения давайте ограничимся методами нулевого порядка (т.е. методы, которые могут вызывать оракул, возвращающий значения целевой функции). Более формально:

- **Цель:** найти $\hat{x} \in [0,1]^d$: $f(\hat{x}) - f^* \leq \varepsilon$, где f удовлетворяет (Л1.3).
- **Класс методов:** методы нулевого порядка.

Рассмотрим следующий довольно незатейливый алгоритм для решения поставленной цели:

Алгоритм Л1.3 Метод равномерного перебора

Вход: целочисленный параметр перебора $p \geq 1$

1: Сформировать $(p+1)^d$ точек вида $x_{(i_1,\ldots,i_d)} = \left(\frac{i_1}{p}, \frac{i_2}{p}, \ldots, \frac{i_d}{p}\right)^\top$, где вектор $(i_1,\ldots,i_d) \in \{\overline{0,p}\}^d$

2: Среди точек $x_{(i_1,\ldots,i_d)}$ найти точку $\hat{x}$ с наименьшим значением целевой функции f.

Выход: $\hat{x}$

Попробуем получить какие-нибудь гарантии на решение, которое находит данный алгоритм.

Теорема Л1.1 (Теорема 1.1.1. из [29])**.** Пусть задача оптимизации (Л1.2) с M-липшицевой целевой функцией f решается с помощью метода равномерного перебора с параметром p (Алгоритм Л1.3). Тогда справедлива следующая оценка сходимости:

$$f(\hat{x}) - f^* \leq \frac{M}{2p},$$

откуда следует, что методу равномерного перебора нужно в худшем случае

$$\left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor + 2\right)^d$$

обращений к оракулу, чтобы гарантировать $f(\hat{x}) - f^* \leq \varepsilon$.

Доказательство. Пусть x^* — решение задачи (возможно, не единственная точка минимума функции f). Тогда в построенной методом сетке из точек найдется такая точка $x_{(i_1,\ldots,i_d)}$, что

$$x := x_{(i_1,\ldots,i_d)} \preceq x^* \preceq x_{(i_1+1,\ldots,i_d+1)} =: x',$$

где знак $\preceq$ применяется покомпонентно. Точки x и x' определяют углы кубика сетки, в котором лежит x^*, то есть $x_i^* \in [x_i, x_i']$ для всех $i = \overline{1,d}$. Отметим, что стороны данного кубика имеют длины $x_i' - x_i = \frac{1}{p}$.

Кроме того, рассмотрим точки $\bar{x}$ и $\tilde{x}$ такие, что $\bar{x} = \frac{x+x'}{2}$ и

$$\tilde{x}_i = \begin{cases} x_i', & \text{если } x_i^* \geq \bar{x}_i, \\ x_i, & \text{иначе.} \end{cases}$$

$\bar{x}$ — центр кубика, а $\tilde{x}$ определяет ближайший к x^* уголок кубика. Заметим, что $\tilde{x}$ принадлежит сетке и $\left|\tilde{x}_i - x_i^*\right| \leq \frac{1}{2p}$, так как в худшем случае x^* расположен в $\bar{x}$, а значит, равноудален от всех уголков. Тогда

$$\left\|\tilde{x} - x^*\right\|_\infty = \max_{i \in \overline{1,d}} |\tilde{x}_i - x_i^*| \leq \frac{1}{2p}.$$

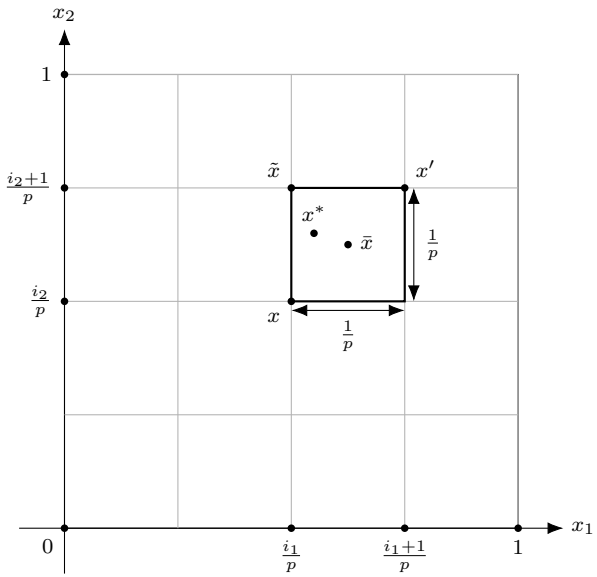

Рис. Л1.1: Построенная сетка и кубик в ней, которому принадлежит точка x^*.

Поскольку $f(\hat{x}) \leq f(\tilde{x})$ (так как $f(\hat{x})$ — минимум по всем точкам сетки), получаем

$$f(\hat{x}) - f^* \leq f(\tilde{x}) - f^* \leq M\|\tilde{x} - x^*\|_\infty \leq \frac{M}{2p}.$$

Выписанная выше оценка достигается методом равномерного перебора за $(p+1)^d$ обращений к оракулу. Ограничим сверху значением ε:

$$f(\hat{x}) - f^* \leq \frac{M}{2p} \leq \varepsilon \implies p \geq \frac{M}{2\varepsilon}.$$

Возьмем $p = \left\lfloor \frac{M}{2\varepsilon} \right\rfloor + 1$, тогда метод сделает $\left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor + 2 \right)^d$ обращений к оракулу нулевого порядка и выдаст ответ с ε-точностью. ■

Интересно оценить порядок величин, которые мы получили:

- Предположим, $M = 2$, $d = 13$ и $\varepsilon = 0.01$, то есть размерность задачи сравнительно небольшая (можно сказать, что никакая для задач оптимизации в современных приложениях) и точность решения задачи не слишком высокая.
- Необходимое число обращений к оракулу: $\left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor + 2 \right)^d = 102^{13} > 10^{26}$.
- Сложность одного вызова оракула не менее 1 арифметической операции.
- Производительность современной видеокарты порядка 10^{14} арифметических операций в секунду (82 TFLOPS на NVIDIA GeForce RTX 4090).
- Общее время: хотя бы 10^{12} секунд, что займёт порядка 30 тысяч лет.

Выводы получаются не очень оптимистичные. Но есть надежда, что наш теоретический анализ из Теоремы Л1.1 плох, либо метод не самый лучший, и существует какой-нибудь зубодробительный алгоритм, который будет давать куда более приятные гарантии на число оракульных вызовов.
В Теореме Л1.1 мы получили так называемые верхние оценки на гарантии нахождения решения, но существует и обратное понятие — нижние оценки.

Определение Л1.4. Пусть нам дан некоторый класс методов, а также некоторый класс задач оптимизации.

- *Верхняя оценка* — гарантия, что рассматриваемый метод из класса методов на любой задаче из класса решаемых задач имеет оракульную/арифметическую/итерационную сложность не хуже, чем утверждает верхняя оценка.
- *Нижняя оценка* — гарантия, что для любого метода из класса методов существует «плохая» задача из класса решаемых задач, такая, что метод имеет оракульную/арифметическую/итерационную сложность не лучше, чем утверждает нижняя оценка.

Напомним, что в этом параграфе мы рассматриваем класс методов нулевого порядка, т.е. все методы, которые каким-либо образом используют информацию о значениях функции (но не градиентов). Также класс задач, которые мы сейчас рассматриваем, описывается с помощью (Л1.2) и (Л1.3).
Нижние оценки нужны, чтобы понять, насколько полученные верхние оценки хороши. В частности, если верхние и нижние оценки совпали, это означает оптимальность предложенного метода для данного класса задач. Но если нижние оценки не совпали с верхними, то это может ничего не значить — возможно, в нижних оценках подобрана недостаточно «плохая» функция, а возможно, при получении верхних оценок пришлось загрубить теоретические выкладки. Нижние оценки можно получать не только для класса методов, но и для определенного метода, чтобы доказать оптимальность и неулучшаемость теоретического анализа и полученной верхней оценки.
Вернемся к нашей задаче: (Л1.2) + (Л1.3) и методам, оперирующим информацией нулевого порядка. Следующая теорема представит нижние оценки на оракульную сложность.

Теорема Л1.2 (Теорема 1.1.2. из [29])**.** Пусть задача оптимизации (Л1.2) с M-липшицевой целевой функцией f решается с помощью методов нулевого порядка с точностью $\varepsilon < \frac{M}{2}$. Тогда справедлива следующая оценка сходимости:

$$\left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor - 1\right)^d - 1$$

обращений к оракулу, чтобы гарантировать $f(\hat{x}) - f^* \leq \varepsilon$.

Доказательство. Доказываем от противного: предположим, что существует такой метод, который решает задачу за $N < \left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor - 1\right)^d - 1$ обращений к оракулу с точностью ε (по функции). Построим такую функцию, на которой метод не сможет найти ε-решение, при помощи сопротивляющегося оракула: пусть изначально наша целевая функция $f(x)$ всюду равна 0. Запустим метод, он запросит значение f в N точках, везде получит 0 и выдаст какую-то точку (возможно, отличную от всех предыдущих N), как ответ. В итоге мы в ходе работы алгоритма заглянули в $N + 1$ точку.

Заметим, что силу непрерывности функции $\frac{1}{x}$, можно подобрать число $\delta > 0$:

$$\frac{M}{2\varepsilon} - 1 < \frac{M}{(2+\delta)\varepsilon}.$$

Тогда, в силу монотонности округления:

$$\left\lfloor \frac{M}{2\varepsilon} \right\rfloor - 1 \le \left\lfloor \frac{M}{(2+\delta)\varepsilon} \right\rfloor.$$

Теперь из этого неравенства и предположения в начале получим оценку на $N+1$:

$$N+1 < \left(\left\lfloor \frac{M}{2\varepsilon} \right\rfloor - 1 \right)^d \le \left\lfloor \frac{M}{(2+\delta)\varepsilon} \right\rfloor^d.$$

Обозначим $p = \left\lfloor \frac{M}{(2+\delta)\varepsilon} \right\rfloor$. Рассмотрим равномерную сетку из $(p+1)^d$ точек, по аналогии с тем, как сделано в методе равномерного перебора. Из этих точек сформируем p^d кубиков с ребрами длины $\frac{1}{p}$. Воспользуемся тем, что смогли заглянуть в $N+1$ точку и $N+1 < p^d$. Тогда по принципу Дирихле (точки — кролики, кубики с углами в соседних вершинах сетки — клетки) найдётся такой кубик

$$C = \left\{ x \;\middle|\; \tilde{x} \preceq x \preceq \tilde{x} + \frac{1}{p}\mathbf{1} \right\},$$

который не содержит внутри себя ни одной из $N+1$ точки (в том числе и выхода метода). Здесь $\tilde{x}$ и $\tilde{x} + \frac{1}{p}\mathbf{1}$ — точки из сетки с шагом $\frac{1}{p}$, а $\mathbf{1}$ — вектор из единиц.

Пусть x^* — это центр кубика C, т.е. $x^* = \tilde{x} + \frac{1}{2p}\mathbf{1}$. Теперь немного модифицируем целевую функцию f:

$$\bar{f}(x) = \min\left\{0,\ M\|x - x^*\|_\infty - \left(1 + \frac{\delta}{2}\right)\varepsilon\right\}.$$

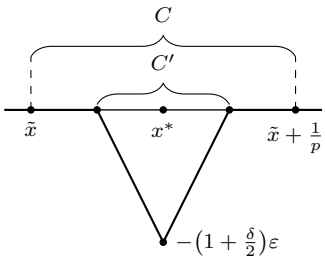

Рис. Л1.2: График функции $\bar{f}(x)$ в окрестности x^*.

Функция $\bar{f}(x)$ липшицева с константой M относительно ℓ_∞-нормы и принимает своё минимальное значение $-\varepsilon\left(1 + \frac{\delta}{2}\right)$ в точке x^*. Более того, по определению функция $\bar{f}(x)$ отлична от нуля только внутри куба

$$C' = \left\{ x \;\middle|\; \|x - x^*\|_\infty < \frac{\varepsilon}{M}\left(1 + \frac{\delta}{2}\right) \right\},$$

который лежит внутри куба C. Это так, поскольку согласно выбору $p = \left\lfloor \frac{M}{(2+\delta)\varepsilon} \right\rfloor$ верно

$$2p = 2\left\lfloor \frac{M}{(2+\delta)\varepsilon} \right\rfloor \leq \frac{2M}{(2+\delta)\varepsilon},$$

откуда получаем

$$\frac{\varepsilon}{M}\left(1 + \frac{\delta}{2}\right) \leq \frac{1}{2p}.$$

Вложение $C' \subset C$ позволяет нам сказать, что алгоритм заглянул только в точки с нулевым значением, поскольку вне C' функция тождественно равна нулю.
Итак, мы привели пример функции, минимум которой достигается в точке x^* и равен $-\varepsilon\left(1 + \frac{\delta}{2}\right)$, при этом рассматриваемый метод выдал точку $\hat{x}$ со значением $f(\hat{x}) = 0$. Следовательно, рассмотренный метод на данной функции нашел решение с точностью $f(\hat{x}) - f^* = \varepsilon\left(1 + \frac{\delta}{2}\right) > \varepsilon$. Противоречие. ■

Получается, что нижняя оценка из Теоремы Л1.2 не особо лучше Теоремы Л1.1. Зависимость $\left(\frac{M}{\varepsilon}\right)^d$ присутствует в обоих результатах. А это значит, что для класса липшицевых функций в общем случае все довольно печально. Нужны дополнительные предположения на задачу (Л1.1), чтобы гарантировать более оптимистичные гарантии. Но это вопрос уже следующих параграфов.

Л1.5 Скорость сходимости методов

Поймём, с каким характером верхних оценок на скорость сходимости/скорость приближения к решению мы хотим и будем иметь дело в дальнейшем.

Определение Л1.5. Выделим основные типы сходимостей:

- *Сублинейная*: $\left\|x^k - x^*\right\|_2 \leq \frac{C}{k^\alpha},\ C > 0,\ \alpha > 0.$
- *Линейная*: $\left\|x^k - x^*\right\|_2 \leq Cq^k,\ C > 0,\ 0 < q < 1.$
- *Сверхлинейная*: $\left\|x^k - x^*\right\|_2 \leq Cq^{k^p},\ C > 0,\ 0 < q < 1,\ p > 1.$
- *Квадратичная*: $\left\|x^k - x^*\right\|_2 \leq Cq^{2^k},\ C > 0,\ 0 < q < 1.$
  Квадратичная локальная: $\left\|x^{k+1} - x^*\right\|_2 \leq C\left\|x^k - x^*\right\|_2^2, C > 0.$

Из определения очевидно, что типы следуют в порядке увеличения скоростей сходимости. Приведённые в Определении Л1.5 типы сходимостей могут относиться как к арифметической, так и к оракульной или итерационной сложностям. Постоим графики представителей каждого из типов сходимостей на Рисунке Л1.3.
Можно лишь примерно оценить, как данные виды сходимостей визуально соотносятся между собой. Однако, спустя 5 итераций сверхлинейная и квадратичная сходимости становятся неразличимы, а после 10 итерации все, кроме сублинейной сходимости, сливаются с нулем. Естественным образом возникает вопрос: как визуально сравнивать скорости сходимостей, когда значения критерия близки к нулю? Решением этой проблемы является логарифмический масштаб оси критерия. Построим новый график с предложенным подходом (Рисунок Л1.4).
Теперь каждый график визуально отделяется от других. Также из рисунка становится понятно, почему линейная скорость получила такое название.
Отдельно рассмотрим последний тип сходимости и поймём, почему он так называется. Запишем неравенство из определения:

$$\left\|x^{k+1} - x^*\right\|_2 \leq C\left\|x^k - x^*\right\|_2^2.$$

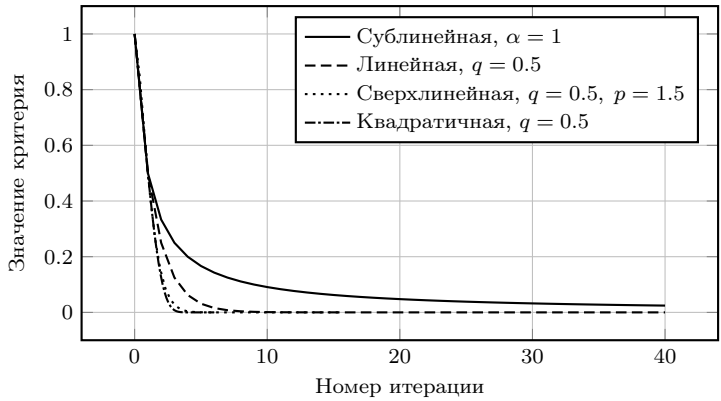

Рис. Л1.3: Основные типы сходимостей, линейный масштаб по оси критерия.

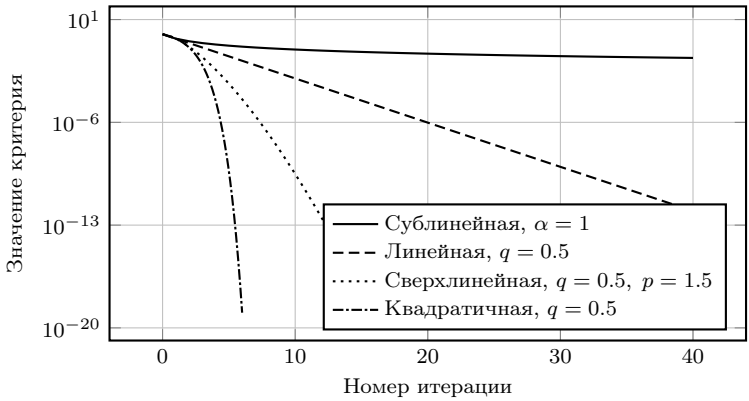

Рис. Л1.4: Основные типы сходимостей, логарифмический масштаб по оси критерия.

Запустим рекурсию: продолжим неравенства, записав оценку сначала для $\|x^k - x^*\|_2$ через $\|x^{k-1} - x^*\|_2$, потом для $\|x^{k-1} - x^*\|_2$ и так далее, пока не дойдем до $\|x^0 - x^*\|_2$.

$$\|x^{k+1} - x^*\|_2 \leq C\|x^k - x^*\|_2^2 \leq C \cdot C^2\|x^{k-1} - x^*\|_2^4 \leq \ldots \leq \left(\prod_{m=0}^{k} C^{2^m}\right)\|x^0 - x^*\|_2^{2^{k+1}}$$
$$= \left(C^{\sum_{m=0}^{k} 2^m}\right)\|x^0 - x^*\|_2^{2^{k+1}}.$$

Если учесть, что $\sum_{m=0}^{k} 2^m = 2^{k+1} - 1$, то можно записать

$$\|x^{k+1} - x^*\|_2 \leq \frac{1}{C}\left(C\|x^0 - x^*\|_2\right)^{2^{k+1}}.$$

Получили определение квадратичной скорости сходимости, где $C \to \frac{1}{C}$, $q \to C\|x^0 - x^*\|_2$. Вспомним требование на q:

$$0 < q = C\|x^0 - x^*\|_2 < 1,$$

что можно понимать как требование на достаточно хорошую стартовую точку:

$$\|x^0 - x^*\|_2 < \frac{1}{C}.$$

Поэтому эта сходимость называется локальной: она начинает работать, только если начальное приближение попало в окрестность решения.

Замечание Л1.6. Из Определения Л1.5, где даны скорости приближения к решению через k арифметических операций/оракульных вызовов/итераций, можно легко получить оценки на соответствующие сложности.
Пусть для некоторого метода мы доказали оценку на сходимость: $\|x^k - x^*\|_2 \leq \frac{C}{k^\alpha}$, где k — номер арифметических операции/оракульного вызова/итерации. Тогда, чтобы гарантировать точность ε по аргументу, нам необходимо сделать $k \geq \left(\frac{C}{\varepsilon}\right)^{1/\alpha}$:

$$\|x^k - x^*\|_2 \leq \frac{C}{k^\alpha} \leq \varepsilon \implies k^\alpha \geq \frac{C}{\varepsilon}.$$

Аналогичные манипуляции можно проделать со всеми типами сходимости из Определения Л1.5.