

Введение в оптимизацию

Звонарев Никита

Слушатели: 4-й курс, специальность СМ-СМ

Санкт-Петербургский Государственный Университет
Кафедра Статистического моделирования



Санкт-Петербург – 2023 г.

Общая постановка задачи

$\mathcal{D} \subset \mathbb{R}^k$ — замкнутое связное множество, $f(x)$ — целевая функция, $f : \mathcal{D} \rightarrow \mathbb{R}$.

Задача минимизации функции:

$$f^* = \min_{x \in \mathcal{D}} f(x).$$

Она же, в виде поиска точки минимума:

$$x^* = \arg \min_{x \in \mathcal{D}} f(x).$$

Вопросы: что конкретное имеется в виду под этой нотацией?

Что именно нужно искать?

Как нужно искать?

Общая постановка задачи

$\mathcal{D} \subset \mathbb{R}^k$ — замкнутое связное множество, $f(x)$ — целевая функция, $f : \mathcal{D} \rightarrow \mathbb{R}$.

Задача минимизации функции:

$$f^* = \min_{x \in \mathcal{D}} f(x).$$

Она же, в виде поиска точки минимума:

$$x^* = \arg \min_{x \in \mathcal{D}} f(x).$$

Вопросы: что конкретное имеется в виду под этой нотацией?

Что именно нужно искать?

Как нужно искать?

(Примерная) классификация методов оптимизации

Определение: метод оптимизации — алгоритм, который начиная с точки x_0 строит последовательность $x_0, x_1, \dots, x_n, \dots$ такую, что $x_n \rightarrow x^* = \arg \min_{x \in \mathcal{D}} f(x)$.

Что нужно искать?

❶ Локальная оптимизация

(Строгий) локальный минимум x^* — существует $\varepsilon > 0$:
 $f(x^*) < f(x)$ для любого x : $\|x - x^*\| < \varepsilon, x \neq x^*$.

❷ Глобальная оптимизация

(Строгий) глобальный минимум x^* : $f(x^*) < f(x)$ для любого $x \in \mathcal{D}, x \neq x^*$.

Как нужно искать?

❶ Детерминированная оптимизация

Обычно используется для поиска локальных минимумов.

❷ Стохастическая оптимизация

Обычно используется для поиска глобального минимума, если f содержит много локальных.

(Примерная) классификация методов оптимизации

Определение: метод оптимизации — алгоритм, который начиная с точки x_0 строит последовательность $x_0, x_1, \dots, x_n, \dots$ такую, что $x_n \rightarrow x^* = \arg \min_{x \in \mathcal{D}} f(x)$.

Что нужно искать?

❶ Локальная оптимизация

(Строгий) локальный минимум x^* — существует $\varepsilon > 0$:
 $f(x^*) < f(x)$ для любого x : $\|x - x^*\| < \varepsilon, x \neq x^*$.

❷ Глобальная оптимизация

(Строгий) глобальный минимум x^* : $f(x^*) < f(x)$ для любого $x \in \mathcal{D}, x \neq x^*$.

Как нужно искать?

❶ Детерминированная оптимизация

Обычно используется для поиска локальных минимумов.

❷ Стохастическая оптимизация

Обычно используется для поиска глобального минимума, если f содержит много локальных.

(Примерная) классификация методов оптимизации

По виду целевой функции:

- 1 Нет предположения о гладкости f
- 2 Есть предположение о гладкости f

По использованию производной:

- 1 Не используется производная
- 2 Используется производная (первая, вторая, ...)

Первые производные образуют *градиент* (gradient), вторые — *матрицу Гессе* (Hessian).

Как считать производную: аналитически, численно, использовать аппроксимацию ...

(Примерная) классификация методов оптимизации

По виду целевой функции:

- 1 Нет предположения о гладкости f
- 2 Есть предположение о гладкости f

По использованию производной:

- 1 Не используется производная
- 2 Используется производная (первая, вторая, ...)

Первые производные образуют *градиент* (gradient), вторые — *матрицу Гессе* (Hessian).

Как считать производную: аналитически, численно, использовать аппроксимацию ...

(Примерная) классификация методов оптимизации

По виду ограничений:

① Безусловная: $\mathcal{D} = \mathbb{R}^k$

② Условная

Линейные/нелинейные ограничения, (пример: “коробка”, ...)

f — линейная, $\mathcal{D}$ ограничен линейными функциями — линейное программирование.

f — квадратичная, $\mathcal{D}$ ограничен линейными функциями — квадратичное программирование.

Предположения о выпуклости области/целевой функции:

① f — не выпуклая или $\mathcal{D}$ — не выпуклое

② f — выпуклая (при этом $\mathcal{D}$ — выпуклое, например $\mathcal{D} = \mathbb{R}^k$)

Если f — строго выпукла, то у неё один строгий локальный минимум (он же строгий глобальный)

(Примерная) классификация методов оптимизации

По виду ограничений:

① Безусловная: $\mathcal{D} = \mathbb{R}^k$

② Условная

Линейные/нелинейные ограничения, (пример: “коробка”, ...)

f — линейная, $\mathcal{D}$ ограничен линейными функциями — линейное программирование.

f — квадратичная, $\mathcal{D}$ ограничен линейными функциями — квадратичное программирование.

Предположения о выпуклости области/целевой функции:

① f — не выпуклая или $\mathcal{D}$ — не выпуклое

② f — выпуклая (при этом $\mathcal{D}$ — выпуклое, например $\mathcal{D} = \mathbb{R}^k$)

Если f — строго выпукла, то у неё один строгий локальный минимум (он же строгий глобальный)

Одномерная оптимизация: линейный поиск

$$k = 1, \mathcal{D} = [a, b] \subset \mathbb{R}.$$

Поиск минимума на решётке с шагом ε (линейный поиск, последовательный поиск, linear search).

“Число узлов” $n = \lfloor \frac{b-a}{\varepsilon} \rfloor$.

“Узлы решётки” $y_j^{(n)} = a + j \frac{b-a}{n}, j = 0, \dots, n$.

$$x_n = \arg \min_{j=0, \dots, n} f(y_j^{(n)}).$$

Одномерная оптимизация: линейный поиск

$$k = 1, \mathcal{D} = [a, b] \subset \mathbb{R}.$$

Поиск минимума на решётке с шагом ε (линейный поиск, последовательный поиск, linear search).

“Число узлов” $n = \lfloor \frac{b-a}{\varepsilon} \rfloor$.

“Узлы решётки” $y_j^{(n)} = a + j \frac{b-a}{n}, j = 0, \dots, n$.

$$x_n = \arg \min_{j=0, \dots, n} f(y_j^{(n)}).$$

Если f — выпуклая, то $|x_n - x^*| \leq \varepsilon$.

Если f — непрерывная, то $|x_n - x^*| = O(\varepsilon)$.

Время работы: $O(1/\varepsilon)$.

Одномерная оптимизация: тернарный (троичный) поиск

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Идея: будем поддерживать отрезок, на котором ищем минимум. На каждой итерации разобьём его на три подотрезка. Выкинем тот, который точно не содержит минимум.

Алгоритм (ternary search)

$l_0 = a$, $r_0 = b$, $i = 0$.

Пока $r_i - l_i > \varepsilon$:

- 1 $\hat{l}_i = (2l_i + r_i)/3$, $\hat{r}_i = (l_i + 2r_i)/3$.
- 2 Если $f(\hat{l}_i) < f(\hat{r}_i)$, то $r_{i+1} = \hat{r}_i$, $l_{i+1} = l_i$, иначе $l_{i+1} = \hat{l}_i$, $r_{i+1} = r_i$.
- 3 Положить $i = i + 1$.

Результат: $(l_i + r_i)/2$

Одномерная оптимизация: тернарный (троичный) поиск

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Идея: будем поддерживать отрезок, на котором ищем минимум. На каждой итерации разобьём его на три подотрезка. Выкинем тот, который точно не содержит минимум.

Алгоритм (ternary search)

$l_0 = a$, $r_0 = b$, $i = 0$.

Пока $r_i - l_i > \varepsilon$:

- 1 $\hat{l}_i = (2l_i + r_i)/3$, $\hat{r}_i = (l_i + 2r_i)/3$.
- 2 Если $f(\hat{l}_i) < f(\hat{r}_i)$, то $r_{i+1} = \hat{r}_i$, $l_{i+1} = l_i$, иначе $l_{i+1} = \hat{l}_i$, $r_{i+1} = r_i$.
- 3 Положить $i = i + 1$.

Результат: $(l_i + r_i)/2$

Одномерная оптимизация: тернарный (троичный) поиск

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Идея: будем поддерживать отрезок, на котором ищем минимум. На каждой итерации разобьём его на три подотрезка. Выкинем тот, который точно не содержит минимум.

Алгоритм (ternary search)

$l_0 = a$, $r_0 = b$, $i = 0$.

Пока $r_i - l_i > \varepsilon$:

- 1 $\hat{l}_i = (2l_i + r_i)/3$, $\hat{r}_i = (l_i + 2r_i)/3$.
- 2 Если $f(\hat{l}_i) < f(\hat{r}_i)$, то $r_{i+1} = \hat{r}_i$, $l_{i+1} = l_i$, иначе $l_{i+1} = \hat{l}_i$, $r_{i+1} = r_i$.
- 3 Положить $i = i + 1$.

Результат: $(l_i + r_i)/2$

Выполнено $|x_n - x^*| \leq \varepsilon$.

Время работы: $O(\log(1/\varepsilon))$.

Одномерная оптимизация: метод золотого сечения

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Проблема тернарного поиска: целевая функция f вычисляется на каждой итерации дважды, в точках $\hat{l}_i$ и $\hat{r}_i$. Если f долго вычисляется, то это может быть неприемлемо.

Идея: как-то иначе разбить отрезок, чтобы f в одной из точек уже была подсчитана на прошлой итерации?

Отношение $\frac{r_i - l_i}{\hat{r}_i - l_i}$ обозначим как Φ . Для простоты: $l_i = 0$, $r_i = 1$, опустим i . Разобьём отрезок симметрично: $\frac{1-0}{\hat{r}-0} = \frac{1-0}{1-\hat{l}} = \Phi$.

Допустим, выкидываем на итерации правый отрезок. Найдём такое Φ , при которой $\hat{r}_{i+1} = \hat{l}_i$. Это означает следующее: $\frac{\hat{r}-0}{\hat{l}-0} = \Phi$.

Одномерная оптимизация: метод золотого сечения

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Проблема тернарного поиска: целевая функция f вычисляется на каждой итерации дважды, в точках $\hat{l}_i$ и $\hat{r}_i$. Если f долго вычисляется, то это может быть неприемлемо.

Идея: как-то иначе разбить отрезок, чтобы f в одной из точек уже была подсчитана на прошлой итерации?

Отношение $\frac{r_i - l_i}{\hat{r}_i - l_i}$ обозначим как Φ . Для простоты: $l_i = 0$, $r_i = 1$, опустим i . Разобьём отрезок симметрично: $\frac{1-0}{\hat{r}-0} = \frac{1-0}{1-\hat{l}} = \Phi$.

Допустим, выкидываем на итерации правый отрезок. Найдём такое Φ , при которой $\hat{r}_{i+1} = \hat{l}_i$. Это означает следующее: $\frac{\hat{r}-0}{\hat{l}-0} = \Phi$.

Одномерная оптимизация: метод золотого сечения

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая.

Проблема тернарного поиска: целевая функция f вычисляется на каждой итерации дважды, в точках $\hat{l}_i$ и $\hat{r}_i$. Если f долго вычисляется, то это может быть неприемлемо.

Идея: как-то иначе разбить отрезок, чтобы f в одной из точек уже была подсчитана на прошлой итерации?

Отношение $\frac{r_i - l_i}{\hat{r}_i - l_i}$ обозначим как Φ . Для простоты: $l_i = 0$, $r_i = 1$, опустим i . Разобьём отрезок симметрично: $\frac{1-0}{\hat{r}-0} = \frac{1-0}{1-\hat{l}} = \Phi$.

Допустим, выкидываем на итерации правый отрезок. Найдём такое Φ , при которой $\hat{r}_{i+1} = \hat{l}_i$. Это означает следующее: $\frac{\hat{r}-0}{\hat{l}-0} = \Phi$.

$\hat{l} = 1 - \frac{1}{\Phi}$, $\hat{r} = \frac{1}{\Phi}$, получаем: $\frac{1}{\Phi-1} = \Phi$, т.е.

$$\Phi^2 - \Phi - 1 = 0.$$

$$\Phi^2 - \Phi - 1 = 0.$$

Положительное решение: $\Phi = \frac{1+\sqrt{5}}{2}$ — золотое сечение.

Алгоритм совпадает с тернарным поиском, кроме:

- 1 $\hat{l}_i = r_i - \frac{r_i - l_i}{\Phi}$, $\hat{r}_i = l_i + \frac{r_i - l_i}{\Phi}$.
- 2 Взятие значения f в нужной точке с предыдущей итерации...

Время работы: $O(\log(1/\varepsilon))$.

Одномерная оптимизация: метод дихотомии

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, f — выпуклая, существует производная.

Идея: будем поддерживать отрезок, на котором ищем минимум.
Посчитаем знак производной в середине отрезка, и в зависимости от этого оставим нужную половину отрезка.

Алгоритм (binary search)

$l_0 = a$, $r_0 = b$, $i = 0$.

Пока $r_i - l_i > \varepsilon$:

- 1 $m_i = (l_i + r_i)/2$.
- 2 Если $f'(m_i) < 0$, то $l_{i+1} = m$, иначе $r_{i+1} = m$.
- 3 Положить $i = i + 1$.

Результат: $(l_i + r_i)/2$

Выполнено $|x_n - x^*| \leq \varepsilon$.

Время работы: $O(\log(1/\varepsilon))$.

Одномерная оптимизация: метод Ньютона

$k = 1$, $\mathcal{D} = [a, b] \subset \mathbb{R}$, у f существует вторая производная.

Идея: будем последовательно строить касательные параболы к f в точке x_n , в качестве x_{n+1} брать минимум у этой параболы.

Дано: x_0 .

Итерация:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

Одномерная оптимизация: теорема Канторовича

Метод Ньютона: $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$.

Теорема

Если существуют такие константы A, B, C , что:

① $\frac{1}{|f''(x)|} < A$ на $[a, b]$.

② $\left| \frac{f'(x)}{f''(x)} \right| < B$ на $[a, b]$.

③ Существует $f'''(x)$ на $[a, b]$, $|f'''(x)| \leq C \leq \frac{1}{2AB}$,

причём $b - a \leq \frac{1}{AB}(1 - \sqrt{1 - 2ABC})$. Тогда

① На $[a, b]$ существует локальный минимум x^*

② Если $x_0 = (a + b)/2$, то $x_n \rightarrow x^*$.

③ Выполнено: $|x_n - x^*| \leq \frac{B}{2^{n-1}}(2ABC)^{2^{n-1}}$.

Можно увидеть, что $|x^* - x_n| = \alpha |x^* - x_{n-1}|^2$ (квадратичная скорость сходимости).

Одномерная оптимизация: как применять

В общем случае f — не выпуклая на $[a, b]$. Идея:

- 1 Разбить отрезок на подотрезки длины ε .
- 2 На каждом из подотрезков запустить метод золотого сечения/дихотомии/Ньютона/...

ε при этом подбирать руками в зависимости от знаний о виде f .

Что использовать в реальной жизни, если есть возможность?

Метод Brent (функция `optimize` в R).

The method used is a combination of golden section search and successive parabolic interpolation, and was designed for use with continuous functions. Convergence is never much slower than that for a Fibonacci search. If f has a continuous second derivative which is positive at the minimum (which is not at lower or upper), then convergence is superlinear, and usually of the order of about 1.324.

Brent, R. (1973) Algorithms for Minimization without Derivatives. Englewood Cliffs N.J.: Prentice-Hall.