ВМК МГУ

Методы одномерной оптимизации

Курс: Методы оптимизации в машинном обучении

Методы оптимизации для оракула нулевого порядка

Рассмотрим задачу одномерной оптимизации вида

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

где функция $f : \mathbb{R} \to \mathbb{R}$ является непрерывной и унимодальной (см. рис. 1, слева). Обозначим через x_{min} решение задачи (1). Предположим, что нам известен интервал $[a, b]$, в котором содержится x_{min}. Предположим также, что в наличии имеется 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. рис. 1, слева). Тогда предположение об унимодальности функции позволяет сократить интервал поиска x_{min} путем сравнения значений $f(x)$ и $f(y)$. Если $f(x) > f(y)$, тогда интервал поиска сокращается до $[x, b]$. В противном случае интервал поиска сокращается до $[a, y]$. 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.

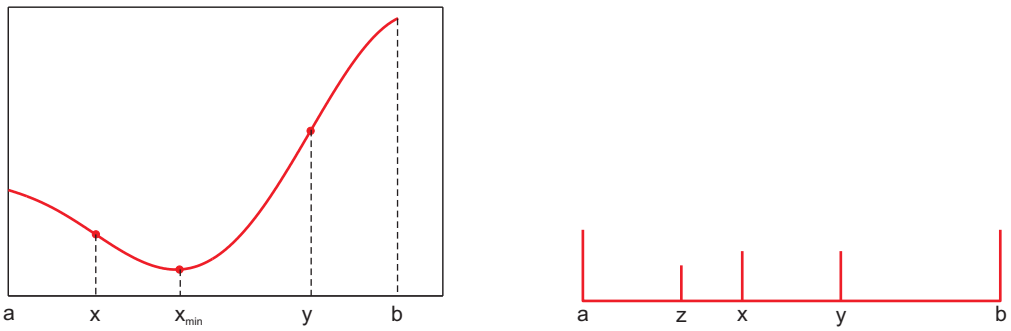

Рис. 1: Слева: унимодальная функция, справа: точки, выбираемые по золотому сечению

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. рис. 1, 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

$$\frac{1/2 + t}{1} = \frac{1/2 - t}{1/2 + t} \;\Rightarrow\; t^2 + 2t - 1/4 = 0 \;\Rightarrow\; t = \frac{\sqrt{5} - 2}{2} \;\Rightarrow\; K = 1/2 + t = \frac{\sqrt{5} - 1}{2} \simeq 0.618.$$

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;

Схема метода золотого сечения представлена в Алгоритме 1. В этой схеме границы 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.

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 равен

$$u = -\frac{b}{2a} = x_2 - \frac{(x_2 - x_1)^2(f_2 - f_3) - (x_2 - x_3)^2(f_2 - f_1)}{2[(x_2 - x_1)(f_2 - f_3) - (x_2 - x_3)(f_2 - f_1)]}.$$

Если $f_2 < f_1$ и $f_2 < f_3$, то точка u гарантированно попадает в интервал $[x_1, x_3]$. Таким образом, внутри интервала $[x_1, x_3]$ определены две точки x_2 и u, с помощью сравнения значений функции f в которых можно сократить интервал поиска (см. рис. 2, слева). 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. рис. 2, справа). Также следует отметить, что на первой итерации метод парабол требует измерения значений функции f в крайних точках интервала оптимизации.

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

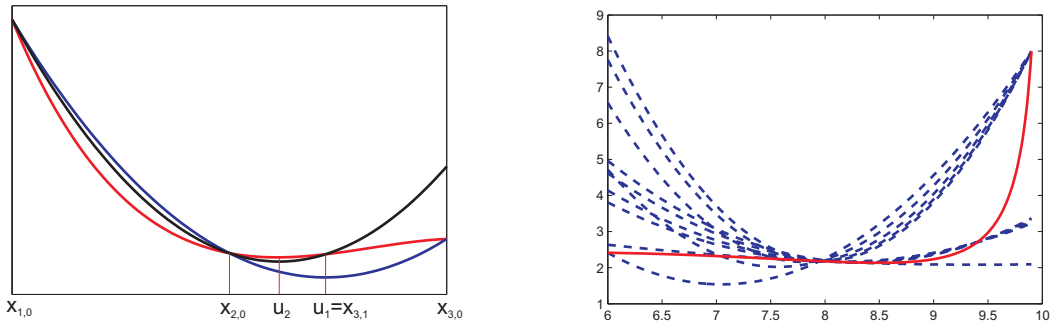

Рис. 2: Иллюстрация работы метода парабол. Слева: первые две итерации метода, красная кривая – оптимизируемая функция, синяя кривая – квадратичное приближение на первой итерации, чёрная кривая – квадратичное приближение на второй итерации. Справа: пример плохой сходимости метода парабол, красная кривая – оптимизируемая функция, синие кривые – итерационные квадратичные приближения.

Брента, который эффективно комбинирует эти две стратегии. В данном методе на каждой итерации отслеживаются значения в шести точках (не обязательно различных): a, c, x, w, v, u. Точки a, c задают текущий интервал поиска решения, x – точка, соответствующая наименьшему значению функции, w – точка, соответствующая второму снизу значению функции, v – предыдущее значение w. В отличие от метода парабол, в методе Брента аппроксимирующая парабола строится с помощью трех наилучших точек x, w, v (в случае, если эти три точки различны и значения в них также различны). При этом минимум аппроксимирующей параболы u принимается в качестве следующей точки оптимизационного процесса, если:

- u попадает внутрь интервала $[a, c]$;
- u отстоит от точки x не более, чем на половину от длины 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[1]. Как правило, именно этот алгоритм используется по умолчанию для одномерной оптимизации в стандартных пакетах оптимизации, например, в Python (функция scipy.optimize.minimize_scalar) или Matlab (функция fminbnd).

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.

Использование производной в оптимизации

Предположим теперь, что $O(x) = f'(x)$, т.е. в данной точке x возможно вычисление значения производной $f'(x)$, но невозможно вычисление $f(x)$, а также других производных. Тогда для решения задачи минимизации функции f можно воспользоваться методами решения уравнения $f'(x) = 0$. Заметим, что в общем случае ноль производной может соответствовать как локальному максимуму, так и минимуму функции. Поэтому после решения уравнения $f'(x) = 0$, вообще говоря, требуется дополнительная проверка решения. Этой проверки можно избежать, если оптимизируемая функция f является выпуклой или строго унимодальной на заданном интервале $[a, b]$.

Методы решения уравнения $f'(x) = 0$

Предположим, что на некотором отрезке $[a, b]$ $f'(a)$ и $f'(b)$ имеют разные знаки (см. рис. 3). В силу непрерывности производной отсюда следует, что внутри интервала $[a, b]$ найдется точка x_*, для которой $f'(x_*) = 0$. Пусть далее проведено измерение значения производной для некоторой промежуточной точки $x \in (a, b)$. Тогда можно сократить интервал поиска x_* в зависимости от знака $f'(x)$. Если $f'(x) > 0$, то интервал сокращается до $[a, x]$, в противном случае – до $[x, b]$. Требуя равенства длин потенциальных интервалов на текущей итерации, получаем, что промежуточную точку x следует выбирать в середине текущего интервала поиска. Данный метод получил

[1] полный код метода см. в [1], стр. 404–405

Алгоритм 2: Комбинированный метод Брента

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
        $v = u, f_v = f_u$;

название деления отрезка пополам и проиллюстрирован на рис. 3. В этом методе интервал поиска на каждой итерации сокращается ровно в два раза. Поэтому метод имеет линейную скорость сходимости с константой 1/2.

Заметим, что в отличие от случая оракула нулевого порядка, знание производной позволяет сокращать интервал поиска с помощью всего одной внутренней точки. В результате метод деления отрезка пополам работает быстрее, чем метод золотого сечения (сокращение интервала в два раза против 0.618 в случае золотого сечения).

По аналогии с методом парабол, для решения уравнения $f'(x) = 0$ можно воспользоваться различными аппроксимациями. Простейшая аппроксимация в данном случае – это приближение функции прямой (секущей) по крайним точкам текущего интервала поиска x_{min}. Пересечение секущей с нулём дает новую точку внутри интервала, по знаку производной в которой данный интервал может быть сокращен. Работа метода секущей проиллюстрирована на рис. 4, слева. Заметим, что построение секущей для $f'(x)$ по двум точкам равносильно аппроксимации параболой для $f(x)$[2]. Поэтому метод секущей наследует все достоинства и недостатки метода парабол, в частности, он обладает суперлинейной скоростью сходимости в близкой окрестности x_{min}, но может долго сходиться на начальных этапах итерационного процесса. Пример долгой сходимости метода секущей показан на рис. 4, справа.

На практике, методы деления отрезка пополам и секущей следует комбинировать, например, как это сделано

[2]С точностью до параметра сдвига по вертикали, который не влияет на положение минимума параболы

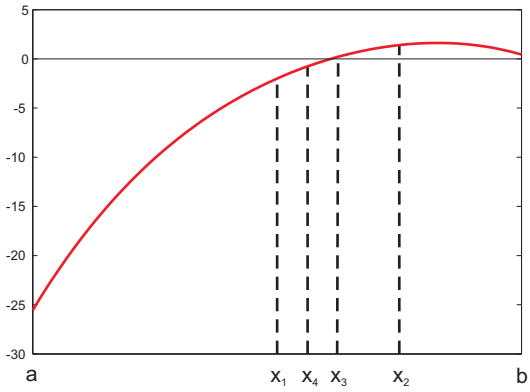

Рис. 3: Иллюстрация работы метода деления отрезка пополам. Красная кривая – функция в уравнении. Показаны первые четыре итерации метода.

в методе Брента[3].

Комбинированный метод Брента с производной

Рассмотрим решение задачи одномерной минимизации в случае наличия оракула первого порядка $O(x) = \{f(x), f'(x)\}$. В этом случае возможно построение различных аппроксимаций оптимизируемой функции на текущем интервале поиска решения с использованием разных данных:

- построение параболы по значениям функции в двух точках и значению производной в одной из них;
- построение параболы по значениям производной в двух точках (эквивалентно методу секущей);
- построение кубической аппроксимации по значениям функции в трех точках и значению производной в одной из них или по значениям функции и производной в двух точках (такая аппроксимация обладает кубической скоростью сходимости в окрестности оптимума);
- и другие.

Модификация метода Брента на случай дополнительного знания производной основана на следующих предложениях:

- для аппроксимаций используется парабола, вычисляемая по значениям производной в двух точках (метод секущей);
- на этапе аппроксимации строится две параболы: через две лучшие точки x и w, а также через лучшую x и третью по качеству точку v;

[3] Детали алгоритм см. http://en.wikipedia.org/wiki/Brent's_method

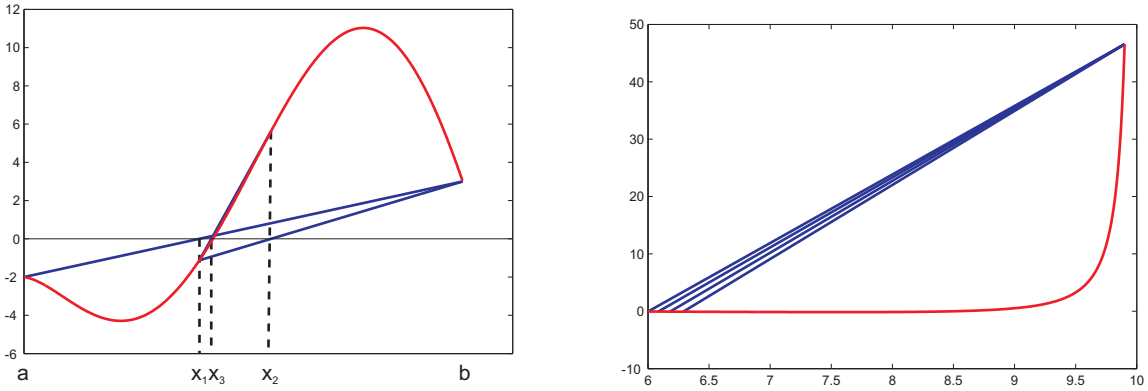

Рис. 4: Иллюстрация работы метода секущей. Слева: первые три итерации метода, красная кривая – функция в уравнении, синие прямые – секущие. Справа: пример долгой сходимости метода секущей, показана каждая 20-ая итерация.

Алгоритм 3: Комбинированный метод Брента с производной

Вход: Интервал оптимизации (a, c), точность ε;
Выход: Точка и значение минимума x_{min}, f_{min};
Инициализация $x = w = v = (a + c)/2$, $f_x = f_w = f_v = f(x)$, $f'_x = f'_w = f'_v = f'(x)$;
Инициализация длины текущего и предыдущего шага $d = e = c - a$;
пока Итерации до сходимости
$g = e, e = d$;
если Выполнен критерий останова: $|x - \frac{a+c}{2}| + \frac{c-a}{2} \le 2(\varepsilon|x| + \frac{\varepsilon}{10})$ то
Выход из цикла;
если Точки x, w и значения f'_x, f'_w – разные то
Параболическая аппроксимация, находим u_1 – минимум параболы;
если $u_1 \in [a, c]$, $(u_1 - x)f'_x \le 0$ и $|u_1 - x| < g/2$ то
Принимаем u_1 как кандидата;
если Точки x, v и значения f'_x, f'_v – разные то
Параболическая аппроксимация, находим u_2 – минимум параболы;
если $u_2 \in [a, c]$, $(u_2 - x)f'_x \le 0$ и $|u_2 - x| < g/2$ то
Принимаем u_2 как кандидата;
если u_1 или u_2 являются кандидатами то
$u = u_{1,2}$, если оба являются кандидатами, то берем u_i, соответствующее меньшей длине шага;
иначе
если $f'_x > 0$ то
$u = (a + x)/2$, $e = x - a$; // Деление отрезка пополам
иначе
$u = (x + c)/2$, $e = c - x$;
если $|u - x| < \varepsilon$ то
$u = x + \text{sign}(u - x)\varepsilon$; // Задаём минимальную близость между u и x
$d = |x - u|$;
Вычисляем $f_u = f(u)$, $f'_u = f'(u)$;
если $f_u \le f_x$ то
если $u \ge x$ то
$a = x$;
иначе
$c = x$;
$v = w, w = x, x = u$, $f_v = f_w, f_w = f_x, f_x = f_u$, $f'_v = f'_w, f'_w = f'_x, f'_x = f'_u$;
иначе
если $u \ge x$ то
$c = u$;
иначе
$a = u$;
если $f_u \le f_w$ или $w = x$ то
$v = w, w = u$, $f_v = f_w, f_w = f_u$, $f'_v = f'_w, f'_w = f'_u$;
иначе если $f_u \le f_v$ или $v = x$ или $v = w$ то
$v = u, f_v = f_u, f'_v = f'_u$;

- на этапе поиска новой точки используется деление отрезка пополам вместо золотого сечения;
- при сокращении интервала поиска решения используются тесты по значениям функции, а не производной, что гарантирует нахождение точки минимума внутри интервалов поиска на всех итерациях.

Схема модифицированного метода Брента представлена в Алгоритме 3[4].

Поиск ограничивающего сегмента

Если начальный интервал поиска минимума функции неизвестен, а минимизируемая функция является многомодальной, то для запуска описанных выше методов минимизации необходимо предварительно найти тройку точек $a < b < c$ таких, что $f(b) < f(a)$ и $f(b) < f(c)$. Для непрерывной функции это условие гарантирует наличие локального минимума внутри (a, c). Для поиска таких троек можно воспользоваться Алгоритмом 4, который комбинирует экстраполяции по параболе и по золотому сечению.

[4] полный код метода см. в [1], стр. 406–408.

Алгоритм 4: Схема поиска ограничивающего сегмента

Вход: Начальная точка a;
Выход: Тройка $a < b < c$: $f_b < f_a$, $f_b < f_c$;
  Взять точку b рядом с a и вычислить f_a, f_b; если $f_b > f_a$, то поменять a и b местами;
  Инициализация $K = (\sqrt{5}+1)/2$, $L = 100$;
  $c = b + K(b-a)$, $f_c = f(c)$;
  пока $f_c < f_b$
    Параболическая экстраполяция через точки a, b, c; u – минимум параболы;
    Предельное $u_{lim} = c + L(c-b)$;
    если $u \in (b, c)$ то
      Вычисляем $f_u = f(u)$;
      если $f_u < f_c$ то
        $a = b, b = u, f_a = f_b, f_b = f_u$;
        Выход из цикла;
      иначе если $f_u > f_b$ то
        $c = u, f_c = f_u$;
        Выход из цикла;
      $u = c + K(c-b)$, $f_u = f(u)$;
    иначе если $u \in (c, u_{lim})$ то
      Вычисляем $f_u = f(u)$;
      если $f_u < f_c$ то
        $b = c, c = u, u = c + K(c-b)$, $f_b = f_c, f_c = f_u, f_u = f(u)$;
    иначе если $u > u_{lim}$ то
      $u = u_{lim}$, $f_u = f(u)$;
    иначе
      $u = c + K(c-b)$, $f_u = f(u)$;
    $a = b, b = c, c = u$, $f_a = f_b, f_b = f_c, f_c = f_u$;

Неточная одномерная оптимизация

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

$$f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) \to \min_{\alpha \geq 0}. \tag{2}$$

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.

Определим условия, характеризующие «значимое уменьшение» значения функции на текущей итерации. Рассмотрим функцию $h(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$. Ее линейное приближение с помощью разложения в ряд Тейлора выглядит как $h(\alpha) \simeq f(\boldsymbol{x}_k) + \alpha \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k$. Рассмотрим следующие прямые:

$$h_B(\alpha) = f(\boldsymbol{x}_k) + \rho\alpha \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k,$$
$$h_C(\alpha) = f(\boldsymbol{x}_k) + (1-\rho)\alpha \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k.$$

Здесь $\rho \in (0, 1/2)$. Функции $h(\alpha), h_B(\alpha), h_C(\alpha)$ показаны на рис. 5, слева. Голдштайн предложил считать некоторую точку α_0 неточным решением задачи (2), если:

$$f(\boldsymbol{x}_k + \alpha_0 \boldsymbol{d}_k) \leq h_B(\alpha_0), \tag{3}$$
$$f(\boldsymbol{x}_k + \alpha_0 \boldsymbol{d}_k) \geq h_C(\alpha_0). \tag{4}$$

Условие (3) гарантирует уменьшение значения функции на текущей итерации, а условие (4) страхует от слишком маленьких шагов в оптимизации. Отдельно условие (3) известно также как условие Армихо.

Предложенные условия (3)-(4), вообще говоря, не гарантируют попадания точки точного минимума α_* в допустимый интервал $[\alpha_1, \alpha_2]$. Поэтому рассматриваются также т.н. условия Вольфа, в которых условие (4) заменено на следующее:

$$0 \geq \nabla f(\boldsymbol{x}_k + \alpha_0 \boldsymbol{d}_k)^T \boldsymbol{d}_k \geq \sigma \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k, \tag{5}$$

где $\sigma \in (0, 1)$. Данное условие проиллюстрировано на рис. 5, справа. Рекомендуемые значения параметров ρ и σ: ситуация $\rho = \sigma = 0.1$ позволяет находить достаточно точный минимум, но, возможно, ценой большого числа

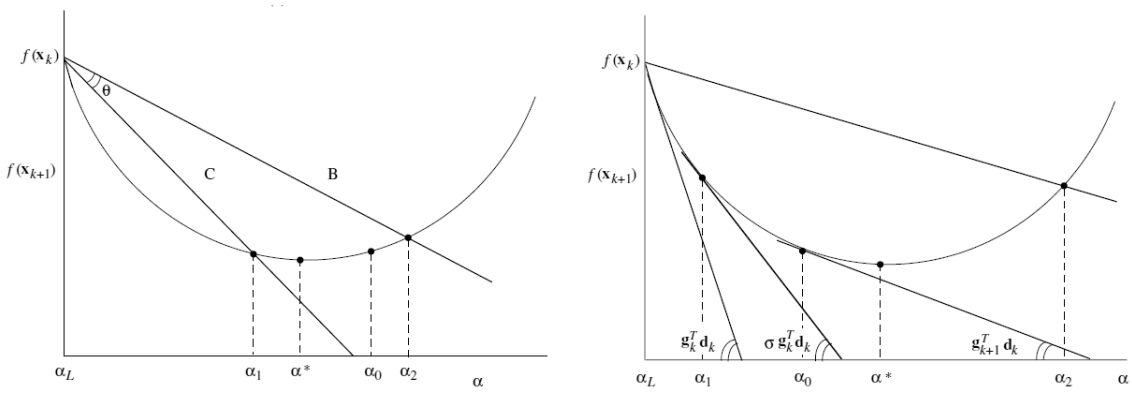

Рис. 5: Иллюстрация достаточных условий для неточной одномерной минимизации. Слева: условия Голдштайна-Деккера, справа: модификация Вольфа-Флетчера.

обращений к оракулу, а ситуация $\rho = 0.1, \sigma = 0.7$ представляет собой хороший компромисс между точностью решения и скоростью работы алгоритма минимизации.

Для поиска точки, удовлетворяющей условию Армихо (3), можно использовать простую схему backtracking. В этой схеме выбирается некоторое начальное значение α_0, а затем новые точки вычисляются как $\alpha_j = \beta\alpha_{j-1}$, где $\beta \in [0.1, 0.8]$ – параметр, задаваемый пользователем. Значения α_j уменьшаются до тех пор, пока не будет выполнено условие (3).

Для поиска точки, удовлетворяющей условиям Вольфа или Голдштайна, предложен ряд комбинированных стратегий, аналогичных методам Брента. Подробнее см. [2], [3].

Список литературы

[1] Numerical Recipes: The Art of Scientific Computing, 3-d edition // Cambridge University Press, 2007.

[2] R. Fletcher. Practical Methods of Optimization, 2-d edition // John Wiley and Sons, 2000.

[3] J.J. More, D.J. Thuente. Line search algorithms with guaranteed sufficient decrease // ACM Transactions on Mathematical Software, 20 (1994), pp. 286—307.