第5章

优化算法基础知识

学习要点

1. 理解优化算法的收敛性和收敛阶等概念.
2. 掌握一维牛顿法和割线法.
3. 掌握区间分割法.
4. 掌握几种常用的线搜索算法及其收敛性分析.

5.1 算法的收敛性与收敛速度

绝大多数非线性优化问题是无法求出其解析解的, 只能通过数值逼近的方法求其近似解.

优化问题数值算法的基本原理是从某个初始点 $x_0 \in \mathbb{R}^n$ 出发, 按照某种计算步骤产生一列点 $\{x_k : k = 0, 1, 2, \cdots\}$, 使得 x_k 逼近该优化问题的最优点 (或局部最优点). 当然, 最优点或局部最优点事先是不知道的, 有时候也用满足某种最优性条件的点代替它, 例如 KKT 点.

用 X^* 表示要寻找的最优点的集合, $\{x_k\}$ 表示由某种算法产生的点列, 如果

$$d(x_k, X^*) := \inf_{x \in X^*} \|x_k - x\|_2 \to 0, \qquad k \to \infty, \tag{5.1}$$

则称 $\{x_k\}$ **弱收敛于** X^*; 如果存在 $x^* \in X^*$ 使得

$$\lim_{k \to \infty} \|x_k - x^*\|_2 = 0, \tag{5.2}$$

则称 $\{x_k\}$ **强收敛于** X^*.

显然, 如果 $\{x_k\}$ 强收敛于 X^*, 则 $\{x_k\}$ 必弱收敛于 X^*.

接下来讨论收敛速度的问题. 设 $\{x_k\}$ 收敛于 $x^* \in X^*$, 对于 $p \geqslant 0$, 称

$$Q_p := \overline{\lim_{k \to \infty}} \frac{\|x_{k+1} - x^*\|_2}{\|x_k - x^*\|_2^p} \tag{5.3}$$

为 $\{x_k\}$ 的**商收敛因子**, 简称为Q **因子**; 称

$$R_p := \begin{cases} \overline{\lim\limits_{k\to\infty}} \|x_k - x^*\|_2^{1/k}, & p = 1 \\ \overline{\lim\limits_{k\to\infty}} \|x_k - x^*\|_2^{1/p^k}, & p > 1 \end{cases} \tag{5.4}$$

为 $\{x_k\}$ 的**根收敛因子**, 简称为R **因子**. 称

$$O_Q := \inf\{p \geqslant 1 :\ Q_p = \infty\} \tag{5.5}$$

为 $\{x_k\}$ 的**商收敛阶**, 简称为Q **收敛阶**; 称

$$O_R := \inf\{p \geqslant 1 :\ R_p = \infty\} \tag{5.6}$$

为 $\{x_k\}$ 的**根收敛阶**, 简称为R **收敛阶**.

对于收敛点列 $\{x_k\}$, 可以证明其 Q 收敛阶不会超过其 R 收敛阶[34]. 目前文献中用得比较多的是 Q 收敛阶, **以后我们提到的收敛因子都是指 Q 收敛因子, 收敛阶都是指 Q 收敛阶**.

定义 5.1 设点列 $\{x_k\}$ 收敛于 x^*, 如果 $Q_1 = 0$, 则称 $\{x_k\}$ **超线性收敛于** x^*; 如果 $0 < Q_1 < 1$, 则称 $\{x_k\}$ **线性收敛于** x^*; 如果 $Q_1 \geqslant 1$, 则称 $\{x_k\}$ **次线性收敛于** x^*.

如果 $\{x_k\}$ 超线性收敛于 x^*, 则有

$$\overline{\lim_{k\to\infty}} \frac{\|x_{k+1} - x^*\|_2}{\|x_k - x^*\|_2} = 0, \tag{5.7}$$

由于

$$\begin{aligned} \frac{\|x_{k+1} - x^*\|_2}{\|x_k - x^*\|_2} &\geqslant \frac{\|x_{k+1} - x^*\|_2}{\|x_k - x_{k+1}\|_2 + \|x_{k+1} - x^*\|_2} \\ &= \frac{\dfrac{\|x_{k+1} - x^*\|_2}{\|x_k - x_{k+1}\|_2}}{1 + \dfrac{\|x_{k+1} - x^*\|_2}{\|x_k - x_{k+1}\|_2}}, \end{aligned} \tag{5.8}$$

同理可得

$$\frac{\|x_{k+1} - x^*\|_2}{\|x_k - x_{k+1}\|_2} \geqslant \frac{\dfrac{\|x_{k+1} - x^*\|_2}{\|x_k - x^*\|_2}}{1 + \dfrac{\|x_{k+1} - x^*\|_2}{\|x_k - x^*\|_2}}, \tag{5.9}$$

因此式 (5.7) 成立当且仅当

$$\overline{\lim_{k\to\infty}} \frac{\|x_{k+1} - x^*\|_2}{\|x_k - x_{k+1}\|_2} = 0. \tag{5.10}$$

这说明当相邻两次迭代的结果相差很小时, x_{k+1} 已经距离 x^* 很近了, 这就为算法提供了一个很好的停机判断准则.

定义 5.2 设点列 $\{x_k\}$ 收敛于 x^*, 如果 $Q_2 = 0$, 则称 $\{x_k\}$ **超二次收敛于** x^*; 如果 $0 < Q_2 < \infty$, 则称 $\{x_k\}$ **二次收敛于** x^*; 如果 $Q_2 = \infty$, 则称 $\{x_k\}$ **次二次收敛于** x^*.

类似地, 可以定义超三次收敛、三次收敛、次三次收敛等概念.

5.2 一维牛顿法与割线法

本节我们考虑一维优化问题

$$\min_{x\in\mathbb{R}} f(x) \tag{5.11}$$

的优化算法. 我们假设 $f(x)$ 具有二阶连续导数. 如果 x^* 是式 (5.11) 的极小点, 则必有 $f'(x^*) = 0$, 因此可以通过求方程 $f'(x) = 0$ 的根求得候选极小点. 当然, 求这个方程的根也并非易事, 需要用逐次逼近的方法求其近似值. **牛顿法** (也称为**牛顿切线法**) 就是这样一种方法.

下面介绍牛顿法的基本思想. 首先选一个初始点 $x_0 \in \mathbb{R}$, 如果 $f'(x_0) = 0$, 则 x_0 就是我们要找的点. 如果 $f'(x_0) \neq 0$, 则可以考虑对 x_0 作适当的修正, 例如使 $x_0 + h$ 比 x_0 更接近 f' 的根 x^*. 如何确定修正量 h 呢? 我们当然希望 $f'(x_0 + h) = 0$, 但这个方程与 $f'(x) = 0$ 是等价的, 无法求出其精确解, 于是考虑在 x_0 附近用一阶 Taylor 多项式逼近 f', 即

$$f'(x_0 + h) = f'(x_0) + f''(x_0)h + o(|h|) \approx f'(x_0) + f''(x_0)h, \tag{5.12}$$

通过求解近似方程 $f'(x_0) + f''(x_0)h = 0$ 确定修正量 h. 这是一个线性方程, 很容易求得其唯一的实根为

$$h = -\frac{f'(x_0)}{f''(x_0)}, \tag{5.13}$$

令

$$x_1 = x_0 + h = x_0 - \frac{f'(x_0)}{f''(x_0)}, \tag{5.14}$$

则在一定的条件下, x_1 确实比 x_0 更接近 x^*.

如果 $f'(x_1) \neq 0$, 则可以重复以上过程对 x_1 进行修正: 在 x_1 点附近用一阶 Taylor 多项式 $f'(x_1) + f''(x_1)h$ 近似代替 $f'(x_1 + h)$, 通过解线性方程 $f'(x_1) + f''(x_1)h = 0$ 确定修正量 $h = -f'(x_1)/f''(x_1)$, 令

$$x_2 = x_1 + h = x_1 - f'(x_1)/f''(x_1). \tag{5.15}$$

如果 $f'(x_2) \neq 0$, 则再重复以上步骤, 直至某次迭代后得到的点 x_k 满足 $f'(x_k) = 0$ 或者其绝对值足够小, 停机并输出 x_k.

以上计算过程可用算法表示如下.

Algorithm 1 (一维牛顿法 I)

Input:

The initial point $x_0 \in \mathbb{R}$;

The tolerance bound ε;

Output:

The approximation of the extreme point x

$x \leftarrow x_0$;

while $|f'(x)| \geqslant \varepsilon$ **do**

compute $h = -f'(x)/f''(x)$;

$x \leftarrow x + h$;

end while

Output x;

关于以上算法的收敛性, 有下列结果.

定理 5.1 设 f 二次连续可微, $f'(x^*) = 0, f''(x^*) \neq 0$, 则当 x_0 充分靠近 x^* 时, 有 $x_k \to x^*$, 且

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = 0, \tag{5.16}$$

即算法 Algorithm 1 是超线性收敛的.

证明 根据算法的迭代公式得

$$\begin{aligned} x_{k+1} - x^* &= x_k - f'(x_k)/f''(x_k) - x^* \\ &= x_k - x^* - \left(f'(x^*) + \int_{x^*}^{x_k} f''(x)\mathrm{d}x\right)/f''(x_k) \\ &= \int_{x^*}^{x_k} [f''(x_k) - f''(x)]\,\mathrm{d}x/f''(x_k), \end{aligned} \tag{5.17}$$

其中, 最后一个等号用到了条件 $f'(x^*) = 0$. 由于 $f''(x^*) \neq 0$, 不妨设 $f''(x^*) > 0$; 由于 f'' 连续, 因此存在 $\delta > 0$ 使得当 $x, y \in [x^* - \delta, x^* + \delta]$ 时恒有 $f''(x) > f''(x^*)/2, |f''(x) - f''(y)| < f''(x^*)/4$, 从而有

$$\frac{|f''(x) - f''(y)|}{f''(x)} < \frac{1}{2}, \tag{5.18}$$

因此当 $x_k \in [x^* - \delta, x^* + \delta]$ 时有

$$|x_{k+1} - x^*| = \left|\int_{x^*}^{x_k} \frac{f''(x_k) - f''(x)}{f''(x_k)}\mathrm{d}x\right| \leqslant \frac{1}{2}|x_k - x^*|, \tag{5.19}$$

由此立刻推出 $x_k \to x^*$. 既然 $x_k \to x^*$, 当 $k \to \infty$ 时必有

$$\left|\frac{f''(x_k)-f''(x)}{f''(x_k)}\right| \to 0, \qquad \forall\, x \in [x^*, x_k], \tag{5.20}$$

而且是一致收敛的, 因此有

$$|x_{k+1}-x^*| = \left|\int_{x^*}^{x_k} \frac{f''(x_k)-f''(x)}{f''(x_k)}\mathrm{d}x\right| = |x_k-x^*|\cdot o(1), \tag{5.21}$$

即式 (5.16) 成立. □

算法 Algorithm 1 求出的只是导数的零点, 既可能是局部极小值点, 也可能是局部极大值点, 一个完善的算法应能够判断是极小值点还是极大值点, 如果落入极大值点, 还要能够跳出来继续寻找极小值点. 将以上因素都考虑进去的算法就是全局牛顿算法, 我们不作深入讨论, 感兴趣的读者可参见文献 [31] 的 2.1 节.

一维牛顿法的关键迭代公式为

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}, \tag{5.22}$$

需要用到二阶导数 $f''(x_k)$. 如果用差商近似代替 $f''(x_k)$, 即

$$f''(x_k) \approx \frac{f'(x_k)-f'(x_{k-1})}{x_k-x_{k-1}}, \tag{5.23}$$

便得到了迭代公式

$$x_{k+1} = x_k - \frac{f'(x_k)(x_k-x_{k-1})}{f'(x_k)-f'(x_{k-1})}, \tag{5.24}$$

由此得到的算法便是**割线法**, 具体算法如 Algorithm 2 所示.

Algorithm 2 (割线法)

Input:

The initial point $x_0, x_1 \in \mathbb{R}$;

The tolerance bound ε;

Output:

The approximation of the extreme point x_{II}

$x_I \leftarrow x_0$;

$x_{II} \leftarrow x_1$;

while $|f'(x_{II})| \geqslant \varepsilon$ **do**

compute $h = -f'(x_{II})(x_{II}-x_I)/(f'(x_{II})-f'(x_I))$;

$x_I \leftarrow x_{II}$;

$x_{II} \leftarrow x_{II} + h$;

end while

Output x_{II};

关于割线法的收敛性的证明以及收敛阶的估计不作深入讨论, 感兴趣的读者可参见文献 [31] 的 2.2 节.

5.3 区间分割法

本节讨论求解**单峰函数**最小值问题的方法——**区间分割法**. 称 f 是区间 $[a,b]$ 上的**单峰函数 (unimodal function)**, 如果存在 $x^* \in [a,b]$ 使得 f 在 $[a,x^*]$ 上严格单调递减, 在 $[x^*,b]$ 上严格单调递增.

从单峰函数的定义不难发现, $[a,b]$ 上的单峰函数 f 有唯一的全局极小值点 x^*. 如何定位极小值点 x^* 呢? 我们可以在区间 $[a,b]$ 上取两点 x_1, x_2 使得

$$a < x_1 < x_2 < b,$$

如果 $f(x_1) < f(x_2)$, 则可以断定 $x^* \in [a,x_2]$; 如果 $f(x_1) > f(x_2)$, 则可以断定 $x^* \in [x_1,b]$; 如果 $f(x_1) = f(x_2)$, 则可以断定 $x^* \in [x_1,x_2]$. 综上所述, 通过计算两点的函数值 $f(x_1)$ 和 $f(x_2)$, 我们可以把 x^* 定位在一个更小的区间中, 这就是区间分割法的核心思想. 通过反复的区间分割, 可以给出极小值点 x^* 任意精度的逼近.

如何选择区间分点 x_1 和 x_2 呢? 最容易想到的就是选择 x_1 和 x_2 为区间 $[a,b]$ 的三等分点:

$$x_1 = a + \frac{1}{3}(b-a), \qquad x_2 = a + \frac{2}{3}(b-a), \tag{5.25}$$

对应的区间分割算法就是算法 Algorithm 3.

Algorithm 3 (区间分割算法 I)

Input:
The initial interval $[a,b]$;
The tolerance bound ε;
Output:
The approximation of the extreme point x_m

$a_m \leftarrow a$;
$b_m \leftarrow b$;
while $b_m - a_m \geqslant \varepsilon$ **do**
  $x_1 \leftarrow a_m + (b_m - a_m)/3$;
  $x_2 \leftarrow a_m + 2(b_m - a_m)/3$;
  if $f(x_1) < f(x_2)$ **then**
    $b_m \leftarrow x_2$;
  else
    if $f(x_1) > f(x_2)$ **then**
      $a_m \leftarrow x_1$;
    else
      $a_m \leftarrow x_1$;

$b_m \leftarrow x_2$;
end if
end if
end while
Output $x_m = (a_m + b_m)/2$;

算法 Algorithm 3 输出结果的精度由最终的小区间 $[a_m, b_m]$ 的长度决定, 满足不等式

$$|x_m - x^*| \leqslant \frac{b_m - a_m}{2}. \tag{5.26}$$

因此算法的收敛速度取决于每次区间分割后极小值点 x^* 所在的估计区间与原来的估计区间相比缩小了多少. 设分割前的区间为 $[a_{m-1}, b_{m-1}]$, 分割后 x^* 所在的区间为 $[a_m, b_m]$, 则 $[a_m, b_m]$ 或者是 $[a_{m-1}, x_2]$, 或者是 $[x_1, b_{m-1}]$, 或者是 $[x_1, x_2]$, 因此分割后与分割前的区间长度之比满足

$$\begin{aligned} \frac{b_m - a_m}{b_{m-1} - a_{m-1}} &\leqslant \frac{\max\{x_2 - a_{m-1}, b_{m-1} - x_1, x_2 - x_1\}}{b_{m-1} - a_{m-1}} \\ &= \frac{\max\{x_2 - a_{m-1}, b_{m-1} - x_1\}}{b_{m-1} - a_{m-1}}. \end{aligned} \tag{5.27}$$

当 x_1 和 x_2 是区间 $[a_{m-1}, b_{m-1}]$ 的三等分点时, 有

$$\frac{b_m - a_m}{b_{m-1} - a_{m-1}} \leqslant \frac{2}{3}. \tag{5.28}$$

经过 m 次区间分割后得到的近似极小值点 x_m 的误差估计为

$$|x_m - x^*| \leqslant \frac{b_m - a_m}{2} \leqslant \frac{1}{2}\left(\frac{2}{3}\right)^m (b - a). \tag{5.29}$$

为了提高计算效率, 我们可以在分割点的选择及算法的设计上做一些改进. 我们选择分点 $a < x < y < b$ 使得

$$x - a = b - y, \qquad \frac{y - a}{b - a} = \frac{x - a}{y - a}, \tag{5.30}$$

令 $\gamma = (y - a)/(b - a)$, 则有

$$\gamma = \frac{1}{\gamma} - 1, \tag{5.31}$$

整理后得到一元二次方程

$$\gamma^2 + \gamma - 1 = 0, \tag{5.32}$$

这个方程有两个实根, 但由于 $\gamma > 0$, 因此只能是

$$\gamma = \frac{\sqrt{5} - 1}{2} \approx 0.618, \tag{5.33}$$

这正是**黄金分割比**. 不难看出

$$\frac{b-y}{b-x}=\frac{x-a}{y-a}=\gamma. \tag{5.34}$$

如果 $f(x)<f(y)$, 则极小值点 x^* 必落在区间 $[a,y]$ 中, 只需令 $x_1=a+(y-x)$, 则 a,x_1,x,y 满足黄金分割条件, 只需再计算 $f(x_1)$ 的值便可进一步分割区间 $[a,y]$, 更精确地定位 x^*; 如果 $f(x)\geqslant f(y)$, 则极小值点 x^* 必落在区间 $[x,b]$ 中, 只需令 $y_1=b-(y-x)$, 则 x,y,y_1,b 满足黄金分割条件, 只需再计算 $f(y_1)$ 的值便可进一步分割区间 $[x,b]$, 更精确地定位 x^*.

现在设计**黄金分割算法**如 Algorithm 4 所示.

Algorithm 4 (黄金分割算法)

Input:
The initial interval $[a,b]$;
The tolerance bound ε;
Output:
The approximation of the extreme point x_m

$a_m \leftarrow a$;
$b_m \leftarrow b$;
$\gamma \leftarrow (\sqrt{5}-1)/2$;
$y \leftarrow a_m+\gamma(b_m-a_m)$;
$x \leftarrow a_m+(b_m-y)$;
$v_x \leftarrow f(x)$;
$v_y \leftarrow f(y)$;
while $b_m-a_m \geqslant \varepsilon$ **do**
  if $v_x < v_y$ **then**
    $b_m \leftarrow y$;
    $y \leftarrow x$;
    $v_y \leftarrow v_x$;
    $x \leftarrow a_m+(b_m-y)$;
    $v_x \leftarrow f(x)$;
  else
    $a_m \leftarrow x$;
    $x \leftarrow y$;
    $v_x \leftarrow v_y$;
    $y \leftarrow b_m-(x-a_m)$;
    $v_y \leftarrow f(y)$;
  end if
end while
Output $x_m=(a_m+b_m)/2$;

黄金分割算法 Algorithm 4 每次分割后所得的估计区间的长度是分割前估计区间的长度的 γ 倍, 因此有误差估计公式

$$|x_m-x^*| \leqslant \frac{b_m-a_m}{2} \leqslant \frac{1}{2}\gamma^m(b-a), \tag{5.35}$$

其中, m 是区间分割的次数. 与算法 Algorithm 3 相比有两点好处: 一是黄金分割比 $\gamma < 2/3$, 从而黄金分割算法收敛得更快; 二是黄金分割算法的每次迭代只需计算一个点的函数值, 而算法 Algorithm 3 的每次迭代需要计算两个点的函数值, 因此黄金分割算法的效率更高.

如果 f 存在一阶导数, 则极小值点 x^* 是唯一满足 $f'(x) = 0$ 的点, 因此可以通过求方程 $f'(x) = 0$ 的根来寻找 x^*. 可以用**对分法**求方程 $f'(x) = 0$ 的近似根: 设 c 是区间 $[a, b]$ 的中点, 如果 $f'(c) < 0$, 则方程的根 $x^* \in [c, b]$, 否则 $x^* \in [a, c]$, 继续对分 $[c, b]$ 或 $[a, c]$ 可得到更小的估计区间, 重复此过程可得到 x^* 的任意精度的逼近. 具体算法如 Algorithm 5 所示.

Algorithm 5 (对分法)

Input:
The initial interval $[a, b]$;
The tolerance bound ε;
Output:
The approximation of the extreme point x_m

$B \leftarrow 1$;
while $b - a \geqslant \varepsilon$ **do**
  $c \leftarrow (a + b)/2$;
  if $f'(c) < 0$ **then**
    $a \leftarrow c$;
  else
    if $f'(c) = 0$ **then**
      Output $x_m = c$;
      $B \leftarrow 0$;
      Stop;
    else
      $b \leftarrow c$;
    end if
  end if
end while
if $B = 1$ **then**
  Output $x_m = (a + b)/2$;
end if

对分法每次区间分割后估计区间的长度缩减为分割前估计区间的长度的一半, 因此有误差估计公式

$$|x_m - x^*| \leqslant \left(\frac{1}{2}\right)^{m+1} (b - a), \tag{5.36}$$

其中, m 是区间分割的次数. 对分法的效率比三等分法和黄金分割法的都高, 前提是 f 可导.

5.4 线 搜 索

线搜索 (line search) 是指找多元函数在一条直线上的最大值点或最小值点. 线搜索问题一般形式为

$$\min_{s>0} f(x_k + sd_k), \tag{5.37}$$

其中, $x_k \in \mathbb{R}^n$ 是某些优化算法迭代过程中到达的某一点, $d_k \in \mathbb{R}^n$ 代表接下来的行走方向, 通常满足 $d_k^{\mathrm{T}} \nabla f(x_k) < 0$, 沿着这个方向走能够保证目标函数 f 的值下降. 线搜索问题之所以重要是因为它出现在许多非线性优化算法的迭代步骤中, 对这些优算法的计算效率都有直接影响.

由于线搜索问题本质上是一维优化问题, 因此可以用前面两节介绍的一维优化算法来求解.

记 $\varphi(s) = f(x_k + sd_k)$, 并假设 f 具有一阶连续偏导数, 则有

$$\varphi'(s) = d_k^{\mathrm{T}} \nabla f(x_k + sd_k), \tag{5.38}$$

因此优化问题 (5.37) 在 $s = s^*$ 处取极小值的必要条件为

$$d_k^{\mathrm{T}} \nabla f(x_k + s^* d_k) = 0. \tag{5.39}$$

如果 f 是二阶连续可微的, 则有

$$\varphi''(s) = d_k^{\mathrm{T}} \nabla^2 f(x_k + sd_k) d_k, \tag{5.40}$$

因此优化问题 (5.37) 在 $s = s^*$ 处取局部极小值的充分条件为

$$d_k^{\mathrm{T}} \nabla f(x_k + s^* d_k) = 0, \qquad d_k^{\mathrm{T}} \nabla^2 f(x_k + s^* d_k) d_k > 0. \tag{5.41}$$

如果 f 是凸函数, 则 φ 也是凸函数, 此时 $\varphi'(s^*) = 0$ 足以保证 φ 在 $s = s^*$ 处取得全局极小值.

精确求解优化问题 (5.37) 称为**精确线搜索 (exact line search)**, 前面两节介绍的牛顿法、割线法和区间分割法都可以用于精确线搜索. 精确线搜索通常需要很多的计算时间, 会严重拖慢优化算法的计算速度, 因此并不常用. 实践中用得更多的是**非精确线搜索 (inexact line search)**, 目的是快速地找到某个近似解 s_I 以满足某些指定的条件, 这些条件能够保证非线性优化算法的整体收敛性.

一种比较常用的非精确线搜索是 **Wolfe 线搜索**, 它寻找 $s_I > 0$ 满足

$$f(x) - f(x + s_I d) \geqslant -s_I b_1 d^{\mathrm{T}} \nabla f(x), \tag{5.42}$$

$$d^{\mathrm{T}} \nabla f(x + s_I d) \geqslant b_2 d^{\mathrm{T}} \nabla f(x), \tag{5.43}$$

其中, b_1 和 b_2 是两个常数, 且满足 $0 < b_1 \leqslant b_2 < 1$[35]. 条件 (5.42) 是为了保证当自变量由 x 走到新的位置 $x + s_I d$ 时目标函数 f 的值下降得足够多 (与步长 s_I 成正比), 通常称