第 3 章：无约束优化

(1)：下降法

授课教师：曹语

课程主页：https://yucaoyc.github.io/math3806

背景和目标

我们已经见过一些典型的无约束凸优化问题：

$$\text{minimize } f(\boldsymbol{x})$$

例如

- 无约束最小二乘：可用于数据拟合
- 无约束几何规划

问题：为什么这里不讨论“无约束线性规划”？

背景和目标

我们已经见过一些典型的无约束凸优化问题：

$$\text{minimize } f(\boldsymbol{x})$$

例如

- 无约束最小二乘：可用于数据拟合
- 无约束几何规划

问题：为什么这里不讨论“无约束线性规划”？

因为线性函数在整个 $\mathbb{R}^n$ 上通常没有最小值：沿某个方向可以一直减小。

本章目标

学会如何用迭代算法一步一步逼近最优解

- 下降法（及其变式）$\rightarrow$ **本次重点**
- 牛顿法（下一部分继续）

整体思路

本节其实在回答 3 个问题：

- 从当前位置出发，**往哪个方向走**会让函数值下降？
- 方向确定后，**一步走多远**比较合适？
- 算法什么时候可以认为**已经够好了，可以停下**？

这就是后面所有公式背后的主线。

情景设定：本节先研究哪一类问题?

暂时只看如下情形：

$$\text{minimize } f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^n$$

并作如下假设：

- f 是凸函数
- ∇f 和 $\nabla^2 f$ 可计算
- 最优解 $\boldsymbol{x}^\star$ 存在，最优值记为 $p^\star = f(\boldsymbol{x}^\star)$

算法： 构造一个点列 $\{\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(k)}, \cdots\}$，希望函数值不断下降：

$$f(\boldsymbol{x}^{(0)}) \geq f(\boldsymbol{x}^{(1)}) \geq \cdots \geq f(\boldsymbol{x}^{(k)}) \geq \cdots$$

也就是说，我们不是一下子跳到最优解，而是像“下山”一样，一步一步往低处走。

两个基本问题

由于函数值一直下降，算法实际上始终在下面的水平集里活动：

$$S = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leqslant f\left(\boldsymbol{x}^{(0)}\right) \right\}$$

为简化起见，假设 S 是闭集。

设计一个下降算法，需要回答两件事：

Q1 已知当前点 $\boldsymbol{x}^{(k)}$，怎样得到下一个点 $\boldsymbol{x}^{(k+1)}$，并保证 $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$?

Q2 什么时候可以停止？也就是怎样判断“已经够接近最优解”？

两个基本问题

由于函数值一直下降，算法实际上始终在下面的水平集里活动：

$$S = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) \leqslant f\left(\boldsymbol{x}^{(0)}\right) \right\}$$

为简化起见，假设 S 是闭集。

设计一个下降算法，需要回答两件事：

Q1 已知当前点 $\boldsymbol{x}^{(k)}$，怎样得到下一个点 $\boldsymbol{x}^{(k+1)}$，并保证 $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$?

Q2 什么时候可以停止？也就是怎样判断“已经够接近最优解”？

一个常见停止准则是 $\left\|\nabla f(\boldsymbol{x}^{(k)})\right\| \leq \eta$。原因是：若 $\boldsymbol{x}^\star$ 是可微问题的最优解，通常满足 $\nabla f(\boldsymbol{x}^\star) = 0$。

目录

几何直觉：像下山一样找路

把 $f(\boldsymbol{x})$ 想象成山地的高度：

- 当前点 $\boldsymbol{x}^{(k)}$ 就是你现在站的位置
- 搜索方向 $\Delta\boldsymbol{x}^{(k)}$ 就是“打算朝哪个方向走”
- 步长 $t^{(k)}$ 就是“这一脚迈多大”

下降法的目标： 每走一步，都让高度 $f(\boldsymbol{x})$ 变低。

后面所有内容都可以理解成：先决定方向，再决定步子大小。

下降方法

从当前点 $\boldsymbol{x}^{(k)}$ 出发，沿某个方向 $\Delta\boldsymbol{x}^{(k)}$ 前进：

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\Delta\boldsymbol{x}^{(k)}, \qquad t^{(k)} > 0$$

- $\Delta\boldsymbol{x}^{(k)}$ 称为**搜索方向**
- $t^{(k)}$ 称为第 k 次迭代的**步长**

为了让函数值下降，考虑一维函数 $g(t) = f(\boldsymbol{x}^{(k)} + t\Delta\boldsymbol{x}^{(k)})$。若希望 $t = 0$ 附近朝前走会让函数值减小，就要有

$$0 > g'(0) = \nabla f(\boldsymbol{x}^{(k)})^\top \Delta\boldsymbol{x}^{(k)}$$

因此需要

$$\nabla f(\boldsymbol{x}^{(k)})^\top \Delta\boldsymbol{x}^{(k)} < 0$$

满足这个不等式的方向 $\Delta\boldsymbol{x}^{(k)}$，称为**下降方向**。

下降方向一定存在吗?

问题：什么时候能找到下降方向?

分两种情况讨论：

- 若 $\nabla f(\boldsymbol{x}^{(k)}) \neq \boldsymbol{0}$，那么取 $\Delta \boldsymbol{x}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$ 即可，因为

$$\nabla f(\boldsymbol{x}^{(k)})^{\top} \Delta \boldsymbol{x}^{(k)} = -\left\|\nabla f(\boldsymbol{x}^{(k)})\right\|_2^2 < 0$$

- 若 $\nabla f(\boldsymbol{x}^{(k)}) = \boldsymbol{0}$，则一阶信息已经不能给出下降方向；对凸可微函数，这时 $\boldsymbol{x}^{(k)}$ 其实已经是全局最优解。

所以：在非最优点处，至少可以用负梯度作为一个下降方向。

通用下降方法

Algorithm 1: 通用下降方法

Input: $\boldsymbol{x}^{(0)}$

1 $k = 0$

2 **while** 终止条件不满足，且 $k < N$ **do**

3 　确定下降方向 $\Delta \boldsymbol{x}^{(k)}$

4 　直线搜索：选择步长 $t^{(k)} > 0$

5 　$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)} \Delta \boldsymbol{x}^{(k)}$

6 　$k \leftarrow k + 1$

7 **end**

此处的通用方法中，有两处有待进一步确认：

任务 1 选一个“往哪里走”的方向

任务 2 决定“沿这个方向走多远”

目录

方式 (1)：精确直线搜索/Exact line search

$$t = \operatorname{argmin}_{s \geqslant 0} f(\boldsymbol{x} + s\Delta\boldsymbol{x})$$

直观理解： 方向先固定，再在这条直线上认真找“最好的那一步”。

优劣对比：

- 好处：每一步通常下降得更充分
- 坏处：每一步都要额外解一个一维优化问题，计算量可能较大

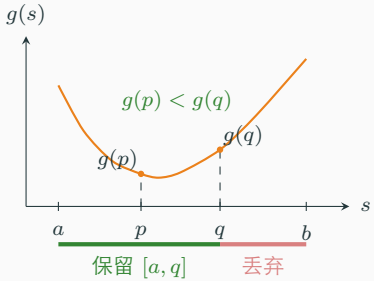
适用场景： 当函数值容易计算，而确定方向很贵时，可以考虑精确直线搜索。

精确直线搜索：实际怎么做？

先把多元问题沿着固定方向改写成一个一元函数：$g(s) = f(\boldsymbol{x} + s\Delta\boldsymbol{x}),\ s \geq 0$
然后求 $t = \operatorname{argmin}_{s\geq 0}\ g(s)$。假设已经找到一个可能包含最小点的区间 $[a, b]$。

例如：黄金分割搜索：

- 在区间 $[a, b]$ 内取两个点 $p = b - \tau(b - a)$，$q = a + \tau(b - a)$，其中 $\tau = \frac{\sqrt{5}-1}{2} \approx 0.618$
- 比较 $g(p)$ 和 $g(q)$
- 若 $g(p) < g(q)$，保留左边区间 $[a, q]$；否则保留右边区间 $[p, b]$
- 重复直到区间足够短



每次都删掉一段不可能包含最优步长的区间。

方式 (2)：回溯直线搜索/Backtracking line search

给定参数 $\alpha \in (0, 0.5), \beta \in (0, 1)$

Algorithm 2: 回溯直线搜索

Input: t_0 (一般选 $t_0 = 1$), 当前位置 $\boldsymbol{x}$, 下降方向 $\Delta \boldsymbol{x}$

Result: t

$t \leftarrow t_0$

while $f(\boldsymbol{x} + t\Delta \boldsymbol{x}) > f(\boldsymbol{x}) + \alpha t \nabla f(\boldsymbol{x})^\top \Delta \boldsymbol{x}$ **do**

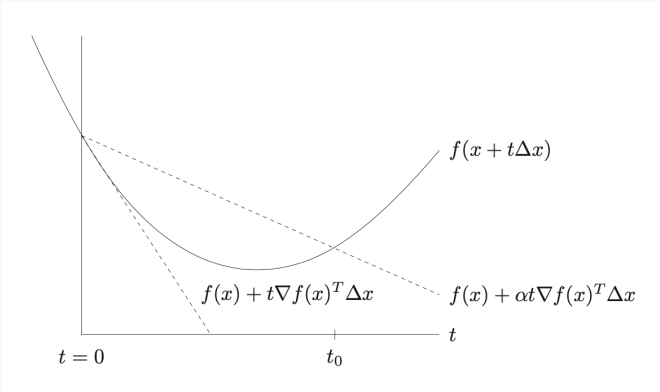
  令 $t \leftarrow \beta t$

end

- 思想：先试较大步长；如果降得不够，把步长乘上 β 缩小
- 由于 $\Delta \boldsymbol{x}$ 是下降方向，只要 t 足够小，上述条件一定能满足

条件 $f(\boldsymbol{x} + t\Delta \boldsymbol{x}) \leq f(\boldsymbol{x}) + \alpha t \nabla f(\boldsymbol{x})^\top \Delta \boldsymbol{x}$ 称为**Armijo 条件**，意思是“这一步至少要有足够明显的下降”。

回溯直线搜索：参数怎么理解?



- α 控制“下降多少才算合格”
- β 控制“步长缩小得有多快”
- 经验上：α 常取 $0.01 \sim 0.3$，β 常取 $0.1 \sim 0.8$

提示： α 太大时要求过严；β 太接近 1 时，每次只缩一点，搜索会比较慢。

目录

梯度下降法

梯度下降法： 考虑

$$\Delta \boldsymbol{x} = -\nabla f(\boldsymbol{x})$$

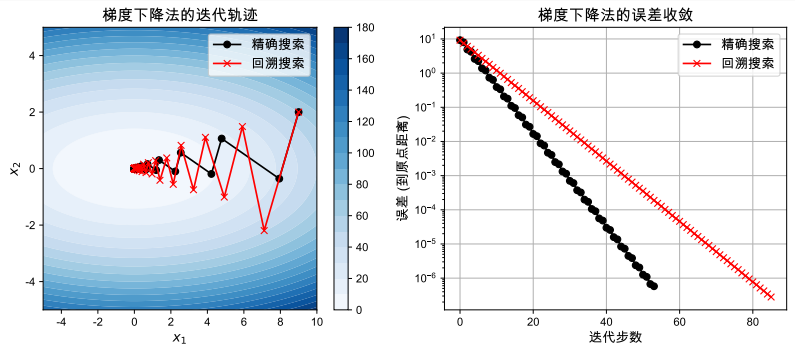
然后搭配精确直线搜索，或者回溯直线搜索来使用。

理解为：沿着函数上升最快方向的反方向前进。

- 为什么自然？因为梯度指向局部上升最快的方向
- 为什么常用？因为它只需要一阶导数，计算和实现都比较直接

实验 1：先看轨迹

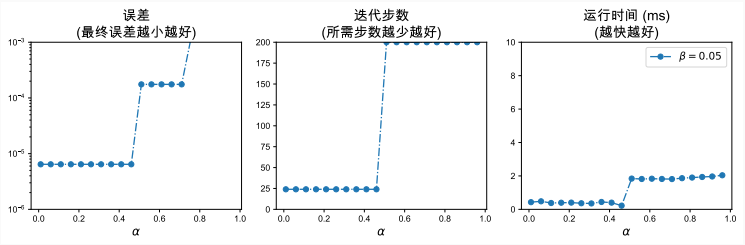
考虑如下的函数 $f(\boldsymbol{x}) = \frac{1}{2}\left(x_1^2 + \gamma x_2^2\right)$；某组实验结果如下（代码见 notebook）：



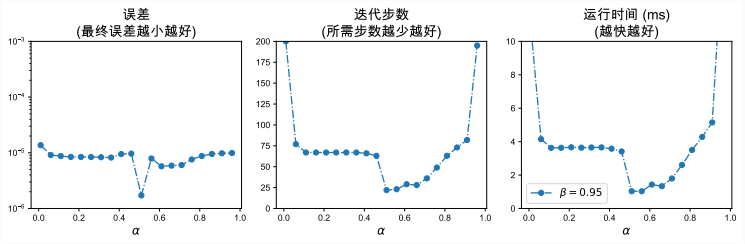
结论：两种线搜索都能收敛。

精确线搜索每一步更“讲究”；回溯线搜索更便宜，通常已经够用。

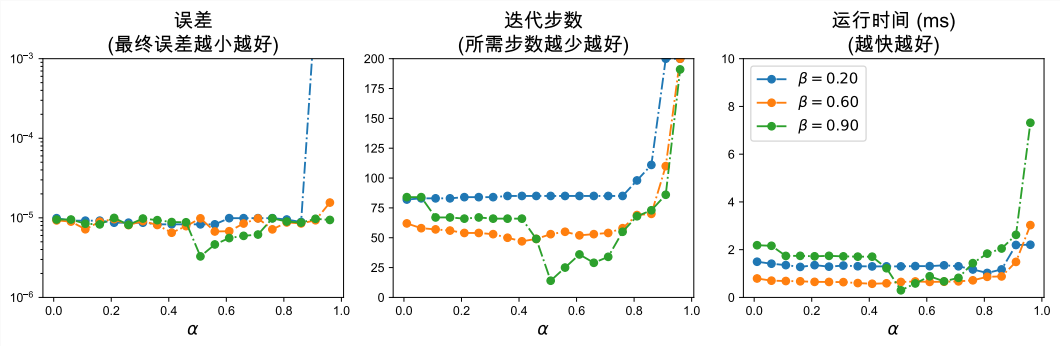
实验 2：回溯直线搜索的参数影响



读图时先看整体趋势：$\alpha < 1/2$ 是基本要求；在常见范围内，它对结果影响通常不大



不太希望 $\beta \approx 1$，因为每次只缩一点，搜索会偏慢



我们可得到如下观察：

- $\alpha < 1/2$ 是基本要求；在常用范围内，它对结果影响通常不大；
- $\beta \approx 1$ 时搜索太慢；$\beta \approx 0$ 时搜索又过于粗糙；
- 在经验取值范围内，α 和 β 往往不太会改变“是否收敛”，主要影响的是效率。

实验 3：为什么有时会走得很慢?

仍考虑二次函数

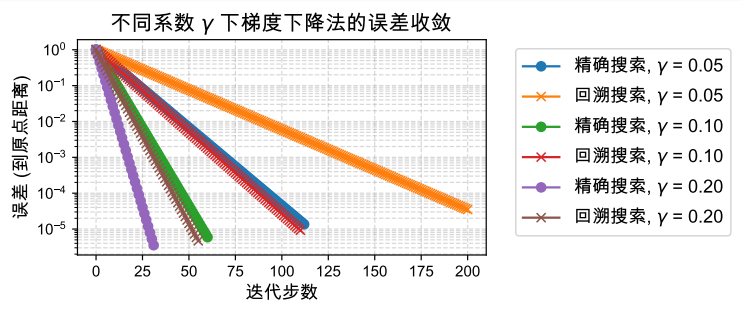
$$f(\boldsymbol{x}) = \frac{1}{2}\left(x_1^2 + \gamma x_2^2\right)$$

- 当 $\gamma \approx 1$ 时，等高线接近圆形，梯度下降通常走得比较顺
- 当 γ 很小或很大时，等高线会变得很狭长，轨迹容易出现之字形摆动
- 对梯度下降法 + 精确直线搜索，可以证明

$$\frac{f\left(\boldsymbol{x}^{(k+1)}\right)}{f\left(\boldsymbol{x}^{(k)}\right)} \leq \left(\frac{\gamma - 1}{\gamma + 1}\right)^2$$

  若初值为 $\boldsymbol{x}^{(0)} = (\gamma, 1)$ 时，等号可以取到。
- 当 $\gamma \to 0$ 或 $\gamma \to \infty$ 时，上式右端趋近于 1，意味着每一步只能减少一点点，所以收敛会慢

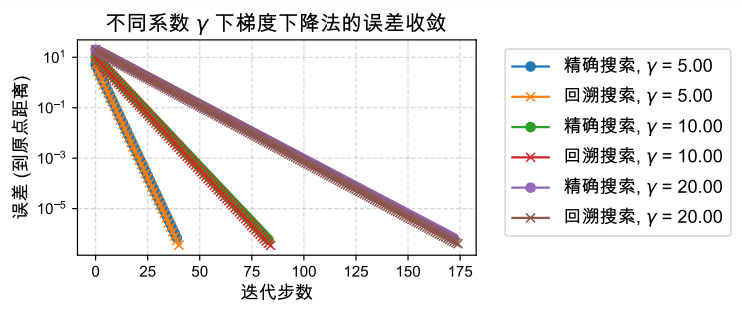
实验 3：问题本身的影响（γ 收缩）



图像验证：

对于初值 $\boldsymbol{x}^{(0)} = (\gamma, 1)$：当 γ 很小时，等高线会非常狭长，梯度下降容易出现来回摆动，因此收敛会变慢。

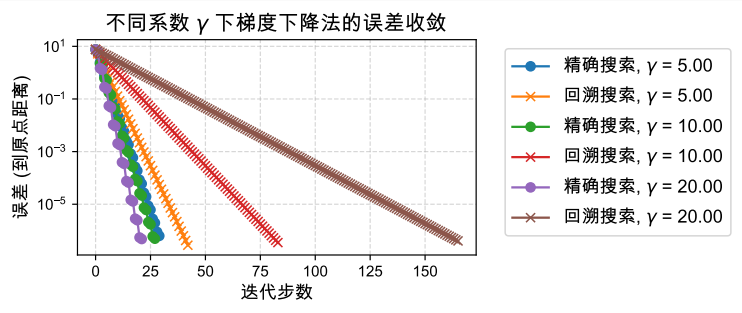
实验 3：问题本身的影响（γ 膨胀）



图像验证：

对于初值 $\boldsymbol{x}^{(0)} = (\gamma, 1)$：类似地，当 γ 很大时，等高线同样非常狭长，梯度下降也容易出现来回摆动，因此问题同样会变难。

实验 4：初值对于问题的影响



观察： 虽然“$\gamma \to \infty$ 或 $\gamma \to 0$ 时收敛变难”往往是规律，但这主要是针对**最差情况**。如果初值足够特殊（恰巧对应理想的下降方向），实际的迭代表现可能挺好。

实验现象总结

- 梯度法常呈现线性收敛：误差大致按固定比例衰减

$$\log e_N \approx -aN + b$$

- 回溯直线搜索中的 α, β 会影响效率，但通常不改变主要结论；很多时候它的效果接近精确直线搜索，而计算代价更低

- 问题本身的性质对收敛速度影响很大。下一部分我们会用条件数来量化这一点

第 (2) 部分的任务：解释这些实验现象背后的理论原因。

目录

最速下降法

最速下降法： 在“单位长度”的方向里，选一个让函数下降最快的方向

$$\Delta \boldsymbol{x} = \arg\min \left\{ \nabla f(\boldsymbol{x})^\top \boldsymbol{v} \mid \|\boldsymbol{v}\| = 1 \right\}$$

- 若范数为 L^2，则 $\Delta \boldsymbol{x} = -\frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|_2}$，即“单位负梯度方向”
- 若范数为 L^1，则 $\Delta \boldsymbol{x} = -\operatorname{sign}\left(\frac{\partial f(\boldsymbol{x})}{\partial x_i}\right)\boldsymbol{e}_i$，其中 i 是梯度绝对值最大的坐标

  例子： 若 $\nabla f(\boldsymbol{x}) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$，则 $\Delta \boldsymbol{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

  这说明：在 L^1 意义下，最速下降法每次只沿一个坐标方向走，因此它也常被称为**坐标下降法**

二次范数下的最速下降

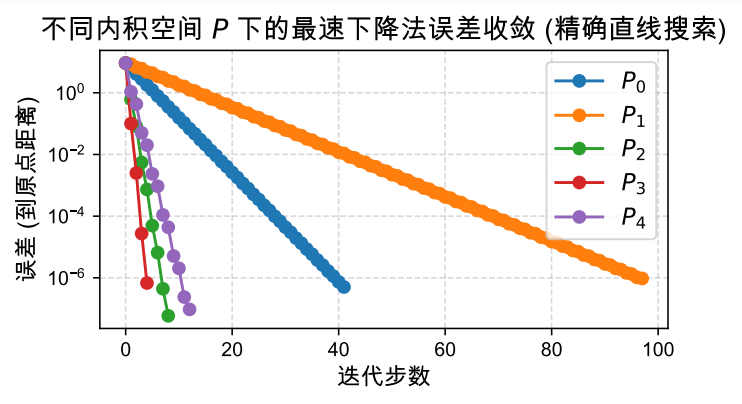
- **(二次 P-范数)** 若 $\|\boldsymbol{v}\|_{\boldsymbol{P}} := \sqrt{\boldsymbol{v}^\top \boldsymbol{P}\boldsymbol{v}} = \left\|\boldsymbol{P}^{1/2}\boldsymbol{v}\right\|_2$，其中 $\boldsymbol{P}$ 是正定矩阵，则

$$\Delta \boldsymbol{x} \propto -\boldsymbol{P}^{-1}\nabla f(\boldsymbol{x})$$

  几何含义： 选不同的 $\boldsymbol{P}$，相当于换一种“衡量一步长短”的方式。

- 如果某个 $\boldsymbol{P}$ 能把狭长的等高线“拉圆一些”，那么下降轨迹通常会更直接、更少摆动
- 因此，$\boldsymbol{P}$ 的选择本质上是在做**坐标缩放**

不同最速下降法的选择



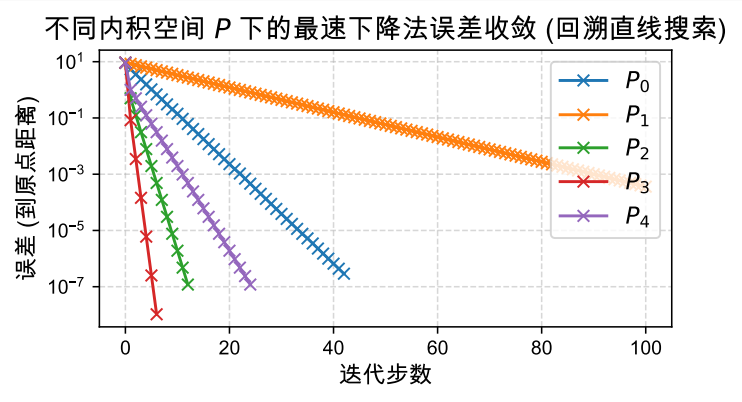
选择不同的

$$\boldsymbol{P}_k = \begin{bmatrix} 1 & 0 \\ 0 & \theta_k \end{bmatrix}$$

$\theta_0 = 1,\ \theta_1 = 1/4,$
$\theta_2 = 4,\ \theta_3 = 4.8,$
$\theta_4 = 10,$ 且 $\gamma = 5$。

观察: 合适的 $\boldsymbol{P}$ 可以明显改善下降轨迹。

最速下降法：实验观察



结论：

实现坐标变换的矩阵 $\boldsymbol{P}$ 的选择很关键。不同的度量方式会直接导致收敛路径的巨大差异。

目录

总结

本节的 4 个结论：

- 下降算法的核心就是两件事：**选方向** 和 **定步长**
- 常见停止准则是 $\|\nabla f(\boldsymbol{x})\| \leq \eta$，因为最优点附近梯度往往接近 0
- 梯度下降法最自然，因为负梯度总是一个下降方向；步长可用精确直线搜索或回溯直线搜索
- 不同的度量下，最速下降法的具体表达式不同
- 收敛速度不仅和算法参数有关，也强烈依赖问题本身的几何性质

阅读作业 & 参考资料：

- 课本第 9.1 - 9.4 章