Fundamentals of Optimization

20 优化入门

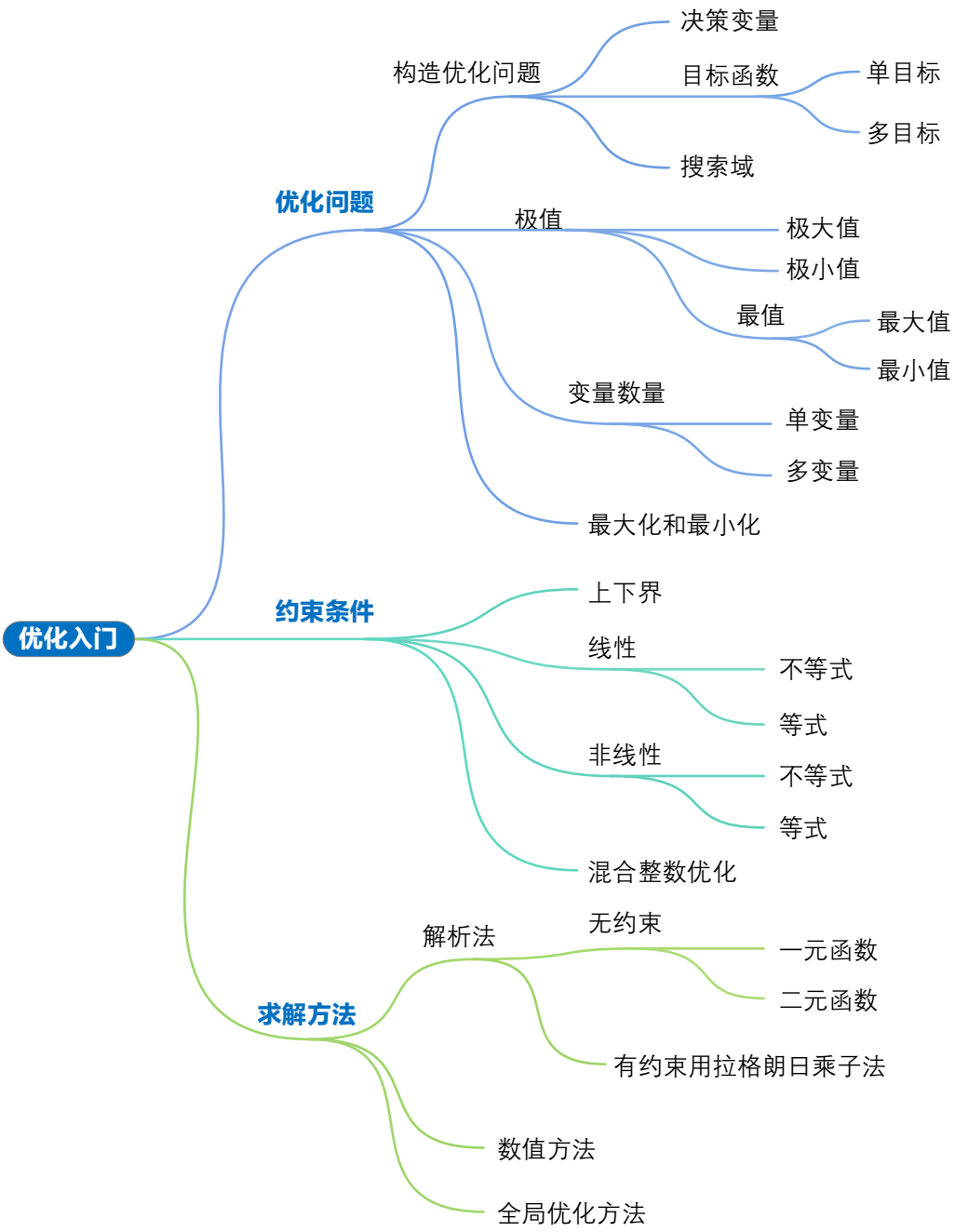
在一定区域内，寻找山峰、山谷

宇宙结构是完美的，是造物主的杰作；因此，宇宙中最大化、最小化准则无处不在。

For since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear.

—— 莱昂哈德·欧拉 (Leonhard Euler) | 瑞士数学家、物理学家 | 1707 ~ 1783

- `scipy.optimize.Bounds()` 定义优化问题中的上下约束
- `scipy.optimize.LinearConstraint()` 定义线性约束条件
- `scipy.optimize.minimize()` 求解最小化优化问题
- `sympy.abc import x` 定义符号变量 x
- `sympy.diff()` 求解符号导数和偏导解析式
- `sympy.Eq()` 定义符号等式
- `sympy.evalf()` 将符号解析式中未知量替换为具体数值
- `sympy.symbols()` 定义符号变量



20.1 优化问题：寻找山峰、山谷

数据科学、机器学习离不开求解优化问题 (optimization problem)；毫不夸张地说，机器学习中所有算法最终都变成求解优化问题。

本书前文聊过一些有关优化问题的概念，比如最大值、最小值、极大值和极小值等概念。有了微积分这个数学工具，我们可以更加深入系统地探讨优化问题这个话题。

简单地说，优化问题是在约束条件下改变变量 (variable)，用某种数学方法，寻找特定目标的最优解 (optimized solution 或 optimal solution 或 optimum)。

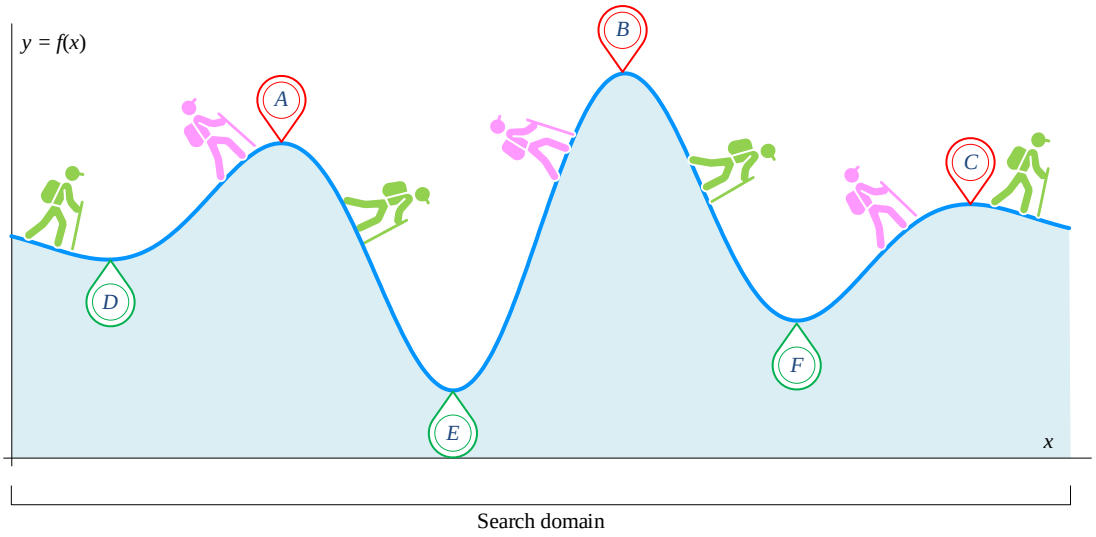


图 1. 爬上寻找山谷和山峰

更通俗地讲，优化问题好比在一定区域范围内，徒步寻找最低的山谷或最高的山峰，如图 1。这个问题的优化变量 (optimization variable) 是登山者在水平方向位置坐标值 x；优化目标 (optimization objective) 是搜索域 (search domain) 内海拔值 y。山谷对应极小值 (minima 或 local minima 或 relative minima)，山峰对应极大值 (maxima 或 local maxima 或 relative maxima)。

从一元函数角度讲，函数 $f(x)$ 在 $x = a$ 点的某个邻域内有定义，对于 a 的去心邻域内任一 x 满足：

$$f(a) < f(x) \tag{1}$$

就称 $f(a)$ 是函数的极小值；也可以说，$f(x)$ 在 a 点处取得极小值，$x = a$ 是函数 $f(x)$ 的极值点。

相反，如果 a 的去心邻域内任一 x 满足：

$$f(a) > f(x) \tag{2}$$

$f(a)$ 是函数的极大值；即 $f(x)$ 在 a 点处取得极大值，同样 $x = a$ 也是函数 $f(x)$ 的极值点。

极值 (extrema 或 local extrema) 是极大值和极小值的统称；白话讲，极值是搜索区域内所有的山峰和山谷，图 1 中 A、B、C、D、E 和 F 这六个点横坐标 x 值对应极值点。

想象一下，爬图 1 这座山的时候，当你爬到某个山峰最顶端时，朝着任何方向迈出一步，对应都是下山，意味着海拔 y 降低；而当你来到一个山谷最低端时，向左或向右迈一步都是会上山，对应海拔 y 抬升。这看似生活常识的认知，实际上是很多优化方法的核心思想。

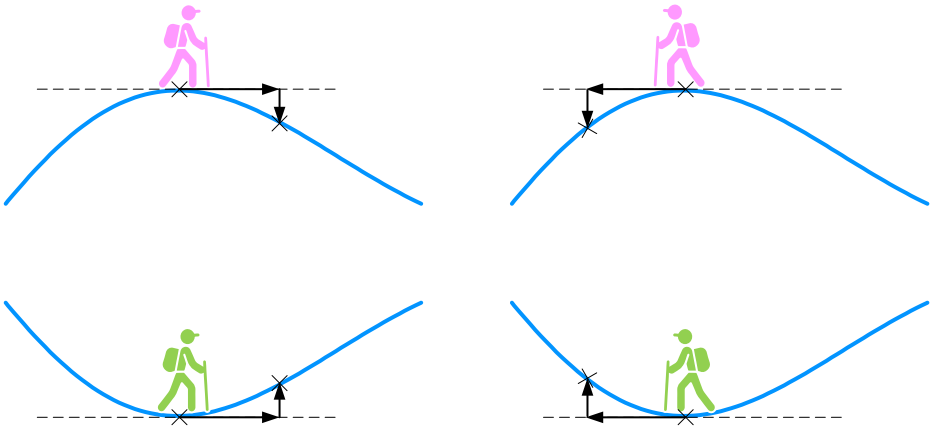


图 2. 上山和下山

如果某个极值是整个指定搜索区域内的极大值或极小值，这个极值又被称作是最大值 (maximum 或 global maximum) 或者最小值 (minimum 或 global minimum)。最大值和最小值统称最值 (global extrema)。

图 1 搜索域内有三座山峰 (A、B 和 C)，即搜索域极大值。而 B 是最高的山峰，因此 B 叫全局最大值 (global maximum)，简称最大值，即站在 B 点一览众山小；A 和 C 是区域极大值 (local maximum)。

从一元函数角度讲，函数 $f(x)$ 在整个搜索域内有定义，对于搜索域内任一 x 满足：

$$f(a) < f(x) \tag{3}$$

就称 $f(a)$ 是函数的最小值，$x = a$ 是函数 $f(x)$ 的全局最优解。

如图 1 所示，搜索域内有三个山谷，D、E 和 F，即极小值。其中，E 是全局极小值 (global minimum)，也叫最小值；D 和 F 是局域极小值 (lobal minimum)。

爬山寻找最高山峰，是最大化优化过程；而寻找最深山谷，便是最小化优化过程。这个寻找方法对应各种优化算法，这是本系列丛书要逐步介绍的内容。

20.2 构造优化问题

最小化优化问题

最小化优化问题可以写成：

$$\arg\min_{x} f(\boldsymbol{x}) \tag{4}$$

arg min 的含义是 argument of the minima；$f(\boldsymbol{x})$ 为目标函数 (objective function)，可以是个函数，也可以是个无法用解析式表达的模型。

比如，迷宫中从 A 点到 B 点，小车可以走走停停，行驶时速度一定，小车对路线记录为短期记忆。目标是不断优化小车自主寻找出口的算法，以使得小车走出迷宫用时最短；走出迷宫的用时就是这个优化问题的目标函数，函数本身显然不能写成一个简单的函数。

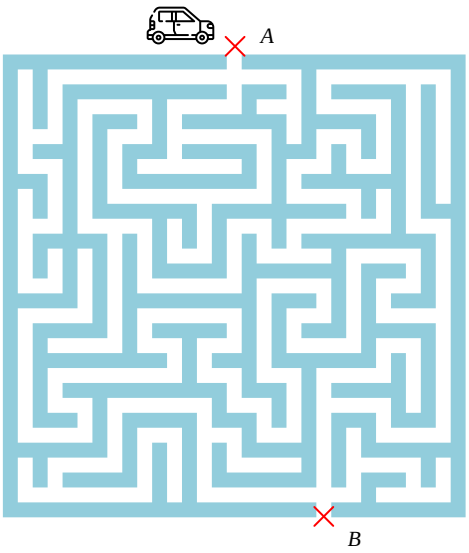


图 3. 小车自主寻找最佳路径

优化变量

$\boldsymbol{x}$ 为优化变量，也叫决策变量；优化变量可以是一个未知量 x，优化变量也可以多个未知量构成的列向量 $\boldsymbol{x} = [x_1, x_2, .., x_D]^T$。

注意，优化变量采用的符号未必都是 x。举个例子，本书前文介绍过一元线性回归。如图 4 所示，蓝色点为样本数据点，找到一条斜线能够很好地描述数据 x 和 y 的关系。如果将斜线写成 $y = ax + b$，显然 a 和 b 就是这个优化问题的变量；如果用 $y = b_1x + b_0$ 这个形式的解析式，b_1 和 b_0 则

是优化问题变量；大家以后肯定会见到很多文献 $y = \theta_1 x + \theta_0$ 这种形式作为线性回归模型，优化问题的变量则变为 θ_1 和 θ_0。

大家可能会问，图 4 这个一元线性回归优化问题的目标函数是什么呢？卖个关子，这个问题答案留到本书文末的“鸡兔同笼三部曲”中回答。

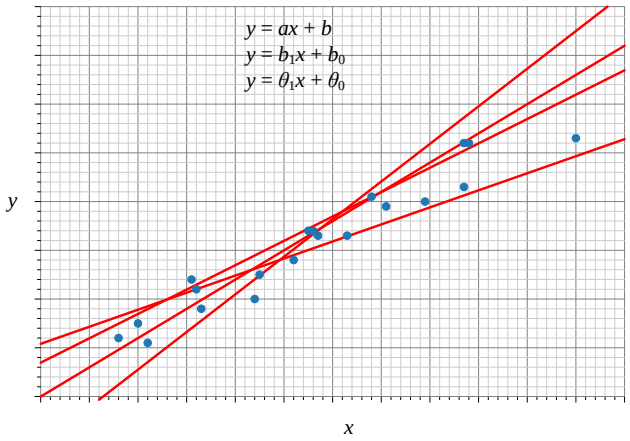


图 4. 一元最小二乘线性回归中的优化变量

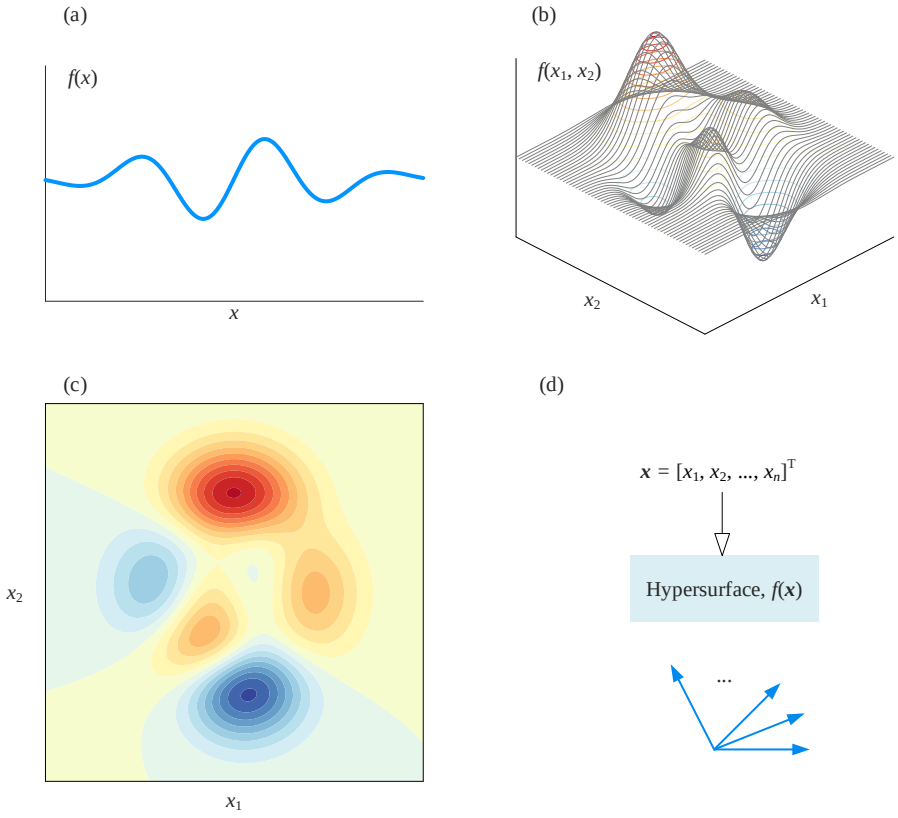


图 5. 目标函数随变量数量变化

如果优化问题的变量只有一个，这类优化叫做单变量优化 (single-variable optimization)；如果优化问题有多个变量，优化问题叫多变量优化 (multi-variable optimization)。

当只有一个优化变量时候，并且目标函数可以写成一元函数 $f(x)$，$f(x)$ 和变量 x 的关系可以在平面上表达，如图 5 (a) 所示。

有两个优化变量的情况，比如 x_1 和 x_2，目标函数为二元函数 $f(x_1, x_2)$、x_1 和 x_2 的关系可以利用三维等高线表达，如图 5 (b)所示。平面等高线也可以展示两个优化变量优化问题，如图 5 (c) 所示。当优化变量数量不断增多，优化变量要写成列向量 $\boldsymbol{x} = [x_1, x_2, ..., x_D]^T$，$f(\boldsymbol{x})$ 在向量 $\boldsymbol{x}$ 构成的多维空间中形成一个超曲面 (hypersurface)。

此外，优化目标可以有一个或多个。具有不止一个目标函数优化问题叫做多目标优化 (multi-objective optimization)。举个例子，本系列丛书将会专门介绍多目标优化。

最大化优化问题

最大化优化问题可以写成：

$$\arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{5}$$

实际上，标准优化问题一般都是最小化优化问题；而最大值问题目标函数改变符号 (乘−1)，就转化为最小值问题：

$$\arg\max_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \Leftrightarrow \quad \arg\min_{\boldsymbol{x}} -f(\boldsymbol{x}) \tag{6}$$

图 6 所示为一个最大值优化问题转化为最小值优化问题。

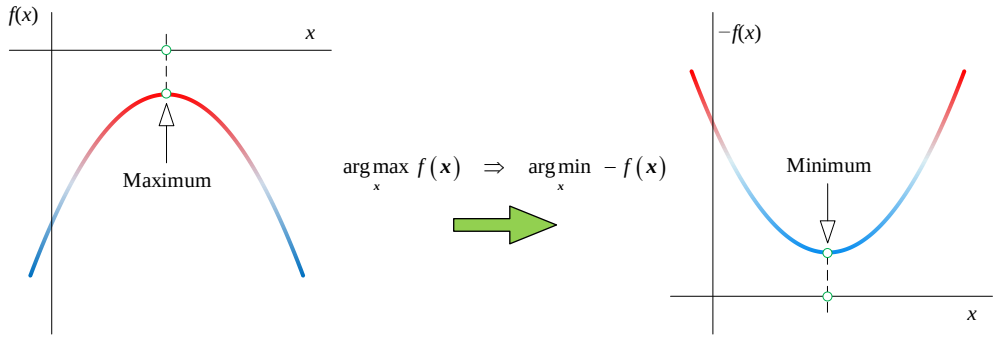


图 6. 优化问题求解最大值和最小值转换

20.3 约束条件：限定搜索区域

优化变量值选取并非随心所欲，必须在一定范围之内。变量的范围叫做定义域 (domain)，也叫做搜索空间 (search space)、选择集 (choice set)；范围内的每一个点为一个潜在解 (candidate solution, feasible solution)。

构建优化变量范围的条件被称作约束条件 (constraints)。根据约束条件的有无，优化问题分为：

- 无约束优化问题 (unconstrained optimization)
- 受约束优化问题 (constrained optimization)

五类约束

多数优化问题都是受约束优化问题，常见的约束条件分为以下几种。

- 上下界 (lower and upper bounds)，$\boldsymbol{x}_{\min} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\max}$
- 线性不等式 (linear inequalities)，$g(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leqslant \boldsymbol{0}$
- 线性等式 (linear equalities)，$h(\boldsymbol{x}) = \boldsymbol{A}_{eq}\boldsymbol{x} - \boldsymbol{b}_{eq} = \boldsymbol{0}$
- 非线性不等式 (nonlinear inequalities)，$c(\boldsymbol{x}) \leqslant 0$
- 非线性等式 (nonlinear equalities)，$c_{eq}(\boldsymbol{x}) = 0$

几种约束条件复合在一起构成优化问题约束。大家应该已经发现，这五类约束条件对应的就是本书前文介绍的不等式。

最小化优化问题

最小化优化结合约束条件，构造完整优化问题。

$$\begin{aligned} &\arg\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \\ &\qquad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ &\qquad \boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\ &\qquad c(\boldsymbol{x}) \leq 0, \ c_{eq}(\boldsymbol{x}) = 0 \end{aligned} \tag{7}$$

其中，subject to 代表“受限于”、“约束于”，常简写成 s.t.；下面，我们一一介绍各种约束条件。

上下界约束

首先讨论上下界约束，用矩阵形式表达。

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \tag{8}$$

其中，列向量 $\boldsymbol{l}$ 为下界 (lower bound 常简写为 lb)，列向量 $\boldsymbol{u}$ 为上界 (upper bound 常简写为 ub)。

图 7 给出的是变量 x_1 取值范围为 $x_1 \geqslant 1$ 对应区间为 $[1, +\infty)$；有了这个搜索范围，我们发现二元函数 $f(x_1, x_2)$ 的最高山峰和最低山谷都被排除在外。Python 中，`float('inf')` 可以表达正无

穷，`float('-inf')` 表达负无穷；优化问题中常用 `numpy.inf` 生成正无穷，`-numpy.inf` 生成负无穷。

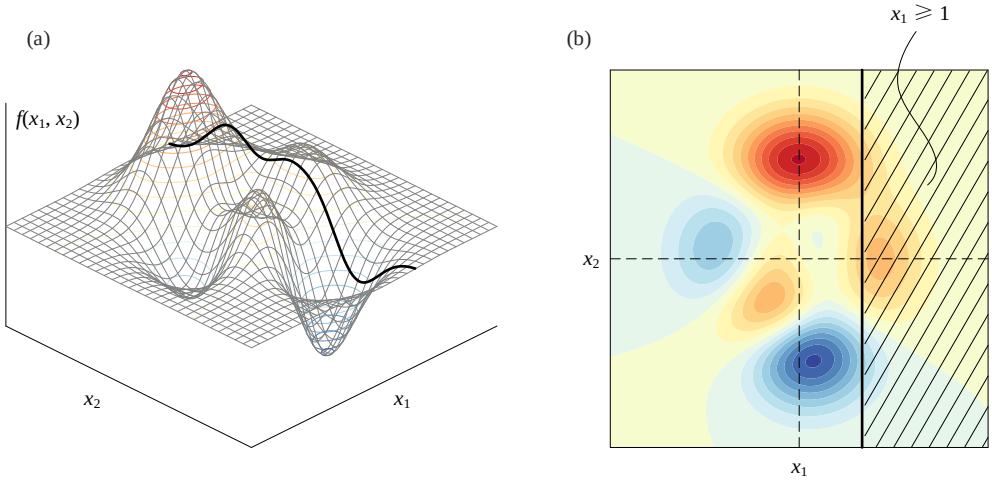


图 7. x_1 的下界为 1，上界为正无穷

图 8 中，变量 x_1 取值范围为 $-2 \leqslant x_1 \leqslant 1$，对应区间为闭区间 $[-2, 1]$；在求解优化问题时，一般的优化器都默认区间为闭区间，比如 $[a, b]$，也就是寻找优化解时，搜索范围包含区间 a 和 b 两端。举个例子，如果一定要将 b 这个端点排除在搜索范围之外，可以用 $b - \varepsilon$ 代替 b，ε 是个极小的正数，比如 $\varepsilon = 10^{-5}$。

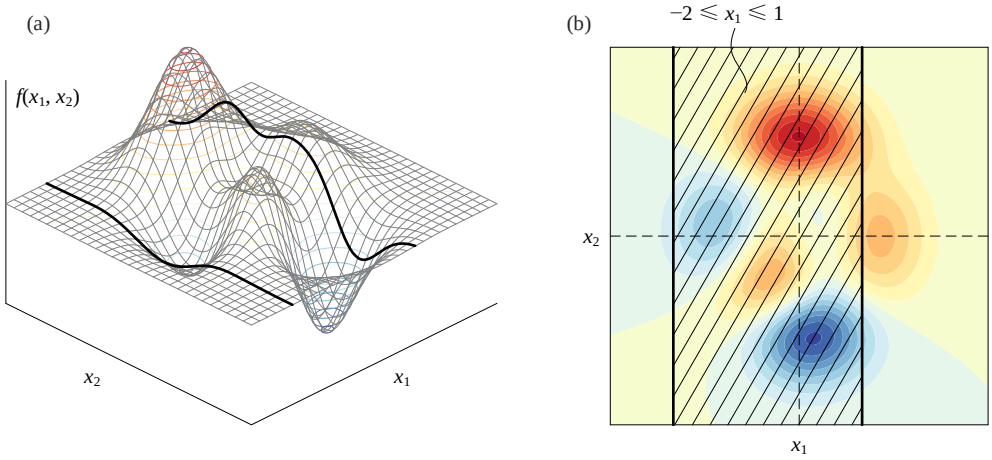


图 8. x_1 的下界为−2，上界为 1

图 9 所示的搜索范围对应，$-1 \leqslant x_2 \leqslant 1$，对应区间为闭区间 $[-1, 1]$。图 10 所示的搜索区域同时满足$-2 \leqslant x_1 \leqslant 1$ 和$-1 \leqslant x_2 \leqslant 1$，写成 (8) 形式为。

$$\underbrace{\begin{bmatrix} -2 \\ -1 \end{bmatrix}}_{\boldsymbol{l}} \leq \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\boldsymbol{u}} \tag{9}$$

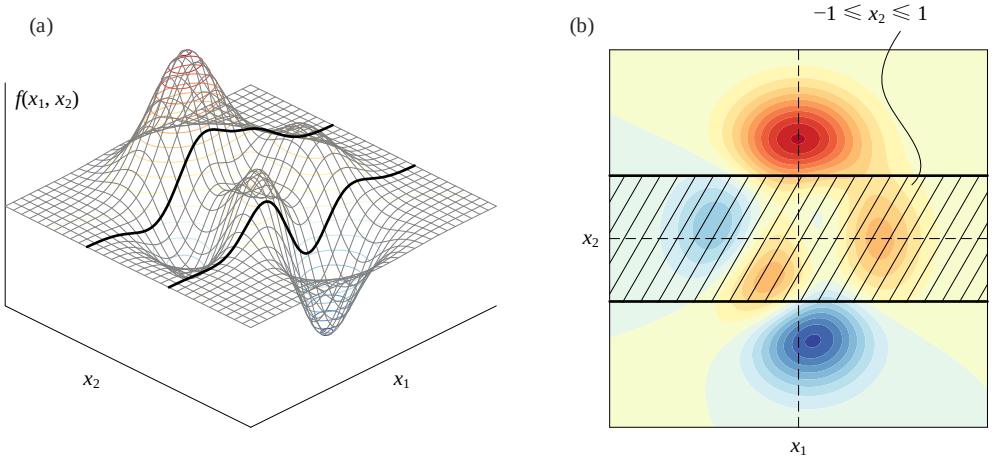


图 9. x_2的下界为−1，上界为 1

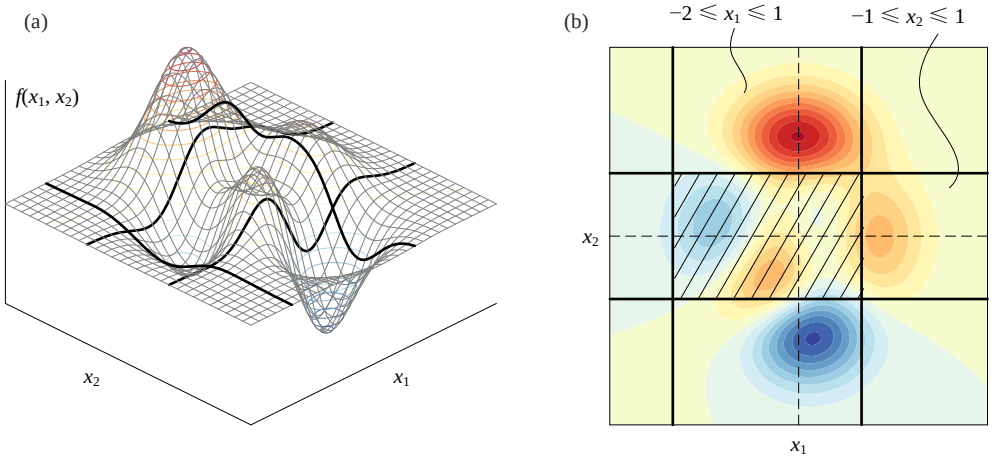


图 10. 同时满足$-2 \leqslant x_1 \leqslant 1$和$-1 \leqslant x_2 \leqslant 1$的搜索区域

线性不等式约束

线性不等式约束表达为。

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \tag{10}$$

也常记做。

$$g(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \boldsymbol{0} \tag{11}$$

图 11 所示的搜索区域为线性不等式约束，满足$x_1 + x_2 - 1 \geqslant 0$；注意，优化问题中约束条件多用“小于等于”，即$-x_1 - x_2 + 1 \leqslant 0$。

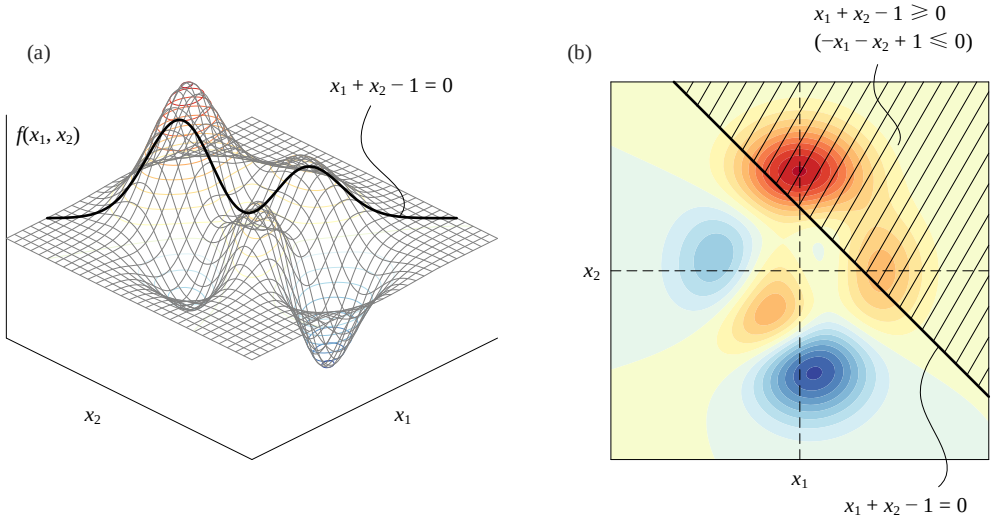


图 11. $x_1 + x_2 - 1 \geqslant 0$ 对应的搜索区域

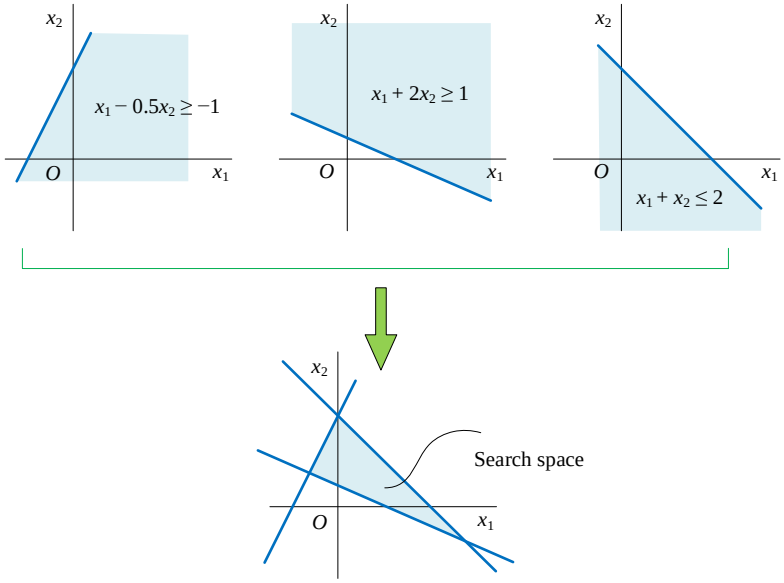


图 12. 三个线性不等式约束构造的搜索区域

下例为三个线性不等式构造的约束条件。

$$\begin{cases} x_1 - 0.5x_2 \geq -1 \\ x_1 + 2x_2 \geq 1 \\ x_1 + x_2 \leq 2 \end{cases} \tag{12}$$

首先将三个不等式所有大于等于号 (≥) 调整为小于等于号 (≤)，得到。

$$\begin{cases} -x_1 + 0.5x_2 \le 1 \\ -x_1 - 2x_2 \le -1 \\ x_1 + x_2 \le 2 \end{cases} \tag{13}$$

然后用矩阵形式描述这一组约束条件。

$$\underbrace{\begin{bmatrix} -1 & 0.5 \\ -1 & -2 \\ 1 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{x} \le \underbrace{\begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}}_{b} \tag{14}$$

这三个线性不等式约束联立在一起便构成图 12 所示搜索空间。

线性等式约束

线性等式约束表达如下。

$$\boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \tag{15}$$

上式常记做 $h(\boldsymbol{x}) = \boldsymbol{A}_{eq}\boldsymbol{x} - \boldsymbol{b}_{eq} = \boldsymbol{0}$；线性等式约束很好理解，线性约束条件对应的搜索范围为一条直线，比如图 11 中黑色线代表线性约束条件 $x_1 + x_2 - 1 = 0$。也就是说，有了这个线性等式约束，我们只能在图 11 黑色曲线上寻找二元函数 $f(x_1, x_2)$ 的最大值或最小值。

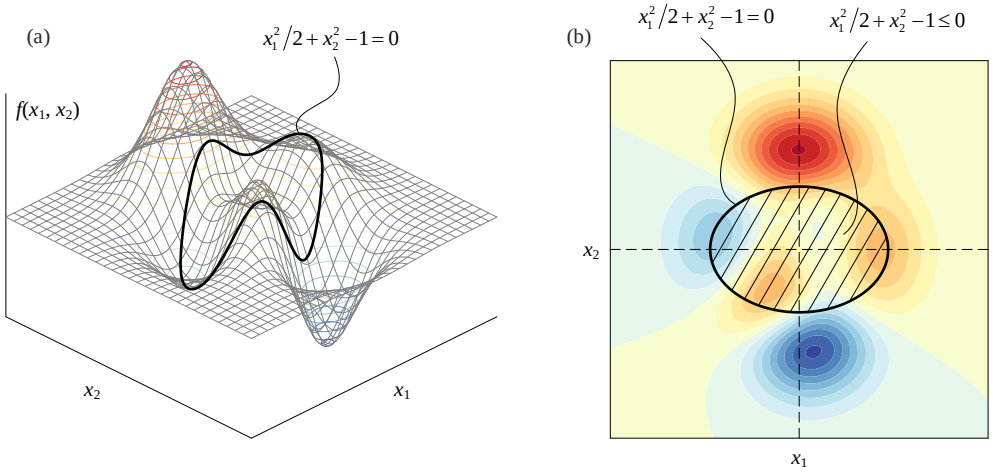


图 13. 非线性不等式约束条件

非线性不等式约束

非线性不等式约束用下式表达。

$$c(\boldsymbol{x}) \le 0 \tag{16}$$

举个例子，图 13 中阴影部分搜索区域对应如下非线性不等式约束条件。

$$\frac{x_1^2}{2}+x_2^2-1\le 0 \tag{17}$$

Python 中，可以同时定义非线性不等式约束的上下界，即。

$$l\le c(\boldsymbol{x})\le u \tag{18}$$

非线性不等式一般通过构造自定义函数完成。

非线性等式约束通过下式表达：

$$c_{\text{eq}}(\boldsymbol{x})=0 \tag{19}$$

非线性等式约束很容易理解；以图 13 为例，满足 $x_1^2/2+x_2^2-1\le 0$ 的搜索区域限定在椭圆上。

极值出现的位置

当约束条件出现时，有些极大值或极小值可能出现在约束条件限定的边界上。如图 14 (a) 所示，给定搜索区域，函数的极大值和极小值均在搜索区域内部；而图 14 (b) 中，$f(x)$ 极小值出现在约束条件右侧边界上；图 14 (c) 中，$f(x)$ 极大值出现在约束条件左侧边界上。

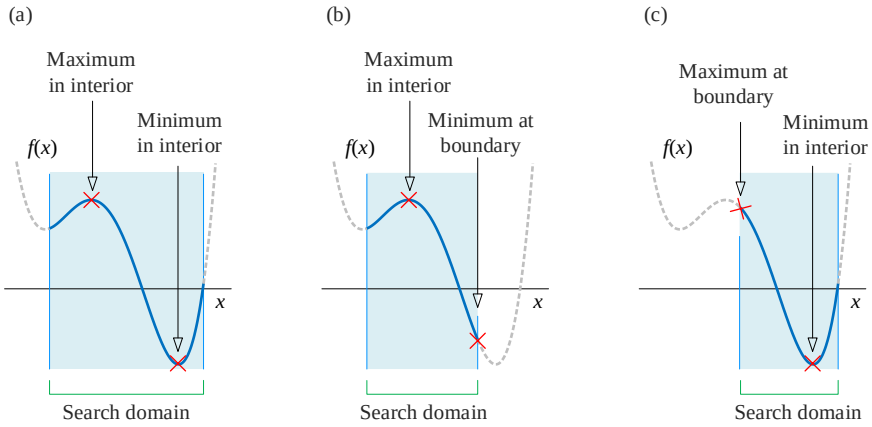


图 14. 极值和约束关系

混合整数优化

很多优化问题要求变量全部为整数，或者部分为整数。混合整数优化 (mixed integer optimization) 可以同时包含整数和连续变量。图 15 所示为在 x_1x_2 平面上三种约束情况：x_1 为整数，x_2 为整数，以及 x_1 和 x_2 均为整数。

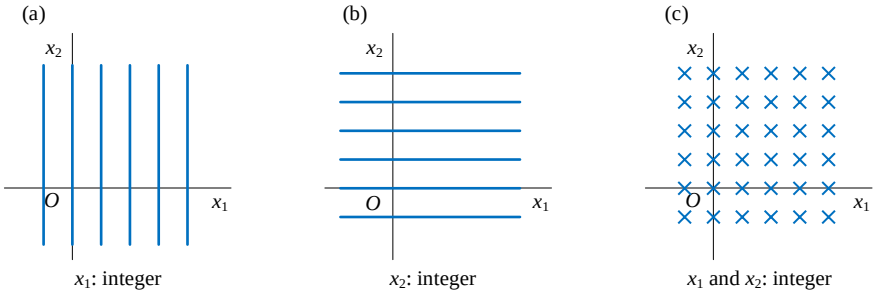


图 15. 混合整数优化

20.4 一元函数极值点判定

本节从最简单的一次函数入手，介绍如何判定极值点。

观察图 16 所示函数，函数为连续函数，没有断点。同样把图 16 看做一座山，不难发现，某个山峰 (极大值) 紧邻的左侧是上坡 (区域递增函数)，而紧邻的右侧是下坡 (区域递降函数)，如图 16 所示。

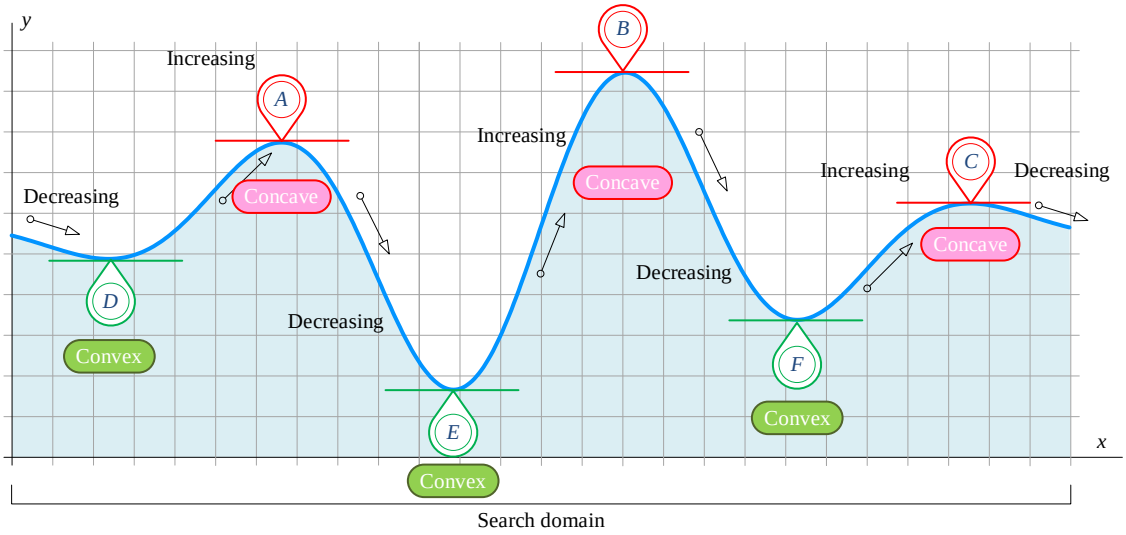


图 16. 极值两侧的增减性质

而某个山谷 (极小值) 则恰好相反，山谷紧邻的左侧是下坡 (区域递减函数)，而紧邻的右侧是上坡 (区域递增函数)。

一阶导数

本书前文在讲解导数时说过，一元函数曲线上某点的导数，就是该点曲线切线斜率；也就是说，我们可以通过导数的正负判断函数极值。如果一元函数处处可导，如图 16 所示，不管站在山谷点、还是山顶点，切线则为水平，即导数为 0。这样，我们便得到一元可导函数极值的一个必要条件。

如果函数 $f(x)$ 在 $x = a$ 处可导，且在该点取得极值，则一阶导数 $f'(a) = 0$。

二阶导数

观察山谷 (极小值点)，D、E 和 F，可以发现这三点区域函数都是局部为凸 (convex)；而山顶，A、B 和 C 所在的局域函数都是局部为凹 (极大值点)。

这样，我们便可以通过二阶导数的正负来进一步判断极值点是极大值还是极小值。再次注意，本书采用的凸凹定义和国内一些教材正好相反。

本书前文介绍，对于一元函数 $f(x)$，函数一阶导数为 0 的点叫驻点 (stationary point)；而驻点可能是一元函数的极大值、极小值，或者是鞍点。

函数 $f(x)$ 在 $x = a$ 处有二阶导数，且一阶导数 $f'(a) = 0$，即 $x = a$ 为驻点；如果二阶导数 $f''(a) > 0$，函数 $f(x)$ 在 $x = a$ 取得极小值；如果二阶导数 $f''(a) < 0$，函数 $f(x)$ 在 $x = a$ 取得极大值。

图 17 所示为最大值和最小值点处函数值、一阶导数和二阶导数变化的细节图。原函数一阶导数的一阶导数是原函数的二阶导数。位于极大值点附近，当 x 增大，二阶导数值需要为负值才能保证一阶导数从正值、穿越 0 点、到负值；而极小值点附近，二阶导数数值需要为正，这样 x 增大一阶导数从负值穿越 0 点，到正值。

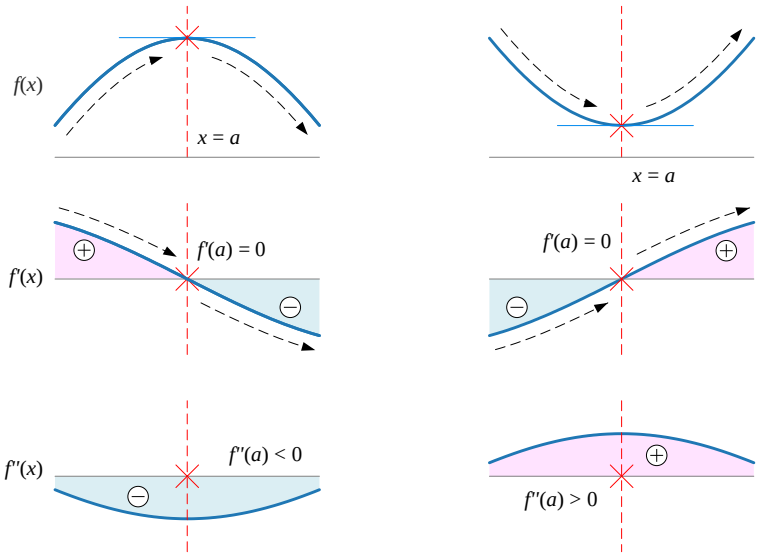


图 17. 极大值和极小值点局部，用正弦余弦重画图

如果二阶导数也为 0，可以用驻点左右一阶导数符号判定。图 18 分别给出三个函数、一阶导数、二阶导数图像。

对于 $f(x) = x^2$，$x = 0$ 处，一阶导数为 0，而且二阶导数为正，容易判断 $x = 0$ 对应函数极小值点；通过进一步判断，函数不存在其他极小值点，因此这个极小值点也是函数的最小值点。

而 $f(x) = x^3$，$x = 0$ 处函数一阶导数和二阶导数都为 0，但是 $x = 0$ 左右一阶导数都为正，显然 $x = 0$ 为函数的鞍点，不是极值点。

对于 $f(x) = x^4$，$x = 0$ 处函数一阶导数和二阶导数虽然也都为 0，但是 $x = 0$ 左右一阶导数分别为正和负，显然 $x = 0$ 为函数的极小值点。

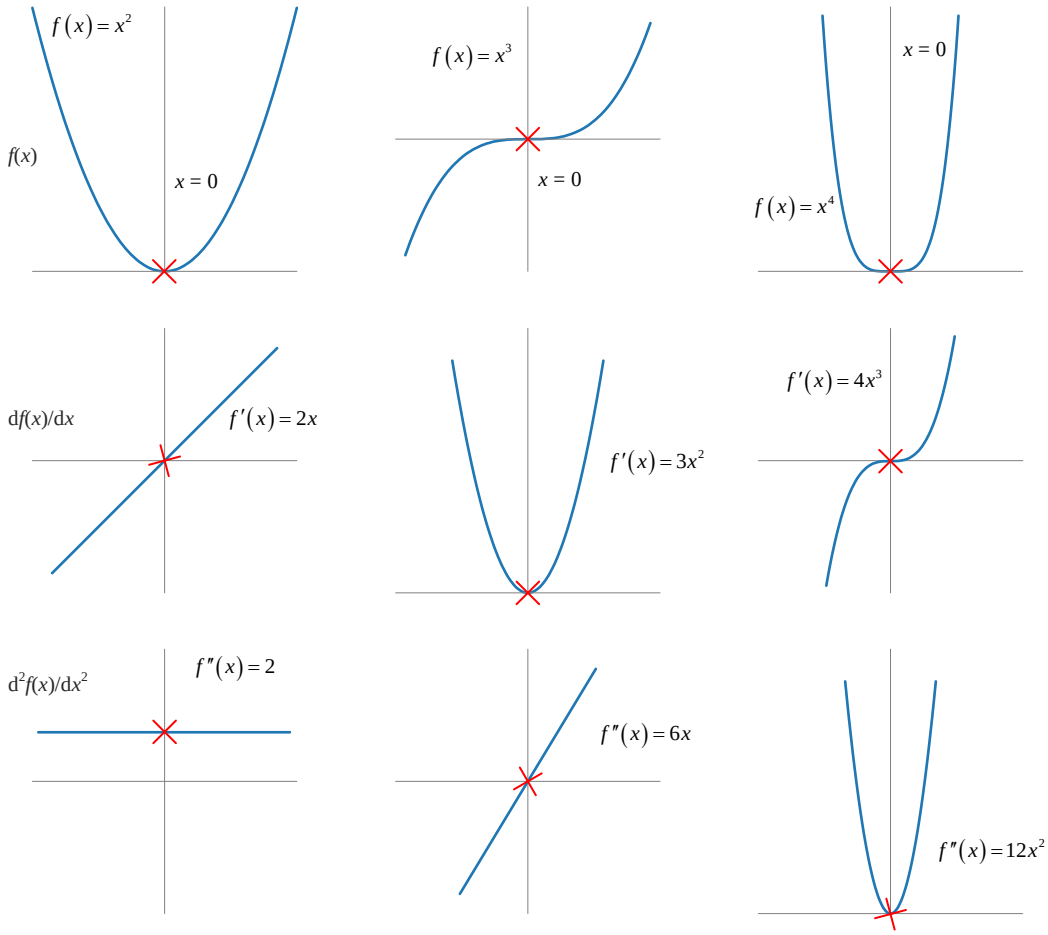


图 18. 通过一阶导数和二阶导数判断极值

寻找极值点

另外，函数在导数不存在的点也可能取得极值；再者，考虑约束条件，函数也可能在约束边界上取得极值。

总结来说，寻找极值时大家需要注意三类点：1) 驻点 (一阶导数为 0 点)，图 19 中 C 和 D 点；2) 不可导点；3) 搜索区域边界点，参考上一节图 14。

注意，本书前文提到导数不存在有又分三种情况：1) 间断点，图 19 中 A 和 B 点；2) 尖点，图 19 中 E 点；3) 切线竖直，即斜率为无穷，图 19 中 F 点。

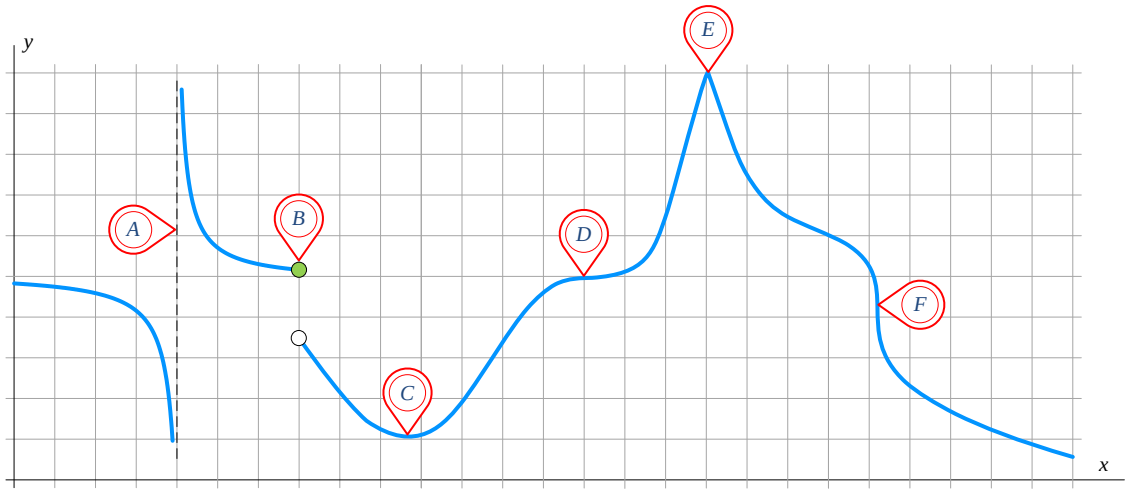


图 19. 一次函数值得关注的几个点

编程求解优化问题

本节最后举个例子，求解无约束条件下如下一元函数 $f(x)$ 的最小值以及对应的优化解。

$$\min_x f(x) = -2x \cdot \exp\left(-x^2\right) \tag{20}$$

函数的一阶导数为。

$$f'(x) = 4x^2 \cdot \exp\left(-x^2\right) - 2\exp\left(-x^2\right) \tag{21}$$

一阶导数为 0 有两个解。

$$x = \pm\frac{\sqrt{2}}{2} \tag{22}$$

请大家自行计算函数二阶导数在 $x = \pm\sqrt{2}/2$ 处具体值。

图 20 (a) 所示为 $f(x)$ 函数图像，容易判断 $x = \sqrt{2}/2$，函数取得最小值。图 20 (b) 所示为 $f(x)$ 函数一阶导数函数图像，$x = \sqrt{2}/2$ 一阶导数为 0。

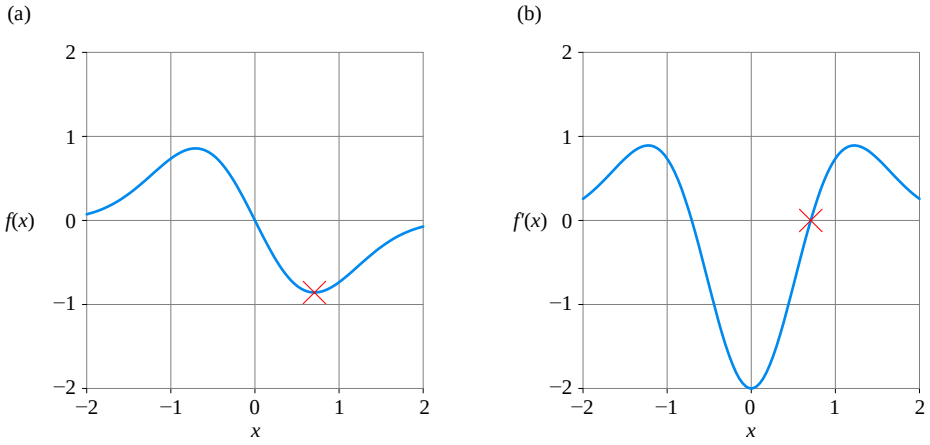


图 20. 一次函数图像和一阶导函数图像，极小值点位置

以下代码完成优化问题求解，并绘制图 20。

```
# Bk3_Ch20_01

import matplotlib.pyplot as plt
from scipy import optimize
import numpy as np
from sympy import lambdify, diff, exp
from sympy.abc import x

f_x = -2*x*exp(-x**2)
obj_f = lambdify(x,f_x)
# objective function

# def obj_f(x):
#     return -4*x*np.exp(-x**2)

result = optimize.minimize_scalar(obj_f)

print('=== Success ===')
print(result.success)

x_min = result.x

x_array = np.linspace(-2,2,100)
y_array = obj_f(x_array)

fig, ax = plt.subplots()

plt.plot(x_array,y_array, color = 'b')
# plot the optimal solution
plt.plot(x_min,obj_f(x_min), color = 'r', marker = 'x',
         markersize = 12)

plt.xlabel('x'); plt.ylabel('f(x)')
plt.xticks(np.linspace(-2, 2, 5)); plt.yticks(np.linspace(-2, 2, 5))
plt.axis('scaled'); ax.set_xlim(-2,2); ax.set_ylim(-2,2)
ax.spines['top'].set_visible(False); ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False); ax.spines['right'].set_visible(False)
ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])
```

```
#%% plot the first-order derivative

f_x_1_diff = diff(f_x,x)
f_x_1_diff_fcn = lambdify(x,f_x_1_diff)
f_x_1_diff_array = f_x_1_diff_fcn(x_array)

fig, ax = plt.subplots()

plt.plot(x_array,f_x_1_diff_array, color = 'b')
# plot the optimal solution
plt.plot(x_min,f_x_1_diff_fcn(x_min), color = 'r', marker = 'x',
         markersize = 12)

plt.xlabel('x'); plt.ylabel('f\'(x)')
plt.xticks(np.linspace(-2, 2, 5)); plt.yticks(np.linspace(-2, 2, 5))
plt.axis('scaled'); ax.set_xlim(-2,2); ax.set_ylim(-2,2)
ax.spines['top'].set_visible(False); ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False); ax.spines['right'].set_visible(False)
ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])
```

20.5 二元函数的极值点判定

二元以及多元函数的极值点判定就没有一元函数那么直接。

一阶偏导

二元函数 $y = f(x_1, x_2)$ 在点 (a, b) 存在分别对 x_1 和 x_2 存在偏导，且在 (a, b) 处有极值，则。

$$f_{x1}(a,b)=0,\quad f_{x2}(a,b)=0 \tag{23}$$

对于二元函数极值的判定，一阶偏导数 $f_{x1}(x_1, x_2) = 0$ 和 $f_{x2}(x_1, x_2) = 0$ 同时成立的点 (x_1, x_2) 为二元函数 $f(x_1, x_2)$ 的驻点；如图 21 所示，驻点可以是极小值、极大值或鞍点。

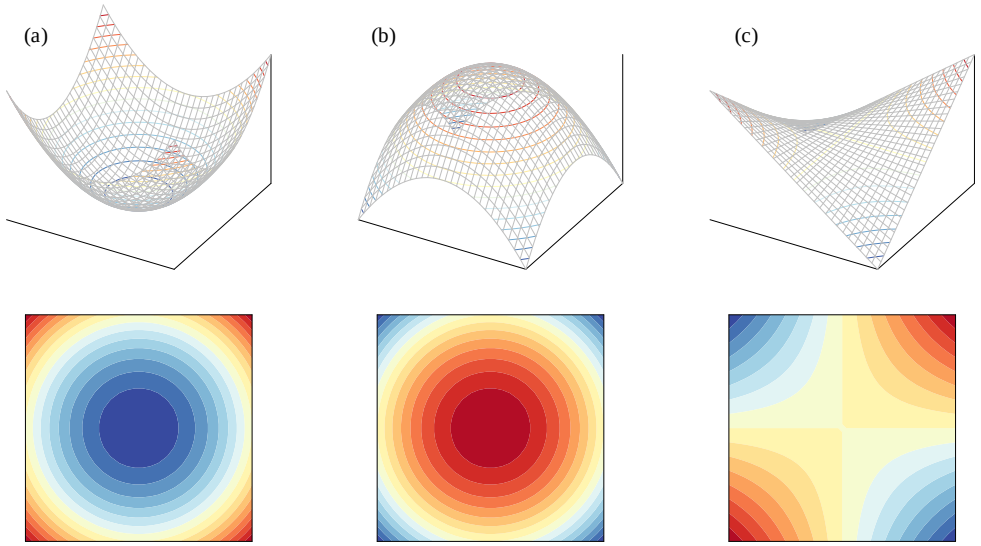


图 21. 二元函数驻点的三种情况

二阶偏导

如果 $f(x_1, x_2)$ 在 (a, b) 邻域内连续，且函数的一阶偏导及二阶偏导连续，令

$$A = f_{x1x1}(a,b), \quad B = f_{x1x2}(a,b), \quad C = f_{x2x2}(a,b) \tag{24}$$

$f(x_1, x_2)$ 在 (a, b) 一阶偏导为 0，$f_{x1}(a, b) = 0$，$f_{x2}(a, b) = 0$，$f(a, b)$ 是否为极值点可以通过如下条件判断：

a) $AC - B^2 > 0$ 存在极值，且当 $A < 0$ 有极大值，$A > 0$ 时有极小值；

b) $AC - B^2 < 0$ 没有极值；

c) $AC - B^2 = 0$，可能有极值，也可能没有极值，需要进一步讨论。

举个例子

在没有约束的条件下，确定如下二元函数的极小值点。

$$\min_{x_1,x_2} f(x_1,x_2) = -2x_1 \cdot \exp\left(-x_1^2 - x_2^2\right) \tag{25}$$

图 22 所示为 $f(x_1, x_2)$ 曲面和等高线图像，显然函数存在一个最小值点。

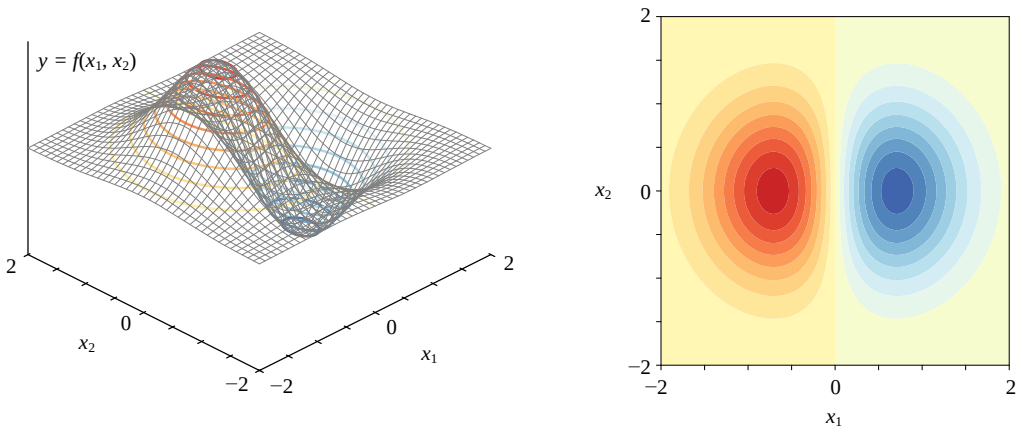


图 22. $f(x_1, x_2)$ 曲面和等高线图像

$f(x_1, x_2)$ 对于 x_1 的一阶偏导 $f_{x1}(x_1, x_2)$ 解析式。

$$f_{x1}(x_1,x_2) = \left(4x_1^2 - 2\right)\exp\left(-x_1^2 - x_2^2\right) \tag{26}$$

再次强调，如图 23 所示，一阶偏导 $f_{x1}(x_1, x_2)$ 也是一个二元函数。图 23 中墨绿色实线对应 $f_{x1}(x_1, x_2) = 0$。

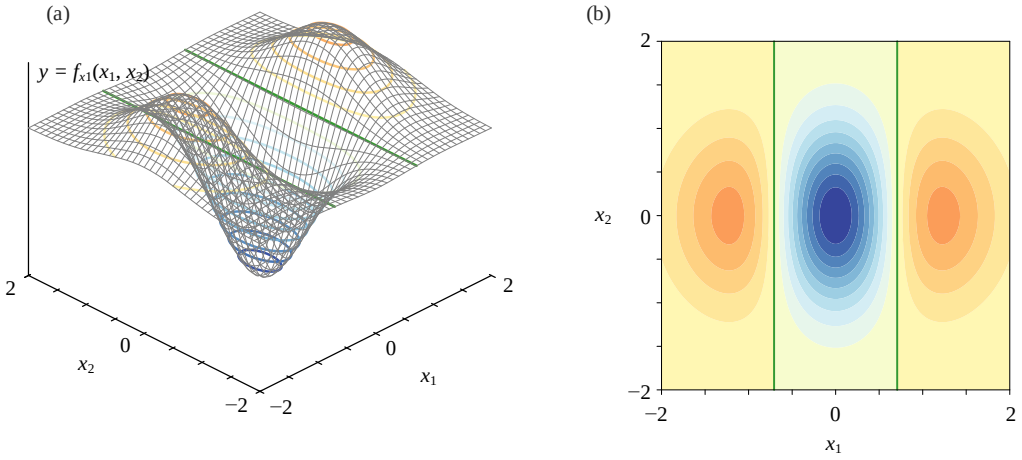


图 23. 一阶偏导 $f_{x1}(x_1, x_2)$ 曲面和等高线图像

$f(x_1, x_2)$ 对于 x_2 的一阶偏导 $f_{x2}(x_1, x_2)$ 解析式。

$$f_{x2}(x_1, x_2) = 4x_1x_2\exp\left(-x_1^2 - x_2^2\right) \tag{27}$$

图 24 所示为一阶偏导 $f_{x2}(x_1, x_2)$ 曲面和等高线图像；图 24 深蓝色曲线对应 $f_{x2}(x_1, x_2) = 0$。图 25 给出的是 $f(x_1, x_2)$ 曲面和等高线，上面墨绿色曲线对应 $f_{x1}(x_1, x_2) = 0$，深蓝色曲线对应 $f_{x2}(x_1, x_2) = 0$。

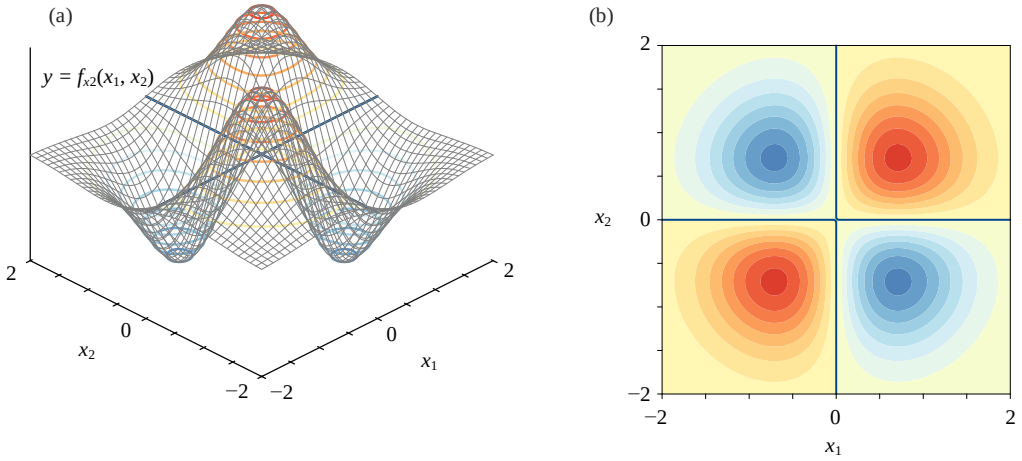


图 24. 一阶偏导 $f_{x2}(x_1, x_2)$ 曲面和等高线图像

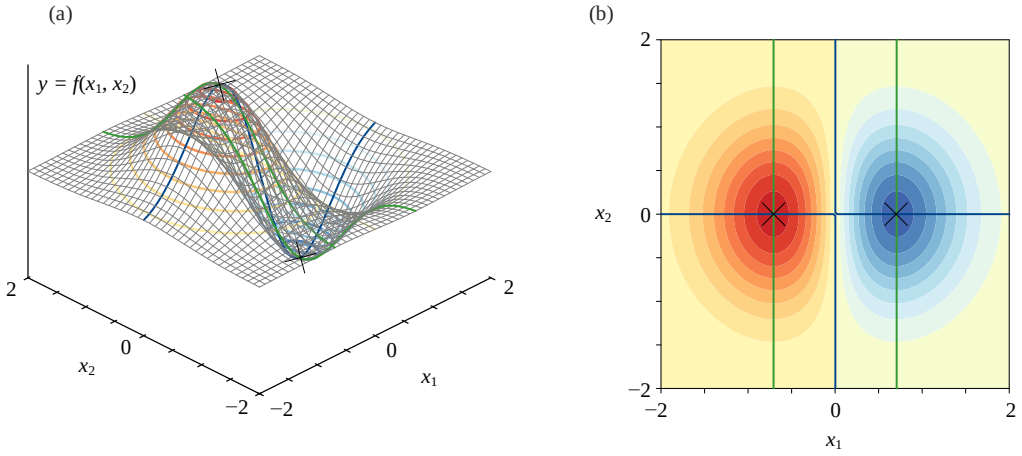


图 25. $f(x_1, x_2)$ 曲面和等高线图像，墨绿色曲线对应 $f_{x1}(x_1, x_2) = 0$，深蓝色曲线对应 $f_{x2}(x_1, x_2) = 0$

$f(x_1, x_2)$ 对 x_1 的二阶偏导。

$$A = f_{x1x1}(x_1, x_2) = 4x_1\left(3 - 2x_1^2\right)\exp\left(-x_1^2 - x_2^2\right) \tag{28}$$

$f(x_1, x_2)$ 对 x_1 和 x_2 混合二阶偏导。

$$B = f_{x1x2}(x_1, x_2) = f_{x2x1}(x_1, x_2) = 4x_2\left(1 - 2x_2^2\right)\exp\left(-x_1^2 - x_2^2\right) \tag{29}$$

$f(x_1, x_2)$ 对 x_2 的二阶偏导。

$$C = f_{x2x2}(x_1, x_2) = 4x_1\left(1 - 2x_2^2\right)\exp\left(-x_1^2 - x_2^2\right) \tag{30}$$

$AC - B^2$ 对应的解析式。

$$AC - B^2 = f_{x2x2}(x_1, x_2) = \left(-32x_1^4 - 32x_1^2x_2^2 + 48x_1^2 - 16x_2^2\right)\exp\left(-x_1^2 - x_2^2\right) \tag{31}$$

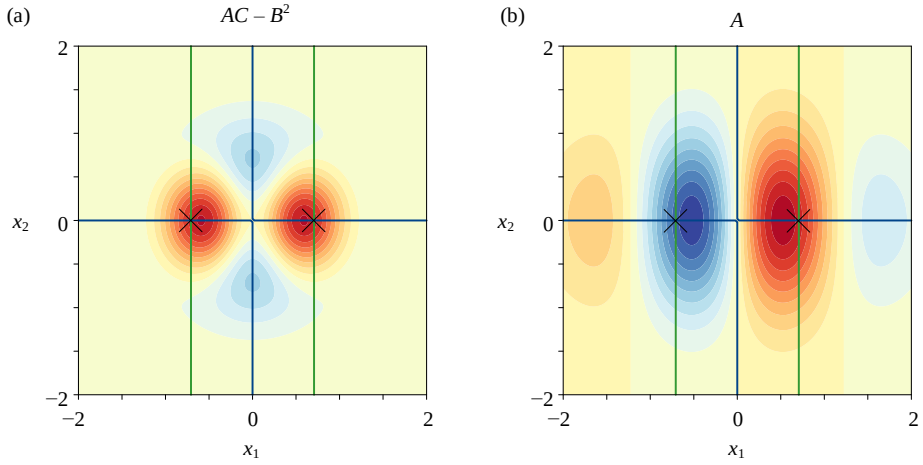


图 26. $AC - B^2$ 和 A 解析式等高线

有约束条件

在 (25) 上，加上非线性约束条件。

$$\begin{aligned} &\min_{x_1,x_2} f(x_1,x_2) = -2x_1 \cdot \exp\left(-x_1^2 - x_2^2\right) \\ &\text{subject to:} \quad |x_1| + |x_2| - 1 \le 0 \end{aligned} \tag{32}$$

图 27 (a) 所示，这个非线性约束条件对优化结果没有影响。

换一个约束条件：

$$\begin{aligned} &\min_{x_1,x_2} f(x_1,x_2) = -2x_1 \cdot \exp\left(-x_1^2 - x_2^2\right) \\ &\text{subject to:} \quad |x_1| + |x_2 + 1| - 1 \le 0 \end{aligned} \tag{33}$$

图 27 (b) 所示，优化结果出现在边界上。

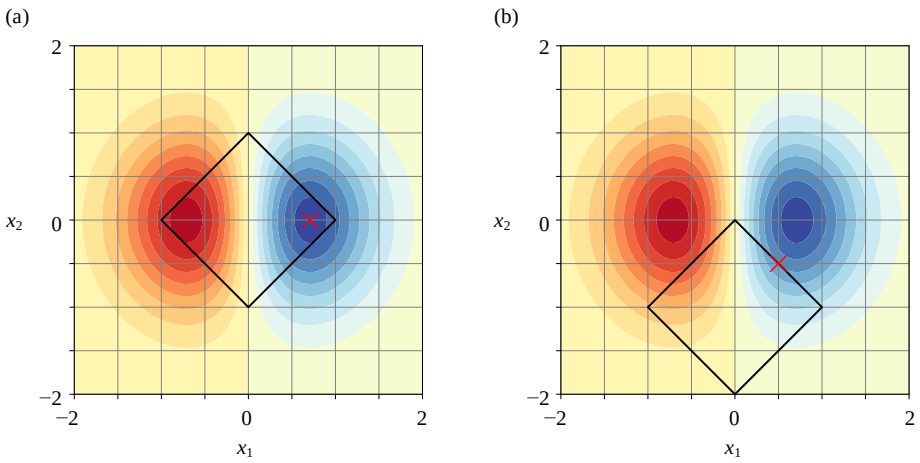
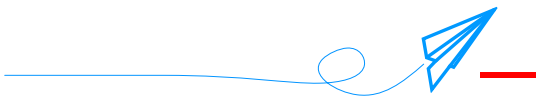


图 27. 两个非线性条件对优化结果影响

以下代码求解含有非线性约束条件的优化问题，并绘制图 27。

```
# Bk3_Ch20_02

import numpy as np
from scipy.optimize import minimize, LinearConstraint, Bounds, NonlinearConstraint
import matplotlib.pyplot as plt

def obj_f(x):
```

```
    x1 = x[0];
    x2 = x[1];

    obj = -2*x1*np.exp(-x1**2 - x2**2)
    return obj

x0 = [1,1]; # initial guess
# linear_constraint = LinearConstraint([1,1],[1],[1])

def nonlinear_c(x):

    x1 = x[0];
    x2 = x[1];

    nlc = np.abs(x1) + np.abs(x2+1) - 1
    return nlc

nlc = NonlinearConstraint(nonlinear_c, -np.inf, 0)

bounds = Bounds([-1.5, -1.5], [1.5, 1.5])

res = minimize(obj_f, x0,
               method='trust-constr',
               bounds = bounds,
               constraints=[nlc])

optimized_x = res.x;

print("==== Optimal solution ====")
print(res.x)

print("==== Optimized objective ====")
print(res.fun)

# Visualization
num = 201; # number of mesh grids
rr = np.linspace(-2,2,num)
xx1,xx2 = np.meshgrid(rr,rr);

yy = obj_f(np.vstack([xx1.ravel(), xx2.ravel()])).reshape((num,num))

fig, ax = plt.subplots()

ax.contourf(xx1,xx2,yy, levels = 20, cmap="RdYlBu_r")

yy_nlc = nonlinear_c(np.vstack([xx1.ravel(), xx2.ravel()])).reshape((num,num))
ax.contour(xx1,xx2,yy_nlc, levels = [0], colors="k")

plt.plot(optimized_x[0],optimized_x[1], 'rx', markersize = 12)

ax.set_xlabel('$\it{x}_1$')
ax.set_ylabel('$\it{x}_2$')
ax.axis('square')
ax.grid(linestyle='--', linewidth=0.25, color=[0.5,0.5,0.5])
ax.set_xlim([rr.min(),rr.max()])
ax.set_ylim([rr.min(),rr.max()])
```



本章浮光掠影地全景介绍了优化问题及求解。本章给出的求解方法是不含约束条件的解析法。本系列丛书后续还会介绍拉格朗日乘子法，它将有约束优化问题转化为无约束；这种方法在很多数据科学和机器学习算法中应用广泛。本系列丛书后续还将介绍各种数值优化算法，以及全局优化算法。