

Errata

勘误表

* 勘误表配合《数学要素》纸质版图书。勘误表不断更新，请大家注意下载最新版本。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

Preface

前言

感谢

首先感谢大家的信任。

作者仅仅是在学习应用数学科学和机器学习算法时，多读了几本数学书，多做了些思考和知识整理而已。知者不言，言者不知。知者不博，博者不知。水平有限，把自己有限所学所思斗胆和大家分享，作者权当无知者无畏。希望大家在 B 站视频下方和 Github 多提意见，让这套书成为作者和读者共同参与创作的优质作品。

特别感谢清华大学出版社的栾大成老师。从选题策划、内容创作、装帧设计，栾老师事无巨细、一路陪伴。每次和栾老师交流，我都能感受到他对优质作品的追求、对知识分享的热情。

出来混总是要还的

曾几何时，考试是我们学习数学的唯一动力。考试是头悬梁的绳，是锥刺股的锥。我们中的绝大多数人从小到大为各种考试埋头题海，数学味同嚼蜡，甚至让人恨之入骨。

数学给我们带来了无尽的折磨。我们憎恨数学，恐惧数学，恨不得一走出校门就把数学抛之脑后、老死不相往来。

可悲可笑的是，我们其中很多人可能会在毕业的五年或十年以后，因为工作需要，不得不重新学习微积分、线性代数、概率统计，悔恨当初没有学好数学、走了很多弯路、没能学以致用，从而迁怒于教材和老师。

这一切不能都怪数学，值得反思的是我们学习数学的方法、目的。

再给自己一个学数学的理由

为考试而学数学，是被逼无奈的举动。而为数学而数学，则又太过高尚而遥不可及。

相信对于绝大部分的我们来说，数学是工具、是谋生手段，而不是目的。我们主动学数学，是想用数学工具解决具体问题。

现在，这套书给大家一个“学数学、用数学”的全新动力——数据科学、机器学习。

数据科学和机器学习已经深度融合到我们生活的方方面面，而数学正是开启未来大门的钥匙。不是所有人生来都握有一副好牌，但是掌握“数学 + 编程 + 机器学习”绝对是王牌。这次，学习数学不再是为了考试、分数、升学，而是投资时间、自我实现、面向未来。

未来已来，你来不来？

本套丛书如何帮到你

为了让大家学数学、用数学，甚至爱上数学，作者可谓颇费心机。在创作这套书时，作者尽量克服传统数学教材的各种弊端，让大家学习时有兴趣、看得懂、有思考、更自信、用得着。

为此，丛书在内容创作上突出以下几个特点：

- ◀ **数学 + 艺术**——全彩图解，极致可视化，让数学思想跃然纸上、生动有趣、一看就懂，同时提高大家的数据思维、几何想象力、艺术感；
- ◀ **零基础**——从零开始学习 Python 编程，从写第一行代码到搭建数据科学和机器学习应用；
- ◀ **知识网络**——打破数学板块之间的壁垒，让大家看到数学代数、几何、线性代数、微积分、概率统计等板块之间的联系，编织一张绵密的数学知识网络；
- ◀ **动手**——授人以鱼不如授人以渔，和大家一起写代码、用 Streamlit 创作数学动画、交互 App；
- ◀ **学习生态**——构造自主探究式学习生态环境“微课视频 + 纸质图书 + 电子图书 + 代码文件 + 可视化工具 + 思维导图”，提供各种优质学习资源；
- ◀ **理论 + 实践**——从加减乘除到机器学习，丛书内容安排由浅入深、螺旋上升，兼顾理论和实践；在编程中学习数学，学习数学时解决实际问题。

虽然本书标榜“从加减乘除到机器学习”，但是建议读者朋友们至少具备高中数学知识。如果读者正在学习或曾经学过大学数学（微积分、线性代数、概率统计），这套书就更容易读了。

聊聊数学

数学是工具。锤子是工具，剪刀是工具，数学也是工具。

数学是思想。数学是人类思想的高度抽象的结晶体。在其冷酷的外表之下，数学的内核实际上就是人类朴素的思想。学习数学时，知其然，更要知其所以然。不要死记硬背公式定理，理解背后的数学思想才是关键。如果你能画一幅图、用大白话描述清楚一个公式、一则定理，这就说明你真正理解了它。

数学是语言。就好比世界各地不同种族有自己的语言，数学则是人类共同的语言和逻辑。数学这门语言极其精准、高度抽象，放之四海而皆准。虽然我们中绝大多数人没有被数学女神选中，不能为人类的对数学认知开疆扩土；但是，这丝毫不妨碍我们使用数学这门语言。就好比，我们不会成为语言学家，我们完全可以使用母语和外语交流。

数学是体系。代数、几何、线性代数、微积分、概率统计、优化方法等等，看似一个个孤岛，实际上都是数学网络的一条条织线。建议大家学习时，特别关注不同数学板块之间的联系，见树，更要见林。

数学是基石。拿破仑曾说“数学的日臻完善和这个国强民富息息相关。”数学是科学进步的根基，是经济繁荣的支柱，是保家卫国的武器，是探索星辰大海的航船。

数学是艺术。数学和音乐、绘画、建筑一样，都是人类艺术体验。通过可视化工具，我们会在看似枯燥的公式、定理、数据背后，发现数学之美。

数学是历史，是人类共同记忆体。“历史是过去，又属于现在，同时在指引未来。”数学是人类的集体学习思考，她把人的思维符号化、形式化，进而记录、积累、传播、创新、发展。从甲

骨、泥板、石板、竹简、木牍、纸草、羊皮卷、活字印刷、纸质书，到数字媒介，这一过程持续了数千年，至今绵延不息。

数学是无穷无尽的**想象力**，是人类的**好奇心**，是自我挑战的**毅力**，是一个接着一个的**问题**，是看似荒诞不经的**猜想**，是一次次胆大包天的**批判性思考**，是敢于站在前人的肩膀之上的**勇气**，是孜孜不倦地延展人类认知边界的**不懈努力**。

家园、诗、远方

诺瓦利斯曾说：“哲学就是怀着一种乡愁的冲动到处去寻找家园。”

在纷繁复杂的尘世，数学纯粹的就像精神的世外桃源。数学是，一束光，一条巷，一团不灭的希望，一股磅礴的力量，一个值得寄托的避风港。

打破陈腐的锁链，把功利心暂放一边，我们一道怀揣一分乡愁、心存些许诗意、踩着艺术维度，投入数学张开的臂膀，驶入她色彩斑斓、变幻无穷的深港，感受久违的归属，一睹更美、更好的远方。

Acknowledgement

致谢

To my parents.

谨以此书献给我的母亲父亲

How to Use the Book

使用本书

丛书资源

本系列丛书提供的配套资源有以下几个：

- ◀ 纸质图书；
- ◀ PDF 文件，方便移动终端学习；请大家注意，纸质图书经过出版社五审五校修改，内容细节上会和 PDF 文件有出入。
- ◀ 每章提供思维导图，纸质书提供全书思维导图海报；
- ◀ Python 代码文件，直接下载运行，或者复制、粘贴到 Jupyter 运行；
- ◀ Python 代码中有专门用 Streamlit 开发数学动画和交互 App 的文件；
- ◀ 微课视频，强调重点、讲解难点、聊聊天。

在纸质书中为了方便大家查找不同配套资源，作者特别设计了如下几个标识。



微课视频

本书配套微课视频均发布在 B 站——生姜 DrGinger：

- ◀ <https://space.bilibili.com/513194466>

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

微课视频是以“聊天”的方式，和大家探讨某个数学话题的重点内容，讲讲代码中可能遇到的难点，甚至侃侃历史、说说时事、聊聊生活。

本书配套的微课视频目的是引导大家自主编程实践、探究式学习，并不是“照本宣科”。

纸质图书上已经写得很清楚的内容，视频课程只会强调重点。需要说明的是，图书内容不是视频的“逐字稿”。

代码文件

本系列丛书的 Python 代码文件下载地址为：

► <https://github.com/Visualize-ML>

Python 代码文件会不定期修改，请大家注意更新。图书配套的 PDF 文件和勘误也会上传到这个 GitHub 账户。因此，建议大家注册 GitHub 账户，给书稿文件夹标星 (star) 或分支克隆 (fork)。

考虑再三，作者还是决定不把代码全文印在纸质书中，以便减少篇幅，节约用纸。

本书编程实践例子中主要使用“鸢尾花数据集”，数据来源是 Scikit-learn 库、Seaborn 库。此外，系列丛书封面设计致敬梵高《鸢尾花》，要是给本系列丛书起个昵称的话，作者乐见“鸢尾花书”。

App 开发

本书几乎每一章都至少有一个用 Streamlit 开发的 App，用来展示数学动画、数据分析、机器学习算法。

Streamlit 是个开源的 Python 库，能够方便快捷搭建、部署交互型网页 App。Streamlit 非常简单易用、很受欢迎。Streamlit 兼容目前主流的 Python 数据分析库，比如 NumPy、Pandas、Scikit-learn、PyTorch、TensorFlow 等等。Streamlit 还支持 Plotly、Bokeh、Altair 等交互可视化库。

本书中很多 App 设计都采用 Streamlit + Plotly 方案。此外，本书专门配套教学视频手把手和大家一起做 App。

大家可以参考如下页面，更多了解 Streamlit：

► <https://streamlit.io/gallery>

► <https://docs.streamlit.io/library/api-reference>

实践平台

本书作者编写代码时采用的 IDE (integrated development environment) 是 Spyder，目的是给大家提供简洁的 Python 代码文件。

但是，建议大家采用 JupyterLab 或 Jupyter notebook 作为本系列丛书配套学习工具。

简单来说，Jupyter 集合“浏览器 + 编程 + 文档 + 绘图 + 多媒体 + 发布”众多功能与一身，非常适合探究式学习。

运行 Jupyter 无需 IDE，只需要浏览器。Jupyter 容易分块执行代码。Jupyter 支持 inline 打印结果，直接将结果图片打印在分块代码下方。Jupyter 还支持很多其他语言，比如 R 和 Julia。

使用 markdown 文档编辑功能，可以编程同时写笔记，不需要额外创建文档。Jupyter 中插入图片和视频链接都很方便。此外，还可以插入 Latex 公式。对于长文档，可以用边栏目录查找特定内容。

Jupyter 发布功能很友好，方便打印成 HTML、PDF 等格式文件。

Jupyter 也并不完美，目前尚待解决的问题有几个。Jupyter 中代码调试不方便，需要安装专门插件（比如 debugger）。Jupyter 没有 variable explorer，要么 inline 打印数据，要么将数据写到 csv 或 Excel 文件中再打开。图像结果不具有交互性，比如不能查看某个点的值，或者旋转 3D 图形，可以考虑安装 (jupyter-matplotlib)。注意，利用 Altair 或 Plotly 绘制的图像支持交互功能。对于自定义函数，目前没有快捷键直接跳转到其定义。但是，很多开发者针对这些问题都开发了插件，请大家留意。

大家可以下载安装 Anaconda，JupyterLab、Spyder、PyCharm 等常用工具都集成在 Anaconda 中。下载 Anaconda 的地址为：

◀ <https://www.anaconda.com/>

学习步骤

大家可以根据自己的偏好制定学习步骤，本书推荐如下步骤。



学完每章后，大家可以在平台上发布自己的 Jupyter 笔记，进一步听取朋友们的意见，共同进步。这样做还可以提高自己学习的动力。

意见建议

欢迎大家对本系列丛书提意见和建议，丛书专属邮箱地址为：

◀ jiang.visualize.ml@gmail.com

也欢迎大家在 B 站视频下方留言互动。

Contents

目录



0.1 本册在全套丛书的定位

本系列丛书有三大板块——编程、数学、实践。机器学习各种算法离不开数学，而《数学要素》一册是“数学”板块的第一册。本书介绍的数学工具是整个“数学”板块的基础，当然也是数据科学和机器学习实践的基础。

《数学要素》一册中编程和可视化无处不在，限于篇幅本书不会专门讲解编程基础内容。因此，建议编程零基础读者先学习《编程不难》和《可视之美》两册内容。当然，根据个人情况，平行学习《数学要素》、《编程不难》和《可视之美》，也是可行的。

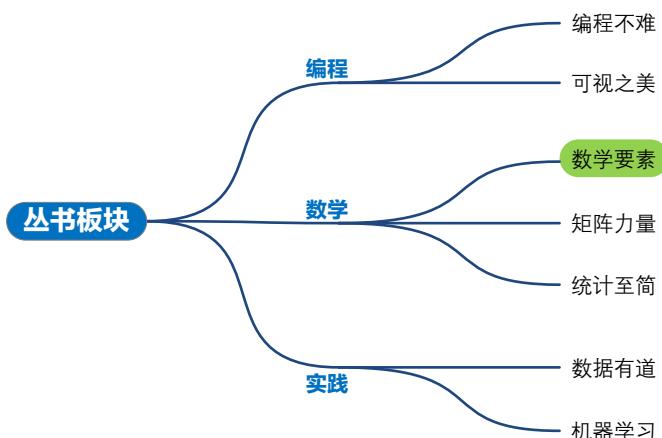


图 1. 本系列丛书板块布局

0.2 结构：7 大板块

本书可以归纳为 7 大板块——基础、坐标系、函数、解析几何、微积分、概率统计、线性代数。

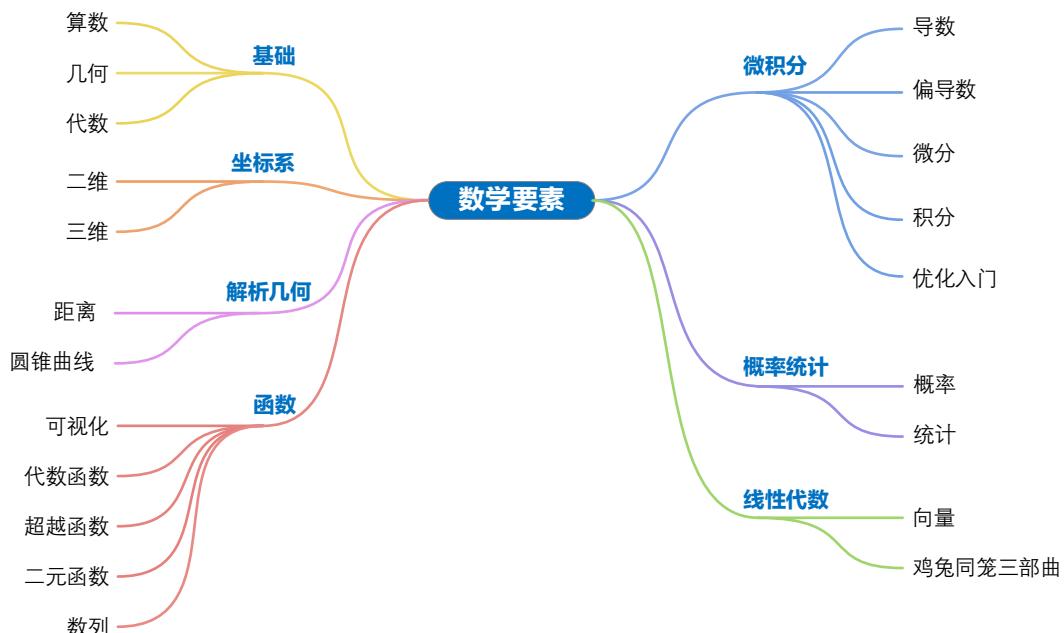


图 2. 《数学要素》板块布局

基础部分

基础部分从加、减、乘、除四则运算讲起。线性代数在机器学习中应用广泛，本书第 1、2 章开门见山讲向量和矩阵的基本运算，也会在本册各个板块见缝插针介绍线性代数基础知识。

本书第 3 章回顾常用几何知识，几何视角是本系列丛书的一大特色。这一章有一大亮点——圆周率估算。圆周率估算本书的一条重要线索，本书会按时间先后顺序介绍如何用不同数学工具估算圆周率。

第 4 章回顾代数知识，里面有两个亮点值得大家特别注意：一个是杨辉三角，本书后面会将杨辉三角和概率统计、随机过程联系起来；另一个是鸡兔同笼问题，本书最后三章都围绕鸡兔同笼这个话题展开。

坐标系

笛卡尔坐标系让几何和代数走到一起，本书第 5、6 两章介绍坐标系有关内容。这两章的一大特色是——代数式可视化，几何体参数化。没有坐标系，就没有函数，也不会有微积分；因此，坐标系的地位毋庸多言。

解析几何

第 7、8、9 三章介绍解析几何内容，其中有两大亮点——距离度量、椭圆。距离度量中，大家要善于用等距线这个可视化工具。此外，大家需要注意欧氏距离并不是唯一的距离度量。第二个亮点是椭圆，椭圆可谓“多面手”，大家很快会看到椭圆在概率统计、线性代数、数学科学、机器学习中大放异彩。

函数

第 10 章到第 14 章都是围绕函数展开。有几点值得强调，学习任何函数时，建议大家编程绘制函数线图，以便观察函数形状、变化趋势。此外，学会利用曲面、剖面线、等高线等可视化工观察分析二元函数。再者，不同函数都有自身特定性质，对应独特应用场景。第 14 章讲解数列，数列可以看成是特殊的函数。本章中，累加、极限这两个知识点特别值得关注，它们都是微积分基础。

微积分

第 15 章到第 19 章讲解微积分以及优化问题内容。牛顿和莱布尼兹分别发明微积分之后，整个数学王国的版图天翻地覆。导数、偏导数、微分、积分给我们提供研究函数性质的量化工具。学好这四章的秘诀就是——几何图解。导数是切线斜率，偏导数是某个变量方向上切线斜率，微分是线性近似，泰勒展开是多项式函数叠加，积分是求面积，二重积分是求体积。数据科学、机器学习中所有算法都可以写成优化问题，而构造、求解优化问题离不开微积分。因此，本书在讲完微积分之后立刻安排了第 19 章，介绍优化问题入门知识。本系列丛书后续还会在各册不断介绍优化方法。

概率统计

第 20、21 两章是概率统计入门。本系列丛书专门有《概率统计》一册系统讲解这个版块，但是这不意味着本书第 20、21 两章内容毫无出彩之处；相反，这两章亮点颇多。第 20 章概率内容实际上是代数部分杨辉三角的延伸，本章用二叉树这个知识点，将代数和概率统计串联在一起。第 20 章最后还介绍了随机过程。第 21 章的关键词就是“图解”，用图像可视化数据，用图像展示概率统计定义。

线性代数

本书最后四章以线性代数收尾。第 22 章可视化向量和向量运算。第 23、24、25 三章是“鸡兔同笼三部曲”，这三章虚构了一个世外桃源，讲述与世隔绝的村民如何利用舶来的线性代数知识，解决村民养鸡养兔时遇到的数学疑难杂症。这三章涉及线性方程组、向量空间、投影、最小二乘法线性回归、马尔科夫过程、特征值分解等内容。这三章一方面给大家展示本书重要数学工具的应用，另外这三章也为本系列丛书《矩阵力量》一册做了内容预告和铺垫。

0.3 特点：知识融合

《数学要素》打破数学板块的藩篱，将算数、代数、线性代数、几何、解析几何、概率统计、微积分、优化方法等板块有机结合在一起。

作为丛书的核心特点，《数学要素》一册内容编排上突出“图解 + 编程 + 机器学习应用”。讲解一些特定数学工具时，本书会穿插介绍其在数据科学和机器学习领域应用场景，让大家学以致用。本书主要使用 Python 的这几个库——Scipy、Numpy、Sympy、Matplotlib、Seaborn 等。

《数学要素》一册还强调数学文化，内容安排上尽可能沿着数学发展先后脉络，为大家展现一幅历史图景。本书还介绍数学史上关键人物，让大家看到数学如何薪火相传、接续发展。

为了帮助大家阅读英文文献以及学术交流，本书还特别总结常用数学知识的英文表述。

下面让我们一起开始《数学要素》一册的学习之旅。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

All Is Number

1 万物皆数

数字统治万物



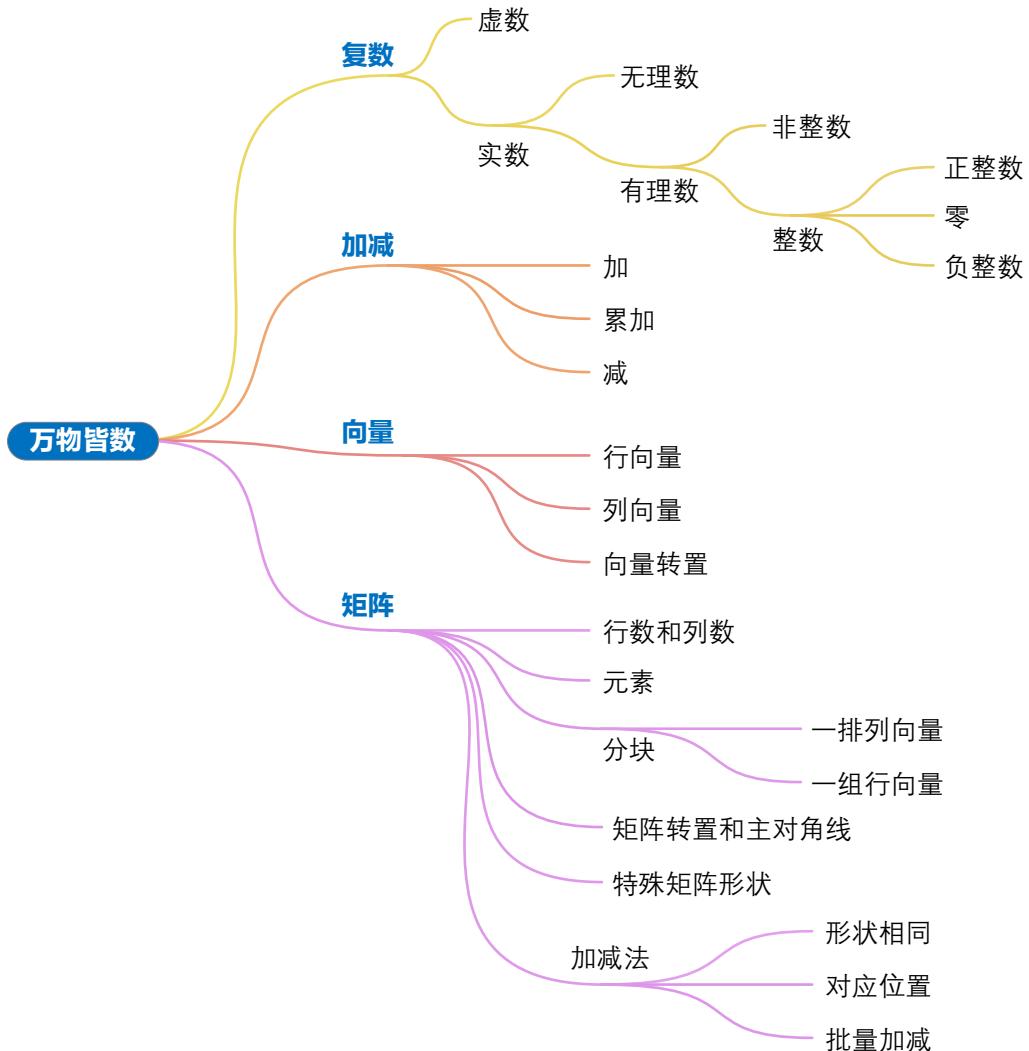
万物皆数。

All is Number.

—— 毕达哥拉斯 (Pythagoras) | 古希腊哲学家、数学家 | 570 ~ 495 BC



- ◀ % 求余数
- ◀ float() 将输入转化为浮点数
- ◀ input() 函数接受一个标准输入数据，返回为 string 类型
- ◀ int() 将输入转化为整数
- ◀ is_integer() 判断是否为整数
- ◀ lambda 构造匿名函数；匿名函数是指一类无需定义函数名的函数或子程序
- ◀ len() 返回序列或者数据帧的数据数量
- ◀ math.e math 库中的欧拉数
- ◀ math.pi math 库中的圆周率
- ◀ math.sqrt(2) math 库计算 2 的平方根
- ◀ mpmath.e mpmath 库中的欧拉数
- ◀ mpmath.pi mpmath 库中的圆周率
- ◀ mpmath.sqrt(2) mpmath 库计算 2 的平方根
- ◀ numpy.add() 向量或矩阵加法
- ◀ numpy.array() 构造数组、向量或矩阵
- ◀ numpy.cumsum() 计算累计求和
- ◀ numpy.linspace() 在指定的间隔内，返回固定步长数组
- ◀ numpy.matrix() 构造二维矩阵
- ◀ print() 在 console 打印
- ◀ range() 返回的是一个可迭代对象，range(10) 返回 0 ~ 9，等价于 range(0, 10); range(1, 11) 返回 1 ~ 10; range(0, -10, -1) 返回 0 ~ -9; range(0, 10, 3) 返回 [0, 3, 6, 9]，步长为 3
- ◀ zip(*) 将可迭代的对象作为参数，让对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。* 代表解包，返回的每一个都是元组类型，而并非是原来的数据类型



1.1 数字和运算：人类思想的伟大飞跃

数字，就是人类思想的空气，无处不在，不可或缺。

大家不妨停止阅读，用一分钟时间，看看自己身边哪里存在数字。

举目四望，你会发现，键盘上有数字，书本印着数字，手机显示数字，交易媒介充满数字，食品有卡路里数值，时钟的数字提醒我们时间，购物扫码本质也是数字。一串串手机号码让人们联通，身份编号是我们的个体的标签 …

当下，数字已经融合到人类生活的方方面面。多数时候，数字像是空气，我们认为它理所应当，甚至忽略了它的存在。

数字是万物的绝对尺度，数字更是一种高阶的思维方式。远古时期，不同地域、不同族群的人类突然意识到，2只鸡，2只兔，2头猪，有一个共性，那就是——2。

2和更多更多数字，以及它们之间加、减、乘、除以及更多复杂运算被抽象出来，这是人类思想的一次伟大飞跃。

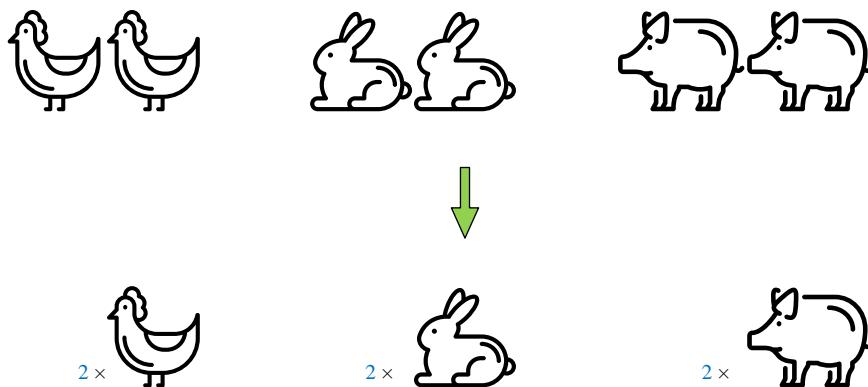


图 1. 数字是人类抽象思维活动的产物

数字这一宝贵的人类遗产，在不同地区、不同种族之间薪火相传。

5000 年前，古巴比伦人将各种数学计算，比如倒数、平方、立方，刻在泥板上。古埃及则是将大量数学知识记录在纸草上。

古巴比伦采用 60 进制。不谋而合，中国自古便发明使用天干地支六十甲子为一个周期来纪年。

今人所说的“阿拉伯数字”实际上是古印度人创造的。古印度发明了**十进制** (decimal system)，而古阿拉伯人将它们发扬光大。中世纪末期，十进制传入欧洲，而日后成为全世界的标准。中国古代也创造了十进制。

有学者认为，人类不约而同地发明并广泛使用十进制，是因为人类有十根手指。人们数数的时候，自然而然地用手指记录。

虽然十进制大行其道，但是其他进制依然广泛运用。比如，[二进制](#) (binary system)、[八进制](#) (octal system) 和[十六进制](#) (hexadecimal system) 经常在电子系统使用。

日常生活中，我们不知不觉中也经常使用其他进制。[十二进制](#) (duodecimal system) 常常出现，比如十二小时制、一年十二个月、[黄道十二宫](#) (zodiac)、[十二地支](#) (Earthly Branches)、[十二生肖](#) (Chinese zodiac)。四进制也不罕见，比如一年四个季度。二十四进制用在一天 24 小时、一年 24 节气。六十进制也很常用，比如一分钟 60 秒、一小时 60 分钟。



图 2. 古巴比伦泥板 (图片来自 Wikipedia)

随着科学技术持续发展，人类的计算也日趋复杂。零、负数、分数、小数、无理数、虚数被发明创造出来。于此同时，人类也在发明改进计算工具，让计算更快、更准。

算盘，作为一种原始的计算工具，现在已经基本绝迹。随着运算量和复杂度不断提高，对运算速度、准确度的需求激增，人类亟需摆脱手工运算，计算器应运而生。

1622 年，英国数学家[威廉·奥特雷德](#) (William Oughtred) 发明计算尺。早期计算尺主要用来四则运算，而后发展到可以用来求对数和三角函数。直到二十世纪后期被便携式计算器代替之前，计算尺一度是科学和工程重要的计算工具。

1642 年，法国数学家[帕斯卡](#) (Blaise Pascal) 发明机械计算器，这个机器可以直接进行加减运算。难以想象，帕斯卡设计自己第一台计算器时未满 19 岁。

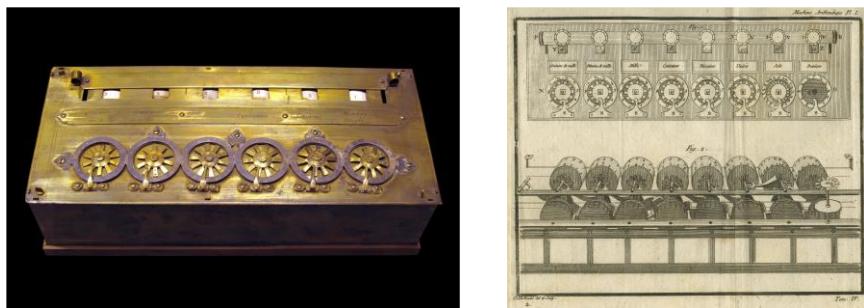


图 3. 帕斯卡机械计算器和原理图纸 (图片来自 Wikipedia)

1822 年前后，英国数学家查尔斯·巴贝奇 (Charles Babbage)，设计完成差分机 (difference engine)。第一台差分机重达 4 吨，最高可以存 16 位数。差分机是以蒸汽机为动力的自动机械计算器，它已经很接近第一台计算机。因此，也有很多人将查尔斯·巴贝奇称作“计算器之父”。

1945 年，“伊尼雅克”ENIAC 诞生。ENIAC 的全称为电子数值积分计算机 (Electronic Numerical Integrator And Computer)。ENIAC 是一台真正意义上的电子计算机。ENIAC 重达 27 吨，占地 167 平方米。

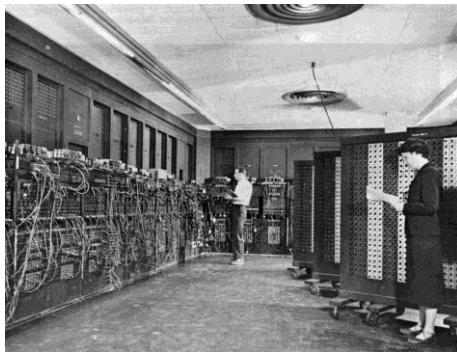


图 4. ENIAC 计算机 (图片来自 Wikipedia)

20 世纪 50 年代电子计算机主要使用真空管，而后开始使用半导体晶体管。半导体使得计算机体积变得更小、成本更低、耗电更少、性能更可靠。进入二十世纪，计算机器的更新迭代，让人目不暇接，甚至让人感觉窒息。

20 世纪 70 年代，集成电路和微处理器先后投入大规模使用，计算机和其他智能设备开始逐渐步入寻常百姓家。现如今，计算的竞赛愈演愈烈，量子计算机的研究进展如火如荼。

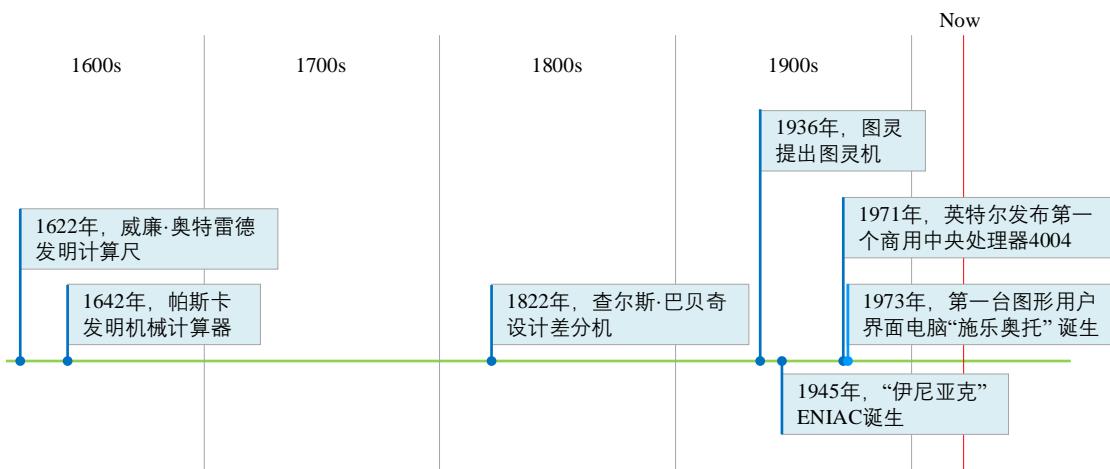


图 5. 计算器发展历史时间轴

到这里，我们不妨停下来，喘口气，回望来时的路。再去看看数字最朴素、最原始、最直觉的形态。

1.2 数字分类：从复数到自然数

本节介绍数字分类，介绍的数字类型如图 6 所示。

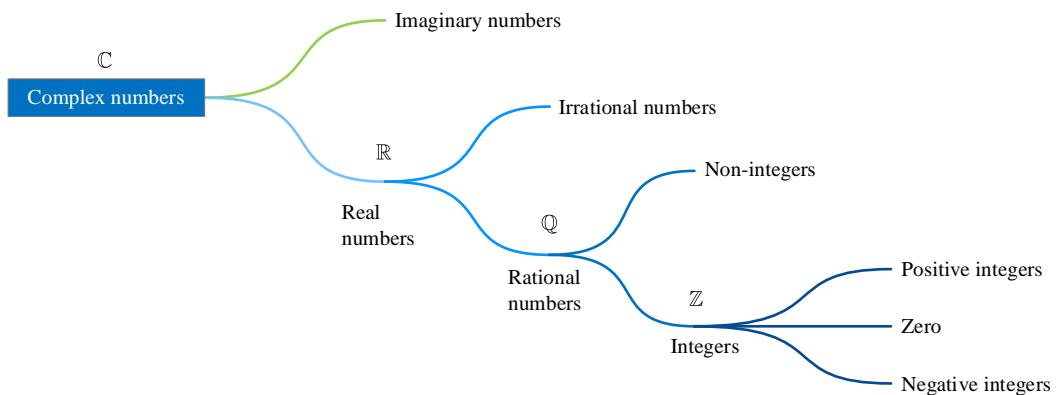


图 6. 数字分类

复数

复数 (complex number) 包括**实数** (real numbers) 和**虚数** (imaginary numbers)。**复数集** (the set of complex numbers) 的记号为 \mathbb{C} 。

集合 (set)，简称**集**，是指具有某种特定性质**元素** (element) 的**总体**。白话说，集合就是一堆东西构成的整体。因此，复数集是所有复数构成的总体。

复数的具体形式如下：

$$a + bi \quad (1)$$

其中， a 和 b 是实数。

(1) 中， i 是**虚数单位** (imaginary unit)， i 的平方等于 -1 ，即：

$$i^2 = -1 \quad (2)$$

笛卡尔 (René Descartes) 最先提出虚数这个概念。而后，**欧拉** (Leonhard Euler) 和**高斯** (Carl Friedrich Gauss) 等人对虚数做了深入研究。

有意思的是，不经意间，(1) 中便使用 a 和 b 来代表实数。用抽象字母来代表具体数值是代数的基础。**代数** (algebra) 的研究对象不仅是数字，还包括各种抽象化的结构。

实数

实数集 (the set of real numbers) 记号为 \mathbb{R} 。实数包括**有理数** (rational numbers) 和**无理数** (irrational numbers)。

(1) 中， $b = 0$ 时，便得到的是实数。如图 7 所示，实数集合可以用**实数轴** (real number line 或 real line) 来展示。

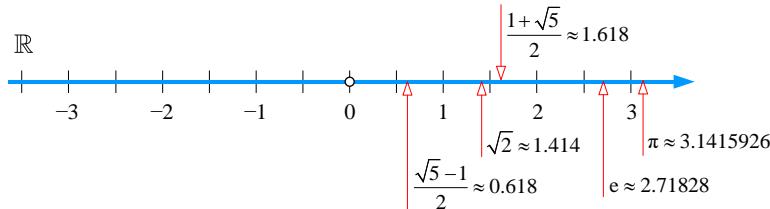


图 7. 实数轴

无意之间，我们又用到了解析几何中重要的工具之——**数轴** (number line)。图 7 这个看似平凡无奇的一根轴，实际上是人类的一个伟大发明创造。

数轴描述一维空间，两根垂直并相交于原点的数轴张成二维直角坐标系，即笛卡尔坐标系 (Cartesian coordinate system)。在二维直角坐标系原点处升起一根垂直平面的数轴便张成了三维直角坐标系。



数轴和坐标系的发明让代数和几何前所未有的结合在一起，这是本书第 5、6 章要介绍的内容。

目前，数学被分割成一个个板块——算数、代数、几何、解析几何、线性代数、概率统计等等——这种安排虽然有利于特定类别数学工具学习，但是板块之间的联系被人为割裂。本书的重要任务之一就是强化各个板块之间的联系，让大家看见森林，而不是一棵棵树。

有理数

有理数集合用 \mathbb{Q} 表示，有理数可以表达为两个整数的商 (quotient of two integers)，形如：

$$\frac{a}{b} \quad (3)$$

其中， a 为分子 (numerator)， b 为分母 (denominator)。(3) 中分母不为零 (the denominator is not equal to zero)。

有理数可以表达成有限小数 (finite decimal 或 terminating decimal)，或者无限循环小数 (repeating decimal 或 recurring decimal)。小数中的圆点叫做小数点 (decimal separator)。

无理数

图 7 所示实数轴上除有理数以外，都是无理数。无理数不能用一个整数或两个整数的商来表示。无理数也叫无限不循环小数 (non-repeating decimal)。

很多重要的数值都是无理数，比如图 7 所示数轴上的圆周率 π (pi)、 $\sqrt{2}$ (the square root of two)、自然常数 e (exponential constant) 和黄金分割比 (golden ratio)。自然常数 e 也叫欧拉数 (Euler's number)。



执行 Bk3_Ch1_01.py 代码，可以打印出 π 、 e 和 $\sqrt{2}$ 的精确值。代码使用了 Math 库中函数，Math 库是 Python 提供的内置数学函数库。

打印结果如下。

```
pi = 3.141592653589793
e = 2.718281828459045
sqrt(2) = 1.4142135623730951
```

下面，我们做一个有趣的实验——打印圆周率和自然常数 e 小数点后 1,000 位数字。图 8 所示为圆周率小数点后 1,000 位，图 9 采用热图的形式展示圆周率小数点后 1,024 位。图 10 自然常数 e 小数点后 1,000 位。

中国古代南北朝时期数学家祖冲之 (429 ~ 500) 曾刷新圆周率估算记录。他估算圆周率在 3.1415926 到 3.1415927 之间，这一记录在之后的约 1000 年内无人撼动。圆周率的估算也是本书的一条重要线索，我们将追随前人足迹，用不同的数学工具估算圆周率。

3.14159265358979323846264338327950288419716939937510	100 digits
5820974944592307816406286208998628034825342117067⑨	
82148086513282306647093844609550582231725359408128	200 digits
4811174502841027019385211055596446229489549303819⑥	
44288109756659334461284756482337867831652712019091	
4564856692346034861045432664821339360726024914127③	300 digits
72458700660631558817488152092096282925409171536436	
78925903600113305305488204665213841469519415116094	400 digits
33057270365759591953092186117381932611793105118548	
0744623799627495673518857527248912279381830119491②	500 digits
98336733624406566430860213949463952247371907021798	
6094370277053921717629317675238467481846766940513②	600 digits
00056812714526356082778577134275778960917363717872	
1468440901224953430146549585371050792279689258923⑤	700 digits
42019956112129021960864034418159813629774771309960	
5187072113499999983729780499510597317328160963185⑨	800 digits
50244594553469083026425223082533446850352619311881	
7101000313783875288658753320838142061717766914730③	900 digits
59825349042875546873115956286388235378759375195778	
1857780532171226806613001927876611195909216420198⑨	1000 digits

图 8. 圆周率小数点后 1000 位

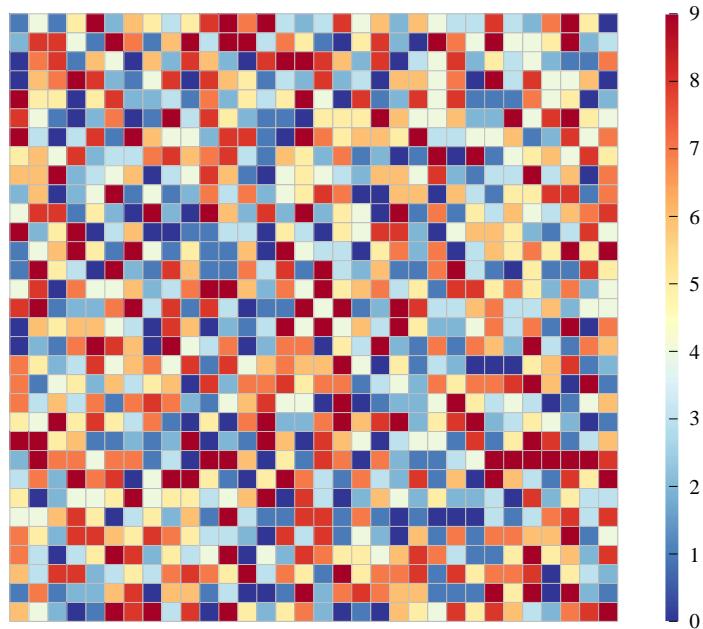
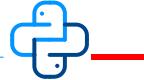


图 9. 圆周率小数点后 1024 位热图

2.71828182845904523536028747135266249775724709369995	100 digits
95749669676277240766303535475945713821785251664274	100 digits
27466391932003059921817413596629043572900334295260	200 digits
59563073813232862794349076323382988075319525101901	200 digits
15738341879307021540891499348841675092447614606680	300 digits
82264800168477411853742345442437107539077744992069	300 digits
55170276183860626133138458300075204493382656029760	400 digits
67371132007093287091274437470472306969772093101416	400 digits
92836819025515108657463772111252389784425056953696	500 digits
77078544996996794686445490598793163688923009879312	500 digits
77361782154249992295763514822082698951936680331825	600 digits
28869398496465105820939239829488793320362509443117	600 digits
30123819706841614039701983767932068328237646480429	700 digits
53118023287825098194558153017567173613320698112509	700 digits
96181881593041690351598888519345807273866738589422	800 digits
87922849989208680582574927961048419844436346324496	800 digits
84875602336248270419786232090021609902353043699418	900 digits
49146314093431738143640546253152096183690888707016	900 digits
76839642437814059271456354906130310720851038375051	1000 digits
01157477041718986106873969655212671546889570350354	1000 digits

图 10. 自然常数 e 小数点后 1000 位



Bk3_Ch1_02.py 打印圆周率、 $\sqrt{2}$ 和自然常数 e，小数点后 1,000 位数字。Mpmath 是一个任意精度浮点运算库。

目前，背诵 pi 的小数点后最多位数的世界吉尼斯纪录是 70,000 位。该纪录由印度人在 2015 年创造，用时近 10 小时。这一纪录的需要背诵的数字量是图 8 的 70 倍，感兴趣的读者可以修改代码获取，并打印保存这些数字。取决于个人电脑的算力，这一过程可能要用时很久。



观察图 8 所示的圆周率小数点后 1,000 位数字，可以发现 0 ~ 9 这 10 个数字反复随机出现。这里，“随机 (random)”是指偶然、随意、无法预测，概率统计中将会大量使用“随机”这个概念。

大家能否凭直觉猜一下哪个数字出现的次数最多？圆周率小数点后 10,000 位、100,000 位，乃至 1,000,000 位，0 ~ 9 这 10 个数字出现的次数又会怎样？

答案就在 Streamlit_Bk3_Ch1_02.py 文件中。我们用 Streamlit 创作了一个数学动画 App 展示圆周率小数点后 0 ~ 9 这 10 个数出现的次数。

整数

整数 (integers) 包括**正整数** (positive integers)、**负整数** (negative integers) 和零。正整数**大于零** (greater than zero)；负整数**小于零** (less than zero)。整数集用 \mathbb{Z} 表达。

整数重要性质之一是——整数相加、相减或相乘结果还是整数。

奇偶性 (parity) 是整数另外一个重要性质。能被 2 整除的整数被称作**偶数** (an integer is called an even integer if it is divisible by two); 否则，**该整数为奇数** (the integer is odd)。



利用以上原理，我们可以写一段 Python 代码，判断数字奇偶性。请大家参考 Bk3_Ch1_03.py。代码中，% 用来求余数。

自然数

自然数 (natural numbers 或 counting numbers) 有些时候指的是正整数，有些时候指的是**非负整数** (nonnegative integer)，这时自然数集合包括“0”。“0”是否属于自然数尚未达成一致意见。

至此，我们回顾了常见数字类型。**表 1** 总结数字类型并给出例子。

表 1. 不同种类数字及举例

英文表达	汉语表达	举例
Complex numbers	复数	$7 + 2i$
Imaginary numbers	虚数	$2i$
Real numbers	实数	7
Irrational numbers	无理数	π, e
Rational numbers	有理数	1.5
Integers	整数	-1, 0, 1
Natural numbers	自然数	9, 18

1.3 加减：最基本的数学运算

本节介绍加、减这两个最基本算数运算。

加法

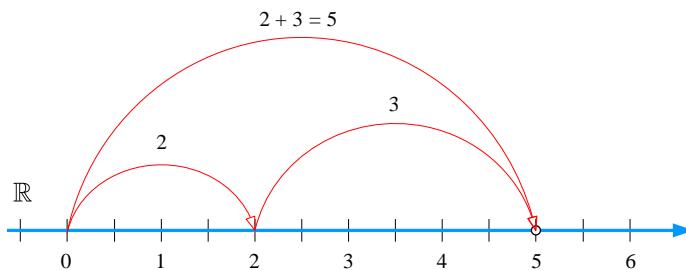
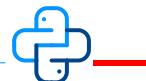
加法 (addition) 的运算符为**加号** (plus sign, plus symbol); 加法运算式中，**等式** (equation) 的左边为**加数** (addend) 和**被加数** (augend, summand)，等式的右边是**和** (sum)。

加法的表达方式多种多样，比如说“**和** (summation)”、“**加** (plus)”、“**增长** (increase)”、“**小计** (subtotal)” 和“**总数** (total)”等等。



图 11. 加法运算

图 12 是在数轴上可视化 $2 + 3 = 5$ 这一加法运算。

图 12. $2 + 3 = 5$ 在数轴上的可视化

Bk3_Ch1_04.py 完成图 12 所示加法运算。

Bk3_Ch1_05.py 对 Bk3_Ch1_04.py 稍作调整，利用 `input()` 函数，让用户通过键盘输入数值。

结果打印如下。

```
Enter first number: 2
Enter second number: 3
The sum of 2 and 3 is 5.0
```

表 2 总结加法的常用英文表达。

表 2. 加法的英文表达

数学表达	英文表达
$1+1=2$	One plus one equals two. The sum of one and one is two. If you add one to one, you get two.
$2+3=5$	Two plus three equals five. Two plus three is equal to five. Three added to two makes five. If you add two to three, you get five.

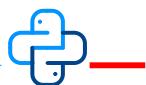
累计求和

对于一行数字，**累计求和** (cumulative sum 或 cumulative total) 得到的结果不是一个总和，而是从左向右每加一个数值，得到的分步结果。比如，自然数 1 到 10 累计求和结果为：

$$1, 3, 6, 10, 15, 21, 28, 36, 45, 55 \quad (4)$$

(4) 累计求和计算过程如下：

$$\begin{array}{r} 1+2+3+4+5+6=21 \\ \hline 1+2+3+4+5=15 \\ \hline 1+2+3+4=10 \\ \hline 1+2+3=6 \\ \hline 1+2=3 \\ \hline 1 \\ 1, 2, 3, 4, 5, 6, 7, 8, \dots \end{array} \quad (5)$$



Bk3_Ch1_06.py 利用 `numpy.linspace(1, 10, 10)` 产生 1~10 这 10 个自然数，然后利用 `numpy.cumsum()` 函数进行累计求和。NumPy 是一个开源的 Python 库，丛书的大量线性代数运算都离不开 NumPy。

减法

减法 (subtraction) 是**加法的逆运算** (inverse operation of addition)，运算符为**减号** (minus sign)。如图 13 所示，减法运算过程是，**被减数** (minuend) 减去**减数** (subtrahend) 得到**差** (difference)。

减法其他名字包括“**减** (minus)”、“**少** (less)”、“**差** (difference)”、“**减少** (decrease)”、“**拿走** (take away)”和“**扣除** (deduct)”等等。



图 13. 减法运算

图 14 所示为在数轴上展示 $5 - 3 = 2$ 的减法运算。

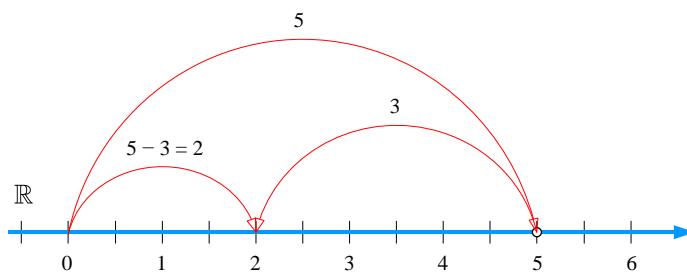
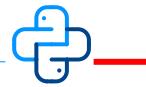


图 14. $5 - 3 = 2$ 在数轴上的可视化



Bk3_Ch1_07.py 完成图 14 给出的减法运算。

相反数

求**相反数** (inverse number 或 additive inverse number) 的过程是**改变符号** (reverses its sign)，这个操作常称作**变号** (sign change)。比如，5 的相反数为 -5 (negative five)。表 3 给出减法常用的英文表达。

表 3. 减法常见英文表达

数学表达	英文表达
$5 - 3 = 2$	Five minus three equals two. Five minus three is equal to two. Three subtracted from five equals two. If you subtract three from five, you get two. If you take three from five, you get two.
$4 - 6 = -2$	Four minus six equals negative two. Four minus six is equal to negative two.

1.4 向量：数字排成行、列

本书的读者会问，明明第一章讲的是算数，怎么一下扯到“向量”这个线性代数的概念。

向量、矩阵等线性代数概念对于数据科学和机器学习至关重要。在机器学习中，数据几乎都是以矩阵形式存储、运算。毫不夸张地说，没有线性代数就没有现代计算机运算。逐渐地，大家会发现算数、代数、解析几何、微积分、概率统计、优化方法并不是一个个孤岛，线性代数正是连接它们的重要桥梁之一。

然而，初学者对向量、矩阵等概念却展现出特别的抗拒，甚至恐惧。

基于以上考虑，本书把线性代数基础概念穿插到各个板块，以便破除大家对线性代数的恐惧，加强大家对这个数学工具的理解。

书归正传。

行向量、列向量

若干数字排成一行或一列，并且用中括号括起来，得到的数组叫做**向量** (vector)。

排成一行的叫做**行向量** (row vector)，排成一列的叫做**列向量** (column vector)。

白话讲，行向量就是表格的一行数字，列向量就是表格的一列数字。如下两例分别展示行向量和列向量：

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}_{1 \times 3}, \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_{3 \times 1} \quad (6)$$

(6) 中下角标， 1×3 代表“1 行、3 列”， 3×1 代表“3 行、1 列”。本书在给出向量和矩阵时，偶尔会以下角标形式展示其形状，比如 $X_{150 \times 4}$ 代表矩阵 X 有 150 行、4 列。

利用 NumPy 库，可以用 `numpy.array([[1, 2, 3]])` 定义 (6) 中行向量。用 `numpy.array([1, 2, 3])` 定义 (6) 中列向量。

⚠ 注意，用 `numpy.array()` 函数定义向量时如果只用一层中括号 `[]`，比如 `numpy.array([1, 2, 3])`，得到的结果只有一个维度。有两层中括号 `[[[]]]`，`numpy.array([[1, 2, 3]])` 得到的结果有两个维度。这一点在 NumPy 库矩阵运算中非常重要。

转置

本系列丛书采用的转置符号为上标 T 。行向量 **转置** (transpose) 得到列向量；同理，列向量转置得到行向量。比如下两例：

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \quad (7)$$

如图 15 所示，转置相当于镜像。图 15 中红线就是镜像轴，红线从第 1 行、第 1 列元素出发，朝向右下方 45° 。

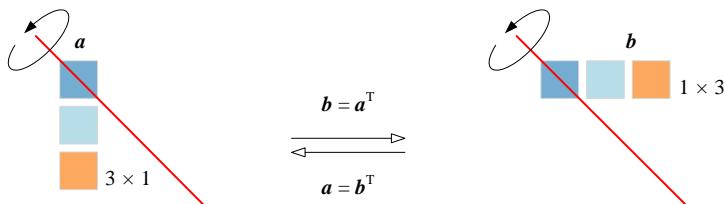


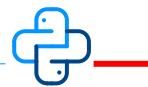
图 15. 向量转置

本系列丛书用加粗、斜体小写字母来代表向量，比如图 15 中向量 a 和向量 b 。

给定如下行向量 a ， a 有 n 个元素，元素本身用小写字母表示，比如：

$$\mathbf{a} = [a_1 \quad a_2 \quad \dots \quad a_n] \quad (8)$$

其中，下角标代表向量元素的序数； $[a_1, a_2, \dots, a_n]$ 读作“ n row vector, a sub one, a sub two, dot dot dot, a sub n ”。



Bk3_Ch1_08.py 定义行向量和列向量，并展示如何通过转置将行向量和列向量相互转换。



本书在介绍线性代数相关知识时，会尽量使用具体数字，而不是变量符号。这样做的考虑是，让读者构建向量和矩阵运算最直观的体验。这给本系列丛书《矩阵力量》一册打下基础。

《矩阵力量》一册则系统讲解线性代数知识，以及线性代数和代数、解析几何、微积分、概率统计、优化方法、数据科学等板块的联系。

1.5 矩阵：数字排列成长方形

矩阵 (matrix) 将一系列数字以长方形方式排列，比如：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3}, \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3 \times 2}, \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}_{2 \times 2} \quad (9)$$

白话说，矩阵将数字排列成表格，有行、有列。(9) 给出三个矩阵，形状分别是 2 行 3 列 (记做 2×3)、3 行 2 列 (记做 3×2) 和 2 行 2 列 (记做 2×2)。

丛书用大写、斜体字母代表矩阵，比如矩阵 A 和矩阵 B 。

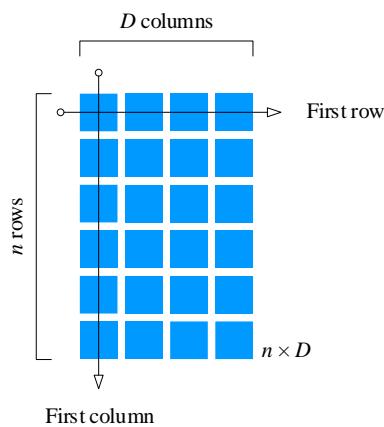


图 16. $n \times D$ 矩阵 X

图 16 所示为一个 $n \times D$ (n by capital D) 矩阵 X ， n 是**矩阵的行数** (number of rows in the matrix)， D 是**矩阵的列数** (number of columns in the matrix)。 X 可以展开写成如下表格形式：

$$\mathbf{X}_{n \times D} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,D} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,D} \end{bmatrix} \quad (10)$$

矩阵 X 中，元素 (element) $x_{i,j}$ 被称作 i,j 元素 (i,j entry 或 i,j element)，也可以说 $x_{i,j}$ 出现在 i 行 j 列 (appears in row i and column j)。比如， $x_{n,1}$ 是矩阵 X 的第 n 行、第 1 列元素。

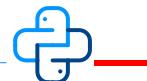
⚠ 再次强调，先说行序号，再说列序号。

本系列丛书中，数据矩阵一般采用大写、粗体、斜体 \mathbf{X} 表达。

表 4 总结如何用英文读矩阵和矩阵元素。

表 4. 矩阵有关英文表达

数学表达	英文表达
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	Two by two matrix, first row one two, second row three four
$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$	m by n matrix, first row a sub one one, a sub one two, dot dot dot, a sub one n second row a sub two one, a sub two two, dot dot dot, a sub two n dot dot dot last row a sub m one, a sub m two, dot dot dot a sub m n
$a_{i,j}$	Lowercase (small) a sub i comma j
$a_{i,j+1}$	Lowercase a double subscript i comma j plus one
$a_{i,j-1}$	Lowercase a double subscript i comma j minus one



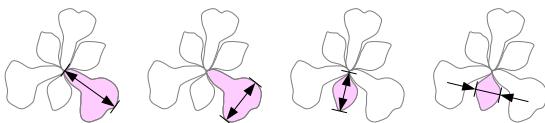
Bk3_Ch1_09.py 利用 `numpy.array()` 定义矩阵，并提取矩阵的某一列、某两列、某一行、某一个位置的具体值。

鸢尾花数据集

绝大多数情况，数据以矩阵形式存储、运算。举个例子，图 17 所示鸢尾花卉数据集，全称为安德森鸢尾花卉数据集 (Anderson's Iris data set)，是植物学家埃德加·安德森 (Edgar Anderson) 在加拿大魁北克加斯帕半岛上的采集的 150 个鸢尾花样本数据。这些数据都属于鸢尾属下的三个亚属。每一类鸢尾花收集了 50 条样本记录，共计 150 条。

图 17 中数据第一列是序号，不算做矩阵元素。但是它告诉我们，鸢尾花数据集有 150 个样本数据，即 $n = 150$ 。紧随其后的是被用作样本定量分析的四个特征——花萼长度 (sepal length)、花萼宽度 (sepal width)、花瓣长度 (petal length) 和花瓣宽度 (petal width)。

图 17 表格最后一列为鸢尾花分类，即标签 (label)。三个标签分别为——山鸢尾 (setosa)、变色鸢尾 (versicolor) 和维吉尼亚鸢尾 (virginica)。最后一列标签算在内，矩阵有 5 列，即 $D = 5$ 。



Index	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Species C
1	5.1	3.5	1.4	0.2	Setosa C_1
2	4.9	3	1.4	0.2	
3	4.7	3.2	1.3	0.2	
...	
49	5.3	3.7	1.5	0.2	
50	5	3.3	1.4	0.2	
51	7	3.2	4.7	1.4	
52	6.4	3.2	4.5	1.5	
53	6.9	3.1	4.9	1.5	
...	
99	5.1	2.5	3	1.1	
100	5.7	2.8	4.1	1.3	
101	6.3	3.3	6	2.5	
102	5.8	2.7	5.1	1.9	
103	7.1	3	5.9	2.1	
...	
149	6.2	3.4	5.4	2.3	
150	5.9	3	5.1	1.8	

图 17. 鸢尾花数据表格，单位为厘米 (cm)

这个 150×5 的矩阵的每一列，即列向量，为鸢尾花一个特征的样本数据。矩阵的每一行，即行向量，代表某一个特定的鸢尾花样本。

鸢尾花数据集可以说是本系列丛书最重要的数据集，没有之一。我们将用各种数学工具从各种视角分析鸢尾花数据。图 18 给出了几个例子，本系列丛书会陪着大家理解其中每幅图的含义。

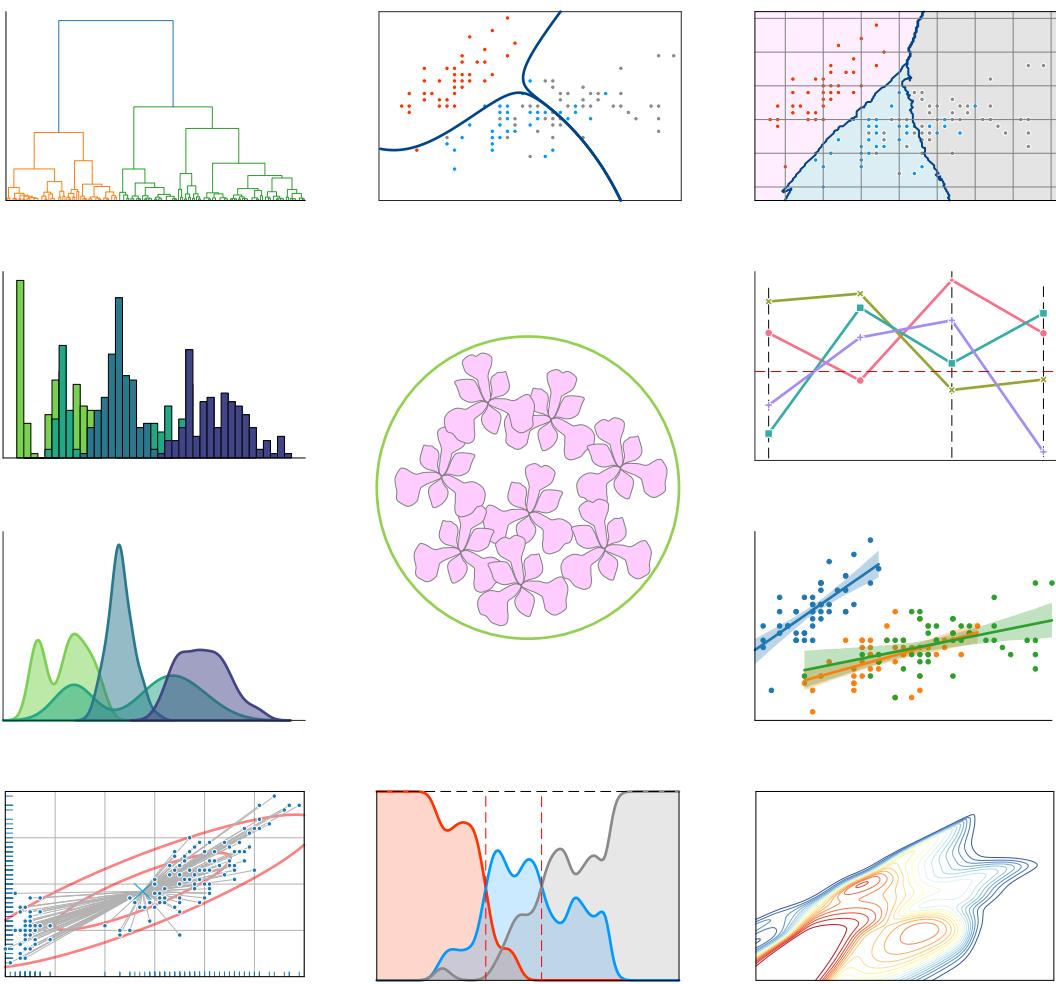


图 18. 用各种概率统计、数据科学、机器学习工具分析鸢尾花数据

矩阵形状记号

大部分数学教科书表达矩阵形状时采用 $m \times n$ ；本系列丛书表达矩阵形状时，一般用 $n \times D$ ， n 表达行数， D 表达列数。

采用 $n \times D$ 这种记号有几方面的考虑。

首先 m 和 n 这两个字母区分度不高。两者长相类似，而且发音相近，这让初学者辨别行、列时很大惑。而 n 和 D ，一个小写字母，一个大写字母，且发音有显著区别，很容易辨识。

此外，处理数据时大家会发现，比如 `pandas.DataFrame` 定义的数据帧中，列代表特征，比如性别、身高、体重、年龄等等；行一般代表样本，比如小张、小王、小姜等等。而统计中，一般用 n 代表样本数，因此决定用 n 来代表矩阵的行数。字母 D 取自 dimension（维度）的首字母，方便记忆。

本系列丛书横跨代数、线性代数、概率统计几个板块， $n \times D$ 这种记法方便大家把矩阵运算和统计知识联系起来。

本书编写之初，也有考虑用 feature (特征) 的首字母 F 来表达矩阵的列数，但最终放弃。一方面，是因为丛书后续会用 F 代表一些特定函数；另一方面， n 和 F 的发音区分度不如 n 和 D 那么高。

基于以上考虑，本系列丛书后续在表达样本数据矩阵形状时都会默认采用 $n \times D$ 这一记法，除非特别说明。

1.6 矩阵：一排列向量，或一组行向量

矩阵可以看做是，若干列向量左右排列，或者若干行向量上下叠放。比如，形状为 2×3 的矩阵可以看成是 3 个列向量左右排列，也可以看成是 2 个行向量上下叠放：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} [1] \\ [4] \end{bmatrix} \begin{bmatrix} [2] \\ [5] \end{bmatrix} \begin{bmatrix} [3] \\ [6] \end{bmatrix} = \begin{bmatrix} [1 & 2 & 3] \\ [4 & 5 & 6] \end{bmatrix} \quad (11)$$

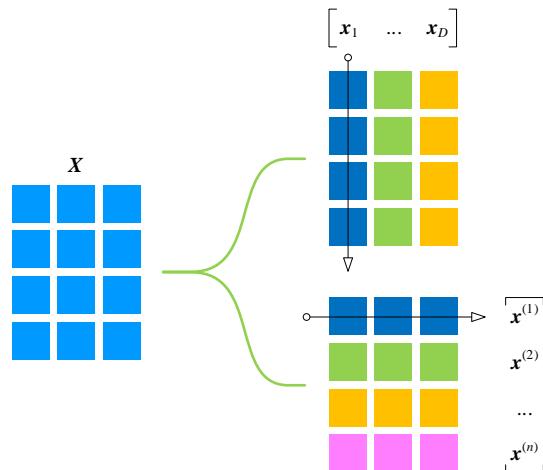


图 19. 矩阵可以分解成一系列行向量或列向量

一般情况，如图 19 所示，形状为 $n \times D$ 的矩阵 X ，可以写成 D 个左右排列的列向量：

$$X_{n \times D} = [x_1 \quad x_2 \quad \cdots \quad x_D] \quad (12)$$

X 也可以写成 n 个行向量上下叠放：

$$X_{n \times D} = \begin{bmatrix} x^{(1)} \\ x^{(2)} \\ \vdots \\ x^{(n)} \end{bmatrix} \quad (13)$$

→ 实际上，(12) 和 (13) 蕴含着一种重要的思想——**矩阵分块** (block matrix 或 partitioned matrix)。本系列丛书《矩阵力量》一册会详细介绍矩阵分块及相关运算规则。

⚠ 注意，为了区分含序号的列向量和行向量，本系列丛书将列向量的序号写成下角标，比如 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_i, \mathbf{x}_D$ 等；将行向量的序号写成上角标加圆括号，比如 $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(j)}, \mathbf{x}^{(n)}$ 等。列索引一般用 i ，行索引一般用 j 。

矩阵转置

矩阵转置 (matrix transpose) 指的是将矩阵的行列互换得到的新矩阵，比如下例：

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3 \times 2}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}_{2 \times 3} \quad (14)$$

(14) 中， 3×2 矩阵转置得到矩阵的形状为 2×3 。

图 20 为矩阵转置示意图，其中红色线为主对角线 (main diagonal)。

⚠ 再次强调，主对角线是从矩阵第 1 行、第 1 列元素出发向右下方倾斜 45° 斜线。

转置前后，矩阵主对角线元素位置不变，比如 (14) 的 1、4 两个元素。向量转置是矩阵转置的特殊形式。

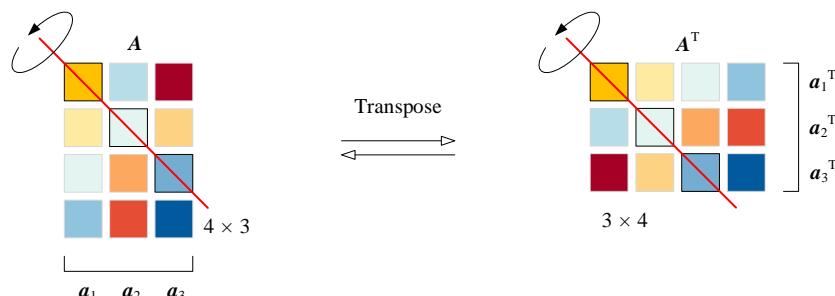


图 20. 矩阵转置

如图 20 所示，将矩阵 A 写成三个列向量左右排列 $[a_1, a_2, a_3]$ ，对 A 转置得到的结果为：

$$A^T = [a_1 \ a_2 \ a_3]^T = \begin{bmatrix} a_1^T \\ a_2^T \\ a_3^T \end{bmatrix} \quad (15)$$

这一点对于转置运算非常重要，再举个具体例子。给定如下矩阵，并将其写成左右排列的列向量：

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \quad (16)$$

(16) 矩阵转置结果为：

$$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \quad \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}^T = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (17)$$

反之，将矩阵 A 写成三个行向量上下叠放，对 A 转置得到的结果为：

$$A^T = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \mathbf{a}^{(3)} \end{bmatrix}^T = \begin{bmatrix} \mathbf{a}^{(1)T} & \mathbf{a}^{(2)T} & \mathbf{a}^{(3)T} \end{bmatrix} \quad (18)$$

请大家根据上式，代入具体值自行完成类似 (17) 的验算。

1.7 矩阵形状：每种形状都有特殊性质和用途

矩阵的一般形状为长方形，但是矩阵还有很多特殊形状。图 21 所示为常见特殊形态矩阵。

很明显，列向量、行向量都是特殊矩阵。

如果列向量的元素都为 1，一般记做 \mathbf{I} 。 \mathbf{I} 被称作全 1 列向量，简称**全 1 向量** (all-ones vector)。

如果列向量的元素都是 0，这种列向量叫做**零向量** (zero vector)，记做 $\mathbf{0}$ 。

行数和列数相同的矩阵叫**方阵** (square matrix)，比如 2×2 矩阵。

对角矩阵 (diagonal matrix) 一般是一个主对角线之外的元素皆为 0 的方阵。

单位矩阵 (identity matrix) 是主对角线元素为 1 其余元素均为 0 的方阵，记做 \mathbf{I} 。

对称矩阵 (symmetric matrix) 是元素相对于主对角线轴对称的方阵。

零矩阵 (null matrix) 一般指所有元素皆为 0 的方阵，记做 \mathbf{O} 。



每一种特殊形状矩阵在线性代数舞台上都扮演特殊的角色，本系列丛书会慢慢讲给大家。

⚠ 值得注意的是，大家会在本系列丛书《矩阵力量》一本中发现，对角矩阵也可以不是方阵。此外，零矩阵也未必都是方阵。

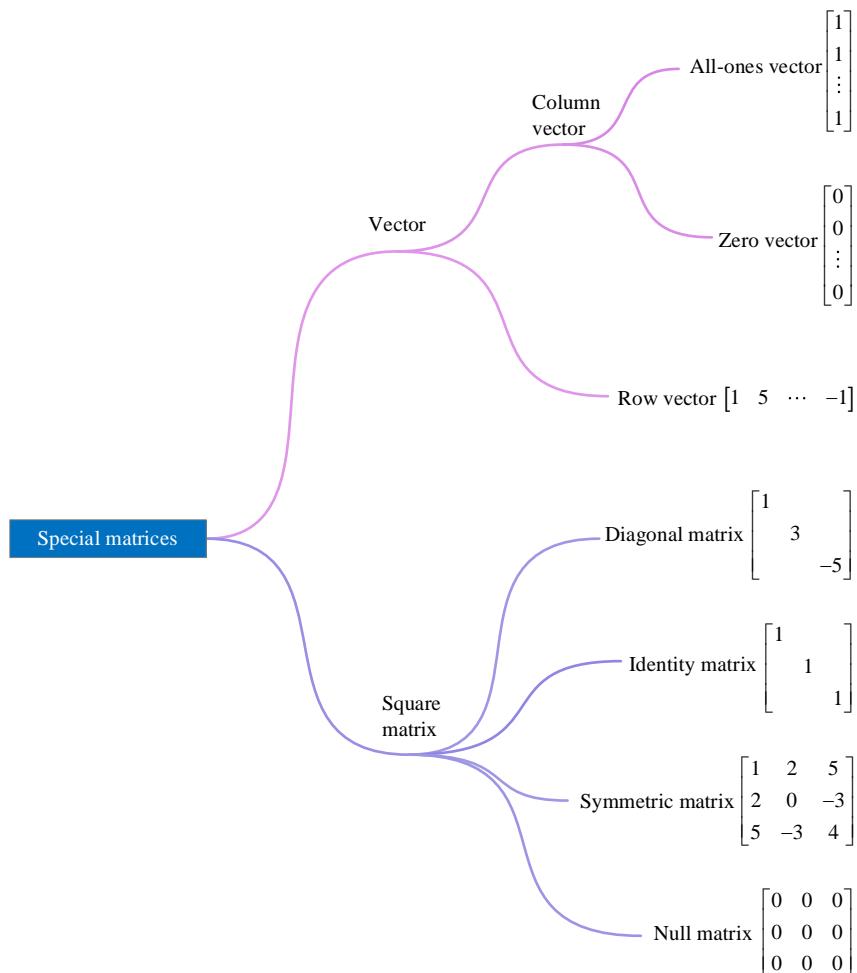


图 21. 常见特殊形态矩阵

1.8 矩阵加减：形状相同，对应位置，批量加减

本节介绍矩阵加减法。矩阵相加减就是批量化完成若干加减运算。矩阵加减可以视作四则运算中加减的高阶版本。

上一节说过，行向量和列向量是特殊的矩阵。两个等长的行向量相加，为对应元素相加，得到还是一个行向量，比如下例：

$$[1 \ 2 \ 3] + [4 \ 5 \ 6] = [1+4 \ 2+5 \ 3+6] = [5 \ 7 \ 9] \quad (19)$$

同理，两个等长行向量相减，就是对应元素相减，得到的也是相同长度行向量：

$$[1 \ 2 \ 3] - [4 \ 5 \ 6] = [1-4 \ 2-5 \ 3-6] = [-3 \ -3 \ -3] \quad (20)$$

(19) 和 (20) 相当于一次性批量完成了三个加减法运算。⚠ 注意，两个矩阵能够完成加减运算的前提——形状相同。

同理，两个等长的列向量相加，得到仍然是一个列向量：

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \\ 9 \end{bmatrix} \quad (21)$$

图 22 所示为两个数字相加的示意图。而图 23 所示为向量求和。



图 22. 数字求和

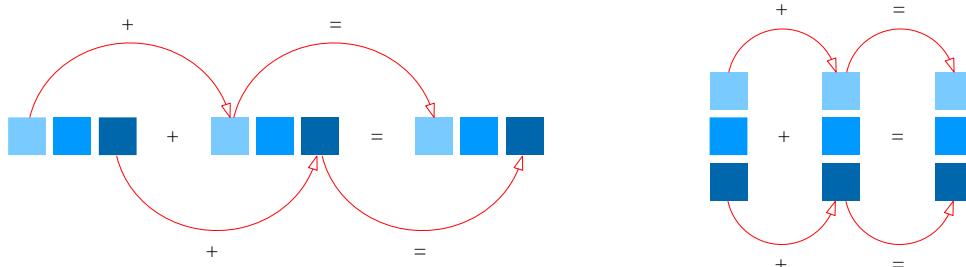


图 23. 向量求和



Bk3_Ch1_10.py 展示了四种计算行向量相加的方式。这四种方法中，当然首推用 NumPy。

矩阵加减

形状相同的两个矩阵相加的结果还是矩阵。运算规则为，对应位置元素相加，形状不变：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 1+1 & 2+0 & 3+0 \\ 4+0 & 5+1 & 6+0 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 2 & 2 & 3 \\ 4 & 6 & 6 \end{bmatrix}_{2 \times 3} \quad (22)$$

两个矩阵相减的运算原理完全相同，比如下例：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 1-1 & 2 & 3 \\ 4 & 5-1 & 6 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 4 & 6 \end{bmatrix}_{2 \times 3} \quad (23)$$



Bk3_Ch1_11.py 用 for 循环完成矩阵加法运算，这种做法绝不推荐！Bk3_Ch1_12.py 利用 NumPy 完成矩阵加法。

⚠ 注意，用 for 循环来解决矩阵相加是最费力的办法，比如 Bk3_Ch1_11.py 代码给出的例子。为了让代码运算效率提高，常用的方法之一就是——向量化 (vectorize)。也就是说，尽量采用向量/矩阵运算，避免循环。



数字和数学是抽象的，它们是人类总结的规律，是人类思想的产物。

“双兔傍地走”中的“双”就是 2；2 这个数字对人类有意义，对兔子自身没有意义；两只兔子自顾自的玩耍，一旁暗中观察的某个人在大脑中思维活动抽象产生了“双”这个数字概念，而且要进一步“辨雄雌”。

试想一个没人类的自然界。那里，天地始交，万物并秀，山川巍峨，江河奔涌，雨润如酥，暗香浮动，芳草萋萋，鹿鸣呦呦，鹰击长空，鱼翔浅底。

试问，这般香格里拉的梦幻世界和数字有什么关系？

然而，本书的读者很快就知道，微观世界中，自然界中，天体运行中，人类通过几千年的观察研究发现，数字、数学规律无处不在；可惜天意从来高难问，大部分规律不为人所知罢了。

这让我们不禁追问，可感知世界万物是否仅仅是表象？世界万物创造动力和支配能量，是否就是数字和数学？我们听到的、看到的、触摸到的，是否都是数字化的，虚拟化的？整个物质世界仅仅是某个巨型计算机模拟的产物？这些问题让我不寒而栗。

老子说，“大道无形，生育天地；大道无情，运行日月。”老子是否真的参透了世间万物，它口中的“大道”是否就是数字、数学规律？

Multiplication and Division

2 乘除

从九九乘法到矩阵乘法



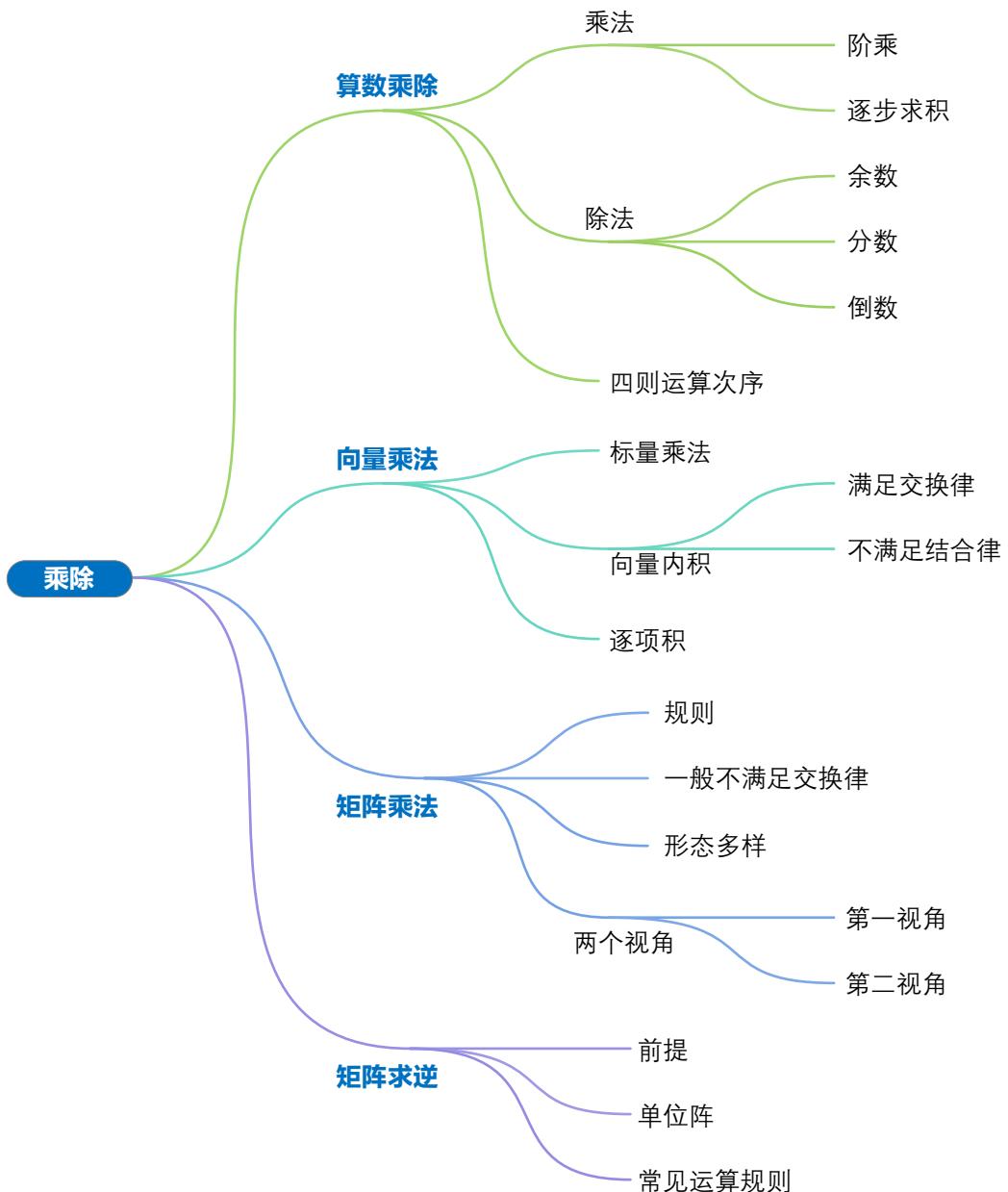
大自然只使用最长的线来编织她的图景；因此，每根织线都能洞见整个大自然的锦绣图景。

Nature uses only the longest threads to weave her patterns, so that each small piece of her fabric reveals the organization of the entire tapestry.

—— 理查德·费曼 (Richard P. Feynman) | 美国理论物理学家 | 1918 ~ 1988



- ◀ `input()` 函数接受一个标准输入数据，返回为字符串 `str` 类型
- ◀ `int()` 将输入转化为整数
- ◀ `math.factorial()` 计算阶乘
- ◀ `numpy.cumprod()` 计算累计乘积
- ◀ `numpy.inner()` 计算行向量的内积，函数输入为两个列向量时得到的结果为张量积
- ◀ `numpy.linalg.inv()` 计算方阵的逆
- ◀ `numpy.linspace()` 在指定的间隔内，返回固定步长的数组
- ◀ `numpy.math.factorial()` 计算阶乘
- ◀ `numpy.random.seed()` 固定随机数发生器种子
- ◀ `numpy.random.uniform()` 产生满足连续均匀分布的随机数
- ◀ `numpy.sum()` 求和
- ◀ `scipy.special.factorial()` 计算阶乘
- ◀ `seaborn.heatmap()` 绘制热图



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

2.1 算术乘除：先乘除，后加减，括号内先算

本节回顾算术乘除法。

乘法

乘法 (multiplication) 算式等号左端是**被乘数** (multiplicand) 和**乘数** (multiplier)，右端是**乘积** (product)。乘法运算符读作**乘** (times 或 multiplied by)。**乘法表** (multiplication table 或 times table) 是数字乘法运算的基础。



图 1. 乘法运算

图 2 所示为数轴上可视化 $2 \times 3 = 6$ 。

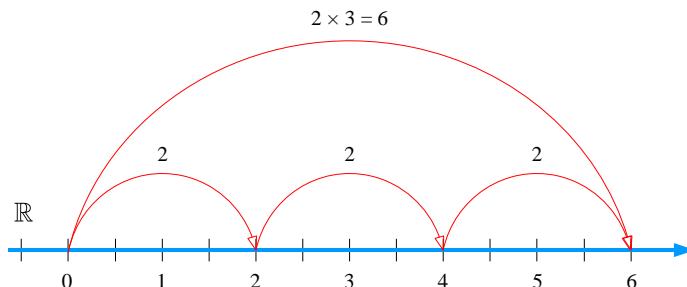


图 2. $2 \times 3 = 6$ 在数轴上的可视化

介绍几个常用乘法符号。乘法符号 \times 用于数字相乘，一般不用在两个变量相乘。在线性代数中， \times 则表示**叉乘** (cross product 或 vector product)，完全另外一回事。

在代数中，两个变量 a 和 b 相乘，可以写成 ab ；这种记法被称作**隐含乘法** (implied multiplication)。 ab 也可以写成 $a \cdot b$ 。

通常，圆点 \cdot 不用在数字相乘，因为它容易和**小数点** (decimal point) 混淆。线性代数中， $a \cdot b$ 表示 a 和 b 两个向量的**标量积** (scalar product)，这是本章后续要介绍的内容。

多提一嘴，乘法计算时，请大家多留意数值单位。举个例子，正方形的边长为 1 m，其面积数值可以通过乘法运算 $1 \times 1 = 1$ 获得，而结果单位为平方米 m²。有一些数值本身**无单位**

(unitless)，比如个数、z 分数。z 分数也叫**标准分数** (standard score)，是概率统计中的一个概念，z 分数是一个数与平均数的差再除以标准差的结果。

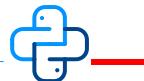


本系列丛书会在《概率统计》一册详细介绍 z 分数这个概念。

表 1 给出和乘法相关的常用英文表达。

表 1. 乘法相关英文表达

数学表达	英文表达
$2 \times 3 = 6$	Two times three equals six. Two multiplied by three equals six. Two cross three equals six. The product of two and three is six. If you multiply two by three, you get six.
$a \cdot b = c$	a times b equals c . a multiplied by b equals c . a dot b equals c . The product of a and b is c .



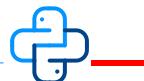
Bk3_Ch2_01.py 完成两个数乘法。Python 中两个数字相乘用 * (asterisk 或 star)。

阶乘

某个正整数的**阶乘** (factorial) 是所有小于及等于该数的正整数的积。比如，5 的阶乘记做 $5!$ ，对应的运算为：

$$5! = 5 \times 4 \times 3 \times 2 \times 1 \quad (1)$$

特别地，定义 0 的阶乘为 $0! = 1$ 。本书有两个重要的数学概念需要用到阶乘——排列组合和泰勒展开。



Python 中可以用 `math.factorial()`、`scipy.special.factorial()`、`numpy.math.factorial()` 计算阶乘。为了帮助大家理解，Bk3_Ch2_02.py 自定义函数求解阶乘。

累计乘积

对于一组数字，**累计乘积** (cumulative product) 也叫**累积乘积**，得到的结果不仅仅是一个乘积，而是从左向右每乘一个数值得到的分步结果。比如，自然数 1 到 10 求累计求积结果为：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

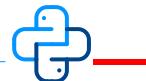
本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800 (2)

(2) 对应的累计乘积过程如下：

$$\begin{array}{r}
 & \overbrace{1 \times 2 \times 3 \times 4 \times 5 \times 6 = 720} \\
 & \overbrace{1 \times 2 \times 3 \times 4 \times 5 = 120} \\
 & \overbrace{1 \times 2 \times 3 \times 4 = 24} \\
 & \overbrace{1 \times 2 \times 3 = 6} \\
 & 1 \times 2 = 2 \\
 1 & \\
 1, 2, 3, 4, 5, 6, 7, 8, \dots
 \end{array} \quad (3)$$



Bk3_Ch2_03.py 利用 numpy.linspace(1, 10, 10) 产生 1 ~ 10 这 10 个自然数，然后利用 numpy.cumprod() 函数来求累计乘积。此外，请大家自行研究如何使用 numpy.arange()，并用这个函数生成 1 ~ 10。

除法

除法 (division) 是**乘法的逆运算** (reverse operation of multiplication)。被**除数** (dividend 或 numerator) **除以** (over 或 divided by) **除数** (divisor 或 denominator) 得到**商** (quotient)。

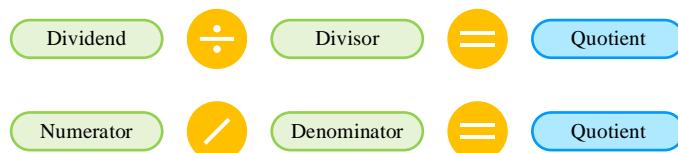
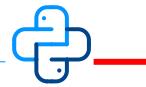


图 3. 除法运算

除法运算有时可以**除尽** (divisible)，比如，**6 可以被 3 除尽** (six is divisible by three)。除法有时也得到**余数** (remainder)，比如 7 除 2 余 1。此外，除法的结果一般用**分数** (fraction) 或**小数** (decimal) 来表达。

表 2. 除法英文表达

数学表达	英文表达
$6 \div 3 = 2$	Six divided by three equals two. If you divide six by three you get two.
$7 \div 2 = 3R1$	Seven over two is three and the remainder is one. Seven divided by two equals three with a remainder of one. Seven divided by two equals three and the remainder is one.



Bk3_Ch2_04.py 完成两个数的除法运算，除法运算符为正斜杠 /。

Bk3_Ch2_05.py 介绍如何求余，求余数的运算符为 %。

分数

最常见的**分数** (fraction) 是**普通分数** (common fraction 或 simple fraction)，由**分母** (denominator) 和**分子** (numerator) 组成，分隔两者的是**分数线** (fraction bar) 或**正斜杠** (forward slash) /。

非零整数 (nonzero integer) a 的**倒数** (reciprocal) 是 $1/a$ 。分数 b/a 的倒数是 a/b 。

表 3 总结常用分数英文表达。

表 3. 分数相关英文表达

数学表达	英文表达
$\frac{1}{2}$, 1/2	One half A half One over two
1:2	One to two
$-\frac{3}{2}$	Minus three-halves Negative three-halves
$\frac{1}{3}$, 1/3	One over three One third
$\frac{1}{4}$, 1/4	One over four One fourth One quarter One divided by four
$1\frac{1}{4}$	One and one fourth
1/5	One fifth
3/5	Three fifths
$\frac{1}{n}$, 1/n	One over n
$\frac{a}{b}$, a/b	a over b a divided by b The ratio of a to b The numerator is a while the denominator is b

2.2 向量乘法：标量乘法、向量内积、逐项积

这一节介绍三种重要的向量乘法：(1) **标量乘法** (scalar multiplication); (2) **向量内积** (inner product); (3) **逐项积** (piecewise product)。

标量乘法

标量乘法运算中，标量乘向量结果还是向量，相当于缩放。

标量乘法运算规则很简单，向量 a 乘以 k , a 的每一个元素均与 k 相乘，比如下例标量 2 乘行向量 $[1, 2, 3]$:

$$2 \times [1 \ 2 \ 3] = [2 \times 1 \ 2 \times 2 \ 2 \times 3] = [2 \ 4 \ 6] \quad (4)$$

再如，标量乘列向量：

$$2 \times \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \times 1 \\ 2 \times 2 \\ 2 \times 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix} \quad (5)$$

图 4 所示为标量乘法示意图。



图 4. 标量乘法

同理，标量 k 乘矩阵 A 结果是 k 与矩阵 A 每一个元素相乘，比如下例：

$$2 \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 2 \times 1 & 2 \times 2 & 2 \times 3 \\ 2 \times 4 & 2 \times 5 & 2 \times 6 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}_{2 \times 3} \quad (6)$$



Bk3_Ch2_06.py 完成向量和矩阵标量乘法。

向量内积

向量内积 (inner product) 结果为标量。向量内积又叫**标量积** (scalar product) 或**点积** (dot product)。

向量内积的运算规则是，两个形状相同向量，对应位置元素一一相乘，再求和。比如下例计算两个行向量内积：

$$[1 \ 2 \ 3] \cdot [4 \ 3 \ 2] = 1 \times 4 + 2 \times 3 + 3 \times 2 = 4 + 6 + 6 = 16 \quad (7)$$

计算两个列向量内积，比如：

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = 1 \times (-1) + 2 \times 0 + 3 \times 1 = -1 + 0 + 3 = 2 \quad (8)$$

图 5 所示为向量内积规则示意图。

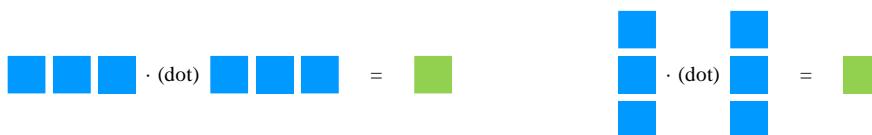


图 5. 向量内积示意图

显然，向量内积满足**交换律** (commutative)，即：

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a} \quad (9)$$

向量内积**对向量加法满足分配律** (distributive over vector addition)：

$$\mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c} \quad (10)$$

显然，向量内积不满足**结合律** (associative)：

$$(\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{c} \neq \mathbf{a} \cdot (\mathbf{b} \cdot \mathbf{c}) \quad (11)$$



Bk3_Ch2_07.py 代码用 `numpy.inner()` 计算行向量的内积；但是，`numpy.inner()` 函数输入为两个列向量时得到的结果为**张量积** (tensor product)。



机器学习和深度学习中，张量积是非常重要的向量运算，本系列丛书将在《矩阵力量》一册会详细介绍。



下面举几个例子，让大家管窥标量积的用途。

给定如下五个数字，

$$1, 2, 3, 4, 5 \quad (12)$$

这五个数字求和，可以用如下标量积计算得到：

$$[1 \ 2 \ 3 \ 4 \ 5]^T \cdot [1 \ 1 \ 1 \ 1 \ 1]^T = 1 \times 1 + 2 \times 1 + 3 \times 1 + 4 \times 1 + 5 \times 1 = 15 \quad (13)$$

前文提过， $[1, 1, 1, 1, 1]^T$ 叫做全 1 向量。

这五个数字的平均值，也可以通过标量积得到：

$$[1 \ 2 \ 3 \ 4 \ 5]^T \cdot [1/5 \ 1/5 \ 1/5 \ 1/5 \ 1/5]^T = 1 \times 1/5 + 2 \times 1/5 + 3 \times 1/5 + 4 \times 1/5 + 5 \times 1/5 = 3 \quad (14)$$

计算五个数字的平方和：

$$[1 \ 2 \ 3 \ 4 \ 5]^T \cdot [1 \ 2 \ 3 \ 4 \ 5]^T = 1 \times 1 + 2 \times 2 + 3 \times 3 + 4 \times 4 + 5 \times 5 = 55 \quad (15)$$

此外，标量积还有重要的几何意义。本书后续将介绍这方面内容。

逐项积

逐项积 (piecewise product)，也叫**阿达玛乘积** (Hadamard product)。两个形状相同向量的逐项积为对应位置元素分别相乘，结果为相同形状的向量。

逐项积的运算符为 \odot 。逐项积相当于算术乘法的批量运算。

举个例子，两个行向量逐项积：

$$[1 \ 2 \ 3] \odot [4 \ 5 \ 6] = [1 \times 4 \ 2 \times 5 \ 3 \times 6] = [4 \ 10 \ 18] \quad (16)$$

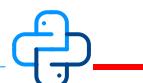
图 6 所示为向量逐项积运算示意图。



图 6. 向量逐项积

同理，两个矩阵逐项积的运算前提——矩阵形状相同。矩阵逐项积运算规则为对应元素相乘，结果形状不变：

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}_{2 \times 3} \odot \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 1 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 1 \times 1 & 2 \times 2 & 3 \times 3 \\ 4 \times (-1) & 5 \times 0 & 6 \times 1 \end{bmatrix}_{2 \times 3} = \begin{bmatrix} 1 & 4 & 9 \\ -4 & 0 & 6 \end{bmatrix}_{2 \times 3} \quad (17)$$



Python 中，对于 `numpy.array()` 定义的形状相同的向量或矩阵，逐项积可以通过 `*` 计算得到。请大家参考 `Bk3_Ch2_08.py`。

2.3 矩阵乘法：最重要的线性代数运算规则

矩阵乘法是最重要线性代数运算，没有之一——这句话并不夸张。

矩阵乘法规则可以视作算术“九九乘法表”的进阶版。

矩阵乘法规则

A 和 B 两个矩阵相乘的前提是矩阵 A 的列数和矩阵 B 的行数相同。 A 和 B 的乘积一般写作 AB 。

⚠ 注意， A 在左边， B 在右边，不能随意改变顺序。也就是说，矩阵乘法一般情况下不满足交换律，即 $AB \neq BA$ 。

A 和 B 两个矩阵相乘 AB 读作“matrix boldface capital A times matrix boldface capital B ”或“the matrix product boldface capital A and boldface capital B ”。

NumPy 中，两个矩阵相乘的运算符为 @，本系列丛书一部分矩阵乘法也会采用 @。比如， AB 也记做 $A @ B$ ：

$$C_{m \times n} = A_{m \times p} B_{p \times n} = A_{m \times p} @ B_{p \times n} \quad (18)$$

如图 7 所示，矩阵 A 的形状为 m 行、 p 列，矩阵 B 的形状为 p 行、 n 列。

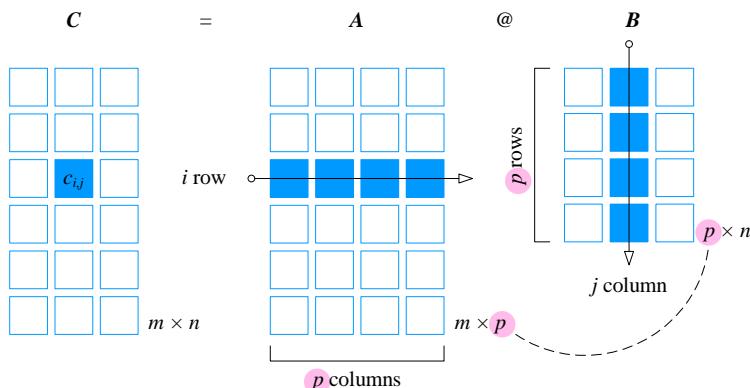


图 7. 矩阵乘法规则

如图 7 所示， A 和 B 相乘得到矩阵 C ， C 的形状为 m 行、 n 列，相当于消去了 p 。

再次强调，矩阵乘法不满足交换律。也就是说，一般情况下式不成立：

$$A_{m \times p} B_{p \times n} \neq B_{p \times n} A_{m \times p} \quad (19)$$

首先， \mathbf{B} 的列数和 \mathbf{A} 的行数很可能不匹配。即便 $m = n$ ，也就是 \mathbf{B} 的列数等于 \mathbf{A} 的行数， \mathbf{BA} 结果很可能不等于 \mathbf{AB} 。

两个 2×2 矩阵相乘

下面，用两个 2×2 矩阵相乘讲解矩阵乘法运算规则。

下式中，矩阵 \mathbf{A} 和 \mathbf{B} 相乘结果为矩阵 \mathbf{C} ：

$$\mathbf{C} = \mathbf{AB} = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_A @ \underbrace{\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}}_B = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_A @ \underbrace{\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}}_B \quad (20)$$

图 8 所示为两个 2×2 矩阵相乘如何得到矩阵 \mathbf{C} 的每一个元素。

矩阵 \mathbf{A} 的第一行元素和矩阵 \mathbf{B} 第一列对应元素分别相乘，再相加，结果为矩阵 \mathbf{C} 的第一行、第一列元素 $c_{1,1}$ 。

矩阵 \mathbf{A} 的第一行元素和矩阵 \mathbf{B} 第二列对应元素分别相乘，再相加，得到 $c_{1,2}$ 。

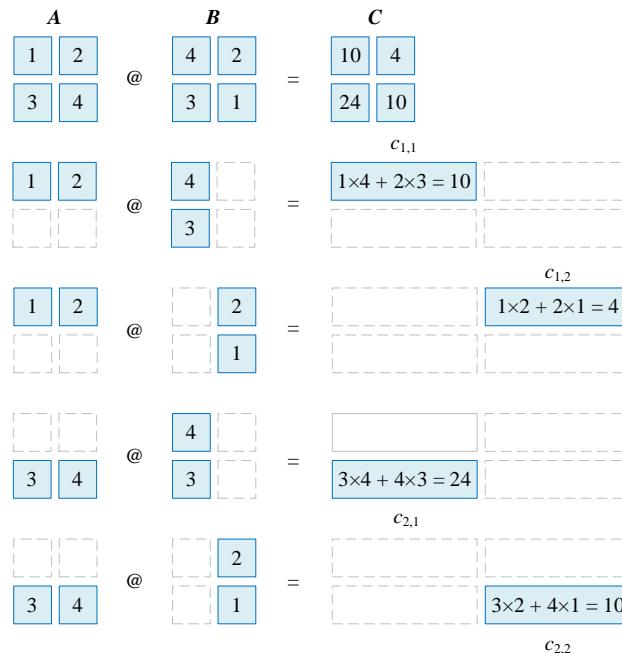
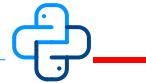


图 8. 矩阵乘法规则，两个 2×2 矩阵相乘为例

同理，依次获得矩阵 \mathbf{C} 的 $c_{2,1}$ 和 $c_{2,2}$ 两个元素。

总结来说， \mathbf{A} 和 \mathbf{B} 乘积 \mathbf{C} 的第 i 行第 j 列的元素 c_{ij} 等于矩阵 \mathbf{A} 的第 i 行的元素与矩阵 \mathbf{B} 的第 j 列对应元素乘积再求和。

⚠ 注意，这个矩阵运算规则既是一种发明创造，也是一种约定成俗。也就是说，这种乘法规则在被法国数学家雅克·菲利普·玛丽·比内 (Jacques Philippe Marie Binet, 1786 ~ 1856) 提出之后，在长期的数学实践中被广为接受。矩阵乘法可谓“成人版九九乘法表”。就像大家儿时背诵九九乘法表时，这里建议大家先把矩阵乘法规则背下来。熟能生巧，慢慢地大家就会通过不断学习认识到这个乘法规则的精妙之处。



Bk3_Ch2_09.py 展示如何完成 (20) 矩阵乘法运算。

矩阵乘法形态

图 9 给出了常见的多种矩阵乘法形态，每一种形态对应一类线性代数问题。图 9 特别高亮矩阵乘法中左侧矩阵的“列”和右侧矩阵的“行”。本系列丛书《矩阵力量》一册将会详细介绍图 9 每一种乘法形态。

这里格外提醒大家，初学者对矩阵乘法会产生一种错误印象，认为这些千奇百怪的矩阵乘法形态就是“奇技淫巧”。这是极其错误的想法！在不断学习中，大家会逐渐领略到每种矩阵乘法形态的力量所在。

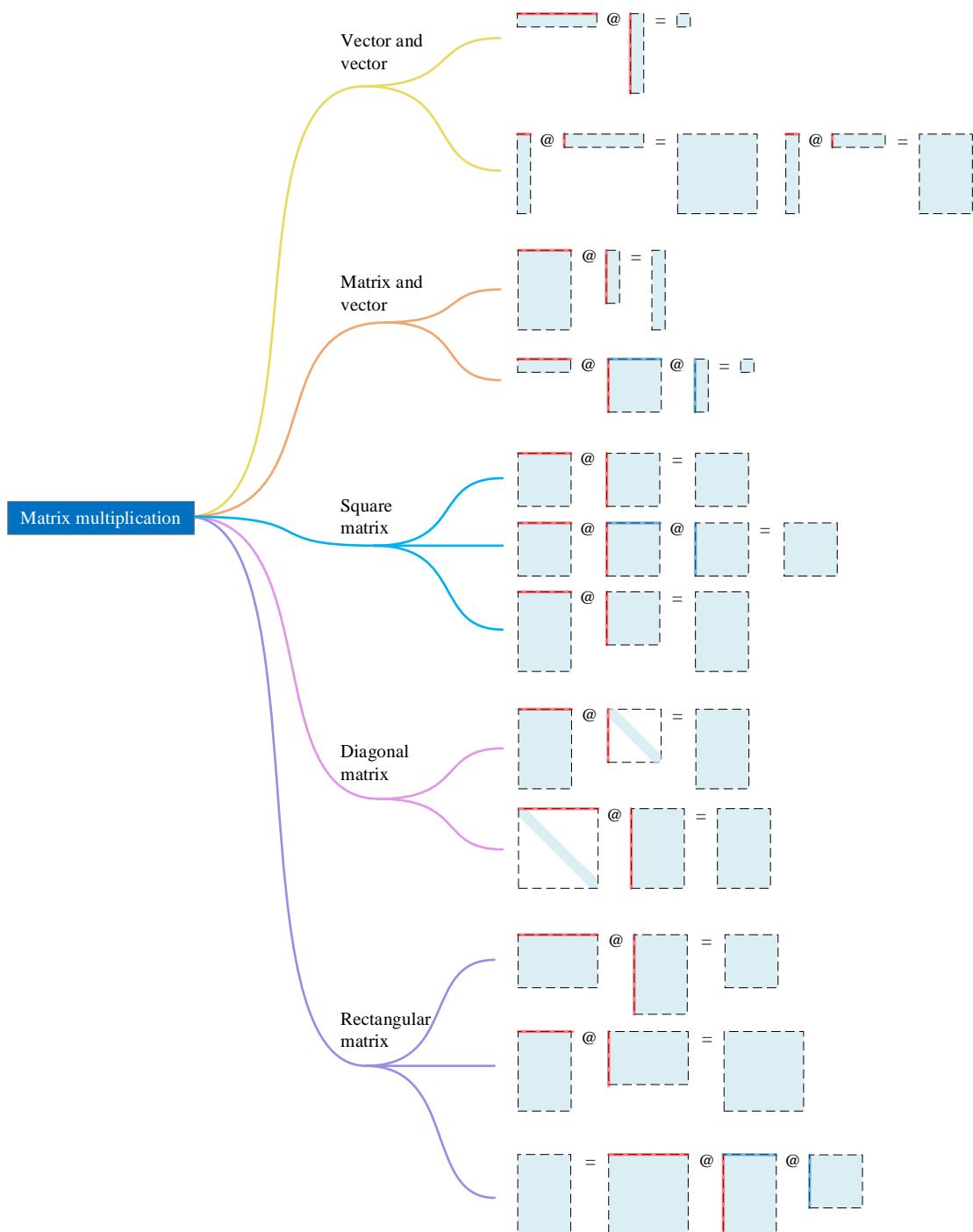


图 9. 矩阵乘法形态多样性

两个向量相乘

本节最后着重讲一下图9最顶上两种向量乘积。这两种特殊形态的矩阵乘法正是理解矩阵乘法规则的两个重要视角。

向量 a 和 b 为等长列向量， a 转置 (a^T) 乘 b 为标量，等价于 a 和 b 的标量积：

$$a^T b = a \cdot b \quad (21)$$

举个例子：

$$a^T b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_{3 \times 1}^T @ \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}_{3 \times 1} = [1 \ 2 \ 3]_{1 \times 3} @ \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}_{3 \times 1} = [1 \ 2 \ 3] \cdot [4 \ 3 \ 2] = 16 \quad (22)$$

列向量 a 乘 b 转置 (b^T)，乘积结果 ab^T 为方阵，也就是行数和列数相同的矩阵：

$$ab^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_{3 \times 1} @ \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}_{3 \times 1}^T = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}_{3 \times 1} @ [4 \ 3 \ 2]_{1 \times 3} = \begin{bmatrix} 4 & 3 & 2 \\ 8 & 6 & 4 \\ 12 & 9 & 6 \end{bmatrix}_{3 \times 3} \quad (23)$$

如果 a 和 b 分别为，请大家自行计算 ab^T 的结果：

$$a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}_{2 \times 1}, \quad b = \begin{bmatrix} 4 \\ 3 \\ 2 \end{bmatrix}_{3 \times 1} \quad (24)$$

⚠ 再次强调，使用 `numpy.array()` 构造向量时，`np.array([1,2])` 构造的是一维数组，不能算是矩阵。而 `np.array([[1,2]])` 构造得到的是 1×2 行向量，是一个特殊矩阵。

→ `np.array([[1],[2]])` 构造的是一个 2×1 列向量，也是个矩阵。本系列丛书会在《矩阵力量》一册介绍更多构造行向量和列向量的办法。

2.4 矩阵乘法第一视角

这一节探讨矩阵乘法的第一视角。

两个 2×2 矩阵相乘

上一节最后介绍， a 和 b 均是形状为 $n \times 1$ 的列向量， $a^T b$ 结果标量，相当于标量积 $a \cdot b$ 。我们可以把 (20) 中 A 写成两个行向量 $a^{(1)}$ 和 $a^{(2)}$ ，把 B 写成两个列向量 b_1 和 b_2 ，即，

$$A = \begin{bmatrix} [1 \ 2] \\ [3 \ 4] \end{bmatrix}_{2 \times 2}, \quad B = \begin{bmatrix} [4] & [2] \\ [3] & [1] \end{bmatrix}_{2 \times 2} \quad (25)$$

这样 AB 矩阵乘积可以写成：

$$\mathbf{A} @ \mathbf{B} = \underbrace{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}}_A @ \underbrace{\begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}}_{\mathbf{B}} = \begin{bmatrix} [1 \ 2] @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} & [1 \ 2] @ \begin{bmatrix} 2 \\ 1 \end{bmatrix} \\ [3 \ 4] @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} & [3 \ 4] @ \begin{bmatrix} 2 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 10 & 4 \\ 24 & 10 \end{bmatrix} \quad (26)$$

也就是说，将位于矩阵乘法左侧的 \mathbf{A} 写成行向量，右侧的 \mathbf{B} 写成列向量。然后，行向量和列向量逐步相乘，得到乘积每个位置的元素。

用符号代替具体数字，(26) 可以写成：

$$\begin{aligned} \mathbf{A} @ \mathbf{B} &= \begin{bmatrix} [a_{1,1} & a_{1,2}]_{1 \times 2} \\ [a_{2,1} & a_{2,2}]_{1 \times 2} \end{bmatrix} \begin{bmatrix} [b_{1,1}]_{2 \times 1} & [b_{1,2}]_{2 \times 1} \\ [b_{2,1}]_{2 \times 1} & [b_{2,2}]_{2 \times 1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \end{bmatrix}_{2 \times 1} [\mathbf{b}_1 \quad \mathbf{b}_2]_{1 \times 2} = \begin{bmatrix} \mathbf{a}^{(1)} \mathbf{b}_1 & \mathbf{a}^{(1)} \mathbf{b}_2 \\ \mathbf{a}^{(2)} \mathbf{b}_1 & \mathbf{a}^{(2)} \mathbf{b}_2 \end{bmatrix}_{2 \times 2} = \begin{bmatrix} (\mathbf{a}^{(1)})^T \cdot \mathbf{b}_1 & (\mathbf{a}^{(1)})^T \cdot \mathbf{b}_2 \\ (\mathbf{a}^{(2)})^T \cdot \mathbf{b}_1 & (\mathbf{a}^{(2)})^T \cdot \mathbf{b}_2 \end{bmatrix}_{2 \times 2} \end{aligned} \quad (27)$$

⚠ 再次强调， $\mathbf{a}^{(1)}$ 是行向量， \mathbf{b}_1 是列向量。

(27) 展示的是矩阵乘法的基本视角，它直接体现的是矩阵乘法规则。

更一般情况

矩阵乘积 \mathbf{AB} 中，左侧矩阵 \mathbf{A} 的形状为 $m \times p$ ，将矩阵 \mathbf{A} 写成一组上下叠放的行向量 $\mathbf{a}^{(i)}$ ：

$$\mathbf{A}_{m \times p} = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \vdots \\ \mathbf{a}^{(m)} \end{bmatrix}_{m \times 1} \quad (28)$$

其中，行向量 $\mathbf{a}^{(i)}$ 列数为 p ，即有 p 个元素。

矩阵乘积 \mathbf{AB} 中，右侧矩阵 \mathbf{B} 的形状为 $p \times n$ 列，将矩阵 \mathbf{B} 写成左右排列的列向量：

$$\mathbf{B}_{p \times n} = [\mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_n]_{1 \times n} \quad (29)$$

其中，列向量 \mathbf{b}_j 行数为 p ，也有 p 个元素。

\mathbf{A} 和 \mathbf{B} 相乘，可以展开写成：

$$\mathbf{A}_{m \times p} @ \mathbf{B}_{p \times n} = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \vdots \\ \mathbf{a}^{(m)} \end{bmatrix}_{m \times 1} [\mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_n]_{1 \times n} = \begin{bmatrix} \mathbf{a}^{(1)} \mathbf{b}_1 & \mathbf{a}^{(1)} \mathbf{b}_2 & \cdots & \mathbf{a}^{(1)} \mathbf{b}_n \\ \mathbf{a}^{(2)} \mathbf{b}_1 & \mathbf{a}^{(2)} \mathbf{b}_2 & \cdots & \mathbf{a}^{(2)} \mathbf{b}_n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}^{(m)} \mathbf{b}_1 & \mathbf{a}^{(m)} \mathbf{b}_2 & \cdots & \mathbf{a}^{(m)} \mathbf{b}_n \end{bmatrix}_{m \times n} = \mathbf{C}_{m \times n} \quad (30)$$

热图

图 10 所示为 **热图** (heatmap) 可视化矩阵乘法。

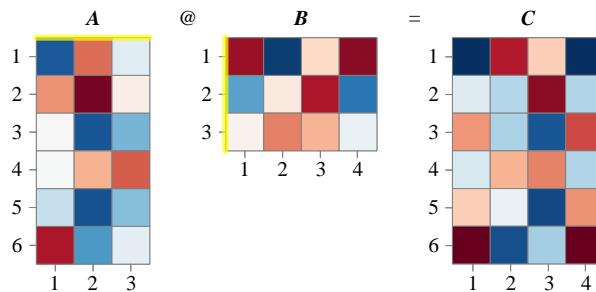


图 10. 矩阵乘法热图展示

具体如图 11 所示， A 中的第 i 行向量 $\mathbf{a}^{(i)}$ 乘以 B 中第 j 列向量 \mathbf{b}_j ，得到标量 $\mathbf{a}^{(i)} \mathbf{b}_j$ ，对应乘积矩阵 C 中第 i 行、第 j 列元素 $c_{i,j}$ ：

$$c_{i,j} = \mathbf{a}^{(i)} \mathbf{b}_j \quad (31)$$

这就是矩阵乘法的第一视角。

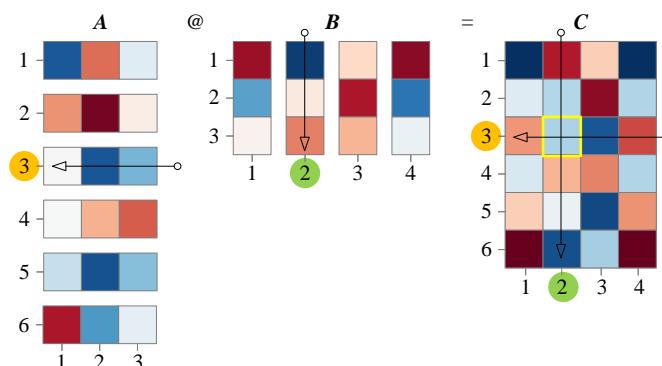
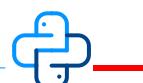


图 11. 矩阵乘法第一视角



代码文件 `Bk3_Ch2_10.py` 中 `Bk3_Ch2_10_A` 部分代码绘制图 10。

代码用 `numpy.random.uniform()` 函数产生满足连续均匀分布的随机数，并用 `seaborn.heatmap()` 绘制热图。热图采用的 colormap 为 '`RdBu_r`'，'`Rd`' 是红色的意思，'`Bu`' 是蓝色，'`_r`' 代表“翻转”。



此外，我们还用 `Streamlit` 制作了展示矩阵乘法运算规则的 App，请大家参考代码文件 `Streamlit_Bk3_Ch2_10.py`。文件中还展示如何使用 `try-except`。

2.5 矩阵乘法第二视角

下面，我们聊一聊矩阵乘法的第二视角。

两个 2×2 矩阵相乘

还是以 (20) 为例， \mathbf{A} 和 \mathbf{B} 相乘，左侧矩阵 \mathbf{A} 写成两个列向量 \mathbf{a}_1 和 \mathbf{a}_2 ，把 \mathbf{B} 写成两个行向量 $\mathbf{b}^{(1)}$ 和 $\mathbf{b}^{(2)}$ ：

$$\mathbf{A} = \begin{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} & \begin{bmatrix} 2 \\ 4 \end{bmatrix} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \begin{bmatrix} 4 & 2 \\ \mathbf{b}^{(1)} & \end{bmatrix} \\ \begin{bmatrix} 3 & 1 \\ \mathbf{b}^{(2)} & \end{bmatrix} \end{bmatrix} \quad (32)$$

这样 \mathbf{AB} 乘积可以展开写成：

$$\mathbf{A} @ \mathbf{B} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} & \begin{bmatrix} 2 \\ 4 \end{bmatrix} \end{bmatrix}}_A @ \underbrace{\begin{bmatrix} \begin{bmatrix} 4 & 2 \\ \mathbf{b}^{(1)} & \end{bmatrix} \\ \begin{bmatrix} 3 & 1 \\ \mathbf{b}^{(2)} & \end{bmatrix} \end{bmatrix}}_B = \begin{bmatrix} 1 \\ 3 \end{bmatrix} @ \underbrace{\begin{bmatrix} 4 & 2 \\ \mathbf{b}^{(1)} & \end{bmatrix}}_{\mathbf{a}_1} + \begin{bmatrix} 2 \\ 4 \end{bmatrix} @ \underbrace{\begin{bmatrix} 3 & 1 \\ \mathbf{b}^{(2)} & \end{bmatrix}}_{\mathbf{a}_2} = \begin{bmatrix} 4 & 2 \\ 12 & 6 \end{bmatrix} + \begin{bmatrix} 6 & 2 \\ 12 & 4 \end{bmatrix} = \begin{bmatrix} 10 & 4 \\ 24 & 10 \end{bmatrix} \quad (33)$$

在这个视角下，我们惊奇发现矩阵乘法竟然变成了“加法”！

用符号代替数字，(33) 可以写成：

$$\begin{aligned} \mathbf{A} @ \mathbf{B} &= \left[\begin{bmatrix} a_{1,1} \\ a_{2,1} \end{bmatrix}_{2 \times 1} \quad \begin{bmatrix} a_{1,2} \\ a_{2,2} \end{bmatrix}_{2 \times 1} \right] \left[\begin{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \end{bmatrix}_{1 \times 2} \\ \begin{bmatrix} b_{2,1} & b_{2,2} \end{bmatrix}_{1 \times 2} \end{bmatrix} \right] \\ &= [\mathbf{a}_1 \quad \mathbf{a}_2]_{1 \times 2} \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \end{bmatrix}_{2 \times 1} = \mathbf{a}_1 \mathbf{b}^{(1)} + \mathbf{a}_2 \mathbf{b}^{(2)} \end{aligned} \quad (34)$$

更一般情况

将矩阵 $A_{m \times p}$ 写成一系列左右排列的列向量：

$$A_{m \times p} = [a_1 \quad a_2 \quad \cdots \quad a_p]_{l \times p} \quad (35)$$

其中，列向量 a_i 元素数量为 m ，即行数为 m 。

将矩阵 $B_{p \times n}$ 写成上下叠放的行向量：

$$B_{p \times n} = \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(p)} \end{bmatrix}_{p \times l} \quad (36)$$

其中，行向量 $b^{(i)}$ 元素数量为 n ，即列数为 n ：

矩阵 A 和矩阵 B 相乘，可以展开写成 p 个 $m \times n$ 矩阵相加，

$$A_{m \times p} @ B_{p \times n} = [a_1 \quad a_2 \quad \cdots \quad a_p]_{l \times p} \begin{bmatrix} b^{(1)} \\ b^{(2)} \\ \vdots \\ b^{(p)} \end{bmatrix}_{p \times l} = \underbrace{a_1 b^{(1)} + a_2 b^{(2)} + \cdots + a_p b^{(p)}}_{p \text{ matrices with shape of } m \times n} = C_{m \times n} \quad (37)$$

我们可以把 $a_k b^{(k)}$ 结果矩阵写成 C_k ，这样 A 和 B 乘积 C 可以写成 $C_k (k = 1, 2, \dots, p)$ 之和，

$$a_1 b^{(1)} + a_2 b^{(2)} + \cdots + a_p b^{(p)} = C_1 + C_2 + \cdots + C_p = C \quad (38)$$

在这个视角下，矩阵的乘法变成若干矩阵的叠加。这是一个非常重要的视角，数学科学和机器学习很多算法都会离不开它。

热图

图 12 给出的是图 11 所示矩阵乘法第二视角的热图。图中三个形状相同矩阵 C_1 、 C_2 、 C_3 相加得到 C 。

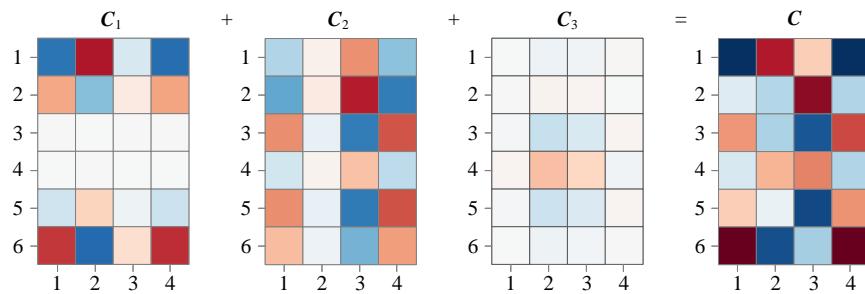


图 12. 矩阵乘法第二视角

如图 13 所示，从图像角度来看，(38) 好比若干形状相同的图片，经过层层叠加，最后获得一幅完整热图。

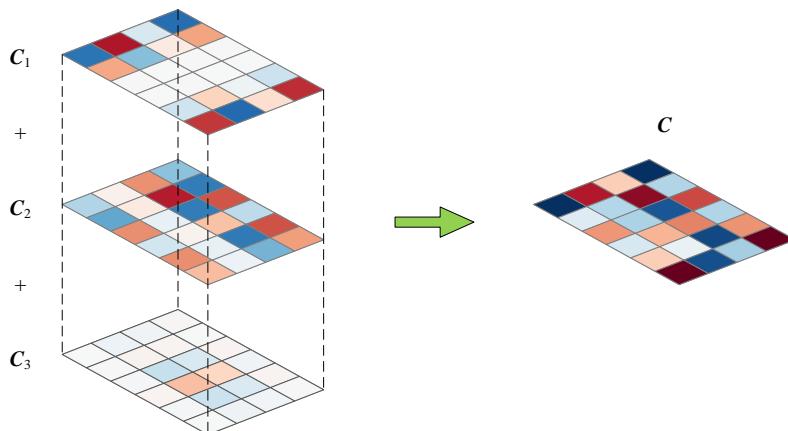
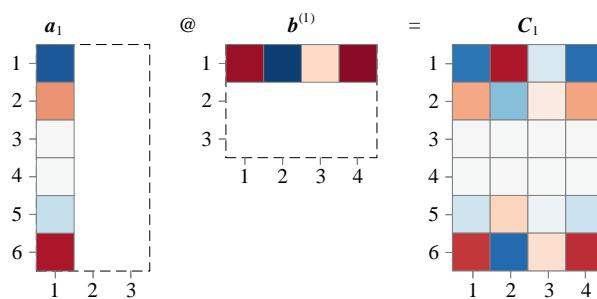
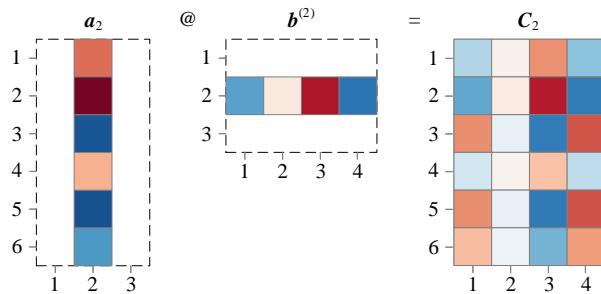
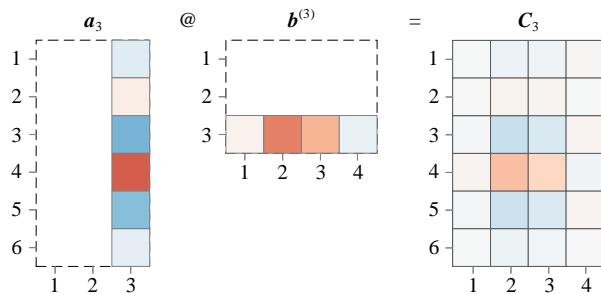
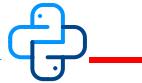
图 13. 三幅图像叠加得到矩阵 C 热图

图 14、图 15、图 16 分别展示如何获得图 12 中矩阵 C_1 、 C_2 、 C_3 热图。

观察热图可以发现一个有意思的现象，列向量乘行向量好像张起了一幅平面。张量积用的就是类似图 14、图 15、图 16 的运算思路。

→ 本系列丛书《矩阵力量》一册会讲解张量积。

图 14. 获得 C_1

图 15. 获得 C_2 图 16. 获得 C_3 

代码文件 Bk3_Ch2_10.py 中 Bk3_Ch2_10_B 部分绘制图 12。



主成分分析 (Principal Component Analysis, PCA) 是机器学习中重要的降维算法。这种算法可以把可能存在线性相关的数据转化成线性不相关的数据，并提取数据中的主要特征。

图 17 中， \mathbf{X} 为原始数据， \mathbf{X}_1 、 \mathbf{X}_2 、 \mathbf{X}_3 给出的是第一、第二、第三主成分。根据热图颜色差异可以看出来，第一主成分解释了原始数据中最大的差异。第二成分则进一步解释剩余数据中最大的差异，以此类推。

图 17 实际上就是本节介绍的矩阵乘法第二视角。本系列丛书会在《矩阵力量》、《概率统计》、《数据科学》三本中从不同视角介绍主成分分析。

《矩阵力量》将从矩阵分解、空间、优化等视角讲解 PCA。《概率统计》将从数据、中心化数据、z 分数、协方差矩阵、相关性系数矩阵这些统计视角来讨论 PCA 不同技术路线之间的差异。《数据科学》则侧重讲解如何在实践中使用 PCA 分析数据，并使用 PCA 结果进行回归分析。

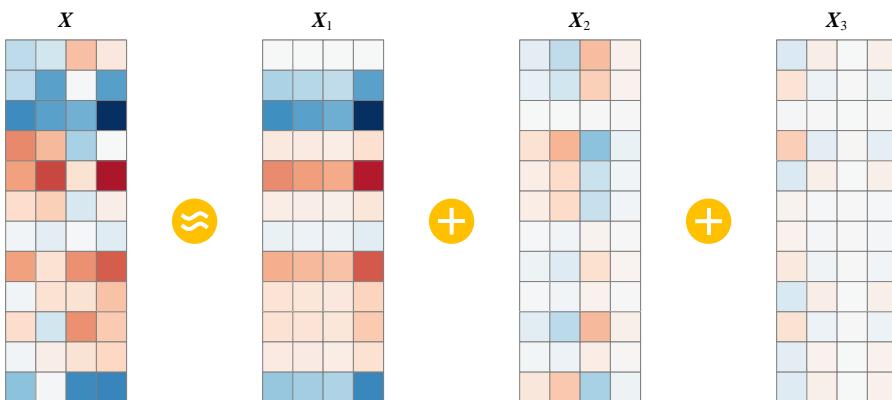


图 17. 用数据热图叠加看主成分分析

2.6 矩阵除法：计算逆矩阵

实际上，并不存在所谓的矩阵除法。所谓矩阵 B 除以矩阵 A ，实际上将矩阵 A 先转化逆矩阵 A^{-1} ，然后计算 B 和逆矩阵 A^{-1} 乘积，即：

$$BA^{-1} = B @ A^{-1} \quad (39)$$

A 如果可逆 (invertible)，仅当 A 为方阵且存在矩阵 A^{-1} 使得下式成立，

$$AA^{-1} = A^{-1}A = I \quad (40)$$

A^{-1} 叫做矩阵 A 的逆 (the inverse of matrix A)。

(40) 中的 I 就是前文介绍过的单位阵 (identity matrix)。 n 阶单位矩阵 (n -square identity matrix 或 n -square unit matrix) 的特点是对角线上的元素为 1，其他为 0，

$$I_{n \times n} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \quad (41)$$

可以用 `numpy.linalg.inv()` 计算方阵的逆。

图 18 所示为方阵 A 和逆矩阵 A^{-1} 相乘得到单位矩阵 I 的热图。

⚠ 注意图中数值仅保留小数点后两位，按图中数值相乘不能准确得到单位矩阵。

$$\begin{array}{c}
 \begin{array}{c} \mathbf{A} \\ \begin{array}{|c|c|c|c|} \hline 0.15 & 0.65 & 0.31 & 0.13 \\ \hline -0.23 & 0.44 & -0.19 & 1.18 \\ \hline 1.39 & -0.35 & 0.88 & 0.09 \\ \hline 0.20 & 1.28 & -1.29 & -1.24 \\ \hline \end{array} \end{array} @ \begin{array}{c} \mathbf{A}^{-1} \\ \begin{array}{|c|c|c|c|} \hline -0.53 & 0.37 & 0.78 & 0.35 \\ \hline 1.11 & 0.06 & -0.13 & 0.17 \\ \hline 1.31 & -0.65 & -0.17 & -0.49 \\ \hline -0.31 & 0.80 & 0.17 & -0.07 \\ \hline \end{array} \end{array} = \begin{array}{c} \mathbf{I} \\ \begin{array}{|c|c|c|c|} \hline 1.00 & 0.00 & 0.00 & 0.00 \\ \hline 0.00 & 1.00 & 0.00 & 0.00 \\ \hline 0.00 & 0.00 & 1.00 & 0.00 \\ \hline 0.00 & 0.00 & 0.00 & 1.00 \\ \hline \end{array} \end{array}
 \end{array}$$

图 18. 方阵 \mathbf{A} 和逆矩阵 \mathbf{A}^{-1} 相乘

一般情况，

$$(\mathbf{A} + \mathbf{B})^{-1} \neq \mathbf{A}^{-1} + \mathbf{B}^{-1} \quad (42)$$

请大家注意以下和矩阵逆有关的运算规则：

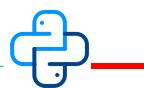
$$\begin{aligned}
 (\mathbf{A}^T)^{-1} &= (\mathbf{A}^{-1})^T \\
 (\mathbf{AB})^{-1} &= \mathbf{B}^{-1}\mathbf{A}^{-1} \\
 (\mathbf{ABC})^{-1} &= \mathbf{C}^{-1}\mathbf{B}^{-1}\mathbf{A}^{-1} \\
 (k\mathbf{A})^{-1} &= \frac{1}{k}\mathbf{A}^{-1}
 \end{aligned} \quad (43)$$

其中，假设 \mathbf{A} 、 \mathbf{B} 、 \mathbf{C} 、 \mathbf{AB} 和 \mathbf{ABC} 逆运算存在。

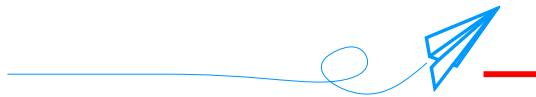
表 4 总结常见矩阵逆相关的英文表达。

表 4. 和矩阵逆相关的英文表达

数学表达	英文表达
\mathbf{A}^{-1}	Inverse of the matrix boldface capital A Matrix boldface capital A inverse
$(\mathbf{A} + \mathbf{B})^{-1}$	Left parenthesis bold face capital A plus boldface capital B right parenthesis superscript minus one Inverse of the matrix sum boldface capital A plus boldface capital B
$(\mathbf{AB})^{-1}$	Left parenthesis boldface capital A times boldface capital B right parenthesis superscript minus one Inverse of the matrix product boldface capital A and boldface capital B
\mathbf{ABC}^{-1}	The product boldface capital A boldface capital B and boldface capital C inverse



Bk3_Ch2_11.py 计算并绘制图 18。



如果学完这一章，大家对矩阵乘法规则还是一头雾水，我只有一个建议——死记硬背！

先别问为什么。就像背诵九九乘法表一样，把矩阵乘法规则背下来。此外，再次强调矩阵乘法等运算不是“奇技淫巧”。后面，大家会逐步意识到矩阵乘法的洪荒伟力。

3 几何

Geometry

音乐之美由耳朵来感受，几何之美让眼睛去欣赏



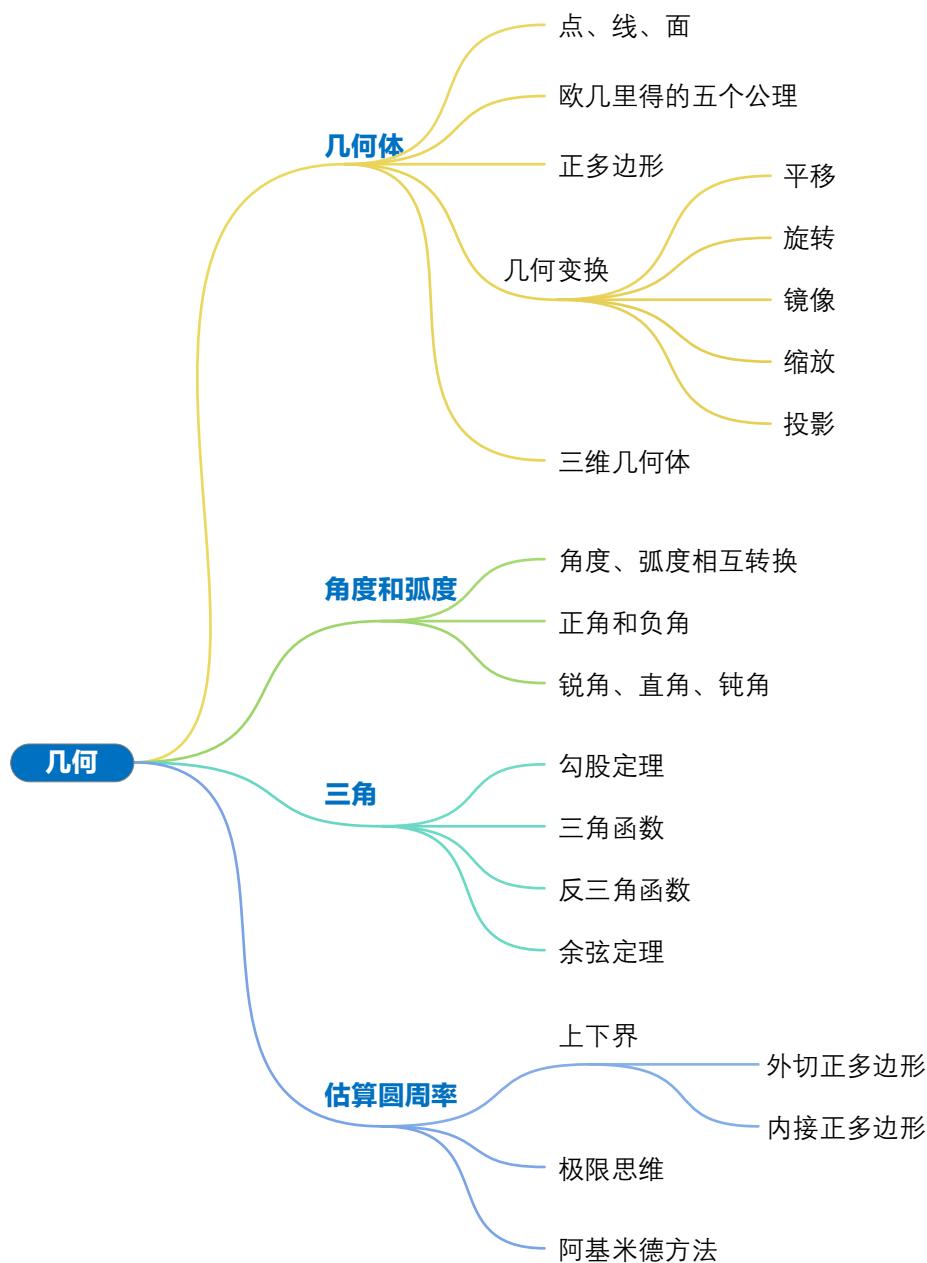
不懂几何，勿入斯门。

Let no one destitute of geometry enter my doors.

—— 柏拉图 (Plato) | 古希腊哲学家 | 424/423 ~ 348/347 BC



- ◀ ax.add_patch() 绘制图形
- ◀ math.degrees() 将弧度转换为角度
- ◀ math.radians() 将角度转换成弧度
- ◀ matplotlib.patches.Circle() 创建正圆图形
- ◀ matplotlib.patches.RegularPolygon() 创建正多边形图形
- ◀ numpy.arccos() 计算反余弦
- ◀ numpy.arcsin() 计算反正弦
- ◀ numpy.arctan() 计算反正切
- ◀ numpy.cos() 计算余弦值
- ◀ numpy.deg2rad() 将角度转化为弧度
- ◀ numpy.rad2deg() 将弧度转化为角度
- ◀ numpy.sin() 计算正弦值
- ◀ numpy.tan() 计算正切值



3.1 几何缘起：根植大地，求索星空

毫不夸张地说，几何思维印刻在人类基因中。生而为人，时时刻刻看到的、触摸的都是各种各样的几何形体。

大家现在不妨停下来看看、摸摸周围环境中的物体，相信你一定会惊叹整个物质世界就是几何的世界。

宏观如天体，微观至原子，几何无处不在。正如**约翰内斯·开普勒**(Johannes Kepler, 1571 ~ 1630) 所言“但凡有物质的地方，就有几何。”

哪怕在遥远的古代文明，人类活动也离不开几何知识，丈量距离、测绘地形、估算面积、计算体积、营造房屋、设计工具、制作车轮、工艺美术…无处不需要几何这个数学工具。

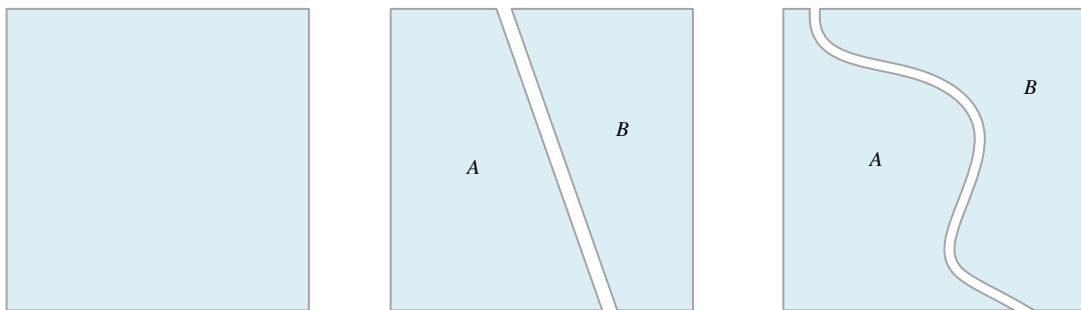


图 1. 各种形状田地地块

几何滥觞于田间地头。在古埃及，尼罗河每年都要淹没河两岸。当洪水退去，留下的肥沃土壤让河两岸平原的农作物生长。但是洪水同样冲走了标示不同耕地界桩。

法老每年都要派大量测量员重新丈量土地。测绘员们用打结的绳子去丈量土地和角度，以便重置这些界桩。计算矩形、三角形农田面积当然简单。对于复杂的几何形体，测绘员经常将土地分割成矩形和三角形来估算土地面积。古埃及的几何知识则随着测量精度提高而不断累积精进。

无独有偶，中国古代重要的数学典籍之一《九章算术》的第一章名为“方田”。这一章多数题目以丈量土地为例，讲解如何计算长方形、三角形、梯形、圆形等等各式几何形状的面积。

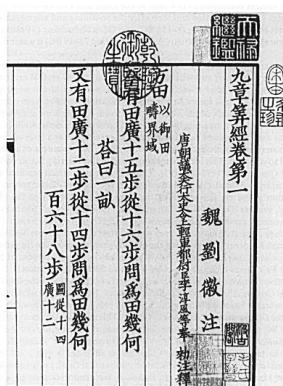


图 2. 《九章算术》第一章开篇

几何学的重大飞跃来自于古希腊。古希腊人创造了几何 geometron (英文：geometry) 这个词；“geo”在希腊语里是“大地”的意思，“metron”的意思是“测量”。

在古希腊，几何学受到高度重视。几何是博雅教育七艺的重要一门课程。据传说，柏拉图学院门口刻着如下这句话：“不懂几何者，不许入内。”

图 3 所示为古希腊几何发展时间轴上重要的数学家，以及同时代的其他伟大思想家。值得注意的是，中国春秋时代的孔子和苏格拉底、柏拉图、亚里士多德，竟然是同属一个时代。东西方两条历史轴线给人平行时空的错觉。

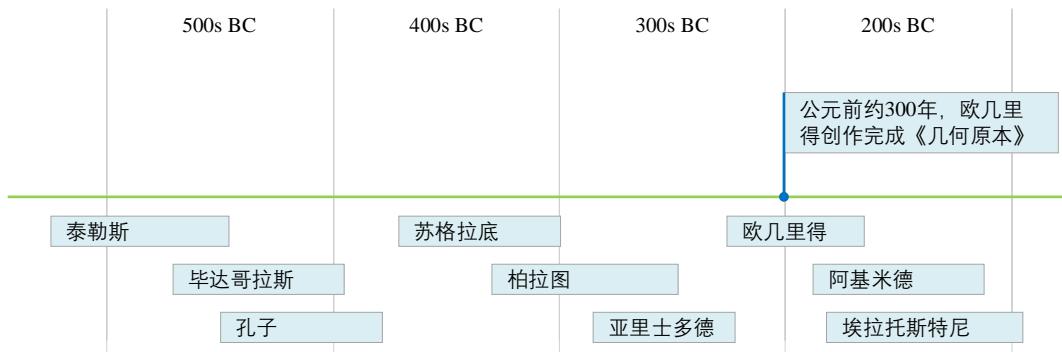


图 3. 古希腊几何发展历史时间轴



欧几里得 (Euclid)

古希腊数学家 | 约公元前330年 ~ 公元前275年

被称为“几何之父”，他的《几何原本》堪称西方现代数学的开山之作



古希腊数学家中关键人物是欧几里得 (Euclid)，他的巨著《几何原本》(The Elements) 首次尝试将几何归纳成一个系统。

不夸张地说，欧几里得《几何原本》是整个人类历史上最成功、影响最深刻的数学教科书，没有之一。《几何原本》不是习题集，它引入严谨的推理，使得数学体系化。

古希腊的几何学发展要远远领先于其他数学门类，可以说古希腊的算术和代数知识也都是建立在几何学基础之上。而代数的大发展要归功于一位波斯数学家——花拉子密，这是我们下一章要介绍的人物。

中文“几何”一词源自于《几何原本》的翻译。1607年，明末科学家徐光启和意大利传教士利玛窦 (Matteo Ricci) 共同翻译完成《几何原本》前六章。

他们确定了包括“几何”、“点”、“直线”、“角”等大量中文译名。“几何”一词的翻译特别精妙，发音取自 geo，而“几何”二字的中文又有“大小如何”的含义。《九章算术》几乎所有的题目都以“几何”这一提问结束，比如“问：为田几何？”

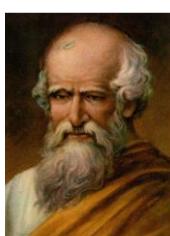


图 4. 《几何原本》1570 年首次被翻译为英文版



图 5. 《中国图说》(China Illustrata) 中插图描绘利玛窦和徐光启

在估算圆周率的竞赛中，阿基米德 (Archimedes) 写下浓墨重彩的一笔。阿基米德利用圆内接正多边形和圆外切正多边形，估算圆周率在 $223/71$ 和 $22/7$ 之间，即 3.140845 和 3.142857 之间。



阿基米德 (Carl Friedrich Gauss)
古希腊数学家、物理学家 | 公元前287年 ~ 公元前212年
常被称作“力学之父”，估算圆周率



公元前 212 年，阿基米德的家乡被罗马军队攻陷时，他还在潜心研究几何问题。罗马士兵闯入他的家，阿基米德大声训斥这些不速之客，“别弄乱我的圆”。但是，罗马士兵还是踩坏了画在沙盘上的几何图形，并杀死了阿基米德。

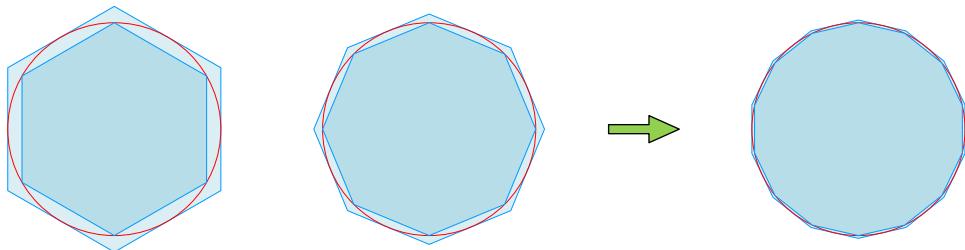


图 6. 圆形内接和外切正 6、正 8 边、正 16 边形

几何学有纬地经天之功。比如，利用相似三角形原理，古希腊数学家**埃拉托斯特尼**(Eratosthenes) 估算地球直径。

正午时分，在点 A (阿斯旺) 太阳光垂直射入深井中，井底可见太阳倒影。此时，在点 B (亚历山大港)，埃拉托斯特尼找人测量一个石塔影子的长度。

利用石塔的高度和影子的长度，埃拉托斯特尼计算得到图 7 中 $\theta = 7^\circ$ ，也就是 A 和 B 两点的距离为整个地球圆周的 $7/360$ 。

埃拉托斯特尼恰好知道 AB 距离，从而估算地球的周长。进而计算得到地球周长在 39,690 千米到 46,620 千米之间，误差约 2%。

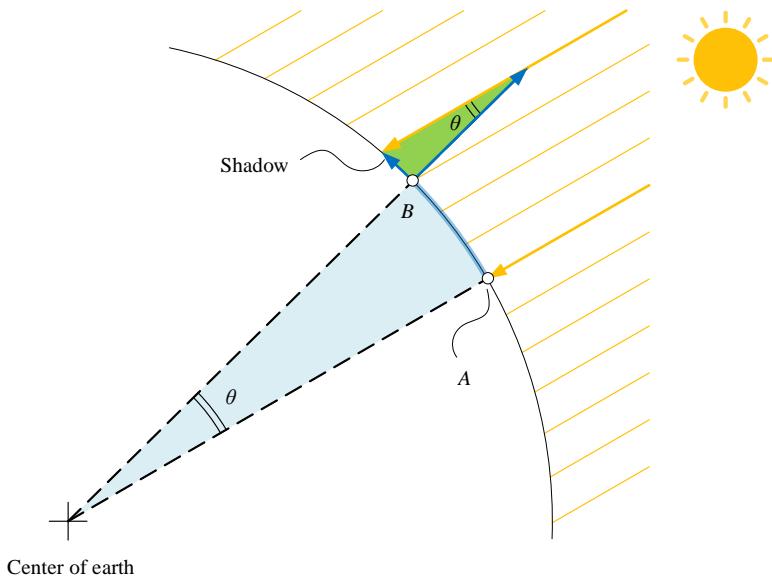


图 7. 埃拉托斯特尼计算地球直径用到的几何知识

托勒密 (Claudius Ptolemy) 在约 150 年创作《天文学大成》 (*Almagest*)。这本书可以说是代表了古希腊天文学的最高水平，它也是古希腊几何思维在天文学领域的结晶。

托勒密总结前人成果，在书中明确提出**地心说** (geocentric model)——地球位于宇宙中心，固定不动，星体绕地球运动。此外，《天文学大成》中给出了人类历史上第一个系统建立的三角函数表。



托勒密 (Claudius Ptolemy)
希腊数学家、天文学 | 100年 ~ 170年
创作《天文学大成》，系统提出地心说



然而，托勒密的地心说被宗教思想奉为圭臬，牢牢禁锢人类长达一千两百多年，直到**哥白尼** (Nicolaus Copernicus, 1473 ~ 1543) 唤醒人类沉睡的思想世界。正是利用古希腊发展的圆锥曲线知识，开普勒提出了行星运动三定律。



圆锥曲线是本书第 8、9 章要介绍的内容。



图 8. 后人绘制的托勒密地心说模型

3.2 点动成线，线动成面，面动成体

点动成线，线动成面，面动成体——相信大家对这句话耳熟能详。点没有维度，线是一维，面是二维，体是三维。当然，在数学的世界，四维乃至多维都是存在的。

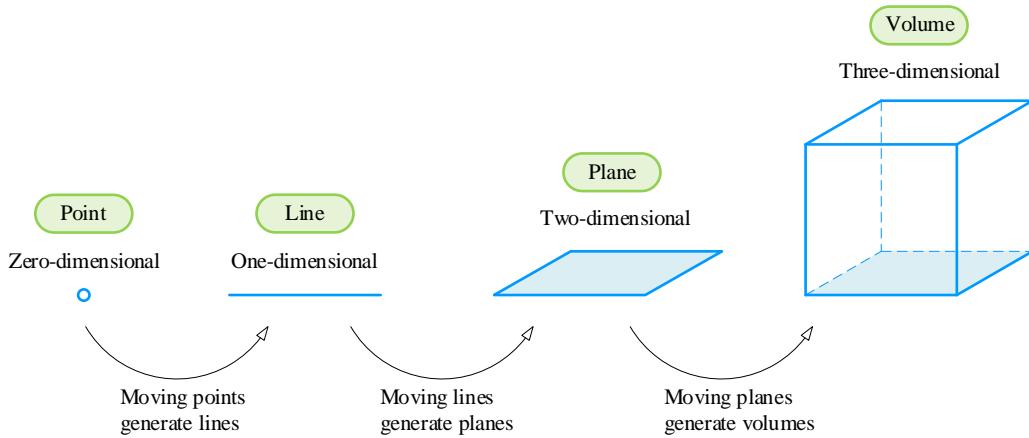


图 9. 点动成线，线动成面，面动成体

点

点确定空间的一个位置，点本身没有长度、面积等几何属性。

所有几何图形都离不开点，图 10 所示为常见的几种点——**端点** (endpoint)、**中点** (midpoint)、**起点** (initial point)、**终点** (terminal point)、**圆心** (center)、**切点** (point of tangency)、**顶点** (vertex)、**交点** (point of intersection)。点和点之间的线段长度叫**距离** (distance)。

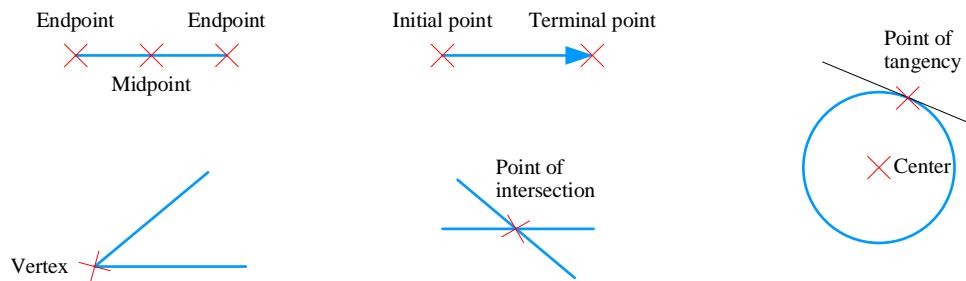


图 10. 几种点

线

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

如图 11 所示，**直线** (line) 沿两个方向无限延伸 (extends in both directions without end)，**没有端点** (has no endpoints)。

射线 (ray 或 half-line) 开始于一端点，仅沿一个方向无限延伸。

线段 (line segment) 有两个**端点** (endpoint)。

向量 (vector) 则是有方向的线段。

线段具有**长度** (length) 这个几何性质，但是没有面积这个性质。

给定参考系，线又可以分为**水平线** (horizontal line)、**斜线** (oblique line) 和**竖直线** (vertical line)。

图 11 还给出其他几种线：**边** (edge)、**曲线** (curve 或 curved line)、**等高线** (contour line)、**法线** (normal line)、**切线** (tangent line)、**割线** (secant line) 等。

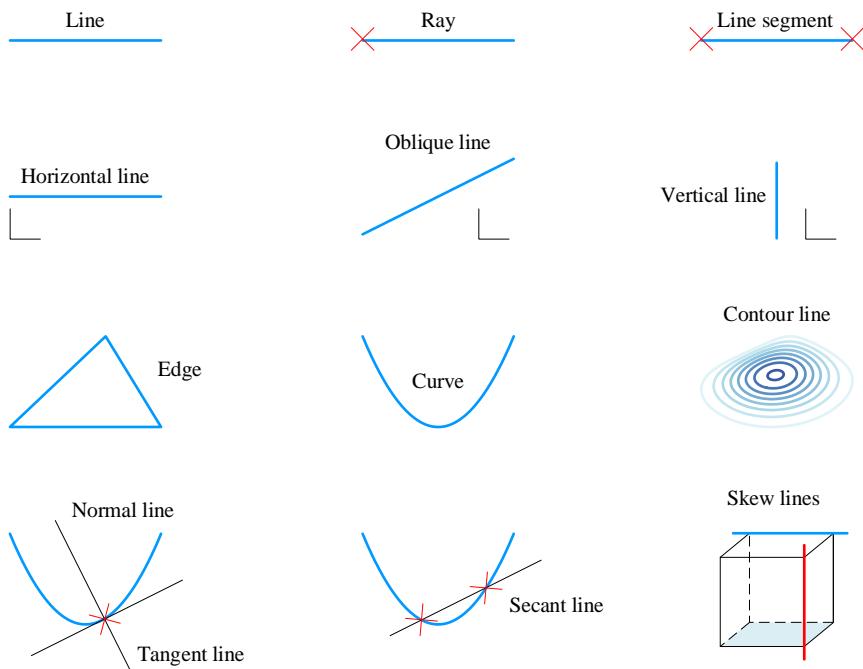


图 11. 几种线

在平面上，线与线之间有四种常见的关系：**平行** (parallel)、**相交** (intersecting)、**垂直** (perpendicular) 和**重合** (coinciding)。

两条线平行可以记作 $l_1 \parallel l_2$ (读作 line l sub one is parallel to the line l sub two)。 l_1 与 l_2 相交于点 P 可以读作“line l sub one intersects the line l sub two at point capital P ”。两条线垂直可以记作 $l_1 \perp l_2$

(读作 line l sub one is perpendicular to the line l sub two)。三维空间中，两条直线还可以互为**异面线** (skew line)。

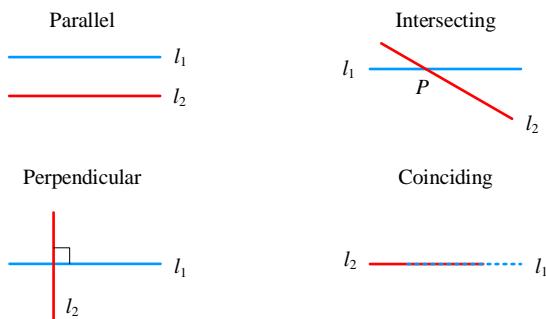


图 12. 平面上两条线的关系

如图 13 所示，可视化时还会用到不同样式的线型，比如**实线** (solid line 或 continuous line)、**粗实线** (heavy solid line 或 continuous thick line)、**点虚线** (dotted line)、**短划线** (dashed line)、**点划线** (dash-dotted line) 等等。

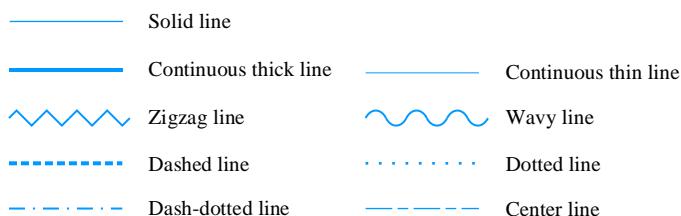


图 13. 几种线的样式

欧几里得的五个公理

在《几何原本》中，欧几里得提出五个公理：

- ◀ 任意两点可以画一条直线；
- ◀ 任意线段都可以无限延伸成一条直线；
- ◀ 给定任意线段，以该线段为半径、一个端点为圆心，可以画一个圆；
- ◀ 所有直角都全等；
- ◀ 两直线被第三条直线所截，如果同侧两内角之和小于两个直角之和，则两条直线则会在该侧相交。

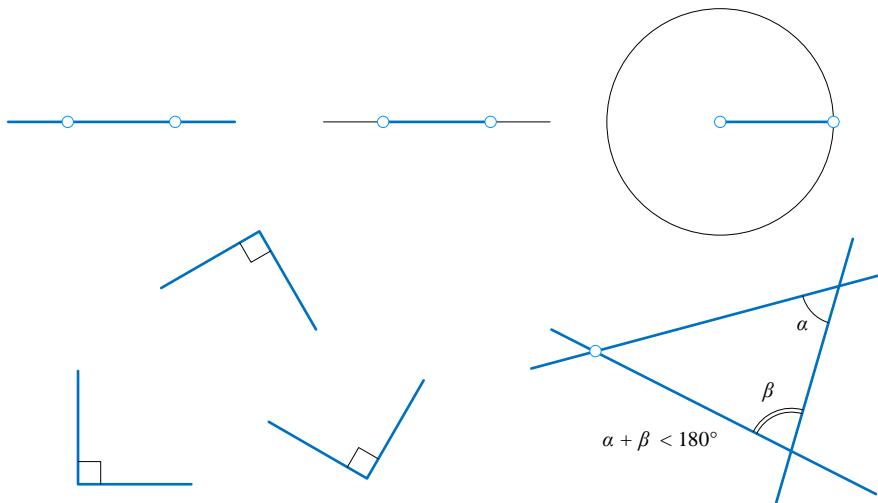


图 14. 欧几里得提出的五个几何公理

以五个公理为基础，欧几里得一步步建立起几何学大厦。坚持第五条定理，我们在欧几里得几何体系之内。而去掉第五条公理，则进入非欧几何体系。值得一提的是，非欧几何中的黎曼几何为爱因斯坦的广义相对论提供了数学工具。

正多边形

正多边形 (regular polygons) 是边长相等的多边形，正多边形内角相等。我们将在圆周率估算中用到正多边形相关知识。

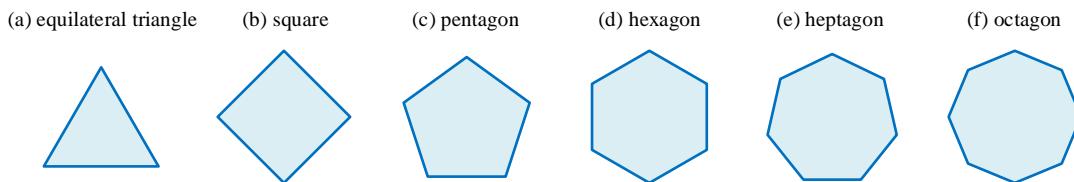


图 15. 六个正多边形



Bk3_Ch3_01.py 绘制图 15 中六个正多边形。

三维几何体

图 16 所示为常见三维几何体，它们依次是：**正球形** (sphere)、**圆柱体** (cylinder)、**圆锥** (cone)、**锥台** (cone frustum)、**正方体**、**正六面体** (cube)、**长方体** (cuboid)、**平行六面体** (parallelepiped)、**四棱台** (square pyramid frustum)、**四棱锥** (square-based pyramid)、**三棱锥** (triangle-based pyramid)、**三棱柱** (triangular prism)、**四面体** (tetrahedron)、**八面体** (octahedron)、**五棱柱** (pentagonal prism)、**六棱柱** (hexagonal prism) 和**五棱锥** (pentagonal pyramid)。

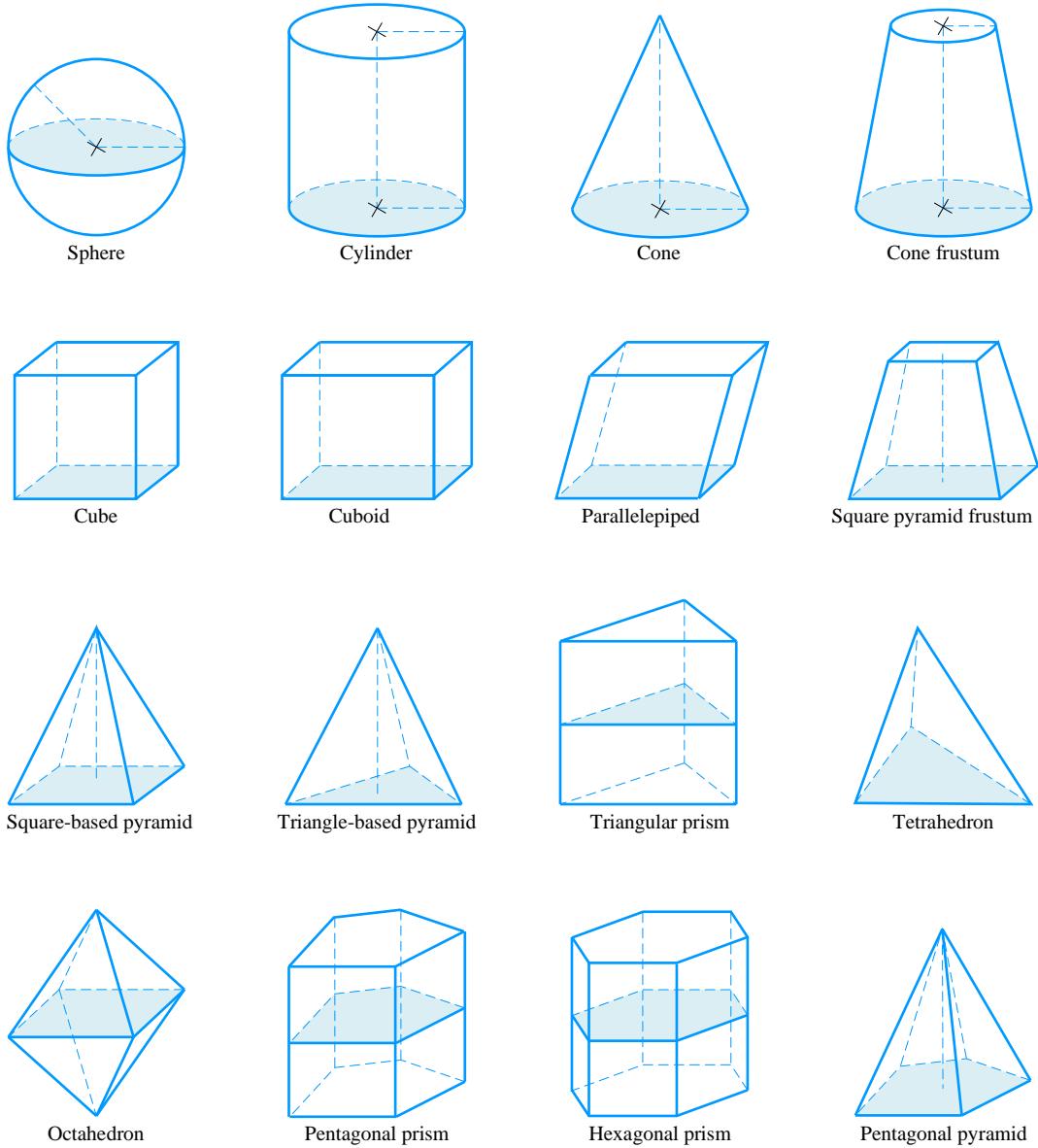


图 16. 常见三维几何体

柏拉图立体 (Platonic solid), 又称正多面体。图 17 所示为五个正多面体，包括**正四面体** (tetrahedron)、**正方体**、**正六面体** (cube)、**正八面体** (octahedron)、**正十二面体** (dodecahedron) 和**正二十面体** (icosahedron)。

正多面体的每个面全等，均为**正多边形** (regular polygons)。图 18 所示为五个正多面体展开得到的平面图形。**表 1** 总结五个正多面体的结构特征。

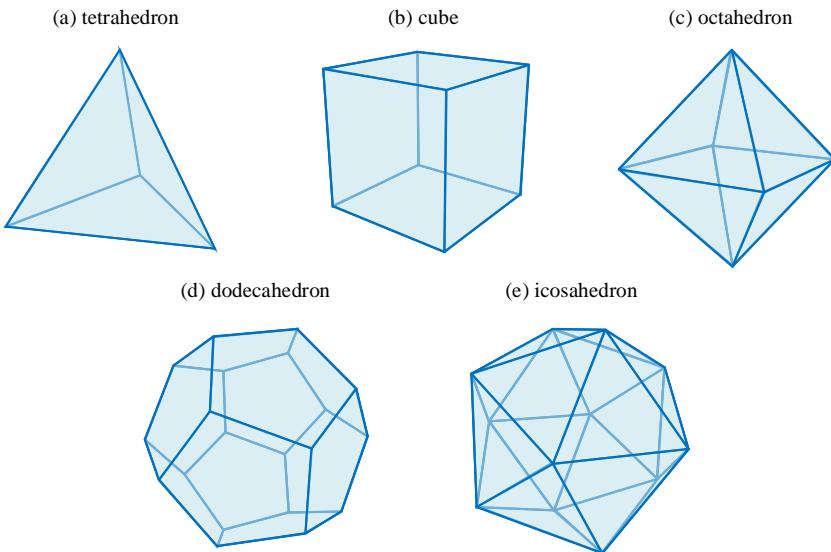


图 17. 五个正多面体

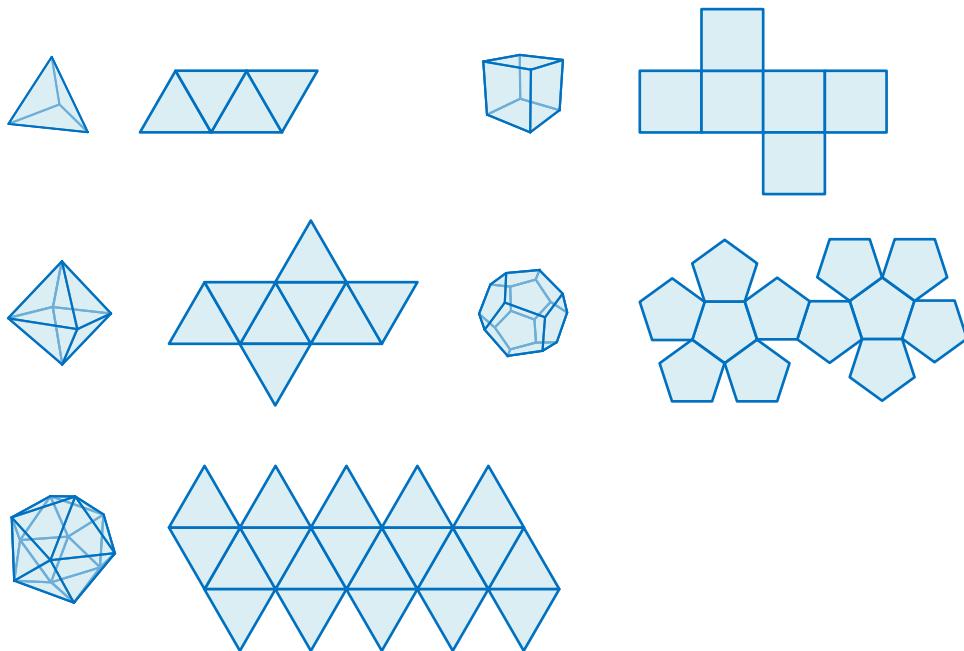


图 18. 五个正多面体展开得到的平面图形

表 1. 正多面体的特征

正多面体	顶点数	边数	面数	面形状
Tetrahedron	4	6	4	Equilateral triangle
Cube	8	12	6	Square
Octahedron	6	12	8	Equilateral triangle
Dodecahedron	20	30	12	Pentagon
Icosahedron	12	30	20	Equilateral triangle

几何变换

几何变换 (geometric transformation) 是本系列丛书中重要的话题之一。我们将在函数变换、线性变换、多元高斯分布等话题中用到几何变换。

如图 19 所示，在平面上，可以通过**平移** (translate)、**旋转** (rotation)、**镜像** (reflection)、**缩放** (scaling)、**投影** (projection) 将某个图形变换得到新的图形。这些几何变换还可以按一定顺序组合完成特定变换。

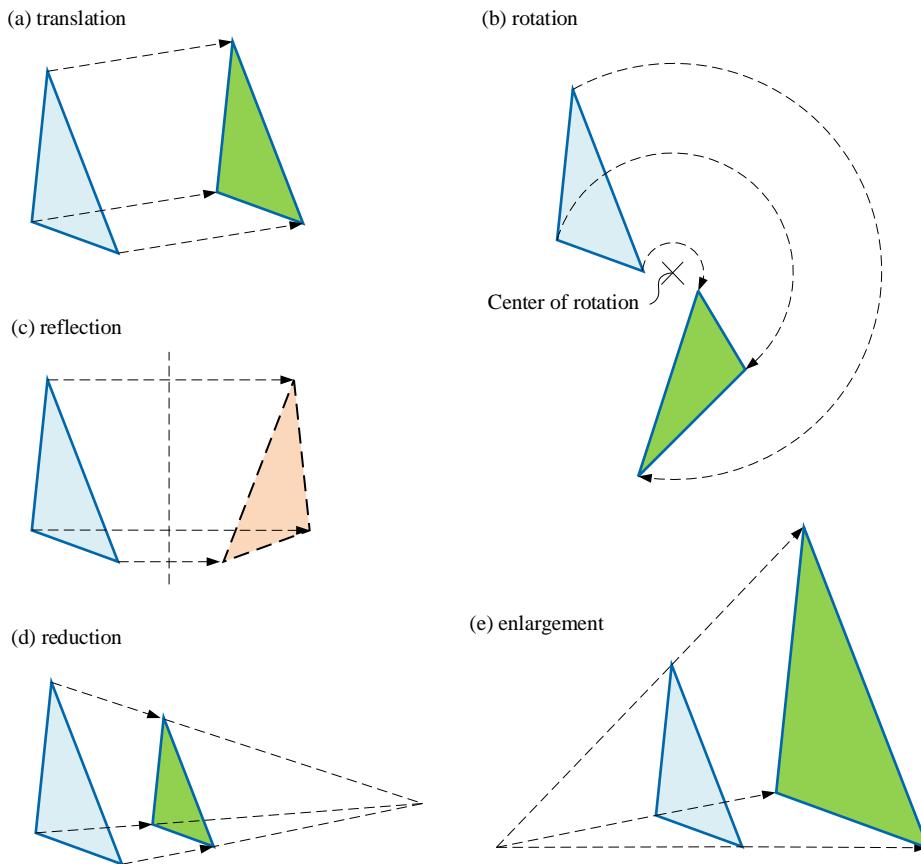


图 19. 常见几何变换

投影

大家平时一定会见到阳光和灯光下各种物体留下的影子，这就是投影。比如，图 20 所示为一个马克杯在不同角度的投影。

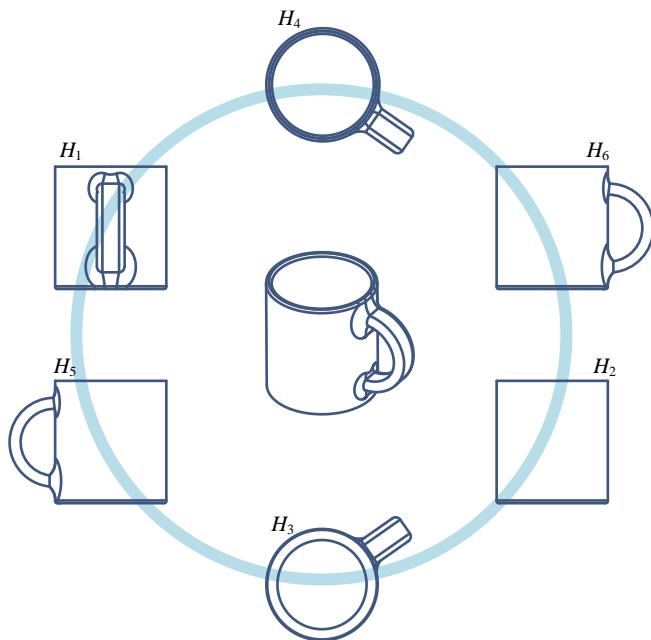


图 20. 咖啡杯在六个方向投影图像

几何中，投影指的是将图形的影子投到一个面或一条线上。如图 21 所示，点可以投影到直线或平面上，影子也是一个点；线段投影到平面上，得到可以是线段。

数学中最常见的投影是正投影。正投影中，投影线（图 21 中虚线）垂直于投影面。线性代数中，我们管这类投影叫**正交投影**（orthogonal projection）

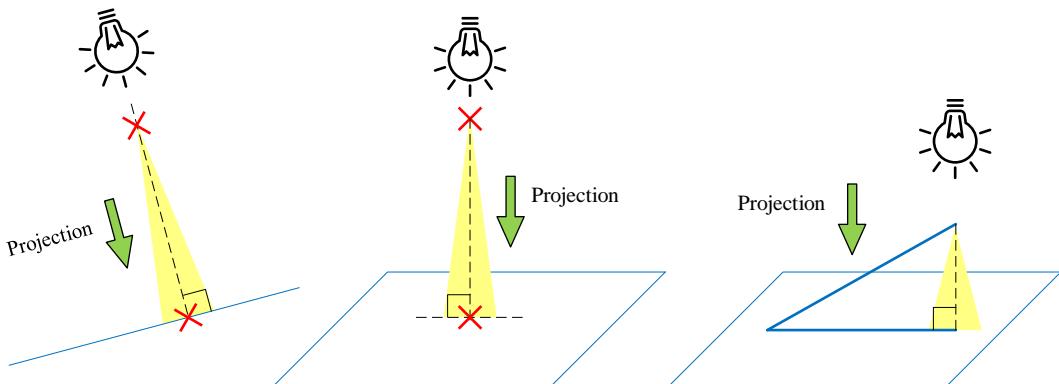


图 21. 投影

3.3 角度和弧度

角度

度 (degree) 是一种常用的角度度量单位。角度可以用量角器 (protractor) 测量。**一周** (a full circle、one revolution 或 one rotation) 对应 360° 。 1° 对应 60 分 (minute)，即 $1^\circ = 60'$ 。 $1'$ 对应 60 秒 (second)，即 $1' = 60''$ 。

形如 25.1875° 被称作**小数角度** (decimal degree)，可以换算得到 $25^\circ 11' 15''$ (twenty five degrees eleven minutes and fifteen seconds)。

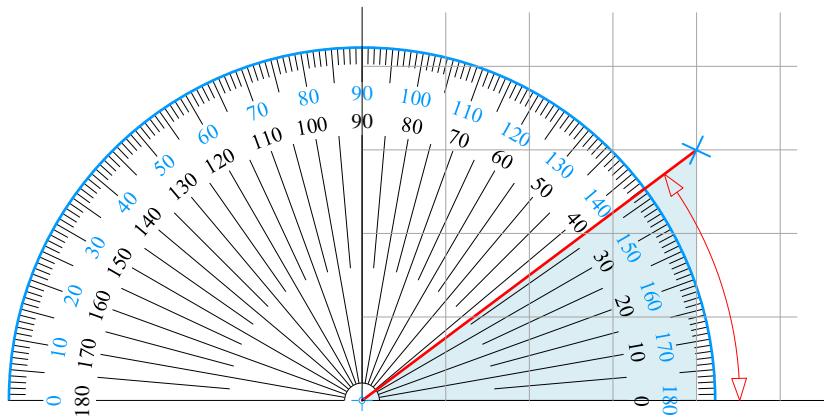


图 22. 量角器测量角度

弧度

弧度 (radian) 常简写作 rad。1 个弧度相当于 $1 \text{ rad} \approx 57.2958^\circ$ 。

在 math 库中，`math.radians()` 函数将角度转换成弧度；`math.degrees()` 将弧度转换为角度。NumPy 中，可以用 `numpy.rad2deg()` 函数将弧度转化为角度，用 `numpy.deg2rad()` 将角度转化为弧度。

常用弧度和角度的换算关系如下：

$$\begin{aligned} 360^\circ &= 2\pi \text{ rad} \\ 180^\circ &= \pi \text{ rad} \\ 90^\circ &= \frac{\pi}{2} \text{ rad} \\ 45^\circ &= \frac{\pi}{4} \text{ rad} \\ 30^\circ &= \frac{\pi}{6} \text{ rad} \end{aligned} \tag{1}$$

正角和负角

如果旋转为逆时针 (counter-clockwise)，角度为正角 (positive angle)。如果旋转为顺时针 (clockwise)，角度为负角 (negative angle)。

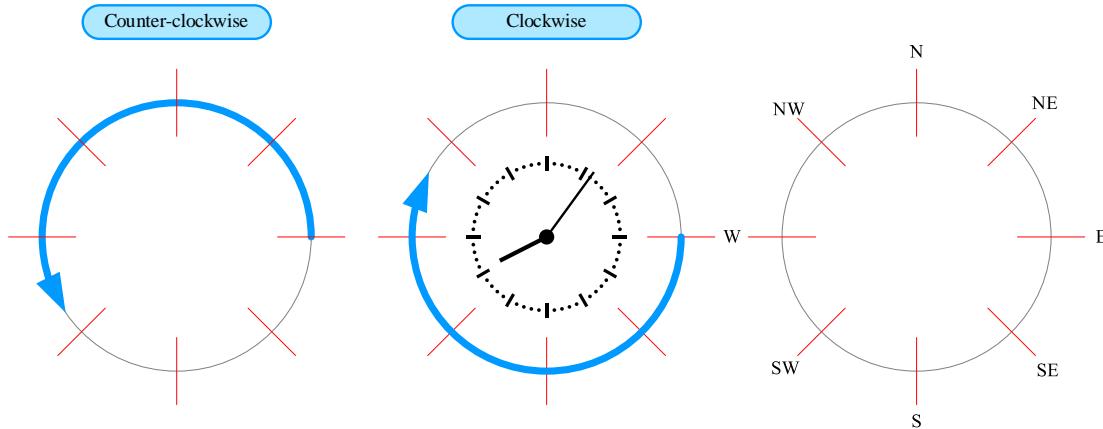


图 23. 逆时针、顺时针和方位

锐角、直角、钝角

锐角 (acute angle) 是指小于 90° 的角，**直角** (right angle) 是指等于 90° 的角，**钝角** (obtuse angle) 是指大于 90° 并且小于 180° 的角。

请大家特别注意这三个角度，在线性代数、数据科学中它们的内涵将得到不断丰富。

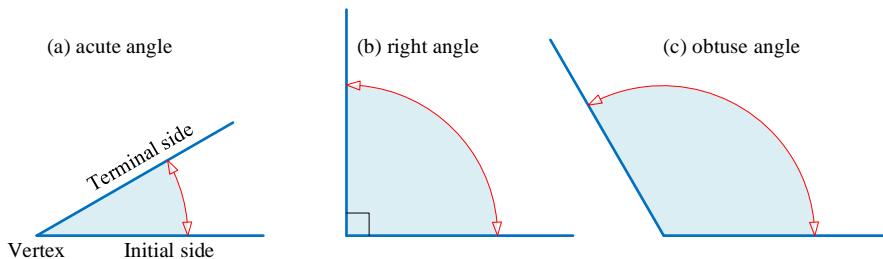


图 24. 锐角、直角和钝角

3.4 勾股定理到三角函数

勾股定理

《周髀算经》编写于公元前一世纪之前，其中记录着商高与周公的一段对话。商高说“故折矩，勾广三，股修四，经隅五。”后人把这一发现简化成——勾三、股四、弦五。《九章算术》的最后一章讲解的也是勾股定理。

满足勾股定理的一组整数，比如(3, 4, 5)，叫做勾股数。

在西方，勾股定理被称作**毕达哥拉斯定理**(Pythagorean Theorem)。

古代很多文明都独立发现了勾股定理。原因也不难理解，古时人们在丈量土地，建造房屋时，都离不开直角。

古埃及人善于使用绳索构造特定几何关系。比如，绳索等距打结，就可以充当带刻度的直尺。绳索一端固定，另外一段绕固定端旋转一周，就可以得到正圆。

古埃及人也发现3:4:5的直角三角形。据此，利用绳索可以轻松获得直角。绳索等距打13个结，形成12段等长线段。按照3:4:5比例分配等距线段，3等分和4等分的两边的夹角便是直角。

勾股定理的一般形式如下：

$$a^2 + b^2 = c^2 \quad (2)$$

如图25所示， a 和 b 为直角边， c 为斜边。图中， a^2 、 b^2 、 c^2 分别为三个正方形的面积。

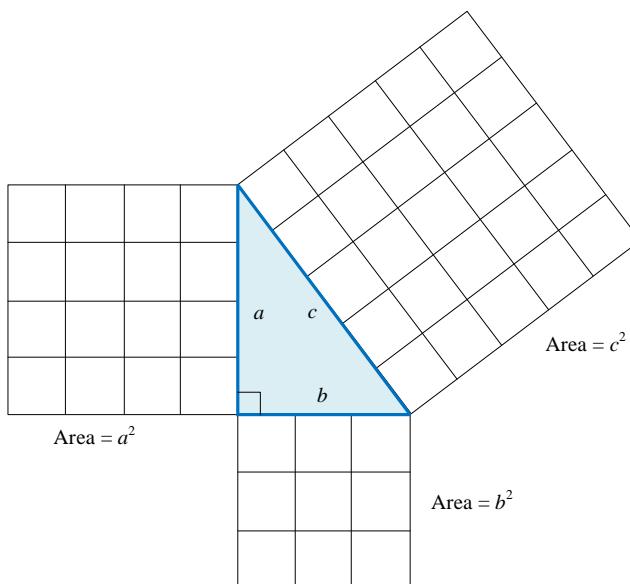


图 25. 图解勾股定理

三角函数

三角函数(trigonometric function)的自变量为弧度角度，因变量为直角三角形斜边、邻边、对边中两个长度的比值。每个比值都有其特定的名字。

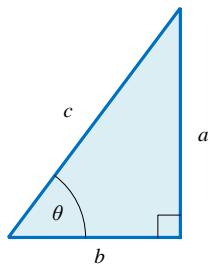


图 26. 直角三角形中定义三角函数

如图 26 所示， θ 的**正弦** (sine) 是对边 a 与斜边 c 的比值：

$$\sin \theta = \frac{a}{c} \quad (3)$$

`numpy.sin()` 可以用来计算正弦值，输入为弧度。

θ 的**余弦** (cosine) 是邻边 b 与斜边 c 的比值：

$$\cos \theta = \frac{b}{c} \quad (4)$$

`numpy.cos()` 可以用来计算余弦值，输入同样为弧度。

θ 的**正切** (tangent) 是对边 a 与邻边 b 的比值：

$$\tan \theta = \frac{a}{b} \quad (5)$$

`numpy.tan()` 可以用来计算正切值，输入也为弧度。

θ 的**余切** (cotangent) 是邻边 b 与对边 a 的比值，是正切的倒数：

$$\cot \theta = \frac{b}{a} = \frac{1}{\tan \theta} \quad (6)$$

θ 的**正割** (secant) 是斜边 c 与邻边 b 的比值，是余弦的倒数：

$$\sec \theta = \frac{c}{b} = \frac{1}{\cos \theta} \quad (7)$$

θ 的**余割** (cosecant) 是斜边 c 与对边 a 的比值，是正弦的倒数：

$$\csc \theta = \frac{c}{a} = \frac{1}{\sin \theta} \quad (8)$$

反三角函数

反三角函数 (inverse trigonometric function) 则是通过三角函数值来反求弧度或角度。表 2 所示为三个常用反三角函数中英文名称、NumPy 函数等。

表 2. 常用三个反三角函数

数学表达	英文表达	中文表达	NumPy 函数
$\arcsin \theta$	arc sine theta inverse sine theta	反正弦	numpy.arcsin()
$\arccos \theta$	arc cosine theta inverse cosine theta	反余弦	numpy.arccos()
$\arctan \theta$	arc tangent theta inverse tangent theta	反正切	numpy.arctan()

余弦定理

本节最后简单介绍**余弦定理** (law of cosines)。给定如图 27 所示三角形，余弦定理的三个等式如下：

$$\begin{cases} a^2 = b^2 + c^2 - 2bc \cdot \cos \alpha \\ b^2 = a^2 + c^2 - 2ac \cdot \cos \beta \\ c^2 = a^2 + b^2 - 2ab \cdot \cos \gamma \end{cases} \quad (9)$$

当 α 、 β 、 γ 三者之一为直角时，(9) 中的一个等式就变成勾股定理等式。

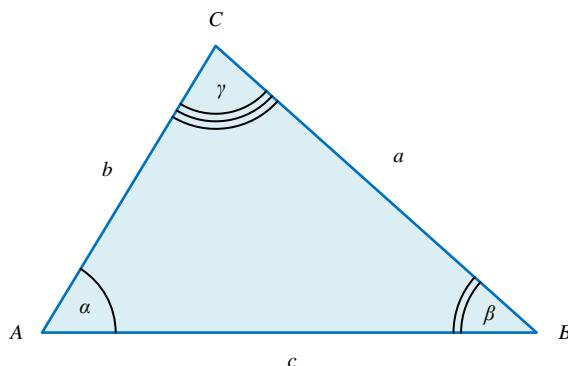


图 27. 余弦定理

在机器学习和数据科学中，余弦定理格外重要。我们会在向量加减法、方差协方差运算、余弦相似度等看到余弦定理的影子。

3.5 圆周率估算初赛：割圆术

圆周率 (π) 是圆的周长和直径之比。

估算圆周率可以看做是不同时空数学家之间的一场竞赛，这场竞赛的标准就是看谁的估算圆周率的精度更准、效率更高。

利用不同的数学工具估算圆周率也是本书一条重要的线索，大家可以从时间维度上看到数学思维、数学工具的迭代发展。

本节介绍数学方法的相当于圆周率估算的“初赛”，这时期数学家使用的数学工具是从几何视角出发的割圆术。

古希腊阿基米德，利用和圆内接正多边形和外切正多边形来估算 π 。阿基米德最后计算到正 96 边形，估算圆周率在 3.1408 到 3.1429 之间。

中国古代魏晋时期的数学家——刘徽（约 225 年 ~ 约 295 年）——用不断增加内接多边形估算圆周率，这种方法被称之为割圆术。

刘徽也用割圆术，从直径为 2 尺的圆内接正六边形开始割圆，依次得正 12 边形、正 24 边形、正 48 边形等等。割得越细，正多边形面积和圆面积差别越小。用他的原话是“割之弥细，所失弥少，割之又割，以至于不可割，则与圆周合体而无所失矣。”这句话中，我们可以体会到“逼近”、“极限”这两个重要的数学思想。

最后，刘徽计算了正 3072 边形的周长，估算得到的圆周率为 3.1416。

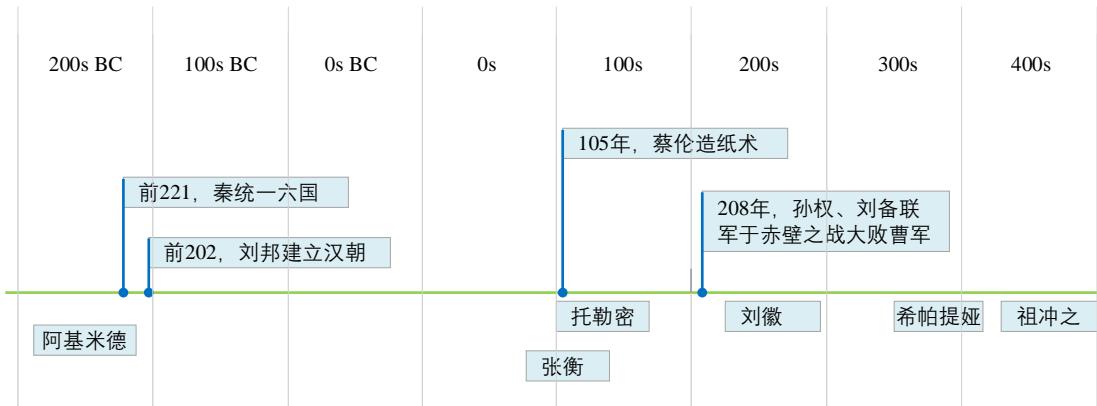


图 28. 圆周率估算的初赛

刘徽之后约 200 年，中国古代南北朝时期数学家祖冲之 (429 ~ 500) 也是用割圆术，最后竟然达到正 12288 边形，估算圆周率为 3.1415926 到 3.1415927 之间。祖冲之再一次刷新圆周率记录，而这一记录几乎保持一千年，直到新的估算圆周率的数学工具横空出世。

内接和外切正多边形

图 29 给出的是正圆内接和外切正 6、8、10、12、14、16 边形。可以发现，正多边形的边数越多，内接和外切正多边形越靠近正圆。

观察图 29，容易发现圆的周长大于圆内接正多边形的边长之和。也就是说，在估算圆周率时，内接正多边形边长和可以作为圆的周长的下边界。

而圆外切正多边形的边长之和大于圆的周长，则为圆周长的上边界。特别地，当正圆为单位圆时，单位圆的周长恰好为 2π ，这方便建立 π 和正多边形边长的联系。

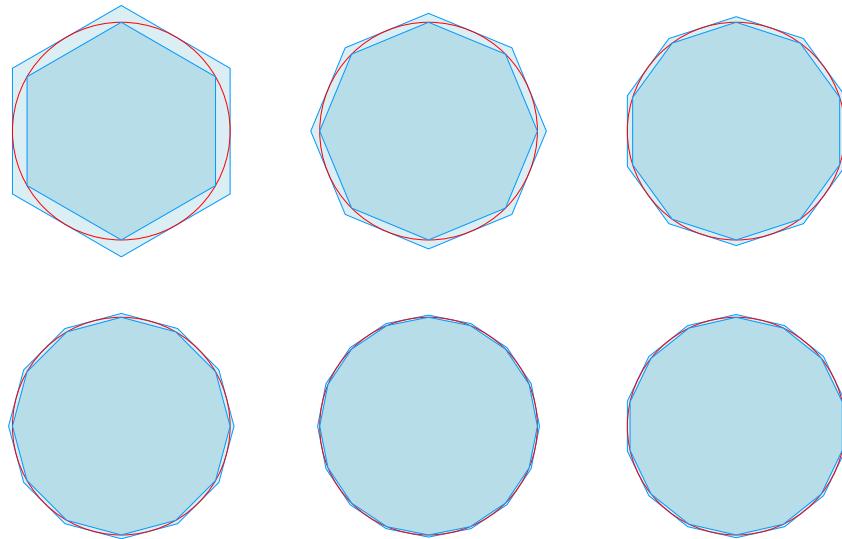
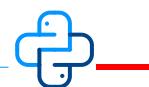


图 29. 圆形内接和外切正 6、8、10、12、14、16 边形



代码文件 Bk3_Ch3_02.py 绘制图 29。

估算圆周率上下界

图 30 给定一个单位圆，单位圆外切和内接相同边数的正多边形。两个正多边形都可以分割为 $2n$ 个三角形，这样圆周 360° (2π) 被均分为 $2n$ 份，每一份对应的角度为：

$$\theta = \frac{2\pi}{2n} = \frac{\pi}{n} \quad (10)$$

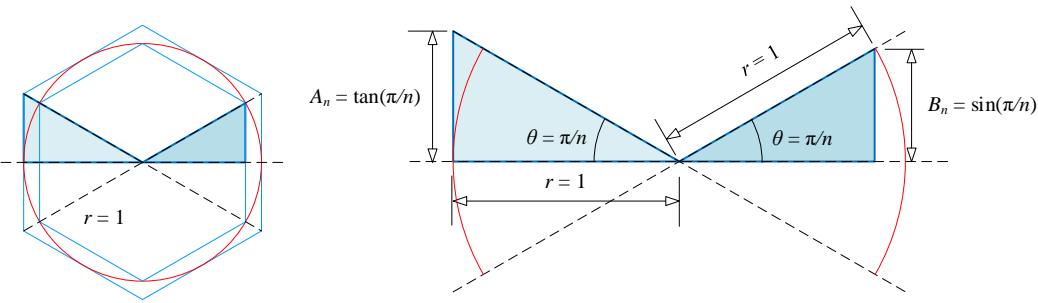


图 30. 圆形内接和外接估算圆周率

外切正多边形的周长是估算单位圆周长的上界：

$$2\pi < 2n \cdot \tan \frac{\pi}{n} \quad (11)$$

即，

$$\pi < n \cdot \tan \frac{\pi}{n} \quad (12)$$

内接正多边形的周长是估算单位圆周长的下界：

$$2n \cdot \sin \frac{\pi}{n} < 2\pi \quad (13)$$

即，

$$n \cdot \sin \frac{\pi}{n} < \pi \quad (14)$$

联合 (12) 和 (14)，可以得到圆周率估算的上下界：

$$n \cdot \sin \frac{\pi}{n} < \pi < n \cdot \tan \frac{\pi}{n} \quad (15)$$

如图 31 所示，随正多边形边数逐步增大，圆周率估算越精确。这张图中， n 不断增大时，绿色和蓝色两条曲线不断收敛 (converge) 于红色虚线，这个过程呈现出极限 (limit) 这一重要数学思想。

在数学上，收敛的意思可以是汇聚于一点，靠近一条线，向某一个值不断靠近。而逼近则是近似，代表高度相似，但是不完全相同。

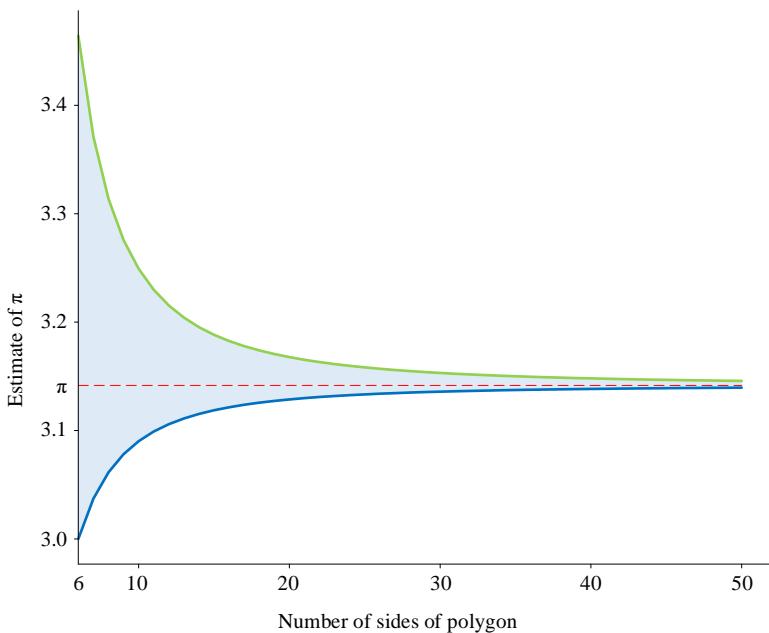
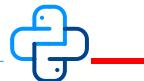


图 31. 随正多边形边数不断增大，圆周率估算越精确



代码文件 Bk3_Ch3_03.py 绘制图 31。



此外，我们还结合 Bk3_Ch3_02.py 和 Bk3_Ch3_03.py，用 Streamlit 制作了估算圆周率的 App，请大家参考代码文件 Streamlit_Bk3_Ch3_03.py。

阿基米德的方法

阿基米德采用另外一种方法，他先用外切和内接正 6 边形，然后逐次加倍边数，到正 12 边形、正 24 边形、正 48 边形，最后到正 96 边形。

根据图 30，对于正 n 边形，令

$$\begin{aligned} B_n &= n \cdot \sin \frac{\pi}{n} \\ A_n &= n \cdot \tan \frac{\pi}{n} \end{aligned} \tag{16}$$

B_n 是 π 的下限， A_n 是 π 的上限。当多边形边数加倍时，即从正 n 边形加倍到正 $2n$ 边形，阿基米德发现如下量化关系：

$$A_{2n} = \frac{2A_n B_n}{A_n + B_n} \\ B_{2n} = \sqrt{A_{2n} B_n}$$
(17)

利用三角恒等式，(17) 中两式不难证明，本书此处省略推导过程。

图 32 所示为阿基米德估算圆周率的结果，可见阿基米德方法收敛过程计算效率更高。

几何思维是刻在人类基因中的思维方式，不难理解为什么不同时空、不同地域的数学家，在最开始估算圆周率时，都不约而同想到用正多边形来近似。圆周率估算的竞赛依然不断进行，随着数学思想和工具的不断进步，新的方法不断涌现。沿着数学发展历史的脉络，本书后续将会介绍更多估算圆周率的估算方法。

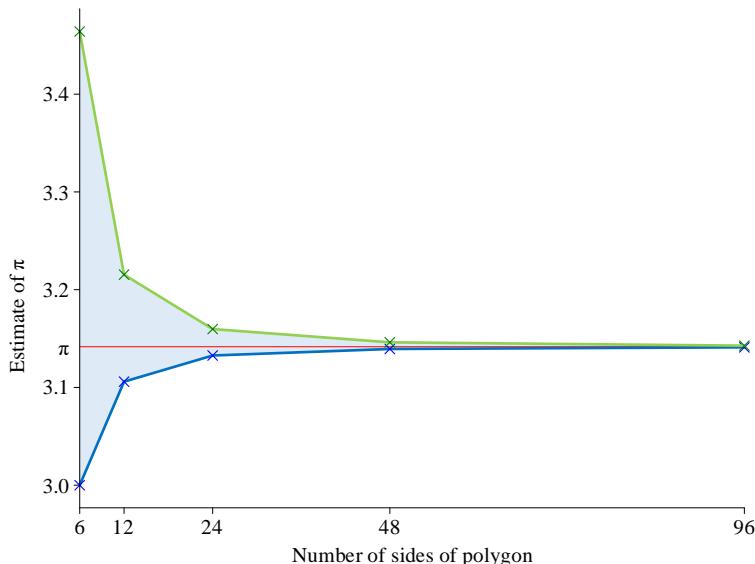


图 32. 阿基米德估算圆周率



Bk3_Ch3_04.py 绘制图 32。



本章蜻蜓点水地介绍了本书后续内容会用到的几何概念。但是，本书要讲述的几何的故事不止于此。

不久之后，在笛卡尔 (René Descartes, 1596 ~ 1650) 手里，几何和代数将完美结合。圆锥曲线很快便革新人类对天体运行规律的认知，颠覆人类的世界观。

斯蒂芬·霍金 (Stephen Hawking, 1942 ~ 2018) 曾说“等式是数学中最无聊的部分，我一直试图从几何视角理解数学。”本书作者也认为几何思维是人类的天然思维方式，因此在讲解数学概念、各种数据科学、机器学习算法时，我们都会多给出一个几何视角，以强化理解。

Algebra

4 代数

代数不过是公式化的几何



代数不过是公式化的几何；几何不过是图形化的代数。

Algebra is but written geometry and geometry is but figured algebra.

——索菲·热尔曼 (Sophie Germain) | 法国女性数学家 | 1776 ~ 1831



- ◀ `difference()` 计算集合的相对补集
- ◀ `interaction()` 计算集合的交集
- ◀ `numpy.roots()` 多项式求根
- ◀ `set()` 构造集合
- ◀ `subs()` 符号代数式中替换
- ◀ `sympy.abc` 引入符号变量
- ◀ `sympy.collect()` 合并同类项
- ◀ `sympy.cos()` 符号运算中余弦
- ◀ `sympy.expand()` 展开代数式
- ◀ `sympy.factor()` 对代数式进行因式分解
- ◀ `sympy.simplify()` 简化代数式
- ◀ `sympy.sin()` 符号运算中正弦
- ◀ `sympy.solvers.solve()` 符号方程求根
- ◀ `sympy.symbols()` 定义符号变量
- ◀ `sympy.utilities.lambdify.lambdify()` 将符号代数式转化为函数
- ◀ `union()` 计算集合的并集

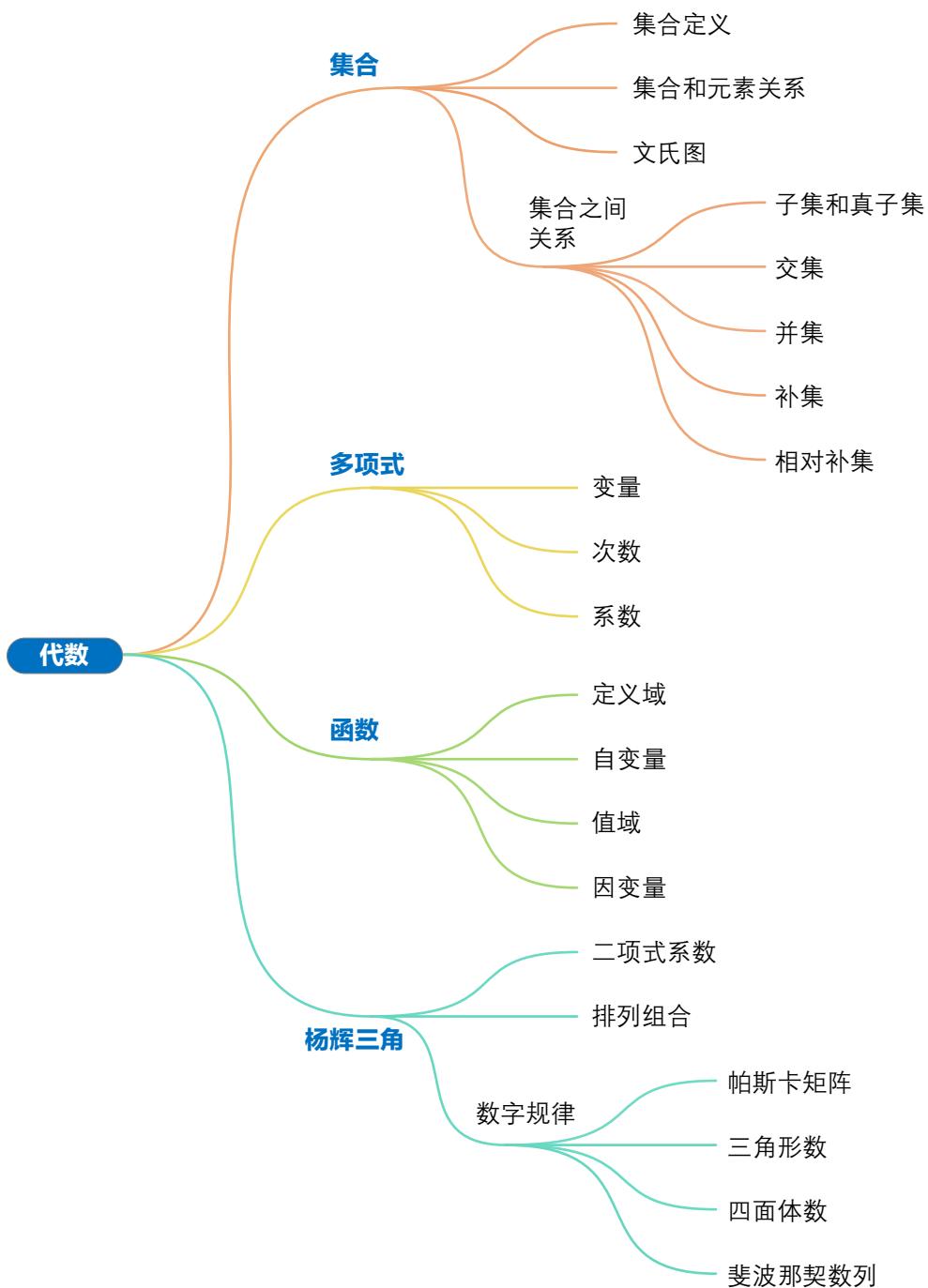
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

4.1 代数的前世今生：薪火相传

思想的传播像火种的接续传递——首先是星星之火，然后是闪烁的炬火，最后是燎原烈焰，排山倒海、势不可挡。

位于埃及境内的亚历山大图书馆曾经一度是古希腊最重要的图书馆，同时也是古希腊重要学术和文化中心。

公元前 47 年，亚历山大图书馆失火，大部分馆藏经典被焚毁。公元 529 年，柏拉图学园和其他所有雅典学校都被迫关闭。

可以想象，柏拉图学园断壁残垣，杂草丛生，物是人非。巢倾卵破，数学家、哲学家们鸟兽散去，远走他乡，衣食无着，寄人篱下，晚景凄凉。

古希腊学术圣火如风中之烛，渐渐燃灭，欧洲则一步步陷入漫漫暗夜。

庆幸的是，西方不亮东方亮；希腊典籍被翻译成阿拉伯语，人类思想的火种在另外一个避风港湾得以保全——巴格达“**智慧宫** (House of Wisdom)”。

在九世纪至十三世纪，智慧宫可以说是整个世界举足轻重的教育学术机构。在智慧宫，东西方科技知识交融发展。值得一提的是，印度十进制数字系统就是在阿拉伯进一步发展，并引入欧洲。因此，十进制数字也被称作阿拉伯数字。

花拉子密 (Muhammad ibn Musa al-Khwarizmi) 是一位波斯数学家、智慧宫的代表性学者。花拉子密在约 820 年，创作完成《代数学》 (*Al-Jabr*)，代数学自此成为一门独立学科。他第一次系统性求解一次方程及一元二次方程，因而被称为“代数之父”。英文中的代数 algebra 一词源自 *Al-Jabr*。值得一提的是，“算法”的英文 algorithm 一词来自于花拉子密 (al-Khwarizmi) 的名字。



图 1. 花拉子密《代数学》封面

好景不长，1258 年蒙古帝国军队的铁蹄大张挞伐，洗劫巴格达，焚毁智慧宫。据说，智慧宫珍贵藏书被丢弃在底格里斯河，河水被染黑长达六个月之久。

相比玉楼金殿、奇珍异宝，记录人类智慧的捆捆羊皮卷、叠叠莎草纸可能显得一分不值。这盏风中摇曳的烛火在两河流域被生生掐灭。

值得宽慰的是，十一世纪开始，十字远征军一次次远征，从阿拉伯人手中取回科学的火种，翻译运动在欧洲兴起，欧洲也渐渐从几百年的暗夜中苏醒。

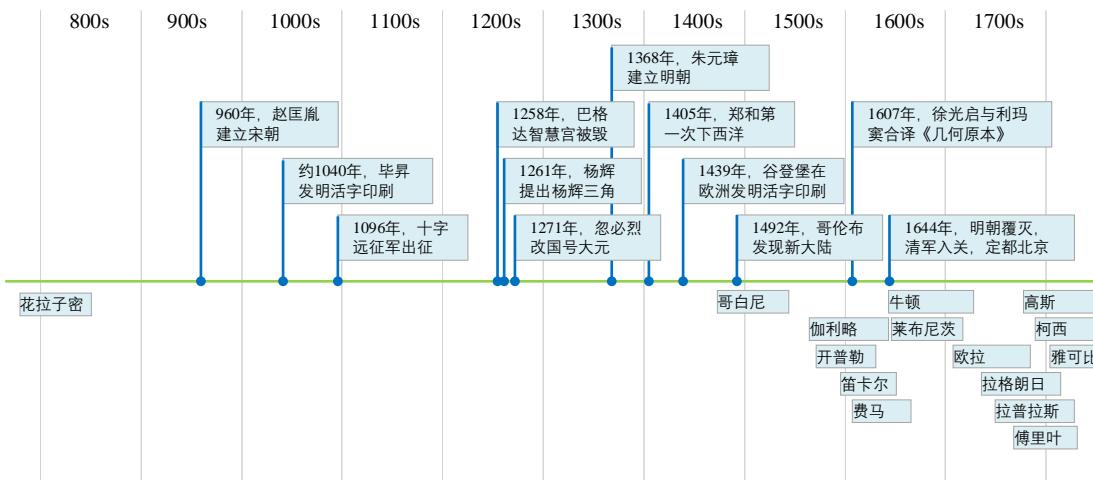


图 2. 西方数学复兴时间轴

十字远征军带回来不仅仅是古希腊的典籍，还有古印度的数学、古代中国的技术发明。这些科学知识在欧洲传播、发展，最终燃成人类思想的熊熊烈焰。

这片思想的火海中绽放出众多绚丽的火焰——伽利略、开普勒、笛卡尔、费马、帕斯卡、牛顿、莱布尼兹、伯努利、欧拉、拉格朗日、拉普拉斯、傅里叶、高斯、柯西 … 本系列丛书会提到他们的名字，以及他们给后来者留下的宝贵知识火种。

4.2 集合：确定的一堆东西

本节回顾集合这个概念。相信本书读者对**集合** (set) 这个概念已经不陌生，我们在本书第 1 章介绍过复数集、实数集、有理数集等等，它们都是集合。

集合是由若干确定的**元素** (member 或 element) 所构成的整体。集合可以分为：**有限元素集合** (finite set)、**无限元素集合** (infinite set) 和**空集** (empty set 或 null set)。

集合与元素

如图 3 所示，集合与元素的关系有两种：**属于** (\in)，**不属于** (\notin)。

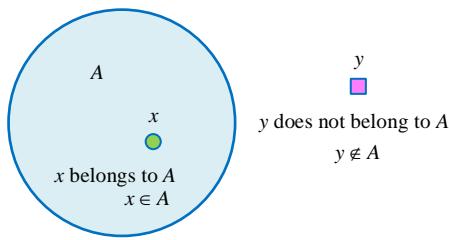


图 3. 属于和不属于

表 1. 集合与元素关系的中英文表达

英文表达	中文表达
x belongs to capital A .	x 属于 A 。
x is a member/element of the set capital A .	x 是集合 A 元素。
x is/lies in the set capital A .	x 在集合 A 之内。
The set capital A includes x .	集合 A 包含 x 。
y does not belong to the set capital A .	y 不属于集合 A 。
y is not a member of the set capital A .	y 不是集合 A 元素。

集合与集合

如果集合 A 中的每一个元素也都是集合 B 中的元素，那么 A 是 B 的**子集** (subset)，记作 $A \subseteq B$ 。而 B 是 A 的**母集**，也称**超集** (superset)。

如果同时满足 $A \subseteq B$ 和 $A \neq B$ ，则称 A 是 B 的**真子集** (A is a proper subset of B)，记作 $A \subset B$ 。

给定 A 和 B 两个集合，由所有属于 A 且属于 B 的元素所组成的集合，叫做 A 与 B 的**交集** (intersection)，记作 $A \cap B$ 。

A 和 B 所有的元素合并组成的集合，叫做 A 和 B 的**并集** (union)，记作 $A \cup B$ 。

补集 (complement) 一般指**绝对补集** (absolute complement)。设 Ω 是一个集合， A 是 Ω 的一个子集，由 Ω 中所有不属于 A 的元素组成的集合，叫做子集 A 在 Ω 中的绝对补集。

A 中 B 的**相对补集** (relative complement)，是所有属于 A 但不属于 B 的元素组成的集合，记做 $A \setminus B$ 或 $A - B$ (set difference of A and B)，也可以读作“ B 在 A 中的相对补集 (the relative complement of B with respect to set A)”。

表 2. 集合与集合关系英文表达

数学表达	英文表达
$A \subset B$	The set capital A is a subset of the set capital B . The set capital B is a superset of the set capital A . The set capital A is contained in the set capital B .
$A \subseteq B$	The set capital A is a subset of or equal to the set capital B .
$A \supset B$	The set capital A contains the set capital B .
$A \supseteq B$	The set capital A contains or is equal to the set capital B .
$A \cap B$	The intersection of the set capital A and the set capital B . A intersection B

	The intersection of A and B
$A \cup B$	The union of the set capital A and the set capital B . Capital A union capital B . The union of A and B .
\bar{A}	The complement of the set capital A .
$A - B$	The relative complement of the set capital B in the set capital A . The relative complement of set capital B with respect to set capital A .
$A \cap (B \cup C)$	The intersection of capital A and the set capital B union capital C .
$(\bar{A} \cup B)$	The complement of the set capital A union capital B .
$\bar{A} \cap \bar{B}$	The intersection of the complement of capital A and the complement of capital B .

文氏图

集合之间的关系也可以用**文氏图** (Venn diagram) 表达。图 4 给出的是两个集合常见关系文氏图。

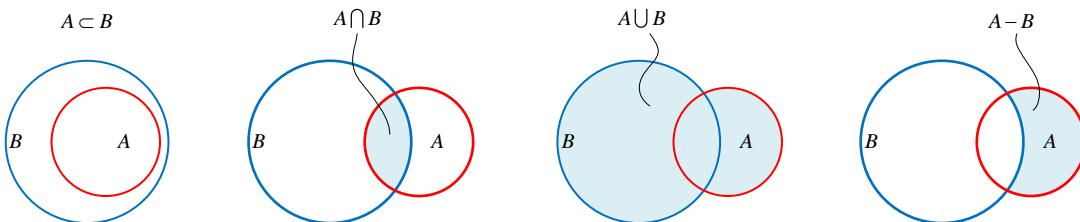


图 4. 两个集合关系文氏图

掷骰子

举个例子，如图 5 所示，掷一枚色子，点数结果构成的集合 Ω 为：

$$\Omega = \{1, 2, 3, 4, 5, 6\} \quad (1)$$

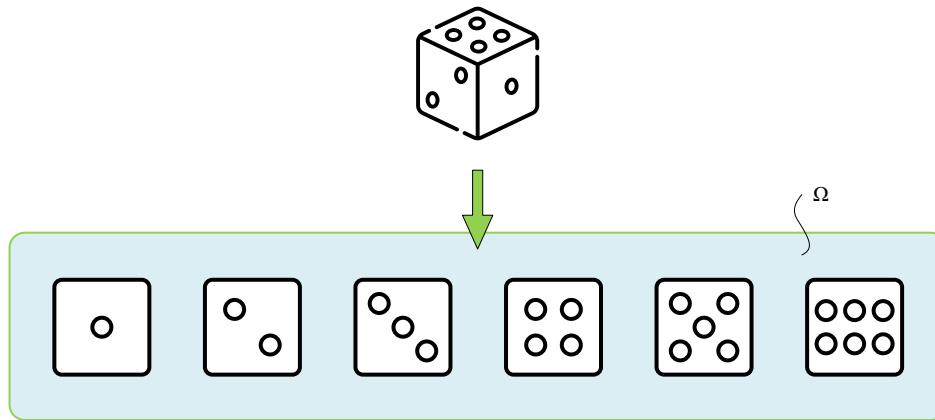


图 5. 投一枚色子点数结果

定义集合 A 为色子点数为奇数：

$$A = \{1, 3, 5\} \quad (2)$$

集合 B 为色子点数为偶数：

$$B = \{2, 4, 6\} \quad (3)$$

集合 C 为色子点数小于 4：

$$C = \{1, 2, 3\} \quad (4)$$

图 6 所示为 A 、 B 、 C 三个集合关系。

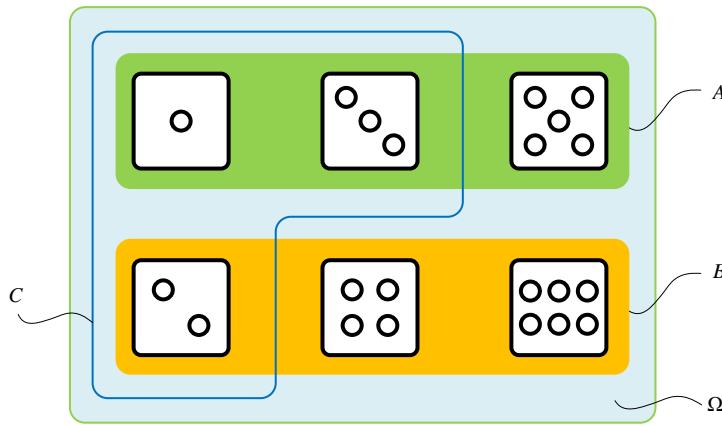


图 6. 色子点数 A 、 B 、 C 三个集合关系

显然， A 和 B 的交集为空，即，

$$A \cap B = \emptyset \quad (5)$$

A 和 B 的并集为全集 Ω ，即，

$$A \cup B = \Omega = \{1, 2, 3, 4, 5, 6\} \quad (6)$$

也就是说， A 在 Ω 中的绝对补集为 B 。

A 和 C 的交集有两个元素：

$$A \cap C = \{1, 3\} \quad (7)$$

A 和 C 的并集有四个元素，即，

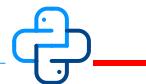
$$A \cup C = \{1, 2, 3, 5\} \quad (8)$$

A 中 C 的相对补集为：

$$A - C = \{5\} \quad (9)$$

C 中 A 的相对补集为：

$$C - A = \{2\} \quad (10)$$



代码文件 Bk3_Ch4_01.py 上述计算。

4.3 从代数式到函数

算数 (arithmetic) 基于**已知量** (known values)。而**代数** (algebra) 基于**未知量** (unknown values)，也称作**变量** (variables)。当然，代数中既有数字也有字母。

现代人一般用 a 、 b 、 c 等代表常数，用 x 、 y 、 z 等代表未知量。这种记法正是约 400 年前笛卡尔提出的。

引入未知量这种数学工具，有助于将数学问题抽象化、一般化。也就是说， $2 + 1$ 、 $6 + 12$ 、 $100 + 150$ 等算式，都可以抽象地写成 $x + y$ 这个代数式。

如图 7 所示，这五个圆形大小明显不同。但是，引入半径 r 这个变量，这些圆形的周长都可以写成 $2\pi r$ ，面积可以写成 πr^2 。将不同的 r 值代入代数式，便可以求得对应圆的周长和面积。

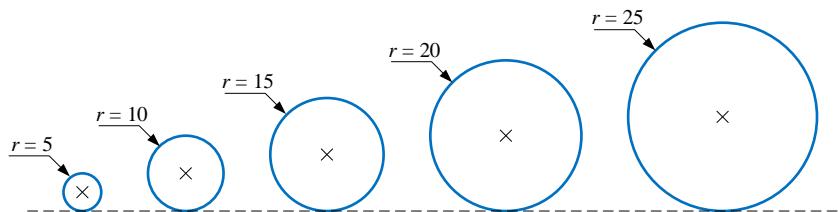


图 7. 不同半径的圆形

多项式

本书最常见的代数式是**多项式** (polynomial)，形如：

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (11)$$

其中， x 为**变量** (variable)， n 为**多项式次数** (degree of a polynomial)， a_0 、 a_1 … a_n 为**系数** (coefficient)。

系数之所以会使用**下角标** (subscript)，是因为字母不够用。类似地，变量多时， x 、 y 、 z 肯定不够用，本系列丛书会用变量加下角标序号 (索引) 来表达变量，比如 x_1 、 x_2 、 x_3 、 x_i 、 x_j 等等。

由数和字母的积组成的代数式叫做**单项式** (monomial)，比如 $a_n x^n$ 。单独的一个数或一个系数也叫做单项式，比如 5 和 a_0 。

一个单项式中，所有变量指数之和，叫做这个单项式的次数。比如， $3x^5$ 的次数是 5， $2xy$ 的次数为 2 ($= 1 + 1$)。

如 (11) 所示，多项式则是由一个个单项式加减构成。

只有一个变量的多项式被称作**一元多项式** (univariate polynomial)。变量多于一个的多项式统称**多元多项式** (multivariate polynomials)。特别地，有两个变量的多项式常被称作**二元多项式** (bivariate polynomial)，比如 $x + y + 8$ 或 $x_1 + x_2 + 8$ 。

最高项次数较小的多项式都有特殊的名字，比如**常数式** (constant equation)、**一次式** (linear equation)、**二次式** (quadratic equation)、**三次式** (cubic equation)、**四次式** (quartic equation) 和**五次式** (quintic equation) 等等。常见多项式及名称总结于表 3 中。

表 3. 常见多项式举例

次数	英文名称	例子
1	linear	$ax + b, \quad a \neq 0$
2	quadratic	$ax^2 + bx + c, \quad a \neq 0$
3	cubic	$ax^3 + bx^2 + cx + d, \quad a \neq 0$
4	quartic	$ax^4 + bx^3 + cx^2 + dx + e, \quad a \neq 0$
5	quintic	$ax^5 + bx^4 + cx^3 + dx^2 + ex + f, \quad a \neq 0$

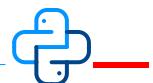
代入具体值

给定如下变量为 x 的三次式：

$$x^3 + 2x^2 - x - 2 \tag{12}$$

令 $x = 1$ ，代入 (12)，得到如下结果：

$$1^3 + 2 \times 1^2 - 1 - 2 = 0 \tag{13}$$

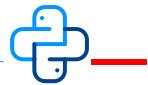


代码文件 `Bk3_Ch4_02.py` 中用 SymPy 中函数来完成上述计算。其中 `sympy.abc` 引入符号变量 x 和 y 。SymPy 是重要的符号运算库，本书将会用到其中的代数式定义、求根、求极限、求导、求积分、级数展开等功能。此外，也可以使用 `sympy.symbols()` 定义更复杂的符号变量。

同样，可以利用`.subs()` 将(12)中的 x 替换成其他符号变量、甚至代数表达式，比如 $x = \cos(y)$ ：

$$(\cos(y))^3 + 2(\cos(y))^2 - \cos(y) - 2 \quad (14)$$

代码文件 `Bk3_Ch4_02.py` 还将 x 替换为 $\cos(y)$ 。



代码文件 `Bk3_Ch4_03.py` 介绍了 SymPy 中几个处理代数式的函数。`sympy.simplify()` 函数可以简化代数式。`sympy.expand()` 可以用来展开代数式。`sympy.factor()` 函数则可以对代数式进行因式分解。`sympy.collect()` 函数用来合并同类项。请大家自行学习。

表 4 总结常用代数式的英文表达。

表 4. 常用代数英文表达

数学表达	英文表达
$x - y$	x minus y
$-x - y - z$	minus x minus y minus z
$x - (y - z)$	x minus the quantity y minus z . x minus open parenthesis y minus z close parenthesis.
$x - (y + z) - t$	x minus the quantity y plus z end of quantity minus t . x minus open parenthesis minus y plus z close parenthesis minus t .
$x(x - y + z)$	x times the quantity x minus y plus z . x times open parenthesis x minus y plus z close parenthesis.
$(x + y)^2$	x plus y all squared
x^3	x to the third x to the third power x raised to the power of three x cubed
$x^5 + 4x^3 - 2x^2$	x to the fifth plus four x to the third minus two x squared
$(x + y)(z + t)$	The sum x plus y times the sum z plus t . The product of the sum x plus y and the sum z plus t . Open parenthesis x plus y close parenthesis times open parenthesis z plus t close parenthesis.
$\left(\frac{x}{y}\right)^2$	x over y all squared
$\frac{x+z}{y}$	The quantity of x plus z divided by y .
$x + \frac{z}{y}$	x plus the fraction z over y .
$t\left(z + \frac{x}{y}\right)$	t times the sum z plus the fraction x over y .

函数

函数 (function) 一词由 **莱布尼兹** (Gottfried Wilhelm Leibniz) 引入。而 $f(x)$ 这个函数记号由 **欧拉** (Leonhard Paul Euler) 发明。欧拉还引入三角函数现代符号，他首创以 e 表记自然对数的底，用希腊字母 Σ 表记累加，以 i 表示虚数单位。

中文“函数”则是由清朝数学家李善兰 (1810 ~ 1882) 翻译。代数、系数、指数、多项式等数学名词中文翻译也是出自李善兰之手。

给定一个集合 X ，对 X 中元素 x 施加映射法则 f ，记作函数 $f(x)$ 。得到的结果 $y = f(x)$ 属于集合 Y 。集合 X 称作**定义域** (domain)， Y 称为**值域** (codomain)。 x 称作**自变量** (an argument of a function 或 an independent variable)， y 称作**因变量** (dependent variable)。

大家应该已经发现，函数 $f: X \rightarrow Y$ 有三个关键要素：定义域 X 、值域 Y 和函数映射规则 f 。

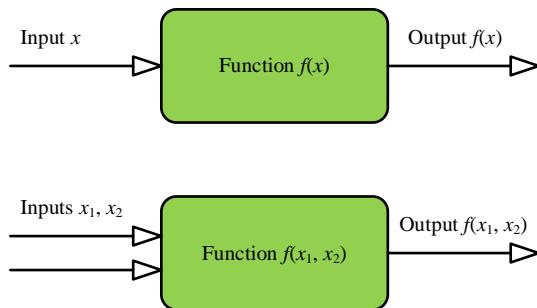
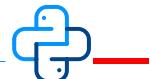


图 8. 一元函数、二元函数的映射

函数的自变量为两个或两个以上时，叫做**多元函数** (multivariate function)。本丛书一般会使用 x 加下角标序号来表达多元函数中的自变量，比如 $f(x_1, x_2, \dots, x_D)$ 函数有 D 个自变量。

为了方便将不同的 x 值代入 (12)，我们可以定义一个函数 $f(x)$ ：

$$f(x) = x^3 + 2x^2 - x - 2 \quad (15)$$



Bk3_Ch4_04.py 将代数式转化为函数，并给 x 赋值得到函数值。

表 5 给出有关函数常用英文表达。

表 5. 常用函数英文表达

数学表达	英文表达
$f(x)$	f_x f of x The function f of x
$f(g(x))$	f composed with g of x f of g of x
$f \circ g(x)$	f composed with g of x f of g of x
$f(x+a)$	f of the quantity x plus a
$f(x, y)$	f of x, y

$f(x_1, x_2, \dots, x_n)$	f of x sub one, x sub two, dot dot dot, x sub n
$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$	f of x equals a sub n times x to the n , plus a sub n minus one times x to the n minus one, plus dot dot dot, plus a sub one times x , plus a sub zero. f of x equals a sub n times x raised to the power of n , plus a sub n minus one times x raised to the power of n minus one, plus dot dot dot, plus a sub one times x , plus a sub zero.
$f(x) = 3x + 5$	f of x equals three times x plus five.
$f(x) = x^2 + 2x + 1$	f of x equals x squared plus two times x plus one.
$f(x) = x^3 - x + 1$	f of x equals x cubed minus x plus one.

为了进一步探讨函数性质，我们亟需一个重要的数学工具——**坐标系** (coordinate system)。坐标系是下一章探讨的内容。

4.4 杨辉三角：代数和几何的完美合体

杨辉三角，也称贾宪三角，又称**帕斯卡三角** (Pascal's triangle)，是二项式系数的一种写法。

二项式系数

如下， $(x+1)^n$ 展开后，按单项 x 的次数从高到低排列，发现单项式系数呈现出特定规律：

$$\begin{aligned}
 (x+1)^0 &= 1 \\
 (x+1)^1 &= x+1 \\
 (x+1)^2 &= x^2 + 2x + 1 \\
 (x+1)^3 &= x^3 + 3x^2 + 3x + 1 \\
 (x+1)^4 &= x^4 + 4x^3 + 6x^2 + 4x + 1 \\
 (x+1)^5 &= x^5 + 5x^4 + 10x^3 + 10x^2 + 5x + 1 \\
 (x+1)^6 &= x^6 + 6x^5 + 15x^4 + 20x^3 + 15x^2 + 6x + 1 \\
 &\dots && \dots
 \end{aligned} \tag{16}$$

图 9 将 (16) 单项式系数以金字塔的结构展示，请读者注意以下规律：

- ◀ 三角形系数呈现对称性，第 k 行有 $k+1$ 个系数；
- ◀ 三角形每一行左右最外侧系数为 1；
- ◀ 除最外两侧系数以外，三角形内部任意系数为左上方和右上方两个系数之和；
- ◀ 第 k 行系数之和为 2^k 。

⚠ 注意，(16) 的第一层对应 $k=0$ 。

杨辉三角中，我们将会看到几何、代数、概率等知识的有趣联系。

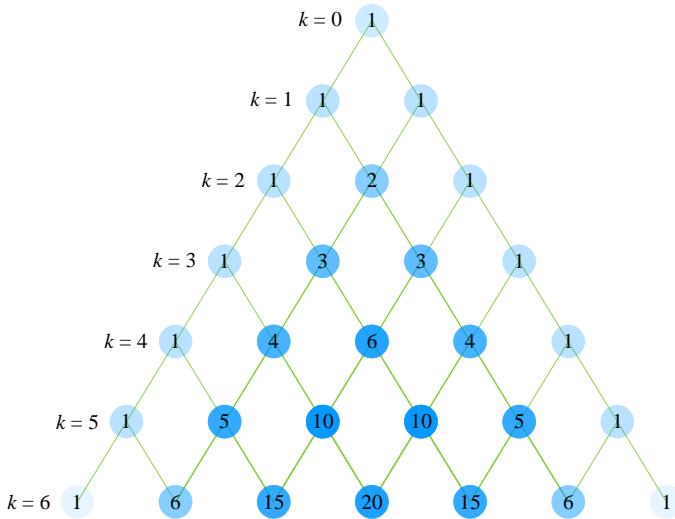


图 9. 杨辉三角

从杨辉三角到概率

比帕斯卡提前 300 年左右，杨辉在自己书中介绍了这个数字规律。杨辉也不是第一位发现者，他在书中也很清楚，这个规律引自贾宪的一部叫做《释锁算术》的数学作品。

按照时间先后顺序，贾宪在 11 世纪北宋时期就发现并推广了这一规律，杨辉只是在 13 世纪南宋时期再次解释。

而帕斯卡等到 1655 年在自己的作品中介绍二项式系数规律。但是，帕斯卡创造性地将它用在解释概率运算，这对概率论发展有开天辟地之功。

概率论是本书第 20 章要介绍的内容。本系列丛书《概率统计》一册将专门介绍概率统计相关内容。

古法七乘方

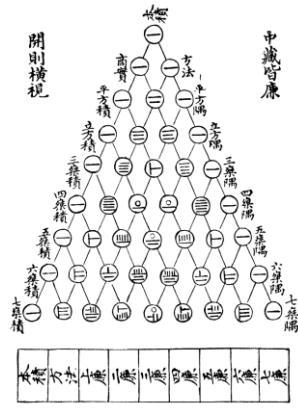


图 10. 元代数学家朱世杰《四元玉鉴》中绘制的杨辉三角

火柴梗图

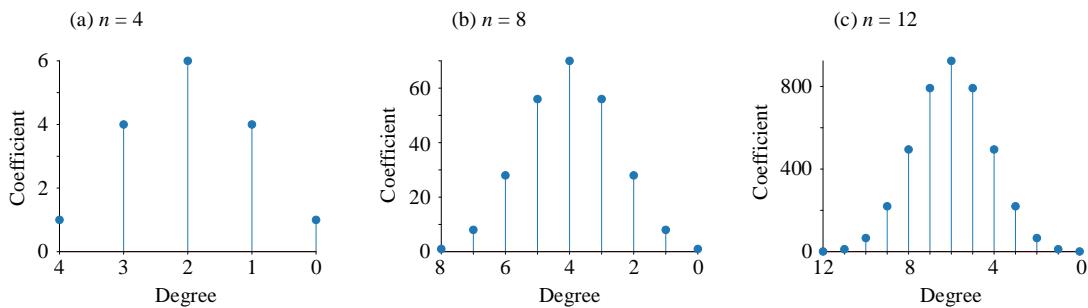
火柴梗图 (stem plot) 可可视化杨辉三角每行单项式系数的规律。图 11 所示为 $n = 4, 8, 12$ 时，二项式展开单项系数规律。

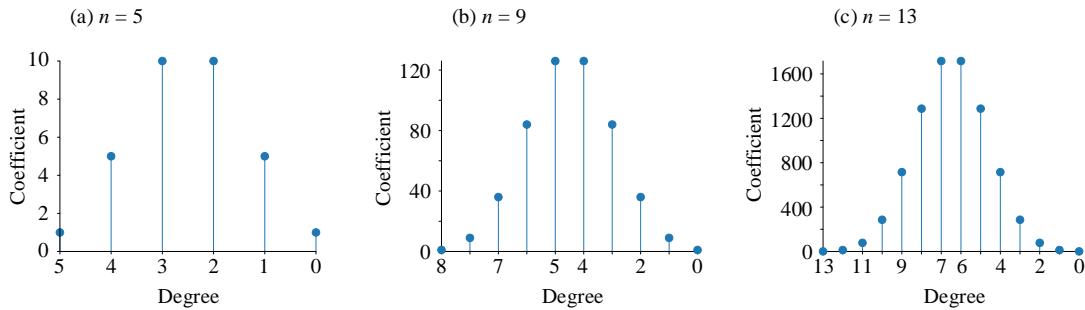
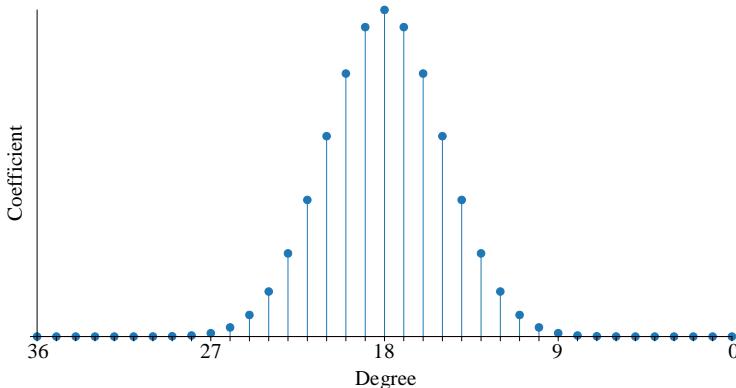
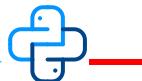
火柴梗图明显呈现中心对称性。 n 为偶数时，对称轴处系数最大。如图 11 所示， n 为奇数时，对称轴附近两个系数为最大值。对称轴左右两侧系数先快速减小，然后再缓慢减小。

随着 n 增大，这一现象更加明显，如图 13 所示。连接图 13 中实心点，我们发现一条优美的曲线呼之欲出，这条曲线就是**高斯函数** (Gaussian function)。



本书第 12 章将介绍高斯函数。

图 11. $n = 4, 8, 12$ 等偶数时，二项式展开单项系数

图 12. $n = 5, 9, 13$ 等奇数时，二项式展开单项系数图 13. $n = 36$ 时，二项式展开单项系数

代码文件 Bk3_Ch4_05.py 绘制图 11 和图 13 两图。



我们在 Bk3_Ch4_05.py 基础上用 Streamlit 制作了展示二项式系数的 App，请大家参考代码文件 Streamlit_Bk3_Ch4_05.py。

4.5 排列组合让二项式系数更具意义

组合数

从 n 个不同元素中，取 m ($m \leq n$) 个元素构成一组，被称作 n 个不同元素中取出 m 个元素的一个组合 (combination)。

⚠ 注意，对于组合来说，组内的元素排序并不重要。

n 个不同元素中取出 m 个元素的所有组合个数叫做组合数，常记做 C_n^m ：

$$C_n^m = C(n, m) = \frac{n!}{(n-m)!m!} \quad (17)$$

其中，! 运算符就是本书第 2 章介绍的阶乘 (factorial)。

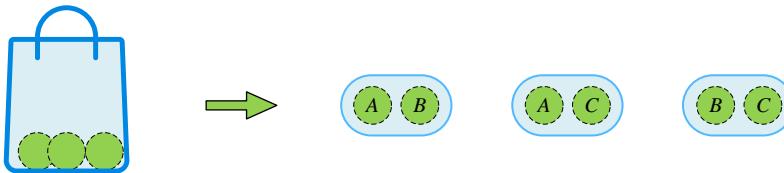


图 14. A 、 B 、 C 三个元素无放回抽取两个，结果有三个组合

举个例子，如图 14 所示，从 A 、 B 、 C 三个元素无放回抽取两个，只要元素相同，不管次序是否相同都算作相同结果。结果有三个组合 AB 、 AC 、 BC ，对应的组合数为：

$$C_3^2 = \frac{3!}{(3-2)!2!} = \frac{6}{2} = 3 \quad (18)$$

逐个抽取个体时，每个被抽到的个体不再放回总体，也就是不再参加下一次抽取，这就是“无放回抽取”。无放回抽取中，总体在抽样过程中逐渐减小。



Bk3_Ch4_06.py 完成上述无放回抽取组合实验。

结果如下。

```
('A', 'B')
('A', 'C')
('B', 'C')
```

组合数表达杨辉三角

用组合数将 $(x + y)^n$ 展开写成：

$$(x + y)^n = C_n^0 x^n y^0 + C_n^1 x^{n-1} y^1 + C_n^2 x^{n-2} y^2 + \cdots + C_n^{n-2} x^2 y^{n-2} + C_n^{n-1} x^1 y^{n-1} + C_n^n x^0 y^n \quad (19)$$

因此，杨辉三角写成图 15 这种形式。

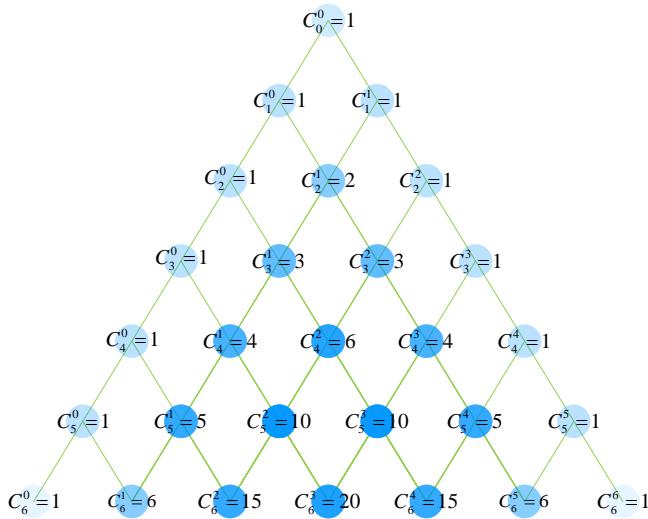


图 15. 用组合数来写杨辉三角

组合数方便解释(19)中各项系数。(19)每一项 x 和 y 的次数之和为 n , 如果某一单项 y 的次数为 k , x 的次数为 $n-k$, 这一项为 $C_n^k x^{n-k} y^k$ 。该项系数 C_n^k 相当于在 n 个 x 或 y 连乘中, 选取 k 个为 y 。

将 $x=y=1$ 代入(19), 可以发现组合数的一个重要规律:

$$2^n = C_n^0 + C_n^1 + C_n^2 + \cdots + C_n^{n-2} + C_n^{n-1} + C_n^n \quad (20)$$

观察图 15 的对称性, 容易发现另外一个组合数规律:

$$C_n^k = C_n^{n-k} \quad (21)$$

排列数

从 n 个不同元素中, 先后取 m ($m \leq n$) 个元素排成一列, 叫做从 n 个元素中取出 m 个元素的一个**排列**(permutation)。排列中, 元素的排序很重要。

n 个不同元素中取出 m 个元素的所有排列的个数叫做排列数, 常记做 P_n^m :

$$P_n^m = P(n, m) = \frac{n!}{(n-m)!} \quad (22)$$

同样, 如图 16 所示, 从 A 、 B 、 C 三个元素无放回先后抽取两个, 结果有 6 个排列 AB 、 BA 、 AC 、 CA 、 BC 、 CB , 即,

$$P_3^2 = \frac{3!}{(3-2)!} = 6 \quad (23)$$

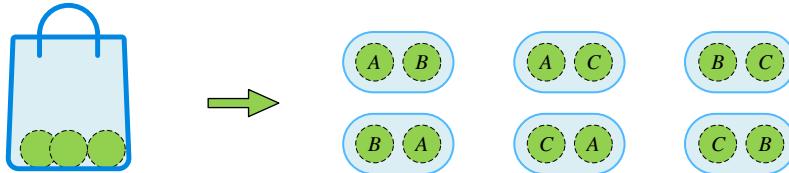
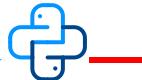


图 16. A、B、C 三个元素无放回抽取两个，结果有 6 个排列



Bk3_Ch4_07.py 完成上述无放回排列实验。

结果为：

```
('A', 'B')
('A', 'C')
('B', 'A')
('B', 'C')
('C', 'A')
('C', 'B')
```

组合数和排列数

比较 (17) 和 (22)，发现排列和组合的关系为：

$$P_n^m = C_n^m \cdot m! \quad (24)$$

可以这样解释上式，先从 n 个元素取出 m 进行组合，组合数为 C_n^m 。然后，把 m 个元素全部排列一遍（也叫全排列），排列数为 $m!$ 。这样， C_n^m 和 $m!$ 乘积便是 n 个元素取出 m 的排列数。



从 A、B、C 三个元素全排列的结果为 ABC、ACB、BAC、BCA、CAB、CBA。代码文件 Bk3_Ch4_08.py 完成上述计算并打印全排列结果。

结果为：

```
('A', 'B', 'C')
('A', 'C', 'B')
('B', 'A', 'C')
('B', 'C', 'A')
('C', 'A', 'B')
('C', 'B', 'A')
```

4.6 杨辉三角隐藏的数字规律

本节简要探讨杨辉三角中隐藏着的有趣数字规律。

帕斯卡矩阵

将杨辉三角数字左对齐，可以得到如下矩阵。这个矩阵常被称作**帕斯卡矩阵** (Pascal matrix):

$$\begin{bmatrix} 1 \\ 1 & 1 \\ 1 & 2 & 1 \\ 1 & 3 & 3 & 1 \\ 1 & 4 & 6 & 4 & 1 \\ 1 & 5 & 10 & 10 & 5 & 1 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{bmatrix} \quad (25)$$

三角形数

(25) 矩阵的第一列均为 1，第二列为自然数，第三列为**三角形数** (triangular number)。

如图 17 所示，如果一定数量圆形紧密排列，可以形成一个等边三角形，这个数量就叫做三角形数。

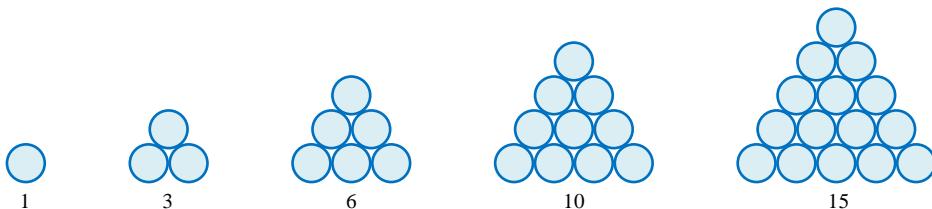


图 17. 三角形数

四面体数

(25) 第四列是叫做**四面体数** (tetrahedral number 或 triangular pyramidal number)。

顾名思义，四面体数就是圆球紧密堆成四面体对应的数字。三角数从 1 累加便可以得到四面体数。也就是把图 17 看成是圆球，将它们一层层摞起来，便得到正四面体。

斐波那契数列

按照图 18 浅黄色线条方向，将杨辉三角每一排数字相加，可以得到如下数字序列：

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55 \quad (26)$$

这便是斐波那契数列 (Fibonacci sequence)。

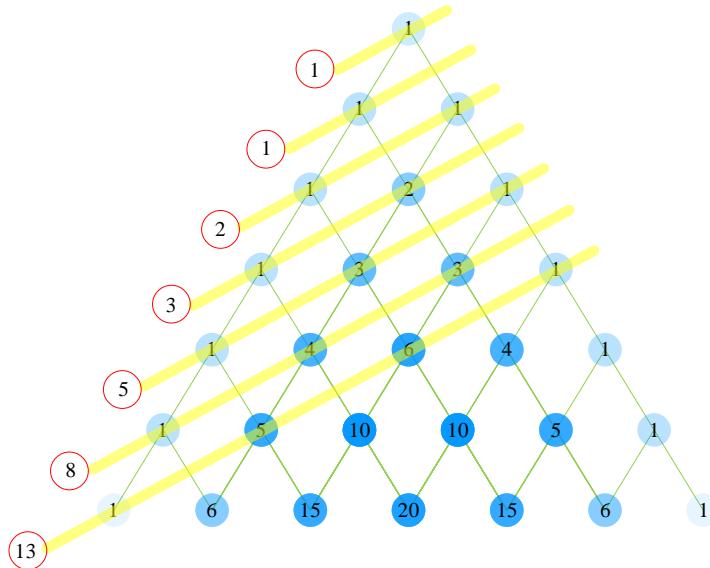


图 18. 杨辉三角和斐波那契数列关系

4.7 方程组：求解鸡兔同笼问题

方程

方程 (equation) 就是含有未知量的等式，比如 $x + 5 = 8$ 。使等式成立的未知量的值叫做方程的根 (root) 或解 (solution)。

一元一次方程 (linear equation in one variable) 可以写成：

$$ax + b = c \quad (27)$$

其中， x 为未知变量， a 、 b 、 c 为实数，且 $a \neq 0$ 。

二元一次方程 (linear equation in two variables)，可以写成：

$$ax + by = c \quad (28)$$

其中， x 和 y 为未知变量， a 、 b 、 c 为实数， $a \neq 0$ 且 $b \neq 0$ 。

用 x_1 和 x_2 作为未知量，(28) 也可以写成：

$$ax_1 + bx_2 = c \quad (29)$$

方程组

方程组 (system of equations) 是指两个或两个以上的方程，一般也会对应两个或两个以上未知量。

约 1500 年前成书的《孙子算经》中记载的“鸡兔同笼”就可以写成二元一次方程组。

鸡兔同笼问题原文是“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”

用现代汉语来说就是：现在笼子里有鸡（雉读作 zhì）和兔子在一起。从上面数一共有 35 个头，从下面数一共有 94 只脚，问一共有多少只鸡、多少只兔子？

用 x 代表鸡， y 代表兔。有 35 个头对应如下方程式：

$$x + y = 35 \quad (30)$$

有 94 只脚对应如下方程式：

$$2x + 4y = 94 \quad (31)$$

联立两个等式得到方程组：

$$\begin{cases} x + y = 35 \\ 2x + 4y = 94 \end{cases} \quad (32)$$

很容易求得，

$$\begin{cases} x = 23 \\ y = 12 \end{cases} \quad (33)$$

也就是，笼子里有 23 只鸡，12 只兔。



本书会在坐标系、线性代数这两个话题中继续有关鸡兔同笼故事。

一元二次方程

一元二次方程 (quadratic equation in one variable) 可以写成：

$$ax^2 + bx + c = 0 \quad (34)$$

其中， a 、 b 、 c 都是实数，且 $a \neq 0$ 。

(34) 的求根公式可以写成：

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (35)$$

(34) 判别式 (discriminant) 是,

$$\Delta = b^2 - 4ac \quad (36)$$

$\Delta > 0$, 一元二次方程有两个实数根; $\Delta = 0$, 一元二次方程有两个相同实数根; $\Delta < 0$, 一元二次方程有两个不同的复数根, 不存在实数根。

多项式求根

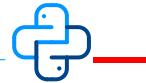
采用 `numpy.roots()` 也可以计算多项式方程的根。给定如下多项式等式:

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 = 0 \quad (37)$$

单项式次数从高到低各项系数作为输入, 用 `numpy.roots([a0, a1, ..., an-1, an])` 函数来求根。此外, `sympy.solvers.solve()` 函数也可以用来求根。

举个例子, 给定三次多项式等式:

$$-x^3 + 0 \cdot x^2 + x + 0 = 0 \quad (38)$$

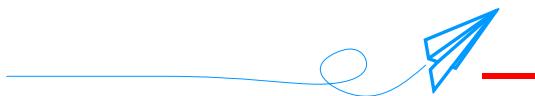


代码文件 `Bk3_Ch4_09.py` 求解 (38) 三个根。

表 6 所示为方程相关的英文表达。

表 6. 方程相关英文表达

$x(y+1)=5$	x times the quantity of y plus one equals five.
$(x+a)(x+b)=0$	The quantity of x plus a times the quantity of x plus b equals zero.
$2x + y = 5$	Two x plus y equals five.
$2x^2 + 3x + 4 = 0$	Two x squared plus three x plus four equals zero.
$2x^3 + 3x^2 + 4 = 0$	Two x cubed plus three x squared plus four equals zero.
$(x+y)/2x=0$	The quantity of x plus y over two x equals zero.
$(x+y)^n=1$	The quantity x plus y to the n th power equals one.
$x^n + x^{n-1} = 5$	x to the n , plus x to the n minus one equals five.



有时候, 知识的传播好似随风潜入夜。更多时候, 是水火不容的碰撞和残酷血腥的竞争。然而, 这场抢占知识高地的竞争从未偃旗息鼓, 大有愈演愈烈之势。

拿破仑曾感叹“数学的发展与国运息息相关”。让数学思想之火熊熊燃烧的是一代代栋梁之才，和保护炬火、席卷八荒的强大力量。两者互为给养、风雨同舟、荣辱与共。

在图2中西方代数复兴的时间轴上，一方面我们看到数学无国界，她在世界各地辗转腾挪、断续发展；另一方面，这个历史的脉络也让人们看到人才培养、聚集、转移，伴随着财富、军事、生产力、政治影响力此消彼长。

阿基米德的血肉之躯不能挡住罗马士兵的刀刃；但是，他的精巧发明曾一度让强敌闻之色变。知识不等同于汗牛充栋、蛛网尘封的藏书，两者可谓天壤之差、云泥之别。掌握、利用知识，让知识成为生产力，才是关键。

5

Cartesian Coordinate System

笛卡尔坐标系

几何代数一相逢，便胜却人间无数



我思，故我在。

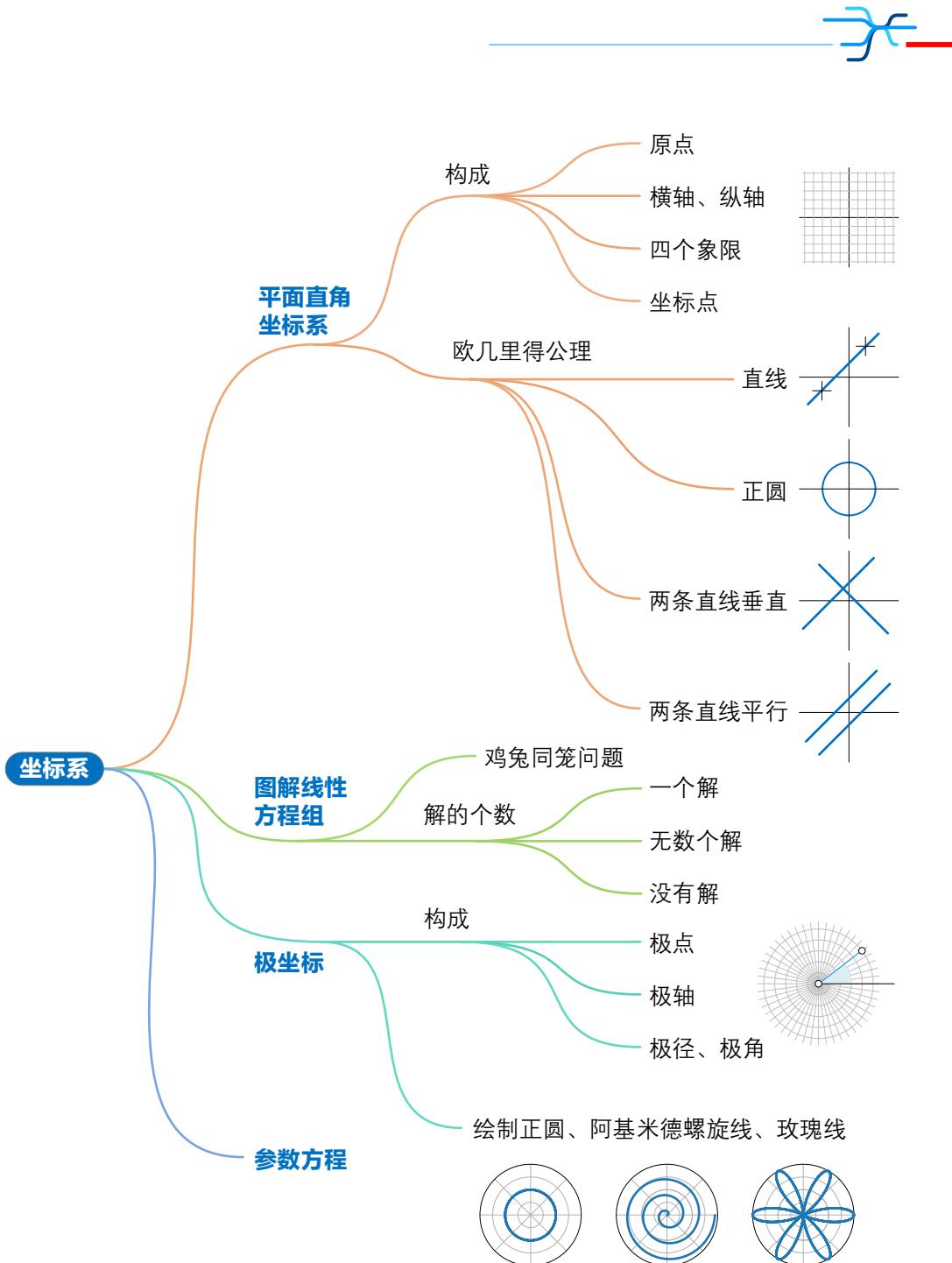
I think, therefore I am.

Cogito ergo sum.

—— 勒内·笛卡尔 (René Descartes) | 法国哲学家、数学家、物理学家 | 1596 ~ 1650



- ◀ Axes3D.plot_surface() 绘制三维曲面
- ◀ matplotlib.pyplot.axhline() 绘制水平线
- ◀ matplotlib.pyplot.axvline() 绘制竖直线
- ◀ matplotlib.pyplot.plot() 绘制线图
- ◀ matplotlib.pyplot.scatter() 绘制散点图
- ◀ matplotlib.pyplot.text() 在图片上打印文字
- ◀ numpy.meshgrid() 生成网格数据
- ◀ plot_parametric() 绘制二维参数方程
- ◀ plot3d_parametric_line() 绘制三维参数方程
- ◀ seaborn.pairplot() 成对散点图
- ◀ seaborn.scatterplot() 绘制散点图
- ◀ sympy.is_decreasing() 判断符号函数的单调性



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

5.1 笛卡尔：我思故我在

笛卡尔 (René Descartes) 在《方法论》(Discourse on the Method) 中写道：“在我看来，任何事情都值得怀疑，但是这个正在思考的个体——我——一定存在。这样，我便得到第一条真理——我思故我在。”



勒内·笛卡尔 (René Descartes)
法国哲学家、数学家和科学家 | 1596年 ~ 1650年
解析几何之父



这一天，房间昏暗，笛卡尔躺在床上、百无聊赖，可能在思考“存在”的问题。一只不速之客闯入他的视野，笛卡尔把目光投向房顶，发现一只苍蝇飞来飞去、嘤嘤作响。

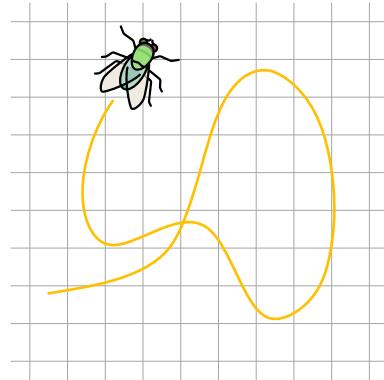


图 1. 笛卡尔眼中的苍蝇飞行

突然之间，一个念头在这个天才的大脑中闪过——要是在屋顶画上方格，我就可以追踪苍蝇的轨迹！

这个开创性的发明像一抹耀眼的光束，瞬间洒满整个屋顶，照亮昏暗房间。它随即射入人类思想的夜空，改变了数学发展的路径。笛卡尔坐标系让几何和代数这两条平行线交织在一起，再也没有分开。

几何形体就像是暗夜中大海上游弋的航船。坐标系就是灯塔，就是指引方位的北斗。代数式每个符号原本瘦骨嶙峋、死气沉沉。坐标系让它们血肉丰满、生龙活虎。

毫不夸张地说，没有笛卡尔坐标系，就不会有函数，更不会有微积分。

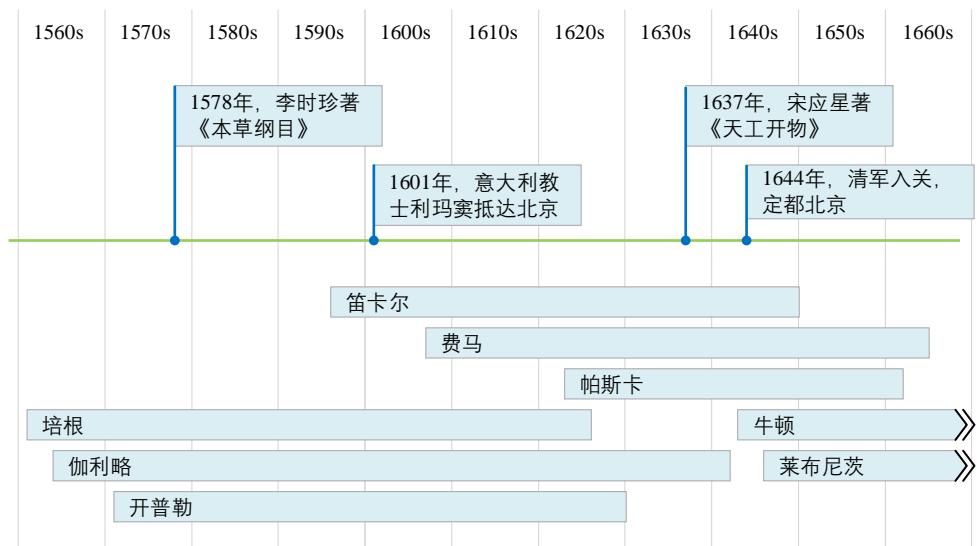


图 2. 笛卡尔时代时间轴

5.2 坐标系：代数可视化，几何参数化

平面直角坐标系

在平面上，**笛卡尔坐标系** (Cartesian coordinate system) 也叫平面直角坐标系。平面直角坐标系是两个相交于**原点** (origin) 相互垂直的实数轴。数学中，平面直角坐标系常记做 \mathbb{R}^2 。

如图 3 所示，平面直角坐标系是“横平竖直”的方格。**横轴** (horizontal number line) 常被称作 x 轴 (x -axis)，**纵轴** (vertical number line) 常被称作 y 轴 (y -axis)。

⚠ 注意，本书也常用 x_1 表示横轴，用 x_2 表示纵轴。

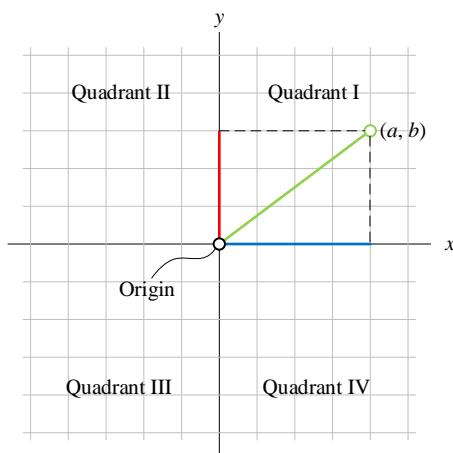


图 3. 笛卡尔坐标系

如图 3 所示，横纵轴将 xy 平面 (xy -plane) 分成四个象限 (quadrants)。象限通常以罗马数字 (Roman numeral) 逆时针方向 (counter-clockwise) 编号。

⚠ 注意，象限不包括坐标轴。

平面上的每个点都可以表示为坐标 (a, b) 。 a 和 b 两个值分别为横坐标 (x -coordinate) 和纵坐标 (y -coordinate)。图 4 所示为平面直角坐标系中 6 个点对应的坐标，请大家自己标出每个点所在象限或横纵轴。

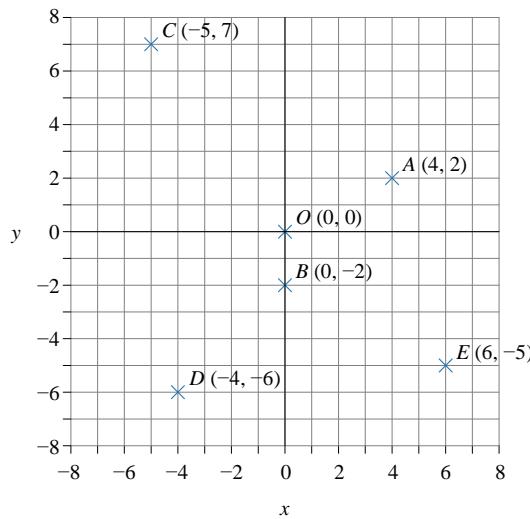


图 4. 平面直角坐标系中 6 个点的位置



代码文件 `Bk3_Ch5_01.py` 绘制图 4 所示平面直角坐标系网格和其中 6 个点，并打印坐标值。

欧几里得的五个公理

有了直角坐标系，欧几里得提出的五个公理就可以很容易被量化，如图 5 和图 6 所示。下面，我们展开讲解。

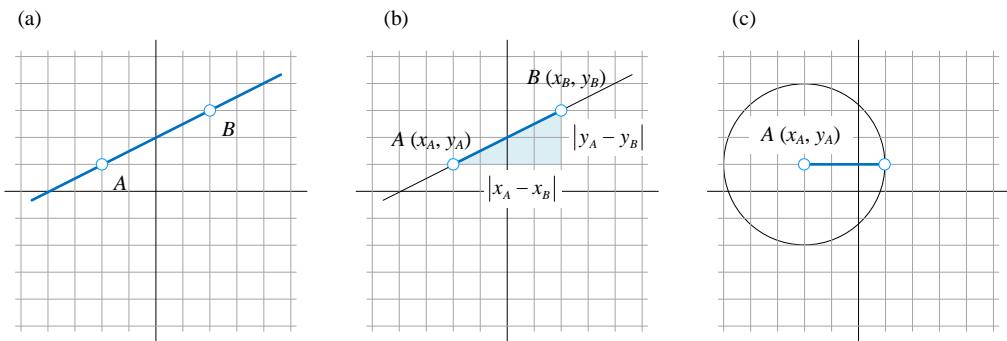


图 5. 在平面直角坐标系中展示直线、线段长度和圆

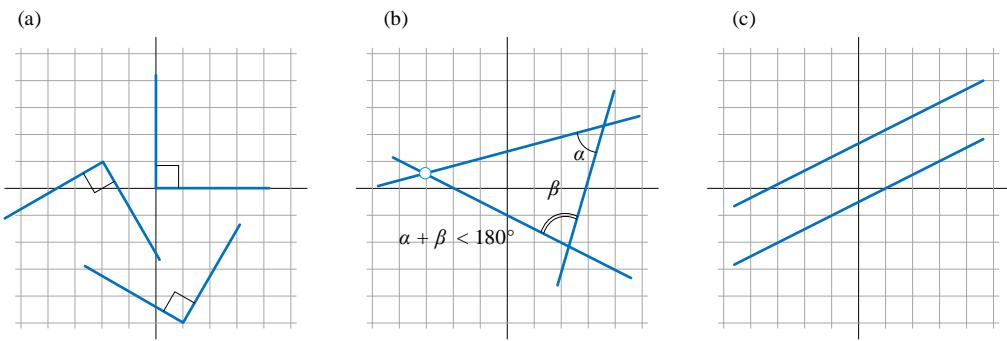


图 6. 在平面直角坐标系中展示直角、相交和平行

直线

如图 5 所示，平面直角坐标系中，任意两点可以画一条直线，这条直线一般对应代数中的二元一次方程：

$$ax + by + c = 0 \quad (1)$$

使用矩阵乘法，(1) 可以写成：

$$\begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + c = 0 \quad (2)$$

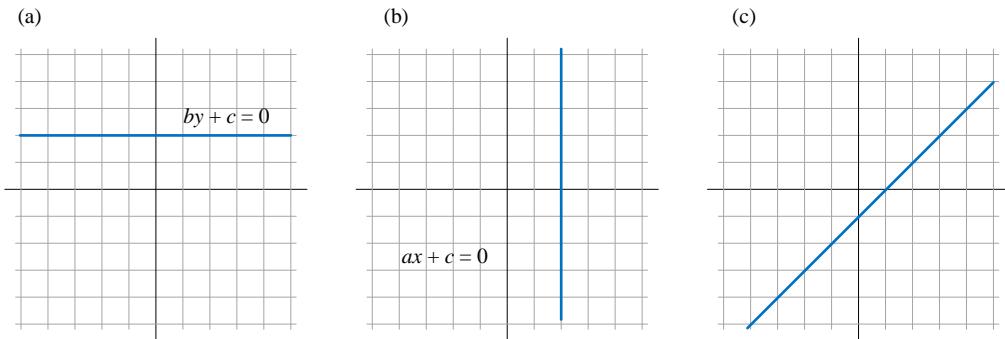


图 7. 平面直角坐标系中三类直线

如图 7 (a) 所示，特别地，当 $a = 0$ 时，直线平行于横轴：

$$by + c = 0 \quad (3)$$

如图 7 (b) 所示，当 $b = 0$ 时，直线平行于纵轴：

$$ax + c = 0 \quad (4)$$

如图 7 (c) 所示，如果 a 、 b 均不为 0，(1) 可以写成：

$$y = -\frac{a}{b}x - \frac{c}{b} \quad (5)$$

当 x 为自变量、 y 为因变量时，(5) 实际上就变成了一元一次函数。其中， $-a/b$ 为直线斜率 (slope)， $-c/b$ 为纵轴截距 (y-intercept)。



本书第 11 章将专门介绍一元一次函数图像。

两点距离

如图 5 (b) 所示， $A(x_A, y_A)$ 和 $B(x_B, y_B)$ 两点之间直线的距离可以用勾股定理获得：

$$AB = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \quad (6)$$

正圆

如图 5 (c) 所示，以 $A(x_A, y_A)$ 点为圆心， r 为半径画一个圆。圆上任意一点 (x, y) 到 $A(x_A, y_A)$ 点的距离为 r ，据此可以构造等式：

$$\sqrt{(x - x_A)^2 + (y - y_A)^2} = r \quad (7)$$

(7) 两边平方得到图 5 (c) 所示圆的解析式：

$$(x - x_A)^2 + (y - y_A)^2 = r^2 \quad (8)$$

使用矩阵乘法，(8) 可以写成：

$$\begin{bmatrix} x - x_A & y - y_A \end{bmatrix} \begin{bmatrix} x - x_A \\ y - y_A \end{bmatrix} - r^2 = 0 \quad (9)$$

特别地，当圆心为原点 $(0, 0)$ ，半径 $r = 1$ 时，圆为 **单位圆** (unit circle)，对应的解析式为：

$$x^2 + y^2 = 1 \quad (10)$$

使用矩阵乘法，(10) 可以写成：

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - 1 = 0 \quad (11)$$

有了平面直角坐标系，单位圆和各种三角函数之间联系就很容易可视化，具体如图 8 所示。

请大家特别注意 θ 为 $\pi/2$ (90°) 的倍数时，即 $\theta = \pi k/2$ (k 为整数)，有些三角函数值为无穷，即没有定义。比如 $\theta = 0$ (0°) 时，点 A 在横轴正半轴上，图 8 中 $\csc(\theta)$ 和 $\cot(\theta)$ 均为无穷。又如 $\theta = \pi/2$ (90°) 时，点 A 在纵轴正半轴上，图 8 中 $\sec(\theta)$ 和 $\tan(\theta)$ 均为无穷。图 9 所示为平面直角坐标系中，角度、弧度和常用三角函数的正负关系。

 当 θ 连续变化时，几个三角函数值也会跟着连续变化，在平面直角坐标系中，我们可以画出三角函数图像。本书第 11 章将介绍常见三角函数的图像。

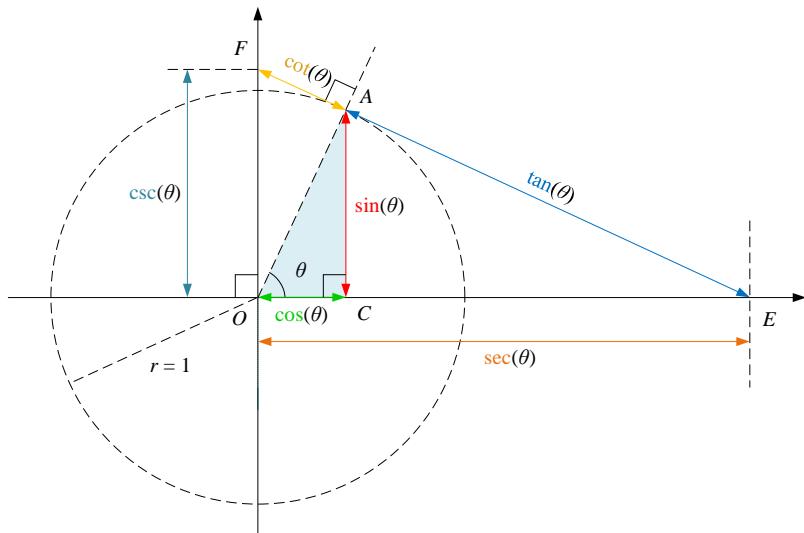


图 8. 三角函数和单位圆的关系

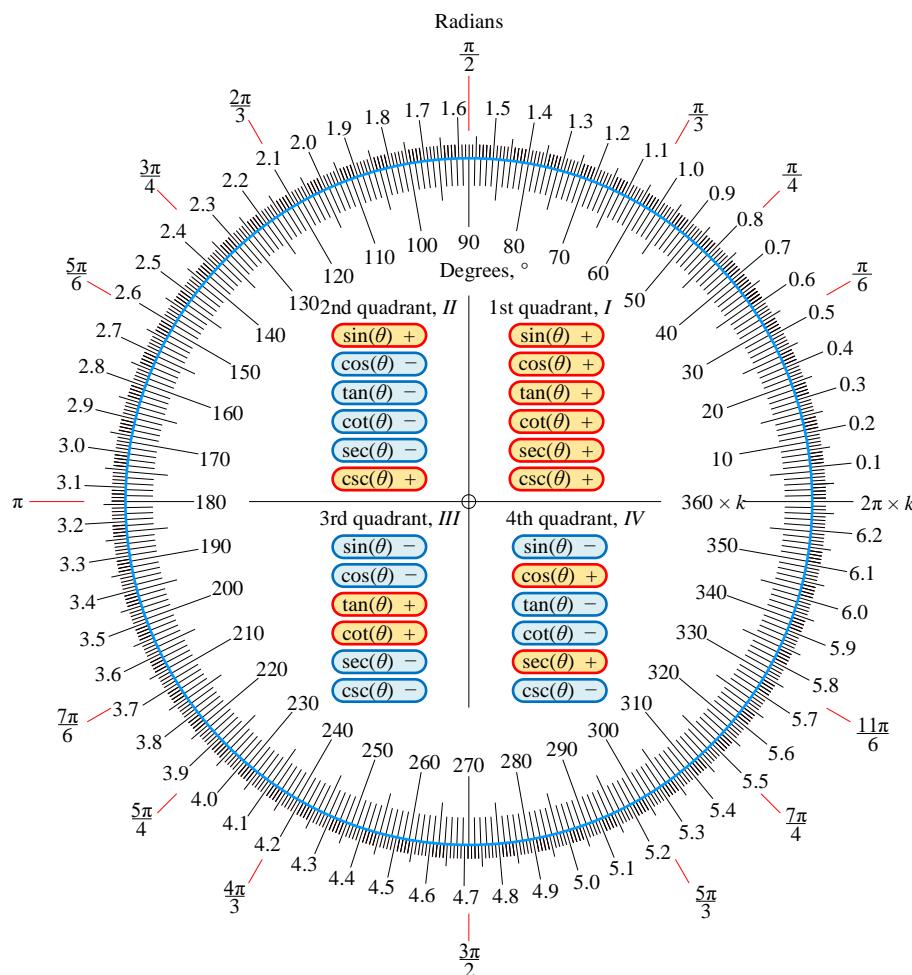


图 9. 平面直角坐标系中，角度、弧度和常用三角函数的正负关系

垂直

平面直角坐标系中，判断垂直变得更加简单。

给定 $ax + by + c = 0$ 和 $\alpha x + \beta y + \gamma = 0$ 两条直线，两者垂直时满足如下条件：

$$a\alpha + b\beta = 0 \quad (12)$$

如果系数 a 、 b 、 α 、 β 均不为 0 时，两条直线若垂直，则两条直线斜率相乘为 -1 ，即，

$$\frac{a}{b} \frac{\alpha}{\beta} = -1 \quad (13)$$

图 10 (a) 所示为两条垂直线，它们分别代表 $y = 0.5x + 2$ 和 $y = -2x - 1$ 这两个一次函数。显然两个一次函数斜率相乘为 $-1 = 0.5 \times (-2)$ 。

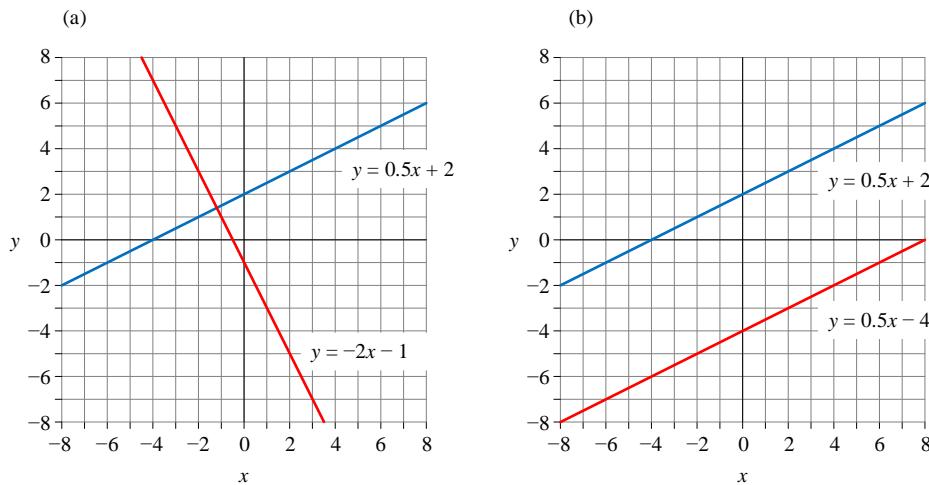


图 10. 两条垂直直线和两条平行线

平行

类似地，如果 $ax + by + c = 0$ 和 $\alpha x + \beta y + \gamma = 0$ 两条直线平行，系数满足：

$$a\beta - b\alpha = 0 \quad (14)$$

如果系数 a 、 b 、 α 、 β 均不为 0 时，两条直线若平行或重合，则两个斜率相同，即，

$$\frac{a}{b} = \frac{\alpha}{\beta} \quad (15)$$

图 10 (b) 所示为两条平行线。图 11 分别展示的是两条水平线和两条竖直线。两条水平线可以视作常数函数，而两条竖直线则不是函数。

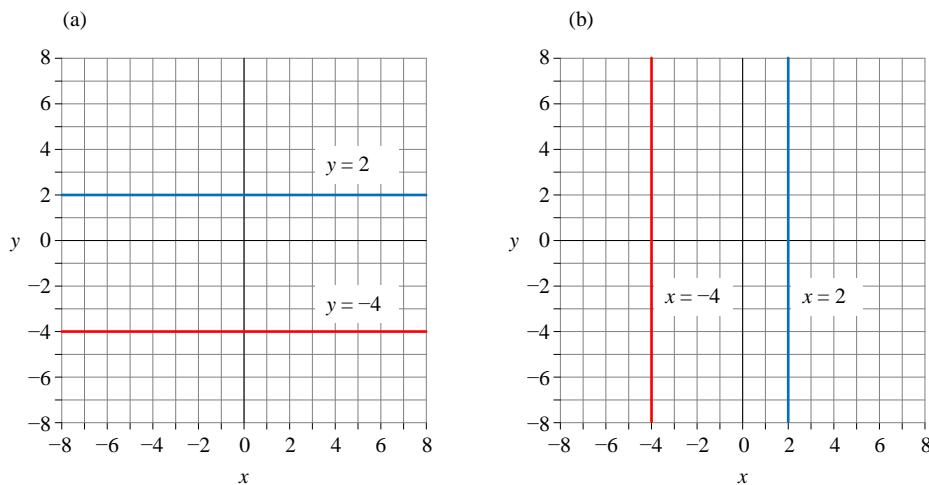


图 11. 两条水平线和两条竖直线

表 1 总结有关坐标系的常用英文表达。

表 1. 有关坐标系的常用英文表达

数学或中文表达	英文表达
(a,b)	The point a, b
$P(a,b)$	The point capital P with coordinates a and b
$P(4,3)$	The x -coordinate of point P is 4; and the y -coordinate of point P is 3. The coordinates of point P are $(4, 3)$. 4 is the x -coordinate and 3 is the y -coordinate P is 4 units to the right of and 3 units above the origin.
第一象限	First quadrant
y 轴正方向	Positive direction of the y -axis
y 轴负方向	Negative direction of the y -axis
x 轴正方向	Positive direction of the x -axis
x 轴负方向	Negative direction of the x -axis
关于 x 轴对称	To be symmetric about the x -axis
关于 y 轴对称	To be symmetric about the y -axis
关于原点对称	To be symmetric about the origin



代码文件 `Bk3_Ch5_02.py` 来绘制图 10 和图 11。

5.3 图解“鸡兔同笼”问题

图解法

有了平面直角坐标系，我们就可以图解本书第 4 章提到的鸡兔同笼问题。

首先构造二元一次方程组，这次用 x_1 代表鸡， x_2 代表兔。

鸡、兔共有 35 个头，对应如下等式：

$$x_1 + x_2 = 35 \quad (16)$$

有 94 只足，对应等式：

$$2x_1 + 4x_2 = 94 \quad (17)$$

联立两个等式，得到方程组：

$$\begin{cases} x_1 + x_2 = 35 \\ 2x_1 + 4x_2 = 94 \end{cases} \quad (18)$$

用图解法，(16) 和 (17) 分别代表平面直角坐标系的两条直线，如图 12。两条直线的交点就是解 $(23, 12)$ 。也就是，笼子里有 23 只鸡，12 只兔。

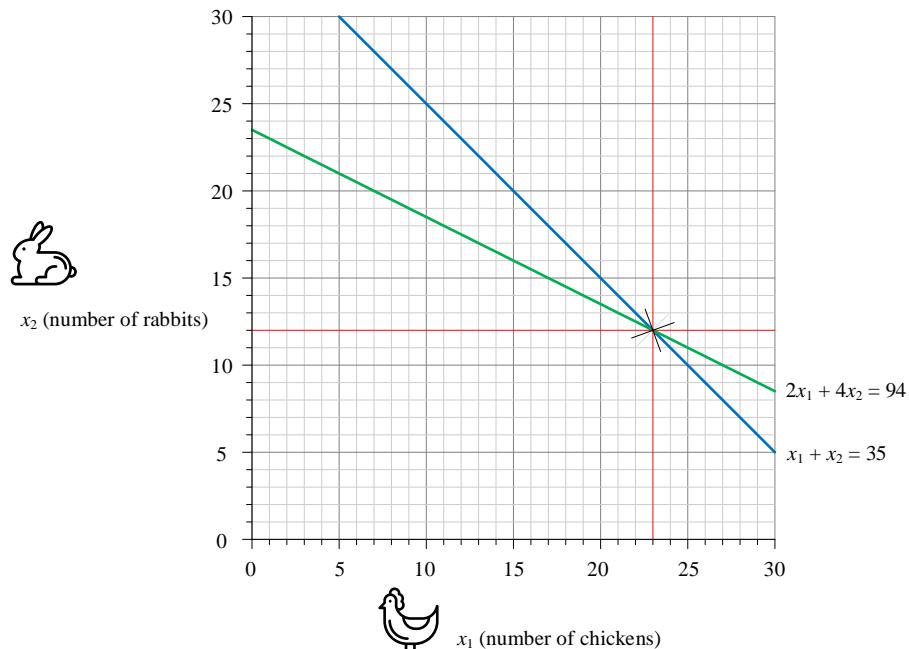


图 12. 鸡兔同笼问题方程组对应的图像

限制条件

实际上，图 12 两条直线并不能准确表达鸡兔同笼问题的全部条件。

鸡兔同笼问题还隐含着限制条件—— x_1 和 x_2 均为非负整数。也就是说，鸡、兔的个数必须是 0 或正整数，不能是小数，更不能是负数。

有了这个条件作为限制，我们便可以获得如图 13 这幅图像。可以看到，方程对应的图像不再是连续的直线，而是一个个点。图 13 的网格交点对应整数坐标点，可以看到所有的 \times 点都在网格交点处。

图 13 中所有的点被限制在第一象限（包含坐标轴），这个区域对应不等式组：

$$\begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \end{cases} \quad (19)$$

不等式区域是下一章要探讨的话题。

从另外一个角度来看，图 13 中 $\textcolor{red}{\times}$ 和 $\textcolor{blue}{\times}$ 两组点对应的横、纵轴坐标值分别构成**等差数列** (arithmetic progression)。

等差数列是指从第二项起，每一项与它的前一项的差等于同一个常数的一种数列。

⚠ 注意，数列也可以看做是定义域不连续的特殊函数。



本书第 14 章将讲解数列相关内容。

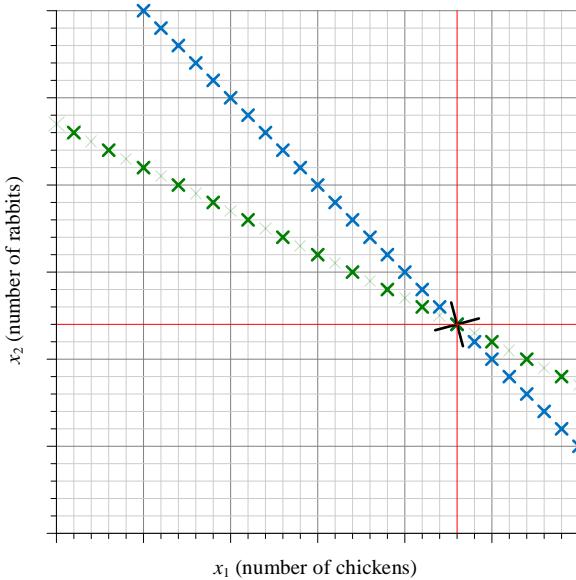


图 13. 鸡兔同笼问题方程组对应的非负整数图像

二元一次方程组解的个数

两个二元一次方程构成的方程组可以有一个解、无数解或者没有解。

有了图像，这一点就很好理解了。图 14 (a) 给出的两条直线相交于一点，也就是二元一次方程组有一个解。

图 14 (b) 给出的两条直线相重合，也就是二元一次方程组无数解。

图 14 (c) 给出的两条直线平行，也就是二元一次方程组没有解。

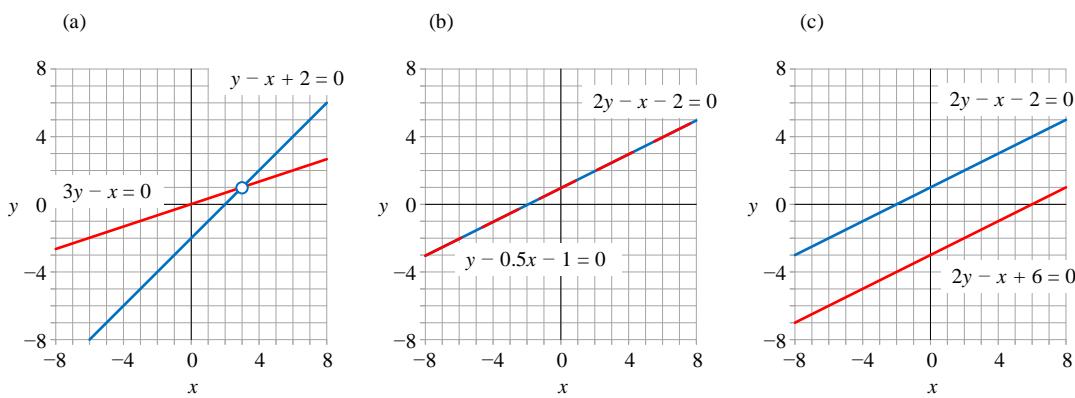


图 14. 两个二元一次方程组有一个解、无数解、没有解



代码文件 Bk3_Ch5_03.py 绘制图 12。代码并没有直接计算出方程组的解，这个任务交给本线性代数相关内容来解决。



我们在 Bk3_Ch5_03.py 基础上，用 Streamlit 制作了绘制平面直线的 App，通过调整参数，请大家观察直线位置变化。请参考代码文件 Streamlit_Bk3_Ch5_03.py。

5.4 极坐标：距离和夹角

极坐标系 (polar coordinate system) 也是常用坐标系。如图 15 左图所示，平面直角坐标系中，位置由横轴、纵轴坐标值确定。而极坐标中，位置由一段距离 r 和一个夹角 θ 来确定。

如图 15 右图所示， O 是极坐标的**极点** (pole)，从 O 向右引一条射线作为**极轴** (polar axis)，规定逆时针角度为正。这样，平面上任意一点 P 的位置可以由线段 OP 的长度 r 和极轴到 OP 的角度 θ 来确定。 (r, θ) 就是 P 点的极坐标。

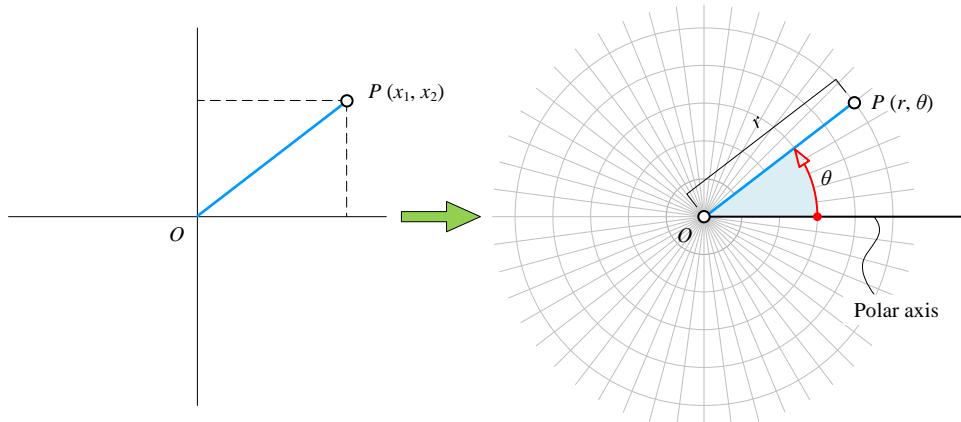


图 15. 从平面直角坐标系到极坐标系

一般， r 称为**极径** (radial coordinate 或 radial distance)， θ 称为**极角** (angular coordinate 或 polar angle 或 azimuth)。

平面上，极坐标 (r, θ) 可以转化为直角坐标系坐标 (x_1, x_2) :

$$\begin{cases} x_1 = r \cdot \cos \theta \\ x_2 = r \cdot \sin \theta \end{cases} \quad (20)$$

平面极坐标让一些曲线可视化变得非常容易。图 16 (a) 所示为极坐标中绘制的正圆，图 16 (b) 所示为阿基米德螺旋线 (Archimedean spiral)，图 16 (c) 为玫瑰线。

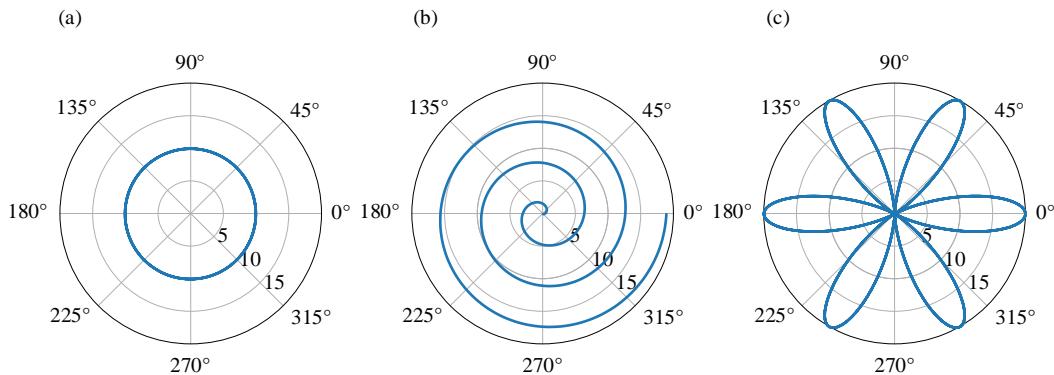
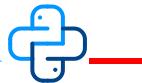


图 16. 平面极坐标中可视化三个曲线



代码文件 Bk3_Ch5_04.py 绘制图 16 三幅图像。

5.5 参数方程：引入一个参数

在平面直角坐标系中，如果曲线上任意一点坐标 x 、 y 都是某个参数（比如 t ）的函数。对于参数任何取值，方程组确定的点 (x_1, x_2) 都在这条曲线上，那么这个方程就叫做曲线的**参数方程** (parametric equation)，比如下例：

$$\begin{cases} x_1 = f(t) \\ x_2 = g(t) \end{cases} \quad (21)$$

图 17 所示为用参数方程法绘制的单位圆，对应的参数方程为：

$$\begin{cases} x_1 = \cos(t) \\ x_2 = \sin(t) \end{cases} \quad (22)$$

其中， t 为参数，取值范围为 $[0, 2\pi]$ 。

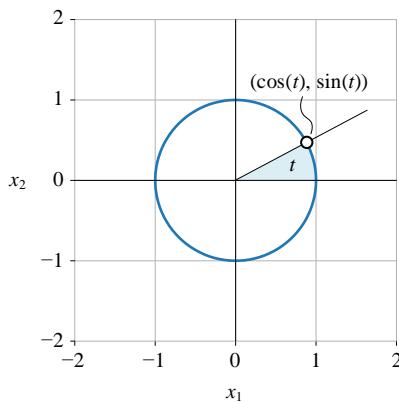


图 17. 参数方程绘制正圆



代码文件 `Bk3_Ch5_05.py` 可以绘制图 17。

我们也可以采用 `sympy` 工具包中的 `plot_parametric()` 函数绘制二维参数方程，代码文件 `Bk3_Ch5_06.py` 便是通过 `t = symbols('t')` 先定义符号变量 `t`。然后，利用 `plot_parametric()` 函数绘制单位圆。

5.6 坐标系必须是“横平竖直的方格”？

本章最后聊一下“坐标系”的内涵。

广义来说，坐标系就是一个定位系统。比如，地球表面可以用经纬度来唯一确定一点，显然经纬度网格不是横平竖直，它更像本章讲到的极坐标。

具体到某一个建筑内的位置时，我们在经纬度基础上加入楼层数这个定位参数。而航空、航天器定位时，会考虑海拔。

现在人类还是生存在地球“表面”。假想在不远的未来，人类可以大规模地在地下、海洋下方，甚至天空中生活，这时人们可能要自然而然地在经纬度基础上再加一个定位值，比如距离地心距离或海拔。三座城市很可能经纬度几乎一致，却分别位于地表、地下和半空中。

坐标系的定义满足实际需求，根据约定俗成怎么方便怎么来。

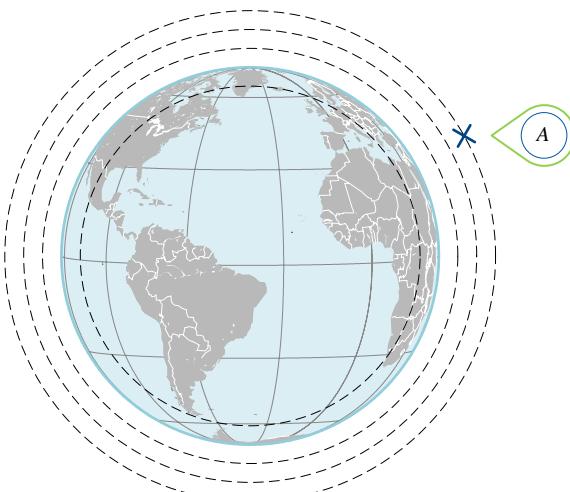


图 18. 经纬度加海拔定位

笛卡尔坐标系是数学中定位平面一点最常用的坐标系。本章给出的直角坐标系都是横平竖直的“方格”，这是因为它们的横纵坐标轴垂直，且尺度完全一致。很多情况，直角坐标系的横纵坐标轴的数值尺度不同，这样我们便获得“长方格”的直角坐标系。

如图 19 所示，横平竖直的方格，经过竖直或水平方向拉伸，得到两个不同长方格。大家会发现，当图像较复杂时，为了突出其细节，本书中很多图像并不绘制网格，而只提供坐标轴上的刻度线和对应刻度值。必要时，在竖直或水平轴具体位置加参考线 (reference line)。

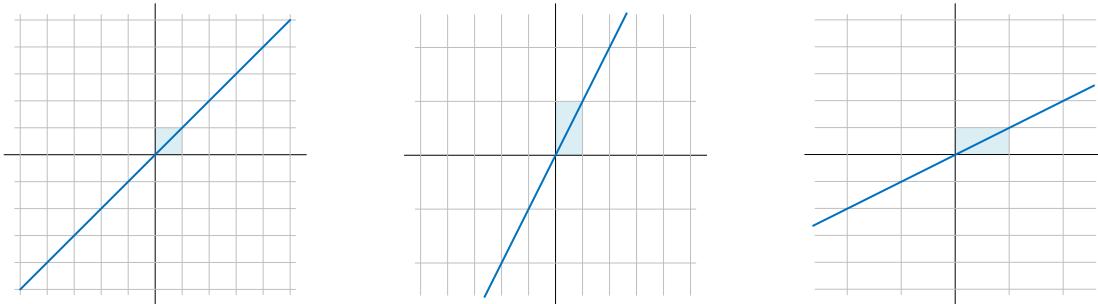


图 19. 直角坐标系，方格到长方

图 19 中三个直角坐标系中方格的大小还是分别保持一致。有些应用场合，一幅图像中方格形状还可能不一致。如图 20 右图所示，图像的纵轴为对数坐标刻度 (logarithmic scale)，这时坐标系方格大小就不再一致。

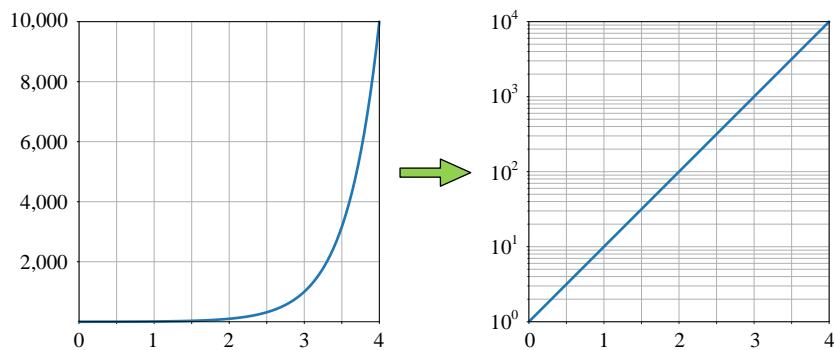
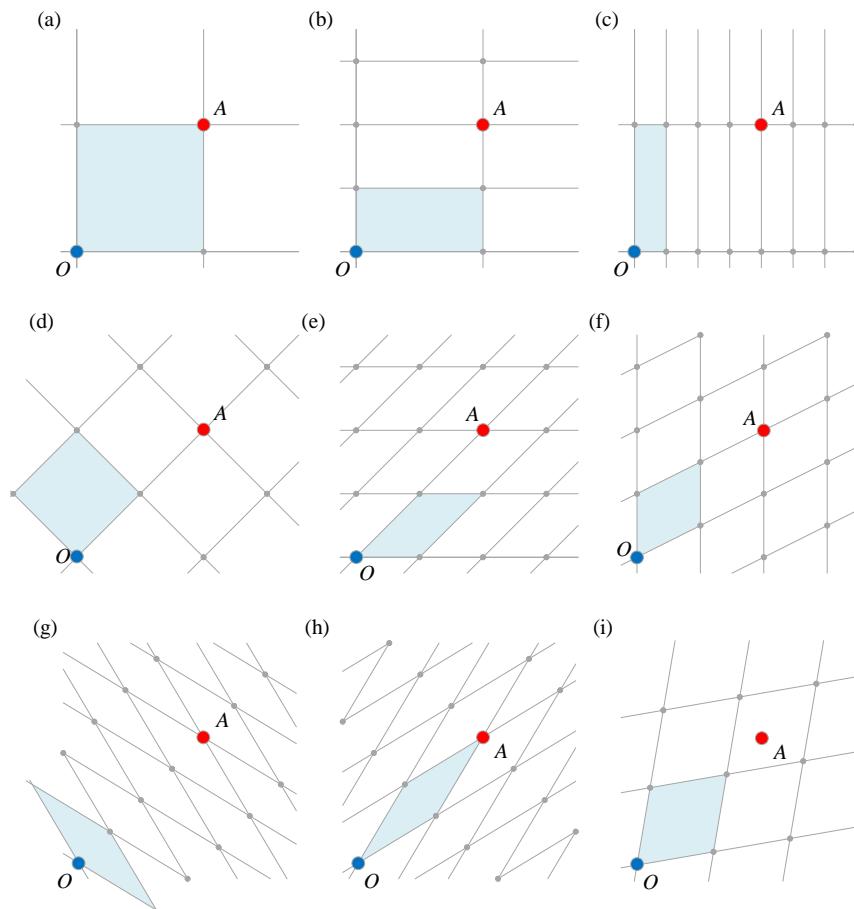


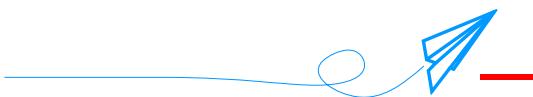
图 20. 直角坐标系到纵轴为对数刻度

再退一步，不管怎么说，图 20 的刻度线还是“横平竖直”。有些时候，“横平竖直”这个限制也可以被打破。图 21 中 (a)、(b) 和 (c) 三幅图坐标网格还是横平竖直。而剩下 6 幅图网格则千奇百怪，对应独特的旋转、伸缩等等几何操作。

即便如此，图 21 中 9 幅图都可以准确定位点 A 和点 O 的位置关系。大家可能已经发现，概括来说，图 21 中每幅图各自的网格都是全等的平行四边形。

图 21. 不同形状平行四边形网格表达点 A 和点 O 关系

本节展示的各种坐标系都束缚在同一个平面内。这个平面最根本的坐标系就是笛卡尔直角坐标系 \mathbb{R}^2 ，而各种坐标系似乎都和笛卡尔坐标系 \mathbb{R}^2 存在某种量化联系。目前我们介绍的数学工具还不足够解析这些量化联系，本系列丛书会讲解更多数学工具，慢慢给大家揭开不同坐标系和笛卡尔直角坐标系的关系。



笛卡尔的坐标系像极了太极八卦。

太极生两仪，两仪生四象，四象生八卦。坐标系的原点就是太极的极，两极阴阳为数轴负和正。横轴 x 和纵轴 y 张成平面 \mathbb{R}^2 ，并将其分成为四个象限。

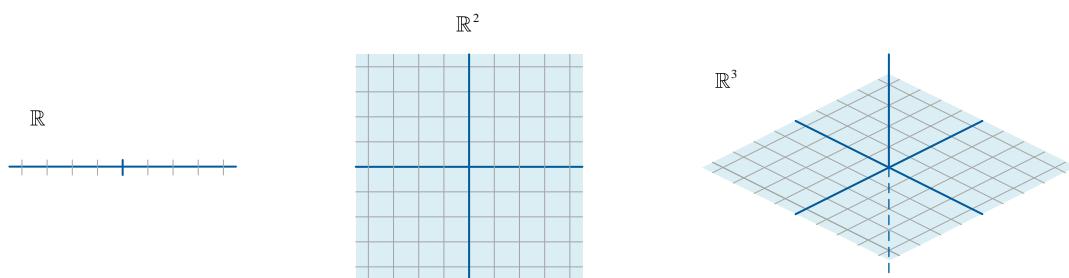


图 22. 数轴、平面直角坐标系、三维直角坐标系

垂直于 \mathbb{R}^2 平面再升起一个 z 轴，便生成一个三维空间 \mathbb{R}^3 。 x 、 y 和 z 轴将三维空间割裂成八个区块。这是下一章要介绍的内容。

坐标系看似有界，但又无界。正所谓大方无隅，大器免成，大音希声，大象无形。

笛卡尔坐标系包罗万象，本章之后的所有数学知识和工具都包含在笛卡尔坐标系这个“大象”之中。

6

Three-Dimensional Coordinate System

三维坐标系

平面直角坐标系上升起一根竖轴



虚空无尽的蔚蓝，神秘深邃的苍穹，漫天飘舞的虫鸟 ...

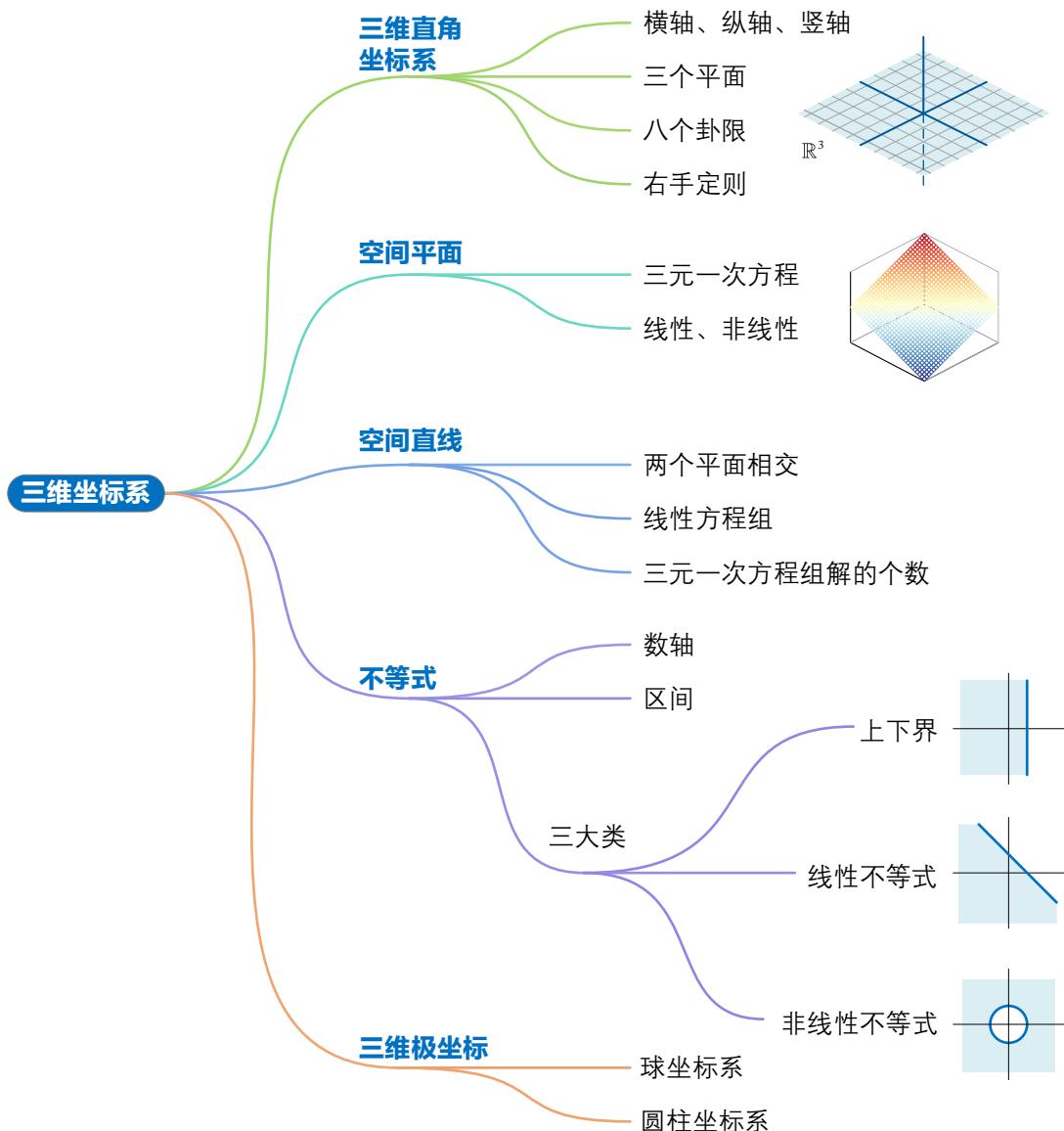
时时刻刻在召唤，“腾空而起吧，人类！”

*The blue distance, the mysterious Heavens, the example of birds and insects flying everywhere —
are always beckoning Humanity to rise into the air.*

—— 康斯坦丁·齐奥尔科夫斯基 (Konstantin Tsiolkovsky) | 俄罗斯火箭专家 | 1857 ~ 1935



- ◀ ax.plot_wireframe() 绘制线框图
- ◀ matplotlib.pyplot.contour() 绘制平面等高线
- ◀ matplotlib.pyplot.contourf () 绘制平面填充等高线
- ◀ numpy.meshgrid() 产生网格化数据
- ◀ numpy.outer() 计算外积
- ◀ plot_parametric() 绘制二维参数方程
- ◀ plot3d_parametric_line() 绘制三维参数方程



6.1 三维直角坐标系

费马 (Pierre de Fermat) 不但独立发明平面直角坐标系，他还在 xy 平面坐标系上插上 z 轴，创造了三维直角坐标系。

三维直角坐标系有三个坐标轴—— x 轴或**横轴** (x -axis), y 轴或**纵轴** (y -axis) 和 z 轴或**竖轴** (z -axis)。本系列丛书也经常使用 x_1 、 x_2 、 x_3 来代表横轴、纵轴和竖轴。

图 1 所示三维直角坐标系有三个平面： xy 平面、 yz 平面、 xz 平面。 x 轴和 y 轴构成 xy 平面， z 轴垂直于 xy 平面； y 轴和 z 轴构成 yz 平面， x 轴垂直于 yz 平面； x 轴和 z 轴构成 xz 平面， y 轴垂直于 xz 平面。这三个平面将三维空间分成了八个部分，称为**卦限** (octant)。

三维直角坐标系内坐标点可以写成 (a, b, c) 。

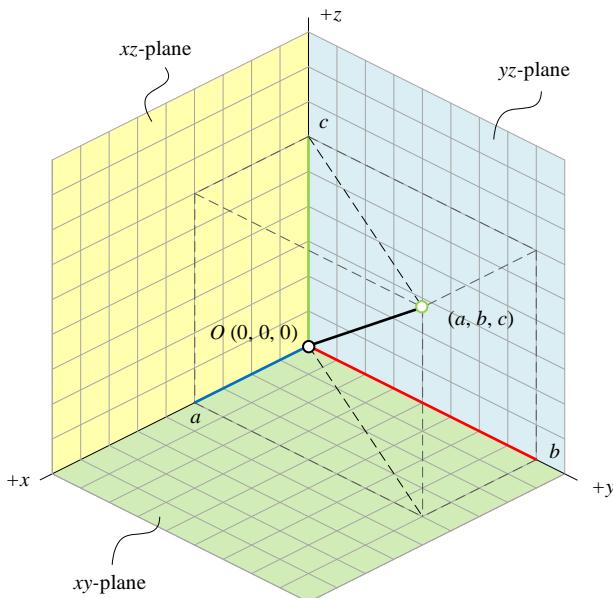


图 1. 三维直角坐标系和三个平面

图 2 所给出三种右手定则，用来确定三维直角坐标系 x 、 y 和 z 轴正方向。比较常用的是图 2 中间这幅图所示定则。

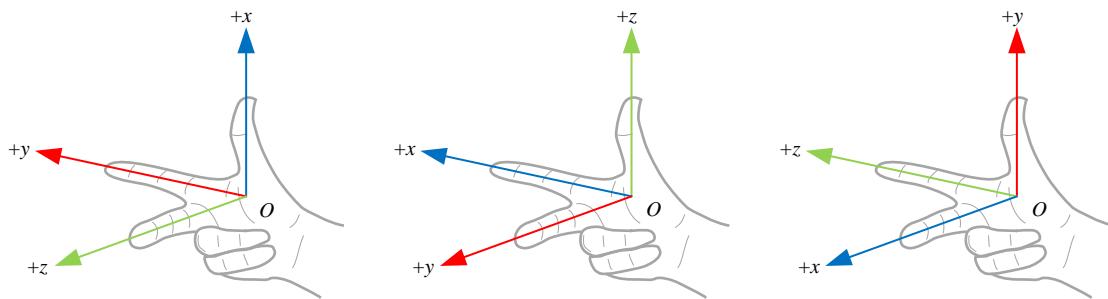


图 2. 右手定则确定三维直角坐标系 x 、 y 和 z 轴正方向

6.2 空间平面：三元一次方程

三维直角坐标系中，平面可以写成如下等式：

$$ax + by + cz + d = 0 \quad (1)$$

其中， x 、 y 、 z 为变量， a 、 b 、 c 、 d 为参数。实际上，这个等式就是代数中的三元一次方程。

利用矩阵乘法，(1) 可以写成：

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + d = 0 \quad (2)$$

第一个平面

举个例子，图 3 所示的平面对应的解析式为：

$$x + y - z = 0 \quad (3)$$

图 3 中网格面的颜色对应 z 的数值。 z 越大，越靠近暖色系； z 越小，越靠近冷色系。

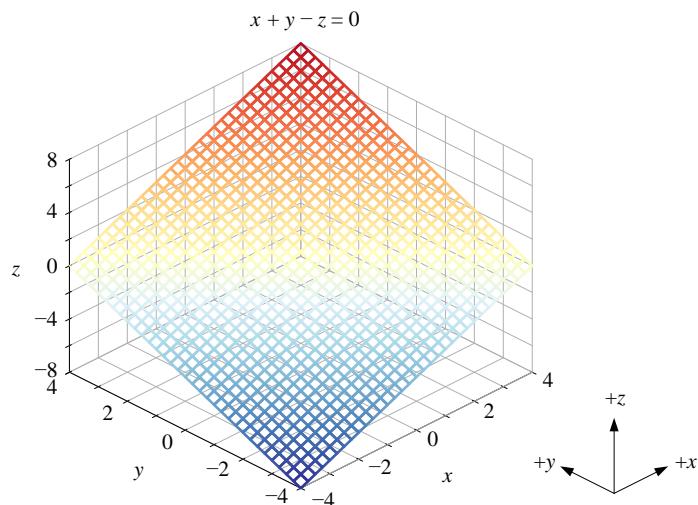


图 3. 等式 $x + y - z = 0$ 对应的平面

以 z 作为因变量、 x 和 y 作为自变量的话，(3) 等价于如下二元函数：

$$z = f(x, y) = x + y \quad (4)$$

第二个平面

图 4 所示平面对应的解析式为：

$$y - z = 0 \quad (5)$$

图 4 中网格面平行于 x 轴，垂直于 yz 平面。从等式上来看，不管 x 取任何值，图 4 平面上的点对应的 y 和 z 都满足 $y - z = 0$ 。

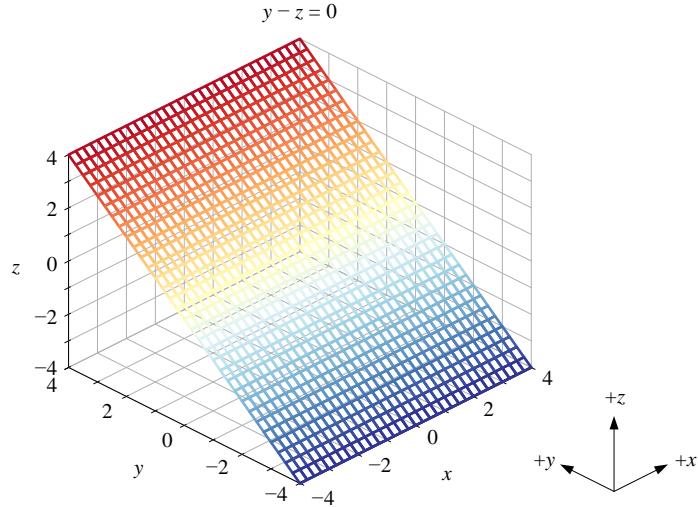


图 4. 等式 $y - z = 0$ 对应的平面

第三个平面

图 5 所示的平面对应的解析式为：

$$x - z = 0 \quad (6)$$

图 5 中网格面平行于 y 轴，垂直于 xz 轴。不管 y 取任何值，图 5 平面上的点 x 和 z 的关系都满足 $x - z = 0$ 。

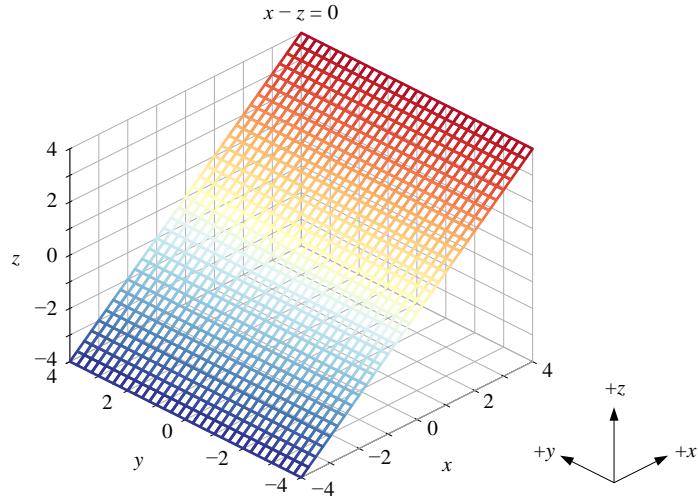
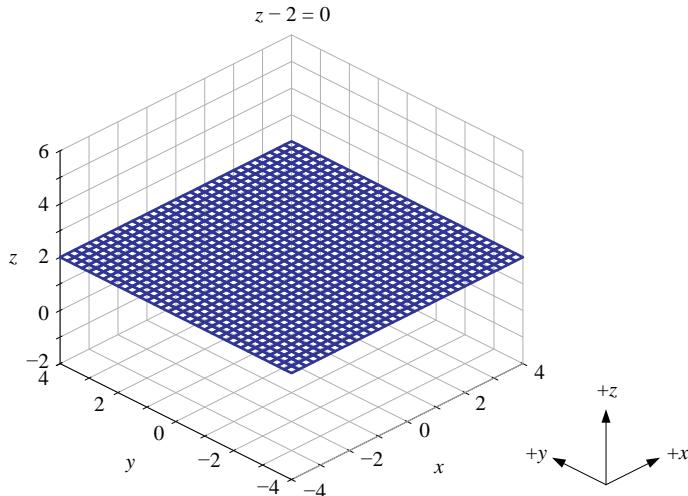


图 5. 等式 $x - z = 0$ 对应的平面

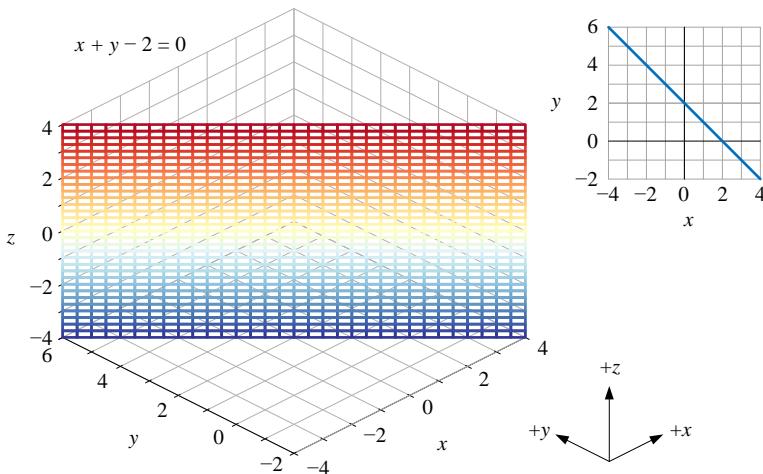
第四个平面

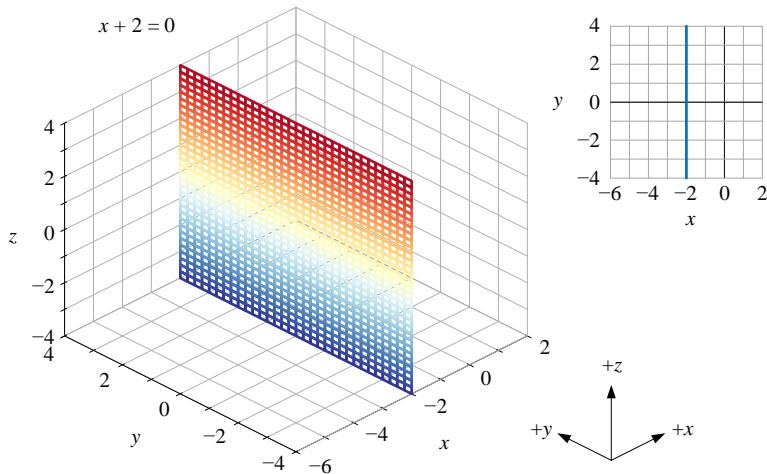
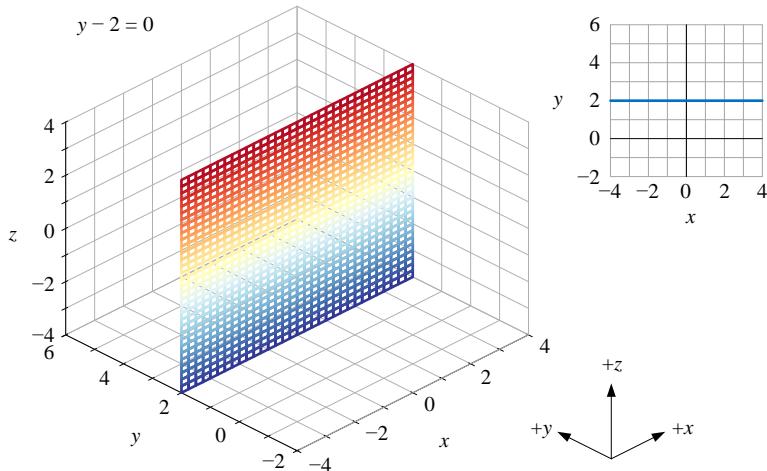
图 6 所示平面对应的等式为 $z - 2 = 0$ ，这个平面显然平行于 xy 平面，垂直 z 轴。从函数角度，这个平面可以看做是二元常数函数，写成 $f(x, y) = c$ 。

图 6. 等式 $z - 2 = 0$ 对应的平面

最后三个例子

图 7 ~ 图 9 三幅图中平面有一个共同特点，它们都垂直于 xy 平面。这三个平面， z 的取值都不影响平面和 xy 平面的相对位置。三个平面都相当于， xy 平面上一条直线沿 z 方向展开。反过来，图 7 ~ 图 9 三幅图中平面在 xy 平面上的投影为一条直线。

图 7. 等式 $x + y - 2 = 0$ 对应的平面

图 8. 等式 $x + 2 = 0$ 对应的平面图 9. 等式 $y - 2 = 0$ 对应的平面

Bk3_Ch6_01.py 绘制本节几幅三维空间平面。



我们在 Bk3_Ch6_01.py 基础上，用 Streamlit 制作了绘制绘制三维空间斜面的 App，通过调整参数，请大家观察斜面位置变化。请参考代码文件 Streamlit_Bk3_Ch6_01.py。



相信大家经常听到“线性”和“非线性”这两个词，下面简单区分两者。

在平面直角坐标系中，**线性** (linearity) 是指量与量之间的关系可以用一条斜线表示，比如 $y = ax + b$ 。平面上，线性函数即一次函数，对应图像为一条斜线。

注意，严格来讲，如果以满足叠加性和齐次性为条件，只有正比例函数是线性函数。

在三维直角坐标系中，“线性”对应几何形式是斜面，也就是二元一次函数，比如 $y = b_1x_1 + b_2x_2 + b_0$ 。

对于多元函数，线性的形式为 $y = b_1x_1 + b_2x_2 + b_3x_3 + \dots + b_nx_n + b_0$ 。在多维空间中，其对应图像是**超平面** (hyperplane)。

图 10 给出线性关系三个例子。

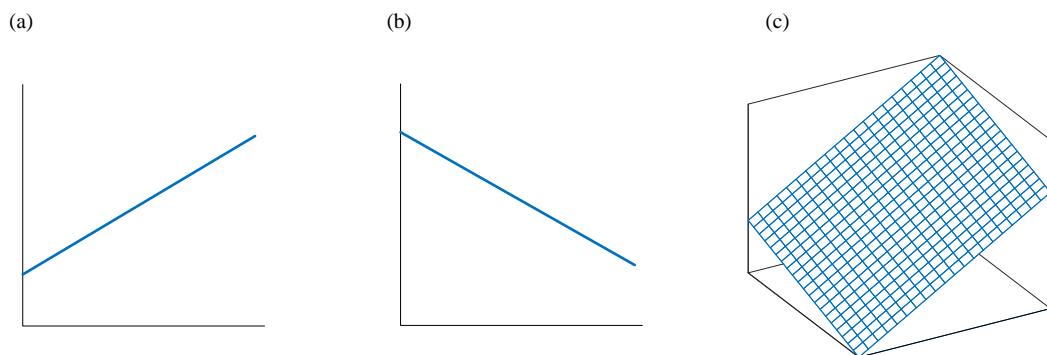


图 10. 线性关系

与线性相对的是**非线性** (nonlinearity)。“非线性”对应的图像不是直线、也不是平面、更不是超平面。平面上，非线性关系可以是曲线、折线，甚至不能用参数来描述。这种不能用参数描述的情况在数学上叫**非参数** (non-parametric)。图 11 给出平面上非线性关系例子。

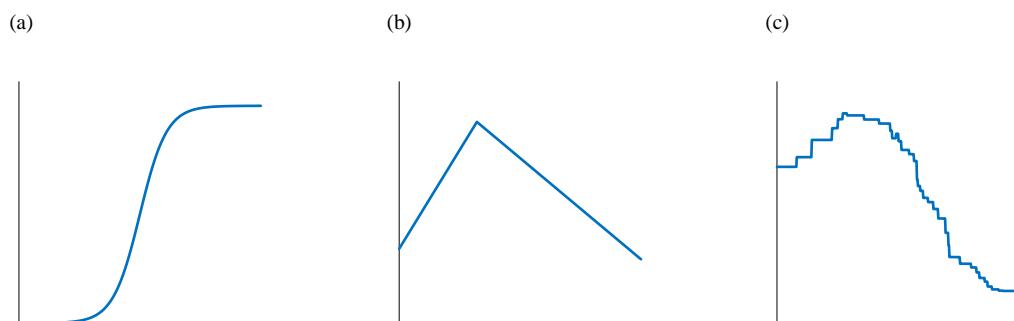


图 11. 非线性关系

机器学习中，回归模型是重要**监督学习** (supervised learning)。回归模型研究变量和自变量之间关系，目的是分析预测。图 12 给出三类回归模型，图 12 (a) 是线性回归模型，图 12 (b)(c) 则是非线性回归模型。

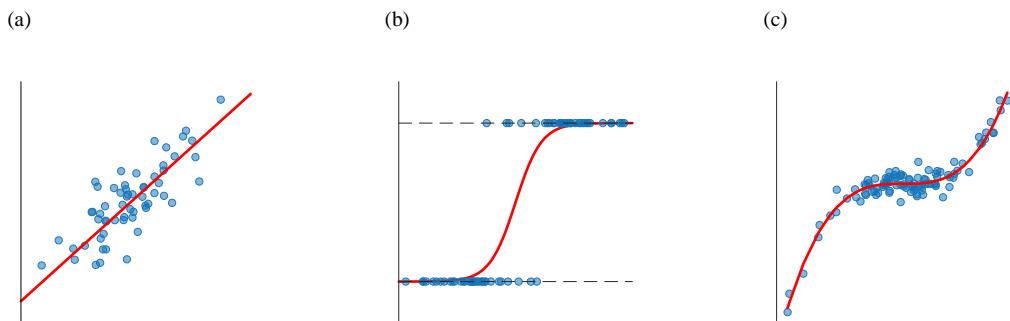


图 12. 机器学习中回归问题

监督学习中，二分类问题很常见，比如将图 13 中蓝色和红色数据点以某种方式分开，分割不同标签数据点的边界线叫**决策边界** (decision boundary)。二分类输出标签一般为 0 (蓝色)、1 (红色)。图 13 (a) 所示为用线性 (一根直线) 决策边界分割蓝色、红色数据点，图 13 (b)(c) 所示为非线性决策边界。

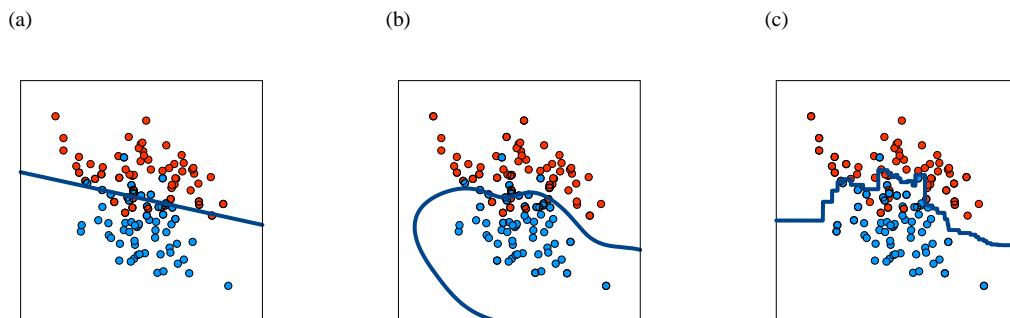


图 13. 机器学习中二分类问题

6.3 空间直线：三元一次方程组

有了三维空间平面，确定一条空间直线则变得很简单——两个平面相交便确定一条空间直线。也就是说，多数情况下，两个三元一次方程确定一条三维空间直线。

举个例子

比如，下例给出两个三元一次方程：

$$\begin{cases} x + y - z = 0 \\ 2x - y - z = 0 \end{cases} \quad (7)$$

上式中，每个方程代表三维空间的一个平面。如图 14 所示，这两个平面相交得到一条直线。

从代数角度，可以这样理解 (7)，这两个三元一次方程构成的方程组有无数组解，这些解都在图 14 所示黑色直线上。

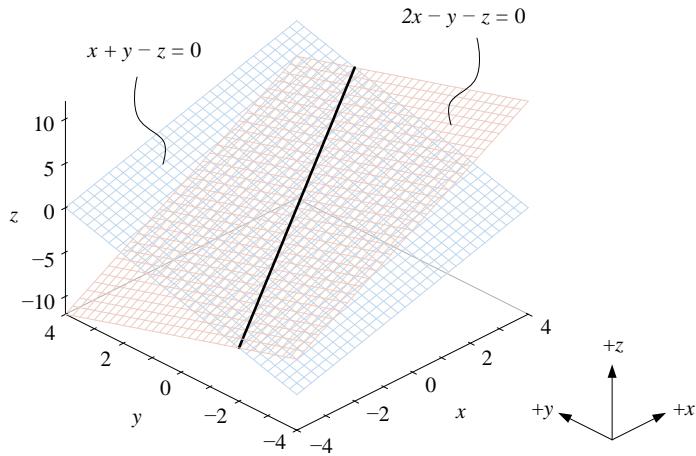


图 14. 两个相交平面确定一条直线

三个平面相交一点

在 (7) 基础上，再加一个三元一次方程，得到如下方程组：

$$\begin{cases} x + y - z = 0 \\ 2x - y - z = 0 \\ -x + 2y - z + 2 = 0 \end{cases} \quad (8)$$

如图 15 所示，这三个平面相交于一点。也就是说，(8) 这个三元一次方程组有唯一解。

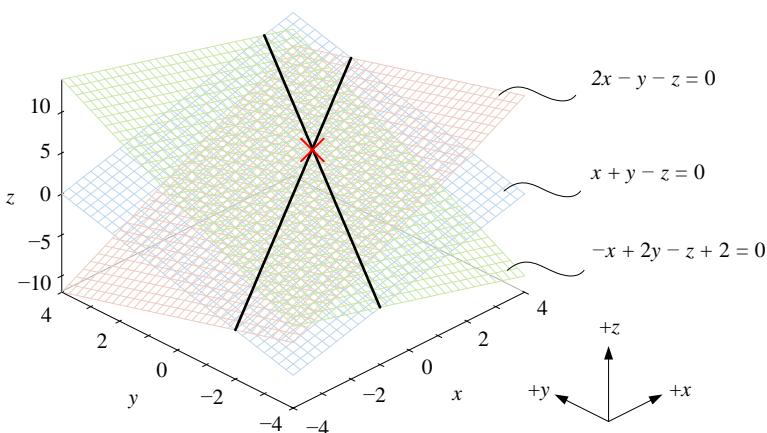


图 15. 三个平面相交于一点

矩阵形式

(8) 一般写成如下矩阵运算形式：

$$\underbrace{\begin{bmatrix} 1 & 1 & -1 \\ 2 & -1 & -1 \\ -1 & 2 & -1 \end{bmatrix}}_A \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix} \quad (9)$$

(9) 这种形式叫做**线性方程组** (system of linear equations)，一般写成 $\mathbf{Ax} = \mathbf{b}$ 。可以想见，当线性方程组的方程数有几百、几千、甚至更多， $\mathbf{Ax} = \mathbf{b}$ 这种形式更规整，更便于计算。而且，对矩阵 A 的各种性质研究，可以判定线性方程组解的特点。



本书最后还会用“鸡兔同笼”问题再次讨论线性方程组。

三元一次方程组解的个数

图 16 所示为三元一次方程组解的个数几种可能性。

如图 16 (a) 所示，当三个平面相交于一点，方程组有且仅有一个解。

如图 16 (b) 所示，当三个平面相交于一条线，方程组有无数组解。无数组解还有其他情况，比如两个平面重合和第三个平面相交，再比如三个平面重合。

图 16 (c)、(d)、(e) 给出的是方程组无解的三种情况。图 16 (c) 中，两个平面平行，分别和第三个平面相交，得到两条交线相互平行。图 16 (d) 中，三个平面平行。图 16 (e) 中，两个平面重合，与第三个平面平行。方程组还有其他无解的情况，比如三个平面两两相交，得到三条交线，而三条交线相互平行。

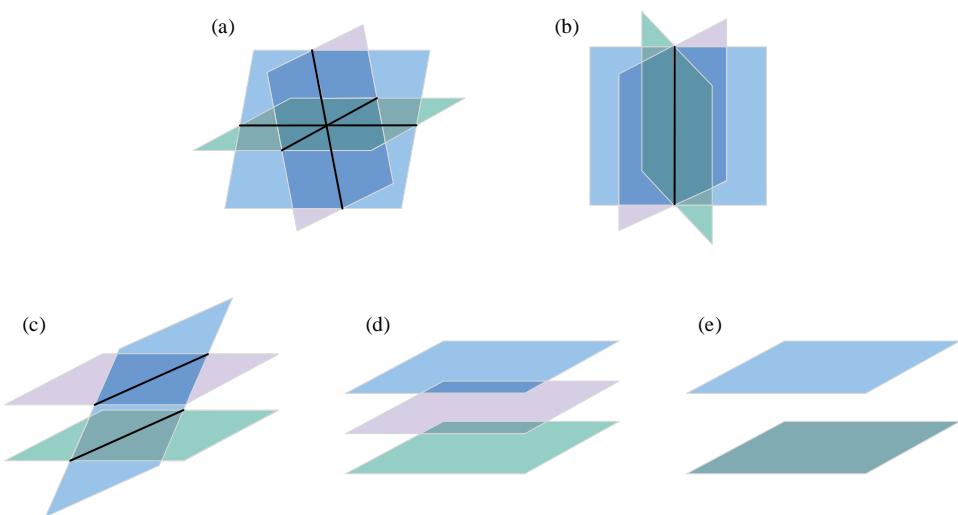
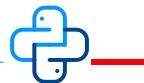


图 16. 图解三元一次方程组解的个数



Bk3_Ch6_02.py 绘制图 14。请大家自行修改代码绘制图 15。

6.4 不等式：划定区域

如图 17 所示，代数中，**等式** (equality) 可以是确定的值 ($x = 1$)、确定的直线 ($x + y = 1$)、确定的曲线 ($x^2 + y^2 = 1$)、确定的平面 ($-x + y - z + 1 = 0$) 等等。

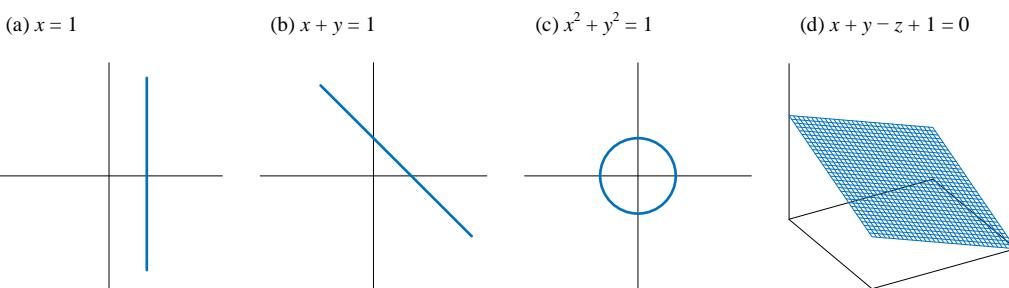


图 17. 等式的几何意义

然而，如图 18 所示，**不等式** (inequality) 的几何意义则是划定区域，比如 x 的取值范围 ($x < 1$)、直线在平面上划定的区域 ($x + y \leq 1$)、曲线在平面上划定的区域 ($x^2 + y^2 > 1$)、平面分割三维空间 ($-x + y - z + 1 < 0$)。

图 18 中当边界为虚线时，意味着划定区域不包括蓝色边界线。

⚠ 注意，图 18 中蓝色箭头指向满足不等式条件区域方向，蓝色箭头和梯度向量 (gradient vector) 有关。本系列丛书《矩阵力量》一册将专门介绍梯度向量相关内容。

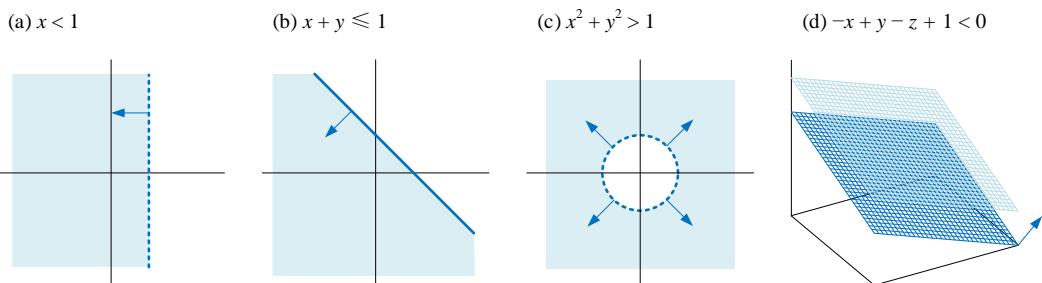


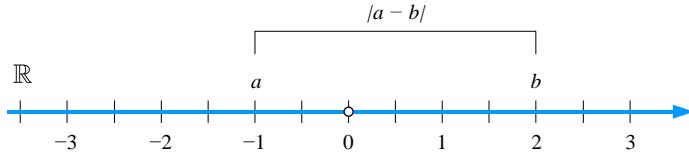
图 18. 不等式的几何意义

此外，图 17 和图 18 这两幅图告诉我们几何视角是理解代数式最直接的方式。本书在讲解每个数学工具式，都会给大家提供几何视角，以便加强理解，请大家格外留意。

数轴、绝对值、大小

为了理解不等式，让我们首先回顾数轴这个概念，数轴上的每一个点都对应一个实数，数轴上原点右侧的数为正数，原点左侧的数为负数。

某个数的**绝对值** (absolute value) 是指，数轴上该数与原点的距离。比如， $|-5| = 5$ (读作 the absolute value of negative five equals five) 可以理解为 -5 距离原点的距离为 5 个单位长度。 x 的绝对值记做 $|x|$ (读作 absolute value of x)。显然，实数绝对值为非负数，即 $|x| \geq 0$ 。

图 19. 实数轴上比较 a 和 b 大小

如果两个实数相等，这就意味着它们位于数轴同一点。当两个数不相等时，位于数轴左侧的数较小。如图 19 所示，实数 a 小于实数 b ，可以表达为 $a < b$ (读作 a is less than b)。也可以说，在数轴上 a 在 b 的左侧 (a is to the left of b on the number line)。

表 1 总结六个不等式符号。这种用**不等号** (inequality sign) 表达的式子被称作为不等式。

表 1. 六个不等式符号

数学表达	英文表达	汉语表达
------	------	------

<	less than	小于
>	greater than	大于
\leq	less than or equal to	小于等于
\geq	greater than or equal to	大于等于
\ll	much less than	远小于
\gg	much greater than	远大于

表 2. 不等式相关的英文表达

数学表达	英文表达
$4 > 3$	Four is greater than three. Three is less than four.
$y \leq 9$	Small y is less than or equal to nine.
$x \geq -1$	Small x is greater than or equal minus one.
$-3 < x < 2$	Small x is greater than minus three and less than two.
$0 \leq x \leq 1$	x is greater than or equal to zero and less than or equal to one.
$a < b$	a is less than b .
$a > b$	a is greater than b .
$a \leq b$	a is less than or equal to b . a is not greater than b .
$a \geq b$	a is greater than or equal to b . a is not less than b .
$a \ll b$	a is much less than b .
$a \gg b$	a is much greater than b .
$a \approx b$	a is approximately equal to b .
$a \neq b$	a is not equal to b .

区间

在数学上，某个变量的上下界可以写成区间。集合角度来看，**区间** (interval) 是指在一定范围的数的集合。

通用的区间记号中，圆括号表示“排除”，方括号表示“包括”。

如图 20 (a) 所示，**开区间** (open interval) 不包括区间左右端点，可以记作 (a, b) ，两端均为圆括号 (parentheses)。

如图 20 (b) 所示，**闭区间** (closed interval) 包括区间两端端点，可以记作 $[a, b]$ ，两端均为**方括号** (square brackets)。

如图 20 (c) 所示，**左开右闭区间** (left-open and right-closed)，可以记做 $(a, b]$ ，不包括区间左端点、包括右端点。如图 20 (d) 所示，**左闭右开区间** (right-open and left-closed)，可以记做 $[a, b)$ ，包括区间左端点、不包括右端点。

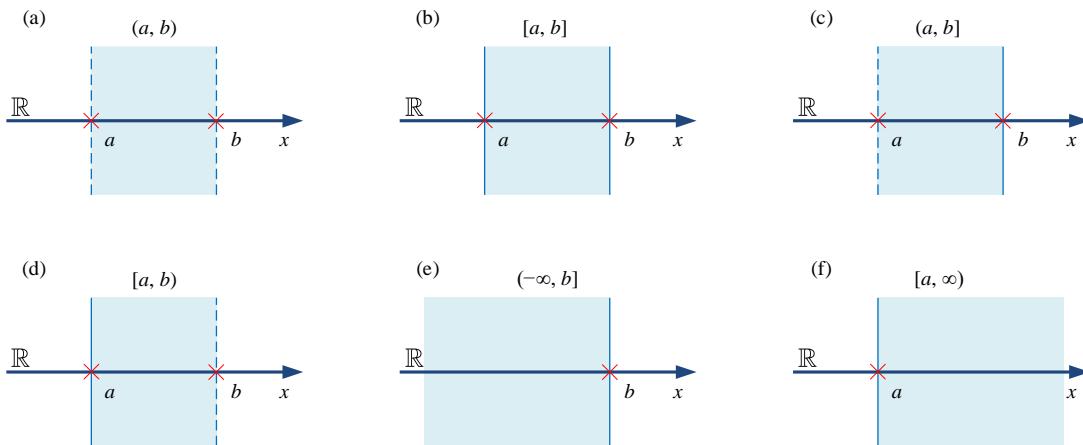


图 20. 六个区间

⚠ 请大家特别注意，在优化问题求解中，如果变量两端均有界，一般只考虑闭区间，即可以取到区间端点数值。也就是，图 20 中 (a)、(b)、(c)、(d) 对应的四个区间在优化问题中等价， a 叫做下界 (lower bound)， b 叫做上界 (upper bound)。

此外，构造优化问题时，一般都将各种不等式符号调整为小于等于号，即“ \leq ”。



本书后文将在第 19 章专门讲解优化问题和约束条件。

区间两端可能**有界** (bounded) 或**无界** (unbounded)，也就是区间某侧可能没有端点，即为无穷。**正无穷** (infinity) 记作 ∞ 或 $+\infty$ ，**负无限** (negative infinity) 记作 $-\infty$ 。

图 20 (e) 所示为**左无界右有界** (left-unbounded and right-bounded) 区间，比如 $(-\infty, b]$ 。

图 20 (f) 所示为**左有界右无界** (left-bounded and right-unbounded) 区间，比如 $[a, \infty)$ 。

左右均无界 (unbounded at both ends)，即 $(-\infty, \infty)$ ，代表整根实数轴。

表 3. 区间相关的英文表达

数学表达	英文表达
(a, b) $\{x \in \mathbb{R} \mid a < x < b\}$	The open interval from a to b . The interval from a to b , exclusive. The values between a and b , but not including the endpoints. x is greater than a and less than b . The set of all x such that x is in between a and b , exclusive.
$[a, b]$ $\{x \in \mathbb{R} \mid a \leq x \leq b\}$	The closed interval from a to b . The interval from a to b , inclusive. The values between a and b , including the endpoints. x is greater than or equal to a and less than or equal to b . The set of all x such that x is in between a and b , inclusive.
$(a, b]$ $\{x \in \mathbb{R} \mid a < x \leq b\}$	The half-open interval from a to b , excluding a and including b . The values between a and b , excluding a and including b . The set of all x such that x is greater than a but less than or equal to b .

$[a, b)$ $\{x \in \mathbb{R} \mid a \leq x < b\}$	The half-open interval from a to b , including a and excluding b . The values between a and b , including a and excluding b . The set of all x such that x is greater than or equal to a but less than b .
--	--

6.5 三大类不等式：约束条件

本节介绍不等式的目的是服务优化问题求解，优化问题中不等式一般分为三大类：

- ◀ **上下界** (lower and upper bounds), 比如 $x > 2$
- ◀ **线性不等式** (linear inequalities), 比如 $x + y \leq 1$
- ◀ **非线性不等式** (nonlinear inequalities), 比如 $x^2 + y^2 \geq 1$

在优化问题中，这些不等式统称为**约束** (constraint)，即限制变量的取值范围。本节后续将采用三种可视化方案呈现不等式划定的区域。

上下界

举个例子，给定 x_1 的取值范围为：

$$x_1 + 1 > 0 \quad (10)$$

首先将上式“大于号”调整为“小于号”，(10) 改写成：

$$-x_1 - 1 < 0 \quad (11)$$

⚠ 注意，本节后续不再区分 $<$ 和 \leq 。

根据 (11)，构造如下二元函数 $f(x_1, x_2)$ ：

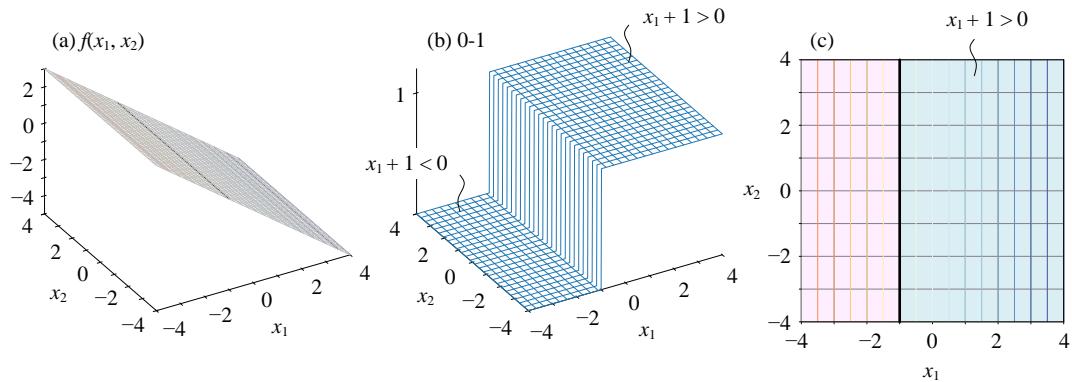
$$f(x_1, x_2) = -x_1 - 1 \quad (12)$$

图 21 (a) 所示为三维直角坐标系中 $f(x_1, x_2)$ 的等高线图 (contour plot)。对于一个二元函数 $f(x_1, x_2)$ ，等高线代表函数值相等的点连成的线，即满足 $f(x_1, x_2) = c$ 。函数等高线类似地形图上海拔高度相同点连成曲线。等高线可以在三维空间展示，也可以在平面上绘制。



对于等高线这个概念陌生的读者不要怕，本书第 10 章将深入介绍等高线。此外，本书第 13 章将专门讲解常用二元函数，本节内容相当于热身。

图 21 (a) 三维等高线采用“红黄蓝”色谱。暖色系颜色等高线对应 $f(x_1, x_2) > 0$ ，即不满足 (11)；冷色系颜色等高线对应 $f(x_1, x_2) < 0$ ，满足 (11)。值得注意的是，图 21 (a) 中等高线相互平行。

图 21. $x_1 + 1 > 0$ 三个可视化方案

然后，我们做一个“二分类”转换，满足(11)不等式的点 (x_1, x_2) 标签设为1(即True)，不满足(11)的点设为0(即False)，这样我们获得图21(b)。相当于把 $f(x_1, x_2)$ 变成一个0-1(False-True)两值阶梯面。

再进一步，将图21(a)等高线投影在 x_1x_2 平面上，获得图21(c)平面等高线。

图21(c)中黑色线就是决策边界，它将整个 x_1x_2 平面划分成两个区域：一个满足(11)，一个不满足(11)。图21(c)中，蓝色阴影区域满足(11)不等式，对应图21(b)中取值为1的区域。粉色阴影区域不满足(11)不等式，对应图21(b)中取值为0的区域。

再举个例子， x_1 的取值范围给定为：

$$-1 < x_1 < 2 \quad (13)$$

其中，-1为下限，2为上限。

利用绝对值运算，将(13)整理为：

$$|x_1 - 0.5| - 1.5 < 0 \quad (14)$$

可以这样理解(14)，数轴上离0.5距离小于1.5所有点的集合。

⚠ 注意，上式也可以看成是一个非线性不等式。

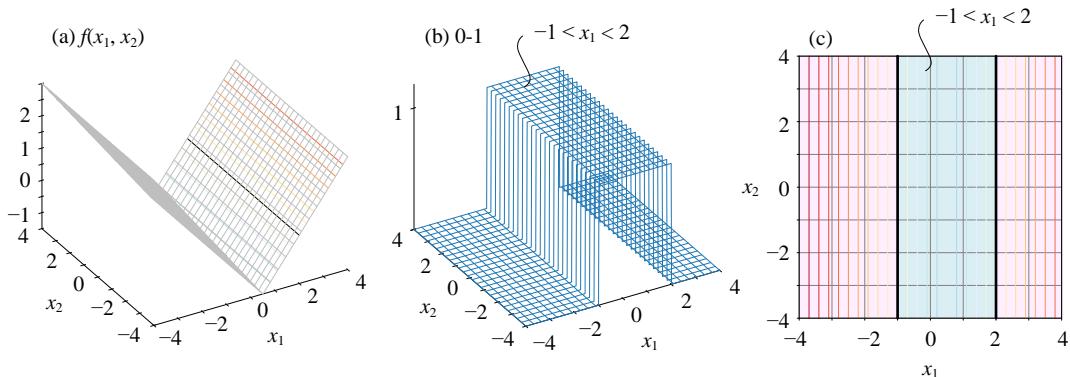
根据(14)，构造如下二元函数 $f(x_1, x_2)$ ：

$$f(x_1, x_2) = |x_1 - 0.5| - 1.5 \quad (15)$$

图22(a)所示为 $f(x_1, x_2)$ 函数在三维直角坐标系中图像，整个曲面呈现V字形。同样，蓝色等高线处满足(14)，而红色等高线处不满足(14)。

图22(b)中取值1的区域满足(14)。

图22(c)中背景色为蓝色区域满足(14)。图22(c)中两条黑色线为决策边界，两者相互平行。

图 22. $-1 < x_1 < 2$ 三个可视化方案

再举个例子，给定 x_2 的取值范围：

$$x_2 < 0 \text{ or } x_2 > 2 \quad (16)$$

⚠ 注意，上式可以看成两个区间构造而成。

将 (16) 整理为：

$$-|x_2 - 1| + 1 < 0 \quad (17)$$

可以这样理解上式，数轴上离 1 距离大于 1 的所有点的集合。

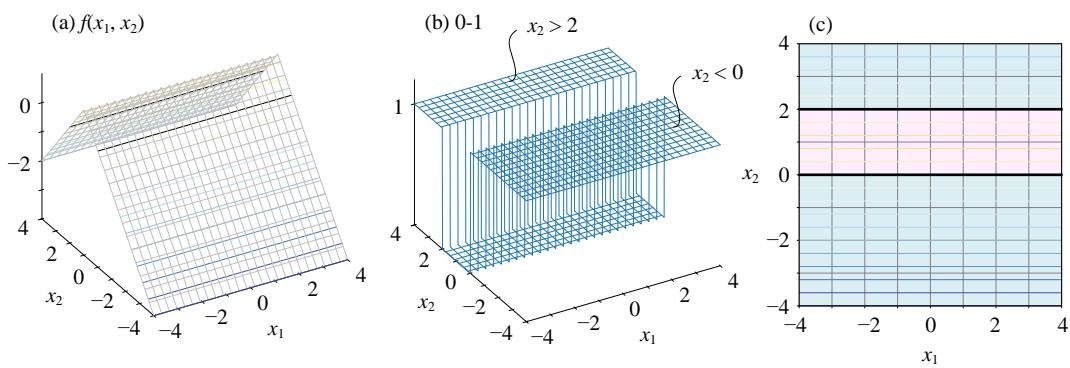
根据 (17) 构造如下二元函数 $f(x_1, x_2)$ ：

$$f(x_1, x_2) = -|x_2 - 1| + 1 \quad (18)$$

图 23 (a) 所示为二元函数 $f(x_1, x_2)$ 在三维直角坐标系中图像。

图 23 (b) 中 1 表示满足 (16)，0 表示不满足 (16)。

图 23 (c) 中蓝色背景色区域满足 (16)。

图 23. $x_2 < 0$ 或 $x_2 > 2$ 三个可视化方案

而几个不等式可以叠加构成不等式组。比如，(13) 和 (16) 叠加得到：

$$\begin{cases} -1 < x_1 < 2 \\ x_2 < 0 \text{ or } x_2 > 2 \end{cases} \quad (19)$$

这相当于在 x_1x_2 平面上，同时限定了 x_1 和 x_2 的取值范围。图 24 所示为同时满足 (19) 两组不等式的区域。请大家根据本节文末代码，自行绘制这两幅图像。

此外，(16) 就相当于两个不等式叠加，请大家用不等式叠加的思路再来分析 (16)。

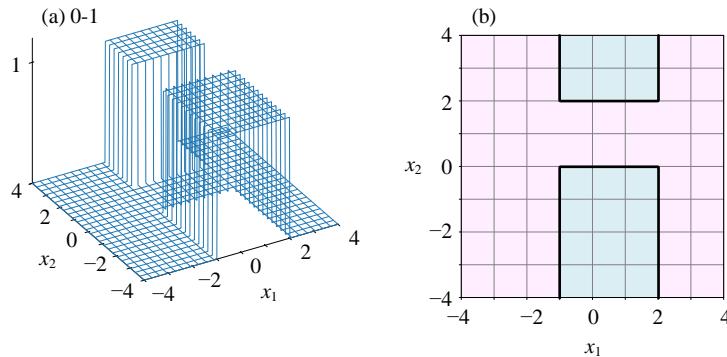


图 24. 同时满足 $-1 < x_1 < 2$ 和 $x_2 < 0$ 或 $x_2 > 2$ 对应区域

线性不等式

线性不等式就是一次不等式，也就是不等式中单项式的变量次数最高为 1 次。线性不等式中可以含有若干未知量。虽然上下界也可以看做是线性不等式，但是在构造优化问题时，我们还是将两类不等式分开处理。

举个例子，给定如下线性不等式：

$$x_1 - x_2 < -1 \quad (20)$$

将 (20) 整理为：

$$x_1 - x_2 + 1 < 0 \quad (21)$$

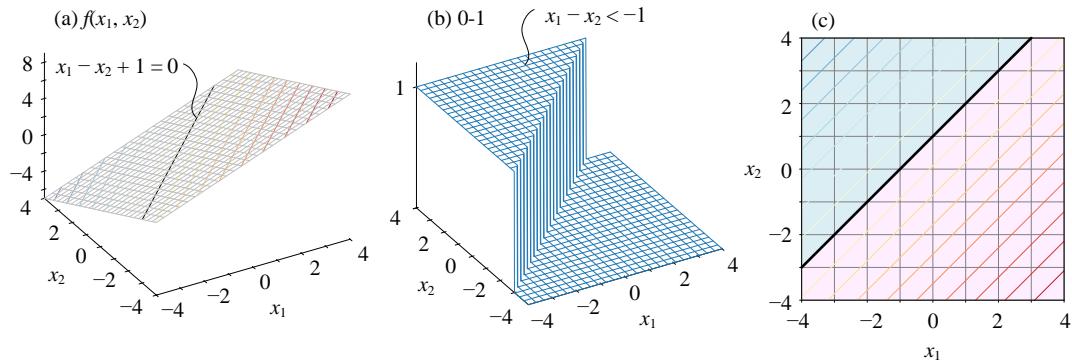
构造如下二元函数 $f(x_1, x_2)$ ：

$$f(x_1, x_2) = x_1 - x_2 + 1 \quad (22)$$

图 25 (a) 所示为 $f(x_1, x_2)$ 在三维直角坐标系的图像为斜面。

图 25 (b) 中取值为 1 的区域满足 (21)。

图 25 (c) 中蓝色阴影的区域满足 (21)，黑色直线对应等式 $x_1 - x_2 + 1 = 0$ 。

图 25. $x_1 - x_2 < -1$ 三个可视化方案

再举一个例子，给定如下线性不等式：

$$x_1 > 2x_2 \quad (23)$$

将 (23) 整理为：

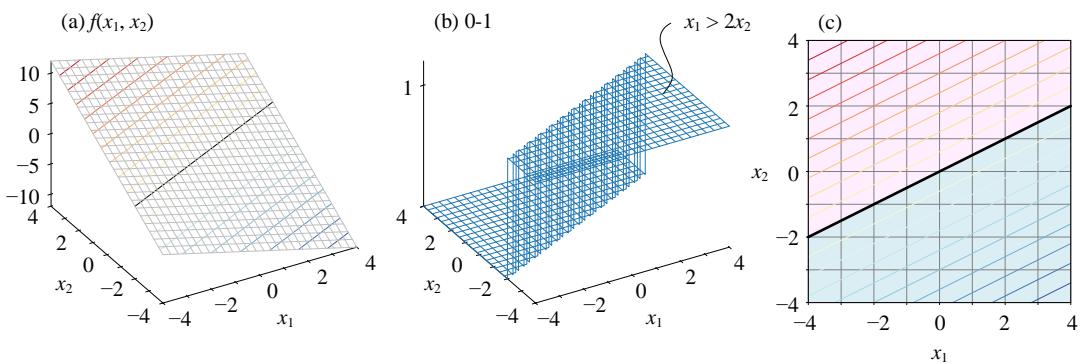
$$-x_1 + 2x_2 < 0 \quad (24)$$

根据 (24)，构造如下二元函数 $f(x_1, x_2)$ ：

$$f(x_1, x_2) = -x_1 + 2x_2 \quad (25)$$

图 26 (a) 中蓝色等高线满足 (23)，而红色等高线不满足 (23)。

图 26 (b) 中取值为 1 和 图 26 (c) 中蓝色阴影区域满足 (23)。

图 26. $x_1 > 2x_2$ 三个可视化方案

请大家将 (20) 和 (23) 两个不等式叠加构造一个不等式组，并绘制类似图 24 两图，可视化其划定的区域。

非线性不等式

除了线性不等式之外，其他各种形式的不等式都可以归类为非线性不等式。下面举三个例子。

给定如下绝对值构造的不等式：

$$|x_1 + x_2| < 1 \quad (26)$$

(26) 整理为：

$$|x_1 + x_2| - 1 < 0 \quad (27)$$

构造如下二元函数 $f(x_1, x_2)$ ：

$$f(x_1, x_2) = |x_1 + x_2| - 1 \quad (28)$$

图 27 (a) 所示为 (28) 对应三维直角坐标系图像。图 27 (b) 中取值为 1 对应的区域和图 27 (c) 中蓝色阴影区域满足 (26)。

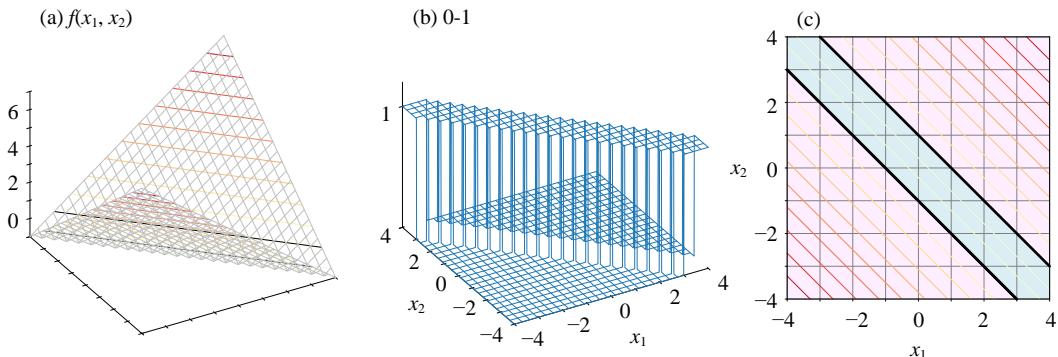


图 27. $|x_1 + x_2| < 1$ 三个可视化方案

此外，(26) 等价于：

$$(x_1 + x_2)^2 < 1 \quad (29)$$

请大家自行绘制 (29) 对应的三幅图像。

第二个例子，也用绝对值构造不等式：

$$|x_1| + |x_2| < 2 \quad (30)$$

将上式整理为：

$$|x_1| + |x_2| - 2 < 0 \quad (31)$$

构造如下二元函数 $f(x_1, x_2)$:

$$f(x_1, x_2) = |x_1| + |x_2| - 2 \quad (32)$$

图 28 (a) 所示为 $f(x_1, x_2)$ 等高线，有意思的是等高线为一个个旋转 45° 的正方形。大家还会在很多不同场合看到类似图像。图 28 (b) 中取值为 1 对应的区域和图 28 (c) 中蓝色阴影区域满足 (30)。

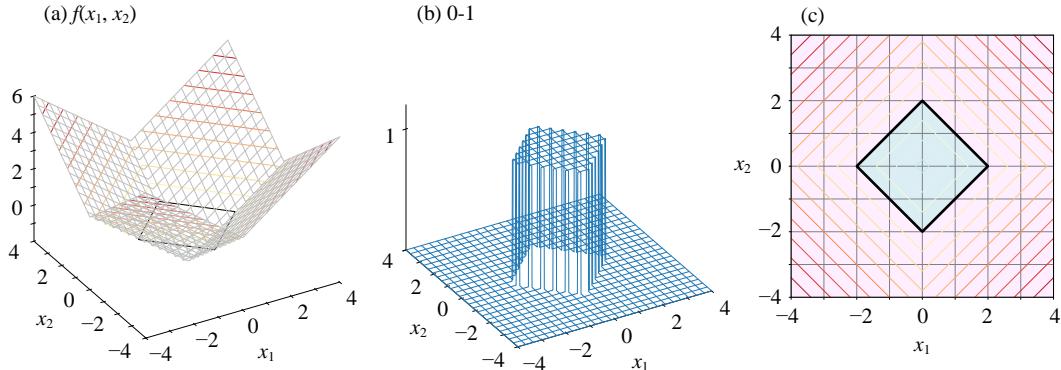


图 28. $|x_1| + |x_2| < 2$ 三个可视化方案

再看个例子，给定如下非线性不等式：

$$x_1^2 + x_2^2 < 4 \quad (33)$$

首先将整理为：

$$x_1^2 + x_2^2 - 4 < 0 \quad (34)$$

在 x_1x_2 平面上，构造如下二元函数 $f(x_1, x_2)$:

$$f(x_1, x_2) = x_1^2 + x_2^2 - 4 \quad (35)$$

图 29 (a) 所示为 (35) 中二元函数对应的曲面，曲面的等高线为同心圆。这种同心圆等高线还会在本书中反复出现，请大家留意。图 29 (b) 中取值为 1 对应的区域和图 28 (c) 中蓝色阴影区域满足 (33)。

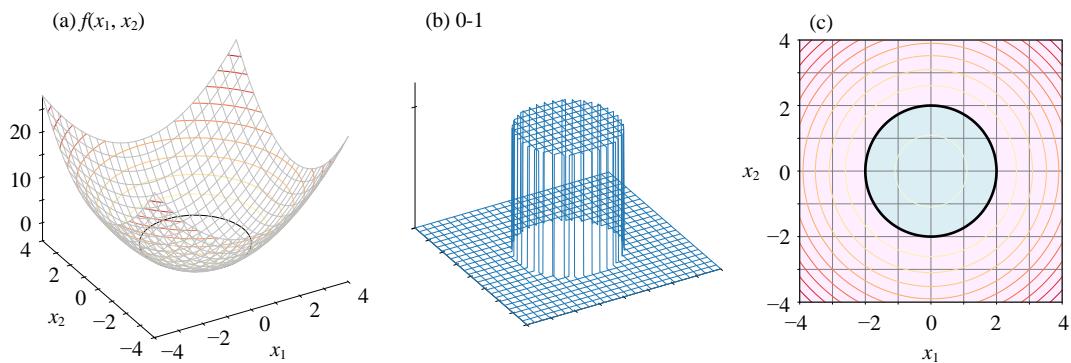


图 29. $x_1^2 + x_2^2 < 4$ 三个可视化方案

此外，(33) 等价于：

$$\sqrt{x_1^2 + x_2^2} < 2 \quad (36)$$

请大家自行绘制 (36) 对应的三幅图像。另外，请将 (26) 和 (33) 两个不等式叠加构造不等式组，并绘制取值区域。



Bk3_Ch6_03.py 绘制本节大部分图像。

6.6 三维极坐标

三维空间中也可以构造类似平面极坐标系的坐标系统，如图 30 (a) 所示的**球坐标系** (spherical coordinate system) 和图 30 (b) 所示的**圆柱坐标系** (cylindrical coordinate system)。

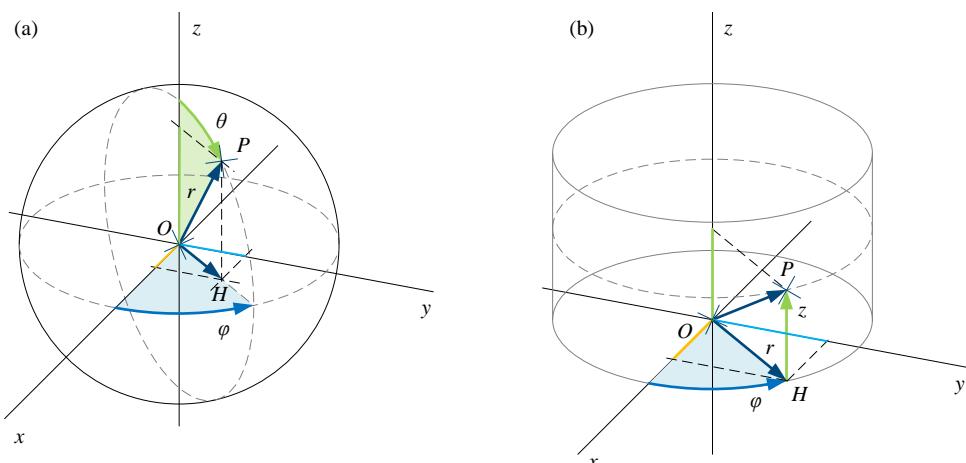


图 30. 球坐标系和圆柱坐标系

球坐标系

图 30 (a) 所示，球坐标相当于由两个平面极坐标系构造。

球坐标系中定位点 P 用的是球坐标 (r, θ, φ) 。其中， r 是 P 与原点 O 之间距离，也叫**径向距离** (radial distance)； θ 是 OP 连线和 z 轴正方向夹角，叫做**极角** (polar angle)； OP 连线在 xy 平面投影线为 OH ， φ 是 OH 和 x 轴正方向夹角，叫做**方位角** (azimuth angle)。

球坐标到三维直角坐标系坐标的转化关系为：

$$\begin{cases} x = r \sin \theta \cdot \cos \varphi \\ OH \\ y = r \sin \theta \cdot \sin \varphi \\ OH \\ z = \underbrace{r \cos \theta}_{PH} \end{cases} \quad (37)$$

图 31 所示正圆球体对应解析式为：

$$x_1^2 + x_2^2 + x_3^2 = r^2 \quad (38)$$

其中， $r = 1$ 。在绘制图 31 中这个正圆球体时，采用的就是球坐标。

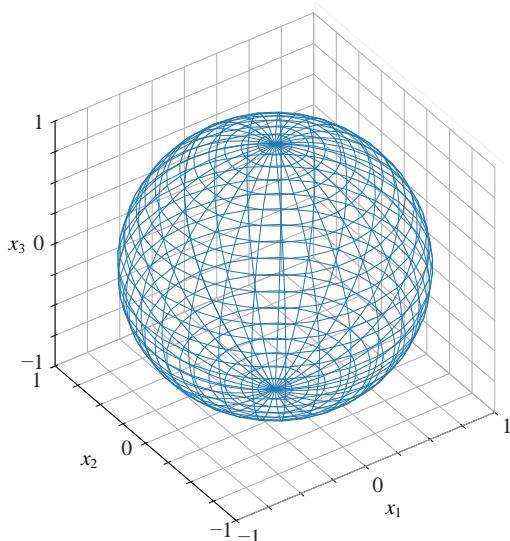
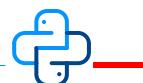


图 31. 球体网格面



Bk3_Ch6_04.py 绘制图 31。

圆柱坐标系

图 30 (b) 所示，圆柱坐标系相当于二维极坐标所在平面上在极点处升起一根 z 轴。

在圆柱坐标系中，点 P 的坐标为 (r, φ, z) 。这时， r 是 P 点与 z 轴的垂直距离； φ 还是 OP 在 xy 平面的投影线 OH 与正 x 轴之间的夹角； z 和三维直角坐标系的 z 一致。

从圆柱坐标到三维直角坐标系坐标转化关系为：

$$\begin{cases} x = r \cos \varphi \\ y = r \sin \varphi \\ z = z \end{cases} \quad (39)$$

上一章介绍的参数方程可以扩展到三维乃至多维。`plot3d_parametric_line()` 函数可以用来绘制参数方程构造的三维线图。

图 32 所示三维线图的参数方程就是采用圆柱坐标：

$$\begin{cases} x_1 = \cos(t) \\ x_2 = \sin(t) \\ x_3 = t \end{cases} \quad (40)$$

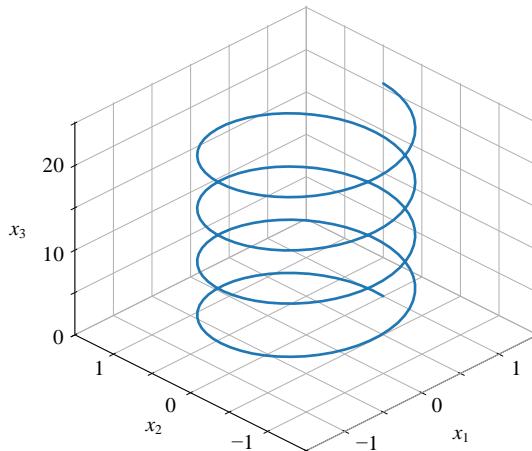
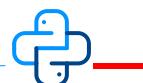


图 32. 三维参数方程线图



`Bk3_Ch6_05.py` 绘制图 32。图 32 也可以用 `plot3d_parametric_line()` 函数绘制，代码文件为 `Bk3_Ch6_06.py`。



坐标系让代数和几何紧密结合，坐标系使几何参数化，让代数可视化。

接下来第 7、8、9 三章，我们聊一聊解析几何相关内容。请大家特别注意距离、椭圆这两个数学工具的应用场合。

坐标系给一个个函数插上了翅膀，让它们能够在二维平面和三维空间自由翱翔。函数是本书第 10 到 13 章重点讲解的内容。



Distance

7 距离

人是万物的尺度



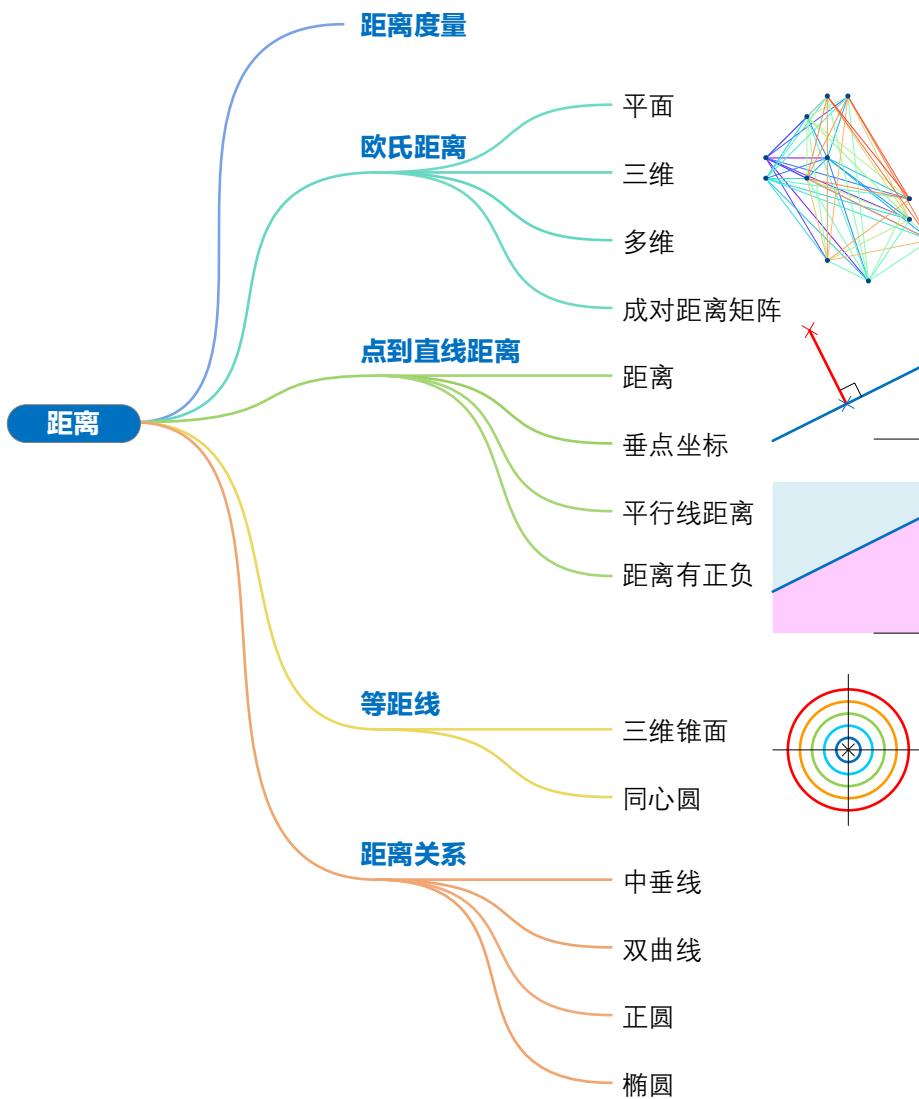
两点之间最短的路径是一条线段。

The shortest path between two points is a straight line.

—— 阿基米德 (Archimedes) | 古希腊数学家、物理学家 | 287 ~ 212 BC



- ◀ `matplotlib.pyplot.axhline()` 绘制水平线
- ◀ `matplotlib.pyplot.axvline()` 绘制竖直线
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `np.abs()` 计算绝对值
- ◀ `numpy.meshgrid()` 获得网格化数据
- ◀ `plot_wireframe()` 绘制三维单色线框图
- ◀ `sympy.abc()` 引入符号变量
- ◀ `sympy.lambdify()` 将符号表达式转化为函数
- ◀ `scipy.spatial.distance.mahalanobis()` 计算欧氏距离



7.1 距离：未必是两点间最短线段

阿基米德说“两点之间最短的路径是一条线段”，这个道理看似再简单不过。扔个肉包子给狗，狗会径直冲向包子。它应该不会拐几个弯，跑出优美曲线给你看。

但是，哪怕最基本的生活经验也会告诉我们，两点之间最短的路径不能简单地用两点线段来描述。图 1 和图 2 所示的四个路径规划就很好地说明这一点。

如果城市街区以正方形方格规划，如图 1 (a) 所示，从 A 点到 B 点，有很多路径可以选择。但是不管怎么选择路径，会发现这些路径都是由横平竖直的线段组合而成，而不是简单的“两点一线”。

图 1 (b) 所示，若城市的街区都是整齐的平行四边形，那么从 A 点到 B 点的路径就要依照四边形边的走势来规划。尽管如此，规划得到的路径依然是直线段的组合。

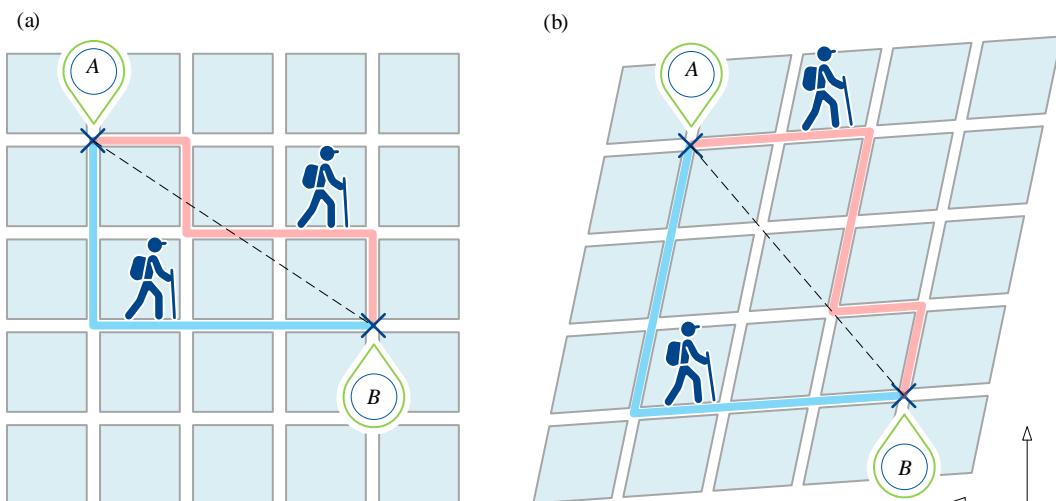


图 1. 两种直线段路径规划

有些城市街区的布置类似极坐标，呈现放射性网状。如图 2 (a)，从 A 点到 B 点的路径，便是直线段和弧线段的结合。

更常见的情况是，路径可能是由不规则曲线、折线构造得到。如图 2 (b) 所示，从 A 点到 B 点，别无选择只能按照一段自由曲线行走。

上述路径规划还是在平面上，这都是理想化的情况。实际情况中度量“距离”要复杂得多。如图 3 所示，计算地球上相隔很远的两个大陆上两点的距离要考虑的是一段弧线的长度。

让我们再增加一些复杂性，考虑山势起伏，具体如图 4 所示。这时，规划 A 点到 B 点的路径难度进一步提高。

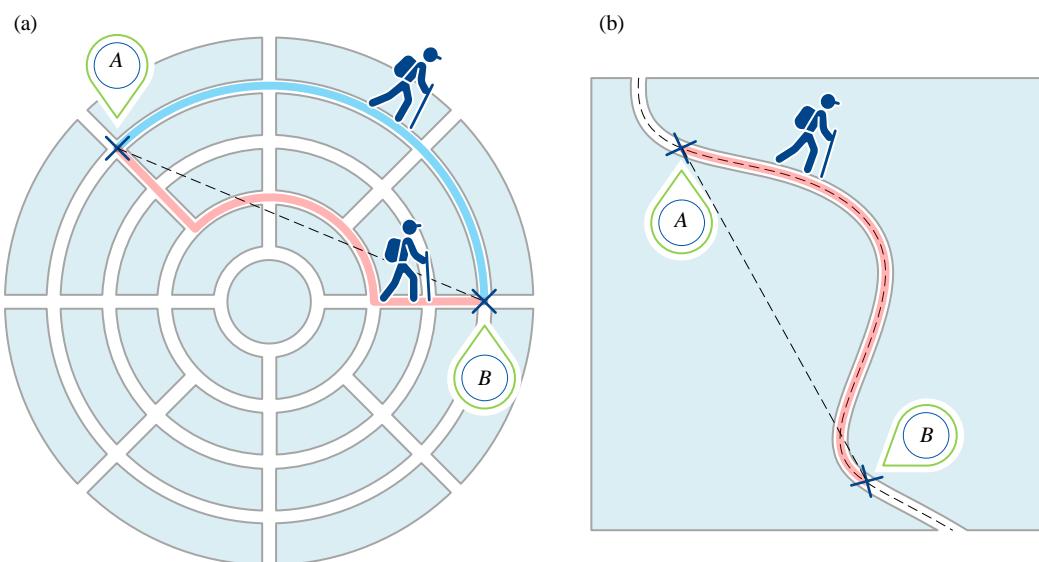


图 2. 两种非线性路径规划

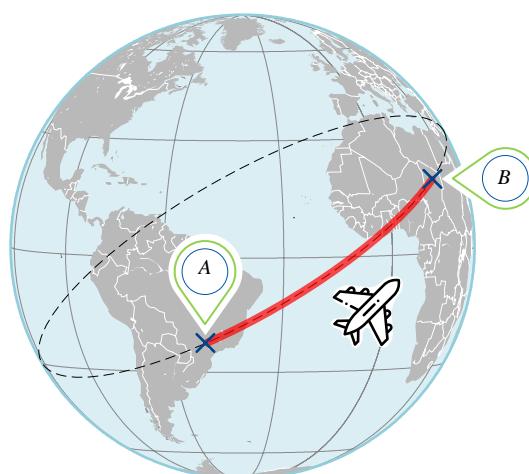
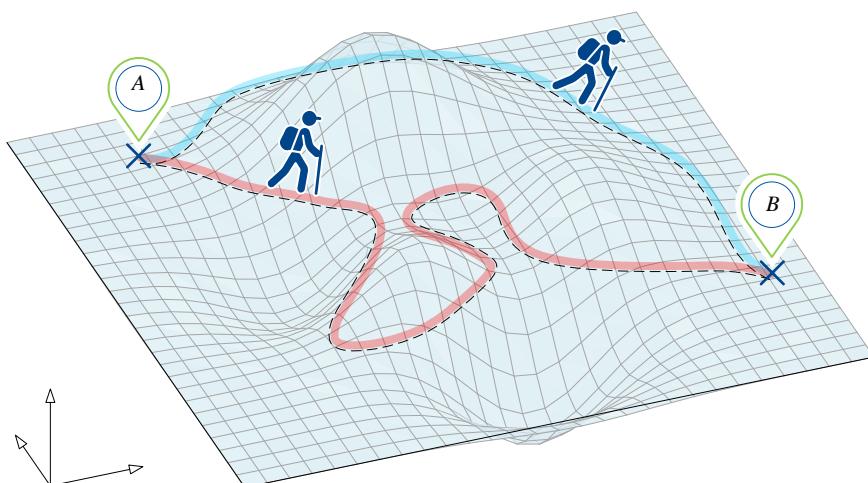


图 3. 地球表面两点距离



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 4. 考虑山势起伏的路径规划

此外，计算距离时还可以考虑数据的分布因素，得到的距离即所谓的**统计距离** (statistical distance)。

如图 5 所示， A 、 B 、 C 和 D 四点和 Q 点的直线距离相同。这个距离又叫欧几里得距离或欧氏距离。图 5 中蓝色散点代表样本数据的分布，考虑数据分布“紧密”情况，不难判断 C 点距离 Q 最近，而 D 距离 Q 最远。

也就是说，地理上的相近，不代表关系的紧密——相隔万里的好友，近在咫尺的路人。

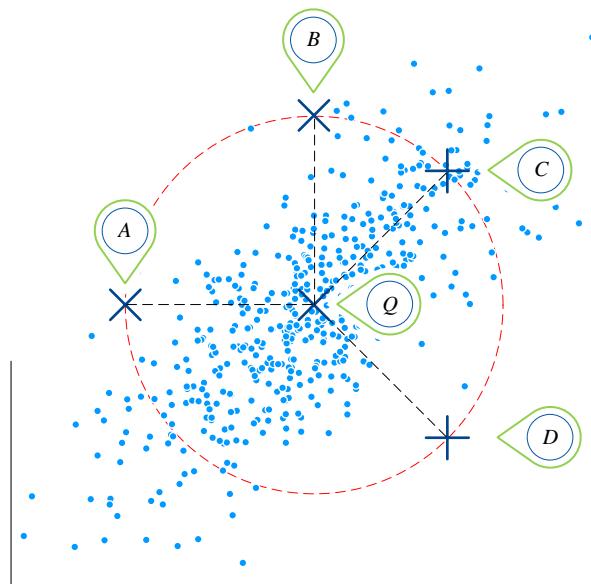


图 5. 考虑数据分布的距离度量

“距离”在数据科学和机器学习中非常重要。本章先从最简单的两点直线距离入手和大家探讨距离这个话题。本系列丛书后续会不断扩展丰富“距离”这个概念。

7.2 欧氏距离：两点间最短线段

两点之间线段长度叫做**欧几里得距离** (Euclidean distance)，或**欧氏距离**，它是最简单的距离度量。

本书前文讲过的绝对值实际上就是一维数轴上的两点距离。如图 6 所示，一维数轴上有 A 和 B 两点，它们的坐标值分别为 x_A 和 x_B 。 A 和 B 两点欧氏距离就是 x_A 和 x_B 之差的绝对值：

$$\text{dist}(A, B) = |x_A - x_B| \quad (1)$$

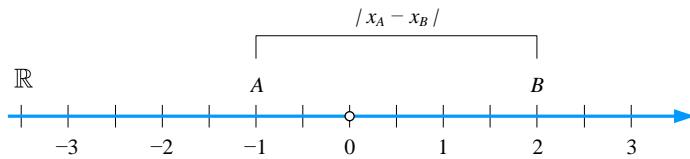


图 6. 实数轴上 A 和 B 距离

二维平面

平面直角坐标系中两点 $A(x_A, y_A)$ 和 $B(x_B, y_B)$ 的欧氏距离就是 AB 线段的长度，可以通过如下公式求得：

$$\begin{aligned} \text{dist}(A, B) &= \sqrt{|x_A - x_B|^2 + |y_A - y_B|^2} \\ &= \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \end{aligned} \quad (2)$$

上式用到的数学工具就是勾股定理。如图 7 所示，直角三角形的两个直角边边长为 $|x_A - x_B|$ 和 $|y_A - y_B|$ 。利用矩阵乘法，(2) 可以写成：

$$\text{dist}(A, B) = \sqrt{\begin{bmatrix} x_A - x_B & y_A - y_B \end{bmatrix} \begin{bmatrix} x_A - x_B \\ y_A - y_B \end{bmatrix}} \quad (3)$$

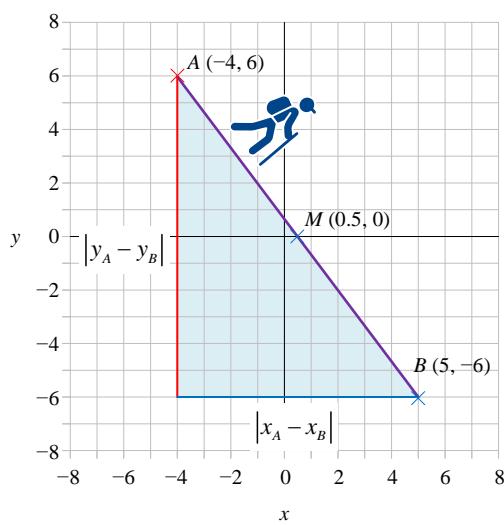
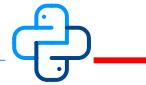


图 7. 平面直角坐标系，两点之间距离

A 和 B 点连线的中点 (midpoint) M 的坐标为：

$$M = \left(\frac{x_A + x_B}{2}, \frac{y_A + y_B}{2} \right) \quad (4)$$



Bk3_Ch7_01.py 绘制图 7。

三维空间

类似地，三维直角坐标系中两点 $A(x_A, y_A, z_A)$ 和 $B(x_B, y_B, z_B)$ 的距离可以通过如下公式求得：

$$\text{dist}(A, B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \quad (5)$$

图 8 给出一个计算三维空间两点欧氏距离的例子。容易发现，计算过程两次使用勾股定理。

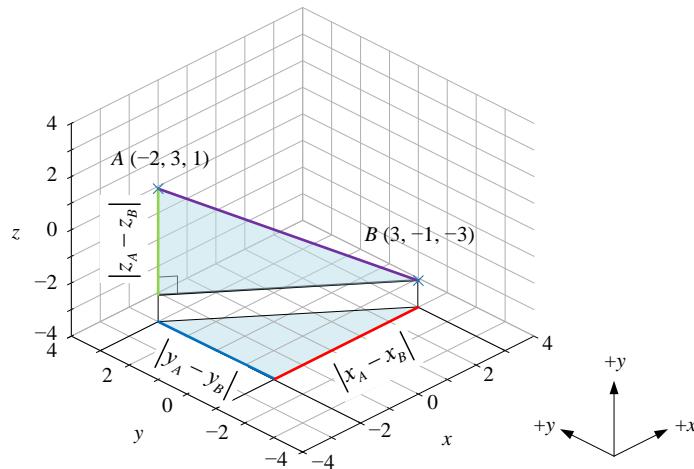


图 8. 三维直角坐标系，两点之间距离

我们也可以把 (5) 推广到多维。 D 维空间两点 $A(x_{1,A}, x_{2,A}, \dots, x_{D,A})$ 和 $B(x_{1,B}, x_{2,B}, \dots, x_{D,B})$ 的距离可以通过如下公式求得：

$$\text{dist}(A, B) = \sqrt{(x_{1,A} - x_{1,B})^2 + (x_{2,A} - x_{2,B})^2 + \dots + (x_{D,A} - x_{D,B})^2} \quad (6)$$

同样利用矩阵乘法，(6) 可以写成：

$$\text{dist}(A, B) = \sqrt{\begin{bmatrix} x_{1,A} - x_{1,B} & x_{2,A} - x_{2,B} & \cdots & x_{D,A} - x_{D,B} \end{bmatrix} \begin{bmatrix} x_{1,A} - x_{1,B} \\ x_{2,A} - x_{2,B} \\ \vdots \\ x_{D,A} - x_{D,B} \end{bmatrix}} \quad (7)$$

对比(5)和(7)，大家已经明白，为什么我们要用 x 加下角标作为变量，因为变量真的不够用。 (x, y, z) 中利用了三个字母代表变量，如果空间的维度为100维，英文字母显然都不够用。采用“变量+下角标索引”这种方式，100维空间坐标可以轻松写成 $(x_1, x_2, x_3, \dots, x_{100})$ 。



Bk3_Ch7_02.py 绘制图8。

成对距离

在数据科学和机器学习实践中，我们经常遇到如图9所示这种多点成对距离(pairwise distance)的情况。在图9这个平面上，一共有12个点。而这12点一共可以构造得到 $66(C_{12}^2)$ 个两点距离。

用什么结构存储、运算及展示这些距离值，成了一个问题。

这时，矩阵就可以派上大用场！

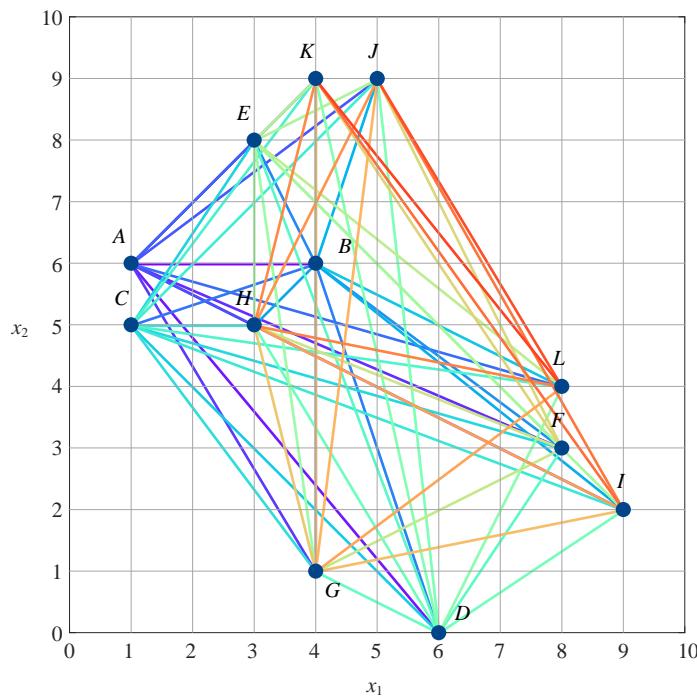


图9. 平面上12个点，成对距离

如图10所示，矩阵的形状为 12×12 ，即12行、12列。矩阵的主对角线元素都是0，这是某点和自身的距离，矩阵非主对角线元素则代表成对距离。

很容易发现，这个矩阵关于主对角线对称。也就是说，我们只需要主对角线斜下方的 66 个元素，或者主对角线斜上方的 66 个元素。这 66 个元素涵盖了我们要保存的所有两点距离。

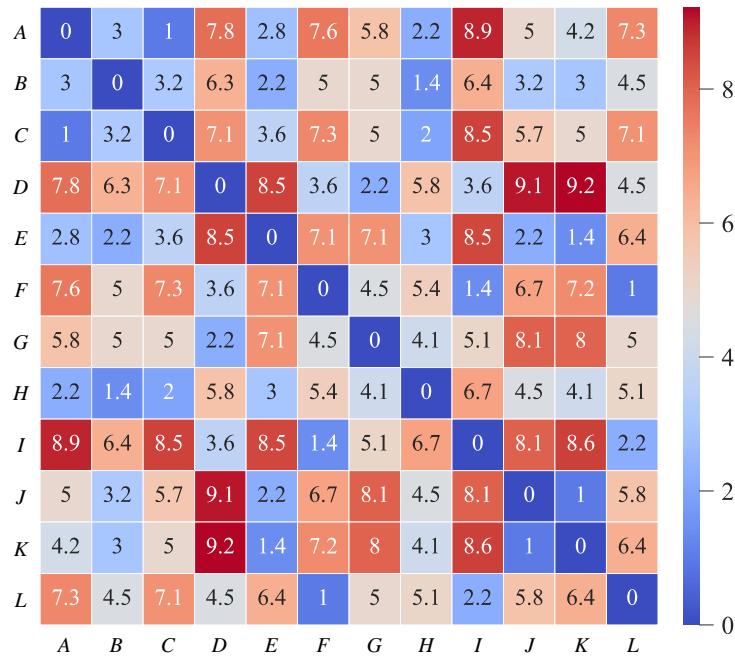


图 10. 成对距离矩阵

下三角矩阵、上三角矩阵

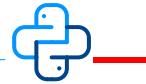
多讲一点，提取类似图 10 方阵中主对角线及其左下方元素，它们单独构成的矩阵叫做**下三角矩阵** (lower triangular matrix) L 。

而主对角线和其右上方元素单独构成的矩阵叫做**上三角矩阵** (upper triangular matrix) R ，其余元素为 0。图 11 给出下三角矩阵和上三角矩阵的示意图。下三角矩阵 L 转置得到的 L^T 便是上三角矩阵 R 。

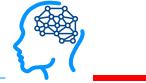
⚠ 注意，图 10 中绿色方框之外代表的元素均为 0。



图 11. 下三角矩阵和上三角矩阵



Bk3_Ch7_03.py 计算成对距离，并且绘制图 9 和图 10。



机器学习很多算法中常常用到**亲近度** (affinity)，亲近度和距离正好相反。两点距离越远，两者亲近度越低；而当它们距离越近，亲近度则越高。

我们假设亲近度的取值在 $[0, 1]$ 这个区间。1 代表两点重合，也就是距离为 0，亲近度最大；亲近度为 0 代表两点相距无穷远。

摆在我面前的一个数学问题就是，如何把距离转化成亲近度。如图 12 所示，我们需要某种“映射”关系，将“欧氏距离”和“亲密度”一一联系起来。

自然而然地，我们就会想到代数中的“函数”。函数就是映射，可以完成数据转换。

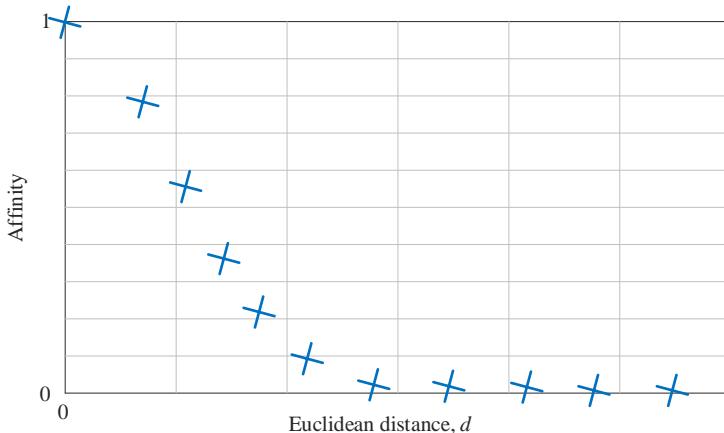
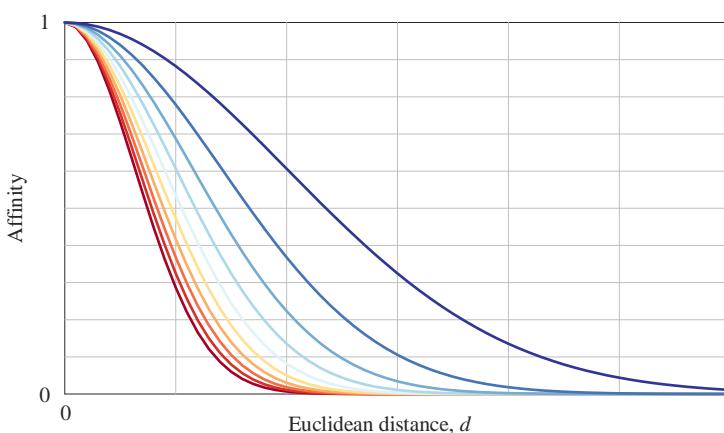


图 12. 如何设计“欧氏距离”到“亲密度”的映射关系

而众多函数当中，高斯函数便可以胜任这一映射要求。高斯函数的一般解析式为：

$$f(d) = \exp(-\gamma d^2) \quad (8)$$

图 13 所示为参数 γ 影响高斯函数右半侧曲线形状。本书后续会介绍高斯函数性质。本系列丛书在《机器学习》一册会专门介绍“亲近度”这个概念。

图 13. 参数 γ 影响高斯函数右半侧曲线形状

7.3 点到直线的距离

给定平面上一条直线 $l: ax + by + c = 0$, 直线外一点 $A(x_A, y_A)$ 到该直线的距离为:

$$\text{dist}(A, l) = \frac{|ax_A + by_A + c|}{\sqrt{a^2 + b^2}} \quad (9)$$

直线 l 上距离 A 最近点的坐标为 $H(x_H, y_H)$:

$$\begin{aligned} x_H &= \frac{b(bx_A - ay_A) - ac}{a^2 + b^2} \\ y_H &= \frac{a(-bx_A + ay_A) - bc}{a^2 + b^2} \end{aligned} \quad (10)$$

A 和 H 的连线得到 AH 线段长度就是 (9)。

特别地, 当 $a = 0$ 时, 直线 l 为水平线。 $A(x_A, y_A)$ 到该直线的距离为:

$$\text{dist}(A, l) = \frac{|by_A + c|}{|b|} \quad (11)$$

当 $b = 0$ 时, 直线 l 为竖直线。 $A(x_A, y_A)$ 到该直线的距离为:

$$\text{dist}(A, l) = \frac{|ax_A + c|}{|a|} \quad (12)$$

举个例子, 图 14 给定直线 $x - 2y - 4 = 0$, $A(-4, 6)$ 到直线距离最近点为 $H(0, -2)$ 。大家可以自己算一下, A 到直线的距离为 8.944。

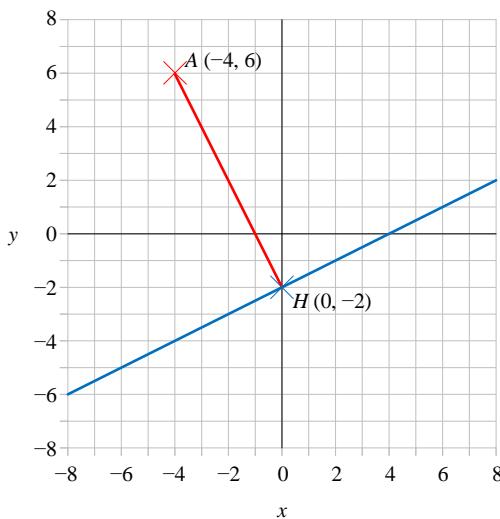


图 14. 平面直角坐标系，点到直线距离



Bk3_Ch7_04.py 计算点 A 到直线距离，并绘制图 14。

平行线间距离

给定如下两条平行线 l_1 和 l_2 对应解析式：

$$\begin{cases} ax + by + c_1 = 0 \\ ax + by + c_2 = 0 \end{cases} \quad (13)$$

其中， $c_1 \neq c_2$ 。

这两条平行线的距离为：

$$\text{dist}(l_1, l_2) = \frac{|c_1 - c_2|}{\sqrt{a^2 + b^2}} \quad (14)$$

在本系列丛书《矩阵力量》一册，我们会利用线性代数运算工具来求解点到直线距离，及两条平行线之间的距离。

距离也可以有“正负”

本章前文介绍的距离都是“非负值”；但是，在机器学习算法中，我们经常会给距离度量加个正负号。下面举几个例子。

如图 15 所示，在数轴上，以 Q 点作为比较的基准点，距离 AQ 和 BQ 的定义分别为：

$$\begin{aligned}\text{dist}(A, Q) &= |x_A - x_Q| \\ \text{dist}(B, Q) &= |x_B - x_Q|\end{aligned}\quad (15)$$

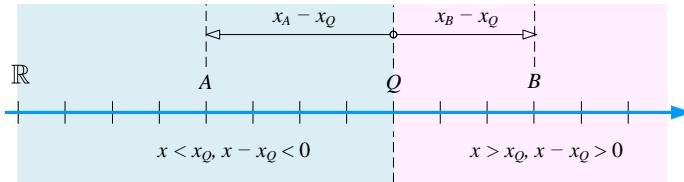


图 15. 一维数轴上距离的正负

将上两式中绝对值去掉，得到：

$$\begin{aligned}\text{dist}(A, Q) &= x_A - x_Q \\ \text{dist}(B, Q) &= x_B - x_Q\end{aligned}\quad (16)$$

图 15 中， A 在 Q 的左边，因此 $x_A - x_Q < 0$ ，也就是距离为“负”；而 B 在 Q 的右边，因此 $x_B - x_Q > 0$ ，也就是距离为“正”。

距离的绝对值告诉我们两点的远近，距离的“正负”符号多了相对位置这层信息。

上一章介绍的不等式有划定区域作用。也就是说，(16) 这种含“正负”的距离把不等式“区域”这层信息也囊括进来。

同理，将 (9) 分子上的绝对值符号去掉，点 A 和直线 l 的距离为：

$$\text{dist}(A, l) = \frac{ax_A + by_A + c}{\sqrt{a^2 + b^2}} \quad (17)$$

以图 16 为例，图中直线 l 的解析式为 $x + y - 1 = 0$ ；这条直线把平面直角坐标系划分成两个区域—— $x + y - 1 > 0$ （暖色背景）和 $x + y - 1 < 0$ （冷色背景）。

根据 (17)，计算 A 和 B 点到直线 l 的含“正负”距离分别为：

$$\text{dist}(A, l) = \frac{3}{\sqrt{2}}, \quad \text{dist}(B, l) = \frac{-5}{\sqrt{2}} \quad (18)$$

根据距离的“正负”符号，可以判断 A 点在 $x + y - 1 > 0$ 这个区域， B 点在 $x + y - 1 < 0$ 这个区域。请大家思考去掉 (14) 分子中绝对值符号后，两条平行线距离分别为正负值所代表的几何含义。

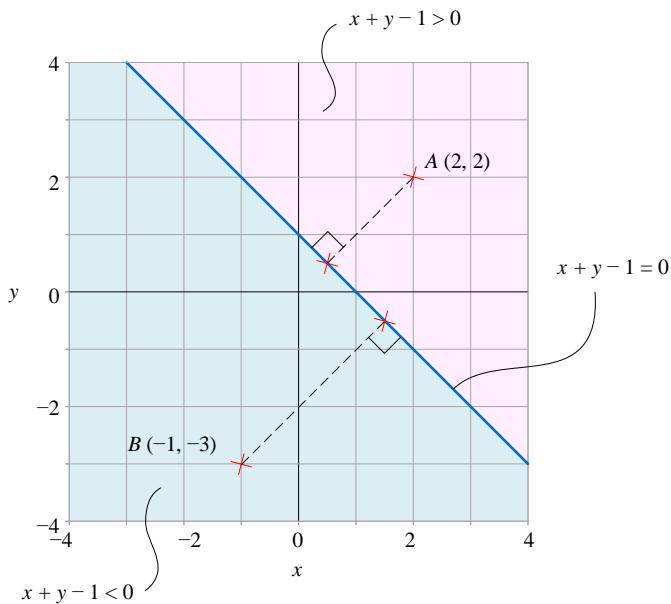


图 16. 点到直线距离的正负



支持向量机 (Support Vector Machine, SVM) 是非常经典的机器学习算法之一。支持向量机既可以用来分类，也可以用来处理回归问题。图 17 所示为支持向量机核心思路。

如图 17 所示，一片湖面左右散布着蓝色 ● 红色 ● 碣石，游戏规则是，皮划艇以直线路径穿越水道，保证船身恰好紧贴礁石。寻找一条直线路径，让该路径通过的皮划艇宽度最大，也就是图 17 中两条虚线之间宽度最大。这个宽度叫做**间隔** (margin)。

图 17 (b) 中加黑圈 ○ 的五个点，就是所谓的**支持向量** (support vector)。图 17 中深蓝色线就是水道，叫做**决策边界** (decision boundary)。决策边界将标签分别为蓝色 ● 红色 ● 数据点“一分为二”，也就是分类。

很明显，图 17 (b) 中规划的路径好于图 17 (a)，因为图 17 (b) 水道间隔明显更宽。而本节介绍的“距离”这个概念在支持向量机算法中扮演重要角色。

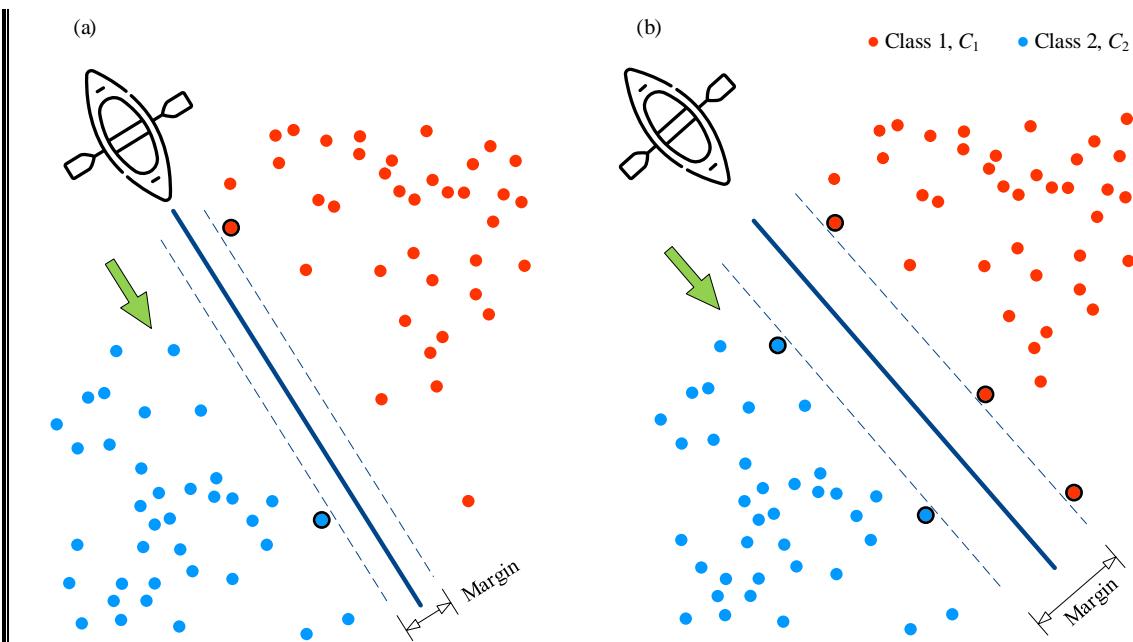


图 17. 支持向量机原理

如图 18 所示，计算“支持向量”A、B、C、D、E 和“决策边界”的距离，用到的就是本节讲到的“点到直线距离”；计算 l_1 和 l_2 “间隔”宽度用到的是“平行线间距离”

而图 18 中暖色和冷色两个区域就是通过不等式划定的区域。暖色区域的样本点分类为红色●，即 C_1 ；冷色区域的样本点分类为蓝色●，即 C_2 。

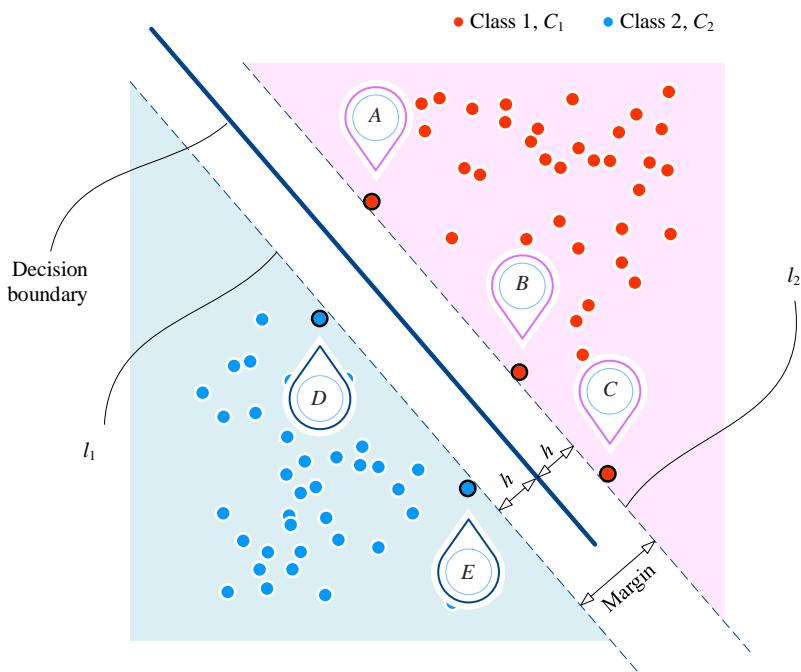


图 18. “距离”在 SVM 算法中扮演的角色

7.4 等距线：换个视角看距离

任意一点 $P(x, y)$ 距离原点 $O(0, 0)$ 的欧氏距离为 r , 对应的解析式为：

$$\text{dist}(P, O) = \sqrt{x^2 + y^2} = r \quad (19)$$

上式左右两侧平方得到下式：

$$x^2 + y^2 = r^2 \quad (20)$$

这样，我们得到一个圆心位于原点、半径为 r 的正圆的解析式。

利用矩阵乘法，(20) 可以写成：

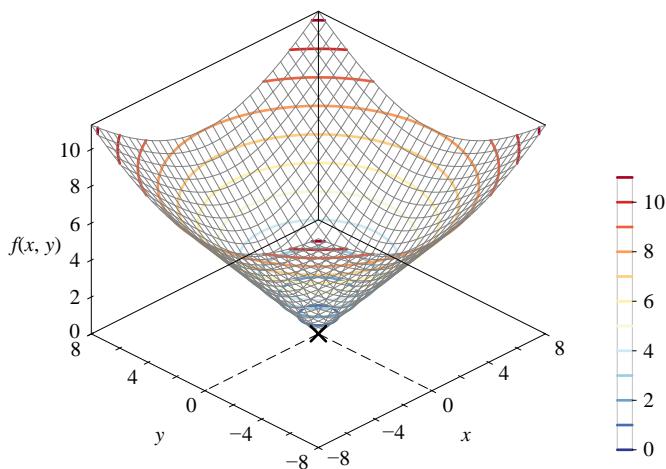
$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = r^2 \quad (21)$$

构造如下二元函数 $f(x, y)$ ：

$$f(x, y) = \sqrt{x^2 + y^2} \quad (22)$$

其中， x 和 y 为自变量。

图 19 所示为 $f(x, y)$ 在三维直角坐标系的曲面形状，这个曲面显然为圆锥。曲面上我们还特地绘制了等高线 (contour line 或 contour)。上一章介绍过，等高线指的是 $f(x, y)$ 上值相等的相邻各点所连成的曲线。

图 19. 三维直角坐标系， $f(x, y)$ 函数曲面

将图 19 等高线投影到 xy 平面上，便得到如图 20 所示的平面等高线，我们管它叫等距线。图 20 中每条等距线对应的就是 $f(x, y) = r$ 截面图像。观察图 19，很容易发现 r 取不同值时对应一系列同心圆。也就是说，距离原点 O 的欧氏距离取不同值时，等距线是一系列同心圆。

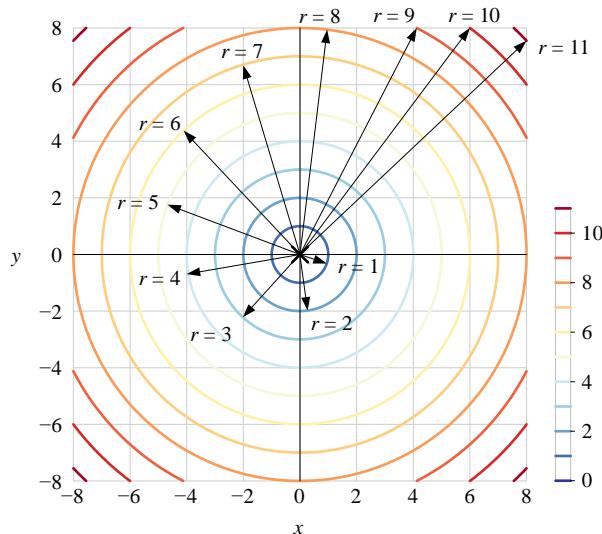
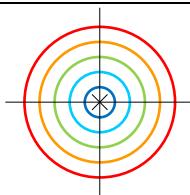


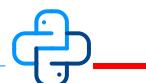
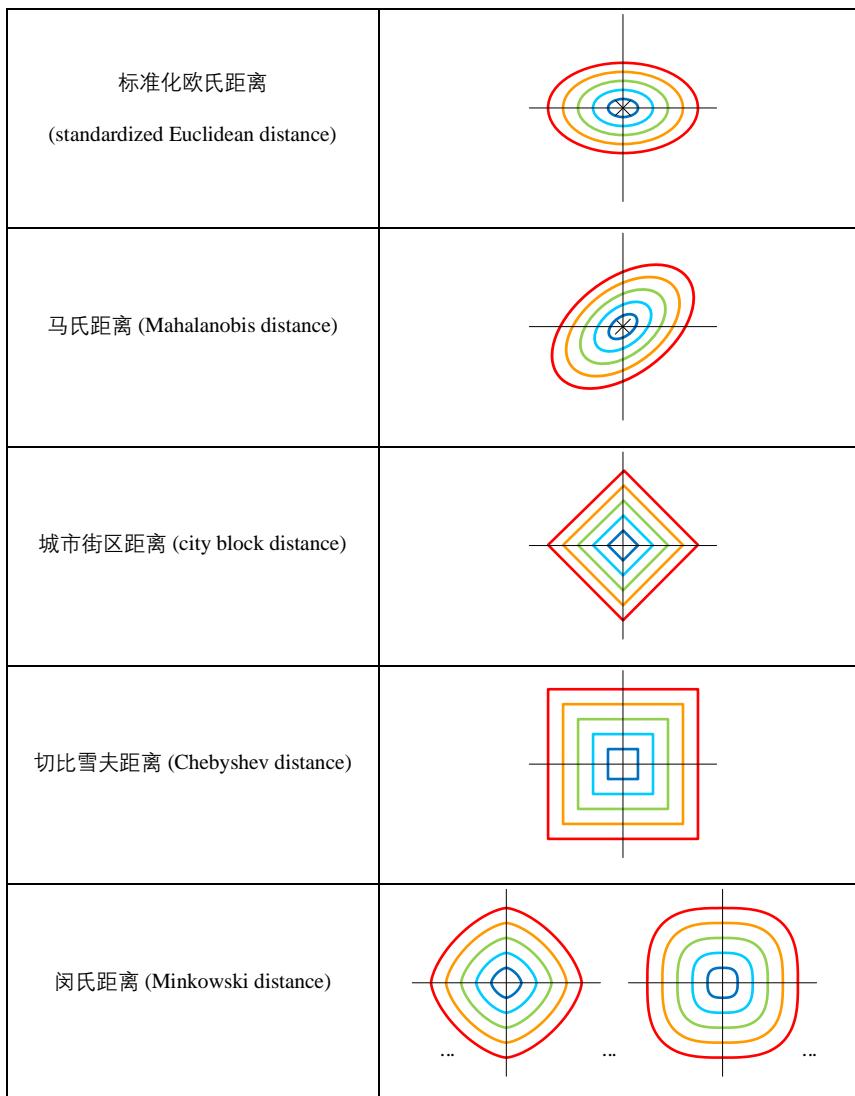
图 20. $f(x, y)$ 函数平面等高线，也就是欧氏距离等距线

⚠ 再次强调，在机器学习中，欧氏距离是最基础的距离度量。本系列丛书会和大家一起探讨各种距离度量，每种距离度量都有自己独特的“等距线”。

表 1 总结常见距离度量平面直角坐标系中的等距线形状。鉴于“距离”的多样性，没有特别说明的话，本书中的“距离”一般指“欧氏距离”。如有必要会专门说明距离度量是哪一种。本系列丛书将一一揭开表 1 距离度量的面纱。

表 1. 常见距离定义及等距线形状

距离度量	平面直角坐标系中等距线形状
欧氏距离 (Euclidean distance)	



Bk3_Ch7_05.py 绘制图 19 和图 20。请大家修改代码绘制 $f(x, y) = x^2 + y^2$ 这个函数的曲面三维等高线和平面等高线图。

7.5 距离间的量化关系

在平面直角坐标系，给定 A 和 B 两点，任意一点 P 到点 A 和点 B 的距离分别为 AP 和 BP 。本节讨论 AP 和 BP 之间存在的一些常见量化关系，以及对应 P 的运动轨迹。这一节同时也引出本书下两章有关圆锥曲线的内容。

中垂线

如果， AP 和 BP 等距，得到的是 A 和 B 两点的中垂线：

$$AP = BP \quad (23)$$

如图 21 所示， A 和 B 两点的中垂线垂直于 AB 线段，并且将 AB 等分。图 21 中的两组等高线，对应的是到 A 和 B 两点等距线，相同颜色代表相同距离。相同颜色等距线的交点显然都在中垂线上。

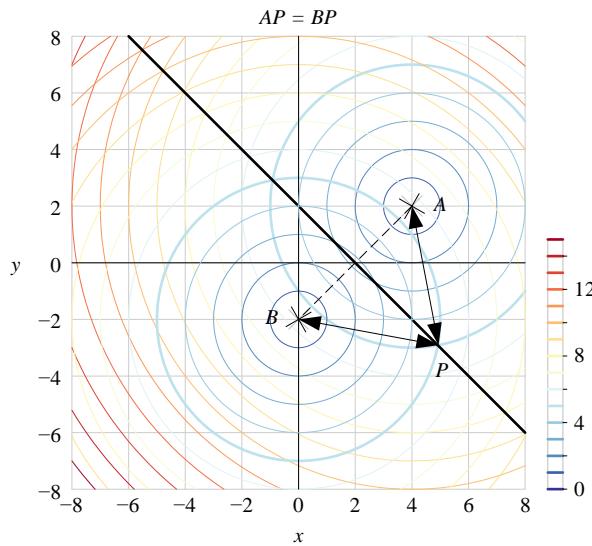


图 21. A 和 B 的中垂线

双曲线

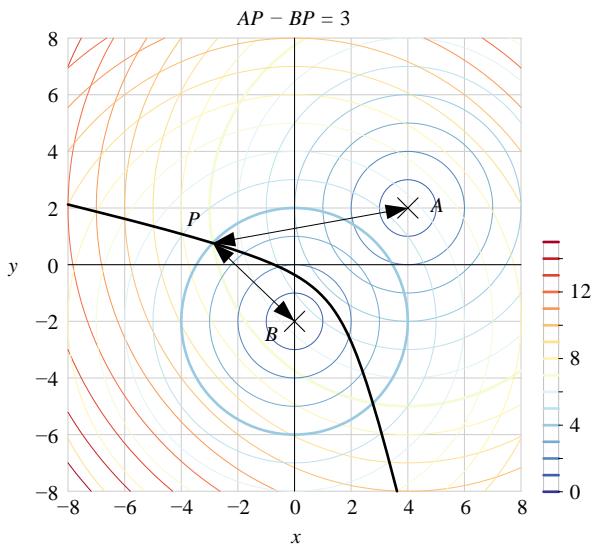
再看一种情况， AP 和 BP 之差为定值：

$$AP - BP = c \quad (24)$$

比如， AP 比 BP 长 3，即：

$$AP - BP = 3 \quad (25)$$

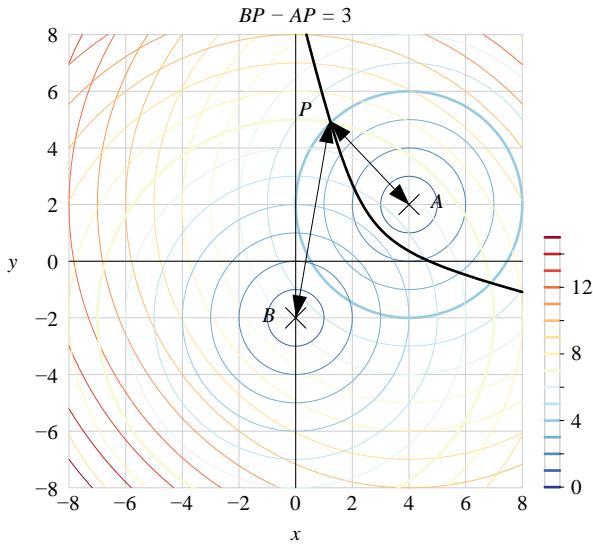
如图 22 所示，我们发现满足 (25) 这种数值关系的 P 构成了一条**双曲线** (hyperbola)。双曲线等圆锥曲线 (conic section) 是本书下两章要介绍的重要内容。

图 22. $AP - BP = 3$ 距离关系构成双曲线左下方一条

将(25)中的3变成-3，也就是说 AP 比 BP 短3，对应如下等式：

$$AP - BP = -3 \quad (26)$$

图23给出的是(26)对应的图像，这时候 P 的轨迹是双曲线右上方那一条。图22和图23构成一对完整的双曲线。

图 23. $BP - AP = 3$ 距离关系构成双曲线右上方一条

正圆

若线段 AP 和 BP 满足倍数关系：

$$AP = c \cdot BP \quad (27)$$

举个例子， AP 是 BP 的两倍：

$$AP = 2BP \quad (28)$$

如图 24 所示，(28) 中 P 轨迹对应的是正圆。有兴趣的读者可以推导这个正圆的解析式。

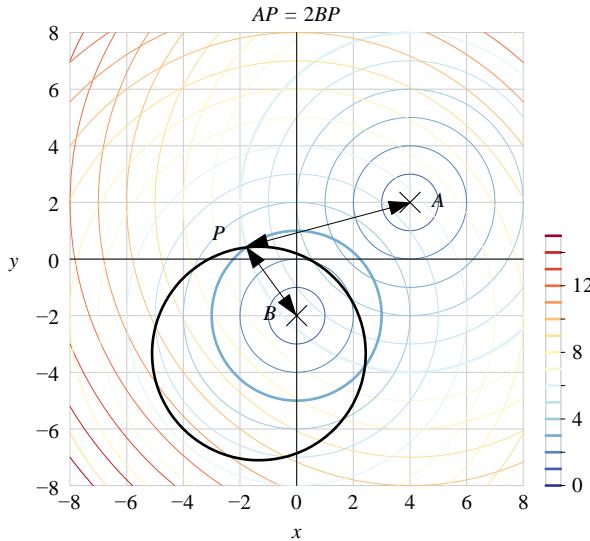


图 24. $AP = 2BP$ 距离关系构成正圆

椭圆

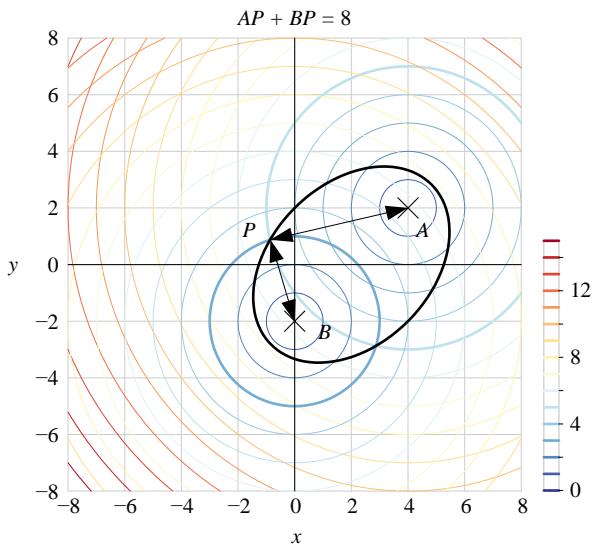
再看一种情况， AP 和 BP 之和为定值，即：

$$AP + BP = c \quad (29)$$

举个例子， AP 和 BP 之和为 8：

$$AP + BP = 8 \quad (30)$$

(30) 中 P 对应的轨迹为一个椭圆 (ellipse)，如图 25 所示。

图 25. $AP + BP = 8$ 距离关系构成椭圆

Bk3_Ch7_06.py 绘制图 21 ~ 图 25。



我们把 Bk3_Ch7_06.py 转化成了展示不同平面形状的 App, 请大家参考代码文件 Streamlit_Bk3_Ch7_06.py。



本章主要介绍了和欧氏距离相关的数学工具, 也特别强调欧氏距离仅仅是众多距离度量之一。为了更好理解其他距离度量的概念和应用场合, 需要大家具备解析几何、线性代数、统计学等知识。随着大家掌握更多数学工具, 本系列丛书将慢慢揭开各种距离度量的面纱, 以及它们在数据科学、机器学习中的重要作用。

Conic Sections

8 圆锥曲线

从解密天体运行，到探索星辰大海



可是，地球确实绕着太阳转。

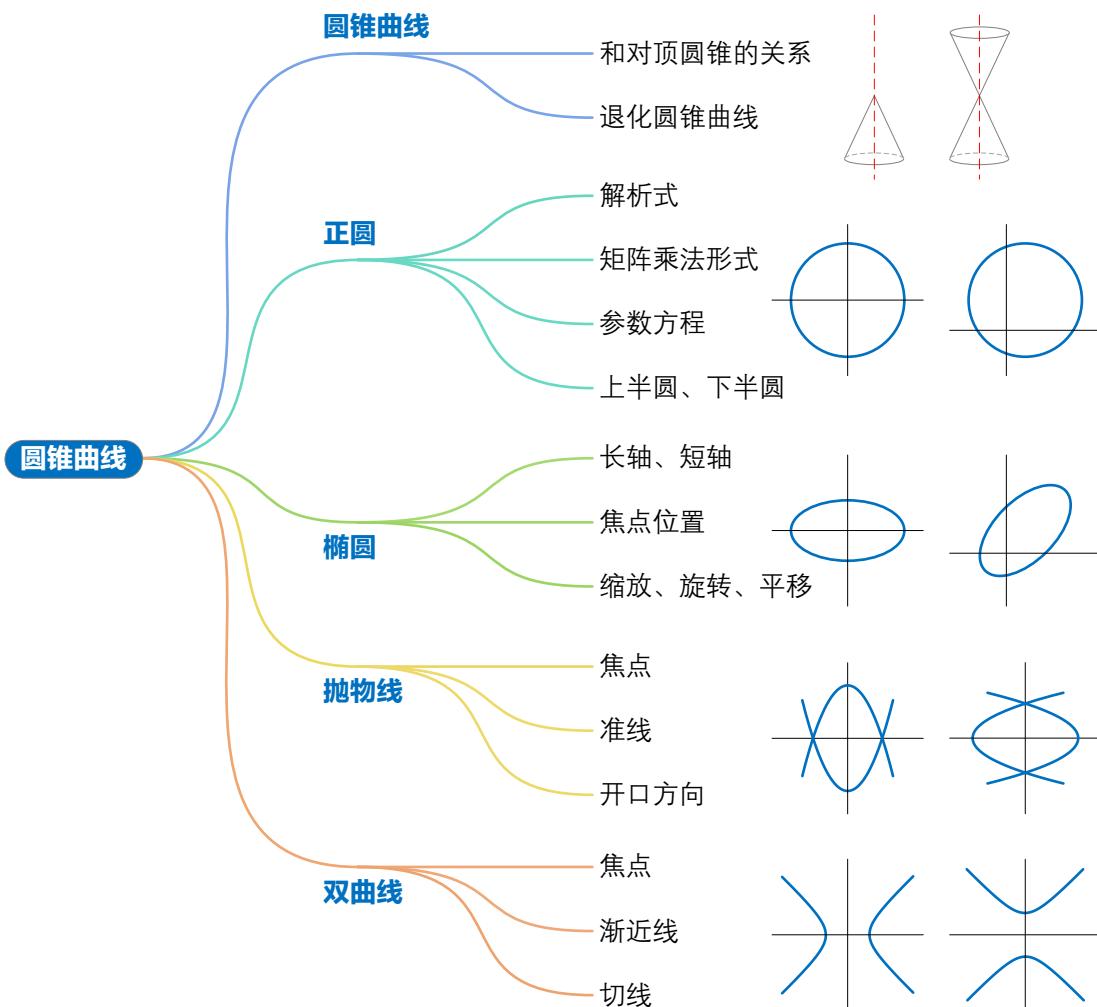
And yet it moves

E pur si muove.

——伽利略·伽利莱 (Galilei Galileo) | 意大利物理学家、数学家及哲学家 | 1564 ~ 1642



- ◀ ax.plot_wireframe() 绘制三维网格图；其中，`ax = fig.add_subplot(projection='3d')`
- ◀ ax.view_init() 设置三维图像观察角度；其中，`ax = fig.add_subplot(projection='3d')`
- ◀ numpy.arange() 根据指定的范围以及设定的步长，生成一个等差数组
- ◀ numpy.cos() 计算余弦
- ◀ numpy.linspace(start, end, num) 生成等差数列，数列 start 和 end 之间（注意，包括 start 和 end 两个数值），数列的元素个数为 num 个
- ◀ numpy.outer(u, v) 当 u 和 v 都为向量时，u 的每一个值代表倍数，使得第二个向量每个值相应倍增。u 决定结果的行数，v 决定结果的列数；当 u 和 v 为多维向量时，按照先行后列展开为向量
- ◀ numpy.sin() 计算正弦
- ◀ sympy.Eq() 定义符号等式
- ◀ sympy.plot(sympy.sin(x)/x, (x, -15, 15), show=True) 绘制符号函数表达式的图像
- ◀ sympy.plot_implicit() 绘制隐函数方程
- ◀ sympy.plot3d(f_xy_diff_x, (x, -2, 2), (y, -2, 2), show=False) 绘制函数的三维图
- ◀ sympy.plotting.plot.plot_parametric() 绘制二维参数方程
- ◀ sympy.plotting.plot.plot3d_parametric_line() 绘制三维参数方程
- ◀ sympy.symbols() 创建符号变量



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

8.1 圆锥曲线外传

自古以来，世界各地的人们在仰望神秘星空时，都会不禁感慨宇宙的浩渺、神秘。

两千三百年前，中国战国时期诗人屈原在《天问》中问到“天何所沓？十二焉分？日月安属？列星安陈？”

在中国古代，“天圆地方”是权威的解释，比如孔子曾说“天道曰圆，地道曰方”。

但是，战国时期的法家创始人之一慎到认为天体是球形，他说“天形如弹丸，半覆地上，半隐地下，其势斜倚”。

东汉张衡无疑推动了中国古代对天体运行规律的认知，他提出“浑天如鸡子，天体圆如弹丸，地如鸡中黄，孤居于内，天大而地小”。

古希腊一众数学家和哲学家对天体运行规律有着相同的探索，其中具有代表性的是**毕达哥拉斯**(Pythagoras)、**亚里士多德**(Aristotle)、**埃拉托斯特尼**(Eratosthenes) 和**托勒密**(Ptolemy)。

古希腊数学家毕达哥拉斯在公元前六世纪，提出地球是球体这一概念。

亚里士多德则以实证的方法得出地球是球形这一结论。比如，他发现月食时，地球投影到月球上的形状为圆形。远航的船舰靠岸时，人们先看到桅杆，再看到船身，最后才能看到整个船身。亚里士多德还发现，越往北走，北极星越高；越往南走，北极星越低。

在地圆说基础上，托勒密建立**地心说**(geocentric model)。地心说被罗马教会奉为圭臬，禁锢了欧洲思想逾千年。

解放人类对天体运行规律认知的数学工具正是圆锥曲线。

古希腊数学家，**梅内克谬斯**(Menaechmus, 380 ~ 320 BC)，开创了圆锥曲线研究。相传，梅内克谬斯是**亚历山大大帝**(Alexander the Great) 的数学老师。亚历山大大帝曾请教梅内克谬斯，想要找到学习几何的终南捷径。梅内克谬斯给出的回答却是“学习几何无捷径”。

抽象的圆锥曲线理论在约 1800 年之后开花结果。这里我们主要介绍四个人物——哥白尼、布鲁诺、开普勒和伽利略。

撼动地心说统治地位的第一人就是波兰天文学家**哥白尼**(Nicolaus Copernicus)。可以说哥白尼革命吹响了现代科学发展的集结号。

1543 年，哥白尼在《天体运行论》(On the Revolutions of the Heavenly Spheres) 提出**日心说**(Heliocentrism)。他认为行星运行轨道为正圆形。细心观察星象后，哥白尼基本上确定地球绕太阳运转，而且每 24 小时完成一周运转。

有趣的是，哥白尼实际上是业余的天文学家，这是典型的“业余”把“专业”干翻在地。



哥白尼 (Nicolaus Copernicus)
波兰数学家、天文学家 | 1473年 ~ 1543年
提出日心说

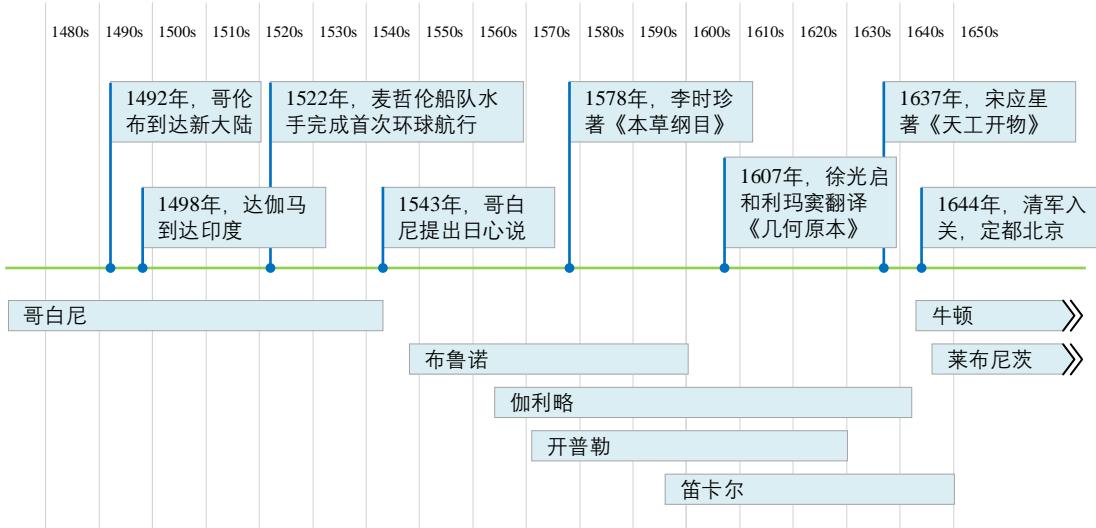


图 1. 哥白尼、布鲁诺、伽利略、开普卡所处的时间轴

这里需要提及的一个人是**布鲁诺** (Giordano Bruno)，他因为对抗教会、传播日心说，被判处长期监禁。1600 年，布鲁诺在罗马鲜花广场被烧死在火刑柱上。

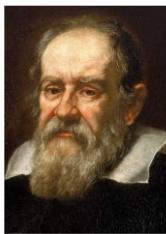
开普勒 (Johannes Kepler) 通过观察和推理提出行星运行轨道为椭圆形，继而提出行星运动三大定理。可以说，圆锥曲线理论是开普勒行星运行研究的核心数学工具。而开普勒的研究对科学技术发展，甚至人类文明进步产生极大的推动力作用。



开普勒 (Johannes Kepler)
德国天文学家、数学家 | 1571年 ~ 1630年
发现行星运行三大定律



伽利略 (Galileo Galilei) 创作的《关于托勒密和哥白尼两大世界体系的对话》(Dialogue Concerning the Two Chief World Systems) 一书中支持哥白尼的理论地球不是宇宙的固定不动的中心。

**伽利略** (Galileo Galilei)意大利天文学家、物理学家和工程师 | 1564年 ~ 1642年
现代物理学之父

因为支持哥白尼日心说，伽利略被罗马宗教裁判所判刑，余生被软禁家中。据传，在被迫放弃日心说主张时，伽利略喃喃自语“可是，地球确实绕着太阳转。”

1979 年，教皇保罗二世代表教廷为伽利略公开平反昭雪，这一道歉迟到了 300 多年。

在伽利略的时代，亚里士多德的世界观处于权威地位。

根据亚里士多德的理论，较重的物体比较轻的物体下降快。在 1800 年时间里，这个观点从未被撼动，直到伽利略爬上比萨斜塔。伽利略将不同重量物体从塔顶抛下，结合之前的斜面试验结果，伽利略提出著名的自由落体定律。

在天文学方面，伽利略首次用望远镜进行天文观测，他发现太阳黑子、月球山系和木星四颗最大的卫星等。

不同于信仰，科学的魅力在于好奇、质疑、实验、推翻、重构，如此往复、迭代上升。

8.2 圆锥曲线：对顶圆锥和截面相交

圆锥、对顶圆锥

顾名思义，**圆锥曲线** (conic section) 和圆锥有直接关系。

图 2 比较**圆锥** (cone) 和**对顶圆锥** (double cone)。圆锥相当于一个直角三角形 (图中蓝色阴影) 以**中轴** (axis) 所在直线旋转得到的形状，直角三角形斜边是圆锥**母线** (generatrix)。白话说，两个全等圆锥，中轴重合、顶对顶安放，便得到对顶圆锥。

反过来，两个全等圆锥，中轴重合、底对底安放，便得到**双锥体** (bicone)。

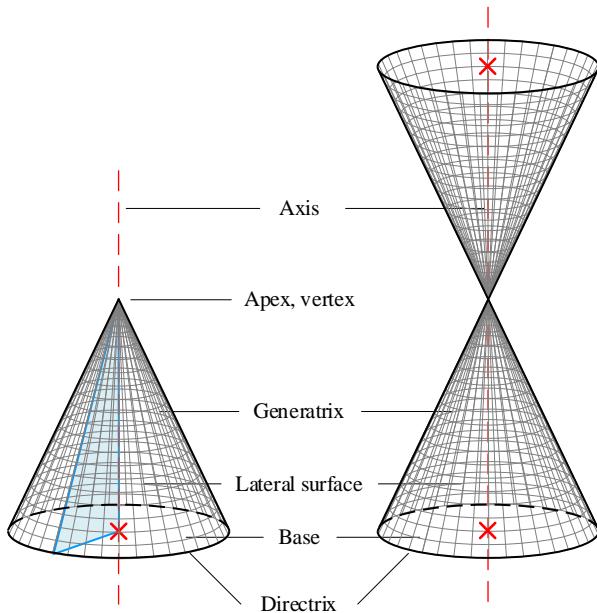


图 2. 圆锥和对顶圆锥

圆锥曲线

圆锥曲线是通过一个对顶圆锥和一个截面 (cutting plane) 相交得到一系列曲线。圆锥曲线主要分为：**正圆** (circle)、**椭圆** (ellipse)、**抛物线** (parabola) 和**双曲线** (hyperbola)。正圆可以视作椭圆的特殊形态。

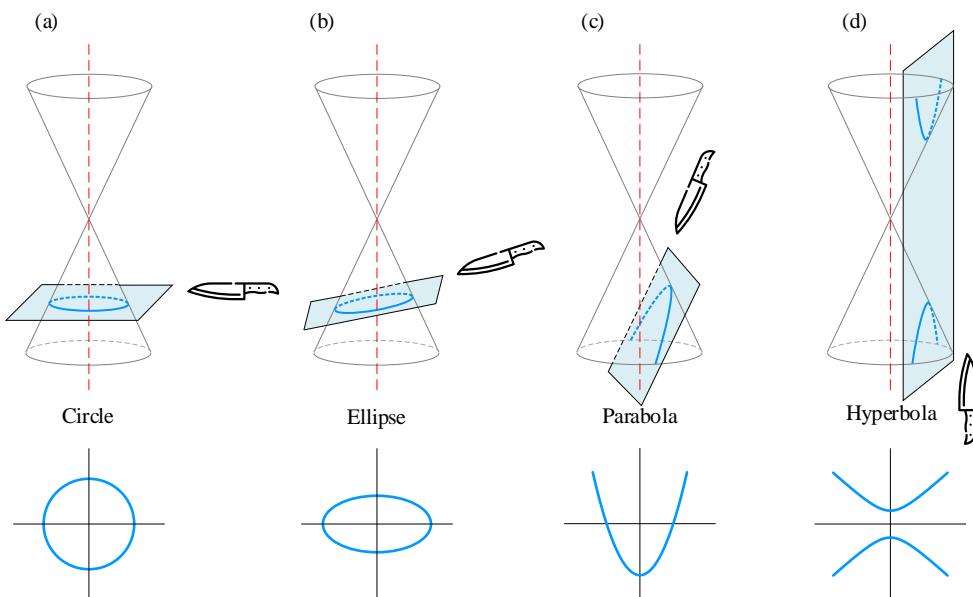


图 3. 四种圆锥曲线

如图 3 (a) 所示，当截面与圆锥中心对称轴垂直，交线为正圆。

当斜面与圆锥相交，交线闭合且不过圆锥顶点，交线为椭圆，如图 3 (b) 所示。

当截面仅与圆锥面一条母线平行，交线仅出现在圆锥面一侧，交线为抛物线，如图 3 (c) 所示。

当截面与两侧圆锥都相交，并且截面不通过圆锥顶点，得到结果是双曲线，如图 3 (d) 所示。

退化圆锥曲线

此外，还有一类圆锥曲线特殊情况——**退化圆锥曲线** (degenerate conic)。

退化双曲线 (degenerate hyperbola) 为两条相交直线，如图 4 (a) 所示。

退化抛物线 (degenerate parabola) 可以是一条直线 (图 4 (b)) 或者两条平行直线。

退化椭圆 (degenerate ellipse) 为一个点，如图 4 (c) 所示。

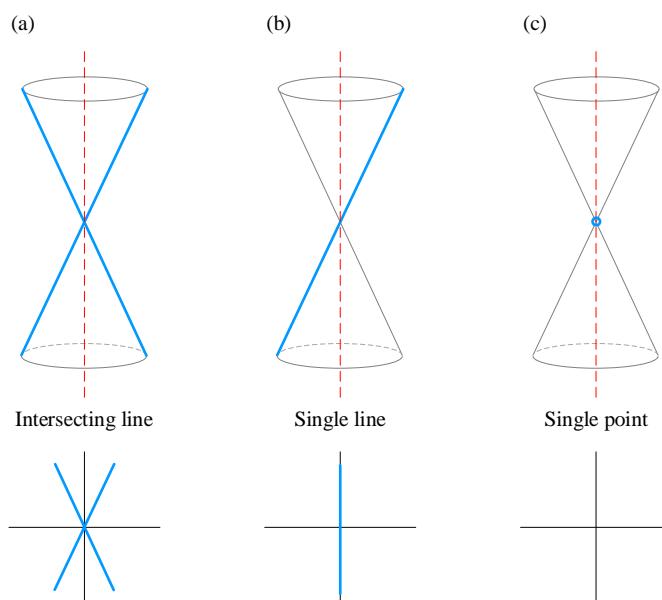


图 4. 三种退化圆锥曲线

8.3 正圆：特殊的椭圆

如图 5 (a) 所示，在 x_1x_2 平面上，圆心位于原点的正圆解析式为：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>
本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>
欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$x_1^2 + x_2^2 = r^2 \quad (1)$$

其中， r 为半径 (radius)。

正圆的周长为 $2\pi r$ ，正圆的面积为 πr^2 。

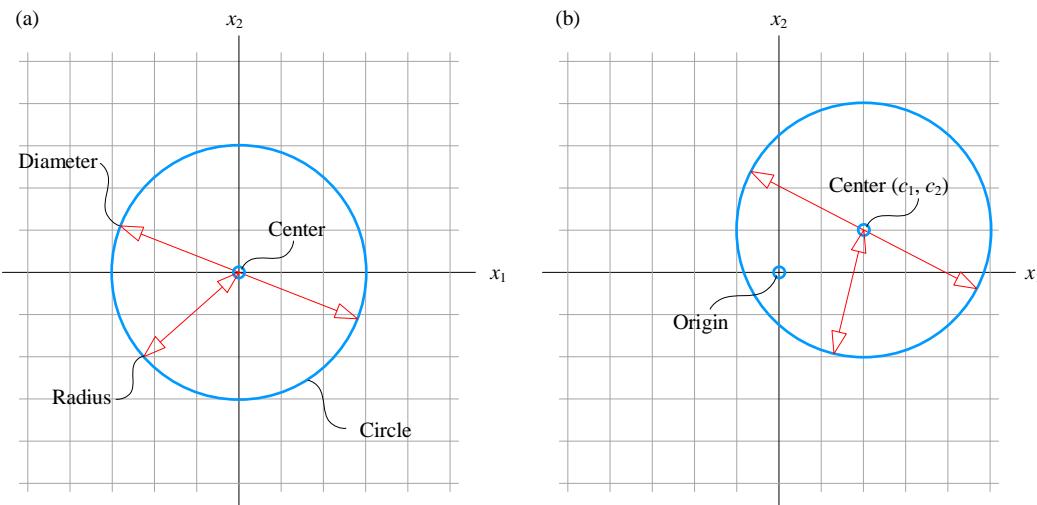


图 5. 正圆

(1) 也可以写成这样的矩阵乘法形式：

$$\mathbf{x}^T \mathbf{x} = r^2 \quad (2)$$

其中，

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (3)$$

(1) 所示解析式对应的参数方程为：

$$\begin{cases} x_1 = r \cos(t) \\ x_2 = r \sin(t) \end{cases} \quad (4)$$

这个正圆的参数方程也可以写成：

$$\begin{cases} x_1 = \frac{1-t^2}{1+t^2} r \\ x_2 = \frac{2t}{1+t^2} r \end{cases} \quad (5)$$

如图 5 (b) 所示，圆心位于 (c_1, c_2) 的正圆解析式为：

$$(x_1 - c_1)^2 + (x_2 - c_2)^2 = r^2 \quad (6)$$

(6) 也可以写成这样的矩阵乘法形式：

$$(\mathbf{x} - \mathbf{c})^T (\mathbf{x} - \mathbf{c}) = r^2 \quad (7)$$

其中，

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \quad (8)$$

(6) 对应的参数方程为：

$$\begin{cases} x_1 = c_1 + r \cos(t) \\ x_2 = c_2 + r \sin(t) \end{cases} \quad (9)$$

上半圆、下半圆

图 5 (a) 所示图像并不是函数，因为一个自变量对应两个因变量的值，这显然不满足函数定义。但是，我们可以用水平线把正圆一切为二，得到上下两个函数，如图 6 所示。

图 6 (a) 所示为**上半圆** (upper semicircle) 函数：

$$f(x_1) = \sqrt{r^2 - x_1^2} \quad (10)$$

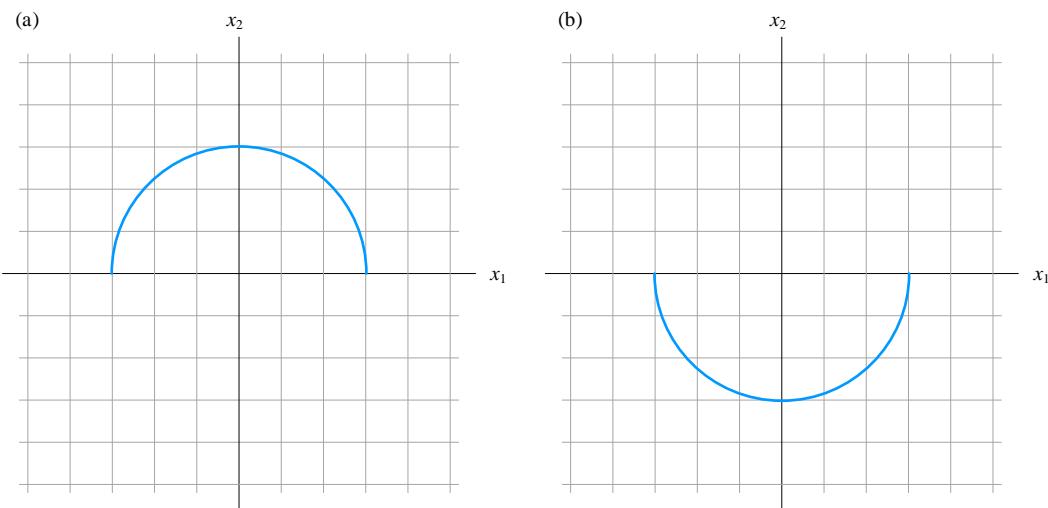


图 6. 上半圆和下半圆函数

图 6 (b) 所示为**下半圆** (lower semicircle) 函数：

$$f(x_1) = -\sqrt{r^2 - x_1^2} \quad (11)$$

表 1. 用英文读正圆解析式

数学表达	英文表达
$x^2 + y^2 = r^2$	x squared plus y squared equals r squared
$y = \pm\sqrt{r^2 - x^2}$	y equals plus or minus square root of the difference of r squared minus x squared
$(x-h)^2 + (y-k)^2 = r^2$	The difference x minus h squared plus the difference y minus k squared equals r squared. The quantity x minus h squared plus the quantity y minus k squared equals r squared.

8.4 椭圆：机器学习的多面手

中心位于原点的正椭圆有两种基本形式，如图 7 所示。

图 7 (a) 所示椭圆的**长轴** (major axis) 位于横轴 x_1 上，对应的解析式为：

$$\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1 \quad (12)$$

其中， $a > b > 0$ 。

长轴，是指通过连接椭圆上的两个点所能获得的最长线段，图 7 (a) 椭圆长轴的长度为 $2a$ 。

与之相反，**短轴** (minor axis) 是通过连接椭圆上的两个点所能获得的最短线段，图 7 (a) 椭圆短轴长度为 $2b$ 。一个椭圆长轴和短轴相互垂直。

长轴的一半被称作**半长轴** (semi-major axis)，短轴的一半被称作**半短轴** (semi-minor axis)。

所谓正椭圆，是指椭圆的长轴位于水平方向或竖直方向。

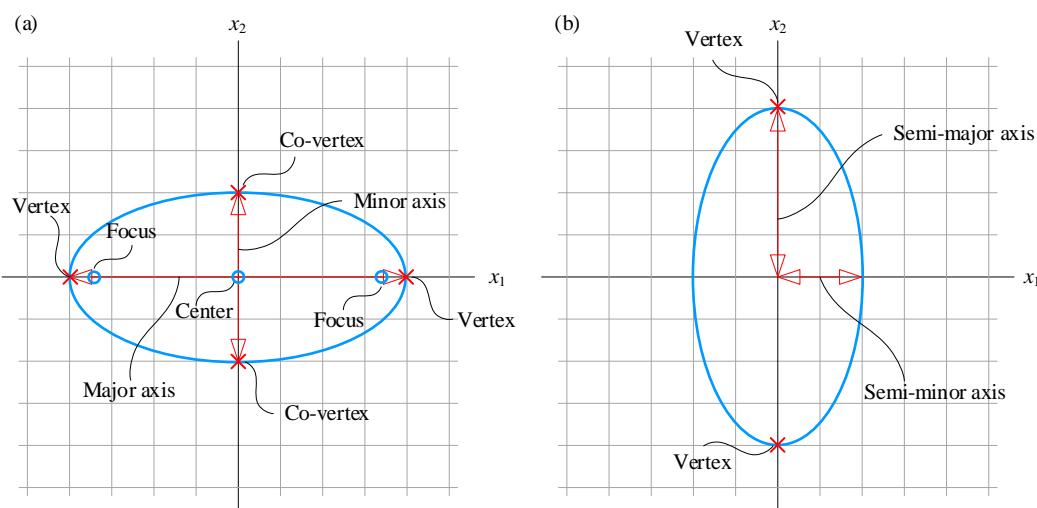


图 7. 中心位于原点的正椭圆

(12) 也可以写成如下矩阵乘法形式：

$$\mathbf{x}^T \begin{bmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{bmatrix} \mathbf{x} = 1 \quad (13)$$

如图 8 所示，椭圆可以看成是正圆朝某一个方向或两个方向缩放得到的结果。这一点从椭圆面积上很容易发现端倪。图 8 所示正圆的面积为 πb^2 ，水平方向拉伸后得到椭圆，对应半长轴为 a ，椭圆面积为 πab 。

在本系列丛书《矩阵力量》一册，大家会知道如下对角方阵对应的几何操作就是“缩放”：

$$\begin{bmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{bmatrix} \quad (14)$$

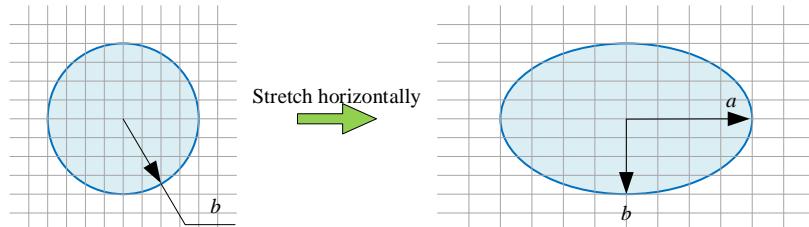


图 8. 椭圆和正圆的关系

(12) 解析式对应的参数方程为：

$$\begin{cases} x_1 = a \cos(t) \\ x_2 = b \sin(t) \end{cases} \quad (15)$$

该参数方程也可以写成：

$$\begin{cases} x_1 = a \frac{1-t^2}{1+t^2} \\ x_2 = b \frac{2t}{1+t^2} \end{cases} \quad (16)$$

表 2. 用英文读椭圆解析式

数学表达	英文表达
$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$	The fraction x squared over a squared plus the fraction y squared over b squared equals one.

图 7 (b) 所示椭圆的长轴位于纵轴 x_2 上，对应的解析式为：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$\frac{x_1^2}{b^2} + \frac{x_2^2}{a^2} = 1 \quad (17)$$

同样 $a > b > 0$ 。

焦点

此外，椭圆的**焦点**（单数 focus，复数 foci）位于椭圆长轴。对于(12)，该椭圆上任意一点 P 到两个焦点 F_1 和 F_2 的距离之和等于 $2a$ 。如图 9 所示，上述关系可以通过下式表达：

$$|PF_1| + |PF_2| = 2a \quad (18)$$

两个焦点 F_1 和 F_2 之间距离被称作为焦距 $2c$ ， c 可以通过下式计算得到：

$$c = \sqrt{a^2 - b^2} \quad (19)$$

由此可以得到图 7 (a) 椭圆两个焦点坐标分别为 $(-c, 0)$ 和 $(c, 0)$ 。图 7 (b) 椭圆两个焦点坐标分别为 $(0, -c)$ 和 $(0, c)$ 。

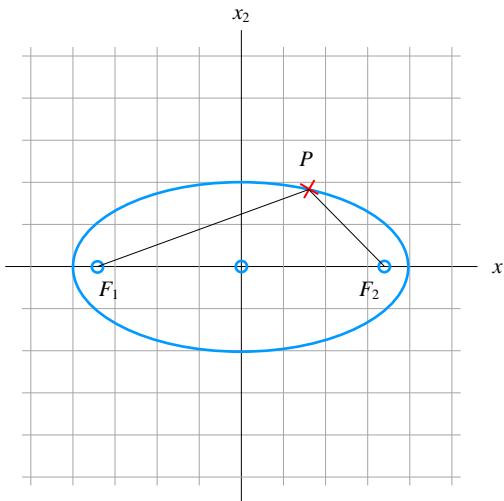


图 9. 椭圆焦点和椭圆的关系

中心移动

图 10 所示为中心在 (c_1, c_2) 的正椭圆。图 10 (a) 所示椭圆的解析式如下：

$$\frac{(x_1 - c_1)^2}{a^2} + \frac{(x_2 - c_2)^2}{b^2} = 1 \quad (20)$$

同样 $a > b > 0$ 。图 10 (b) 所示椭圆的解析式如下：

$$\frac{(x_1 - c_1)^2}{b^2} + \frac{(x_2 - c_2)^2}{a^2} = 1 \quad (21)$$

图 10 中椭圆实际上是图 7 经过平移得到的。

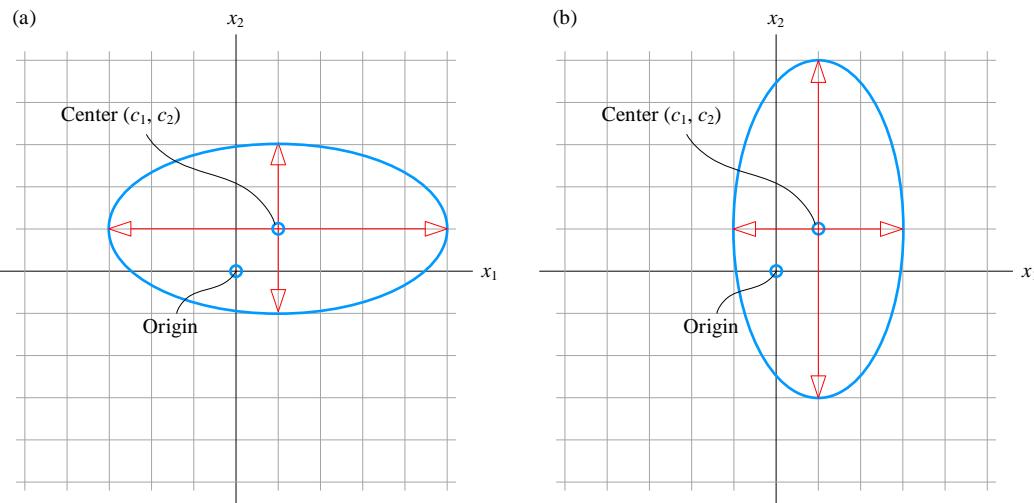


图 10. 中心偏离原点椭圆

8.5 旋转椭圆：几何变换的结果

(12) 椭圆逆时针旋转 θ 后得到椭圆对应的解析式：

$$\frac{[x_1 \cos(\theta) + x_2 \sin(\theta)]^2}{a^2} + \frac{[x_1 \sin(\theta) - x_2 \cos(\theta)]^2}{b^2} = 1 \quad (22)$$

顺时针旋转 θ 后得到椭圆解析式：

$$\frac{[x_1 \cos(\theta) - x_2 \sin(\theta)]^2}{a^2} + \frac{[x_1 \sin(\theta) + x_2 \cos(\theta)]^2}{b^2} = 1 \quad (23)$$

举个例子

中心位于原点，长轴位于横轴的正椭圆在旋转之前解析式为：

$$\frac{x_1^2}{4} + x_2^2 = 1 \quad (24)$$

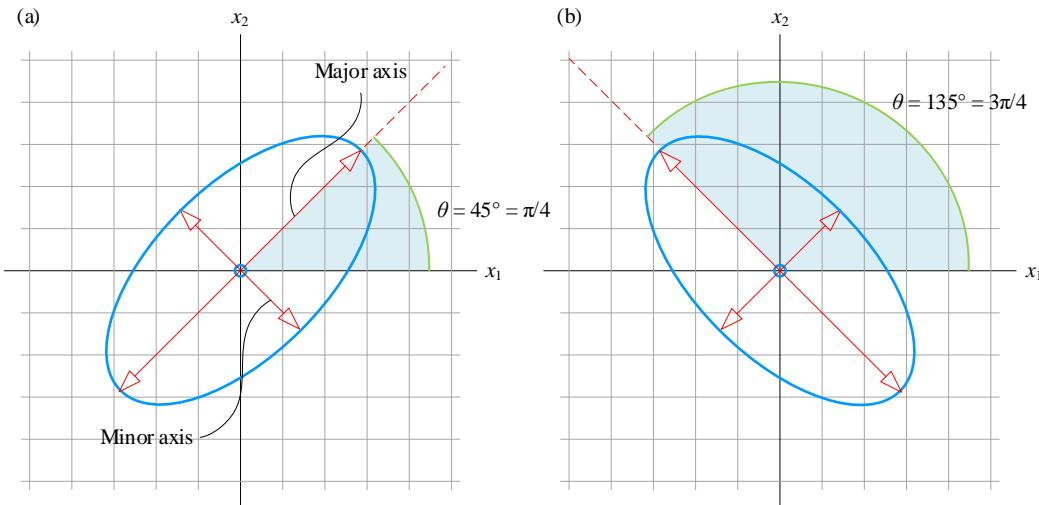


图 11. 旋转椭圆

图 11 (a) 所示为绕中心(原点)逆时针旋转 $\theta = 45^\circ = \pi/4$ 获得的椭圆，长轴位于第一和第三象限。对应解析式如下：

$$\frac{[x_1 \cos(45^\circ) + x_2 \sin(45^\circ)]^2}{4} + [x_1 \sin(45^\circ) - x_2 \cos(45^\circ)]^2 = 1 \quad (25)$$

整理解析式得到：

$$\frac{5x_1^2}{8} - \frac{3x_1x_2}{4} + \frac{5x_2^2}{8} = 1 \quad (26)$$

图 11 (b) 所示为绕中心(原点)逆时针旋转 $\theta = 135^\circ = 3\pi/4$ 获得的椭圆，它的长轴位于第二和第四象限。对应解析式如下：

$$\frac{[x_1 \cos(135^\circ) + x_2 \sin(135^\circ)]^2}{4} + [x_1 \sin(135^\circ) - x_2 \cos(135^\circ)]^2 = 1 \quad (27)$$

整理解析式得到：

$$\frac{5x_1^2}{8} + \frac{3x_1x_2}{4} + \frac{5x_2^2}{8} = 1 \quad (28)$$

对比 (26) 和 (28)，发现解析式仅仅差在 $3x_1x_2/4$ 项的正负号上。

单位圆到椭圆

平面上，对单位圆进行一系列几何变换操作，可以得到中心位于任意一点的旋转椭圆。

如图 12 所示，单位圆(蓝色)，首先经过缩放得到长轴位于横轴的椭圆(绿色)，绕中心旋转之后得到旋转椭圆(橙黄)，最后中心平移得到红色椭圆。

→ 多提一嘴，椭圆的缩放、旋转、平移等操作，和本系列丛书后续线性代数中仿射变换直接相关，请大家格外留意。

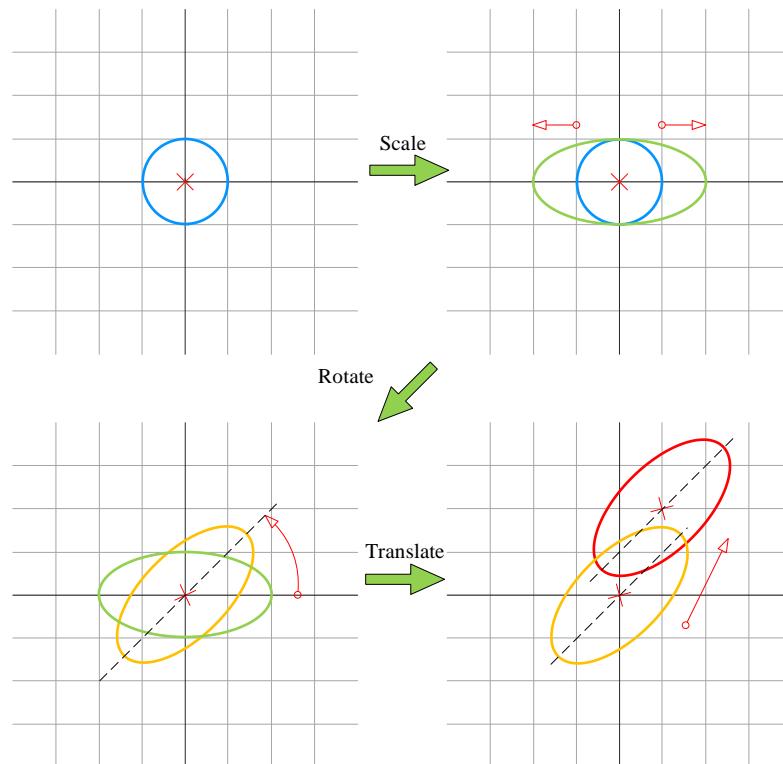
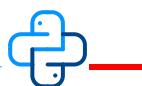


图 12. 正圆经过缩放、旋转和平移得到椭圆



Bk3_Ch8_01.py 绘制平移和旋转椭圆。



椭圆，这个高中阶段学过的数学概念，和数据科学、机器学习有什么关系？

看似平淡无奇的椭圆，其实与数据科学和机器学习有着特别密切的关系。这里，我们蜻蜓点水地聊一下。

图 13 所示为不同相关性系数条件下，二元高斯分布 PDF 曲面和等高线。在平面等高线的图像中，我们看到的是一系列同心椭圆。

不同相关性系数条件下，满足特定二元高斯分布的随机数可以看到椭圆的影子，具体如图 14 所示。

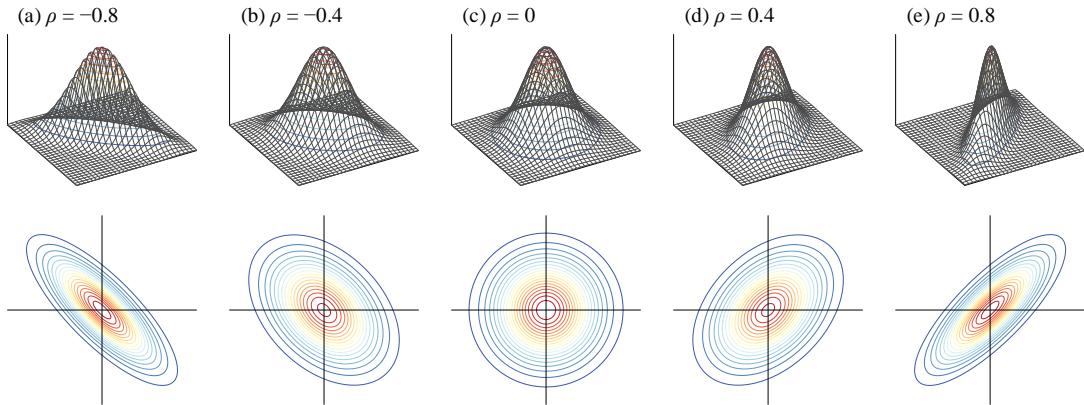
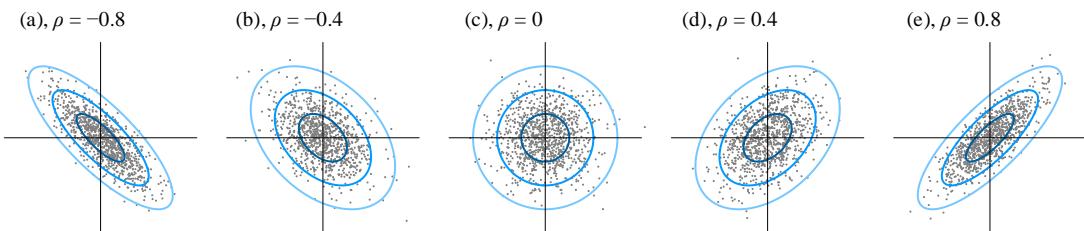


图 13. 不同相关性系数，二元高斯分布 PDF 曲面和等高线

图 14. 相关系数 ρ 不同时，散点和椭圆关系

上一章简单提到马氏距离这个距离度量。如图 15 所示，马氏距离计算过程就是椭圆几何变换过程。

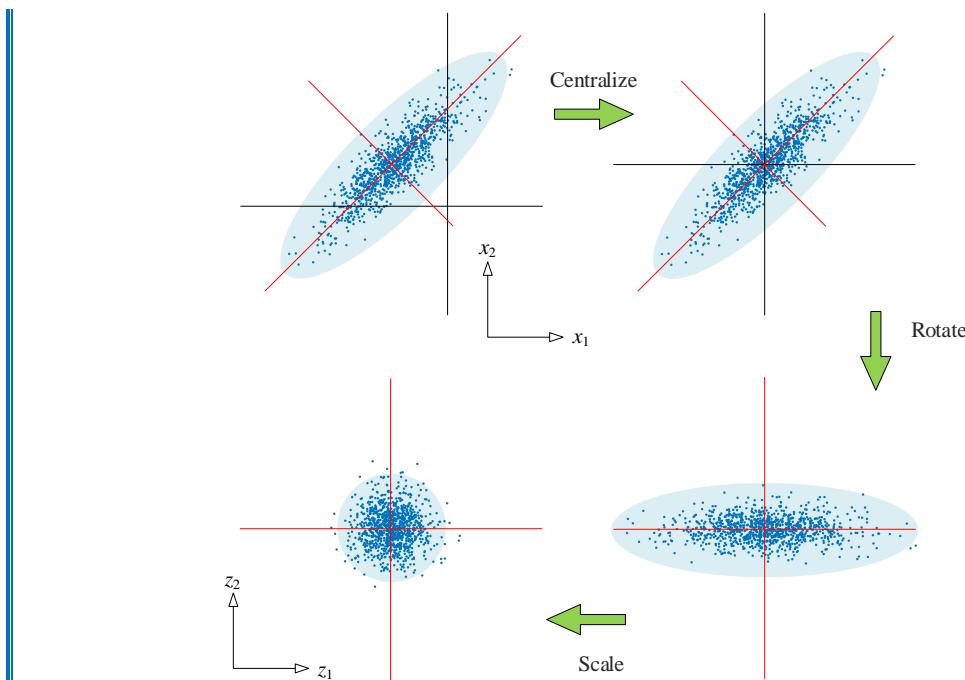


图 15. 马氏距离计算过程

本书前文提到的一元线性回归和椭圆也息息相关。从概率角度，线性回归的本质就是条件概率。图 16 所示为条件概率曲面等高线，黑色斜线代表线性回归。我们也能看到椭圆身在其中。

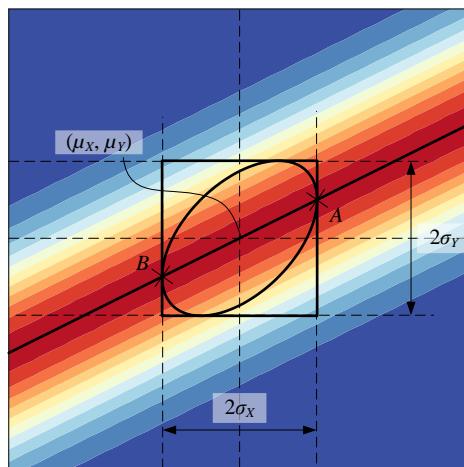


图 16. 条件概率曲面等高线

主成分分析 (Principal Component Analysis, PCA) 是机器学习中重要的降维算法。从几何角度来看，如图 17 所示，主成分分析实际上就是在寻找椭圆的长轴。

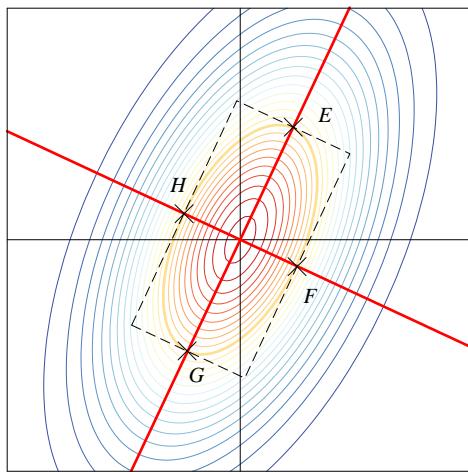


图 17. 主成分分析

在一些分类和聚类算法确定的决策边界中，我们也可以看到椭圆及其他圆锥曲线。图 18 所示为高斯朴素贝叶斯分类计算得到的决策边界。图 19 所示为高斯混合模型确定的决策边界。

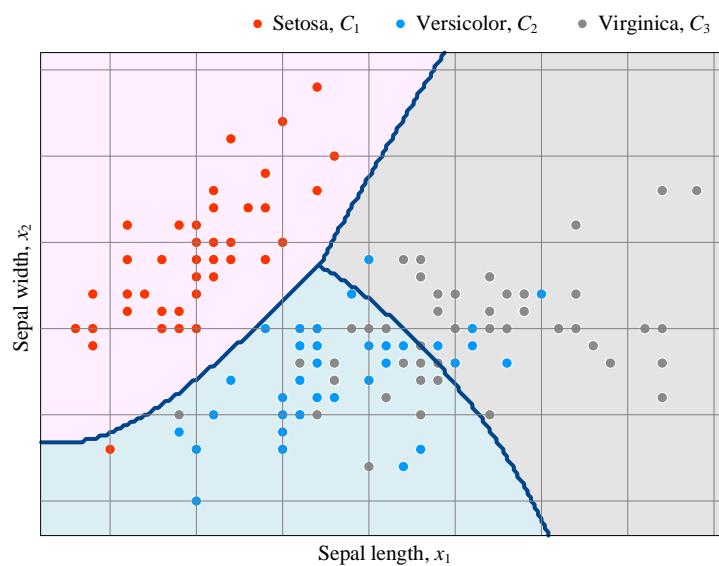


图 18. 高斯朴素贝叶斯分类确定的决策边界

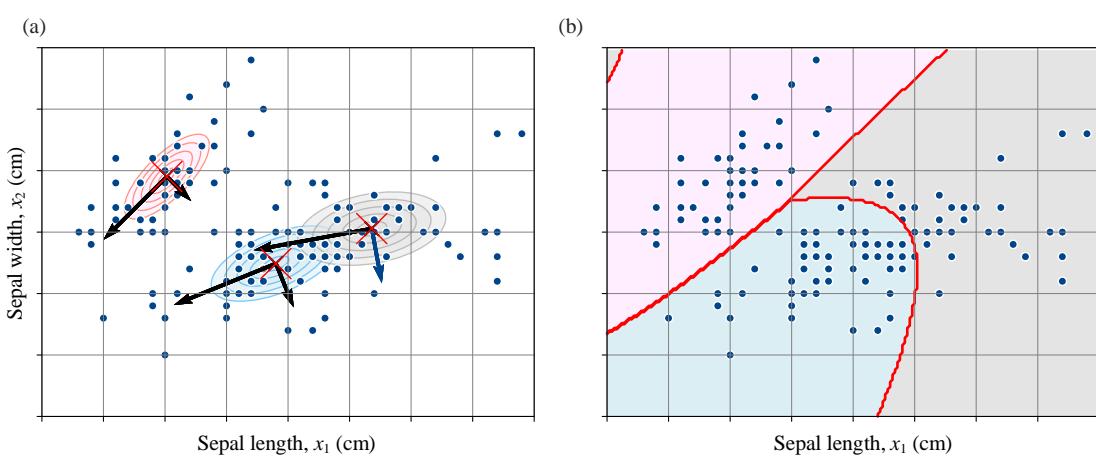


图 19. 高斯混合模型确定的决策边界

总而言之，椭圆在数据科学和机器学习这两个话题中有很多“戏份”，本系列丛书将为大家一一讲述这些有关椭圆的故事。因此，也请大家耐心学完本章和下一章内容。

8.6 抛物线：不止是函数

如图 20 (a) 所示，顶点在原点，对称轴位于 x_2 纵轴，开口向上抛物线解析式如下：

$$4px_2 = x_1^2, \quad p > 0 \quad (29)$$

这条抛物线顶点位于原点 $(0, 0)$ ，**焦点** (focus) 位于 $(0, p)$ ，**准线** (directrix) 位于 $y = -p$ 。当 p 小于 0 时，形如 (29) 抛物线开口朝下。

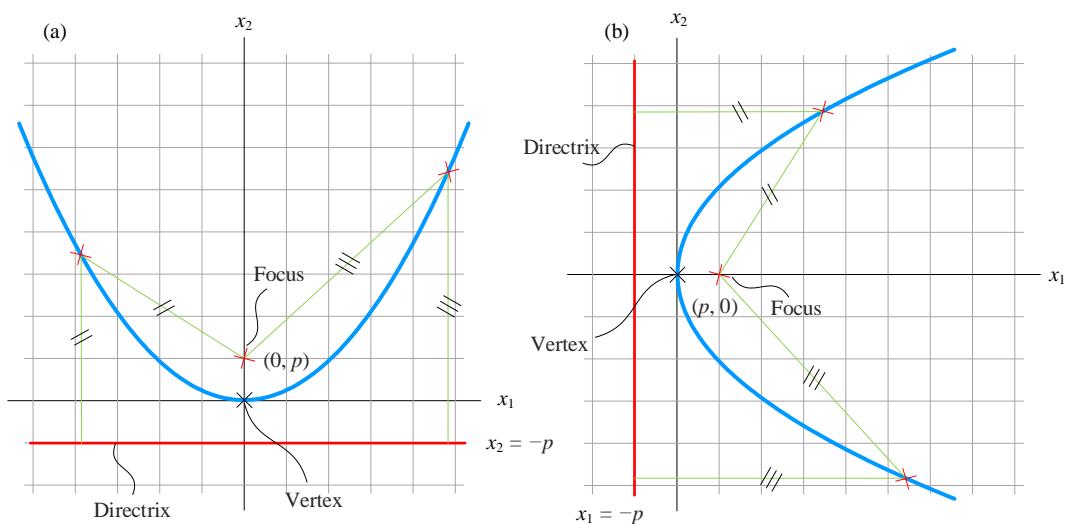


图 20. 抛物线

平面上，抛物线上每一点与焦点之间距离等于点和准线之间距离。这一规律可以用来推导得到抛物线解析式。设抛物线任意一点为 (x_1, x_2) ，该点与准线间距离，等于该点和焦点间距离，如下式：

$$\begin{aligned} x_2 - (-p) &= \sqrt{(x_1 - 0)^2 + (x_2 - p)^2}, \quad p > 0 \\ \Rightarrow (x_2 + p)^2 &= x_1^2 + (x_2 - p)^2 \\ \Rightarrow 4px_2 &= x_1^2 \end{aligned} \tag{30}$$

图 20 (b) 所示，顶点在原点，对称轴位于 x_1 横轴，开口向右抛物线解析式如下：

$$4px_1 = x_2^2, \quad p > 0 \tag{31}$$

当 p 小于 0 时，形如 (31) 抛物线开口朝左。

平移

同样，抛物线也可以整体平移。形如下式的抛物线，顶点位于 (h, k) ，开口朝上或朝下，具体方向由 p 正负决定：

$$4p(x_2 - k) = (x_1 - h)^2 \tag{32}$$

(32) 所示抛物线对称轴为 $x_1 = h$ ，准线位于 $x_2 = k - p$ ，焦点所在位置为 $(h, k + p)$ 。

如下抛物线，顶点同样位于 (h, k) ， p 正负决定开口朝左或朝右：

$$4p(x_1 - h) = (x_2 - k)^2 \tag{33}$$

(33) 所示抛物线对称轴为 $x_2 = k$ ，准线位于 $x_1 = h - p$ ，焦点所在位置为 $(h + p, k)$ 。

图 21 所示为四种抛物线，请大家参考前文代码自行绘制图 21。

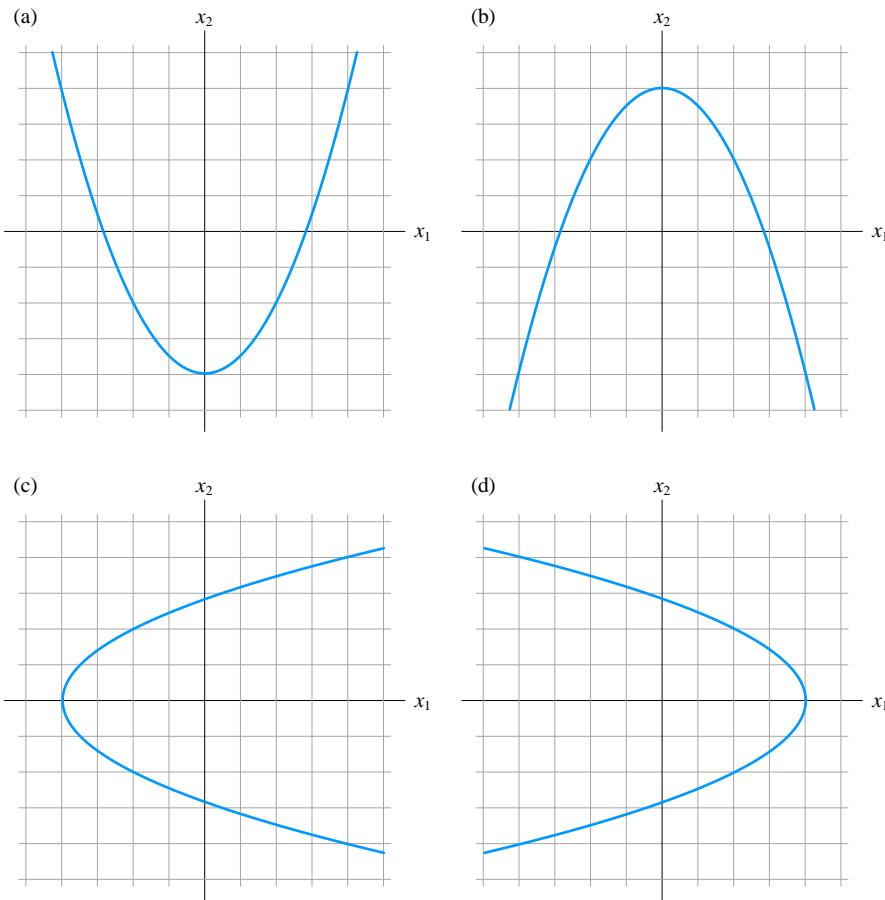


图 21. 四种抛物线

8.7 双曲线：引力弹弓的轨迹

图 22 (a) 所示为，焦点位于横轴，顶点位于原点双曲线，对应解析式形式为：

$$\frac{x_1^2}{a^2} - \frac{x_2^2}{b^2} = 1 \quad a, b > 0 \quad (34)$$

如图 22 (a) 所示，双曲线顶点位于原点，焦点位于 $F_1(-c, 0)$ 和 $F_2(c, 0)$ ， c 可以通过下式计算得到：

$$c^2 = a^2 + b^2, \quad c > 0 \quad (35)$$

双曲线上任意一点 P 到两个焦点 F_1 和 F_2 距离差值为 $2a$ ：

$$|PF_1| - |PF_2| = \pm 2a \quad (36)$$

图 22 (b) 所示为焦点位于纵轴双曲线，上下开口。这种双曲线标准式如下：

$$\frac{x_2^2}{b^2} - \frac{x_1^2}{a^2} = 1 \quad a, b > 0 \quad (37)$$

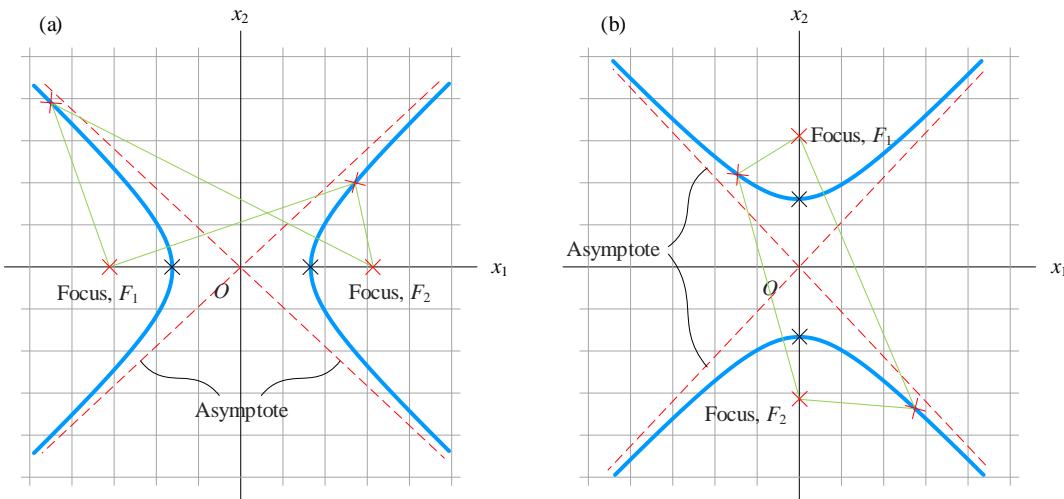


图 22. 两种标准双曲线形式

渐近线、切线斜率

图 22 (a) 所示双曲线的两条渐近线 (asymptote) 表达式如下：

$$x_2 = \pm \frac{b}{a} x_1 \quad (38)$$

图 23 所示为左右开口双曲线右侧部分不同切点处若干条渐变切线。容易发现，图 23 切线斜率，要么大于 b/a ，要么小于 $-b/a$ 。也就是说，切线在 $(-\infty, -b/a)$ 和 $(b/a, +\infty)$ 两个区间之内。

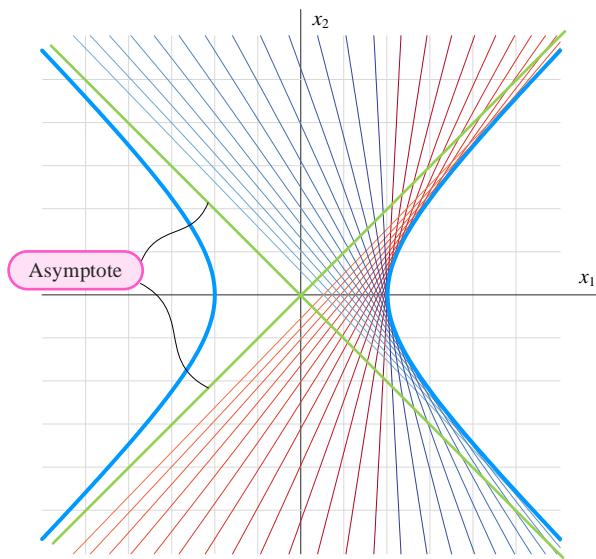


图 23. 双曲线右侧曲线切线

平移

双曲线中心也可以平移，比如将(34)所示双曲线中心平移至 (h, k) ，得到的双曲线解析式如下：

$$\frac{(x_1-h)^2}{a^2} - \frac{(x_2-k)^2}{b^2} = 1 \quad a, b > 0 \quad (39)$$

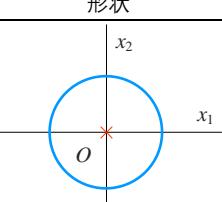
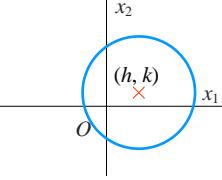
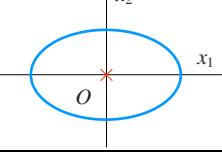
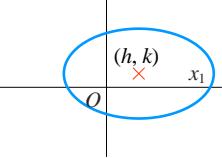
(39) 对应双曲线的两条渐近线解析式如下：

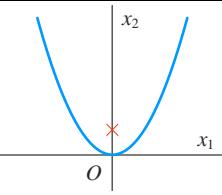
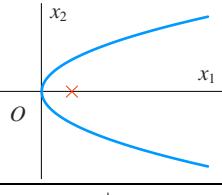
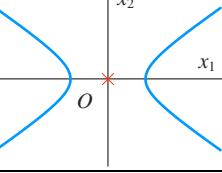
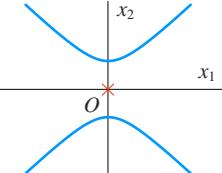
$$x_2 = k \pm \frac{b}{a}(x_1 - h) \quad (40)$$

圆锥曲线参数方程

表 3 总结常见圆锥曲线参数方程。

表 3. 常见圆锥曲线参数方程

形状	一般式	参数方程
	圆心在原点，半径为 r 圆形 $x_1^2 + x_2^2 = r^2$	$\begin{cases} x_1 = r \cos(t) \\ x_2 = r \sin(t) \end{cases}$ 或 $\begin{cases} x_1 = \frac{1-t^2}{1+t^2} \\ x_2 = \frac{2t}{1+t^2} \end{cases}$
	圆心在 (h, k) ，半径为 r 圆形 $(x_1 - h)^2 + (x_2 - k)^2 = r^2$	$\begin{cases} x_1 = h + r \cos(t) \\ x_2 = k + r \sin(t) \end{cases}$
	椭圆中心在原点，长轴和焦点位于横轴，半长轴为 a ，半短轴为 b $\frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = 1$	$\begin{cases} x_1 = a \cos(t) \\ x_2 = b \sin(t) \end{cases}$ 或 $\begin{cases} x_1 = a \frac{1-t^2}{1+t^2} \\ x_2 = b \frac{2t}{1+t^2} \end{cases}$
	椭圆，中心在 (h, k) ，半长轴为 a ，半短轴为 b $\frac{(x_1 - h)^2}{a^2} + \frac{(x_2 - k)^2}{b^2} = 1$	$\begin{cases} x_1 = h + a \cos(t) \\ x_2 = k + b \sin(t) \end{cases}$

	抛物线，焦点位于纵轴 $4px_2 = x_1^2, p > 0$	$\begin{cases} x_1 = t \\ x_2 = \frac{t^2}{4p} \end{cases}$
	抛物线，焦点位于横轴 $4px_1 = x_2^2, p > 0$	$\begin{cases} x_1 = \frac{t^2}{4p} \\ x_2 = t \end{cases}$
	双曲线，焦点位于横轴 $\frac{x_1^2}{a^2} - \frac{x_2^2}{b^2} = 1$	$\begin{cases} x_1 = a \sec(t) \\ x_2 = b \tan(t) \end{cases} \text{ 或 } \begin{cases} x_1 = a \frac{1+t^2}{1-t^2} \\ x_2 = b \frac{2t}{1-t^2} \end{cases}$
	双曲线，焦点位于纵轴 $\frac{x_2^2}{b^2} - \frac{x_1^2}{a^2} = 1$	$\begin{cases} x_1 = a \tan(t) \\ x_2 = b \sec(t) \end{cases} \text{ 或 } \begin{cases} x_1 = a \frac{2t}{1-t^2} \\ x_2 = b \frac{1+t^2}{1-t^2} \end{cases}$



科技进步发展从来不是正道坦途、一帆风顺，这条道路蜿蜒曲折、荆棘密布，很多人甚至为之付出生命。

即便如此，科学思想一旦被提出，就像是一颗种子种在了人类思想的土壤中。这些种子们早晚会生根发芽，开花结果。圆锥曲线就是个很好的例子。

圆锥曲线提出千年以后，人们利用这个数学工具解密了天体运行规律。此后几百年，圆锥曲线就会助力人类飞出地球摇篮，探索无边的深空。

9

Dive into Conic Sections

深入圆锥曲线

探寻和数据科学、机器学习之间联系



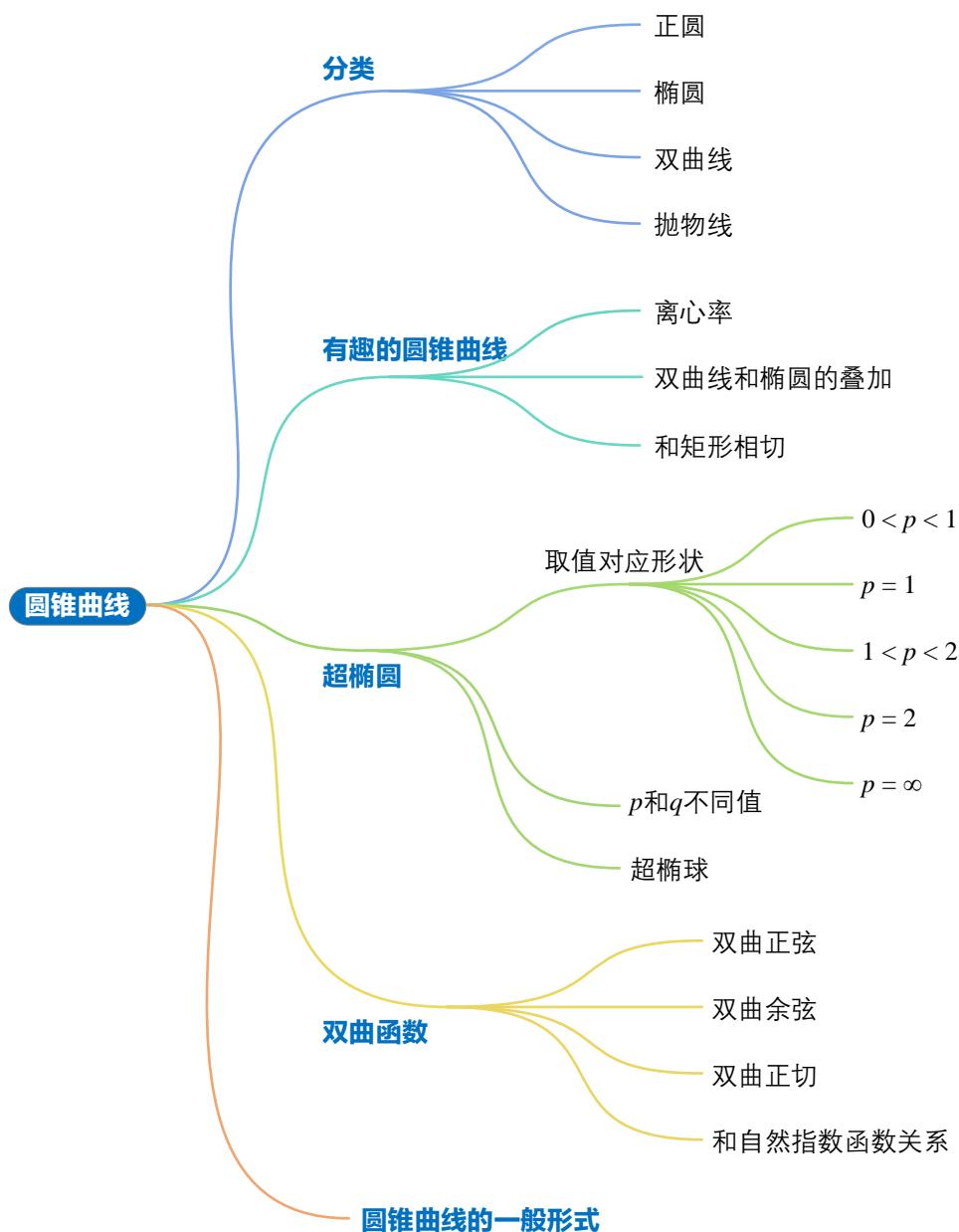
地球是人类的摇篮，但我们不能永远生活在摇篮里。

Earth is the cradle of humanity, but one cannot live in a cradle forever.

——康斯坦丁·齐奥尔科夫斯基 (Konstantin Tsiolkovsky) | 俄罗斯火箭专家 | 1857 ~ 1935



- ◀ `matplotlib.patches.Rectangle()` 绘制通过定位点，以及设定宽度和高度的矩形
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `numpy.cosh()` 双曲余弦函数
- ◀ `numpy.isinf()` 判断是否存在无穷
- ◀ `numpy.maximum()` 计算最大值
- ◀ `numpy.sinh()` 双曲正弦函数
- ◀ `numpy.tanh()` 双曲正切函数
- ◀ `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.plot_implicit()` 绘制隐函数方程
- ◀ `sympy.symbols()` 定义符号变量



9.1 圆锥曲线：探索星辰大海

虽然正圆、椭圆、抛物线、双曲线这样的数学概念现在见诸于中学课本，但是它们现如今依旧展现着巨大能量。比如，在星辰大海的征途中，圆锥曲线扮演重要角色。

图 1 所示为四种航天器轨道。当航天器以**第一宇宙速度** (first cosmic velocity) 绕地运行时，运行的轨道为**正圆轨道** (circular orbit)，第一宇宙速度因此被称作**环绕速度** (orbit speed)。提高航天器绕行速度，轨道变为**椭圆轨道** (elliptical orbit)。

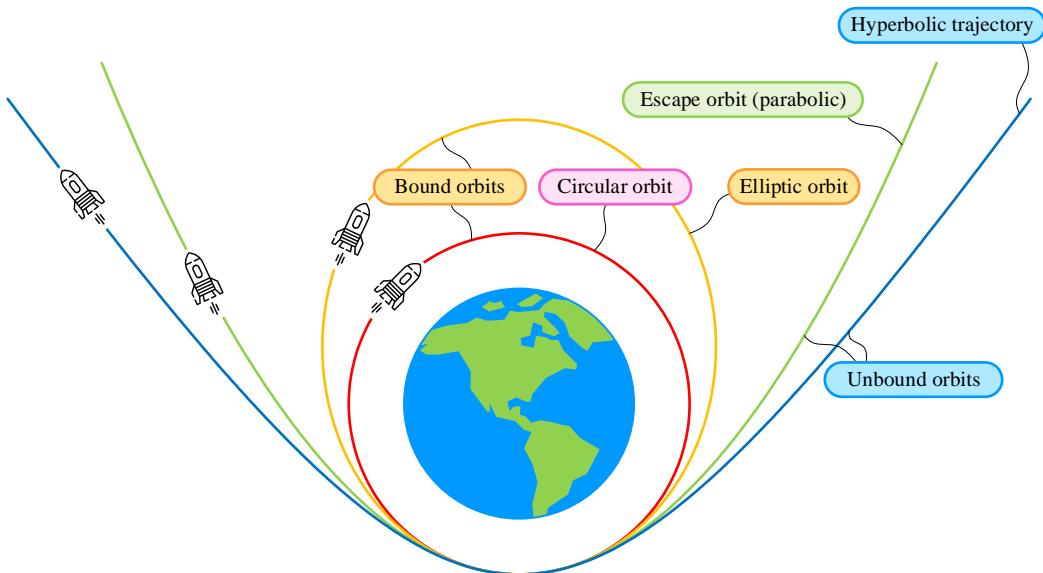


图 1. 航天器的几种轨道

继续提高绕行速度，当航天器速度达到**第二宇宙速度** (second cosmic velocity) 时，航天器便达到逃离地球所需速度，这一速度也叫**逃逸速度** (escape velocity)。这时，航天器运行轨道变为**抛物线轨道** (parabolic trajectory) 或**双曲线轨道** (hyperbolic trajectory)。这种条件下，航天器可以脱离地球的引力场而成为围绕太阳运行的人造行星。

探索火星约每 26 个月有一个发射窗口，这是因为地球在低轨道绕太阳运行，而火星在高轨道绕行。地球和火星的公转周期不同，两个行星大约每 26 个月“相遇”一次，也就是说地球与火星之间的距离最近。

如图 2 所示，探索火星需要利用**霍曼转移轨道** (Hohmann transfer orbit)。简单来说，霍曼轨道是一条椭圆形的轨道，通过两次加速将航天器从地球所在的低轨道送入火星运动的高轨道。

航天器首先进入绕太阳圆周运动的低轨道。

太空船在低轨道 A 点处上瞬间加速后，进入一个椭圆形的转移轨道。注意，加速瞬间火星位于 B。太空船由此椭圆轨道的近拱点开始，抵达远拱点后再瞬间加速，进入火星所在的目标轨道。反过来，霍曼转移轨道亦可将太空船送往较低的轨道，不过是两次减速而非加速。

拱点 (apsis) 在天文学中是指椭圆轨道上运行天体（比如地球）最接近或最远离它的引力中心（比如太阳）的点。最靠近引力中心的点称为**近拱点 (periapsis)**；而距离引力中心最远的点就称为**远拱点 (apoapsis)**。

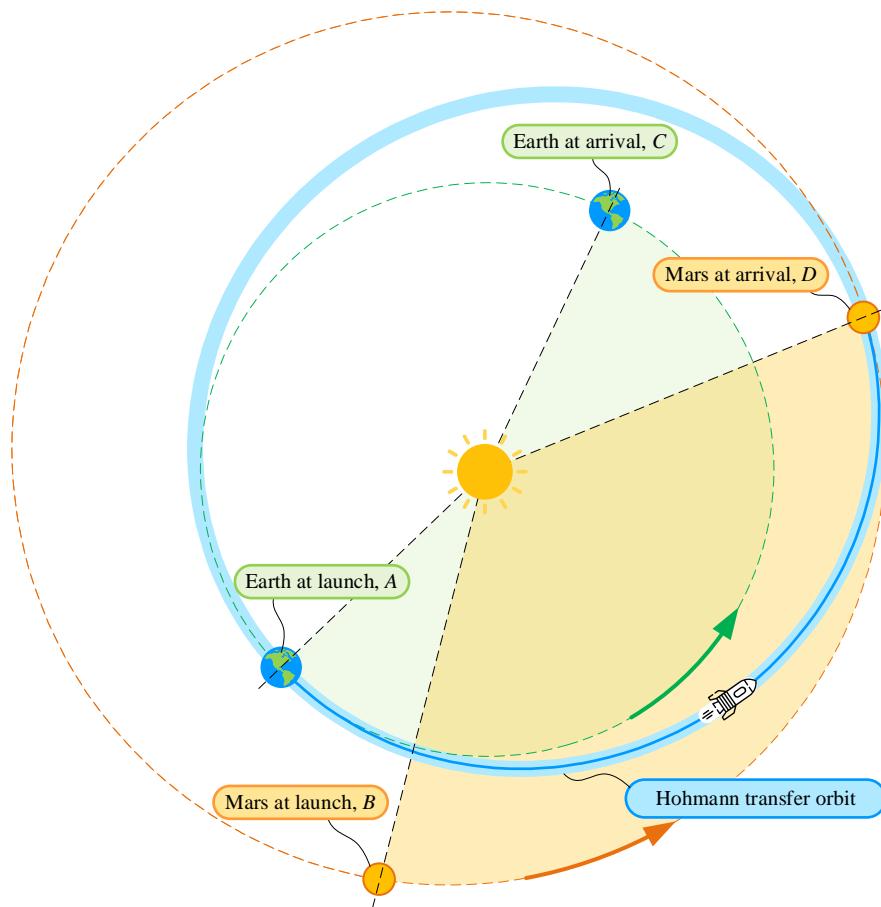


图 2. 探索火星的霍曼轨道

9.2 离心率：联系不同类型圆锥曲线

不同类型圆锥曲线可以通过同**离心率 (eccentricity) e** 联系起来：

$$x_2^2 = 2px_1 + (e^2 - 1)x_1^2, \quad e \geq 0 \quad (1)$$

正圆的离心率 $e = 0$, 椭圆的离心率 $0 < e < 1$, 抛物线离心率 $e = 1$, 双曲线离心率 $e > 1$ 。(1) 对应的这一组曲线共用 $(0, 0)$ 这个顶点。

当 $p = 1$ 时, 离心率 e 取不同数值, 可以得到如图 3 所示一组圆锥曲线。

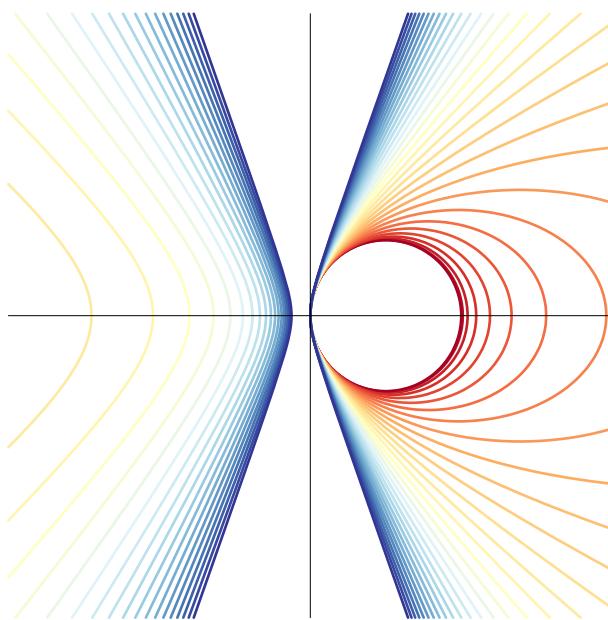


图 3. 离心率连续变化条件下一组圆锥曲线



Bk3_Ch9_01.py 绘制图 3。代码采用等高线方式可视化圆锥曲线。本书之后的圆锥曲线都会再用这种可视化方案。

9.3 一组有趣的圆锥曲线

本节介绍一组有趣的圆锥曲线, 解析式如下:

$$\underbrace{\frac{x_1^2}{m^2} + \frac{x_2^2}{n^2}}_{\text{Ellipse}} - 2\rho \underbrace{\frac{x_1 x_2}{mn}}_{\text{Hyperbola}} = 1 \quad (2)$$

其中, $m > 0$, $n > 0$ 。

上式可以看做是椭圆和双曲线的“叠加”。 $x_1 x_2 = 1$ 实际上是一个旋转双曲线。参数 ρ 可以视作调节双曲线“影响力”的参数, ρ 越大双曲线的影响越强。

点 $(\pm m, 0)$ 、 $(0, \pm n)$ 都满足(2)，也就是说这四个点都在圆锥曲线上。

图4所示为当 $m = n = 1$ 时，且 $\rho \geq 0$ ，圆锥曲线随 ρ 变化。而图5所示为当 $m = n = 1$ 时，且 $\rho \leq 0$ ，圆锥曲线随 ρ 变化。不难发现， $-1 < \rho < 1$ 时，椭圆的影响力占上风。而 $|\rho| > 1$ ，双曲线影响力更大。当 $\rho = \pm 1$ 时，椭圆和双曲线势均力敌。

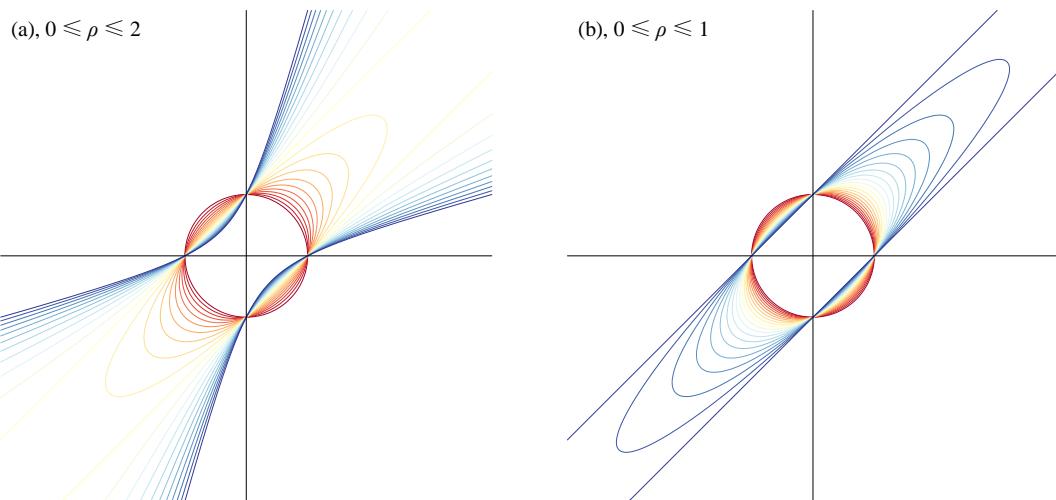


图4. $m = n = 1$, 圆锥曲线随 ρ 变化, ρ 非负

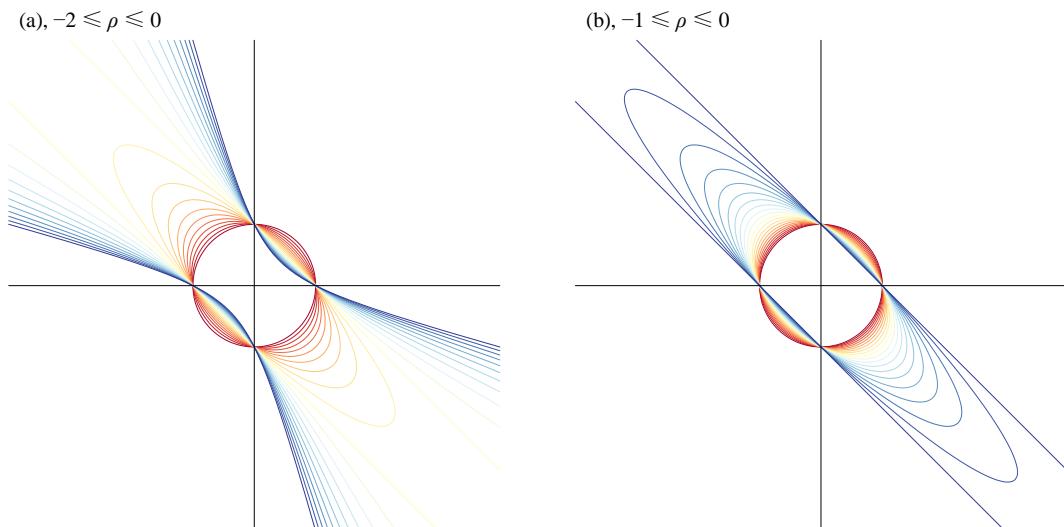


图5. $m = n = 1$, 圆锥曲线随 ρ 变化, ρ 非正

当 $m = n = 1$ 时，且 $\rho = 1$ 时，(2) 为：

$$(x - y)^2 = 1 \quad (3)$$

以上解析式对应两条直线：

$$x - y = 1, \quad x - y = -1 \quad (4)$$

当 $m = n = 1$ 时，且 $\rho = -1$ 时，(2) 也对应两条直线。

图 6 所示为 $m = 2, n = 1$ ，圆锥曲线随 ρ 变化， ρ 的变化范围为 $[-2, 2]$ 。

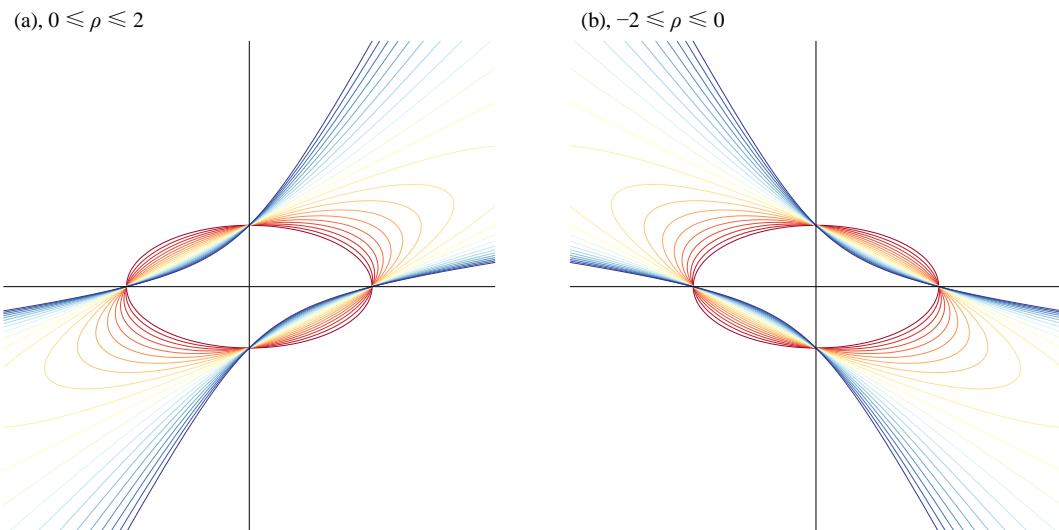
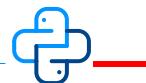


图 6. $m = 2, n = 1$ ，圆锥曲线随 ρ 变化， ρ 的变化范围为 $[-2, 2]$

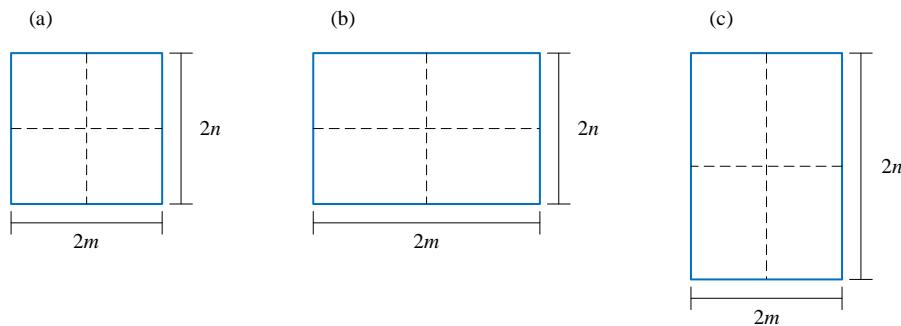


Bk3_Ch9_02.py 绘制图 4、图 5 和图 6 几幅图像。

9.4 特殊椭圆：和给定矩形相切

这一节，我们要在特殊条件约束下绘制椭圆。

给定如图 7 所示的三类矩形，假定它们的中心都位于原点。本节绘制和矩形四个边相切的椭圆。椭圆可以是正椭圆，也可以是旋转椭圆。

图 7. m 、 n 大小关系不同的矩形

对上一节 (2) 稍作修改，得到如下解析式：

$$\frac{x_1^2}{m^2} + \frac{x_2^2}{n^2} - \frac{2\rho x_1 x_2}{mn} = 1 - \rho^2 \quad (5)$$

ρ 取值范围在 -1 和 1 之间。大家很快就会发现参数 ρ 影响椭圆的倾斜程度。

(5) 可以进一步写成：

$$\frac{1}{1-\rho^2} \left(\frac{x_1^2}{m^2} + \frac{x_2^2}{n^2} - \frac{2\rho x_1 x_2}{mn} \right) = 1 \quad (6)$$

如图 8 所示，以矩形的中心为原点构造平面直角坐标系，容易计算得到矩形和椭圆相切的切点 A 、 B 、 C 、 D 的坐标为：

$$A(m, \rho n), \quad B(\rho m, n), \quad C(-m, -\rho n), \quad D(-\rho m, -n) \quad (7)$$

⚠ 请大家格外注意 AC 连线，我们将在本系列丛书的条件概率和线性回归话题中谈到这条直线。

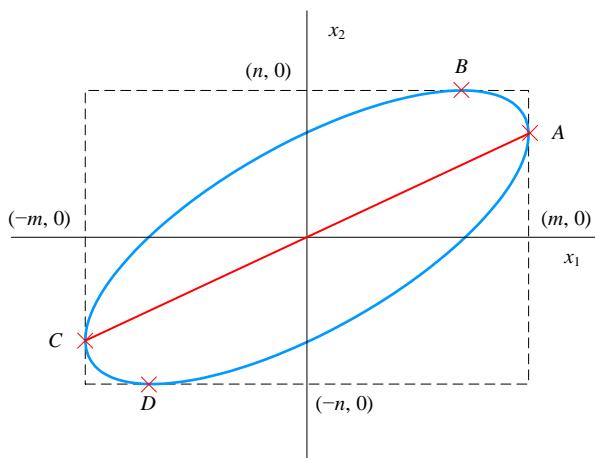


图 8. 四个切点的位置

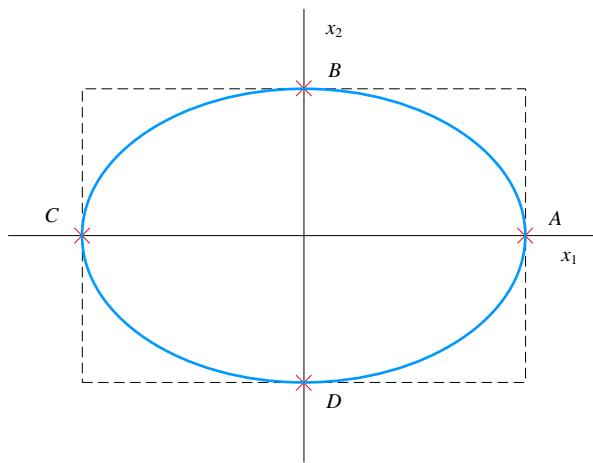
正椭圆

当 $\rho = 0$ 时，椭圆为正椭圆，即，

$$\frac{x_1^2}{m^2} + \frac{x_2^2}{n^2} = 1 \quad (8)$$

如图 9 所示，椭圆和矩形相切的四个切点 A 、 B 、 C 、 D 的坐标为：

$$A(m, 0), \quad B(0, n), \quad C(-m, 0), \quad D(0, -n) \quad (9)$$

图 9. 当 $\rho = 0$ 时，四个切点的位置

线段

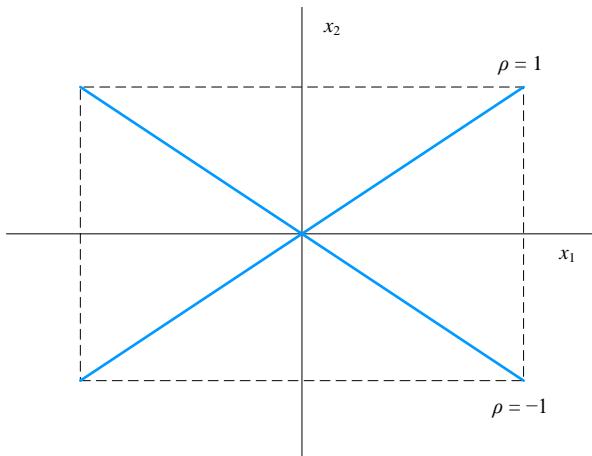
当 $\rho = 1$ 时，椭圆退化为一条线段，对应解析式为：

$$\frac{x_1}{m} - \frac{x_2}{n} = 0 \quad (10)$$

当 $\rho = -1$ 时，椭圆也是一条线段：

$$\frac{x_1}{m} + \frac{x_2}{n} = 0 \quad (11)$$

两种情况对应的图像为图 10。

图 10. 当 $\rho = \pm 1$ 时，椭圆退化成线段

旋转椭圆

图 11 所示为，当 $m = n$ ，椭圆形状随参数 ρ 变化。当 ρ 靠近 0 时，椭圆形状越接近正圆； ρ 的绝对值越靠近 1，椭圆越扁，形状越接近线段。此外，请大家格外关注切点位置随 ρ 如何移动。

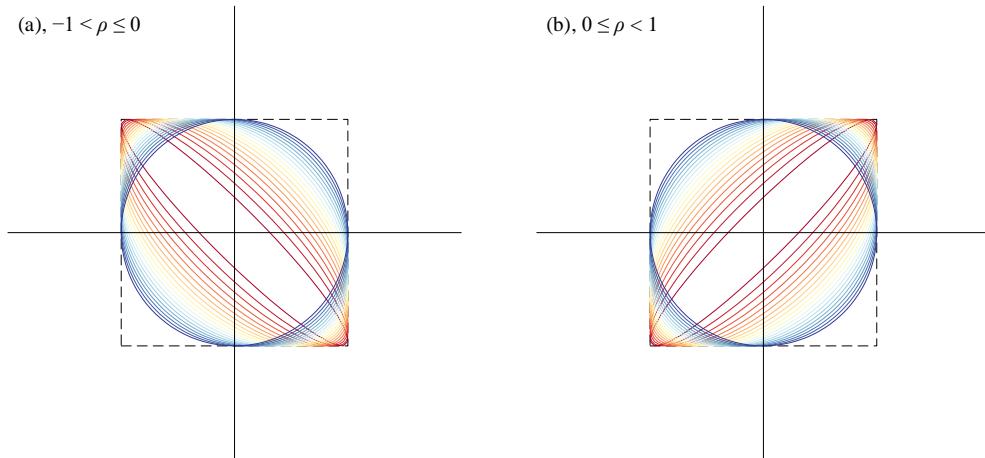
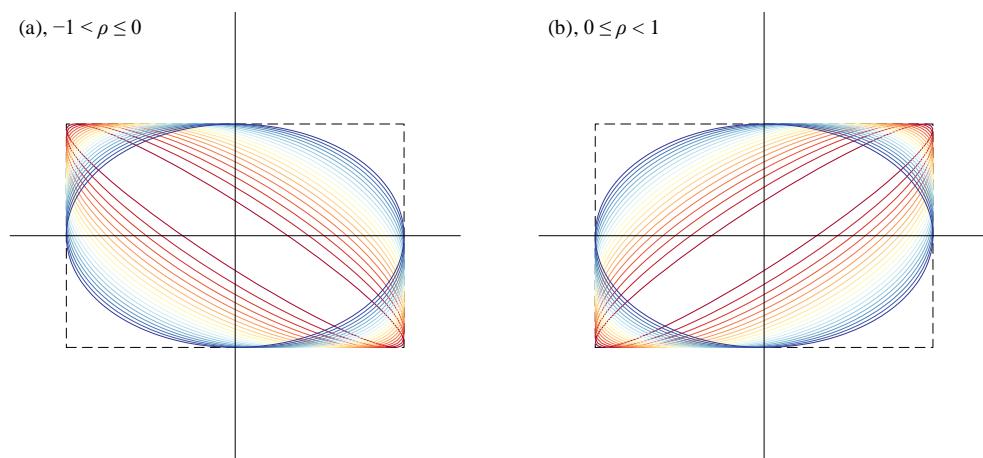
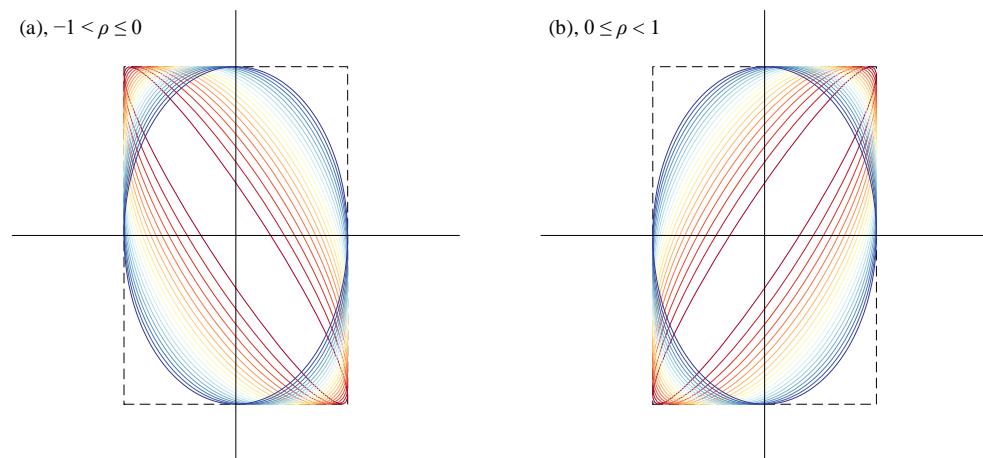
图 11. $m = n$ 时，和给定正方形相切椭圆

图 12 和图 13 分别展示 $m > n$ 和 $m < n$ 两种情况条件下，椭圆形状随 ρ 变化。

图 12. $m > n$ 时，和给定矩形相切椭圆图 13. $m < n$ 时，和给定矩形相切椭圆

二元高斯分布

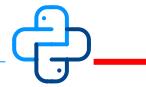
我们之所以讨论这种特殊形态的椭圆，是因为它和二元高斯分布的概率密度函数直接相关。

二元高斯分布 (bivariate Gaussian distribution) 的概率密度函数 $f_{X,Y}(x,y)$ 解析式如下：

$$f_{X,Y}(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho_{X,Y}^2}} \times \exp\left(\underbrace{\frac{-1}{2}\frac{1}{(1-\rho_{X,Y}^2)}\left(\left(\frac{x-\mu_X}{\sigma_X}\right)^2 - 2\rho_{X,Y}\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right) + \left(\frac{y-\mu_Y}{\sigma_Y}\right)^2\right)}_{\text{Ellipse}}\right) \quad (12)$$

其中， μ_X 和 μ_Y 分别为随机变量 X 、 Y 的期望值。 σ_X 和 σ_Y 分别为随机变量 X 、 Y 的均方差； $\rho_{X,Y}$ 为 X 和 Y 线性相关系数，取值区间为 $(-1, 1)$ 。

相信大家已经在 (12) 看到了 (6)。



Bk3_Ch9_03.py 绘制图 11、图 12、图 13。



我们把 Bk3_Ch9_03.py 转化成了一个 App，大家可以调节不同参数观察椭圆形状变化，以及切点位置。请大家参考代码文件 Streamlit_Bk3_Ch9_03.py。

9.5 超椭圆：和范数有关

超椭圆 (superellipse) 是对椭圆的拓展，最常见的超椭圆的解析式为：

$$\left| \frac{x_1}{a} \right|^p + \left| \frac{x_2}{b} \right|^p = 1 \quad (13)$$

一般情况， p 为大于 0 的数值。

特别地，当 $p = 2$ ，(13) 所示为椭圆解析式。

还有两个特殊的情况，当 $p = 1$ 时，超椭圆图形为菱形：

$$\left| \frac{x_1}{a} \right| + \left| \frac{x_2}{b} \right| = 1 \quad (14)$$

当 $p = +\infty$ 时，超椭圆图形为长方形，对应的解析式为：

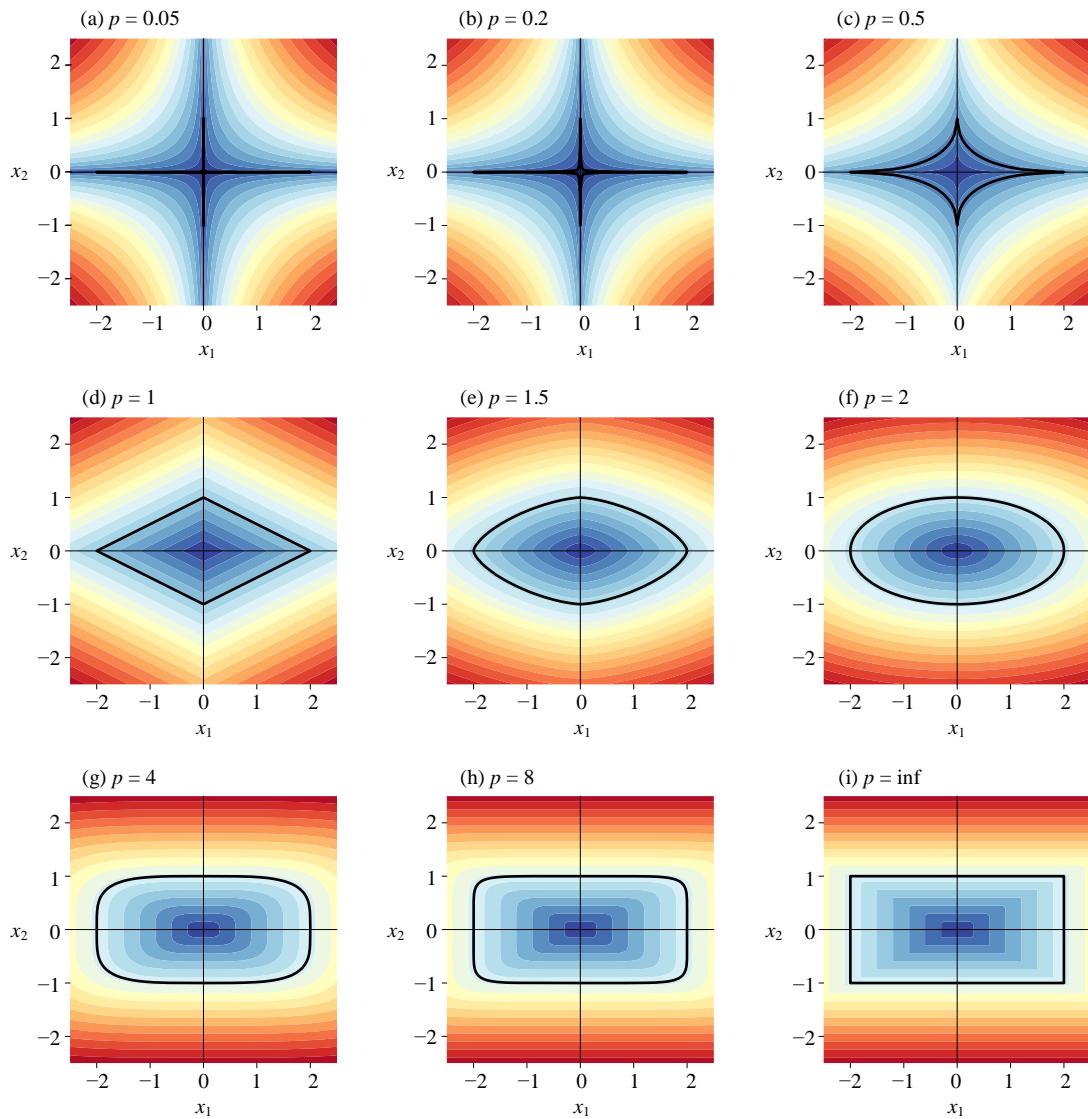
$$\max \left(\left| \frac{x_1}{a} \right|, \left| \frac{x_2}{b} \right| \right) = 1 \quad (15)$$

第一个例子

当 $a = 2$, $b = 1$ 时，超椭圆的解析式为：

$$\left| \frac{x_1}{2} \right|^p + \left| \frac{x_2}{1} \right|^p = 1 \quad (16)$$

图 14 所示为 p 取不同值时，超椭圆的形状。

图 14. 超椭圆 p 取不同值时，超椭圆的形状， $a = 2, b = 1$

第二个例子

当 $a = 1, b = 1$ 时，超椭圆的解析式为：

$$|x_1|^p + |x_2|^p = 1 \quad (17)$$

图 15 所示为 p 取不同值时，超椭圆的形状。

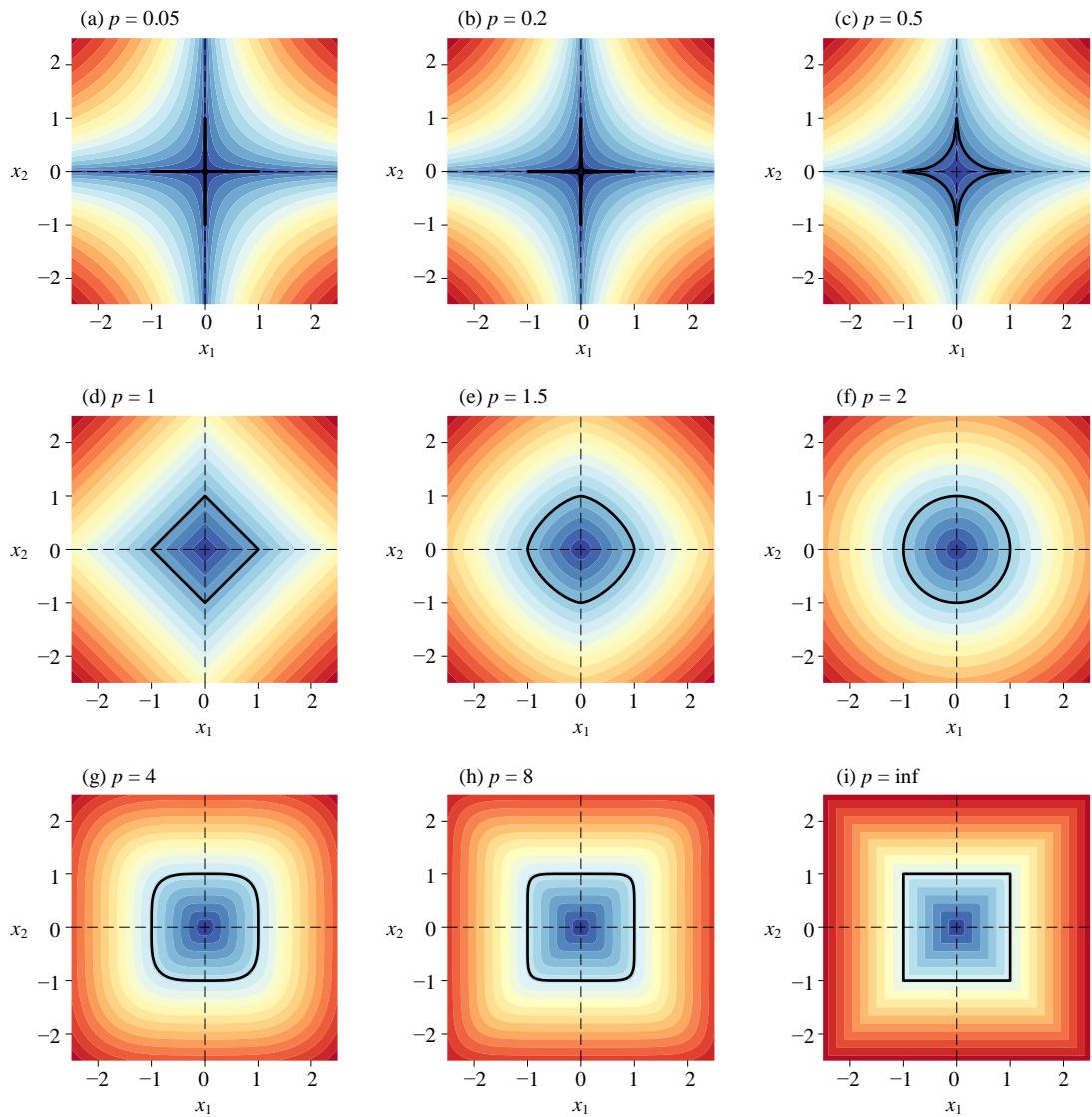


图 15. 超椭圆 \$p\$ 取不同值时，超椭圆的形状，\$a = 1, b = 1\$

\$p\$ 和 \$q\$ 两个参数

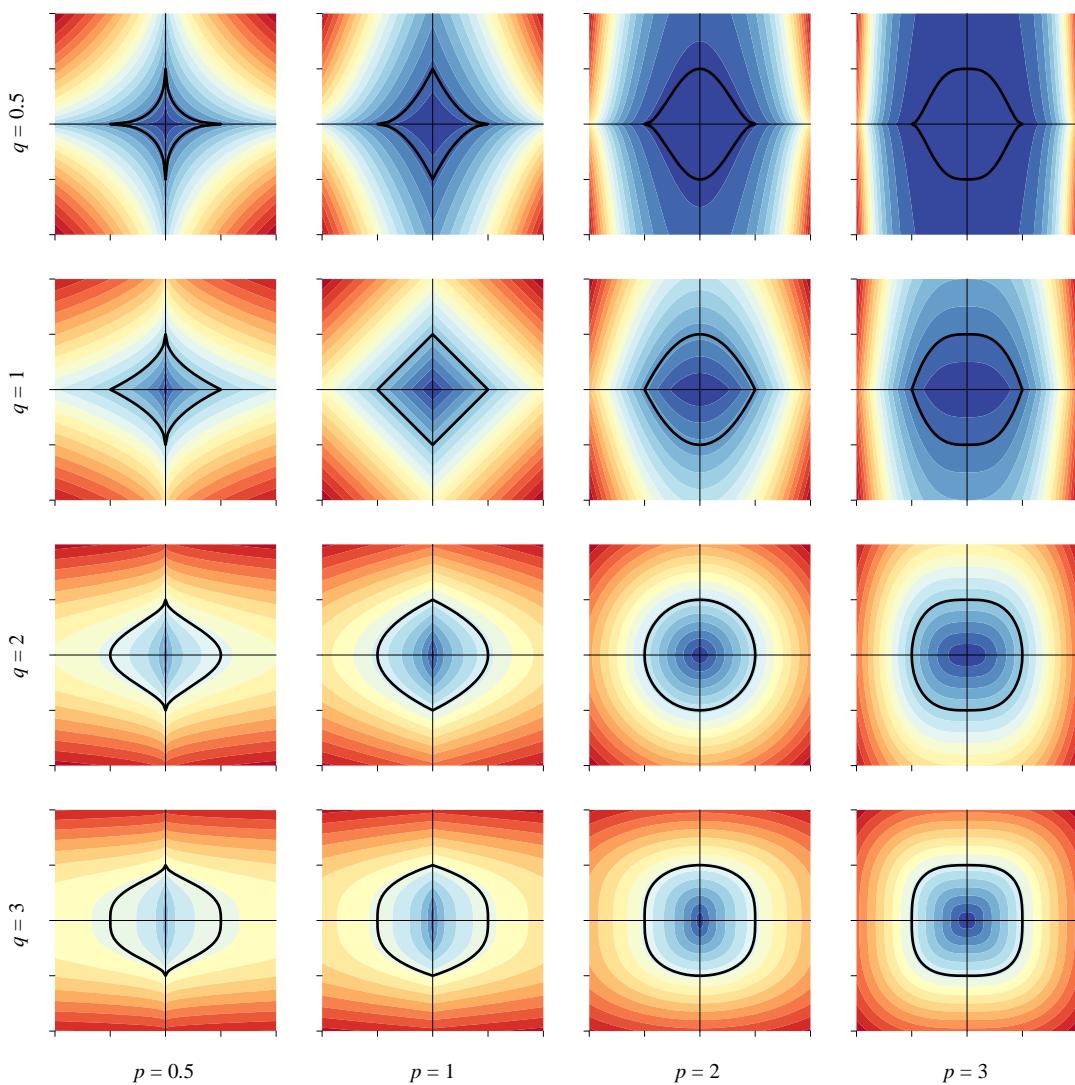
将(13)解析式进一步推广，得到如下二维平面的超椭圆解析式：

$$\left| \frac{x_1}{a} \right|^p + \left| \frac{x_2}{b} \right|^q = 1 \quad (18)$$

其中，\$p\$ 和 \$q\$ 为正数。

举个例子，当 \$a = 1, b = 1\$ 时，(18) 对应的超椭圆的解析式为：

$$|x_1|^p + |x_2|^q = 1 \quad (19)$$

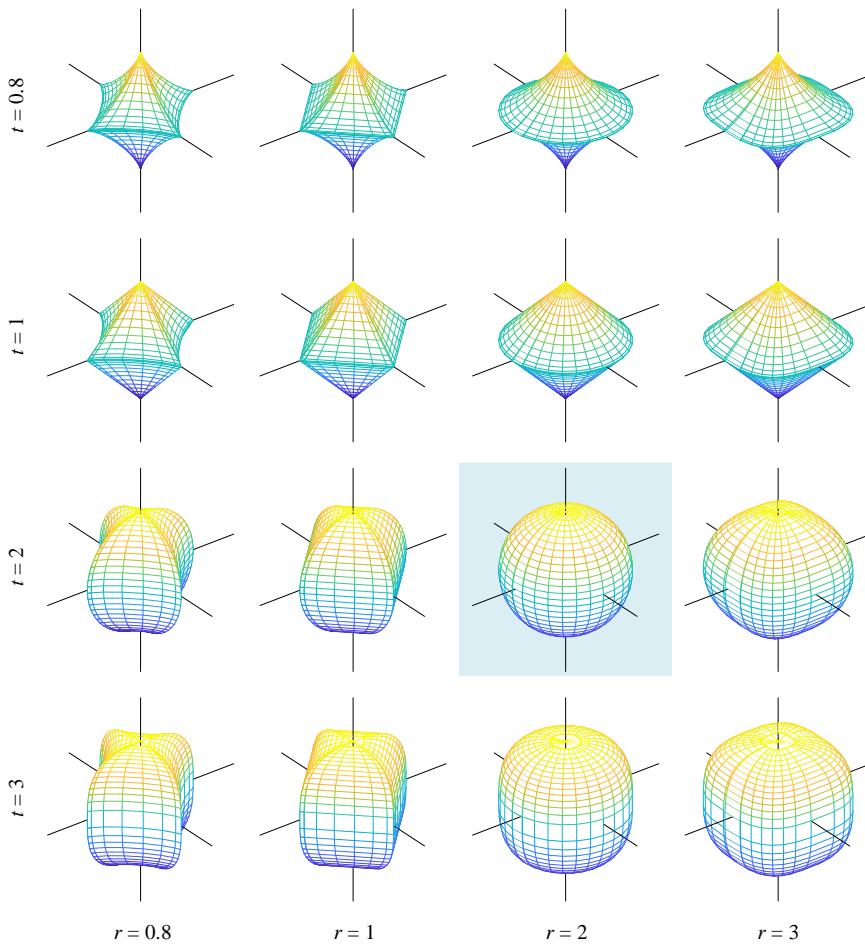
图 16 所示为 p 和 q 取不同值时，(19) 对应超椭圆的形状。图 16. p 和 q 取不同值时，超椭圆的形状， $a = 1, b = 1$

超椭球

从二维到三维，可以得到**超椭球** (superellipsoid) 的解析式：

$$\left(\left| \frac{x_1}{a} \right|^r + \left| \frac{x_2}{b} \right|^r \right)^{\frac{t}{r}} + \left| \frac{x_3}{c} \right|^t = 1 \quad (20)$$

图 17 所示为 $a = 1$ 和 $b = 1$ ， t 和 r 取不同值时，超椭球的形状。

图 17. t 和 r 取不同值时，超椭球的形状， $a = 1, b = 1$ 

本节介绍的超椭圆和 L^p 范数紧密联系。 L^p 范数的定义如下：

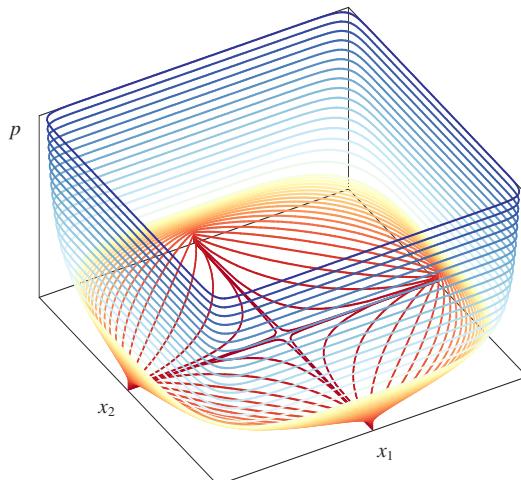
$$\|\mathbf{x}\|_p = \left(|x_1|^p + |x_2|^p + \dots + |x_D|^p \right)^{1/p} = \left(\sum_{i=1}^D |x_i|^p \right)^{1/p} \quad (21)$$

其中，

$$\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_D]^T \quad (22)$$

图 18 所示为随着 p 增大， L^p 范数等距线一层层包裹。在数据科学和机器学习中， L^p 范数常用来度量距离。当 $p = 2$ ，(21) 就是 L^2 范数，这便是前文介绍的欧氏距离。

本系列丛书将在《矩阵力量》一册系统讲解范数。

图 18. 随着 p 增大, L^p 范数等距线一层层包裹

Bk3_Ch9_04.py 绘制图 14、图 15、图 16。



在 Bk3_Ch9_04.py 基础上, 我们做了一个 App, 大家可以调节参数观察超椭圆形状变化。
请大家参考代码文件 Streamlit_Bk3_Ch9_04.py。

9.6 双曲函数：基于单位双曲线

当 $a = 1$ 和 $b = 1$ 时, 双曲线为**单位双曲线** (unit hyperbola):

$$x_1^2 - x_2^2 = 1 \quad a, b > 0 \quad (23)$$

类似前文提到过的三角函数和单位圆之间关系, 单位双曲线可以用来定义**双曲函数** (hyperbolic function)。

如图 19 所示, 最基本的双曲函数是双曲正弦函数 $\sinh()$ 和双曲余弦函数 $\cosh()$ 。

双曲正切 $\tanh()$, 可以通过如下比例计算得到:

$$\tanh \theta = \frac{\sinh \theta}{\cosh \theta} \quad (24)$$



双曲正切函数 $\tanh()$ 是 S 型函数中重要的一种，本书第 12 章将深入介绍。

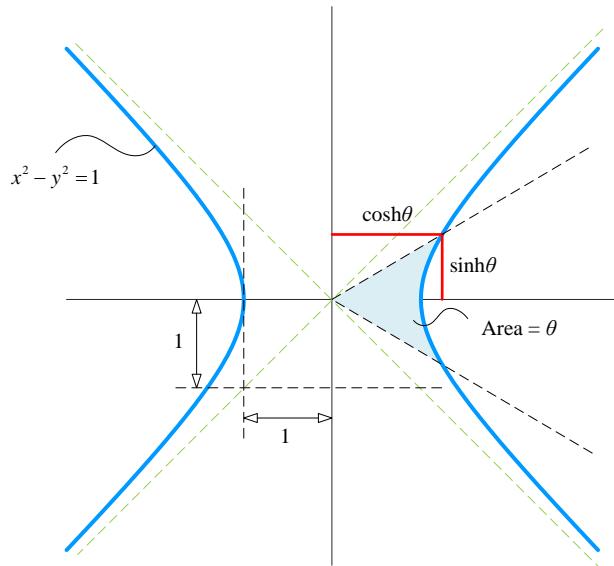


图 19. 单位双曲线和双曲函数的关系

图 20 所示为 $\sinh\theta$ 、 $\cosh\theta$ 和 $\tanh\theta$ 三个函数之间的图像关系。

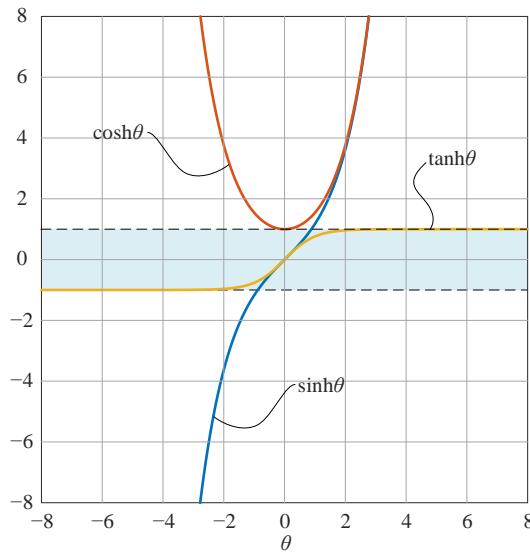


图 20. $\sinh\theta$ 、 $\cosh\theta$ 和 $\tanh\theta$ 三者关系

表 1. 用英文表达双曲函数

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

数学表达	英文表达	中文表达
$\sinh \theta$	hyperbolic sine theta sinh /sintʃ/ theta	双曲正弦
$\cosh \theta$	hyperbolic co sine theta cosh /kɒʃ/ theta	双曲余弦
$\tanh \theta$	hyperbolic tangent theta tanh /taentʃ/ theta	双曲正切

和指数函数关系

此外， $\sinh \theta$ 、 $\cosh \theta$ 和 $\tanh \theta$ 三个函数和指数函数 $\exp(\theta)$ 存在以下关系：

$$\begin{aligned}\sinh \theta &= \frac{\exp(\theta) - \exp(-\theta)}{2} \\ \cosh \theta &= \frac{\exp(\theta) + \exp(-\theta)}{2} \\ \tanh \theta &= \frac{\sinh \theta}{\cosh \theta} = \frac{\exp(\theta) - \exp(-\theta)}{\exp(\theta) + \exp(-\theta)}\end{aligned}\quad (25)$$

图 21 所示为 $\sinh \theta$ 和 $\cosh \theta$ 与指数函数关系。

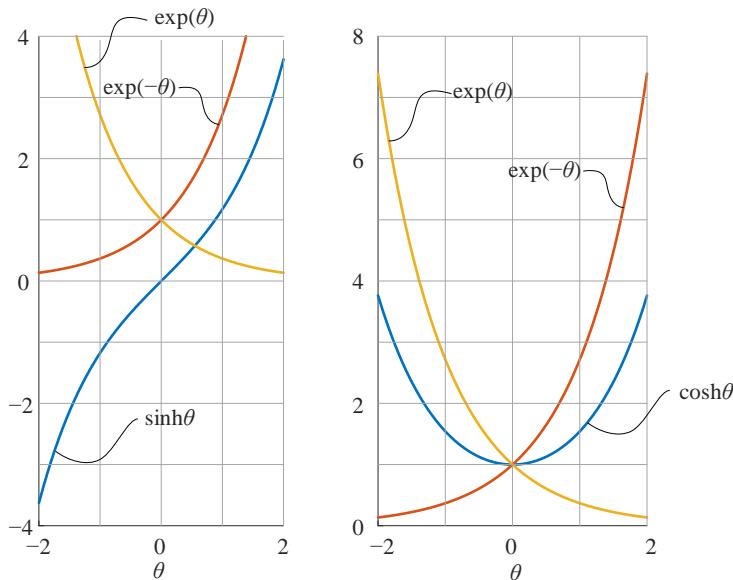


图 21. $\sinh \theta$ 和 $\cosh \theta$ 与指数函数关系

9.7 圆锥曲线一般式

圆锥曲线的一般形如下：

$$Ax_1^2 + Bx_1x_2 + Cx_2^2 + Dx_1 + Ex_2 + F = 0 \quad (26)$$

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

满足下列条件，圆锥曲线为正圆：

$$Ax_1^2 + Cx_2^2 + Dx_1 + Ex_2 + F = 0, \quad A = C \quad (27)$$

满足下列条件，圆锥曲线为正椭圆，即没有旋转：

$$Ax_1^2 + Cx_2^2 + Dx_1 + Ex_2 + F = 0, \quad A \neq C, \quad AC > 0 \quad (28)$$

满足下列条件，圆锥曲线为正双曲线：

$$Ax_1^2 + Cx_2^2 + Dx_1 + Ex_2 + F = 0, \quad AC < 0 \quad (29)$$

满足下列任一等式，圆锥曲线为正抛物线：

$$\begin{cases} Ax_1^2 + Dx_1 + Ex_2 + F = 0 \\ Cx_2^2 + Dx_1 + Ex_2 + F = 0 \end{cases} \quad (30)$$

⚠ 注意 当 $B \neq 0$ 时，圆锥曲线存在旋转，需要通过 $B^2 - 4AC$ 来判断圆锥曲线类型。

$B^2 - 4AC < 0$ 时，圆锥曲线为椭圆； $B^2 - 4AC = 0$ ，圆锥曲线为抛物线； $B^2 - 4AC > 0$ 时，圆锥曲线为双曲线。

大家可能会问，为何要采用 $B^2 - 4AC$ 来判断圆锥曲线类型？我们将在《矩阵力量》回答这个问题。

矩阵运算

把 (26) 写成如下矩阵运算式：

$$\frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 2A & B \\ B & 2C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} D \\ E \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + F = 0 \quad (31)$$

进一步写成：

$$\frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} + \mathbf{w}^T \mathbf{x} + F = 0 \quad (32)$$

其中，

$$\mathbf{Q} = \begin{bmatrix} 2A & B \\ B & 2C \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} D \\ E \end{bmatrix} \quad (33)$$

目前不需要大家掌握 (31) 这个矩阵运算式。我们也将再《矩阵力量》一册深入分析这个等式。



正如牛顿所言，“我不知道世人看我的眼光。依我看来，我不过是一个在海边玩耍的孩子，不时找到几个光滑卵石、漂亮贝壳，而惊喜万分。而展现在我面前的是，真理的浩瀚海洋，静候探索。”

人类何尝不是在宇宙某个角落玩耍的一群孩子，手握的知识不过沧海一粟，却雄心万丈一心要去探索星辰大海。

但也正是这群孩子将无数的不可能变成了可能，现在他们已经在地月系、甚至太阳系的边缘跃跃欲试。

今人不见古时月，今月曾经照古人。宇宙的星辰大海一直都在人类眼前，它从未走远。路漫漫其修远兮，吾将上下而求索。

地球不过是人类的摇篮，我们的征途是星辰大海。这句话含蓄而浪漫。刘慈欣《三体》中则说的更为露骨而冷酷——“我们都是阴沟里的虫子，但总还是得有人仰望星空。”

10

Functions Meet Coordinate Systems

函数

从几何图形角度探究



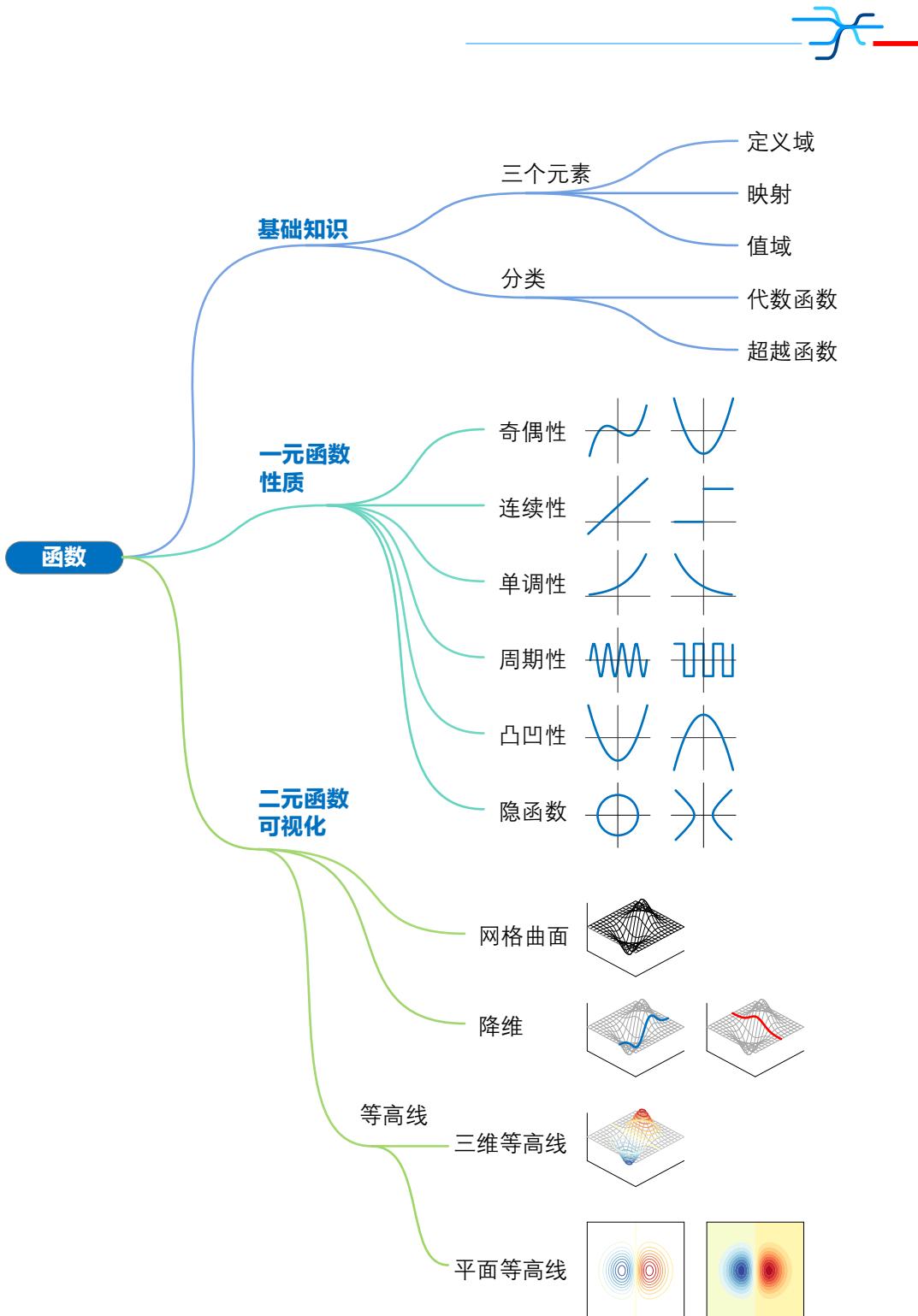
音乐是一种隐藏的数学实践，它是大脑潜意识下的计算。

Music is the hidden arithmetical exercise of a mind unconscious that it is calculating.

——戈特弗里德·莱布尼茨 (Gottfried Wilhelm Leibniz) | 德意志数学家、哲学家 | 1646 ~ 1716



- ◀ `matplotlib.pyplot.axhline()` 绘制水平线
- ◀ `matplotlib.pyplot.axvline()` 绘制竖直线
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `numpy.linspace()` 在指定的间隔内，返回固定步长的数据
- ◀ `numpy.meshgrid()` 获得网格数据
- ◀ `plot_wireframe()` 绘制三维单色线框图
- ◀ `sympy.abc` 引入符号变量
- ◀ `sympy.diff()` 对符号函数求导
- ◀ `sympy.exp()` 符号运算中以 e 为底的指数函数
- ◀ `sympy.Interval` 定义符号区间
- ◀ `sympy.is_increasing` 判断符号函数的单调性
- ◀ `sympy.lambdify()` 将符号表达式转化为函数



10.1 当代数式遇到坐标系

坐标系给每个冷冰冰的代数式赋予生命。图 1 ~ 图 3 给出了九幅图像，他们多数是函数，也有隐函数和参数方程。

建议大家盯着每幅图像看一会儿，你会惊奇地发现，坐标系给这些函数插上了翅膀，让他们在空间腾跃、讲述自己的故事。

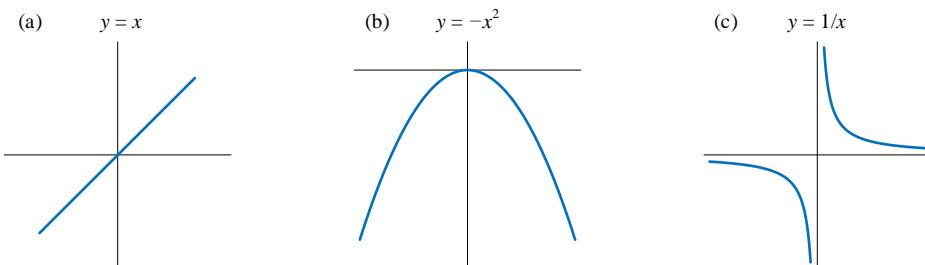


图 1. 一次函数、二次函数和反比例函数

线性函数 $y = x$ 是个坚毅果敢、埋头苦干的家伙。你问他，“你要去哪？”他莫不做声，自顾自地向着正负无穷，无限延伸，直到世界尽头。

抛物线 $y = -x^2$ 像一条腾出水面的锦鲤，在空中划出一道优美的弧线，他飞跃龙门、修成正果。从此岸到彼岸，离家越远，心就离家越近。

反比例函数 $y = 1/x$ 像一个哲学家，他在讲述——太极者，无极而生，动静之机，阴阳之母也。物极必反，任何事物都有两面，而且两面会互相转化。

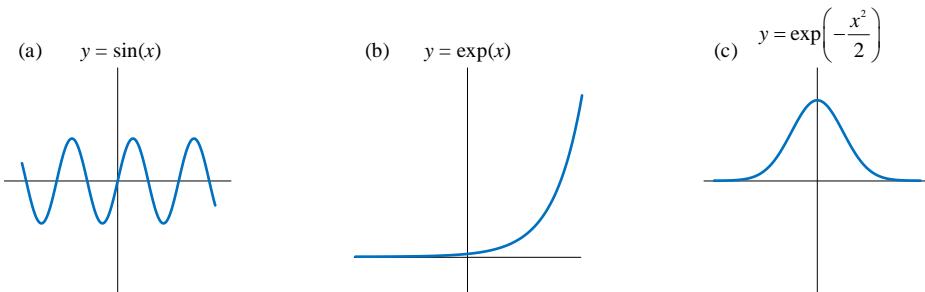


图 2. 正弦函数、指数函数和高斯函数

海水无风时，波涛安悠悠。正弦函数 $y = \sin(x)$ 像是海浪，永远波涛澎湃。它代表着生命的律动，你仿佛能够听到它的脉搏砰砰作响。

指数函数 $y = \exp(x)$ 就是那条巨龙。起初，他韬光养晦、潜龙勿用。万尺高楼起于累土，他不知疲倦、从未停歇。你看他，越飞越快，越升越高，如今飞龙在天。

君不见黄河之水天上来，奔流到海不复回。优雅而神秘，高斯函数 $y = \exp(-x^2/2)$ 好比高山流水。上善若水，涓涓细流，利万物而不争。

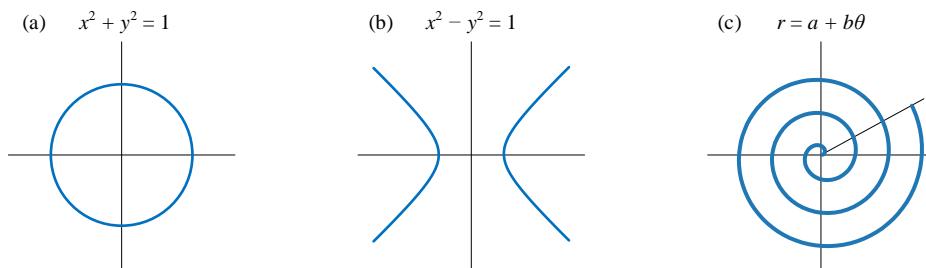


图 3. 正圆、双曲线和阿基米德螺旋线，非函数

海上生明月，天涯共此时。 $x^2 + y^2 = 1$ 是挂在天上的白玉盘，是家里客厅的圆饭桌，是捧在手里的圆月饼。转了一圈，圆心是家。

人有悲欢离合，月有阴晴圆缺，此事古难全。造化弄人， $x^2 + y^2 = 1$ 正号 + 改为负号 -，就变成双曲线 $x^2 - y^2 = 1$ 。两条曲线隔空相望，如此的近，又如此的远。像牛郎和织女，盈盈一水间，脉脉不得语。

阿基米德螺旋线好似夜空中的银河星系，把我们的目光从人世的浮尘，拉到深蓝的虚空，让我们片刻间忘却了这片土地的悲欢离合。

10.2 一元函数：一个自变量

如果函数 f 以 x 作为唯一输入值，输出值写作 $y = f(x)$ ，函数是一元函数。也就是说，有一个自变量的函数叫做**一元函数** (univariate function)。

本书前文介绍过，函数输入值 x 构成的集合叫做**定义域**，函数输出值 y 构成的集合叫做**值域**。

⚠ 注意，定义域中任一 x 在值域中有唯一对应的 y 。当然，不同 x 可以对应一样的函数值 $f(x)$ 。

平面直角坐标系中，一般用**线图** (line plot 或 line chart) 作为函数的可视化方案。图 4 展示的便是一元函数的映射关系，以及几种一元函数示例。

白话说，函数就是一种数值转化。本书第 6 章讲解不等式时，我们做过这样一个实验，给满足不等式条件的变量一个标签——1 (True)；不满足不等式的变量结果为 0 (False)。这实际上也是函数映射，输入为定义域内自变量的取值，输出为两值之一——0 或 1。

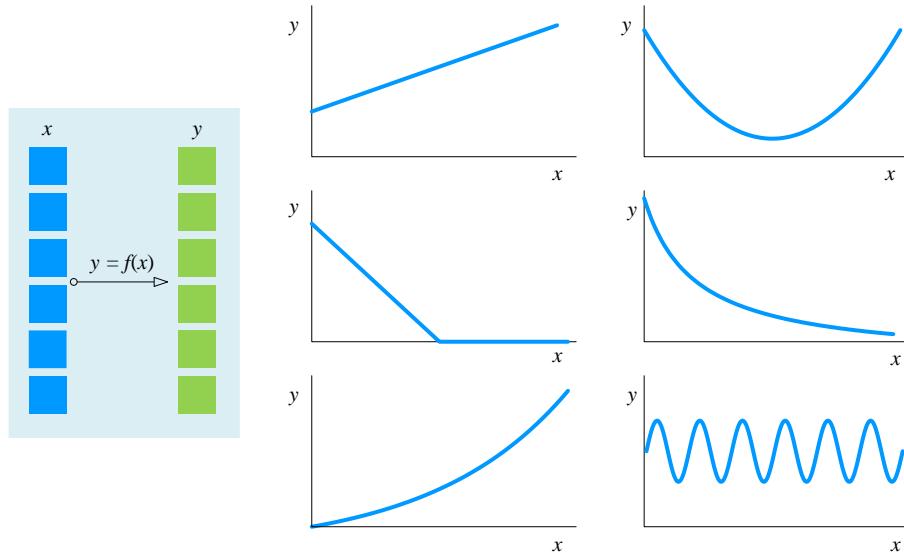


图 4. 一元函数

数据科学和机器学习中常用的函数一般分为代数函数 (algebraic function) 和超越函数 (transcendental function)。代数函数是指通过常数与自变量相互之间有限次的加、减、乘、除、有理指数幂和开方等运算构造的函数。本书将绝对值函数也归类到代数函数中。

超越函数指的是“超出”代数函数范畴的函数，比如对数函数、指数函数、三角函数等等。

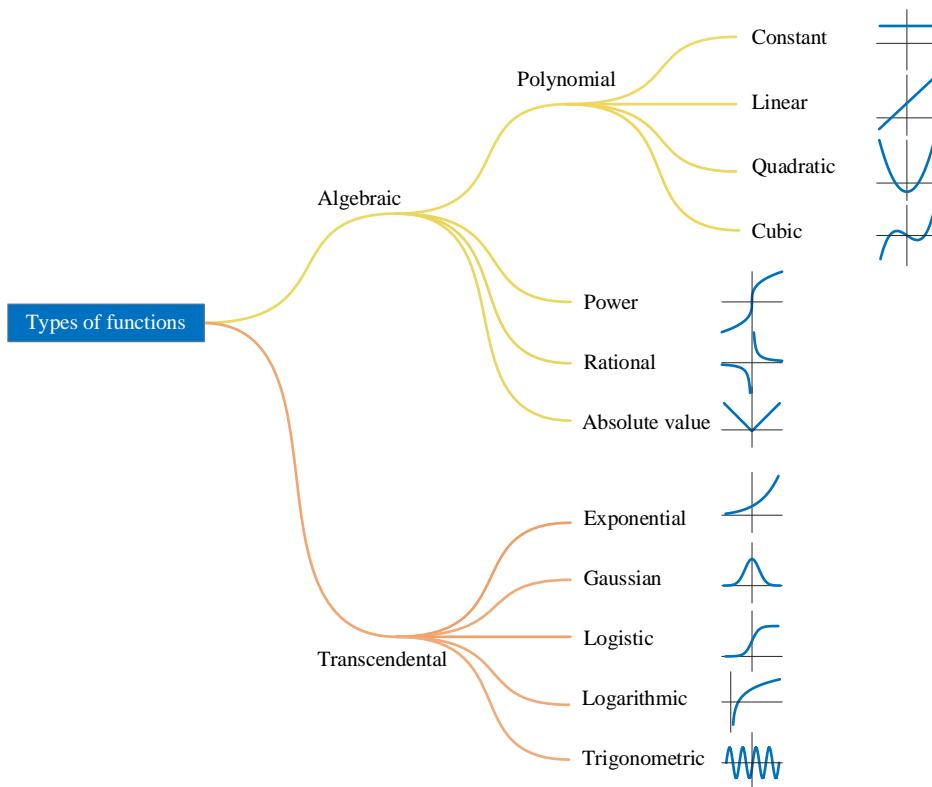


图 5. 常见函数分类



函数在机器学习中扮演重要角色。下面以神经网络 (neural network) 为例，简单介绍函数的作用。

神经网络的核心思想是模拟人脑神经元 (neuron) 的工作原理。图 6 展示神经元基本生物学结构。神经元细胞体 (cell body) 的核心是细胞核 (nucleus)，细胞核周围围绕着树突 (dendrite)。树突接受外部刺激，并将信号传递至神经元内部。

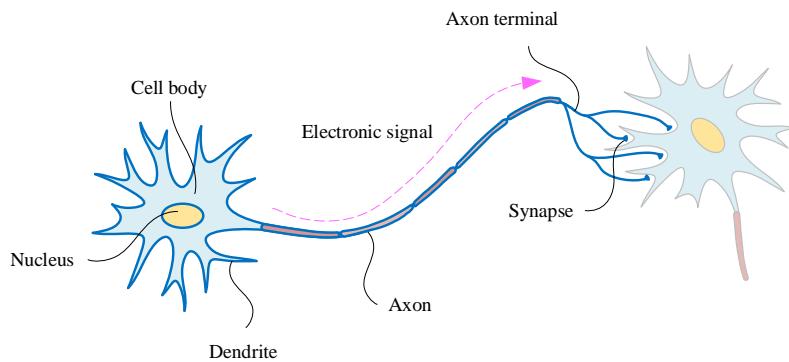


图 6. 神经元结构

细胞体汇总不同树突刺激，当刺激达到一定程度时，激发细胞兴奋状态；否则，细胞处于抑制状态。轴突 (axon) 则负责将兴奋状态通过轴突末端 (axon terminal) 的突触 (synapse) 等结构传递到另一个神经元或组织细胞。

图 7 可看作是对神经元简单模仿。神经元模型的输入 x_1, x_2, \dots, x_D 类似于神经元的树突， x_i 取值为简单的 0 或 1。这些输入分别乘以各自权重，再通过求和函数汇集到一起得到 x 。接着， x 值再通过一个判别函数 $f()$ 得到最终的值 y 。

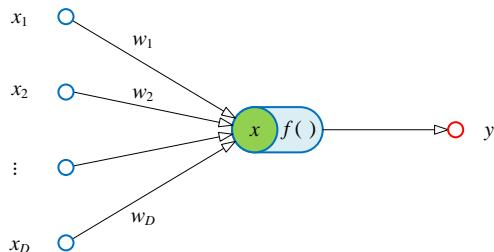


图 7. 最简单的神经网络模型

图 8 展示的便是几种常见的判别函数 $f()$ 及其对应图像。

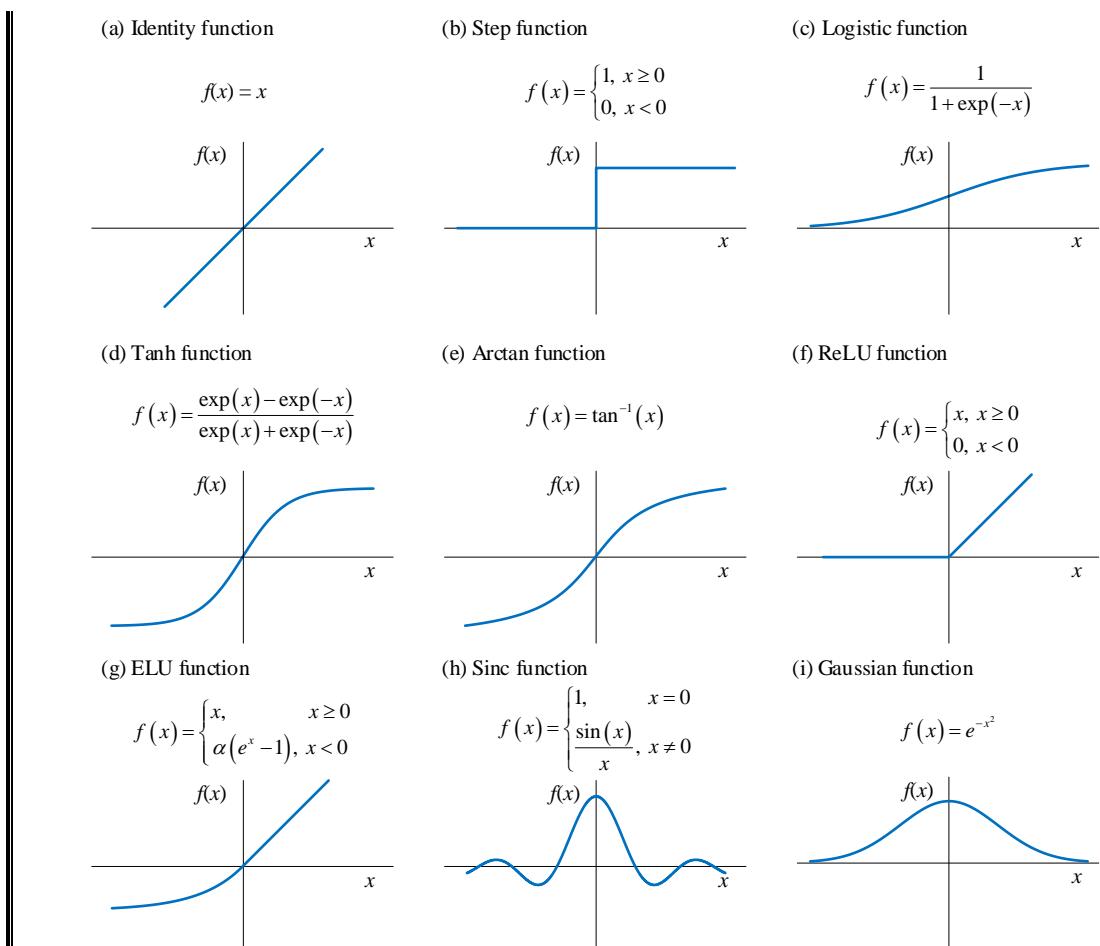


图 8. 几种神经网络中常见的判别函数

10.3 一元函数性质

学习函数时，请大家关注函数这几个特征：形状及变化趋势、自变量取值范围、函数值取值范围、函数性质等。

下面，本节利用图9介绍一元函数常见性质。

奇偶性

图9(a)所示函数为**偶函数** (even function)。

$f(x)$ 若为偶函数，对于定义域内任意 x 如下关系都成立：

$$f(x) = f(-x) \quad (1)$$

从几何角度， $f(x)$ 若为偶函数，函数图像关于纵轴对称。

如图9 (b) 所示，如果 $f(x)$ 为**奇函数** (odd function)，对于定义域内任意 x 如下关系都成立：

$$f(-x) = -f(x) \quad (2)$$

从几何角度， $f(x)$ 若为奇函数，函数图像关于原点对称。

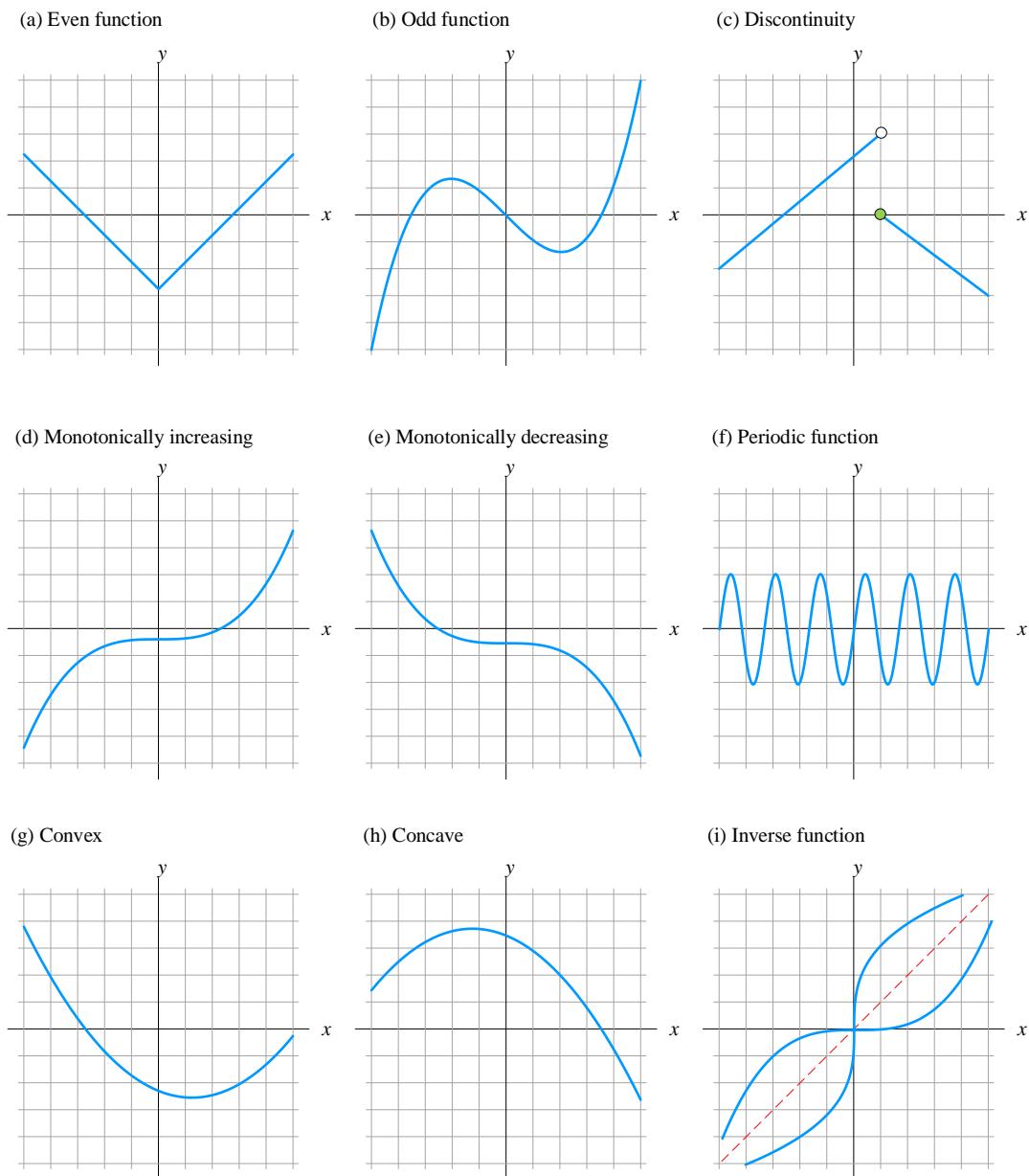


图 9. 一元函数常见性质

连续性

简单来说，**连续函数** (continuous function) 是指函数 $y = f(x)$ 当自变量 x 的变化很小时，所引起的因变量 y 的变化也很小，即没有函数值突变。与之相对的就是**不连续函数** (discontinuous function)。图 9 (c) 给出的是函数存在**不连续** (discontinuity)。

如图 10 所示，不连续函数有几种：**渐近线间断** (asymptotic discontinuity)，**点间断** (point discontinuity)，还有**跳跃间断** (jump discontinuity)。

在学习**极限** (limit) 之后，函数的**连续性** (continuity) 更容易被定义。此外，函数的连续性和**可导性** (differentiability) 有着密切联系。



导数是本书第 15 章要讨论的内容。

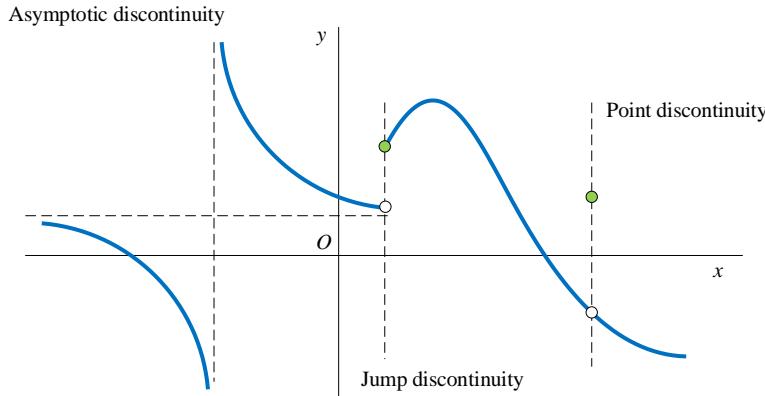
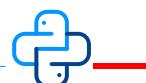


图 10. 几种不连续函数特征

单调性

图 9 (d) 和(e) 描述的是函数**单调性** (monotonicity)。图 9 (d) 给出的函数为**单调递增** (monotonically increasing)，图 9 (e) 对应的函数则是**单调递减** (monotonically decreasing)。



`sympy.is_decreasing()` 可以用来判断符号函数的单调性。Bk3_Ch10_01.py 展示如何判断函数在不同区间内单调性。

周期性

图 9 (f) 所示为函数**周期性** (periodicity)。如果函数 f 中不同位置 x 满足下式，则函数为周期函数：

$$f(x+T) = f(x) \quad (3)$$

其中， T 为周期 (period)，也叫最小正周期。三角函数就是典型的周期函数。图 11 给出的是四个其他周期函数的例子。

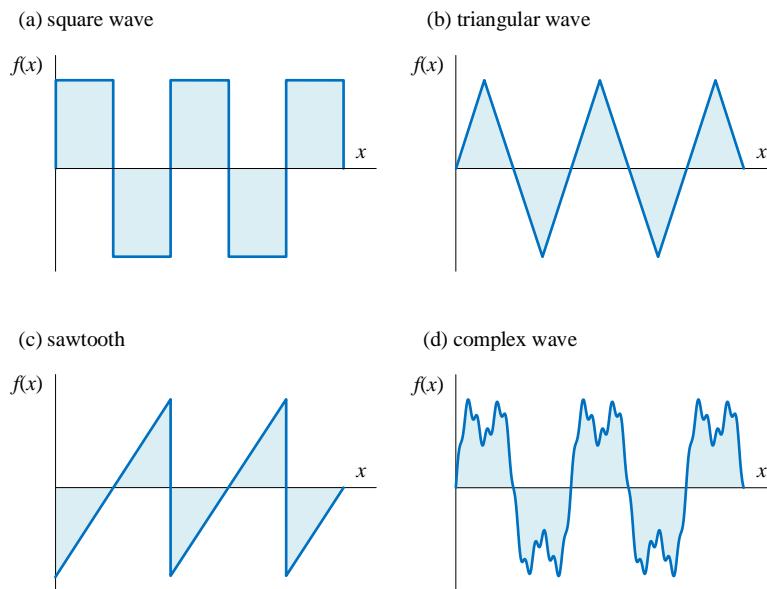


图 11. 四个周期函数

凸凹性

图 9 (g) 所示为凸函数 (convex function)，图 9 (h) 所示为凹函数 (concave function)。

⚠ 注意，国内数学教材对凸凹的定义，可能和本书正好相反。

下面聊一下凸凹函数的确切定义和特点。

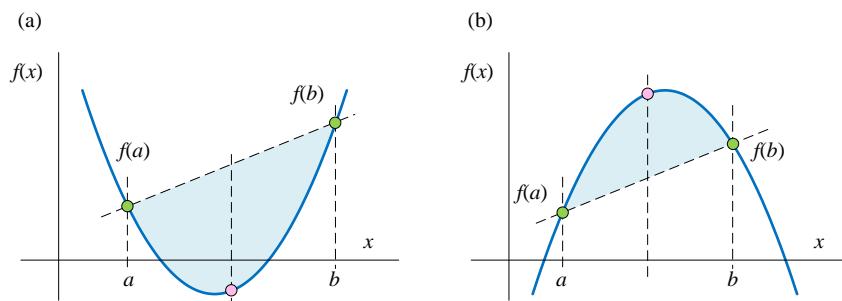


图 12. 函数凸凹性

如图 12 (a) 所示，若 $f(x)$ 在区间 I 有定义，如果对于任意 $a, b \in I$ ，且 $a \neq b$ ，如果满足：

$$f\left(\frac{a+b}{2}\right) < \frac{f(a)+f(b)}{2} \quad (4)$$

则称 $f(x)$ 在该区间内为凸函数。

如图 12 (b) 所示，如果对于任意 $a, b \in I$, 且 $a \neq b$, 如果满足：

$$f\left(\frac{a+b}{2}\right) > \frac{f(a)+f(b)}{2} \quad (5)$$

则称 $f(x)$ 在该区间内为凹函数。

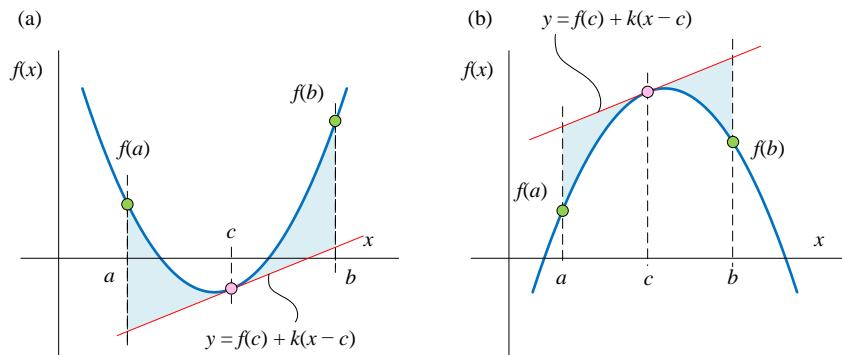


图 13. 切线角度看函数凸凹性

再从切线角度来看函数凸凹性。如图 13 所示，在 (a, b) 区间内一点 $x = c$ ，在函数上 $(c, f(c))$ 做一条切线，切线的解析式为：

$$y = f(c) + k(x - c) \quad (6)$$

如图 13 (a) 所示，如果函数为凸函数，当 $x \neq c$ ，函数 $f(x)$ 图像在切线上方，也就是说：

$$f(x) > f(c) + k(x - c), \quad x \in (a, b), x \neq c \quad (7)$$

有人可能会问，(6) 的 k 是什么？具体值是什么？ k 就是函数在 $x = c$ 切线的斜率。

如图 13 (b) 所示，如果函数为凹函数，当 $x \neq c$ ，函数 $f(x)$ 图像在切线下方，即：

$$f(x) < f(c) + k(x - c), \quad x \in (a, b), x \neq c \quad (8)$$



此外，函数的凸凹性和极值有着密切联系，本书第 19 章将介绍。

反函数

反函数 (inverse function) $x = f^{-1}(y)$ 的定义域、值域分别是函数 $y = f(x)$ 的值域、定义域。图 9 (i) 给出的是函数 f 和其反函数 f^{-1} ，两者关系如下：

$$f^{-1}(f(x)) = x \quad (9)$$

原函数和反函数的图像关于 $y = x$ 直线对称。此外，并不是所有函数都存在反函数。

隐函数

隐函数 (implicit function) 是由**隐式方程** (implicit equation) 所隐含定义的函数。比如，隐式方程 $F(x_1, x_2) = 0$ 描述 x_1 和 x_2 两者关系。

不同于一般函数，很多隐函数较难分离自变量和因变量，比如图 14 所示两个例子。和函数一样，隐函数可以扩展到多元，比如图 15 所示为三元隐函数的例子。后续，我们会专门介绍如何用 Python 绘制如图 14 所示的隐函数图像。

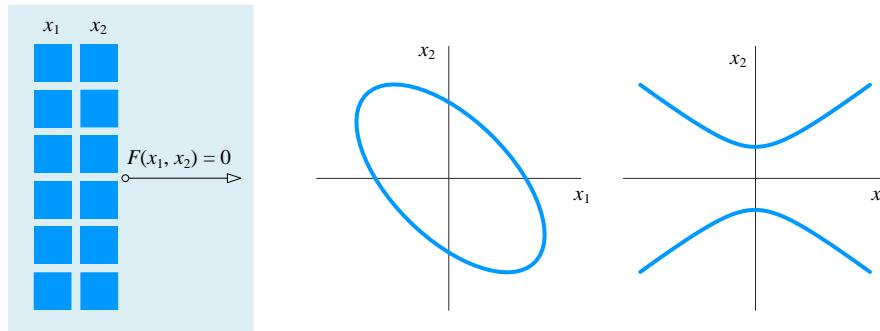


图 14. 二元隐函数

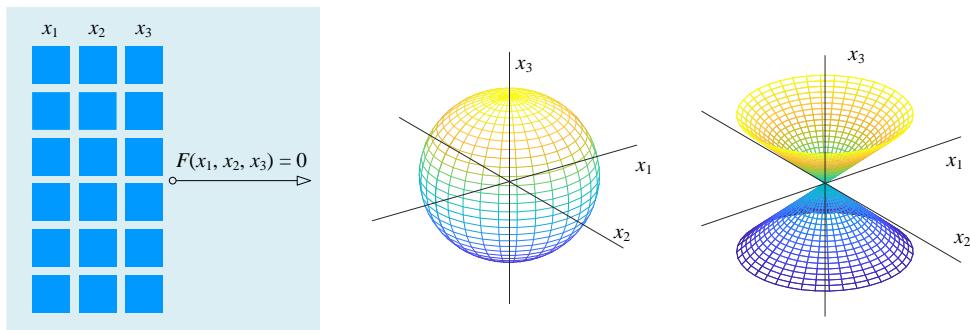


图 15. 三元隐函数

变化率和面积

很多数学问题要求我们准确地计算函数的变化率。从几何角度，如图 16(a) 所示，函数上某一点切线的斜率正是函数的变化率。微积分中，这个函数变化率叫做**导数** (derivative)。

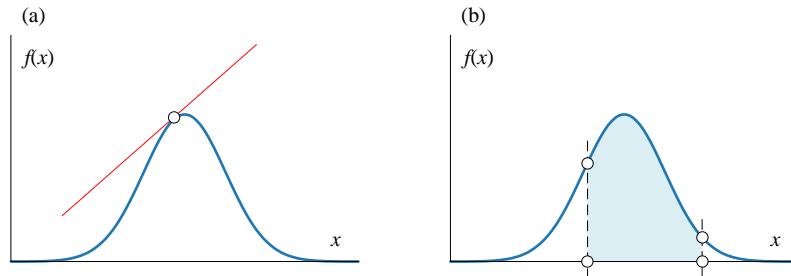


图 16. 函数的变化率和面积

进一步细看图 16(a) 给出的函数数值变化。如图 17 所示，很明显在 A 和 B 两个区域，随着 x 增大 $f(x)$ 增大，也就是变化率为正。即 A 和 B 两个区域，在函数曲线上任意一点做切线，切线的斜率为正。

但是，在 A 这个区域， x 增大时， $f(x)$ 增速加快，也就是函数“变化率的变化率”为正；而在 B 区域， x 增大时， $f(x)$ 增速逐步放缓，即函数“变化率的变化率”为负。



这个“变化率的变化率”就是二阶导数，这是本书第 15 章要介绍的内容。

再看 C 和 D 两个区域，随 x 增大 $f(x)$ 减小，即变化率为负。也就是说，在这两个区域，函数曲线上任意一点做切线，切线的斜率为负。不同的是，在 C 区域， x 增大， $f(x)$ 加速下降；在 D 区域， x 增大， $f(x)$ 下降逐步放缓。

大家试着在 A、B、C 和 D 区内函数曲线上分别找一点画切线，看一下切线是在函数曲线的“上方”，还是“下方”。并对应分析这个特征和“变化率的变化率”正负的关系。

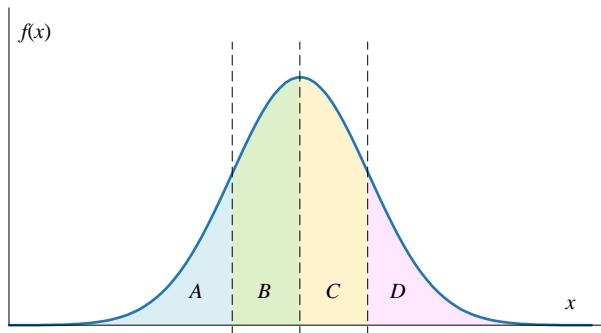


图 17. 细看函数的变化率

如图 16 (b) 所示，一些数学问题求解面积时，需要计算某个函数图形在一定取值范围和横轴围成几何图形的面积，这就要求大家了解**积分** (integral) 这个数学工具。

→ 本书第 15 ~ 18 章会着重介绍导数和积分这两个数学工具。

10.4 二元函数：两个自变量

有两个自变量函数叫做**二元函数** (bivariate function)，比如 $y = f(x_1, x_2)$ 。本书常常借助三维直角坐标系可视化二元函数。图 18 所示为二元函数映射关系以及几个示例。

举个例子，二元一次函数 $y = f(x_1, x_2) = x_1 + x_2$ 有 x_1 和 x_2 两个自变量。当 x_1 和 x_2 取值分别为 $x_1 = 2, x_2 = 4$ ，函数值 $f(x_1 = 2, x_2 = 4) = 2 + 4 = 6$ 。

此外，有多个自变量函数叫做**多元函数** (multivariate function)，比如 $y = f(x_1, x_2, \dots, x_D)$ 有 D 个自变量。

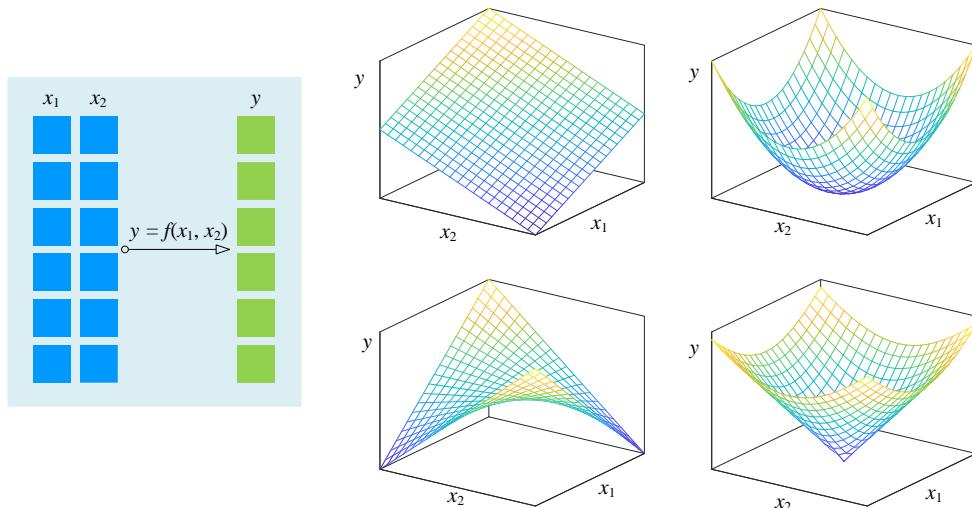


图 18. 二元函数

网格化数据

为了获得 $f(x_1, x_2)$ 在三维空间的图形，需要提供一系列整齐的网格化坐标值 (x_1, x_2) ，如下：

$$(x_1, x_2) = \begin{bmatrix} (-4, -4) & (-2, -4) & (0, -4) & (2, -4) & (4, -4) \\ (-4, -2) & (-2, -2) & (0, -2) & (2, -2) & (4, -2) \\ (-4, 0) & (-2, 0) & (0, 0) & (2, 0) & (4, 0) \\ (-4, 2) & (-2, 2) & (0, 2) & (2, 2) & (4, 2) \\ (-4, 4) & (-2, 4) & (0, 4) & (2, 4) & (4, 4) \end{bmatrix} \quad (10)$$

将上述坐标点 x_1 和 x_2 分离并写成两个矩阵形式：

$$x_1 = \begin{bmatrix} -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -4 & -4 & -4 & -4 & -4 \\ -2 & -2 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix} \quad (11)$$

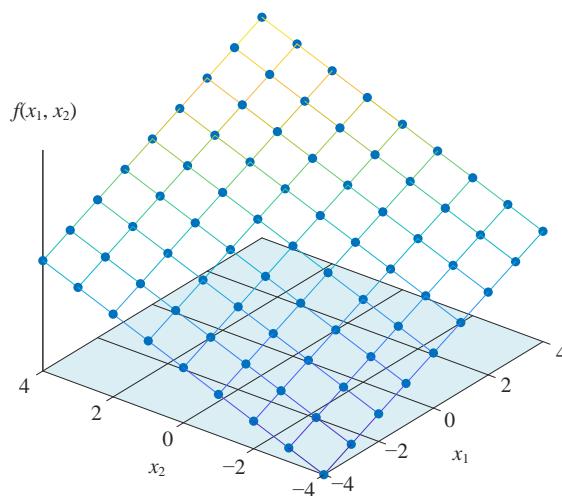
式中， x_1 的每个值代表点的横坐标值， x_2 的每个值代表点的纵坐标值。`numpy.meshgrid()` 可以用来获得网格化数据。

⚠ 注意 上式中 x_1 和 x_2 仅仅是示意，本书矩阵一般记号都是大写字母、粗体、斜体，比如 \mathbf{A} 、 \mathbf{V} 、 \mathbf{X} 等。

$y = f(x_1, x_2) = x_1 + x_2$ 这个二元函数便是将 (11) 相同位置的数值相加得到函数值 $f(x_1, x_2)$ 的矩阵：

$$f(x_1, x_2) = x_1 + x_2 = \begin{bmatrix} -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \\ -4 & -2 & 0 & 2 & 4 \end{bmatrix} + \begin{bmatrix} -4 & -4 & -4 & -4 & -4 \\ -2 & -2 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 4 & 4 \end{bmatrix} = \begin{bmatrix} -8 & -6 & -4 & -2 & 0 \\ -6 & -4 & -2 & 0 & 2 \\ -4 & -2 & 0 & 2 & 4 \\ -2 & 0 & 2 & 4 & 6 \\ 0 & 2 & 4 & 6 & 8 \end{bmatrix} \quad (12)$$

图 19 所示为 $f(x_1, x_2) = x_1 + x_2$ 对应的三维空间平面。函数 $f()$ 则代表某种规则，将网格化数据从 x_1x_2 平面映射到三维空间。

图 19. $f(x_1, x_2) = x_1 + x_2$ 对应的三维空间平面

这就是前文说的，在绘制函数图像时，比如二元函数曲面，实际上输入的函数值都是离散的、网格化的。当然，网格越密，函数曲面越精确。

实际应用中，网格的疏密可以根据函数的复杂度调整。比如图 19 这幅平面图像很简单，因此可以用比较稀疏的网格来呈现图像；但是，对于比较复杂的函数，网格则需要设置的密一些，也就是步长小一些。

一个复杂曲面

下面看一个复杂二元函数 $f(x_1, x_2)$ 对应的曲面。

图 20 对应的函数解析式为：

$$f(x_1, x_2) = 3(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \exp(-x_1^2 - x_2^2) - \frac{1}{3} \exp(-(x_1+1)^2 - x_2^2) \quad (13)$$

相对图 19，图 20 的网格更为密集，这是为了更准确地观察分析这个比较复杂曲面的各种特征。本章后续有关二元函数的可视化方案，都是以上述二元函数作为例子。

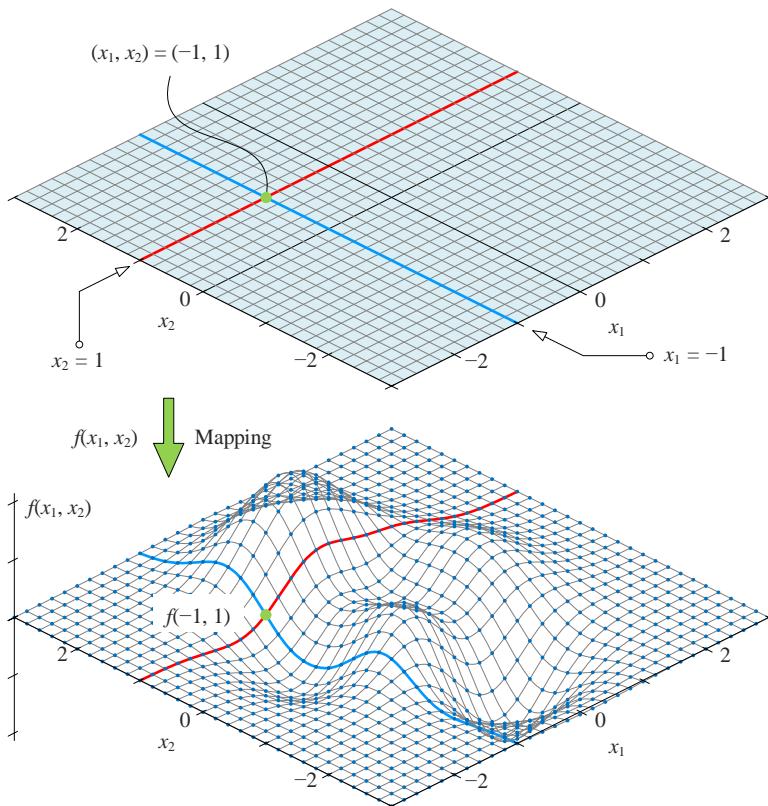


图 20. 网格化数据与二元函数映射



代码文件 `Bk3_Ch10_02.py` 中 `Bk3_Ch10_02_A` 部分绘制图 20 二元函数 $f(x_1, x_2)$ 对应的网格曲面。

10.5 降维：二元函数切一刀得到一元函数

如图 21 所示为二元函数两种可视化工具——剖面线、等高线。

本节介绍剖面线，它相当于在曲面上沿着横轴或纵轴切一刀。我们关注的是截面处曲线的变化趋势，“切一刀”这个过程相当于降维。

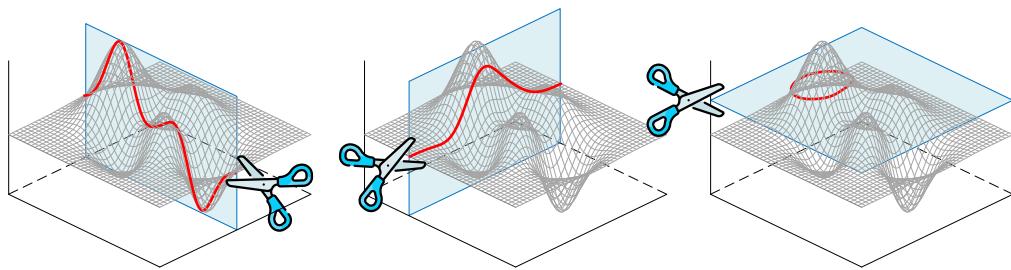


图 21. 函数降维

x_1y 平面方向剖面线

以(13)所示 $f(x_1, x_2)$ 二元函数为例，如果自变量 x_2 固定在 $x_2 = c$ ，只有自变量 x_1 变化， $f(x_1, x_2 = c)$ 则相当于是 x_1 的一元函数。

图 22 中彩色曲线所示为 x_2 固定在几个具体值 c 时， $f(x_1, x_2 = c)$ 随 x_1 变化的剖面线。这些剖面线就是一元函数。

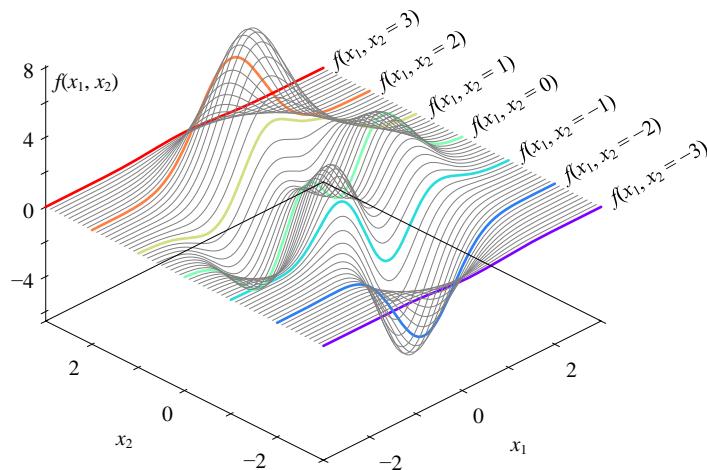
利用一元函数性质，我们可以分析曲面在不同位置的变化趋势。

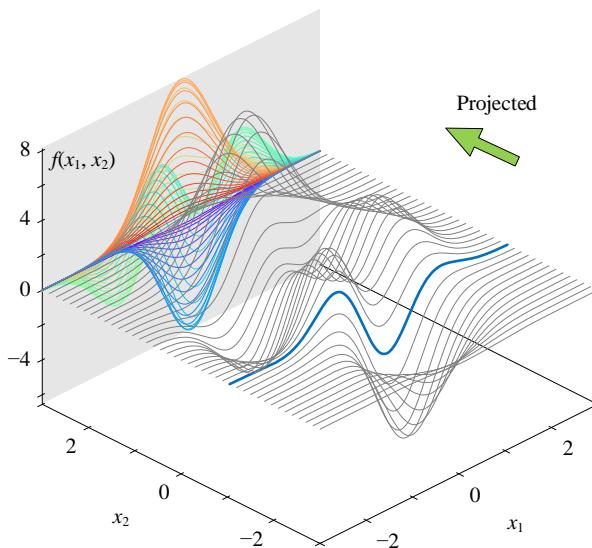
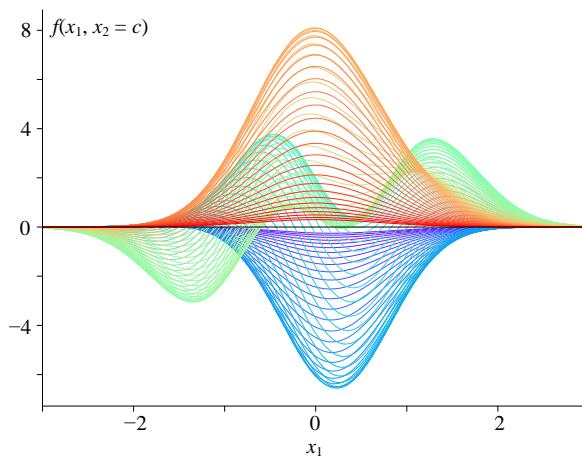
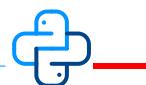
如图 23 所示，将一系列 $f(x_1, x_2 = c)$ 剖面线投影在 x_1y 平面上，给每条曲线涂上不同颜色，可以得到图 24。



本书第 16 章要介绍的偏导数 (partial derivative) 就是研究这些剖面线的变化率的数学工具。

⚠ 注意，通过剖面线得出的“局部”结论不能推广到整个二元函数。

图 22. 自变量 x_2 固定，自变量 x_1 变化

图 23. 将 $f(x_1, x_2)$ 剖面线投影到 x_1y 平面图 24. 剖面线在 x_1y 平面上投影

代码文件 Bk3_Ch10_02.py 中 Bk3_Ch10_02_B 部分绘制图 22、图 23、图 24。

x_2y 平面方向剖面线

自变量 x_1 固定，只有自变量 x_2 变化， $f(x_1, x_2)$ 则相当于是 x_2 的一元函数。图 25 中彩色曲线所示为 x_1 固定在具体值 c 时， $f(x_1 = c, x_2)$ 随 x_2 变化。

如图 26 所示，将 $f(x_1 = c, x_2)$ 剖面线投影在 x_2y 平面上。给每条曲线涂上不同颜色，可以得到图 27。

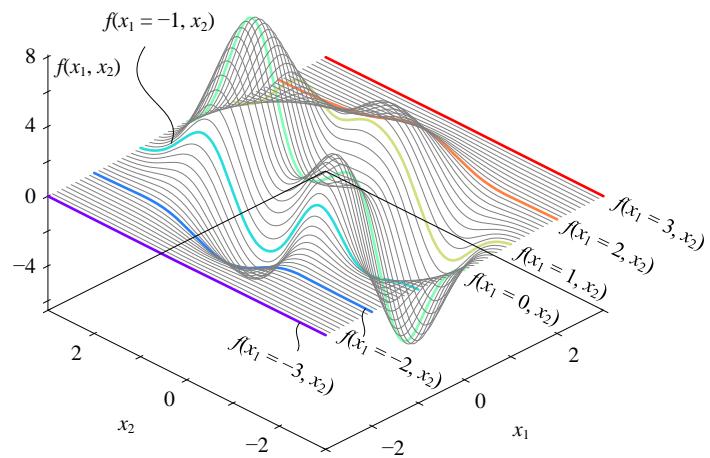


图 25. 自变量 x_1 固定, 自变量 x_2 变化

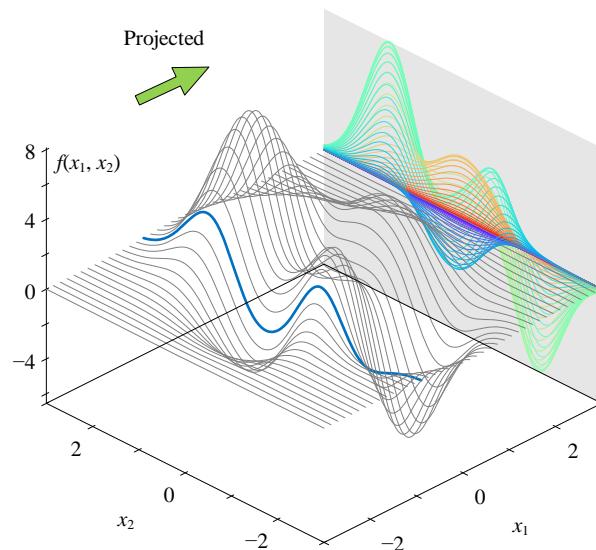
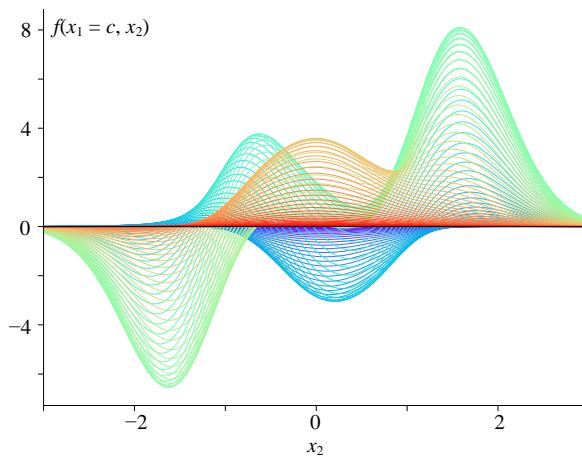
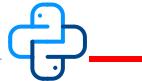


图 26. 将 $f(x_1, x_2)$ 剖面线投影到 x_2y 平面

图 27. 剖面线在 x_2y 平面投影

代码文件 Bk3_Ch10_02.py 中 Bk3_Ch10_02_C 部分绘制图 25、图 26、图 27。

10.6 等高线：由函数值相等点连成

把图 28 所示 $f(x_1, x_2)$ 曲面比作一座山峰，函数值越大，相当于山峰越高。图中用暖色色块表达山峰，用冷色色块表达山谷。

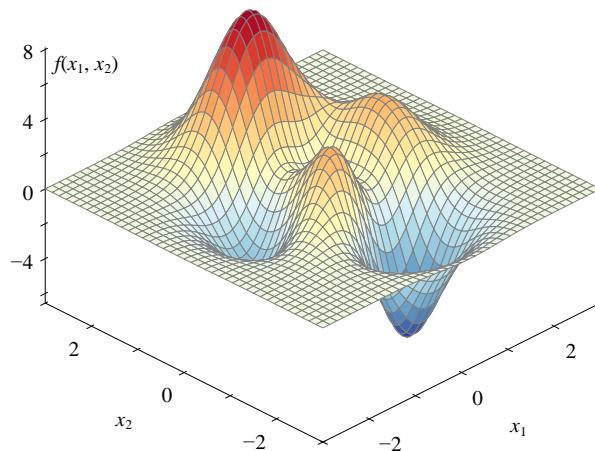


图 28. 用冷暖色表示函数的不同高度取值

等高线

三维等高线 (contour line) 和平面等高线是研究二元函数重要的手段之一。上一章在讲不等式时，我们简单提过等高线。简单来说，曲面某一条等高线就是函数值 $f(x_1, x_2)$ 相同，即 $f(x_1, x_2) = c$ 的相邻点连接构成的曲线。

当 c 取不同值时，便可以得到一系列对应不同高度的等高线，获得的图像便是三维等高线图，如图 29 所示彩色线。这些曲线可以是闭合曲线，也可以非闭合。

将这些曲线垂直投影到水平面上，得到平面等高线图，如图 30 所示。

生活中，等高线有很多其他形式，比如等温线、等压线、等降水线等等。

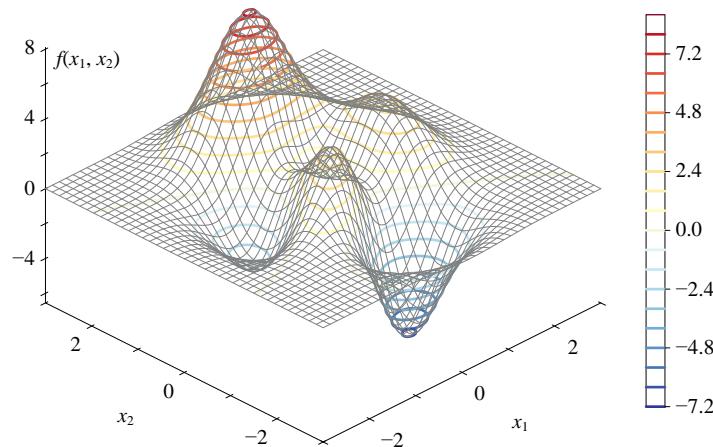


图 29. 二元函数三维等高线

图 30 所示 $f(x_1, x_2)$ 三维等高线相当于图 29 在 x_1x_2 平面上的投影结果。平面等高线图中，每条不同颜色的曲线代表一个具体函数取值。把二元函数比作山峰的话，等高线越密集的区域，坡度越陡峭。相反，等高线越平缓的区域，坡面越平坦。



本书第 16 章将介绍的偏导数这个数学工具可以用来量化“陡峭”和“平坦”。

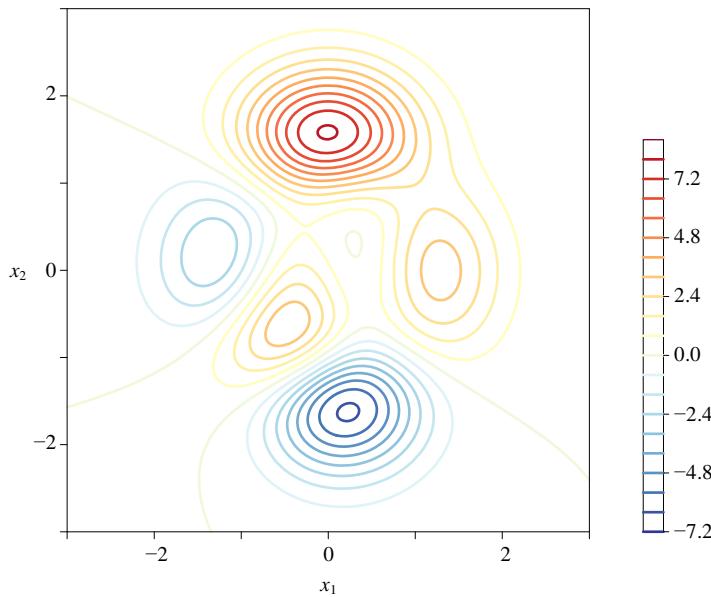


图 30. 二元函数的平面等高线

填充等高线

本书还常用填充等高线来可视化二元函数。

图 31 所示为 $f(x_1, x_2)$ 三维坐标系中在 x_1x_2 平面上得到平面填充等高线。图 32 就是填充等高线在 x_1x_2 平面上的投影结果。

⚠ 注意， 在填充等高线图中，同一个颜色色块代表函数范围在某一特定区间内 $[c_i, c_{i+1}]$ 。

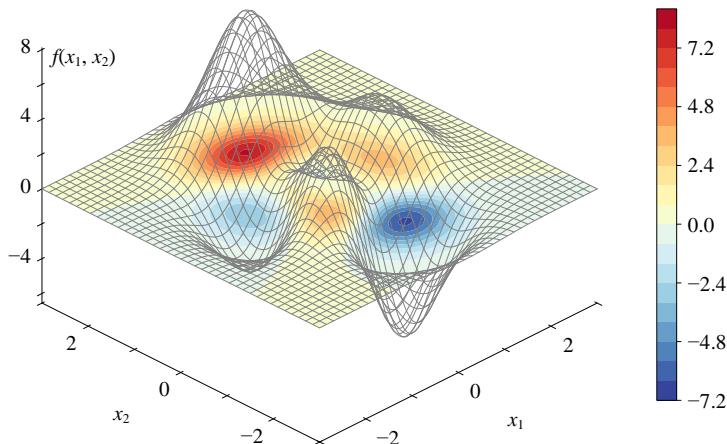


图 31. 三维曲面投影在水平面上得到平面填充等高线

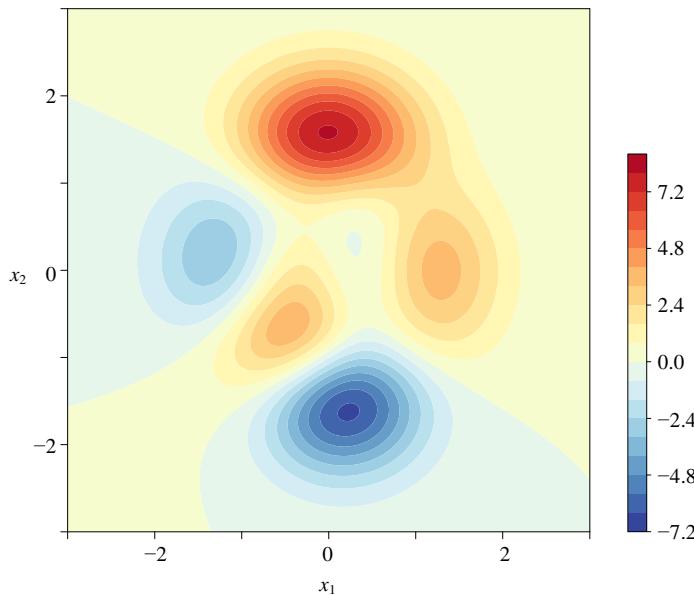


图 32. 平面填充等高线图



代码文件 Bk3_Ch10_02.py 中 Bk3_Ch10_02_D 部分绘制图 28 ~ 图 32。



在 Bk3_Ch10_02.py 基础上，我们做了一个 App 用来交互呈现曲面特征。请大家参考代码文件 Streamlit_Bk3_Ch10_02.py。这个代码有个特别之处，我们用了 Plotly 库中的 3D 交互绘图函数。



没有坐标系，就没有函数。坐标系给函数以生命。希望大家，在学习任何函数时，首先想到的是求助于坐标系。

特别强调，在描绘函数形状和变化趋势时，千万不能按自己审美偏好“手绘”函数图像！哪怕技艺精湛，手绘函数也不能准确描绘函数的每一处细节。

函数图像必须通过编写代码可视化！而且得到的曲线，千万不要“手动”改变某点取值，否则篡改数据！

即便是编程绘制的图像也不是百分之百准确无误，因为这些图像是散点连接而成的。只不过当这些点和点之间步长较小时，图像看上去连续光滑罢了。

作者在很多数学教科书中，看到很多不负责任的“手绘”函数图像。作者本人特别不能容忍“手绘”高斯函数或高斯分布概率密度函数曲线，这简直就是暴殄天物！

11

Algebraic Functions

代数函数

自变量有限次加、减、乘、除、有理指数幂和开方



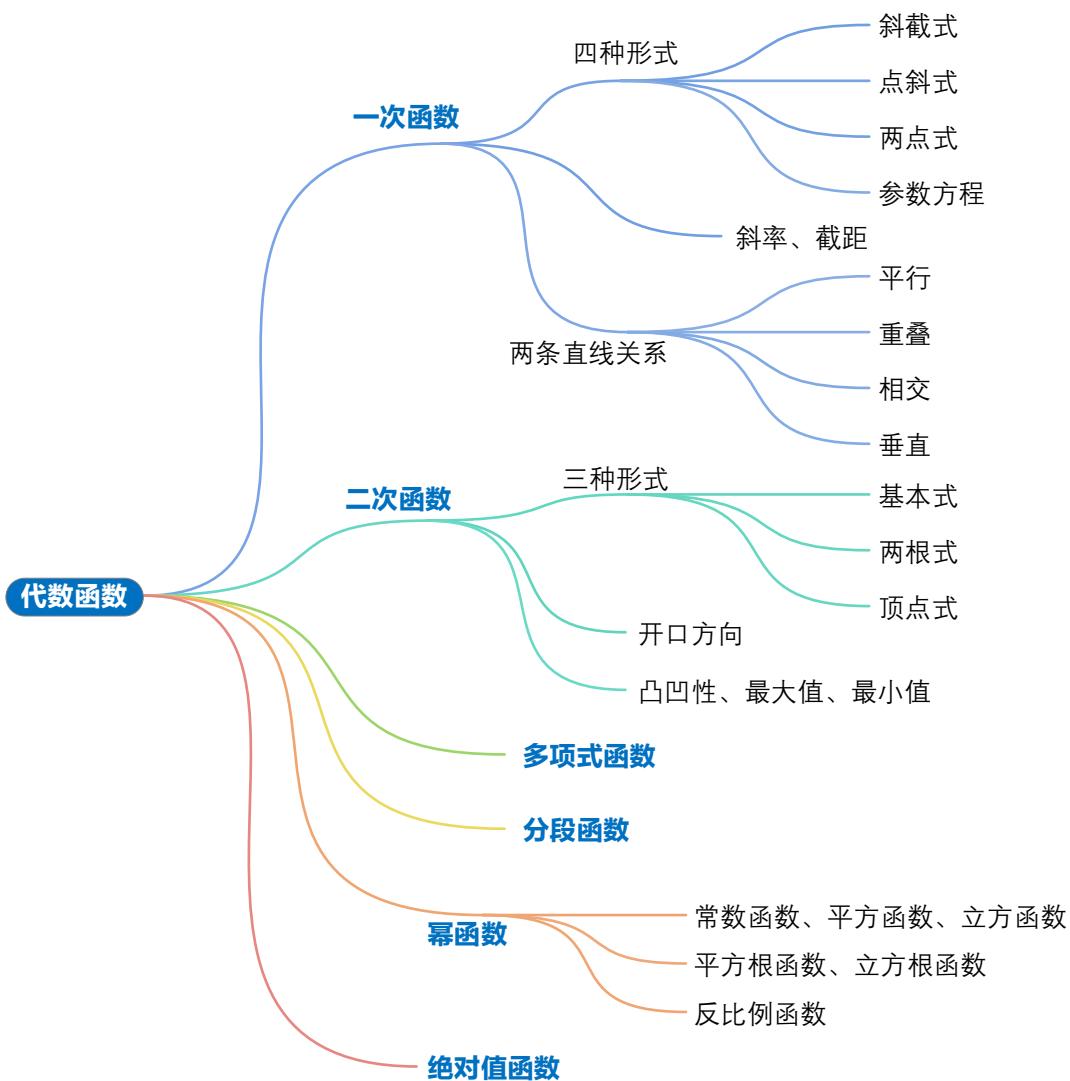
数学不分种族、不分地域；对于数学来说，其文化世界自成一国。

Mathematics knows no races or geographical boundaries; for mathematics, the cultural world is one country.

—— 大卫·希尔伯特 (David Hilbert) | 德国数学家 | 1862 ~ 1943



- ◀ matplotlib.pyplot.axhline() 绘制水平线
- ◀ matplotlib.pyplot.axvline() 绘制竖直线
- ◀ matplotlib.pyplot.contour() 绘制平面等高线
- ◀ matplotlib.pyplot.contourf() 绘制平面填充等高线
- ◀ matplotlib.pyplot.grid() 绘制网格
- ◀ matplotlib.pyplot.plot() 绘制线图
- ◀ matplotlib.pyplot.show() 显示图片
- ◀ matplotlib.pyplot.xlabel() 设定 x 轴标题
- ◀ matplotlib.pyplot.ylabel() 设定 y 轴标题
- ◀ numpy.absolute() 计算绝对值
- ◀ numpy.array() 创建 array 数据类型
- ◀ numpy.cbrt(x) 计算立方根
- ◀ numpy.ceil() 计算向上取整
- ◀ numpy.floor() 计算向下取整
- ◀ numpy.linspace() 产生连续均匀向量数值
- ◀ numpy.meshgrid() 创建网格化数据
- ◀ numpy.sqrt() 计算平方根



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

11.1 初等函数：数学模型的基础

大家在中学时代都接触过的初等函数是最朴实无华的数学模型，它们是复杂数学模型的基础。本节以二次函数为例介绍如何利用初等函数进行数学建模。

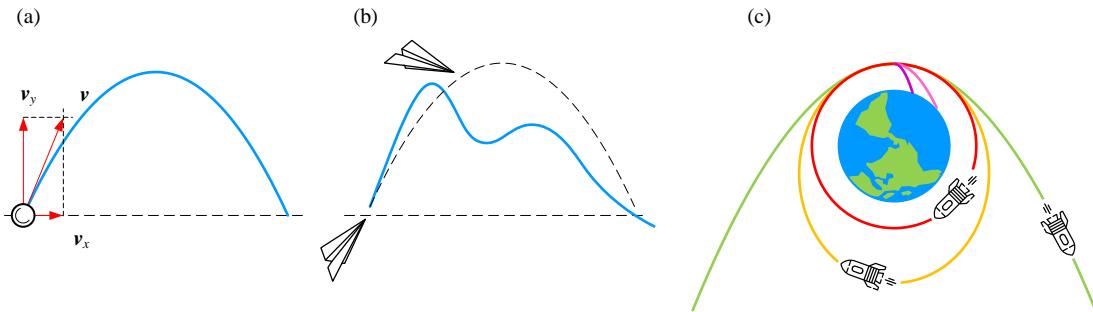


图 1. 抛物线

如图 1 (a) 所示，斜向上方抛起一个小球，忽略空气阻力影响，小球在空中划出的一道曲线就可以用抛物线描述。这条抛物线就是二次函数。小球在空中不同时刻的位置，以及最终的落点，都可以通过二次函数这个模型计算得到。

同样的仰角，斜向上抛出一个纸飞机，纸飞机在空中的飞行轨迹就不得不考虑纸飞机外形、空气气流这些因素。如图 1 (b) 所示，抛物线已经不足以描述纸飞机的轨迹。

类似的，很多应用场景都需要对抛物线模型进行修正。比如，击打网球时，施加旋转可以改变网球飞行轨迹。射击时，枪管膛线让子弹旋转飞行，这必然会让其行进轨迹发生变化。发射炮弹时，空气阻力与炮弹外型和飞行速度有密切关系，这显然会影响炮弹飞行轨迹和落点。

此外，认为抛射物体轨迹为抛物线至少基于几个假设前提。比如，忽略空气阻力的影响；再比如，假设大地平坦；同时假设物体受到的地球引力垂直大地，如图 2 (a)。

准确地来说，抛射体受到的重力实际上是指向地心，如图 2 (b)。也就是说物体在空中飞行时，加速度朝着地球中心，它的轨迹实际上是椭圆的一部分。

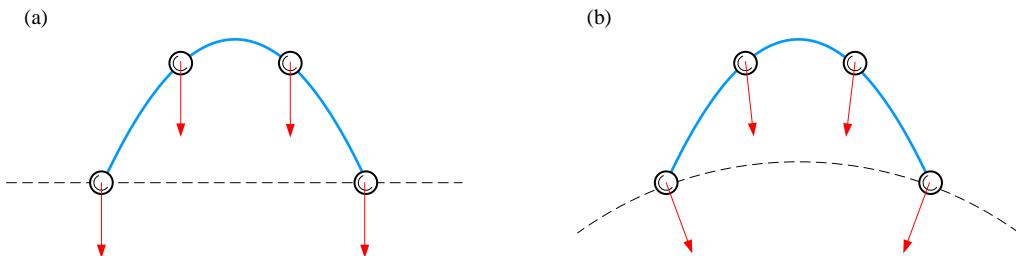


图 2. 引力方向

再进一步，远程炮弹飞行就需要考虑地心引力变化、地球自转；深空探测时，飞行器轨迹还需要考虑不同星体之间的引力作用，甚至来自太阳的光压等等因素。

假设前提是每个数学模型应用基础。数学模型毕竟是对现实世界各种现象的高度抽象概括，必须忽略一些次要因素，设定必要的假设前提，才能把握主要矛盾。

算力有限时，对抛射一个实心小球建模时，显然不会考虑小球的气动因素，更不会考虑引力场因素。但是，模拟不同击打技巧对网球飞行轨迹的影响，就不得不考虑网球旋转和空气流动这些因素。

模拟洲际导弹弹道时，气动布局、空气流体、地球自转、引力场等因素就不再是次要因素，必须考虑这些因素才能准确判断炮弹飞行轨迹以及落点。

人类在抛物线、空气动力学方面的进步，很大程度上来自于对弹道的研究。不得不承认，科学技术的确是把双刃剑，备战和战争有些时候是人类自然科学知识进步的加速器。

11.2 一次函数：一条斜线

四种形式

一次函数 (linear function) 有几种不同的形式构造：

- ◀ **斜截式** (slope-intercept form), 如图 3 (a) 所示；
- ◀ **点斜式** (point-slope form), 如图 3 (b) 所示；
- ◀ **两点式** (two-point form), 如图 3 (c) 所示；
- ◀ **参数方程** (parametric equation), 如图 3 (d) 所示。

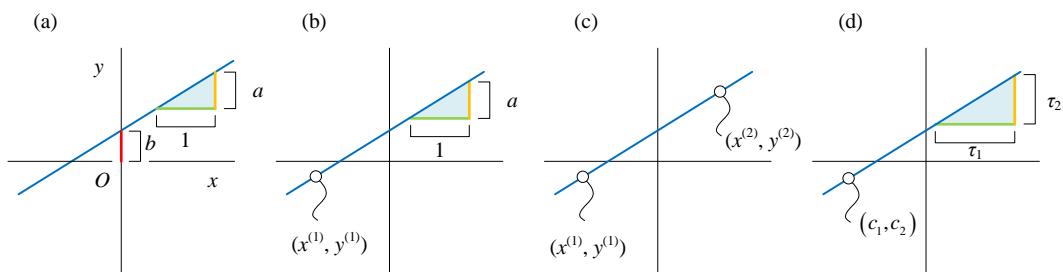


图 3. 一次函数的几种构造方法

斜截式

斜截式一次函数形式如下：

$$y = f(x) = ax + b \quad (1)$$

斜截式需要两个参数——斜率 (a) 和 y 轴截距 (b)。

⚠ 注意，对于一次函数，斜率 (a) 不能为 0。

当 $a = 0$ 时，(1) 为 **常数函数** (constant function)。也就是说，**零斜率** (zero slope) 对应常数函数，即**水平线** (horizontal line)。

无定义斜率 (undefined slope) 代表一条**竖直线** (vertical line)，此时图像虽然是一条直线，垂直于横轴，但它并不是函数。

对于(1)，当 $b = 0$ 时，函数为**比例函数** (proportional function)，而 a 则叫做**比例常数** (constant ratio 或 proportionality constant)。比例函数是特殊的一次函数。

→ 本书第 24 章将比较一次函数(含 y 轴截距)、比例函数(不含 y 轴截距)两种形式的线性回归模型。

一次函数有两种斜率：**正斜率** (positive slope) 和**负斜率** (negative slope)。图 4 (a) 所示一次函数斜率大于 0，单调递增。图 4 (b) 所示斜率小于 0，一次函数单调递减。

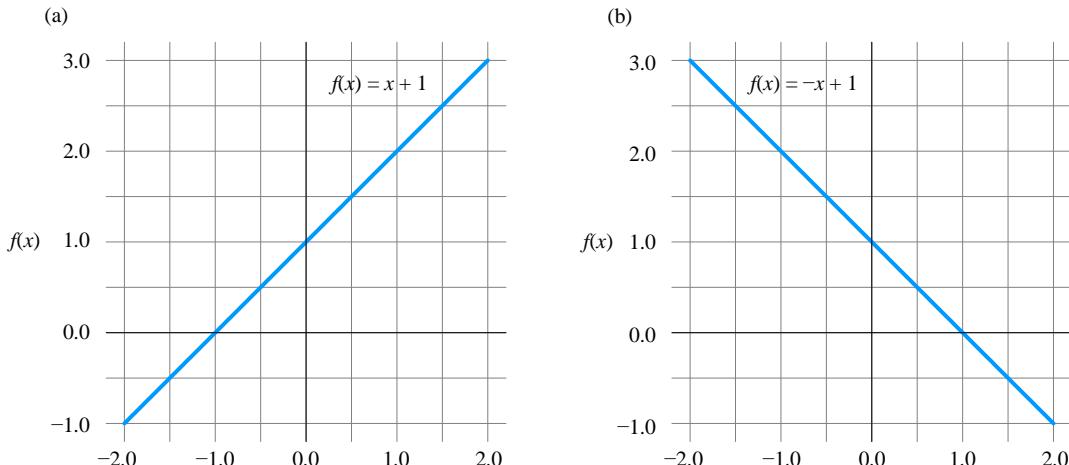


图 4. 一次函数

简单函数通过叠加和复合可以得到更复杂的函数，这是分析理解函数的重要视角。如图 5 所示， $f(x) = x + 1$ 可以看成是比例函数 $f_1(x) = x$ 和常数函数 $f_2(x) = 1$ 叠加得到。 $f(x) = -x + 1$ 可以看成是比例函数 $f_1(x) = -x$ 和常数函数 $f_2(x) = 1$ 叠加得到。

从几何视角来看，比例函数 $f_1(x) = x$ 图像沿 y 轴向上移动 1 个单位就得到 $f(x) = x + 1$ 。

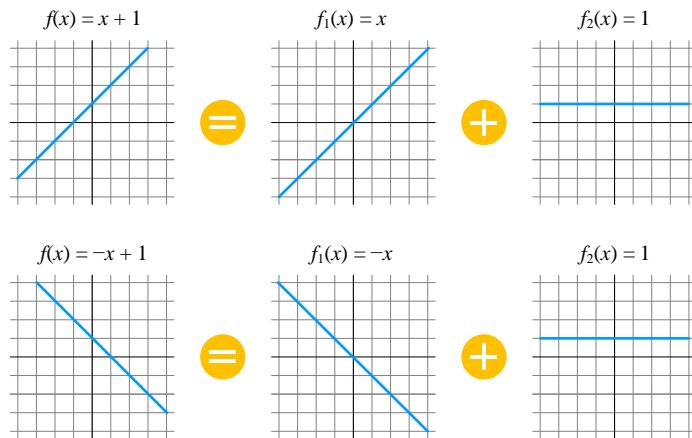
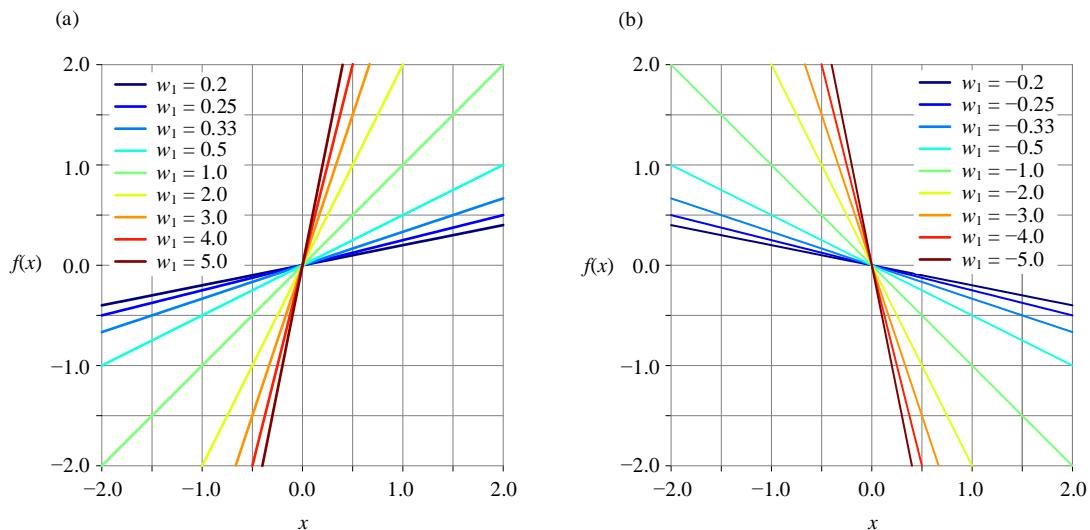


图 5. 函数叠加

斜率

图 6 所示为一次函数 $y = w_1x$ 随斜率变化，有些场合我们用 w_1 代表斜率。 w_1 的绝对值越大，一次函数图像越陡峭。

图 6. 一次函数 $y = w_1x$ 随斜率变化

截距

图 7 所示为一次函数随 y 轴截距变化情况。调整一次函数 y 轴截距大小，相当于图像上下平移。

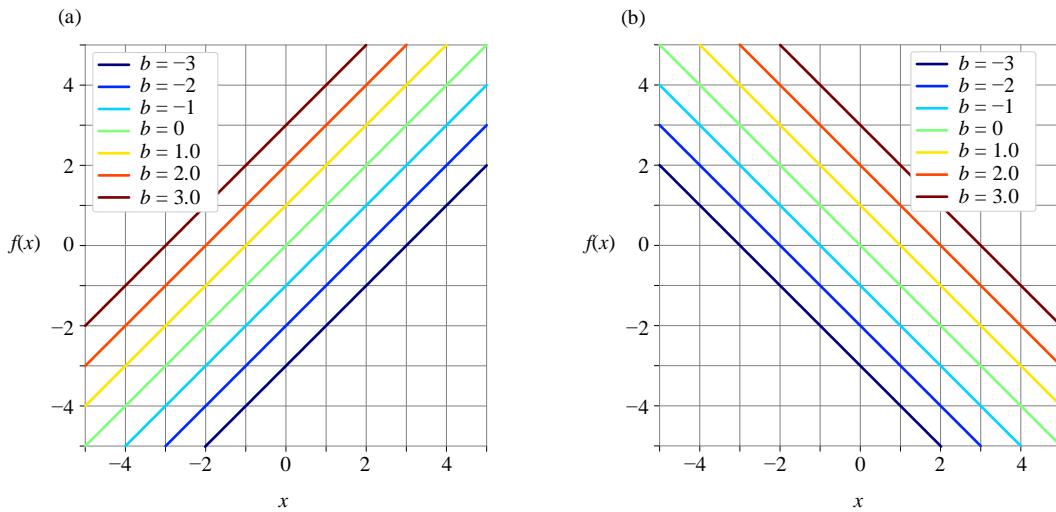


图 7. 一次函数随截距变化

两条直线关系

如果两条直线斜率相同，它们相互平行 (parallelize)，如图 8 (a)，或重合 (coincide)，如图 8 (b)。图 8 (c) 所示为两条直线相交 (intersect)，有唯一交点。

如果两个一次函数的斜率乘积为-1，则两者垂直 (perpendicular)，如图 8 (d)。

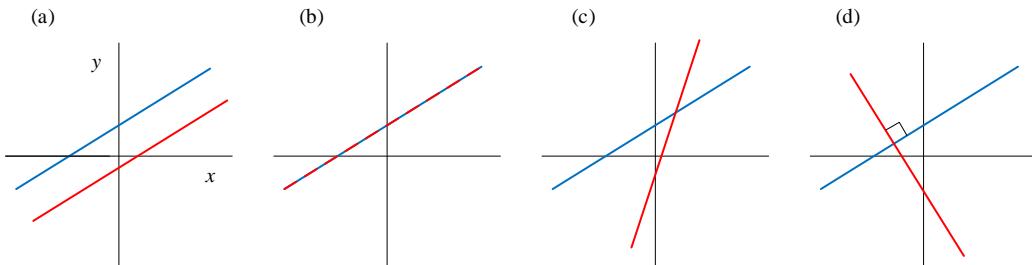


图 8. 两条之间的关系

点斜式

一次函数的第二种是点斜式：

$$y - y^{(1)} = f(x) - y^{(1)} = a(x - x^{(1)}) \quad (2)$$

也就是说，给定斜率 a 和直线上的一个点 $(x^{(1)}, y^{(1)})$ ，便可以确定平面上一条直线。

两点式

第三种形式是两点式，即两点确定一条直线。如下一次函数通过 $(x^{(1)}, y^{(1)})$ 和 $(x^{(2)}, y^{(2)})$ 两点：

$$y - y^{(1)} = \underbrace{\frac{y^{(2)} - y^{(1)}}{x^{(2)} - x^{(1)}}}_{\text{Slope}} (x - x^{(1)}) \quad (3)$$

其中， $x^{(1)} \neq x^{(2)}$ 。

两点式可以展开写成：

$$(y - y^{(1)})(x^{(2)} - x^{(1)}) = (y^{(2)} - y^{(1)})(x - x^{(1)}) \quad (4)$$



一次函数虽然看着简单，大家千万不要小瞧。数据科学和机器学习中很多算法都离不开一次函数，比如**简单线性回归** (Simple Linear Regression)。简单线性回归也叫一元线性回归模型，是指模型中只含有一个自变量和一个因变量，模型假设自变量和因变量之间存在线性关系。

人们常用有限的样本数据去探寻变量之间的规律，并以此作为分析或预测的工具。如图 9 所示，从给定的样本数据来看 x 和 y 似乎存在某种线性关系。通过一些算法，我们可以找到图 9 中那条红色斜线，它就是简单线性回归模型。而简单线性回归采用的解析式便是一元函数的斜截式。

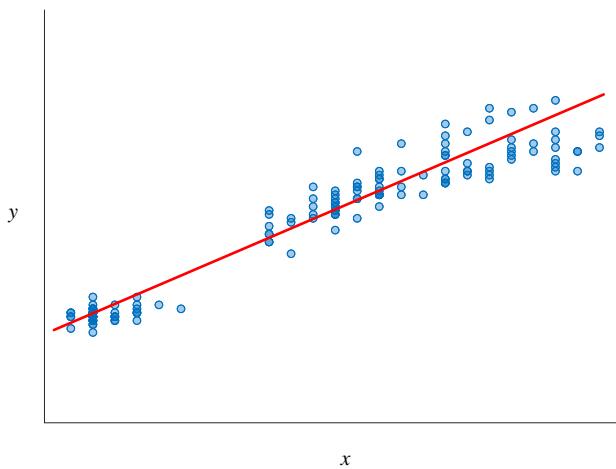
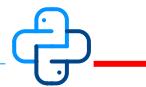


图 9. 简单线性回归



Bk3_Ch11_01.py 绘制图 6、图 7。代码中一元函数自变量 x 取值一般是等差数列。

`numpy.linspace(start, end, num)` 可以用来生成等差数列，数列 start 和 end 之间（包括 start 和 end 两个端点数值），数列的元素个数为 num 个，得到的结果数据类型为 array。

11.3 二次函数：一条抛物线

二次函数 (quadratic function) 是**二次多项式函数** (second order polynomial function 或 second degree polynomial function)。二次函数图象是**抛物线** (parabola)，可以**开口向上** (open upward) 或**开口向下** (open downward)，**对称轴平行于纵轴** (the axis of symmetry is parallel to the y-axis)。

三种形式

二次函数解析式有三种形式：

- ◀ **基本式** (standard form), 如图 10 (a);
- ◀ **两根式** (factored form), 如图 10 (b);
- ◀ **顶点式** (vertex form), 如图 10 (c)。

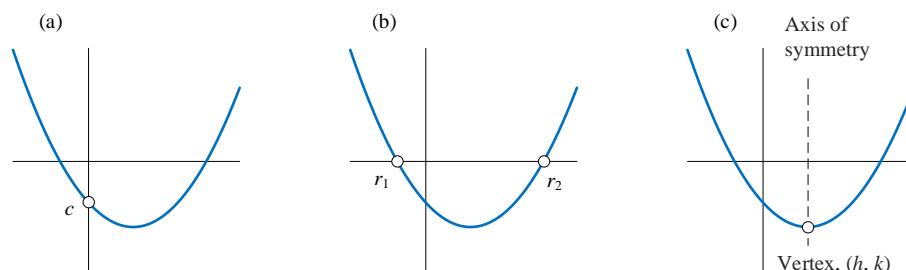


图 10. 二次函数的几种构造方法

基本式

二次函数的基本式如下：

$$f(x) = ax^2 + bx + c, \quad a \neq 0 \quad (5)$$

其中， a 被称作**二次项系数** (quadratic coefficient)，注意 a 不为零。 b 是**一次项系数** (linear coefficient)； c 叫**常数项** (constant term)，也叫 y 轴截距 (y -intercept)。

图 11 (a) 所示二次函数开口向上，**顶点** (vertex) 位于 y 轴，对称轴为 y 轴。图 11 (b) 所示二次函数开口向下。

图 11 (a) 二次函数为凸，顶点位置对应函数**最小值** (minimum)。图 11 (b) 二次函数为凹，顶点位置对应函数**最大值** (maximum)。

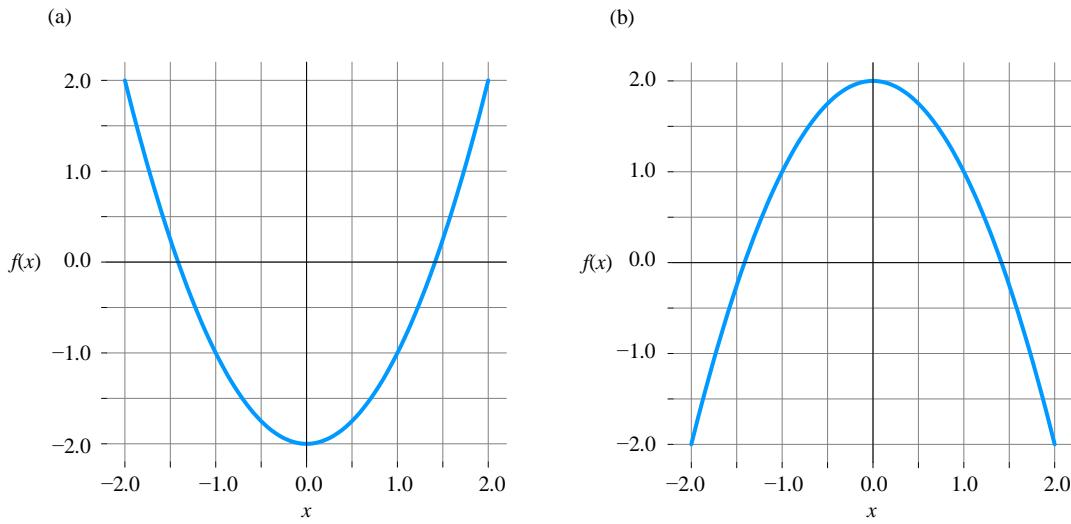


图 11. 不同开口方向二次函数



大家应该听过**极大值** (maxima 或 local maxima 或 relative maxima)、**极小值** (minima 或 local minima 或 relative minima)、**最大值** (maximum 或 global maximum 或 absolute maximum)、**最小值** (minimum 或 global minimum 或 absolute minimum) 等数学概念。

这里，我们用白话比较一下这几个概念，让大家有一个直观印象。

极大值和极小值统称**极值** (extrema 或 local extrema)，最大值和最小值统称**最值** (global extrema)。极值是就局部而言，而最值是整体来看。极值是局部的最大或最小值，而最值是整体的最大或最小值。

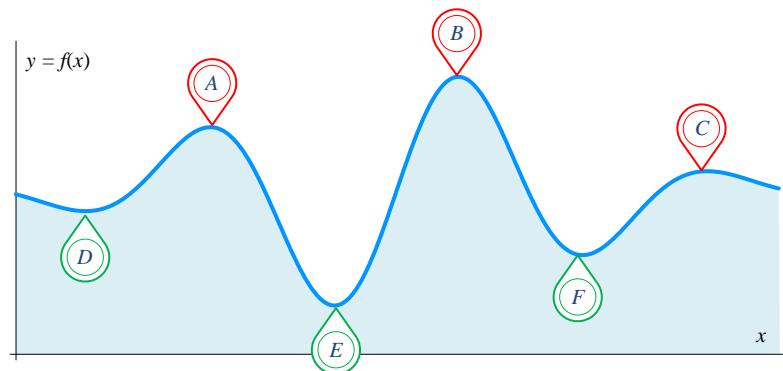


图 12. 极值和最值

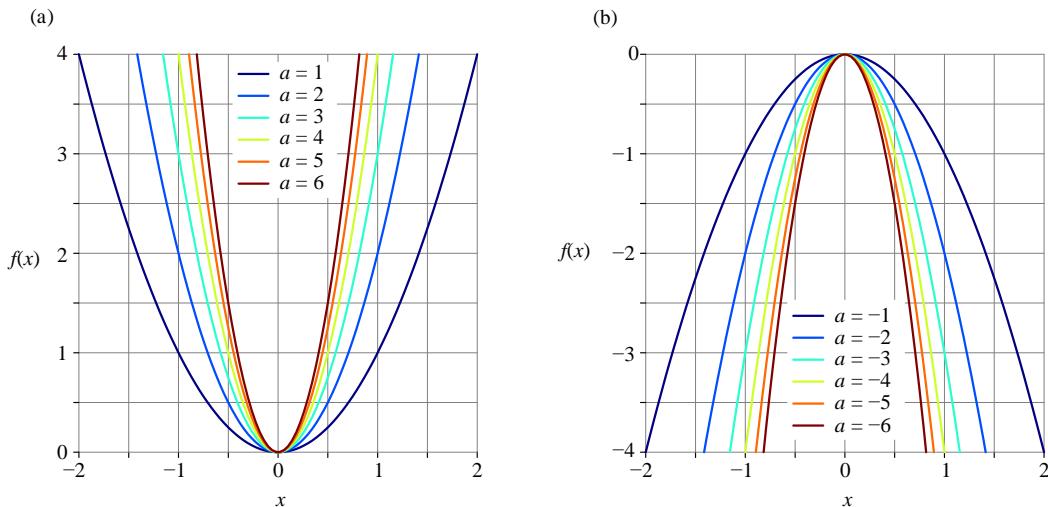
把图 12 函数图像看成一座山峰， A 、 B 、 C 、 D 、 E 、 F 都是极值，即山峰和山谷的总和。其中， A 、 B 、 C 为极大值，即山峰； D 、 E 、 F 为极小值，即山谷。

显然， B 是最高的山峰，也就是最大值，也叫全局最大值。而 E 是最低的山谷，也就是最小值，也叫全局最小值。

回过头来再看图 11，图 11(a) 中开口朝上抛物线的顶点对应最低的山谷，即全局最小值；图 11(b) 中开口朝下抛物线的顶点为最高的山峰，即全局最大值。

开口大小

图 13 所示为二次函数图像开口大小随系数 a 变化。 a 的绝对值越大，开口越小，对应二次函数图像变化越剧烈。

图 13. 二次函数随 a 变化

两根式

如果 $f(x) = 0$ 存在两个实数根的话，二次函数可以写成两根式：

$$f(x) = a(x - r_1)(x - r_2), \quad a \neq 0 \quad (6)$$

其中， r_1 和 r_2 为二次方程的根。

顶点式

二次函数另外一个常见的形式是顶点式，具体形式如下：

$$f(x) = a(x - h)^2 + k, \quad a \neq 0 \quad (7)$$

其中， h 和 k 分别为顶点的横纵坐标值。

图 14 (a) 所示为函数图像和 h 的关系，显然 h 影响函数在水平方向位置。图 14 (b) 所示为函数图像和 k 的关系， k 影响函数在竖直方向位置。

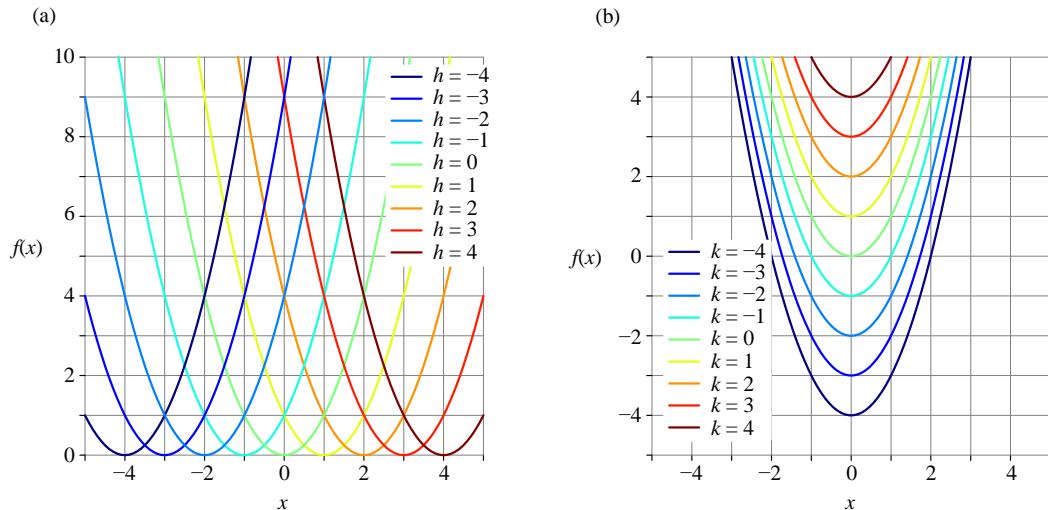
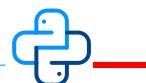


图 14. 二次函数随 h 和 k 变化

前文提到，二次函数顶点可以是函数的最大值或最小值。该顶点也是函数**单调性** (monotonicity) 的分水岭，即**拐点** (turning point)。以 h 为界，二次函数**单调区间** (intervals of monotonicity) 分别为 $(-\infty, h)$ 和 $(h, +\infty)$ 。



Bk3_Ch11_02.py 绘制图 13 和图 14。

11.4 多项式函数：从叠加角度来看

多项式函数 (polynomial function) 相当于一次和二次函数的推广，具体形式如下：

$$y = f(x) = a_K x^K + a_{K-1} x^{K-1} + \dots + a_2 x^2 + a_1 x + a_0 = \sum_{i=0}^K a_i x^i \quad (8)$$

其中，最高次项系数 a_K 不为 0， K 为最高次项次数。

图 15 几幅分图分别展示常数函数、一次到五次函数图像。可以这样理解，任何五次多项式函数都是图 15 所示的图像分别乘以相应系数叠加而成。

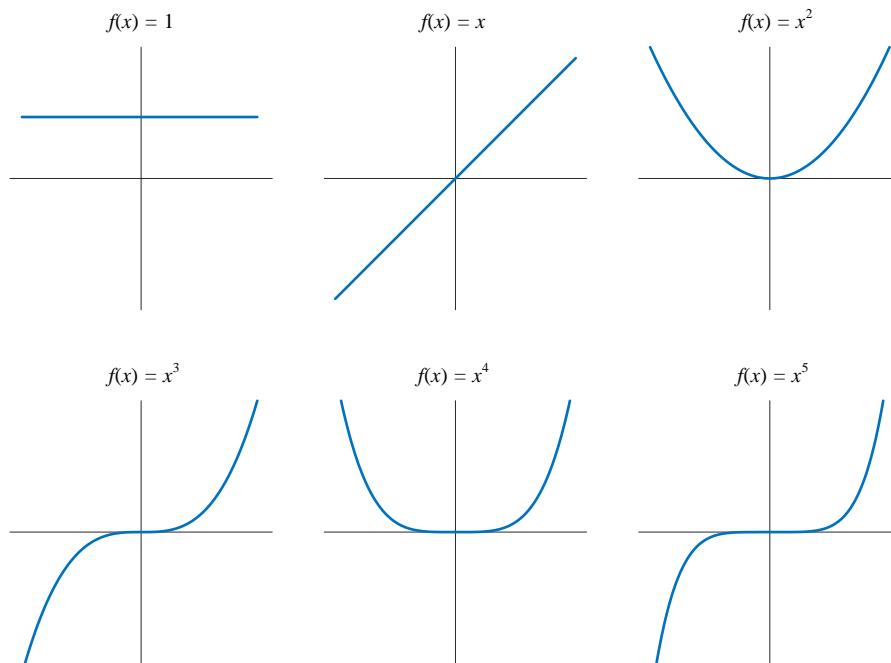


图 15. 常数函数到五次函数

三次函数

三次函数 (cubic function, polynomial function of degree 3) 的形式为：

$$y = f(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0 \quad (9)$$

举两个三次函数的例子：

$$\begin{aligned} y &= f(x) = x^3 - x \\ y &= f(x) = -x^3 + x \end{aligned} \quad (10)$$

这两个三次函数可以看做是 x^3 和 x 经过加减运算组合而成，如图 17 所示。

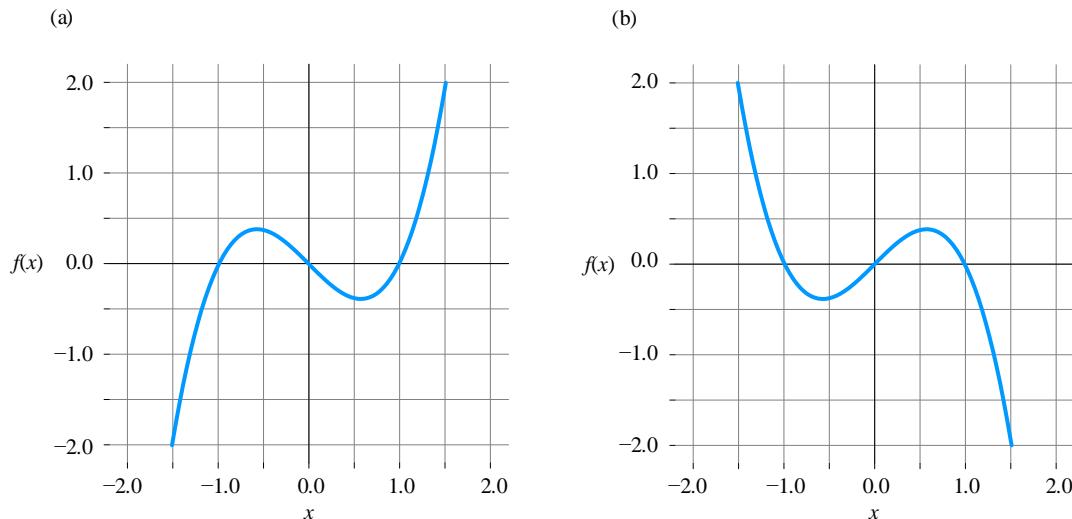


图 16. 两个三次函数

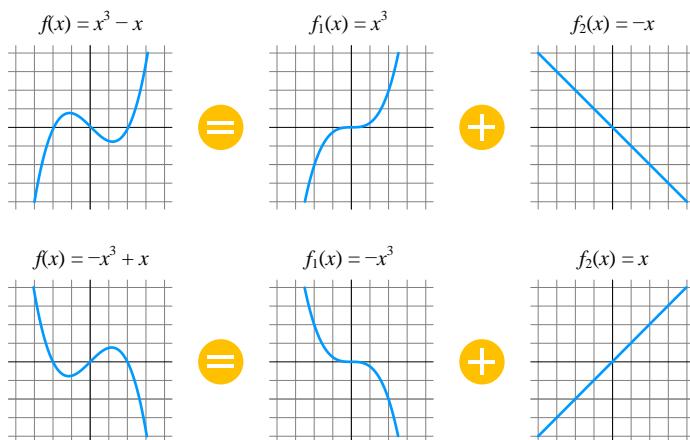


图 17. 函数叠加得到三次函数



前面讲过一次函数可以用在一元线性回归。线性回归虽然简单好用，但是并非万能。图 18 给出的数据具有明显的“非线性”特征，显然不适合用线性回归来描述。

多项式回归可以胜任很多非线性回归应用场合，多项式回归采用的数学模型就是多项式函数。

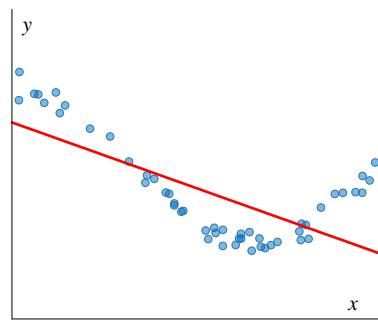


图 18. 线性回归失效的例子

图 19 (a)、(b)、(c) 比较三条拟合曲线，它们分别采用二次到四次一元多项式回归模型拟合样本数据。多项式回归的最大优点就是可以通过增加自变量次数，达到对数据更好的拟合；但是，对于多项式回归，自变量次数越高，越容易产生过度拟合 (overfitting) 问题，如图 19 (d)。

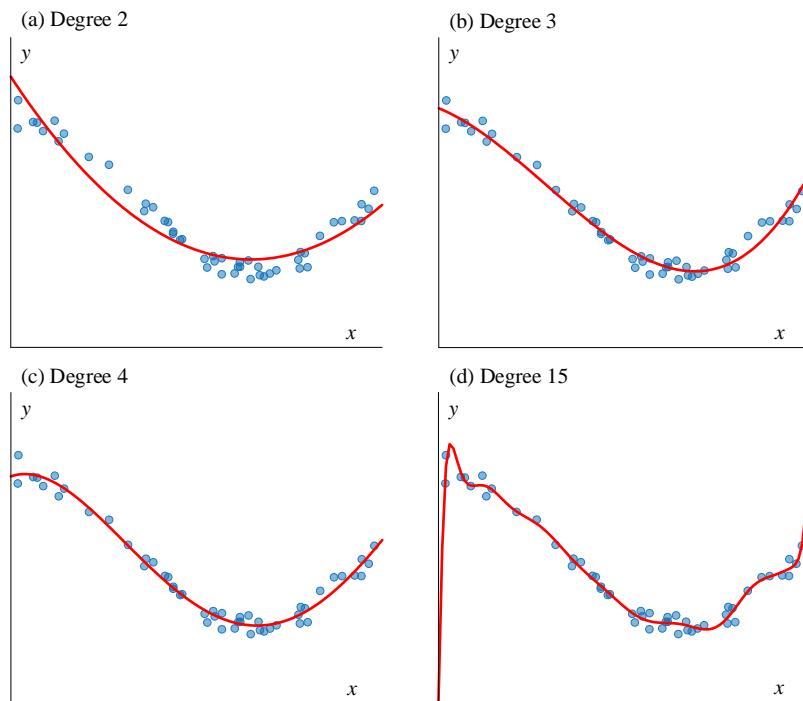
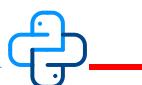


图 19. 逐渐增加多项式回归次数

使用过于复杂的模型是导致过拟合的重要原因之一。过拟合模型过度捕捉训练数据中的细节信息，甚至是噪音。但是，使用过拟合模型分析预测新样本数据时，往往结果较差。本系列丛书会在《数学科学》一册专门讲解多项式回归。



Bk3_Ch11_03.py 绘制图 4、图 11 和图 16。

11.5 幂函数：底数为自变量

幂函数 (power function) 是形如下式的函数：

$$f(x) = k \cdot x^p \quad (11)$$

其中，自变量 x 为**底数** (base)， p 为**指数** (exponent 或 power)。白话说，幂就是一个数和它自己相乘的积，比如， $xx = x^2$ 是二次幂， $xxx = x^3$ 是三次幂， $xxxx = x^4$ 是四次幂。

表 2 总结常用的幂函数，请大家关注不同函数的自变量取值范围。

表 1. 几个常用幂函数

幂函数	例子	图像
常数函数 (constant function)	$f(x) = 1 = x^0$	
恒等函数 (identity function)	$f(x) = x = x^1$	
平方函数 (square function)	$f(x) = x^2$	
立方函数 (cubic function)	$f(x) = x^3$	
反比例函数 (reciprocal function)	$f(x) = \frac{1}{x} = x^{-1}$ $x \neq 0$	
反比例平方函数 (reciprocal squared function)	$f(x) = \frac{1}{x^2} = x^{-2}$ $x \neq 0$	

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

平方根函数 (square root function)	$f(x) = \sqrt{x} = x^{\frac{1}{2}}$ $x \geq 0$	
立方根函数 (cubic root function)	$f(x) = \sqrt[3]{x} = x^{\frac{1}{3}}$	

平方根函数

图 20 (a) 所示红色曲线为**平方根函数** (square root function), 对应函数式为:

$$y = f(x) = \sqrt{x} = x^{\frac{1}{2}} \quad (12)$$

⚠ 注意上式函数定义域 $x \geq 0$, 即非负实数; 函数值域也是非负实数。

`numpy.sqrt(x)` 可以用来计算平方根, 也可以用 `x**(1/2)` 来计算。

图 20 (a) 还比较了平方根函数和二次函数, 两个函数的定义域显然不同。

立方根函数

图 20 (b) 所示红色曲线为**立方根函数** (cubic root function), 对应函数式为:

$$y = f(x) = \sqrt[3]{x} = x^{\frac{1}{3}} \quad (13)$$

`numpy.cbrt(x)` 可以用来计算立方根。Python 中 `x**(1/3)` 不可以计算负数立方根。

图 20 (b) 还比较了立方根函数和三次函数 $f(x) = x^3$, 两者互为反函数。

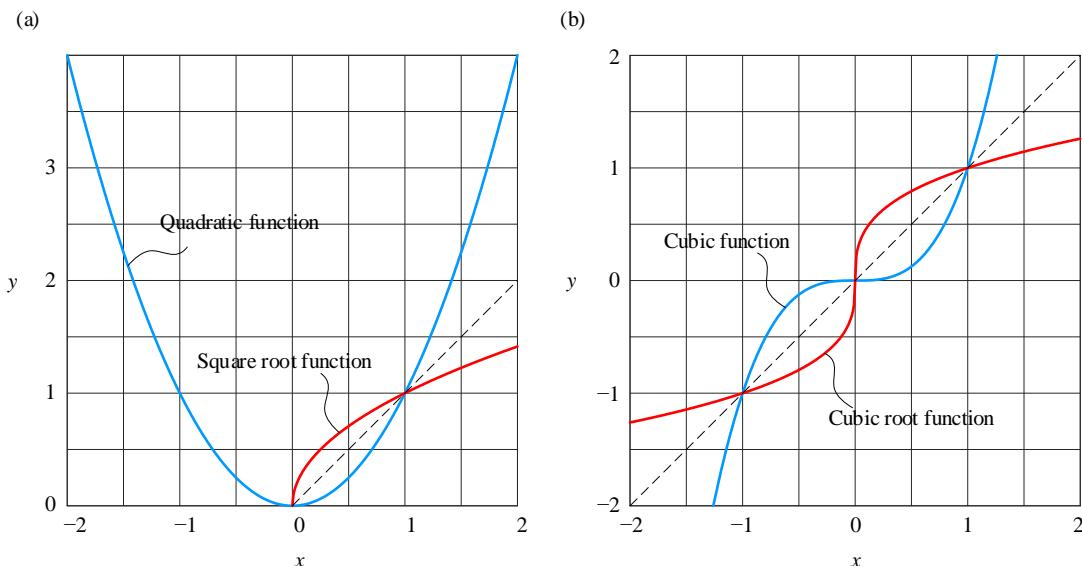


图 20. 平方根和立方根函数

奇偶性

如图 21 (a) 所示，当 p 为偶数时，幂函数为偶函数，图像关于 y 轴对称。 p 值越大， x 绝对值增大时，函数值越快速接近正无穷。

如图 21 (b) 所示，当 p 为奇数时，幂函数为奇函数，图像关于原点对称。 p 值越大， x 绝对值增大时，函数值越快速接近正无穷或负无穷。

此外，图 21 两幅图中所有函数可以写作 $f(x) = x^p$ ，所有曲线都经过 $(1, 1)$ 。

请大家修改前文代码自行绘制图 21。

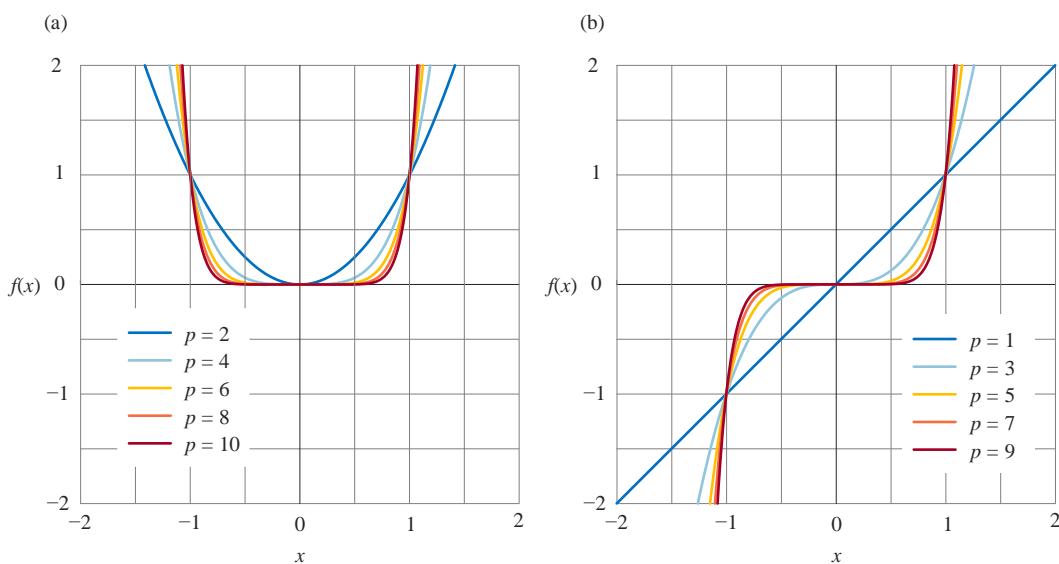
图 21. 幂函数 p 分别为偶数和奇数时，图像特征

表 2. 用英文读乘幂

数学表达	英文表达
x^n	x to the n x to the n -th x to the n -th power the n -th power of b x raised to the n -th power x raised to the power of n x raised by the exponent of n
a^2	a squared the square of a a raised to the second power a to the second
a^3	a cubed the cube of a a to the third
2^5	the fifth power of 2 2 raised to the fifth power 2 to the power of 5 2 to the fifth power 2 to the fifth 2 to the five
$y = 2^x$	y equals 2 to the power of x
$\sqrt{2} = 2^{\frac{1}{2}}$	square root of two
$\sqrt[3]{2} = 2^{\frac{1}{3}}$	cube root of two cubic root of two
$\sqrt[2]{64} = 8$	The square root of sixty four is eight.
$\sqrt[3]{64} = 4$	The cube root of sixty four is four.
$\sqrt[6]{64} = 2$	The sixth root of sixty four is two.
$\sqrt[c]{a^b}$	c -th root of a raised to the b power

反比例函数

反比例函数 (inversely proportional function) 的一般式：

$$y = f(x) = \frac{k}{x} \quad (14)$$

如图 22 所示，和 $y = f(x) = 1/x$ 相比， $|k| > 1$ 时，双曲线朝远离原点方向拉伸； $|k| < 1$ 将双曲线向靠近原点方向压缩。

⚠ 注意，反比例函数实际上是旋转双曲线。

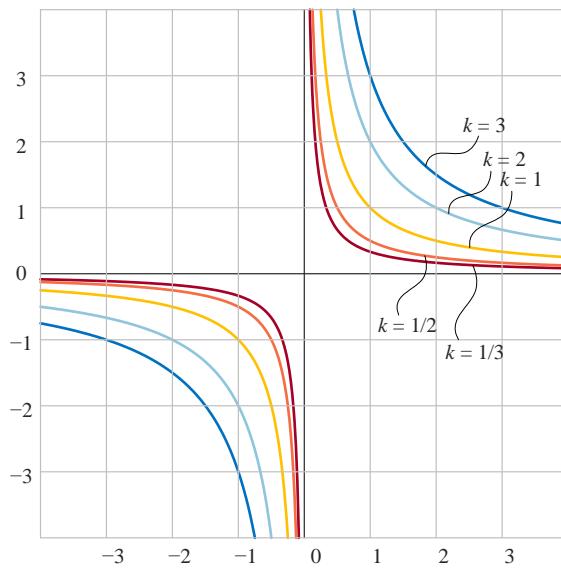
图 22. k 取不同值时反比例函数图像 $f(x) = k/x$

表 3. 用英文读比例函数

数学表达	英文表达
$y = \frac{k}{x}$	x is inversely proportional to y
$h = \frac{k}{t^2}$	h is inversely proportional to the square of t h varies inversely with the square of t

渐近线

如图 22 所示反比例函数有两条渐近线 (asymptote)，**水平渐近线** (horizontal asymptote) $y = 0$ 和**竖直渐近线** (vertical asymptote) $x = 0$ 。

所谓渐近线是指与曲线极限相关的一条直线，当曲线上某动点沿该曲线的一个分支移向无穷远时，动点到该渐近线的垂直距离趋于零。图 22 中，当 x 从右侧接近竖直渐近线，函数值无约束地接近**正无穷** (positive infinity)；相反，当 x 从左侧接近竖直渐近线，函数值无约束地接近**负无穷** (negative infinity)。

有理函数

反比例函数移动之后可以得到最简单的**有理函数** (rational function)，解析式如下：

$$f(x) = \frac{k}{x-h} + a \quad (15)$$

其中， $x \neq h$ 。

h 左右移动竖直渐近线， a 上下移动水平渐近线，比如下例：

$$f(x) = \frac{1}{x-1} + 1 \quad (16)$$

其中， $x \neq 1$ 。如图 23 所示， $y = 1$ 为 (16) 对应反比例函数的水平渐近线； $x = 1$ 为竖直渐近线。

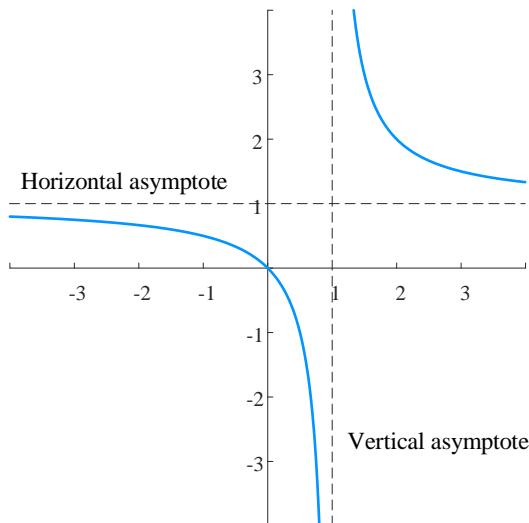


图 23. 反比例函数两条渐近线

11.6 分段函数：不连续函数

分段函数 (piecewise function) 是一类不连续函数；分段函数是自变量 x 的不同的取值范围有不同的解析式的函数。

⚠ 注意，分段函数不能算做代数函数。

图 24 对应下例分段函数：

$$f(x) = \begin{cases} 4 & x < -2 \\ -1 & -2 \leq x < 3 \\ 3 & 3 \leq x \end{cases} \quad (17)$$

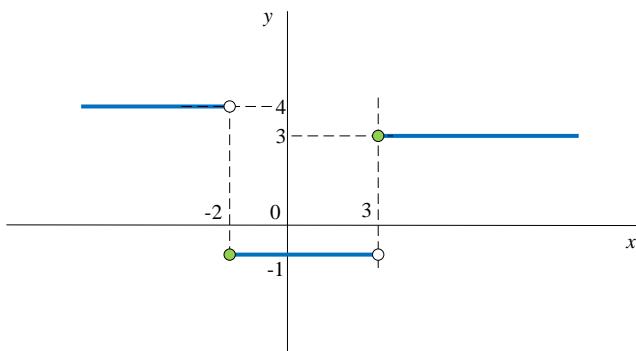
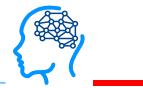


图 24. 分段函数



插值 (interpolation) 指的是通过已知离散数据点，在一定范围内推导求得新数据点的方法。**线性插值** (linear interpolation) 是指插值函数为一次函数。

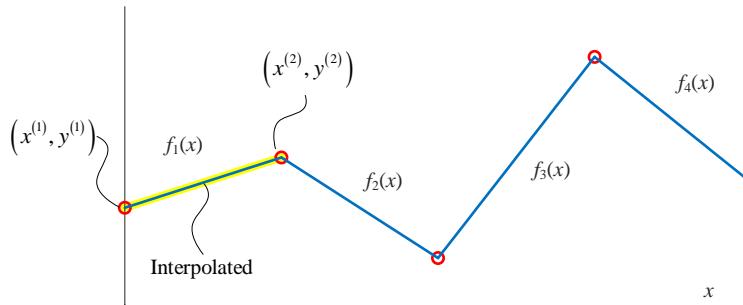


图 25. 一次函数两点式用于线性插值

插值函数是分段函数时，也称**分段插值** (piecewise interpolation)，每两个相邻的数据点之间便是一个分段函数：

$$f(x) = \begin{cases} f_1(x) & x^{(1)} \leq x < x^{(2)} \\ f_2(x) & x^{(2)} \leq x < x^{(3)} \\ \dots & \dots \\ f_{n-1}(x) & x^{(n-1)} \leq x < x^{(n)} \end{cases} \quad (18)$$

如图 25 所示，所有红色的圆点为已知离散数据点。

相邻两点连接得到的线段解析式便是线性插值分段函数。两点式公式常用在线性插值。举个例子，利用一次函数两点式，给定的两点 $(x^{(1)}, y^{(1)})$ 和 $(x^{(2)}, y^{(2)})$ 可以确定分段函数 $f_1(x)$ 。

比较图 9 和图 25，我们很容易发现回归和插值的明显区别。如图 9 所示，回归绝不要求红色线（模型）穿越所有样本点。而如图 25 所示，插值则要求分段函数穿越所有已知数据点。

除了线性插值之外，本系列丛书还要介绍其他常见插值方法。

绝对值函数

绝对值函数 (absolute value function) 可以看做是分段函数。绝对值函数的一般式为：

$$f(x) = k|x - h| + a \quad (19)$$

举个最简单的例子：

$$f(x) = k|x| \quad (20)$$

对于 $f(x) = k|x|$ 函数， $x = 0$ 为 $f(x)$ 的尖点，它破坏了函数的光滑。图 26 所示为 k 影响绝对值函数 $f(x) = k|x|$ 的开口大小； k 的绝对值越大，绝对值函数开口越小。

`numpy.absolute()` 计算绝对值。请大家自行编写代码绘制图 26，并讨论 k 为不同负整数时函数图像特点。

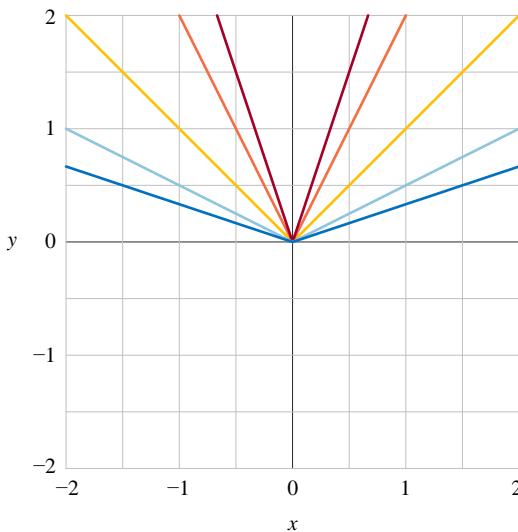
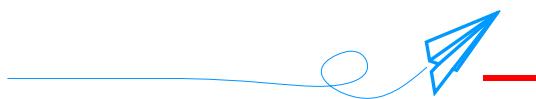


图 26. k 影响绝对值函数 $f(x) = k|x|$

严格来讲，绝对值函数不属于代数函数；但是，绝对值函数可以写成自变量的指数幂和开方形式。比如 (20) 可以写成：

$$f(x) = k\sqrt{x^2} \quad (21)$$



本章除了介绍几种常见的代数函数以外，还有一个要点——参数对函数形状、性质的影响。请大家思考这几个问题。

一次函数的斜率和截距，如何影响函数图像？

哪个参数影响二次函数的开口方向和大小？二次函数什么时候存在最大值或最小值？二次函数的对称轴位置？

用“叠加”这个思路，请大家想一下多项式函数 $f(x) = x^4 + 2x^3 - x^2 + 5$ 相当于由哪些函数构造而成？它们各自的函数图像分别怎样？

12

Transcendental Functions

超越函数

超出代数函数范围的函数



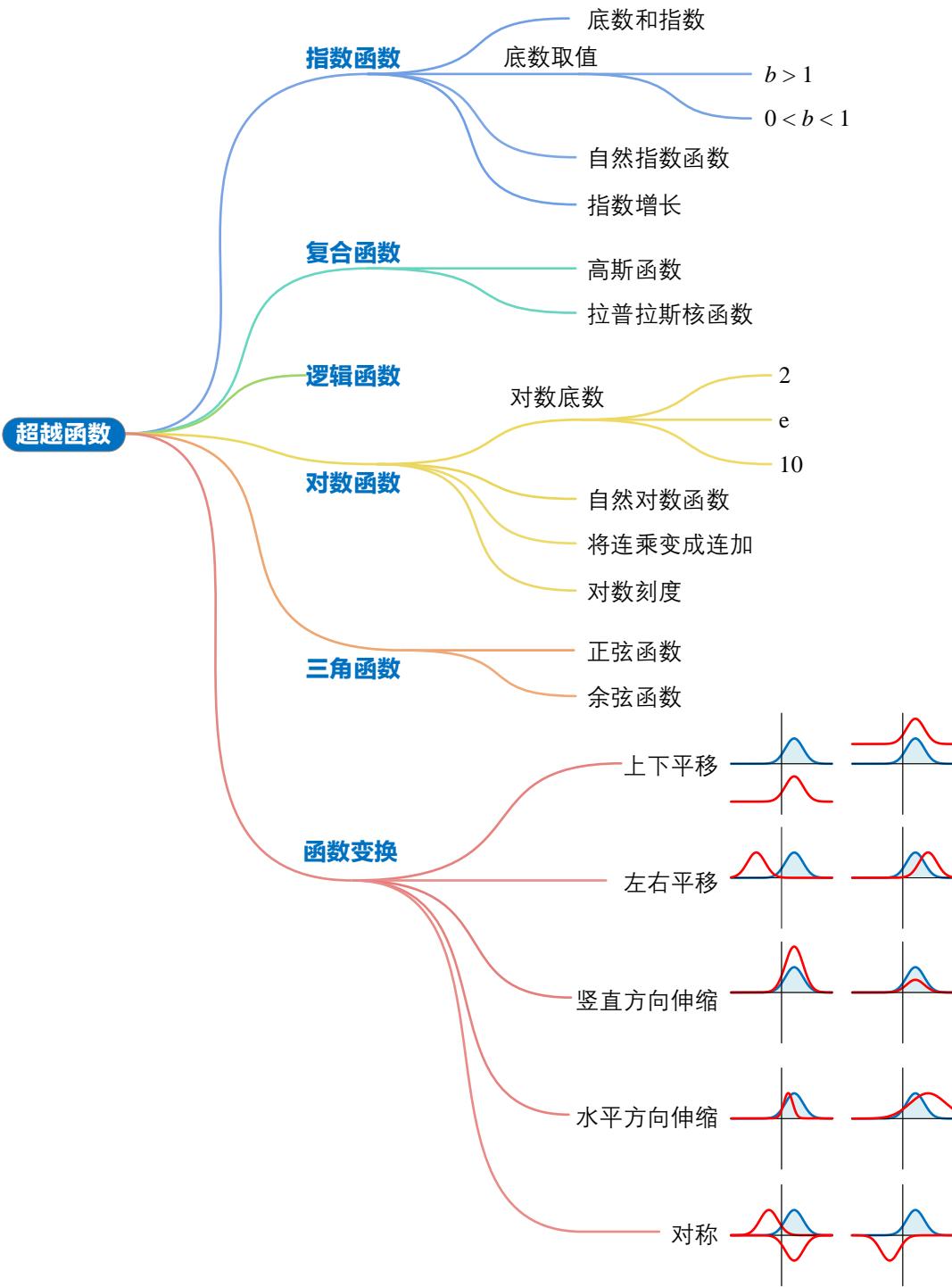
科学只不过是日常思维的提炼。

The whole of science is nothing more than a refinement of everyday thinking.

——阿尔伯特·爱因斯坦 (Albert Einstein) | 理论物理学家 | 1879 ~ 1955



- ◀ `matplotlib.pyplot.axhline()` 绘制水平线
- ◀ `matplotlib.pyplot.axvline()` 绘制竖直线
- ◀ `matplotlib.pyplot.contour()` 绘制平面等高线
- ◀ `matplotlib.pyplot.contourf()` 绘制平面填充等高线
- ◀ `matplotlib.pyplot.grid()` 绘制网格
- ◀ `matplotlib.pyplot.plot()` 绘制线图
- ◀ `matplotlib.pyplot.show()` 显示图片
- ◀ `matplotlib.pyplot.xlabel()` 设定 x 轴标题
- ◀ `matplotlib.pyplot.ylabel()` 设定 y 轴标题
- ◀ `numpy.absolute()` 计算绝对值
- ◀ `numpy.array()` 创建 array 数据类型
- ◀ `numpy.cbrt()` 计算立方根
- ◀ `numpy.ceil()` 计算向上取整
- ◀ `numpy.cos()` 计算余弦
- ◀ `numpy.floor()` 计算向下取整
- ◀ `numpy.linspace()` 产生连续均匀向量数值
- ◀ `numpy.log()` 底数为 e 自然对数函数
- ◀ `numpy.log10()` 底数为 10 对数函数
- ◀ `numpy.log2()` 底数为 2 对数函数
- ◀ `numpy.meshgrid()` 创建网格化数据
- ◀ `numpy.power()` 乘幂运算
- ◀ `numpy.sin()` 计算正弦
- ◀ `numpy.sqrt()` 计算平方根



12.1 指数函数：指数为自变量

指数函数 (exponential function) 的一般形式为：

$$f(x) = b^x \quad (1)$$

其中， b 为**底数** (base)，自变量 x 为**指数** (exponent)。

⚠ 注意，幂函数的自变量为底数，而指数函数的自变量为指数。

图 1 所示为当底数取不同值时指数函数图像，这几条曲线都经过 $(0, 1)$ 。

请大家区分底数 $b > 1$ 和 $0 < b < 1$ 两种情况对应的指数函数图像。 $b > 1$ 时， $f(x) = b^x$ 单调递增； $0 < b < 1$ 时， $f(x) = b^x$ 单调递减。

绘图时，可以用 `numpy.linspace()` 产生 x 数据，然后用 `b**x` 或 `numpy.power(b, x)` 计算指数函数值 b^x 。

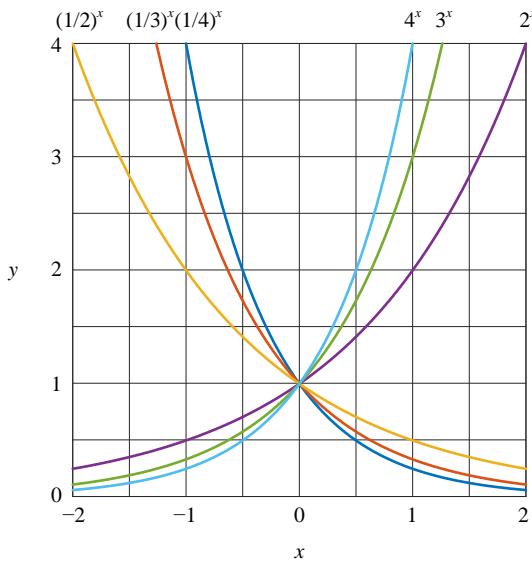


图 1. 不同底数的指数函数

自然指数函数

更多情况，指数函数指的是**自然指数函数** (natural exponential function)：

$$f(x) = e^x = \exp(x) \quad (2)$$

自然指数函数中的“自然”指的是自然常数 e 为底数， $e \approx 2.718$ 。

(1) 可以转化成以 e 为底数的函数：

$$y = f(x) = b^x = e^{\ln b \cdot x} = \exp(\ln b \cdot x) \quad (3)$$

表 1. 用英文读指数

数学表达	英文表达
e^x	e raised to the x th power e to the x e to the power of x exponential of x exponential x
$y = e^x$	y equals (is equal to) exponential x
$y = b^x$	y equals (is equal to) b to the x y equals (is equal to) b raised to the power of x
e^{x+y}	e to the quantity x plus y power e raised to the power of x plus y
$e^x + y$	the sum of e to the x and y e to the x power plus y
$e^x e^y$	the product of e to the x power and e to the y power
$e^x y$	the product of e to the x power and y e raised to the x power times y

指数增长

指数增长 (exponential growth) 模型就是用如下指数函数来表达：

$$G(t) = (1+r)^t \quad (4)$$

其中， r 为年化增长率， t 是年限。

当增长率 r 取不同值时，指数增长模型 $G(t)$ 和年限对应的关系如图 2 所示。**翻倍时间** (doubling time) 指的是当增长翻倍时所用的时间。图 2 中平行于横轴的虚线就是增长翻倍所对应的高度。

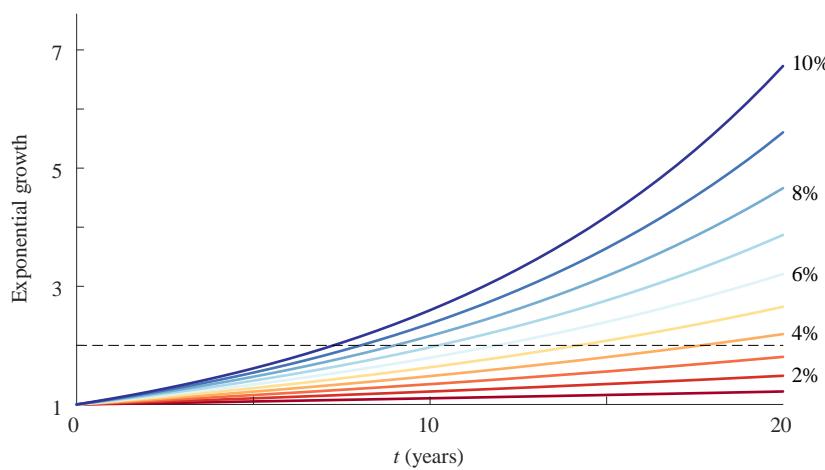


图 2. 指数增长

12.2 对数函数：把连乘变成连加

对数函数 (logarithmic function) 解析式如下：

$$y = f(x) = \log_b(x) \quad (5)$$

其中， b 为**对数底数** (logarithmic base)， $b > 0$ 且 $b \neq 1$ 。上述对数函数的定义域为 $x > 0$ 。

如图 3 所示， $b > 1$ 时， $f(x)$ 在定义域上为单调增函数。 $0 < b < 1$ 时， $f(x)$ 在定义域上为单调减函数。

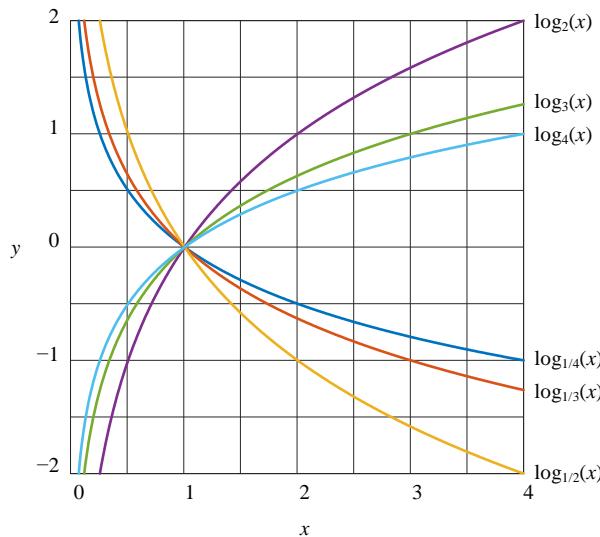


图 3. 不同底数的对数函数

自然对数函数

图 4 中蓝色曲线为**自然对数函数** (natural logarithmic function)，函数如下：

$$y = f(x) = \ln(x) = \log_e(x) \quad (6)$$

如图 4 所示，自然指数函数 (红色) 和自然对数函数 (蓝色) 互为反函数，两条曲线关于图中划线对称。

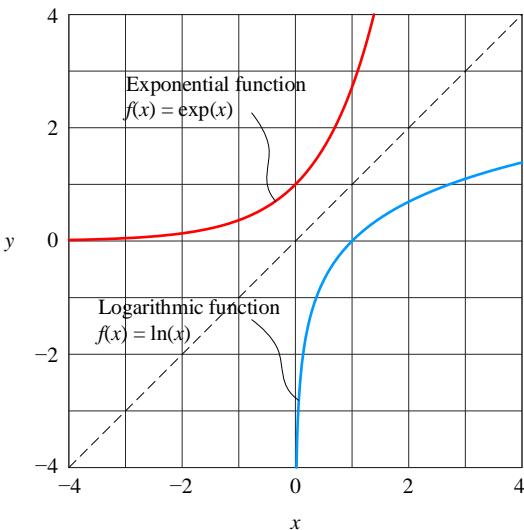


图 4. 自然对数函数和自然指数函数

Numpy 提供三个特殊底数对数函数运算：

- ◀ 底数为 2 对数函数 $\log_2(x)$, 函数为 `numpy.log2()`;
- ◀ 底数为 e 自然对数函数 $\ln(x)$, 函数为 `numpy.log()`;
- ◀ 底数为 10 对数函数 $\log_{10}(x)$, 函数为 `numpy.log10()`。

其他底数对数函数运算可以利用如下公式完成：

$$\log_b x = \frac{\ln(x)}{\ln(b)} \quad (7)$$

对数运算特点

请大家关注以下几个对数运算规则：

$$\begin{aligned} \log_b x &= \frac{\log_k x}{\log_k b} \\ \log_b x &= \frac{\log_{10} x}{\log_{10} b} = \frac{\ln x}{\ln b} \\ \log_b x^m &= \frac{m}{n} \log_b x \\ x &= b^{\log_b(x)} \\ x^{\log_b(y)} &= y^{\log_b(x)} \end{aligned} \quad (8)$$

对数的一个重要的性质是，把连乘变成连加：

$$\log_b(xyz) = \log_b x + \log_b y + \log_b z \quad (9)$$

我们会考虑用对数运算把连乘变成连加，是因为连乘不容易求偏导，而对连加求偏导则容易很多。特别地，高斯函数存在 $\exp()$ 项， $\ln()$ 可以把指数项变成求和形式。再者 $\ln()$ 不改变单调性。

在概率计算中，概率值累计乘积会出现数值非常小的正数情况，比如 $1e-30$ (10^{-30})。由于计算机的精度是有限的，无法识别这一类数据。而取对数之后，更易于计算机的识别。比如，对 $1e-30$ 以 10 为底取对数后得到 -30 。

对数刻度

对数刻度 (logarithmic scale 或 logarithmic axis) 是一种**非线性刻度** (nonlinear scale)，常用来描述较大的数值。图 5 (a) 的横轴和纵轴都是**线性刻度** (linear scale)，图中一元一次函数 $f(x) = x$ 为一条直线。图 5 (b) 的横轴为对数刻度，图中对数函数 $f(x) = \ln(x)$ 为一条直线。图 5 (c) 的纵轴为对数刻度，其中指数函数 $f(x) = 10^x$ 为一条直线。图 5 (d) 中，横轴、纵轴都是对数刻度，图中一元一次函数 $f(x) = x$ 还是一条直线。

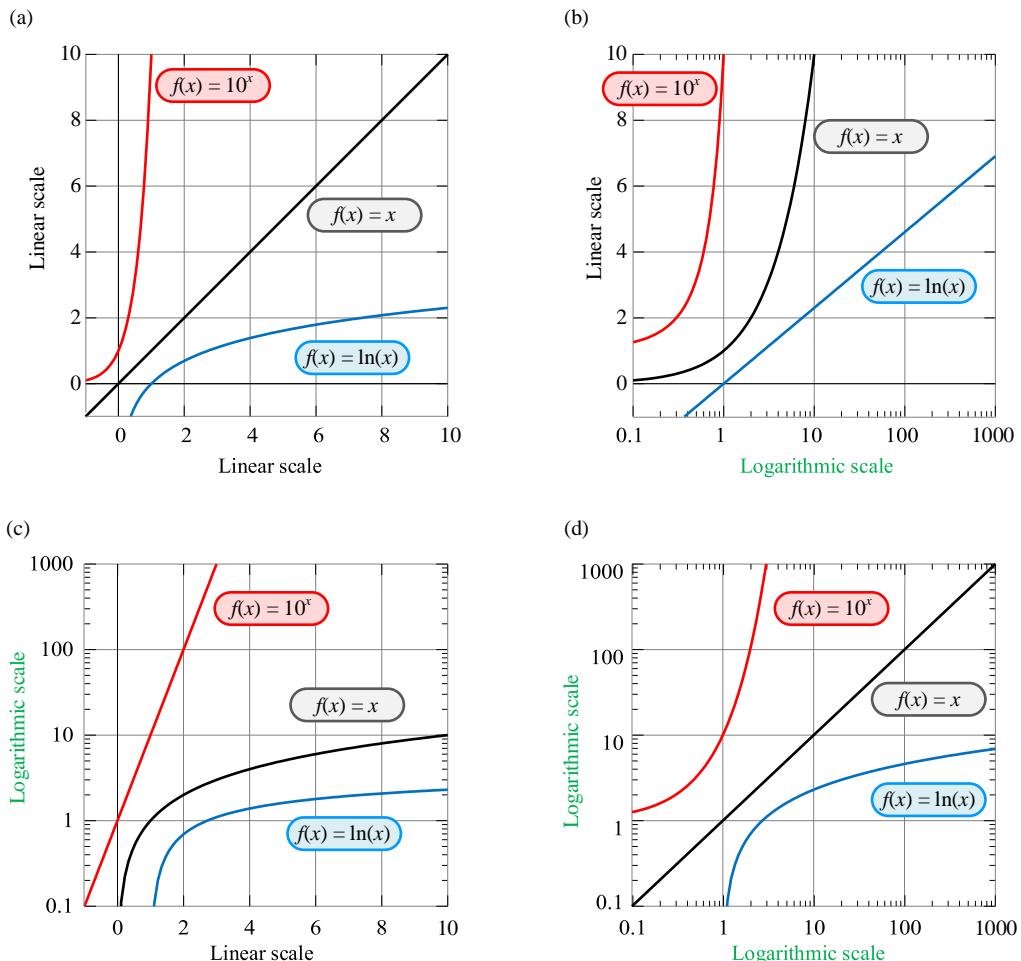
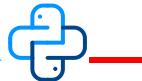


图 5. 几种对数刻度

表 2. 用英文读对数

数学表达	英文表达
$\log_4 x$	Logarithm of x with base four
$y = \log_a x$	y is the logarithm of x to the base a . y is equal to \log base a of x .
$\ln y$	Log y to the base e Log to the base e of y Natural log (of) y
$\log_2 8 = 3$	The log base 2 of 8 is equal to 3. The logarithm of 8 with base 2 is 3. Log base 2 of 8 is 3.
$\log_4 16 = 2$	The log base 4 of 16 is equal to 2.
$\log_2 \frac{1}{8} = -3$	The log base 2 of 1/8 is equal to -3.
$\log_6 108 = 3$	The logarithm of 108 to the base 6 is 3 3 is the logarithm of 108 to the base 6.



Bk3_Ch12_01.py 绘制图 5。

12.3 高斯函数：高斯分布之基础

复合函数 (function composition) 通俗地说就是函数套函数，是把几个简单的函数复合得到一个较为复杂的函数。

高斯函数

自然指数函数经常和其他函数构造复合函数。比如，自然指数函数复合二次函数，得到**高斯函数** (Gaussian function)：

$$f(x) = \exp(-\gamma x^2) \quad (10)$$

其中， γ 为参数， $\gamma > 0$ 。高斯函数的定义域是 $(-\infty, +\infty)$ ，而值域是 $(0, 1]$ 。**(10)** 给出的高斯函数关于 $x = 0$ 对称。

⚠ 注意，高斯函数无限接近 0，却不到达 0。

图 6 (a) 所示为 γ 决定高斯函数的形状。

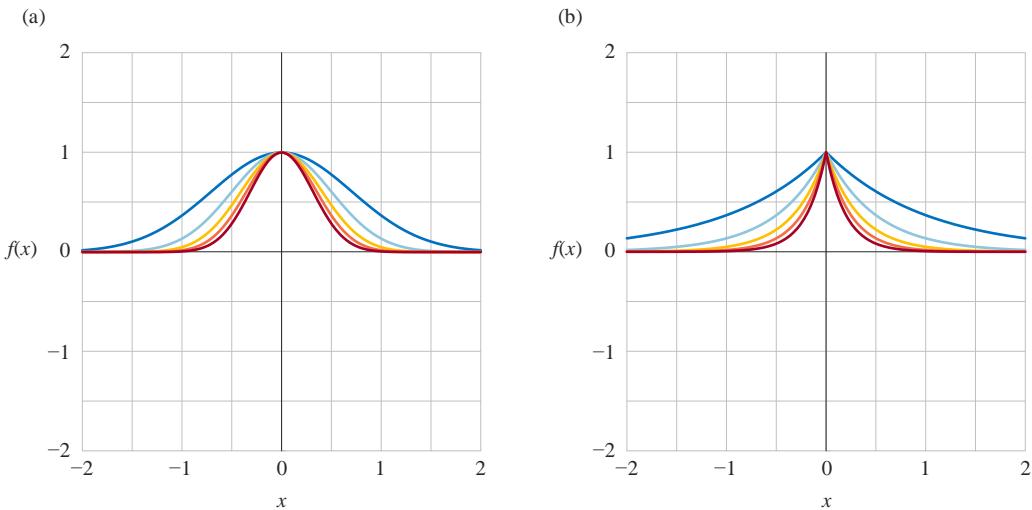


图 6. 高斯函数和拉普拉斯核函数

最基本的高斯函数为：

$$f(x) = \exp(-x^2) \quad (11)$$

如下也是常用的高斯函数的一般形式：

$$f(x) = a \cdot \exp\left(\frac{-(x-b)^2}{2c^2}\right) \quad (12)$$

上述高斯函数关于 $x=b$ 对称。

(11) 通过缩放、平移等变换来可以得到 (12)。函数变换是本章最后要讲解的内容。

高斯函数和**高斯分布** (Gaussian distribution) 的**概率密度函数** (Probability Density Function, PDF) 直接相关。高斯函数可以进一步推广得到**径向基核函数** (radial basis function, RBF)。

下一章将介绍二元高斯函数的性质。此外，鉴于高斯函数的重要性，本书后续导数、积分相关内容都会以高斯函数作为实例。

高斯小传

高斯函数以著名数学家**高斯** (Carl Friedrich Gauss) 命名。

在数据科学和机器学习领域，高斯的名字无处不在，比如大家耳熟能详的高斯核函数、高斯消去、高斯分布、高斯平滑、高斯朴素贝叶斯、高斯判别分析、高斯过程、高斯混合模型等等。并不是高斯发明了这些算法；而是，后来人在创造这些算法时，都用到了高斯分布。

被称作数学王子的高斯，出身贫寒。母亲做过女佣近乎文盲，父亲多半生靠体力讨生活。据说，高斯自幼喜欢读书，特别是和数学相关的书籍；渴望学习、热爱知识是他的驱动力，而不是颜如玉、黄金屋。



卡尔·弗里德里希·高斯 (Carl Friedrich Gauss)

德国数学家、物理学家、天文学家 | 1777 ~ 1855

常被称作“数学王子”，在数学的每个领域开疆拓土。丛书关键词：● 等差数列 ● 高斯分布
● 最小二乘法 ● 高斯朴素贝叶斯 ● 高斯判别分析 ● 高斯过程 ● 高斯混合模型 ● 高斯核函数

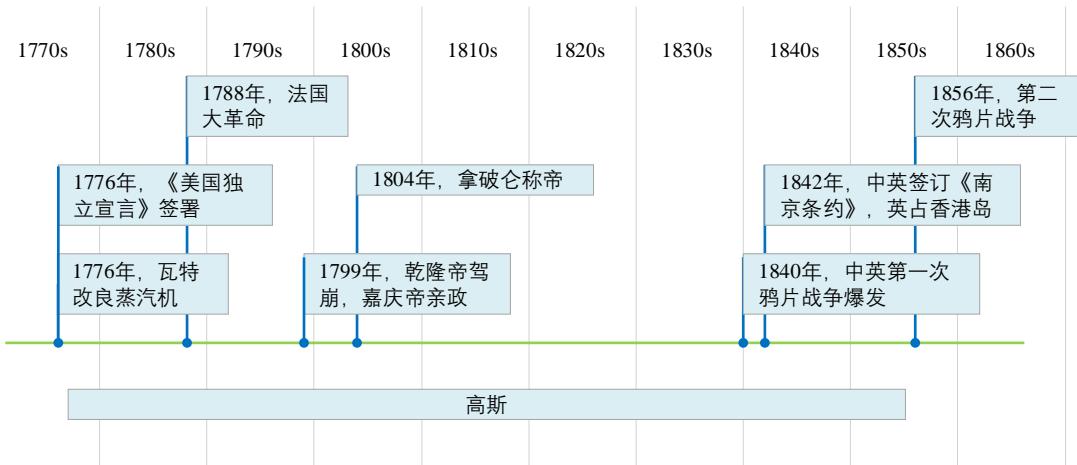


图 7. 高斯所处时代大事记

拉普拉斯核函数

此外，绝对值函数 $|x|$ 和指数函数的复合，得到的是一元**拉普拉斯核函数** (Laplacian kernel function)：

$$f(x) = \exp(-\gamma|x|) \quad (13)$$

图 6 (b) 所示为 γ 决定拉普拉斯核函数的形状。

⚠ 注意，图 6 (b) 中拉普拉斯核函数在 $x = 0$ 处有“尖点”，它破坏了函数的平滑。拉普拉斯核函数也经常出现在机器学习一些算法当中。

12.4 逻辑函数：在 0 和 1 之间取值

逻辑函数 (logistic function) 也可以视作是自然指数函数扩展得到的复合函数。下式为最简单的一元逻辑函数：

$$f(x) = \frac{1}{1 + \exp(-x)} = \frac{\exp(x)}{1 + \exp(x)} \quad (14)$$

更一般的一元逻辑函数形式为：

$$f(x) = \frac{1}{1 + \exp(-(b_0 + b_1 x))} \quad (15)$$

可以明显发现逻辑函数的取值范围在 0 和 1 之间，函数无限接近 0 和 1，却不能达到。如图 8 所示， b_1 影响图像的陡峭程度，注意图中 $b_0 = 0$ 。

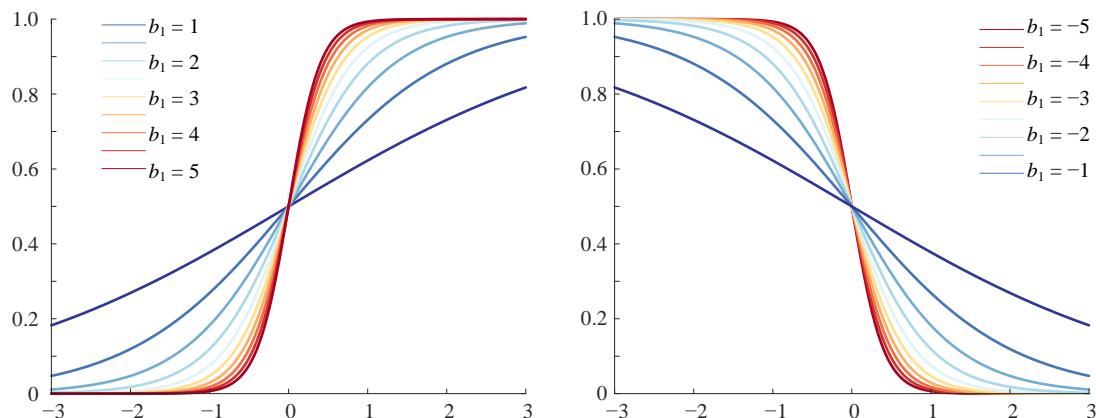


图 8. b_1 影响逻辑函数的陡峭程度

中心点位置

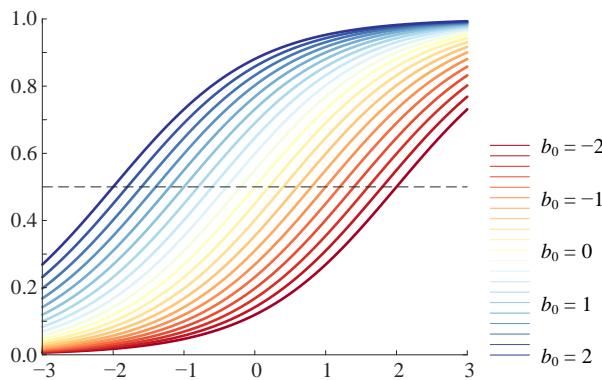
下面确定 $f(x) = 1/2$ 位置，令，

$$f(x) = \frac{1}{1 + \exp(-(b_0 + b_1 x))} = \frac{1}{2} \quad (16)$$

整理得到：

$$x = -\frac{b_0}{b_1} \quad (17)$$

这个点被称作为逻辑函数中心所在位置。图 9 中 $b_1 = 1$ ， b_0 决定逻辑函数中心所在位置。

图 9. $b_1 = 1$ 时, b_0 决定逻辑函数中心位置

逻辑回归 (logistic regression) 模型基于逻辑函数。逻辑回归虽然被称作回归模型，但是它经常用来做分类，特别是二分类。

上一章我们看到线性回归中输出值 y 是连续值。如图 10 所示，逻辑回归中 y 可以为离散值，比如 0、1。因此，我们也可以把逻辑函数想象成一个开关。此外，逻辑回归可以看做是在线性回归基础上增加了一个非线性映射。

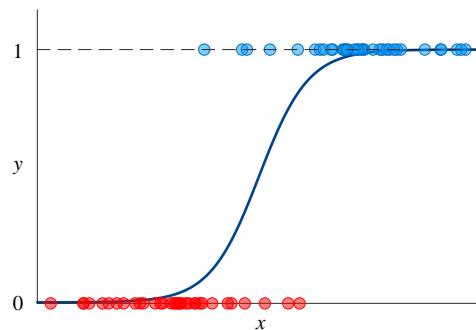


图 10. 逻辑回归可以用来做二分类

逻辑斯谛增长

自然界和人类社会中，指数增长这种 **J 型增长** (J-shaped growth) 一般只在一段时间内存在。各种条件会限制增长幅度，比如人口增长不可能一直按指数增长持续下去，毕竟地球的**承载能力** (carrying capability) 有限。

而逻辑曲线可以用来模拟人口增长的 **S 型增长曲线** (S-shaped growth curve)，如图 11 所示。

S 型增长曲线中，开始阶段类似指数增长。

然后，随着种群个体不断增多，受限于有限资源，个体之间对食物、生存空间等关键资源争夺越来越激烈，增长阻力变得越来越大，增速开始放慢。

最后，增长逐渐停止，趋向于瓶颈。

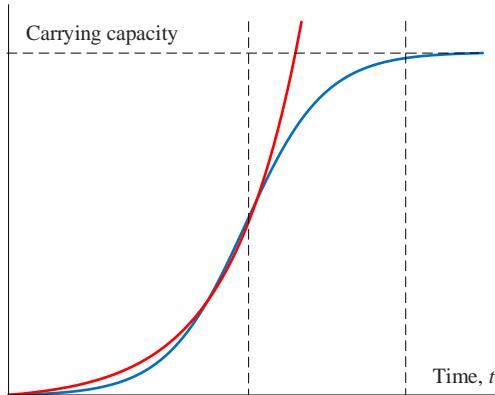


图 11. 逻辑函数模拟人口增长

S型函数

逻辑函数是 **S型函数** (sigmoid function 或 S-shaped function) 的一种。S型函数因其函数图像形状类似字母 S 而得名。机器学习和深度学习中，S型函数经常出现。

常见的 S型函数还有双曲正切函数 $f(x) = \tanh(x)$ 、反正切函数 $f(x) = \arctan(x)$ 、误差函数 $f(x) = \text{erf}(x)$ 等。



本书会在第 18 章专门介绍误差函数。

一些代数函数也可以归类为 S型函数，比如：

$$f(x) = \frac{x}{1+|x|}, \quad f(x) = \frac{x}{\sqrt{1+x^2}} \quad (18)$$

图 12 比较几种常用的 S型函数曲线。请注意，图 12 中反正切函数为 $f(x) = \frac{2}{\pi} \arctan\left(\frac{\pi}{2}x\right)$ 。计算双曲正切函数用的函数 `numpy.tanh()`。误差函数用的是符号函数 `sympy.erf()`。

双曲正切函数 $f(x) = \tanh(x)$ 和 (14) 逻辑函数的关系为：

$$f(x) = \frac{1}{1+\exp(-x)} = \frac{\exp(x)}{1+\exp(x)} = \frac{1}{2} + \frac{1}{2} \tanh\left(\frac{x}{2}\right) \quad (19)$$

图 12 中，除 (19) 以外，函数的取值范围都是 $(-1, 1)$ 。这些函数都无限接近 -1 和 1 ，但是不能达到。

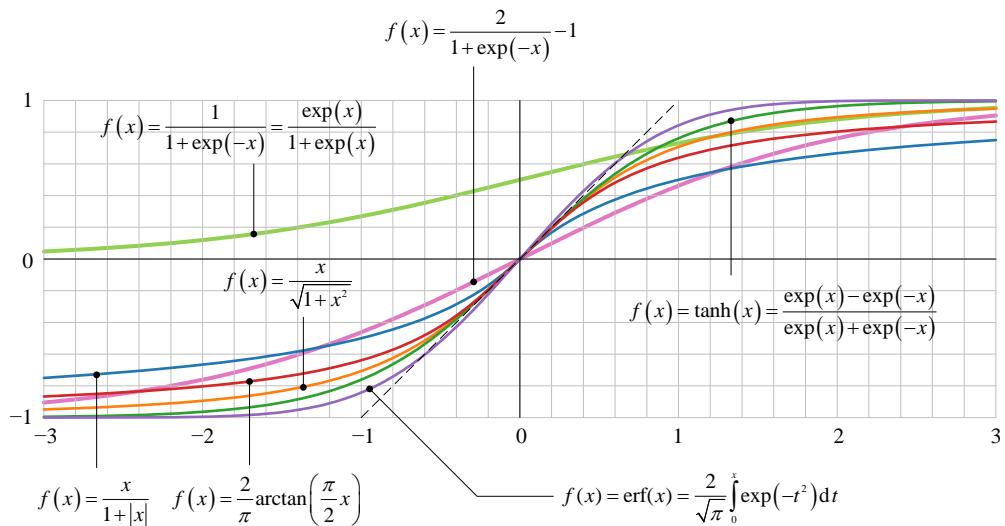


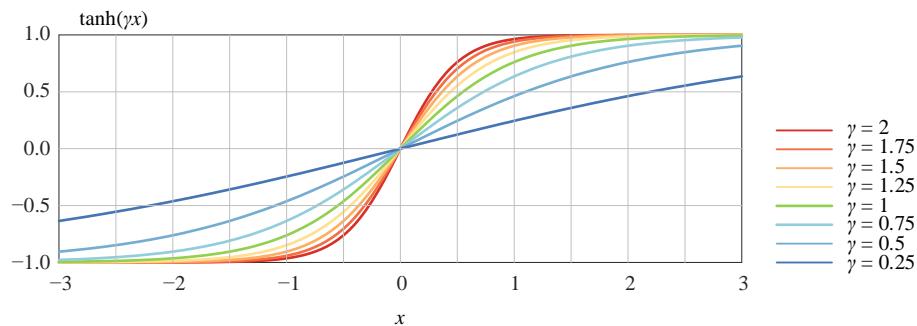
图 12. 比较常用的几种 S 型函数曲线形状

tanh() 函数

在很多机器学习算法中，sigmoid 函数特指 $\tanh()$ 函数。给定如下 $\tanh()$ 函数一般式：

$$f(x) = \tanh(\gamma x) \quad (20)$$

图 13 所示为 γ 如何影响曲线形状。

图 13. γ 影响双曲正切函数形状

12.5 三角函数：周期函数的代表

本节介绍几个常用的三角函数。**三角函数** (trigonometric function 或 circular function) 是一类**周期函数** (periodic function)。

正弦函数

正弦波 (sine wave 或 sinusoid) 是一种常见的波形，比如物理学中的**正弦交流电** (sinusoidal alternating current)。图 14 (a) 所示为如下最基本的**正弦函数** (sine function)：

$$y = f(x) = \sin(x) \quad (21)$$

图 14 (a) 所示正弦函数定义域为整个实数域。 $f(x) = \sin(x)$ 函数是**奇函数** (odd function)，关于原点对称。这个函数的**周期** (period) 是 $T = 2\pi$ ，函数值域是 $[-1, 1]$ 。准确来说，这个周期 T 是最小正周期。

图 14 (a) 所示正弦函数取得极大值 1 对应的 x 为：

$$x = \frac{\pi}{2} + 2\pi n \quad (22)$$

其中， n 为整数。

正弦函数的极小值为 -1 对应 x 为：

$$x = -\frac{\pi}{2} + 2\pi n \quad (23)$$

`numpy.sin()` 函数可以用来完成正弦计算。

余弦函数

图 14 (b) 所示为如下**余弦函数** (cosine function)：

$$y = f(x) = \cos(x) \quad (24)$$

图 14 (b) 所示余弦函数是偶函数，关于纵轴对称； $y = \cos(x)$ 相当于图 14 (a) 正弦函数水平向左移动 $\pi/2$ 。余弦函数也是周期为 2π 的周期函数。`numpy.cos()` 函数可以用来完成余弦计算。

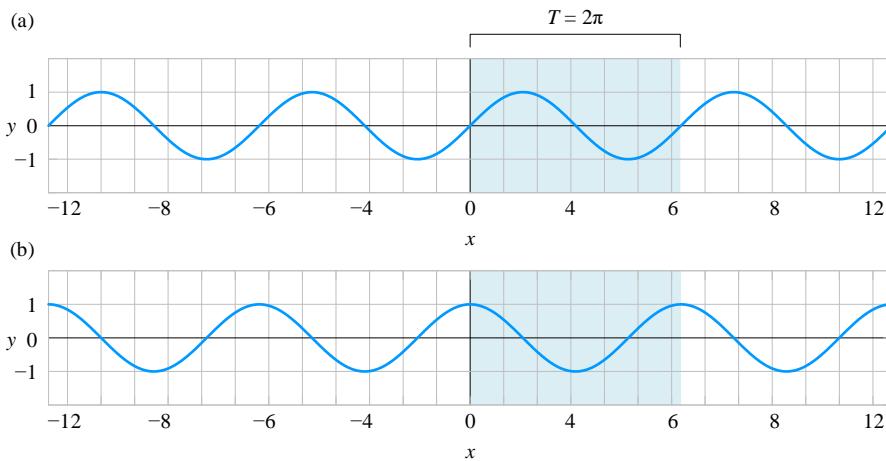


图 14. 正弦函数和余弦函数

表 3 总结了六个常用三角函数图像及性质。

表 3. 六个三角函数的图像和性质

函数	性质	图像
正弦 (sine) $y = \sin(x)$ numpy.sin()	定义域：整个实数集 值域： $[-1, 1]$ 最小正周期： 2π 奇函数，图像关于原点对称 极大值为 1，极小值为 -1	
余弦 (cosine) $y = \cos(x)$ numpy.cos()	定义域：整个实数集 值域： $[-1, 1]$ 最小正周期： 2π 偶函数，图像关于 y 轴对称 极大值为 1，极小值为 -1	
正切 (tangent) $y = \tan(x)$ 也记做 $y = \text{tg}(x)$ numpy.tan()	定义域： $\left\{x \mid x \neq k\pi + \frac{\pi}{2}, k \in \mathbb{Z}\right\}$ 值域：整个实数集 最小正周期： π 奇函数，图像关于原点对称 不存在极值	
余切 (cotangent) $y = \cot(x)$ 也记做 $y = \text{ctg}(x)$ 1/numpy.tan()	定义域： $\{x \mid x \neq k\pi, k \in \mathbb{Z}\}$ 值域：整个实数集 最小正周期： π 奇函数，图像关于原点对称 不存在极值	
正割 (secant) $y = \sec(x)$ 1/numpy.cos()	定义域： $\left\{x \mid x \neq k\pi + \frac{\pi}{2}, k \in \mathbb{Z}\right\}$ 值域： $ \sec(x) \geq 1$ 最小正周期： 2π 偶函数，图像关于 y 轴对称 不存在极值	

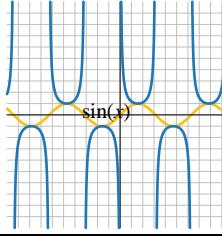
余割 (cosecant) $y = \csc(x)$ $1 / \text{numpy.sin}()$	定义域: $\{x \mid x \neq k\pi, k \in \mathbb{Z}\}$ 值域: $ \csc(x) \geq 1$ 最小正周期: 2π 奇函数, 图像关于原点对称 不存在极值	
--	--	---

表 4. 用英文读三角函数

数学表达	英文表达
$\sin \theta + x$	Sine of theta, that quantity plus x .
$\sin(\theta + \omega)$	Sine of sum theta plus omega. Sine of the quantity theta plus omega.
$\sin(\theta) \cdot x$	Sine theta times x .
$\sin(\theta\omega)$	Sine of the product theta time omega.
$(\sin \theta^2) \cdot x$	Sine of theta squared, that quantity times x .
$(\sin^2 \theta) \cdot x$	Sine squared of theta, that quantity times x .

12.6 函数变换：平移、缩放、对称

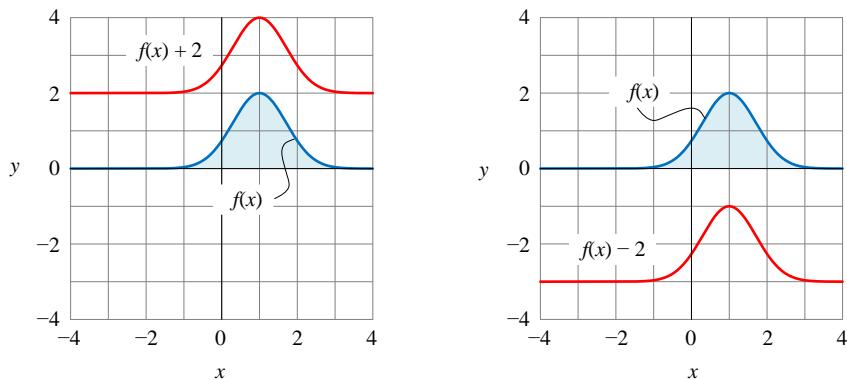
本章最后利用高斯函数和大家探讨函数变换。常见的函数变换有三种：平移、缩放和对称。

给定某个函数 $y = f(x)$ 解析式为：

$$f(x) = 2 \exp(-(x-1)^2) \quad (25)$$

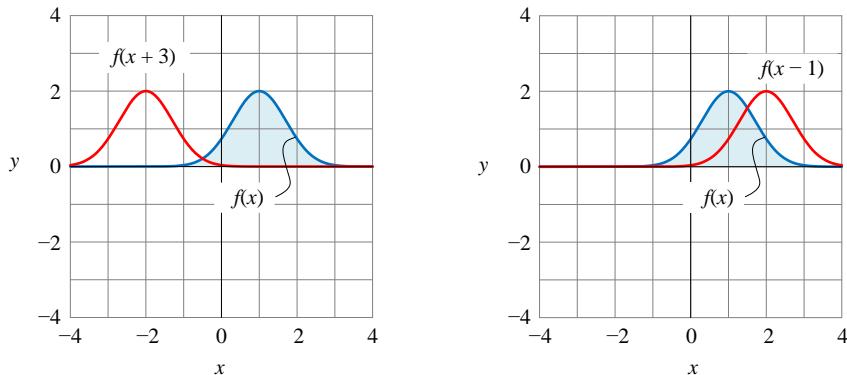
平移

如图 15 所示，相对 $y = f(x)$, $f(x) + c$ 竖直向上平移 c 单位 (vertical shift up by c units); $f(x) - c$ 将原函数竖直向下平移 c 单位 (vertical shift down by c units)。

图 15. 原函数 $y=f(x)$ 上下平移

如图 16 所示，相对 $y=f(x)$ ， $f(x+c)$ 相当于函数向左平移 c 单位 (horizontal shift left by c units)， $c > 0$ ； $f(x-c)$ 相对原函数向右平移 c 单位 (horizontal shift right by c units)。

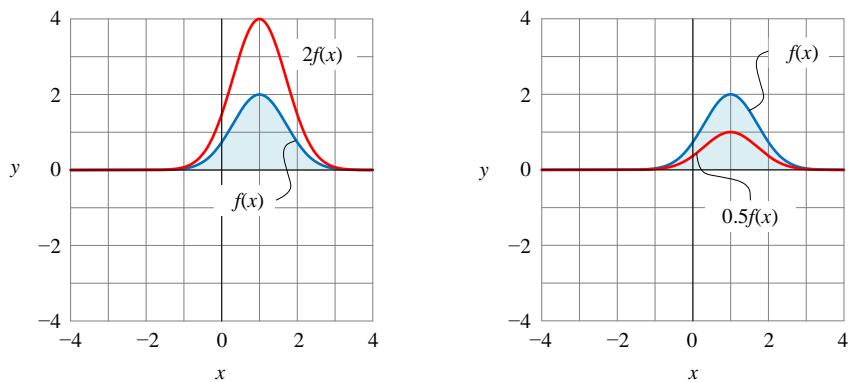
⚠ 注意，水平平移不影响函数图像和横轴包围的面积。

图 16. 原函数 $y=f(x)$ 左右平移

缩放

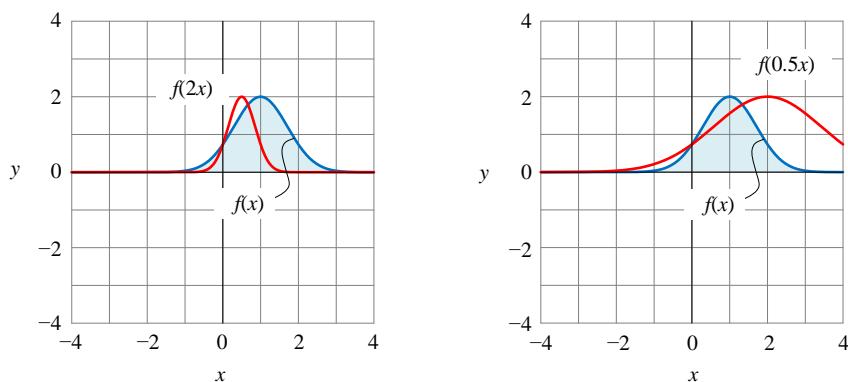
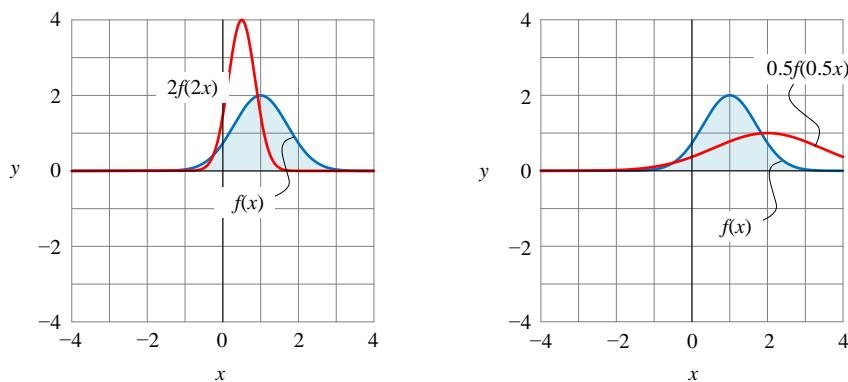
如图 17 所示，相对 $y=f(x)$ ， $cf(x)$ 相当于函数 **竖直方向缩放** (vertical scaling)。 $c > 1$ 时， $cf(x)$ **竖直方向拉伸** (vertical stretch)； $0 < c < 1$ 时， $cf(x)$ **竖直方向压缩** (vertical compression)。

⚠ 注意，这种几何变换等比例缩放面积。

图 17. 原函数 $y = f(x)$ 竖直方向伸缩

如图 18 所示，相对 $y = f(x)$, $f(cx)$ 相当于函数**水平方向伸缩** (horizontal scaling)。 $c > 1$ 时，**水平方向压缩** (horizontal compression); $0 < c < 1$ 时，**水平方向拉伸** (horizontal stretch)。面积等比例缩放，缩放比例为 $1/c$ 。

如图 19 所示，相对于 $f(x)$, $cf(cx)$ 面积不变。

图 18. 原函数 $y = f(x)$ 水平方向伸缩图 19. 原函数 $y = f(x)$ 水平方向、竖直方向同时伸缩

对称

如图 20 所示，相对 $y = f(x)$ ， $f(-x)$ 相当于函数关于 y 轴对称 (reflection about y axis)。 $-f(x)$ 相当于函数关于 x 轴对称 (reflection about x axis)。而 $f(x)$ 和 $-f(-x)$ 关于原点对称。

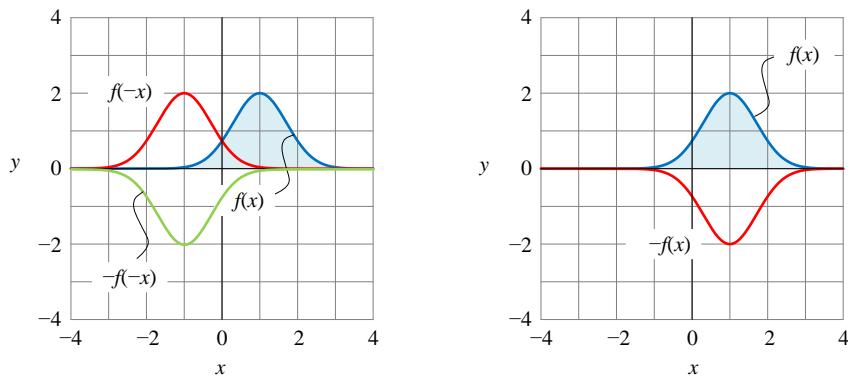


图 20. 原函数 $y = f(x)$ 关于横轴、纵轴对称



一元高斯分布的概率密度函数解析式如下：

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (26)$$

其中， μ 为均值， σ 为标准差。

高斯分布的概率密度函数实际上可以通过高斯函数经过函数变换得到。

观察 (26) 中指数部分存在两个函数变换——横轴缩放 (σ)、横轴平移 (μ)。

令，

$$z = \frac{x-\mu}{\sigma} \quad (27)$$

将 (27) 代入 (26)，整理得到：

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-1}{2}z^2\right) \quad (28)$$

(28) 中分母 $\sigma\sqrt{2\pi}$ ，起到的是纵向缩放作用，保证曲线下方面积为 1。理解这步变换需要积分知识。图 21 所示为以上三步几何变换。

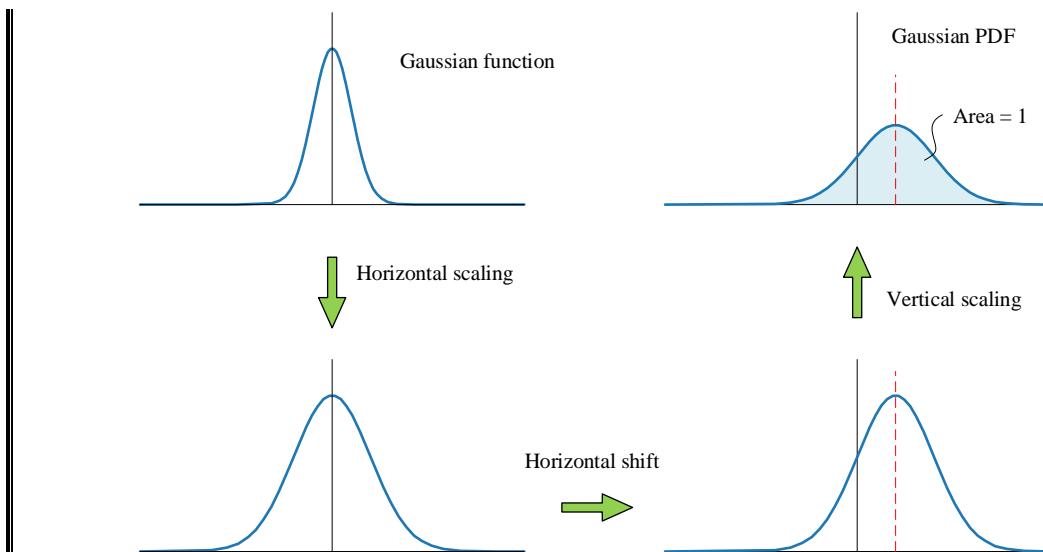
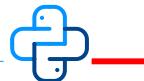


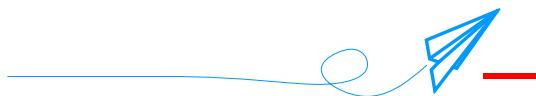
图 21. 高斯函数三步几何变换



Bk3_Ch12_02.py 绘制图 15~图 20。



在 Bk3_Ch12_02.py 基础上，我们做了一个 App 用来交互呈现不同参数对高斯函数的形状和位置影响。请大家参考代码文件 Streamlit_Bk3_Ch12_02.py。



本章有两个要点——函数在数值转化的作用、函数变换。

机器学习各种算法中，函数起到数据转化的作用，比如把取值在正负无穷之间的数值转化在 0 和 1 之间。本系列丛书《数据科学》一册会专门探讨这个话题。

平移、缩放、对称等函数变换是几何变换在函数上的应用。请大家格外注意，函数变换过程前后，形状、单调性、极值点、对称轴、面积等性质的变化。

13 Bivariate Functions 二元函数 从三维几何图形角度理解



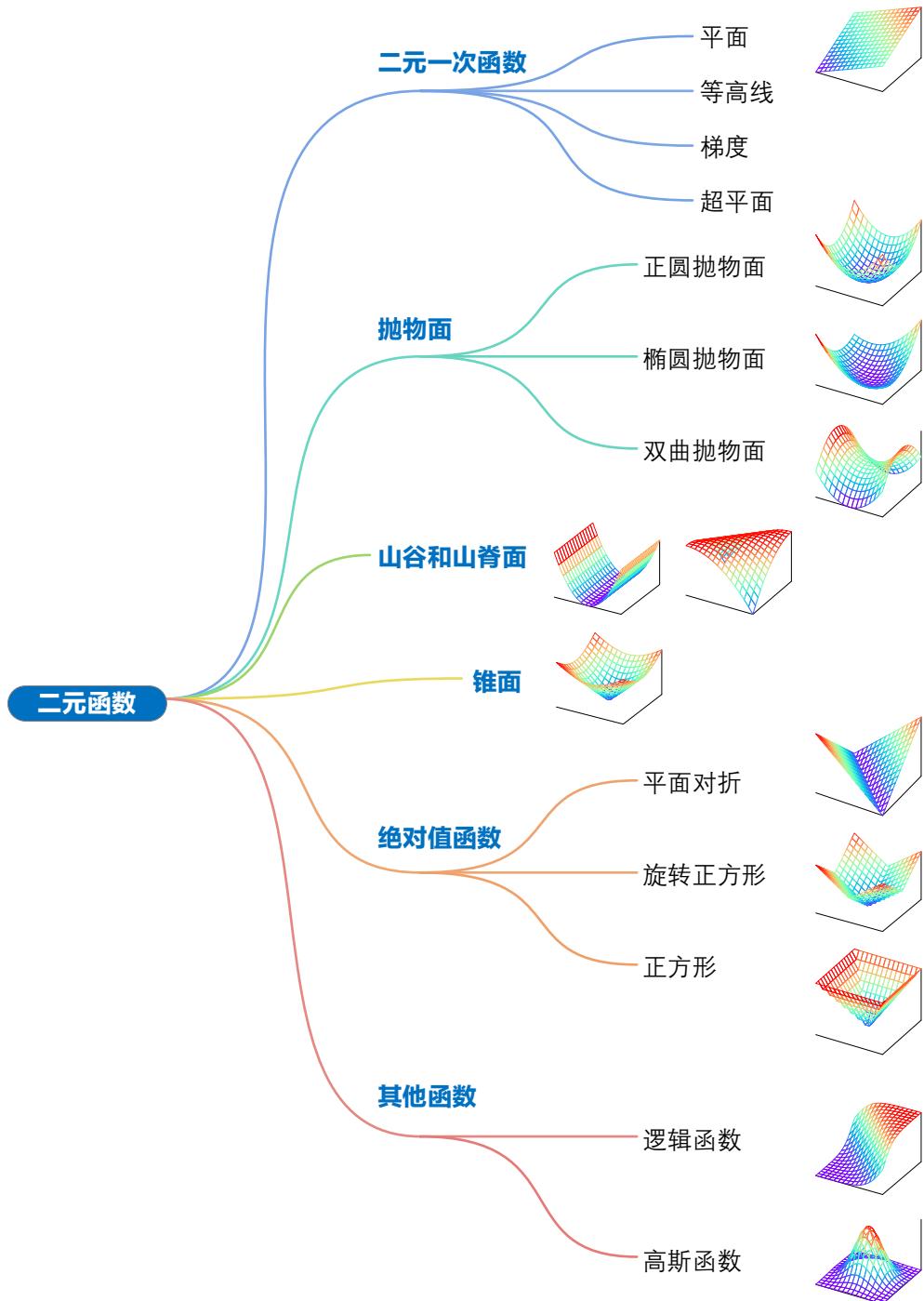
当然，我们可以使用任何需要符号。不要嘲笑符号。发明它们，它们很强大。事实上，很大程度数学就是在发明更好的符号。

We could, of course, use any notation we want; do not laugh at notations; invent them, they are powerful. In fact, mathematics is, to a large extent, invention of better notations.

——理查德·费曼 (Richard P. Feynman) | 美国理论物理学家 | 1918 ~ 1988



- ◀ `Axes3D.plot_surface()` 绘制三维曲面
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `numpy.linspace()` 在指定的间隔内，返回固定步长的数据
- ◀ `numpy.meshgrid()` 生成网格数据



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

13.1 二元一次函数：平面

二元一次函数是一元一次函数的扩展，一般式如下：

$$y = f(x_1, x_2) = w_1 x_1 + w_2 x_2 + b \quad (1)$$

当 w_1 和 w_2 均为 0 时， $f(x_1, x_2) = b$ 为二元常数函数，平行于 $x_1 x_2$ 水平面。

用矩阵乘法，(1) 可以写成：

$$y = f(x_1, x_2) = \mathbf{w}^T \mathbf{x} + b \quad (2)$$

其中，

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (3)$$

当 y 取一定值时，比如 $y = c$ ，平面退化为一条直线：

$$w_1 x_1 + w_2 x_2 + b = c \quad (4)$$

从另外一个角度， c 相当于 $f(x_1, x_2)$ 平面的某一条等高线，即 $f(x_1, x_2) = c$ 等高线为直线。

举个例子

图 1 所示图像对应如下解析式：

$$y = f(x_1, x_2) = x_1 + x_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (5)$$

图 1 (a) 所示为 (5) 对应平面，图中黑色直线对应 $x_1 + x_2 = 0$ ，即 $x_2 = -x_1$ 。图 1 (b) 所示 $f(x_1, x_2)$ 平面等高线都平行于 $x_1 + x_2 = 0$ 。由于 $f(x_1, x_2)$ 为线性函数，因此等高线平行，且间距相同。

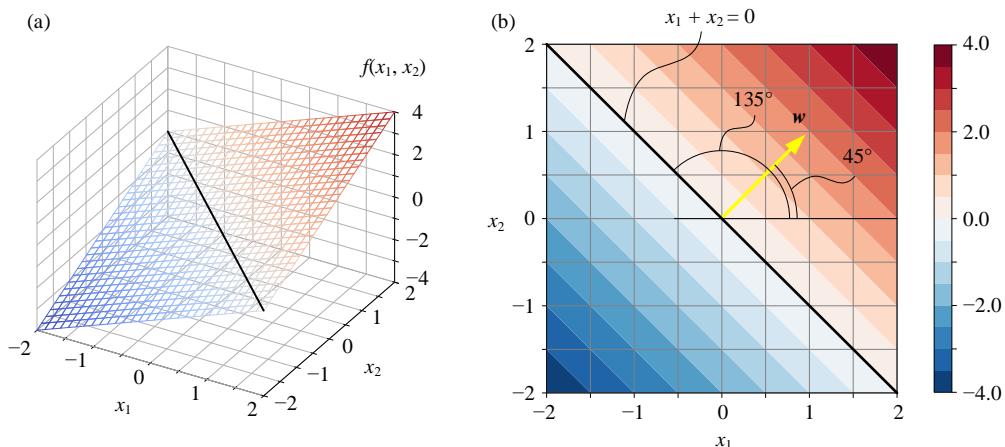


图 1. $f(x_1, x_2) = x_1 + x_2$ 网格图和等高线图

图 1 (b) 中黄色箭头为 $f(x_1, x_2)$ 增大方向，箭头和 x_1 轴正方向夹角为 45° 。有心的读者可能发现，黄色箭头对应的向量就是 w ：

$$w = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (6)$$

图 1 (b) 中， w 向量垂直于等高线并指向 $f(x_1, x_2)$ 增大方向。这并非巧合，实际上 w 向量便是 **梯度向量** (gradient vector)。本书在前文讲解不等式时，提到过梯度这个概念，不过当时我们关注的是梯度的反方向，即梯度下降方向。

本系列丛书内容不断深入，大家会理解 w 的几何意义以及梯度向量这一重要概念。这里先给大家留下一个印象。

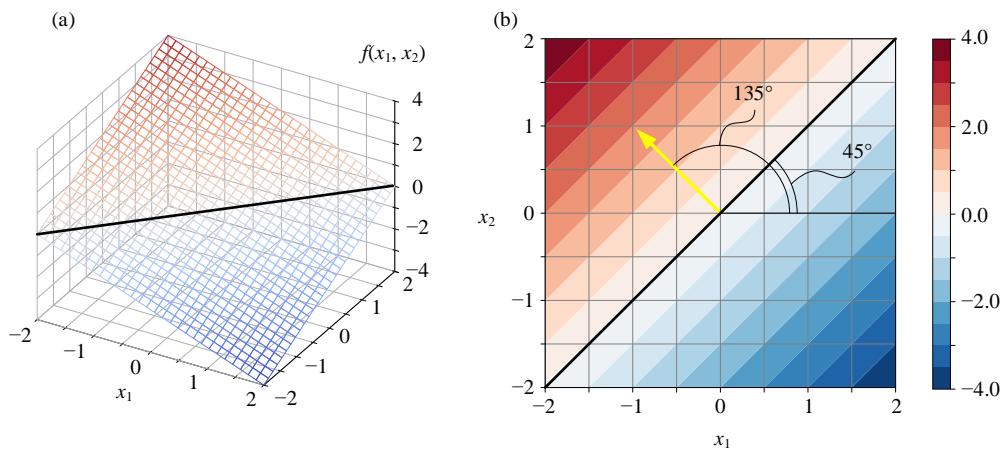
此外，相信大家已经意识到向量是个多面手，向量不仅仅是一列或一行数，还是有方向的线段。大家会在本系列丛书《矩阵力量》经常听到这句提醒——有向量的地方，就有几何！希望大家在看到向量出现时，多从几何视角思考向量的几何内涵。

第二个例子

图 2 所示平面对应解析式如下：

$$y = f(x_1, x_2) = -x_1 + x_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (7)$$

图 2 (b) 中黄箭头同样指向 $f(x_1, x_2)$ 增大方向，对应 (7) 中 w 。箭头和 x_1 轴正方向夹角为 135° 。

图 2. $f(x_1, x_2) = -x_1 + x_2$ 网格图和等高线图

等高线平行纵轴

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

当 $w_1 = -1, w_2 = 0, b = 0$ 时, $f(x_1, x_2)$ 平面高度仅仅受到 x_1 影响。图 3 所示图像对应如下解析式:

$$y = f(x_1, x_2) = -x_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (8)$$

图 3 (a) 所示平面平行于 x_2 轴, 即纵轴。图 3 (b) 所示 $f(x_1, x_2)$ 平面等高线同样平行于 x_2 轴。图 3 (b) 中, 黄色箭头为函数 $f(x_1, x_2)$ 增大方向, 箭头平行 x_1 轴朝左, 即朝向 x_1 轴负方向。

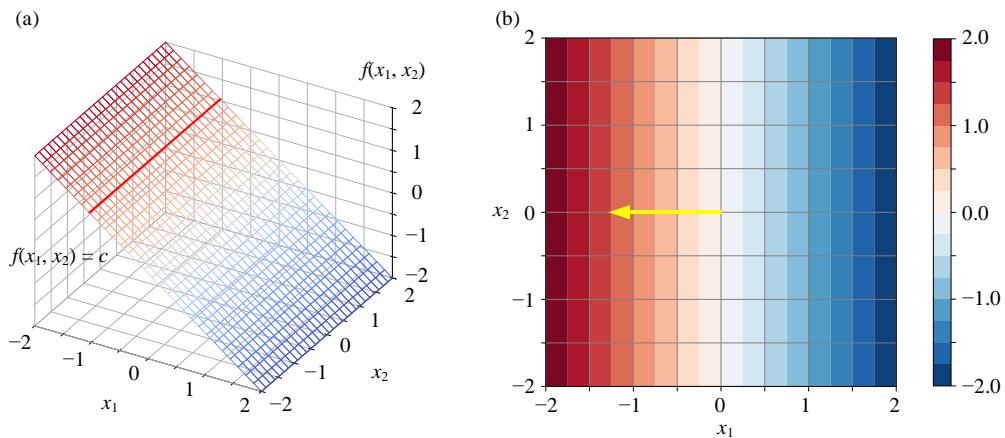


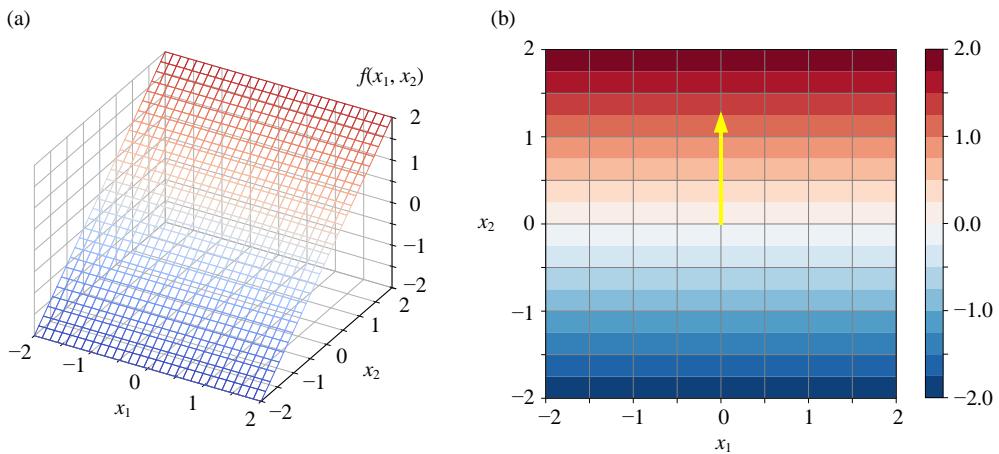
图 3. $f(x_1, x_2) = -x_1$ 网格图和等高线图

等高线平行横轴

当 $w_1 = 0, w_2 = 1, b = 0$ 时, $f(x_1, x_2)$ 平面仅仅受到 x_2 影响。图 4 所示图像对应如下解析式:

$$y = f(x_1, x_2) = x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (9)$$

图 4 (a) 所示平面平行于 x_1 轴。图 4 (b) 所示 $f(x_1, x_2)$ 平面等高线同样平行于 x_1 轴。图 4 (b) 中黄色箭头同样为 $f(x_1, x_2)$ 增大方向, 箭头指向 x_2 轴正方向。

图 4. $f(x_1, x_2) = x_2$ 网格图和等高线图

平面叠加

如图 5 所示，若干平面叠加得到的还是平面。函数 $f_i(x_1, x_2)$ 中下角标 i 为函数序号，不同序号代表不同函数。

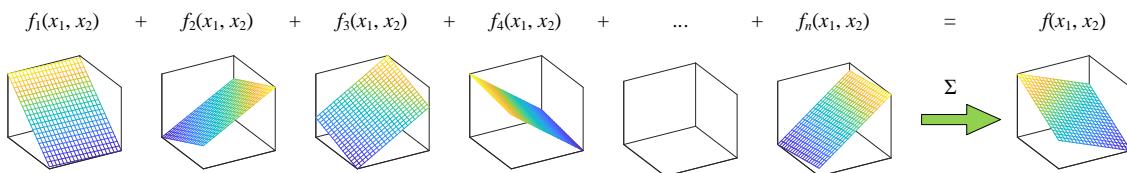


图 5. 若干平面叠加得到的还是平面

超平面

一次函数中变量数量继续增多时，将获得**超平面** (hyperplane)，对应的解析式如下：

$$y = f(x_1, x_2, \dots, x_D) = w_1 x_1 + w_2 x_2 + \dots + w_D x_D + b \quad (10)$$

将 (10) 写成矩阵运算形式：

$$y = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b \quad (11)$$

其中，

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_D \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \quad (12)$$

平面直线、三维空间直线、三维空间平面可以借助不同数学工具进行描述，如表 1 所总结。请读者格外注意区分，代数中函数、方程式、参数方程三个概念之间区别。

表 1. 不同数学工具描绘直线和平面

	类型	图像			
$f(x_1) = w_1x_1 + b$	函数				
$w_1x_1 + w_2x_2 + b = 0$	方程式				
$\begin{cases} x_1 = c_1 + \tau_1 t \\ x_2 = c_2 + \tau_2 t \end{cases}$	参数方程				
$f(x_1, x_2) = w_1x_1 + w_2x_2 + b$	函数				
$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$	方程式				
$\begin{cases} x_1 = c_1 + \tau_1 t \\ x_2 = c_2 + \tau_2 t \\ x_3 = c_3 + \tau_3 t \end{cases}$	参数方程				



本书前文介绍过一元线性回归，回归模型中只含有一个自变量和一个因变量。从图像上看，一元线性回归模型就是一条直线。

自变量的个数增加到两个，我们便得到二元线性回归。二元线性回归解析式可以写成 $y = b_0 + b_1x_1 + b_2x_2$ ，这就是我们本节介绍的二元一次函数，对应的图像是一个平面。

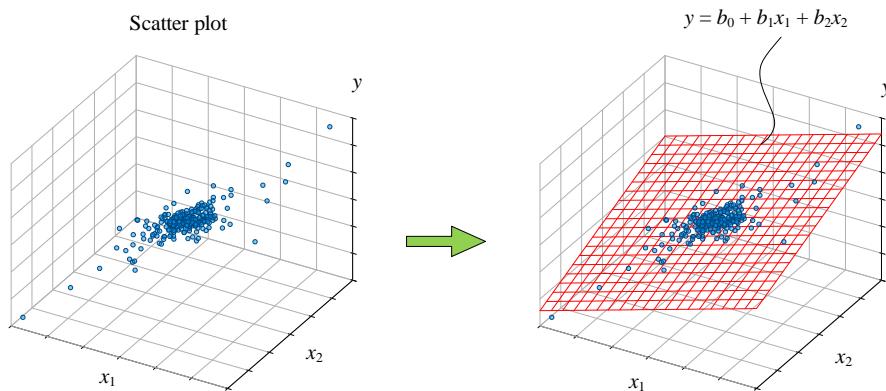
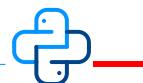


图 6. 从散点图到二元回归平面

图 6 左图是三维直角坐标系散点图。通过观察散点图，我们可以发现因变量随自变量变化的大致趋势。

图 6 右图中红色平面就是二元线性回归模型对应的图形，这个平面试图用一个平面（线性）解释自变量和因变量之间的量化关系。



Bk3_Ch13_01.py 绘制图 1 到图 4。代码中创建了三个自定义函数，用来可视化。另外，请大家修改 Bk3_Ch13_01.py 并绘制本章后续图像。

13.2 正圆抛物面：等高线为正圆

正圆抛物面 (circular paraboloid) 的是**抛物面** (paraboloid) 的一种特殊形式，它的等高线为正圆。正圆抛物面的最简单的形式为：

$$y = f(x_1, x_2) = a(x_1^2 + x_2^2) \quad (13)$$

(13) 可以写成如下矩阵运算形式：

$$y = f(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} a & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = a \mathbf{x}^T \mathbf{x} = a \|\mathbf{x}\|^2 \quad (14)$$

向量的模

请大家格外注意，(14) 可以写成 $y = f(x_1, x_2) = a\|\mathbf{x}\|^2$ 这种形式，其中 $\|\mathbf{x}\|$ 叫向量 \mathbf{x} 的模 (norm)。

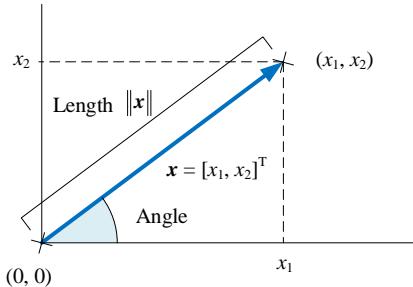


图 7. 向量有大小和方向两个性质

如图 7 所示，有了坐标系，向量 \mathbf{x} 可以理解为平面上有方向的线段，它有大小和方向两个性质。 $\|\mathbf{x}\|$ 为向量 \mathbf{x} 的模，就是向量的长度，定义为：

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2} \quad (15)$$

观察 (15)，利用勾股定理， $\|\mathbf{x}\|$ 相当于 (x_1, x_2) 和原点 $(0, 0)$ 之间的距离，即欧氏距离。而 (14) 相当于欧氏距离的平方。



建议大家回想本书第 7 章介绍的“等距线”这个概念，回忆欧氏距离对应的等距线有怎样特点。本书第 22 章将继续这一话题。

开口朝上

图 8 所示为正圆抛物面开口朝上，对应的解析式如下：

$$y = f(x_1, x_2) = x_1^2 + x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{x}^T \mathbf{x} = \|\mathbf{x}\|^2 \quad (16)$$

观察图 8 (b)，三维等高线为一系列同心正圆。观察等高线变化和曲面，可以发现等高线越密集，曲面变化越剧烈，也就是说曲面坡面越陡峭。图 8 所示曲面最小值点为 $(0, 0)$ 。



注意，图 8 (b) 中黄色箭头不再平行。但是，不同位置的黄色箭头都垂直于等高线，并指向函数增大方向。



要想获得黄色箭头 (梯度向量) 准确解析式就需要用到偏导数这个数学工具，偏导数是本书第 16 章要介绍的内容。

另外，当 x_1 为定值时，比如 $x_1 = 1$ ，得到的曲线为抛物线：

$$y = f(x_1=1, x_2) = 1 + x_2^2 \quad (17)$$

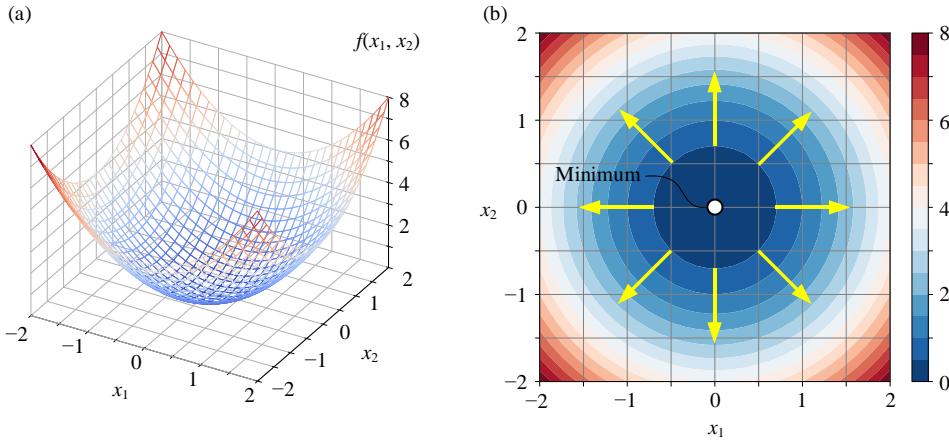


图 8. 开口朝上正圆抛物面，网格图和等高线图

开口朝下

图 9 所示同样为正圆抛物面，但开口朝下，解析式如下所示：

$$y = f(x_1, x_2) = -x_1^2 - x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -\mathbf{x}^\top \mathbf{x} \quad (18)$$

图 9 所示曲面在 $(0, 0)$ 处取得最大值点。

图 9 (b) 中不同位置的黄色箭头也都垂直于等高线，并指向函数增大方向。图 8 (b) 中箭头发散，但是图 9 (b) 箭头汇聚。这和曲面的凸凹性有关。图 8 (a) 曲面为凸面，而图 9 (a) 曲面为凹面。

值得注意的是，图 8 关于 x_1x_2 平面镜像便得到图 9。

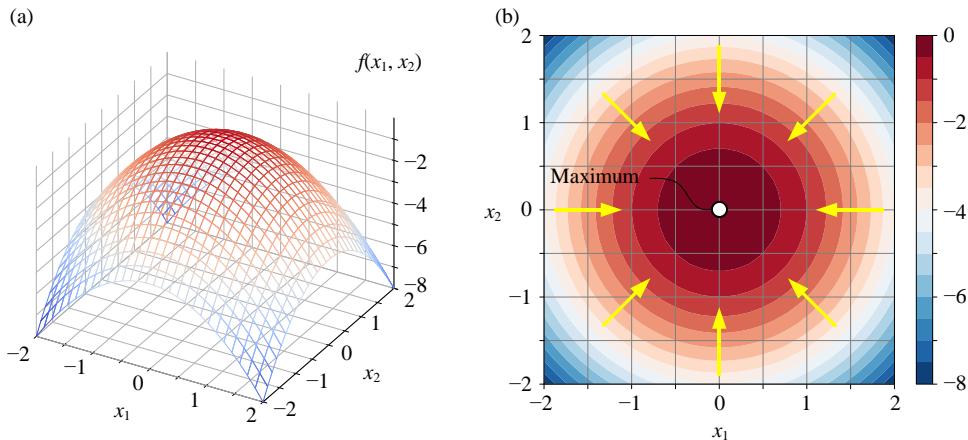


图 9. 开口朝下正圆抛物面，网格图和等高线图

平移

本书前文介绍过函数变换思想，在三维直角坐标系中，将(18)中二元函数变量 (x_1, x_2) 平移 (c_1, c_2) 得到：

$$y = f(x_1, x_2) = -(x_1 - c_1)^2 - (x_2 - c_2)^2 = -(\mathbf{x} - \mathbf{c})^\top (\mathbf{x} - \mathbf{c}) = -\|\mathbf{x} - \mathbf{c}\|^2 \quad (19)$$

其中， $\mathbf{c} = [c_1, c_2]^\top$ 。

举个例子，当 $\mathbf{c} = [1, 1]^\top$ 时，(19)对应的抛物面曲面和等高线如图10所示。图9图像在 x_1x_2 平面平移 $\mathbf{c} = [1, 1]^\top$ ，得到图10图像。正圆抛物面的中心移动到了 $(1, 1)$ 。相应地，最大值点也移动到了 $(1, 1)$ 。

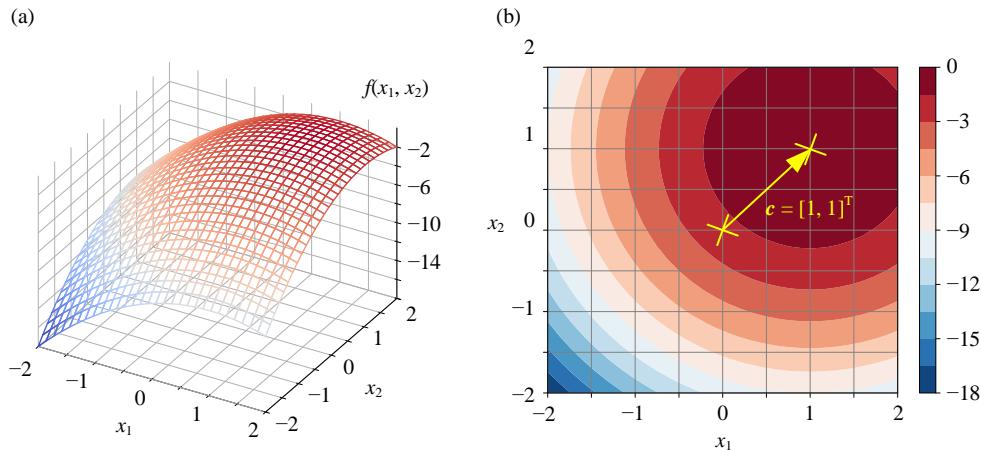


图 10. 抛物面平移

13.3 椭圆抛物面：等高线为椭圆

开口朝上

开口朝上**椭圆抛物面** (elliptic paraboloid) 的一般形式为：

$$y = f(x_1, x_2) = \frac{x_1^2}{a^2} + \frac{x_2^2}{b^2} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (20)$$

其中， a 和 b 都不为 0。特别地，当 $a^2 = b^2$ 时，椭圆抛物面便是正圆抛物面。

将(20)写成：

$$y = f(x_1, x_2) = \mathbf{x}^T \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix} \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix} \mathbf{x} = \left(\begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix} \mathbf{x} \right)^T \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix} \mathbf{x} \quad (21)$$

我们可以发现，从几何视角来看，上式中的对角方阵起到的就是“缩放”这个几何操作。

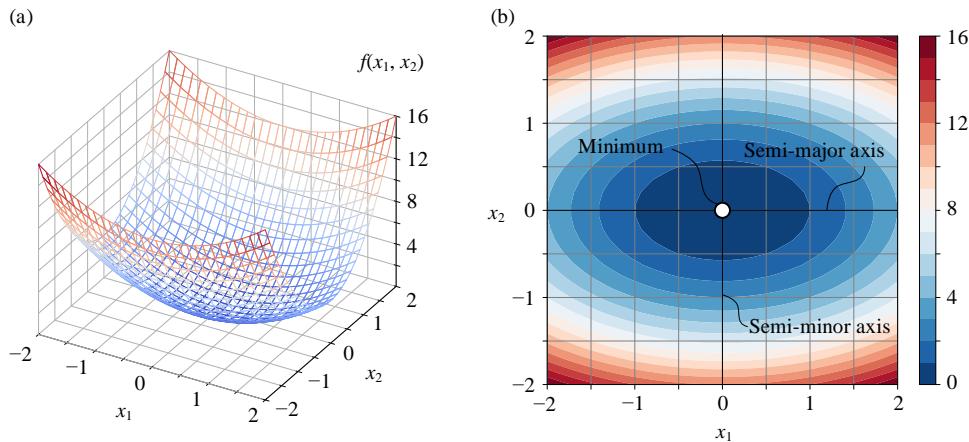


图 11. 开口朝上正椭圆抛物面，等高线为正椭圆，半长轴位于 x_1 轴，网格图和等高线图

举个例子

图 11 所示椭圆抛物面开口朝上，解析式如下：

$$y = f(x_1, x_2) = x_1^2 + 3x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (22)$$

图 11 所示椭圆抛物面的最小值点位于 $(0, 0)$ 。图 8 在 x_2 轴方向以一定比例缩放便得到图 11。

如图 11 (b) 所示，三维等高线为一系列椭圆。这些椭圆为正椭圆，其半长轴位于 x_1 轴。

回顾一下前文介绍过的椭圆相关概念。**长轴** (major axis) 是过焦点与椭圆相交的线段长，也叫做椭圆最长的直径；**半长轴** (semi-major axis) 是椭圆长轴的一半长。**短轴** (minor axis) 为椭圆最短的直径，**半短轴** (semi-minor axis) 为短轴的一半。

开口朝下

图 12 所示正椭圆抛物面开口朝下，对应解析式如下：

$$y = f(x_1, x_2) = -3x_1^2 - x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} -3 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (23)$$

如图 12 (b) 所示，三维等高线为正椭圆，半长轴位于 x_2 轴。图 12 所示曲面最大值点位于 $(0, 0)$ 。

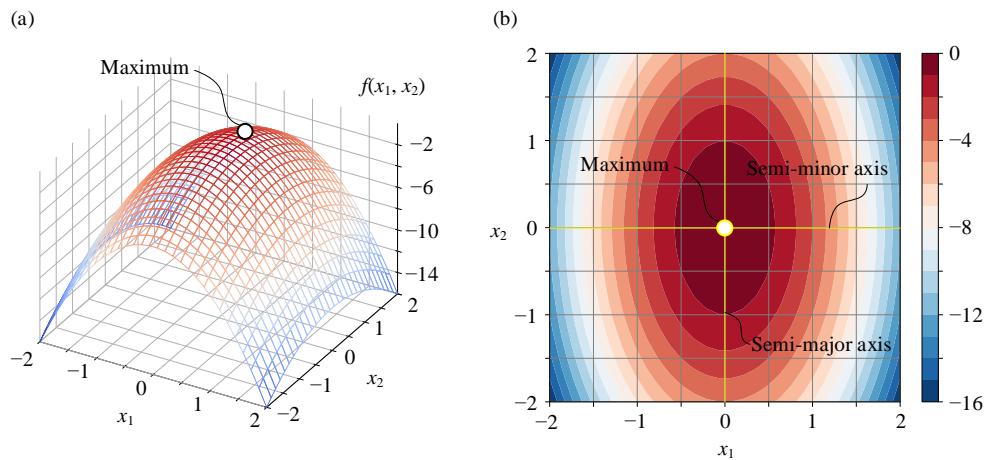


图 12. 开口朝下正椭圆抛物面，半长轴位于 x_2 轴，网格图和等高线图

旋转

图 13 所示旋转椭圆抛物面开口朝上，解析式如下：

$$y = f(x_1, x_2) = x_1^2 + x_1 x_2 + x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 1/2 \\ 1/2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (24)$$

观察图 13 (b) 可以容易发现三维等高线不再是正椭圆，而是旋转椭圆。旋转椭圆的长半轴和 x_1 轴正方向夹角 135° 。

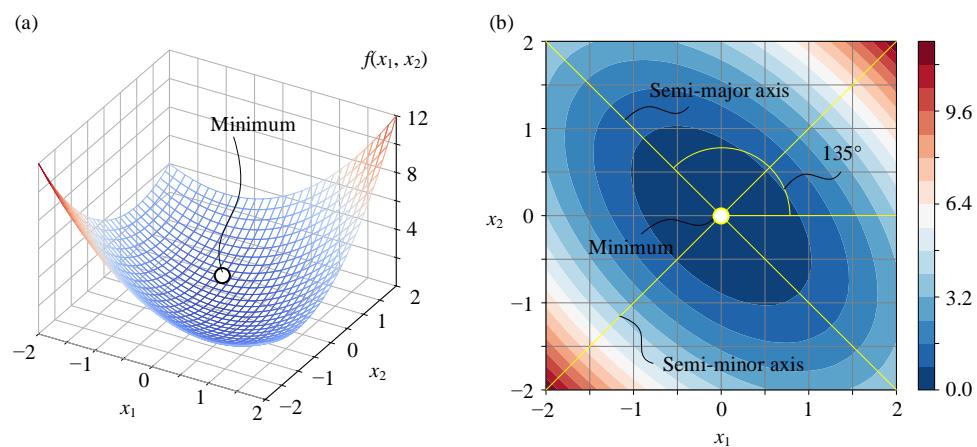


图 13. 开口朝上旋转椭圆抛物面，网格图和等高线图

图 14 所示为旋转椭圆抛物面开口朝下，对应解析式如下：

$$y = f(x_1, x_2) = -x_1^2 + x_1 x_2 - x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^\top \begin{bmatrix} -1 & 1/2 \\ 1/2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (25)$$

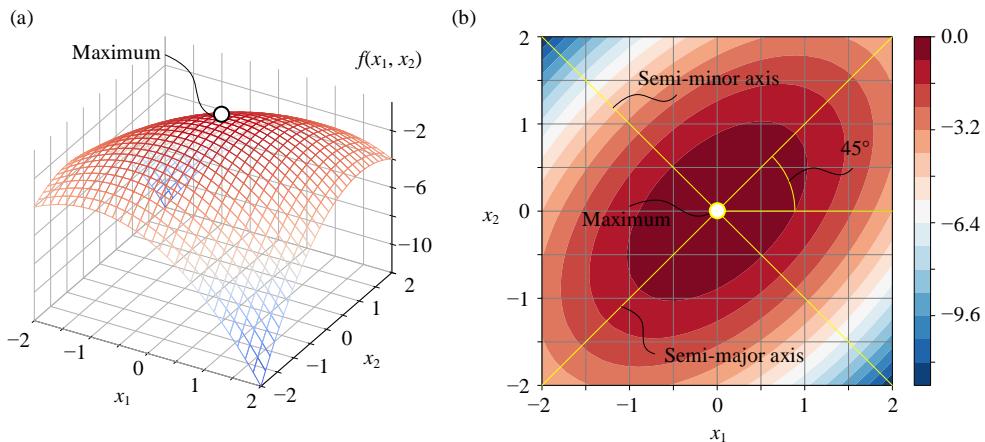
图 14 三维等高线椭圆旋转方向和图 13 正好相反。图 14 最大值点位于 $(0, 0)$ 。

图 14. 开口朝下旋转椭圆抛物面，网格图和等高线图



在多元线性回归中，为了简化模型复杂度，可以引入**正则项** (regularizer)。正则项的目的是“收缩”，即让某些估计参数变小，甚至为 0。

L2 正则是常见的正则方法之一。图 15 左上旋转椭圆抛物面上红叉 \times 对应的位置就是二元线性回归中最优参数 b_1 和 b_2 (不考虑常数 b_0) 所在位置。

从几何角度，引入 L2 正则项，就相当于在旋转抛物面上叠加一个正圆抛物面。观察图 15 右图，可以发现引入正圆抛物面后，参数 b_1 和 b_2 位置相对于更靠近原点。这便是 L2 正则项 (正圆曲面) 起到的作用。

L2 正则项权重越大，其影响越大，即红叉 \times 位置越靠近原点。

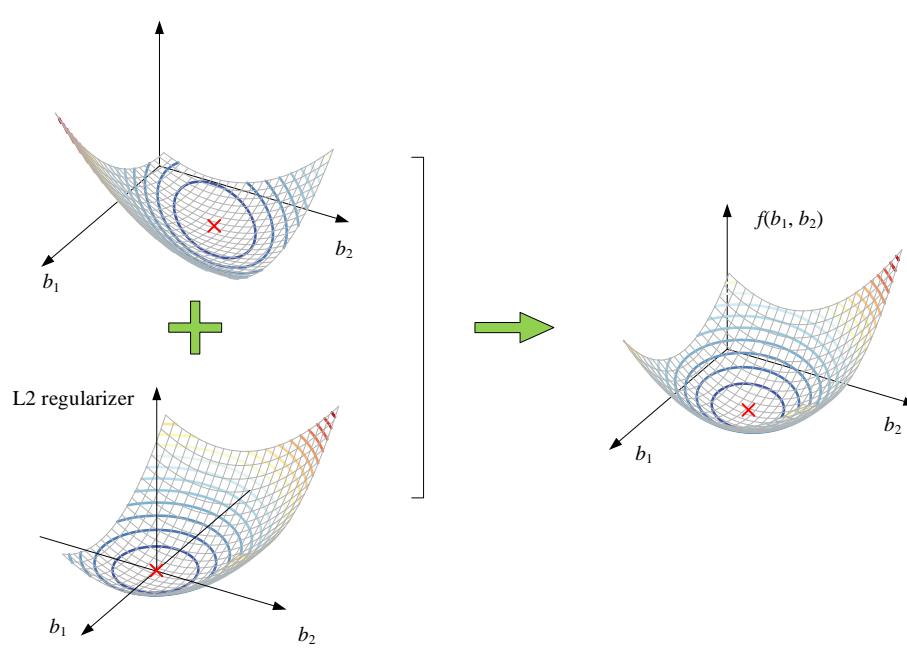


图 15. 线性回归中，L2 正则化相当于椭圆抛物面和正圆抛物面叠加

椭圆相关性质对于数据科学和机器学习很多算法至关重要，本系列丛书后续将继续探讨椭圆和其他数学知识的联系。

13.4 双曲抛物面：马鞍面

双曲抛物面 (hyperbolic paraboloid)，也叫**马鞍面** (saddle surface)，因其形状酷似马鞍而得名。双曲抛物面的一般形式为：

$$y = f(x_1, x_2) = \frac{x_1^2}{a^2} - \frac{x_2^2}{b^2} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1/a^2 & 0 \\ 0 & -1/b^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (26)$$

举个例子

图 16 所示双曲抛物面解析式为：

$$y = f(x_1, x_2) = x_1^2 - x_2^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (27)$$

观察图 16 (b)，可以发现三维等高线为一系列双曲线。而曲面中心点，也称作**鞍点** (saddle point)，鞍点既不是曲面的最大值点也不是最小值点。有关鞍点的性质，本系列丛书会逐步介绍。

本章前文看到正圆和椭圆抛物面的等高线为闭合曲线；而图 16 (b) 中等高线不再闭合。此外请大家自行在图 16 (b) 中四条黑色等高线不同点处，画出前文介绍的黄色箭头（即梯度向量）；要求是，箭头垂直该点处等高线，并指向函数增大方向（朝向暖色系）。

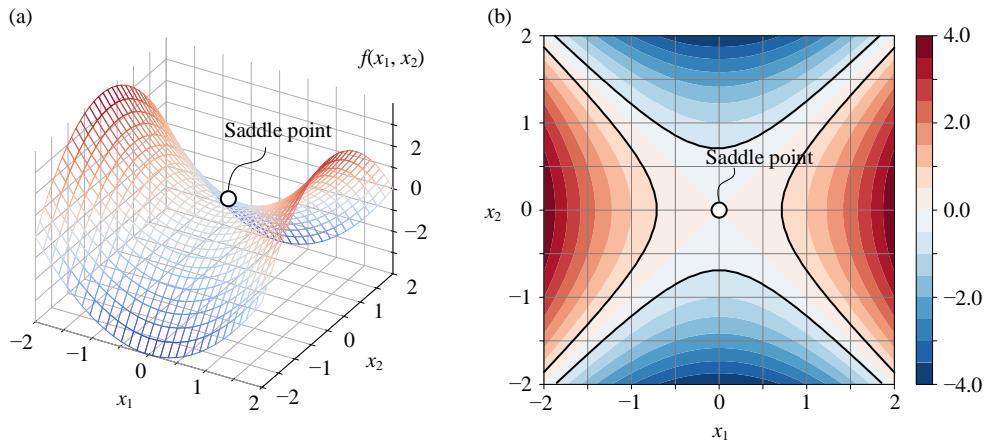


图 16. 双曲抛物面，网格图和等高线图

旋转

图 17 所示为旋转双曲线抛物面，解析式如下：

$$y = f(x_1, x_2) = x_1 x_2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 0 & 1/2 \\ 1/2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (28)$$

(28) 图 17 (b) 所示等高线实际上是一系列反比例函数曲线。

比较图 16 (b)，可以发现图 17 (b) 中双曲线旋转 45 度。

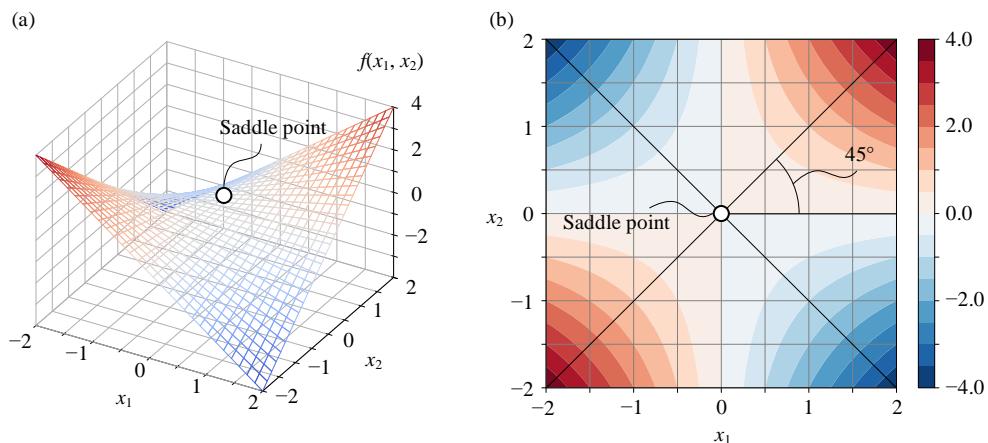


图 17. 旋转双曲抛物面，网格图和等高线图

13.5 山谷和山脊：无数极值点

本节介绍山谷面和山脊面，和它们的几何特征。

山谷面

图 18 所示为山谷面 (valley surface)，对应解析式如下：

$$y = f(x_1, x_2) = x_1^2 = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (29)$$

观察图 18 (b) 可以发现，山谷面存在无数极小值点，并且这些极小值点均在一条直线上。

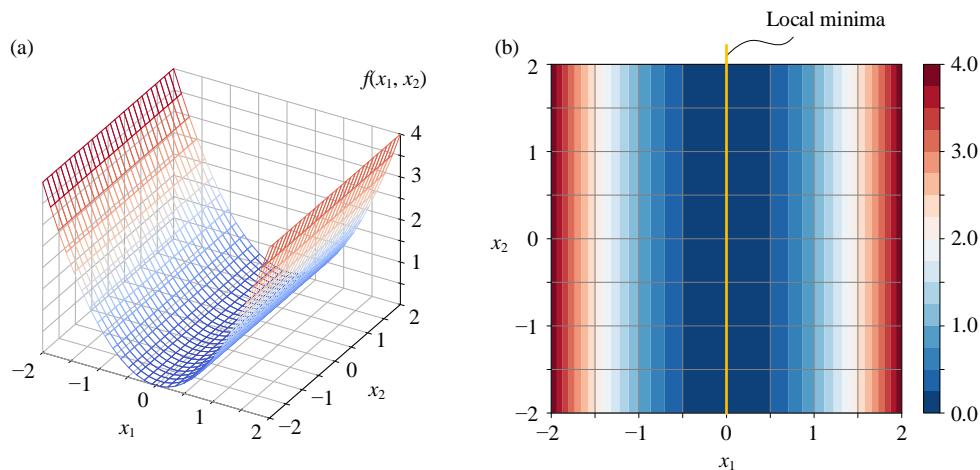


图 18. 山谷面，网格图和等高线图

叠加

如图 19 所示，如下正圆抛物面可以看做由两个山谷面叠加得到：

$$y = f(x_1, x_2) = x_1^2 + x_2^2 \quad (30)$$

很多曲面都可以看做是若干不同类型曲面叠加而成。这个几何视角对于理解一些机器学习和数据科学算法非常重要。

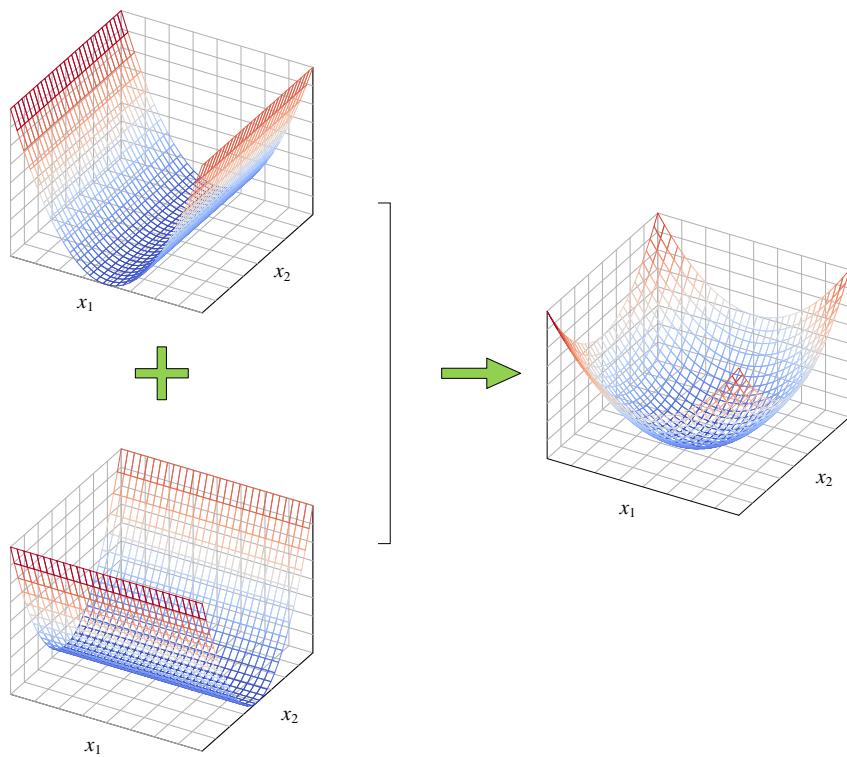


图 19. 两个山谷面合成得到正圆面

山脊面

图 20 所示为旋转**山脊面** (ridge surface), 解析式如下:

$$y = f(x_1, x_2) = -\frac{x_1^2}{2} + x_1 x_2 - \frac{x_2^2}{2} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} -1/2 & 1/2 \\ 1/2 & -1/2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad (31)$$

图 20 (b) 告诉我们, 山脊面有一系列极大值点, 它们在同一条斜线上。

也请大家在图 20 (b) 中黑色等高线不同点绘制梯度方向箭头。

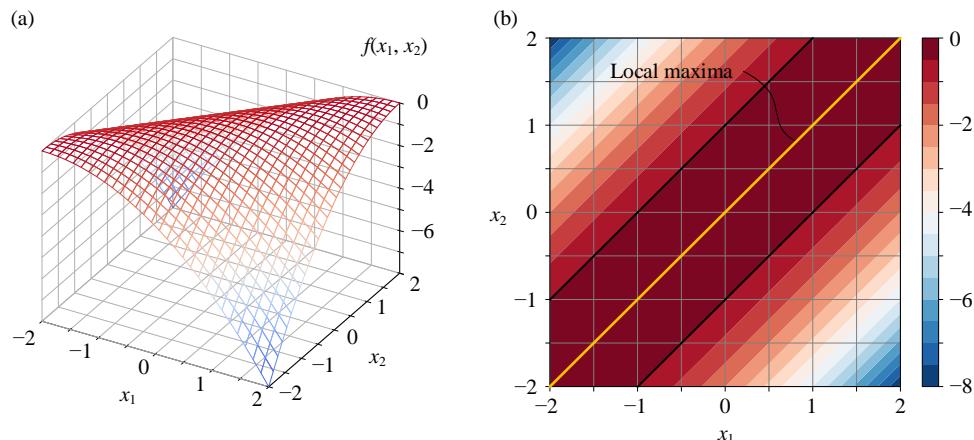


图 20. 旋转山脊面, 网格图和等高线图



大家可能已经发现本章前文介绍的平面或二次曲面都可以写成如下一般式：

$$f(x_1, x_2) = ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2 + f \quad (32)$$

在 Bk3_Ch13_01.py 基础上，我们做了一个 App 用来交互呈现不同参数对上述函数对应的曲面影响。并采用 Plotly 呈现交互 3D 曲面。请参考 Streamlit_Bk3_Ch13_01.py。

13.6 锥面：正圆抛物面开方

开口朝上

开口朝上正圆抛物面解析式开平方取正，便得到锥面。图 21 所示锥面 (cone surface) 开口朝上，对应解析式如下：

$$y = f(x_1, x_2) = \sqrt{x_1^2 + x_2^2} = \sqrt{\mathbf{x}^\top \mathbf{x}} = \|\mathbf{x}\| \quad (33)$$

观察图 21 (b) 可以发现，锥面的等高线为一系列同心圆。

图 21 所示曲面在 (0, 0) 处取得最小值。但是 (0, 0) 并不光滑，该点为尖点。

⚠ 注意， 在这个尖点处，无法找到曲面的切线或切面。

值得注意的是，图 21 (b) 中不同等高线之间均匀渐变，这显然不同于图 8 (b)。为了更好地量化比较，请大家试着写代码绘制 $y = |x|$ 和 $y = x^2$ 这两个函数，观察曲线变化大家就会理解为什么图 21 等高线均匀变化，而图 8 等高线离中心越远越密集。这个分析思路就是通过“降维”来分析二元、多元函数，即固定其他变量，观察函数随某个特定变量变化。



本书第 16 章介绍的偏导数这个工具，用的也是“降维”这个思路。

前文说过，向量模 $\|\mathbf{x}\|$ 代表向量长度，也就是距离，即欧氏距离。图 21 (b) 中不同等高线代表和 $(0, 0)$ 距离相同，这些等高线就是欧氏距离“等距线”。

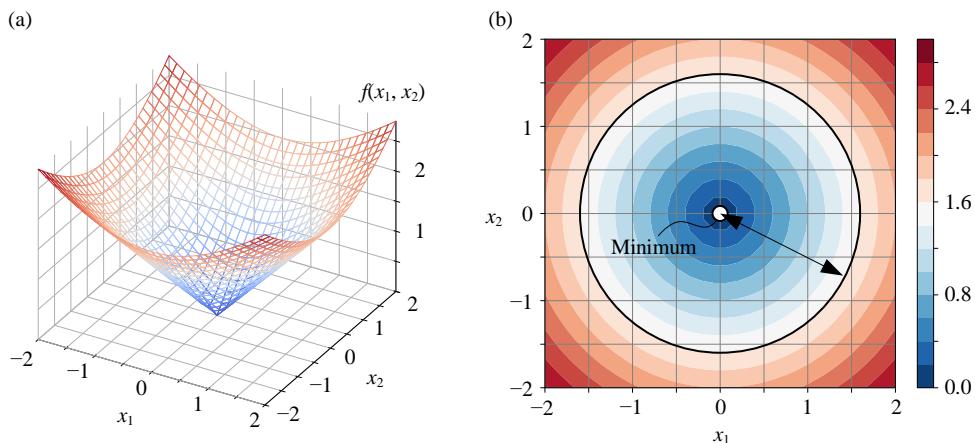


图 21. 正圆锥面，开口朝上，网格图和等高线图

开口朝下

(33) 解析式加个负号便得到如图 22 所示开口向下锥面，解析式如下：

$$y = f(x_1, x_2) = -\sqrt{x_1^2 + x_2^2} \quad (34)$$

图 22 (b) 锥面等高线同样为一系列均匀渐变同心圆，锥面在 $(0, 0)$ 取得最大值。最大值点处也是尖点。

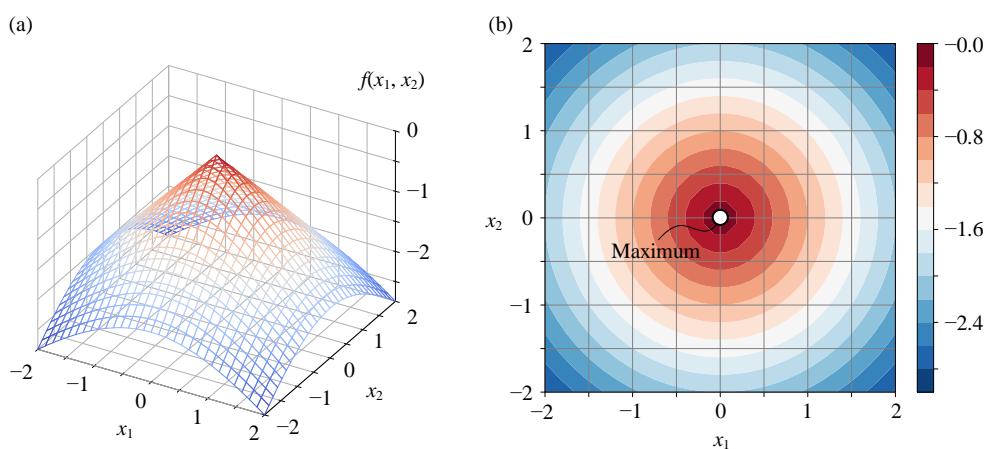


图 22. 正圆锥面，开口朝下，网格图和等高线图

对顶圆锥

中轴保持在一条直线上，将图 21 和图 22 两个圆锥面在顶点处拼接在一起便获得如图 23 所示**对顶圆锥** (double cone 或 vertically opposite circular cone)。大家在前文已经看到对顶圆锥和圆锥曲线之间的关系。

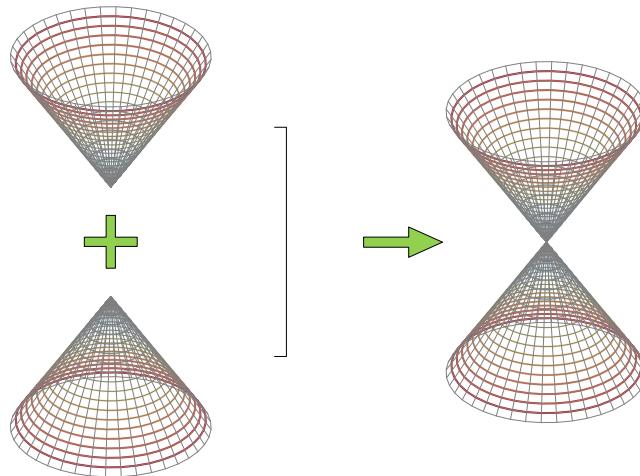


图 23. 对顶圆锥



Bk3_Ch13_02.py 绘制图 23 中开口朝上的圆锥面。注意，图 23 中网格面在极坐标系中生成。

13.7 绝对值函数：和超椭圆有关

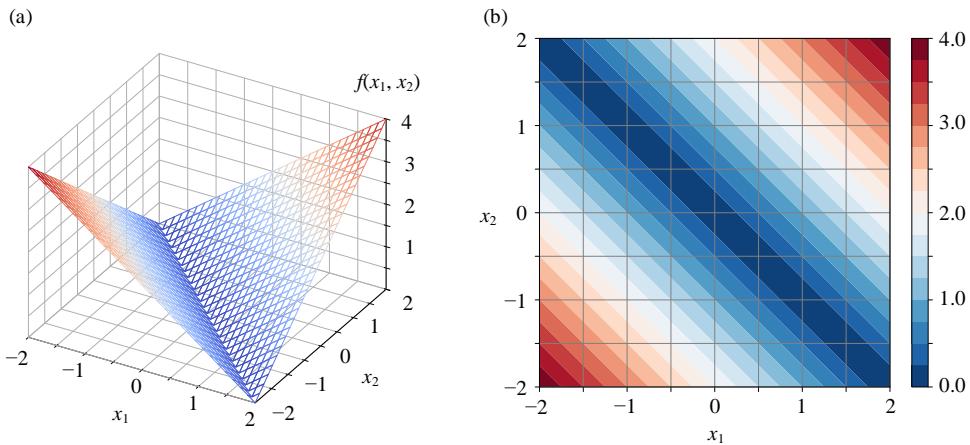
本节将绝对值函数扩展到二元，本节将构造三个不同绝对值函数。

平面对折

第一个例子， $x_1 + x_2$ 取绝对值，具体解析式如下：

$$y = f(x_1, x_2) = |x_1 + x_2| \quad (35)$$

如图 24 所示，(35) 相当于 $f(x_1, x_2) = x_1 + x_2$ 平面对折。

图 24. $f(x_1, x_2) = |x_1 + x_2|$ 空间形状

此外，(35) 相当于旋转山谷面解析式开平方取正：

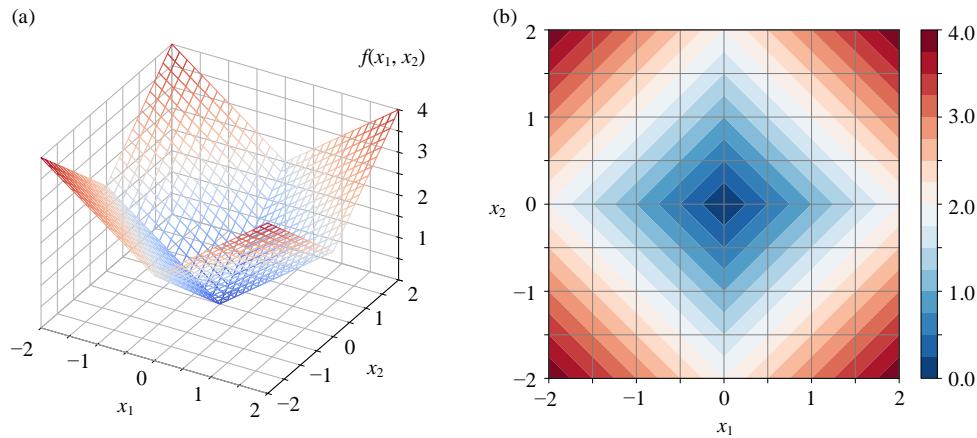
$$y = f(x_1, x_2) = \sqrt{(x_1 + x_2)^2} \quad (36)$$

旋转正方形

第二个例子， x_1 和 x_2 分别取绝对值再求和，解析式如下：

$$y = f(x_1, x_2) = |x_1| + |x_2| \quad (37)$$

图 25 所示 $f(x_1, x_2) = |x_1| + |x_2|$ 等高线图像为一系列旋转正方形。

图 25. $f(x_1, x_2) = |x_1| + |x_2|$ 空间形状

正方形

第三个绝对值函数的例子为， x_1 和 x_2 分别取绝对值，比大小后、取两者中最大值，如下：

$$y = f(x_1, x_2) = \max(|x_1|, |x_2|) \quad (38)$$

如图 26 所示， $\max(|x_1|, |x_2|)$ 对应三维等高线为正方形。

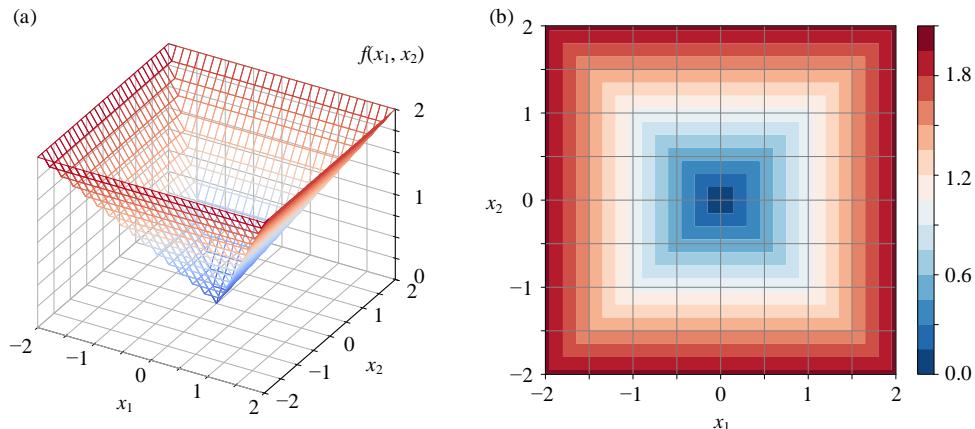


图 26. $f(x_1, x_2) = \max(|x_1|, |x_2|)$ 空间形状



本节介绍的三个绝对值函数和本书第 9 章介绍的超椭圆存在联系。此外，本系列丛书后续还会介绍它们和 L^p 范数、距离度量之间的联系。

实际上，上一节介绍的锥面也可以看做是一种绝对值函数：

$$y = f(x_1, x_2) = \sqrt{|x_1|^2 + |x_2|^2} = \|\mathbf{x}\| \quad (39)$$

⚠ 请大家注意区分绝对值和向量模这两个数学概念。



本章前文介绍过，引入正则项可以简化多元线性回归。

除了 L2 正则项，L1 正则项也经常使用。

如图 27 所示，引入 L1 正则项，相当于在旋转抛物面上叠加一个解析式为 $f(b_1, b_2) = \alpha(|b_1| + |b_2|)$ 绝对函数曲面。观察图 27 右图，发现曲面出现“折痕”，这些“折痕”来自于 L1 正则项曲面，它们破坏了曲面的光滑。

引入 L1 正则项，参数 b_1 和 b_2 位置更靠近原点。特别地，当 L1 正则项权重增大到一定程度， b_1 或 b_2 优化解可以 0。也就是说，红叉 **×** 位置可能在横轴或者纵轴上。这种特性是 L2 正则项不具备的。

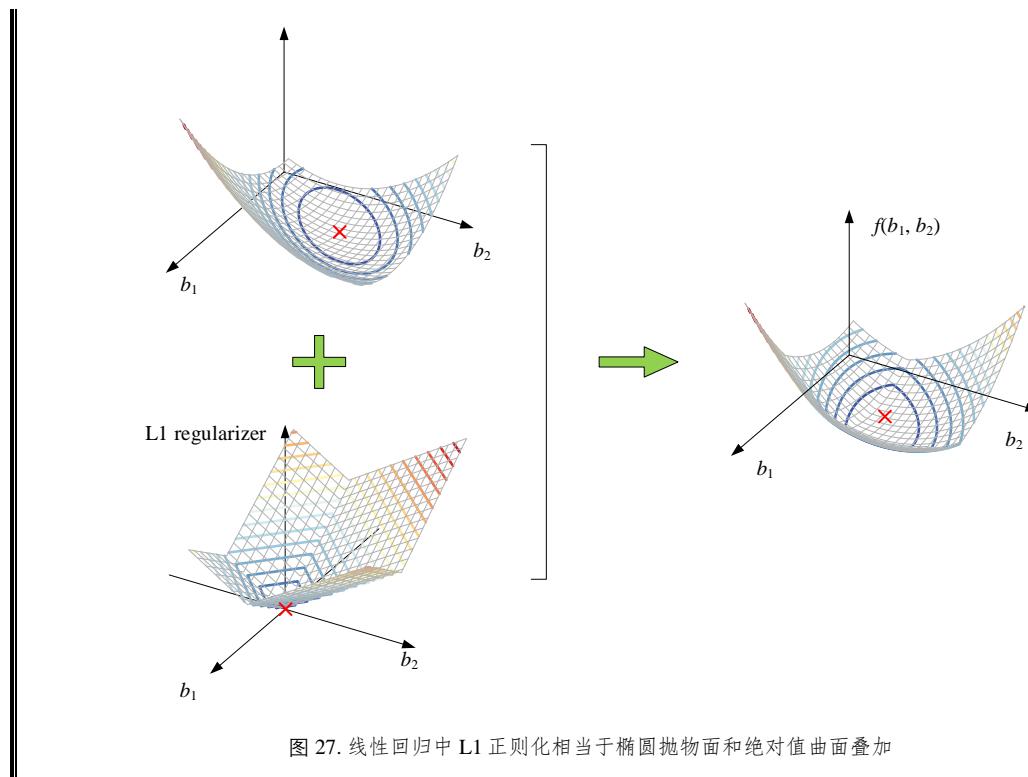


图 27. 线性回归中 L1 正则化相当于椭圆抛物面和绝对值曲面叠加

13.8 逻辑函数：从一元到二元

本节将一元逻辑函数推广到二元。二元逻辑函数对应的一般解析式如下：

$$y = f(x_1, x_2) = \frac{1}{1 + \exp(-(w_1 x_1 + w_2 x_2 + b))} \quad (40)$$

写成矩阵运算形式：

$$y = f(\mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{w}^\top \mathbf{x} + b))} \quad (41)$$

⚠ 注意，(41) 可以看做一个复合函数。

举个例子

当 $w_1 = 1, w_2 = 1, b = 0$ 时，(40) 可以写成：

$$y = f(x_1, x_2) = \frac{1}{1 + \exp(-(x_1 + x_2))} \quad (42)$$

观察图 28 曲面可以发现，当 $x_1 + x_2$ 趋近正无穷时，(42) 趋近 1，却无法达到 1。当 $x_1 + x_2$ 趋向于负无穷时，(42) 趋近 0，却无法达到 0。

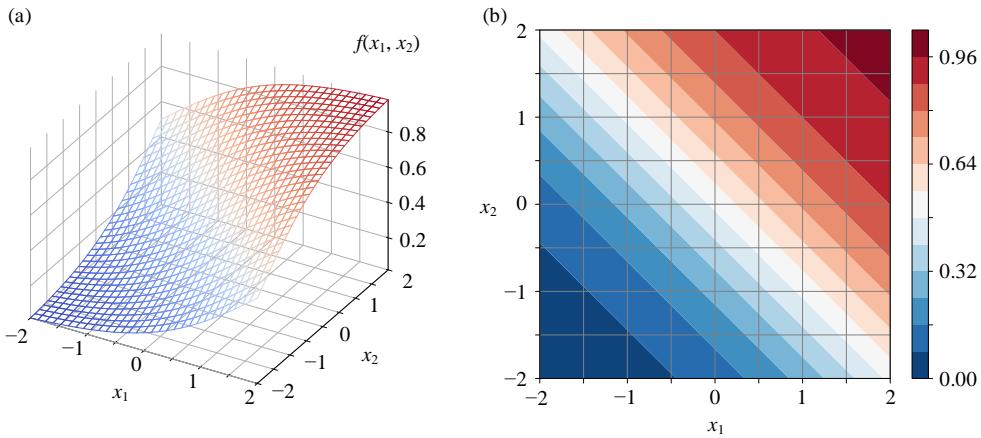


图 28. $f(x_1, x_2) = 1/(1 + \exp(-(x_1+x_2)))$ 空间形状

再举个例子

当 $w_1 = 4, w_2 = 4, b = 0$ 时，(40) 可以写成：

$$y = f(x_1, x_2) = \frac{1}{1 + \exp(-4(x_1 + x_2))} \quad (43)$$

图 29 所示为 (43) 对应的曲面。对比图 28 和图 29，不难发现，当 w_1 和 w_2 增大后，坡面变得陡峭。

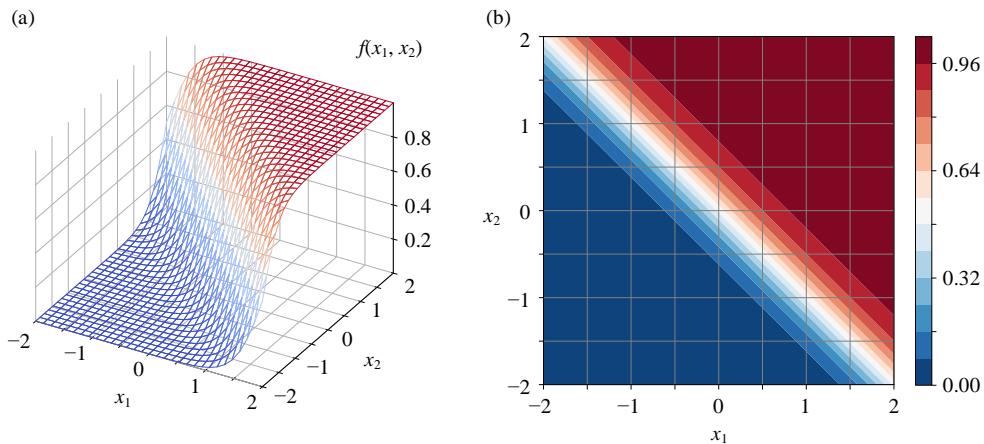


图 29. $f(x_1, x_2) = 1/(1 + \exp(-4(x_1+x_2)))$ 空间形状

二元 tanh() 函数

上一章提到，逻辑函数是 S 型函数的一种；而机器学习中，sigmoid 函数很多时候特指 $\tanh()$ 函数。二元 $\tanh()$ 函数形式如下：

$$y = f(x_1, x_2) = \tanh(\gamma(w_1 x_1 + w_2 x_2) + r) \quad (44)$$

写成矩阵运算形式：

$$y = f(\mathbf{x}) = \tanh(\gamma \mathbf{w}^T \mathbf{x} + r) \quad (45)$$

举个例子

$\gamma = 1, w_1 = 1, w_2 = 1, r = 0$ 时，

$$y = f(x_1, x_2) = \tanh(x_1 + x_2) \quad (46)$$

图 30 所示为 (46) 对应的曲面以及平面等高线。当 γ 增大时，曲面也变得陡峭。比如，图 31 对应 $\gamma = 4, w_1 = 1, w_2 = 1, r = 0$ 函数曲面。

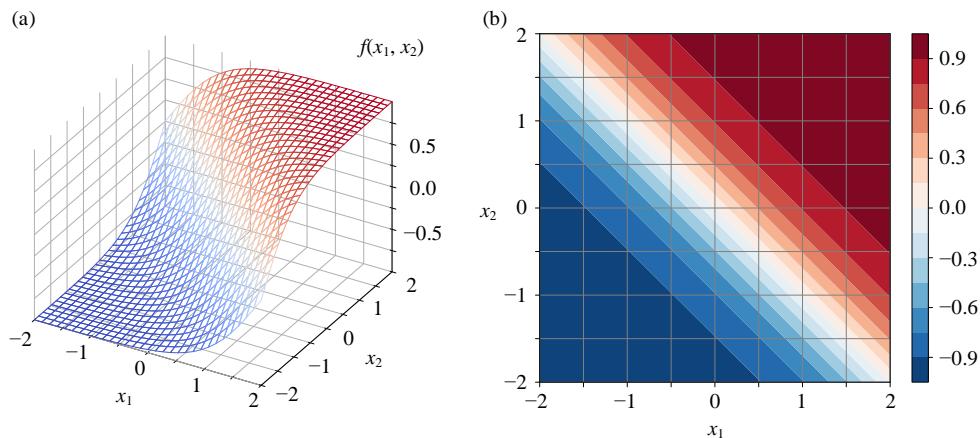


图 30. 二元 Sigmoid 核函数， $\gamma = 1, w_1 = 1, w_2 = 1, r = 0$

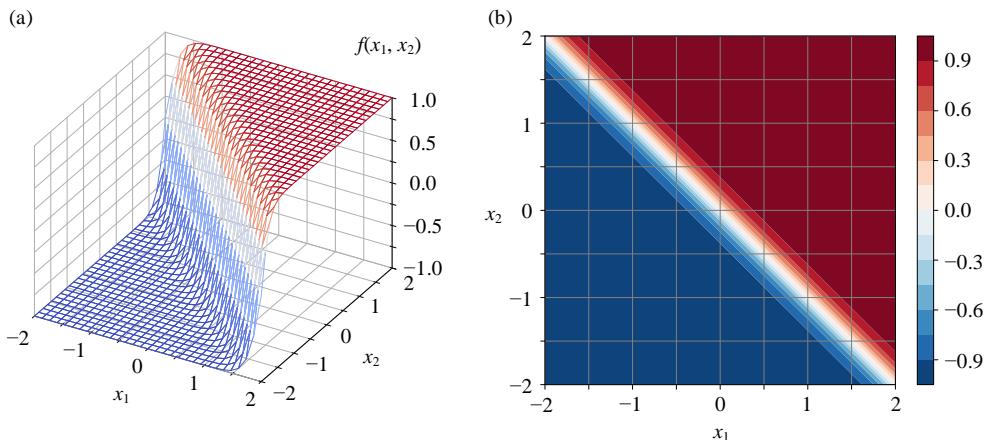


图 31. 二元 Sigmoid 核函数， $\gamma = 4, w_1 = 1, w_2 = 1, r = 0$

13.9 高斯函数：机器学习的多面手

本节将一元高斯函数推广到二元。

二元高斯函数的一般形式为：

$$y = f(x_1, x_2) = \exp\left(-\gamma((x_1 - c_1)^2 + (x_2 - c_2)^2)\right) \quad (47)$$

举个例子

$\gamma = 1, c_1 = 0, c_2 = 0$ 时，二元高斯函数函数解析式为：

$$y = f(x_1, x_2) = \exp(-(x_1^2 + x_2^2)) = \exp(-\mathbf{x}^\top \mathbf{x}) = \exp(-\|\mathbf{x}\|^2) \quad (48)$$

图 32 所示为 (48) 对应的曲面和平面等高线。

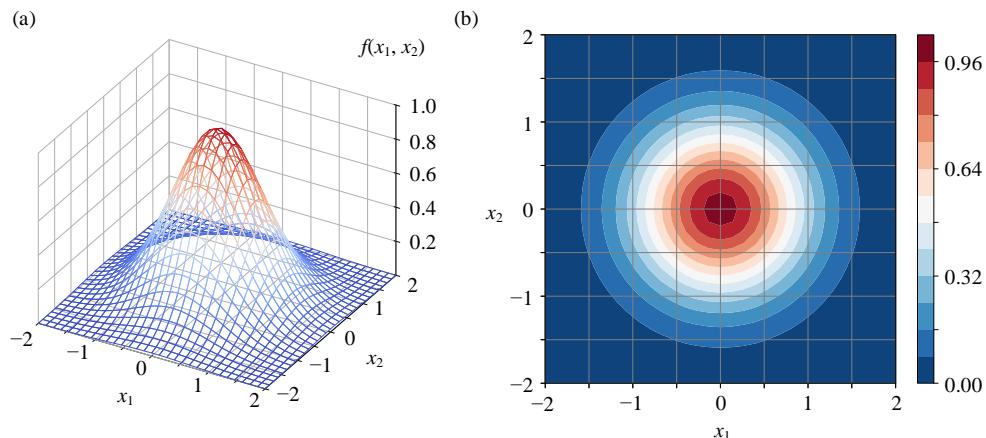


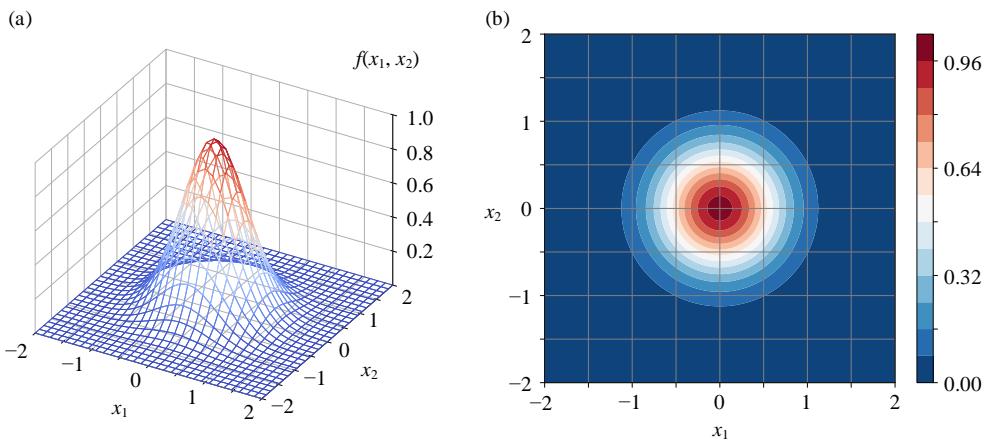
图 32. 高斯核曲面, $\gamma = 1, c_1 = 0, c_2 = 0$

再举个例子

$\gamma = 2, c_1 = 0, c_2 = 0$ 时，二元高斯函数为：

$$y = f(x_1, x_2) = \exp(-2(x_1^2 + x_2^2)) = \exp(-2\mathbf{x}^\top \mathbf{x}) = \exp(-2\|\mathbf{x}\|^2) \quad (49)$$

图 33 所示为 (49) 对应的曲面和平面等高线。比较图 32 和图 33，可以发现随着 γ 增大，曲面变得更尖，更陡峭。

图 33. 高斯核曲面, $\gamma = 2$, $c_1 = 0$, $c_2 = 0$ 

本书前文简单介绍过一种重要的机器学习方法——**支持向量机**。如图 34 所示, SVM 基本原理是找到一条灰色“宽带”, 将绿色点和蓝色点分开, 并让灰色“间隔 (margin)”最宽。

灰色“间隔”中心线 (图 34 中红色直线) 便是分割边界, 即分类**决策边界** (decision boundary)。

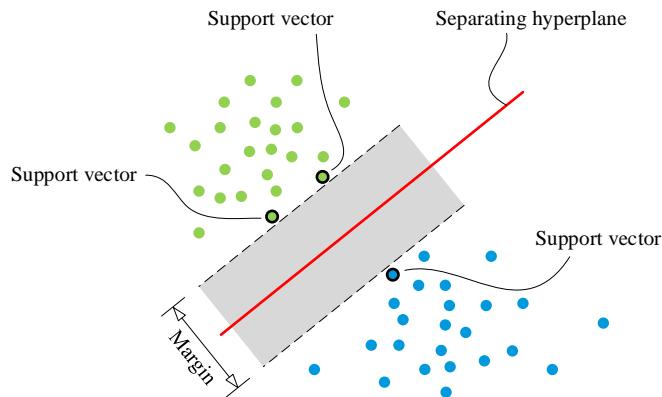


图 34. 支持向量机原理

但是, 实际情况却是, 很多数据并不能用一条直线将不同标签样本分类, 比如图 35 所示情况。

对于这种情况, 我们需要采用**核技巧** (kernel trick)。核技巧的基本思路就是将数据映射到高维空间中, 让数据在这个高维空间中线性可分。

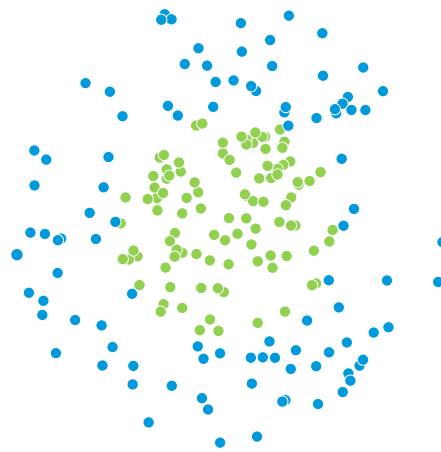


图 35. 线性不可分数据

核技巧原理如图 36 所示。原数据线性不可分，显然不能用一条直线将数据分成两类。

但是，将原来二维数据投射到三维空间之后，就可以用一个平面将数据轻易分类。这个投射规则便是**核函数** (kernel function)，而高斯函数是重要的核函数之一。图 36 (b) 右图是由若干高斯函数叠加而成。

红色等高线便是分类决策边界。

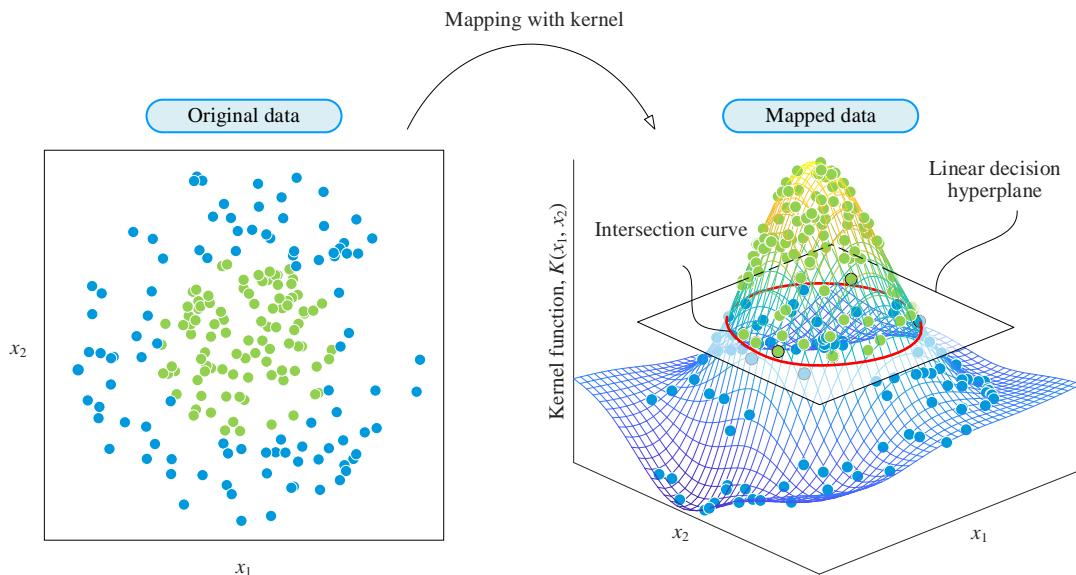


图 36. SVM 核技巧



本节将一元函数推广到二元情况，并将它们和几何、优化、机器学习联系起来。虽然，这样显得“急功近利”，但是必须承认带着“学以致用”目标学习数学将大大提高学习效率。

14 Sequences 数列 也是一种特殊函数



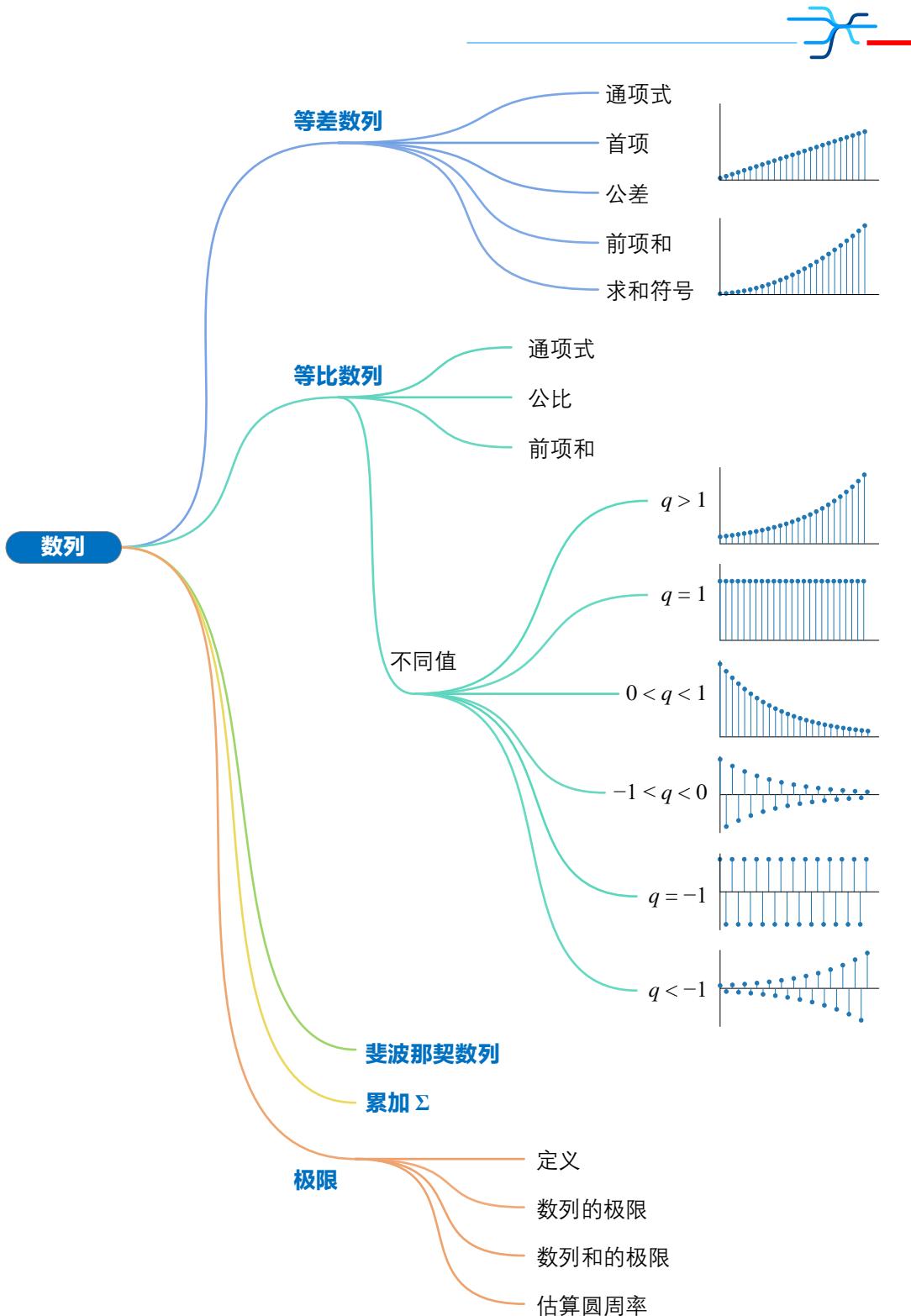
有数字的地方，就存在美。

Wherever there is number, there is beauty.

——普罗克洛 (Proclus) | 古希腊哲学家 | 412 ~ 485



- ◀ `numpy.sum()` 计算数列和
- ◀ `numpy.cumsum()` 计算累积和
- ◀ `numpy.cumprod()` 计算累积乘积
- ◀ `numpy.arange()` 根据指定的范围以及设定的步长，生成一个等差数列，数据类型为数组
- ◀ `matplotlib.pyplot.stem()` 绘制火柴梗图



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

14.1 芝诺悖论：阿基里斯追不上乌龟

芝诺悖论 (Zeno's paradoxes) 中最有名的例子莫过于“阿基里斯追乌龟”。

阿基里斯 (Achilles) 是古希腊神话中的勇士，可谓飞毛腿。而乌龟的奔跑速度仅仅是阿基里斯的 $1/10$ 。赛跑比赛时，阿基里斯让乌龟在自己前面 100 米处起跑，他自己在后面追。

根据芝诺悖论，阿基里斯不可能追上乌龟。

芝诺的逻辑是这样的，赛跑过程中，阿基里斯必须先追到 100 米处，而此时乌龟已经向前爬行了 10 米。此时，相当于乌龟还是领先阿基里斯 10 米，这算是一个新的起跑点。

阿基里斯继续追乌龟，当他跑了 10 米之后，乌龟则又向前爬了 1 米。

于是，阿基里斯还需要再追上 1 米，于此同时乌龟又向前爬了 $1/10$ 米。

如此往复，结论是阿基里斯永远也追不上乌龟。

为了方便可视化，假设乌龟爬行速度是阿基里斯奔跑速度的 $1/2$ 。设定，阿基里斯奔跑速度 10 m/s ，神龟爬 (飞) 行速度 5 m/s 。

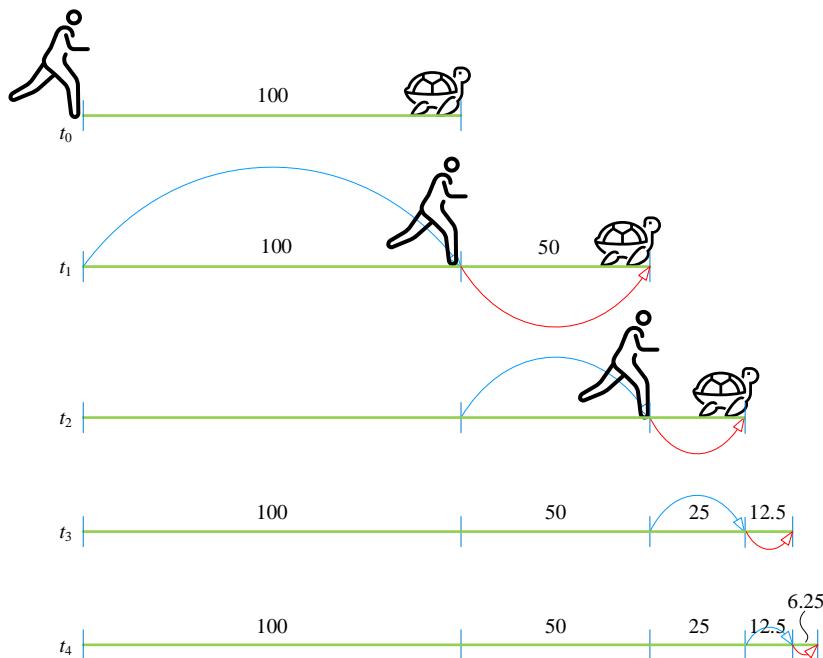


图 1. 阿基里斯追乌龟

$t_0 = 0 \text{ s}$ 时刻，乌龟在阿基里斯前方 100 米处，两者同时起跑。

$t_1 = 10 \text{ s}$, 阿基里斯跑了 10 s , 追到 100 m ; 而这段时间, 乌龟向前跑了 50 m 。因此, 此刻乌龟领先优势为 50 m 。

$t_2 = 10 + 5 \text{ s}$, 阿基里斯又跑了 5 s , 又追了 50 m 。 5 s 时间, 乌龟跑了 25 m , 而此时乌龟领先优势为 25 m 。

$t_3 = 10 + 5 + 2.5 \text{ s}$, 阿基里斯又跑了 2.5 s , 再追了 25 m 。此刻乌龟领先优势为 12.5 m 。

$t_4 = 10 + 5 + 2.5 + 1.25 \text{ s}$, 阿基里斯再追了 12.5 m , 此刻乌龟领先优势为 6.25 m 。

时间无限可分, 距离无限可分, 上述过程无穷尽也, 似乎阿基里斯永远也追不上乌龟。

同向追赶问题

看到这里, 大家一定会有不同意见。

这分明就是一道小学算数的“同向追赶问题”, 距离之差 (100 m) 除以速度之差 (5 m/s) 就等于阿基里斯追上乌龟所需的时间为 20 s 。而 20 s 时间, 阿基里斯一共跑了 200 m 。

这种解题思路固然正确。但是, 解题过程的前提条件是, 假设阿基里斯恰好追上了乌龟!

解题技巧当然重要。实际上, 看似无比荒诞的阿基里斯追乌龟问题, 其中蕴含的数学思想才是内核。

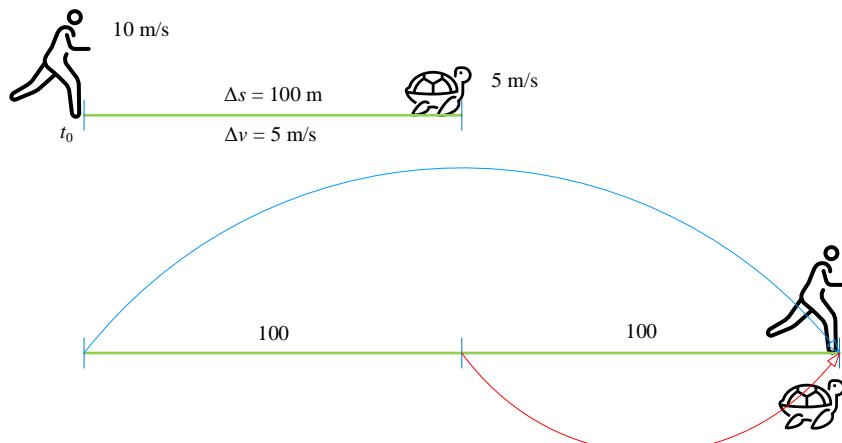


图 2. 小学数学同向追趕問題

一尺之棰，日取其半，万世不竭

下面用数列这个数学工具来分析上述问题。数列 (sequence) 是指按照一定规则排列的一列数。

将图 1 中每段时间间隔写成一个数列:

$$10, 5, 2.5, 1.25, 0.625, 0.3125, 0.15625, 0.078125, \dots \quad (1)$$

图 3 (a) 用火柴梗图可视化 (1) 数列。

追赶时间的逐项和也写成一个数列累加：

$$10, 15, 17.5, 18.75, 19.375, 19.6875, 19.84375, 19.921875, \dots \quad (2)$$

图 3 (b) 所示火柴梗图为 (2) 数列。可以发现上述数列似乎逐渐趋向于 20，即阿基里斯追上乌龟所需要的时间。

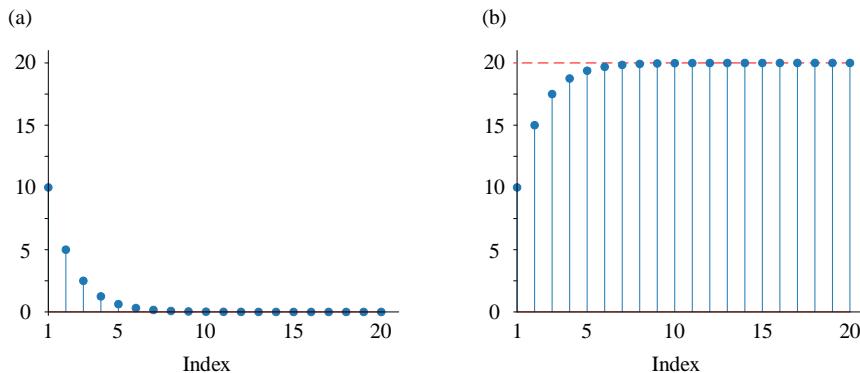


图 3. 时间间隔数列和时间逐项和数列

将阿基里斯在不同时刻之间奔跑的距离写成一列数：

$$100, 50, 25, 12.5, 6.25, 3.125, 1.5625, 0.78125, \dots \quad (3)$$

容易发现，这个数列就是大家熟悉的等比数列。上述数列的逐项和构成了一个新数列：

$$100, 150, 175, 187.5, 193.75, 196.875, 198.4375, 199.21875, \dots \quad (4)$$

可以发现上述数列似乎逐渐趋向于 200，即阿基里斯追上乌龟总共奔跑的距离。

这体现的正是庄子的哲学观点——“一尺之棰，日取其半，万世不竭。”如图 4 所示，面积为 1 的正方形，每次取一半，如此往复，没有尽头。

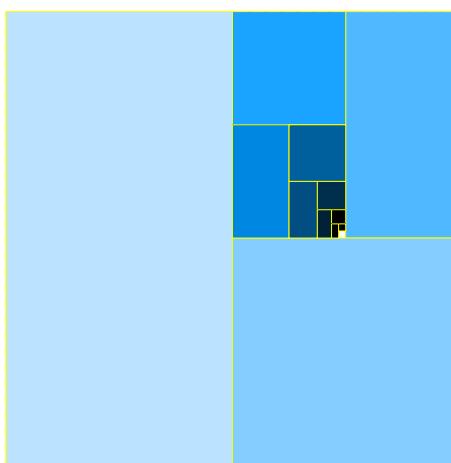


图 4. 日取其半，万世不竭

大家经常会遇到类似比较 1 和 0.99999... 大小之类的数学问题，其中蕴含的数学思想就是极限。本章就讲解数列、数列前 n 项和、极限等数学工具。

14.2 数列分类

几种常见的数列：

- ◀ **等差数列** (arithmetic sequence 或 arithmetic progression)，图 5 (a) 为递增等差数列，图 5 (b) 为递减等差数列；
- ◀ **等比数列** (geometric sequence 或 geometric progression)，图 5 (c) 为递增等比数列，图 5 (d) 为递减等比数列；
- ◀ **正负相间数列** (sign sequence 或 bipolar sequence)，如图 5 (e) 和 (f) 所示；
- ◀ **斐波那契数列** (Fibonacci sequence)，如图 5 (g) 所示；
- ◀ **随机数列** (random sequence)，如图 5 (h) 所示。

此外，根据数列项的数量，数列可以分为**有限项数列** (finite sequence) 和**无限项数列** (infinite sequence)。

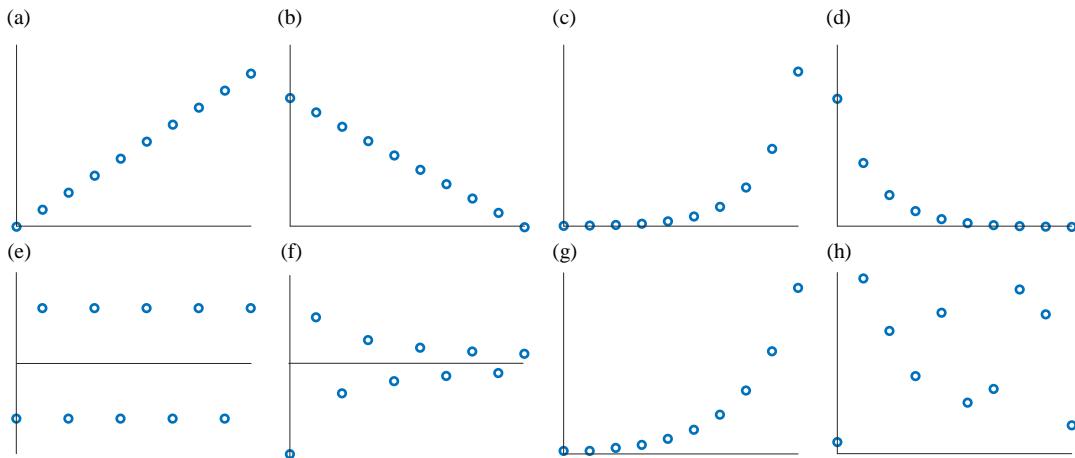


图 5. 几种数列

14.3 等差数列：相邻两项差相等

等差数列指的是数列中任何相邻两项的差相等，比如 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...

等差数列中相邻两项差值常被称作**公差** (common difference)。将数列的第 k 项用一个具体含有参数 k 式子表示出来，称作该数列的通项公式。

等差数列通项公式 a_k 的一般式如下：

$$a_k = a + (k - 1) \cdot d \quad (5)$$

a_1 (读作 a sub one) 为数列第一项，即**首项** (initial term) $a_1 = a$; d 为公差； k 为**项数** (number of terms)。numpy.arange() 和 numpy.linspace() 可以用来生成等差数列。

(5) 所示等差数列前 k 项之和为 S_k :

$$\begin{aligned} S_k &= a + (a + d) + (a + 2d) + \dots + (a + (k - 1) \cdot d) \\ &= \sum_{i=1}^k (a + (i - 1) \cdot d) = a \cdot k + \frac{k(k - 1)}{2} \cdot d \end{aligned} \quad (6)$$

上式中， i 为**索引** (index) 也叫序号； Σ 是求和符号，是希腊字母 σ 的大写，读作 sigma。numpy.sum() 可以用来计算数列和。

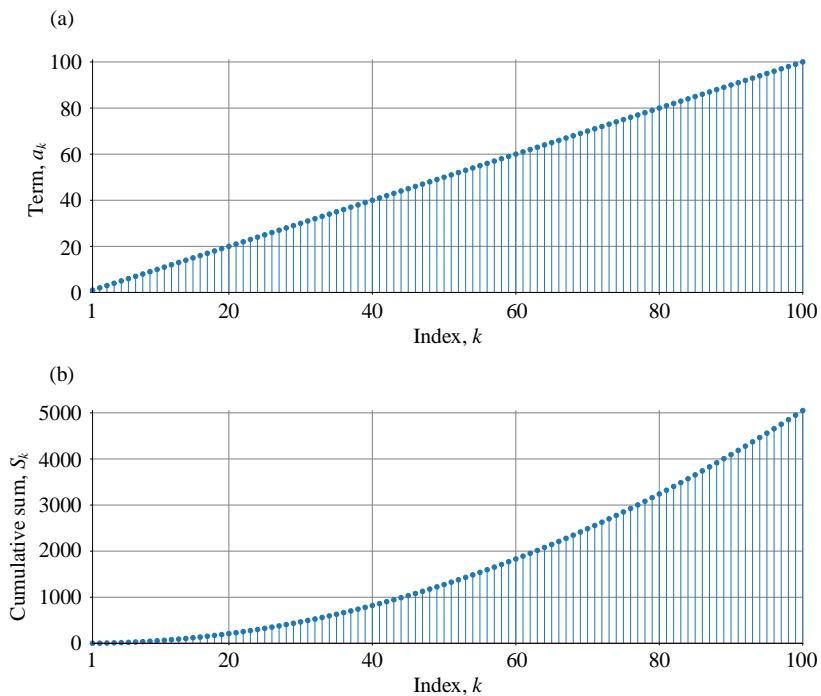
⚠ 注意区分， Π 是求积符号，它是希腊字母 π 的大写。

欧拉 (Leonhard Euler) 最先使用 Σ 来表达求和。

相信读者还记得，等差数列求和的计算方法——首项加末项之和，乘以项数，然后除 2。相传，这个等差数列求和方法是**高斯** (Johann Carl Friedrich Gauss) 年仅 10 岁的时候发现的。

本书 1 章介绍，给定一列数，除了求和之外，还有**累计求和** (cumulative sum)。比如，等差数列 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 的累积和为 1, 3 (1 + 2), 6 (1 + 2 + 3), 10 (1 + 2 + 3 + 4), 15, 21, 28, 36, 45, 55。累积和的最后一项也是数列之和。numpy.cumsum() 可以用来计算数列累计求和。

下面，我们编写一段 Python 代码，计算等差数列 1, 2, 3, 4, 5, ..., 99, 100 之和，以及数列累积和。并绘制图 6 两图；图 6 (a) 为 a_k 随序数变化，图 6 (b) 为 S_k 和随序数变化。

图 6. 等差数列 $1, 2, 3, 4, 5, \dots, 99, 100$ 和数列累积和

函数视角

从函数角度，数列也是函数， k 为自变量， a_k 为因变量。需要特殊强调的是 k 的取值为正整数。如图 6 (a) 所示，(5) 可以看做特殊的一次函数。

也从函数角度来看，(6) 中 k 为自变量、取值同样为正整数， S_k 为因变量。如图 6 (b) 所示，(6) 可以看做特殊的二次函数。

数列作为一种特殊的函数，也具有各种函数性质。

$d > 0$ ，如图 7 (a) 所示数列 a_k 递增； $d < 0$ ，如图 7 (c) 所示数列 a_k 递减。

$d > 0$ ，如图 7 (b) 所示 S_k 图像开口向上，呈现出凸性； $d < 0$ ，数列递减，如图 7 (d) 所示 S_k 图像开口向下，呈现出凹性。

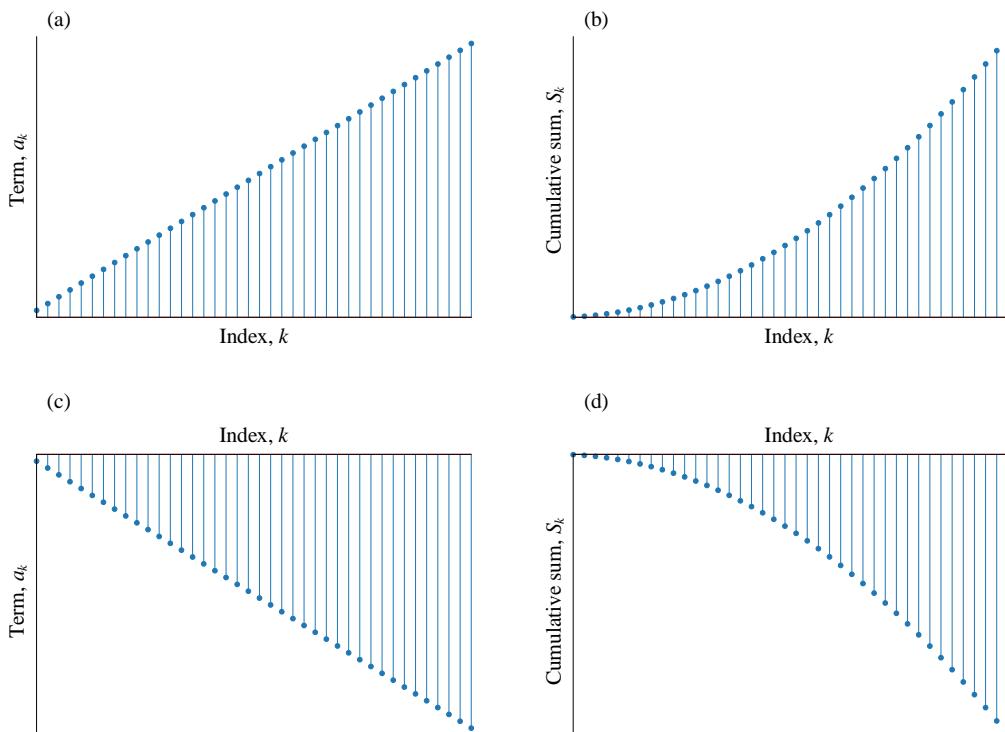


图 7. 公差为正、负两种情况

表 1. 数列英文表达

数学表达	英文表达
$a_n + a_{n-1} + \dots + a_1 + a_0$	a sub n plus a sub n minus one plus dot dot dot plus a sub one plus a sub zero. a sub n plus a sub n minus one plus ellipsis plus a sub one plus a sub zero.
$a_n \cdot a_{n-1} \cdot \dots \cdot a_1 \cdot a_0$	a sub n times a sub n times one plus dot dot dot times a sub one times a sub zero.
$a_0 + x(a_1 + x(a_2 + \dots))$	a sub zero plus x times quantity of a sub one plus x times quantity of a sub two plus dot dot dot
$(a_n - a_{n-1})^2$	a sub n minus a sub quantity n minus one all squared.



Bk3_Ch14_01.py 计算并绘制图 6。

14.4 等比数列：相邻两项比值相等

等比数列指的是数列中任何相邻两项比值相等，比如 $2, 4, 8, 16, 32, 64, 128, \dots$

等比数列的比值被称作为**公比** (common ratio)。等比数列第 k 项 a_k 的一般式如下：

$$a_k = aq^{k-1} = \frac{a}{q} q^k \quad (7)$$

其中， a 为首项， q 为公比。

⚠ 注意 q 不为 0。

从函数角度，(7) 为特殊的指数函数—— q 为底数，自变量 k 为指数， k 的取值范围为正整数。

(7) 所示等比数列前 k 项之和 S_k 为：

$$\begin{aligned} S_k &= a + aq + aq^2 + aq^3 + \cdots + aq^{k-1} \\ &= \sum_{i=0}^{k-1} a \cdot q^i = \frac{a(q^k - 1)}{(q - 1)} = \frac{a}{q-1} q^k - \frac{a}{q-1} \end{aligned} \quad (8)$$

⚠ 注意，上式中 q 不为 1。从函数角度，(7) 也是指数函数。

请大家回忆等比数列求和技巧。首先，计算 S_k 和 q 乘积：

$$S_k q = aq + aq^2 + aq^3 + \cdots + aq^{k-1} + aq^k \quad (9)$$

(9) 和 (8) 等式左右分别相减，并整理得到 S_k ：

$$\begin{aligned} S_k q - S_k &= S_k (q - 1) = aq^k - a = a(q^k - 1) \\ \Rightarrow S_k &= \frac{a(q^k - 1)}{(q - 1)} \end{aligned} \quad (10)$$

函数视角

图 8 展示六种等比 q 取不同值时，等比数列的特点。

当 $q > 1$ 时，等比数列呈现出**指数增长** (exponential growth)，如图 8 (a) 所示。

当 $q = 1$ 时，等比数列退化成常数数列，如图 8 (b) 所示。

当 $0 < q < 1$ 时，等比数列呈现出**衰退** (decay)，如图 8 (c) 所示。

当 $-1 < q < 0$ 时，等比数列呈现两种特性：**振荡** (oscillate) 和**收敛** (converge)，如图 8 (d) 所示。

当 $q = -1$ 时，等比数列只是反复振荡，也就是正负相间数列，如图 8 (e) 所示。

当 $q < -1$ 时，等比数列振荡**发散** (diverge)，如图 8 (e) 所示。

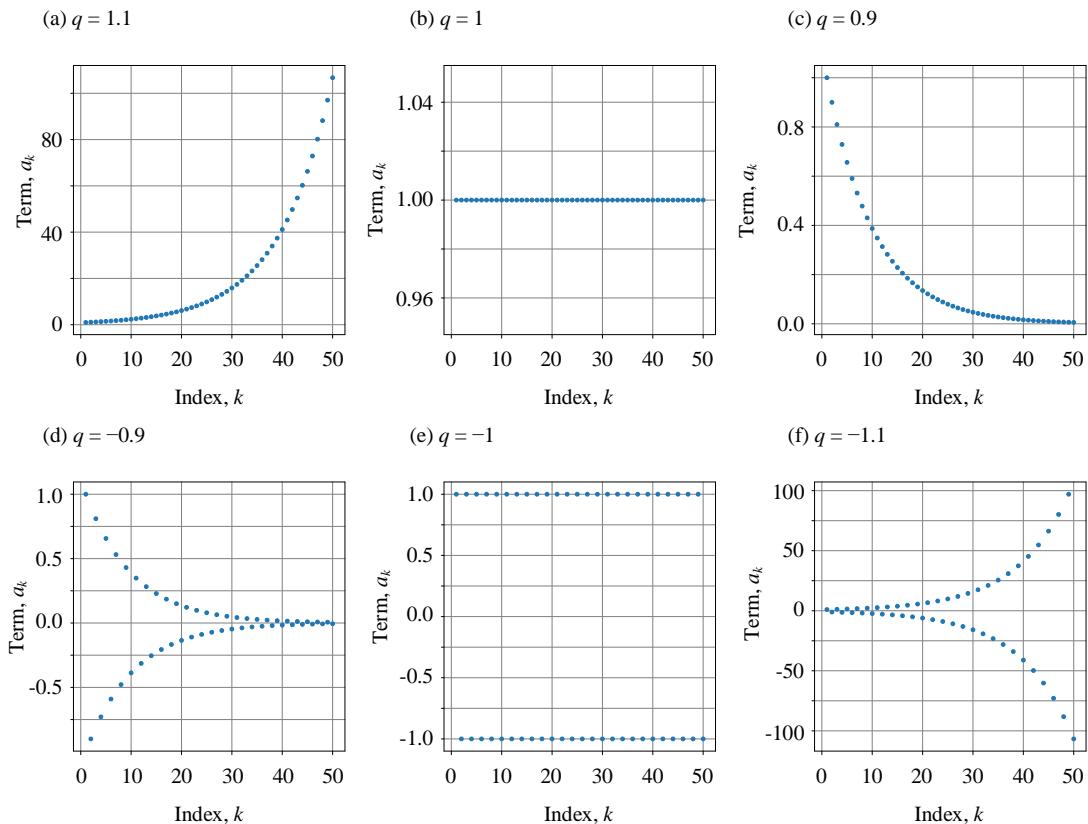
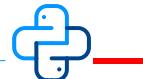


图 8. 六种等比数列趋势



Bk3_Ch14_02.py 绘制图 8 几幅子图。



在 Bk3_Ch14_02.py 基础上，我们做了一个 App 用来交互呈现 q 对数列的影响。并采用 Plotly 呈现交互散点图像。请参考 Streamlit_Bk3_Ch14_02.py。



数列在大数据和机器学习中有着广泛应用，下面举一个例子介绍等比数列在 **指数加权移动平均** (Exponentially Weighted Moving Average, EWMA) 方法的应用。

一般情况，求解**平均值** (Simple Average, SA) 时，对不同时间点的观察值赋予相同的权重，比如下式：

$$SA = \frac{1}{n} (s_1 + s_2 + s_3 + \dots + s_n) \quad (11)$$

其中，所有观察值中 s_1 为最旧的数据， s_n 为最新数据。

采用 EWMA 可以保证越新的观察值享有更高的权重，这样估算得到的平均值能够反映数据近期趋势：

$$EWMA = \frac{(1-\lambda)}{1-\lambda^n} (\lambda^{n-1}s_1 + \lambda^{n-2}s_2 + \lambda^{n-3}s_3 + \dots + \lambda^0s_n) \quad (12)$$

(12) 中的 λ 为 **衰减系数** (decay factor)，取值范围在 0 ~ 1 之间。 λ 越小，衰减越明显。

可以发现索引为 i 的权重 w_i 计算式如下所示：

$$w_i = \frac{(1-\lambda)}{1-\lambda^n} \lambda^{n-i} \quad (13)$$

索引连续变化时，权重 w_i 便构成一个等比数列。图 9 所示为 EWMA 权重随衰减系数变化情况。

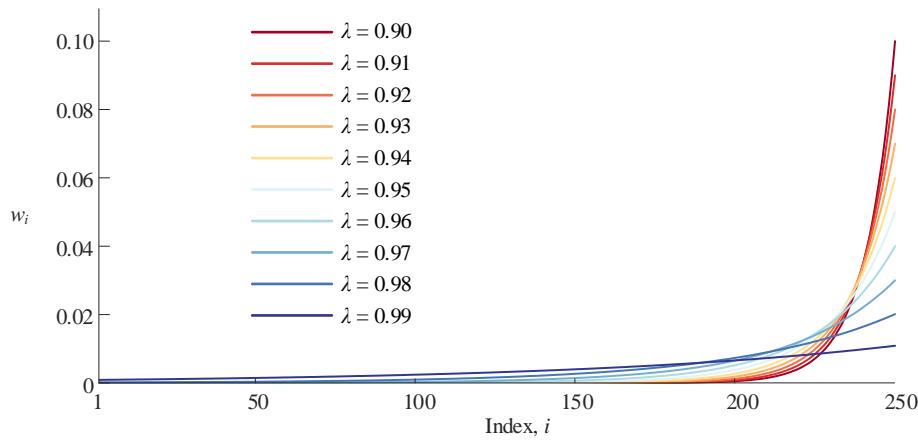


图 9. EWMA 权重随衰减系数变化

EWMA 权重一个重要的性质是，所有权重之和为 1，也就是，

$$\sum_{i=1}^n w_i = \frac{(1-\lambda)}{1-\lambda^n} (\lambda^{n-1} + \lambda^{n-2} + \lambda^{n-3} + \dots + \lambda^0) = 1 \quad (14)$$

更多有关时间序列、移动平均 MA、EWMA 方法及其应用，请读者阅读本系列丛书《数据科学》一册。

14.5 斐波那契数列

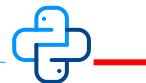
本书第 4 章介绍过斐波那契数列和杨辉三角的关系。斐波那契数列 (Fibonacci sequence)，又被称作黄金分割数列。

斐波那契数列可以通过如下递归 (recursion) 方法获得：

$$\begin{cases} F_0 = 0 \\ F_1 = F_2 = 1 \\ F_n = F_{n-1} + F_{n-2}, \quad n > 2 \end{cases} \quad (15)$$

于是，包括第 0 项，斐波那契数列的前 10 项为：

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 \quad (16)$$



Bk3_Ch14_03.py 产生并打印斐波那契数列。

黄金分割

斐波那契数列和**黄金分割** (golden ratio) 有着密切联系。图 10 所示为利用斐波那契数列构造的矩形，这个矩形是对黄金分割矩形的近似。黄金矩形的长宽比例 φ 为：

$$\varphi = \frac{\sqrt{5} + 1}{2} \approx 1.61803 \quad (17)$$

图 10 所示矩形的长宽比例为：

$$\frac{21+13}{21} \approx 1.61905 \quad (18)$$

图 10 所示的螺旋线叫做**斐波那契螺旋线** (Fibonacci spiral)，它是对**黄金螺旋线** (golden spiral) 的近似。



本系列丛书将在《矩阵力量》一本介绍如何用**特征值分解** (eigen decomposition) 求解斐波那契数列通项式。

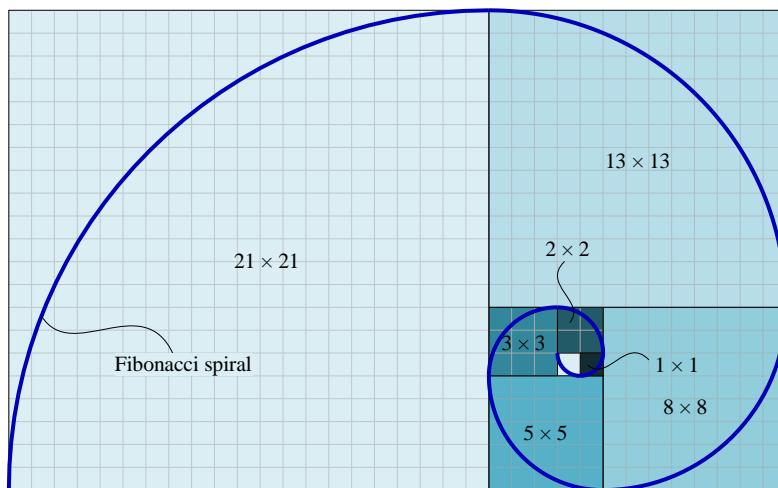


图 10. 斐波那契数列和黄金分割关系

14.6 累加：大写西格玛

求和符号 (summation symbol) —— 大写西格玛 Σ (capital sigma) —— 是表达求和的便捷记法。

以下式为例， a_i 描述求和中的每一项，下角标 i 代表**索引** (index variable 或 index)，也叫序号：

$$\sum_{i=1}^n a_i = a_1 + a_2 + \dots + a_{n-1} + a_n \quad (19)$$

Σ 下侧和上侧的数字分别代表了**求和索引下限** (lower bound of summation) 和**求和索引上限** (upper bound of summation)，如图 11 所示。

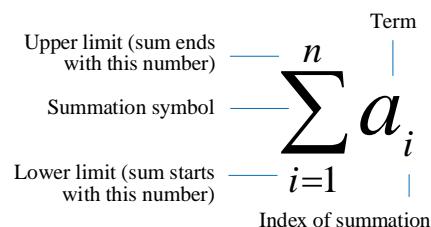


图 11. 大西格玛求和记号

常用表达索引的字母有 i 、 j 、 k 、 m 、 n 等，采用什么索引字母并不影响求和结果，比如：

$$\sum_{i=1}^{100} a_i = \sum_{j=1}^{100} a_j = \sum_{k=1}^{100} a_k \quad (20)$$

Σ 中索引上下限可以都是具体正整数，比如：

$$\sum_{i=1}^5 a_i = a_1 + a_2 + a_3 + a_4 + a_5 \quad (21)$$

Σ 中索引上下限也可以都是代数符号，比如：

$$\sum_{i=m}^n a_i = a_m + a_{m+1} + \cdots + a_{n-1} + a_n \quad (22)$$

Σ 的索引也可以是满足集合运算的标签：

$$\sum_{i \in S} a_i \quad (23)$$

降维

如图 12 所示，当索引 i 在一定范围变化时，比如 $1 \sim n$ ，数列 $\{a_i\}$ ($i = 1 \sim n$) 本身相当于一个数组，而索引 i 像是方向。

对 $\{a_i\}$ ($i = 1 \sim n$) 求和，相当于在 i 方向上将数组“压扁”，得到一个标量 $\sum_i a_i$ 。 $\sum_i a_i$ 除以 n (也就是数列元素个数)，便得到平均数。

从空间角度来看，数列 $\{a_i\}$ 是一维数组，索引 i 就是它的维度。而求和运算 $\sum_i a_i$ 相当于“降维”，得到的“和”只是一个数值，没有维度。

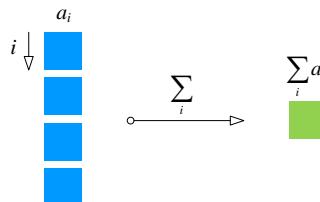


图 12. 从数组角度看求和

线性代数运算

看到这里，大家是否想得到，此处图 12 数列 $\{a_i\}$ 相当于一个列向量 a ，即：

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad (24)$$

本书第 2 章在讲解向量和矩阵时提到过，对列向量 a 所有元素求和可以利用如下向量内积或矩阵运算得到，

$$\sum_{i=1}^n a_i = \mathbf{I} \cdot \mathbf{a} = \mathbf{a} \cdot \mathbf{I} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \mathbf{a}^\top \mathbf{I} = \mathbf{I}^\top \mathbf{a} = [1 \ 1 \ \cdots \ 1] @ \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad (25)$$

其中， \mathbf{I} 就是全 1 列向量，和 \mathbf{a} 等长。

这样，我们就把数列求和、向量、向量内积、矩阵乘法这几个概念联系起来。

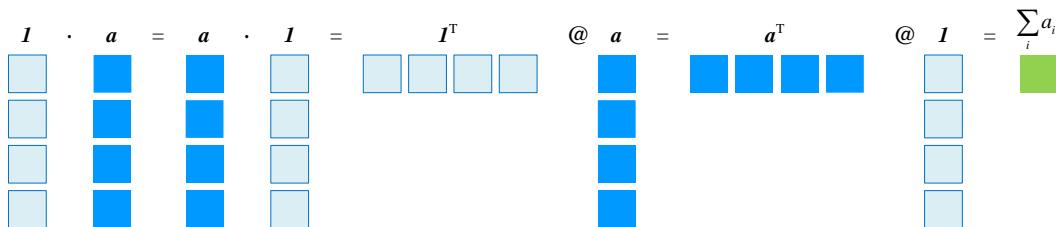


图 13. 从向量内积和矩阵乘法角度看求和

下面介绍几个有关求和符号的重要法则。

对数运算把连乘变连加

类似 Σ , Π (capital pi) 用来标记多项相乘, 比如:

$$\prod_{i=1}^n a_i = a_1 a_2 \cdots a_{n-1} a_n \quad (26)$$

本书第 12 章提到, 对数运算将连乘转化为连加, 比如:

$$\ln\left(\prod_{i=1}^n a_i\right) = \ln a_1 + \ln a_2 + \cdots + \ln a_{n-1} + \ln a_n = \sum_{i=1}^n \ln a_i \quad (27)$$

乘系数

常数 c 乘 a_i , 再求和 $\sum_{i=1}^n c a_i$, 等同于常数 c 乘 $\sum_{i=1}^n a_i$:

$$\sum_{i=1}^n c a_i = c \left(\sum_{i=1}^n a_i \right) = c \sum_{i=1}^n a_i \quad (28)$$

其中, $c \times$ 相当于“缩放”, \sum_i 相当于“降维”。如图 14 所示, (28) 相当于在说, “缩放 \rightarrow 降维”等价于“降维 \rightarrow 缩放”。

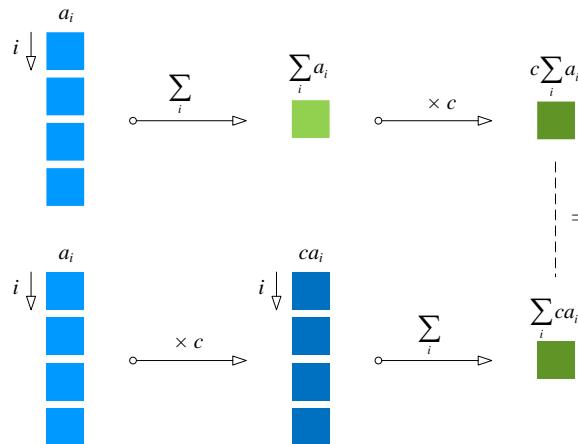


图 14. 乘系数

特别地，如果 Σ 内为一常数 a ，则，

$$\sum_{i=1}^n a = na \quad (29)$$

分段求和

可以根据索引排列，将求和分割成几个部分分别求和，比如：

$$\begin{aligned} \sum_{i=1}^n a_i &= (a_1 + a_2 + \dots + a_k) + (a_{k+1} + a_{k+2} + \dots + a_n) \\ &= \sum_{i=1}^k a_i + \sum_{i=k+1}^n a_i \end{aligned} \quad (30)$$

分段求和常用在对齐不同长度数组，以便化简计算。

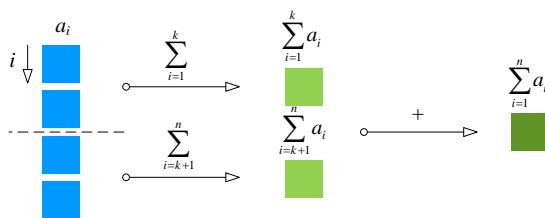


图 15. 分段求和

两项相加减

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

拥有相同索引的两项相加再求和，等于分别求和再相加，即，

$$\sum_{i=1}^n (a_i + b_i) = \sum_{i=1}^n a_i + \sum_{i=1}^n b_i \quad (31)$$

上述法则也适用于减法，即，

$$\sum_{i=1}^n (a_i - b_i) = \sum_{i=1}^n a_i - \sum_{i=1}^n b_i \quad (32)$$

平方

Σ 内为 a_i 的平方，则，

$$\sum_{i=1}^n (a_i^2) = \sum_{i=1}^n a_i^2 = a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2 \quad (33)$$

如图 16 所示，利用前文 (24) 定义的列向量 \mathbf{a} ，(33) 等价于：

$$\sum_{i=1}^n a_i^2 = \mathbf{a} \cdot \mathbf{a} = \mathbf{a}^\top \mathbf{a} \quad (34)$$

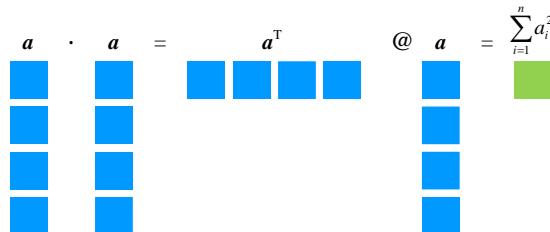


图 16. 从向量内积和矩阵乘法角度看 $\sum_{i=1}^n a_i^2$

$\sum_{i=1}^n a_i$ 的平方则为：

$$\left(\sum_{i=1}^n a_i \right)^2 = (a_1 + a_2 + a_3 + \cdots + a_n)^2 \quad (35)$$

显然，(33) 和 (35) 不相同：

$$\sum_{i=1}^n (a_i^2) \neq \left(\sum_{i=1}^n a_i \right)^2 \quad (36)$$

乘法

拥有相同索引的 a_i 和 b_i 相乘，再求和：

$$\sum_{i=1}^n (a_i b_i) = a_1 b_1 + a_2 b_2 + a_3 b_3 + \cdots + a_n b_n \quad (37)$$

定义列向量 \mathbf{b} , \mathbf{b} 和 \mathbf{a} 形状相同， \mathbf{b} 的元素为 b_i 。如图 17 所示， $\sum_{i=1}^n (a_i b_i)$ 等价于：

$$\sum_{i=1}^n (a_i b_i) = \mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a} = \mathbf{a}^\top \mathbf{b} = \mathbf{b}^\top \mathbf{a} \quad (38)$$

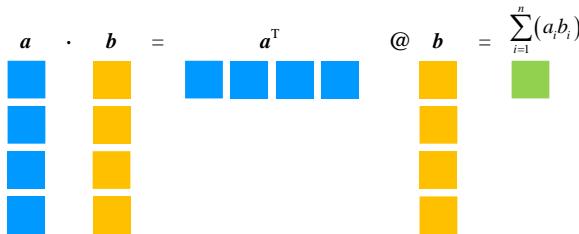


图 17. 从向量内积和矩阵乘法角度看 $\sum_{i=1}^n (a_i b_i)$

$\sum_{i=1}^n a_i$ 和 $\sum_{i=1}^n b_i$ 相乘展开得到：

$$\left(\sum_{i=1}^n a_i \right) \left(\sum_{i=1}^n b_i \right) = \sum_{i=1}^n a_i \sum_{i=1}^n b_i = (a_1 + a_2 + \cdots + a_n)(b_1 + b_2 + \cdots + b_n) \quad (39)$$

显然，(37) 和 (39) 不相同：

$$\sum_{i=1}^n (a_i b_i) \neq \left(\sum_{i=1}^n a_i \right) \left(\sum_{i=1}^n b_i \right) \quad (40)$$

二重求和

一些情况，我们需要用到二重求和记号 $\Sigma\Sigma$ ，比如：

$$\begin{aligned} \sum_{i=1}^3 \sum_{j=2}^4 a_i b_j &= \sum_{i=1}^3 a_i b_2 + a_i b_3 + a_i b_4 \\ &= (a_1 b_2 + a_1 b_3 + a_1 b_4) + (a_2 b_2 + a_2 b_3 + a_2 b_4) + (a_3 b_2 + a_3 b_3 + a_3 b_4) \end{aligned} \quad (41)$$

⚠ 注意，式中内层 Σ 索引为 j ，外层 Σ 索引为 i 。先对索引 j 求和，再对索引 i 求和。注意上式中，每个元素只有一个索引，相当于只有一个维度。

两个索引

实践中，经常遇到的情况是一项有两个、甚至更多索引，比如 a_{ij} 有两个索引 i 和 j 。

根据本节前文分析思路，举个例子，索引 i 的取值范围为 $1 \sim n$ ，索引 j 的取值范围为 $1 \sim m$ ，数组 a_{ij} 相当于有 i 和 j 两个维度。

这是否让大家想到了本书第 1 章讲过的矩阵，如图 18 所示， a_{ij} 相当于矩阵 A 的第 i 行、第 j 列元素。

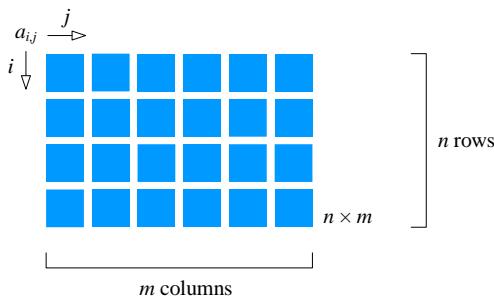


图 18. $n \times m$ 矩阵 A

本节后续内容就围绕图 19 热图给出的二维数组展开，这个数组有 8 行、12 列。

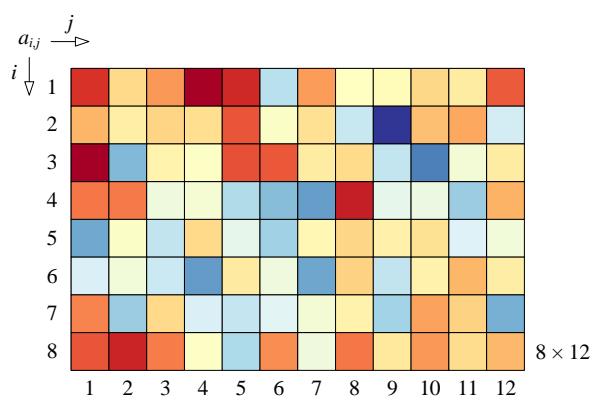


图 19. 8×12 数组

偏求和

下面先介绍单维求和，也就是沿着一个索引求和。我们给它取个名字，叫“偏求和”。

→ 这个“偏”字呼应本书第 16、18 章要介绍的“偏导数”、“偏微分”、“偏积分”等概念。

首先聊聊 $a_{i,j}$ 对索引 i 偏求和：

$$\sum_{i=1}^n a_{i,j} \Rightarrow \sum_{i=1}^n a_{i,1}, \sum_{i=1}^n a_{i,2}, \dots \sum_{i=1}^n a_{i,m}$$

Sum over i

(42)

我们发现，得到的不是一个求和，而是 m 个和。也就是说，图 18 中每一列数值求和，每一列都有一个“偏求和”结果。

白话说， $a_{i,j}$ 就是个“表格”， $\sum_i a_{i,j}$ 就是按列求和，每列一个和。

如图 20 所示， $\sum_{i=1}^n a_{i,7}$ 代表对数组第 7 列元素求和。

⚠ 注意，为了简化数学表达，我们也常用 $\sum_i a_{i,j}$ 代表对索引 i 的求和，求和上下限不再给出。

$\sum_i a_{i,j}$ 除以 n ，得到的就是每一列元素的平均数。

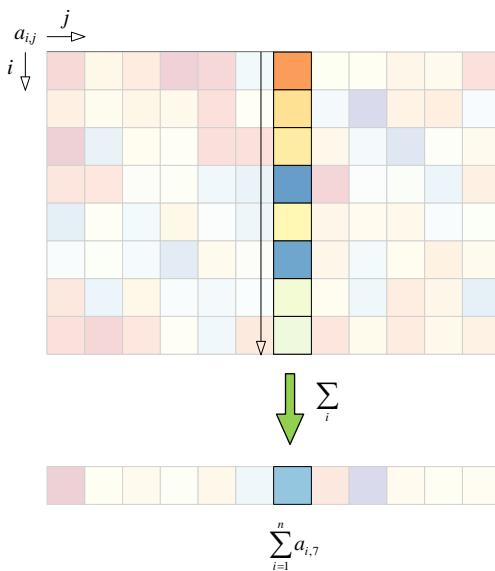
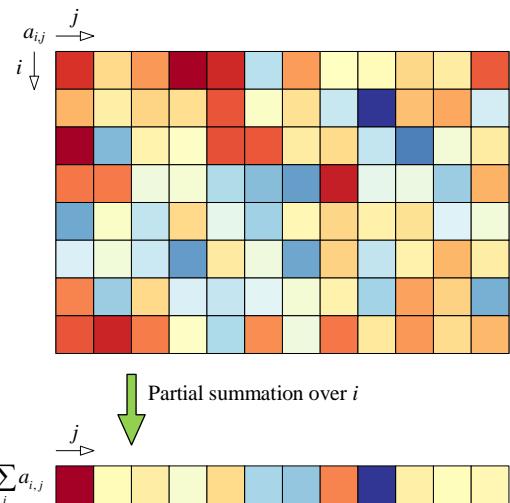


图 20. 将二维数组的第 7 列求和

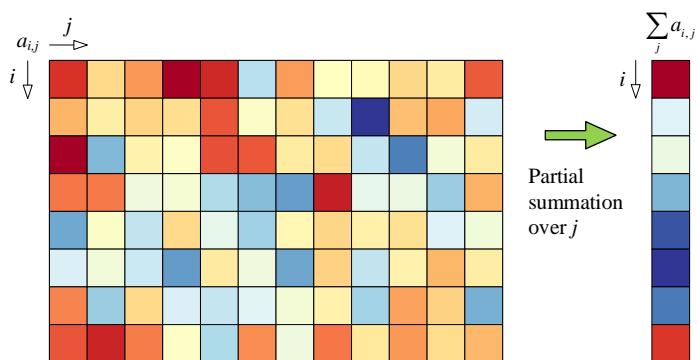
这相当于矩阵 $a_{i,j}$ 沿着索引 i 被“压扁”。如图 21 所示，原来数据有两个维度——索引 i 和 j ；而现在只剩一个维度——索引 j 。

图 21. 将二维数组沿着索引 i 代表的方向“压扁”

同理，如图 22 所示， $a_{i,j}$ 对索引 j 偏求和 $\sum_j a_{i,j}$ ，相当于沿着索引 j 方向将数组“压扁”； $\sum_j a_{i,j}$ 只剩 i 这一个维度。

白话说， $a_{i,j}$ 就是个“表格”， $\sum_j a_{i,j}$ 就是按行求和，每行一个和。

$\sum_j a_{i,j}$ 除以 m ，得到的就是每一行元素的平均数。

图 22. 将二维数组沿着索引 j 代表的方向“压扁”

多重求和

而 $\sum_i a_{i,j}$ 和 $\sum_j a_{i,j}$ 沿着各自剩余最后一个方向再次“压扁”，得到的就是 $a_{i,j}$ 所有元素的和。

这种情况，求和顺序不影响结果，即：

$$\sum_{i=1}^n \sum_{j=1}^m a_{i,j} = \sum_{j=1}^m \sum_{i=1}^n a_{i,j}$$

Sum over j Sum over i

(43)

上式也可以写作：

$$\sum_{j,i} a_{i,j} = \sum_{i,j} a_{i,j}$$
(44)

下标“ j, i ”表示先对 j 求和、再对 i 求和；下标“ i, j ”表示先对 i 求和、再对 j 求和。

图 23 所示为上述计算的过程分解。

⚠ 注意，只有数组是“方方正正”的结构时，上式才成立；否则，求和先后顺序会影响到结果。这一点和多重积分中积分先后顺序原理一致。

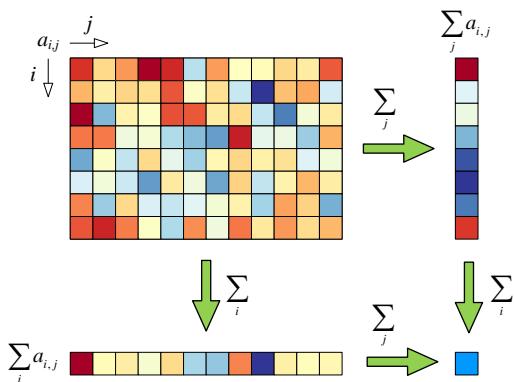


图 23. 求和顺序不影响结果

举个例子：

$$\begin{aligned} \sum_{i=1}^3 \sum_{j=2}^4 a_{i,j} &= \sum_{i=1}^3 a_{i,2} + a_{i,3} + a_{i,4} \\ &= (a_{1,2} + a_{1,3} + a_{1,4}) + (a_{2,2} + a_{2,3} + a_{2,4}) + (a_{3,2} + a_{3,3} + a_{3,4}) \end{aligned}$$
(45)

调换求和顺序，得到：

$$\begin{aligned} \sum_{j=2}^4 \sum_{i=1}^3 a_{i,j} &= \sum_{j=2}^4 a_{1,j} + a_{2,j} + a_{3,j} \\ &= (a_{1,2} + a_{2,2} + a_{3,2}) + (a_{1,3} + a_{2,3} + a_{3,3}) + (a_{1,4} + a_{2,4} + a_{3,4}) \end{aligned}$$
(46)

可以发现 (45) 和 (46) 两式相等。

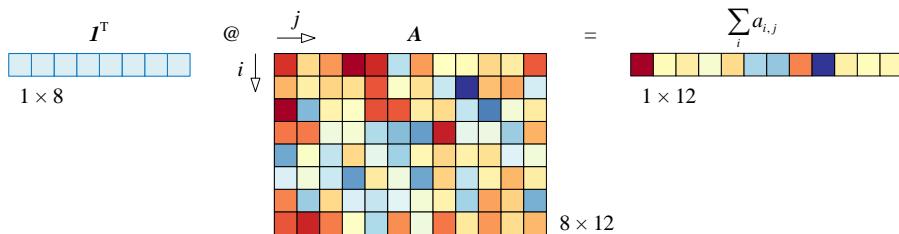
矩阵运算视角

前文介绍的“偏求和”和多重求和都可以通过矩阵运算得到结果。

如图 24 所示， $a_{i,j}$ 对索引 i 偏求和等价于如下矩阵运算：

$$\mathbf{I}^T \mathbf{A} = \sum_i a_{i,j} \quad (47)$$

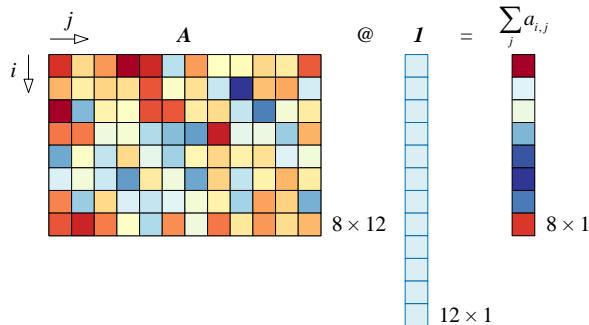
⚠ 注意，(47) 中全 1 列向量 \mathbf{I} 的形状为 8×1 ，转置得到 \mathbf{I}^T 的形状为 1×8 。

图 24. 计算矩阵 A 每列元素和

如图 24 所示， a_{ij} 对索引 j 偏求和等价于如下矩阵运算：

$$\mathbf{A}\mathbf{I} = \sum_j a_{i,j} \quad (48)$$

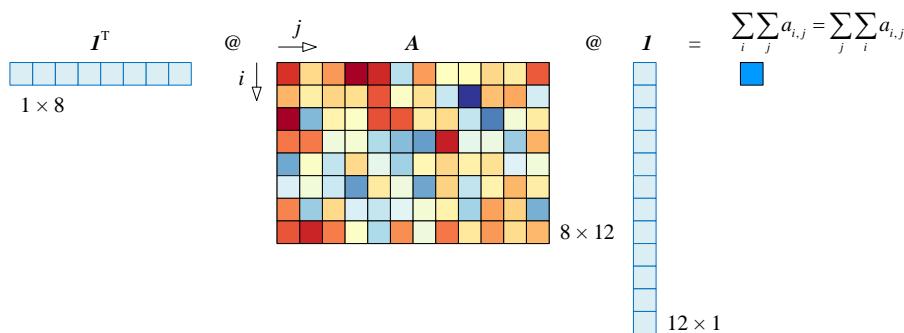
⚠ 注意，(48) 中全 1 列向量 \mathbf{I} 的形状为 12×1 。

图 25. 计算矩阵 A 每行元素和

如图 26 所示，求矩阵 A 所有元素之和对应的矩阵运算为：

$$\sum_i \sum_j a_{i,j} = \sum_j \sum_i a_{i,j} = \mathbf{I}^T \mathbf{A}\mathbf{I} \quad (49)$$

⚠ 再次强调，(49) 中两个全 \mathbf{I} 向量形状不同。

图 26. 计算二维矩阵 A 所有元素之和

两个以上索引

某一项可能有超过两个索引，比如 $a_{i,j,k}$ 有三个索引，数组 $\{a_{i,j,k}\}$ 相当于有三个维度。

图 27 所示为三维数组的多重求和运算，求和的顺序为“ j, k, i ”。

首先， $a_{i,j,k}$ 沿 j 索引求和，得到 $\sum_j a_{i,j,k}$ 。相当于一个立方体“压扁”为一个平面，三维降维到二维。

然后，再沿 k 索引求和，进一步将平面“压扁”得到一维数组。此时，数组只有一个索引 i 。

最后，沿着 i 再求和，得到一个标量 $\sum_{j,k,i} a_{i,j,k}$ 。

请大家自行绘制按照“ i, j, k ”这个顺序求和得到 $\sum_{i,j,k} a_{i,j,k}$ 过程示意图。

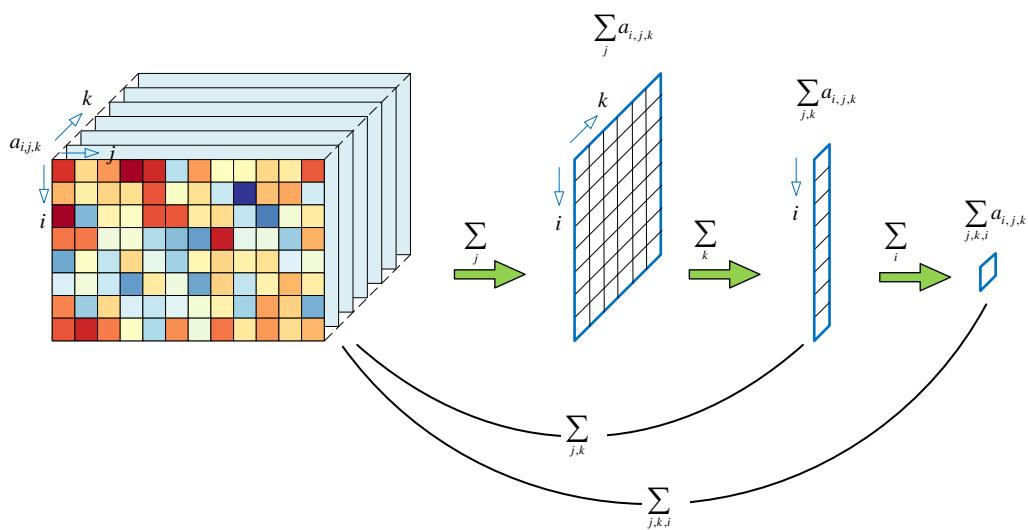


图 27. 三维数组的求和运算

对于多维数组，建议大家了解一下 Python 中 xarray 这个工具包。此外，Pandas 数据帧也是处理多维数组不错的工具。本系列丛书会介绍 xarray 和 Pandas 这两个工具。

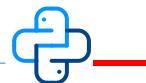
爱因斯坦曾经提出过以自己名字命名的求和法则，叫做**爱因斯坦求和约定** (Einstein summation convention)。Numpy 中 numpy.einsum() 函数的运算规则就是基于爱因斯坦求和约定。



爱因斯坦求和约定简化多维数组求和运算，本系列丛书《矩阵力量》将展开介绍。

表 2. 求和和求积的英文表达

数学表达	英文表达
$\sum_{i=1}^n a_i$	The sum of all the terms (small) a sub (small) i , where i takes the integers from one to (small) n . The sum from (small) i equals one to (small) n , (small) a sub i . The sum as (small) i runs from one to n of (small) a sub (small) i .
$\sum_{n=1}^5 2n$	The sum of 2 times n as n goes from 1 to 5. The summation of the expression $2n$ for integer values of n from 1 to 5.
$\prod_{i=1}^n a_i$	The multiplication of all the terms a sub i , where i takes the values from one to n . The product from i equals 1 to n of a sub i .
$\prod_{i=1}^{\infty} y_i$	The product from i equals one to infinity of y sub i .



Bk3_Ch14_04.py 绘制本节二维数组热图，并计算“偏求和”。请大家自行计算这个二维数组所有元素之和。

14.7 数列极限：微积分的一块基石

数列极限

数列 $\{a_n\}$ 极限存在的确切定义如下。

设 $\{a_n\}$ 为一数列，如果存在常数 C ，对于任意给定的正数 ε ，不管 ε 有多小，总存在正整数 N ，使得 $n > N$ 时，如下不等式均成立，

$$|a_n - C| < \varepsilon \quad (50)$$

那么就称常数 C 是数列 $\{a_n\}$ 的极限；也可以说，数列 $\{a_n\}$ 收敛于 C ：

$$\lim_{n \rightarrow \infty} a_n = C \quad (51)$$

其中， \lim 是英文 limit 的缩写， $n \rightarrow \infty$ 表达 n 趋向无穷。

如果，极限 C 不存在，则称数列 $\{a_n\}$ 极限不存在。

几个例子

给定如下等比数列：

$$a_n = \frac{1}{2^n} \quad (52)$$

当 n 趋向于无穷，数列值趋向于零：

$$\lim_{n \rightarrow \infty} a_n = \lim_{n \rightarrow \infty} \frac{1}{2^n} = 0 \quad (53)$$

对于如下数列，当 n 趋向无穷时，数列值在两个定值之间震荡；因此，不存在极限：

$$a_n = (-1)^n \quad (54)$$

对于如下数列，当 n 趋向无穷时，数列值急速增加至无穷，即发散；因此，也不存在极限：

$$a_n = 2^n \quad (55)$$

收敛的数列，可以自下而上收敛、自上而下收敛、振荡收敛。

图 28 给出了三个收敛数列的例子。图 28 (c) 对应的数列如下：

$$\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e \quad (56)$$

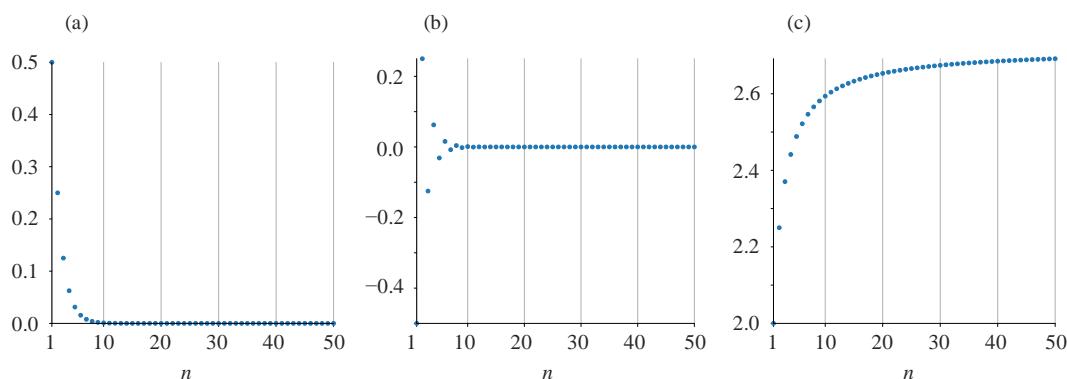


图 28. 收敛数列

数列和的极限

此外，数列之和也可以收敛。下式就是一个收敛的数列之和，数列之和随 n 变化趋势如图 29 (a) 所示：

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = \sum_{n=0}^{\infty} \frac{1}{2^n} = 2 \quad (57)$$

如图 29 (b) 所示，以下数列之和也是收敛于 1：

$$\frac{1}{1 \times 2} + \frac{1}{2 \times 3} + \frac{1}{3 \times 4} + \frac{1}{4 \times 5} + \frac{1}{5 \times 6} + \dots = \sum_{n=1}^{\infty} \frac{1}{n(n+1)} = 1 \quad (58)$$

如图 29 (c) 所示，自然对数底数 e 也可以用数列和的极限来近似：

$$\sum_{k=0}^{\infty} \frac{1}{k!} = 1 + \frac{1}{1} + \frac{1}{1 \times 2} + \frac{1}{1 \times 2 \times 3} + \frac{1}{1 \times 2 \times 3 \times 4} + \dots = e \quad (59)$$

但是 $1/n$ 这个数列之和并不收敛：

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots = \sum_{n=1}^{\infty} \frac{1}{n} \quad (60)$$

如果在以上数列中每一项增加正负号交替，这个数列之和收敛：

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \dots = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} = \ln(2) \quad (61)$$

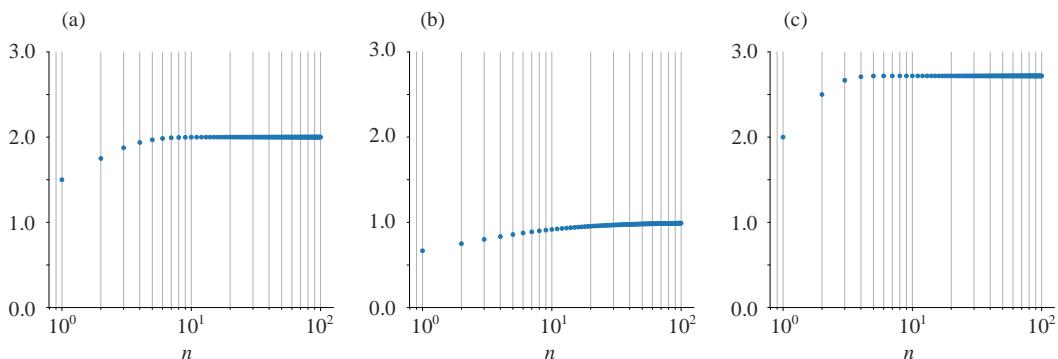
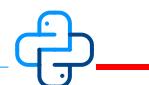


图 29. 数列之和收敛



Bk3_Ch14_05.py 绘制图 29。代码中利用 `sympy.limit_seq()` 函数计算极限值。

14.8 数列极限估算圆周率

本书第 3 章介绍，古代数学家通过割圆术不断提高圆周率的估算精度。随着数学方法的发展，很多数学家发现可以用数列和来逼近圆周率。这是圆周率估算的一次颠覆性进步。

比如莱布尼兹发现，如下数列之和逼近 $\pi/4$ ：

$$\frac{\pi}{4} \approx 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^{n+1}}{2n-1} \quad (62)$$

即

$$\pi = 4 \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \quad (63)$$

图 30 所示为上式随着 k 不断增加逼近圆周率值。

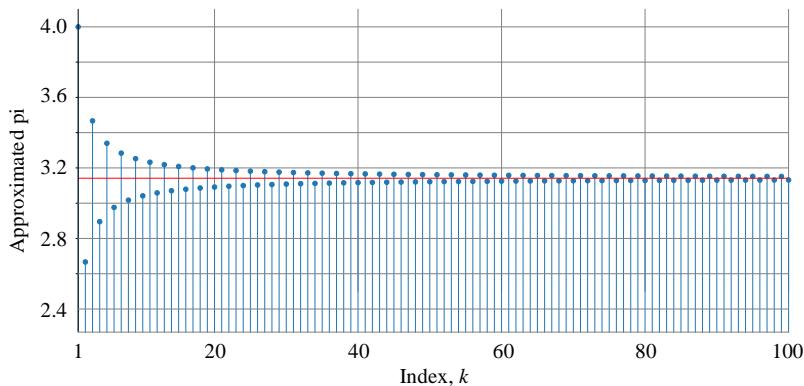


图 30. 数列之和逼近圆周率



Bk3_Ch14_06.py 绘制图 30。



本章介绍的大西格玛求和、极限这两个数学工具是微积分的基础。

大家在学习大西格玛求和时，请务必从几何、数据、维度、矩阵运算这几个视角分析求和运算；不然，复杂多层求和运算会让大家晕头转向。

此外，有了数列和极限这两个数学概念，我们在圆周率估算方法上又进一步。

15 导数

Derivative
函数切线斜率，即变化率



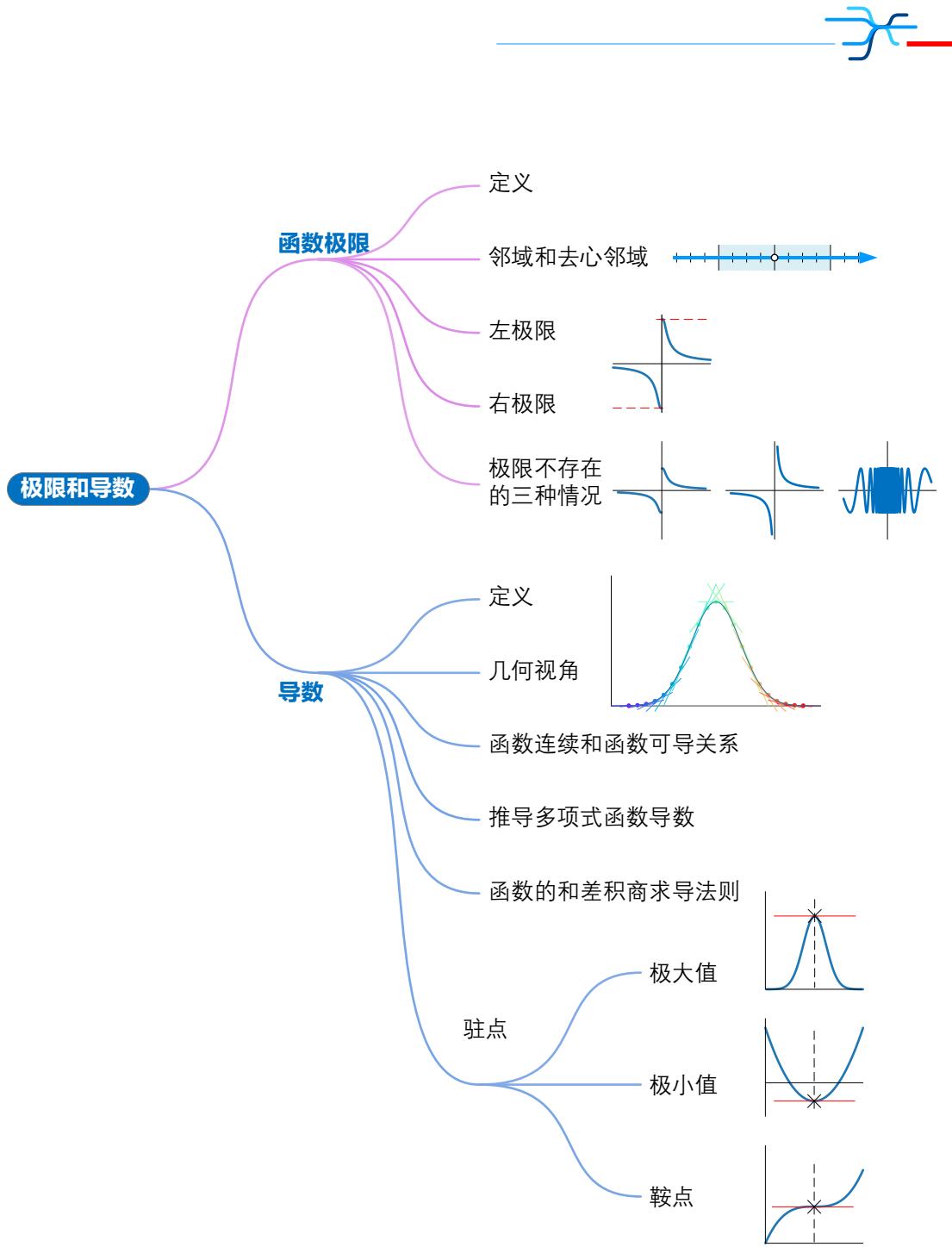
微积分是现代数学的第一个成就，它的重要性怎么评价都不为过。我认为它比其他任何东西都更明确地定义了现代数学的起源。而作为其逻辑发展的数学分析系统，仍然是精确思维的最大技术进步。

The calculus was the first achievement of modern mathematics and it is difficult to overestimate its importance. I think it defines more unequivocally than anything else the inception of modern mathematics; and the system of mathematical analysis, which is its logical development, still constitutes the greatest technical advance in exact thinking.

—— 约翰·冯·诺伊曼 (John von Neumann) | 美国籍数学家 | 1903 ~ 1957



- ◀ sympy.abc import x 定义符号变量 x
- ◀ sympy.diff() 求解符号函数导数和偏导解析式
- ◀ 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.limit() 求解极限
- ◀ sympy.plot_implicit() 绘制隐函数方程
- ◀ sympy.series() 求解泰勒展开级数符号式
- ◀ sympy.symbols() 定义符号变量



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

15.1 牛顿小传

“如果说我比别人看得更远，那是因为我站在巨人们的肩上。”

1642年年底，**艾萨克·牛顿**(Sir Isaac Newton)呱呱坠地，同年年初伽利略驾鹤西征。牛顿从伽利略手中接过了智慧火炬。这可能完全是巧合，但又何尝不是某种命中注定。在伽利略等科学先驱者开垦的沃土上，即便没有培育出牛顿，也会注定会造就马顿、羊顿、米顿…



艾萨克·牛顿(Sir Isaac Newton)
英国物理学家、数学家 | 1643年 ~ 1727年
提出万有引力定律、牛顿运动定律，与莱布尼茨共同发明微积分

年轻的牛顿坐在果园里，思考物理学。苹果熟了，从树上落下，砸到了牛顿的脑门。牛顿发出了一个惊世疑问，苹果为什么会下落？

是的，苹果为什么会下落，而不是飞向更遥远的天际？对这些问题的系统思考让牛顿提出万有引力定律。

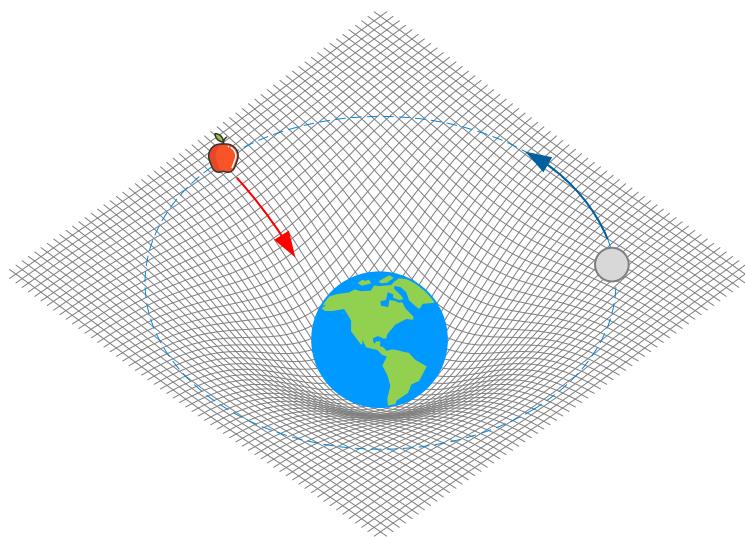


图 1. 地球引力场作用下的月球和苹果

牛顿的成就不止于此。他提出三大运动定律，并出版《自然哲学的数学原理》(*Mathematical Principles of Natural Philosophy*)，他利用三棱镜发现七色光谱，发明反射望远镜，并提出光的微

粒说，他和莱布尼茨分别独立发明微积分等等。任何人有其中任意一个贡献，就可以留名青史；然而，牛顿一个人完成上述科学进步。



自然和自然规律隐藏在黑暗之中。

上帝说：交给牛顿吧，

于是一切豁然开朗。

Nature and Nature's laws lay hid in night:

God said, Let Newton be! and all was light.

—— 亚历山大·蒲柏 (Alexander Pope) | 英国诗人 | 1688 ~ 1744

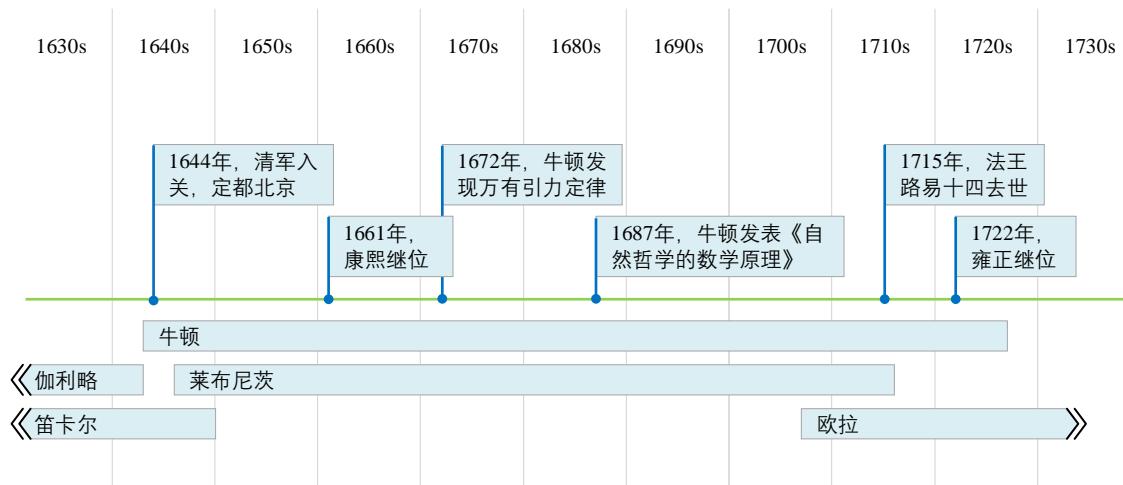


图 2. 牛顿时代时间轴

15.2 极限：研究微积分的重要数学工具

微积分 (calculus) 是研究实数域上函数的微分与积分等性质的学科，而极限是微积分最重要数学工具。**连续** (continuity)、**导数** (derivative) 和**积分** (integral) 这些概念都是通过极限来定义。

上一章简单介绍了数列极限、数列和的极限。本节主要介绍函数极限。

函数极限

首先聊一下函数极限的定义。

设函数 $f(x)$ 在点 a 的某一个去心邻域内有定义，如果存在常数 C ，对于任意给定正数 ε ，不管它多小，总存在正数 δ ，使得 x 满足如下不等式时，

$$0 < |x - a| < \delta \quad (1)$$

对应函数值 $f(x)$ 都满足，

$$|f(x) - C| < \varepsilon \quad (2)$$

常数 C 就是函数 $f(x)$ 当 $x \rightarrow a$ 时的极限，记做：

$$\lim_{x \rightarrow a} f(x) = C \quad (3)$$

举个例子

给定如下函数：

$$f(x) = \left(1 + \frac{1}{x}\right)^x \quad (4)$$

如图 3 所示，当 x 趋向正无穷，函数极限为 e ：

$$\lim_{x \rightarrow +\infty} f(x) = e \quad (5)$$

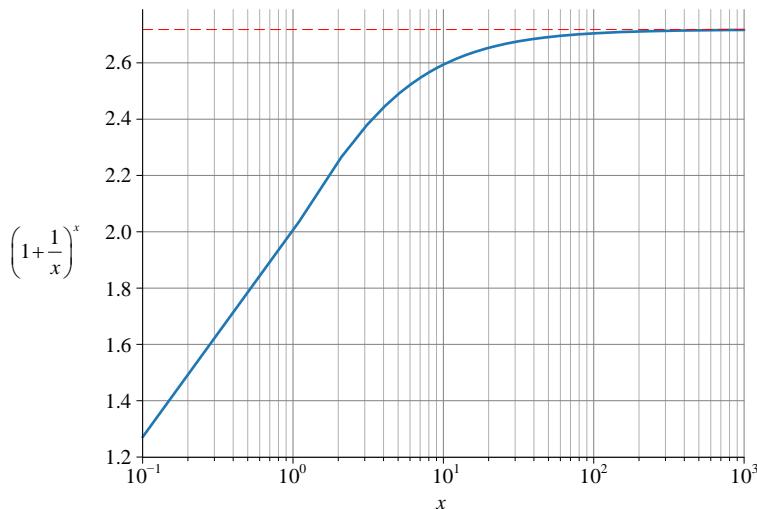


图 3. 当 x 趋向正无穷，函数 $f(x)$ 极限值

邻域

解释一下邻域这个概念。邻域 (neighbourhood) 实际上就是一个特殊的开区间。如图 4 所示，点 a 的 h ($h > 0$) 邻域满足 $a - h < x < a + h$ 。

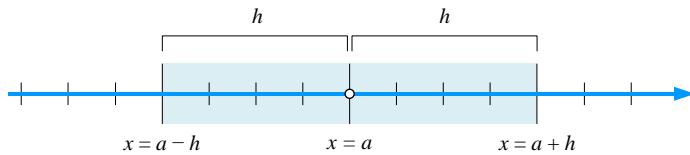
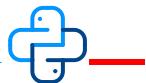


图 4. 邻域

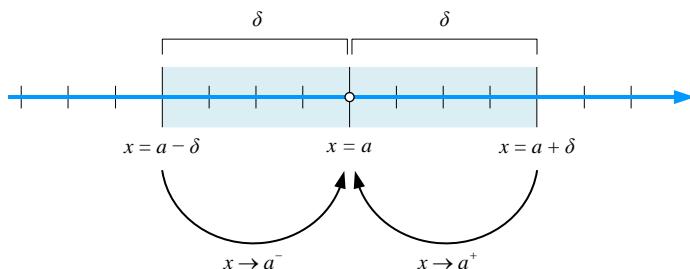
a 为邻域的中心， h 为邻域的半径。而去心邻域 (deleted neighborhood 或 punctured neighborhood) 指的是，在 a 的邻域中去掉 a 的集合。



Bk3_Ch15_01.py 计算极限并绘制图 3。

15.3 左极限、右极限

请注意 (1) 的绝对值符号。如图 5 所示，这代表着 x 从右 ($x > a$)、左 ($x < a$) 两侧趋向 a 。下面，我们聊一聊分别从右侧和左侧趋向于 a 有怎样的区别和联系。

图 5. x 分别从左右两侧趋向 a

右极限

将 (1) 绝对值符号去掉取正得到，

$$0 < x - a < \delta \quad (6)$$

称之为 x 从右侧趋向 a ，记做 $x \rightarrow a^+$ 。

随之，将(3)中极限条件改为 $x \rightarrow a^+$ ， C 叫做函数 $f(x)$ 的**右极限**(right-hand limit或right limit)，记做：

$$\lim_{x \rightarrow a^+} f(x) = C \quad (7)$$

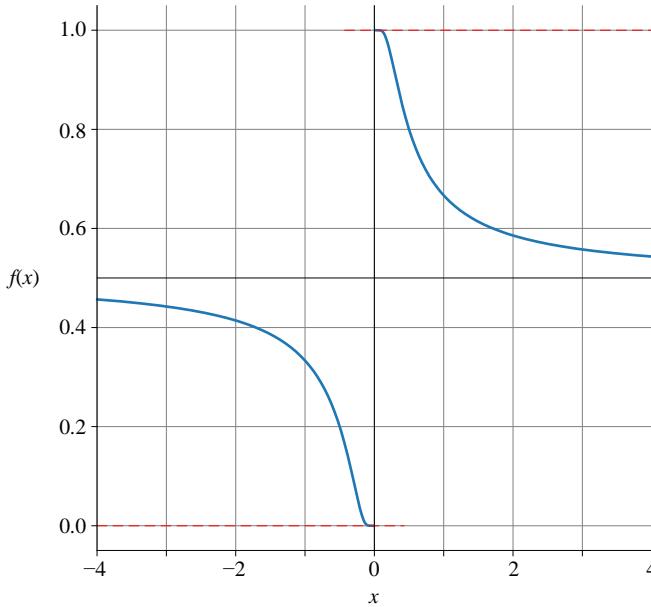


图 6. 函数 $f(x)$ 左右极限不同

左极限

相反，如果将(1)绝对值符号去掉并取负，

$$-\delta < x - a < 0 \quad (8)$$

称之为 x 从左侧趋向 a ，记做 $x \rightarrow a^-$ 。

将(3)中极限条件改为 $x \rightarrow a^-$ ， C 叫做函数 $f(x)$ 的**左极限**(left-hand limit或left limit)，记做：

$$\lim_{x \rightarrow a^-} f(x) = C \quad (9)$$

当(7)和(9)都成立时，(3)才成立。也就是，当 $x \rightarrow a$ 时函数 $f(x)$ 极限存在的充分必要条件是，左右极限均存在且相等。

极限不存在

⚠请大家格外注意，即便左右极限均存在，如果两者不相等，则极限不存在。

如图6所示，函数在 $x=0$ 的右极限为1：

$$\lim_{x \rightarrow 0^+} \frac{1}{1 + 2^{-1/x}} = 1 \quad (10)$$

而函数在 $x = 0$ 的左极限为 0：

$$\lim_{x \rightarrow 0^-} \frac{1}{1 + 2^{-1/x}} = 0 \quad (11)$$

显然函数在 $x = 0$ 处不存在极限。此外， $f(x)$ 在 $x = 0$ 处没有定义。

$\lim_{x \rightarrow a} f(x)$ 不存在可能有三种情况：(a) $f(x)$ 在 $x = a$ 处左右极限不一致；(b) $f(x)$ 在 $x = a$ 处趋向无穷；(c) $f(x)$ 在趋向 $x = a$ 时在两个定值之间震荡。这三种情况分别对应图 7 三幅子图。

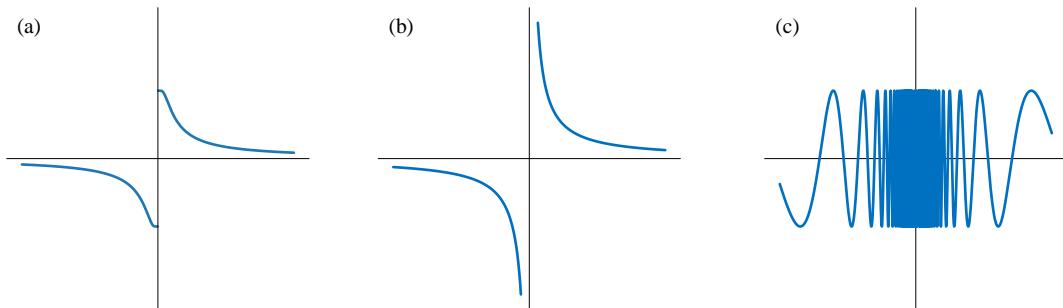


图 7. 极限不存在的三种情况

表 1. 极限的英文表达

数学表达	英文表达
$\lim_{\Delta x \rightarrow 0} f(x) = b$	As delta x approaches 0, the limit for f of x equals b .
$\Delta x \rightarrow 0$	Delta x approaches zero.
$\Delta x \rightarrow 0^+$	Delta x goes to zero from the right. Delta x approaches to zero from the right.
$\Delta x \rightarrow 0^-$	Delta x goes to zero from the left. Delta x approaches to zero from the left.
$\lim_{\Delta x \rightarrow 0}$	The limit as delta x approaches zero. The limit as delta x tends to zero.
$\lim_{x \rightarrow a^+}$	The limit as x approaches a from the right. The limit as x approaches a from the above.
$\lim_{x \rightarrow a^-}$	The limit as x approaches a from the left. The limit as x approaches a from the below.
$\lim_{x \rightarrow c} f(x) = L$	The limit of $f(x)$ as x approaches c is L .
$\lim_{n \rightarrow \infty} a_n = L$	The limit of a sub n as n approaches infinity equals L .
$\lim_{x \rightarrow -\infty} f(x) = L_1$	the limit of f of x as x approaches negative infinity is capital L sub one.
$\lim_{x \rightarrow +\infty} f(x) = L_2$	the limit of f of x as x approaches positive infinity is capital L sub two.



Bk3_Ch15_02.py 求函数左右极限，并绘制图 6。

15.4 几何视角看导数：切线斜率

导数 (derivative) 描述函数在某一点处的变化率。几何角度来看，导数可以视作函数曲线切线斜率。

切线斜率

举个中学物理中的例子，加速度 a 是速度 v 的变化率，速度 v 是距离 s 的变化率。

如图 8 所示，匀速直线运动中，距离函数 $s(t)$ 对于时间 t 是一个一次函数。从图像角度来看， $s(t)$ 是一条斜线。

$s(t)$ 图像的切线斜率不随时间变化，也就是说匀速直线运动的速度函数 $v(t)$ 的图像为常数函数。

而 $v(t)$ 的切线斜率为 0，说明加速度 $a(t)$ 图像为取值为 0 的常数函数。

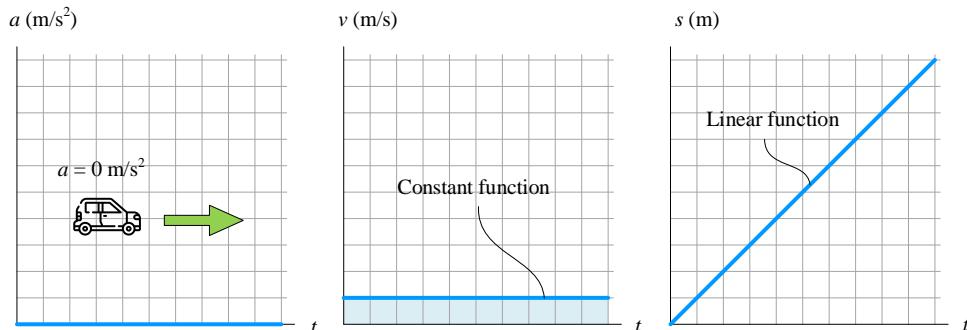


图 8. 匀速直线运动：加速度、速度、距离图像

再看个例子。如图 9 所示，对于匀加速直线运动，距离函数 $s(t)$ 对于时间 t 是一个二次函数。从图像上看， $s(t)$ 在不同时间 t 位置，切线斜率不同。随着 t 增大，切线斜率不断增大，说明运动速度随 t 增大而增大。

完成本章学习后，大家会知道二次函数的导数是一次函数，也就是说速度函数 $v(t)$ 的图像为一次函数。

显然，速度函数 $v(t)$ 的切线斜率不随时间变化。因此， $a(t)$ 图像为常数函数，即加速度为定值。

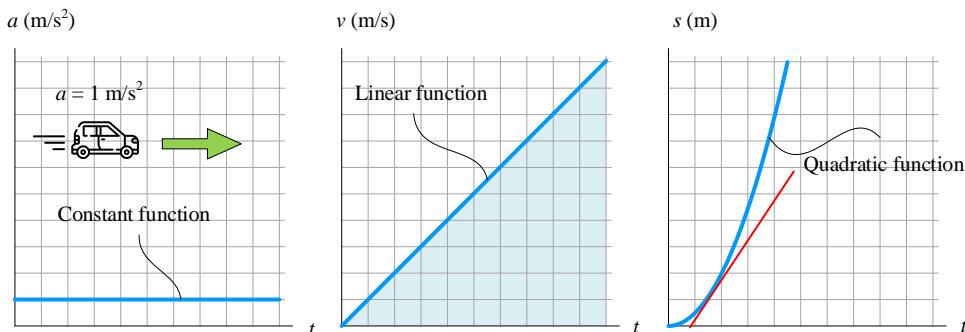


图 9. 匀加速直线运动：加速度、速度、距离

换个角度来看， $a(t)$ 和横轴在一定时间范围，比如在 $[t_1, t_2]$ 围成的面积就是速度变化 $v_2 - v_1$ 。同理， $v(t)$ 和横轴在 $[t_1, t_2]$ 围成的面积就是距离变化 $s_2 - s_1$ 。完成这个运算的数学工具就是第 18 章要介绍的定积分。简单来说，积分就是求面积、体积。

再从数值单位变化角度，如图 9 三幅子图纵轴所示，加速度的单位为 m/s^2 ，速度的单位为 m/s ，距离的单位是 m 。距离 (m) 随时间 (s) 的变化，单位上就是 m/s ；速度 (m/s) 随时间 (s) 的变化，单位上就是 m/s/s ，即 m/s^2 。

反向来看，加速度 (m/s^2) 到速度 (m/s) 就是求面积的过程。加速度纵轴的单位为 m/s^2 ，而横轴的单位为 s ，因此结果的单位为 $\text{m/s}^2 \times \text{s}$ ，即 m/s 。同理，速度纵轴的单位为 m/s ，横轴单位为 s ，因此结果的单位为 $\text{m/s} \times \text{s}$ ，即 m 。

⚠️ 再次提醒大家，不管是加减乘除，还是微分积分，注意数值单位。

函数导数定义

下面看一下函数导数的确切定义。

对于函数 $y = f(x)$ 自变量 x 在 a 点处一个微小增量 Δx ，会导致函数值增量 $\Delta y = f(a + \Delta x) - f(a)$ 。

当 Δx 趋向于 0 时，函数值增量 Δy 和自变量增量 Δx 比值的极限存在，则称 $y = f(x)$ 在 a 处可导 (function f of x is differentiable at a)。这个极限值便是函数 $f(x)$ 在 a 点处一阶导数值：

$$f'(a) = f'(x)|_{x=a} = \frac{df(x)}{dx}|_{x=a} = \lim_{\Delta x \rightarrow 0} \frac{f(a + \Delta x) - f(a)}{\Delta x} \quad (12)$$

如图 10 所示，从几何角度看，随着 Δx 不断减小，割线不断接近切线。

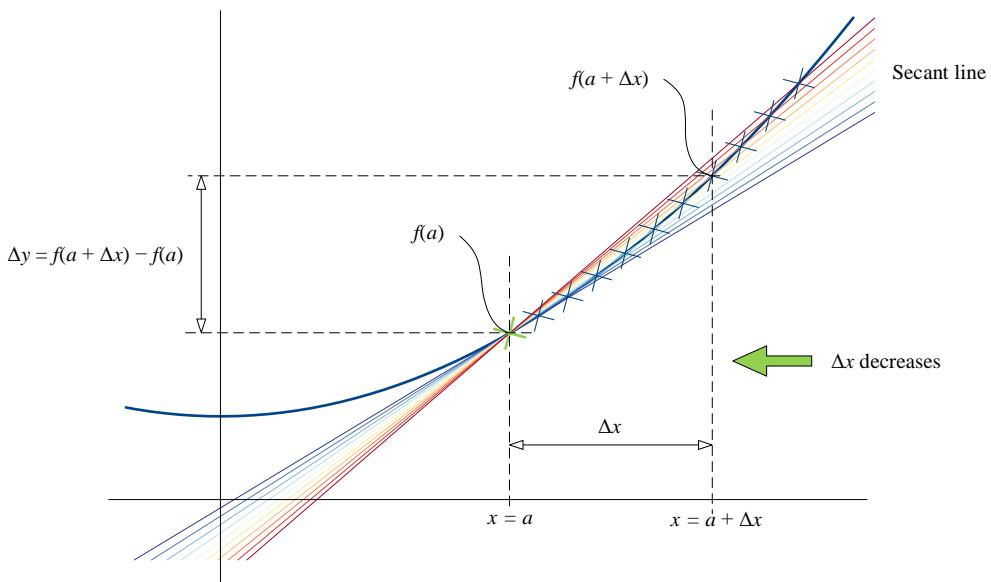
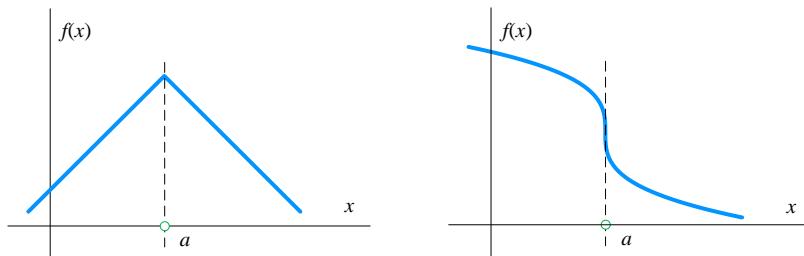


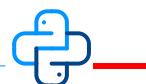
图 10. 导数就是变化率

Δ 和 d 都是“差”(difference) 的含义。但是， Δ 代表近似值，比如 $\Delta x \rightarrow 0$ ；而 d 是精确值，比如 dx 。白话说， dx 是 Δx 趋于 0 的精确值。

如果函数 $y = f(x)$ 在 $x = a$ 可导 (differentiable) 则函数在该点处连续 (continuous)；但是，函数在某一点处连续并不意味着函数可导，如图 11 所示两种情况。

图 11. 函数在 $x = a$ 不存在导数的两种情况

⚠ 再次注意，本书用 x_1 、 x_2 、 x_3 等等表达变量，而不是变量 x 取值。如果有必要对自变量取值进行编号，本书会使用上标记法 $x^{(1)}$ 、 $x^{(2)}$ 、 $x^{(3)}$ 等等。



Bk3_Ch15_03.py 绘制图 10。



在 Bk3_Ch15_03.py 基础上，我们做了一个 App 用来演示在函数曲线不同点如何用割线近似函数切线斜率。请参考 Streamlit_Bk3_Ch15_03.py。

15.5 导数也是函数

导数也常被称作导数函数或导函数，因为导数也是函数。

图 12 所示函数曲线在不同点处切线斜率随着自变量 x 变化。再次强调，函数 $f(x)$ 对自变量 x 的一阶导数 $f'(x)$ 也是一个函数，它的自变量也是 x 。 $f'(x)$ 可以读作 (f prime of x)。

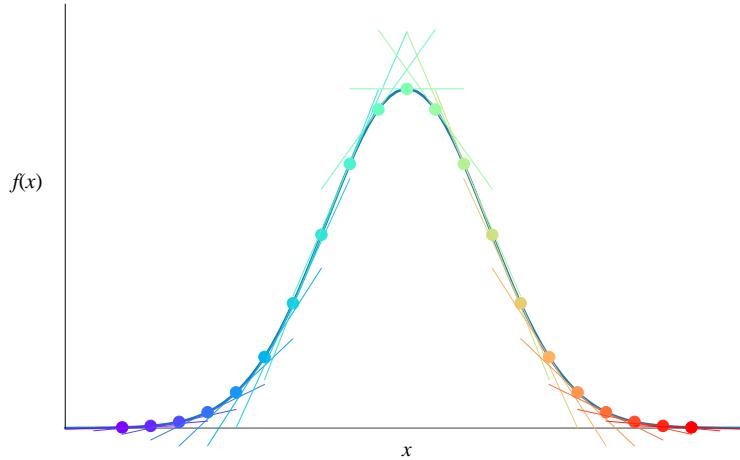


图 12. 函数不同点处切线斜率随着自变量 x 变化

一阶导数

给定二次函数， $f(x) = x^2$ 。下面利用 (12) 推导它的一阶导数：

$$\begin{aligned}
 f'(x) &= \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{(x + \Delta x)^2 - x^2}{\Delta x} \\
 &= \lim_{\Delta x \rightarrow 0} \frac{2x\Delta x + \Delta x^2}{\Delta x} \\
 &= \lim_{\substack{\Delta x \rightarrow 0 \\ \rightarrow 0}} 2x + \Delta x = 2x
 \end{aligned} \tag{13}$$

几何角度来看， $f(x) = x^2$ 相当于边长为 x 的正方形面积。图 13 所示为当 x 增加到 $x + \Delta x$ 时，函数值变化对应正方形面积变化。 x 到 $x + \Delta x$ ，正方形面积增加 $2x\Delta x + \Delta x^2$ 。

根据导数定义，函数导数为比值 $(2x\Delta x + \Delta x^2)/\Delta x = 2x + \Delta x$ ；当 $\Delta x \rightarrow 0$ 时，可以消去 Δx 一项。

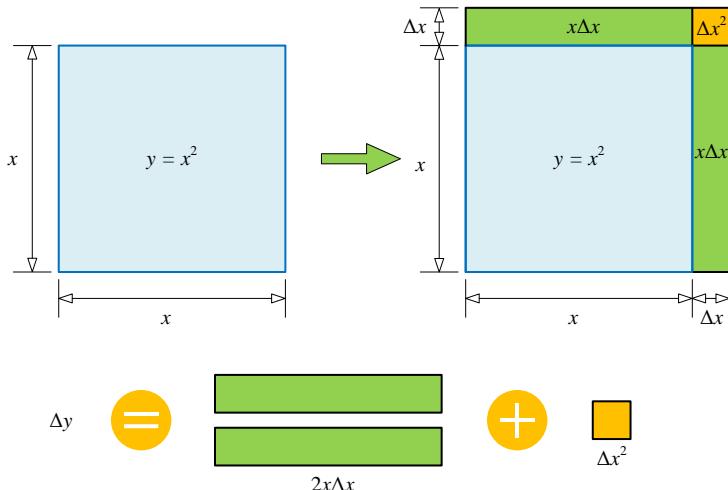


图 13. 几何角度推导 $f(x) = x^2$ 的一阶导数

类似地，推导 $f(x) = x^n$ 的导数， n 为大于 1 的正整数：

$$\begin{aligned}
 f'(x) &= \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \rightarrow 0} \frac{(x + \Delta x)^n - x^n}{\Delta x} \\
 &= \lim_{\Delta x \rightarrow 0} \frac{x^n + nx^{n-1}\Delta x + \frac{n(n-1)}{2}x^{n-2}\Delta x^2 + \dots + \Delta x^n - x^n}{\Delta x} \\
 &= \lim_{\Delta x \rightarrow 0} nx^{n-1} + \underbrace{\frac{n(n-1)}{2}x^{n-2}\Delta x + \dots + \Delta x^{n-1}}_{\rightarrow 0} = nx^{n-1}
 \end{aligned} \tag{14}$$

举个例子

图 14 (a) 所示函数如下：

$$f(x) = x^2 + 2 \tag{15}$$

根据前文推导，它的一阶导数解析式如下：

$$f'(x) = 2x \tag{16}$$

如图 14 (b) 所示，(15) 这个二次函数的一阶导数图像为一条斜线。

$x < 0$ 时， x 增大 $f(x)$ 减小，此时函数导数为负。 $x > 0$ ， x 增大 $f(x)$ 增大，函数导数为正。值得注意的是 $x = 0$ ， $f(x)$ 取得 **最小值** (minimum)，此处函数 $f(x)$ 导数值为 0。

而对(16)再求一阶导数得到的结果是常数函数，具体如所示图14(c)。

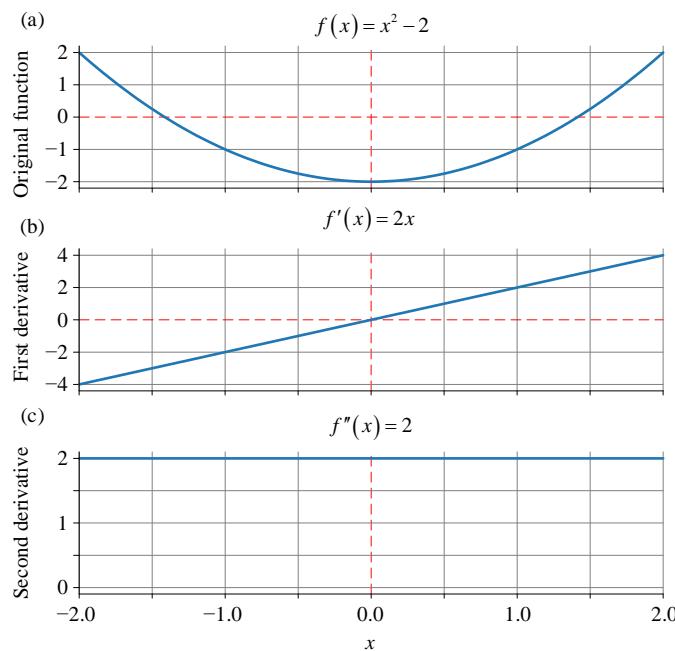


图 14. 二次函数、一阶导数、二阶导数

表2总结了常用函数导数及图像，请大家自行绘制这些图像。

表 2. 常用函数导数及图像

函数	函数图像举例	一阶导数	一阶导数图像举例
常数函数 $f(x) = C$		$f'(x) = 0$	
一次函数 $f(x) = ax$		$f'(x) = a$	

二次函数 $f(x) = ax^2 + bx + c$		$f'(x) = 2ax + b$	
幂函数 $f(x) = x^p$		$f'(x) = px^{p-1}$	
正弦函数 $f(x) = \sin x$		$f'(x) = \cos x$	
余弦函数 $f(x) = \cos x$		$f'(x) = -\sin x$	
指数函数 $f(x) = b^x$ ($b > 0, b \neq 1$)		$f'(x) = \ln b \cdot b^x$	
自然指数函数 $f(x) = e^x = \exp(x)$		$f'(x) = e^x = \exp(x)$	

对数函数 $f(x) = \log_b x$ ($x > 0, b > 0, b \neq 1$)	$f(x) = \log_{10}(x)$	$f'(x) = \frac{1}{\ln b \cdot x}$	$f'(x) = 1/(\ln 10 \cdot x)$
自然对数函数 $f(x) = \ln x$ ($x > 0$)	$f(x) = \ln(x)$	$f'(x) = \frac{1}{x}$	$f(x) = 1/x$
高斯函数 $f(x) = \exp(-\gamma x^2)$	$f(x) = \exp(-x^2)$	$f'(x) = -2\gamma x \exp(-\gamma x^2)$	$f(x) = -2x \exp(-x^2)$

二阶导数

(15) 这个二次函数的二阶导数是其一阶导数的一阶导数,

$$f''(x) = 2 \quad (17)$$

如图 14 (c) 所示, (15) 这个二次函数的二阶导数图像为一条水平线, 即常数函数。

图 15 所示为, 高斯函数以及其一阶导数和二阶导数函数图像。



容易发现, 函数 $f(x)$ 在 $x = 0$ 处取得最大值, 对应的一阶导数为 0, 二阶导数为负。这一点对于理解一元函数的极值非常重要, 本书第 19 章将深入介绍。

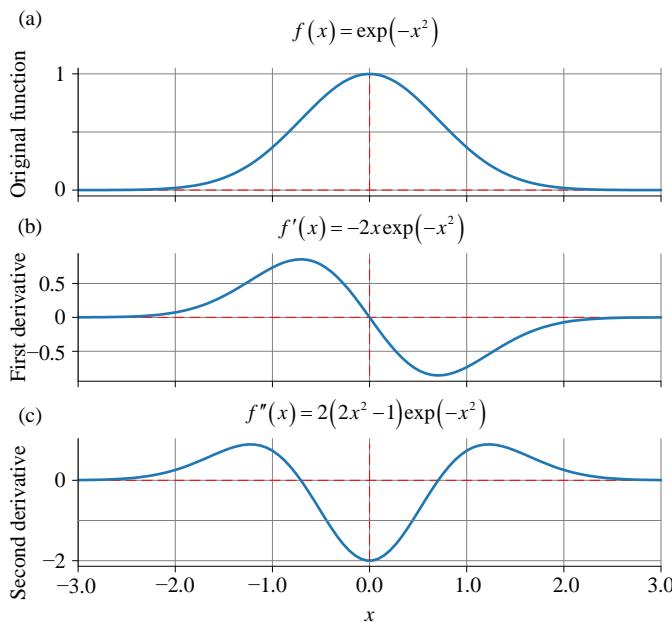


图 15. 高斯函数、一阶导数、二阶导数

图 16 所示为，三次函数图像，以及其一阶导数和二阶导数函数图像。容易发现， $x = 0$ 处函数一阶导数为 0；但是， $x = 0$ 既不对应函数最大值，也不是最小值。

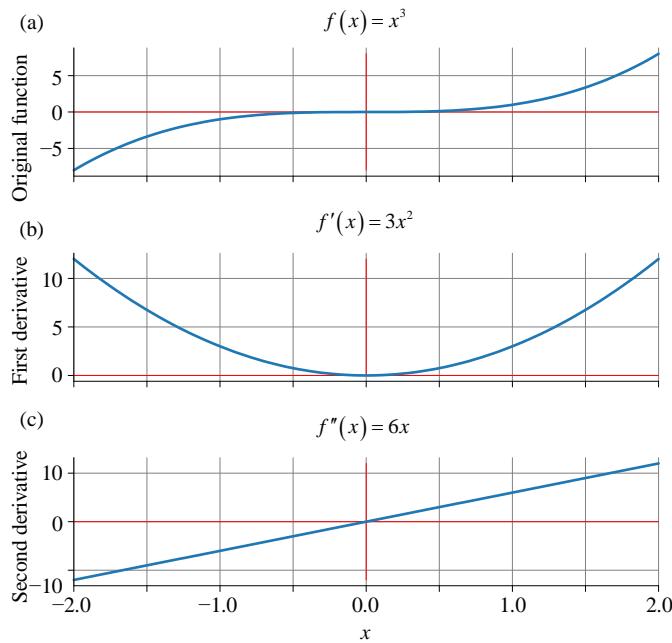


图 16. 三次函数、一阶导数、二阶导数

驻点

有了以上分析，我们可以聊一聊驻点这个概念。

对于一元函数 $f(x)$ ，**驻点** (stationary point) 是函数一阶导数为 0 的点。从图像上来看，一元函数 $f(x)$ 在驻点处的切线平行于 x 轴。

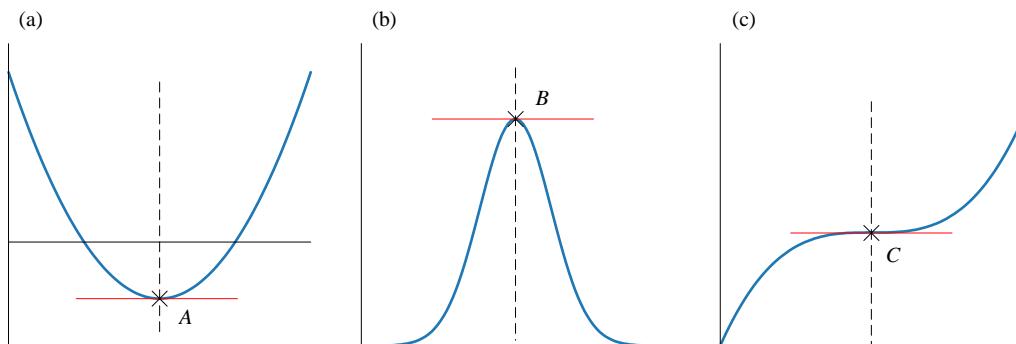


图 17. 驻点可能是极小值、极大值或鞍点

如图 17 所示，驻点可能是一元函数的极小值、极大值、鞍点。

⚠ 注意，这里我们没有用最大值和最小值，这是因为函数可能存在不止一个“山峰”或“山谷”。



本书第 19 章将在讲解优化问题时深入探讨这些概念。

表 3. 常用导数法则

和	$(f(x) + g(x))' = f'(x) + g'(x)$
差	$(f(x) - g(x))' = f'(x) - g'(x)$
积	$(f(x) \cdot g(x))' = f'(x) \cdot g(x) + f(x) \cdot g'(x)$
商	$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{g^2(x)}$
倒数	$\left(\frac{1}{f(x)}\right)' = \frac{-f'(x)}{f^2(x)}$

表 4. 导数相关的英文表达

数学表达	英文表达
dy	differential of y

$\frac{dy}{dx}$	the derivative of y with respect to x the derivative with respect to x of y $d y$ by $d x$ $d y$ over $d x$
$d f(x)$	The derivative of f of x
$\frac{d f(x)}{dx}$	The derivative of f of x with respect to x
$\frac{d f(a)}{dx}$	the derivative of f with respect to x at a $d y$ by $d x$ at a $d y$ over $d x$ at a
$\frac{dx^3}{dx} = 3x^2$	The derivative of x cubed with respect to x equals three x squared.
$\frac{d^2 y}{dx^2}$	d two y by $d x$ squared the second derivative of y with respect to x
$\frac{d^2 x^3}{dx^2} = 6x$	The second derivative of x cubed with respect to x equals to six x .
$\frac{d^n y}{dx^n}$	n th derivative of y with respect to x
$f'(x)$	f dash x f prime of x the derivative of f of x with respect to x the first-order derivative of f with respect to x
$f'(a)$	f prime of a
$f''(x)$	f double-dash x f double prime of x the second derivative of f with respect to x the second-order derivative of f with respect to x
$f'''(x)$	f triple prime of x f triple-dash x f treble-dash x the third derivative of f with respect to x the third-order derivative of f with respect to x
$f^{(4)}(x)$	the fourth derivative of f with respect to x the fourth-order derivative of f with respect to x
$f^{(n)}(x)$	the n th derivative of f with respect to x the n th-order derivative of f with respect to x f to the n th prime of x
$f'(g(x))$	f prime of g of x f prime at g of x
$f'(g(x))g'(x)$	the product of f prime of g of x and g prime of x
$(f(x)g(x))'$	the quantity of f of x times g of x , that quantity prime
$f'(x)g(x) + f(x)g'(x)$	f prime of x times g of x , that product plus f of x times g prime of x
$\left(\frac{f(x)}{g(x)}\right)'$	the quantity f of x over g of x , that quantity prime
$\frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}$	the fraction, the numerator is f prime of x times g of x , that product minus f of x times g prime of x , the denominator is g squared of x



Bk3_Ch15_04.py 绘制图 14；请读者修改代码绘制本节其他图像。本节代码采用 `sympy.abc import x` 定义符号变量，然后利用 `sympy.diff()` 计算一阶导数函数符号式；利用 `sympy.lambdify()` 将符号式转换成函数。



每个天才的诞生都需要时代、社会、思想的土壤。牛顿之所以成为牛顿，是一代代巨匠层层累土的结果。

牛顿开创经典牛顿力学体系，以此为基础的牛顿机械论自然观让当时人类思想界的面貌天翻地覆，它是人类文明的划时代的里程碑。必须认识到牛顿的力学体系是基于哥白尼、开普勒、伽利略等人知识之上的继承和发展。在牛顿所处的时代，哥白尼的日心说已经深入人心，开普勒提出行星运动三定律，伽利略发现惯性定律和自由落体定律。此外，牛顿之所以能发明微积分，离不开笛卡尔创立的解析几何。

人类知识体系是由一代代学者不断继承发展而丰富壮大的。每一个发现、每一条定理，都是知识体系重要的一环，它们既深受前辈学者影响，又启迪后世学者。

16

Partial Derivative

偏导数

只对多元函数一个变量求导，其他变量保持定值



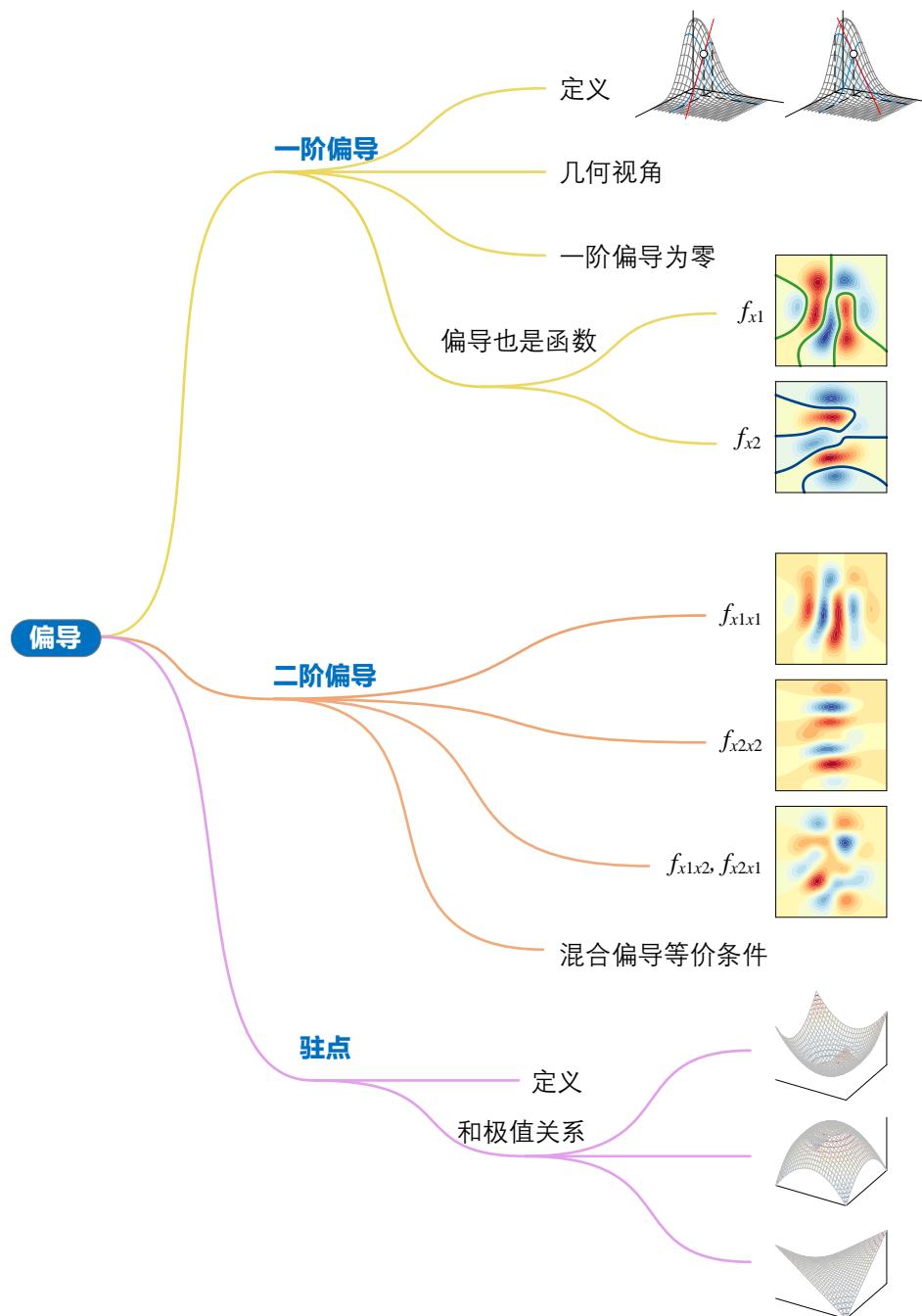
我不知道世人看我的眼光。依我看来，我不过是一个在海边玩耍的孩子，不时找到几个光滑卵石、漂亮贝壳，而惊喜万分；而展现在我面前的是，真理的浩瀚海洋，静候探索。

I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the sea-shore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.

——艾萨克·牛顿 (Isaac Newton) | 英国数学家、物理学家 | 1643 ~ 1727



- ◀ `ax.plot_surface()` 绘制三维曲面图
- ◀ `ax.plot_wireframe()` 绘制线框图
- ◀ `matplotlib.pyplot.contour()` 绘制等高线图
- ◀ `matplotlib.pyplot.contourf()` 绘制填充等高线图
- ◀ `sympy.abc` 引入符号变量
- ◀ `sympy.diff()` 求解符号导数和偏导解析式
- ◀ `sympy.exp()` 符号自然指数函数
- ◀ `sympy.lambdify()` 将符号表达式转化为函数
- ◀ `sympy.symbols()` 定义符号变量



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

16.1 几何角度看偏导数

上一章介绍一元函数导数时，我们知道它是一元函数的变化率。从几何角度来看，导数就是一元函数曲线上某点切线的斜率。

之前我们聊过，一般情况下二元函数 $f(x_1, x_2)$ 可以视作曲面。如图 1 所示， $f(x_1, x_2)$ 函数曲面上某一点 $(a, b, f(a, b))$ 如果光滑，该点处有无数条切线。

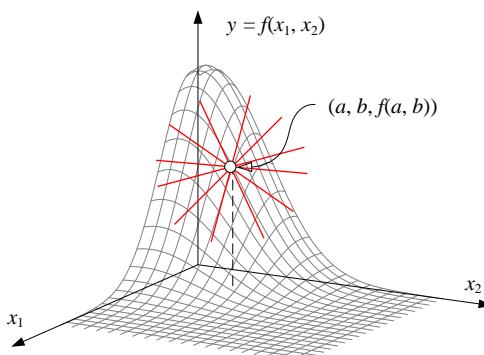


图 1. 光滑 $f(x_1, x_2)$ 某点的切线有无数条

而我们特别关注的两条切线是图 2 (a) 和 (b) 红色直线对应的切线。图 2 (a) 中切线平行于 x_1y 平面，图 2 (b) 中切线平行于 x_2y 平面。这用到的就是本书第 10 章介绍的剖面线思想。

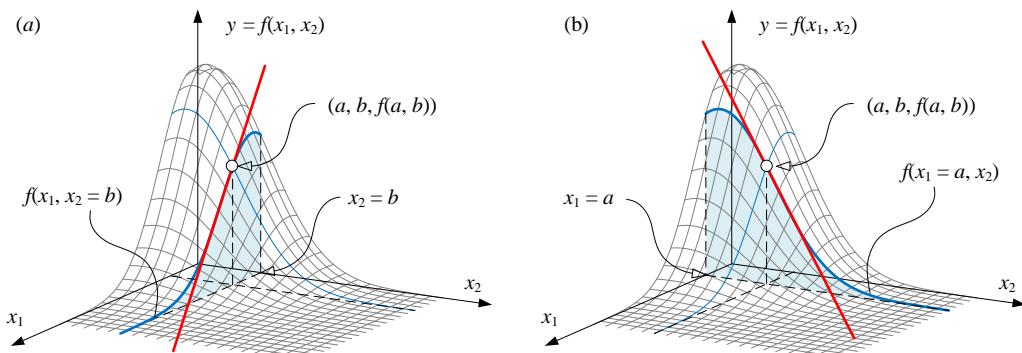


图 2. 几何视角看 $f(x_1, x_2)$ 偏导

把 x_2 固定在 b ，即 $x_2 = b$ ，图 2 (a) 切线斜率代表二元函数 $f(x_1, x_2)$ 在 (a, b) 处沿着 x_1 的变化率。把 x_1 固定在 a ，即 $x_1 = a$ ，图 2 (b) 切线斜率代表二元函数 $f(x_1, x_2)$ 在 (a, b) 处沿着 x_2 的变化率。**偏导数** (partial derivative) 正是研究这种二元乃至多元函数变化率的工具。

对于多元函数 $f(x_1, x_2, \dots, x_D)$ 来说，偏导数是关于函数的某一个特定变量 x_i 的导数，而保持其他变量恒定。

本节通过二元函数介绍偏导数的定义。

偏导数定义

设 $f(x_1, x_2)$ 是定义在 \mathbb{R}^2 上的二元函数， $f(x_1, x_2)$ 在点 (a, b) 的某一邻域内有定义。

将 x_2 固定在 $x_2 = b$ ， $f(x_1, x_2)$ 则变成一个关于 x_1 的一元函数 $f(x_1, b)$ 。 $f(x_1, b)$ 在 $x_1 = a$ 处关于 x_1 可导，则称 $f(x_1, x_2)$ 在点 (a, b) 处关于 x_1 可偏微分 (partially differentiable)。

用极限， $f(x_1, x_2)$ 在点 (a, b) 处关于 x_1 的偏导定义为：

$$f_{x_1}(a, b) = \frac{\partial f}{\partial x_1} \Big|_{\substack{x_1=a \\ x_2=b}} = \lim_{\Delta x_1 \rightarrow 0} \frac{f\left(a + \Delta x_1, b\right) - f(a, b)}{\Delta x_1} \quad (1)$$

图 2 (a) 网格面为 $f(x_1, x_2)$ 函数曲面。从几何角度看偏导数，平行 x_1y 平面，在 $x_2 = b$ 切一刀得到浅蓝色的剖面线，偏导 $f_{x_1}(a, b)$ 就是蓝色剖面线在 $(a, b, f(a, b))$ 一点的切线的斜率。⚠ 注意，在三维直角坐标系中，该切线平行 x_1y 平面。

类似地， $f(x_1, x_2)$ 在 (a, b) 点对于 x_2 的偏导可以定义为：

$$f_{x_2}(a, b) = \frac{\partial f}{\partial x_2} \Big|_{\substack{x_1=a \\ x_2=b}} = \lim_{\Delta x_2 \rightarrow 0} \frac{f\left(a, b + \Delta x_2\right) - f(a, b)}{\Delta x_2} \quad (2)$$

也从几何角度分析，如图 2 (b) 所示，偏导 $f_{x_2}(a, b)$ 就是蓝色剖面线在 $(a, b, f(a, b))$ 一点的切线斜率。该切线平行 x_2y 平面。

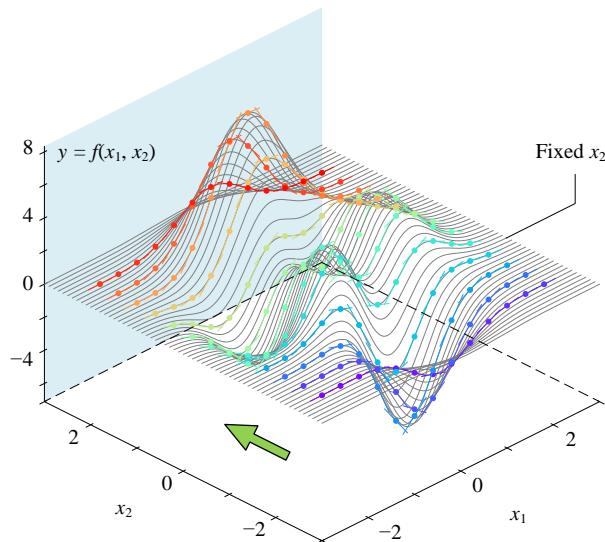
一个多极值曲面

下面用如下这个较复杂二元函数 $f(x_1, x_2)$ 讲解偏导：

$$f(x_1, x_2) = 3(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) \exp(-x_1^2 - x_2^2) - \frac{1}{3} \exp(-(x_1+1)^2 - x_2^2) \quad (3)$$

对 x_1 偏导

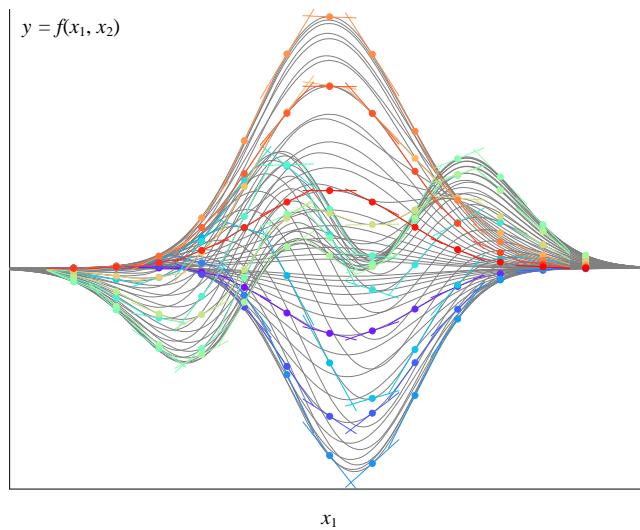
图 3 给出 $f(x_1, x_2)$ 曲面上一系列散点。在每一个散点处，绘制平行于 x_1y 平面的切线，这些切线的斜率就是该点处 $f(x_1, x_2)$ 对 x_1 的偏导 $\partial f / \partial x_1 = f_{x_1}$ 。

图 3. $f(x_1, x_2)$ 曲面上不同点处绘制 $f(x_1, x_2 = b)$ 切线

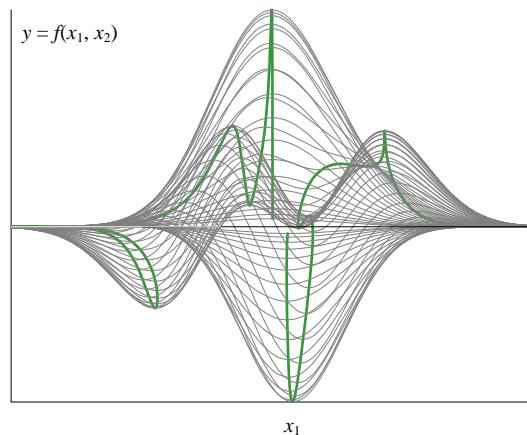
将这些切线投影到 x_1y 平面得到图 4。

如前文所述，固定 $x_2 = b$ ， $f(x_1, x_2)$ 这个二元函数变成了一个关于 x_1 的一元函数 $f(x_1, x_2 = b)$ 。不同 b 值对应不同的 $f(x_1, x_2 = b)$ 函数，对应图 4 中不同曲线。

在这些 $f(x_1, x_2 = b)$ 一元函数曲线上某点做切线，切线斜率就是二元函数 $f(x_1, x_2)$ 对 x_1 偏导。

图 4. $f(x_1, x_2 = b)$ 函数和切线在 x_1y 平面投影

再次观察图 4，发现每一条曲线都能找到至少一条切线平行于 x_1 轴，也就是切线斜率为 0。将这些切线斜率为 0 的点连在一起可以得到图 5 中绿色曲线。不难看出，绿色曲线经过曲面的每个“山峰”和“山谷”，也就是二元函数极大值和极小值。这一点观察对后续优化问题求解很重要。

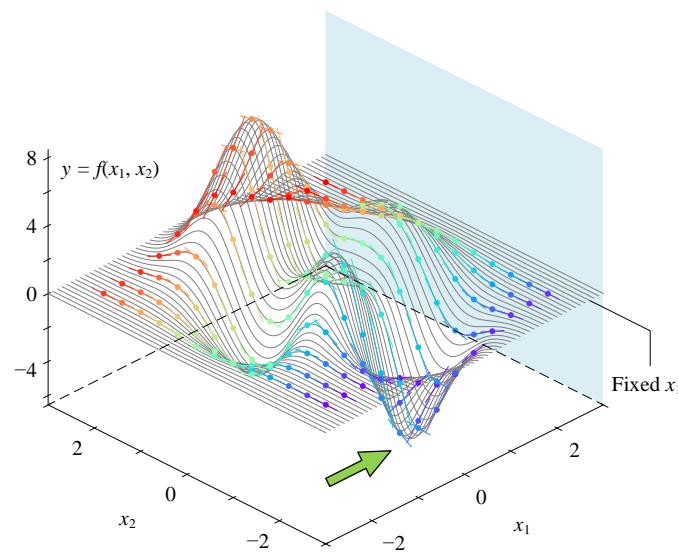
图 5. 将满足 $f_{x1}(x_1, x_2) = 0$ 的点连成线

对 x_2 偏导

下面，我们用同样几何视角分析 $f(x_1, x_2)$ 对 x_2 的偏导 $\partial f / \partial x_2 = f_{x2}$ 。

如图 6 所示，绘制 $f(x_1, x_2)$ 曲面上不同位置平行于 x_2y 平面的切线。而这些切线斜率就是不同点处 $f(x_1, x_2)$ 对 x_2 的偏导 $\partial f / \partial x_2 = f_{x2}$ 。

将这些切线投影到 x_2y 平面得到图 7。图中曲线都相当是一元函数，曲线上不同点切线斜率就是偏导。偏导用到的思维实际上也相当于“降维”，将三维曲面投影到平面上得到一系列曲线，然后再研究“变化率”。也就是说，偏导的内核实际上还是一元函数导数。

图 6. $f(x_1, x_2)$ 曲面上不同点处绘制 $f(x_1 = a, x_2)$ 切线

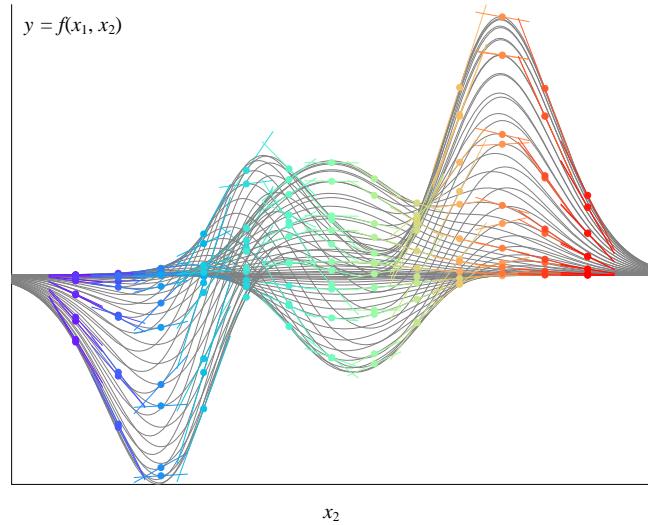
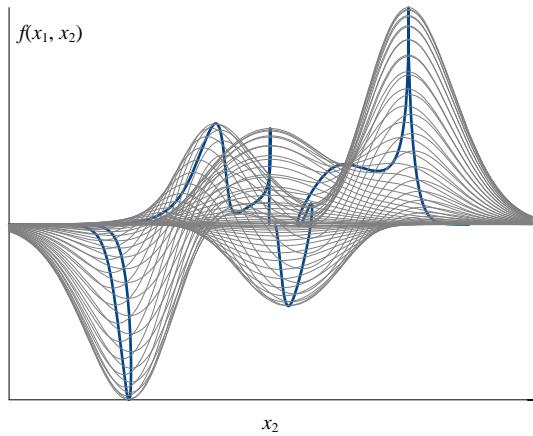
图 7. $f(x_1 = a, x_2)$ 函数和切线在 x_2y 平面投影

图 8 所示深蓝色曲线满足 $f_{x_2}(x_1, x_2) = 0$ 。同样，我们发现这条深蓝色曲线经过曲面的“山峰”和“山谷”。本章后文会换一个视角来看图 5 中绿色曲线和图 8 深蓝色曲线。

图 8. 将满足 $f_{x_2}(x_1, x_2) = 0$ 的点连成线

本章开头说到，光滑曲面任意一点有无数条切线；也就是说，给定曲面一点 $(a, b, f(a, b))$ 从不同角度都可以获得曲面在该点处切线。而对 x_1 偏导和对 x_2 偏导只能帮助我们定义两条切线。

大家可能会问，如何确定其他方向上切线斜率？这些“偏导数”又叫什么？

目前，我们已经掌握的数学工具尚不足以解决这个问题。我们把它留给本系列丛书《矩阵力量》一册。

表 1. 偏导数的英文表达

数学表达	英文表达
∂	Partial d, curly d, curved d, del
∂y	Partial y The partial derivative of y
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x Partial y over partial x Partial derivative with respect to x of y
$\frac{\partial^2 y}{\partial x^2}$	Partial two y by partial x squared The second partial derivative of y with respect to x
$\frac{\partial^2 f}{\partial x \partial y}$	Second partial derivative of f, first with respect to x and then with respect to y
$\frac{\partial f}{\partial x_1}$	The partial derivative of f with respect to x_1 sub one Partial d f over partial x_1 sub one
$\frac{\partial^2 f}{\partial x_1^2}$	The second partial derivative of f with respect to x_1 sub one Partial two f by partial x_1 sub one squared

16.2 偏导也是函数

上一章说到导数也叫导函数，这是因为导数也是函数；同样，偏导也叫偏导函数，因为它也是函数。

对 x_1 偏导

计算(3)给出的二元函数 $f(x_1, x_2)$ 对于 x_1 的一阶偏导 $f_{x1}(x_1, x_2)$ 解析式：

$$\begin{aligned}
 f_{x1}(x_1, x_2) = & -6x_1(1-x_1)^2 \exp(-x_1^2 - (x_2+1)^2) \\
 & -2x_1(10x_1^3 - 2x_1 + 10x_2^5) \exp(-x_1^2 - x_2^2) \\
 & -\frac{1}{3}(-2x_1 - 2) \exp(-x_2^2 - (x_1+1)^2) \\
 & +(6x_1 - 6) \exp(-x_2^2 - (x_1+1)^2) \\
 & +(30x_1^2 - 2) \exp(-x_1^2 - x_2^2)
 \end{aligned} \tag{4}$$

可以发现， $f_{x1}(x_1, x_2)$ 也是一个二元函数。

图 9 所示为 $f_{x1}(x_1, x_2)$ 曲面，请大家格外注意图中绿色等高线，它们对应 $f_{x1}(x_1, x_2) = 0$ (图 5 中绿色曲线)。

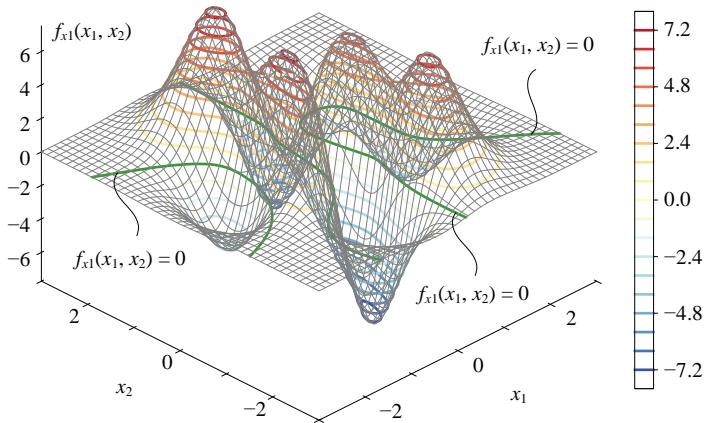
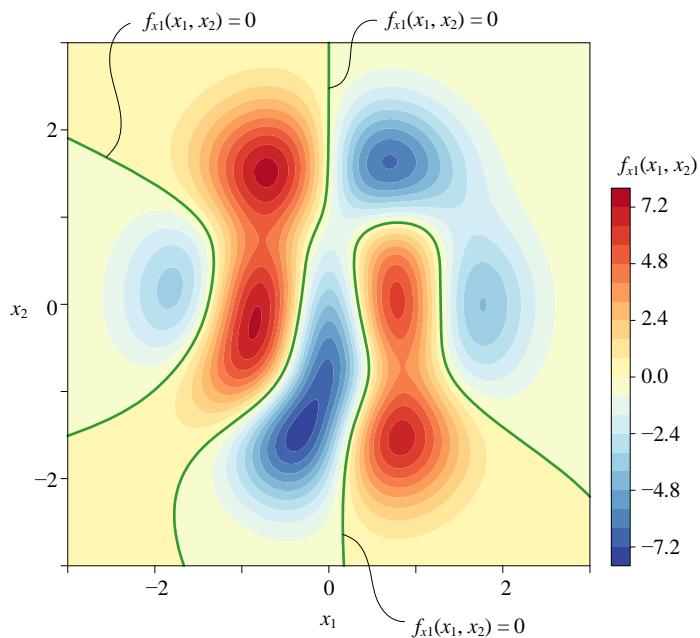
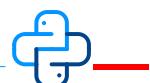
图 9. 二元函数 $f(x_1, x_2)$ 对 x_1 一阶偏导 $f_{x1}(x_1, x_2)$ 曲面

图 10 所示为 $f_{x1}(x_1, x_2)$ 平面填充等高线，从这个视角看 $f_{x1}(x_1, x_2) = 0$ 对应的绿色等高线更加方便。本章末将探讨绿色等高线和 $f(x_1, x_2)$ 曲面极值点的关系。

图 10. $f_{x1}(x_1, x_2)$ 平面填充等高线

代码文件 Bk3_Ch16_01.py 中 Bk3_Ch16_01_A 部分绘制图 9 和图 10。

对 x_2 偏导

配合前文代码，请自行计算 $f(x_1, x_2)$ 对于 x_2 的一阶偏导 $f_{x_2}(x_1, x_2)$ 解析式。图 11 所示为 $f_{x_2}(x_1, x_2)$ 曲面。

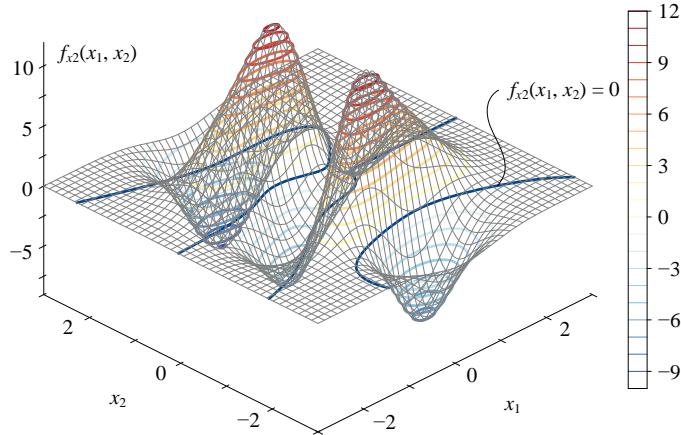


图 11. 二元函数 $f(x_1, x_2)$ 对 x_2 一阶偏导 $f_{x_2}(x_1, x_2)$ 曲面

图 12 所示为 $f_{x_2}(x_1, x_2)$ 曲面填充等高线，图中深蓝色等高线对应 $f_{x_2}(x_1, x_2) = 0$ 。

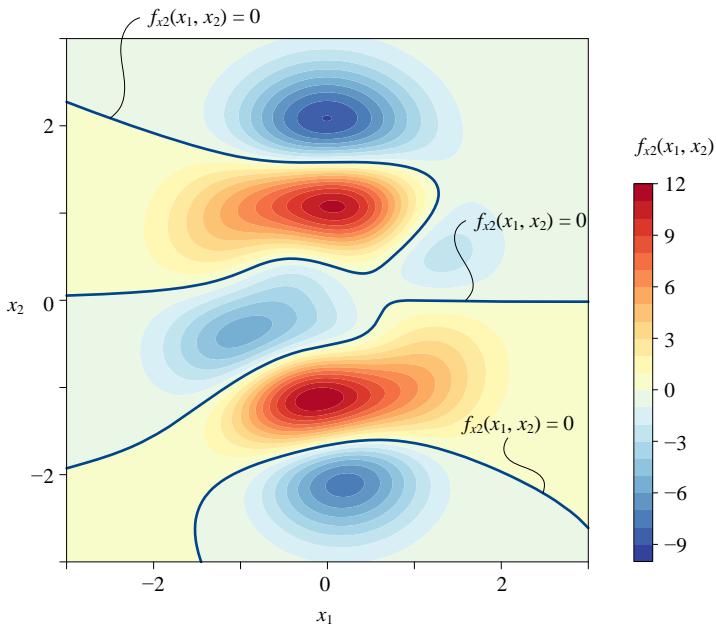
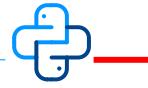


图 12. $f_{x_2}(x_1, x_2)$ 平面填充等高线



代码文件 Bk3_Ch16_01.py 中 Bk3_Ch16_01_B 部分绘制图 11 和图 12。

16.3 二阶偏导：一阶偏导函数的一阶偏导

假设某个二元函数 $f(x_1, x_2)$ 对 x_1 、 x_2 分别具有偏导数 $f_{x1}(x_1, x_2)$ 、 $f_{x2}(x_1, x_2)$ 。上一节内容告诉我们 $f_{x1}(x_1, x_2)$ 、 $f_{x2}(x_1, x_2)$ 也是关于 x_1 、 x_2 的二元函数。

如果一阶偏导函数 $f_{x1}(x_1, x_2)$ 、 $f_{x2}(x_1, x_2)$ 也有其各自一阶偏导数，则称“一阶偏导的一阶偏导”是 $f(x_1, x_2)$ 的二阶偏导数。

对 x_1 二阶偏导

$f_{x1}(x_1, x_2)$ 对 x_1 求一阶偏导得到 $f(x_1, x_2)$ 对 x_1 的二阶偏导，记做：

$$\frac{\partial}{\partial x_1} \left(\frac{\partial f}{\partial x_1} \right) = \frac{\partial^2 f}{\partial x_1^2} = f_{x1x1} = (f_{x1})_{x1} \quad (5)$$

图 13 所示为二阶偏导 f_{x1x1} 曲面和平面填充等高线。

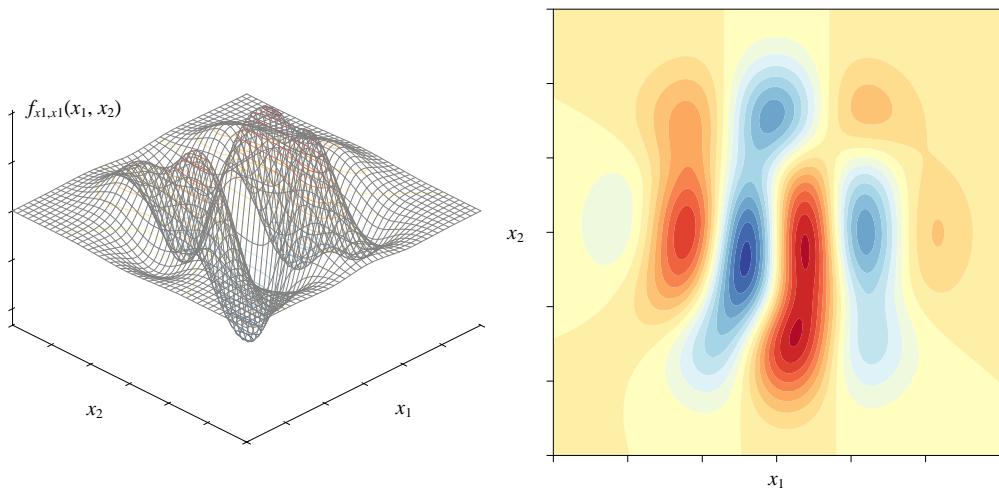
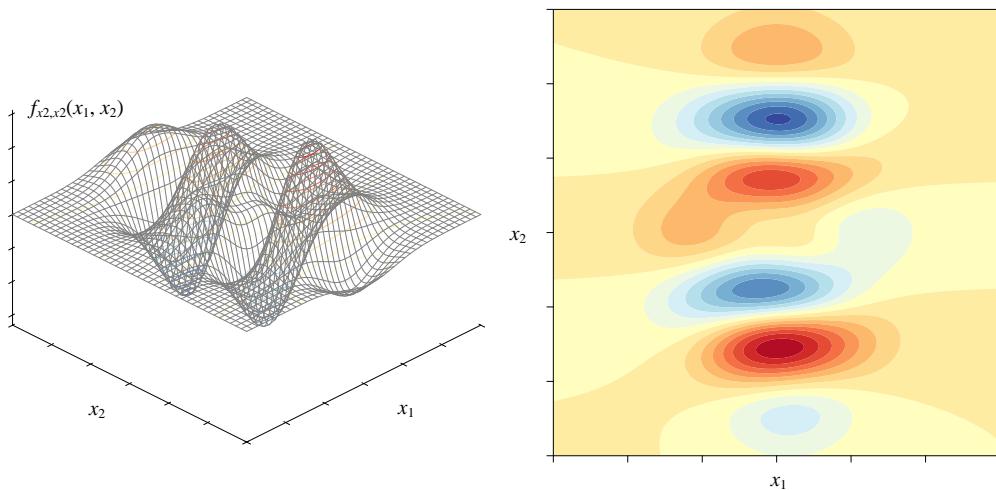


图 13. 二阶偏导 f_{x1x1} 曲面和平面填充等高线

对 x_2 二阶偏导

$f_{x2}(x_1, x_2)$ 对 x_2 求一阶偏导得到 $f(x_1, x_2)$ 对 x_2 的二阶偏导：

$$\frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_2} \right) = \frac{\partial^2 f}{\partial x_2^2} = f_{x2x2} = (f_{x2})_{x2} \quad (6)$$

图 14 所示为二阶偏导 f_{x2x2} 曲面和平面填充等高线。图 14. 二阶偏导 f_{x2x2} 曲面和平面填充等高线

二阶混合偏导

$f_{x1}(x_1, x_2)$ 对 x_2 求一阶偏导得到 $f(x_1, x_2)$ 先对 x_1 、后对 x_2 二阶混合偏导，记做：

$$\frac{\partial}{\partial x_2} \left(\frac{\partial f}{\partial x_1} \right) = \frac{\partial^2 f}{\partial x_1 \partial x_2} = f_{x1x2} = (f_{x1})_{x2} \quad (7)$$

⚠ 请大家注意偏导先后顺序，先 x_1 后 x_2 。

$f_{x2}(x_1, x_2)$ 对 x_1 求一阶偏导得到 $f(x_1, x_2)$ 先对 x_2 、后对 x_1 二阶混合偏导：

$$\frac{\partial}{\partial x_1} \left(\frac{\partial f}{\partial x_2} \right) = \frac{\partial^2 f}{\partial x_2 \partial x_1} = f_{x2x1} = (f_{x2})_{x1} \quad (8)$$

⚠ 再次请大家注意混合偏导的先后顺序。不同教材的记法存在顺序颠倒。为了方便大部分读者习惯，本章混合偏导记法采用同济大学编写的《高等数学》中记法规则。

如果函数 $f(x_1, x_2)$ 在某个特定区域内两个二阶混合偏导 f_{x2x1}, f_{x1x2} 连续，那么这两个混合偏导数相等，即：

$$\frac{\partial^2 f}{\partial x_2 \partial x_1} = \frac{\partial^2 f}{\partial x_1 \partial x_2} \quad (9)$$

对于(3)，函数的二阶偏导连续。因此， f_{x2x1} 和 f_{x1x2} 等价。图15所示为二阶偏导 $f_{x1x2}(=f_{x2x1})$ 曲面和填充等高线。

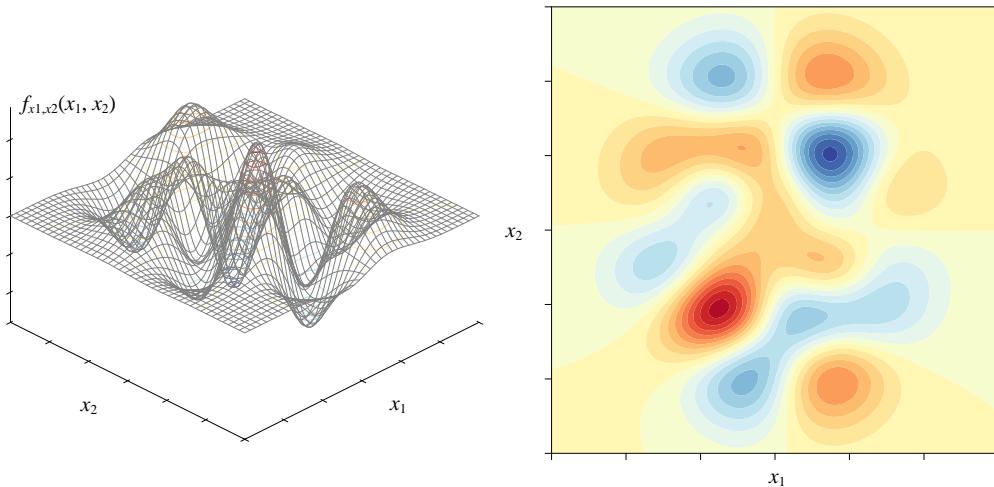


图15. 二阶偏导 $f_{x1x2}(=f_{x2x1})$ 曲面和填充等高线

和杨辉三角的联系

图16所示为偏导数和杨辉三角的联系。

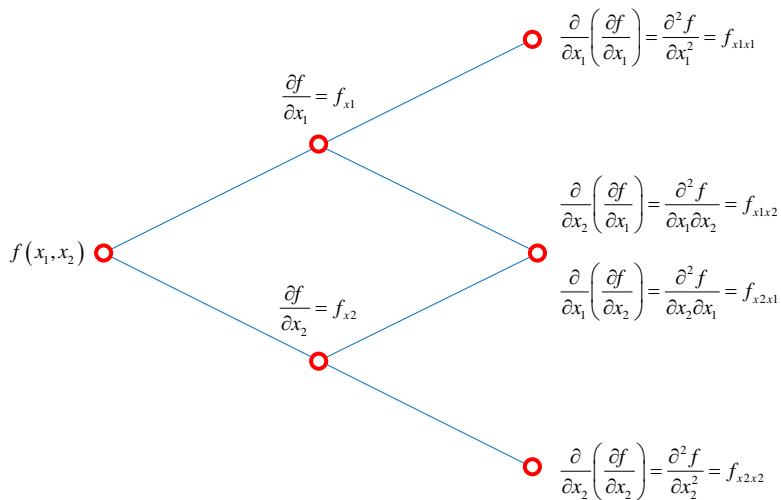


图16. 杨辉三角在偏导数的应用



代码文件 Bk3_Ch16_01.py 中 Bk3_Ch16_01_C 部分绘制图13、图14、图15。

16.4 二元曲面的驻点：一阶偏导为 0

上一章介绍过驻点这个概念。对于一元函数 $f(x)$ ，驻点处函数一阶导数为 0。从几何图像来看， $f(x)$ 在驻点的切线平行于横轴。驻点可能对应一元函数的极小值、极大值或鞍点。

而对于二元函数，驻点对应两个一阶偏导为 0 的点。几何角度，驻点处切面平行于水平面。

对 x_1 一阶偏导为 0

图 9 和图 10 给出 $f_{x1}(x_1, x_2) = 0$ 对应的坐标点 (x_1, x_2) 位置。如果将满足 $f_{x1}(x_1, x_2) = 0$ 等式的所有点映射到 $f(x_1, x_2)$ 曲面上，可以得到图 17 的绿色曲线。

仔细观察图 17 中绿色曲线，它们都经过 $f(x_1, x_2)$ 曲面上的极大值和极小值点。这一点，在图 18 填充等高线上看的更清楚。

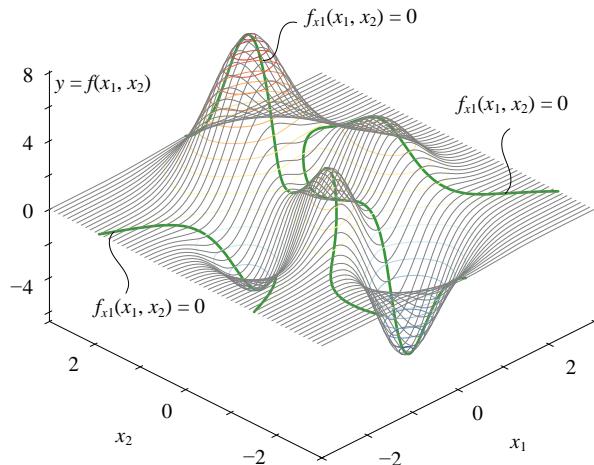
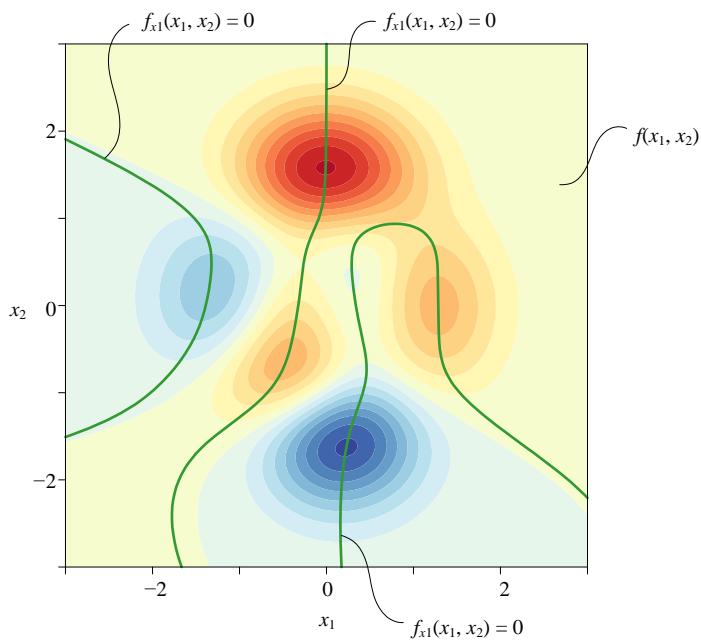
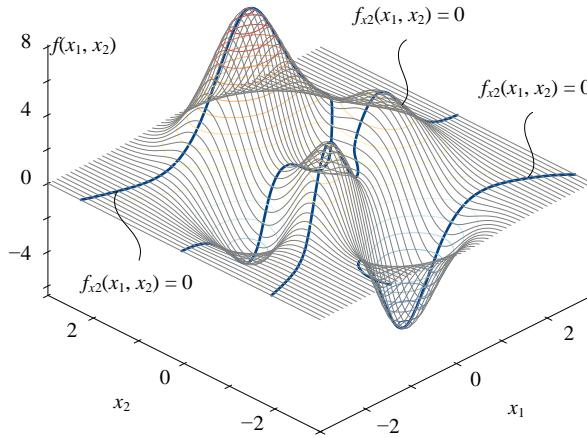


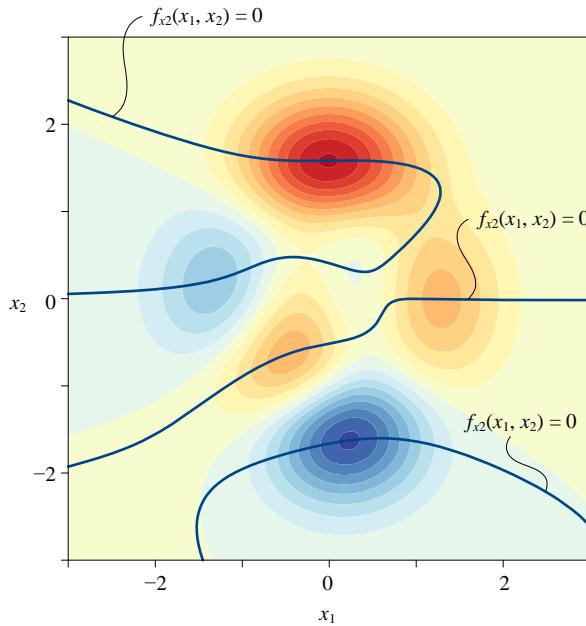
图 17. $f_{x1}(x_1, x_2) = 0$ 投影在 $f(x_1, x_2)$ 曲面上

图 18. 将 $f_{x1}(x_1, x_2) = 0$ 投影在 $f(x_1, x_2)$ 曲面填充等高线上

对 x_2 一阶偏导为 0

同理，图 11 和图 12 给出 $f_{x2}(x_1, x_2) = 0$ 对应的坐标点 (x_1, x_2) 位置。将满足 $f_{x2}(x_1, x_2) = 0$ 等式的所有点映射到 $f(x_1, x_2)$ 曲面上，得到图 19 的蓝色曲线。图 19 中蓝色曲线也都经过 $f(x_1, x_2)$ 曲面上的极大值和极小值点。图 20 所示为平面填充等高线图。

图 19. $f_{x2}(x_1, x_2) = 0$ 投影在 $f(x_1, x_2)$ 曲面上

图 20. 将 $f_{x2}(x_1, x_2) = 0$ 投影在 $f(x_1, x_2)$ 曲面填充等高线上

二元函数驻点

将 $f_{x1}(x_1, x_2) = 0$ (绿色曲线) 和 $f_{x2}(x_1, x_2) = 0$ (蓝色曲线) 同时映射到 $f(x_1, x_2)$ 曲面，得到图 21。

$f(x_1, x_2)$ 曲面山峰和山谷，也就是极大和极小值点，正好都位于蓝色和绿色曲线的交点处。

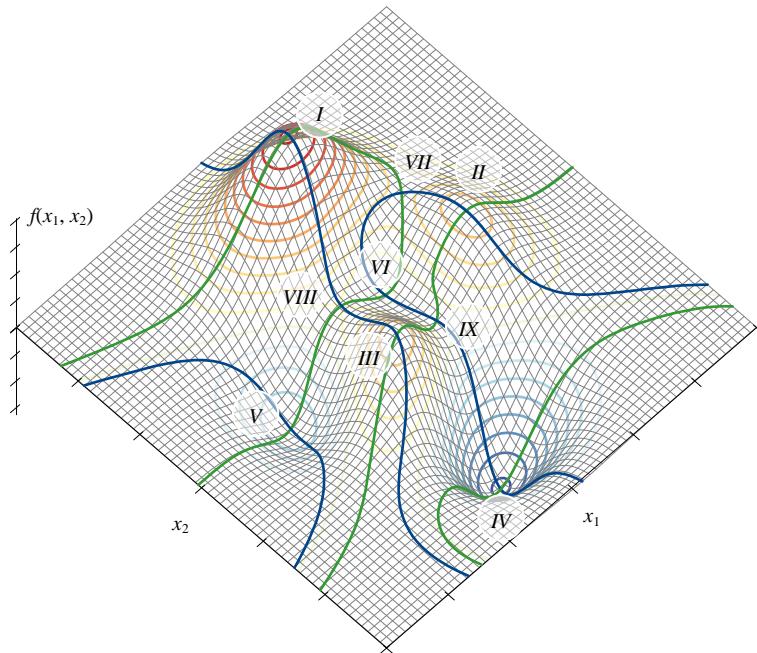
图 21. $f_{x1}(x_1, x_2) = 0$ 和 $f_{x2}(x_1, x_2) = 0$ 同时投影在 $f(x_1, x_2)$ 曲面上

图 22 给出的等高线更容易发现，I、II、III 点为极大值点，其中 I 为最大值点。IV、V、VI 为极小值点，其中 IV 为最小值点。

于此同时，我们也发现还有三个蓝绿曲线的交点 VII、VIII、IX，它们既不是极大值点，也不是极小值点。VII、VIII、IX 就是所谓的鞍点。

比如，在 IX 点，沿着绿色线向 IV 运动是下山，而沿着蓝色线向 III 运动是上山。

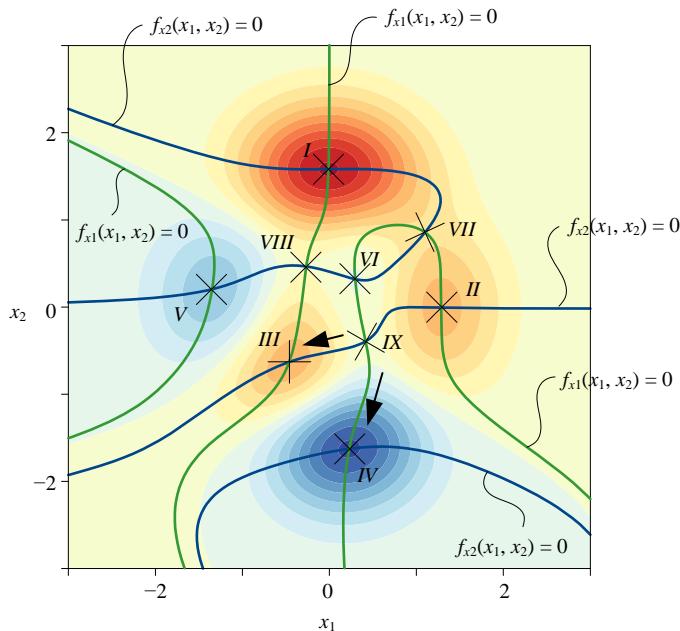


图 22. $f_{x1}(x_1, x_2) = 0$ 和 $f_{x2}(x_1, x_2) = 0$ 同时投影在 $f(x_1, x_2)$ 曲面填充等高线



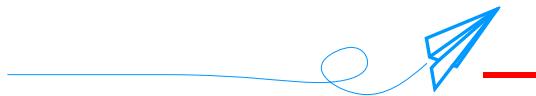
对于具有多个“山峰”和“山谷”的曲面，利用一阶偏导为 0 来判断极值点显然不充分。本书将在第 19 章介绍如何判断二元函数的极值点。



代码文件 Bk3_Ch16_01.py 中 Bk3_Ch16_01_D 部分绘制图 18、图 20、图 21、图 22 四副图像。请读者自行绘制图 17 和图 19 两幅图像。



在 Bk3_Ch16_01.py 基础上，我们做了一个 App 并用 Plotly 绘制偏导函数的 3D 交互曲面。请参考 Streamlit_Bk3_Ch16_01.py。



一元函数导数是函数变化率，几何角度是曲线切线斜率。本章利用“降维”这个思路，将一元函数导数这个数学工具拿来分析二元函数；对于二元函数或多元函数，我们给这个数学工具取了个名字叫做“偏导数”。“偏”字就是只考虑一个变量，或一个维度的意思。我们在介绍大西格玛 Σ 时，也创造了“偏求和”这个概念；在之后的积分内容中，我们还会见到“偏积分”。

本章还利用剖面线和等高线这两个可视化工具分析二元函数特征。请大家格外注意二元函数鞍点的性质。

17 Differential 微分

微分是线性近似



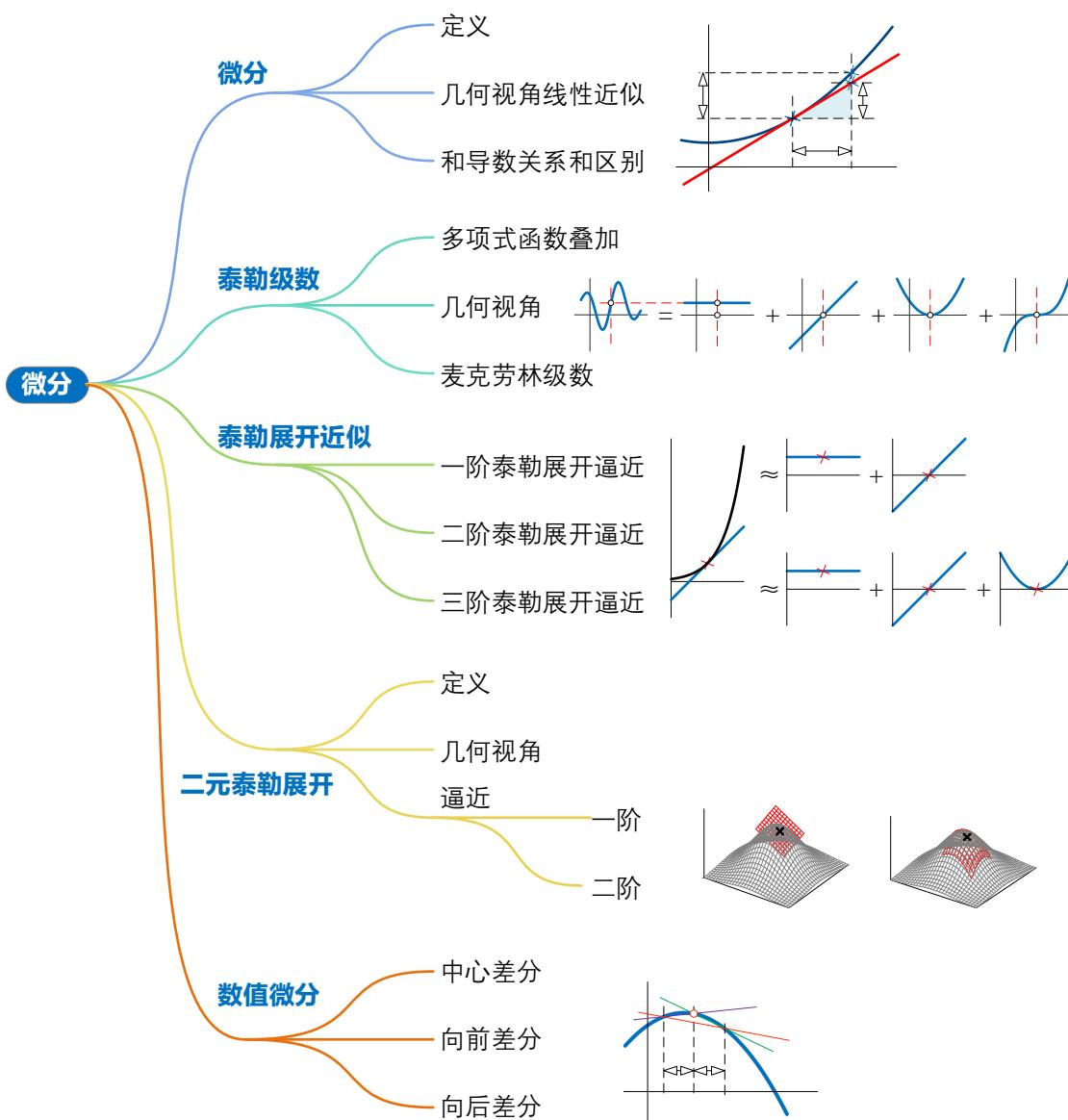
我看的比别人更远，那是因为我站在一众巨人们的肩膀之上。

If I have seen further than others, it is by standing upon the shoulders of giants.

——艾萨克·牛顿 (Isaac Newton) | 英国数学家、物理学家 | 1643 ~ 1727



- ◀ `numpy.meshgrid()` 获得网格数据
- ◀ `from 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.evalf()` 将符号解析式中未知量替换为具体数值
- ◀ `sympy.lambdify()` 将符号表达式转化为函数
- ◀ `sympy.series()` 求解泰勒展开级数符号式
- ◀ `sympy.symbols()` 定义符号变量



17.1 几何角度看微分：线性近似

微分 (differential) 是函数的局部变化的一种线性描述。如图 1 所示，微分可以近似地描述当函数自变量取值出现足够小 Δx 变化时，函数值变化 Δy 。

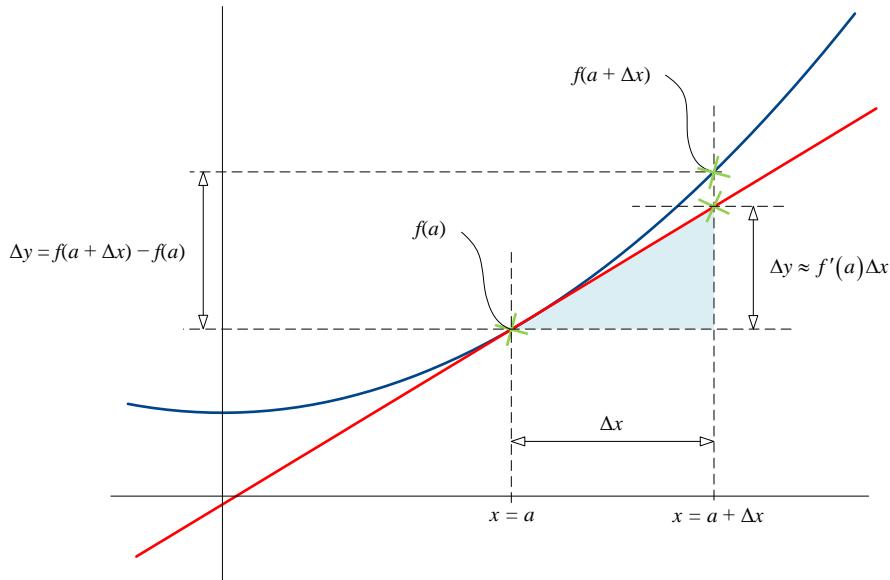


图 1. 对一元函数来说，微分是线性近似

假设函数 $f(x)$ 在某个区间内有定义。给定该区间内一点 a ，当 a 变动到 $a + \Delta x$ (也在该区间内) 时，函数实际增量为 Δy ：

$$\Delta y = f(a + \Delta x) - f(a) \quad (1)$$

而增量 Δy 可以近似为：

$$\Delta y = f(a + \Delta x) - f(a) \approx f'(a)\Delta x \quad (2)$$

其中， $f'(a)$ 为函数在 $x = a$ 处一阶导数。本书第 15 章讲过，函数 $f(x)$ 在某一点处一阶导数值是函数在该点处切线的斜率值。

整理上式， $f(a + \Delta x)$ 的近似写成：

$$f(a + \Delta x) \approx f'(a)\Delta x + f(a) \quad (3)$$

令

$$x = a + \Delta x \quad (4)$$

(3) 可以写成：

$$f(x) \approx f'(a)(x-a) + f(a) \quad (5)$$

上式就是一次函数的点斜式。一次函数通过 $(a, f(a))$ 这点，斜率为 $f'(a)$ 。

如图 1 所示，从几何角度，微分用切线这条斜线代替曲线。实践中，复杂的非线性函数可以通过局部线性化来简化运算。

图 2 和图 3 分别所示为高斯函数和其一阶导数函数在若干点处的切线。

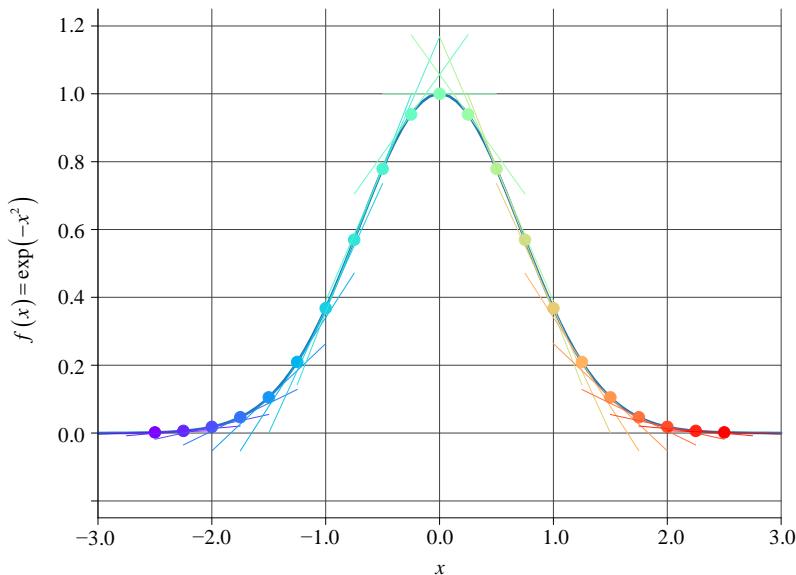


图 2. 高斯函数不同点处切线

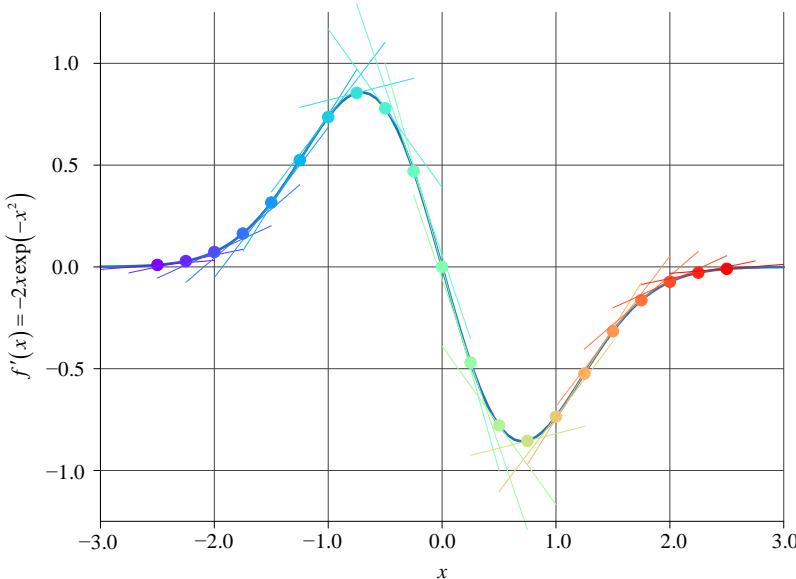


图 3. 高斯函数一阶导数不同点处切线

17.2 泰勒级数：多项式函数近似

英国数学家布鲁克·泰勒 (Sir Brook Taylor) 在 1715 年发表了泰勒级数 (Taylor's theorem)。泰勒级数是一种强大的函数近似工具。



布鲁克·泰勒 (Brook Taylor)
英国数学家 | 1685年 ~ 1731年
以泰勒公式和泰勒级数闻名

当展开点 (expansion point) 为 $x = a$ 时，一元函数 $f(x)$ 泰勒展开 (Taylor expansion) 形式为：

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n \\ &= f(a) + \underbrace{\frac{f'(a)}{1!}(x-a)}_{\text{Constant}} + \underbrace{\frac{f''(a)}{2!}(x-a)^2}_{\text{Linear}} + \underbrace{\frac{f'''(a)}{3!}(x-a)^3}_{\text{Quadratic}} + \dots \end{aligned} \quad (6)$$

其中， a 为展开点 (expansion point)。式中的阶乘是多项式求导产生的。展开点为 0 的泰勒级数又叫做麦克劳林级数 (Maclaurin series)。

如图 4 所示，泰勒展开相当于一系列多项式函数叠加，用来近似某个复杂函数。

⚠ 注意，图中常数函数图像对应的高度 $f(a)$ 提供了 $x=a$ 处 $f(x)$ 的函数值。而剩余其他多项式函数在展开点 $x=a$ 处函数值均为 0。

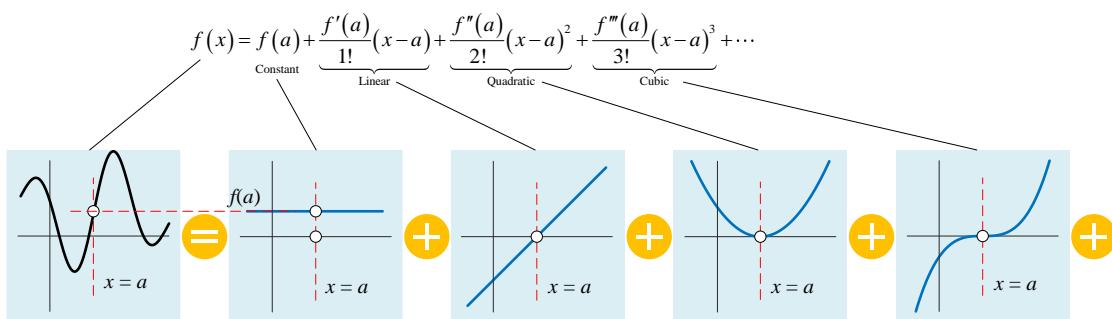


图 4. 一元函数泰勒展开原理

实际应用中，在应用泰勒公式近似计算时需要截断，也就是只取有限项。

上一节介绍微分时，(5) 实际上就是泰勒公式取前两项，即用“常数函数 + 一次函数”叠加近似原函数 $f(x)$ ：

$$f(x) \approx f(a) + \underbrace{\frac{f'(a)}{1!}(x-a)}_{\text{Linear}} = f(a) + f'(a)(x-a) \quad (7)$$

在(7)“常数函数+一次函数”基础上，再增加“二次函数”成分，我们便得到二次近似：

$$f(x) \approx f(a) + \underbrace{\frac{f'(a)}{1!}(x-a)}_{\text{Linear}} + \underbrace{\frac{f''(a)}{2!}(x-a)^2}_{\text{Quadratic}} \quad (8)$$

图5和图6分别所示为高斯函数和其一阶导数函数的二次近似。泰勒公式把复杂函数转换为多项式叠加。相较其他函数，多项式函数更容易计算微分、积分。本章后续将会介绍利用泰勒展开近似。

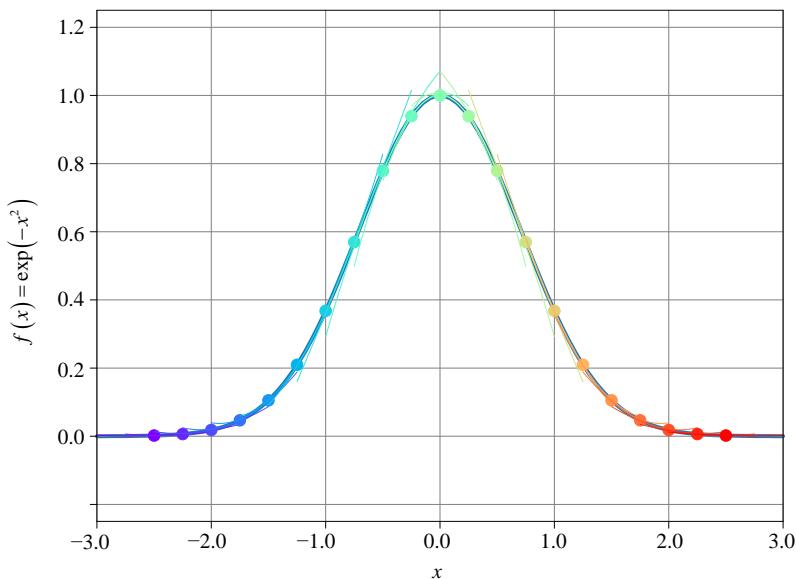


图5. 高斯函数不同点处二次近似

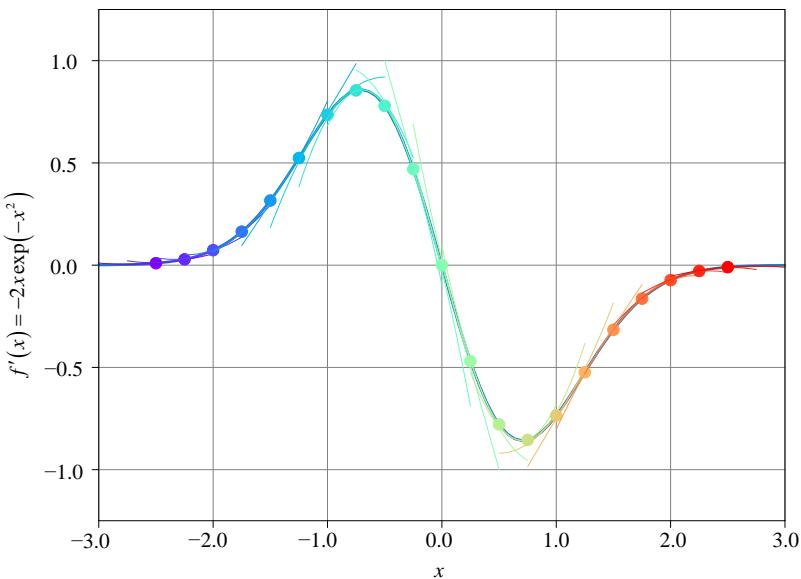


图 6. 高斯函数一阶导数不同点处二次近似



Bk3_Ch17_01.py 绘制图 5 和图 6。



在 Bk3_Ch17_01.py 基础上，我们做了一个 App 展示曲线上不同点一次和二次近似。请参考 Streamlit_Bk3_Ch17_01.py。

17.3 多项式近似和误差

再次强调，泰勒展开的核心是用一系列多项式函数叠加来逼近某个函数。实际应用中，泰勒级数常用来近似计算复杂非线性函数，并估计误差。

给定原函数 $f(x)$ 为自然指数函数：

$$f(x) = \exp(x) = e^x \quad (9)$$

在 $x=0$ 处，该函数的泰勒级数展开为：

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \frac{x^0}{0!} + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \dots = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \dots \quad (10)$$

如前文所述，在具体应用场合，泰勒公式需要截断，只取有限项进行近似运算。一个函数的有限项的泰勒级数叫做泰勒展开式。

常数函数

在 $x=0$ 点处， $f(x)$ 函数值为：

$$f(0) = \exp(0) = 1 \quad (11)$$

图 7 所示为用常数函数来近似原函数：

$$f_0(x) = \underset{\text{Constant}}{1} \quad (12)$$

图 8 比较原函数和常数函数，并给出误差随 x 变化。常数函数为平行横轴的直线，它的估计能力显然明显不足。

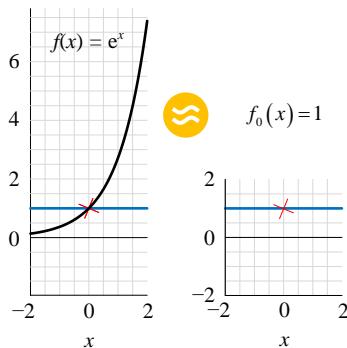


图 7. 常数函数近似

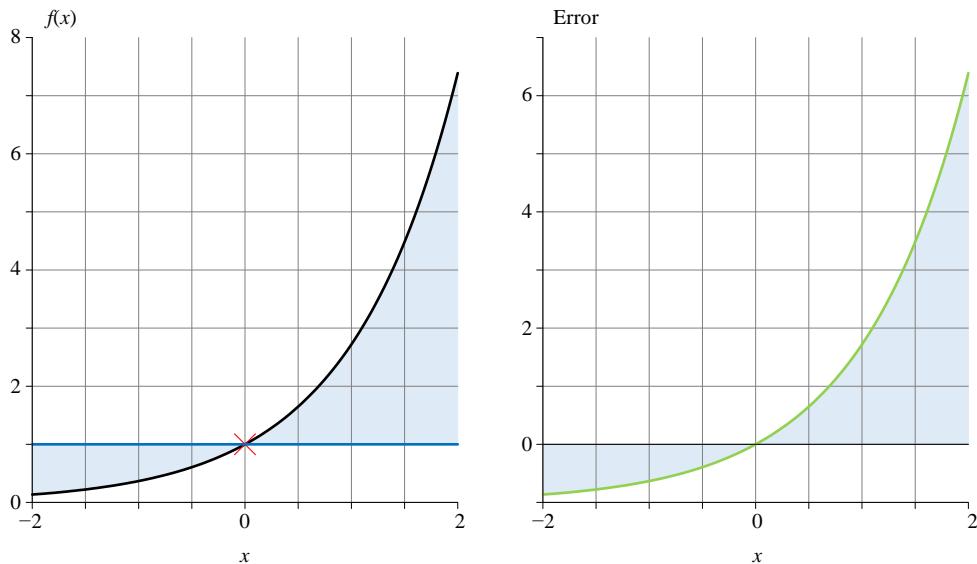


图 8. 常数函数近似及误差

一次函数

原函数 $f(x)$ 一阶导数为：

$$f'(x) = \exp(x) \quad (13)$$

$x = 0$ 处一阶导数为切线斜率：

$$f'(0) = \exp(0) = 1 \quad (14)$$

用一次函数来近似原函数：

$$f_1(x) = \begin{array}{c} 1 \\ \text{Constant} \end{array} + \begin{array}{c} x \\ \text{Linear} \end{array} \quad (15)$$

图 9 所示为“常数函数 + 一次函数”近似的原理。

叠加常数函数和一次函数，常被称作一阶泰勒展开 (first-order Taylor polynomial/expansion/approximation 或 first-degree Taylor polynomial)。一阶泰勒展开是最常用的逼近手段。

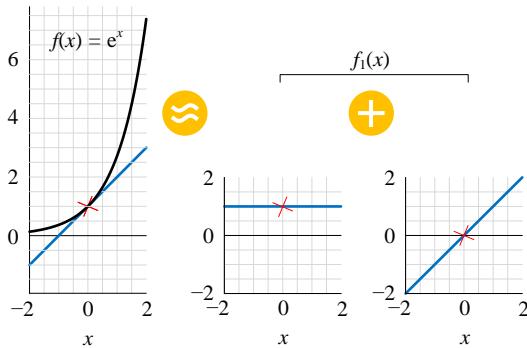


图 9.“常数函数 + 一次函数”近似

原函数和泰勒多项式的差被称作为泰勒公式的余项，即误差：

$$R(x) = f(x) - f_1(x) = f(x) - \left(\underset{\text{Constant}}{1} + \underset{\text{Linear}}{x} \right) \quad (16)$$

图 10 所示为一阶泰勒展开近似和误差；离展开点 $x = a$ 越远，误差越大。

也就是说，非线性函数在 $x = a$ 附近可以用这个一次函数近似。当 x 远离 a ，这个近似就变得越不准确。

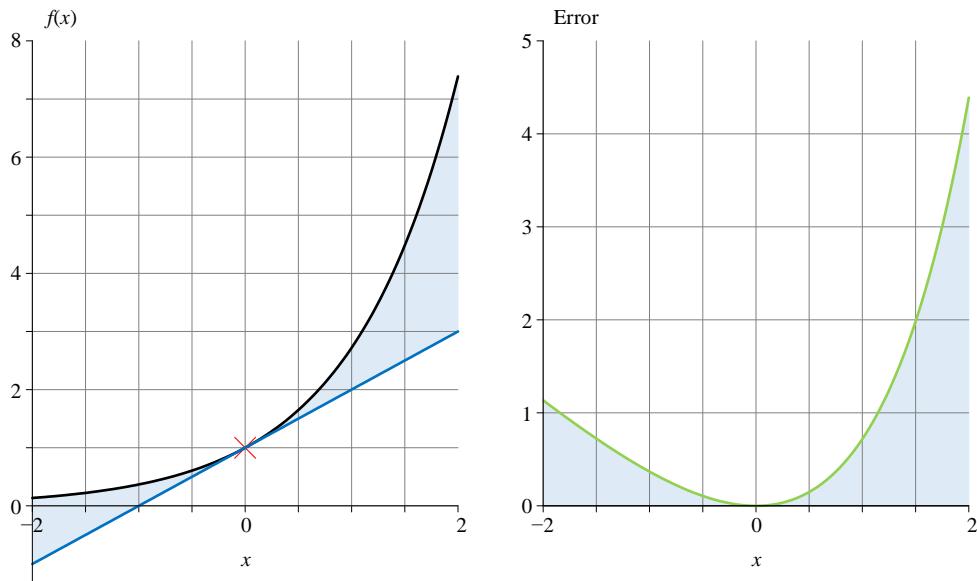


图 10. 一阶泰勒展开近似及误差

二次函数

用二次多项式函数近似原函数：

$$f_2(x) = \underbrace{1}_{\text{Constant}} + \underbrace{x}_{\text{Linear}} + \underbrace{\frac{x^2}{2}}_{\text{Quadratic}} \quad (17)$$

上式也叫二阶泰勒展开 (second-order Taylor polynomial/expansion/approximation)。图 11 所示，二阶泰勒展开叠加了三个成分，“常数函数 + 一次函数 + 二次函数”。

图 12 给出的是二阶泰勒展开近似及误差。相较图 10，图 12 中误差明显变小。

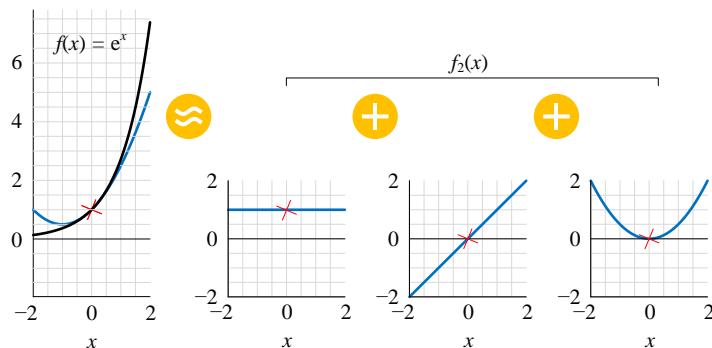


图 11. “常数函数 + 一次函数 + 二次函数”近似原函数

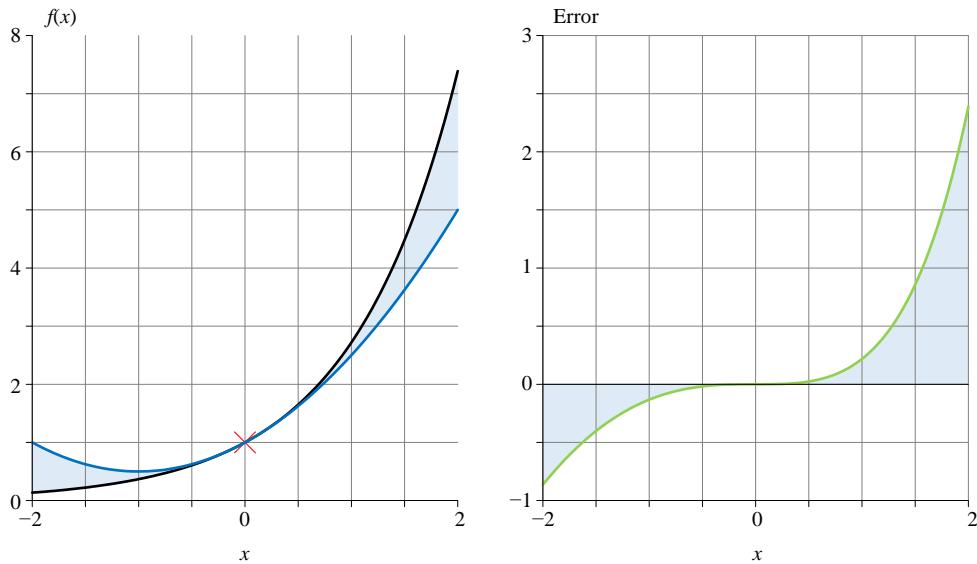


图 12. 二阶泰勒展开近似及误差

三次函数

用三次多项式函数近似原函数：

$$f_3(x) = \begin{array}{c} 1 \\ \text{Constant} \end{array} + \begin{array}{c} x \\ \text{Linear} \end{array} + \begin{array}{c} x^2 \\ \text{Quadratic} \end{array} + \begin{array}{c} x^3 \\ \text{Cubic} \end{array} \quad (18)$$

图 13 所示为“常数函数 + 一次函数 + 二次函数 + 三次函数”叠加近似原函数。比较图 12 和图 14，增加三次项后，逼近效果提高，误差进一步减小。

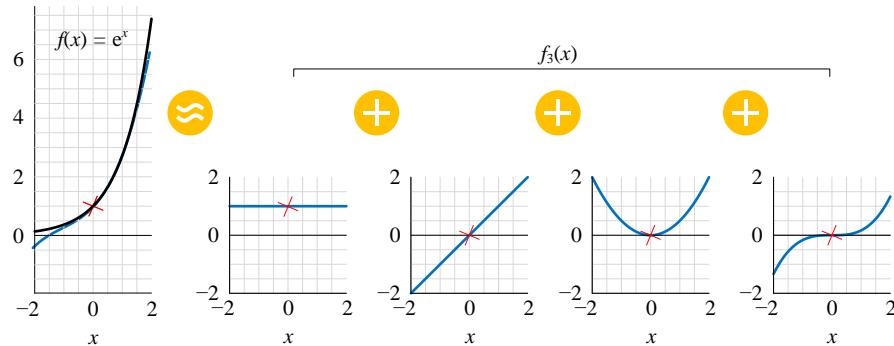


图 13.“常数函数 + 一次函数 + 二次函数 + 三次函数”近似原函数

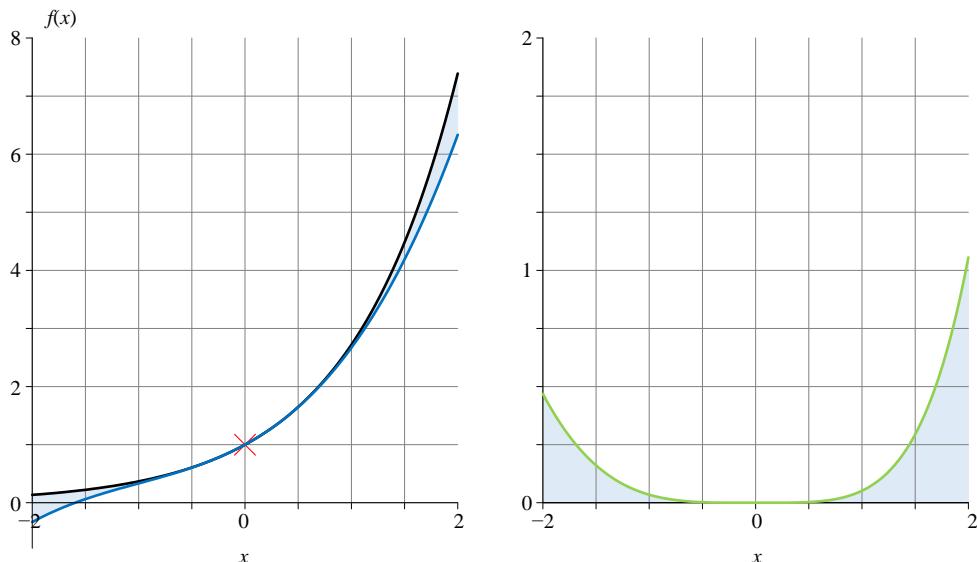


图 14. 三函数近似原函数及误差

四次函数

图 15 所示为用四次多项式函数近似原函数：

$$f(x) = \exp(x) \approx f_4(x) = \begin{matrix} \text{Constant} & + & \text{Linear} & + & \frac{x^2}{2} & + & \frac{x^3}{6} & + & \frac{x^4}{24} \end{matrix} \quad (19)$$

一般来说，泰勒多项式展开的项数越多，也就是说多项式幂次越高，逼近效果越好；但是，实际应用中，线性逼近和二次逼近用的最为广泛。对于误差分析本书不做探讨，对于误差分析感兴趣的同学可以自行学习。

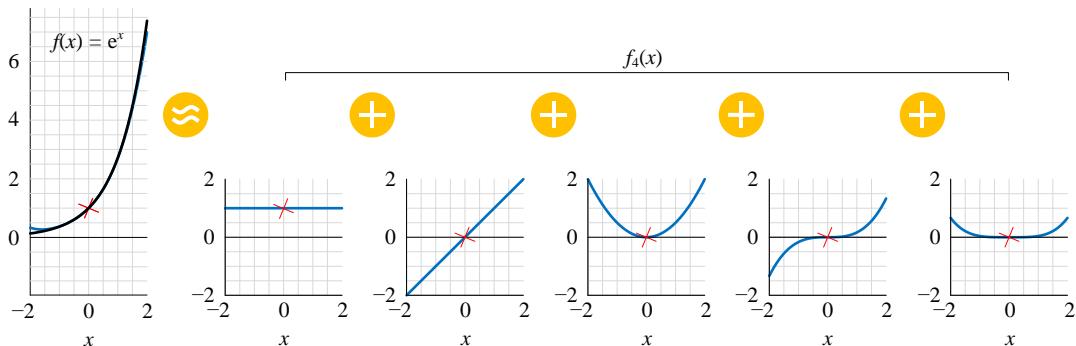
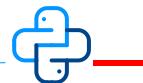


图 15.“常数函数 + 一次函数 + 二次函数 + 三次函数 + 四次函数”近似原函数



Bk3_Ch17_02.py 绘制本节图像；请大家改变展开点位置，比如 $x=1$ 、 $x=-1$ ，并观察比较近似及误差。



在 Bk3_Ch17_02.py 基础上，我们做了一个 App 比较不同泰勒展开项数对逼近结果的影响。请参考 Streamlit_Bk3_Ch17_02.py。

17.4 二元泰勒展开：用多项式曲面近似

上一节介绍的一元泰勒展开也可以扩展到多元函数。本节以二元函数为例介绍多元函数泰勒展开。

给定二元函数 $f(x_1, x_2)$ ，它的泰勒展开可以写成：

$$\begin{aligned}
 f(x_1, x_2) = & \underbrace{f(a, b)}_{\text{Constant}} + \underbrace{f_{x_1}(a, b)(x_1 - a) + f_{x_2}(a, b)(x_2 - b)}_{\text{Plane}} \\
 & + \underbrace{\frac{1}{2!} \left[f_{x_1 x_1}(a, b)(x_1 - a)^2 + 2f_{x_1 x_2}(a, b)(x_1 - a)(x_2 - b) + f_{x_2 x_2}(a, b)(x_2 - b)^2 \right]}_{\text{Quadratic}} + \dots
 \end{aligned} \tag{20}$$

⚠ 注意，式中假定两个混合偏导相同，即 $f_{x_1 x_2}(a, b) = f_{x_2 x_1}(a, b)$ 。

将(20)写成矩阵运算形式：

$$f(x_1, x_2) = f(a, b) + \begin{bmatrix} f_{x_1}(a, b) \\ f_{x_2}(a, b) \end{bmatrix}^T \begin{bmatrix} x_1 - a \\ x_2 - b \end{bmatrix} + \frac{1}{2!} \begin{bmatrix} x_1 - a \\ x_2 - b \end{bmatrix}^T \begin{bmatrix} f_{x_1 x_1}(a, b) & f_{x_1 x_2}(a, b) \\ f_{x_2 x_1}(a, b) & f_{x_2 x_2}(a, b) \end{bmatrix} \begin{bmatrix} x_1 - a \\ x_2 - b \end{bmatrix} + \dots \tag{21}$$

如图16所示，从几何角度，二元函数泰勒展开相当于，水平面、斜面、二次曲面、三次曲面等多项式曲面叠加。

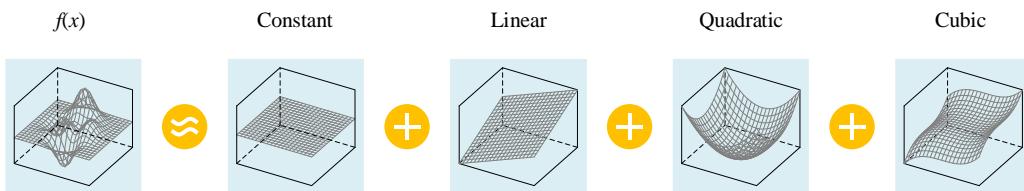


图 16. 二元函数泰勒展开原理

举个例子

给定如下二元高斯函数：

$$y = f(x_1, x_2) = \exp(-\left(x_1^2 + x_2^2\right)) \tag{22}$$

二元函数 $f(x_1, x_2)$ 一阶偏导：

$$\begin{aligned}
 f_{x_1}(x_1, x_2) &= \frac{\partial f}{\partial x_1}(x_1, x_2) = -2x_1 \exp(-\left(x_1^2 + x_2^2\right)) \\
 f_{x_2}(x_1, x_2) &= \frac{\partial f}{\partial x_2}(x_1, x_2) = -2x_2 \exp(-\left(x_1^2 + x_2^2\right))
 \end{aligned} \tag{23}$$

图17所示为两个一阶偏导函数的平面等高线图；图中 \times 为展开点位置，水平面位置坐标 $(-0.1, -0.2)$ 。

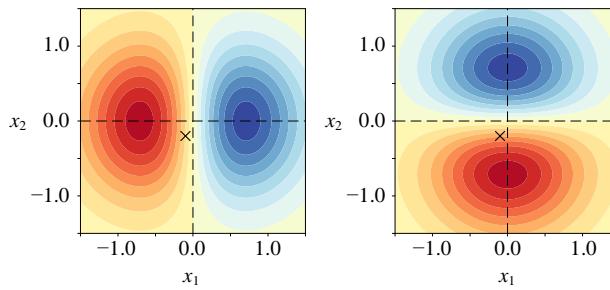


图 17. 一阶偏导数曲面等高线

(22) 中二元函数 $f(x_1, x_2)$ 二阶偏导：

$$\begin{aligned} f_{x_1x_1}(x_1, x_2) &= \frac{\partial^2 f}{\partial x_1^2}(x_1, x_2) = (-2 + 4x_1^2)\exp(-x_1^2 - x_2^2) \\ f_{x_2x_2}(x_1, x_2) &= \frac{\partial^2 f}{\partial x_2^2}(x_1, x_2) = (-2 + 4x_2^2)\exp(-x_1^2 - x_2^2) \\ f_{x_1x_2}(x_1, x_2) &= \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_1, x_2) = 4x_1 x_2 \exp(-x_1^2 - x_2^2) \\ f_{x_2x_1}(x_1, x_2) &= \frac{\partial^2 f}{\partial x_2 \partial x_1}(x_1, x_2) = 4x_1 x_2 \exp(-x_1^2 - x_2^2) \end{aligned} \quad (24)$$

显然，两个混合偏导相同，即，

$$f_{x_1x_2}(x_1, x_2) = f_{x_2x_1}(x_1, x_2) \quad (25)$$

图 18 所示为两个二阶偏导函数的平面等高线图。

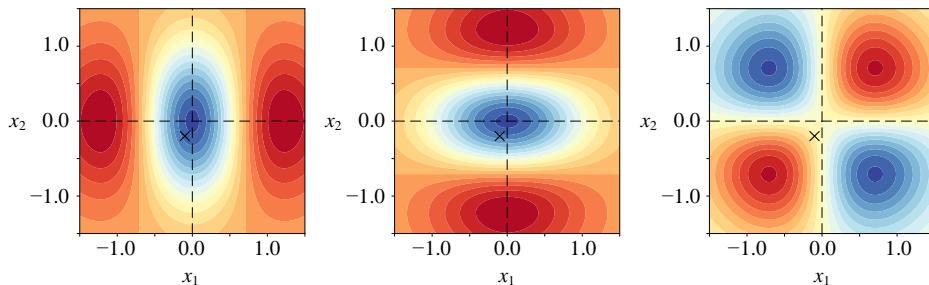


图 18. 二阶偏导数曲面等高线

展开点 $\times (-0.1, -0.2)$ 处函数值、一阶、二阶偏导数具体值为：

$$f(-0.1, -0.2) = 0.951, \quad \begin{cases} f_{x_1}(-0.1, -0.2) = 0.190 \\ f_{x_2}(-0.1, -0.2) = 0.380 \end{cases}, \quad \begin{cases} f_{x_1x_1}(-0.1, -0.2) = -1.864 \\ f_{x_2x_2}(-0.1, -0.2) = -1.750 \\ f_{x_1x_2}(-0.1, -0.2) = f_{x_2x_1}(-0.1, -0.2) = 0.076 \end{cases} \quad (26)$$

常数函数

类似前文，我们本节也采用逐步分析。首先用二元常函数数来估计 $f(x_1, x_2)$ ：

$$f(x_1, x_2) \approx \underbrace{f(a, b)}_{\text{Constant}} \quad (27)$$

这相当于用一个平行于 x_1x_2 平面的水平面来近似 $f(x_1, x_2)$ 。

图 19 所示为用常数函数估计二元高斯函数，常数函数对应解析式为：

$$f(x_1, x_2) \approx f(-0.1, -0.2) = 0.951 \quad (28)$$

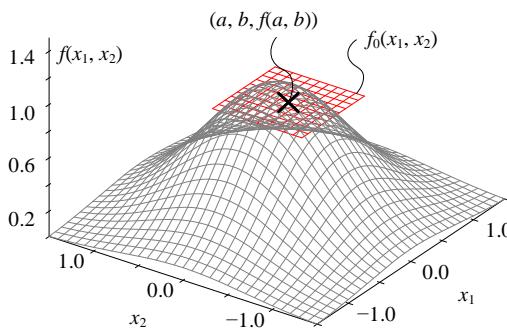


图 19. 用常数函数估计二元高斯函数

一次函数

用一次泰勒展开估计 $f(x_1, x_2)$ ：

$$f(x_1, x_2) \approx \underbrace{f(a, b)}_{\text{Constant}} + \underbrace{f_{x_1}(a, b)(x_1 - a) + f_{x_2}(a, b)(x_2 - b)}_{\text{Plane}} \quad (29)$$

相当于“水平面 + 斜面”叠加来近似 $f(x_1, x_2)$ 。

图 20 所示为一阶泰勒展开估计原函数，二元一次函数对应解析式为：

$$\begin{aligned} f(x_1, x_2) &\approx f(-0.1, -0.2) + f_{x_1}(-0.1, -0.2)(x_1 - (-0.1)) + f_{x_2}(-0.1, -0.2)(x_2 - (-0.2)) \\ &= 0.951 + 0.190(x_1 + 0.1) + 0.380(x_2 + 0.2) \end{aligned} \quad (30)$$

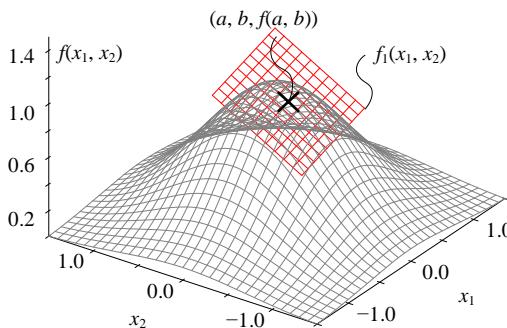


图 20. 用二元一次函数估计二元高斯函数

二次函数

用二次泰勒展开估计 $f(x_1, x_2)$:

$$\begin{aligned} f(x_1, x_2) \approx & \underbrace{f(a, b)}_{\text{Constant}} + \underbrace{f_{x_1}(a, b)(x_1 - a) + f_{x_2}(a, b)(x_2 - b)}_{\text{Plane}} \\ & + \underbrace{\frac{1}{2!} \left[f_{x_1 x_1}(a, b)(x_1 - a)^2 + 2f_{x_1 x_2}(a, b)(x_1 - a)(x_2 - b) + f_{x_2 x_2}(a, b)(x_2 - b)^2 \right]}_{\text{Quadratic}} \end{aligned} \quad (31)$$

相当于“水平面 + 斜面 + 二次曲面”叠加来近似 $f(x_1, x_2)$ 。

图 21 所示为二阶泰勒展开估计原函数，二元二次函数对应解析式为：

$$\begin{aligned} f_2(x_1, x_2) = & 0.951 + 0.190(x_1 + 0.1) + 0.380(x_2 + 0.2) \\ & + \frac{1}{2} \left[-1.864(x_1 + 0.1)^2 + 0.152(x_1 + 0.1)(x_2 + 0.2) - 1.750(x_2 + 0.2)^2 \right] \end{aligned} \quad (32)$$

请大家用本节代码自行展开整理上式。

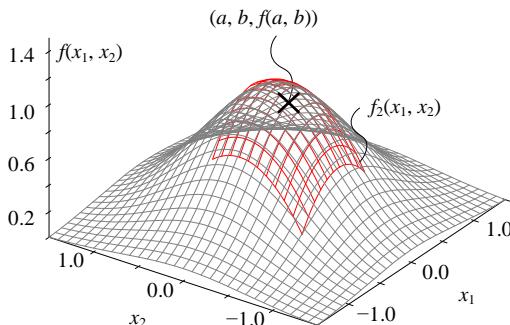
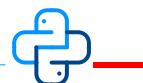


图 21. 用二次函数估计二元高斯函数



Bk3_Ch17_03.py 绘制图 19、图 20、图 21 三幅图。建议大家自己用 Streamlit 把这个代码改成一个 App。

17.5 数值微分：估算一阶导数

并不是所有函数都能得到导数的解析解，很多函数需要用数值方法近似求得导数。数值方法就是“近似”。

三种方法

本节介绍三种一次导数的数值估算方法：向前差分 (forward difference)、向后差分 (backward difference)、中心差分 (central difference)，具体如图 22 所示。

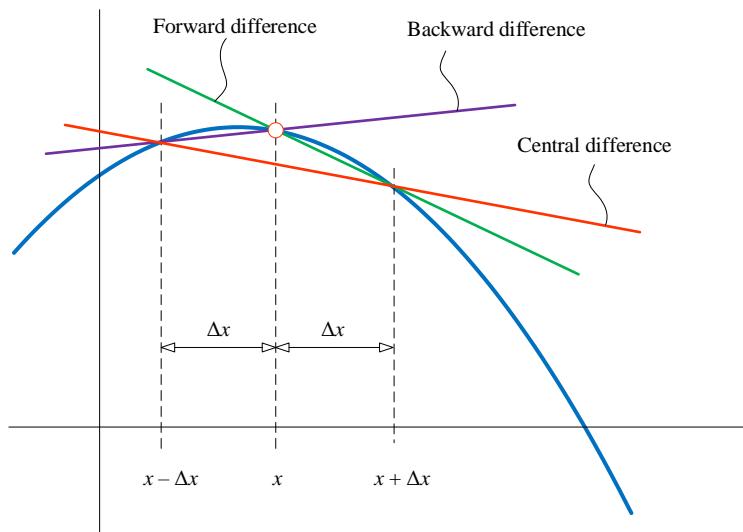


图 22. 三种一次导数的数值估计方法

一阶导数向前差分的具体公式为：

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x} \quad (33)$$

一阶导数向后差分的公式如下：

$$f'(x) \approx \frac{f(x) - f(x - \Delta x)}{\Delta x} \quad (34)$$

一阶导数的中心差分形式如下：

$$f'(x) \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \quad (35)$$

举个例子

给定如下高斯函数：

$$f(x) = \exp(-x^2) \quad (36)$$

我们可以很容易计算得到它的一阶导数函数解析式为：

$$f'(x) = -2x \exp(-x^2) \quad (37)$$

图 23 所示为高斯函数和它的一阶导数图像。同时，我们用三种不同的数值方法在 x 取不同值时估算 (37)。

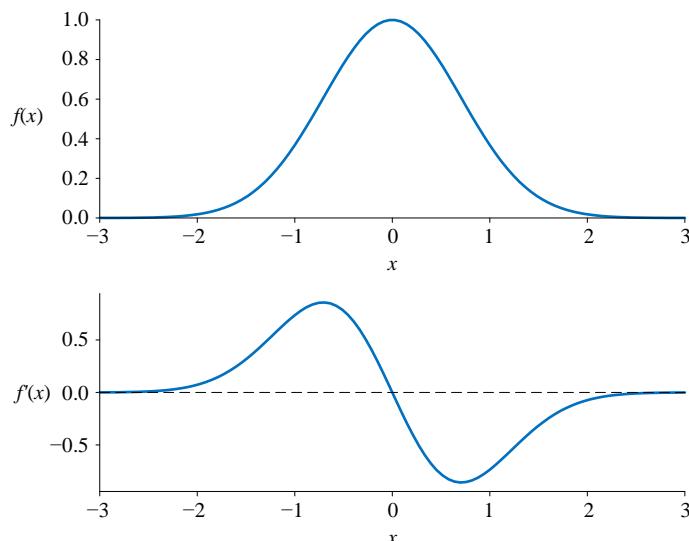


图 23. 高斯函数和它的一阶导数图像

设定 $\Delta x = 0.2$ ，图 24 对比中心差分、向前差分、向后差分结果。图 24 中，中心差分的结果相对好一些。

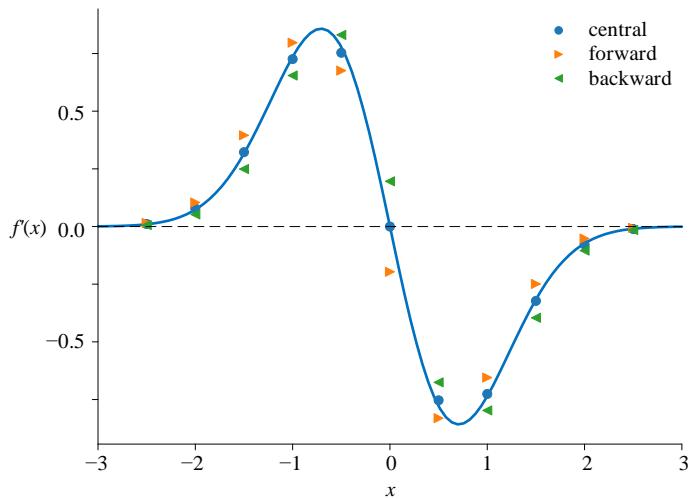
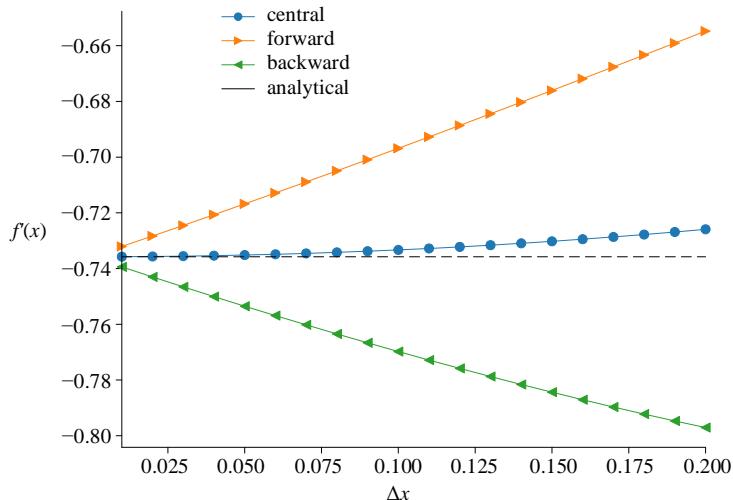
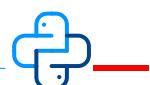
图 24. 对比中心差分、向前差分、向后差分结果, $\Delta x = 0.2$

图 25 所示为 $x = 1$, 步长 Δx 取不同值时, 中心差分、向前差分、向后差分结果对比。很明显当 Δx 减小时, 中心差分更快地收敛于解析解, 具有更高的精度。

图 25. 步长 Δx 不同时, 中心差分、向前差分、向后差分结果

Bk3_Ch17_04.py 完成三种差分运算, 并绘制图 24 和图 25。



本章有三个关键点——微分、泰勒展开、数值微分。

再次强调，导数是函数的变化率，而微分是函数线性近似。从几何视角来看，微分用切线近似非线性函数。

泰勒展开是一系列多项式函数叠加，用来近似某个复杂函数；最为常用的一阶泰勒展开和二阶泰勒展开。

并不是所有的函数都能很容易求得导数，对于求导困难的函数，我们可以采用数值微分的方法。本章介绍了三种不同的方法。

18

Fundamentals of Integral

积分

源自于求面积、体积等数学问题



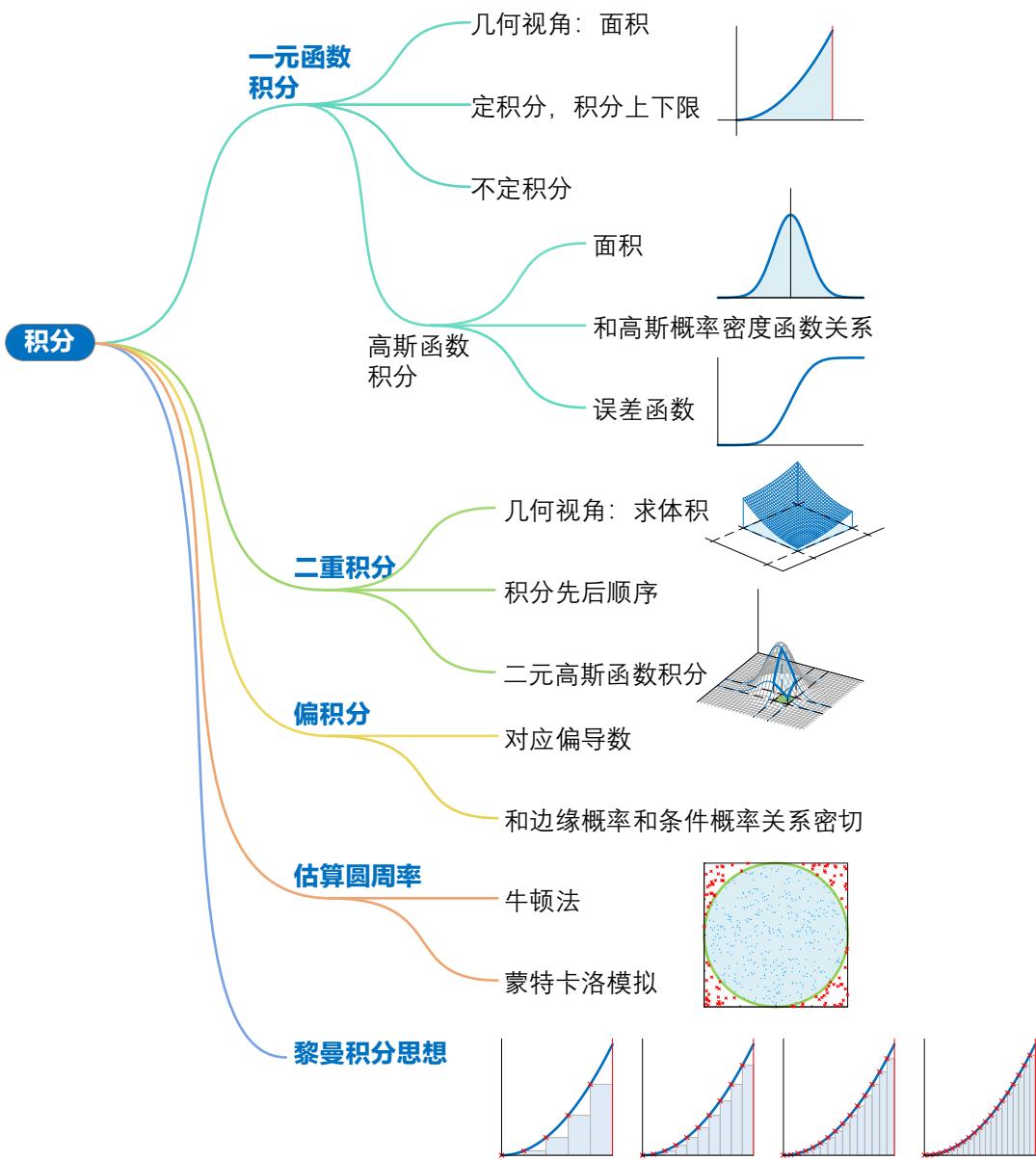
有苦才有甜。

He who hasn't tasted bitter things hasn't earned sweet things.

——戈特弗里德·莱布尼茨 (Gottfried Wilhelm Leibniz) | 德意志数学家、哲学家 | 1646 ~ 1716



- ◀ `numpy.vectorize()` 将自定义函数向量化
- ◀ `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.integrate()` 符号积分
- ◀ `sympy.symbols()` 定义符号变量



18.1 莱布尼茨：既生瑜，何生亮

实际上，人类对积分的探索要远早于微分。古时候，各个文明都在探索不同方法计算不规则形状的长度、面积、体积，人类几何知识则在这个过程中不断进步并且体系化。前文介绍过早期数学家估算圆周率时用内接或外切正多边形近似正圆，其中蕴含的数学思想也是积分的基础。

积分的本来含义就是求和，拉丁语 summa 首字母 s 纵向拉伸，便得到积分符号 \int 。积分符号 \int 的发明者便是莱布尼茨 (Gottfried Wilhelm Leibniz)。



戈特弗里德·威廉·莱布尼茨 (Gottfried Wilhelm Leibniz)
德国哲学家、数学家 | 1646年 ~ 1716年
和牛顿先后独立发明了微积分，创造的微积分符号至今被广泛使用

莱布尼茨是十七世纪少有的通才，这个德国人是律师、哲学家、工程师，更是优秀的数学家。

牛顿和莱布尼茨各自独立发明微积分，两者就微积分发明权争执了很长时间。牛顿在十七世纪的学术界呼风唤雨，是学术天空中最耀眼的一颗星辰，莱布尼茨和其他学者的光芒则显得暗淡很多。很可能是因为这个原因，英国皇家学会公开判定“牛顿是微积分的第一发明人”。

但是，莱布尼茨显得大度很多，他公开表示“在从世界开始到牛顿生活的时代的全部数学中，牛顿的工作超过了一半。”

不管谁发明了微积分，莱布尼茨的微积分数学符号被后世广泛采用，这也算是一种胜利。

18.2 从小车匀加速直线运动说起

回顾本书第 15 章讲解导数时给出的匀加速直线运动的例子。

如图 1 所示，匀加速直线运动中，加速度 $a(t)$ 是常数函数，图像是水平线 $a(t) = 1$ (忽略单位)。时间 $0 \sim t$ ，水平线和横轴围成的面积是个矩形。容易求解矩形面积，这个面积对应速度函数 $v(t) = t$ 。

显然， $v(t)$ 是一个一次函数，图像为一条通过原点的斜线。时间范围为 $0 \sim t$ ， $v(t)$ 斜线和横轴围成的面积是个三角形。三角形的面积对应距离函数 $s(t) = t^2/2$ 。而 $s(t)$ 是个二次函数，图像为抛物线。

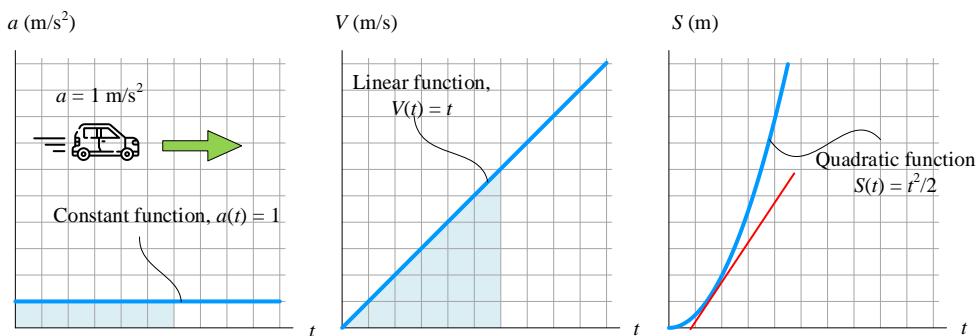
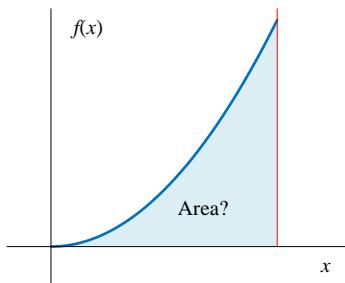


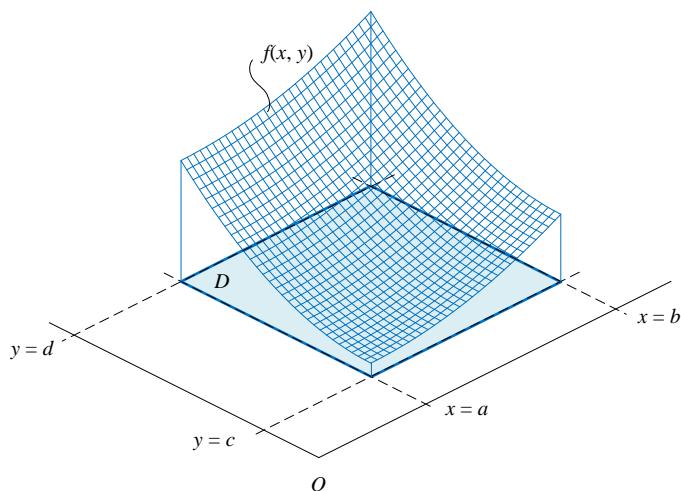
图 1. 匀加速直线运动：加速度、速度、距离

求解矩形面积和三角形面积显然难不倒我们。但是，当我们把问题的难度稍微提高。比如，将变量从 t 换成 x ，把距离函数写成 $f(x) = x^2/2$ 。

如图 2 所示， x 在一定区间内，二元函数 $f(x)$ 曲线和横轴构成这块形状不规则图形，要精确计算它的面积，怎么办呢？

图 2. $f(x)$ 在固定区间积分求面积

还有，如何计算图 3 中 $f(x, y)$ 曲面在 D 区域内和水平面围成的几何形体体积？

图 3. $f(x, y)$ 在区域 D 进行二重积分求体积

解决这些问题需要借助本章要讲解的重要数学工具——积分。

18.3 一元函数积分

导数、偏导、积分、二重积分

本书第 15 章聊过，导数关注变化率。对于一元函数，图 4 所示，几何角度来看，导数相当于曲线切线斜率。而对于二元函数来说，偏导数是二元函数曲面某点在特定方向的切线斜率。微分，则是线性近似。

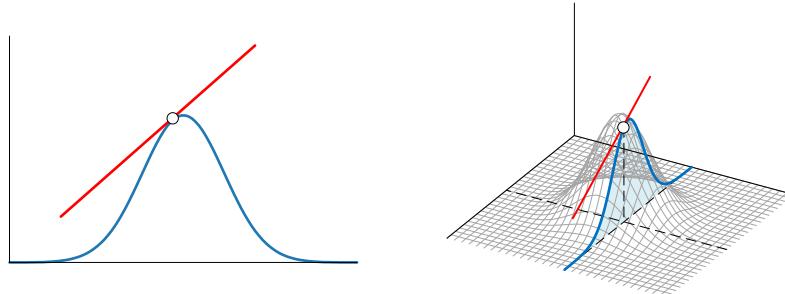


图 4. 几何视角看导数、偏导数、微分等数学工具

积分是微分的逆运算，积分关注变化累积，比如曲线面积、曲面体积，如图 5 所示。导数、微分、积分这个数学工具合称微积分，微积分是定量研究变化过程的重要数学工具。

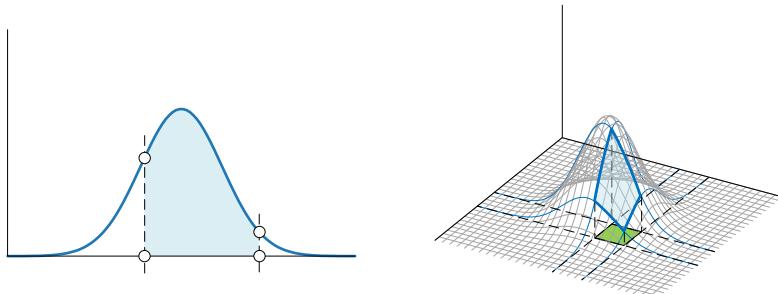


图 5. 几何视角看积分、多重积分等数学工具

一元函数积分

一元函数 $f(x)$ 自变量 x 在区间 $[a, b]$ 上定积分运算记做：

$$\int_a^b f(x) dx \quad (1)$$

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

其中， a 叫积分下限 (lower bound)， b 叫积分上限 (upper bound)。

⚠ 注意，积分有正负之分，也就是说面积值有正负。

如图 6 所示，对于一元函数，曲线在横轴之上包围的面积为正，曲线在横轴之下包围的面积为负。

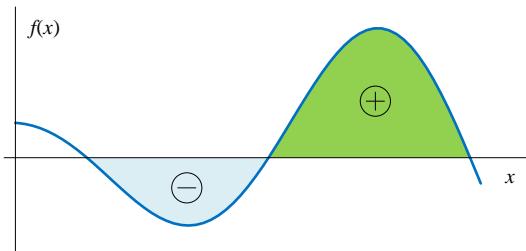


图 6. 积分有正负之分

类比的话，一次函数积分类似如下累加：

$$\sum_{i=1}^n a_i = a_1 + a_2 + \cdots + a_{n-1} + a_n \quad (2)$$

举个例子

图 7 (a) 所示为如下一次函数的定积分：

$$\int_0^1 \left(x^2 + \frac{1}{2} \right) dx = \left(\frac{1}{3}x^3 + \frac{1}{2}x \right)_0^1 = \frac{5}{6} \approx 0.8333 \quad (3)$$

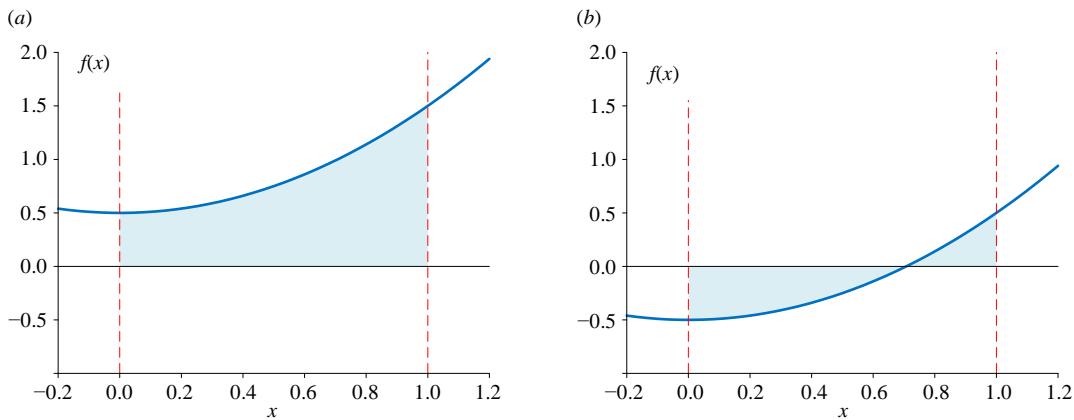


图 7. 两个函数的定积分

图 7 (b) 所示为如下函数的定积分：

$$\int_0^1 \left(x^2 - \frac{1}{2} \right) dx = \left(\frac{1}{3}x^3 - \frac{1}{2}x \right)_0^1 = -\frac{1}{6} \approx -0.1667 \quad (4)$$

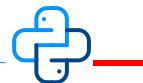
图 7 (a) 函数图像向下移动一个单位便得到图 7 (b)。因此，(3) 和 (4) 两个定积分存在以下关系：

$$\int_0^1 \left(x^2 + \frac{1}{2} \right) dx - \int_0^1 \left(x^2 - \frac{1}{2} \right) dx = 1 \quad (5)$$

若定积分存在，定积分则是一个具体的数值；而不定积分结果一般是一个函数表达式。

表 1. 积分的英文表达

数学表达	英文表达
$\int_1^3 x^3 dx$	The integral from one to three of x cubed dx
$\int f(x)dx$	The integral of f of x dx The indefinite integral of f with respect to x
$\int_a^b f(x)dx$	The integral from a to b of f of x dx



Bk3_Ch18_01.py 计算定积分并绘制图 7 两幅子图。

18.4 高斯函数积分

高斯函数积分有自己的名字——高斯积分 (Gaussian integral)。

前文提过，高斯函数和高斯分布 (Gaussian distribution) 联系紧密；因此，高斯积分在概率统计中也有扮演重要角色。

坐标变换方法可以求解高斯积分；但是，本书不会介绍如何推导高斯积分，这部分内容留给感兴趣的读者自己探索。



本章想从几何视角和大家聊聊有关高斯积分的一些重要性质，这部分内容和本系列丛书后续内容高斯分布有着密切联系。

对于一元高斯函数积分，请大家首先留意如下积分结果：

$$\int_{-\infty}^{\infty} \exp(-x^2) dx = \sqrt{\pi} \quad (6)$$

如图 8 所示，高斯函数 $f(x) = \exp(-x^2)$ 和整个 x 轴围成的面积为 $\sqrt{\pi}$ 。再次强调，图 8 中高斯函数趋向正、负无穷时，函数值无限接近 0，但是达不到 0。

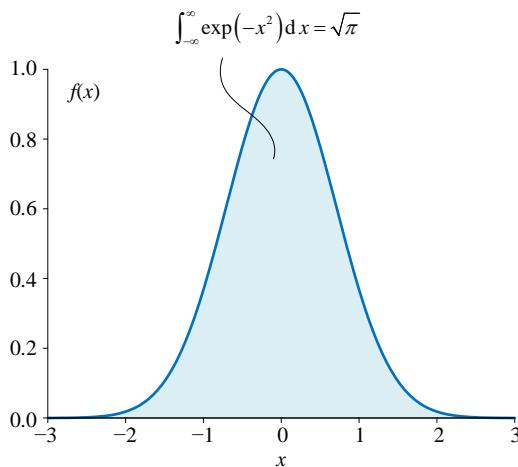


图 8. 高斯函数正负无穷积分面积

定积分

再举个定积分的例子，给定如下积分上下限，计算高斯函数定积分：

$$\int_{-0.5}^1 \exp(-x^2) dx \approx 1.208 \quad (7)$$

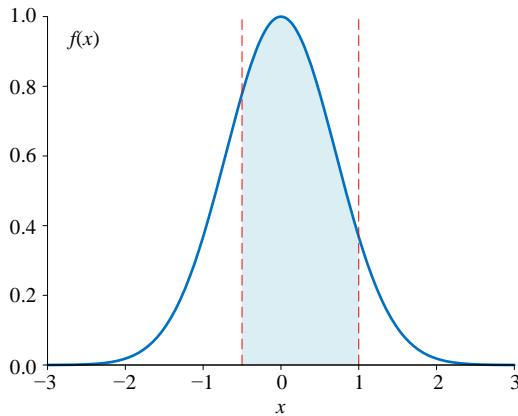


图 9. 高斯函数定积分

前文提过不定积分存在的话，函数的不定积分结果是函数。比如，对高斯函数从 $-\infty$ 积分到 x 得到：

$$F(x) = \int_{-\infty}^x \exp(-t^2) dt \quad (8)$$

图 10 中蓝色曲线所示为上述高斯积分 $F(x)$ 随 x 变化。

⚠ 注意，高斯函数积分没有解析解。

这样 (7) 可以用 $F(x)$ 计算如下定积分：

$$\int_{-0.5}^1 \exp(-x^2) dx = F(1) - F(-0.5) \approx 1.633 - 0.425 = 1.208 \quad (9)$$

图 10 所示为利用 $F(x)$ 计算 (7) 定积分原理。

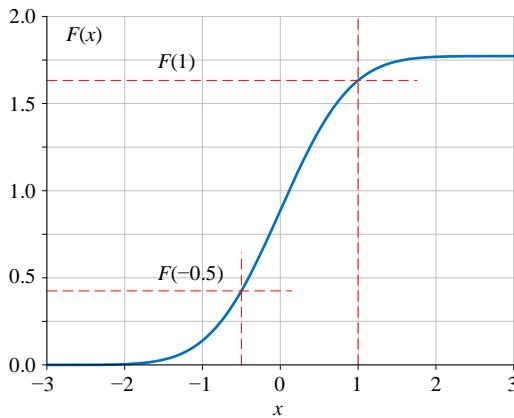


图 10. 用 $F(x)$ 计算高斯定积分

另外注意下面这个积分公式：

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = 1 \quad (10)$$

看到这个公式，大家是否联想到一元标准正态分布概率密度函数 PDF。分母上为 $\sqrt{2\pi}$ 作用是归一化，也就是让函数和整个横轴围成的面积为 1。这解释了为什么高斯分布概率密度函数的分母上有 $\sqrt{2\pi}$ 这个缩放系数。



Bk3_Ch18_02.py 计算高斯函数积分并且绘制图 9 和图 10。

18.5 误差函数：S型函数的一种

通过调取代码结果，大家可能已经发现高斯函数积分结果是用 erf() 函数来表达的。

erf(x) 函数就是鼎鼎有名的误差函数 (error function)：

$$\operatorname{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^x \exp(-t^2) dt = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt \quad (11)$$

误差函数是利用高斯积分定义的，它没有一般意义上的解析式。

前文提过，误差函数是 S 型函数的一种。误差函数在概率统计、数据科学、机器学习中应用广泛。

一般情况，误差函数自变量 x 的取值为正值，但是为了计算方便，erf() 的输入也可以是负值。 x 为负值时，下式成立：

$$\operatorname{erf}(x) = -\operatorname{erf}(-x) \quad (12)$$

图 11 所示为误差函数图像。

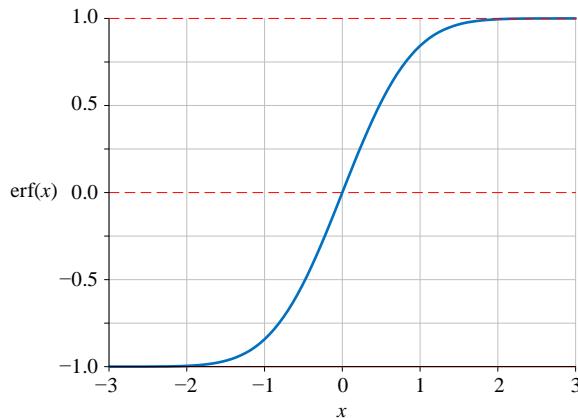
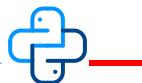


图 11. 误差函数

高斯函数从 $-\infty$ 积分到 x 对应的高斯积分和误差函数的关系为：

$$F(x) = \int_{-\infty}^x \exp(-t^2) dt = \frac{\sqrt{\pi}}{2} \operatorname{erf}(x) + \frac{\sqrt{\pi}}{2} \quad (13)$$

通过上述公式可以看出误差函数先是通过纵轴缩放，再沿纵轴平移得到高斯积分。



Bk3_Ch18_03.py 绘制图 11。注意，`sympy.erf()` 可以接受负值。

18.6 二重积分：类似二重求和

先对 x 积分

给定积分区域 $D = \{(x, y) \mid a < x < b, c < y < d\}$, $f(x, y)$ 二重积分记做：

$$\int_c^d \int_a^b f(x, y) dx dy \quad (14)$$

请注意上式二重积分的先后次序，先对 x 积分，再对 y 积分。也就是说，内部这一层 $\int_{x=a}^{x=b} f(x, y) dx$ 先消去 x ，变成有关 y 的一元函数；然后再对 y 积分：

$$\int_c^d \int_a^b f(x, y) dx dy = \int_{y=c}^{y=d} \underbrace{\int_{x=a}^{x=b} f(x, y) dx}_{\text{A function of } y} dy \quad (15)$$

Eliminate x

$\int_{x=a}^{x=b} f(x, y) dx$ 相当于降维，也就是“压缩”。

从几何角度，如图 12 所示，当 $y = c$ 时， $\int_{x=a}^{x=b} f(x, y=c) dx$ 结果为图中暖色阴影区域面积，也就是压缩为一个值。

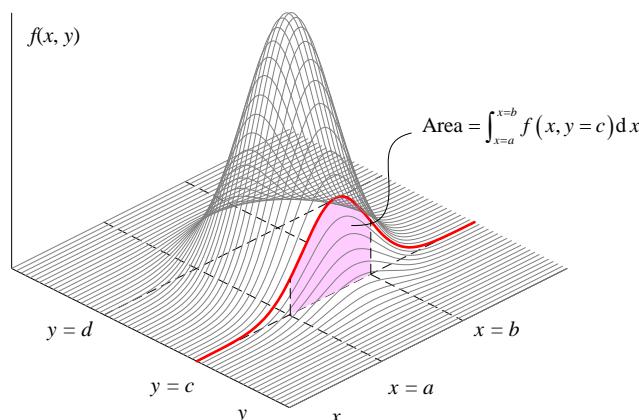


图 12. $f(x, y)$ 先对 x 积分，相当于沿 x 轴压缩

先对 y 积分

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

如果调换积分顺序，先对 y 积分， $\int_{y=c}^{y=d} f(x, y) dy$ 相当于消去 y ，得到有关 x 的一元函数；然后对 x 积分：

$$\int_a^b \int_c^d f(x, y) dy dx = \underbrace{\int_{x=a}^{x=b} \overbrace{\int_{y=c}^{y=d} f(x, y) dy}^{\text{Eliminate } y} dx}_{\text{A function of } x} \quad (16)$$

如图 13 所示，当 $x = a$ 时， $\int_{y=c}^{y=d} f(x=a, y) dy$ 结果为图中冷色阴影区域面积，即压缩为一个值。

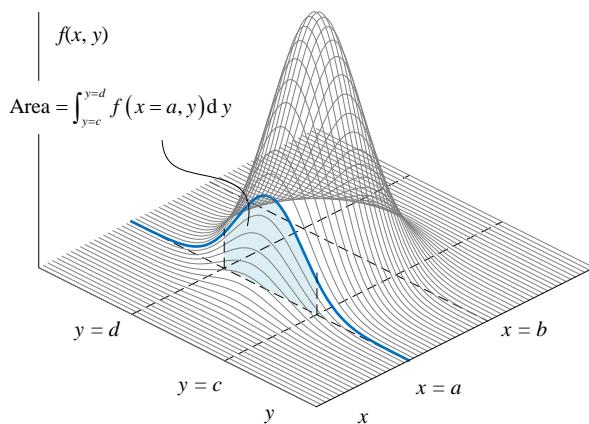


图 13. $f(x, y)$ 先对 y 积分，相当于沿 y 轴压缩

调换积分顺序

特别地，如果 $f(x, y)$ 在矩形局域 $D = \{(x, y) \mid a < x < b, c < y < d\}$ 连续，如下二重积分先后顺序可以调换：

$$\int_c^d \int_a^b f(x, y) dx dy = \int_a^b \int_c^d f(x, y) dy dx \quad (17)$$

白话说，如果积分的区域相对于坐标系“方方正正”，积分顺序可以调换。

⚠ 千万注意，二重积分、多重积分中，积分先后顺序不能随意调换，上述例子仅仅是个特例而已。有关多重积分顺序调换内容，本书不展开讲解。

类比的话，二重积分类似如下二重累加：

$$\sum_{i=1}^n \sum_{j=1}^m a_{i,j} \quad (18)$$

二元高斯函数

举个例子，给定二元高斯函数 $f(x, y)$,

$$f(x, y) = \exp(-x^2 - y^2) \quad (19)$$

$f(x, y)$ 的二重不定积分 $F(x, y)$ 可以用误差函数表达为：

$$F(x, y) = \int_{-\infty}^y \int_{-\infty}^x \exp(-u^2 - v^2) du dv = \frac{\pi}{4} \operatorname{erf}(x) \operatorname{erf}(y) + \frac{\pi}{4} \operatorname{erf}(x) + \frac{\pi}{4} \operatorname{erf}(y) + \frac{\pi}{4} \quad (20)$$

图 14 所示为 $F(x, y)$ 曲面以及三维等高线。图 15 所示为 $F(x, y)$ 曲面在 xz 平面、 yz 平面投影，可以发现投影得到的曲线形状类似误差函数。

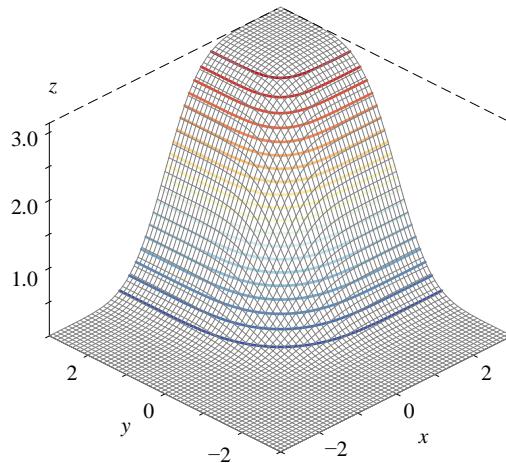


图 14. 二元高斯函数二重不定积分 $F(x, y)$ 曲面

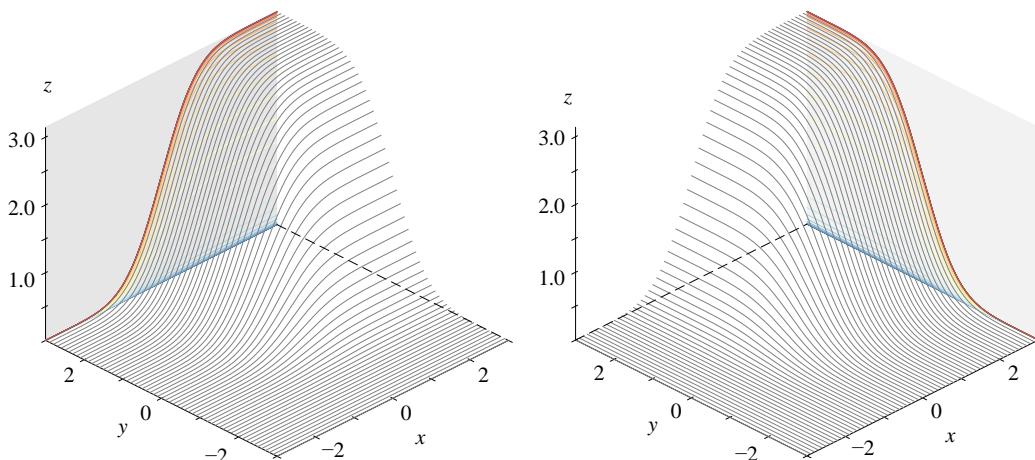
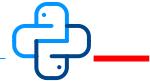


图 15. $F(x, y)$ 曲面在 xz 平面、 yz 平面投影

特别地， $f(x, y)$ 曲面和整个水平面围成的体积为 π ，即，

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(-x^2 - y^2) dx dy = \pi \quad (21)$$



Bk3_Ch18_04.py 完成本节二重积分计算。

18.7 “偏积分”：类似偏求和

本书第 14 章介绍过“偏求和”、偏导数，“偏”字的意思是考虑一个变量，其他变量视为定值。本节自创一个积分概念——“偏积分”，如下两式就是“偏积分”：

$$\begin{aligned} & \int_a^b f(x, y) dx \\ & \int_c^d f(x, y) dy \end{aligned} \quad (22)$$

类比的话，偏积分类似前文介绍的偏求和：

$$\sum_{i=1}^n a_{i,j}, \quad \sum_{j=1}^m a_{i,j} \quad (23)$$

对 y 偏积分

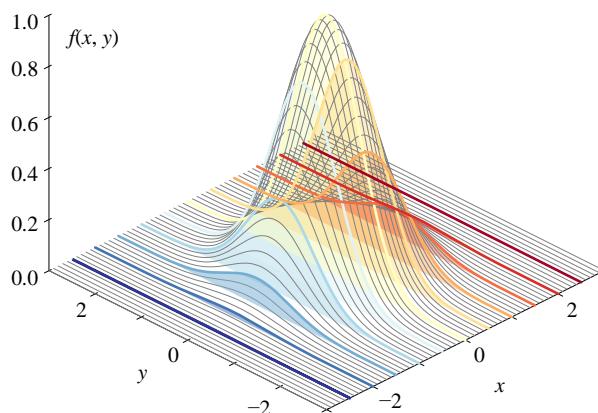
给定二元高斯函数 $f(x, y)$ ：

$$f(x, y) = \exp(-x^2 - y^2) \quad (24)$$

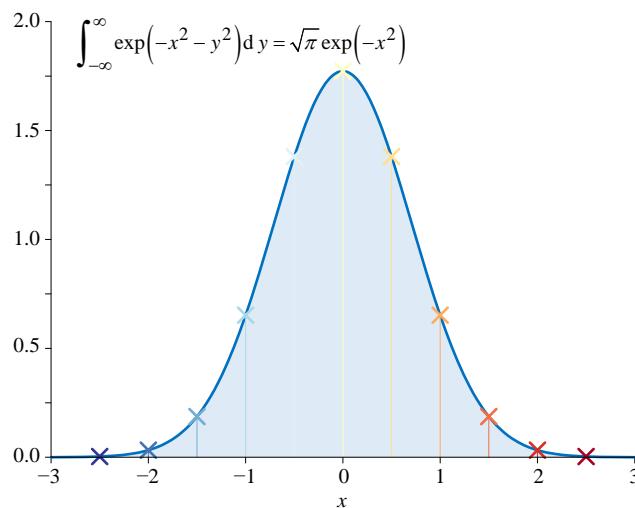
$f(x, y)$ 对于 y 从负无穷到正无穷偏积分，得到结果变成了关于 x 的高斯函数：

$$\int_{-\infty}^{\infty} \exp(-x^2 - y^2) dy = \sqrt{\pi} \exp(-x^2) \quad (25)$$

从几何角度来看，如图 16 所示， $f(x, y)$ 对 y 从负无穷到正无穷偏积分，相当于 x 取某个值时，比如 $x = c$ ，对二元高斯函数 $f(x, y)$ 曲线做个剖面，剖面线（图 16 彩色曲线）和其水平面投影构成面积（图 16 彩色阴影区域）就是偏积分结果。

图 16. 二元高斯函数 $f(x, y)$ 对 y 偏积分

正如图 17 所示，(25) 的偏积分结果是有关 x 的一元高斯函数。

图 17. 对 y 偏积分的结果是关于 x 的高斯函数

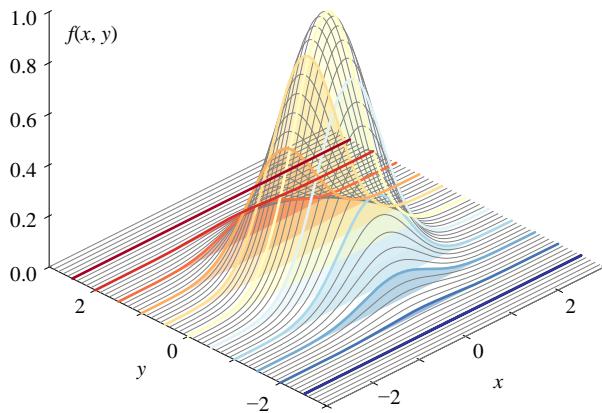
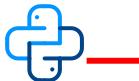
对 x 偏积分

类似地，二元高斯函数 $f(x, y)$ 对 x 从负无穷到正无穷偏积分，结果为关于 y 的高斯函数：

$$\int_{-\infty}^{\infty} \exp(-x^2 - y^2) dx = \sqrt{\pi} \exp(-y^2) \quad (26)$$

图 18 为上式的几何含义。

⚠ 再次注意，偏积分是我们创造的一个词，对应偏导数。预告一下，“偏积分”这个概念在概率统计中会帮助我们理解连续随机变量的边缘概率和条件概率。

图 18. 二元高斯函数 $f(x, y)$ 对 x 偏积分

Bk3_Ch18_05.py 计算高斯二元函数偏积分。

18.8 估算圆周率：牛顿法

本书前文介绍过估算圆周率的不同方法。随着数学工具的不断升级，有了微积分这个强有力的工具，我们可以介绍牛顿估算圆周率的方法。

图 19 中给出的函数 $f(x)$ 是某个圆形的上半圆。这个圆的中心位于 $(0.5, 0)$ ，半径为 0.5。上半圆函数 $f(x)$ 的解析式为：

$$f(x) = \sqrt{x - x^2} \quad (27)$$

在这个半圆中，划定图 19 左图所示的阴影区域，它对应的圆心角度为 60° 。

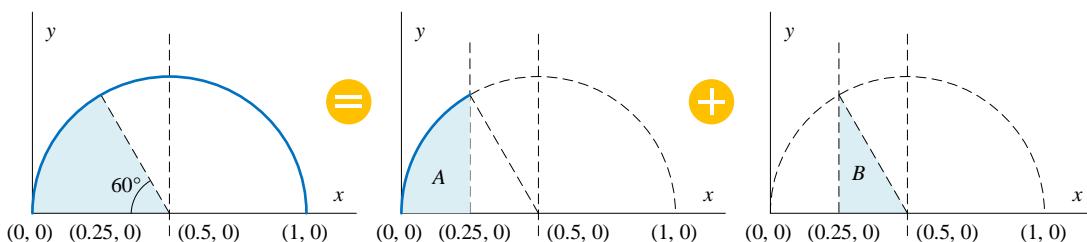


图 19. 面积关系

整个阴影区域的面积为 $\pi/24$ 。而这个区域的面积可以分成 A 和 B 两部分。

B 部分为直角三角形，面积很容易求得，具体值为：

$$B = \frac{\sqrt{3}}{32} \quad (28)$$

因此， A 的面积为扇形面积减去 B 的面积：

$$A = \frac{\pi}{24} - \frac{\sqrt{3}}{32} \quad (29)$$

整理 (29)，得到圆周率和 A 的关系：

$$\pi = 24 \times \left(\frac{\sqrt{3}}{32} + A \right) \quad (30)$$

而面积 A 可以通过如下定积分得到：

$$A = \int_0^{\frac{1}{4}} \sqrt{x - x^2} dx = \int_0^{\frac{1}{4}} \sqrt{x} \sqrt{1-x} dx \quad (31)$$

其中 $\sqrt{1-x}$ 可以用泰勒展开写成：

$$\sqrt{1-x} = 1 - \frac{1}{2}x - \frac{1}{8}x^2 - \frac{1}{16}x^3 - \frac{5}{128}x^4 \dots \quad (32)$$

(32) 代入积分式 (31)，得到：

$$\begin{aligned} A &= \int_0^{\frac{1}{4}} \sqrt{x - x^2} dx \\ &= \int_0^{\frac{1}{4}} x^{\frac{1}{2}} \left(1 - \frac{1}{2}x - \frac{1}{8}x^2 - \frac{1}{16}x^3 - \frac{5}{128}x^4 \dots \right) dx \\ &= \int_0^{\frac{1}{4}} \left(x^{\frac{1}{2}} - \frac{1}{2}x^{\frac{3}{2}} - \frac{1}{8}x^{\frac{5}{2}} - \frac{1}{16}x^{\frac{7}{2}} - \frac{5}{128}x^{\frac{9}{2}} \dots \right) dx \\ &= \left[\frac{2}{3}x^{\frac{3}{2}} - \frac{1}{5}x^{\frac{5}{2}} - \frac{1}{28}x^{\frac{7}{2}} - \frac{1}{72}x^{\frac{9}{2}} - \frac{5}{704}x^{\frac{11}{2}} \dots \right]_{x=0}^{x=\frac{1}{4}} \\ &= \frac{2}{3 \times 2^3} - \frac{1}{5 \times 2^5} - \frac{1}{28 \times 2^7} - \frac{1}{72 \times 2^9} - \frac{5}{704 \times 2^{11}} \dots \end{aligned} \quad (33)$$

这样 A 可以写成级数求和：

$$A = -\sum_{n=0}^{\infty} \frac{(2n)!}{2^{4n+2} (n!)^2 (2n-1)(2n+3)} \quad (34)$$

于是圆周率可以通过下式近似得到：

$$\pi = 24 \times \left(\frac{\sqrt{3}}{32} - \sum_{n=0}^{\infty} \frac{(2n)!}{2^{4n+2} (n!)^2 (2n-1)(2n+3)} \right) \quad (35)$$

图 20 所示为圆周率估算结果随 n 增加而不断收敛。观察曲线，可以发现这个估算过程收敛的速度很快。以上就是牛顿估算圆周率的方法。

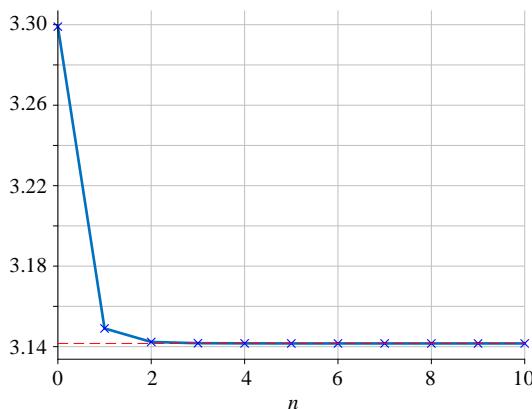


图 20. 牛顿方法估算圆周率



Bk3_Ch18_06.py 绘制图 22。



本章前文黎曼积分的思想——通过无限逼近来确定积分值。利用这一思路，我们也可以估算圆周率。如图 21 所示，我们可以用不断细分的正方形估算单位圆的面积，从而估算圆周率。而这一思路实际上就是蒙特卡洛模拟估算圆周率的内核。

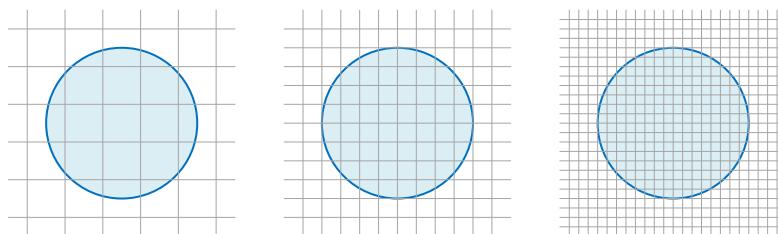


图 21. 用不断细分的正方形估算单位圆面积

蒙特卡洛模拟 (Monte Carlo simulation) 在大数据分析和机器学习中占据重要的位置。蒙特卡洛模拟以摩纳哥的赌城蒙特卡罗命名，是一种使用随机数并以概率理论为指导的数值计算方法。

下面，简单介绍利用如何用蒙特卡洛模拟估算圆周率 π 。

在如图 21 所示单位圆 ($r = 1$) 的周围，构造一个以圆心为中心、以圆直径为边长的外切正方形。圆形面积 A_{circle} 和正方形面积 A_{square} 容易求得：

$$\begin{cases} A_{\text{circle}} = \pi r^2 = \pi \\ A_{\text{square}} = (2r)^2 = 4 \end{cases} \quad (36)$$

进而求得圆周率 π 和两个面积的比例关系：

$$\pi = 4 \times \frac{A_{\text{circle}}}{A_{\text{square}}} \quad (37)$$

然后，在这个正方形区域内产生满足均匀随机分布的 n 个数据点。生活中均匀随机分布无处不在。大家可以想象一下，一段时间没有人打理的房间内，落满灰尘。不考虑房间内特殊位置（窗口、暖气口等）的气流影响，灰尘的分布就类似“均匀随机分布”。

统计落入圆内的数据点个数 m 与总数据点总数 n 的比值，这个比值即为圆面积和正方形面积之比近似值。带入 (37) 可得：

$$\pi \approx 4 \times \frac{m}{n} \quad (38)$$

图 22 所示为四个蒙特卡洛模拟实验，随机点总数 n 分别为 100、500、1000 和 5000。可以发现随着 n 增大，估算得到的圆周率 π 不断接近真实值。

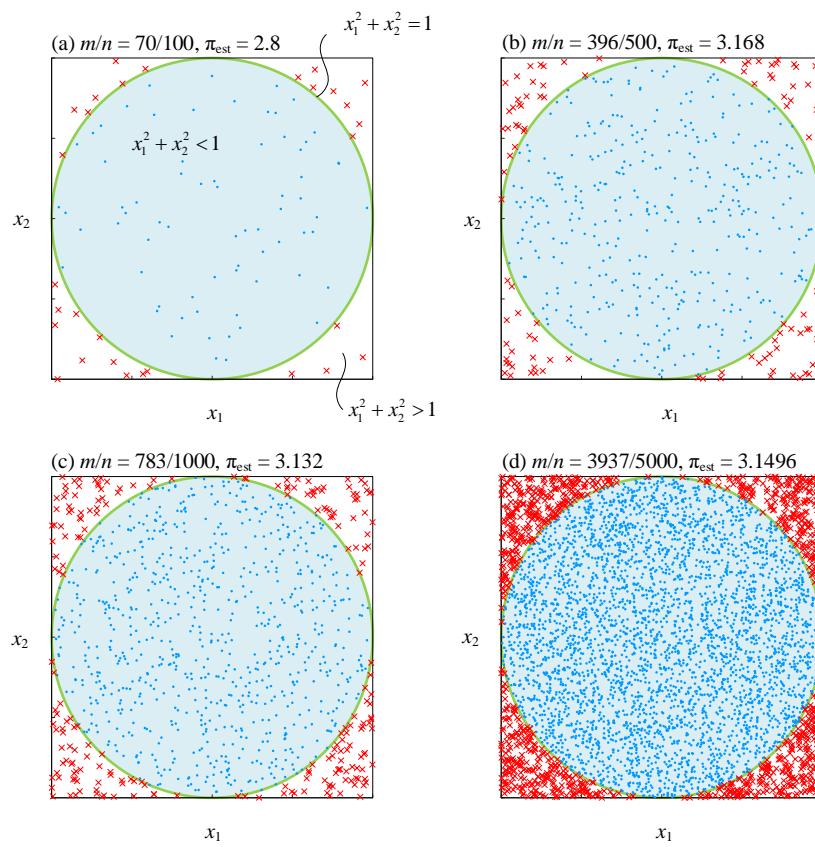


图 22. 蒙特卡洛模拟方法估算圆周率

这种估算圆周率的方法思想来源于在 18 世纪提出的布丰投针问题 (Buffon's needle problem)。实际上，布丰投针实验要比这里介绍的蒙特卡洛模拟方法更为复杂。本系列丛书将在《概率统计》一本书中和大家探讨布丰投针这一经典实验以及如何用 Python 编写代码实现模拟。

18.9 数值积分：黎曼求积

有些函数看着不复杂，竟然也没有积分解析解，比如高斯函数 $f(x) = \exp(-x^2)$ 。因为高斯函数积分很常用，人们还创造出误差函数。对于没有解析解的积分，我们通常使用数值积分方法。本节讨论如何用数值方法估算积分。

将平面图形切成细长条

德国数学家黎曼 (Bernhard Riemann, 1826 ~ 1866) 提出了一个求积解决方案——将不规则图形切成细长条。然后这些细长条近似看成一个个矩形，计算出它们的面积。这些矩形面积求和，可以用来近似不规则形状的面积。

狭长长方形越细，也就是图 23 中 Δx 越小，长方形越贴合区域形状，就越能精确估算面积。特别地，当细长条的宽度 Δx 趋近于 0 时，得到的面积的极限值就是不规则形状的面积。

看到图 23 这幅图，大家有没有想到本书第 3 章介绍圆周率估算时，刘徽说的“割之弥细，所失弥少，割之又割，以至于不可割，则与圆周合体而无所失矣。”两者思想如出一辙。

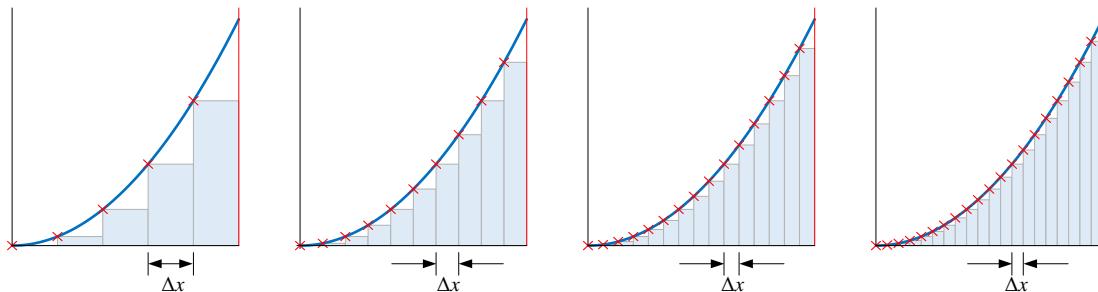


图 23. 细长条切得越细，面积估算越精确

将立体图形切成细高立方体

也用黎曼求积思路计算体积。如图 24 所示，我们可以用一个个细高立方体体积之和来近似估算几何体的体积。

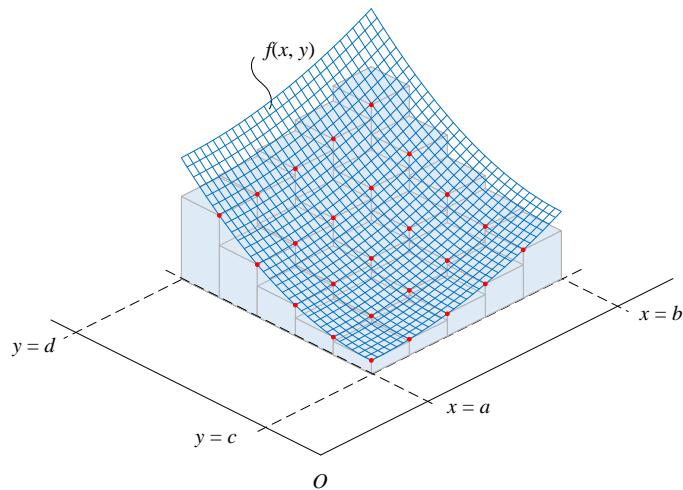


图 24. 将不规则几何体分割成细高立方体

如图 25 所示，随着细长立方体不断变小，这些立方体的体积之和不断接近不规则几何体的体积。

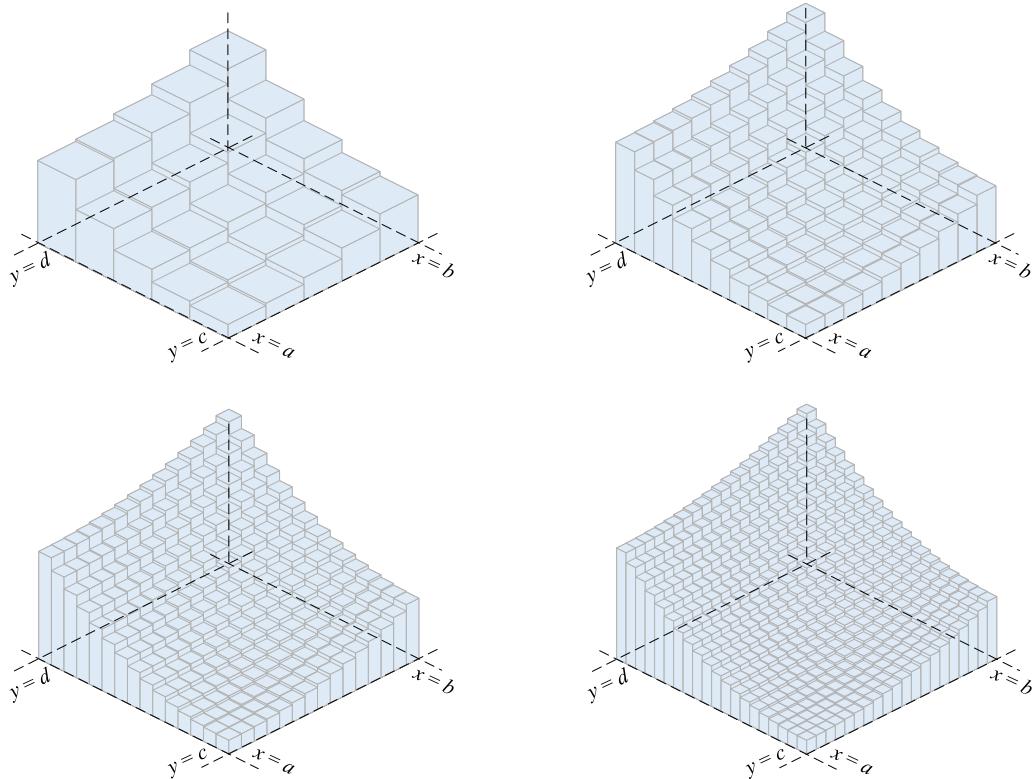


图 25. 随着细长立方体不断变小，立方体体积之和不断接近不规则形体的真实体积

基本数值积分方法

如图 26 (a) 所示，在 $[a, b]$ 区间内，为估算函数在区间内和 x 轴形成的面积，用左侧 a 点的函数值 $f(a)$ 进行积分估值运算：

$$\int_a^b f(x) dx \approx (b-a)f(a) \quad (39)$$

这种方法叫做向前差分，也叫 left Riemann sum。这实际上就是用矩阵面积估算函数积分。

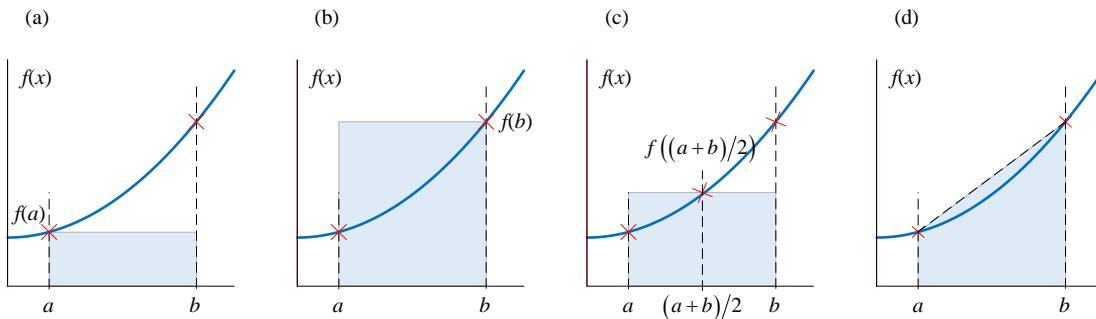


图 26. 四种不同方法

如图 26 (b) 所示，用 $[a, b]$ 区间右侧 b 点函数值 $f(b)$ 进行积分估值运算叫做向后差分，也叫 right Riemann sum：

$$\int_a^b f(x) dx \approx (b-a)f(b) \quad (40)$$

如图 26 (c) 所示，用 $[a, b]$ 区间中间点 $(a+b)/2$ 函数值 $f((a+b)/2)$ 作为矩形高度来估值叫做中值差分，也叫 middle Riemann sum：

$$\int_a^b f(x) dx \approx (b-a)f\left(\frac{a+b}{2}\right) \quad (41)$$

图 26 中前三种都是用矩形面积估算积分。

图 26 (d) 给出的是所谓梯形法，这种方法用 $f(a)$ 和 $f(b)$ 的平均值：

$$\int_a^b f(x) dx \approx (b-a)\left(\frac{f(a)+f(b)}{2}\right) \quad (42)$$

如果数值积分采用固定步长 Δx 。把 $[a, b]$ 区间分成 n 段， Δx 为：

$$\Delta x = \frac{b-a}{n} \quad (43)$$

当然，我们也可以采用可变步长，这不是本节要介绍的内容。

实践中，我们还会用到其他的数值积分方法。它们的差别一般在于两点之间的插值方法，也就是用什么样的简单函数尽可能逼近原函数。比如，图 26 前三种方法采用的是水平线（常数函数），只不过水平线的高度不同而已。图 26 中第四种方法采用两点之间斜线，即一次函数。再比如，辛普森法（Simpson method）用抛物线插值，牛顿-柯蒂斯法（Newton-Cotes method）采用的是 Lagrange 插值。

代码实现

本节仅介绍如何用代码实现图 26 中前三种数值积分方法。

图 27、图 28、图 29 三幅图对比步长 Δx 分别取 0.2、0.1、0.05 时三种数值积分法结果。图 30 所示为随分段数 n 增大三种数值积分结果不断收敛过程。容易发现，middle Riemann sum 更快地逼近真实值，它的精度显然更高。感兴趣的读者，可以自行了解数值积分中代数精度和误差等概念。

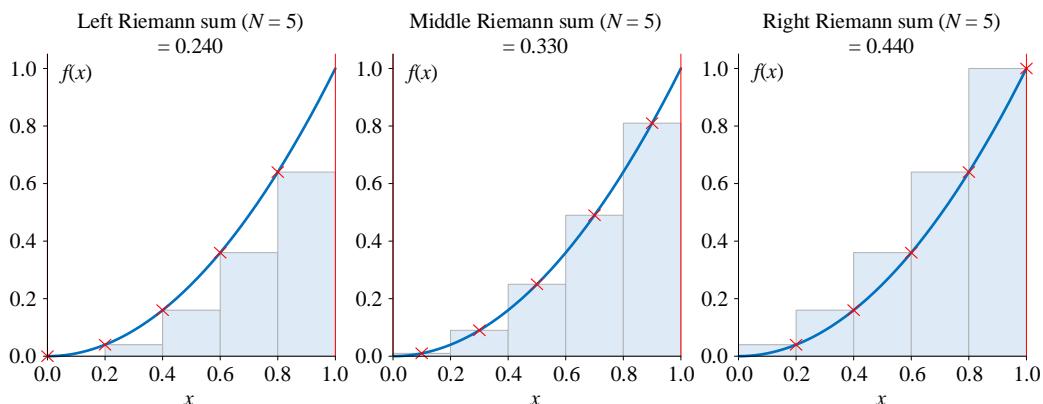


图 27. 步长 $\Delta x = 0.2$

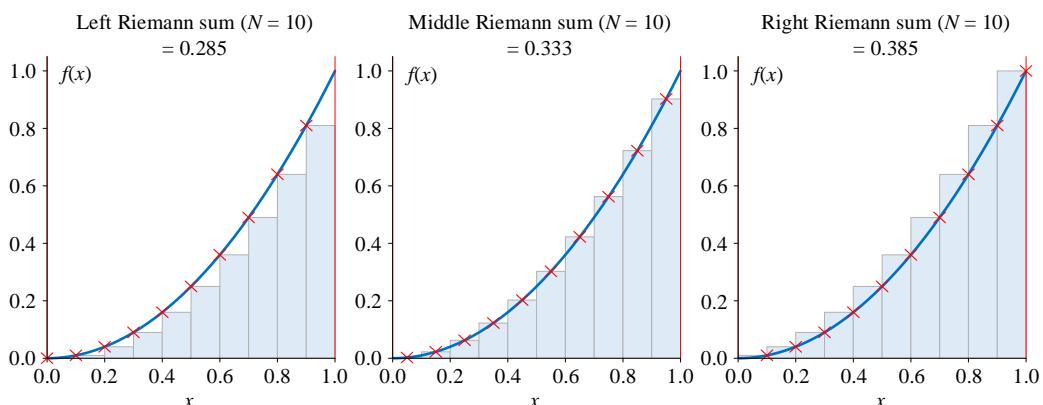
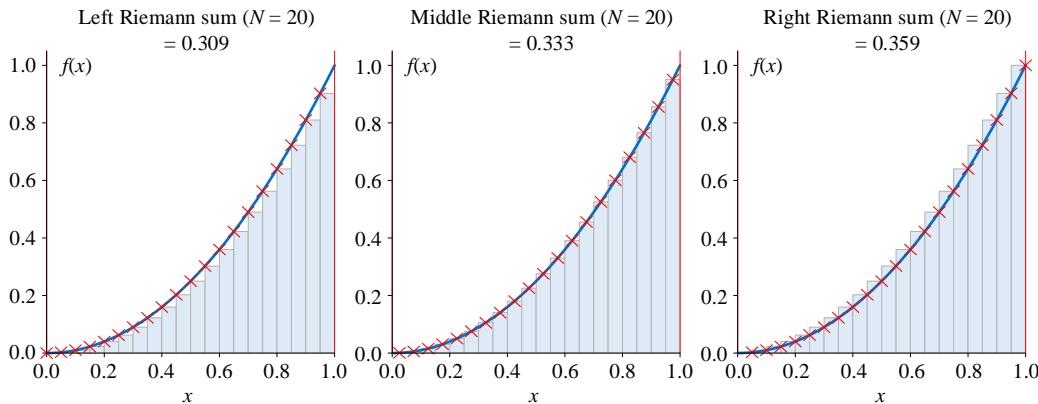
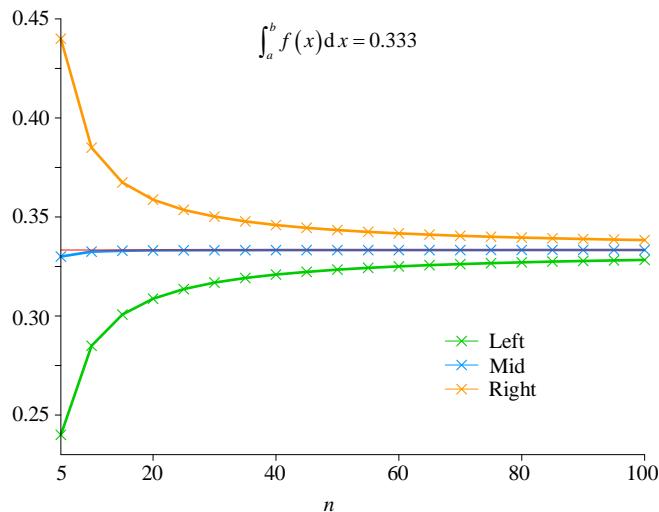


图 28. 步长 $\Delta x = 0.1$

图 29. 步长 $\Delta x = 0.05$ 图 30. 三种数值积分随分段数 n 变化

Bk3_Ch18_07.py 绘制图 27、图 28、图 29。请大家自行编写代码绘制图 30。



在 Bk3_Ch18_07.py 基础上，我们做了一个 App 展示步长 Δx 对数值积分结果影响。请参考 Streamlit_Bk3_Ch18_07.py。

二重数值积分

本节最后介绍如何用代码实现二重数值积分。图 31 所示为某个二元函数曲面，我们要计算曲面和水平面在图中给出的区域内包围的体积。

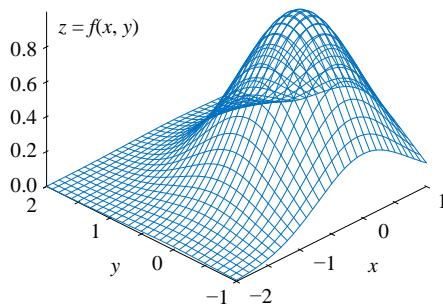


图 31. 二元函数曲面

图 32 所示为用数值积分方法，不断减小在 x 和 y 方向的步长，从而提高二重积分估算精度。

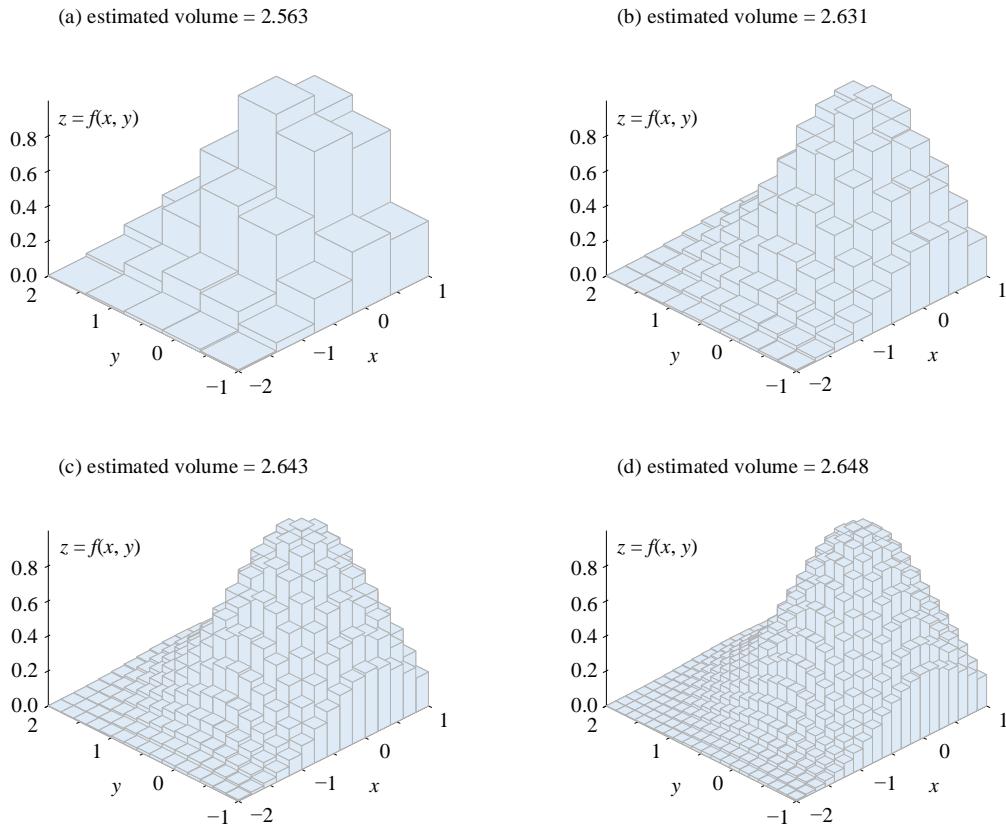
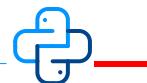


图 32. 不断减小步长提高估算精度



Bk3_Ch18_08.py 绘制图 32。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com



本书有关微积分的内容到此告一段落。机器学习特别是深度学习中，还有两个重要的微积分话题——自动求导、卷积。很遗憾，限于篇幅，本书只能蜻蜓点水般聊一聊在数据科学、机器学习最常用的微积分内容。本书介绍的微积分内容只是整个微积分体系的冰山一角，希望读者日后能够更全面学习提高。

有了微积分这个数学工具，下一章初步探讨优化问题相关内容。

19

Fundamentals of Optimization

优化入门

在一定区域内，寻找山峰、山谷



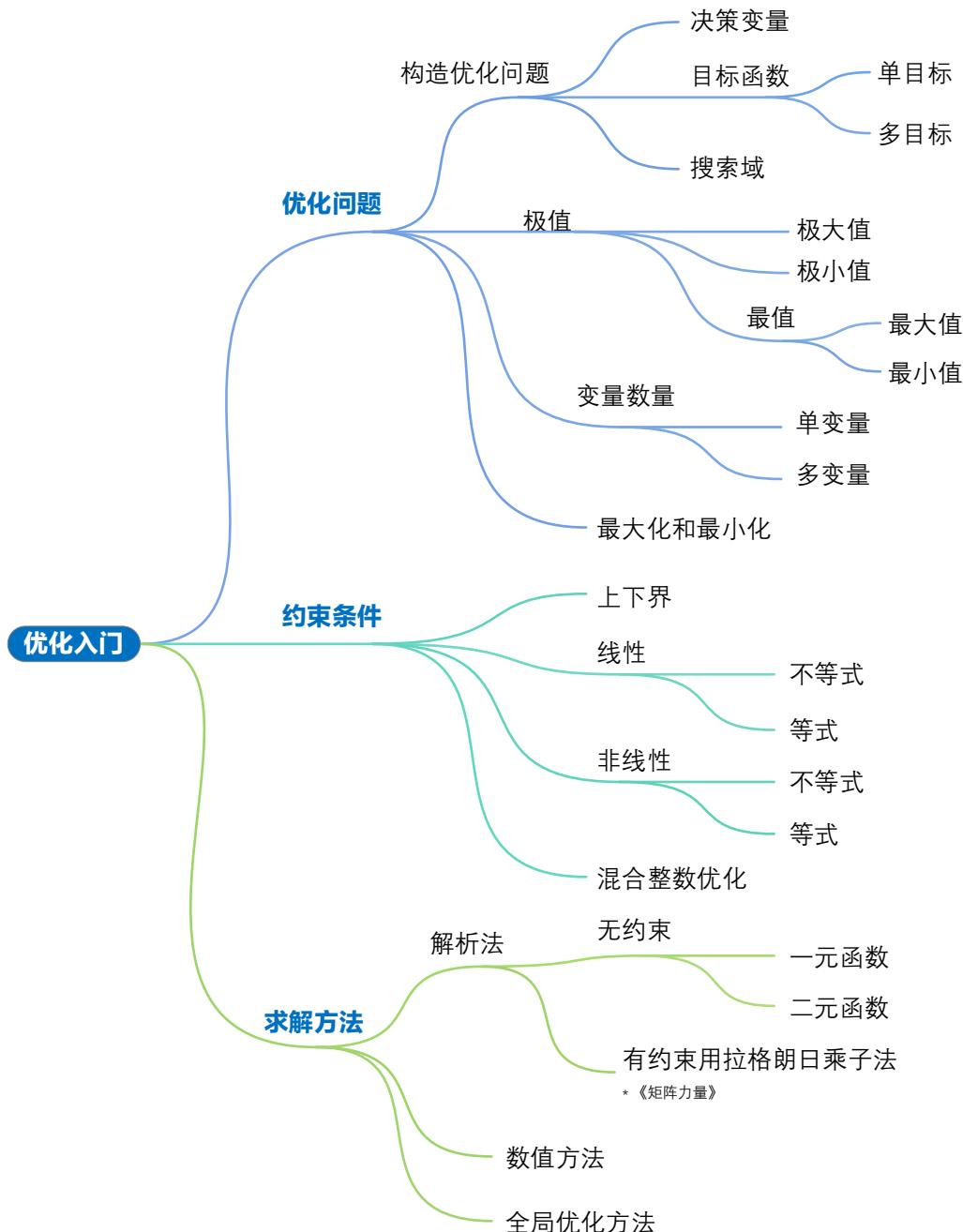
宇宙结构是完美的，是造物主的杰作；因此，宇宙中最大化、最小化准则无处不在。

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、下界约束
- ◀ `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



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

19.1 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。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，优化问题是在给定约束条件下改变**变量** (variable 或 optimization 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)。

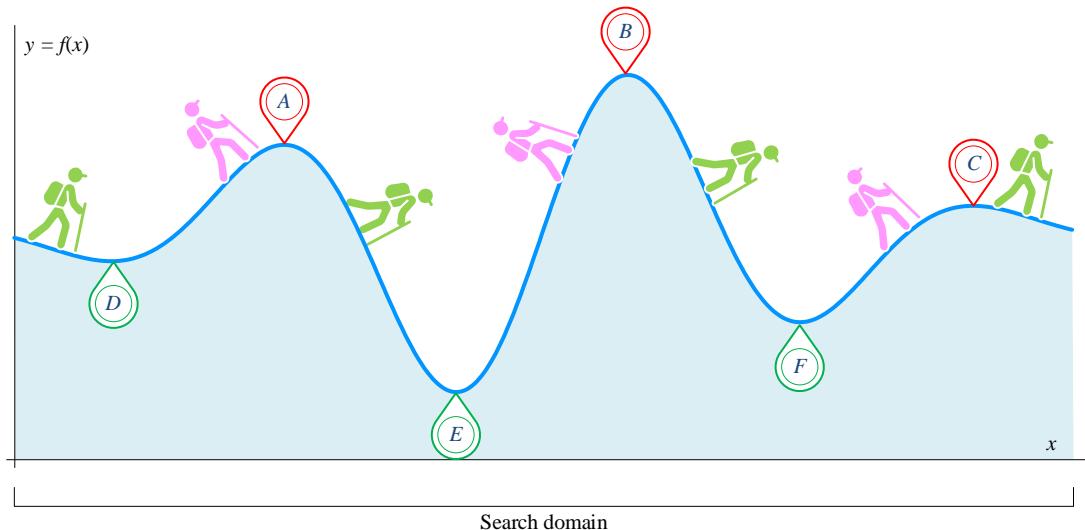


图 1. 爬上寻找山谷和山峰

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。

图 1 所示这个优化问题的变量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)。

极值

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：

$$f(a) < f(x) \quad (1)$$

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：

$$f(a) > f(x) \quad (2)$$

$f(a)$ 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，当你爬到某座山峰最顶端时，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，实际上是很优化方法的核心思想。

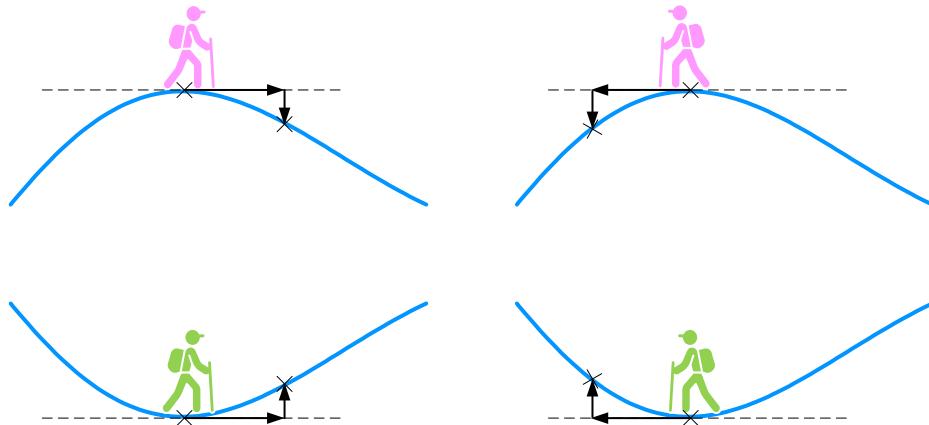


图 2. 上山和下山

最值

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：

$$f(a) < f(x) \quad (3)$$

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**局部极小值** (lobal minimum)。

总结来说，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。

19.2 构造优化问题

最小化优化问题

最小化优化问题可以写成：

$$\arg \min_x f(x) \quad (4)$$

$\arg \min$ 的含义是 argument of the minima； x 是自变量，多变量一般写成列向量； $f(x)$ 为**目标函数** (objective function)，它可以是个函数解析式 (比如 $f(x) = x^2$)，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，这个目标函数本身显然不能写成一个简单的函数。

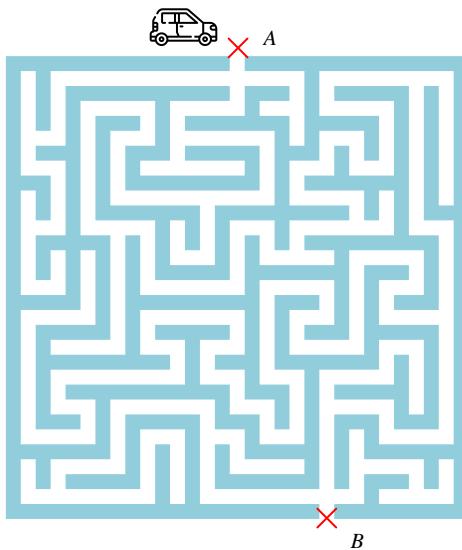


图 3. 小车自主寻找最佳路径

优化变量

前文提过， 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 ，优化变量也可以是多个未知量构成的列向量，比如 $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ 。

⚠ 注意，优化变量采用的符号未必都是 x 。以一元线性回归为例，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 $y = \theta_1x + \theta_0$ 或 $y = w_1x + w_0$ 作为线性回归模型，优化问题的变量则变为 θ_1 和 θ_0 ，或 w_1 和 w_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。

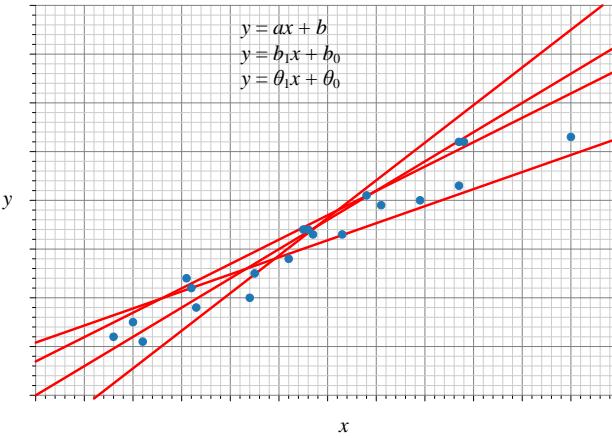


图 4. 一元最小二乘线性回归中的优化变量

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，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 $f(x_1, x_2)$ 时， $f(x_1, x_2)$ 、 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 $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ ， $f(\mathbf{x})$ 在多维空间中形成一个**超曲面** (hypersurface)。

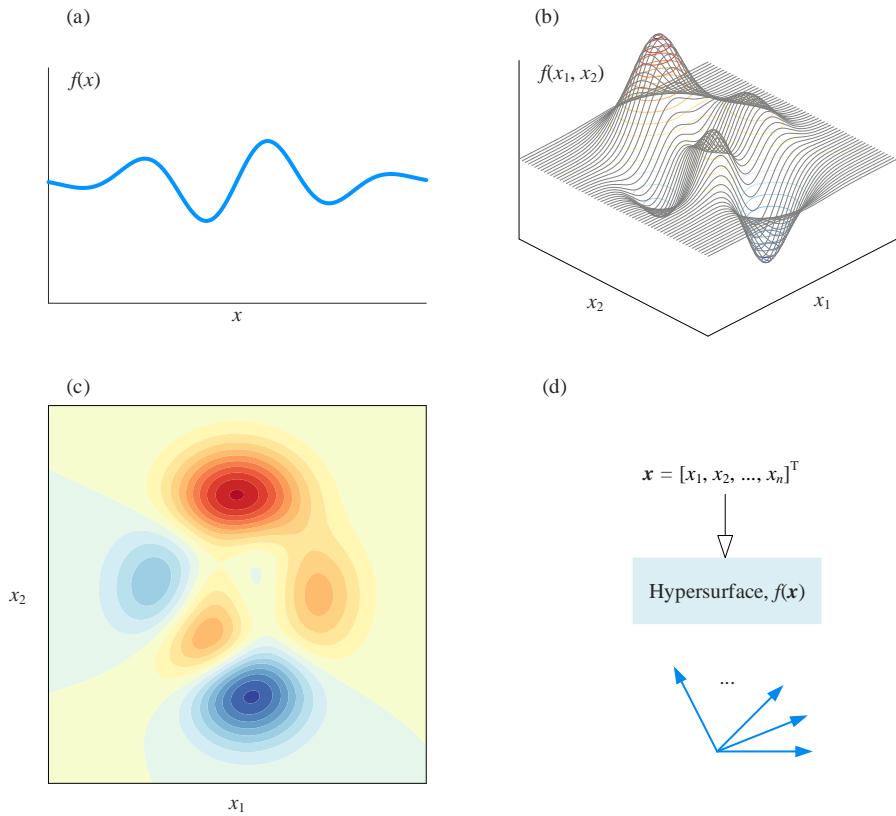


图 5. 目标函数随变量数量变化

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)。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：

$$\arg \max_{\mathbf{x}} f(\mathbf{x}) \quad (5)$$

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

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

图 6 所示为一个最大化问题转化为最小化问题。

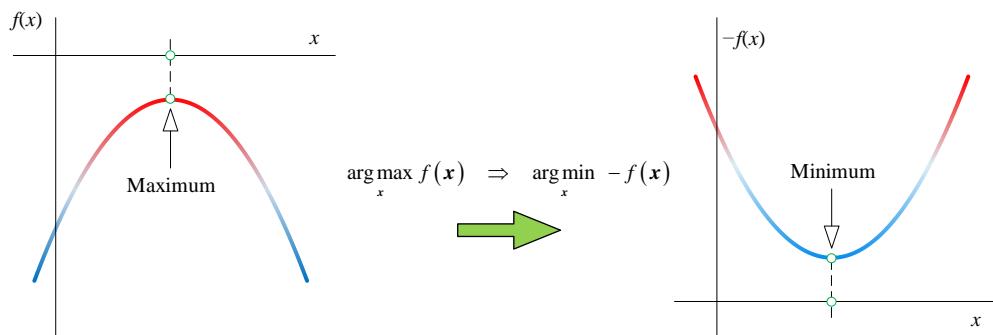


图 6. 最大化问题转化为最小化问题

19.3 约束条件：限定搜索区域

优化变量取值并非随心所欲，必须在一定范围之内。变量的取值范围叫做**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 或 feasible solution)。

优化变量取值范围的条件被称作**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), $l \leq x \leq u$
- ◀ **线性不等式** (linear inequalities), $g(x) = Ax - b \leq 0$
- ◀ **线性等式** (linear equalities), $h(x) = A_{eq}x - b_{eq} = 0$
- ◀ **非线性不等式** (nonlinear inequalities), $c(x) \leq 0$
- ◀ **非线性等式** (nonlinear equalities), $c_{eq}(x) = 0$

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 是本书第 6 章介绍的几种不等式。

最小化问题

结合约束条件，构造完整最小化问题：

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

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：

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

其中，列向量 \mathbf{l} 为 **下界** (lower bound 常简写为 lb)，列向量 \mathbf{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 $x_1 \geq 1$ 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。优化问题构造约束时，常用 `numpy.inf` 生成正无穷，`-numpy.inf` 生成负无穷。

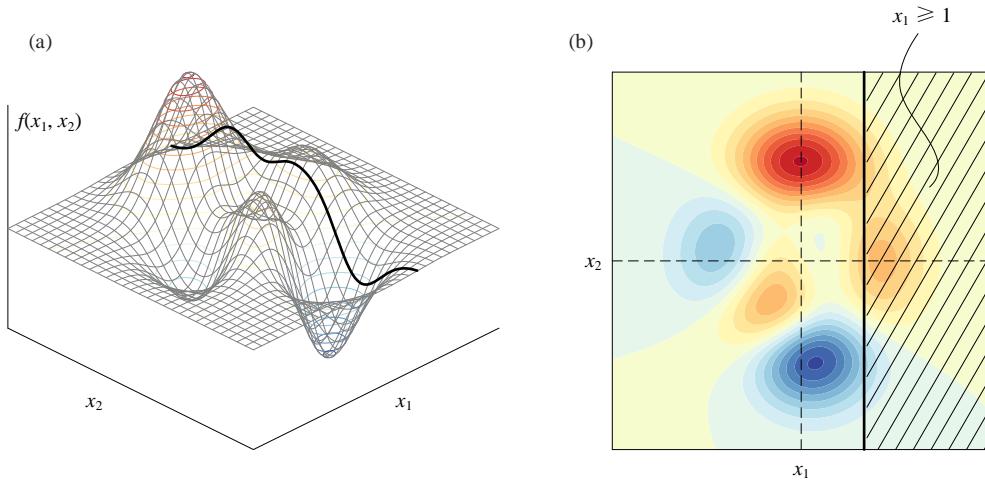


图 7. x_1 的下界为 1，上界为正无穷

图 8 中，变量 x_1 取值范围为 $-2 \leq x_1 \leq 1$ ，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。

如果一定要将 b 这个端点排除在搜索范围之外，可以用 $b - \varepsilon$ 替代 b ， ε 是个极小的正数，比如 $\varepsilon = 10^{-5}$ 。

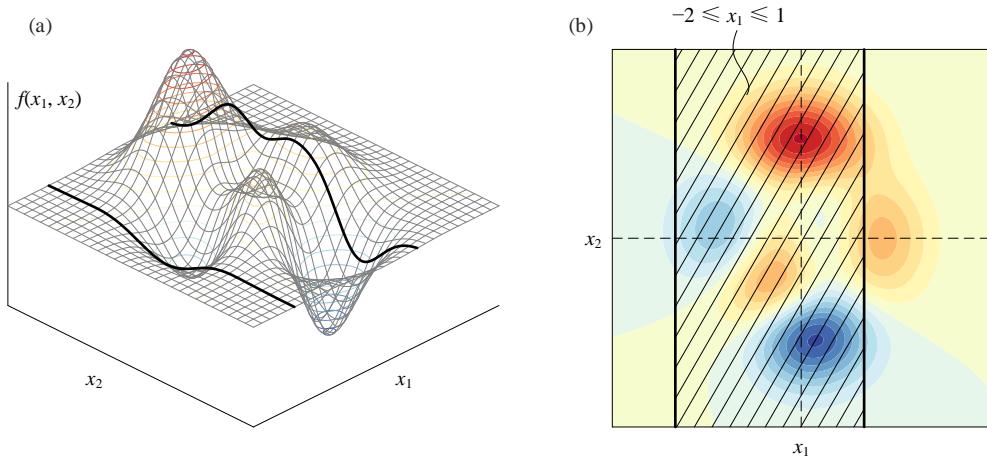


图 8. x_1 的下界为 -2，上界为 1

图 9 所示的搜索范围对应， $-1 \leq x_2 \leq 1$ ，对应区间为闭区间 $[-1, 1]$ 。图 10 所示的搜索区域同时满足 $-2 \leq x_1 \leq 1$ 和 $-1 \leq x_2 \leq 1$ ，将两者写成 (8) 形式得到：

$$\begin{bmatrix} -2 \\ -1 \end{bmatrix} \leq \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (9)$$

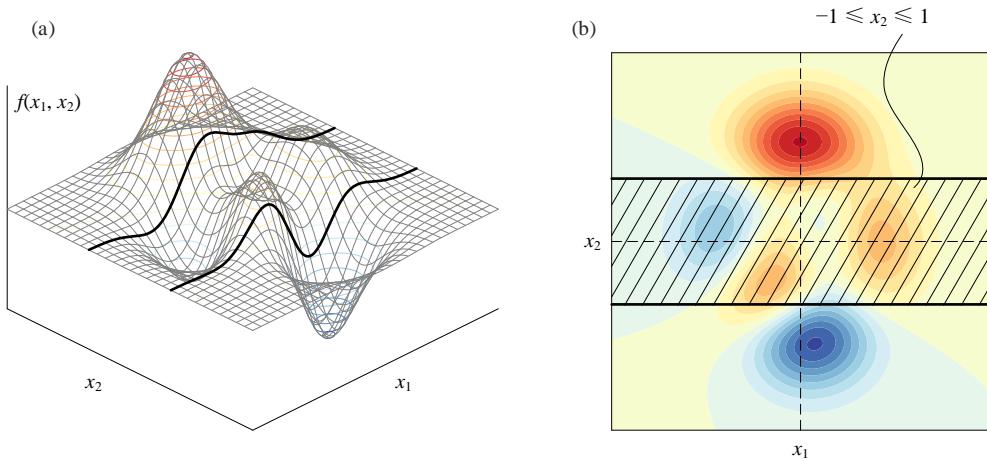
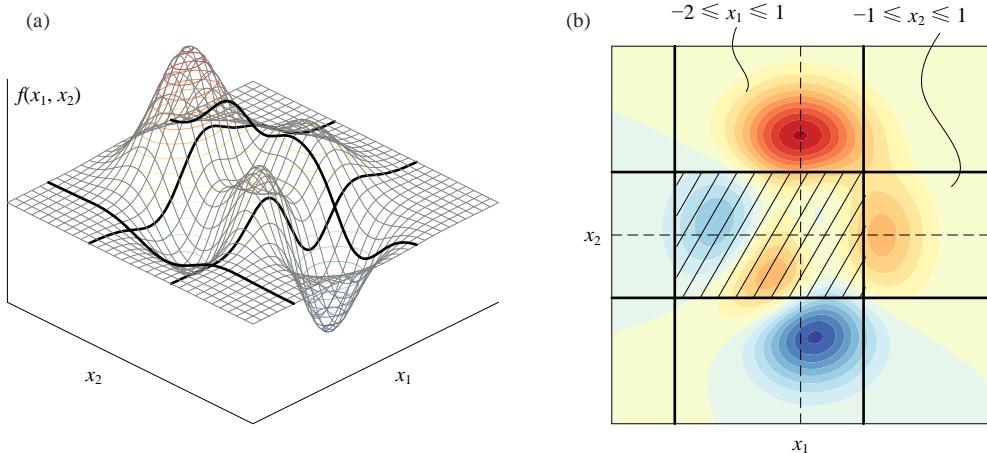


图 9. x_2 的下界为 -1，上界为 1

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

线性不等式约束

线性不等式约束表达为：

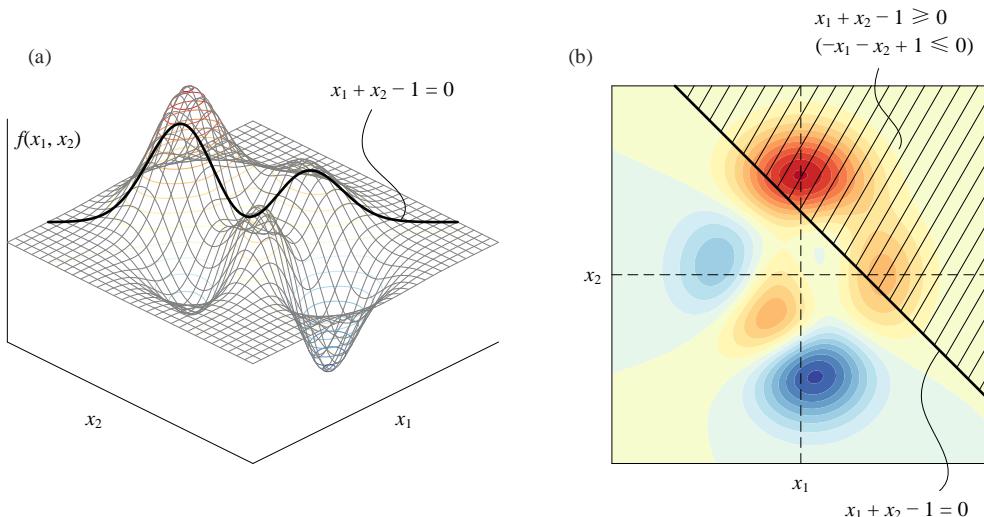
$$Ax \leq b \quad (10)$$

也常记做，

$$g(x) = Ax - b \leq 0 \quad (11)$$

图 11 所示的搜索区域为线性不等式约束，满足 $x_1 + x_2 - 1 \geq 0$ 。

⚠ 再次强调，优化问题中约束条件一般用“小于等于”，比如 $-x_1 - x_2 + 1 \leq 0$ 。

图 11. $x_1 + x_2 - 1 \geq 0$ 对应的搜索区域

线性不等式组

下例为三个线性不等式构造的约束条件：

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

首先将三个不等式所有大于等于号 (\geq) 调整为小于等于号 (\leq)，得到：

$$\begin{cases} -x_1 + 0.5x_2 \leq 1 \\ -x_1 - 2x_2 \leq -1 \\ x_1 + x_2 \leq 2 \end{cases} \quad (13)$$

然后用矩阵形式描述这一组约束条件：

$$\underbrace{\begin{bmatrix} -1 & 0.5 \\ -1 & -2 \\ 1 & 1 \end{bmatrix}}_A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} \quad (14)$$

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。

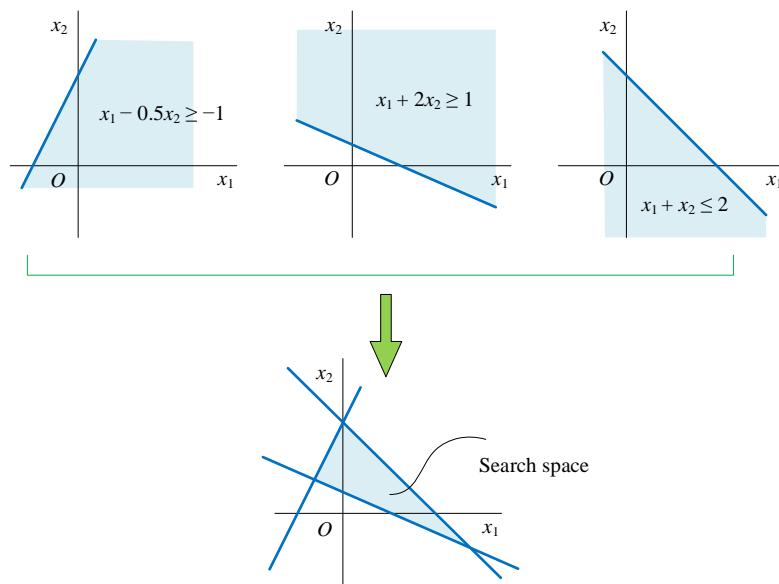


图 12. 三个线性不等式约束构造的搜索区域

线性等式约束

线性等式约束表达如下：

$$A_{\text{eq}} \mathbf{x} = \mathbf{b}_{\text{eq}} \quad (15)$$

上式常记做 $h(\mathbf{x}) = A_{\text{eq}} \mathbf{x} - \mathbf{b}_{\text{eq}} = \mathbf{0}$ 。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、一个平面或超平面。

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。

非线性不等式约束

非线性不等式约束用下式表达：

$$c(\mathbf{x}) \leq 0 \quad (16)$$

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：

$$\frac{x_1^2}{2} + x_2^2 - 1 \leq 0 \quad (17)$$

Python 中，可以同时定义非线性不等式约束的上下界，即，

$$l \leq c(\mathbf{x}) \leq u \quad (18)$$

非线性不等式一般通过构造自定义函数完成。

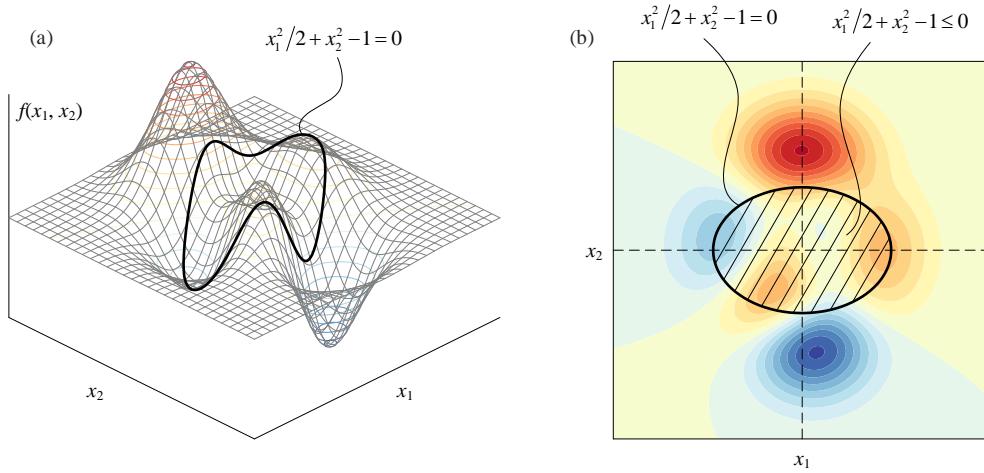


图 13. 非线性不等式约束条件

非线性等式约束通过下式定义：

$$c_{\text{eq}}(\mathbf{x}) = 0 \quad (19)$$

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

最值出现的位置

当约束条件存在时，有些最大值或最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，函数的最大值和最小值均在搜索区域内部。

而图 14 (b) 中， $f(x)$ 最小值出现在约束条件右侧边界上。

图 14 (c) 中， $f(x)$ 最大值出现在约束条件左侧边界上。

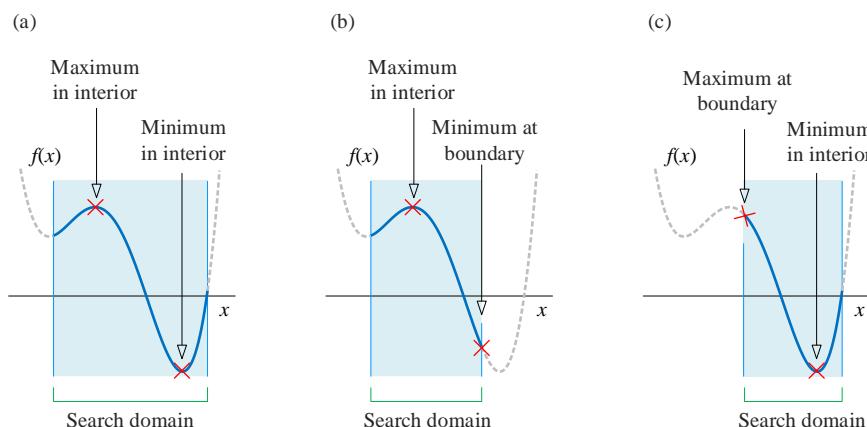


图 14. 最值和约束关系

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。

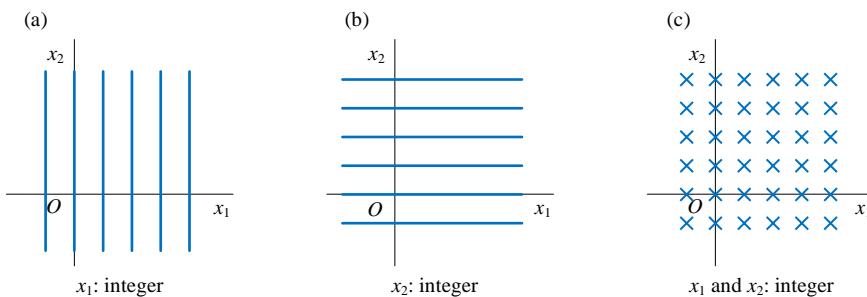


图 15. 混合整数优化

19.4 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。

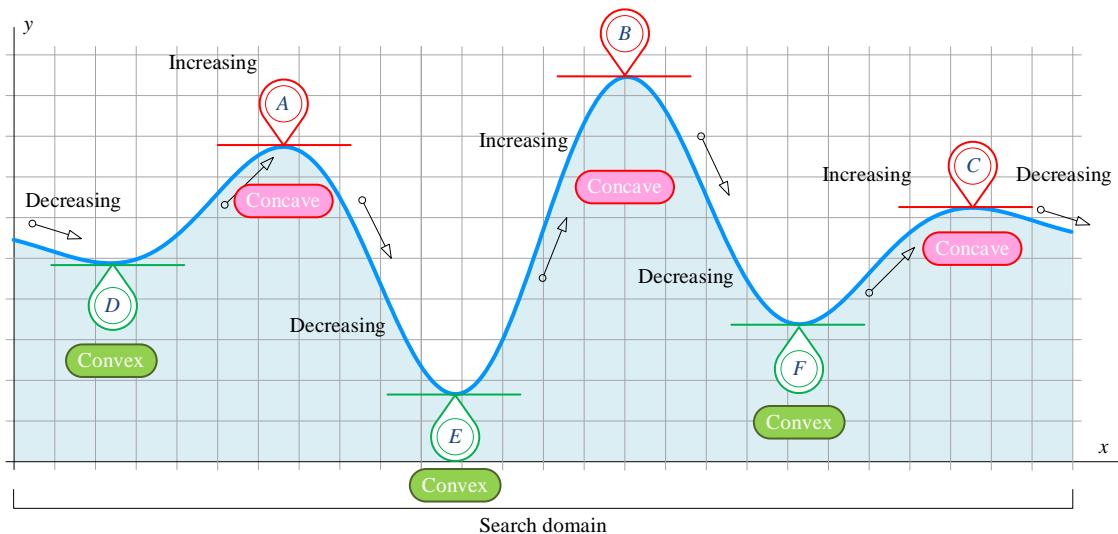


图 16. 极值两侧的增减性质

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）。不管是山峰还是山谷，只要函数光滑连续，极值点处切线为水平线。

一阶导数

本书第 15 章在讲解导数时说过，一元函数曲线上某点的导数，就是该点曲线切线斜率。

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。

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。

这样，我们便得到一元可导函数极值的一个必要条件，而非充分条件。

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 $f'(a) = 0$ 。

要判断极值点是极大值，还是极小值点，我们需要利用二阶导数进行进一步分析。

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(concave)。大家经常听说的凸优化(convex optimization)正是研究定义于凸集中的凸函数最小化的问题。

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。

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。

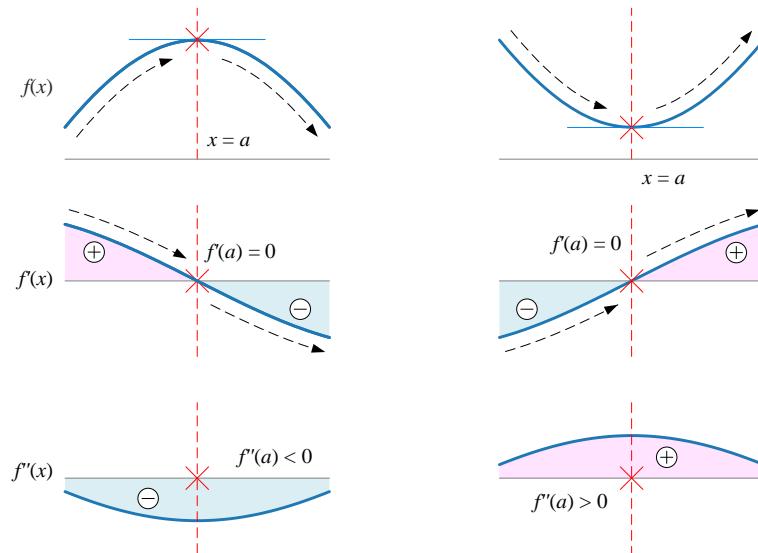


图 17. 极大值和极小值点局部, 用正弦余弦重画图

一阶导数和二阶导数均为 0

如果函数驻点的一阶导数和二阶导数都为 0, 可以用驻点左右一阶导数符号判定极值。图 18 分别给出原函数、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负和正, 显然 $x = 0$ 为函数的极小值点。

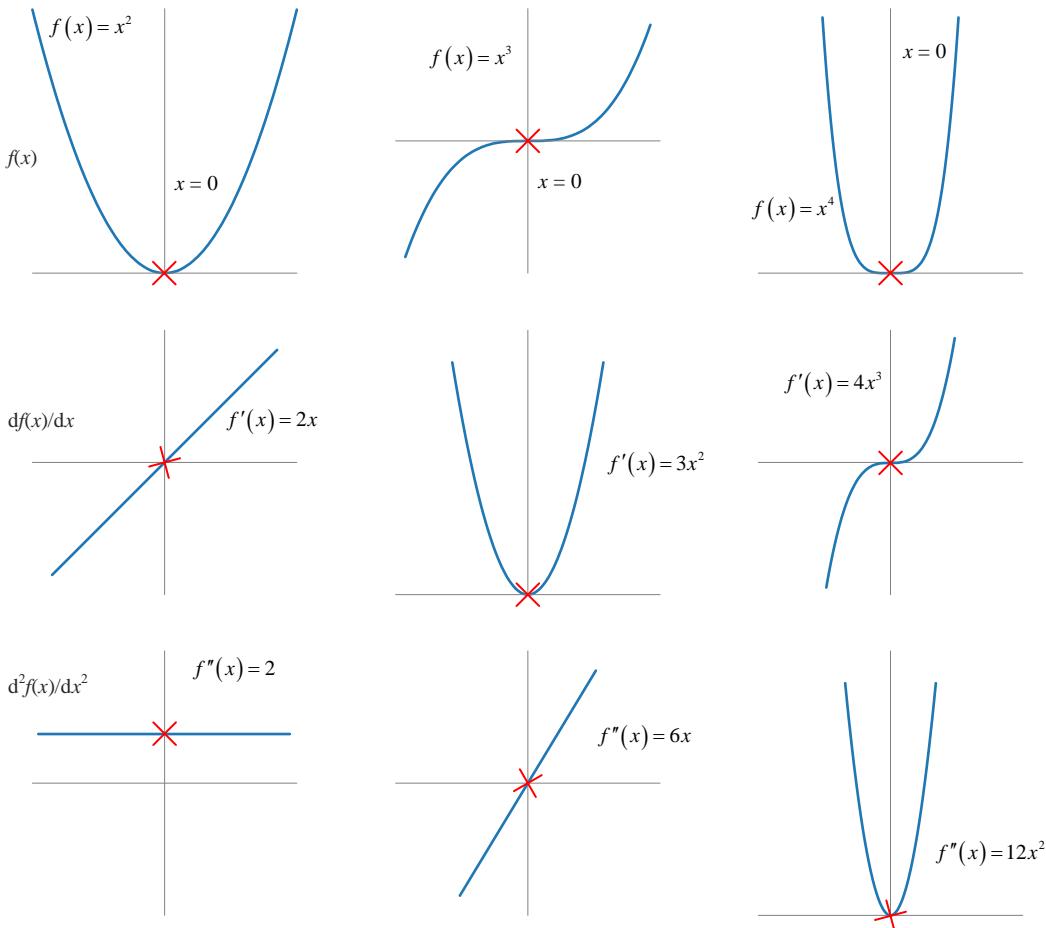


图 18. 通过一阶导数和二阶导数判断极值

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。

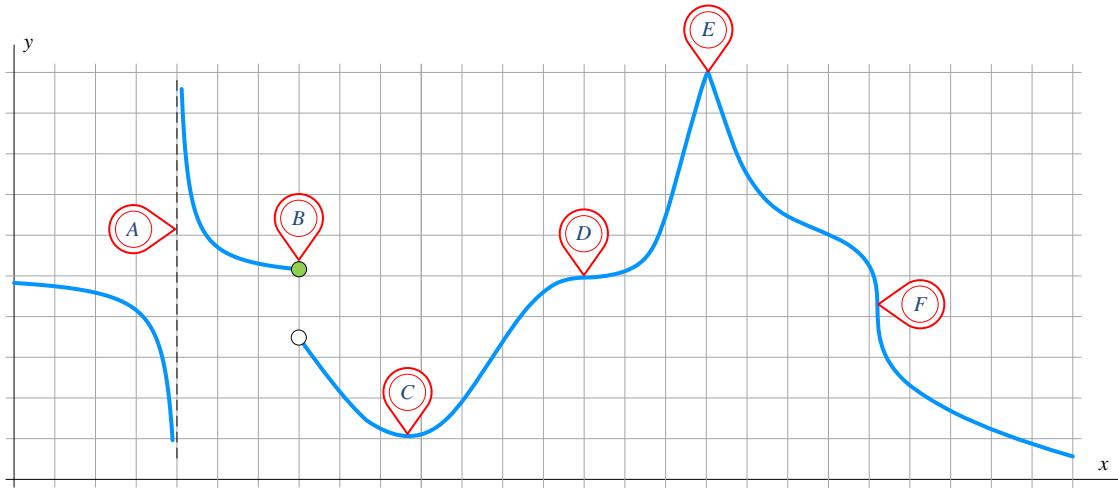


图 19. 一次函数值得关注的几个点

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：

$$\min_x f(x) = -2x \cdot \exp(-x^2) \quad (20)$$

函数的一阶导数为：

$$f'(x) = 4x^2 \cdot \exp(-x^2) - 2\exp(-x^2) \quad (21)$$

一阶导数为 0 有两个解，分别是：

$$x = \pm \frac{\sqrt{2}}{2} \quad (22)$$

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。

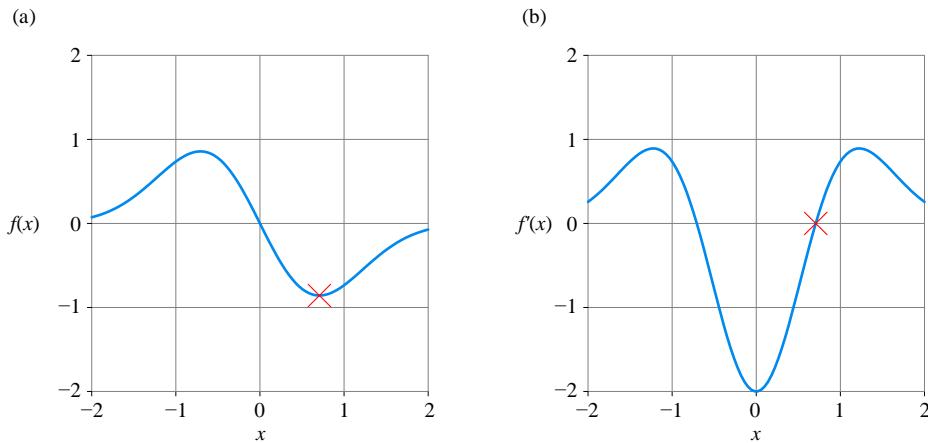
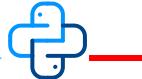


图 20. 一次函数图像和一阶导函数图像，极小值点位置



Bk3_Ch19_01.py 完成优化问题求解，并绘制图 20。

19.5 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，

$$f_{x_1}(a, b) = 0, \quad f_{x_2}(a, b) = 0 \quad (23)$$

对于二元函数极值的判定，一阶偏导数 $f_{x_1}(x_1, x_2) = 0$ 和 $f_{x_2}(x_1, x_2) = 0$ 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。

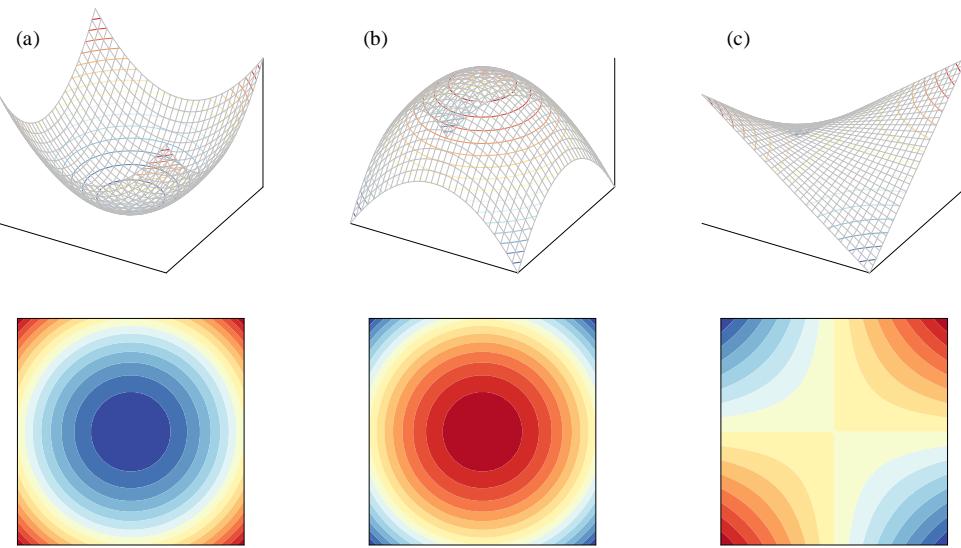


图 21. 二元函数驻点的三种情况

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，

$$A = f_{x_1 x_1}(a, b), \quad B = f_{x_1 x_2}(a, b), \quad C = f_{x_2 x_2}(a, b) \quad (24)$$

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

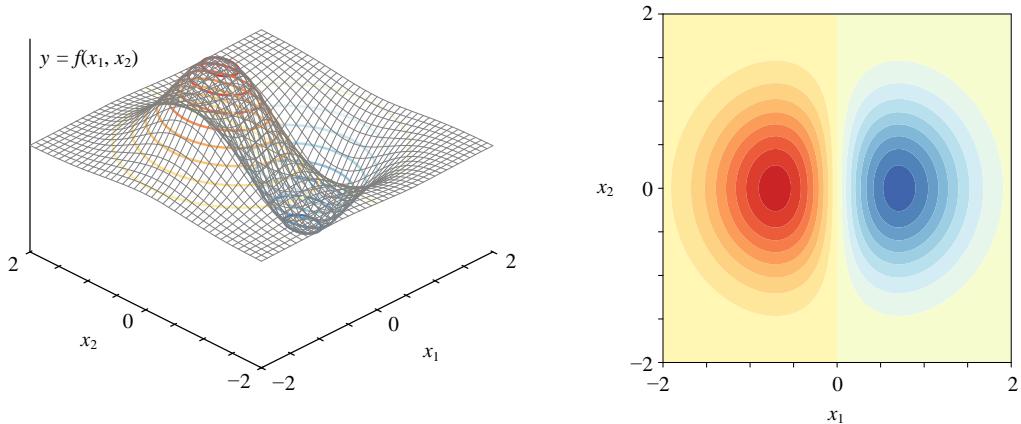
- a) $AC - B^2 > 0$ 存在极值，且当 $A < 0$ 有极大值， $A > 0$ 时有极小值；
- b) $AC - B^2 < 0$ 没有极值；
- c) $AC - B^2 = 0$, 可能有极值，也可能没有极值，需要进一步讨论。

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：

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

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。

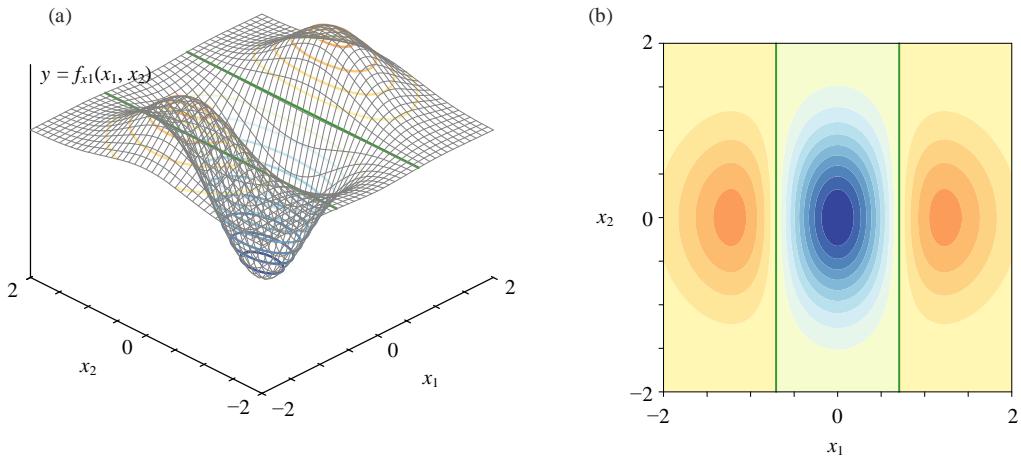
图 22. $f(x_1, x_2)$ 曲面和等高线图像

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

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

⚠ 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。

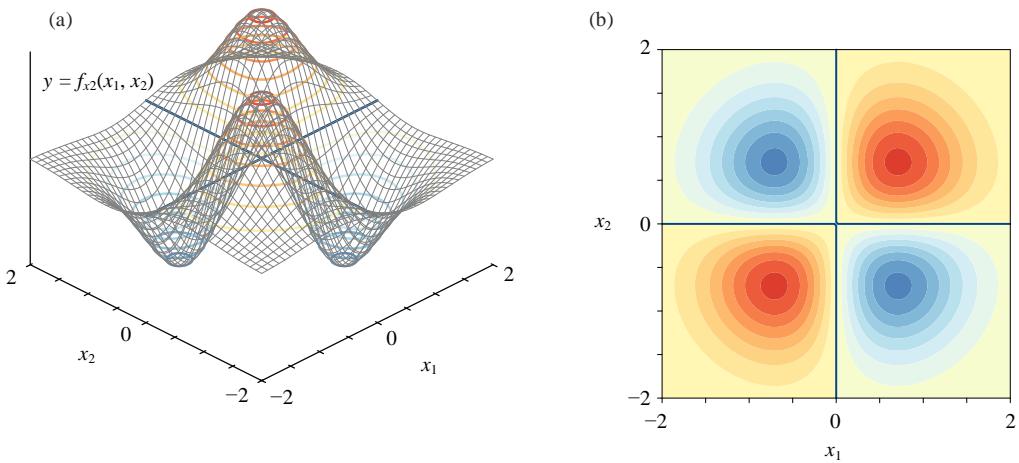
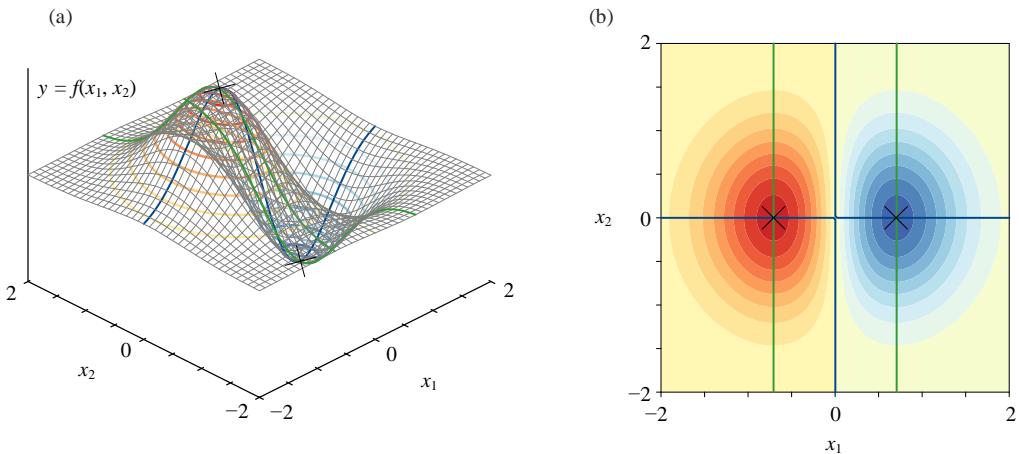
图 23 中墨绿色实线对应 $f_{x1}(x_1, x_2) = 0$ 。

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：

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

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 中深蓝色曲线对应 $f_{x2}(x_1, x_2) = 0$ 。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:

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

$f(x_1, x_2)$ 对 x_1 和 x_2 混合二阶偏导:

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

$f(x_1, x_2)$ 对 x_2 的二阶偏导:

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

$AC - B^2$ 对应的解析式为:

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

如图 26 (a) 所示另个鞍点处, $AC - B^2$ 均大于 0, 这说明两点均为极值点; 根据图 26 (b) 可以判定极值类型。

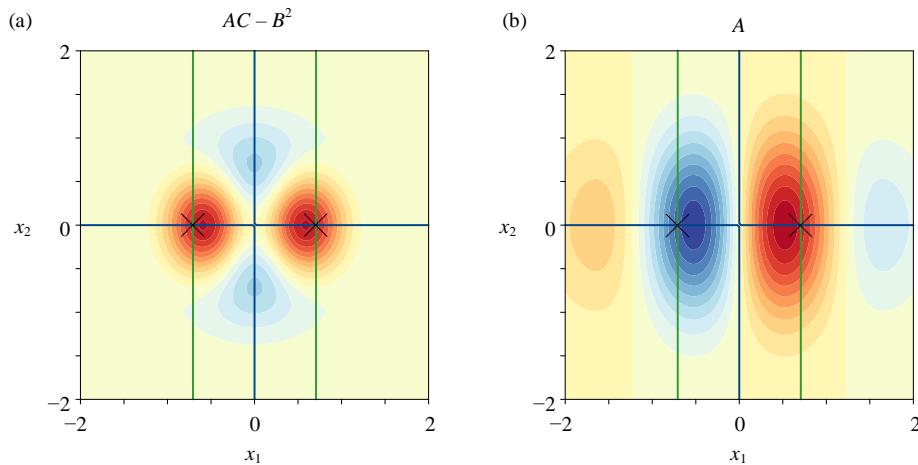


图 26. $AC - B^2$ 和 A 解析式等高线

有约束条件

在 (25) 基础上, 加上非线性约束条件:

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

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:

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

图 27 (b) 所示, 优化结果出现在边界上。

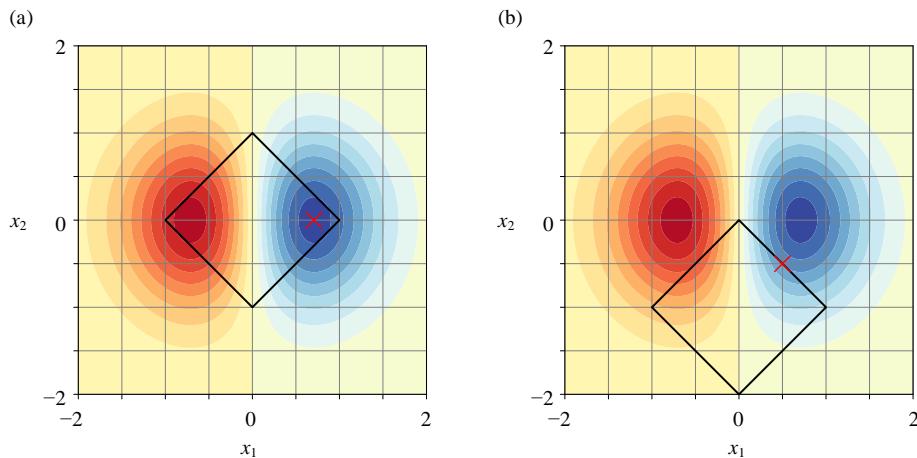
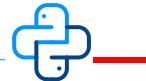


图 27. 两个非线性条件对优化结果影响



`Bk3_Ch19_02.py` 求解含有非线性约束条件的优化问题，并绘制图 27。



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。

20 Fundamentals of Probability 概率入门

从杨辉三角到古典概率模型



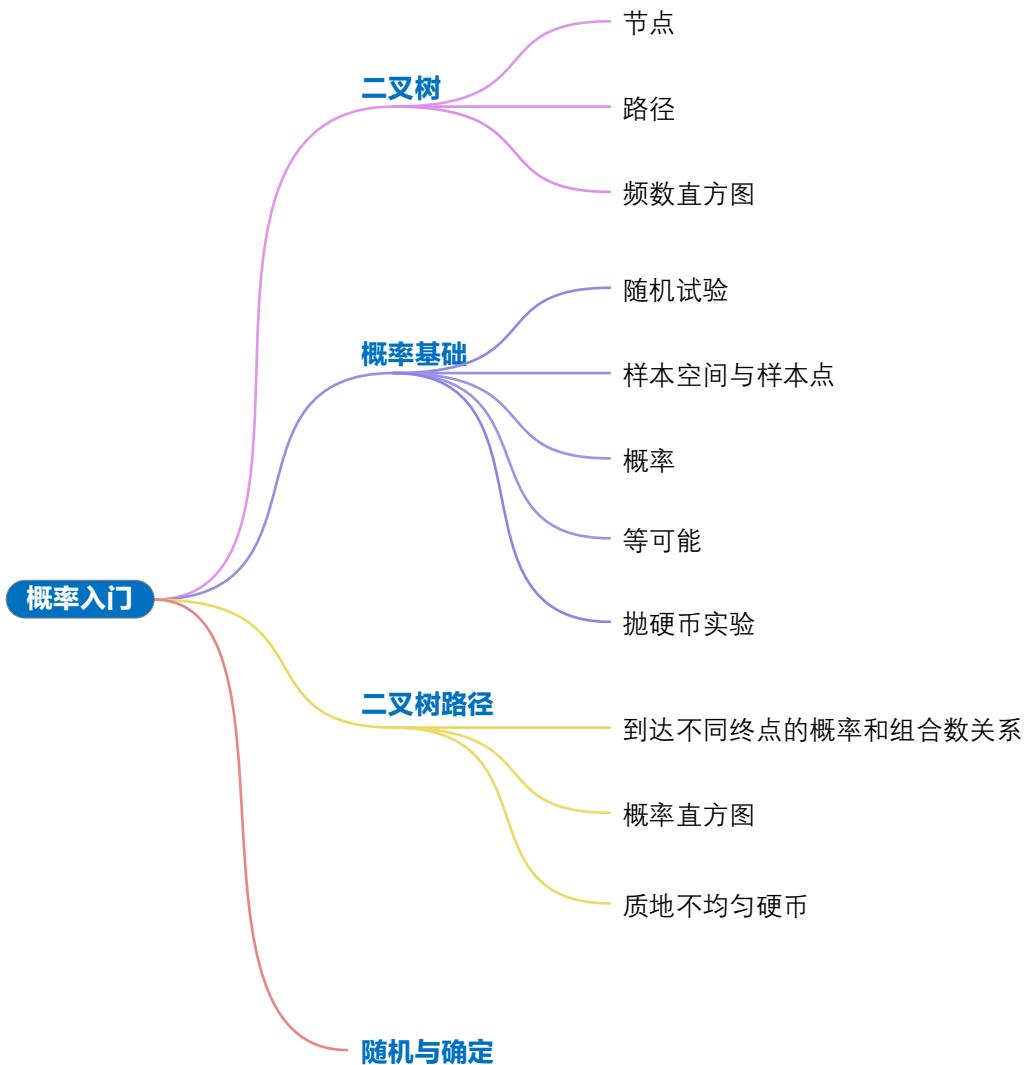
这个世界的真正逻辑是概率的推演。

The true logic of this world is the calculus of probabilities.

—— 詹姆斯·克拉克·麦克斯韦 (James Clerk Maxwell) | 英国数学物理学家 | 1831 ~ 1879



- ◀ ax.invert_xaxis() 调转 x 轴
- ◀ ax1.spines['right'].set_visible(False) 除去图像右侧黑框线
- ◀ ax1.spines['top'].set_visible(False) 除去图像上侧黑框线
- ◀ itertools.combinations() 无放回抽取组合
- ◀ itertools.combinations_with_replacement() 有放回抽取组合
- ◀ itertools.permutations() 无放回排列
- ◀ matplotlib.pyplot.barh() 绘制水平直方图
- ◀ matplotlib.pyplot.stem() 绘制火柴梗图
- ◀ numpy.concatenate() 将多个数组进行连接
- ◀ numpy.stack() 将矩阵叠加
- ◀ numpy.zeros_like() 用来生成和输入矩阵形状相同的零矩阵
- ◀ scipy.special.binom() 产生二项式系数
- ◀ sympy.Poly 将符号代数式转化为多项式



20.1 概率简史：出身赌场

概率是现代人类的自然思维方式。大家在日常交流时，用到“预测”、“估计”、“肯定”、“百分之百的把握”、“或许”、“百分之五十可能性”、“大概”、“可能”、“恐怕”、“绝无可能”等字眼时，思维已经进入概率的范畴。

概率论的目的就是将这些字眼数学化、量化。

意大利学者吉罗拉莫·卡尔达诺 (Girolamo Cardano, 1501 ~ 1576) 可以说是文艺复兴时期百科全书式人物。他做过执业医生，第一个发表三次代数方程式的一般解法，他还是赌场常胜将军。

卡尔达诺死后才向世人公布自己创作的赌博秘籍《论赌博的游戏》(Book on Games of Chance)，这本书首次对概率进行系统介绍。他在书中用投色子游戏讲解等可能事件和其他概率概念。值得一提的是，卡尔达诺的父亲和达芬奇是好友。和达芬奇一样，卡尔达诺也是私生子。

概率论的基本原理是在帕斯卡 (Blaise Pascal, 1623 ~ 1662) 和费马 (Pierre de Fermat, 1607 ~ 1665) 的一系列来往书信中搭建起来的。他们在书信中讨论的是著名的赌博奖金分配问题。

举个例子说明赌博奖金分配问题。A、B 两人玩抛硬币游戏，每次抛一枚硬币，硬币朝上 A 得一分，硬币朝下 B 得一分，谁先得到 10 分谁就赢得所有奖金。但是，游戏进行到途中突然中断，此时 A 得分 7 分，B 得分 5 分，两人此时应该如何分配奖金？

在帕斯卡和费马的讨论中，他们提出了枚举法。一些书信中也能看到他们谈到利用杨辉三角和二项式展开求解赌博奖金分配问题。

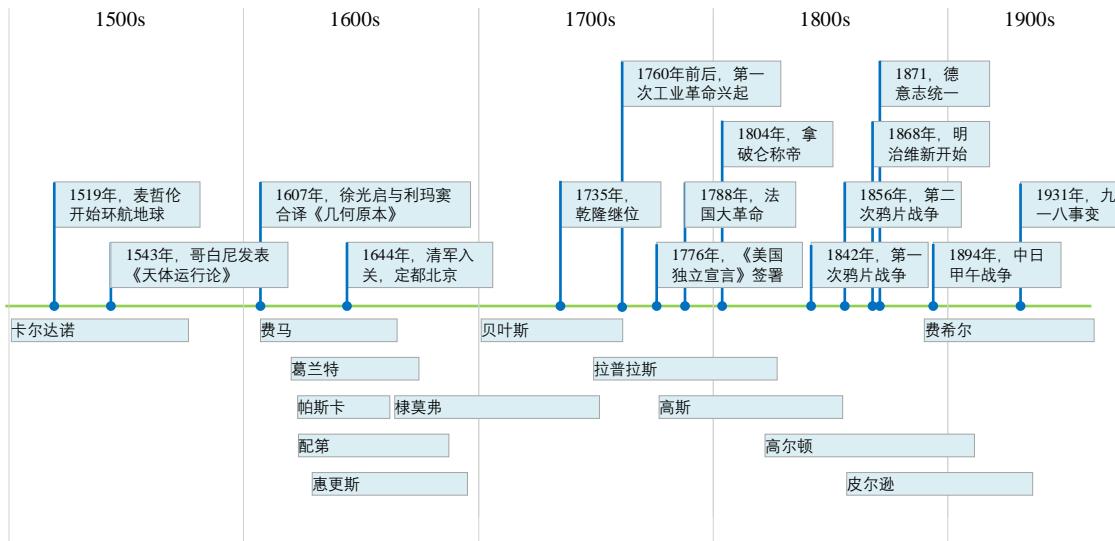


图 1. 概率论、统计学发展时间轴

克里斯蒂安·惠更斯 (Christiaan Huygens, 1629 ~ 1695) 扩展了帕斯卡和费马的理论。惠更斯 1657 年发表了《论赌博中的计算》(On Reasoning in Games of Chance)，被很多人认为是概率论诞生的标志。

法国数学家**亚伯拉罕·棣莫弗**(Abraham de Moivre, 1667 ~ 1754)继续推动概率论的发展，他首先提出正态分布、中心极限定理等。在处理莱布尼兹-牛顿微积分发明权之争，棣莫弗还被选做裁决人之一。

贝叶斯(Thomas Bayes, 1701 ~ 1761)在自己的论文《解决机会学说中的问题》(An Essay Towards Solving a Problem in the Doctrine of Chances)中探讨了条件概率，这使得贝叶斯成为贝叶斯学派的开山鼻祖。

在概率领域，**高斯**(Carl Friedrich Gauss, 1777 ~ 1855)发明最小二乘法。虽然正态分布常被称作高斯分布，但是高斯不是正态分布的第一发明者。

弗朗西斯·高尔顿(Francis Galton, 1822 ~ 1911)则提出回归、相关系数等重要统计学概念。有趣的是，高尔顿是查尔斯·达尔文的表弟。

概率论和统计学两门学科相互交融，而且发展历史跨度很大，太多学者起到推动力作用。很可惜，限于篇幅本节只能走马观花用几句话概括关键人物的生平。

20.2 二叉树：一生二、二生三

杨辉三角可谓是算数、代数、几何、数列、概率的完美结合体。沿着帕斯卡和费马的思路，本章从杨辉三角入手来和大家探讨概率论的核心思想。

本节首先从一个全新视角解读杨辉三角——**二叉树**(binomial tree)。将本书第4章介绍的杨辉三角逆时针旋转90度，得到图2这个二叉树。图2中每个点称作**节点**(node)。

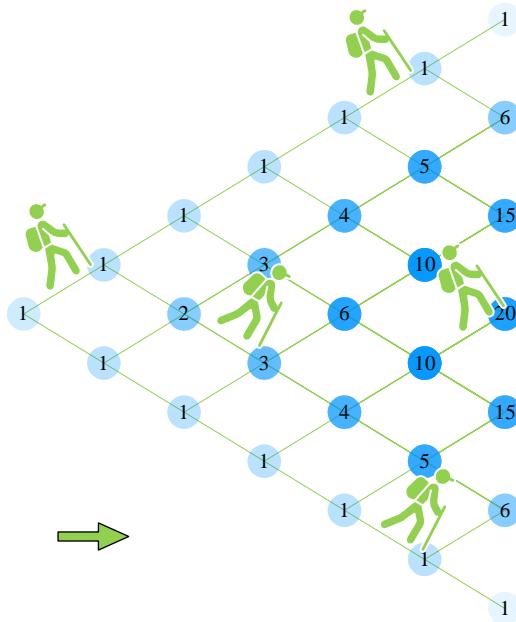


图 2. 杨辉三角逆时针旋转 90 度得到一个二叉树

试想，一名登山者从最左侧初始点出发，沿着二叉树规划的路径向右移动，到达最右侧任意节点结束。途中每个节点处，登山者可以向右上方或右下方走，但是不能往回走。

这样，图2中的数字便有了另外一层内涵——登山者到达对应节点的可能路径。

二叉树原理

下面解释一下原理。

如图3所示，当 $n=1$ 时，二叉树叫做**一步二叉树** (one-step binomial tree)。也就是说，登山者从初始点出发，只有两个路径到达两个不同终点。

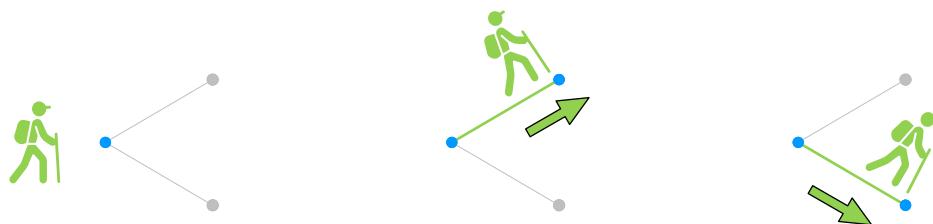


图 3. $n = 1$, 向上、向下走的路径

如图4所示， $n=2$ 时，二叉树为**两步二叉树** (two-step binomial tree)。从起点到终点，一共有4条路径，二项式系数1、2、1则相当于到达对应A、B、C终点的可能路径数量。

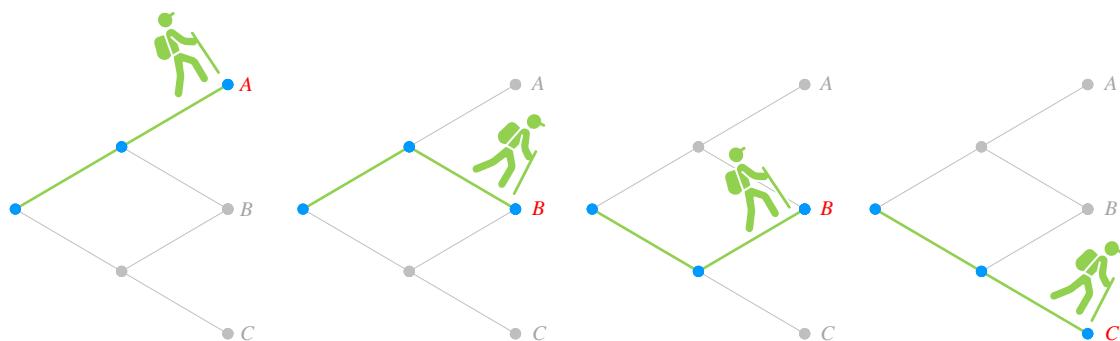


图 4. $n = 2$, 通向最终节点路径

当二叉树的层数不断增多，到达终点的路径的数量呈现指数增长趋势。

如图5(a)所示， $n=3$ 时，路径数量为 $8 (= 1 + 3 + 3 + 1 = 2^3)$ 。如图5(b)所示， $n=4$ 时，路径数量为 $16 (= 1 + 4 + 6 + 4 + 1 = 2^4)$ 。如图5(c)所示， $n=5$ 时，路径数量为 $32 (= 1 + 5 + 10 + 10 + 5 + 1 = 2^5)$ 。

这个结果也不难理解，二叉树每增加一层，登山者就多一次二选一的机会。从路径数量角度，就是再乘 2。

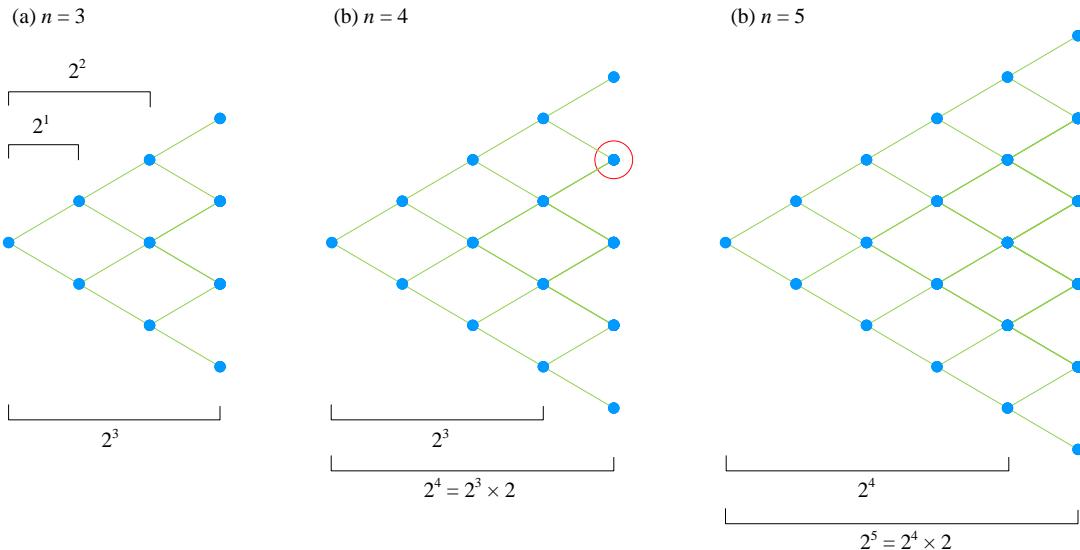


图 5. $n = 3, 4, 5$, 通向最终节点路径

图 6 所示为 4 条到达图 5 (b) 二叉树画红圈终点节点路径。4 这个结果和组合数有着密切关系。下面我们聊一下如何用组合数解释到达不同终点路径数。

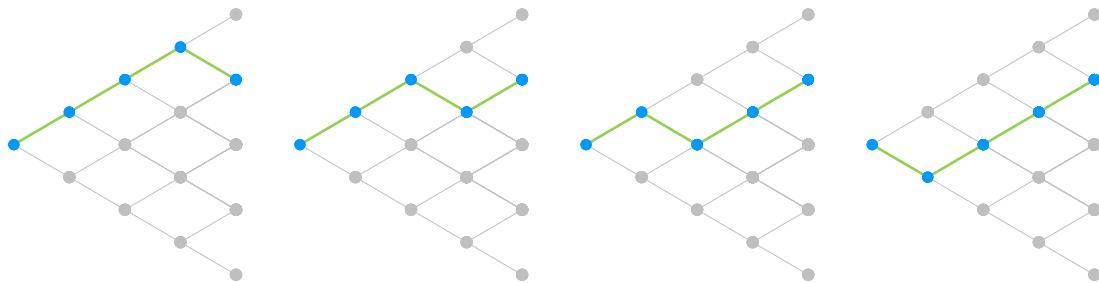


图 6. 四条到达同一终点节点的路径

组合数

利用**水平条形图** (horizontal bar graph) 可可视化图 5 二叉树路径数。如图 7 所示， $n = 3$ 时，到达二叉树终点节点的路径分别有 1、3、3、1 条，总共有 8 条路径，写成组合数：

$$C_3^0 + C_3^1 + C_3^2 + C_3^3 = 1 + 3 + 3 + 1 = 8 = 2^3 \quad (1)$$

大家可能会问，组合数在这里扮演的角色是什么？

很容易理解，登山者在图 7 所示二叉树需要做三次“向上走或向下走”的决策。

C_3^0 可以理解为，3 次决策中 0 次向下； C_3^1 可以理解为，3 次决策中 1 次向下； C_3^2 可以理解为，3 次决策中 2 次向下； C_3^3 可以理解为，3 次决策中 3 次向下。

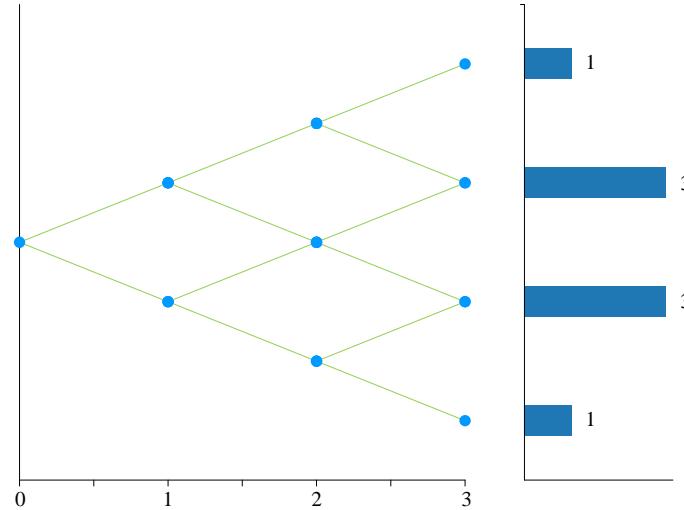


图 7. $n = 3$, 二叉树路径数分布

如图 8 所示， $n = 4$ 时，到达二叉树终点节点的路径分别有 1、4、6、4、1 条，总共有 16 条路径：

$$C_4^0 + C_4^1 + C_4^2 + C_4^3 + C_4^4 = 1 + 4 + 6 + 4 + 1 = 16 = 2^4 \quad (2)$$

也就是说，这种情况登山者面临 4 次“二选一”的决策。

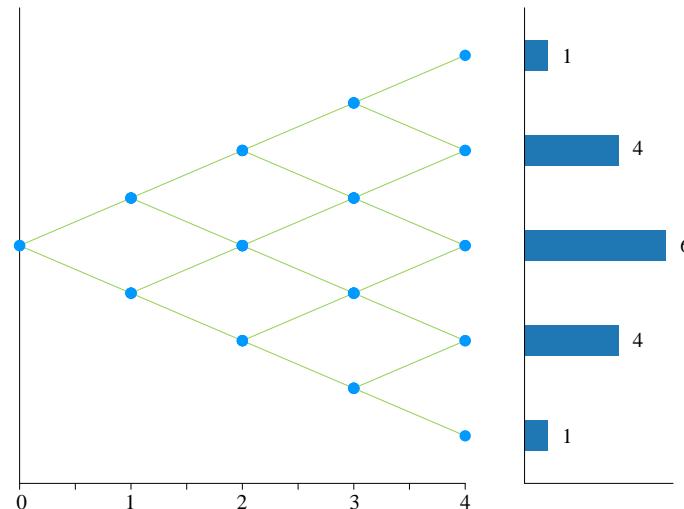


图 8. $n = 4$, 二叉树路径数分布

如图9所示， $n=5$ 时，登山者有5次“二选一”决策，到达二叉树终点节点的路径分别有1、5、10、10、5、1条，总共有32条路径：

$$C_5^0 + C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5 = 1 + 5 + 10 + 10 + 5 + 1 = 32 = 2^5 \quad (3)$$

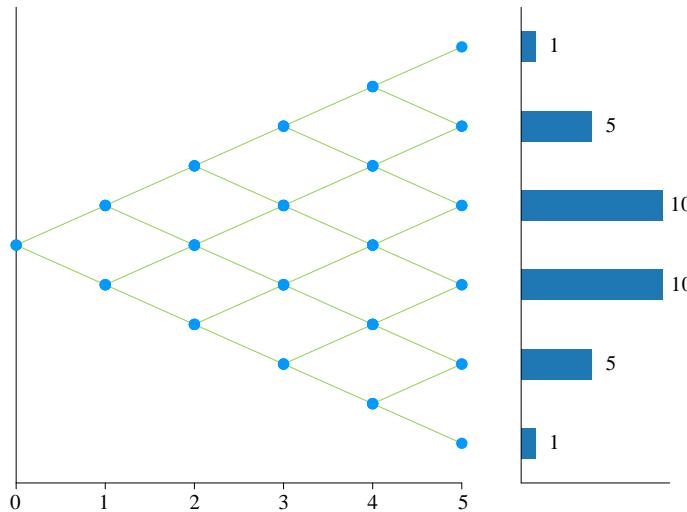
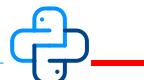


图 9. $n=5$, 二叉树路径数分布

从概率统计角度，图9右侧的直方图常被称作**频数直方图** (frequency histogram)。频数也称次数，是对总数据按某种标准进行分组，统计出各个组内含个体的个数。

杨辉三角和二叉树体现出来的规律像极了老子所言“道生一，一生二，二生三，三生万物。”



代码文件 Bk3_Ch20_1.py 中 Bk3_Ch20_1_A 部分绘制图 7 ~ 图 9。

20.3 抛硬币：正反面概率

确定与随机

在自然界和社会实践活动中，人类遇到的各种现象可分为两大类：确定现象，随机现象。

随机现象的准确定义是：在一定条件下，出现的可能结果不止一个，事前无法确切知道哪一个结果一定会出现，但大量重复试验中其结果又具有统计规律的现象称为随机现象。

一年 24 节气轮替，太阳东升西落，这是确定性现象。某一年是干旱少雨，还是洪涝灾害频发，某一天是否会下雨，什么时候下雨，降水量多大，这些事情的结果都是随机的。

天地不仁，以万物为刍狗——感觉这句就是在说随机性。

但是，随机之中有确定。举个例子，抛一枚硬币，谁也不能准确预测硬币落地时是正面还是反面朝上。但是，大量抛硬币，却发现硬币的正反面平均值有一定的规律。

人类虽然不能百分之百准确预测明年今天的晴雨状况。但是，通过研究大量气象数据，我们可以找到降水周期性规律，并在一定范围内预测降水量。

在微观、少量、短期尺度上，我们看到的更多的是不确定、不可预测、随机；但是，站在宏观、大量、更长的时间尺度上，我们可以发现确定、模式、规律。

随机试验

随机试验 (random experiment) 是在相同条件下对某随机现象进行的大量重复观测。随机试验需要满足三个条件：

- a) 可重复，在相同条件下试验可以重复进行；
- b) 结果集合明确，每次试验的可能结果不止一个，并且能事先明确试验的所有可能结果；
- c) 单次试验结果不确定，进行一次试验之前不能确定哪一个结果会出现，但必然出现结果集合中的一个。

给定一个随机试验，所有的结果构成的集合为样本空间 Ω 。样本空间 Ω 中的每一个元素为一个样本点。

概率

概率 (probability) 反映随机事件出现的可能性大小。

给定任意一个事件 A ， $\Pr(A)$ 为事件 A 发生的概率 (the probability of event A occurring)。

⚠ 注意本书概率记法， \Pr 为正体。

对于任意事件 A ， A 发生的概率满足：

$$\Pr(A) \geq 0 \tag{4}$$

整个样本空间 Ω 的概率为 1，即，

$$\Pr(\Omega) = 1 \tag{5}$$

空集 \emptyset 不包含任何样本点，也称作不可能事件，因此对应的概率为 0，即：

$$\Pr(\emptyset) = 0 \tag{6}$$

白话说，一定会发生的事情，概率值为 1 (100%)；一定不会发生的事情，概率值为 0 (0%)。不一定会发生的事情，概率值在 0、1 之间。这就是量化“可能性”的基础。

等可能

等可能性是指设一个试验的所有可能产生的结果有 n 个，它们都是随机事件，每次试验有且只有其中的一个结果出现。

如果每个结果出现的机会均等，那么说这 n 个事件的发生是等可能试验的结果。设样本空间 Ω 由 n 个等可能的试验结果构成，事件 A 的概率为：

$$\Pr(A) = \frac{n_A}{n} \quad (7)$$

其中， n_A 为含于事件 A 的试验结果数量。

这种基本事件个数有限且等可能的概率模型，称为古典概率模型。所谓概率模型是对不确定现象的数学描述。

抛硬币

举最简单的例子，抛一枚硬币，1 代表落地结果正面、0 代表反面。抛一枚硬币的可能结果样本空间 Ω 为：

$$\Omega = \{0, 1\} \quad (8)$$

根据生活常识，如果硬币质地均匀。获得正面和反面的概率相同均为 $1/2$ ，即等可能：

$$\Pr(0) = \Pr(1) = \frac{1}{2} \quad (9)$$

连续抛 100 枚硬币，并记录每次硬币正 (1)、反面 (0) 结果。图 10 所示为每一次试验硬币正反面结果以及累计结果平均值变化。可以发现，随着抛硬币的次数不断增多，硬币正反面平均值愈靠近 $1/2$ 。

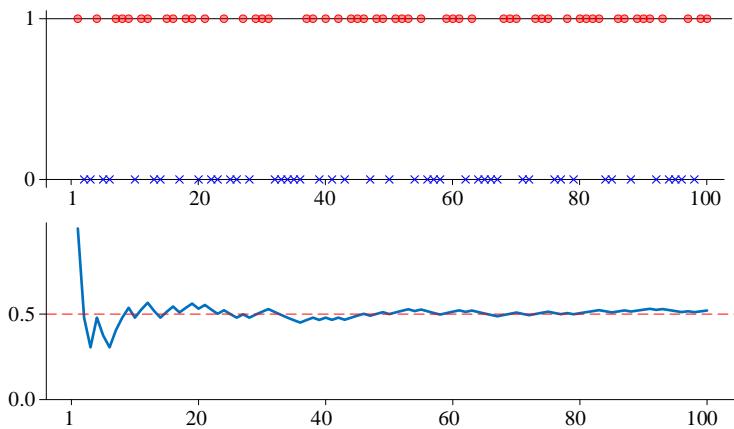
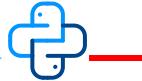


图 10. 抛硬币 100 次试验，硬币正反面结果，以及平均值变化



Bk3_Ch20_2.py 绘制图 10。

20.4 聊聊概率：向上还是向下

本节引入概率，给杨辉三角增加一个新视角。

登山者在二叉树始点或中间节点时，都会面临“向上”或“向下”这种二选一抉择。如果登山者通过抛硬币，决定每一步的行走路径——正面，向右上走；反面，向右下走。

生活经验告诉我们，如果硬币质地均匀，抛硬币时获得正面和反面的可能性相同。这个可能性，就是上一节提到的概率。

对于图 11 (a)，当登山者位于红色点 \bullet ，他通过抛一枚硬币决定向上走和向下走的概率（可能性）相同，均为 0.5 (50%)。

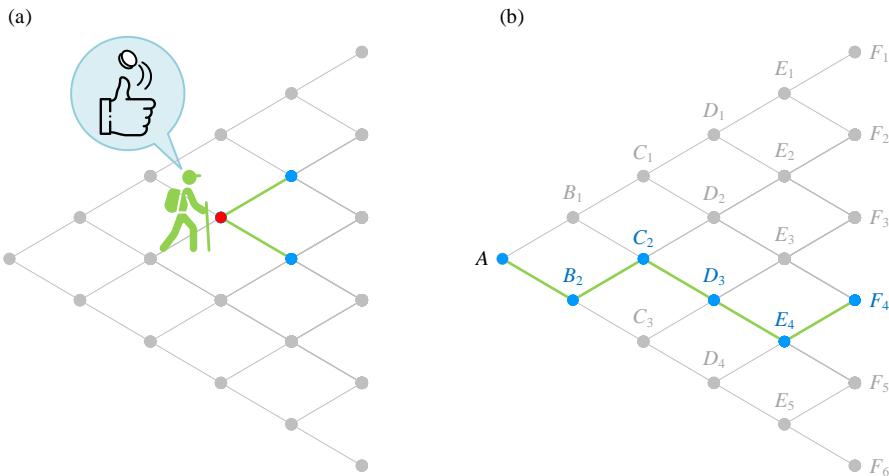


图 11. 二叉树路径与可能性

等可能角度

通过本章前文学习，大家已经清楚图 11 (b) 二叉树一共有 32 条路径。显然，从初始点到某一特定终点节点，登山者采用任意路径的可能性相同。也就是说图 11 (b) 中 $A \rightarrow B_2 \rightarrow C_2 \rightarrow D_3 \rightarrow E_4 \rightarrow F_4$ 这条路径被采纳的概率（可能性）为：

$$\Pr(A \rightarrow B_2 \rightarrow C_2 \rightarrow D_3 \rightarrow E_4 \rightarrow F_4) = \frac{1}{32} = 0.03125 = 3.125\% \quad (10)$$

二选一角度

再换一个角度，登山者在 A 、 B 、 C 、 D 、 E 这 5 个节点都面临二选一的抉择，而选择向上或向下的概率均为 $1/2$ ；因此，登山者选择图 11 (b) 中 $A \rightarrow B_2 \rightarrow C_2 \rightarrow D_3 \rightarrow E_4 \rightarrow F_4$ 路径的概率为：

$$\Pr(A \rightarrow B_2 \rightarrow C_2 \rightarrow D_3 \rightarrow E_4 \rightarrow F_4) = \left(\frac{1}{2}\right)^5 = \frac{1}{32} = 0.03125 = 3.125\% \quad (11)$$

结果和 (10) 完全一致。

组合数

图 11 (b) 二叉树从起点 A 到终点 ($F_1 \sim F_6$) 一共有 32 条路径，而到达 F_4 点一共有 10 条路径。也就是说从 A 点出发，最终到达 F_4 点的概率为：

$$\Pr(F_4) = \frac{C_5^3}{2^5} = \frac{10}{32} = 0.3125 = 31.25\% \quad (12)$$

同理，我们可以计算得到到达 F_1 、 F_2 、 F_3 、 F_5 、 F_6 这几个终点的概率：

$$\begin{aligned}
 \Pr(F_1) &= \frac{C_5^0}{2^5} = \frac{1}{32} = 0.03125 \\
 \Pr(F_2) &= \frac{C_5^1}{2^5} = \frac{5}{32} = 0.15625 \\
 \Pr(F_3) &= \frac{C_5^2}{2^5} = \frac{10}{32} = 0.3125 \\
 \Pr(F_4) &= \frac{C_5^3}{2^5} = \frac{10}{32} = 0.3125 \\
 \Pr(F_5) &= \frac{C_5^4}{2^5} = \frac{5}{32} = 0.15625 \\
 \Pr(F_6) &= \frac{C_5^5}{2^5} = \frac{1}{32} = 0.03125
 \end{aligned} \tag{13}$$

举个例子，从 A 点出发，不管中间走那条路线，到达 F_2 的概率为 15.625%。

这些概率值求和，得到结果为 1；这就是说，按照既定规则，登山者从起点出发，必然到达终点。1 量化了“必然”这一论述：

$$\begin{aligned}
 \left(\frac{1}{2} + \frac{1}{2}\right)^5 &= C_5^0\left(\frac{1}{2}\right)^5 + C_5^1\left(\frac{1}{2}\right)^5 + C_5^2\left(\frac{1}{2}\right)^5 + C_5^3\left(\frac{1}{2}\right)^5 + C_5^4\left(\frac{1}{2}\right)^5 + C_5^5\left(\frac{1}{2}\right)^5 \\
 &= \frac{1}{32} + \frac{5}{32} + \frac{10}{32} + \frac{10}{32} + \frac{5}{32} + \frac{1}{32} \\
 &= 0.03125 + 0.15625 + 0.3125 + 0.3125 + 0.15625 + 0.03125 = 1
 \end{aligned} \tag{14}$$

概率直方图

将上述概率值做成水平条形图，放在二叉树路径的右侧，我们得到图 12。

这种直方图被称作**概率直方图** (probability histogram)。大家可能已经发现，图 9 所示的频数直方图结果除以总数 32，就得到图 12 这幅概率直方图。也就是说，频数直方图和概率直方图可以很容易相互转化。注意，直方图可以展示频数、概率值、概率密度值。概率密度积分后得到概率值。

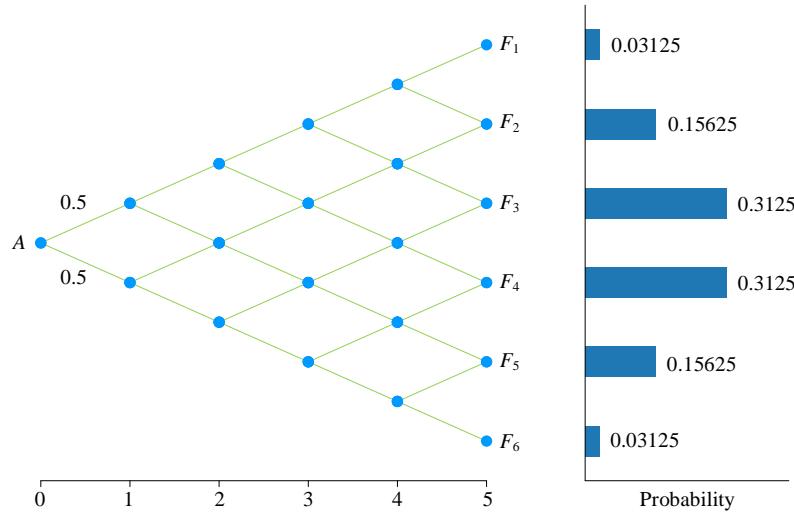


图 12. $n = 5$ ，到达二叉树终点节点概率分布，向上、向下概率均为 0.5

20.5 一枚质地不均匀的硬币

前文假设硬币质地均匀，即抛一枚硬币获得正面背面朝上的概率相同，均为 0.5 (50%)；但是，假设一种情况，硬币质地不均匀，抛这枚硬币时，得到正面的可能性为 60%，反面的可能性为 40%。

下面计算一下抛这枚硬币决定在图 11 所示二叉树中登山者从起点到达终点的选取不同路径的可能性。

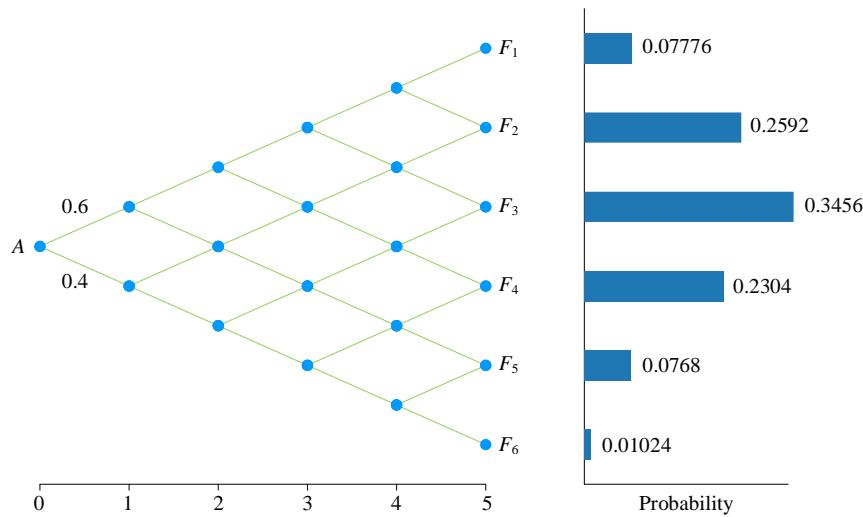
在五次“二选一”的决策中，向上走的可能性为 0.6，向下走的可能性为 0.4，利用组合数容易得到，到达 F_1 、 F_2 、 F_3 、 F_4 、 F_5 、 F_6 对应的概率分别为：

$$\begin{aligned}\Pr(F_1) &= C_5^0 \times 0.6^5 \times 0.4^0 = 0.07776 \\ \Pr(F_2) &= C_5^1 \times 0.6^4 \times 0.4^1 = 0.2592 \\ \Pr(F_3) &= C_5^2 \times 0.6^3 \times 0.4^2 = 0.3456 \\ \Pr(F_4) &= C_5^3 \times 0.6^2 \times 0.4^3 = 0.2304 \\ \Pr(F_5) &= C_5^4 \times 0.6^1 \times 0.4^4 = 0.0768 \\ \Pr(F_6) &= C_5^5 \times 0.6^0 \times 0.4^5 = 0.01024\end{aligned}\tag{15}$$

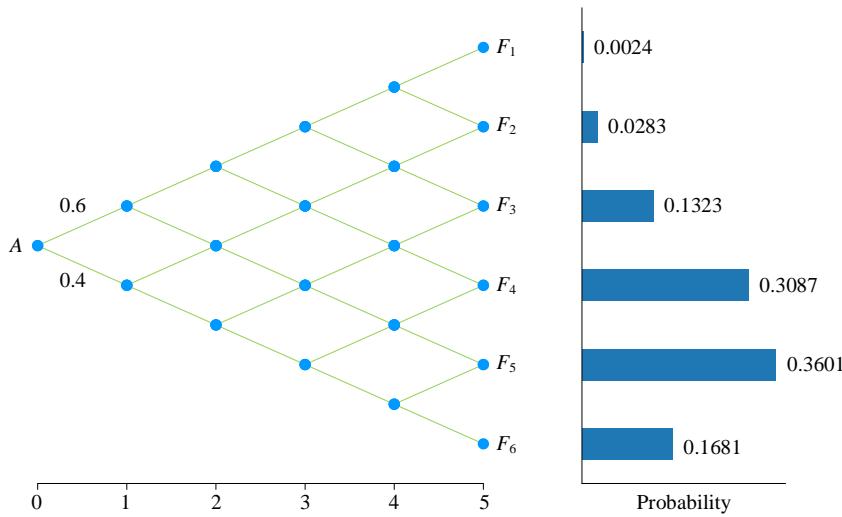
到达 F_1 、 F_2 、 F_3 、 F_4 、 F_5 、 F_6 对应的概率之和仍然为 1：

$$\begin{aligned}(0.6 + 0.4)^5 &= \underbrace{C_5^0 \times 0.6^5 \times 0.4^0}_{\Pr(F_1)} + \underbrace{C_5^1 \times 0.6^4 \times 0.4^1}_{\Pr(F_2)} + \underbrace{C_5^2 \times 0.6^3 \times 0.4^2}_{\Pr(F_3)} + \\ &\quad \underbrace{C_5^3 \times 0.6^2 \times 0.4^3}_{\Pr(F_4)} + \underbrace{C_5^4 \times 0.6^1 \times 0.4^4}_{\Pr(F_5)} + \underbrace{C_5^5 \times 0.6^0 \times 0.4^5}_{\Pr(F_6)} \\ &= 0.07776 + 0.2592 + 0.3456 + 0.2304 + 0.0768 + 0.01024 = 1\end{aligned}\tag{16}$$

但是对比图 12 和图 13，容易发现登山者倾向于“向上走”；这显然是因为硬币不均匀，抛硬币得到正面的概率高于反面。而且图 13 右侧的概率直方图不再对称。

图 13. $n = 5$, 到达二叉树终点节点概率分布, 向上、向下概率分别为 0.6、0.4

如果我们恰好能够找到另外一枚质地不均匀的硬币, 抛这枚硬币时, 得到正面的可能性为 30%, 反面的可能性为 70%。登山者通过抛这枚硬币确定向上走或向下走, 如图 14 所示, 登山者更倾向于向下走。

图 14. $n = 5$, 到达二叉树终点节点概率分布, 向上、向下概率分别为 0.3、0.7

这一节的内容, 实际上就是我们要在丛书《概率统计》一本中要讲解的**二项式分布**(binomial distribution)。概率是数据科学和机器学习中重要的板块, 本系列丛书《概率统计》一本将全面讲解。



代码文件 Bk3_Ch20_1.py 中 Bk3_Ch20_1_B 部分绘制图 12、图 13、图 14。请读者修改代码中 p 值。



在 Bk3_Ch20_1.py 基础上，我们做了一个 App 展示不同概率值对到达终点不同点概率的影响。请参考 Streamlit_Bk3_Ch20_1.py。

20.6 随机中有规律

本节还是用二叉树来探讨随机和确定之间的辩证关系。

在给定的二叉树网格中，登山者在不同节点“随机”确定向上走、向下走，得到的结果就是一种**随机漫步** (random walk)。

图 15 所示为 20 步二叉树网格，根据前文所学，我们知道从起点到终点，这个网格对应 2^{20} (1048576) 条路径。图 15 四幅图给出的是登山者“可能”走的 2、4、8、16 条随机路径。

随着路径数量增多，我们似乎可以预感，到达终点时登山者在中间的可能性会高于两端。

为了验证这一直觉，并相对准确确定登山者到达终点位置的规律，我们不断增加随机路径的数量，并根据终点位置绘制频率直方图。如图 16 所示，50、100、5000 条随机路径条件下，登山者终点位置概率直方图。

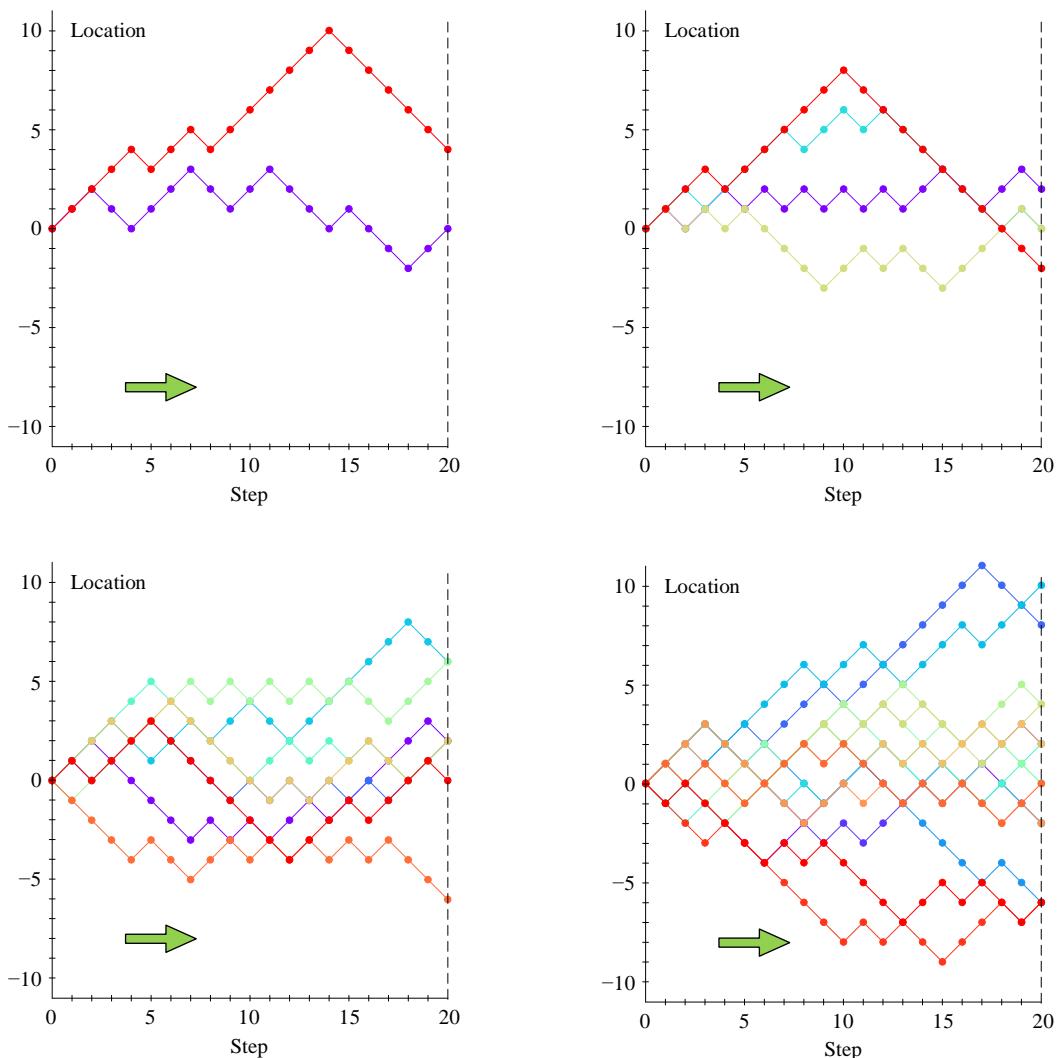


图 15. 随机漫步，2、4、8、16 条路径

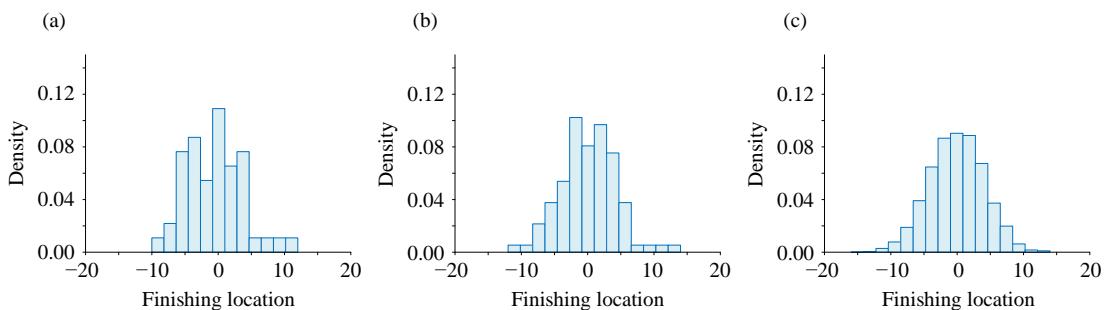


图 16. 随机漫步结束位置概率直方图，50、100、5000 条路径，纵轴代表概率密度

实际上，二叉树网格限制了登山者向上或向下运动的步幅。更进一步，如果我们放开二叉树网格的限制，让登山者按照某种规律自行决定向上或向下的步幅，就可以得到图 17。

单看图中任意一条或几条路径，我们很难抓住任何规律；但是随着随机路径的数量不断增多，运动的规律就不言自明了。

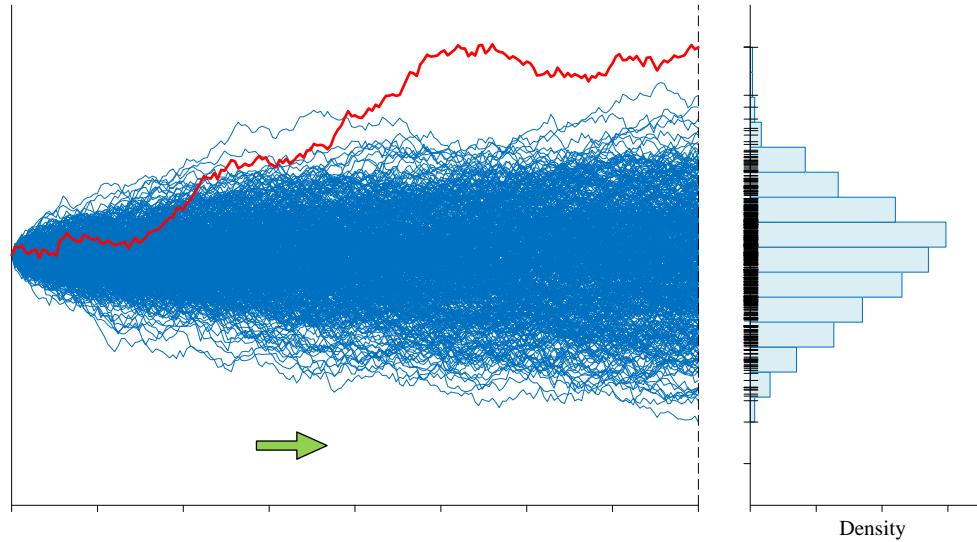


图 17. 不受二叉树网格限制的随机漫步

生活中这种随机中存在规律的情况不胜枚举。

举个例子，图 18 左图所示为一段时间内某只股票日收益率，红线以上为股价上涨，红线以下为股价下跌。单看某几天的股价涨跌很难把握住规律。但是，把一段时间内股价的日收益率数据绘制成直方图，如图 18 右图，我们就可以发现股价涨跌规律的端倪。

当然，为了得出更有意义的结论，我们还需要掌握更多的概率统计工具。本系列丛书将在《概率统计》和《数据科学》两本书中介绍更多概率统计知识，以及如何将它们应用到数据分析和预测实践中。

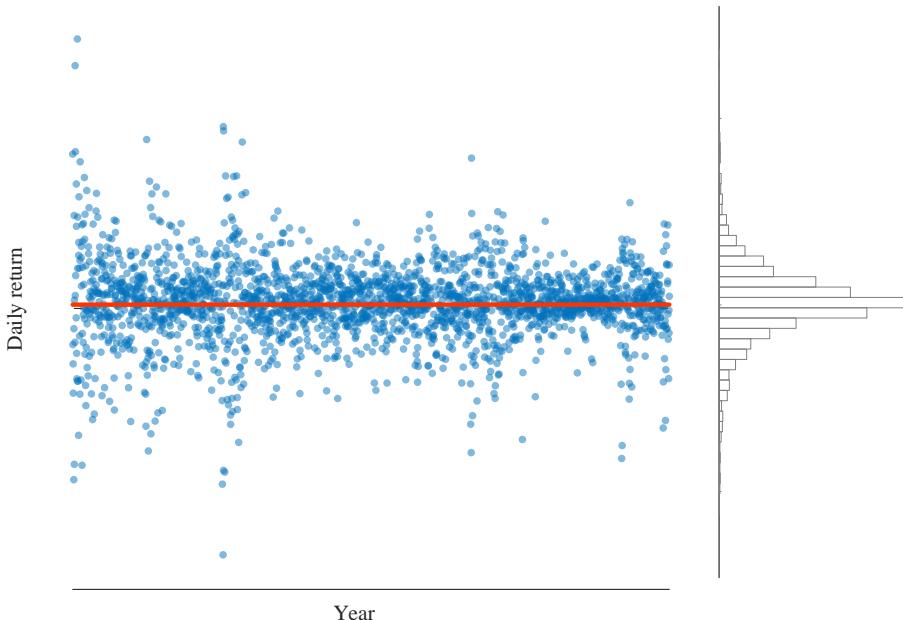


图 18. 股价日收益率和一段时间内的分布情况

高斯分布

观察图 16、图 17、图 18 直方图，似乎某种神秘的规律，或者一条神秘的曲线，呼之欲出。这就是“宇宙终极分布”——**高斯分布** (Gaussian distribution)。

高斯分布是众多概率分布中较为常用的一种。所谓**概率分布** (probability distribution) 描述随机变量取值的概率规律。

下式是高斯分布的**概率密度函数** (probability density function, PDF) 曲线解析式：

$$f_x(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (17)$$

其中， μ 为均值， σ 为标准差；下一章将介绍均值和标准差。

满足 (17) 的高斯分布常记做 $N(\mu, \sigma^2)$ 。连续型随机变量的概率密度函数 PDF 描述随机变量的在某个确定的取值点附近的可能性的函数。

(17) 实际上就是本书第 12 章介绍过的高斯函数通过函数变换得到的解析式。

图 19 所示为三个不同参数的一元高斯分布概率密度函数曲线。高斯分布，形态上极富美感；公式优雅精巧，包含数学中两个重要两个无理数 π 和 e 。高斯分布可以解释自然界很多纷繁复杂的规律；有人说，高斯分布似乎代表着宇宙幕后终极秩序。



本系列丛书《概率统计》一册将深入介绍高斯分布。

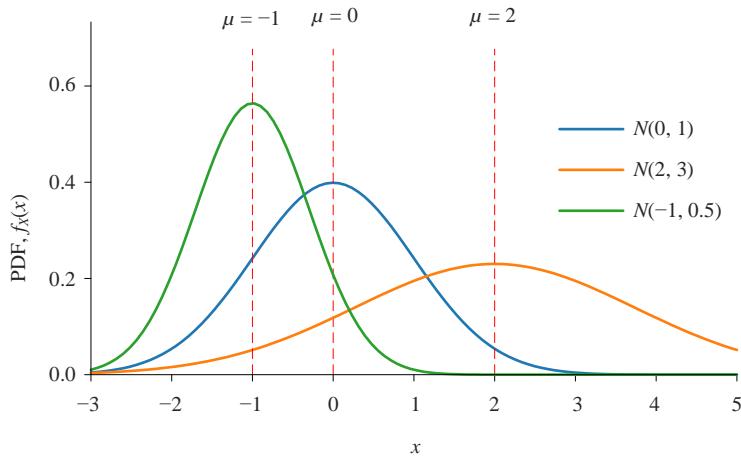


图 19. 三个不同一元高斯分布的概率密度函数曲线



本书前文利用杨辉三角，将算数、代数、几何、数列等数学知识联系起来，本章又将杨辉三角的触角伸到二叉树、概率和随机等概念；这正是是丛书的重要目的之一——打破数学板块之间的壁垒，将它们有机联结。

希望大家通过本章的学习能够获得有关概率和随机的直观感受。随着，本系列丛书内容的不断深入，大家不但能够获得解释随机现象的数学工具，还能将它们用在解决数据科学和机器学习具体问题中去。

21

Fundamentals of Statistics

统计入门

以鸢尾花数据为例



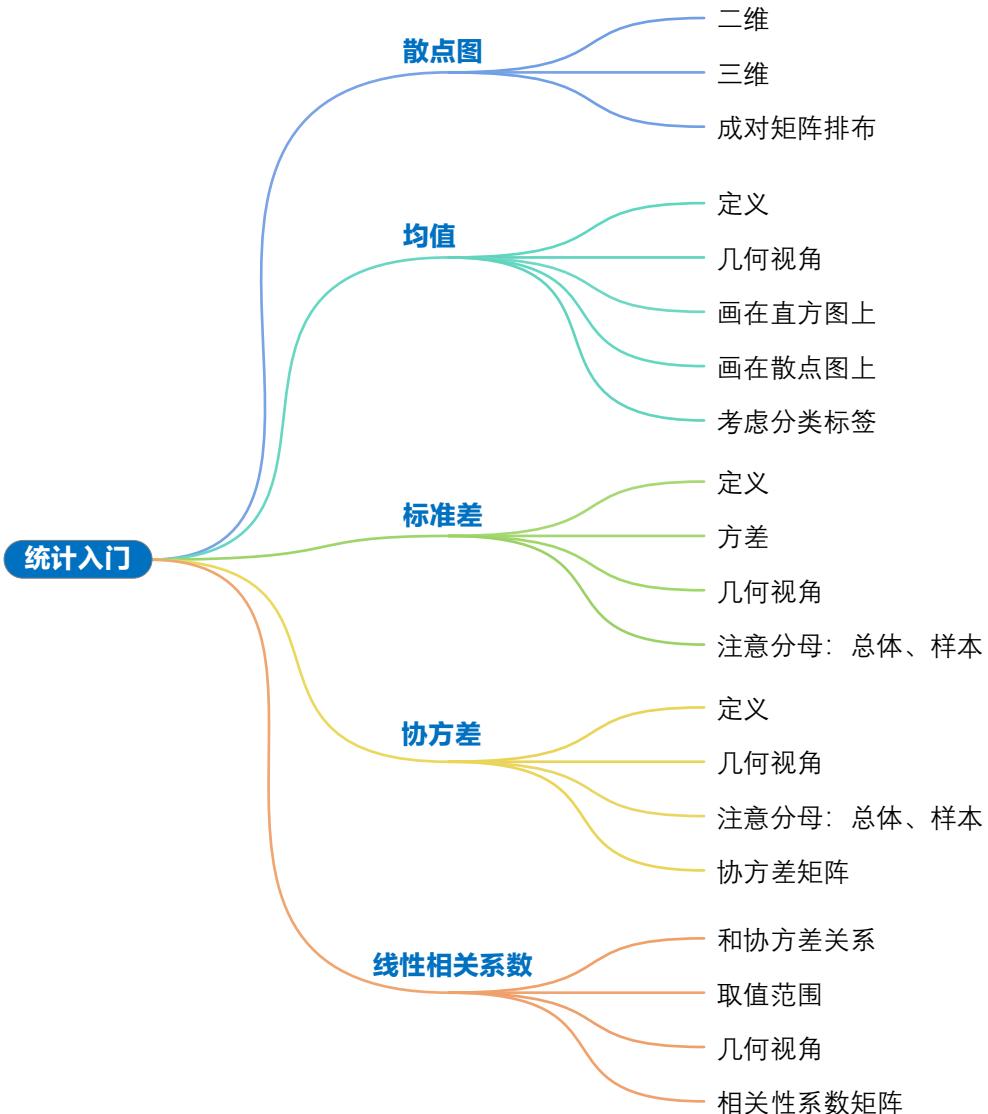
有朝一日，对于所有人，统计思维就像读写能力一样重要。

Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write.

—— 赫伯特·乔治·威尔斯 (H. G. Wells) | 英国科幻小说家 | 1866 ~ 1946



- ◀ seaborn.heatmap() 绘制热图
- ◀ seaborn.histplot() 绘制频率/概率直方图
- ◀ seaborn.pairplot() 绘制对分析图
- ◀ seaborn.lineplot() 绘制线图



21.1 统计的前世今生：强国知十三数

现在，“概率”和“统计”两个词如影随形。统计搜集、整理、分析、研究数据，从而寻找规律。概率论是统计推断的基础。基于特定条件，概率量化事件的可能性。

现代统计学的主要数学基础是概率论；但是，统计的出现远早于概率。通过上一章学习，我们了解了概率出生草莽；但是，统计学却是衔玉而生。

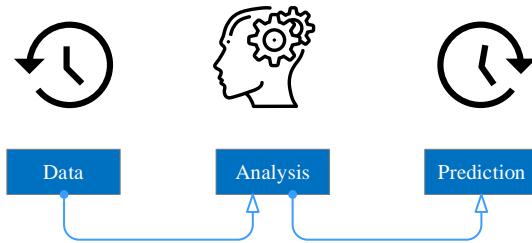


图 1. 统计和概率关系

统计学的初衷就是为国家管理提供可靠数据。英语中 statistics 是源于现代拉丁语 statisticum collegium (国会)。

战国思想家商鞅 (390 BC ~ 338 BC) 提出“强国知十三数”，他为秦国制定的统计内容包含“十三数”——“竟内仓、口之数，壮男、壮女之数，老、弱之数，官、士之数，以言说取食者之数，利民之数，马、牛、刍藁之数。欲强国，不知国十三数，地虽利，民虽众，国愈弱至削。”

简单说，商鞅认为和国家存亡攸关的统计数字包括粮仓、金库、壮年男子、壮年女子、老年人、体弱者、官吏、士卒、游说者、工商业者、牲畜和饲料。刍藁 (chú gǎo) 为饲养牲畜的草料。

商鞅强调统计数字对王朝兴亡至关重要。他说，“数者，臣主之术而国之要也。故万国失数而国不危，臣主失数而不乱者，未之有也。”大意是，统计数字是治国之术和国家根本；没有统计数字，君主便无法治国理政，国家就要危乱。

阿拉伯学者肯迪 (Al-Kindi, 801 ~ 873) 创作的《密码破译》 (*Manuscript on Deciphering Cryptographic Messages*) 书中，介绍如何使用统计数据和频率分析进行密码破译。肯迪和本书前文介绍的花拉子密 (Muhammad ibn Musa al-Khwarizmi) 都供职于巴格达“智慧宫” (House of Wisdom)。

英国经济学家约翰·葛兰特 (John Graunt, 1620 ~ 1674) 在 1663 年发表了《对死亡率表的自然与政治观察》 (*Natural and Political Observations Made Upon the Bills of Mortality*)，被誉为人口统计学的开山之作，他本人也常被称作“人口统计学之父”。

本章内容以鸢尾花数据为例，用最少的公式，尽量从几何可视化视角给大家介绍统计的入门知识。

21.2 散点图：当数据遇到坐标系

本书第1章以表格的形式介绍过鸢尾花数据。有了坐标系，类似鸢尾花这样的样本数据就可以在纸面飞跃。

本节介绍样本数据重要的可视化方案之一——**散点图** (scatter plot)。散点图将二维样本数据以点的形式展现在直角坐标系上。

图2(a)所示为鸢尾花数据中花萼长度和花萼宽度两个特征的散点图。散点图中每一个点代表一朵鸢尾花，横坐标值代表花萼长度，纵坐标值代表花萼宽度。

我们知道鸢尾花数据集一共有150个数据点，分成3大类，也就是对应3个不同的标签。在图2(a)散点图基础上，用不用颜色区分分类标签，我们可以得到图2(b)。

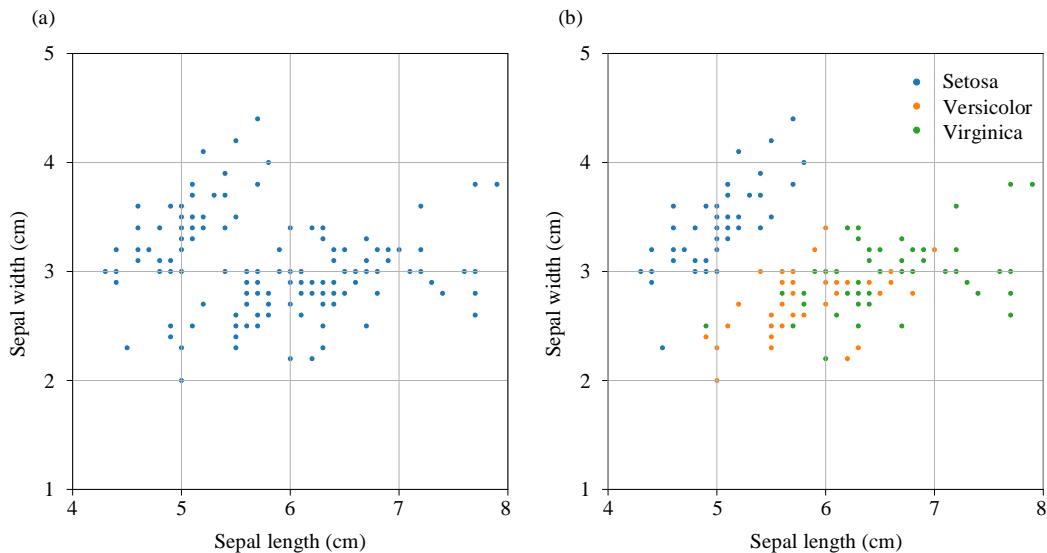


图2. 花萼长度、花萼宽度特征数据散点图

我们也可以在三维直角坐标系中绘制散点图。图3(a)所示为花萼长度、花萼宽度、花瓣长度三个特征的散点图。

在图3(a)基础上，如果加上分类标签，我们可以得到图3(b)。

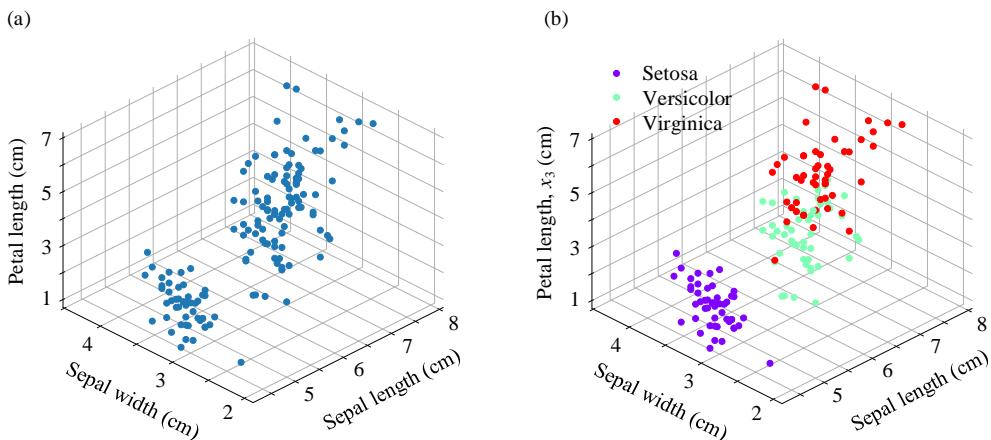


图 3. 花萼长度、花萼宽度、花瓣长度散点图

成对特征散点图

大家可能会问，鸢尾花有 4 个特征（花萼长度、花萼宽度、花瓣长度、花瓣宽度）；有没有什么可视化方案能够展示所有的特征？

答案是成对特征散点图。

如图 4 所示，16 幅子图被安排成 4×4 矩阵的形式。其中，12 幅散点图为成对特征关系，对角线上的 4 幅图像叫做**概率密度估计** (probability density estimation) 曲线。

简单来说，概率密度估计曲线展示数据分布情况，类似于上一章介绍的频率直方图。本系列丛书《概率统计》一册将专门讲解概率密度估计。

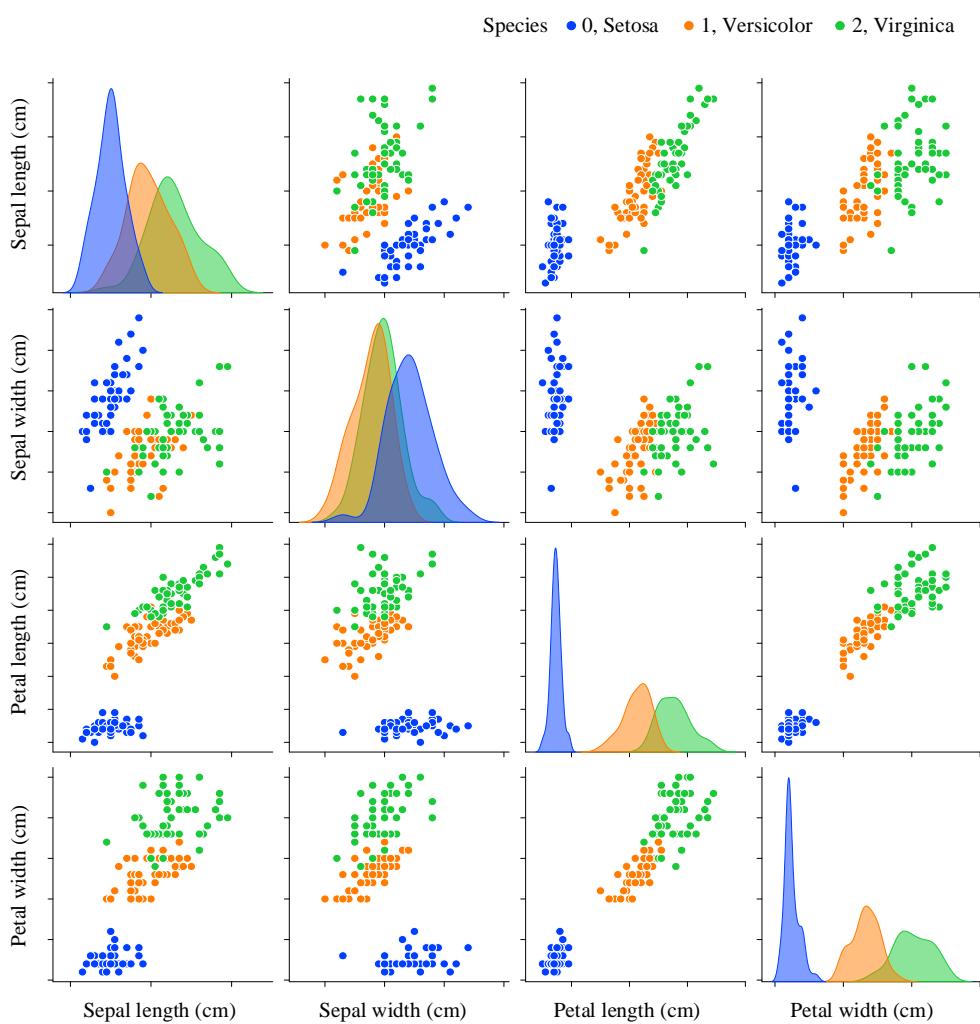


图 4. 鸢尾花数据成对特征散点图，考虑分类标签

散点图的作用

利用散点图，我们可以发现数据的集中、分布程度，比如数据主要集中在哪些区域。

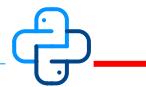
散点图也会揭示不同特征之间可能存在的量化关系，比如图 4 中花瓣长度和宽度数据关系似乎能够用一条直线来表达。这就是**线性回归** (linear regression) 的思路。

此外，我们还可以利用散点图发现数据是否存在离群值。**离群值** (outlier) 指的是，和其他数据相比，数据中有一个或几个样本数值差异较大。



本系列丛书《数据科学》一册将讲解发现数据中离群值的常用算法。

本节采用可视化的方式来描绘数据，实际应用中，我们经常需要量化数据的集中、分散程度，以及不同特征之间的关系。这就需要大家了解均值、方差、标准差、协方差、相关性这些概念。这是本章后续要介绍的内容。



代码文件 Bk3_Ch21_1.py 中 Bk3_Ch21_1_A 部分绘制本节图像。

21.3 均值：集中程度

大家对均值这个概念应该不陌生。

均值 (average 或 mean)，也叫平均值，**算数平均数** (arithmetic average 或 arithmetic mean)。均值代表一组数据集中趋势。

均值对应的运算是，一组数据中所有数据先求和，再除以这组数据的个数。比如鸢尾花萼特征数据 $\{x_1^{(1)}, x_1^{(2)}, \dots, x_1^{(150)}\}$ 有 150 个值，它们的平均值为：

$$\mu_1 = \frac{1}{n} \left(\sum_{i=1}^n x_1^{(i)} \right) = \frac{x_1^{(1)} + x_1^{(2)} + \dots + x_1^{(150)}}{150} \quad (1)$$

从几何角度，如图 5 所示，算数平均值相当于找到一个平衡点。

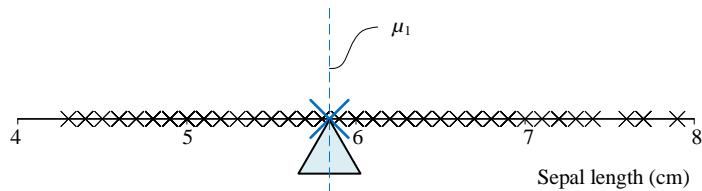


图 5. 均值相当于找到数据的平衡点

以鸢尾花为例，它的样本数据在花萼长度、花萼宽度、花瓣长度和花瓣宽度四个特征的均值分别为：

$$\mu_1 = 5.843, \mu_2 = 3.057, \mu_3 = 3.758, \mu_4 = 1.199 \quad (2)$$

图 6 所示为鸢尾花四个特征均值在频数直方图位置。

⚠ 注意在计算这四个均值时，我们并没有考虑鸢尾花的分类标签。

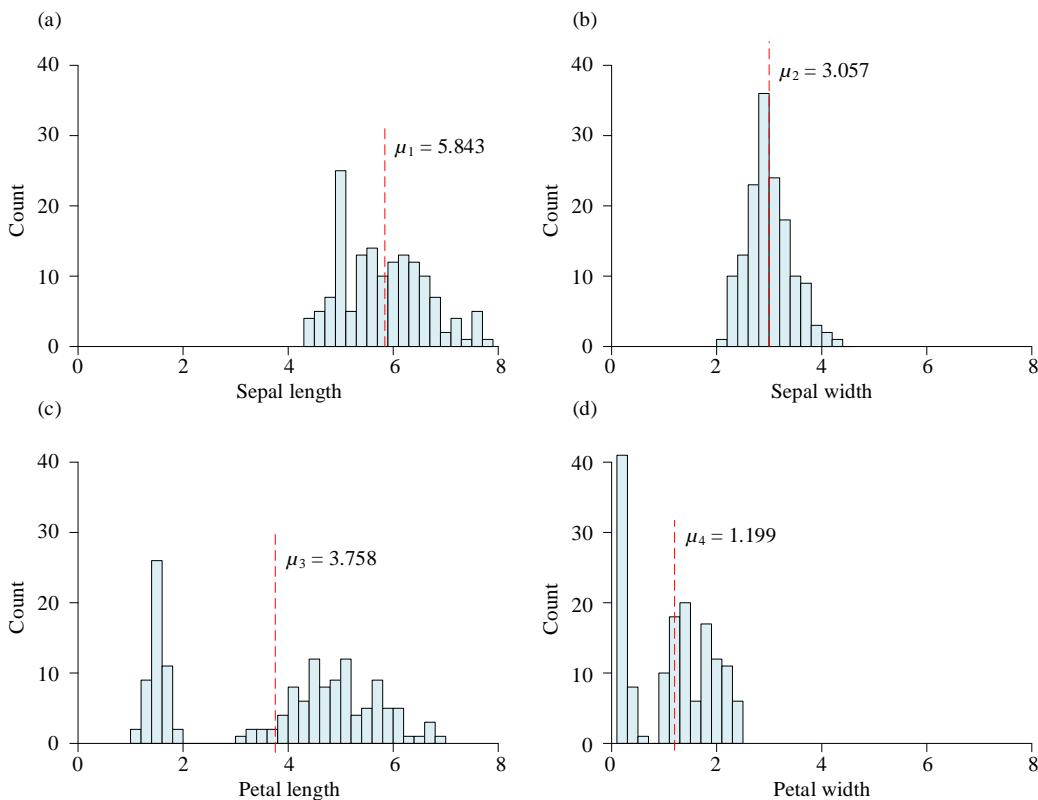


图 6. 鸢尾花四个特征数据均值在直方图位置

考虑分类

当然，我们在计算均值的时候，也可以考虑分类。

以鸢尾花数据为例，很多应用场合需要计算满足某个条件的均值，比如标签为 virginica 样本数据的花萼长度。

在图 4 基础上，我们可以三类不同标签条件下样本数据均值位置可视化，这样便得到图 7。图 7 中 \times 、 \times 、 \times 分别代表 setosa、versicolor、virginica 三个不同标签均值的位置。

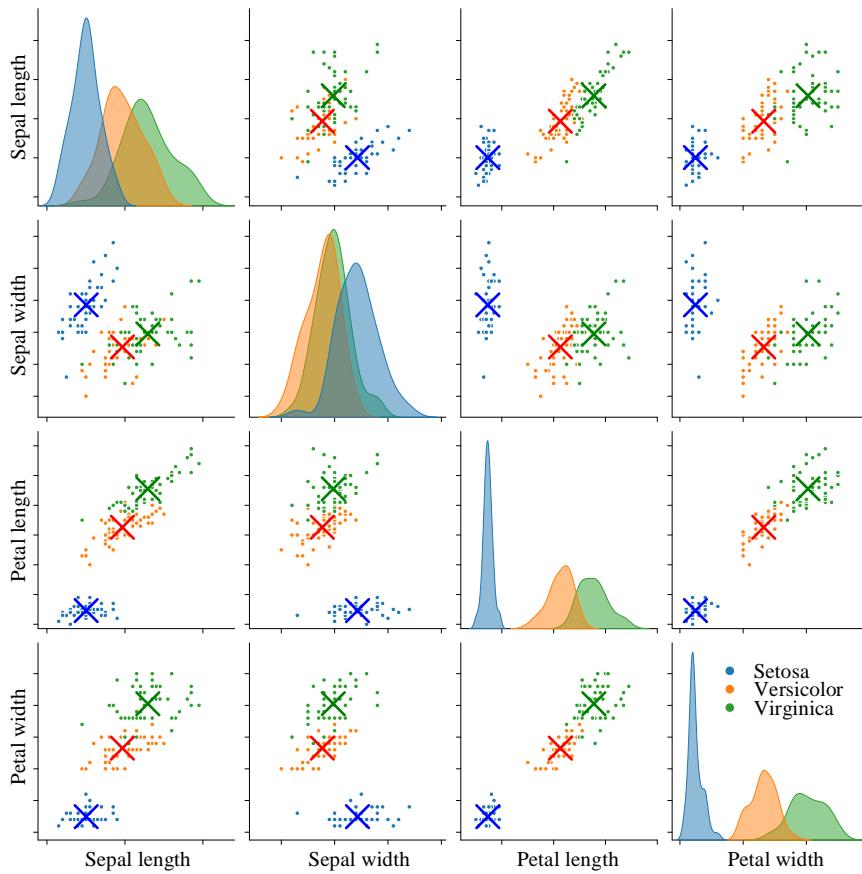


图 7. 均值在散点图上的位置，考虑分类标签



代码文件 Bk3_Ch21_1.py 中 Bk3_Ch21_1_B 部分计算均值并绘制图 6。

21.4 标准差：离散程度

标准差 (standard deviation) 描述一组数值以均值 μ 为基准的分散程度。如果数据为样本，比如鸢尾花数据 $\{x_1^{(1)}, x_1^{(2)}, \dots, x_1^{(150)}\}$ 标准差为：

$$\sigma_1 = \sqrt{\frac{1}{150-1} \sum_{i=1}^{150} (x_1^{(i)} - \mu_1)^2} \quad (3)$$

⚠ 注意，(3) 根号内分式的分母为 $(150 - 1)$ ，不是 150 。

标准差的平方为**方差**(variance)：

$$\text{var}(X_1) = \sigma_1^2 = \frac{1}{150-1} \sum_{i=1}^{150} (x_1^{(i)} - \mu_1)^2 \quad (4)$$

如图8所示， $x_1^{(i)} - \mu_1$ 代表 $x_1^{(i)}$ 和 μ_1 距离；而 $(x_1^{(i)} - \mu_1)^2$ 代表以 $|x_1^{(i)} - \mu_1|$ 为边长正方形的面积。

(4) 相当于这些正方形面积求平均值。

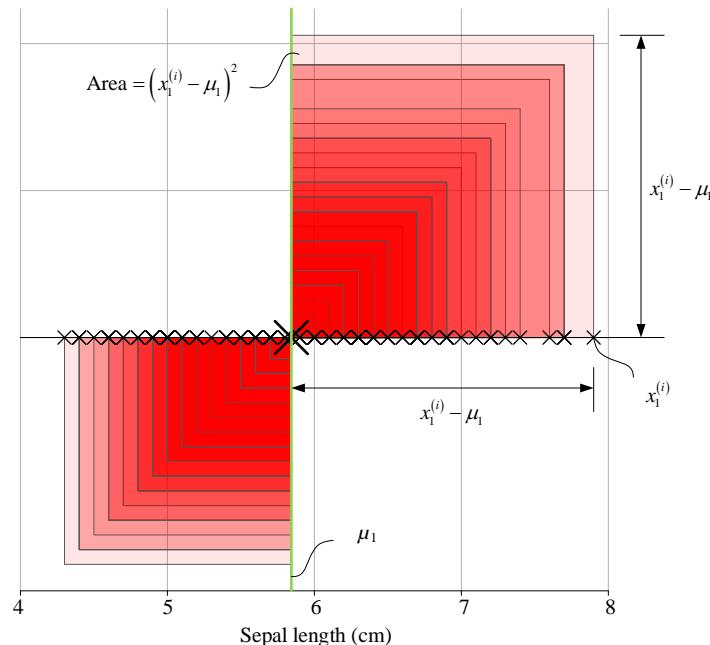


图 8. 几何视角看方差

⚠ 注意，标准差的单位和样本数据相同；但是，方差的单位是样本数据单位的平方。比如，鸢尾花花萼长度的单位是厘米 cm，因此这个特征上样本数据的标准差对应的单位也是厘米 cm，而方差的单位是平方厘米 cm²。所以在同一幅图上，我们常会看到 μ 、 $\mu \pm \sigma$ 、 $\mu \pm 2\sigma$ 、 $\mu \pm 3\sigma$ 等。

计算鸢尾花样本数据四个特征的标准差：

$$\sigma_1 = 0.825, \sigma_2 = 0.434, \sigma_3 = 1.759, \sigma_4 = 0.759 \quad (5)$$

上式这些数值的单位都是厘米 cm。

图 9 所示为鸢尾花四个特征数据均值 μ 、标准差 $\mu \pm \sigma$ 在频数直方图位置。

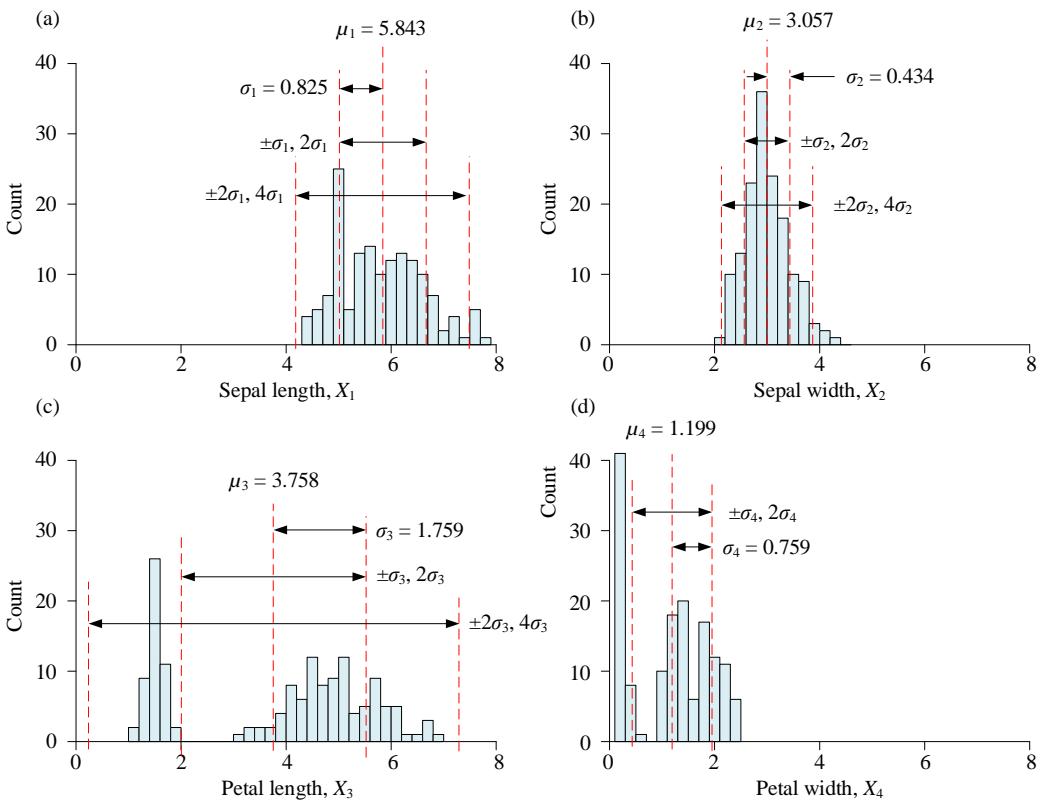
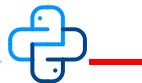


图 9. 鸢尾花四个特征数据均值、标准差在直方图位置



代码文件 Bk3_Ch21_1.py 中 Bk3_Ch21_1_C 部分计算标准差并绘制图 9。

21.5 协方差：联合变化程度

协方差 (covariance) 描述的是随机变量联合变化程度。白话讲，以图 4 中花瓣长度和宽度数据关系为例，我们发现如果样本数据的花瓣长度越长，其花瓣宽度很大可能也越宽。这就是联合变化。而协方差以量化的方式来定量分析这种联合变化程度。

定义第 i 朵花的花萼长度和花萼宽度的取值为 $(x_1^{(i)}, x_2^{(i)})$ ($i = 1, \dots, 150$)，花萼长度和宽度的协方差为：

$$\text{cov}(X_1, X_2) = \frac{1}{150-1} \sum_{i=1}^{150} (x_1^{(i)} - \mu_1)(x_2^{(i)} - \mu_2) \quad (6)$$

如图 10 所示，从几何视角， $(x_1^{(i)} - \mu_1)(x_2^{(i)} - \mu_2)$ 相当于以 $(x_1^{(i)} - \mu_1)$ 和 $(x_2^{(i)} - \mu_2)$ 为边的矩形面积。

⚠ 注意这个面积有正负。

当 $(x_1^{(i)} - \mu_1)$ 和 $(x_2^{(i)} - \mu_2)$ 同号，面积为正，对应图 10 中红色矩形。也就是说，红色矩形越多说明，花萼长度越长，花萼宽度越宽；或者，花萼长度越短，花萼宽度越窄。

当 $(x_1^{(i)} - \mu_1)$ 和 $(x_2^{(i)} - \mu_2)$ 异号，面积为负，对应图 10 中蓝色矩形。蓝色矩形越多说明，花萼长度越长，花萼宽度越窄；花萼长度越短，花萼宽度越长。

这些矩形的面积的平均值便是协方差。同样在计算协方差时，对于样本，分母为 $n - 1$ ；对于总体，分母为 n 。

可以这样理解，当 X_1 和 X_2 联合变化越强，某个颜色（红色或蓝色）矩形面积之和越大；当 X_1 和 X_2 联合变化弱的时候，红色和蓝色矩形面积之和越趋向于 0，也就是颜色越“平衡”。

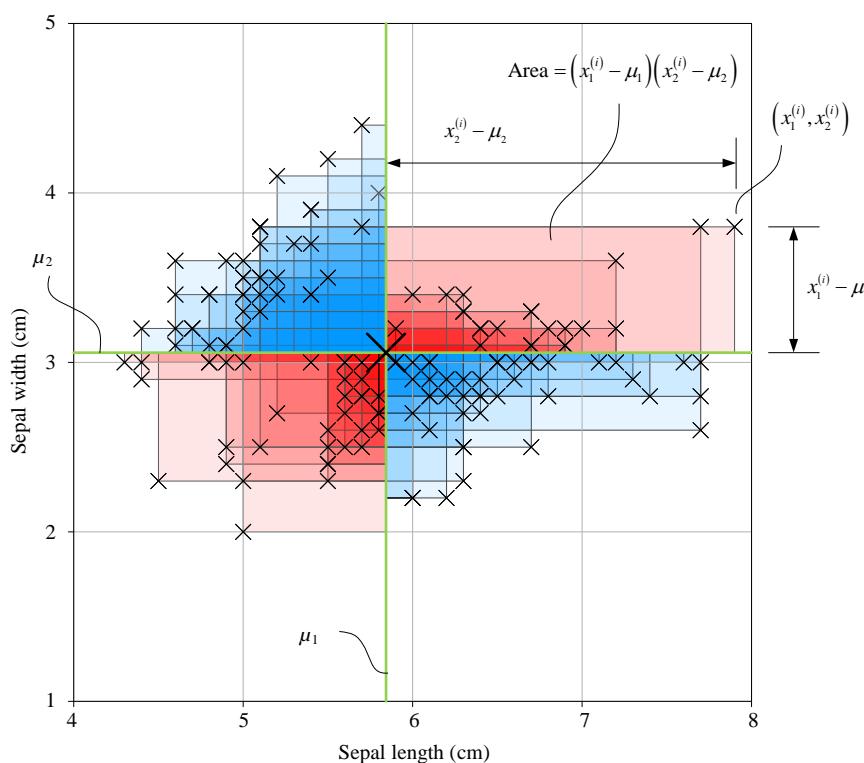


图 10. 几何视角看协方差

协方差矩阵

以鸢尾花为例，对于不同成对的特征，我们可以获得如下 6（对应组合数 C_4^2 ）个协方差值：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$\begin{aligned}
 \text{cov}(X_1, X_2) &= -0.042 \\
 \text{cov}(X_1, X_3) &= 1.274 \\
 \text{cov}(X_1, X_4) &= 0.516 \\
 \text{cov}(X_2, X_3) &= -0.330 \\
 \text{cov}(X_2, X_4) &= -0.122 \\
 \text{cov}(X_3, X_4) &= 1.296
 \end{aligned} \tag{7}$$

可以想象，如果我们有更多的特征，成对协方差值不计其数。整理和储存这些数据需要很好的结构。矩阵就是最好的解决办法。

由方差和协方差构成的矩阵叫做**协方差矩阵** (covariance matrix)，也叫方差-协方差矩阵 (variance-covariance matrix)。

以鸢尾花四个特征为例，这个协方差矩阵为 4×4 矩阵：

$$\boldsymbol{\Sigma} = \begin{bmatrix} \text{cov}(X_1, X_1) & \text{cov}(X_1, X_2) & \text{cov}(X_1, X_3) & \text{cov}(X_1, X_4) \\ \text{cov}(X_2, X_1) & \text{cov}(X_2, X_2) & \text{cov}(X_2, X_3) & \text{cov}(X_2, X_4) \\ \text{cov}(X_3, X_1) & \text{cov}(X_3, X_2) & \text{cov}(X_3, X_3) & \text{cov}(X_3, X_4) \\ \text{cov}(X_4, X_1) & \text{cov}(X_4, X_2) & \text{cov}(X_4, X_3) & \text{cov}(X_4, X_4) \end{bmatrix} \tag{8}$$

协方差矩阵为方阵。矩阵中对角线上元素为方差。

也就是说，某个随机变量和自身求协方差，得到的就是方差，比如下例：

$$\text{cov}(X_1, X_1) = \text{var}(X_1) \tag{9}$$

协方差矩阵中非对角线上元素为协方差。容易知道，下式成立：

$$\text{cov}(X_i, X_j) = \text{cov}(X_j, X_i) \tag{10}$$

这就解释了为什么协方差矩阵为对称矩阵。

对于鸢尾花数据，它的协方差矩阵 $\boldsymbol{\Sigma}$ 具体值为：

$$\boldsymbol{\Sigma} = \begin{bmatrix} 0.686 & -0.042 & 1.274 & 0.516 \\ -0.042 & 0.190 & -0.330 & -0.122 \\ 1.274 & -0.330 & 3.116 & 1.296 \\ 0.516 & \underbrace{-0.122}_{\text{Sepal width, } X_2} & 1.296 & 0.581 \end{bmatrix} \leftarrow \begin{array}{l} \text{Sepal length, } X_1 \\ \text{Sepal width, } X_2 \\ \text{Petal length, } X_3 \\ \text{Petal width, } X_4 \end{array} \tag{11}$$

图 14 所示为鸢尾花数据协方差矩阵热图。

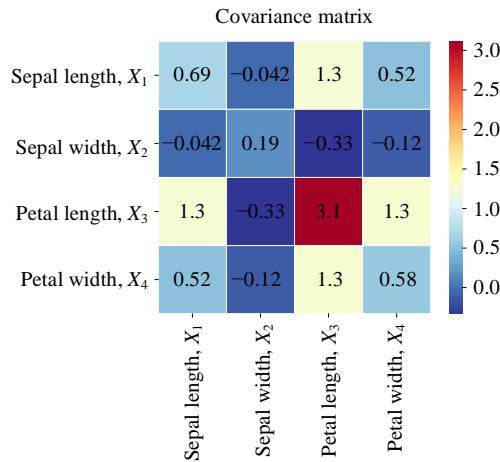


图 11. 鸢尾花数据协方差矩阵热图

考虑标签

当然，在计算协方差时，我们也可以考虑到数据标签。图 12 所示为三个不同标签数据各自的协方差矩阵热图。

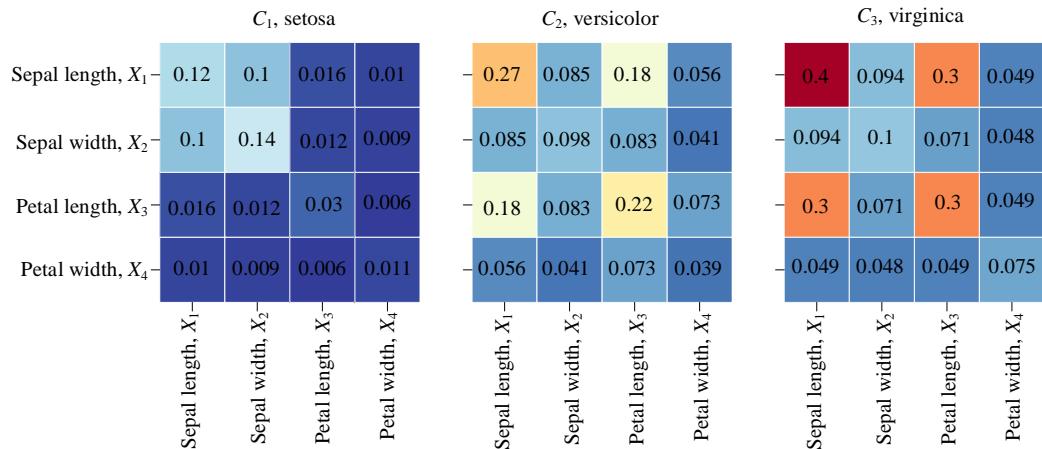


图 12. 协方差矩阵热图，考虑分类



代码文件 Bk3_Ch21_1.py 中 Bk3_Ch21_1_D 部分绘制本节热图。

21.6 线性相关系数：线性关系强弱

有了上一节的协方差，我们就可以定义**线性相关系数** (linear correlation coefficient 或 correlation coefficient)。线性相关系数也叫**皮尔逊相关系数** (Pearson correlation coefficient)，它刻画随机变量线性关系的强度，具体定义为：

$$\rho_{1,2} = \text{corr}(X_1, X_2) = \frac{\text{cov}(X_1, X_2)}{\sigma_1 \sigma_2} \quad (12)$$

ρ 的取值范围在 $[-1, 1]$ 。观察 (12)，可以发现 ρ 相当于协方差归一化。也相当于对两个随机变量的 z 分数求协方差：

$$\rho_{1,2} = \text{corr}(X_1, X_2) = \text{cov}\left(\frac{X_1 - \mu_1}{\sigma_1}, \frac{X_2 - \mu_2}{\sigma_2}\right) \quad (13)$$

归一化的线性相关系数比协方差更适合横向比较。

采用和图 10 一样的几何视角，我们来看一下在不同相关性系数条件下，红色和蓝色矩形面积的特征。

如图 13 所示，当 $\rho = 0.9$ 时，矩形的颜色几乎都是红色；当 ρ 逐步减小到 0.3 时，红色矩形依然主导，但是蓝色矩形不断变多，也就是红蓝色趋向均衡。

相反，当 $\rho = -0.9$ 时，矩形的颜色中蓝色居多，而且面积和的比例明显压倒优势；当 ρ 逐步增大到 -0.3 时，红色矩形增多，面积增大。

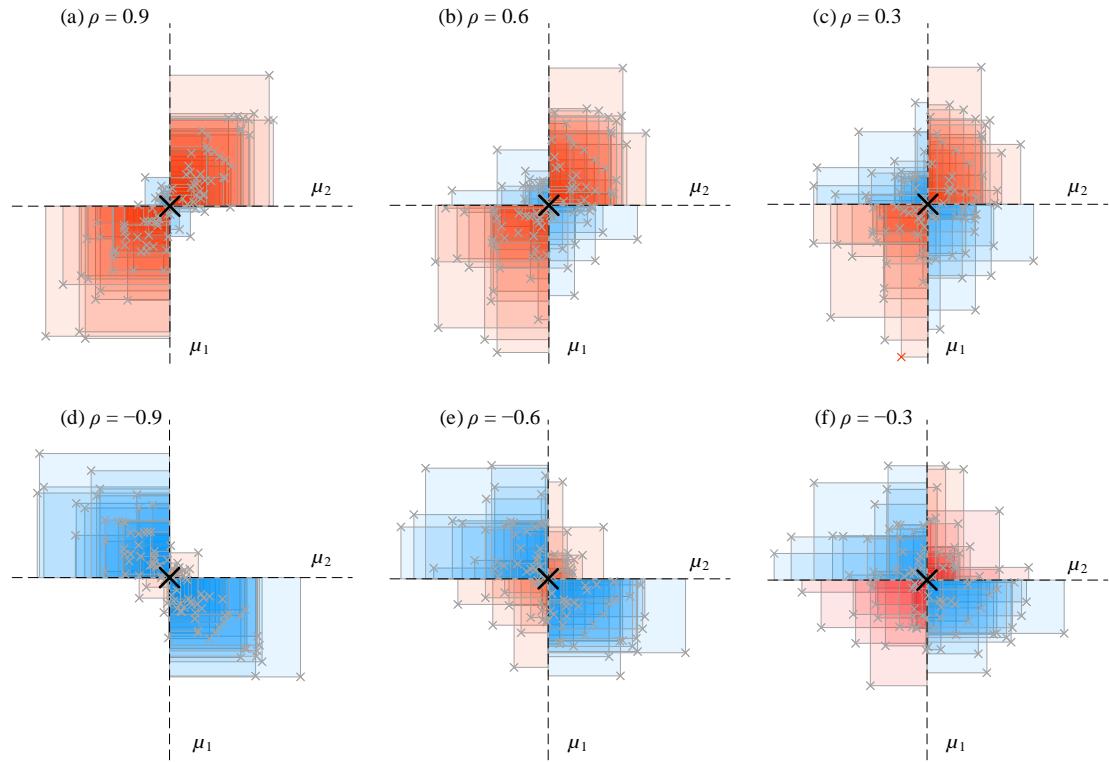


图 13. 几何视角看相关系数

某个随机变量和自身求线性关系系数，结果为 1，比如下例：

$$\text{corr}(X_1, X_1) = \frac{\text{var}(X_1)}{\sigma_1 \sigma_1} = 1 \quad (14)$$

容易知道，下式成立：

$$\text{corr}(X_i, X_j) = \text{corr}(X_j, X_i) \quad (15)$$

相关系数矩阵

类似上一节讲过的协方差矩阵，而相关系数构成的矩阵叫做**相关系数矩阵** (correlation matrix) \mathbf{P} ；以鸢尾花四个特征为例，其相关系数矩阵为 4×4 ：

$$\mathbf{P} = \begin{bmatrix} 1 & \rho_{1,2} & \rho_{1,3} & \rho_{1,4} \\ \rho_{2,1} & 1 & \rho_{2,3} & \rho_{2,4} \\ \rho_{3,1} & \rho_{3,2} & 1 & \rho_{3,4} \\ \rho_{4,1} & \rho_{4,2} & \rho_{4,3} & 1 \end{bmatrix} \quad (16)$$

线性相关系数的主对角元素为 1，这是因为随机变量和自身的线性相关系数为 1；非对角线元素为成对相关系数。

鸢尾花数据的相关性系数矩阵 \mathbf{P} 具体为：

$$\mathbf{P} = \begin{bmatrix} 1.000 & -0.118 & 0.872 & 0.818 \\ -0.118 & 1.000 & -0.428 & -0.366 \\ 0.872 & -0.428 & 1.000 & 0.963 \\ 0.818 & \underbrace{-0.366}_{\text{Sepal width, } X_2} & \underbrace{0.963}_{\text{Petal length, } X_3} & \underbrace{1.000}_{\text{Petal width, } X_4} \end{bmatrix} \quad \begin{array}{l} \leftarrow \text{Sepal length, } X_1 \\ \leftarrow \text{Sepal width, } X_2 \\ \leftarrow \text{Petal length, } X_3 \\ \leftarrow \text{Petal width, } X_4 \end{array} \quad (17)$$

图 14 所示为 \mathbf{P} 的热图。观察相关性系数矩阵 \mathbf{P} ，可以发现花萼长度 X_1 和花萼宽度 X_2 线性负相关，花瓣长度 X_3 和花萼宽度 X_2 线性负相关，花瓣宽度 X_4 和花萼宽度 X_2 线性负相关。

当然，鸢尾花数据集样本数量有限，通过样本数据得出的结论远不足以推而广之。

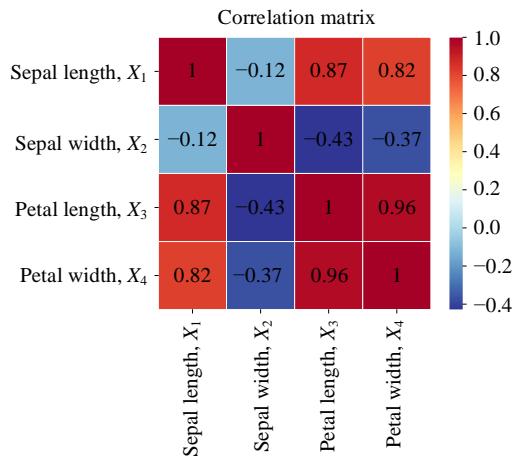


图 14. 鸢尾花数据相关性系数矩阵热图

考虑标签

图 15 为考虑分类标签条件下的协方差矩阵热图。

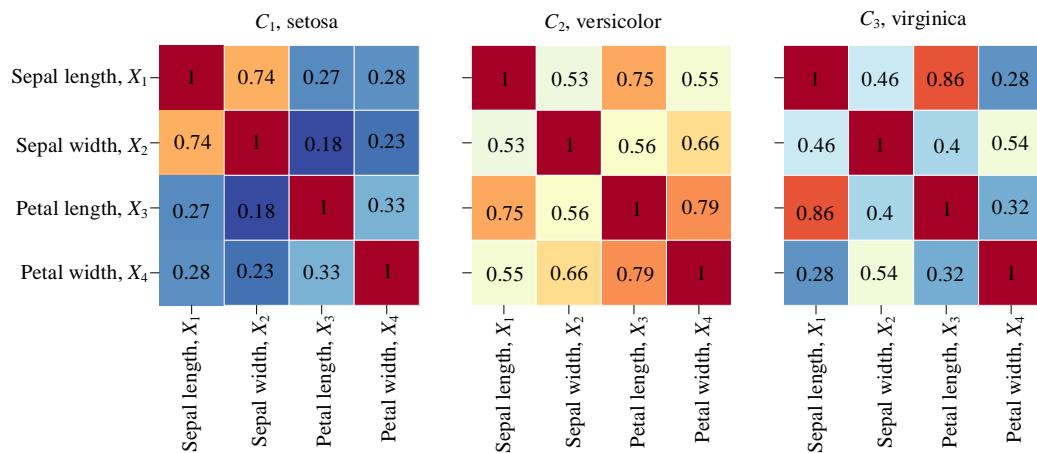
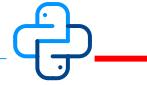


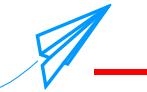
图 15. 相关性系数矩阵热图，考虑分类标签



代码文件 Bk3_Ch21_1.py 中 Bk3_Ch21_1_E 部分绘制本节热图。



在 Bk3_Ch21_1.py 基础上，我们做了一个 App 以鸢尾花数据为例展示如何用 Plotly 绘制具有交互性质的统计图像。请参考 Streamlit_Bk3_Ch21_1.py。



概率统计是数学中很大的一个版块，本书用两章的内容浮光掠影地介绍概率统计的入门知识，目的是让大家了解概率统计中重要概念，并建立它们和其他数学知识的联系。

概率统计，特别是多元概率统计，是数据科学和机器学习很多算法中重要的数学工具。本系列丛书将会在《概率统计》和大家系统探讨。



Vectors Meet Coordinate Systems

向量

向量遇见坐标系



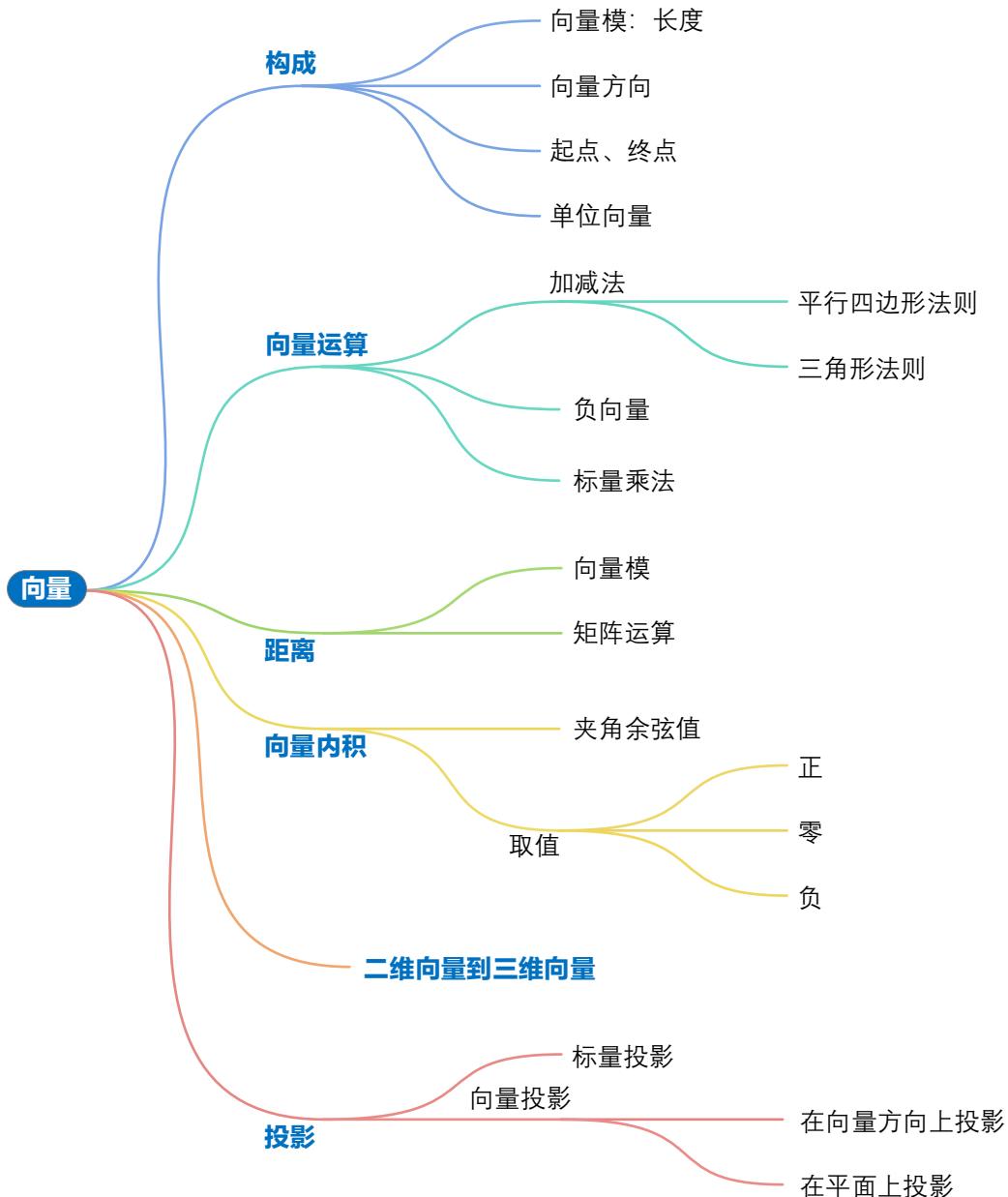
曾几何时，代数和几何形单影只、踽踽独行；它们各自蜗步难移、难成大器。然而，代数和几何结合之后，便珠联璧合、琴瑟和鸣；两者取长补短、急流勇进、日臻完美。

As long as algebra and geometry proceeded along separate paths, their advance was slow and their applications limited. But when these sciences joined company, they drew from each other fresh vitality and thenceforward marched on at a rapid pace toward perfection.

—— 约瑟夫·拉格朗日 (Joseph Lagrange) | 法国籍意大利裔数学家和天文学家 | 1736 ~ 1813



- ◀ `matplotlib.pyplot.annotate()` 在平面坐标系标注
- ◀ `matplotlib.pyplot.quiver()` 绘制箭头图
- ◀ `numpy.arccos()` 反余弦
- ◀ `numpy.degrees()` 将弧度转化为角度
- ◀ `numpy.dot()` 计算向量标量积。值得注意的是，如果输入为一维数组，`numpy.dot()` 输出结果为标量积；如果输入为矩阵，`numpy.dot()` 输出结果为矩阵乘积，相当于矩阵运算符`@`
- ◀ `numpy.linalg.norm()` 计算范数



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

22.1 向量：有大小、有方向

向量简史

很多的数学工具的发明的时候，并没有具体的用途。科学史上经常发生的情况是，几十年之后、甚至几百年之后，科学家应用某个被尘封的数学工具，完成科学技术的巨大飞跃。本书前文介绍的圆锥曲线就是很好的例子。

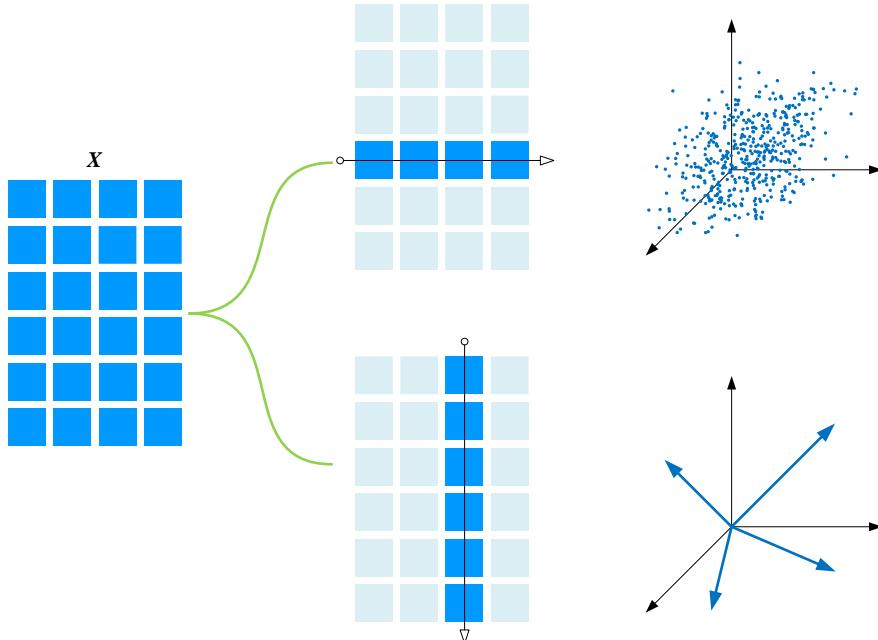
但是，也有部分数学工具是为了更好地描述其他学科发现的新理论而创造发展的，比如说向量这个概念。

向量的发明经过了漫长的 200 年岁月。几乎难以想象，现在大家熟知的向量的记法和运算规则，竟然是在 19 世纪末才加入数学这个大家庭。

1865 年开始，苏格兰数学物理学家 **麦克斯韦** (James Clerk Maxwell) 逐步提出将电、磁场、光统一起来的麦克斯韦方程组。为了更好描述麦克斯韦方程组，美国科学家 Josiah Willard Gibbs 和英国科学家 Oliver Heaviside 分别独立发明了向量的现代记法。

向量是一行或一列数字

本书第 1 章就介绍了向量。从数据角度，向量无非就是一列或一行数字。如图 1 所示，数据矩阵 X 的每一行是一个行向量，代表一个观察值； X 的每一列为一个列向量，代表某个特征上的所有数据。



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

图 1. 观察数据的两个角度

向量的几何意义

但是有了坐标系，向量便不再是无趣的数字，它们化身成一只只离弦之箭，在空间腾飞。

向量 (vector) 是**既有长度又有方向的量** (a quantity that possesses both magnitude and direction)。物理学中，**位移** (displacement)、**速度** (velocity)、**加速度** (acceleration)、**力** (force) 等物理量都是向量。

如图 2 所示，位移向量的大小代表前进的距离大小，而向量的方向代表位移方向。

和向量相对的是标量。**标量** (scalar, scalar quantity) 是有大小没有方向的量，用实数表示。

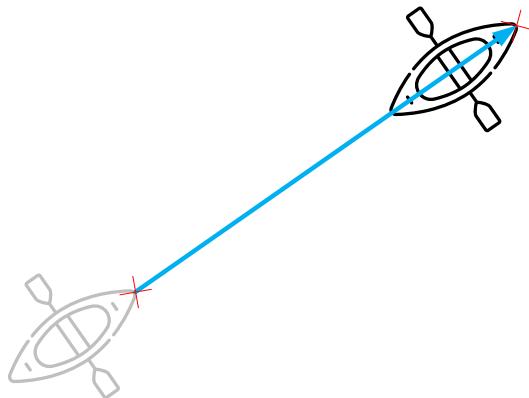


图 2. 位移向量

向量定义

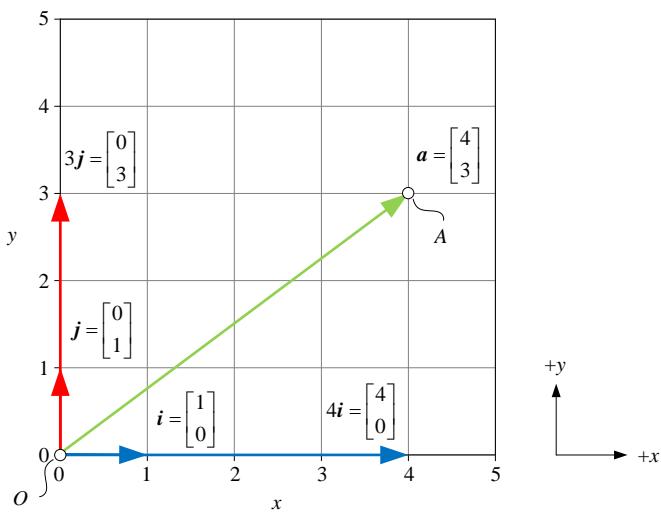
给定一个列向量 a :

$$a = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad (1)$$

如图 3 所示，向量 a 可以表达为一个带箭头的线段，它以原点 $O(0, 0)$ 为起点指向终点 $A(4, 3)$ 。因此，向量 a 也可以写作 \overrightarrow{OA} :

$$\overrightarrow{OA} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad (2)$$

本系列丛书很少使用 \overrightarrow{OA} 这种向量记法，我们一般使用小写字母，斜体、粗体，来代表向量，比如 a 、 x 、 x_1 、 $x^{(1)}$ 。

图 3. 平面直角坐标系，向量 a 的定义

向量模

如图 3 所示，向量 a 的长度就是线段 OA 的长度：

$$\|a\| = \sqrt{4^2 + 3^2} = 5 \quad (3)$$

$\|a\|$ 计算向量 a 的长度， $\|a\|$ 叫向量模，也叫 L^2 范数。 L^2 范数是 L^p 范数的一种。

向量分解

类似物理学中力的分解，一个向量还可以表达成若干向量的和。比如，向量 a 可以写成如下两个向量的和：

$$a = 4i + 3j = 4 \times \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 3 \times \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad (4)$$

i 和 j 常被称作横轴和纵轴上的单位向量，具体定义为：

$$i = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad j = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (5)$$

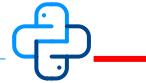
单位向量 (unit vector) 是指模 (长度) 等于 1 的向量，即：

$$\|i\| = 1, \quad \|j\| = 1 \quad (6)$$

本系列丛书后续也会使用 e_1 和 e_2 代表横轴纵轴的单位向量。

任何非零向量除以自身的模，得到向量方向上的单位向量。(1) 中向量 a 的单位向量为：

$$\frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{1}{5} \times \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} \quad (7)$$



Bk3_Ch22_1.py 绘制图 3。

22.2 几何视角看向量运算

有了平面直角坐标系，向量的加法、减法和标量乘法更容易理解。

向量加法

图 4 所示 \mathbf{a} 和 \mathbf{b} 两个向量相加对应的等式为：

$$\mathbf{a} + \mathbf{b} = \begin{bmatrix} 4 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \end{bmatrix} \quad (8)$$

如图 4 所示，从几何角度来看， \mathbf{a} 和 \mathbf{b} 两个向量相加相当于物理学中两个力的合成。

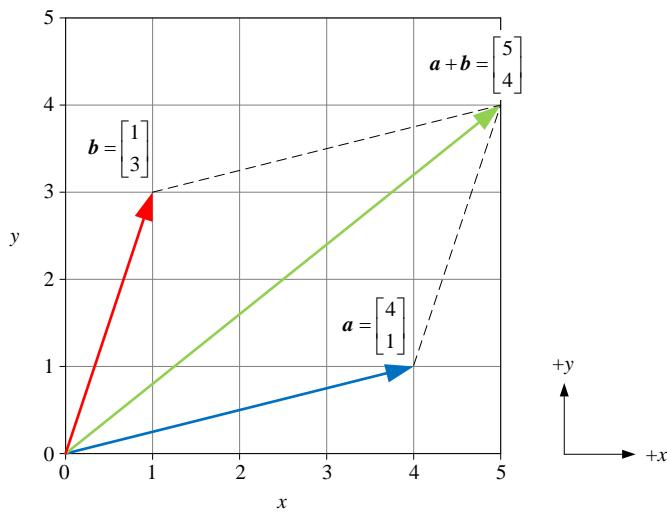


图 4. \mathbf{a} 和 \mathbf{b} 两个向量相加

正四边形和三角形法则

平面直角坐标系上，两个向量相加有两种方法——**正四边形法则** (parallelogram law)、**三角形法则** (triangle law)。

图 5 (a) 所示为平行四边形法则。将向量 a 和 b 平移至公共起点，以 a 和 b 的两条边作平行四边形， $a + b$ 为公共起点所在平行四边形对角线。

三角形法则更为常用。如图 5 (b) 所示，在平面内，将向量 a 和 b 首尾相连， $a + b$ 的结果为向量 a 的起点与向量 b 的终点相连构成的向量。也就是说 $a + b$ 始于 a 的起点，指向 b 终点。

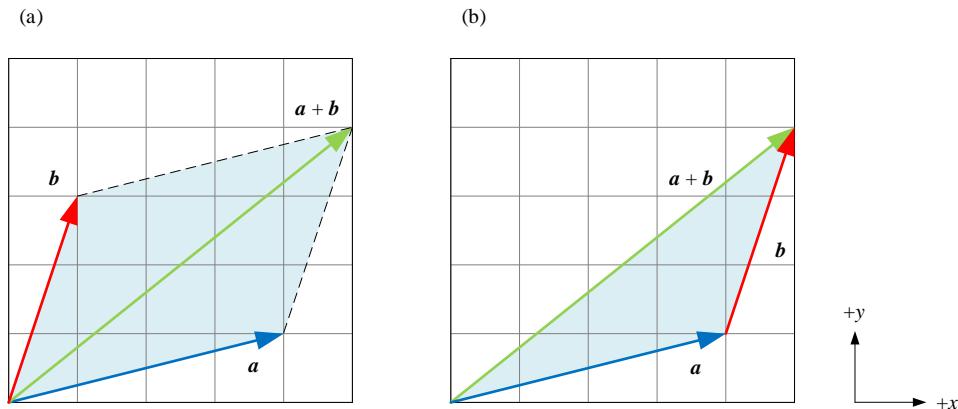


图 5. 平行四边形法则和三角形法则计算 a 和 b 两个向量相加

n 个向量相加， $a_1 + a_2 + \dots + a_n$ ，将它们首尾相连，第一个向量 a_1 的起点与最后一个向量 a_n 的终点相连构成的向量就是这几个向量的和。图 6 所示为采用三角形法则计算 5 个向量相加。

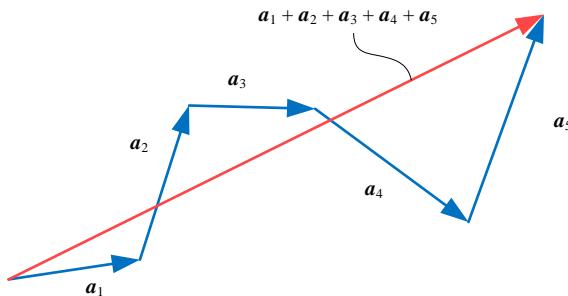


图 6. 三角形法则计算 5 个向量相加

向量减法

a 和 b 两个向量相减：

$$\mathbf{a} - \mathbf{b} = \begin{bmatrix} 4 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \quad (9)$$

从几何角度，也可以用三角形法则来求解。

如图 7 所示，将 \mathbf{a} 和 \mathbf{b} 两个向量平移至公共起点，以 \mathbf{a} 和 \mathbf{b} 作为两边构造三角形，向量 \mathbf{a} 和 \mathbf{b} 的终点连线为第三条边。这个三角形的第三边就是 $\mathbf{a} - \mathbf{b}$ 的结果， $\mathbf{a} - \mathbf{b}$ 的方向为减向量 \mathbf{b} 终点指向被减向量 \mathbf{a} 终点。

此外， $\mathbf{a} - \mathbf{b}$ 可以视作 \mathbf{a} 和 $-\mathbf{b}$ 相加。 $-\mathbf{b}$ 叫做 \mathbf{b} 的负向量。

从数字角度， \mathbf{b} 和 $-\mathbf{b}$ 对应元素为相反数。从几何角度， \mathbf{b} 和 $-\mathbf{b}$ 大小相同、方向相反。也就是说， \mathbf{b} 和 $-\mathbf{b}$ 起点、终点存在相互调换关系。

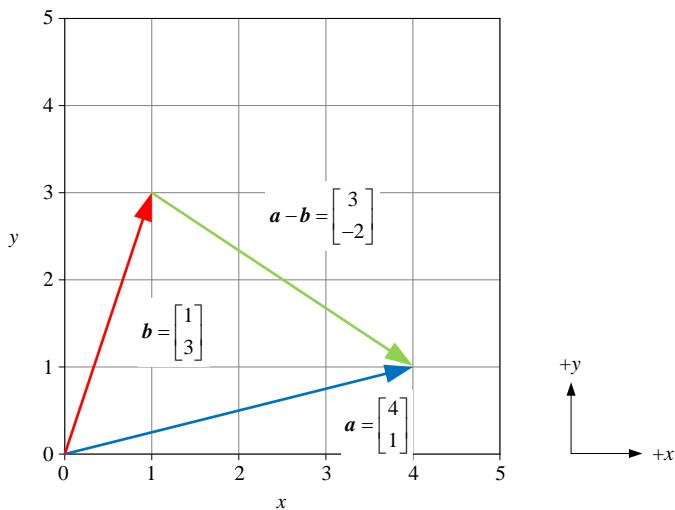


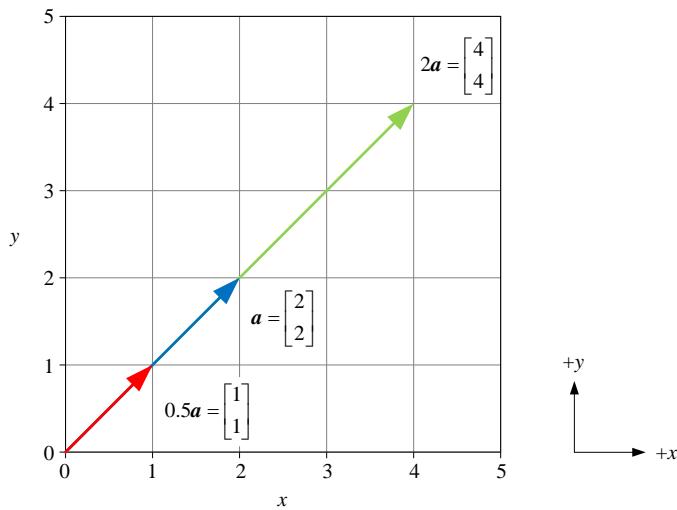
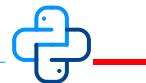
图 7. \mathbf{a} 和 \mathbf{b} 两个向量相减，三角形法则

标量乘法

图 8 所示 \mathbf{a} 的两个标量乘法：

$$0.5\mathbf{a} = 0.5 \times \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad 2\mathbf{a} = 2 \times \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \quad (10)$$

从几何角度，向量的标量乘法就是向量的缩放——长度缩放，方向不变或反向。

图 8. 向量 a 的标量乘法

Bk3_Ch22_2.py 绘制图 4、图 7、图 8。

22.3 向量简化距离运算

本书第 7 章介绍计算两点之间的距离，我们管它叫欧氏距离。本节引入向量，让距离计算变得更直观。

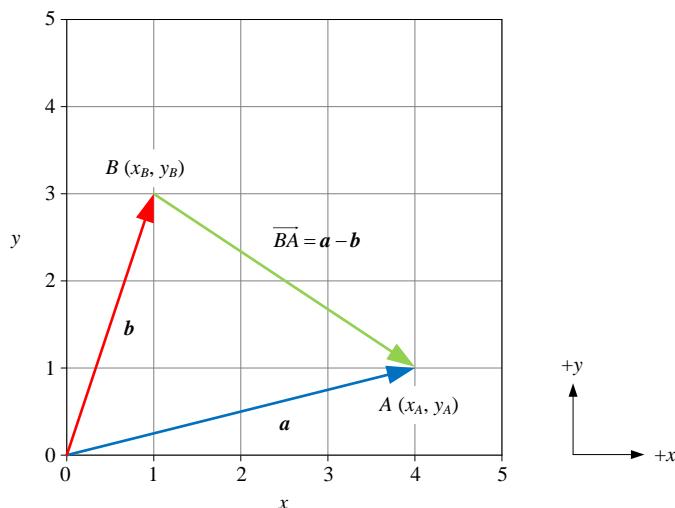
勾股定理扩展

先看第一种解释。图 9 所示，给定 $A(x_A, y_A)$ 和 $B(x_B, y_B)$ 两点， B 为起点 A 点为终点的向量 \overrightarrow{BA} 可以写作：

$$\overrightarrow{BA} = \begin{bmatrix} x_A - x_B \\ y_A - y_B \end{bmatrix} \quad (11)$$

根据勾股定理， \overrightarrow{BA} 的向量模便对应 AB 线段的长度：

$$\|\overrightarrow{BA}\| = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} = \sqrt{(4-1)^2 + (1-3)^2} = \sqrt{13} \quad (12)$$

图 9. 计算 A 和 B 距离，即 AB 长度

向量模

另外一个角度，将 A 和 B 的坐标写成向量 a 和 b ，线段 AB 长度等价于 $a - b$ 的模：

$$d = \|a - b\| = \sqrt{(a - b) \cdot (a - b)} \quad (13)$$

代入图 9 具体值， $a - b$ 为：

$$a - b = \begin{bmatrix} 4 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \quad (14)$$

将 (14) 代入 (13)，得到线段 AB 长度：

$$d = \|a - b\| = \sqrt{3^2 + (-2)^2} = \sqrt{13} \quad (15)$$

如果 a 、 b 均为列向量，用矩阵乘法来写 (13)，可以得到：

$$d = \sqrt{(a - b)^T (a - b)} = \sqrt{\begin{bmatrix} 3 \\ -2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}} = \sqrt{13} \quad (16)$$

22.4 向量内积与向量夹角

给定向量 a 和 b ，

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad (17)$$

根据本书第 2 章所讲，向量 \mathbf{a} 和 \mathbf{b} 内积为：

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_D b_D = \sum_{i=1}^D a_i b_i \quad (18)$$

有了几何视角，向量 \mathbf{a} 和 \mathbf{b} 的内积有了一个新的定义方式：

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta \quad (19)$$

图 10 所示， \mathbf{a} 和 \mathbf{b} 的内积为 \mathbf{a} 的模乘向量 \mathbf{b} 在向量 \mathbf{a} 方向上投影的分量值。 \mathbf{b} 在 \mathbf{a} 方向上投影的分量值为 $\|\mathbf{b}\| \cos \theta$ ，这个值也叫 \mathbf{b} 在 \mathbf{a} 方向上的 **标量投影** (scalar projection)。

此外，(19) 也可以理解为， \mathbf{a} 在 \mathbf{b} 方向上的投影分量值 $\|\mathbf{a}\| \cos \theta$ ， $\|\mathbf{a}\| \cos \theta$ 再乘 \mathbf{b} 得到内积结果 $\|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$ 。本章后文会专门讲解标量投影。

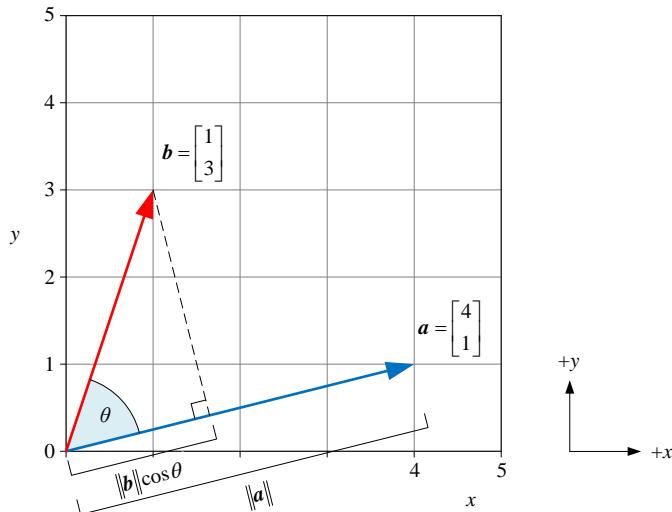


图 10. 内积的定义

向量夹角

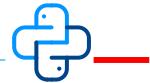
这样，向量 \mathbf{a} 和 \mathbf{b} 的夹角 θ 的余弦值可以通过下式求得：

$$\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} \quad (20)$$

举个例子，图 4 中 \mathbf{a} 和 \mathbf{b} 夹角余弦值：

$$\cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} = \frac{1 \times 4 + 3 \times 1}{\sqrt{10} \sqrt{17}} \approx 0.537 \quad (21)$$

通过反余弦求得 \mathbf{a} 和 \mathbf{b} 夹角角度约为 $\theta = 57.53^\circ$ 。



Bk3_Ch22_3.py 代码计算 \mathbf{a} 和 \mathbf{b} 夹角角度 θ 。

内积正、负、零

如图 11 所示，两个向量内积结果为正，说明向量夹角在 0° 到 90° （包括 0° ，不包括 90° ）。

内积结果为 0，说明两个向量垂直。

内积结果为负，向量夹角在 90° 到 180° （包括 180° ，不包括 90° ）。

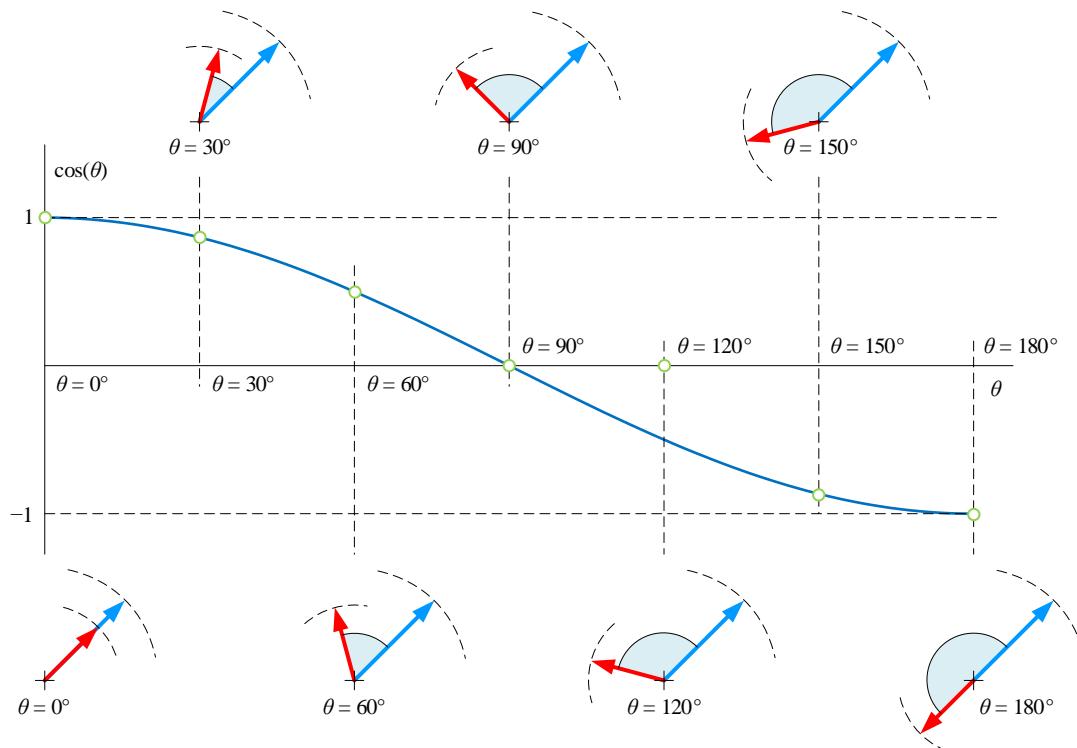


图 11. 向量夹角和余弦值关系

在平面直角坐标系中， i 和 j 相互垂直，因此两者内积为 0：

$$\mathbf{i} \cdot \mathbf{j} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \mathbf{i}^T \mathbf{j} = [1 \ 0] @ \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 0 \quad (22)$$

22.5 二维到三维

本章前文定义 i 和 j 为二维平面直角坐标系横轴和纵轴上的单位向量，在它们的基础上各加一行 0 就可以得到三维坐标系的横轴和纵轴单位向量：

$$\mathbf{i} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{j} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad (23)$$

另外，再加一个表达 z 轴正方向的单位向量 k ，

$$\mathbf{k} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (24)$$

利用 i 、 j 、 k 也可以在三维直角坐标系中构造图 3 向量 a ：

$$\mathbf{a} = 4\mathbf{i} + 3\mathbf{j} = 4 \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 3 \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 0 \end{bmatrix} \quad (25)$$

具体如图 12 所示。向量 a “趴”在 xy 平面上。

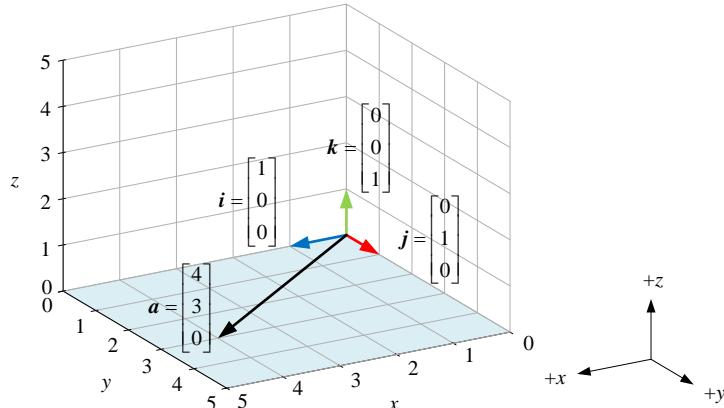


图 12. 三维直角坐标系，向量 a 的定义

投影

图 13 所示为三维直角坐标系中向量 c 和 a 的关系为：

$$\mathbf{c} = \mathbf{a} + 5\mathbf{k} = \begin{bmatrix} 4 \\ 3 \\ 0 \end{bmatrix} + 5 \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \quad (26)$$

观察图 13，发现向量 \mathbf{a} 相当于 \mathbf{c} 在 xy 平面的投影。白话说，在向量 \mathbf{c} 正上方点一盏灯，在水平面内的影子就是 \mathbf{a} 。这就是我们在本书第 3 章介绍的 **投影** (projection)。

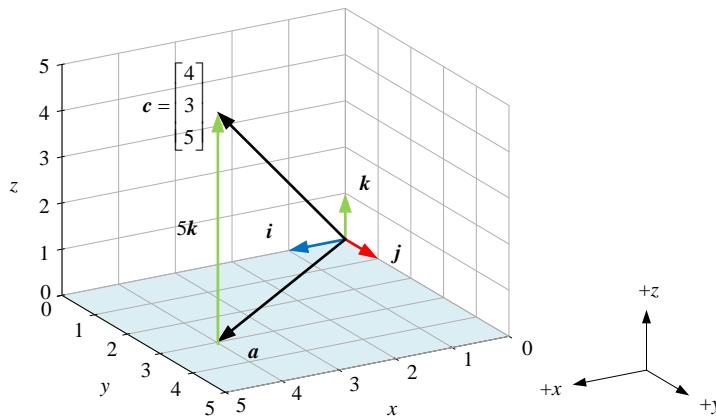
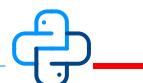


图 13. 三维直角坐标系，向量 \mathbf{a} 和向量 \mathbf{c} 的关系

值得注意的，向量 \mathbf{c} 和 \mathbf{a} 之差为 $5\mathbf{k}$ ，而 $5\mathbf{k}$ 垂直于 xy 平面。也就是说， $5\mathbf{k}$ 垂直 \mathbf{i} ，同时垂直于 \mathbf{j} 。即， $5\mathbf{k}$ 和 \mathbf{i} 内积为 0， $5\mathbf{k}$ 和 \mathbf{j} 内积为 0：

$$5\mathbf{k} \cdot \mathbf{i} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 0, \quad 5\mathbf{k} \cdot \mathbf{j} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 0 \quad (27)$$



Bk3_Ch22_4.py 绘制图 13。

22.6 投影：影子的长度

投影分为两种——**标量投影** (scalar projection) 和**向量投影** (vector projection)。

标量投影

向量标量投影结果为标量。

利用向量内积的计算原理，向量 \mathbf{b} 在向量 \mathbf{a} 方向上投影得到的线段长度就是标量投影：

$$\|\mathbf{b}\| \cos \theta = \|\mathbf{b}\| \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|} \quad (28)$$

如图 14 所示， \mathbf{b} 在 \mathbf{a} 方向上的标量投影为：

$$\|\mathbf{b}\| \cos \theta = \frac{7}{\sqrt{17}} \quad (29)$$

\mathbf{a} 在 \mathbf{b} 方向上的标量投影为：

$$\|\mathbf{a}\| \cos \theta = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{b}\|} = \frac{7}{\sqrt{10}} \quad (30)$$

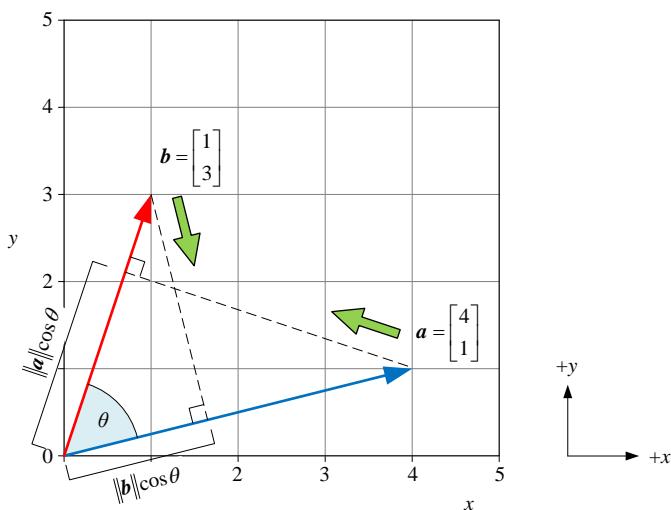


图 14. 标量投影

向量投影

向量投影则是在标量基础上，加上方向。也就是说 \mathbf{b} 在 \mathbf{a} 方向上标量投影，再乘 \mathbf{a} 的单位向量：

$$\text{proj}_{\mathbf{a}} \mathbf{b} = \|\mathbf{b}\| \cos \theta \frac{\mathbf{a}}{\|\mathbf{a}\|} = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|} \frac{\mathbf{a}}{\|\mathbf{a}\|} \quad (31)$$

特别地，如果 \mathbf{v} 为单位向量， \mathbf{a} 在 \mathbf{v} 方向上标量投影为：

$$\|\mathbf{a}\| \cos \theta = \frac{\mathbf{a} \cdot \mathbf{v}}{\|\mathbf{v}\|} = \mathbf{a} \cdot \mathbf{v} \quad (32)$$

如图 15 所示， \mathbf{a} 在单位向量 \mathbf{v} 方向上向量投影为：

$$\text{proj}_v \mathbf{a} = (\|\mathbf{a}\| \cos \theta) \mathbf{v} = (\mathbf{a} \cdot \mathbf{v}) \mathbf{v} \quad (33)$$

本系列丛书里面最常见的就是上述投影规则。若 \mathbf{a} 和 \mathbf{v} 均为列向量，可以用矩阵乘法规则重新写 (33)：

$$\text{proj}_v \mathbf{a} = (\mathbf{a}^\top \mathbf{v}) \mathbf{v} = (\mathbf{v}^\top \mathbf{a}) \mathbf{v} \quad (34)$$

投影是线性代数中有关向量最重要的几何操作，没有之一。投影会经常出现在本系列丛书不同板块中。本节会多花一些笔墨和大家聊聊向量投影，给大家建立更加直观的印象。

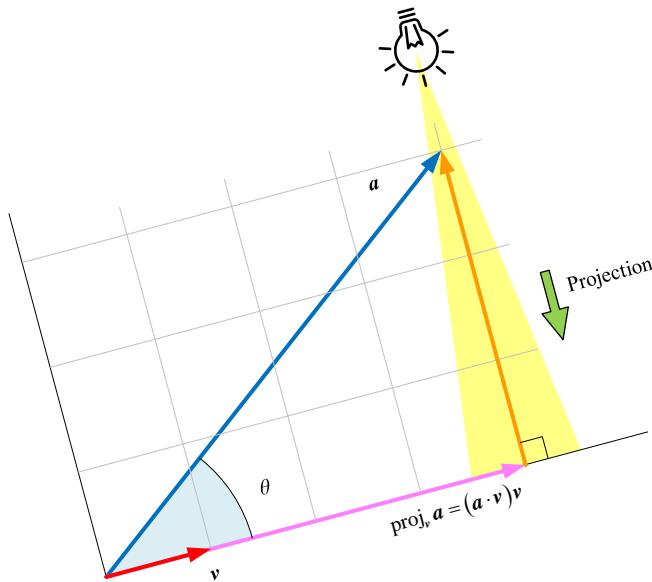


图 15. 正交投影的意义

二维向量投影

如图 16 所示，向量 \mathbf{a} 在 \mathbf{i} 方向上向量投影为：

$$\text{proj}_i \mathbf{a} = \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \mathbf{i} = 4\mathbf{i} = \begin{bmatrix} 4 \\ 0 \end{bmatrix} \quad (35)$$

\mathbf{i} 就是单位向量。

用矩阵乘法计算 (35)：

$$\begin{aligned}\text{proj}_i \mathbf{a} &= \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix}^T @ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) \mathbf{i} = \left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right) \mathbf{i} = 4\mathbf{i} \\ &= \mathbf{i} \left(\mathbf{i}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right) = \mathbf{i} @ \mathbf{i}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \end{bmatrix}\end{aligned}\tag{36}$$



在本系列丛书《矩阵力量》一册中，我们会给上式 $\mathbf{i} @ \mathbf{i}^T$ 一个全新的名字——张量积。

\mathbf{a} 在 \mathbf{j} 方向上向量投影：

$$\text{proj}_j \mathbf{a} = \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix} @ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \mathbf{j} = 3\mathbf{j} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}\tag{37}$$

用矩阵乘法计算 (37)：

$$\begin{aligned}\text{proj}_j \mathbf{a} &= \left(\begin{bmatrix} 4 \\ 3 \end{bmatrix}^T @ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) \mathbf{j} = \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right) \mathbf{j} = 3\mathbf{j} \\ &= \mathbf{j} \left(\mathbf{j}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} \right) = \mathbf{j} @ \mathbf{j}^T @ \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}\end{aligned}\tag{38}$$

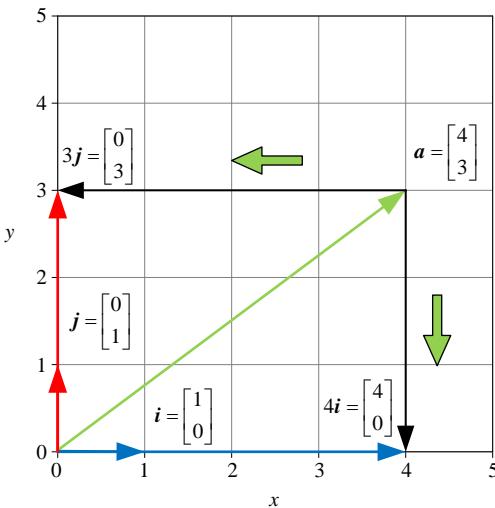


图 16. \mathbf{a} 分别在 \mathbf{i} 和 \mathbf{j} 方向上向量投影

三维向量投影

我们再看三维向量投影。如图 17 所示， \mathbf{c} 在 \mathbf{i} 方向上标量投影：

$$\text{proj}_i \mathbf{c} = \left(\begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} @ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \mathbf{i} = 4\mathbf{i} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}\tag{39}$$

用矩阵乘法写 (39)，得到：

$$\begin{aligned}\text{proj}_i \mathbf{c} &= \left([4 \ 3 \ 5] @ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \mathbf{i} = \left([1 \ 0 \ 0] @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right) \mathbf{i} = 4\mathbf{i} \\ &= \mathbf{i} \left(\mathbf{i}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right) = \mathbf{i} @ \mathbf{i}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}\end{aligned}\quad (40)$$

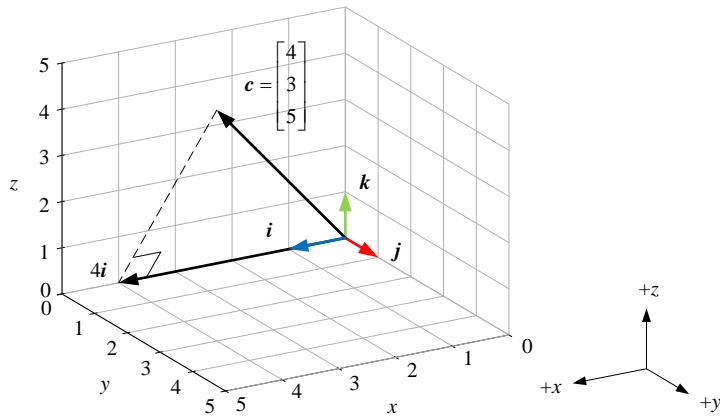


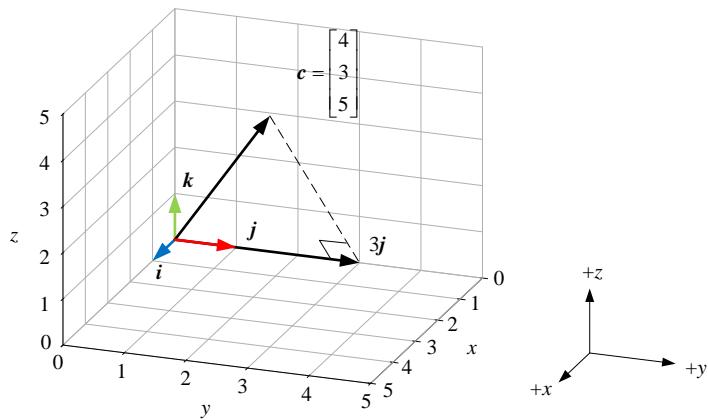
图 17. 向量 \mathbf{c} 在向量 \mathbf{i} 方向上投影

如图 18 所示， \mathbf{c} 在 \mathbf{j} 方向上标量投影为：

$$\text{proj}_j \mathbf{c} = \left(\begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) \mathbf{j} = 3\mathbf{j} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}\quad (41)$$

同样，用矩阵乘法写 (41)：

$$\begin{aligned}\text{proj}_j \mathbf{c} &= \left([4 \ 3 \ 5] @ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right) \mathbf{j} = \left([0 \ 1 \ 0] @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right) \mathbf{j} = 3\mathbf{j} \\ &= \mathbf{j} \left(\mathbf{j}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right) = \mathbf{j} @ \mathbf{j}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 0 \end{bmatrix}\end{aligned}\quad (42)$$

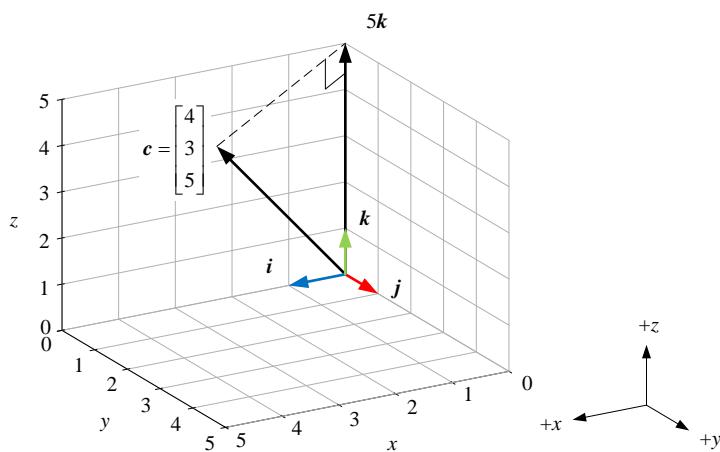
图 18. 向量 c 在向量 j 方向上投影

如图 19 所示， c 在 k 方向上标量投影：

$$\text{proj}_k \mathbf{c} = \left(\begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \mathbf{k} = 5\mathbf{k} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \quad (43)$$

用矩阵乘法写 (43)：

$$\begin{aligned} \text{proj}_k \mathbf{c} &= \left[\begin{bmatrix} 4 & 3 & 5 \end{bmatrix} @ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right] \mathbf{k} = \left[\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right] \mathbf{k} = 5\mathbf{k} \\ &= \mathbf{k} \left(\mathbf{k}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} \right) = \mathbf{k} @ \mathbf{k}^T @ \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \end{aligned} \quad (44)$$

图 19. 向量 c 在向量 k 方向上投影

平面投影

本节最后聊一下三维向量在各个平面投影。

以图 20 为例，因为 i 和 j 张起了 xy 平面，向量 c 在 xy 平面投影，相当于向量 c 分别在 i 和 j 向量投影，再合成。

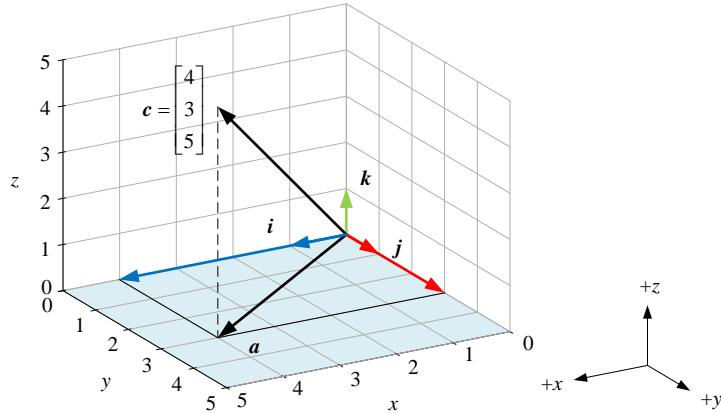


图 20. 向量 c 在 xy 平面投影

换个角度求解这个问题。 c 在 xy 平面上投影为 a :

$$\mathbf{a} = xi + yj \quad (45)$$

其中， x 和 y 为未知量；为了求解 x 和 y ，我们需要构造两个等式。

首先计算 $c - a$,

$$\mathbf{c} - \mathbf{a} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} - (xi + yj) = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} - x \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - y \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 3-x \\ 4-y \\ 5 \end{bmatrix} \quad (46)$$

根据前文分析，我们知道 $c - a$ 分别垂直 i 和 j ，这样我们可以构造两个等式：

$$\begin{aligned} (\mathbf{c} - \mathbf{a}) \cdot \mathbf{i} &= 0 \\ (\mathbf{c} - \mathbf{a}) \cdot \mathbf{j} &= 0 \end{aligned} \quad (47)$$

将 (46) 代入 (47) 得到：

$$\begin{bmatrix} 3-x \\ 4-y \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 0, \quad \begin{bmatrix} 3-x \\ 4-y \\ 5 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 0 \quad (48)$$

整理得到方程组，并求解 x 和 y :

$$\begin{cases} 3-x=0 \\ 4-y=0 \end{cases} \Rightarrow \begin{cases} x=3 \\ y=4 \end{cases} \quad (49)$$

这样计算得到， \mathbf{c} 在 xy 平面上投影为 $\mathbf{a} = 3\mathbf{i} + 4\mathbf{j}$ 。

图 21 和图 22 分别所示为向量 \mathbf{c} 在 yz 和 xz 平面投影，请读者自行计算投影结果。

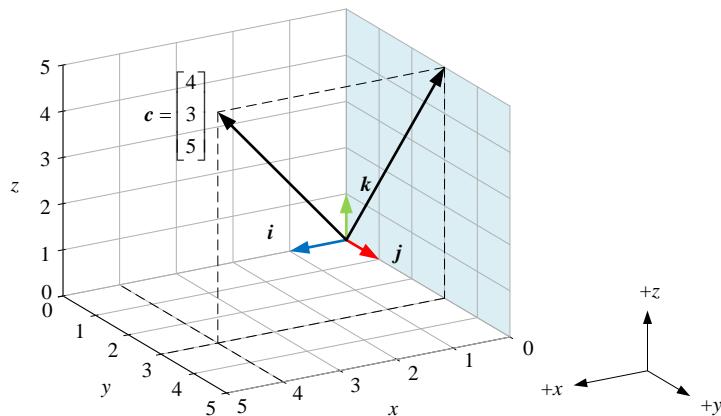


图 21. 向量 \mathbf{c} 在 yz 平面投影

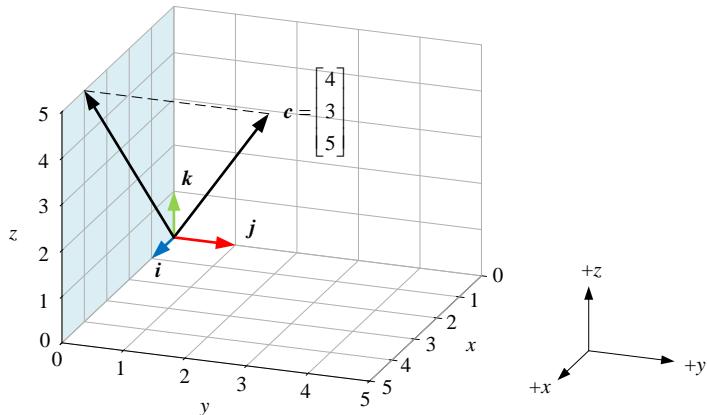


图 22. 向量 \mathbf{c} 在 xz 平面投影



向量是线性代数中的多面手，它可以是一行或一列数，也可以是矩阵的一行或一列；有了坐标系，向量和空间中的点、线等元素建立的连线，这时向量摇身一变，变成了有方向的线段。

正是因为向量有几何内涵，线性代数的知识都可以用几何视角来理解。有向量的地方，就有几何。本系列丛书介绍在线性代数知识时都会给出几何视角，请大家格外留意。

下面，请大家准备开始本书最后“鸡兔同笼三部曲”的学习之旅。

23

Fundamentals of Linear Algebra

鸡兔同笼 1

之从《孙子算经》到线性代数



这就是数学。

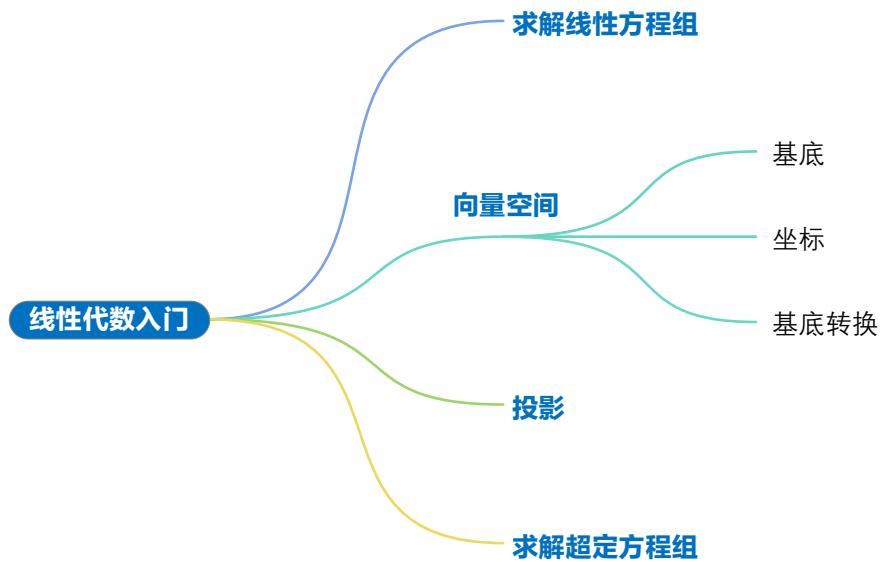
她提醒你无形灵魂的存在，她赋予数学发现以生命；她唤醒沉睡的心灵，她净化蒙尘的心智；她给思想以光辉。她涤荡与生俱来的蒙昧与无知。

This, therefore, is mathematics: she reminds you of the invisible form of the soul; she gives life to her own discoveries; she awakens the mind and purifies the intellect; she brings light to our intrinsic ideas; she abolishes oblivion and ignorance which are ours by birth.

—— 普罗克洛 (Proclus) | 古希腊哲学家 | 412 ~ 485



- ◀ `matplotlib.pyplot.quiver()` 绘制箭头图
- ◀ `numpy.column_stack()` 将两个矩阵按列合并
- ◀ `numpy.linalg.inv()` 矩阵求逆
- ◀ `numpy.linalg.solve()` 求解线性方程组
- ◀ `numpy.matrix()` 创建矩阵
- ◀ `sympy.solve()` 求解符号方程组
- ◀ `sympy.solvers.solveset.linsolve()` 求解符号线性方程组



23.1 从鸡兔同笼说起

云山青青，风泉泠泠，山色可爱，泉声可听。土地平旷，屋舍俨然，阡陌交通，鸡犬相闻。

崇山峻岭之中，茂林修竹深处，有个小村，村中有五十户人家。大伙儿甘其食，美其服，安其居，乐其俗。黄发垂髫，怡然自乐。

村民善养鸡兔，又善筹算。在这个与世隔绝的小村庄，鸡兔同笼这样的经典数学问题，代代流传，深入人心。村民中有个农夫，他特别痴迷数学。最近他手不释卷地阅读一本叫《线性代数》的舶来经典。

本书最后三章给大家说说村民在养鸡养兔遇到的数学问题，讲讲农夫如何用学到的线性代数工具帮助大伙儿解决这些问题。

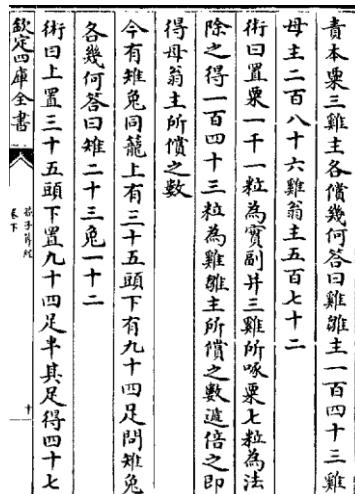


图 1. 《孙子算经》中的鸡兔同笼问题(来源：<https://cnkgraph.com/>)

鸡兔同笼原题

《孙子算经》中鸡兔同笼问题这样说，“今有雉兔同笼，上有三十五头，下有九十四足，问雉兔各几何？”

本书前文构造二元一次方程组，用代数方法解决鸡兔同笼问题：

$$\begin{cases} x_1 + x_2 = 35 \\ 2x_1 + 4x_2 = 94 \end{cases} \quad (1)$$

其中， x_1 为鸡数量， x_2 为兔数量。

求得笼子里有 23 只鸡，12 只兔：

$$\begin{cases} x_1 = 23 \\ x_2 = 12 \end{cases} \quad (2)$$

此外，本书之前也介绍过利用坐标系图解鸡兔同笼问题。

线性方程组

农夫决定用自己刚刚学过的线性代数知识解决“鸡兔同笼”这个数学问题。

(1) 中第一个等式写成矩阵运算形式，得到：

$$1 \cdot x_1 + 1 \cdot x_2 = 35 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 35 \end{bmatrix} \quad (3)$$

(1) 第二个等式也写成类似形式：

$$2 \cdot x_1 + 4 \cdot x_2 = 94 \Rightarrow \begin{bmatrix} 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 94 \end{bmatrix} \quad (4)$$

结合 (3) 和 (4)，农夫便用矩阵形式写出了鸡兔同笼问题的线性方程组：

$$\begin{cases} 1 \cdot x_1 + 1 \cdot x_2 = 35 \\ 2 \cdot x_1 + 4 \cdot x_2 = 94 \end{cases} \Rightarrow \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 35 \\ 94 \end{bmatrix} \quad (5)$$

(5) 可以写成：

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (6)$$

其中，

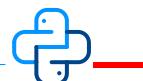
$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 35 \\ 94 \end{bmatrix} \quad (7)$$

\mathbf{x} 是未知变量构成的列向量， \mathbf{A} 为方阵且可逆， \mathbf{x} 可以利用下式求得：

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \quad (8)$$

代入具体数值计算得到 \mathbf{x} ：

$$\mathbf{x} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 35 \\ 94 \end{bmatrix} = \begin{bmatrix} 2 & -0.5 \\ -1 & 0.5 \end{bmatrix} \begin{bmatrix} 35 \\ 94 \end{bmatrix} = \begin{bmatrix} 23 \\ 12 \end{bmatrix} \quad (9)$$



Bk3_Ch23_1.py 完成上述运算。

23.2 “鸡” 向量与“兔” 向量

农夫观察矩阵 A ，发现它是由两个列向量左右排列构造而成：

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \quad \begin{array}{l} \text{Head} \\ \text{Feet} \end{array} \quad (10)$$

由此，农夫将矩阵 A 写成 a_1 和 a_2 两个左右排列的列向量：

$$[a_1 \ a_2] \quad (11)$$

农夫特别好奇 a_1 和 a_2 这两个向量的具体含义，他决定深入分析一番。

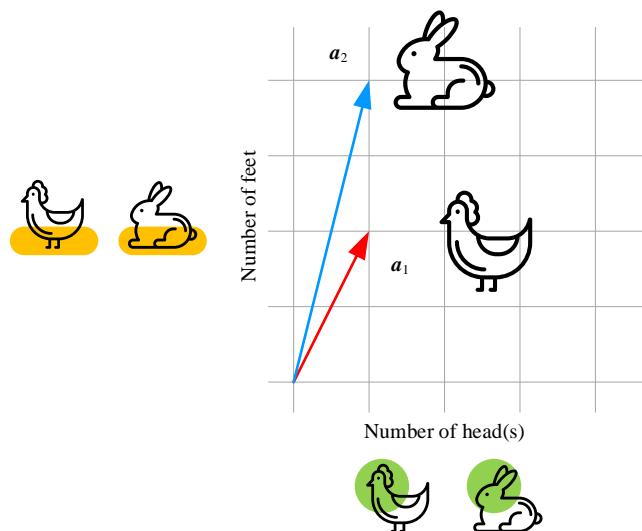


图 2. 鸡向量 a_1 和兔向量 a_2

农夫认为 a_1 代表一只鸡，特征是一个头、两只脚：

$$a_1 = \begin{bmatrix} \# \text{head} \\ 1 \\ \# \text{feet} \\ 2 \end{bmatrix} \quad (12)$$

a_2 代表一只兔，特征是有一个头、四只脚：

$$a_2 = \begin{bmatrix} \# \text{head} \\ 1 \\ \# \text{feet} \\ 4 \end{bmatrix} \quad (13)$$

农夫决定管 a_1 叫“鸡向量”， a_2 叫“兔向量”。

图 2 所示为鸡向量 a_1 和兔向量 a_2 。图 2 中坐标轴的横轴为头的数量，纵轴为脚的数量。 e_1 代表“头”向量， e_2 代表“脚”向量。显然， e_1 是横轴单位向量， e_2 是纵轴单位向量。

分解

如图 3 所示，鸡向量 a_1 可以写成：

$$a_1 = \begin{bmatrix} \# \text{ head} \\ 1 \\ \# \text{ feet} \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = [e_1 \ e_2] \begin{bmatrix} 1 \\ 2 \end{bmatrix} = e_1 + 2e_2 \quad (14)$$

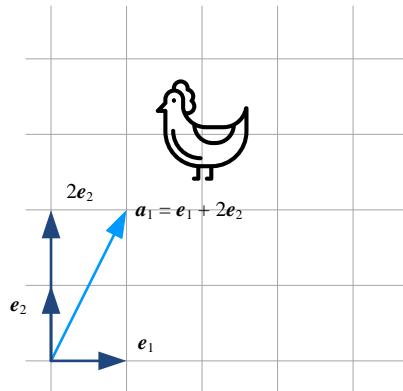
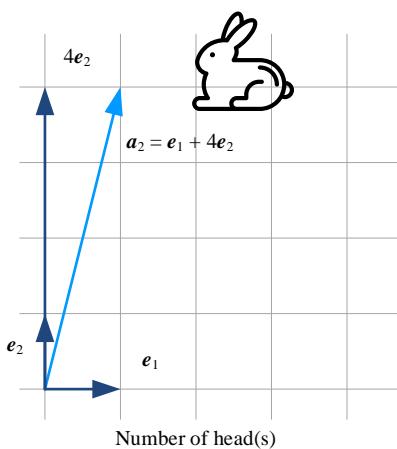


图 3. 鸡向量 a_1

如图 4 所示，兔向量 a_2 可以写成：

$$a_2 = \begin{bmatrix} \# \text{ head} \\ 1 \\ \# \text{ feet} \\ 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = [e_1 \ e_2] \begin{bmatrix} 1 \\ 4 \end{bmatrix} = e_1 + 4e_2 \quad (15)$$

图 4. 兔向量 a_2

再谈鸡兔同笼

回到鸡兔同笼问题， x_1 代表鸡的数量， x_2 为兔的数量。农夫将 $A = [a_1, a_2]$ 代入 (6)，得到：

$$\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 35 \\ 94 \end{bmatrix} \quad \begin{array}{c} \text{一只鸡} \\ \text{一只兔} \\ \text{一只鸡} \\ \text{一只兔} \end{array} \quad (16)$$

白话说，(16) 代表 x_1 份 a_1 和 x_2 份 a_2 组合，得到 b 向量。

为了方便可视化，农夫将向量 b 改为如下具体值。也就是鸡兔同笼问题条件变为：鸡兔同笼有 3 个头、8 只脚。

农夫把线性方程组写成：

$$\begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix} \quad \begin{array}{c} \text{一只鸡} \\ \text{一只兔} \\ \text{一只鸡} \\ \text{一只兔} \end{array} \quad (17)$$

农夫此刻在思考这个问题， x 和 b 具体代表什么？

(17) 等式左边的列向量 $x = [x_1, x_2]^T$ 代表鸡兔数量，而 (17) 右侧 b 代表头、脚数量。

坐标系角度

从坐标系的角度来看， x 在“鸡-兔系”中，而 b 在“头-脚系”中。

图 5 左侧方格就是“头-脚系”，而图 5 右侧平行四边形网格便是“鸡-兔系”。

“头-脚系”中，“头”向量 e_1 和“脚”向量 e_2 ，张成了方格面。白话说，在“头-脚系”中，农夫看到的是鸡兔的头和脚数。

“鸡-兔系”中，“鸡”向量 a_1 和“兔”向量 a_2 ，张成了平行四边形网格。在“鸡-兔系”中，农夫认为自己关注的是鸡兔具体只数。

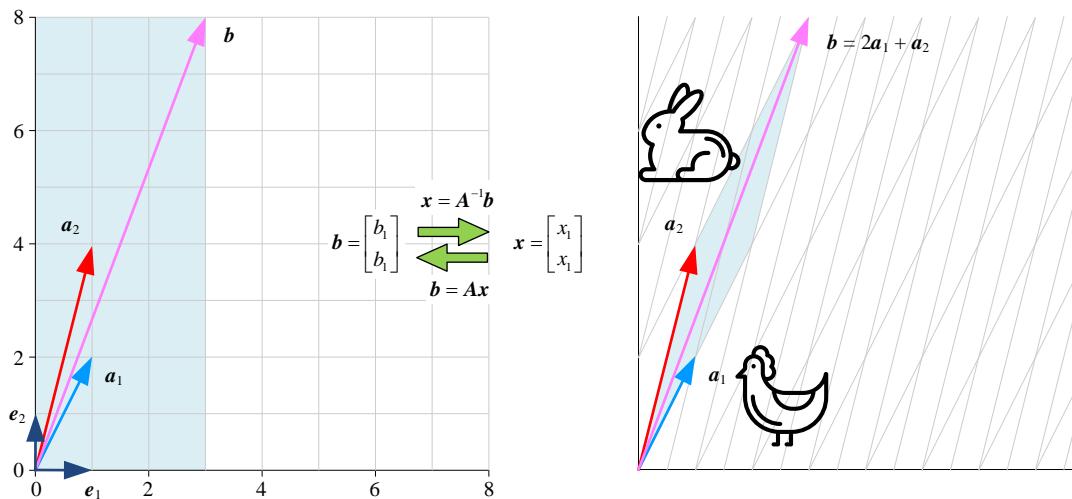


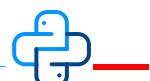
图 5. “头-脚系”和“鸡-兔系”相互转换

A 作为桥梁，完成从“鸡-兔系” x 向“头-脚系” b 转换：

$$x \rightarrow b : Ax = b \quad (18)$$

反方向来看， A^{-1} 完成“头-脚” b 向“鸡-兔” x 转换：

$$b \rightarrow x : A^{-1}b = x \quad (19)$$



Bk3_Ch23_2.py 绘制图 5。



在 Bk3_Ch23_2.py 基础上，我们做了一个 App 用来可视化矩阵 A 对网格形状的影响。请参考 Streamlit_Bk3_Ch23_2.py。

23.3 那几只毛绒耳朵

农夫看了看同处一笼的鸡兔，突然发现在头、脚之外，赫然独立几只可爱至极的毛绒耳朵。

他突然想到，除了查头数、脚数之外，查毛绒耳朵的数量应该更容易确定兔子的数量！虽然，生理学角度，鸡也有耳朵，但是极不容易被发现。

加了毛绒耳朵这个特征之后，二维向量就变成了三维向量。

鸡向量 a_1 变为：

$$a_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

(20)

兔向量 a_2 变为：

$$a_2 = \begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix}$$

(21)

在平面直角坐标系中，升起第三个维度——毛绒耳朵数量，农夫便得到如图 6 所示的三维直角坐标系。其中， e_3 代表“毛绒耳朵”向量。

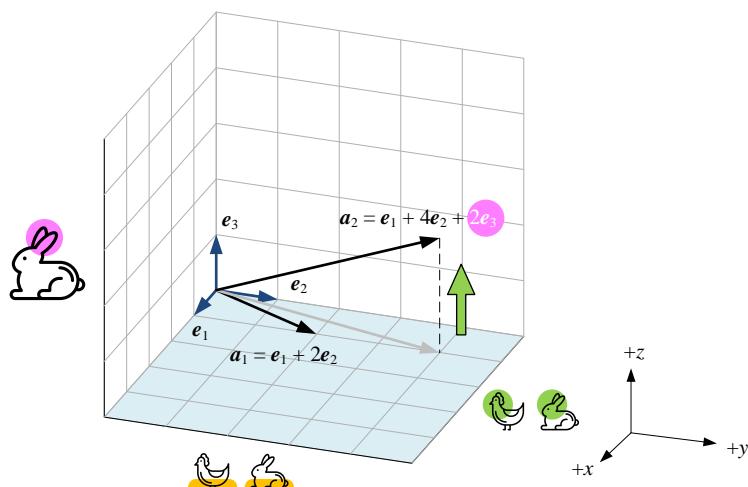


图 6. 三维直角坐标系中的鸡向量 a_1 和兔向量 a_2

图 6 中，一只鸡一个头、两只脚、没有毛绒耳朵，因此鸡向量 a_1 为：

$$a_1 = e_1 + 2e_2 \quad (22)$$

观察图 6，鸡向量 a_1 还“趴”在水平面上，这是因为鸡没有毛绒耳朵！

一只兔有一个头、四只脚、两个毛绒耳朵， a_2 写成：

$$a_2 = e_1 + 4e_2 + 2e_3 \quad (23)$$

而兔向量 a_2 还已经“立”在水平面之外，就是因为那两只毛绒耳朵（撸撸）。

计算头、脚、毛绒耳朵数量

如果给定一笼鸡兔的鸡和兔的数量，让大家求解头、脚、毛绒耳朵数量，就是从“鸡-兔系”到“头-脚-毛绒耳朵系”的转化。

假设有鸡 10 只 (x_1)、兔 5 只 (x_2)，可以通过下式计算头、脚和毛绒耳朵数量：

$$\mathbf{b} = [\mathbf{a}_1 \quad \mathbf{a}_2] \mathbf{x} = \begin{bmatrix} 1 & 1 \\ 2 & 4 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 10 \\ 5 \end{bmatrix} = \begin{bmatrix} 15 \\ 40 \\ 10 \end{bmatrix} \quad (24)$$

这样，通过上述计算，农夫便完成了从“鸡-兔系”到“头-脚-毛绒耳朵系”的转换。这个过程是从二维到三维，相当于“升维”。

23.4 “鸡兔”套餐

村子里来个小贩买小鸡和小兔，但可惜不单独售卖。

小贩提供两种套餐捆绑销售： A 套餐，3 鸡 1 兔； B 套餐，1 鸡 2 兔。

这可难坏了农夫，因为他想买 10 只鸡、10 只兔。该怎么组合 A 、 B 两种套餐？

农夫想了想，发现这不就是个“鸡兔同笼”问题的升级版嘛！下面，农夫决定用线性代数这个万能工具试试看。

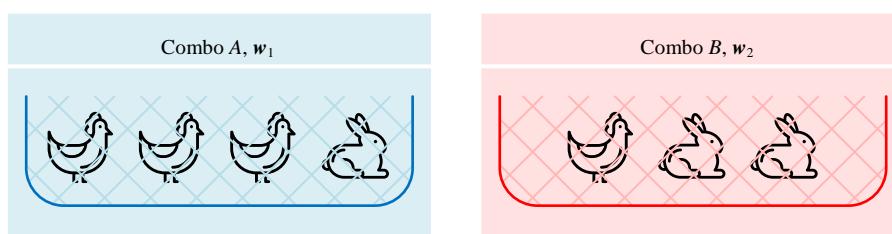


图 7. 鸡兔 A 、 B 套餐

A - B 套餐系

农夫将 A 、 B 套餐记做列向量 w_1 和 w_2 ，具体取值如下：

$$\mathbf{w}_1 = \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad (25)$$

农夫想买 10 只鸡、10 只兔，记做 \mathbf{a} ：

$$\mathbf{a} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (26)$$

令，所需套餐 A 的数量为 x_1 ，套餐 B 的数量为 x_2 ，构造如下等式：

$$x_1 \mathbf{w}_1 + x_2 \mathbf{w}_2 = x_1 \begin{bmatrix} 3 \\ 1 \end{bmatrix} + x_2 \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (27)$$

即：

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (28)$$

如图 8 所示，向量 \mathbf{a} 在“鸡-兔系”到“A-B 套餐系”的不同意义。图 8 (a) 给出的是鸡兔数量，图 8 (b) 展示的套餐数量。

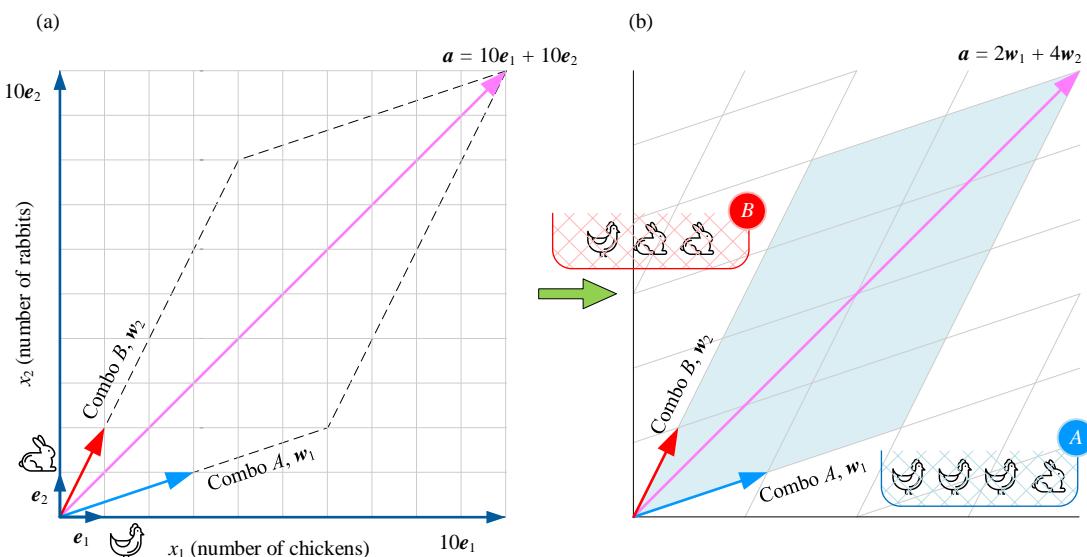


图 8. 向量 \mathbf{a} 在“鸡-兔系”到“A-B 套餐系”的不同意义

这样农夫求得向量 \mathbf{x} 为：

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}^{-1} @ \begin{bmatrix} 10 \\ 10 \end{bmatrix} = \begin{bmatrix} 0.4 & -0.2 \\ -0.2 & 0.6 \end{bmatrix} @ \begin{bmatrix} 10 \\ 10 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \quad (29)$$

线性组合

也就是说，农夫可以买 2 份 A 套餐、4 份 B 套餐，这样一共买到 10 只鸡、10 只兔，对应算式为：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>
本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>
欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$2\mathbf{w}_1 + 4\mathbf{w}_2 = 2 \times \begin{bmatrix} 3 \\ 1 \end{bmatrix} + 4 \times \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (30)$$

翻阅《线性代数》，农夫发现(30)这个等式就叫**线性组合**(linear combination)。书上管 \mathbf{w}_1 和 \mathbf{w}_2 叫做**基底**(basis)，写成 $\{\mathbf{w}_1, \mathbf{w}_2\}$ 。也就是说，图9左图的基底为 $\{\mathbf{a}_1, \mathbf{a}_2\}$ ，右图的基底为 $\{\mathbf{w}_1, \mathbf{w}_2\}$ 。

白话说，就是用2份 \mathbf{w}_1 向量、4份 \mathbf{w}_2 向量混合得到向量 \mathbf{a} 。通过线性组合的向量仍在平面之内。

如图9所示，农夫发现，如果只看网格的话，(30)中数学运算完成了“鸡-兔系”到“A-B套餐系”的坐标系转化。

$(10, 10)$ 是向量 \mathbf{a} 在“鸡-兔系”的坐标。

而2份A套餐、4份B套餐相当于 $(2, 4)$ 是向量 \mathbf{a} 在“A-B套餐系”的坐标。

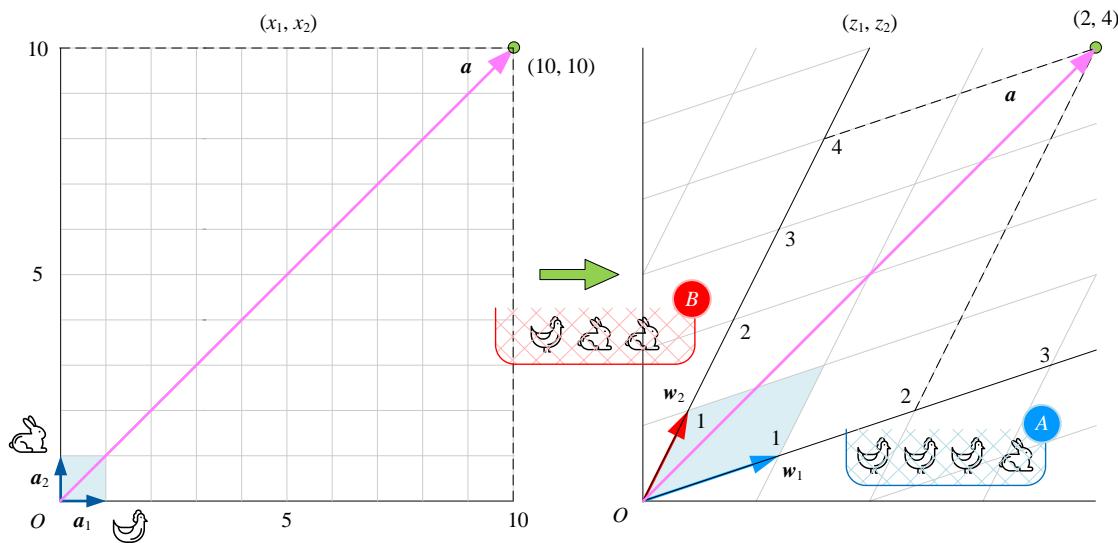


图9. 坐标系转换，“鸡-兔系”到“A-B套餐系”

基底变换

农夫回想，不管是从“头、脚系”到“鸡-兔系”，还是从“鸡-兔系”到“A-B套餐系”，都叫做**基底变换**(change of basis)。

对于向量 \mathbf{a} ，在基底 $\{\mathbf{a}_1, \mathbf{a}_2\}$ 下，坐标值为 $[x_1, x_2]^T$ ：

$$\mathbf{a} = \mathbf{x} = \mathbf{I}\mathbf{x} = [\mathbf{a}_1 \quad \mathbf{a}_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \mathbf{a}_1 + x_2 \mathbf{a}_2 \quad (31)$$

也就是：

$$\mathbf{a} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (32)$$

同一个向量 \mathbf{a} , 在基底 $\{\mathbf{w}_1, \mathbf{w}_2\}$ 下, 坐标值为 $[z_1, z_2]^T$:

$$\mathbf{a} = \mathbf{W}\mathbf{z} = \underbrace{\begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}}_z = z_1 \mathbf{w}_1 + z_2 \mathbf{w}_2 \quad (33)$$

即:

$$\mathbf{a} = \underbrace{\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}}_{\mathbf{W}} \underbrace{\begin{bmatrix} 2 \\ 4 \end{bmatrix}}_z = \begin{bmatrix} 10 \\ 10 \end{bmatrix} \quad (34)$$

联立 (31) 和 (33) 得到:

$$\mathbf{x} = \mathbf{W}\mathbf{z} \quad (35)$$

即,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 \end{bmatrix}}_{\mathbf{W}} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad (36)$$

新坐标 \mathbf{z} , 可以通过下式得到:

$$\mathbf{z} = \mathbf{W}^{-1}\mathbf{x} \quad (37)$$

也就是说, \mathbf{W} 是新旧坐标转换的桥梁。如图 9 所示, 转换前后, 网格形状发生变化, 但是平面还是那个平面。

“线性代数真是有趣、有用! ”, 农夫喃喃自语。



Bk3_Ch23_3.py 绘制本节两个坐标系网格图像。

23.5 套餐转换：基底转换

前来买鸡兔的村民在小贩周围越聚越多, 大家都说套餐 A 和 B 组合太繁琐, 纷纷抱怨。

为了方便村民买鸡兔, 小贩推出两个新套餐 C 和 D: 套餐 C, 两只小鸡; 套餐 D, 两只小兔。也就是说, 鸡兔都是成对贩售。

农夫决定用刚刚学过的基底转换思路来看看这个新基底。

令第三个基底 $\{\mathbf{v}_1, \mathbf{v}_2\}$ 代表“C-D 套餐系”。在基底 $\{\mathbf{v}_1, \mathbf{v}_2\}$ 中, 向量 \mathbf{a} 可以写成:

$$\mathbf{a} = \mathbf{Vs} = \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}}_V \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = s_1 \mathbf{v}_1 + s_2 \mathbf{v}_2 \quad (38)$$

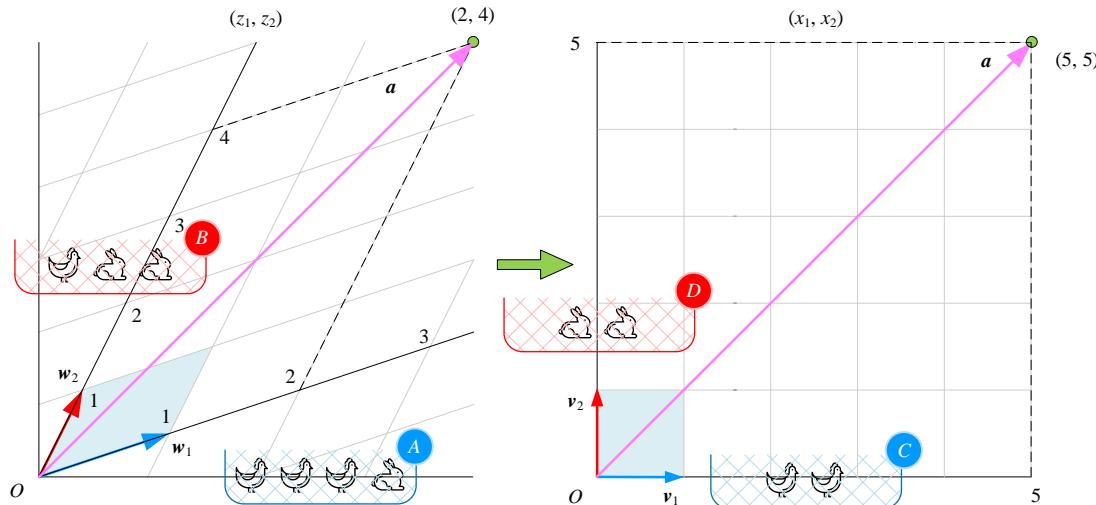


图 10. 套餐转换，“A-B 套餐系”到“C-D 套餐系”

联立 (33) 和 (38), 得到:

$$\mathbf{Wz} = \mathbf{Vs} \quad (39)$$

也就是说, s 可以通过下式得到:

$$\mathbf{s} = \mathbf{V}^{-1} \mathbf{Wz} \quad (40)$$

而 \mathbf{V} 为:

$$\mathbf{V} = \underbrace{\begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 \end{bmatrix}}_V = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \quad (41)$$

这样, 向量 a 从 $\{w_1, w_2\}$ 基底到 $\{v_1, v_2\}$ 基底, 新坐标 s 为:

$$\mathbf{s} = \underbrace{\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}}_V^{-1} \underbrace{\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}}_W \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad (42)$$

也就是说, 农夫想要买 10 只鸡、10 只兔的话, 需要 5 份套餐 C 和 5 份套餐 D。

23.6 猪引发的投影问题

农夫突然改了主意，他对小贩说，我想买 10 只鸡、10 只兔，还要买 5 只猪！

小贩很无奈，说小猪早就卖断货了。

农夫略有所思，说了句，“我和你之间，存在 5 只猪的距离。”

从向量角度，农夫自己想买 10 只鸡、10 只兔、5 只猪，可以写成向量 y ：

$$\mathbf{y} = \begin{bmatrix} 10 \\ 10 \\ 5 \end{bmatrix} \quad (43)$$

然而，小贩提供的“ $A-B$ 套餐”只能满足农夫部分需求，记做向量 a ：

$$\mathbf{a} = x_1 \mathbf{w}_1 + x_2 \mathbf{w}_2 = x_1 \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix} \quad (44)$$

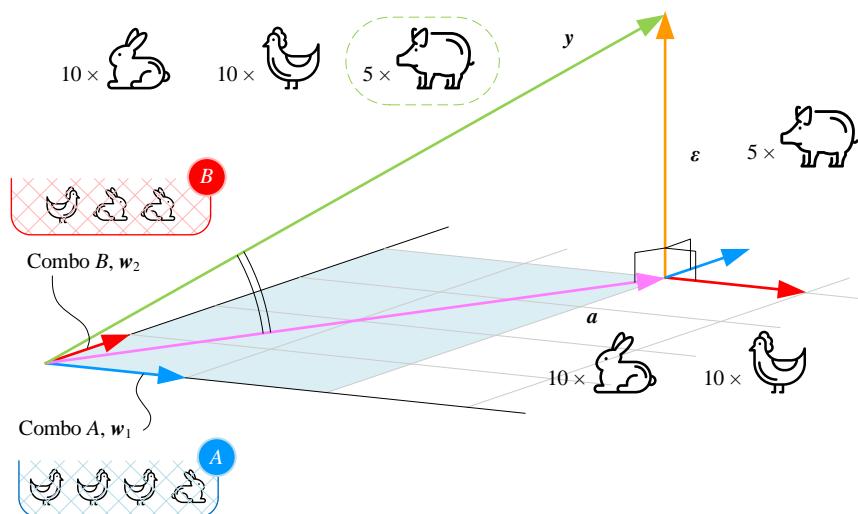


图 11. 农夫的需求和小贩提供的“ $A-B$ 套餐”平面存在 5 只猪的距离

农夫的需求 y 和 a 的“差距”记做 ϵ ，计算得到具体值：

$$\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{a} = \begin{bmatrix} 10 \\ 10 \\ 5 \end{bmatrix} - \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} \quad (45)$$

垂直

如图 11 所示，容易发现 ε 垂直于 w_1 、 w_2 、 a 。下面，农夫用刚学的向量内积证明一下。

首先， ε 垂直于 w_1 ：

$$w_1 \cdot \varepsilon = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = 3 \times 0 + 1 \times 0 + 0 \times 5 = 0 \quad (46)$$

ε 垂直于 w_2 ：

$$w_2 \cdot \varepsilon = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = 1 \times 0 + 2 \times 0 + 0 \times 5 = 0 \quad (47)$$

ε 垂直于 a ：

$$a \cdot \varepsilon = \begin{bmatrix} 10 \\ 10 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} = 10 \times 0 + 10 \times 0 + 0 \times 5 = 0 \quad (48)$$

也就是说， ε 垂直于 w_1 和 w_2 张成的平面。

从投影的角度来看，向量 y 在“ $A-B$ 套餐”平面的投影为 a 。

“真是有向量的地方，就有几何啊！”，这是农夫自己学习线性代数悟出的一句真经。

23.7 黄鼠狼惊魂夜：“鸡飞兔脱”与超定方程组

夜黑风高，农夫突然听到鸡叫犬吠！

他赶紧捡了件衣服披在身上，提起油灯、夺门而出。在赶去鸡窝路上，他发现了黄鼠狼的脚印，“大事不妙！”

农夫慌忙跑到鸡兔窝，看到鸡飞兔跳、惊慌失措。

担心黄鼠狼抓走了鸡兔，农夫心急如焚，他举高油灯，凑近笼子，数了又数。几遍下来，数字都对不上，自己更是头晕眼花。

他找来隔壁的甲、乙、丙、丁四人，让甲、乙数头，让丙、丁数脚。过了一阵，甲说有 30 个头，乙说有 35 个头；丙说有 90 只脚，丁说有 110 只脚。

这可难坏了农夫，他可怎么估算鸡兔各自的数量？

他决定也用线性代数工具试试。

农夫先列出来方程组：

$$\begin{cases} x_1 + x_2 = 30 \\ x_1 + x_2 = 35 \\ 2x_1 + 4x_2 = 90 \\ 2x_1 + 4x_2 = 110 \end{cases} \quad (49)$$

农夫首先拿出图解法这个利器！

图 12 所示为四条直线对应的图像，发现它们一共存在 4 个交点，没有一组确切解。

代数角度，上述方程组叫做**超定方程组** (overdetermined system)。两个方程两个未知数，显然所需的方程组远超未知数数量。

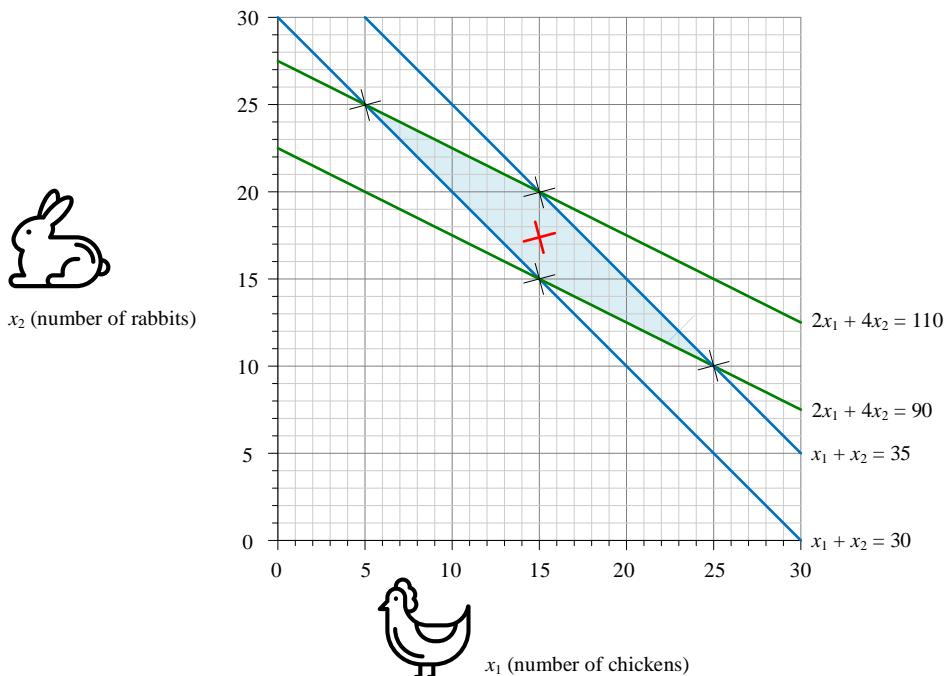


图 12. 超定方程组图像

农夫将 (49) 写成矩阵的形式：

$$\underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 4 \\ 2 & 4 \end{bmatrix}}_A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 30 \\ 35 \\ 90 \\ 110 \end{bmatrix} \quad (50)$$

也就是说：

$$\mathbf{Ax} = \mathbf{b} \quad (51)$$

A 不是方阵，显然不存在逆。

在《线性代数》这本经典中，农夫发现了一个全新的解法。他将 (51) 左右分别乘 A^T ：

$$A^T A x = A^T b \quad (52)$$

$A^T A$ 为 2×2 方阵，且存在逆。

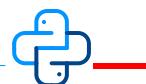
这样，(52) 可以整理为：

$$x = (A^T A)^{-1} A^T b \quad (53)$$

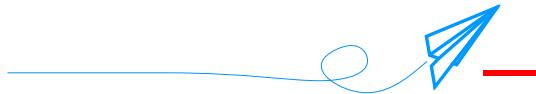
代入具体值，得到 x 的估算解：

$$x = \left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 4 \\ 2 & 4 \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 4 \\ 2 & 4 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 2 & 4 \\ 2 & 4 \end{bmatrix}^T \begin{bmatrix} 30 \\ 35 \\ 90 \\ 110 \end{bmatrix} = \begin{bmatrix} 10 & 18 \\ 18 & 34 \end{bmatrix}^{-1} \begin{bmatrix} 465 \\ 865 \end{bmatrix} = \begin{bmatrix} 15 \\ 17.5 \end{bmatrix} \quad (54)$$

农夫发现这个解恰好在图 12 四个交点构成平行四边形的中心位置，“神奇，真是神奇！”



Bk3_Ch23_4.py 完成上述矩阵运算。



微风丝丝缕缕，细雨点点滴滴。

微风夹着细雨，掠过田间地头，摇晃着杨柳梢，吹洗一池荷花。杨柳依依，荷风香气。微风轻轻悄悄地划过鸡舍兔笼，踮着脚尖走过睡熟的牧童。微风看了一眼灯下苦读的农夫，舞动着农夫书桌上跳跃的烛火。

折腾了一天一夜，小村似乎安静下来。

殊不知，大风起兮，云飞扬，远处一场风暴正在酝酿。

未完待续。

24

A Story of OLS Linear Regression

鸡兔同笼 2

之线性回归风暴



只有带着对数学纯粹的爱去接近她，数学才会向你展开它的神秘所在。

Mathematics reveals its secrets only to those who approach it with pure love, for its own beauty.

——阿基米德 (Archimedes) | 古希腊数学家、物理学家 | 287 ~ 212 BC



- ◀ matplotlib.pyplot.contour() 绘制等高线图
- ◀ matplotlib.pyplot.scatter() 绘制散点图
- ◀ numpy.array() 创建 array 数据类型
- ◀ numpy.linalg.inv() 矩阵求逆
- ◀ numpy.linalg.solve() 求解线性方程组
- ◀ numpy.linspace() 产生连续均匀向量数值
- ◀ numpy.meshgrid() 创建网格化数据
- ◀ numpy.random.randint() 产生随机整数
- ◀ numpy.random.seed() 设定初始化随机状态
- ◀ plot_wireframe() 绘制三维单色线框图
- ◀ seaborn.scatterplot() 绘制散点图
- ◀ sympy.abc 引入符号变量
- ◀ sympy.diff() 求解符号导数和偏导解析式
- ◀ sympy.evalf() 将符号解析式中未知量替换为具体数值
- ◀ sympy.simplify() 简化代数式
- ◀ sympy.solvers.solve() 符号方程求根
- ◀ sympy.symbols() 定义符号变量
- ◀ sympy.utilities.lambdify.lambdify() 将符号代数式转化为函数

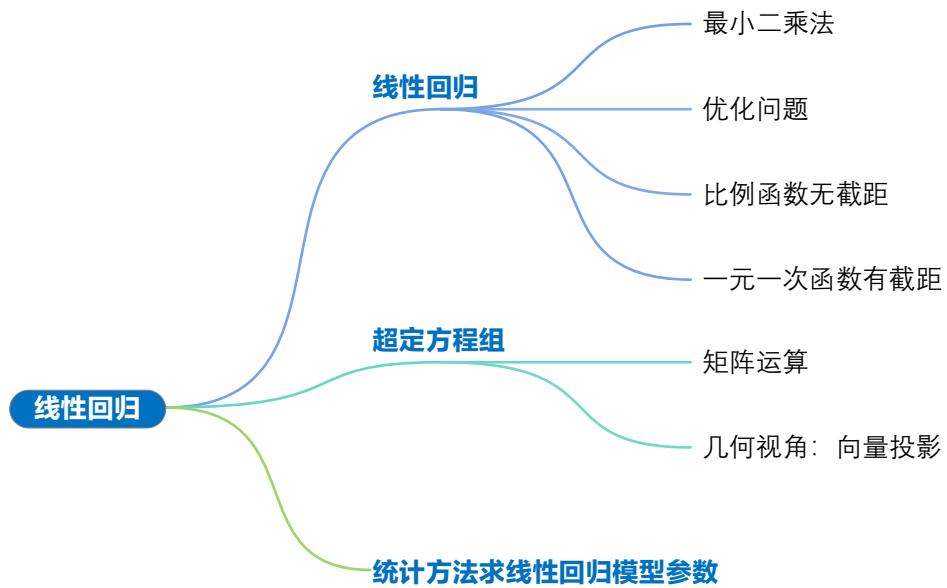
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com



24.1 鸡兔数量的有趣关系

清江一曲抱村流，长夏江村事事幽。

舶来的线性代数知识悄悄地改变着小村，村民们凡事都要用这个数学工具探究一番。

大家这次盯上了一个养鸡养兔的小妙招。老人常言“两鸡一兔，百毒不入”。也就是说，不管最开始养多少鸡兔，当鸡兔大概达到 2:1 这个比例，便达到某种神奇的平衡，鸡兔都健健康康。

农夫决定一探究竟，他搜集村中 20 位养鸡大户的鸡兔数量，总结在表 1。

表 1. 20 户农户鸡兔数量关系

养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88	44	33	57	16	22	52
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52	20	18	33	12	11	28

将表 1 数据以散点方式绘制在方格纸上得到图 1。老农隐隐觉得这个 2:1 的比例关系好像的确存在。

但是，农夫并不满足于此，他想找到鸡兔达到平衡时确切的数学关系。于是乎，他想到了比例函数和一元函数，决心探究一番。

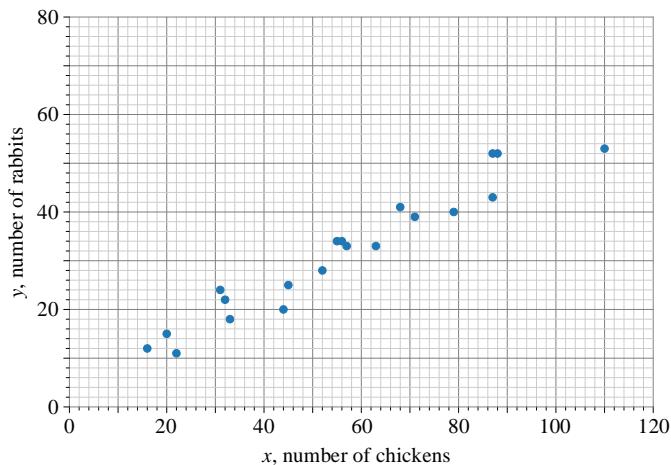


图 1. 平衡时，各家鸡兔数量关系

24.2 试试比例函数： $y = ax$

观察图 1，农夫首先想到用比例函数。

比例函数假设平衡时鸡兔数量似乎呈现某种比例关系：

$$\hat{y} = ax \quad (1)$$

其中， a 为比例系数。为了区分数据 y ， \hat{y} 上加了个帽子表示预测。

农夫在方格纸上，用红笔画出一系列通过原点的斜线，得到图 2。

老农先是觉得 a 取 0.5 比较好，但是又觉得 a 取 0.6 也不差。他隐约觉得 a 应该在 0.5 和 0.6 之间。

如何找到合理的 a 值？

这个问题让他陷入了沉思。显然，他需要找到一条红线足够靠近图 2 所有散点。那么，问题来了——如何量化“足够靠近”？

他决定先找几个值试试看。

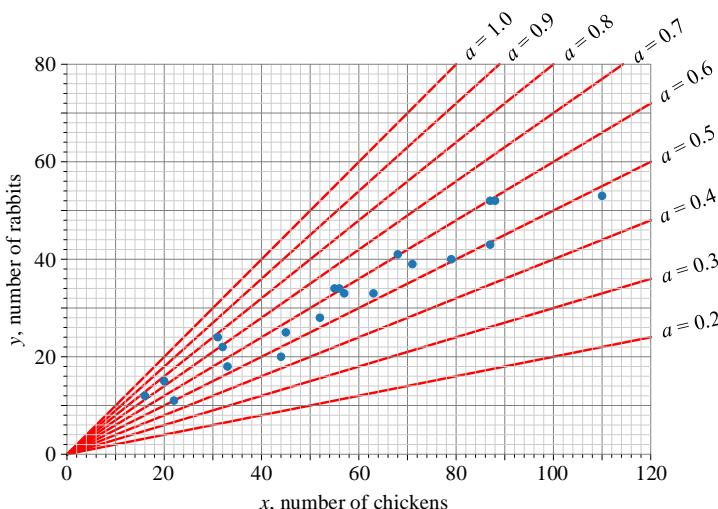


图 2. 平衡时，各家鸡兔数量好像呈现某种比例关系

$a = 0.4$

农夫先试了比例值 $a = 0.4$ ，这时比例函数为：

$$\hat{y} = 0.4x \quad (2)$$

将 $x = 110$ （鸡的数量）代入 (2)，得到 44（兔的数量）这个预测值。

$$\hat{y}|_{x=110} = 0.4 \times 110 = 44 \quad (3)$$

当 $x = 110$ 时，真实值 y 和预测值 \hat{y} 两者的误差 e 为：

$$e|_{x=100} = y - \hat{y} = 53 - 44 = 9 \quad (4)$$

农夫觉得从这个误差值入手，可能会找到合适的 a 值，并确定一条合理的比例函数。

于是乎，农夫开始计算 $\hat{y} = 0.4x$ 这个比例函数条件下图 1 中每个点的误差值。

最后，他得到图 3。图中，竖直黄色线段代表实际数据和比例函数估值之间的误差，也就是不同 x 对应的 e 。

农夫翻阅舶来的数学典籍，发现了**最小二乘法** (Least Squares 或 Least Squares Estimator)。仔仔细研读后，他决定拿来一试。

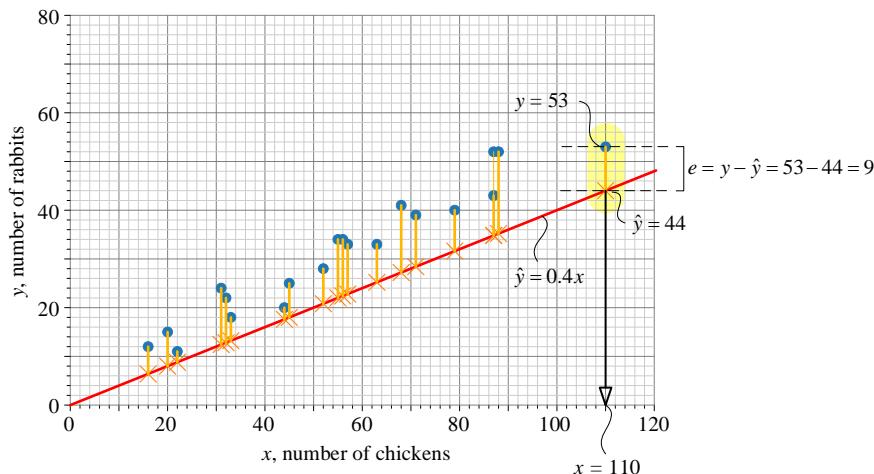


图 3. $a = 0.4$ 时，实际数据和比例函数估值之间的误差

24.3 最小二乘法

书上写道：“最小二乘法通过最小化误差的平方和，寻找数据的最佳回归参数匹配。”

误差平方和最小化

农夫已经得到了一系列 e 值，只需要对 e 平方！

他把计算得到的分步数据记录在表 2 中。表第一、二行数值分别为鸡、兔实际数量，第三行为 $\hat{y} = 0.4x$ 估算得到的兔子数量，第四行为误差 $e = y - \hat{y}$ ，即实际兔数减去估算兔数，第五行为误差的平方值 e^2 。

表 2 第五行 e^2 求和得到误差平方和为 1756.28。

表 2. a 取 0.4 时，估计值、误差、误差平方

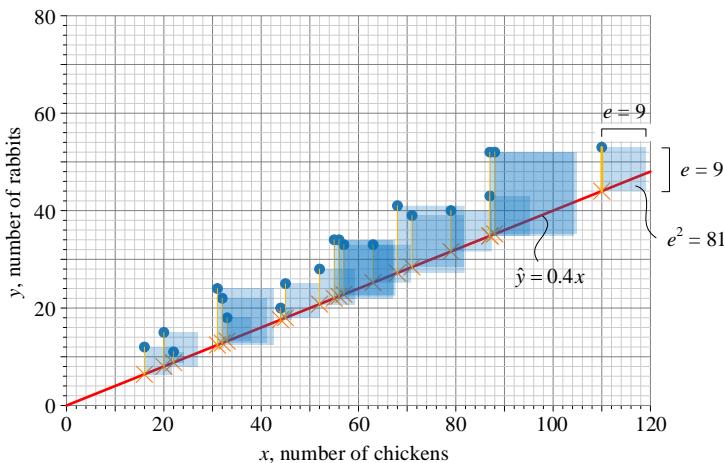
养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52
$\hat{y} = 0.4x$ 估算兔数	12.8	44	28.4	31.6	18	8	22.4	22	34.8	27.2	34.8	25.2	12.4	35.2
误差 e	9.2	9	10.6	8.4	7	7	11.6	12	17.2	13.8	8.2	7.8	11.6	16.8
误差平方 e^2	84.6	81	112.3	70.5	49	49	134.5	144	295.8	190.4	67.2	60.8	134.5	282.2

农夫突然意识到， e^2 不就是以 e 的绝对值为边长的正方形面积嘛！真是“行到水穷处，坐看云起时。”

有了这个几何视角，他绘制得到了图 4。图 4 中所有的正方形的边长为不同 x 位置的误差 e 绝对值。将这些蓝色正方形面积相加得到面积和，即误差之和：

$$\sum_{i=1}^{20} (e^{(i)})^2 = \sum_1^{20} (y^{(i)} - \hat{y}^{(i)})^2 = \sum_1^{20} (y^{(i)} - ax^{(i)})^2 \quad (5)$$

找到让上式值最小的 a ，就可以让图 4 中正方形面积之和最小。

图 4. $a = 0.4$ 时，可视化误差平方

$a = 0.5$

他决定再试几个值，比如 $a = 0.5$ 时，比例函数为：

$$\hat{y} = 0.5x \quad (6)$$

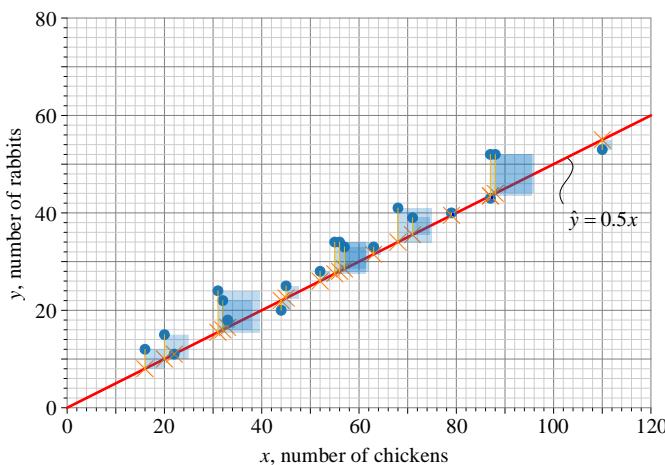
表 3 给出 a 取 0.5 时，不同 x 对应的估计值 \hat{y} 、误差 e 、误差平方 e^2 。

经过计算可以发现 (6) 这个比例函数模型条件下，误差平方和为 422。

几何角度来看，图 5 中的正方形面积之和看上去确实比图 4 要小。

表 3. a 取 0.5 时，估计值、误差、误差平方

养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52
$\hat{y} = 0.5x$ 估算兔数	16	55	35.5	39.5	22.5	10	28	27.5	43.5	34	43.5	31.5	15.5	44
误差 e	6	-2	3.5	0.5	2.5	5	6	6.5	8.5	7	-0.5	1.5	8.5	8
误差平方 e^2	36	4	12.25	0.25	6.25	25	36	42.25	72.25	49	0.25	2.25	72.25	64

图 5. $a = 0.5$ 时，可视化误差平方 **$a = 0.6$**

农夫又试了试 $a = 0.6$ ，比例函数为：

$$\hat{y} = 0.6x \quad (7)$$

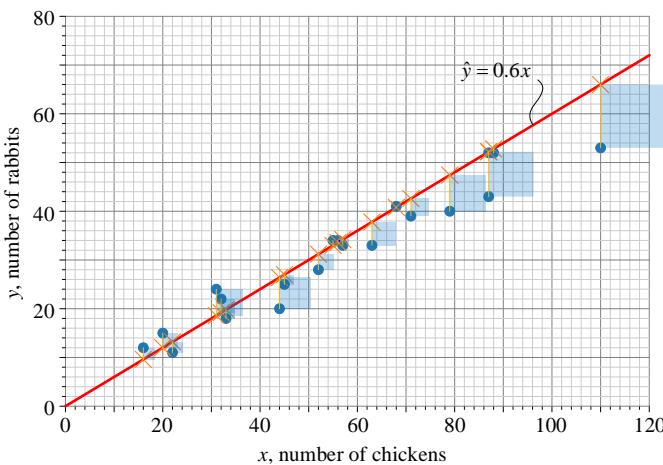
经过表 4 计算求得误差平方和为 396.28。图 6 可视化误差平方和。

农夫感觉到，似乎在 0.5 和 0.6 之间存在一个更好的 a ，能够让误差平方和最小。

但是，这样徒手计算，一个一个值算，终究不是办法。

表 4. a 取 0.6 时，估计值、误差、误差平方

养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52
$\hat{y} = 0.6x$ 估算兔数	19.2	66	42.6	47.4	27	12	33.6	33	52.2	40.8	52.2	37.8	18.6	52.8
误差 e	2.8	-13	-3.6	-7.4	-2	3	0.4	1	-0.2	0.2	-9.2	-4.8	5.4	-0.8
误差平方 e^2	7.84	169	12.96	54.76	4	9	0.16	1	0.04	0.04	84.64	23.04	29.16	0.64

图 6. $a = 0.6$ 时，可视化误差平方

目标函数

观察 (5)，他发现 $x^{(i)}$ 和 $y^{(i)}$ 都是给定数值，而式中唯一的变量就是 a 。也就是说，把 a 看做一个未知数，(5) 可以写成一个函数 $f(a)$ ：

$$f(a) = \sum_{i=1}^{20} (y^{(i)} - ax^{(i)})^2 \quad (8)$$

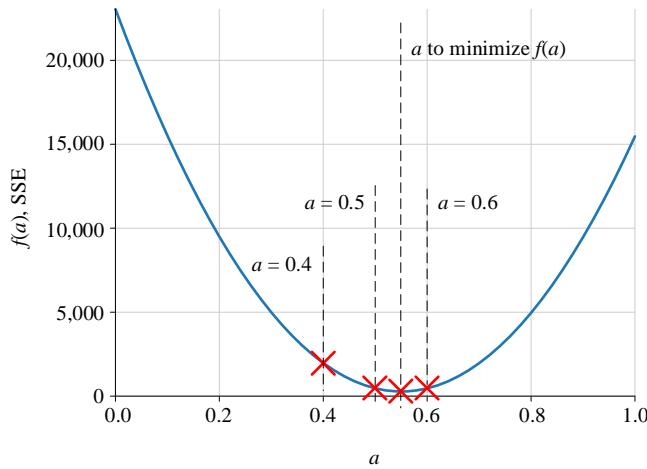
而最小化误差对应的就是让上述函数值取得最小值！农夫想到这里，高兴地不住拍手。

农夫把所有的 $x^{(i)}$ 和 $y^{(i)}$ 代入上式，整理并得到函数具体解析式：

$$f(a) = 65428a^2 - 72228a + 20179 \quad (9)$$

他惊奇地发现，竟然得到了一元二次函数！这个函数，我懂啊！

如图 7 所示，这个一元二次函数的图像是一条开口朝上的抛物线，具有凸性。显然，函数在对称轴处取得最小值。而 (9) 这个一元二次函数就是优化问题中的目标函数，优化变量为 a 。

图 7. 函数 $f(a)$ 图像

解析法求解优化问题

利用导数这个数学工具，对 $f(a)$ 求一阶导数，得到 $f'(a)$ ：

$$f'(a) = 130856a - 72228 \quad (10)$$

$f'(a) = 0$ 得到 $f(a)$ 取得最小值时 a 的值，记做 a^* ：

$$a^* = \frac{18057}{32714} \approx 0.552 \quad (11)$$

这个 a^* 就是农夫要找的最佳 a 值，它让误差平方和最小。

此时，对应的最优比例函数为：

$$\hat{y} = 0.552x \quad (12)$$

带回检验

农夫决定用“土办法”再算算 a^* 对应的估计值、误差、误差平方这几个数值，他得到表 5。

此时，误差平方和为 245.32，明显小于 $a = 0.5$ 或 $a = 0.6$ 这两种情况。

他不忘绘制图 6，看看正方形的面积到底怎样。

表 5. a 取 0.5504 时，估计值、误差、误差平方

养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52
$\hat{y} = 0.5504x$ 估算兔数	17.6	60.5	39.1	43.5	24.8	11.0	30.8	30.3	47.9	37.4	47.9	34.7	17.1	48.4
误差 e	4.4	-7.5	-0.1	-3.5	0.2	4.0	3.2	3.7	4.1	3.6	-4.9	-1.7	6.9	3.6
误差平方 e^2	19.2	56.9	0.0	12.1	0.1	15.9	10.1	13.9	16.9	12.8	23.9	2.8	48.1	12.7

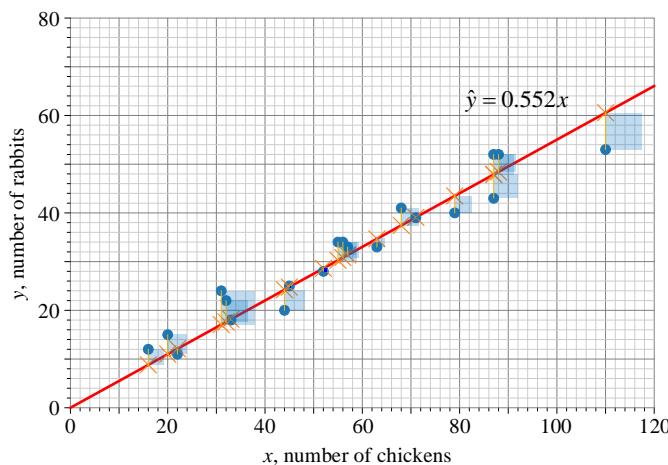


图 8. $a = 0.552$ 时，可视化误差平方

农夫如获至宝，不住地说“最小二乘法，好！真好！”。

他回过头再次翻阅数学典籍，又仔仔细细把最小二乘方法反复研读几遍。兴奋之余，他想让自己的数学模型再复杂一点，决定试试一元一次函数。



Bk3_Ch24_1.py 绘制本节图像，并求解最优化问题。

24.4 再试试一次函数： $y = ax + b$

农夫知道，比例函数通过原点，也就是纵轴截距为 0。而一元函数则没有这个限制。

他决定试一下如下这个一元函数，看看是否有更好的结果：

$$\hat{y} = ax + b \quad (13)$$

这个一元函数对应的误差平方和为：

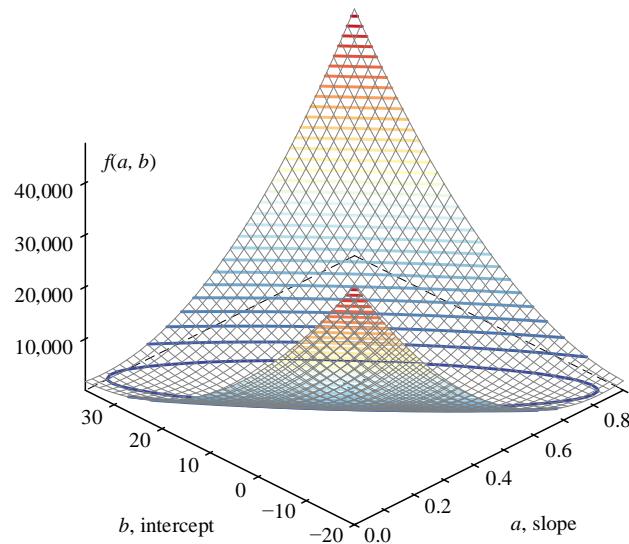
$$\sum_{i=1}^{20} (e^{(i)})^2 = \sum_{i=1}^{20} (y^{(i)} - \hat{y}^{(i)})^2 = \sum_{i=1}^{20} (y^{(i)} - ax^{(i)} - b)^2 \quad (14)$$

式中， $x^{(i)}$ 和 $y^{(i)}$ 都是给定的样本数据。也就是说，上式有两个自变量，有两个需要优化的参数 a 、 b 。

农夫还是决定暴力求解一番，将 $x^{(i)}$ 和 $y^{(i)}$ 代入 (14)，整理并得到二元函数 $f(a, b)$ 解析式：

$$f(a, b) = 65428a^2 + 1784ab - 72228a + 14b^2 - 1014b + 20179 \quad (15)$$

这不就是二元二次函数，我也懂啊！几何角度不就是个开口朝上的旋转椭圆面嘛！农夫再次惊叹数学的精妙！

图 9. 误差平方和 $f(a, b)$ 随 a 、 b 变化构造的开口向上抛物曲面

用偏导求极值点

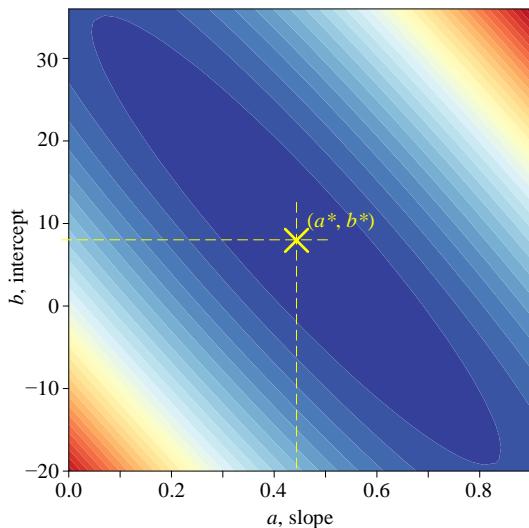
计算 $f(a, b)$ 最小值极值点处，利用 $f(a, b)$ 对 a 、 b 求偏导为 0 为条件，构造两个等式：

$$\begin{cases} \frac{\partial f}{\partial a} = 130856a + 1784b - 72228 = 0 \\ \frac{\partial f}{\partial b} = 1784a + 28b - 1014 = 0 \end{cases} \quad (16)$$

联立等式，求得最优解：

$$\begin{cases} a^* = \frac{513}{1157} \approx 0.4434 \\ b^* = \frac{18429}{2314} \approx 7.9641 \end{cases} \quad (17)$$

图 10 告诉我们这个最优解就在旋转椭圆中心位置。农夫看着图 10，嘴里叨叨着“椭圆真是个好东西！哪都离不开它！”

图 10. $f(a, b)$ 平面等高线和最优解位置

(17) 对应的一次函数：

$$\hat{y} = 0.4434x + 7.9641 \quad (18)$$

这就是农夫要找的最佳一次函数！

带回检验

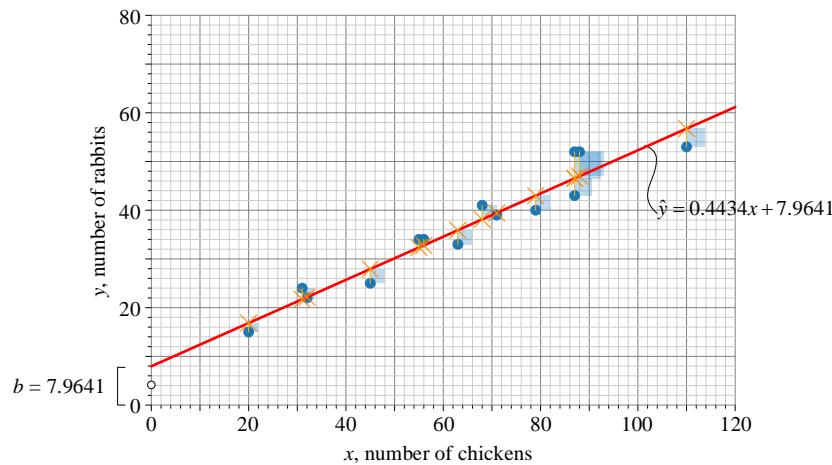
农夫还是想用“土办法”再验算一遍！

他用(18)一步步仔细运算，并将分步结果记录在表6中。农夫最终求得误差平方和为128.67，这比之前比例函数对应的误差平方和还要小。

他不怕麻烦，又画了图11。图中，一次函数的截距为正。

表 6. $a = 0.4434$ 、 $b = 7.9641$ 时，估计值、误差、误差平方

养鸡数量 x	32	110	71	79	45	20	56	55	87	68	87	63	31	88
养兔数量 y	22	53	39	40	25	15	34	34	52	41	43	33	24	52
$\hat{y} = 0.4863x + 4.312$	19.9	57.8	38.8	42.7	26.2	14.0	31.5	31.1	46.6	37.4	46.6	34.9	19.4	47.1
误差 e	2.1	-4.8	0.2	-2.7	-1.2	1.0	2.5	2.9	5.4	3.6	-3.6	-1.9	4.6	4.9
误差平方 e^2	4.5	23.1	0.0	7.5	1.4	0.9	6.0	8.7	28.9	13.1	13.1	3.8	21.3	23.9

图 11. $a = 0.4434$ 、 $b = 7.9641$ 时，可视化误差平方

Bk3_Ch24_2.py 绘制本节图像，并求解优化问题。



在 Bk3_Ch24_2.py 基础上，我们做了一个 App，大家可以输入不同的一次函数 a 和 b 两个参数，绘制直线并和线性回归直线比较误差结果。请参考 Streamlit_Bk3_Ch24_2.py。

24.5 再探黄鼠狼惊魂夜：超定方程组

突然间，一道灵光闪过！

农夫回想，那夜黄鼠狼来偷鸡抓兔，邻居甲、乙、丙、丁四人数头数、脚数时，为了估算鸡兔数量，他采用的舶来线性代数典籍中的超定方程组的求解方法。

回过头来看自己手中的线性回归问题，“这不也是一个超定方程组吗？！”

比例函数

他立刻摊开纸，把表 1 数据写成列向量形式：

$$\mathbf{x} = \begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 22 \\ 53 \\ \vdots \\ 28 \end{bmatrix} \quad (19)$$

农夫将比例函数模型写成：

$$\mathbf{y} = a\mathbf{x} \quad (20)$$

即

$$\begin{bmatrix} 22 \\ 53 \\ \vdots \\ 28 \end{bmatrix} = a \begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix} \quad (21)$$

只有一个未知数 a ，但是方程组有 20 个方程，这显然也是一个超定方程组！

农夫顿时兴奋起来，他用黄鼠狼惊魂夜一模一样的方法求解：

$$a = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \mathbf{y} \quad (22)$$

实际上， $\mathbf{x}^T \mathbf{x}$ 是一个 1×1 矩阵，也就是一个数字，即标量。它的逆就是 $\mathbf{x}^T \mathbf{x}$ 这个数字的倒数。

将 \mathbf{x} 和 \mathbf{y} 具体数值代入 (22)，得到：

$$a = \left(\begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix}^T @ \begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix} \right)^{-1} @ \begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix}^T @ \begin{bmatrix} 22 \\ 53 \\ \vdots \\ 28 \end{bmatrix} = 0.552 \quad (23)$$

农夫惊呼，“得来全不费工夫啊！”这个结果和他用最小二乘法得到结果完全一致。

一元函数

灵光再现，他立刻疾步多取回些纸笔，将一元函数这个模型也写成矩阵形式：

$$\begin{bmatrix} 22 \\ 53 \\ \vdots \\ 28 \end{bmatrix} = a \begin{bmatrix} 32 \\ 110 \\ \vdots \\ 52 \end{bmatrix} + b \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad (24)$$

即，

$$\hat{\mathbf{y}} = a\mathbf{x} + b\mathbf{I} \quad (25)$$

\mathbf{I} 叫做全 1 列向量。

只有二个未知数 a 、 b ，但是方程组有 20 个方程，这明显也是一个超定方程组。

将 (25) 写成：

$$\hat{\mathbf{y}} = \underbrace{[\mathbf{I} \quad \mathbf{x}]}_{\mathbf{X}} \begin{bmatrix} b \\ a \end{bmatrix} \quad (26)$$

令，

$$\mathbf{X} = [\mathbf{I} \quad \mathbf{x}] = \begin{bmatrix} 1 & 32 \\ 1 & 110 \\ \vdots & \vdots \\ 1 & 52 \end{bmatrix} \quad (27)$$

(26) 则写成：

$$\hat{\mathbf{y}} = \mathbf{X} \begin{bmatrix} b \\ a \end{bmatrix} \quad (28)$$

求解超定方程组，得到：

$$\begin{bmatrix} b \\ a \end{bmatrix} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (29)$$

将 \mathbf{X} 和 \mathbf{y} 具体数值代入 (29)，得到：

$$\begin{bmatrix} b \\ a \end{bmatrix} = \left(\begin{bmatrix} 1 & 32 \\ 1 & 110 \\ \vdots & \vdots \\ 1 & 52 \end{bmatrix}^T @ \begin{bmatrix} 1 & 32 \\ 1 & 110 \\ \vdots & \vdots \\ 1 & 52 \end{bmatrix} \right)^{-1} @ \left(\begin{bmatrix} 1 & 32 \\ 1 & 110 \\ \vdots & \vdots \\ 1 & 52 \end{bmatrix}^T @ \begin{bmatrix} 22 \\ 53 \\ \vdots \\ 28 \end{bmatrix} \right) = \begin{bmatrix} 14 & 892 \\ 892 & 65428 \end{bmatrix}^{-1} \begin{bmatrix} 507 \\ 36114 \end{bmatrix} = \begin{bmatrix} 7.9641 \\ 0.4434 \end{bmatrix} \quad (30)$$

几何视角

农夫知道，凡是向量的地方，就有几何！

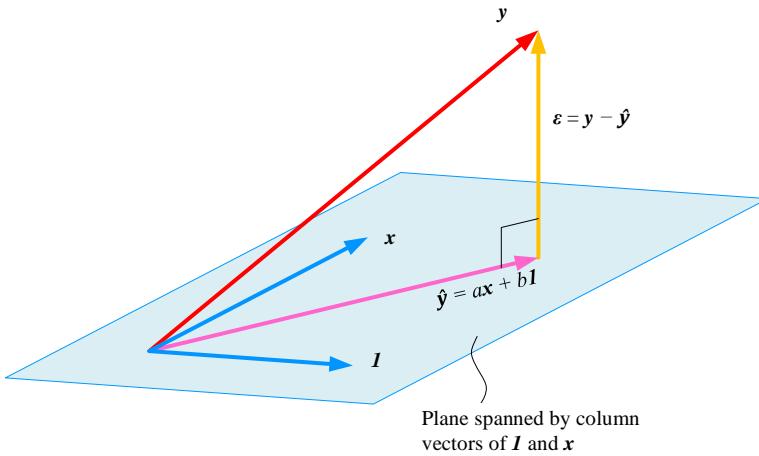


图 12. 几何角度解释一元最小二乘结果，二维平面

上述解法肯定可以通过几何角度解释。如图 12 所示，将 y 向量向 x 和 I 张成的平面 H 投影，得到结果为向量 \hat{y} ；而误差 ϵ 可以写成：

$$\epsilon = y - \hat{y} = y - (ax + bI) \quad (31)$$

误差 ϵ 显然垂直于 H ，即 ϵ 分别垂直 I 和 x 。

也就是说：

$$\begin{aligned} \epsilon \perp I &\Rightarrow I^T \epsilon = 0 \Rightarrow I^T(y - (ax + bI)) = 0 \\ \epsilon \perp x &\Rightarrow x^T \epsilon = 0 \Rightarrow x^T(y - (ax + bI)) = 0 \end{aligned} \quad (32)$$

以上两式合并：

$$\underbrace{[I \ x]}_X^T \left(y - X \begin{bmatrix} b \\ a \end{bmatrix} \right) = 0 \quad (33)$$

整理得到：

$$X^T X \begin{bmatrix} b \\ a \end{bmatrix} = X^T y \quad (34)$$

等式左右分别左边乘以 $X^T X$ 的逆，不就得到 (29) 嘛！

“嗟夫！我的神仙姑奶奶！”这个结果让农夫惊呆了半晌。

带回检验

醒过神来，他把比例函数和一次函数对应的图像都画在一幅图上，如图 13。

“朝闻道夕死可矣！”

线性代数的魅力让农夫彻底折服。

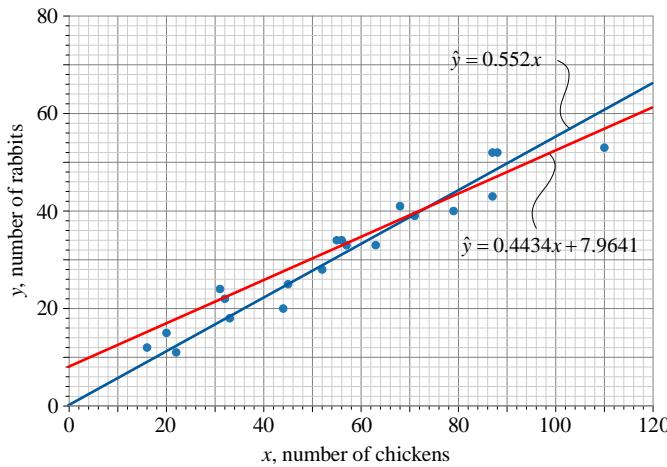


图 13. 比较比例模型和线性模型



Bk3_Ch24_3.py 求解本节优化问题，并绘制图 13。

24.6 统计方法求解回归参数

突然间，农夫想起前几日在一本叫做《概率统计》的数学典籍中一个有趣的公式，它赶忙取来典籍，找到如下这个公式：

$$y = \rho_{X,Y} \frac{\sigma_Y}{\sigma_X} (x - \mu_X) + \mu_Y = \underbrace{\rho_{X,Y} \frac{\sigma_Y}{\sigma_X} x}_{a} + \underbrace{\left(-\rho_{X,Y} \frac{\sigma_Y}{\sigma_X} \mu_X + \mu_Y \right)}_{b} \quad (35)$$

其中， μ_X 为 X 均值， μ_Y 为 Y 均值； σ_X 为 X 的标准差， σ_Y 为 Y 的标准差； $\rho_{X,Y}$ 为 X 和 Y 的相关性系数。

农夫意识到，从统计角度，也可以用 (35) 计算一元一次线性回归模型。

他赶紧利用表 1 数据计算得到均值、标准差和相关性系数等值：

$$\begin{cases} \mu_X = 63.714 \\ \mu_Y = 36.214 \end{cases}, \quad \begin{cases} \sigma_X = 25.712 \\ \sigma_Y = 11.826 \end{cases}, \quad \rho_{X,Y} = 0.96397 \quad (36)$$

这样可以计算得到参数 a 和 b ：

$$a = \rho_{x,y} \frac{\sigma_y}{\sigma_x} = 0.96397 \times \frac{11.826}{25.712} = 0.4434 \quad (37)$$

$$b = -\rho_{x,y} \frac{\sigma_y}{\sigma_x} \mu_x + \mu_y = -0.96397 \times \frac{11.826}{25.712} \times 63.714 + 36.214 = 7.9641$$

这和前面的几种方法结果完全吻合！农夫顿悟，原来最小二乘法线性回归是几何、向量、优化、概率统计的完美合体！

他不忘绘制图 14 这幅图，农夫发现图中回归直线通过 (μ_x, μ_y) 这点。在《概率统计》这本书上，农夫又发现了图 15 这幅图。据书上所讲，图中椭圆和**二元高斯分布** (bivariate Gaussian distribution)、**条件概率** (conditional probability) 都有密切关系。农夫感慨，“书山有路啊，学海无涯啊！”

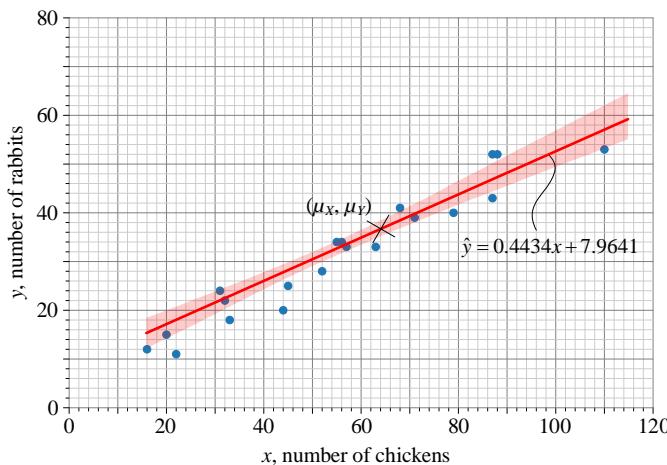


图 14. 利用统计方法获得线性模型

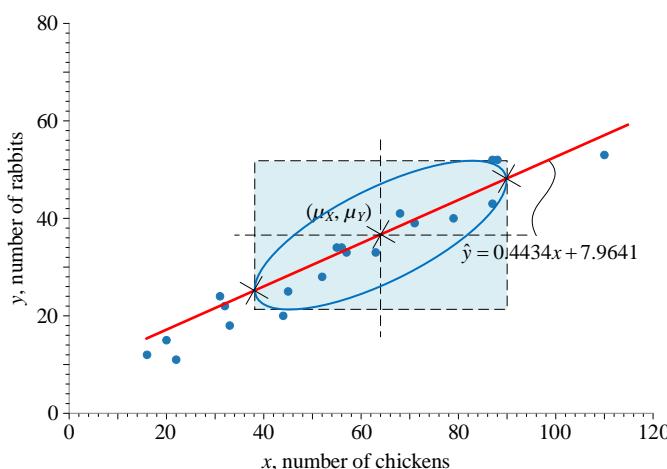
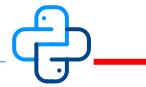


图 15. 条件概率视角



Bk3_Ch24_4.py 代码绘制图 14。



农夫落笔刹那，毫无防备之间，黑云压城城欲摧。

农夫赶忙起身关紧门窗，只见窗外云浪翻腾，道道电光从西方汹涌而来！闪电撕开天幕，大有列缺霹雳、丘峦崩摧之状。

瞬时，天河倾注，拳头大的雨滴敲击着大地，冲刷每一条沟壑，涤荡每一片浮尘。

农夫却毫无惧色，他喜出望外，仰天长啸道，“天上之水啊！上善若水啊！好水，好水！”

未完待续。

25

Probability Meets Linear Algebra

鸡兔同笼 3

鸡兔互变之马尔科夫奇妙夜



我们必须知道，我们终将知道。

Wir müssen wissen. Wir werden wissen.

We must know. We shall know.

—— 大卫·希尔伯特 (David Hilbert) | 德国数学家 | 1862 ~ 1943



- ◀ `numpy.diag()` 以一维数组的形式返回方阵的对角线元素，或将一维数组转换成对角阵
- ◀ `numpy.linalg.eig()` 特征值分解
- ◀ `numpy.linalg.inv()` 矩阵求逆
- ◀ `numpy.matrix()` 构造二维矩阵
- ◀ `numpy.meshgrid()` 产生网格化数据
- ◀ `numpy.vstack()` 垂直堆叠数组
- ◀ `seaborn.heatmap()` 绘制热图

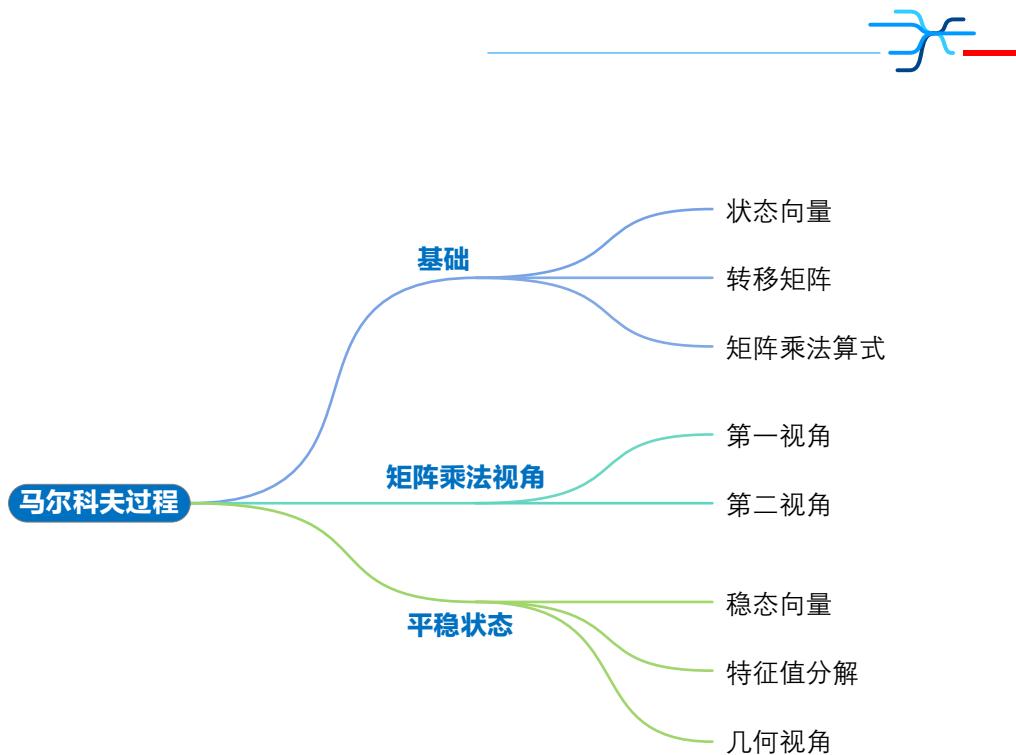
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com



25.1 鸡兔互变奇妙夜

怪哉，怪哉！

接连数月，村民发现一件奇事——夜深人静时，同笼鸡兔竟然互变！一些小兔变成小鸡，而一些小鸡变成小兔。

村民奔走相告，大家都惊呼，“我们都疯了！”

而一众村民中，农夫则显得处变不惊。在农夫眼里，村里发生的鸡兔互变像极了老子说的“祸兮，福之所倚；福兮，祸之所伏。”

农夫对村民说，“大家不要怕，恐惧都是来自于未知。我们必须知道，我们终将知道！福祸相生，是福不是祸，是祸躲不过。”

面对这个鸡兔互变的怪相，农夫决定用线性代数这个利器探究一番。

鸡兔互变过程图

农夫先是连续几日统计村里的鸡兔数量，他有个意想不到的发现——每晚有 30% 的小鸡变成小兔，其他小鸡不变；与此同时，每晚有 20% 小兔变成小鸡，其余小兔不变。变化前后鸡兔总数不变。

他先画了图 1 这幅图，用来描述鸡兔互变的比例。这个比例也就是概率值，即发生变化的可能性。

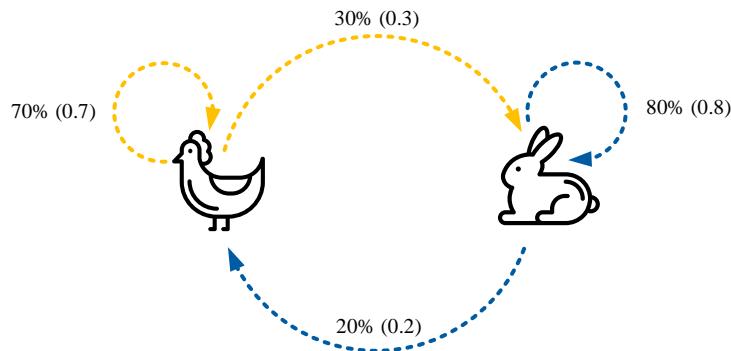


图 1. 鸡兔互变的比例

矩阵乘法

农夫想试试用矩阵乘法来描述这一过程。

第 k 天，鸡兔的比例用列向量 $\pi(k)$ 表示，比如：

$$\pi(k) = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} \quad (1)$$

其中， $\pi(k)$ 第一行元素代表小鸡的比例 (0.3, 30%)， π 第二行元素代表小兔的比例 (0.7, 70%)。

第 $k + 1$ 天，鸡兔的比例用列向量 $\pi(k+1)$ 表示。鸡兔互变的比例写成方阵 T ，这样 $k \rightarrow k + 1$ 变化过程可以写成：

$$k \rightarrow k + 1: T\pi(k) = \pi(k + 1) \quad (2)$$

农夫翻阅线性代数典籍时发现 T 和 π 都有自己专门的名称： T 叫**转移矩阵** (transition matrix)；列向量 π 叫做**状态向量** (state vector)。

而整个鸡兔互变的过程也有自己的名称——**马尔可夫过程** (Markov process)。

转移矩阵

鸡兔互变中，转移矩阵 T 为：

$$T = \begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix} \quad (3)$$

图 2 所示为转移矩阵 T 每个元素的具体含义。

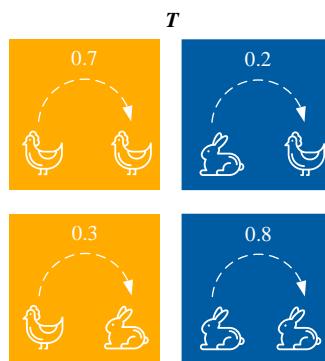


图 2. 转移矩阵 T

图 3 所示为用矩阵运算描述 $k \rightarrow k + 1$ 鸡兔互变过程。

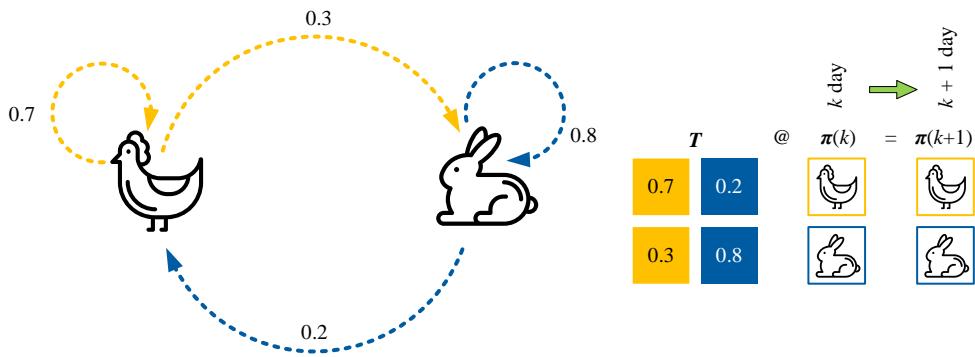


图 3. 用矩阵运算描述鸡兔互变

农夫注意到 T 矩阵的每一列概率值相加为 1。也就是说，这个 2×2 的方阵 T 还可以写成：

$$T = \begin{bmatrix} p & q \\ 1-p & 1-q \end{bmatrix} \quad (4)$$

其中， $p = 0.7$, $q = 0.2$ 。

代入具体数值

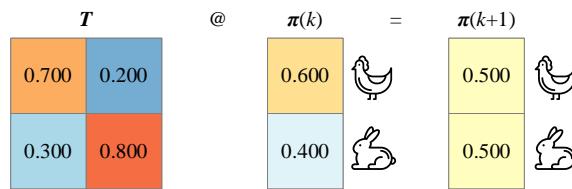
农夫假设，第 k 天鸡兔的比例为 60% 和 40%， $\pi(k)$ 为：

$$\pi(k) = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} \quad (5)$$

第 $k+1$ 天，鸡兔比例为：

$$k \rightarrow k+1: \quad T\pi(k) = \underbrace{\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}}_T \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \pi(k+1) \quad (6)$$

农夫想到这一计算可以用热图表达，于是他画了图 4。

图 4. 第 k 天 \rightarrow 第 $k+1$ 天，状态转换运算热图

而第 $k+2$ 天状态向量 $\pi(k+2)$ 和第 $k+1$ 天状态向量 $\pi(k+1)$ 关系为：

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。
版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

$$k+1 \rightarrow k+2: \quad T\pi(k+1) = \pi(k+2) \quad (7)$$

联立 (6) 和 (7)，得到第 $k+2$ 天状态向量 $\pi(k+2)$ 和第 k 天状态向量 $\pi(k)$ 关系：

$$k \rightarrow k+2: \quad T^2\pi(k) = \pi(k+2) \quad (8)$$

图 5 所示为，第 k 天 → 第 $k+2$ 天，状态转换运算热图。

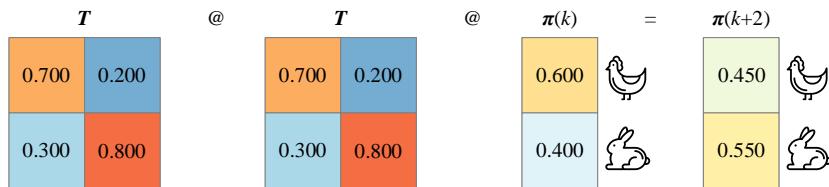


图 5. 第 k 天 → 第 $k+2$ 天，状态转换运算热图

另一种形式

农夫在查找参考书时发现，也有很多典籍用行向量表达状态向量，即对等式 (2) 左右转置：

$$\pi(k)^T T^T = \pi(k+1)^T \quad (9)$$

这样，(6) 可以写成：

$$\pi(k+1)^T = [0.6 \quad 0.4] \begin{bmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{bmatrix} = [0.5 \quad 0.5] \quad (10)$$

这种情况，转移矩阵的每一行概率值相加为 1。对应的矩阵运算热图为图 6。

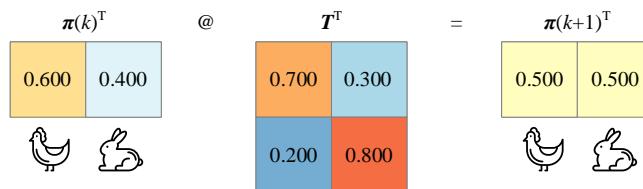
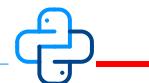


图 6. 第 k 天 → 第 $k+1$ 天，状态转换运算热图，注意状态向量为行向量



Bk3_Ch25_1.py 计算状态向量转化，并绘制图 4 和图 5 两幅热图。

25.2 第一视角：“鸡/兔→鸡” 和 “鸡/兔→兔”

农夫想到自己学习矩阵乘法时，书上讲过矩阵乘法有两个主要视角。他想先用矩阵乘法第一视角来分析(2)矩阵运算式。

他把 T 写成两个行向量 $t^{(1)}$ 和 $t^{(2)}$ 上下叠加，代入(2)得到：

$$\begin{bmatrix} t^{(1)} \\ t^{(2)} \end{bmatrix} \pi(k) = \begin{bmatrix} t^{(1)} \pi(k) \\ t^{(2)} \pi(k) \end{bmatrix} = \underbrace{\begin{bmatrix} \pi_1(k+1) \\ \pi_2(k+1) \end{bmatrix}}_{\pi(k+1)} \quad (11)$$

鸡/兔→鸡

农夫发现只看(11)第一行运算的话，它代表的转化是“鸡/兔 → 鸡”，如图7所示：

$$\begin{bmatrix} t^{(1)} \end{bmatrix} \pi(k) = \begin{bmatrix} t^{(1)} \pi(k) \end{bmatrix} = \begin{bmatrix} \pi_1(k+1) \end{bmatrix} \quad (12)$$

也就是说，上式代表第 k 天的鸡、兔，在第 $k+1$ 天变为鸡。

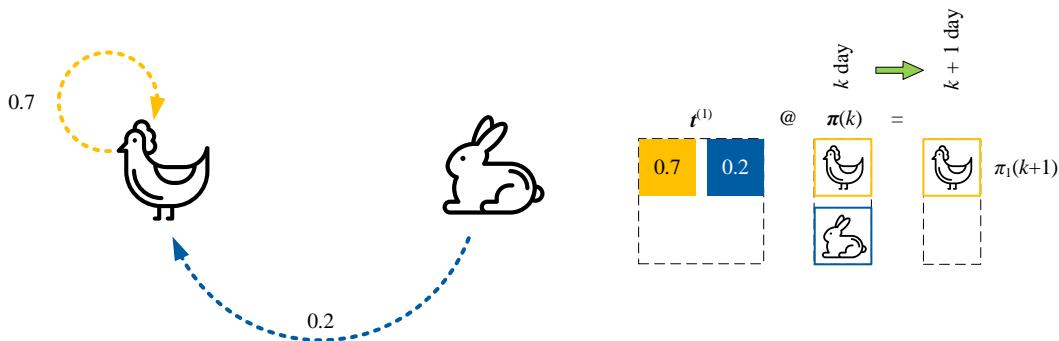
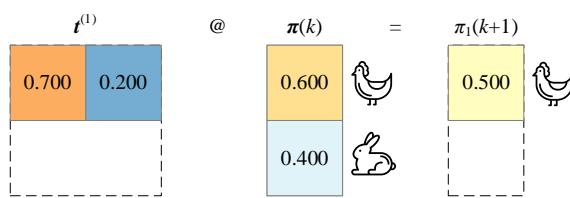


图 7. 鸡/兔→鸡

代入具体值，得到：

$$\begin{bmatrix} 0.7 & 0.2 \end{bmatrix} @ \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.5 \end{bmatrix} \quad (13)$$

第 k 天的鸡兔的比例分别为 60% 和 40%，到了 $k+1$ 天，鸡的比例为 50%。图 8 所示为上述运算热图。

图 8. 第 k 天 → 第 $k+1$ 天，鸡/兔→鸡

鸡/兔→兔

图 9 所示为 (11) 第二行运算，它代表“鸡/兔 → 兔”。也就是说，第 k 天的鸡、兔，第 $k+1$ 天变为兔：

$$\begin{bmatrix} t^{(2)} \end{bmatrix} \pi(k) = \begin{bmatrix} t^{(2)} \pi(k) \end{bmatrix} = \begin{bmatrix} \pi_2(k+1) \end{bmatrix} \quad (14)$$

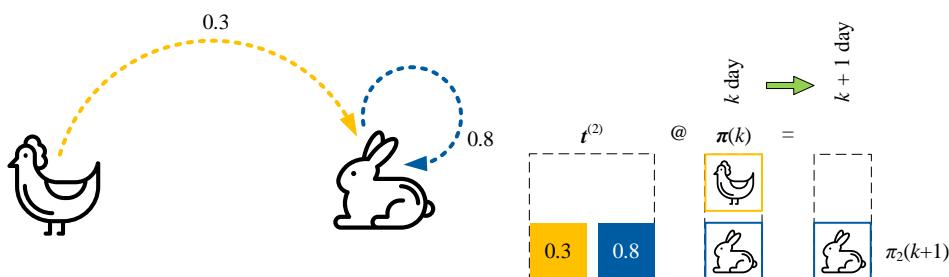
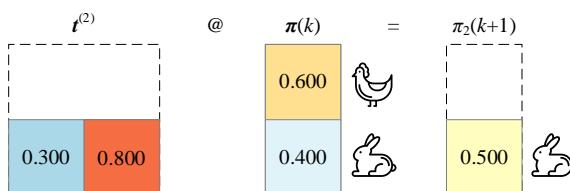


图 9. 鸡/兔→兔

图 10 所示为第 k 天的鸡兔的比例分别为 60% 和 40%，到了 $k+1$ 天，兔的比例也为 50%：

$$\begin{bmatrix} 0.3 & 0.8 \end{bmatrix} @ \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.5 \end{bmatrix} \quad (15)$$

图 10. 第 k 天 → 第 $k+1$ 天，鸡/兔→鸡

这就是利用矩阵乘法第一视角来分析状态转化运算。

25.3 第二视角：“鸡→鸡/兔” 和 “兔→鸡/兔”

农夫继续用矩阵乘法第二视角分析 (2) 矩阵运算式。

他将转移矩阵 T 写成左右排列列向量 t_1 和 t_2 ，代入 (2) 展开得到：

$$\underbrace{[t_1 \quad t_2]}_{T} \underbrace{\begin{bmatrix} \pi_1(k) \\ \pi_2(k) \end{bmatrix}}_{\pi(k)} = \pi_1(k)t_1 + \pi_2(k)t_2 = \begin{bmatrix} \pi_1(k+1) \\ \pi_2(k+1) \end{bmatrix} \quad (16)$$

其中， π_1 代表鸡的比例， π_2 代表兔的比例。

矩阵乘法第二视角将矩阵乘法 $T\pi(k) = \pi(k+1)$ 转化为矩阵加法 $\pi_1(k)t_1 + \pi_2(k)t_2$ 。农夫考虑分别分析 $\pi_1(k)t_1$ 和 $\pi_2(k)t_2$ 代表的具体含义。

(16) 这个式子让农夫看着头大，他决定代入具体鸡兔数值。

鸡→鸡/兔

假设第 k 天，鸡兔的比例仍为 60%、40%：

$$\pi(k) = \begin{bmatrix} \pi_1(k) \\ \pi_2(k) \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} \quad (17)$$

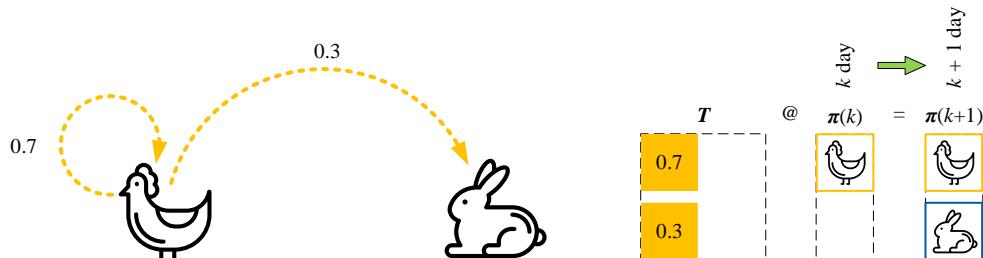
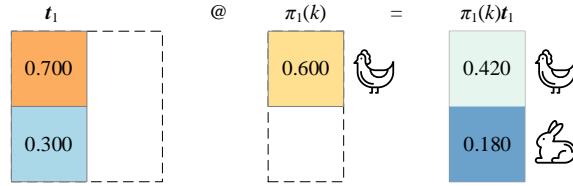


图 11. 鸡→鸡/兔

如图 11 所示， $\pi_1(k)t_1$ 代表“鸡 → 鸡/兔”。第 k 天，鸡的比例为 0.6，这些鸡在第 $k+1$ 天变成占总体比例 0.42 的鸡和 0.18 的兔：

$$\pi_1(k)t_1 = 0.6 \times \begin{bmatrix} 0.7 \\ 0.3 \end{bmatrix} = \begin{bmatrix} 0.42 \\ 0.18 \end{bmatrix} \quad (18)$$

图 12 所示为 (18) 运算热图。

图 12. 第 k 天 → 第 $k + 1$ 天，鸡→鸡/兔

兔→鸡/兔

如图 13 所示， $\pi_2(k)t_2$ 代表“兔 → 鸡/兔”。第 k 天，兔的比例为 0.4，这些兔在第 $k + 1$ 天变成占总体比例 0.08 的鸡和 0.32 的兔：

$$\pi_2(k)t_2 = 0.4 \times \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix} = \begin{bmatrix} 0.08 \\ 0.32 \end{bmatrix} \quad (19)$$

图 14 热图对应上述运算。

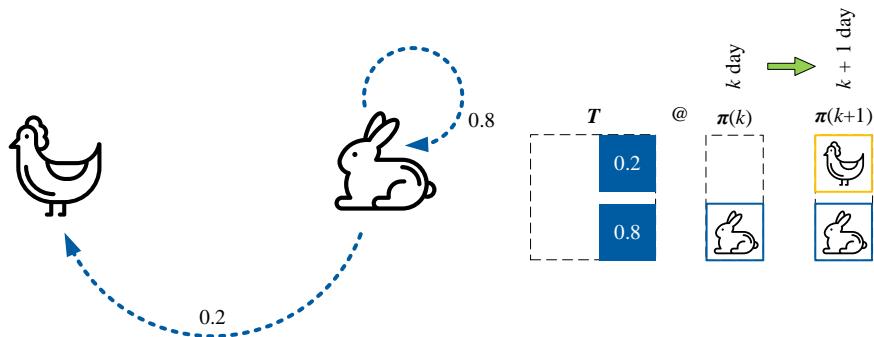
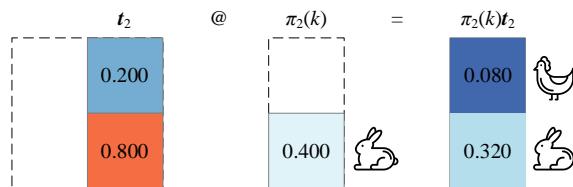
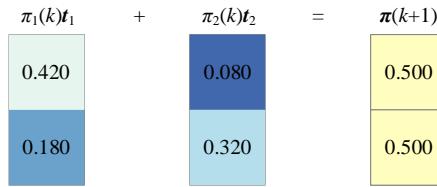


图 13. 兔→鸡/兔

图 14. 第 k 天 → 第 $k + 1$ 天，兔→鸡/兔

如图 15 热图所示，将 (18) 和 (19) 相加，得到第 $k + 1$ 天状态向量 $\pi(i + 1)$ ：

$$\pi(k+1) = \begin{bmatrix} 0.42 \\ 0.18 \end{bmatrix} + \begin{bmatrix} 0.08 \\ 0.32 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad (20)$$

图 15. 第 k 天 → 第 $k+1$ 天, 鸡/兔→鸡/兔

这就是利用矩阵乘法第二视角来分析状态转化运算。

25.4 连续几夜鸡兔转换

农夫把自己所学所想和村民分享后，大家都觉得线性代数有趣，认为农夫的分析有道理。大家纷纷加入农夫成立的“线代探秘小组”，学线代、用线代，并继续探究鸡兔互变这个疑难杂症。

有“线代探秘小组”成员发现，虽然连日来各家鸡兔互变没有停止，但是全村的鸡兔比例似乎达到了某种平衡。真是丈二和尚摸不着头脑！

农夫想用线性代数方法来看看连续几晚鸡兔互变有何有趣特征。

第 0 天，为初始状态，记做 $\pi(0)$ 。

第 1 天，状态向量 $\pi(1)$ 为：

$$0 \rightarrow 1: \quad T\pi(0) = \pi(1) \quad (21)$$

第 2 天，状态向量 $\pi(2)$ 和 $\pi(0)$ 关系为：

$$0 \rightarrow 2: \quad T\pi(1) = T^2\pi(0) = \pi(2) \quad (22)$$

第 3 天，状态向量 $\pi(3)$ 和 $\pi(0)$ 关系为：

$$0 \rightarrow 3: \quad T\pi(2) = T^3\pi(0) = \pi(3) \quad (23)$$

这样 $0 \rightarrow k$ 变化过程可以写成：

$$0 \rightarrow k: \quad T^k\pi(0) = \pi(k) \quad (24)$$

12 夜

农夫想算算连续 12 夜，在不同鸡兔初始比例状态 $\pi(0)$ 条件下，鸡兔达到平衡时比例特点。

图 16 所示的五种情况为鸡的初始比例更高，经过连续 12 夜的变化，农夫发现鸡兔的比例都达到了 40%、60%，也就是 4:6。

这个结果让农夫和“线代探秘小组”组员都眼前一亮！

而图 17 对应的一种情况是，鸡兔的初始比例相同，都是 50%；12 夜之后，鸡兔比例还是 40%、60%。

图 18 所示的五种情况是，初始状态 $\pi(0)$ 时，兔的比例更高。有趣的是，12 夜之后，鸡兔比例最终还是达到 40%、60%。

农夫觉得可以初步得出结论，在给定的转移矩阵 T 前提下，不管鸡兔初始比例 $\pi(0)$ 如何，结果都达到了一定的平衡，也就是：

$$T\pi = \pi \quad (25)$$

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
1.000	0.700	0.550	0.475	0.437	0.419	0.409	0.405	0.402	0.401	0.401	0.400	0.400
0.000	0.300	0.450	0.525	0.563	0.581	0.591	0.595	0.598	0.599	0.599	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.900	0.650	0.525	0.462	0.431	0.416	0.408	0.404	0.402	0.401	0.400	0.400	0.400
0.100	0.350	0.475	0.538	0.569	0.584	0.592	0.596	0.598	0.599	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.800	0.600	0.500	0.450	0.425	0.412	0.406	0.403	0.402	0.401	0.400	0.400	0.400
0.200	0.400	0.500	0.550	0.575	0.588	0.594	0.597	0.598	0.599	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.700	0.550	0.475	0.438	0.419	0.409	0.405	0.402	0.401	0.401	0.400	0.400	0.400
0.300	0.450	0.525	0.562	0.581	0.591	0.595	0.598	0.599	0.600	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.600	0.500	0.450	0.425	0.412	0.406	0.403	0.402	0.401	0.400	0.400	0.400	0.400
0.400	0.500	0.550	0.575	0.588	0.594	0.597	0.598	0.599	0.600	0.600	0.600	0.600

图 16. 连续 12 夜鸡兔互变比例，鸡的初始比例更高

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.500	0.450	0.425	0.412	0.406	0.403	0.402	0.401	0.400	0.400	0.400	0.400	0.400
0.500	0.550	0.575	0.588	0.594	0.597	0.598	0.599	0.600	0.600	0.600	0.600	0.600

图 17. 连续 12 夜鸡兔互变比例，鸡和兔的初始比例一样高

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400
0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.300	0.350	0.375	0.387	0.394	0.397	0.398	0.399	0.400	0.400	0.400	0.400	0.400
0.700	0.650	0.625	0.613	0.606	0.603	0.602	0.601	0.600	0.600	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.200	0.300	0.350	0.375	0.387	0.394	0.397	0.398	0.399	0.400	0.400	0.400	0.400
0.800	0.700	0.650	0.625	0.613	0.606	0.603	0.602	0.601	0.600	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.100	0.250	0.325	0.362	0.381	0.391	0.395	0.398	0.399	0.399	0.400	0.400	0.400
0.900	0.750	0.675	0.638	0.619	0.609	0.605	0.602	0.601	0.601	0.600	0.600	0.600

$\pi(0)$	$\pi(1)$	$\pi(2)$	$\pi(3)$	$\pi(4)$	$\pi(5)$	$\pi(6)$	$\pi(7)$	$\pi(8)$	$\pi(9)$	$\pi(10)$	$\pi(11)$	$\pi(12)$
0.000	0.200	0.300	0.350	0.375	0.388	0.394	0.397	0.398	0.399	0.400	0.400	0.400
1.000	0.800	0.700	0.650	0.625	0.613	0.606	0.602	0.603	0.602	0.601	0.600	0.600

图 18. 连续 12 夜鸡兔互变比例，兔的初始比例更高

求解平衡状态

农夫把 (25) 代入 (4)，得到：

$$\begin{bmatrix} p & q \\ 1-p & 1-q \end{bmatrix} \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} \quad (26)$$

另外，状态向量本身元素相加为 1，由此农夫得到两个等式：

$$\begin{cases} p\pi_1 + q\pi_2 = \pi_1 \\ \pi_1 + \pi_2 = 1 \end{cases} \quad (27)$$

求解二元一次线性方程组得到：

$$\begin{cases} \pi_1 = \frac{q}{1-p+q} \\ \pi_2 = \frac{1-p}{1-p+q} \end{cases} \quad (28)$$

农夫记得他假设 $p = 0.7$, $q = 0.2$, 代入 (28) 得到：

$$\begin{cases} \pi_1 = 0.4 \\ \pi_2 = 0.6 \end{cases} \quad (29)$$

也就鸡兔互变平衡时，稳态向量 π 为：

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 \\ \pi_2 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} \quad (30)$$

这和农夫之前做的模拟实验结果完全一致！真可谓“山重水复疑无路，柳暗花明又一村。”

也就是说， T 乘上 (30) 中的稳态向量 π ，结果还是稳态向量 π ：

$$T\boldsymbol{\pi} = \boldsymbol{\pi} \Rightarrow \underbrace{\begin{bmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{bmatrix}}_T \underbrace{\begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}}_{\boldsymbol{\pi}} = \underbrace{\begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix}}_{\boldsymbol{\pi}} \quad (31)$$

农夫突然记起这就是前几日他读到的**特征值分解** (eigen decomposition)! 书上反复提到特征值分解的重要性，农夫今天也见识到这个数学利器的伟力。



Bk3_Ch25_2.py 绘制本节 11 幅热图。



在 Bk3_Ch25_2.py 基础上，我们做了一个 App 用热图展示不同的初始状态到稳态向量的演变过程。请参考 Streamlit_Bk3_Ch25_2.py。

25.5 有向量的地方，就有几何

农夫学习线性代数时，总结了几句真经。其中一句就是——有向量的地方，就有几何。

他决定透过几何这个视角来看看状态向量的变化。

农夫把图 16、图 17、图 18 对应的 11 种状态向量的初始值画在平面直角坐标系中，用“有方向的线段”代表具体向量数值。

在他画的图 19 这 11 幅子图中，紫色向量代表鸡兔初始比例状态 $\pi(0)$ ，红色向量代表经过 12 夜鸡兔互变后 $\pi(12)$ 的位置。

农夫发现不管初始比例状态 $\pi(0)$ 如何，也就是紫色向量位于任何方位，经过 12 夜持续变化，红色向量 $\pi(12)$ 的位置几乎完全一致。

特别地，如图 19 (g) 所示，当初始比例 $\pi(0)$ 就是稳态向量时：

$$\pi(0) = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} \quad (32)$$

转移矩阵 T 没有改变 $\pi(0)$ 的方向。农夫查阅典籍发现，这个向量也有自己的名字，它叫做 T 的**特征向量** (eigenvector)。

而且，他发现变化过程，向量终点都落在一条直线上。这条直线代表——鸡、兔比例之和为 1。

农夫在图 19 中还画了另外一组向量，这些向量都是**单位向量** (unit vector)，对应：

$$\frac{\pi}{\|\pi\|} \quad (33)$$

这一组向量终点都落在单位圆上，因为它们的模都是 1。

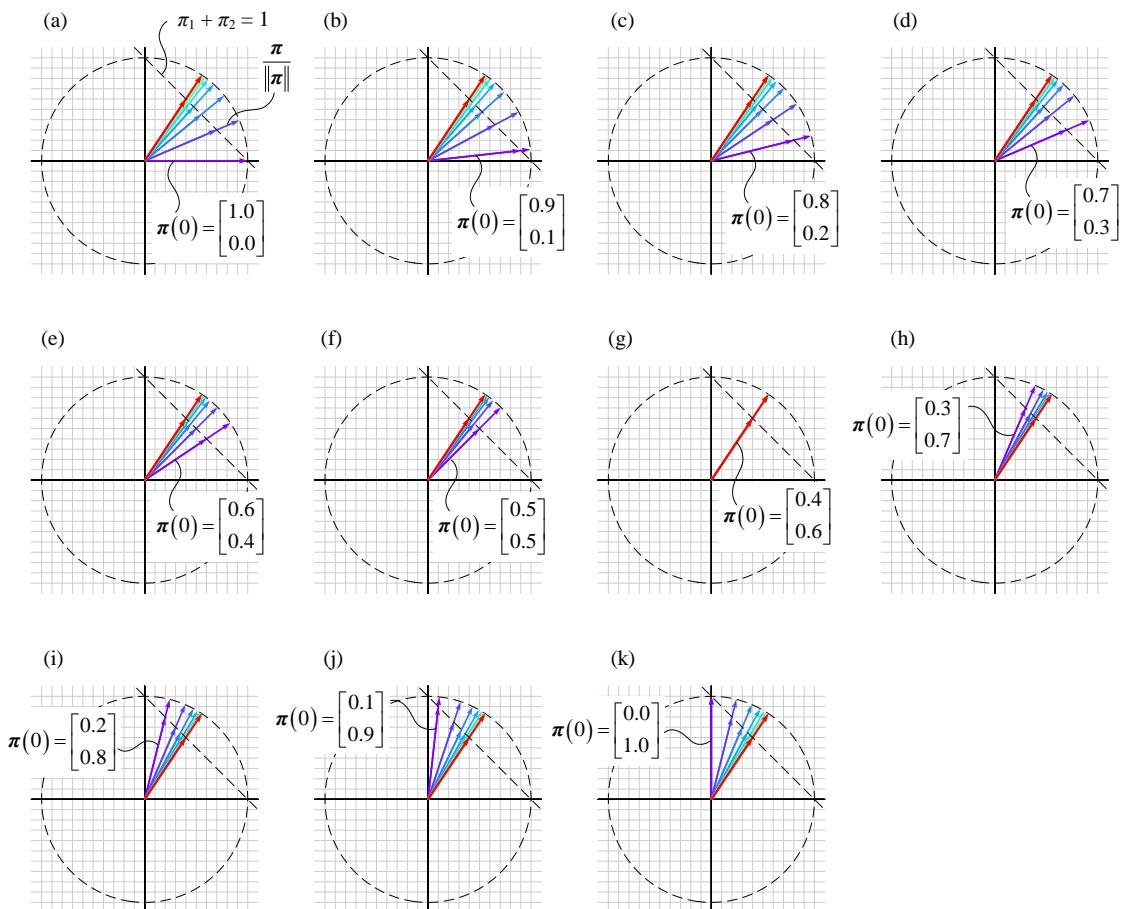
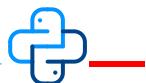


图 19. 连续 12 夜鸡兔互变比例，几何视角



Bk3_Ch25_3.py 绘制图 19。



在 Bk3_Ch25_3.py 基础上，我们做了一个 App 用箭头图展示不同的初始状态到稳态向量的演变过程。请参考 Streamlit_Bk3_Ch25_3.py。

25.6 彩蛋

至此，小村村民心中一块大石头算是落地了。对于“鸡兔互变”这个奇事，大伙儿也都见怪不怪了！

前后脚的事儿，村民发现鸡兔互变也停了。笑容在大伙儿脸上绽开，农夫把全村老少都邀到自家菜园，要好好欢庆一番！

大伙儿都没闲着，摘果蔬，网肥鱼，蒸米饭，取美酒，摆桌椅，嘉宾纷沓，鼓瑟吹笙，烹羊宰牛且为乐，会须一饮三百杯 …

这阵仗吓坏了的一笼鸡兔，它们蜷缩一团，瑟瑟发抖。农夫见状，撸着一只毛绒兔耳朵说，“你们这次立了大功，留着过年吧！”

欢言酌春酒，摘我园中蔬。微雨从东来，好风与之俱。

变与不变

书到用时方恨少，腹有诗书气自华，农夫这次让大伙儿理解了这两句话的精髓。

经过这场线性代数风暴之后，小村村民白天田间耕作时都会怀揣一本数学典籍，一得片刻休息，大伙儿分秒必争、手不释卷。夜深人静时，焚膏继晷、挑灯夜读者甚多。学数学，用数学，成了小村新风尚。

大伙儿似乎也不再惧怕未知，因为“我们必须知道，我们终将知道。”

渐渐地，这个曾经与世隔绝的小村处处都在变化，村民们也都肉眼可见地变化。你让我说，小村和村民哪里发生了变化？我也说不上。反正，时时刻刻都在变化，感觉一切都在变得更好。

而不变的是，小村还是那个小村，村民还是咱们这五十几户村民。

云山青青，风泉泠泠。山色依旧可爱，泉声更是可听。

(镜头拉远拉高) 一川松竹任横斜，有人家，被云遮。



东风升，云雾腾。

紫气东来，祥云西至。

鸡兔同笼引发的思想风暴，似乎给这个沉睡数百年的村庄带了什么，也似乎带走了什么。

好像什么都没有发生，又好像要发生什么。

往时曾发生的，来日终将发生。