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**，是敢于站在前人的臂膀之上的**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

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

Introduction

0 绪论

图解 + 编程 + 实践 + 数学板块融合

0.1 本册在全套丛书的定位

欢迎大家来到“鸢尾花书”最后一本——《机器学习》！

《数据有道》和《机器学习》两册是丛书“实践”板块的两本书。“数学”板块三本书为“实践”板块两本，特别是《机器学习》打下了坚实的数学基础。因此，数学基础不强的读者，不建议跳过“数学”直接学习本册。《数据有道》关注回归、降维这两类算法，它们都是特征工程的利器。而《机器学习》则在分类、聚类算法着墨更多。

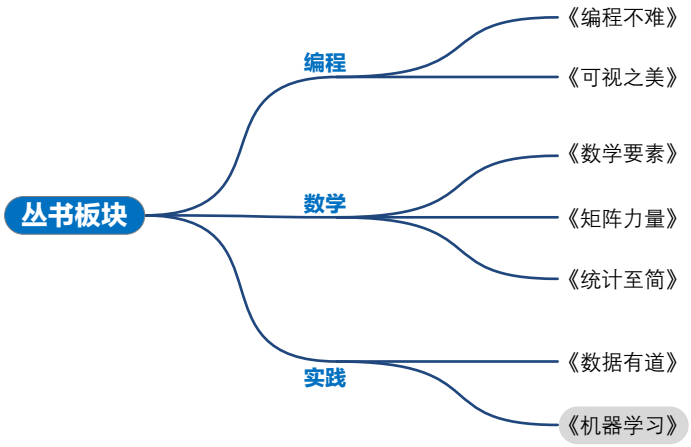


图 1. 本系列丛书板块布局

0.2 结构：2 大板块

根据机器学习有监督、无监督学习，本书主要分成两大板块。本书最后两章归为“其他”，但这并不代表这两章不重要。

图 2.《机器学习》板块布局

有监督学习

第 1 章首先给大家展示了 Scikit-learn 的机器学习算法模型地图，本书介绍的算法几乎都包含在这幅地图之中。

然后第 2 到 11 章主要介绍有关有监督学习内容。第 2 章首先介绍 k 近邻算法，这个算法基本思想是“小范围投票，少数服从多数”，它可以用来分类，也可以用来回归。

第 3 章总结鸢尾花书常见的距离度量。大家必须掌握不同距离度量的特点和应用场合。

第 4、5 两章介绍朴素贝叶斯分类。有关朴素贝叶斯分类算法，希望大家记住“假设特征之间条件独立，最大化后验概率”。这两章的区别在于概率密度估算方法，第 4 章利用高斯 KDE，第 5 章用多元高斯分布。

第 6 章介绍高斯判别分析，算法特点是“假设后验概率为高斯分布，最小化分类错误”。线性判别、二次判别都包含在高斯判别之中。

第 7、8 章介绍支持向量机。支持向量机的特点是间隔最大化，支持向量确定决策边界。第 8 章着重介绍核技巧，将样本数据映射到高维特征空间中，使数据在高维空间中线性可分。支持向量机既可以用来分类，也可以用来回归。

想要理解支持向量机绝对离不开《矩阵力量》中各种线性代数工具，特别是《矩阵力量》第 19 章内容、以及有关格拉姆矩阵的知识。

第 9 章讲解决策树，大家注意理解信息熵、信息增益等概念。

第 10 章介绍高斯过程，这种算法集合了高斯分布、条件概率、协方差矩阵、随机过程等数学工具，理解上不是很容易。高斯过程可以解决分类、回归两类问题。

第 11 章讲解回归，这一章也是综述，“鸟瞰”本系列丛书介绍的各种回归方法。

无监督学习

第 12 到 18 章为“无监督学习”板块。

第 12 章介绍 k 均值聚类，算法特点是簇内距离和最小、迭代求解。注意，k 均值聚类的 k 不同于 k-NN 中的 k。

第 13 章介绍高斯混合模型。高斯混合模型组合若干高斯分布，期望最大化。高斯混合模型求解离不开第 14 章讲解的最大期望算法。最大期望算法的特点是迭代优化两步走：E 步，M 步；最大化对数似然函数。

第 15 章介绍层次聚类。层次聚类基于数据之间距离，自下而上聚合，或自上而下分裂。

第 16 章介绍密度聚类 DBSCAN，算法特点是利用数据分布紧密程度聚类。Scikit-learn 中 OPTICS 算法类似 DBSCAN。

第 17 章讲解谱聚类。谱聚类通过构造无向图，降维聚类。本章略微介绍有关图论的内容，但是没有展开。

第 18 章是本系列丛书有关降维的综述。这一章回顾了奇异值分解、主成分分析、典型相关分析，还介绍了核主成分分析、独立成分分析、流形学习等算法。

0.3 特点：经典 + 综述

机器学习、深度学习算法不断涌现，让人目不暇接。限于作者知识水平、本书篇幅，本册在选取算法模型的标准只有一个——经典。从“经典”算法角度切入，《数据有道》、《机器学习》两册的目标是覆盖 Scikit-learn 库的常用函数。

本书还有一个特点就是提供“综述”，比如距离、回归、降维、优化这四章。请大家注意，这里的“综述”仅仅是对本系列丛书相关内容的总结和适度扩展。

本书还有一个特点是“理论 + 实践”。在学习本书时，希望大家不仅仅满足于会“调包”，也就是调用 Scikit-learn 各种函数，更要理解这些算法背后的数学理论。因此，本书给出适度的数学推导以及扩展阅读。

本书也有几个短板。其中之一是本书不涉及集成学习、神经网络、强化学习、深度学习、自然语言处理等话题。其次，本书也不涉及机器学习理论。虽然《数据有道》一册介绍过很多特征工程的工具，但是本书没有专门讲解特征工程章节。还有，本书也没有讨论如何部署机器学习模型。这些话题留给大家“按需”学习。

最后，欢迎大家来到“鸢尾花书”的收官之旅！

2 k 最近邻分类

k-nearest neighbors algorithm

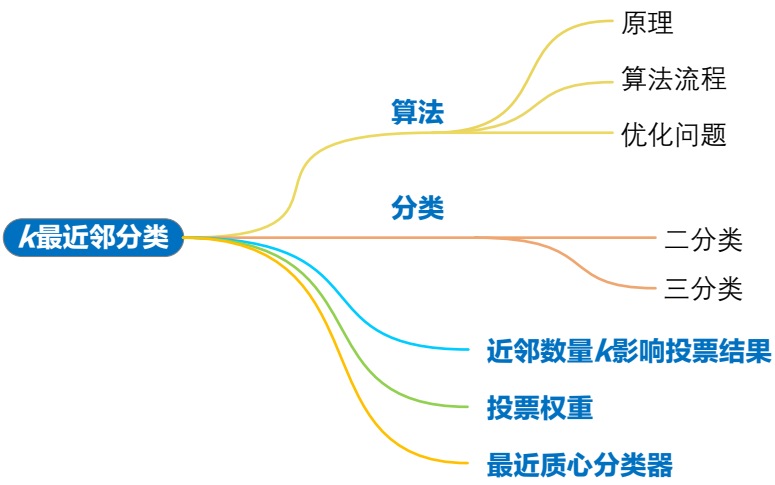
小范围投票，少数服从多数

如果一台计算机能够欺骗人类，让人类相信它也是人类一员；那么，这台计算机值得被称作智能机器。

A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.

—— 艾伦·图灵 (Alan Turing) | 英国计算机科学家、数学家，人工智能之父 | 1912 ~ 1954

- `enumerate()` 函数用于将一个可遍历的数据对象，比如列表、元组或字符串等，组合为一个索引序列，同时列出数据和数据下标，一般用在 `for` 循环当中
- `matplotlib.pyplot.contour()` 绘制等高线线图
- `matplotlib.pyplot.contourf()` 绘制填充等高线图
- `matplotlib.pyplot.scatter()` 绘制散点图
- `numpy.array()` 创建 `array` 数据类型
- `numpy.c_()` 按列叠加两个矩阵
- `numpy.r_()` 按行叠加两个矩阵
- `numpy.ravel()` 将矩阵扁平化
- `numpy.linspace()`产生连续均匀向量数值
- `numpy.meshgrid()` 创建网格化数据
- `seaborn.scatterplot()` 绘制散点图
- `sklearn.datasets.load_iris()` 加载鸢尾花数据集
- `sklearn.neighbors.KNeighborsClassifier` 为 k-NN 分类算法函数；函数常用的 `mehods` 为 `fit(X, y)` 和 `predit(q)`；`fit(X, y)`用来加载样本数据，`predit(q)`用来预测查询点 `q` 的分类
- `sklearn.neighbors.NearestCentroid` 最近质心分类算法函数



2.1 k 近邻分类原理：近朱者赤，近墨者黑

k 近邻算法 (*k*-nearest neighbors algorithm, *k*-NN) 是最基本监督学习方法之一。这种算法的优点是简单易懂，不需要训练过程，对于非线性分类问题表现良好。然而，它也存在一些缺点，例如需要大量存储训练集、预测速度较慢、对于高维数据容易出现维数灾难等。此外，在选择 k 值时需要进行一定的调参工作，以保证算法的准确性和泛化能力。

⚠ 注意，*k*-NN 中的 k 指的是“近邻”的数量。

原理

k-NN 思路很简单——“近朱者赤，近墨者黑”。更准确地说，小范围投票，少数服从多数 (majority rule)，如图 1。

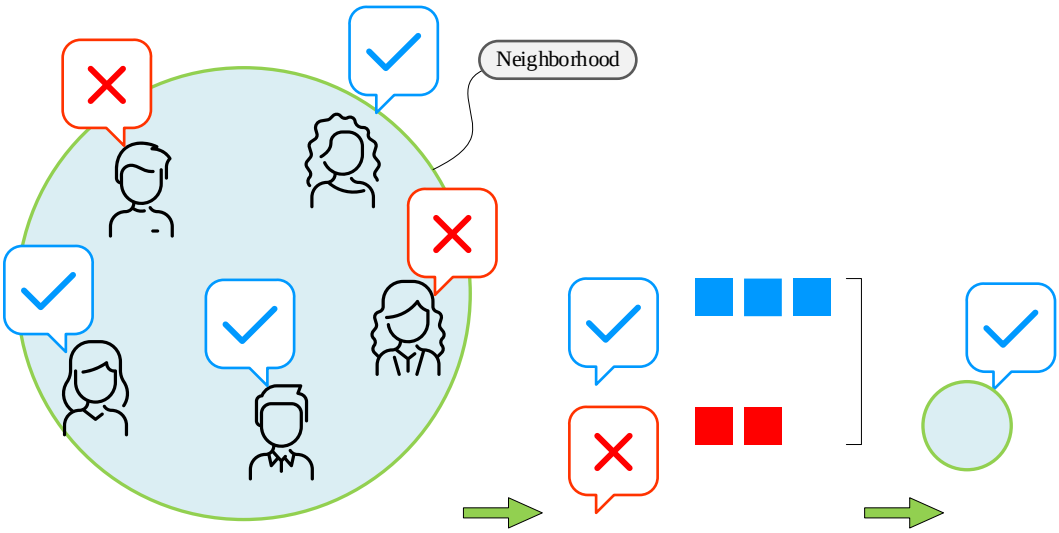


图 1. k 近邻分类核心思想——小范围投票，少数服从多数

算法流程

给定样本数据 $\boldsymbol{X}$ $(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)})$，分别对应已知标签 $\boldsymbol{y}$ $(y^{(1)}, y^{(2)}, \ldots, y^{(n)})$。**查询点** (query point) $\boldsymbol{q}$ 标签未知，待预测分类。

k-NN 近邻算法流程如下：

- 计算样本数据 $\boldsymbol{X}$ 任意一点 $\boldsymbol{x}$ 和查询点 $\boldsymbol{q}$ 距离；
- 找 $\boldsymbol{X}$ 中距离查询点 $\boldsymbol{q}$ 最近的 k 个样本，即 k 个“近邻”；

◀ 根据 k 个邻居已知标签，直接投票或加权投票；k 个邻居出现数量最多的标签即为查询点 $\boldsymbol{q}$ 预测分类结果。

优化问题

用公式表示，k-NN 算法的优化目标如下，**预测分类** (predicted classification) $\hat{y}$：

$$\hat{y}(\boldsymbol{q}) = \arg\max_{C_k} \sum_{i \in kNN(\boldsymbol{q})} I\left(y^{(i)} = C_k\right) \tag{1}$$

其中，kNN($\boldsymbol{q}$)为查询点 $\boldsymbol{q}$ 近邻构成的集合，C_k 为标签为 C_k 的样本数据集合，$k = 1, 2, \ldots, K$。I 为**指示函数** (indicator function)，表示“一人一票”；当 $y^{(i)} = C_k$ 成立时，$I = 1$；否则，$I = 0$。

下面以二分类为例，和大家讲解如何理解 k-NN 算法。

2.2 二分类：非红，即蓝

平面可视化

假设，数据 $\boldsymbol{X}$ 有两个特征，即 $D = 2$；$\boldsymbol{X}$ 两个特征分别为 x_1 和 x_2。也就是说，在 x_1x_2 平面上，$\boldsymbol{X}$ 的第一列数值为横坐标，$\boldsymbol{X}$ 的第二列数值为纵坐标。

$\boldsymbol{y}$ 有两类标签 $K = 2$，即 C_1 和 C_2；红色 ● 表示 C_1，蓝色 ● 表示 C_2。

$\boldsymbol{X}$ 和 $\boldsymbol{y}$ 数据形式及平面可视化如图 2 所示。

显然这是个**二分类** (binary classification, bi-class classification) 问题，查询点 $\boldsymbol{q}$ 的分类可能是 C_1 (红色 ●)，或者 C_2 (蓝色 ●)。

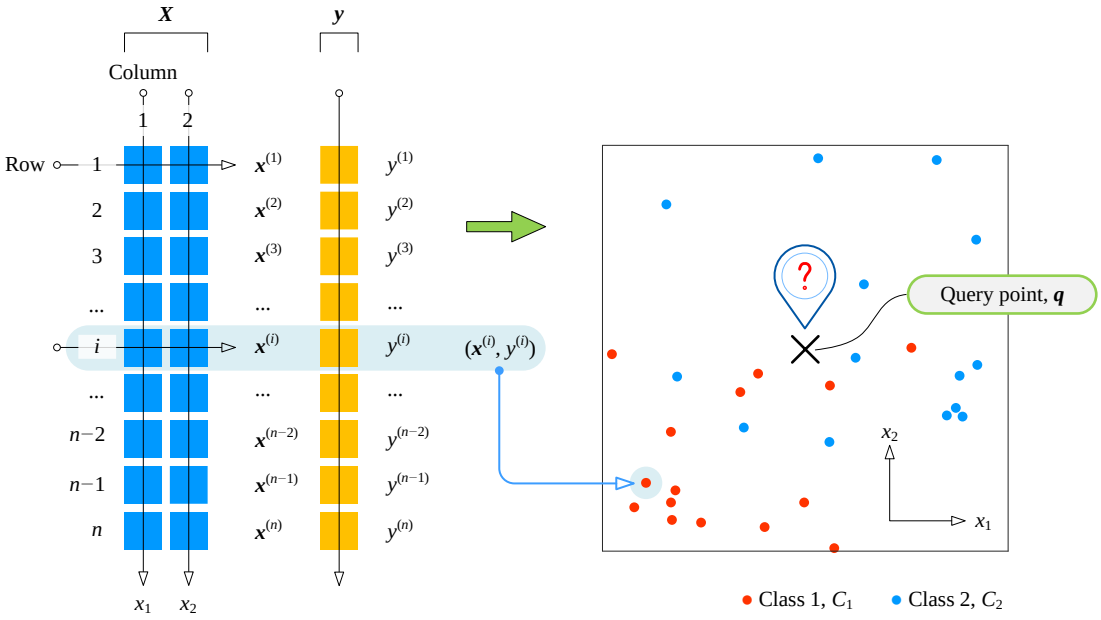


图 2. 两特征 ($D = 2$) 含标签样本数据可视化

四个近邻投票

对于二分类问题，即 $K = 2$，(1) 可以写成：

$$\hat{y}(\boldsymbol{q}) = \max_{C_1, C_2} \left\{ \sum_{i \in kNN(\boldsymbol{q})} I\left(y^{(i)} = C_1\right), \sum_{i \in kNN(\boldsymbol{q})} I\left(y^{(i)} = C_2\right) \right\} \tag{2}$$

在图 3 所示平面上，× 为查询点 $\boldsymbol{q}$，以行向量表达。

如果设定“近邻”数量 $k = 4$，以查询点 $\boldsymbol{q}$ 为圆心圈定的圆形“近邻社区”里有 4 个样本数据点 ($\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$、$\boldsymbol{x}^{(3)}$ 和 $\boldsymbol{x}^{(4)}$)。4 个点中，样本点 $\boldsymbol{x}^{(1)}$ 距离查询点 $\boldsymbol{q}$ 距离 d_1 最近，样本点 $\boldsymbol{x}^{(4)}$ 距离查询点 $\boldsymbol{q}$ 距离 d_4 最远。

显然，查询点 $\boldsymbol{q}$ 近邻社区中四个查询点中，投票为“三比一”——3 个“近邻”标签为 C_1 (红色 ●)，1 个“近邻”标签为 C_2 (蓝色 ●)。也就是：

$$\begin{aligned} \sum_{i \in kNN(\boldsymbol{q})} I\left(y^{(i)} = C_1\right) &= 3 \\ \sum_{i \in kNN(\boldsymbol{q})} I\left(y^{(i)} = C_2\right) &= 1 \end{aligned} \tag{3}$$

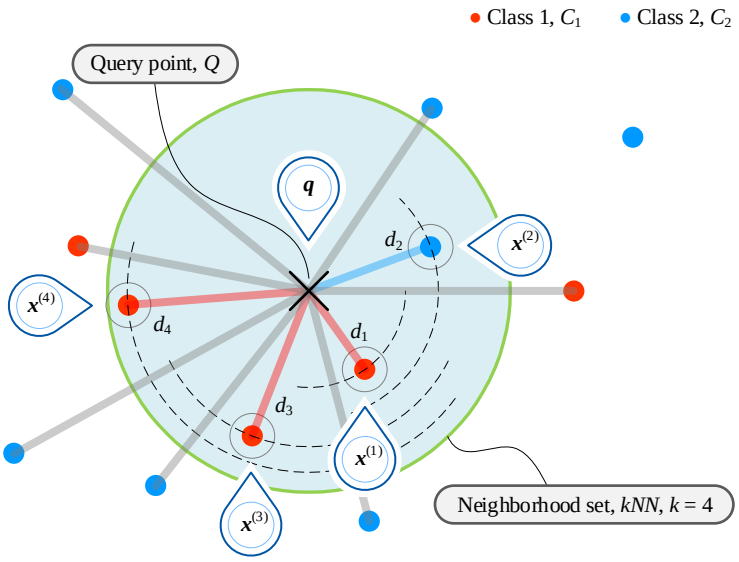


图 3. k 近邻原理

将具体分类标签带入 (2)，可以得到：

$$\hat{y}(\boldsymbol{q}) = \max_{C_{1,2}} \left\{ 3_{(C_1)}, 1_{(C_2)} \right\} = C_1 \tag{4}$$

由于近邻不分远近，投票权相同。图 3 中距离线段线宽代表投票权。少数服从多数，在 $k = 4$ 的条件下，红色 ● “胜出”！因此，查询点 $\boldsymbol{q}$ 的预测分类为 C_1 (红色 ●)。

需要引起注意的是，近邻数量 k 是自定义输入；观察图 3 可以发现，当 k 增大时，查询点 $\boldsymbol{q}$ 的预测分类可能会发生变化。下一节将会讨论近邻数量 k 如何影响分类预测结果。

使用函数

`sklearn.neighbors.KNeighborsClassifier` 为 Scikit-learn 工具包 k-NN 分类算法函数。函数默认的近邻数量 `n_neighbors` 为 5，默认距离度量 `metric` 为**欧氏距离** (Euclidean distance)。这个函数常用的 methods 为 `fit(X, y)` 和 `predit(q)`；`fit(X, y)` 用来拟合样本数据，`predit(q)` 用来预测查询点 $\boldsymbol{q}$ 的分类。

本书下一章将总结常见距离度量。

2.3 三分类：非红，要么蓝，要么灰

鸢尾花分类问题为三分类问题，即 $K = 3$。图 4 每个圆点 ● 代表一个数据点。其中，● 代表分类为 setosa (C_1, $y = 0$)，● 代表 versicolor (C_2, $y = 1$)，● 代表 virginica (C_3, $y = 2$)。

图 4 所示为利用 KNeighborsClassifier 获得的鸢尾花分类结果。输入数据选取鸢尾花数据 2 个特征——花萼长度 x_1，和花萼宽度 x_2。用户输入的近邻数量 n_neighbors 为 4。请大家注意，图 4 平面一些位置数据点存在叠合，也就是说一个圆点代表不止一个数据点。

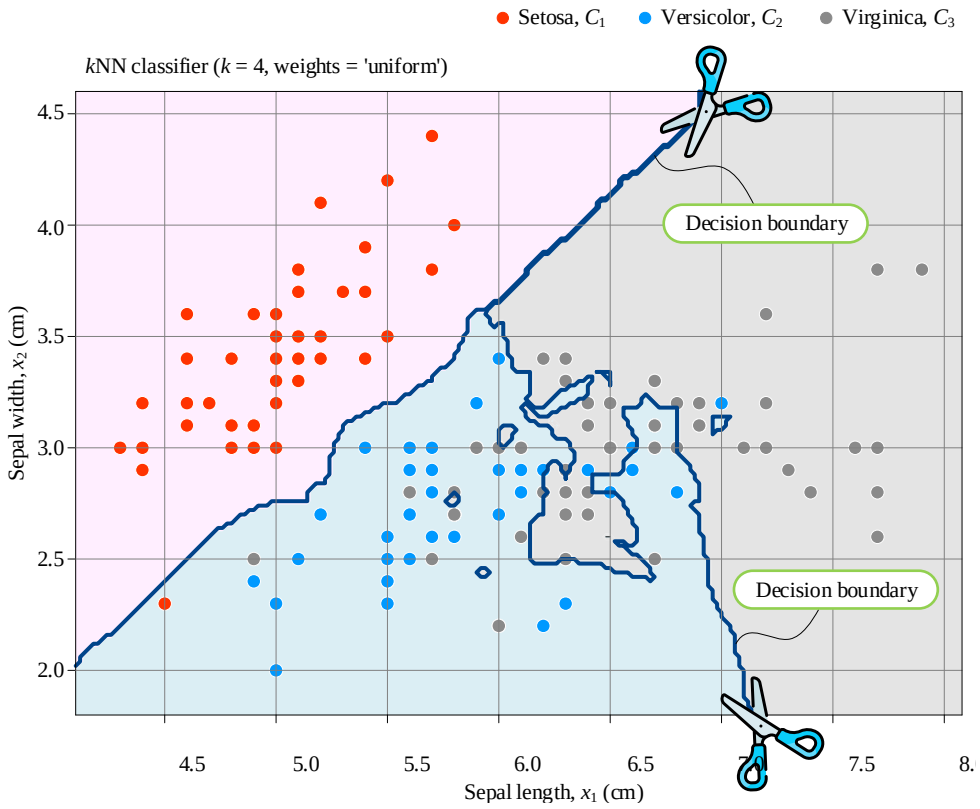


图 4. k 近邻分类，k = 4，采用 2 个特征 (花萼长度 x_1，和花萼宽度 x_2) 分类三种鸢尾花

⚠ 注意，欧几里德距离，也称欧氏距离，是最常见的距离度量，本章出现的距离均为欧氏距离。此外，本节利用直接投票 (等权重投票)，而本章第三节将讲解加权投票原理。

决策边界

图 4 中深蓝色曲线为**决策边界** (decision boundary)。如果决策边界是直线、平面或超平面，那么这个分类问题是线性的，分类是线性可分的；否则，分类问题非线性。图 4 所示 k-NN 算法决策边界杂乱无章，肯定是非线性，甚至不可能用某个函数来近似。

很多分类算法获得的决策边界都可以通过简单或者复杂函数来描述，比如一次函数、二次函数、二次曲线等等；这类模型也称**参数模型** (parametric model)。与之对应的是，类似 k-NN 这样的学习算法得到的决策边界为**非参数模型** (non-parametric model)。

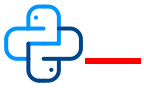
k-NN 基于训练数据，更准确地说是把训练数据以一定的形式存储起来完成学习任务，而不是泛化得到某个解析解进行数据分析或预测。

所谓**泛化能力** (generalization ability) 是指机器学习算法对全新样本的适应能力。适应能力越强，泛化能力越强；否则，泛化能力弱。

举个简单例子解释“泛化能力弱”这一现象；一个学生平时做了很多练习题，每道练习题目都烂熟于心；这个学生虽然刻苦练习，可惜他就题论题，不能举一反三，考试做新题时，分数总是很低。

每当遇到一个新查询点，k-NN 分类器分析这个新查询点与早前存储样本数据的关系，并据此把一个预测分类值赋给新查询点。值得注意的是，这些样本数据是以树形结构存储起来，常见的算法是 *kd* 树。

提醒大家注意，学习每一种学习算法时，注意观察决策边界形状特点，并总结规律。



代码 Bk7_Ch02_01.py 可以用来实现本节分类问题，并绘制图 4。

2.4 近邻数量 k 影响投票结果

近邻数量 k 为用户输入值，而 k 值直接影响查询点分类结果；因此，选取合适 k 值格外重要。本节和大家探讨近邻数量 k 对分类结果影响。

图 5 所示为 k 取四个不同值时，查询点 $\boldsymbol{q}$ 预测分类结果变化情况。如图 5 (a) 所示，当 $k = 4$ 时，查询点 $\boldsymbol{q}$ 近邻中，3 个近邻为 ● (C_1)，1 个近邻为 ● (C_2)；采用等权重投票，查询点 $\boldsymbol{q}$ 预测分类为● (C_1)。

当近邻数量 k 提高到 8 时，近邻社区中，4 个近邻为 ● (C_1)，4 个近邻为 ● (C_2)，如图 5 (b) 所示；等权重投票的话，两个标签各占 50%。因此 $k = 8$ 时，查询点 $\boldsymbol{q}$ 恰好在决策边界上。

如图 5 (c) 所示，当 $k = 12$ 时，查询点 $\boldsymbol{q}$ 近邻中 5 个为 ● (C_1)，7 个为 ● (C_2)；等权重投票条件下，查询点 $\boldsymbol{q}$ 预测标签为 ● (C_2)。当 $k = 16$ 时，如图 5 (c) 所示，查询点 $\boldsymbol{q}$ 预测标签同样为 ● (C_2)。

k-NN 算法选取较小的 k 值虽然能准确捕捉训练数据的分类模式；但是，缺点也很明显，容易受到噪声影响。

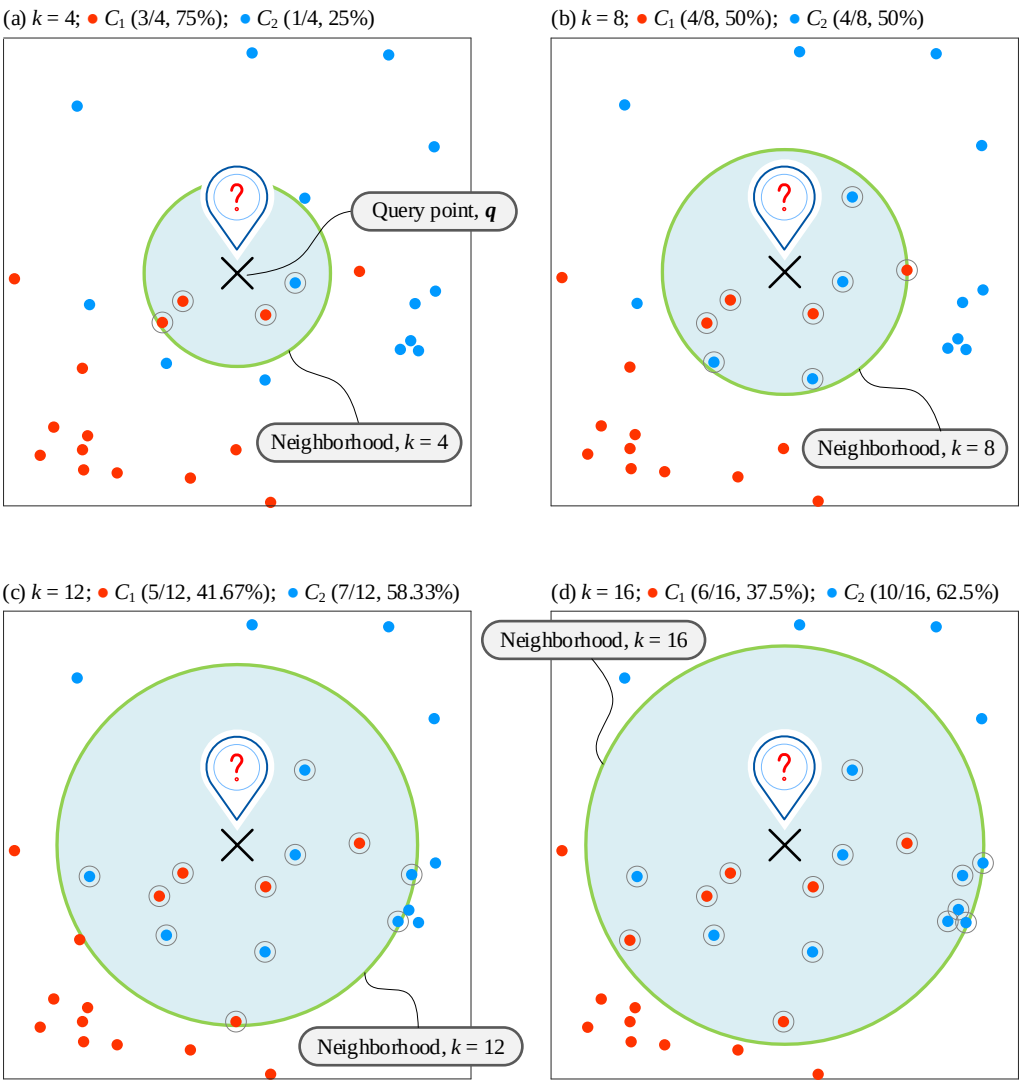


图 5. 近邻数量 k 值影响查询点的分类结果

影响决策边界形状

图 6 所示为 k 选取不同值时对鸢尾花分类影响。观察图 6 四副子图可以发现，当 k 逐步增大时，局部噪音样本对边界的影响逐渐减小，边界形状趋于平滑。

较大的 k 是会抑制噪声的影响，但是使得分类界限不明显。举个极端例子，如果选取 k 值为训练样本数量，即 $k = n$，采用等权重投票，这种情况不管查询点 $\boldsymbol{q}$ 在任何位置，预测结果仅有一个。这种训练得到的模型过于简化，忽略样本数据中有价值的信息。

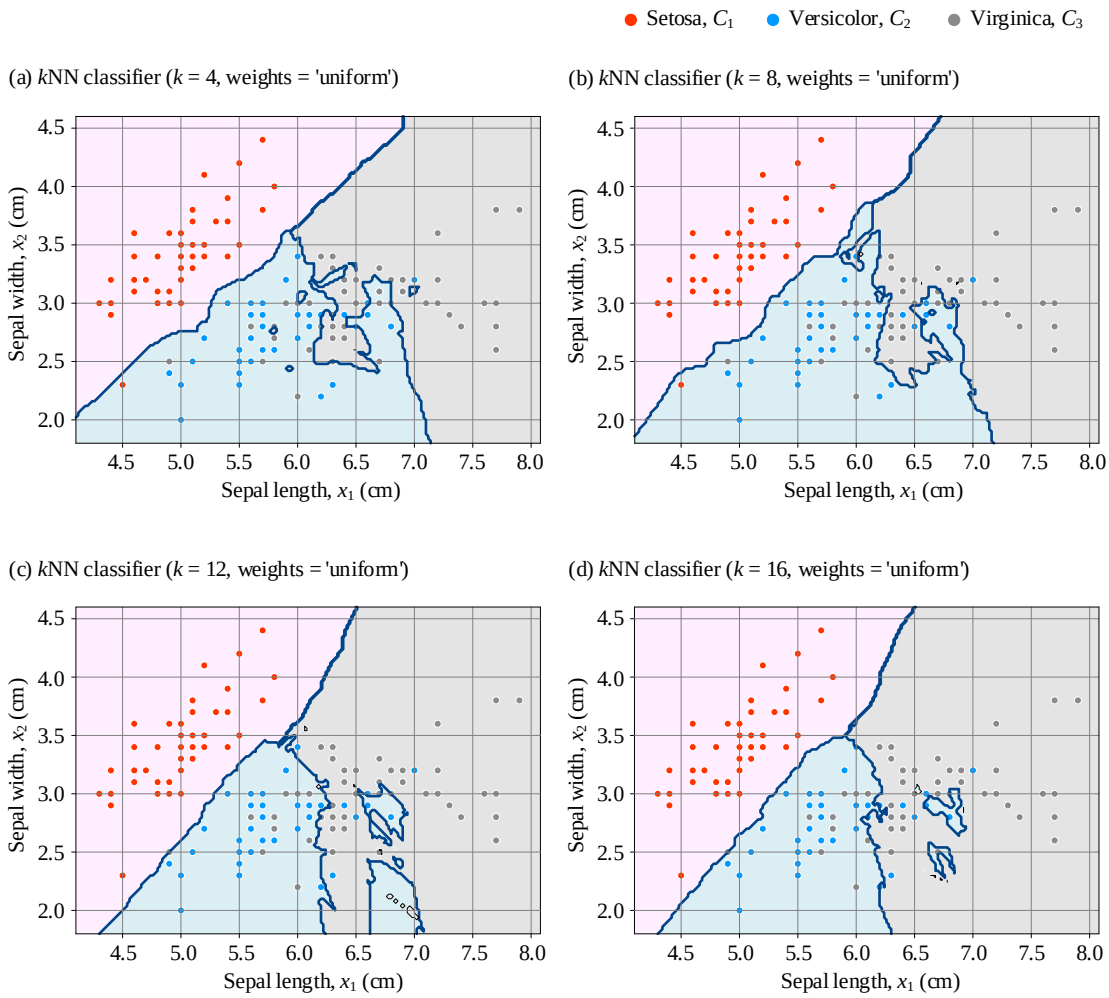


图 6. *k*-NN，*k* 选取不同值时对鸢尾花分类影响

图 7 所示为 $k = 50$ 时得到的决策边界。

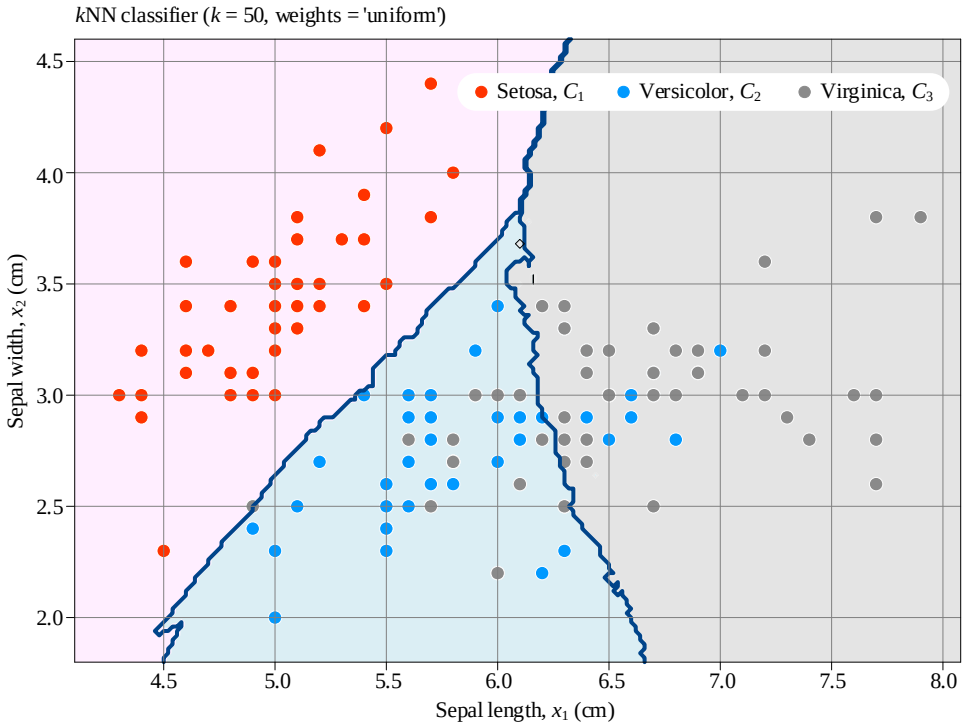


图 7. k = 50 时，鸢尾花分类决策边界，k-NN，等权重投票

代码 Bk7_Ch02_02.py 可以获得图 6 和图 7。这个代码利用 matplotlib 实现交互，大家可以比较它和 streamlit 哪个制作 App 更方便。

2.5 投票权重：越近，影响力越高

本章前文强调，在“近邻社区”投票时，采用的是“等权重”方式；也就是说，只要在“近邻社区”之内，无论距离远近，一人一票，少数服从多数。

前文 k 近邻分类函数，默认等权重投票，默认值 `weights = 'uniform'`。但是，很多 k 近邻分类问题采用加权投票则更有效。

如图 8 所示，每个近邻的距离线段线宽 w_i 代表各自投票权重。距离查询点越近的近邻，投票权重 w_i 越高；相反，越远的近邻，投票权重 w_i 越低。

对应的优化问题变成：

$$\hat{y}(\boldsymbol{q}) = \arg\max_{C_k} \sum_{i \in kNN(\boldsymbol{q})} w_i \cdot I\left(y^{(i)} = C_k\right) \tag{5}$$

`sklearn.neighbors.KNeighborsClassifier` 函数中，可以设定投票权重与查询点距离成反比，`weights = 'distance'`。

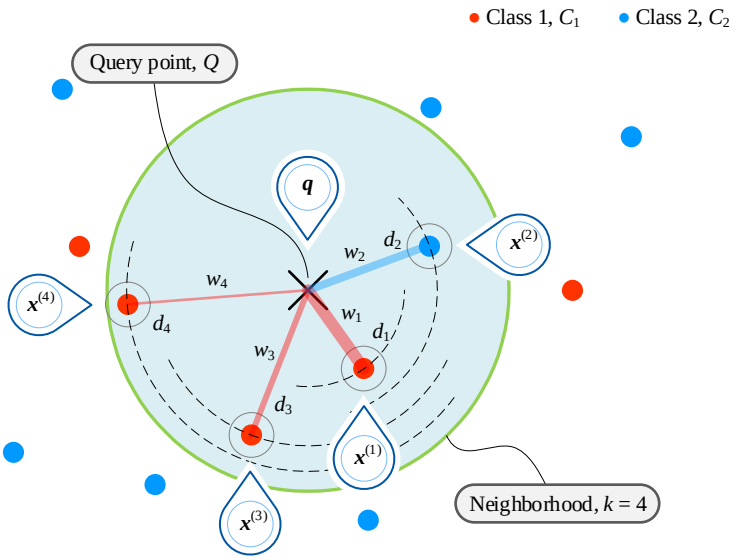


图 8. k 近邻原理，加权投票

此外，近邻投票权 w_i 还可以通过**归一化** (normalization) 处理，如下式：

$$w_i = \frac{\max\left(d_{\text{NN}}\right) - d_i}{\max\left(d_{\text{NN}}\right) - \min\left(d_{\text{NN}}\right)} \tag{6}$$

d_{NN} 为所有近邻距离构成的集合，$\max(d_{\text{NN}})$ 和 $\min(d_{\text{NN}})$ 分别计算得到近邻距离最大和最小值。加权投票权重还可以采用距离平方的倒数，这种权重随着距离增大衰减越快。使用 scikit-learn 的 kNN 分类器时，大家可以自定义加权投票权重函数。

决策边界

图 9 所示为，近邻数量为 k = 50 条件下，`weights = 'distance'` 时，k 近邻分类算法获得决策边界。

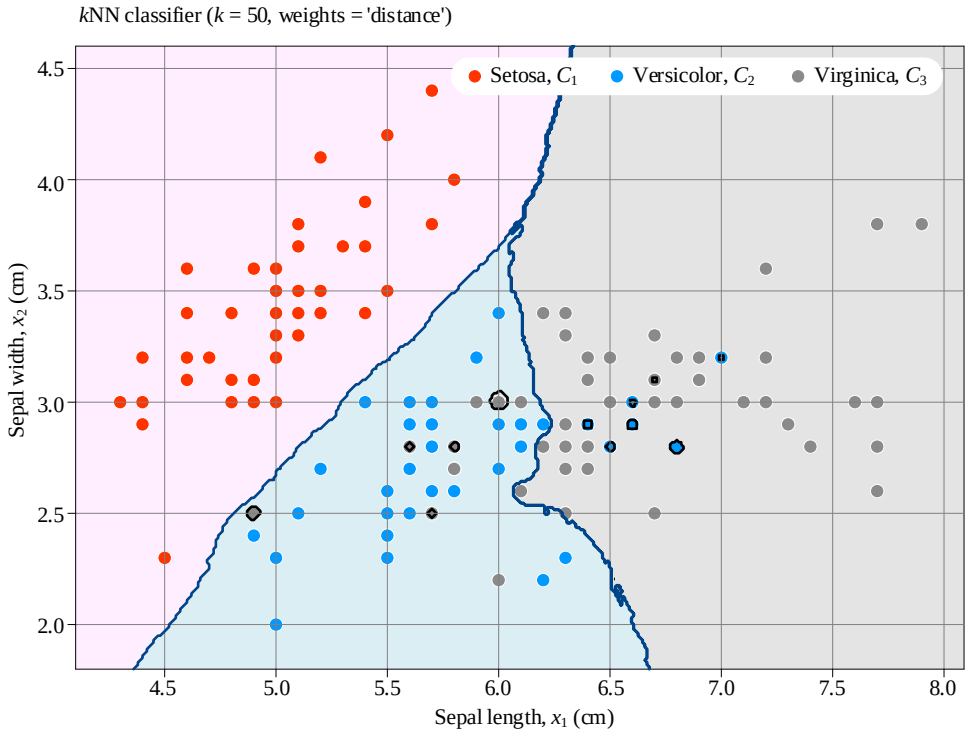


图 9. k = 50 时，鸢尾花分类决策边界，投票权重与查询点距离成反

2.6 最近质心分类：分类边界为中垂线

最近质心分类器 (Nearest Centroid Classifier, NCC) 思路类似 k-NN。

本章前文讲过，k-NN 以查询点为中心，圈定 k 个近邻，近邻投票。而最近质心分类器，先求解得到不同类别样本数据簇质心位置 $\boldsymbol{\mu}_m$ (m = 1, 2, …, K)；查询点 $\boldsymbol{q}$ 距离哪个分类质心越近，其预测分类则被划定为这一类。因此，最近质心分类器不需要设定最近邻数量 k。

《矩阵力量》第 22 章已经讨论过**数据质心** (centroid) 这个概念，它的具体定义如下：

$$\boldsymbol{\mu}_k = \frac{1}{\text{count}(C_k)} \sum_{i \in C_k} \boldsymbol{x}^{(i)} \tag{7}$$

其中，count() 计算某个标签为 C_k 的子集样本数据点的数量。

注意，上式假定 $\boldsymbol{x}^{(i)}$ 和 $\boldsymbol{\mu}_k$ 均为列向量。

分类函数

Python 工具包完成最近质心分类的函数为 `sklearn.neighbors.NearestCentroid`。图 10 所示为通过最近质心分类得到的鸢尾花分类决策边界。图 10 中 $\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$ 和 $\boldsymbol{\mu}_3$ 三点分别为 ● setosa (C_1, $y = 0$)、● versicolor (C_2, $y = 1$)和 ● virginica (C_3, $y = 2$) 的质心所在位置。

大家可能已经发现，图 10 中每段决策边界就是两个质心的中垂线！

《矩阵力量》第 19 章讲解过中垂线，请大家回顾。

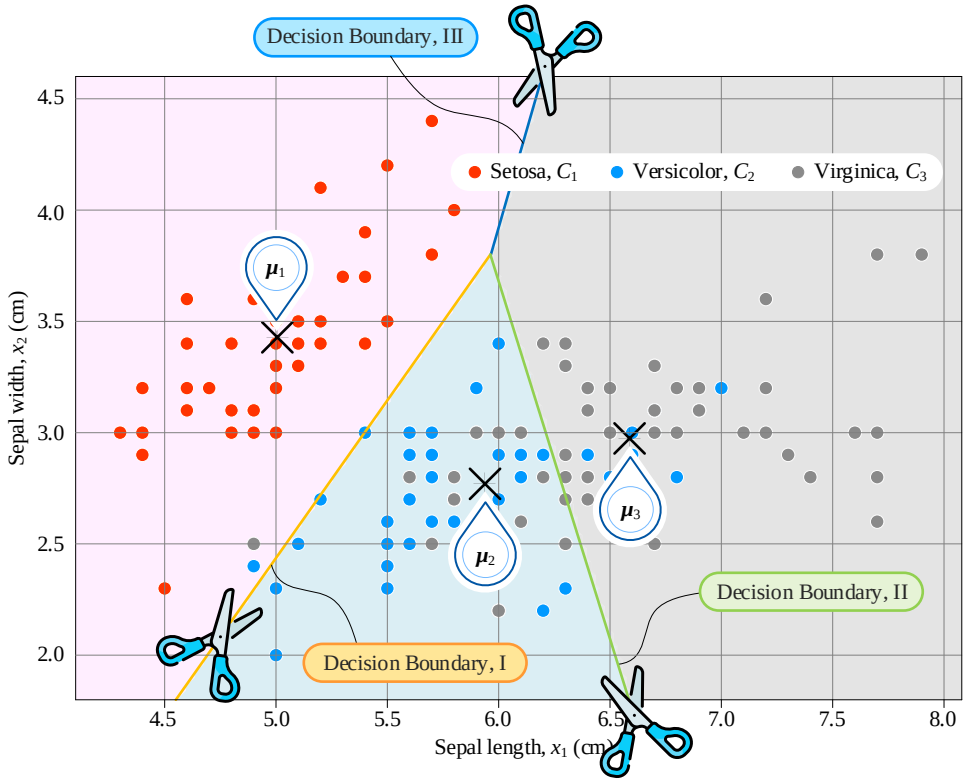


图 10. 鸢尾花分类决策边界，最近质心分类

图解原理

图 11 所示为最近质心分类器边界划分原理图。

平面上，A 和 B 两点中垂线上每一点距离 A 和 B 相等。中垂线垂直于 AB 线段，并经过 AB 线段中点。图 11 中决策边界无非就是，$\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$ 和 $\boldsymbol{\mu}_3$ 三个质心点任意两个构造中垂线。

如图 11 所示，为了确定查询点 $\boldsymbol{q}$ 的预测分类，计算 $\boldsymbol{q}$ 到 $\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$ 和 $\boldsymbol{\mu}_3$ 三个质心点距离度量。比较 AQ、BQ 和 CQ 三段距离长度，发现 CQ 最短，因此查询点 $\boldsymbol{q}$ 预测分类为 ● virginica (C_3)。

图 11 有专门的名字——**沃罗诺伊图** (Voronoi diagram)。本书将会在 K 均值聚类一章介绍。

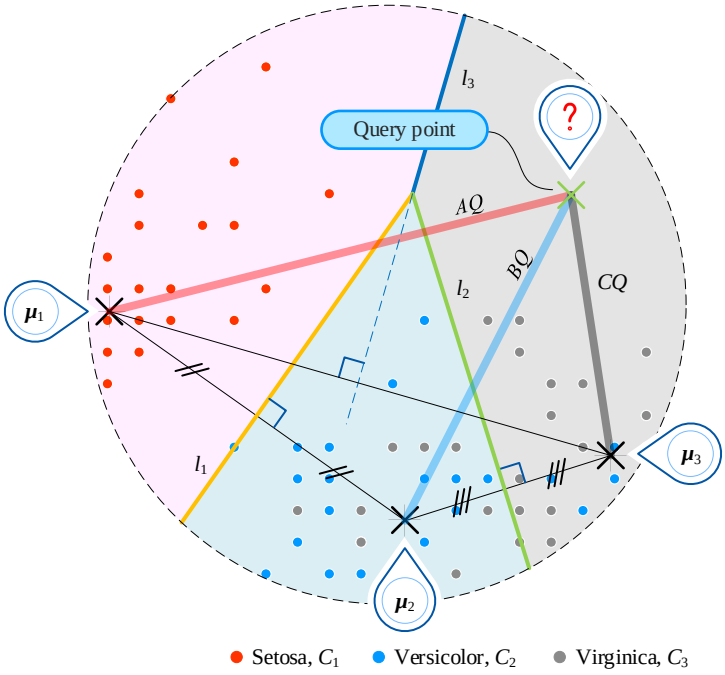


图 11. 最近质心分类决策边界原理

收缩阈值

`sklearn.neighbors.NearestCentroid` 函数还提供**收缩阈值** (shrink threshold)，获得**最近收缩质心** (nearest shrunken centroid)。说的通俗一点，根据收缩阈值大，每个类别数据质心向样本数据总体质心 $\boldsymbol{\mu}_X$ 靠拢。图 12 展示的是随着收缩阈值不断增大，分类数据质心不断向 $\boldsymbol{\mu}_X$ 靠拢，分类边界不断变化的过程。

`NearestCentroid` 函数定义收缩阈值如何工作。对此感兴趣的话，大家可以自行打开 `NearestCentroid` 函数，查找 `if self.shrink_threshold:` 对应的一段。

代码 Bk7_Ch02_03.py 绘制图 12 所示四幅图像。

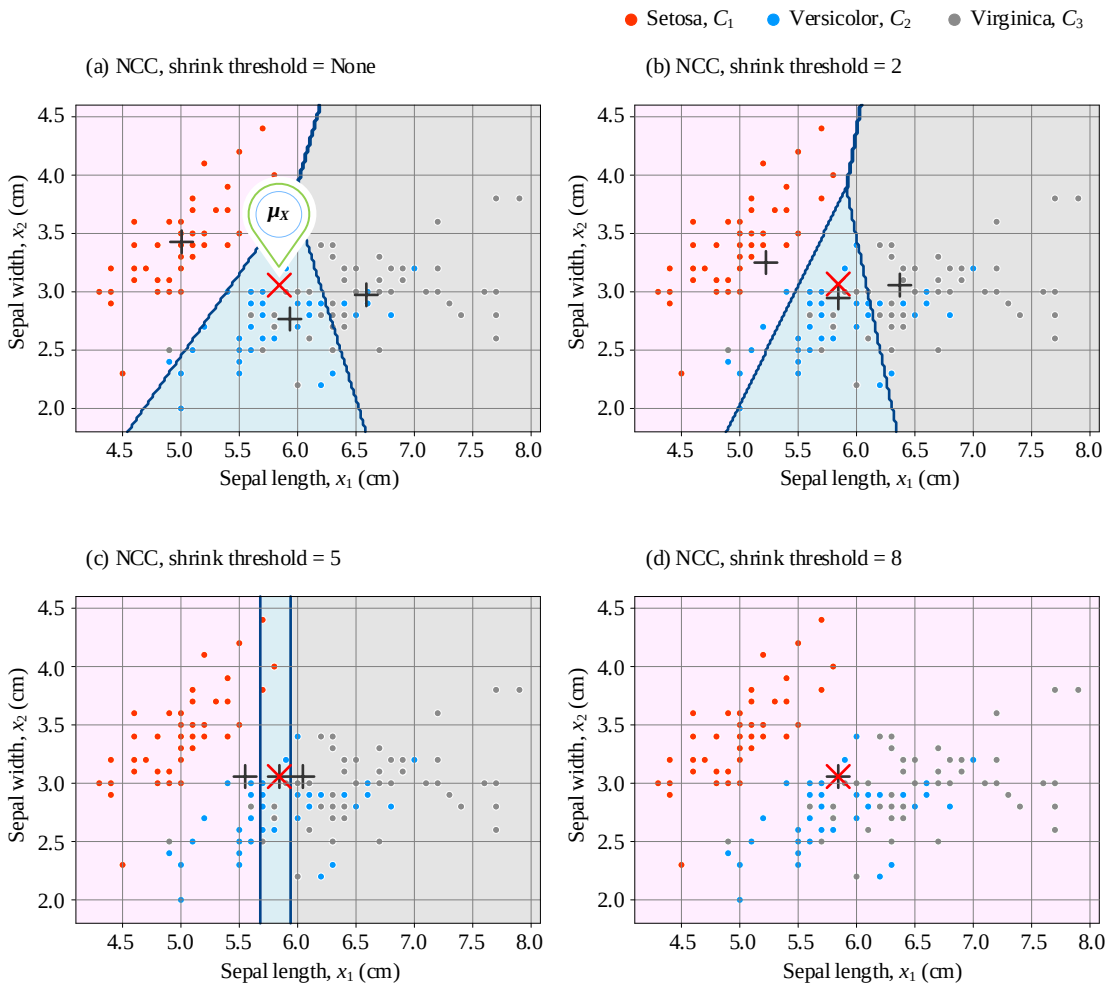


图 12. 收缩阈值增大对决策边界影响

本章探讨最简单的监督学习方法之一——最近邻 *k*-NN。最近邻方法可以用于分类问题，也可以用于回归问题。本书后文将介绍如何用最近邻 *k*-NN 完成回归任务。使用 *k*-NN 算法时，要注意近邻 *k* 值选择、距离度量，以及是否采用加权投票。

此外，最近质心分类 NCC 可以看做 *k*-NN 的简化版本，NCC 利用某一类成员质心表示该类别数据，不需要用户提供近邻数量 *k* 值，决策边界为中垂线。

最近邻这一思路是很多其他机器学习算法的基础，比如 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)、**流形学习** (manifold learning) 和**谱聚类** (spectral clustering) 也是基于最近邻思想。

本章给出的例子中距离度量均为欧氏距离；而实际应用中，距离度量种类繁多，需要大家理解距离的具体定义以及优缺点。下一章，我们聊一聊几种常见距离度量。

3 Distance Metrics
距离

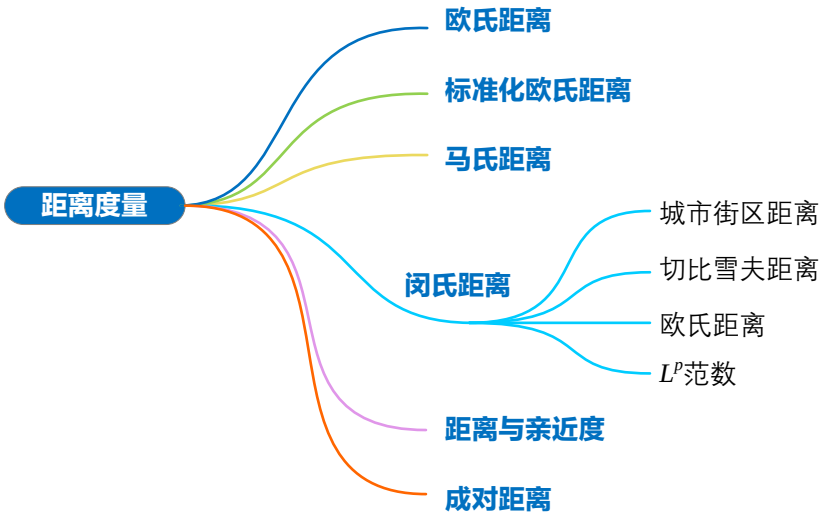
鸢尾花书有关距离度量的综述

人是万物的尺度：是存在事物存在的尺度，也是不存在事物不存在的尺度。

Man is the measure of all things: of things which are, that they are, and of things which are not, that they are not.

—— 普罗泰戈拉 (Protagoras) | 古希腊哲学家 | 490 ~ 420 BC

- `metrics.pairwise.linear_kernel()` 计算线性核成对亲近度矩阵
- `metrics.pairwise.manhattan_distances()` 计算成对城市街区距离矩阵
- `metrics.pairwise.paired_cosine_distances(X,Q)` 计算 X 和 Q 样本数据矩阵成对余弦距离矩阵
- `metrics.pairwise.paired_euclidean_distances(X,Q)` 计算 X 和 Q 样本数据矩阵成对欧氏距离矩阵
- `metrics.pairwise.paired_manhattan_distances(X,Q)` 计算 X 和 Q 样本数据矩阵成对城市街区距离矩阵
- `metrics.pairwise.polynomial_kernel()` 计算多项式核成对亲近度矩阵
- `metrics.pairwise.rbf_kernel()` 计算 RBF 核成对亲近度矩阵
- `metrics.pairwise.sigmoid_kernel()` 计算 sigmoid 核成对亲近度矩阵
- `numpy.diag()` 如果 A 为方阵，`numpy.diag(A)` 函数提取对角线元素，以向量形式输入结果；如果 a 为向量，`numpy.diag(a)` 函数将向量展开成方阵，方阵对角线元素为 a 向量元素
- `numpy.linalg.inv()` 计算逆矩阵
- `numpy.linalg.norm()` 计算范数
- `scipy.spatial.distance.chebyshev()` 计算切比雪夫距离
- `scipy.spatial.distance.cityblock()` 计算城市街区距离
- `scipy.spatial.distance.euclidean()` 计算欧氏距离
- `scipy.spatial.distance.mahalanobis()` 计算马氏距离
- `scipy.spatial.distance.minkowski()` 计算闵氏距离
- `scipy.spatial.distance.seuclidean()` 计算标准化欧氏距离
- `sklearn.metrics.pairwise.euclidean_distances()` 计算成对欧氏距离矩阵
- `sklearn.metrics.pairwise_distances()` 计算成对距离矩阵



3.1 还聊距离

上一章在讲解 *k*-NN 分类算法时，默认距离度量为欧几里得距离，实际应用中还有大量其他距离可供选择。

大家对距离这个概念应该非常熟悉，我们从《数学要素》第 7 章"开始就不断丰富"距离"的内涵。我们在《矩阵力量》第 3 章专门介绍了基于 L^p 范数的几种距离度量，在《统计至简》第 15 章专门讲解了马氏距离。

本章便专门总结并探讨常用的几个距离度量：

- **欧氏距离** (Euclidean distance)
- **标准化欧氏距离** (standardized Euclidean distance)
- **马氏距离** (Mahalanobis distance, Mahal distance)
- **城市街区距离** (city block distance)
- **切比雪夫距离** (Chebyshev distance)
- **闵氏距离** (Minkowski distance)
- **余弦距离** (cosine distance)
- **相关性距离** (correlation distance)

本章最后探讨距离和亲近度的关系。

3.2 欧氏距离：最常见的距离

欧几里得距离，也称**欧氏距离** (Euclidean distance)。欧氏距离是机器学习中常用的一种距离度量方法，适用于处理连续特征的数据。其特点是简单易懂、计算效率高，但容易受到数据维度、特征尺度、特征量纲影响。

任意样本数据点 $\boldsymbol{x}$ 和查询点 $\boldsymbol{q}$ 欧氏距离定义如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\| = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{q})} \tag{1}$$

其中，$\boldsymbol{x}$ 和 $\boldsymbol{q}$ 为列向量。欧氏距离本质上就是 $\boldsymbol{x}-\boldsymbol{q}$ 的 L^2 范数。从几何视角来看，二维欧氏距离可以看做同心正圆，三维欧氏距离可以视作同心正球体，等等。

当特征数为 D 时，上式展开可以得到：

$$d(\boldsymbol{x},\boldsymbol{q})=\sqrt{(x_1-q_1)^2+(x_2-q_2)^2+\ldots+(x_D-q_D)^2} \tag{2}$$

特别地，当特征数量 $D=2$ 时，$\boldsymbol{x}$ 和 $\boldsymbol{q}$ 两点欧氏距离定义为：

$$d(\boldsymbol{x},\boldsymbol{q})=\sqrt{(x_1-q_1)^2+(x_2-q_2)^2} \tag{3}$$

举个例子

如果查询点 $\boldsymbol{q}$ 有两个特征，并位于原点，即：

$$\boldsymbol{q}=\begin{bmatrix}0\\0\end{bmatrix} \tag{4}$$

如图 1 所示，三个样本点 $\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$的位置如下：

$$\boldsymbol{x}^{(1)}=\begin{bmatrix}-5 & 0\end{bmatrix},\quad \boldsymbol{x}^{(3)}=\begin{bmatrix}4 & 3\end{bmatrix},\quad \boldsymbol{x}^{(3)}=\begin{bmatrix}3 & -4\end{bmatrix} \tag{5}$$

根据 (1) 可以计算得到三个样本点 $\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$ 距离查询点 $\boldsymbol{q}$ 之间欧氏距离均为 5：

$$\begin{cases} d_1=\sqrt{([0\quad 0]-[-5\quad 0])([0\quad 0]-[-5\quad 0])^{\mathrm{T}}}=\sqrt{[5\quad 0][5\quad 0]^{\mathrm{T}}}=\sqrt{25+0}=5 \\ d_2=\sqrt{([0\quad 0]-[4\quad 3])([0\quad 0]-[4\quad 3])^{\mathrm{T}}}=\sqrt{[-4\quad -3][-4\quad -3]^{\mathrm{T}}}=\sqrt{16+9}=5 \\ d_3=\sqrt{([0\quad 0]-[3\quad -4])([0\quad 0]-[3\quad -4])^{\mathrm{T}}}=\sqrt{[-3\quad 4][-3\quad 4]^{\mathrm{T}}}=\sqrt{9+16}=5 \end{cases} \tag{6}$$

⚠ 注意，行向量和列向量的转置关系，本章后续不再区分行、列向量。

如图 1 所示，当 d 取定值时，上式相当于以 (q_1, q_2) 为圆心的正圆。

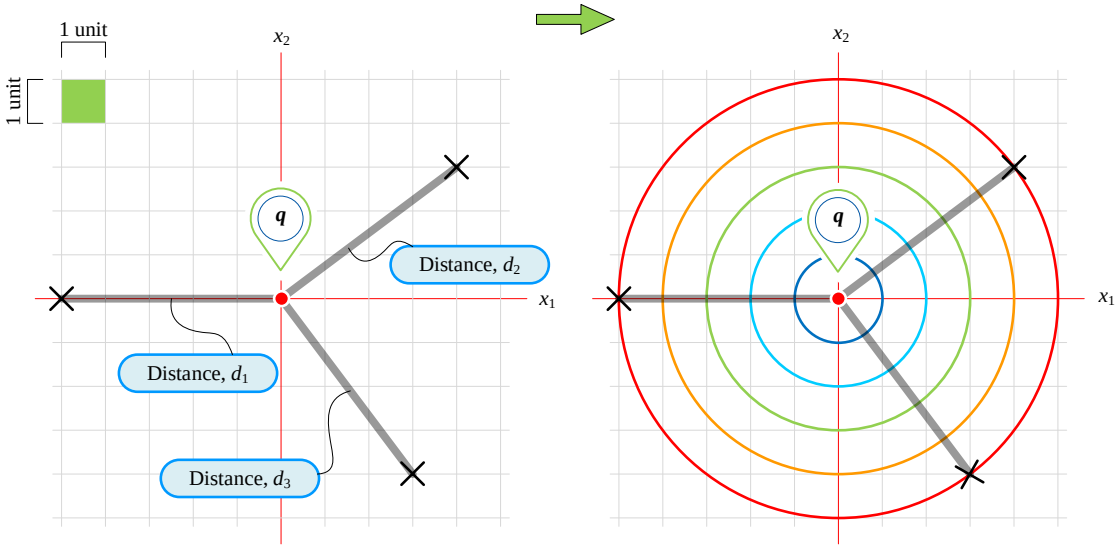


图 1. 3 特征 ($D=2$) 欧几里得距离

代码 Bk7_Ch03_01.py 计算两点欧氏距离。`scipy.spatial.distance.euclidean()`为计算欧氏距离的函数。

成对距离

如图 1 所示，三个样本点 $\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$ 之间也存在两两距离，我们管它们叫做**成对距离** (pairwise distance)。图 2 所示为平面上 12 个点的成对距离。成对距离结果一般以矩阵方式呈现。

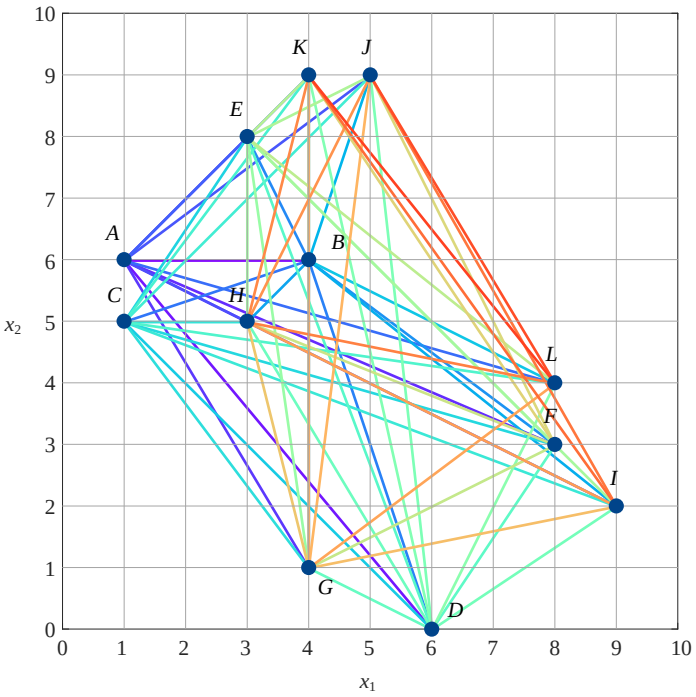


图 2. 平面上 12 个点，成对距离，来自鸢尾花书《数学要素》

代码 Bk7_Ch03_02.py 计算图 1 中三个样本点之间的成对欧氏距离。本章最后一节将专门介绍成对距离。

3.3 标准化欧氏距离：考虑标准差

标准化欧氏距离 (standardized Euclidean distance) 是一种将欧氏距离进行归一化处理的方法，适用于处理特征间尺度差异较大的数据。其特点是能够消除不同特征之间的度量单位和尺度差异，从而减少距离计算结果偏差。优点是比欧氏距离更具有鲁棒性和稳定性，缺点是对于一些特征较为稀疏的数据，可能存在一些计算上的困难。

定义

标准化欧氏距离定义如下。

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^{\mathrm{T}} \boldsymbol{D}^{-1}\boldsymbol{D}^{-1}(\boldsymbol{x}-\boldsymbol{q})} \tag{7}$$

其中，$\boldsymbol{D}$ 为对角方阵，对角线元素为标准差，运算如下：

$$\boldsymbol{D} = \operatorname{diag}\left(\operatorname{diag}(\boldsymbol{\Sigma})\right)^{\frac{1}{2}} = \operatorname{diag}\left(\operatorname{diag}\begin{bmatrix} \sigma_1^2 & \rho_{1,2}\sigma_1\sigma_2 & \cdots & \rho_{1,D}\sigma_1\sigma_D \\ \rho_{1,2}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2,D}\sigma_2\sigma_D \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1,D}\sigma_1\sigma_D & \rho_{2,D}\sigma_2\sigma_D & \cdots & \sigma_D^2 \end{bmatrix}\right)^{\frac{1}{2}} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D \end{bmatrix} \tag{8}$$

回忆《矩阵力量》介绍过有关 diag() 函数的说明。如果 $\boldsymbol{A}$ 为方阵，diag($\boldsymbol{A}$) 函数提取对角线元素，结果为向量；如果 $\boldsymbol{a}$ 为向量，diag($\boldsymbol{a}$) 函数将向量 $\boldsymbol{a}$ 展开成对角方阵，方阵对角线元素为 $\boldsymbol{a}$ 向量元素。NumPy 中完成这一计算的函数为 `numpy.diag()`。

将 (8) 带入 (7) 得到：

$$\begin{aligned} d(\boldsymbol{x},\boldsymbol{q}) &= \sqrt{\begin{bmatrix} x_1-q_1 & x_2-q_2 & \cdots & x_D-q_D \end{bmatrix}\begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2 \end{bmatrix}^{-1}\begin{bmatrix} x_1-q_1 & x_2-q_2 & \cdots & x_D-q_D \end{bmatrix}^{\mathrm{T}}} \\ &= \sqrt{\frac{(x_1-q_1)^2}{\sigma_1^2}+\frac{(x_2-q_2)^2}{\sigma_2^2}+\ldots+\frac{(x_D-q_D)^2}{\sigma_D^2}} = \sqrt{\sum_{j=1}^{D}\left(\frac{x_j-q_j}{\sigma_j}\right)^2} \end{aligned} \tag{9}$$

(9) 可以记做：

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{z_1^2+z_2^2+\ldots+z_D^2} = \sqrt{\sum_{j=1}^{D} z_j^2} \tag{10}$$

其中，z_j 为：

$$z_j = \frac{x_j-q_j}{\sigma_j} \tag{11}$$

上式类似 Z 分数。

《统计至简》第 9 章专门介绍 Z 分数，请大家回顾。

正椭圆

对于 $D = 2$，两特征的情况，标准化欧氏距离平方可以写成：

$$d^2 = \frac{(x_1 - q_1)^2}{\sigma_1^2} + \frac{(x_2 - q_2)^2}{\sigma_2^2} \tag{12}$$

可以发现，上式代表的形状是以 (q_1, q_2) 为中心的正椭圆。观察 (12)，可以发现，标准化欧氏距离引入数据每个特征标准差，但是没有考虑特征之间的相关性。图 3 中，网格的坐标已经转化为“标准差”，而标准欧氏距离等距线为正椭圆。

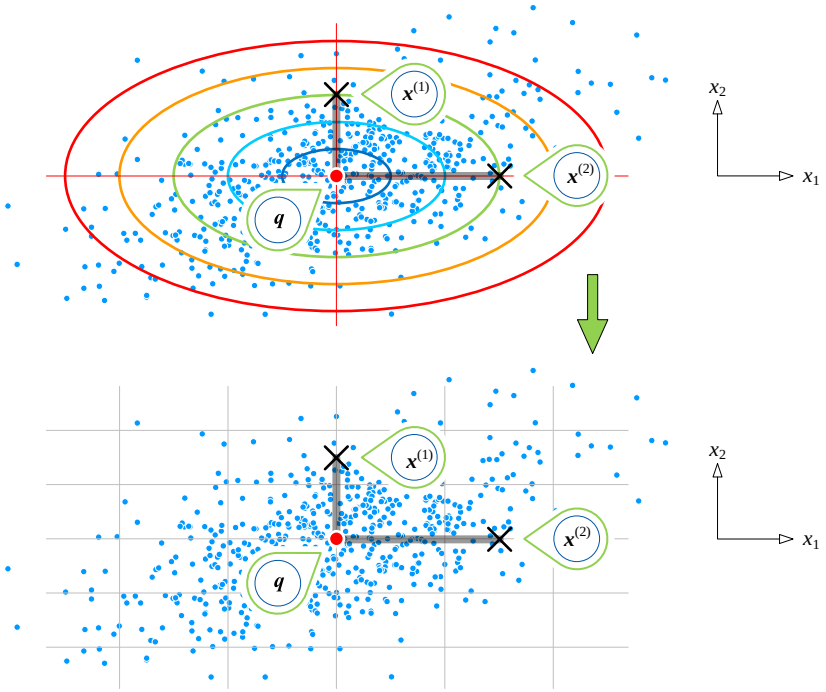


图 3. 2 特征 ($D = 2$) 标准化欧氏距离

几何变换视角

如图 4 所示，从几何变换角度，标准化欧氏距离相当于对 $\boldsymbol{X}$ 数据每个维度，首先**中心化** (centralize)，然后利用标准差进行**缩放** (scale)；但是，标准化欧氏距离没有旋转操作，也就是没有正交化。

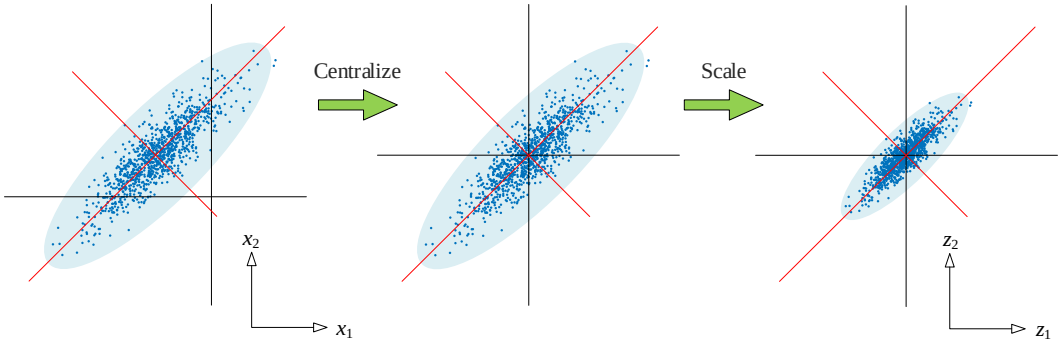


图 4. 标准化欧氏距离运算过程

计算标准化欧氏距离的函数为 `scipy.spatial.distance.seuclidean()`。代码 Bk7_Ch03_03.py 计算本节标准化欧氏距离。

3.4 马氏距离：考虑标准差和相关性

本系列丛书《矩阵力量》和《统计至简》从不同角度讲过马氏距离，本节稍作回忆。

马氏距离，**马哈距离** (Mahalanobis distance, Mahal distance)，全称马哈拉诺比斯距离，是机器学习中常用的一种距离度量方法，适用于处理高维数据和特征之间存在相关性的情况。其特点是考虑到特征之间的相关性，从而在计算距离时可以更好地描述数据之间的相似程度。优点是能够提高模型的准确性，缺点是对于样本数较少的情况下容易过拟合，计算量较大，同时对数据的分布形式存在假设前提 (多元正态分布)。

马氏距离定义如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{q})} \tag{13}$$

其中，$\boldsymbol{\Sigma}$ 为协方差矩阵，$\boldsymbol{q}$ 一般是样本数据的质心。

注意，马氏距离的单位是“标准差”。比如，马氏距离计算结果为 3，应该称作 3 个标准差。

特征值分解：缩放 → 旋转 → 平移

$\boldsymbol{\Sigma}$ 谱分解得到：

$$\boldsymbol{\Sigma} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}} \tag{14}$$

其中，$\boldsymbol{V}$ 为正交矩阵。

$\boldsymbol{\Sigma}^{-1}$ 的特征值分解可以写成：

$$\boldsymbol{\Sigma}^{-1} = \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}\right)^{-1} = \left(\boldsymbol{V}^{\mathrm{T}}\right)^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^{-1} = \boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^{\mathrm{T}} \tag{15}$$

将 (15) 代入 (13) 得到：

$$d\left(\boldsymbol{x},\boldsymbol{\mu}\right) = \left\| \underbrace{\boldsymbol{\Lambda}^{\frac{-1}{2}}}_{\text{Scale}} \underbrace{\boldsymbol{V}^{\mathrm{T}}}_{\text{Rotate}} \underbrace{\left(\boldsymbol{x}-\boldsymbol{\mu}\right)}_{\text{Centralize}} \right\| \tag{16}$$

其中，$\boldsymbol{q}$ 列向量完成**中心化** (centralize)，$\boldsymbol{V}$ 矩阵完成**旋转** (rotate)，$\boldsymbol{\Lambda}$ 矩阵完成**缩放** (scale)。

旋转椭圆

如图 5 所示，当 $D = 2$ 时，马氏距离的等距线为旋转椭圆。

大家如果对这部分内容感到陌生，请回顾《矩阵力量》第 20 章、《统计至简》第 23 章。

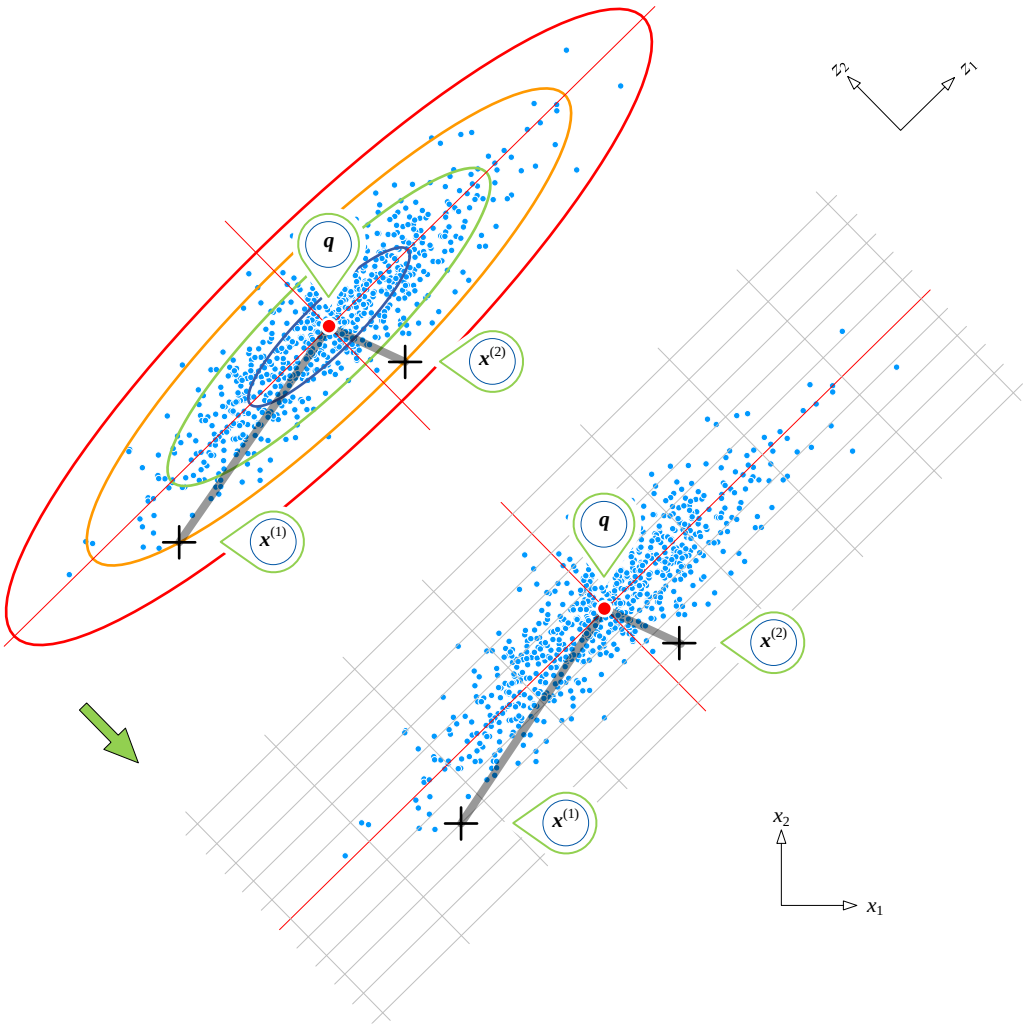


图 5. 2 特征 (D = 2) 马氏距离

代码 Bk7_Ch03_04.py 计算图 5 两个点的马氏距离。

举例

下面，我们用具体数字举例讲解如何计算马氏距离。

给定质心 $\boldsymbol{\mu} = [0, 0]^T$。两个样本点的坐标分别为。

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -3.5 & -4 \end{bmatrix}^T, \quad \boldsymbol{x}^{(2)} = \begin{bmatrix} 2.75 & -1.5 \end{bmatrix}^T \tag{17}$$

计算得到 $\boldsymbol{x}^{(1)}$和 $\boldsymbol{x}^{(2)}$距离 $\boldsymbol{\mu}$ 之间欧氏距离 (L^2 范数) 分别为 5.32 和 3.13。

假设方差协方差矩阵 $\boldsymbol{\Sigma}$ 取值如下。

$$\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{18}$$

观察如上矩阵，可以发现 x_1 和 x_2 特征各自的方差均为 2，两者协方差为 1；计算得到 x_1 和 x_2 特征相关性为 0.5。根据 $\boldsymbol{\Sigma}$ 计算 $\boldsymbol{x}^{(1)}$和 $\boldsymbol{x}^{(2)}$距离 $\boldsymbol{\mu}$ 之间马氏距离为。

$$\begin{aligned} d_1 &= \sqrt{\left([-3.5 \quad -4] - [0 \quad 0]\right) \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \left([-3.5 \quad -4] - [0 \quad 0]\right)^{\mathrm{T}}} \\ &= \sqrt{[-3.5 \quad -4] \cdot \frac{1}{3} \cdot \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} [-3.5 \quad -4]^{\mathrm{T}}} = 3.08 \\ d_2 &= \sqrt{\left([2.75 \quad -1.5] - [0 \quad 0]\right) \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1} \left([2.75 \quad -1.5] - [0 \quad 0]\right)^{\mathrm{T}}} \\ &= \sqrt{[2.75 \quad -1.5] \cdot \frac{1}{3} \cdot \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} [2.75 \quad -1.5]^{\mathrm{T}}} = 3.05 \end{aligned} \tag{19}$$

可以发现，$\boldsymbol{x}^{(1)}$和 $\boldsymbol{x}^{(2)}$ 和 $\boldsymbol{\mu}$ 之间马氏距离非常接近。

3.5 城市街区距离：L^1范数

城市街区距离 (city block distance)，也称**曼哈顿距离** (Manhattan distance)，和欧氏距离本质上都是 L^p 范数。请大家注意区别两者等高线。

城市街区距离具体定义如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x} - \boldsymbol{q}\|_1 = \sum_{j=1}^{D} |x_j - q_j| \tag{20}$$

其中，j 代表特征序号。

城市街区距离就是我们在《矩阵力量》第 3 章中介绍的 L^1 范数。

将 (20) 展开得到下式：

$$d(\boldsymbol{x},\boldsymbol{q}) = |x_1 - q_1| + |x_2 - q_2| + \ldots + |x_D - q_D| \tag{21}$$

特别地，当 $D = 2$ 时，城市街区距离为：

$$d(\boldsymbol{x},\boldsymbol{q}) = |x_1 - q_1| + |x_2 - q_2| \tag{22}$$

旋转正方形

如图 6 所示，城市街区距离的等距线为旋转正方形。图中，$\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$ 和 $\boldsymbol{q}$ 欧氏距离均为 5，但是城市街区距离分别为 5、7 和 7。

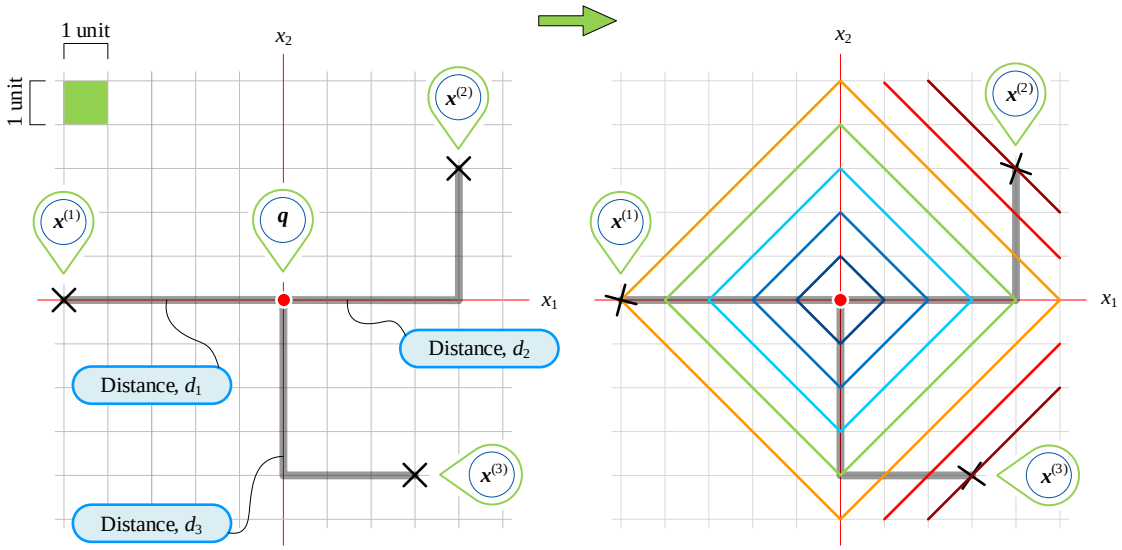


图 6. 2 特征 (D = 2) 城市街区距离

代码 Bk7_Ch03_05.py 给出两种方法计算得到图 6 所示城市街区距离。

3.6 切比雪夫距离：L^∞范数

切比雪夫距离 (Chebyshev distance)，具体如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\|_\infty = \max_j\left\{\left|x_j - q_j\right|\right\} \tag{23}$$

切比雪夫距离就是我们在《矩阵力量》第 3 章中介绍的 L^∞范数。

将 (23) 展开得到下式：

$$d(\boldsymbol{x},\boldsymbol{q}) = \max\left\{\left|x_1 - q_1\right|,\ \left|x_2 - q_2\right|,\ \ldots,\ \left|x_D - q_D\right|\right\} \tag{24}$$

特别地，当 D = 2 时，切比雪夫距离为：

$$d(\boldsymbol{x},\boldsymbol{q}) = \max\left\{\left|x_1 - q_1\right|,\ \left|x_2 - q_2\right|\right\} \tag{25}$$

正方形

如图 7 所示，切比雪夫距离等距线为正方形。前文提到，$\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$ 和 $\boldsymbol{q}$ 欧氏距离相同，但是切比雪夫距离分别为 5、4 和 4。

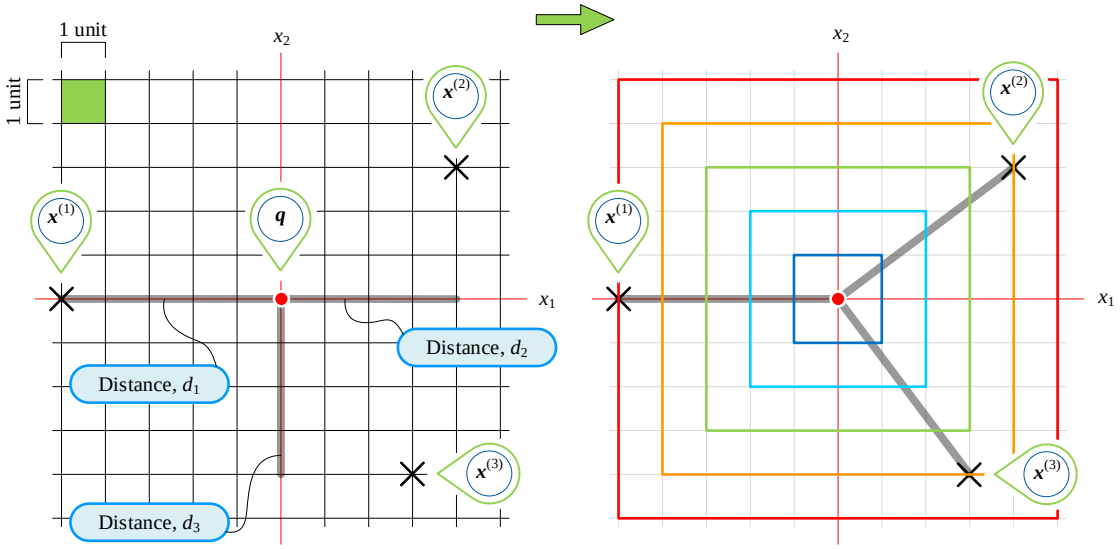


图 7. 2 特征 (D = 2) 切比雪夫距离

代码 Bk7_Ch03_06.py 计算图 7 所示切比雪夫距离。

3.7 闵氏距离：L^p 范数

闵氏距离 (Minkowski distance) 类似 L^p 范数，对应定义如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\|_p = \left(\sum_{j=1}^{D}\left|x_j - q_j\right|^p\right)^{1/p} \tag{26}$$

⚠ 注意，$p \geqslant 1$ 时上式才叫向量范数。

计算闵氏距离的函数为 `scipy.spatial.distance.minkowski()`。

图 8 所示为 p 取不同值时，闵氏距离等距线图。特别地，p = 1 时，闵氏距离为城市街区距离；p = 2 时，闵氏距离为欧氏距离；$p \to \infty$ 时，闵氏距离为切比雪夫距离。

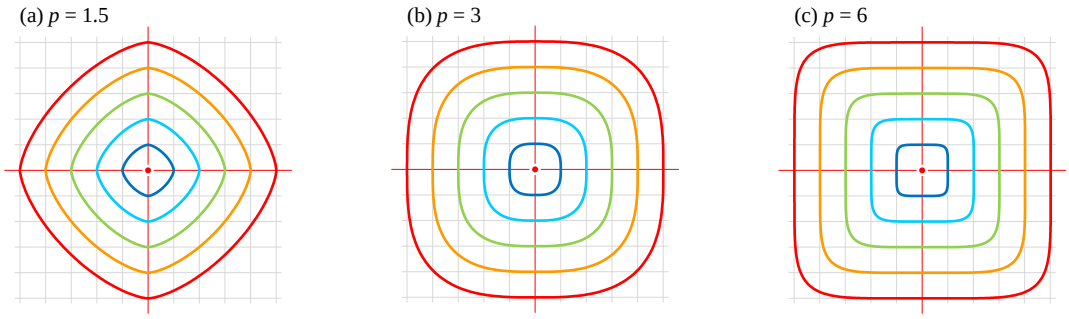


图 8. 闵氏距离 ($D = 2$)，p 取不同值

3.8 距离与亲近

本节介绍和距离相反的度量——**亲近度** (affinity)。两个样本数据距离越远，两者亲近度越低；而当它们距离越近，亲近度则越高。亲近度，也称**相似度** (similarity)。

余弦相似度

《矩阵力量》第 2 章讲过，**余弦相似度** (cosine similarity) 用向量夹角的余弦值度量样本数据的相似性。$\boldsymbol{x}$ 和 $\boldsymbol{q}$ 两个向量的余弦相似度具体定义如下：

$$k(\boldsymbol{x},\boldsymbol{q}) = \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} = \frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} \tag{27}$$

如图 9 所示，如果两个向量方向相同，则夹角 θ 余弦值 $\cos(\theta)$ 为 1；如果，两个向量方向完全相反，夹角 θ 余弦值 $\cos(\theta)$ 为 −1。因此余弦相似度取值范围在 $[-1, +1]$ 之间。

⚠ 注意，余弦相似度和向量模无关，仅仅与两个向量夹角有关。

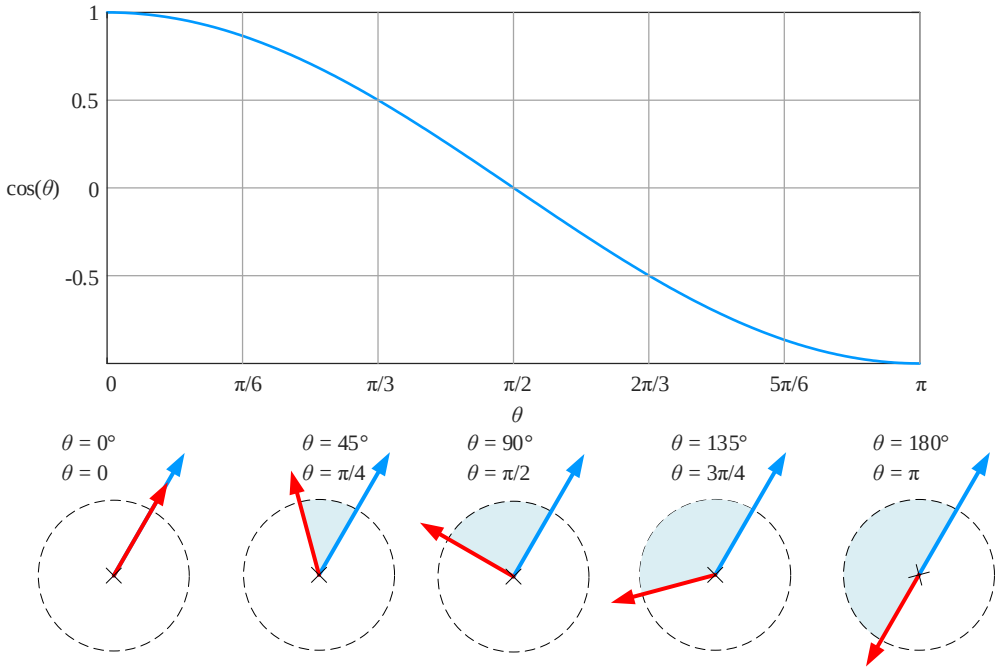


图 9. 余弦相似度

举个例子

给定如下两个向量具体值：

$$\boldsymbol{x}=\begin{bmatrix}8 & 2\end{bmatrix}^{\mathrm{T}},\quad \boldsymbol{q}=\begin{bmatrix}7 & 9\end{bmatrix}^{\mathrm{T}} \tag{28}$$

将 (28) 代入 (27) 得到：

$$k(\boldsymbol{x},\boldsymbol{q})=\frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|}=\frac{8\times7+2\times9}{\sqrt{8^2+2^2}\times\sqrt{7^2+9^2}}=\frac{74}{\sqrt{68}\times\sqrt{130}}=0.7871 \tag{29}$$

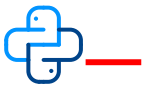
代码 Bk7_Ch03_07.py 得到和 (29) 一致结果。

余弦距离

余弦距离 (cosine distance) 的定义如下：

$$d(\boldsymbol{x},\boldsymbol{q})=1-k(\boldsymbol{x},\boldsymbol{q})=1-\frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|}=1-\frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} \tag{30}$$

余弦相似度的取值范围 [−1, +1] 之间，因此余弦距离的取值范围为 [0, 2]。



代码计算 (28) 中两个向量的余弦距离，结果为 0.2129。也可以采用 `scipy.spatial.distance.pdist(X, 'cosine')` 函数计算余弦距离。

相关系数相似度

相关系数相似度 (correlation similarity) 定义如下：

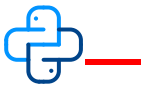
$$k(\boldsymbol{x},\boldsymbol{q}) = \frac{(\boldsymbol{x}-\bar{x})^{\mathrm{T}}(\boldsymbol{q}-\bar{q})}{\|\boldsymbol{x}-\bar{x}\|\|\boldsymbol{q}-\bar{q}\|} = \frac{(\boldsymbol{x}-\bar{x})\cdot(\boldsymbol{q}-\bar{q})}{\|\boldsymbol{x}-\bar{x}\|\|\boldsymbol{q}-\bar{q}\|} \tag{31}$$

其中，$\bar{x}$ 为列向量 $\boldsymbol{x}$ 元素均值；$\bar{q}$ 为列向量 $\boldsymbol{q}$ 元素均值。

观察 (31)，发现相关系数相似度类似余弦相似度；稍有不同的是，相关系数相似度需要“中心化”向量。

还是以 (28) 为例，计算 $\boldsymbol{x}$ 和 $\boldsymbol{q}$ 两个向量的相关系数相似度。将 (28) 代入 (31) 可以得到：

$$\begin{aligned} k(\boldsymbol{x},\boldsymbol{q}) &= \frac{\left(\begin{bmatrix}8 & 2\end{bmatrix}^{\mathrm{T}} - \frac{8+2}{2}\right)\cdot\left(\begin{bmatrix}7 & 9\end{bmatrix}^{\mathrm{T}} - \frac{7+9}{2}\right)}{\|\boldsymbol{x}-\bar{\boldsymbol{x}}\|\|\boldsymbol{q}-\bar{\boldsymbol{q}}\|} \\ &= \frac{\begin{bmatrix}3 & -3\end{bmatrix}^{\mathrm{T}}\cdot\begin{bmatrix}-1 & 1\end{bmatrix}^{\mathrm{T}}}{\left\|\begin{bmatrix}3 & -3\end{bmatrix}^{\mathrm{T}}\right\|\left\|\begin{bmatrix}-1 & 1\end{bmatrix}^{\mathrm{T}}\right\|} = \frac{-6}{6} = -1 \end{aligned} \tag{32}$$



代码 Bk7_Ch03_09.py 计算得到两个向量的相关系数距离为 2。也可以采用 `scipy.spatial.distance.pdist(X, 'correlation')` 函数计算相关系数距离。

核函数亲近度

不考虑常数项，**线性核** (linear kernel) 亲近度定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q} = \boldsymbol{x}\cdot\boldsymbol{q} \tag{33}$$

对比 (27) 和 (33)，(27) 分母上 $\|\boldsymbol{x}\|$ 和 $\|\boldsymbol{q}\|$ 分别对 $\boldsymbol{x}$ 和 $\boldsymbol{q}$ 归一化。

`sklearn.metrics.pairwise.linear_kernel` 为 scikit-learn 工具箱中计算线性核亲近度函数。

将 (28) 代入 (33)，得到线性核亲近度为：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = 8\times 7 + 2\times 9 = 74 \tag{34}$$

多项式核 (polynomial kernel) 亲近度定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \left(\gamma \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q} + r\right)^{d} = \left(\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r\right)^{d} \tag{35}$$

其中，d 为多项式核次数，γ 为系数，r 为常数。

多项式核亲近度函数为 `sklearn.metrics.pairwise.polynomial_kernel`。

Sigmoid 核 (sigmoid kernel) 亲近度定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \tanh\left(\gamma \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q} + r\right) = \tanh\left(\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r\right) \tag{36}$$

Sigmoid 核亲近度函数为 `sklearn.metrics.pairwise.sigmoid_kernel`。

最常见的莫过于，**高斯核** (Gaussian kernel) 亲近度，即**径向基核函数** (radial basis function kernel, RBF kernel)：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma \left\|\boldsymbol{x}-\boldsymbol{q}\right\|^2\right) \tag{37}$$

(37) 中 $\left\|\boldsymbol{x}-\boldsymbol{q}\right\|^2$ 为欧氏距离的平方，(37) 也可以写作：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma d^2\right) \tag{38}$$

其中，d 为欧氏距离 $\left\|\boldsymbol{x}-\boldsymbol{q}\right\|$。高斯核亲近度取值范围为 [0, 1]；距离值越小，亲近度越高。高斯核亲近度函数为 `sklearn.metrics.pairwise.rbf_kernel`。

图 10 所示为，γ 取不同值时，高斯核亲近度随着欧氏距离 d 变化。聚类算法经常采用高斯核亲近度。

从“距离 → 亲近度”转换角度来看，多元高斯分布分子中高斯函数完成的就是马氏距离 d 到概率密度 (亲近度) 的转化：

$$f_{\chi}(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}} = \frac{\exp\left(-\frac{1}{2}d^2\right)}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}} \tag{39}$$

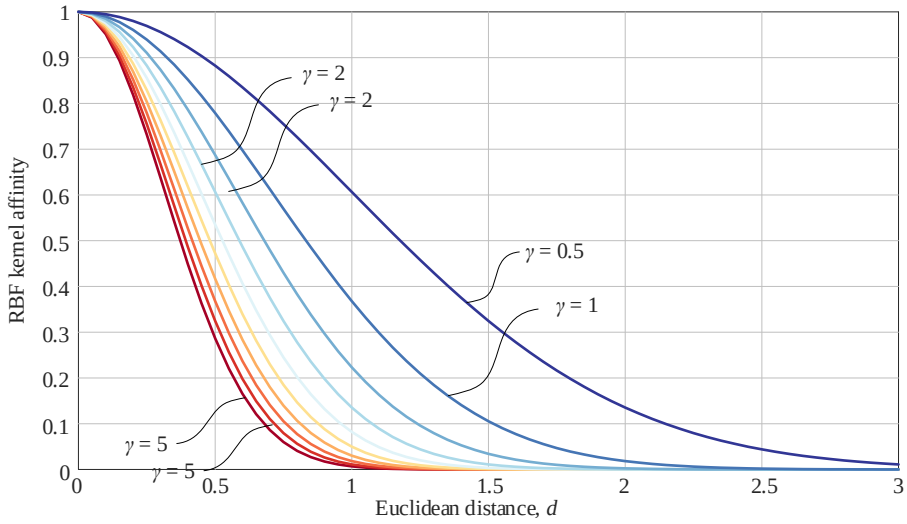


图 10. 高斯核亲近度随欧氏距离变化

拉普拉斯核 (Laplacian kernel) 亲近度，定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma \left\| \boldsymbol{x}-\boldsymbol{q} \right\|_1\right) \tag{40}$$

其中，$\left\| \boldsymbol{x}-\boldsymbol{q} \right\|_1$ 为城市街区距离。

图 11 所示为，γ 取不同值时，拉普拉斯核亲近度随着城市街区距离 d 变化。拉普拉斯核亲近度对应函数为 `sklearn.metrics.pairwise.laplacian_kernel`。

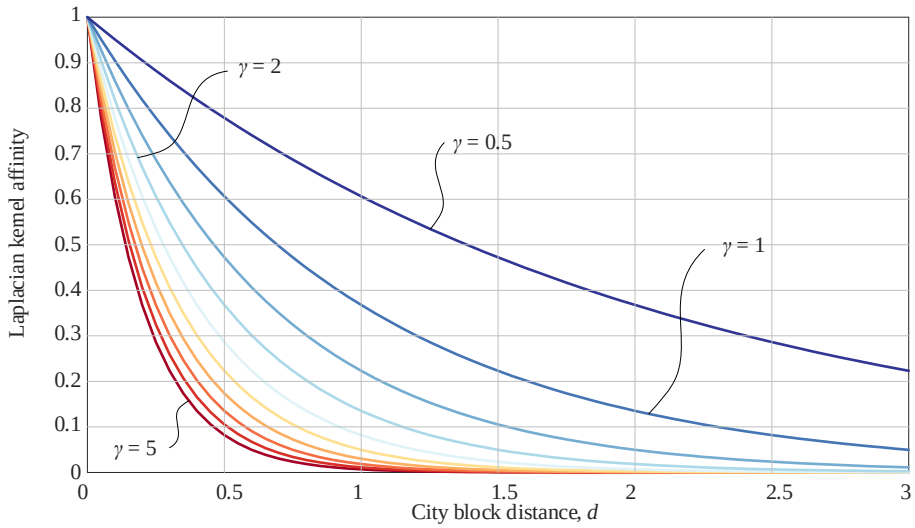


图 11. 拉普拉斯核亲近度随距离变化

3.9 成对距离、成对亲近度

《矩阵力量》反复强调，样本数据矩阵 $\boldsymbol{X}$ 每一列代表一个特征，而每一行代表一个样本数据点，比如：

$$\boldsymbol{X}_{n\times D}=\begin{bmatrix}\boldsymbol{x}^{(1)}\\ \boldsymbol{x}^{(2)}\\ \vdots\\ \boldsymbol{x}^{(n)}\end{bmatrix} \tag{41}$$

本书中，$\boldsymbol{x}^{(i)}$ 多数时候被当做是列向量，此时 $\boldsymbol{X}$ 为：

$$\boldsymbol{X}_{n\times D}=\begin{bmatrix}\boldsymbol{x}^{(1)\mathrm{T}}\\ \boldsymbol{x}^{(2)\mathrm{T}}\\ \vdots\\ \boldsymbol{x}^{(n)\mathrm{T}}\end{bmatrix} \tag{42}$$

$\boldsymbol{X}$ 样本点之间距离构成的**成对距离矩阵** (pairwise distance matrix) 形式如下：

$$\boldsymbol{D}_{n\times n}=\begin{bmatrix}0 & d_{1,2} & d_{1,3} & \cdots & d_{1,n}\\ d_{2,1} & 0 & d_{2,3} & \cdots & d_{2,n}\\ d_{3,1} & d_{3,2} & 0 & \cdots & d_{3,n}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ d_{n,1} & d_{n,2} & d_{n,3} & \cdots & 0\end{bmatrix} \tag{43}$$

每个样本数据点和自身的距离为 0，因此 (43) 主对角线为 0。很显然矩阵 $\boldsymbol{D}$ 为对称矩阵，即 $d_{i,j}$ 和 $d_{j,i}$ 相等。

图 12 给定 12 个样本数据点坐标点。

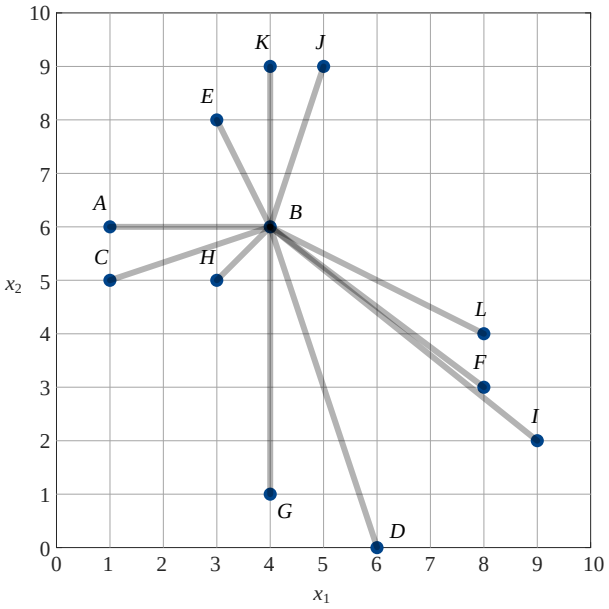


图 12. 样本数据散点图和成对距离

利用 `sklearn.metrics.pairwise.euclidean_distances`，我们可以计算图 12 数据点的成对欧氏距离矩阵。图 13 所示为欧氏距离矩阵数据构造的热图。

实际上，我们关心的成对距离个数为：

$$C_n^2 = \frac{n(n-1)}{2} \tag{44}$$

也就是说，(43) 中不含对角线的下三角矩阵包含的信息足够使用。

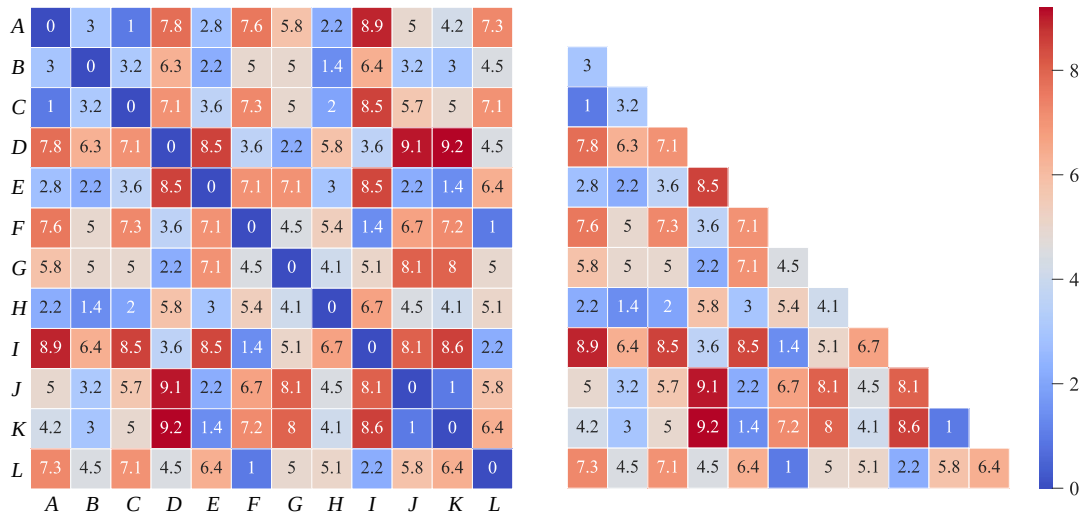


图 13. 样本数据成对距离矩阵热图

表 1 总结计算成对距离、亲近度矩阵常用函数。

表 1. 计算成对距离/亲近度矩阵常见函数

函数	描述
metrics.pairwise.cosine_similarity()	计算余弦相似度成对矩阵
metrics.pairwise.cosine_distances()	计算成对相似性距离矩阵
metrics.pairwise.euclidean_distances()	计算成对欧氏距离矩阵
metrics.pairwise.laplacian_kernel()	计算拉普拉斯核成对亲近度矩阵
metrics.pairwise.linear_kernel()	计算线性核成对亲近度矩阵
metrics.pairwise.manhattan_distances()	计算成对城市街区距离矩阵
metrics.pairwise.polynomial_kernel()	计算多项式核成对亲近度矩阵
metrics.pairwise.rbf_kernel()	计算 RBF 核成对亲近度矩阵
metrics.pairwise.sigmoid_kernel()	计算 sigmoid 核成对亲近度矩阵
metrics.pairwise.paired_euclidean_distances(X,Q)	计算 X 和 Q 样本数据矩阵成对欧氏距离矩阵
metrics.pairwise.paired_manhattan_distances(X,Q)	计算 X 和 Q 样本数据矩阵成对城市街区距离矩阵
metrics.pairwise.paired_cosine_distances(X,Q)	计算 X 和 Q 样本数据矩阵成对余弦距离矩阵

代码 Bk7_Ch03_10.py 可以绘制图 12、图 13。

树形图

图 13 数据矩阵是很多机器学习算法的起点；看似杂乱无章的图 13，实际上隐含很多重要信息。下面介绍**树形图** (dendrogram)，让大家领略成对距离/亲近度矩阵的力量。

下载 12 只股票历史股价，初值归一走势如图 14 所示。计算日对数回报率，然后估算相关系数矩阵，如图 15 热图所示。相关系数相当于亲近度，相关系数越高，说明股票涨跌趋势越相似。利用树形图，我们可以清楚看到各种股票之间的关联。

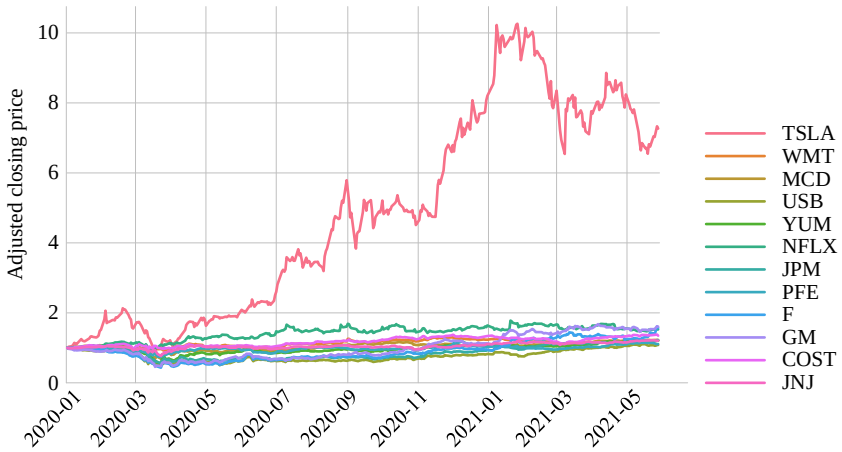


图 14. 12 只股票股价水平，初始股价归一化

PFE 和 JNJ 同属医疗，WMT 和 COST 同属零售，F 和 GM 同属汽车，USB 和 JPM 同属金融，MCD 和 YUM 同属餐饮；因此，它们之间相关性高并不足为奇。但是，本应该离汽车更近的 TSLA，却展现出和 NFLX 更高的相似性。

图 16 给出的树形图，直观地表达样本数据之间的距离/亲密度关系。树形图纵坐标高度表达不同数据之间的距离。

USB 和 JPM 之间相关性系数最高，因此 USB 和 JPM 距离最近，所以在树形图中首先将这两个节点相连，形成一个新的节点。然后，MCD 和 YUM 形成一个节点，F 和 GM 形成一个节点 … 依据这种方式，树形自下而上不断聚拢。有关树形图的原理，本书将在层次聚类一章中讲解。

图 16 树形图将股票按照相似度重新排列顺序。图 16 热图发生有意思的变化，热图中出现一个个色彩相近“方块”。每一个“方块”实际上代表着一类相似的数据点。因此，树形图很好揭示股票之间的相似性关系，这便是**聚类** (clustering) 算法的一种思路。

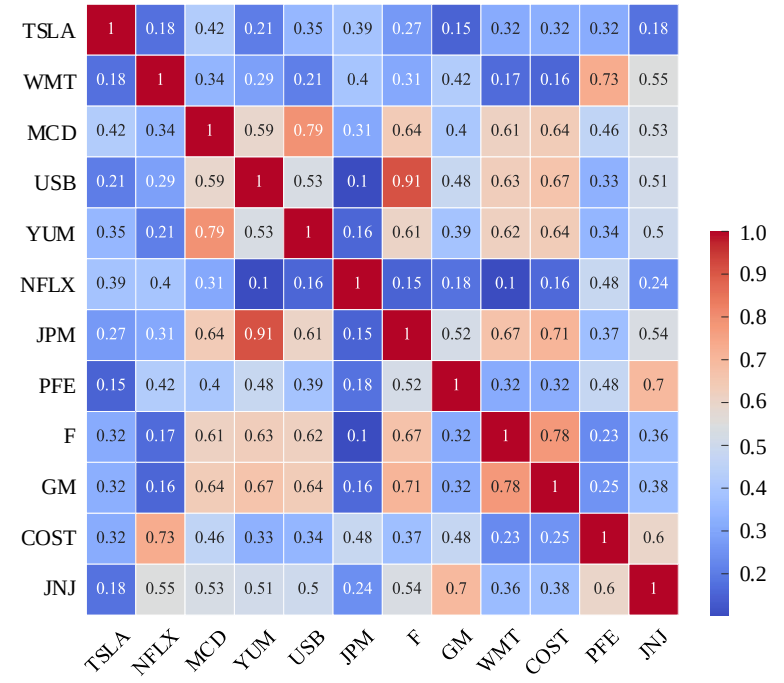


图 15. 12 只股票相关性热图

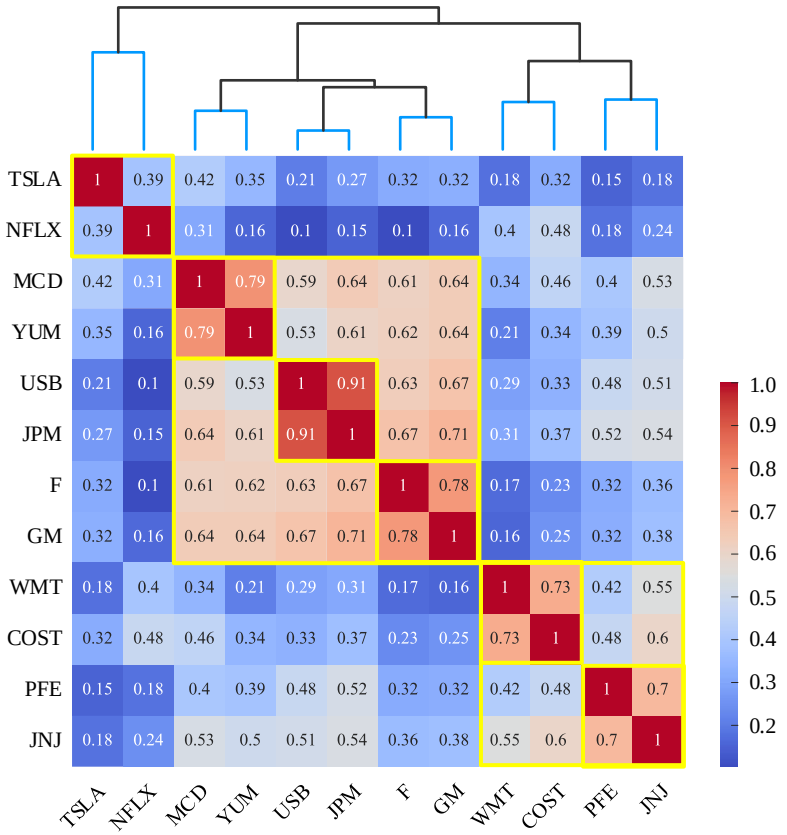
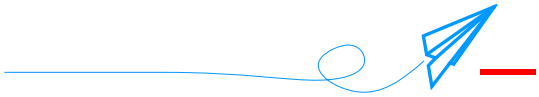


图 16. 根据树形图重组相关性热图

代码 Bk7_Ch03_11.py 绘制图 14、图 15 和图 16。



本章主要介绍了几种常见的距离度量和亲近度。机器学习中的距离，并不简单指的是“两点一线”，需要具体问题具体分析。特别希望大家能够结合丛书之前讲解的有关椭圆、矩阵转化和统计相关内容，强化对马氏距离的理解。此外，“远亲不如近邻”，两个点距离越近，两个点的“亲近度”或“相似度”也就越高。

Naive Bayes Classifier

4 朴素贝叶斯分类

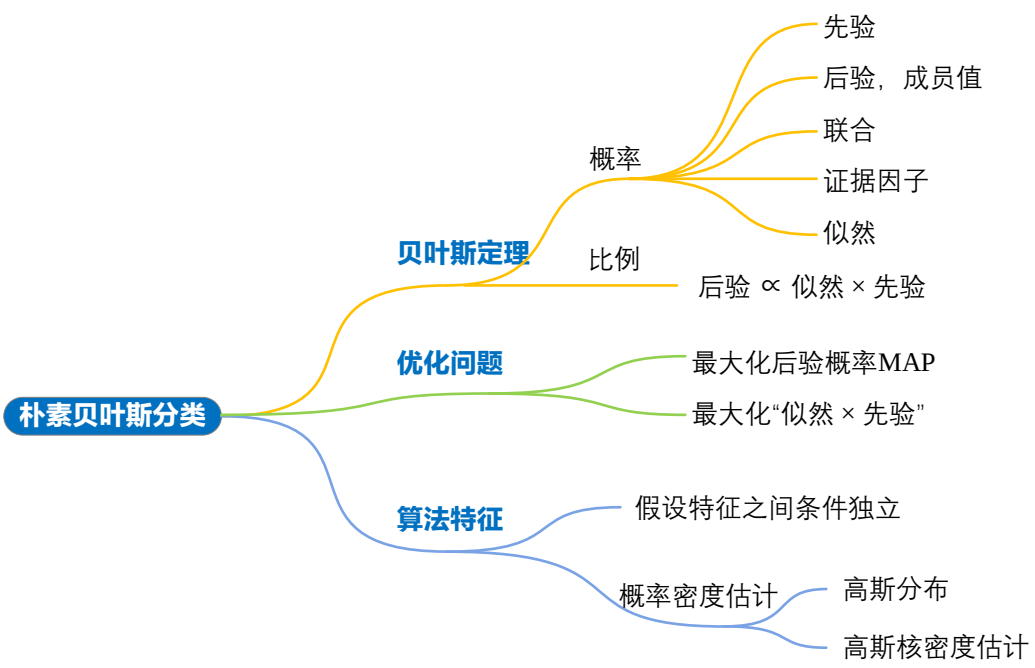
假设特征之间条件独立，最大化后验概率

大家使用朴素贝叶斯分类器时，假设特征 (条件) 独立。之所以称之“朴素”，是因为那真是个“天真”的假设。

A learner that uses Bayes' theorem and assumes the effects are independent given the cause is called a Naïve Bayes classifier. That's because, well, that's such a naïve assumption.

—— 佩德罗·多明戈斯 (Pedro Domingos) | 《终极算法》作者，华盛顿大学教授 | 1965 ~

- `matplotlib.axes.Axes.contour()` 绘制平面和空间等高线图
- `matplotlib.Axes3D.plot_wireframe()` 绘制三维单色网格图
- `matplotlib.pyplot.bar()` 绘制直方图
- `seaborn.barplot()` 绘制直方图
- `seaborn.displot()` 绘制一元和二元条件边际分布
- `seaborn.jointplot()` 同时绘制分类数据散点图、分布图和边际分布图



4.1 重逢贝叶斯

贝叶斯是我们的老朋友，《统计至简》薄频率派，厚贝叶斯派，这本书用了很大篇幅介绍了**贝叶斯定理** (Bayes' theorem) 和应用。《统计至简》第 18、19 章介绍贝叶斯分类的理论基础，第 20、21、22 章介绍贝叶斯统计推断。

本章和下一章，贝叶斯定理将专门用来解决数据分类问题。这种分类方法叫做**朴素贝叶斯分类** (Naive Bayes classification)。简单来说，朴素贝叶斯分类是一种基于贝叶斯定理和特征条件独立假设的分类方法，其原理是利用已知分类标记的训练数据，计算每个类别的条件概率分布，并根据贝叶斯定理计算未知样本属于每个类别的后验概率，最终将样本分配给具有最高概率的类别。其优点包括算法简单、计算高效、在处理大规模数据时表现良好，适用于多分类问题和高维数据；缺点是对特征的条件独立性要求较高，可能导致分类准确度下降，同时对于连续型变量的处理也存在一定困难。

本章和下一章共用一个思维导图。

托马斯·贝叶斯 (Thomas Bayes) | 英国数学家 | 1702 ~ 1761
贝叶斯统计的开山鼻祖，以贝叶斯定理闻名于世。关键词：● 贝叶斯定理 ● 朴素贝叶斯分类 ● 贝叶斯回归 ● 贝叶斯派

分类原理

朴素贝叶斯分类核心思想是比较后验概率大小。

比如，对于二分类问题 ($K = 2$)，就是比较某点 $\boldsymbol{x}$ 处，后验概率 (posterior) $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 的大小。

后验概率 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 本质上是条件概率 (conditional probability)。白话说，$f_{Y|X}(C_1 \mid \boldsymbol{x})$ 代表“给定 $\boldsymbol{x}$ 被分类为 C_1 的概率”，$f_{Y|X}(C_1 \mid \boldsymbol{x})$ 代表“给定 $\boldsymbol{x}$ 被分类为 C_2 的概率”。

如果 $f_{Y|X}(C_1 \mid \boldsymbol{x}) > f_{Y|X}(C_2 \mid \boldsymbol{x})$，$\boldsymbol{x}$ 被预测分类为 C_1；反之，$f_{Y|X}(C_1 \mid \boldsymbol{x}) < f_{Y|X}(C_2 \mid \boldsymbol{x})$，$\boldsymbol{x}$ 就被预测分类为 C_2。倘若 $f_{Y|X}(C_1 \mid \boldsymbol{x}) = f_{Y|X}(C_2 \mid \boldsymbol{x})$，该点便在决策边界 (decision boundary) 上。

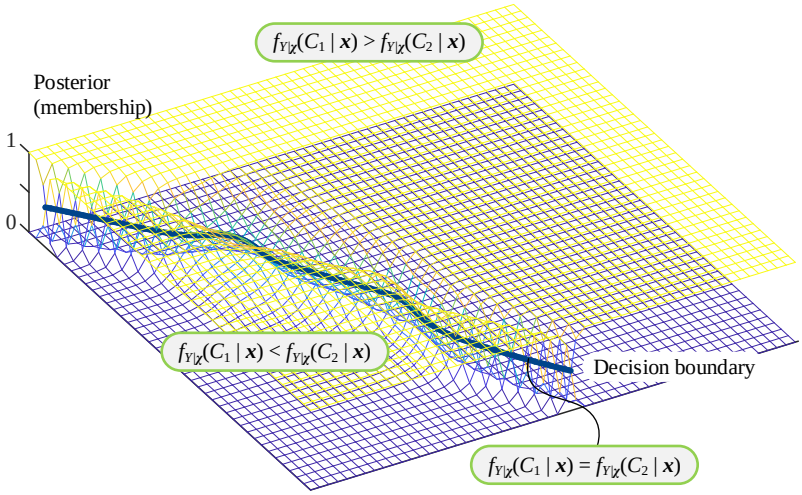


图 1. 二分类，比较后验概率大小，基于 KDE

比较图 1 所示 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 两个曲面。大家肯定已经发现，$f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 的取值在 [0, 1] 之间。实际上，$f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 并不是概率密度，它们本身就是概率。《统计至简》一册几次强调过这一点。

根据 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 两个曲面高度值，即概率值，我们可以确定决策边界 (图 1 中深蓝色实线)。

此外，对于二分类问题，$f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 之和为 1，下面简单证明一下。

全概率定理、贝叶斯定理

对于二分类问题，根据全概率定理 (law of total probability) 和贝叶斯定理 (Bayes' theorem)，$f_X(\boldsymbol{x})$ 可以通过下式计算得到：

$$\begin{aligned} f_X(\boldsymbol{x}) &= f_{X,Y}(\boldsymbol{x}, C_1) + f_{X,Y}(\boldsymbol{x}, C_2) \\ &= f_{Y|X}(C_1|\boldsymbol{x})\, f_X(\boldsymbol{x}) + f_{Y|X}(C_2|\boldsymbol{x})\, f_X(\boldsymbol{x}) \end{aligned} \tag{1}$$

$f_X(\boldsymbol{x})$ 不为 0 时，(1) 左右消去 $f_X(\boldsymbol{x})$，得到：

$$1 = f_{Y|X}(C_1|\boldsymbol{x}) + f_{Y|X}(C_2|\boldsymbol{x}) \tag{2}$$

白话解释，对于二分类问题，某点 $\boldsymbol{x}$ 要么属于 C_1，要么属于 C_2。

成员值：比较大小

后验概率值 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 取值在 [0, 1] 之间，且满足 (2)；因此，后验概率也常被称作成员值 (membership score)。

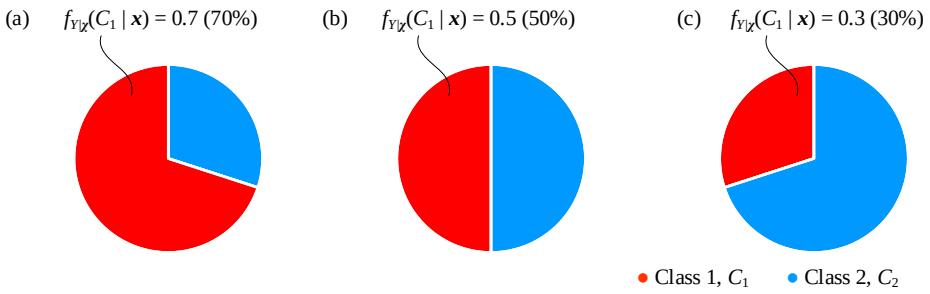


图 2. 二分类成员值

如图 2 (a) 所示，$f_{Y|X}(C_1 \mid \boldsymbol{x}) = 0.7$ (70%)，也就是说 $\boldsymbol{x}$ 属于 C_1 的可能性为 70%，即成员值为 0.7。这种情况，$\boldsymbol{x}$ 预测分类为 C_1。

$f_{Y|X}(C_1 \mid \boldsymbol{x}) = 0.5$ (50%) 时，对于二分类问题，$\boldsymbol{x}$ 应该位于决策边界上，相当于“骑墙派”，如图 2 (b) 所示。

若 $f_{Y|X}(C_1 \mid \boldsymbol{x}) = 0.3$ (30%)，$\boldsymbol{x}$ 属于 C_1 成员值为 0.3。显然，$\boldsymbol{x}$ 应该被预测分类为 C_2，如图 2 (c) 所示。

仅对于二分类问题，如果 $f_{Y|X}(C_1 \mid \boldsymbol{x}) > 0.5$，可以预测 $\boldsymbol{x}$ 分类为 C_1。

联合概率：比较大小

根据贝叶斯定理，对于二分类问题，证据因子 $f_X(\boldsymbol{x})$ 不为 0 时，后验概率 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 为：

$$
\begin{cases}
\underbrace{f_{Y|X}\left(C_1 \middle| \boldsymbol{x}\right)}_{\text{Posterior}} = \dfrac{\overbrace{f_{X,Y}\left(\boldsymbol{x}, C_1\right)}^{\text{Joint}}}{\underbrace{f_X\left(\boldsymbol{x}\right)}_{\text{Evidence}}} \\
\underbrace{f_{Y|X}\left(C_2 \middle| \boldsymbol{x}\right)}_{\text{Posterior}} = \dfrac{\overbrace{f_{X,Y}\left(\boldsymbol{x}, C_2\right)}^{\text{Joint}}}{\underbrace{f_X\left(\boldsymbol{x}\right)}_{\text{Evidence}}}
\end{cases}
\tag{3}
$$

观察 (3)，发现分母上均为证据因子 $f_X(\boldsymbol{x})$。这说明，后验概率 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 正比于联合概率 (joint probability, joint) $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$，即：

$$
\begin{cases}
\underbrace{f_{Y|X}\left(C_1 \middle| \boldsymbol{x}\right)}_{\text{Posterior}} \propto \underbrace{f_{X,Y}\left(\boldsymbol{x}, C_1\right)}_{\text{Joint}} \\
\underbrace{f_{Y|X}\left(C_2 \middle| \boldsymbol{x}\right)}_{\text{Posterior}} \propto \underbrace{f_{X,Y}\left(\boldsymbol{x}, C_2\right)}_{\text{Joint}}
\end{cases}
\tag{4}
$$

对于二分类问题，比较联合概率 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 大小，便可以预测分类！

图 3 给出的是某个二分类问题中，联合概率 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 两个曲面。通过比较 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 两个曲面高度，我们可以得出和图 1 一样的分类结论。

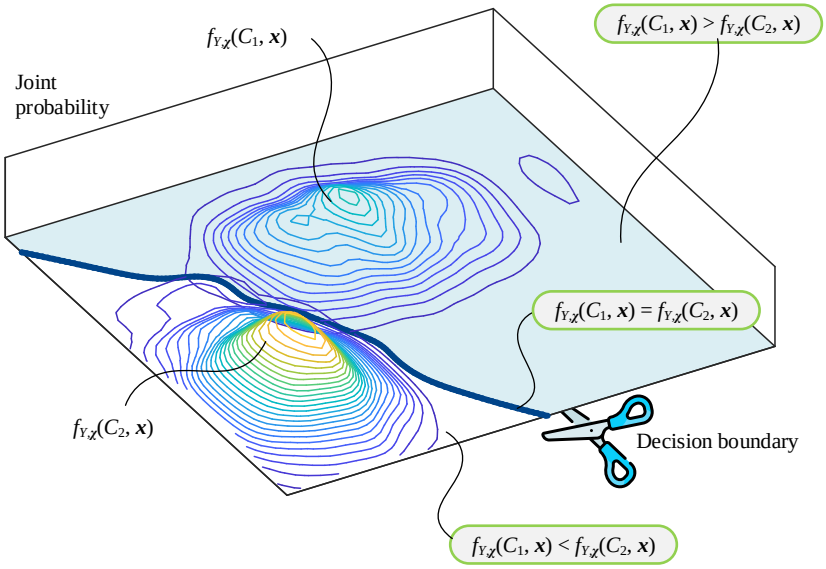


图 3. 二分类，比较联合概率大小，基于 KDE

推广：从二分类到多分类

根据前文分析，我们可以总结得到朴素贝叶斯分类优化问题——最大化后验概率：

$$\hat{y} = \arg\max_{C_k} f_{Y|X}\left(C_k \middle| \boldsymbol{x}\right) \tag{5}$$

其中，$k = 1,2,\ldots K$。

证据因子 $f_X(\boldsymbol{x})$ 不为 0 时，后验概率正比于联合概率，即：

$$\underbrace{f_{Y|X}\left(C_k \middle| \boldsymbol{x}\right)}_{\text{Posterior}} \propto \underbrace{f_{X,Y}\left(\boldsymbol{x}, C_k\right)}_{\text{Joint}} \tag{6}$$

因此，(5) 等价于：

$$\hat{y} = \arg\max_{C_k} f_{X,Y}\left(\boldsymbol{x}, C_k\right) \tag{7}$$

由于后验 ∝ 似然 × 先验，最大化后验概率等价于最大化“似然 × 先验”。

至此，我们解决了朴素贝叶斯分类的“贝叶斯”部分，下一节讨论何谓“朴素”。

阅读这一节感到吃力的话，请大家回顾《统计至简》第 18、19 章内容。

4.2 朴素贝叶斯的“朴素”之处

朴素贝叶斯分类，何以谓之“朴素”？

本章副标题已经给出答案——假设特征之间条件独立 (conditional independence)！

独立指两个事件 A、B 之间没有任何关联，即 A 的发生与 B 的发生互不影响，可以表示为 $\Pr(A \cap B) = \Pr(A) \times \Pr(B)$。条件独立指在已知某些条件下，两个事件 A、B 之间没有任何关联，比如给定条件 C 下，A 的发生与 B 的发生互不影响，可以表示为 $\Pr(A \cap B \mid C) = \Pr(A \mid C) \times \Pr(B \mid C)$。

特征独立

对于 x_1 和 x_2 两特征情况，“特征独立”指的是：

$$f_{\chi}(\boldsymbol{x}) = f_{X1,X2}(x_1, x_2) = f_{X1}(x_1) f_{X2}(x_2) \tag{8}$$

$f_{X1}(x_1)$ 和 $f_{X2}(x_2)$ 为两个特征上的边际概率密度函数，如图 4 所示。

推广到 D 个特征情况，“特征独立”指的是：

$$f_{\chi}(\boldsymbol{x}) = f_{X1}(x_1) f_{X2}(x_2) \ldots f_{X_D}(x_D) = \prod_{j=1}^{D} f_{X_j}(x_j) \tag{9}$$

图 4 中等高线为“特征独立”条件下，证据因子 $f_{\chi}(\boldsymbol{x})$ 概率密度分布。不知道大家看到这幅图时，是否想到《矩阵力量》中讲过的向量张量积。

$f_{X1}(x_1)$ 和 $f_{X2}(x_2)$ 描述 X_1 和 X_2 两特征的分布还比较准确。但是，假设特征独立，用 (8) 估算证据因子概率密度 $f_{\chi}(\boldsymbol{x})$ 时，偏差很大。比较图 4 等高线和散点分布就可以看出来。

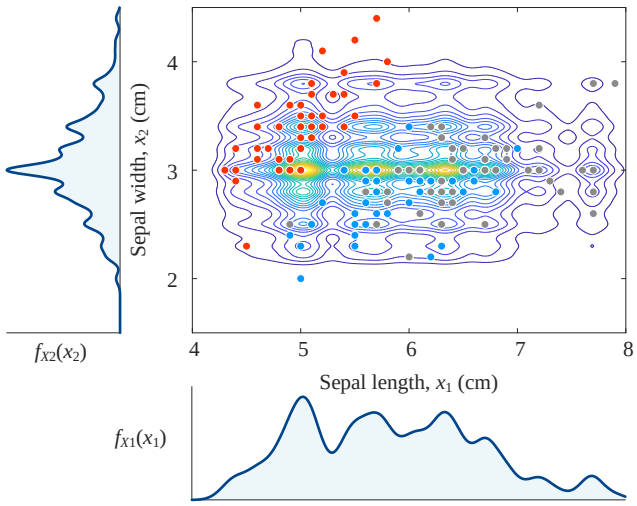


图 4. “特征独立”条件下，证据因子 $f_{\chi}(\boldsymbol{x})$ 概率密度，基于 KDE

特征条件独立

对于两特征 ($D = 2$)、两分类 ($K = 2$) 情况，“特征条件独立”指的是：

$$\begin{cases} \underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_1\right)}_{\text{Likelihood}} = \underbrace{f_{X1|Y}\left(x_1|C_1\right) f_{X2|Y}\left(x_2|C_1\right)}_{\text{Conditional independence}} \\ \underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_2\right)}_{\text{Likelihood}} = \underbrace{f_{X1|Y}\left(x_1|C_2\right) f_{X2|Y}\left(x_2|C_2\right)}_{\text{Conditional independence}} \end{cases} \tag{10}$$

推广到 D 个特征情况，“特征条件独立”假设下，似然概率为：

$$\underbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}_{\text{Likelihood}} = f_{X1|Y}\left(x_1|C_k\right) f_{X2|Y}\left(x_2|C_k\right) \ldots f_{XD|Y}\left(x_D|C_k\right) = \prod_{j=1}^{D} f_{Xj|Y}\left(x_j|C_k\right) \tag{11}$$

⚠ 请大家格外注意，A 和 B 相互独立，无法推导得到 A 和 B 条件独立。而 A 和 B 条件独立，也无法推导得到 A 和 B 相互独立。《统计至简》第 3 章专门介绍过条件独立。

特征条件独立 → 联合概率

根据贝叶斯定理，联合概率为：

$$\underbrace{f_{\boldsymbol{X},Y}\left(\boldsymbol{x},C_k\right)}_{\text{Joint}} = \underbrace{p_Y\left(C_k\right)}_{\text{Prior}} \underbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}_{\text{Likelihood}} \tag{12}$$

⚠ 注意，先验概率 $p_Y(C_k)$ 为概率质量函数 (probability mass function, PMF)。这是因为 Y 是离散随机变量，Y 的取值为分类标签 C_1、C_2 … C_K，并非连续。

将 (11) 代入 (12)，可以得到“特征条件独立”条件下，联合概率为：

$$\underbrace{f_{\boldsymbol{X},Y}\left(\boldsymbol{x},C_k\right)}_{\text{Joint}} = \underbrace{p_Y\left(C_k\right)}_{\text{Prior}} \underbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}_{\text{Likelihood}} = \underbrace{p_Y\left(C_k\right)}_{\text{Prior}} \underbrace{\prod_{j=1}^{D} f_{Xj|Y}\left(x_j|C_k\right)}_{\text{Conditional independence}} \tag{13}$$

“朴素”贝叶斯优化问题

有了本节分析，基于 (13)，(7) 所示朴素贝叶斯优化问题可以写成：

$$\hat{y} = \underset{C_k}{\arg\max}\, p_Y\left(C_k\right) \prod_{j=1}^{D} f_{Xj|Y}\left(x_j|C_k\right) \tag{14}$$

这样，我们便解决了“朴素贝叶斯”中的“朴素”部分！

朴素贝叶斯分类流程

图 5 所示为朴素贝叶斯分类流程图，图中散点数据为鸢尾花前两个特征——花萼长度、花萼宽度。

图 5 中概率密度基于核密度估计 (Kernel Density Estimation, KDE)。《统计至简》第 17 章介绍过 KDE 方法。

请大家现在快速浏览这幅图，完成本章学习之后，再回过头来再仔细观察图 5 细节。

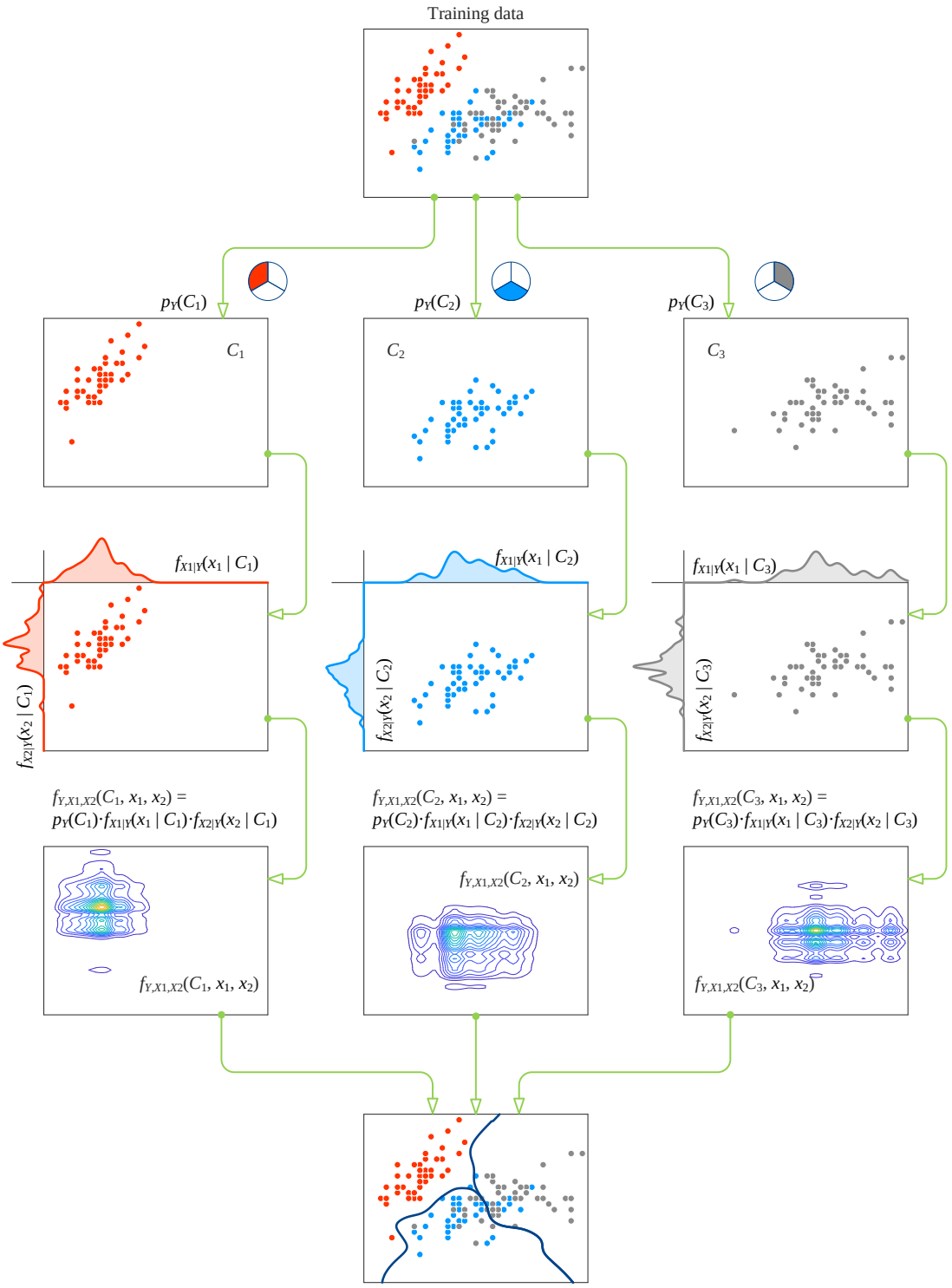


图 5. 朴素贝叶斯分类过程，基于 KDE

4.3 先验概率

先验概率计算最为简单。鸢尾花数据 C_1、C_2 和 C_3 三类对应的先验概率为：

$$p_Y(C_1)=\frac{\text{count}(C_1)}{\text{count}(\Omega)},\quad p_Y(C_2)=\frac{\text{count}(C_2)}{\text{count}(\Omega)},\quad p_Y(C_3)=\frac{\text{count}(C_3)}{\text{count}(\Omega)}, \tag{15}$$

鸢尾花数据共有 150 个数据点，count(Ω) = 150；而 C_1、C_2 和 C_3 三类各占 50，因此，

$$p_Y(C_1)=p_Y(C_2)=p_Y(C_3)=\frac{50}{150}=\frac{1}{3} \tag{16}$$

图 6 所示为鸢尾花数据先验概率结果。

⚠ 注意，一般情况各类数据先验概率并不相等。

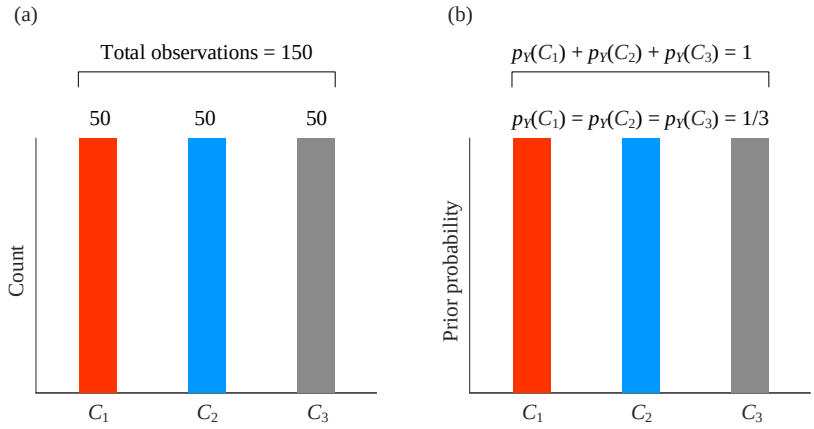


图 6. 鸢尾花数据先验概率

4.4 似然概率

根据前三节所述，朴素贝叶斯分类算法核心在于三方面：(1) 贝叶斯定理建立似然概率、先验概率和后验概率三者联系；(2) 估算似然概率时，假设特征之间条件独立；(3) 优化目标为，最大化后验概率，或最大化联合概率 (似然 × 先验)。

根据 (13)，想要获得联合概率，就先需要利用“特征条件独立”计算得到似然概率。

下面，我们利用花萼长度 (x_1) 和花萼宽度 (x_2) 两个特征 (D = 2)，解决鸢尾花三分类 (K = 3, C_1、C_2 和 C_3) 问题。本节先讨论如何获得 C_1、C_2 和 C_3 似然概率密度。

C_1 的似然概率

图 7 所示为求解似然概率密度 $f_{\boldsymbol{X}|Y}(\boldsymbol{x} \mid C_1)$ 的过程。只考虑 setosa (C_1, $y = 0$) 样本数据点 ●，分别估算两个特征的条件边际分布 $f_{X1|Y}(x_1 \mid C_1)$ 和 $f_{X2|Y}(x_2 \mid C_1)$。

需要特别注意的是，图 7 中，$f_{X1|Y}(x_1 \mid C_1)$ 和 $f_{X2|Y}(x_2 \mid C_1)$ 曲线覆盖阴影区域面积均为 1。

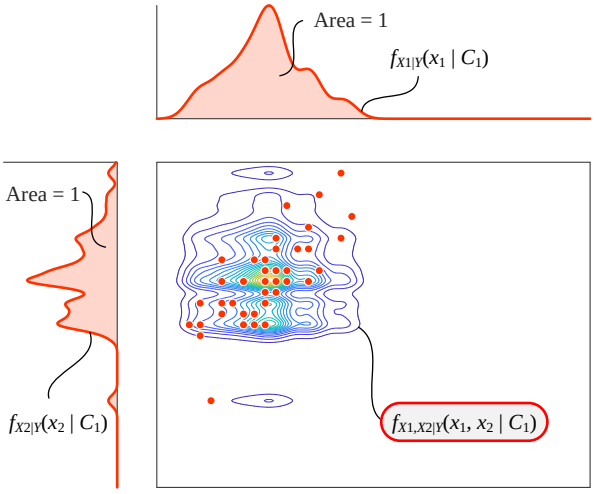


图 7. 分类 C_1 样本数据，鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率密度 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$

根据 (11)，似然概率 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 可以通过下式计算得到：

$$f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_1\right) = f_{X1,X2|Y}\left(x_1, x_2|C_1\right) = f_{X1|Y}\left(x_1|C_1\right) \cdot f_{X2|Y}\left(x_2|C_1\right) \tag{17}$$

得到的 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 结果对应图 7 中等高线。而 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 曲面和水平面围成几何体的体积为 1，也就是说，$f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 在 $\mathbb{R}^2$ 的二重积分结果为 1，这个值是概率。而 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 的“偏积分”为条件边际分布 $f_{X1|Y}(x_1 \mid C_1)$ 或 $f_{X2|Y}(x_2 \mid C_1)$，它们还是概率密度，并非概率值。

《数学要素》第 14 章聊过“偏求和”，第 18 章聊过“偏积分”，建议大家回顾。

本章估算条件边际分布时用的是高斯核密度估计方法。下一章则采用二元高斯分布 (Gaussian distribution) 来估算条件边际分布。因此，下一章的分类算法被称作，高斯朴素贝叶斯分类 (Gaussian Naïve Bayes classification)。

C_2 和 C_3 的似然概率

类似 (17)，C_2 和 C_3 似然概率可以通过下式估算得到：

$$\begin{cases} f_{X1,X2|Y}\left(x_1, x_2|C_2\right) = f_{X1|Y}\left(x_1|C_2\right) \cdot f_{X2|Y}\left(x_2|C_2\right) \\ f_{X1,X2|Y}\left(x_1, x_2|C_3\right) = f_{X1|Y}\left(x_1|C_3\right) \cdot f_{X2|Y}\left(x_2|C_3\right) \end{cases} \tag{18}$$

图 8 和图 9 等高线分别对应似然概率密度函数 $f_{X1,X2|Y}(x_1, x_2 \mid C_2)$ 和 $f_{X1,X2|Y}(x_1, x_2 \mid C_3)$ 结果。有了上一节的先验概率和本节得到的似然概率密度，我们可以求解联合概率。

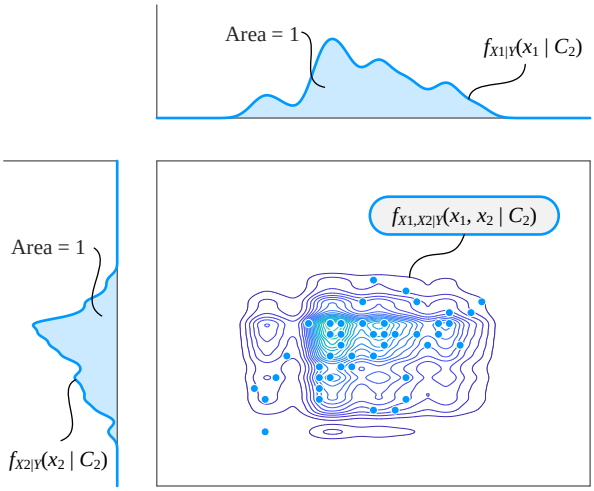


图 8. 分类 C_2 样本数据，鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率密度函数 $f_{X1,X2|Y}(x_1, x_2 \mid C_2)$

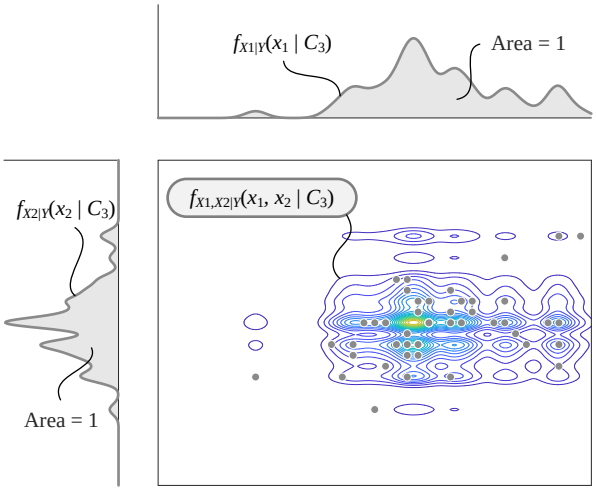


图 9. 分类 C_3 样本数据，鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率密度函数 $f_{X1,X2|Y}(x_1, x_2 \mid C_3)$

4.5 联合概率

C_1 的联合概率

根据 (13) 可以计算得到联合概率。对于鸢尾花三分类问题，假设“特征条件独立”，利用贝叶斯定理，联合概率 $f_{X1,X2,Y}(x_1, x_2, C_1)$ 可以通过下式得到：

$$\begin{aligned} \underbrace{f_{X1,X2,Y}\left(x_1,x_2,C_1\right)}_{\text{Joint}} &= \underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_1\right)}_{\text{Likelihood}}\underbrace{p_Y\left(C_1\right)}_{\text{Prior}} \\ &= \underbrace{f_{X1|Y}\left(x_1|C_1\right)\cdot f_{X2|Y}\left(x_2|C_1\right)}_{\text{Conditional independence}}\underbrace{p_Y\left(C_1\right)}_{\text{Prior}} \end{aligned} \tag{19}$$

利用 (17)，我们已经得到似然概率密度曲面 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$。(16) 给出先验概率 $p_Y(C_1)$，代入 (19) 可以求得联合概率 $f_{X1,X2,Y}(x_1, x_2, C_1)$：

$$\underbrace{f_{X1,X2,Y}\left(x_1,x_2,C_1\right)}_{\text{Joint}} = \underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_1\right)}_{\text{Likelihood}} \times \underbrace{\frac{1}{3}}_{\text{Prior}} \tag{20}$$

容易发现，先验概率 $p_Y(C_1) = 1/3$ 相当于一个缩放系数。

图 10 所示为联合概率 $f_{X1,X2,Y}(x_1, x_2, C_1)$ 概率密度曲面。图 10 的 z 轴数值为概率密度值，并非概率。

我们知道似然概率密度曲面 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$ 和水平面围成三维形状的体积为 1。而图 10 中联合概率 $f_{X1,X2,Y}(x_1, x_2, C_1)$ 和水平面围成体积为 $p_Y(C_1) = 1/3$。也就是说，$f_{X1,X2,Y}(x_1, x_2, C_1)$ 在 $\mathbb{R}^2$ 的二重积分结果为 1/3，这个值是概率值。

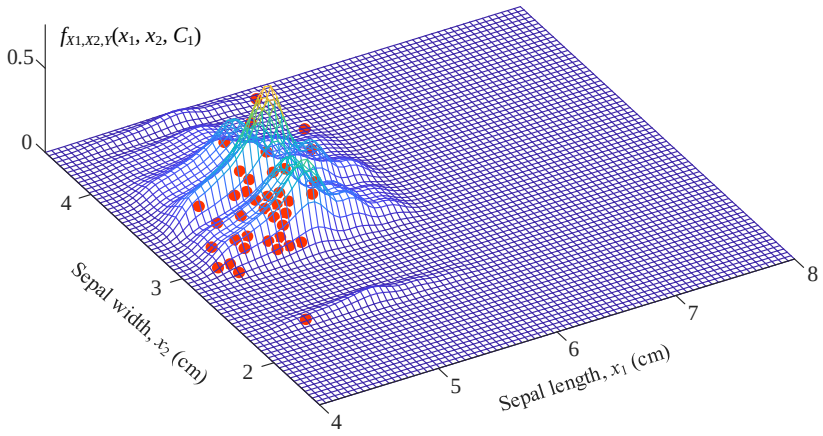


图 10. $f_{X1,X2,Y}(x_1, x_2, C_1)$ 概率密度曲面，基于 KDE

C_1 和 C_2 的联合概率

类似地，我们可以计算得到另外两个联合概率 $f_{X1,X2|Y}(x_1, x_2 \mid C_2)$ 和 $f_{X1,X2|Y}(x_1, x_2 \mid C_3)$，对应曲面分别如图 11 和图 12 所示。

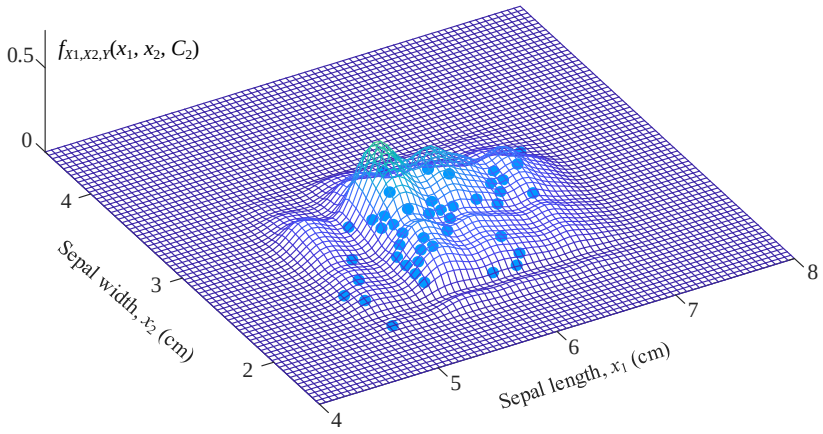


图 11. $f_{X1,X2,Y}(x_1, x_2, C_2)$ 概率密度曲面，基于 KDE

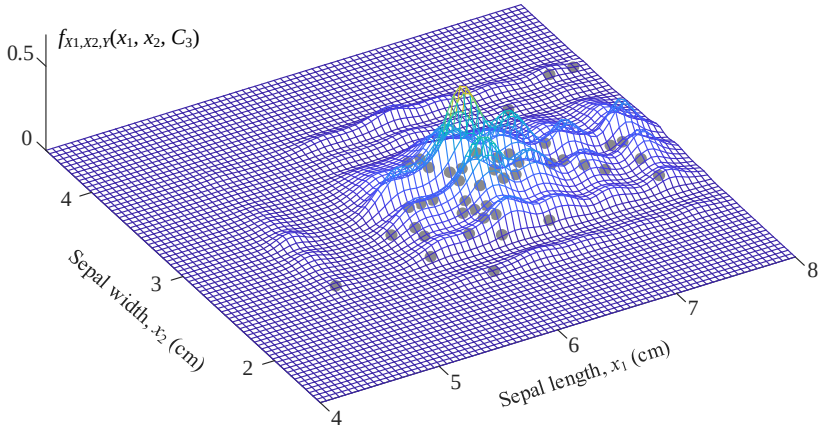


图 12. $f_{X1,X2,Y}(x_1, x_2, C_3)$ 概率密度曲面，基于 KDE

分类

至此，根据 (7) 我们可以比较上述三个联合概率密度曲面高度，从而获得决策边界。图 13 所示为采用朴素贝叶斯分类算法，基于 KDE 估算条件边际概率密度，得到的鸢尾花三分类边界。

请大家注意，目前 Python 的 Scikit-learn 工具包暂时不支持基于 KDE 的朴素贝叶斯分类。Scikit-learn 提供基于高斯分布的朴素贝叶斯分类器，这是下一章要介绍的内容。另外，KDE 朴素贝叶斯分类得到的决策边界不存在解析解。而高斯朴素贝叶斯分类得到的决策边界存在解析解。

利用 (7) 思想——比较联合概率大小——我们已经完成分类问题。但是，一般情况我们都会求出证据因子，并求得后验概率。如前文所述，后验概率又叫成员值，可以直接表达分类可能性百分比，便于可视化和解释结果。根据贝叶斯公式，要想得到后验概率，需要求得证据因子，这是下一节介绍的内容。

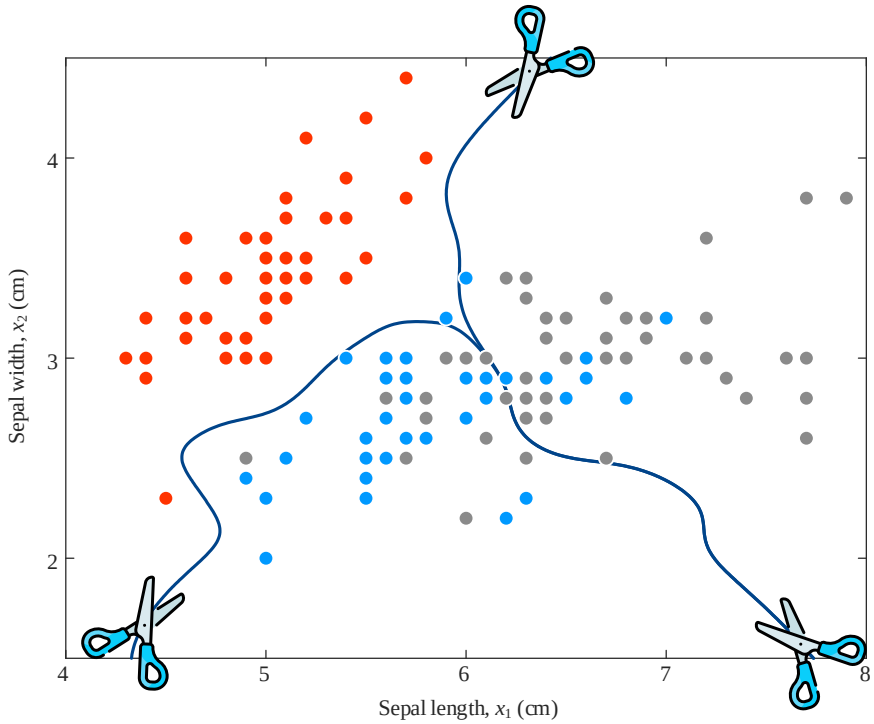


图 13. 朴素贝叶斯决策边界，基于核密度估计 KDE

4.6 证据因子

假设特征条件独立，利用全概率定理和 (13)，证据因子 $f_X(\boldsymbol{x})$ 概率密度可以通过下式计算得到：

$$\underbrace{f_X(\boldsymbol{x})}_{\text{Evidence}} = \sum_{k=1}^{K}\left\{\underbrace{f_{X,Y}(\boldsymbol{x},C_k)}_{\text{Joint}}\right\} = \sum_{k=1}^{K}\left\{\underbrace{p_Y(C_k)}_{\text{Prior}}\underbrace{f_{X|Y}(\boldsymbol{x}|C_k)}_{\text{Likelihood}}\right\} = \sum_{k=1}^{K}\left\{\underbrace{p_Y(C_k)}_{\text{Prior}}\underbrace{\prod_{j=1}^{D}f_{X_j|Y}(x_j|C_k)}_{\text{Conditional independence}}\right\} \tag{21}$$

两特征、三分类问题

当 $K = 3$ 时，对于两特征分类问题，证据因子 $f_{X1,X2}(x_1, x_2)$ 可以利用下式求得：

$$\begin{aligned}\underbrace{f_{X1,X2}(x_1,x_2)}_{\text{Evidence}} &= \underbrace{f_{X1,X2,Y}(x_1,x_2,C_1)}_{\text{Joint}} + \underbrace{f_{X1,X2,Y}(x_1,x_2,C_2)}_{\text{Joint}} + \underbrace{f_{X1,X2,Y}(x_1,x_2,C_3)}_{\text{Joint}} \\ &= \underbrace{p_Y(C_1)}_{\text{Prior}}\underbrace{f_{X1,X2|Y}(x_1,x_2|C_1)}_{\text{Likelihood}} + \underbrace{p_Y(C_2)}_{\text{Prior}}\underbrace{f_{X1,X2|Y}(x_1,x_2|C_2)}_{\text{Likelihood}} + \underbrace{p_Y(C_3)}_{\text{Prior}}\underbrace{f_{X1,X2|Y}(x_1,x_2|C_3)}_{\text{Likelihood}}\end{aligned} \tag{22}$$

这步计算很容易理解，对于鸢尾花数据，上一节得到的三个联合概率曲面 (图 10 ~ 图 12) 叠加便得到证据因子 $f_{X1,X2}(x_1, x_2)$ 概率密度曲面。图 14 所示为运算过程。图 14 实际上也是一种概率密度估算的方法。

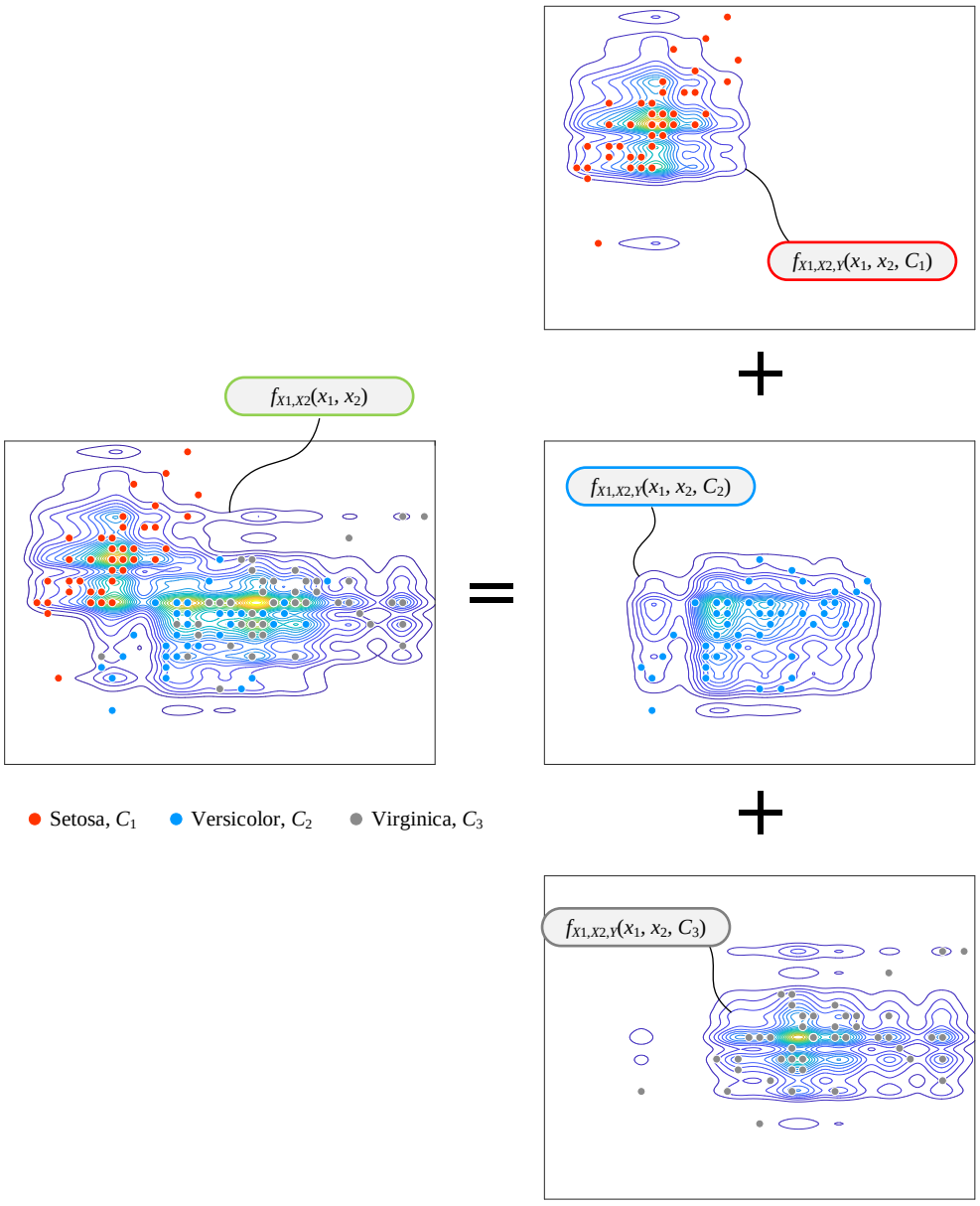


图 14. 估算证据因子概率密度，基于 KDE

概率密度估算

图 15 所示为利用“特征条件独立”构造得到的证据因子 $f_{X1,X2}(x_1, x_2)$ 概率密度曲面。$f_{X1,X2}(x_1, x_2)$ 概率密度曲面和水平面构成的几何形体体积为 1。

图 4 所示为假设“特征独立”条估算的证据因子概率密度曲面。前文提过，图 4 这个曲面没有准确捕捉样本数据分布特点；然而，图 15 曲面较为准确描述样本数据分布。

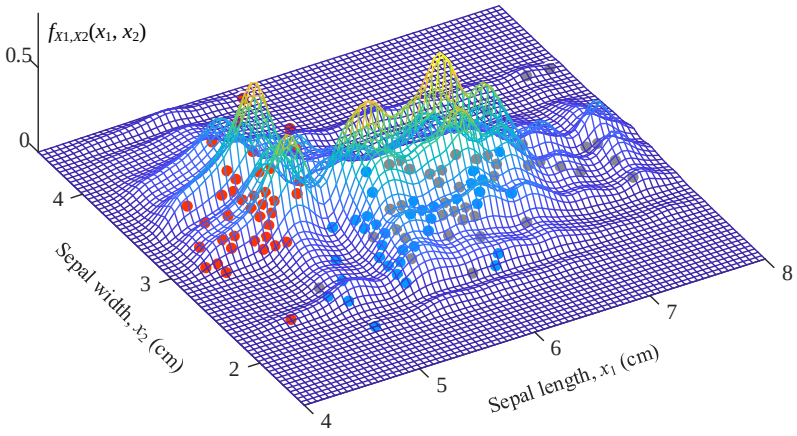


图 15. 估算得到的概率密度曲面，特征条件独立，基于 KDE

4.7 后验概率：成员值

有了前两节计算得到联合概率和证据因子，本节我们计算后验概率。

当 $K = 3$ 时，如果证据因子 $f_{X1,X2}(x_1, x_2)$ 不为 0，后验概率 $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 可以通过下式得到：

$$\underbrace{f_{Y|X1,X2}\left(C_1|x_1,x_2\right)}_{\text{Posterior}} = \frac{\overbrace{f_{X1,X2,Y}\left(x_1,x_2,C_1\right)}^{\text{Joint}}}{\underbrace{f_{X1,X2}\left(x_1,x_2\right)}_{\text{Evidence}}} \tag{23}$$

白话来讲，后验概率 $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 的含义是，给定 (x_1, x_2) 的具体值，分类标签为 C_1 的可能性多大？所以，$f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 并不是概率密度，$f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 是概率。

图 16 所示为后验概率 $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 曲面，容易发现曲面高度在 $[0, 1]$ 之间。

同理，可以计算得到另外两个后验概率 $f_{Y|X1,X2}(C_2 \mid x_1, x_2)$ 和 $f_{Y|X1,X2}(C_3 \mid x_1, x_2)$。比较三个后验概率曲面高度关系，可以得到和图 13 完全一致的决策边界。

对于三分类问题，后验概率 (成员值) 存在以下关系：

$$\underbrace{f_{Y|X1,X2}\left(C_1|x_1,x_2\right)}_{\text{Posterior}} + \underbrace{f_{Y|X1,X2}\left(C_2|x_1,x_2\right)}_{\text{Posterior}} + \underbrace{f_{Y|X1,X2}\left(C_3|x_1,x_2\right)}_{\text{Posterior}} = 1 \tag{24}$$

白话说，给定平面上任意一点 (x_1, x_2)，它的分类可能性只有三个——C_1、C_2、C_3。因此，上式中，三个条件概率之和为 1。

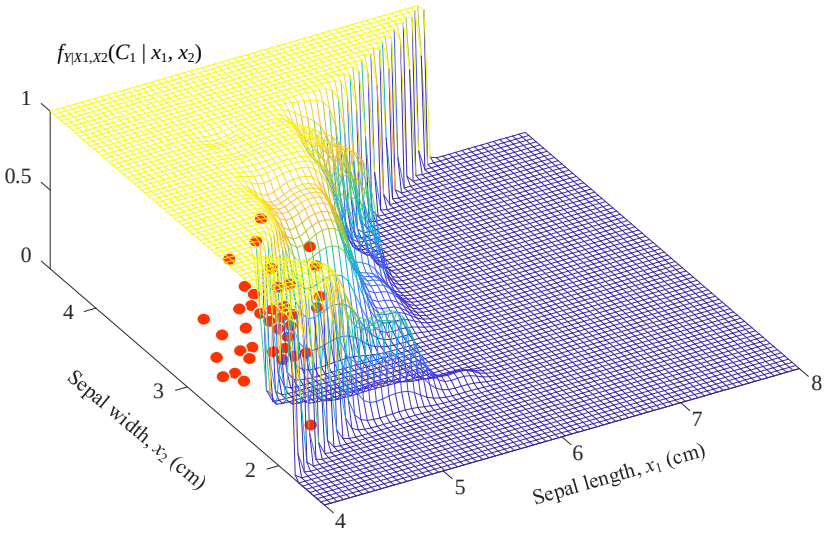


图 16. $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 后验概率曲面，基于 KDE

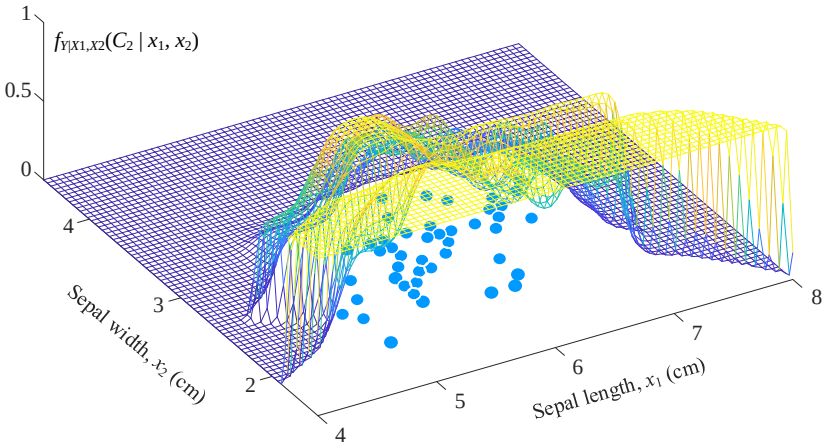


图 17. $f_{Y|X1,X2}(C_2 \mid x_1, x_2)$ 后验概率曲面，基于 KDE

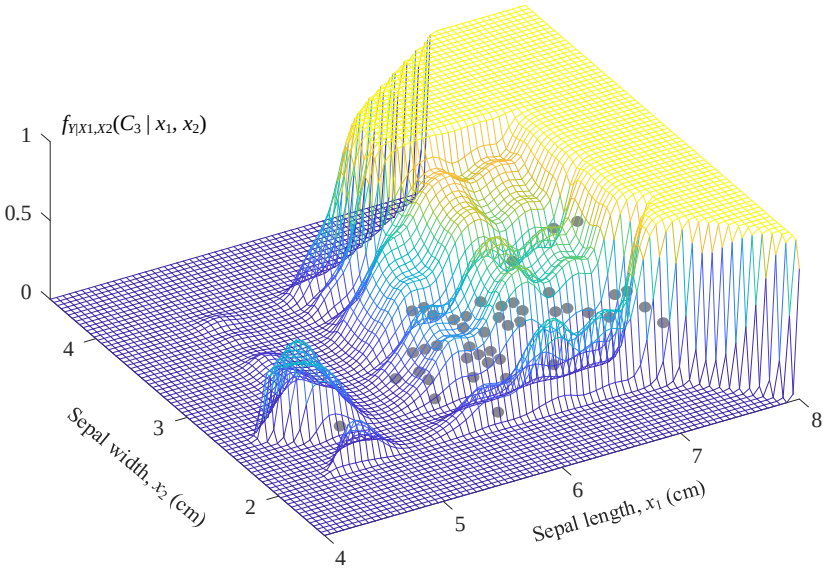
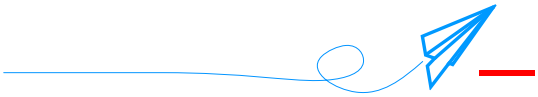


图 18. $f_{Y|X1,X2}(C_3 \mid x_1, x_2)$ 后验概率曲面，基于 KDE



本章最后请大家特别注意以下几点：

- 贝叶斯定理和全概率定理是朴素贝叶斯分类器的理论基础；
- 朴素贝叶斯分类器的“朴素”来自假设“特征条件独立”；
- 后验 $\propto$ 似然 $\times$ 先验；
- 比较联合概率 (似然 $\times$ 先验) 大小，可以预测分类；
- 假设“特征条件独立”，联合概率叠加得到证据因子，这是一种概率密度估算方法；
- 后验概率，本身就是概率值，取值范围在 $[0, 1]$ 之间；
- 比较后验概率大小，同样可以预测分类。

下一章介绍高斯朴素贝叶斯。下一章采用和本章几乎一致的内容安排，请大家对照阅读。这两章共用一个思维导图。

5 高斯朴素贝叶斯

Gaussian Naïve Bayes

假设特征之间条件独立，且条件概率分布服从高斯分布

给我带来极大的喜悦的是，求知的过程，而不是占有知识。

It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.

—— 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

- `matplotlib.pyplot.contour()` 绘制等高线线图
- `matplotlib.pyplot.contourf()` 绘制填充等高线图
- `matplotlib.pyplot.scatter()` 绘制散点图
- `numpy.array()` 创建 array 数据类型
- `numpy.c_()` 按列叠加两个矩阵
- `numpy.linspace()`产生连续均匀向量数值
- `numpy.meshgrid()` 创建网格化数据
- `numpy.r_()` 按行叠加两个矩阵
- `numpy.ravel()` 将矩阵扁平化
- `seaborn.scatterplot()` 绘制散点图
- `sklearn.datasets.load_iris()` 加载鸢尾花数据集
- `sklearn.naive_bayes.GaussianNB` 高斯朴素贝叶斯分类算法函数

5.1 高斯你好

高斯的足迹几乎踏遍数学的每个角落，他所到之处都留下了自己的名字。哪怕在机器学习算法中，“高斯”这个金字招牌也反复出现。比如，本书提到几种算法：

- **高斯朴素贝叶斯** (Gaussian Naive Bayes)
- **高斯判别分析** (Gaussian discriminant analysis)
- **高斯过程** (Gaussian process)
- **高斯混合模型** (Gaussian mixture model)

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，都利用了**高斯分布** (Gaussian distribution)。

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

原理

上一章介绍了朴素贝叶斯分类，这种分类算法思路核心在于如下三点：

- 贝叶斯定理；
- 假设特征之间条件独立 (朴素之处)；
- 优化目标为最大化后验概率，或最大化联合概率 (似然 × 先验)。

上一章在估算单一特征条件边际分布时，采用高斯核密度估计 KDE。而本章介绍的**高斯朴素贝叶斯** (Gaussian Naive Bayes) 最大不同在于，采用高斯分布估计单一特征条件边际分布。

最大化后验概率

朴素贝叶斯分类的优化目标可以是——最大化后验概率。对于二分类问题，直接比较 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 两个后验概率大小，就可以预测分类。

图 1 所示为基于高斯分布得到的 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 两个后验概率曲面。比较上一章基于 KDE 的后验概率曲面，可以发现高斯后验概率曲面非常平滑。图 1 中深蓝色曲线就是决策边界。这条决策边界实际上是二次曲线。

这一点，我们将会在下一章讲解**高斯判别分析** (Gaussian Discriminant Analysis, GDA) 时深入介绍。

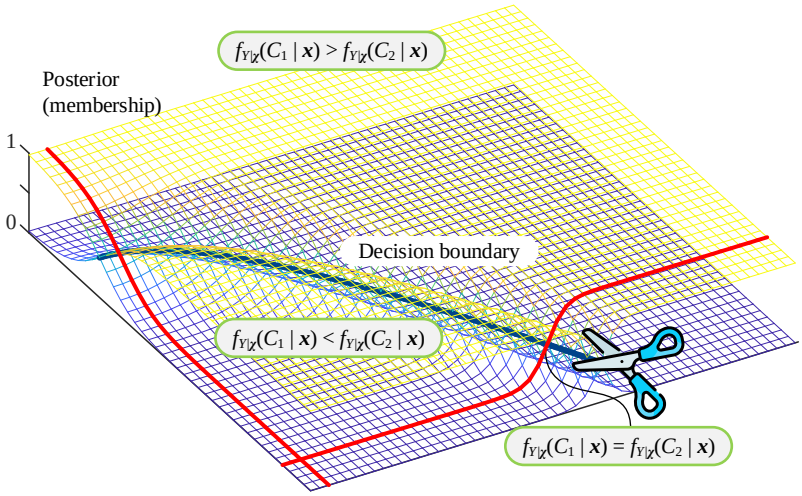


图 1. 二分类，比较后验概率大小，基于高斯分布

最大化联合概率

上一章提到，朴素贝叶斯分类的优化目标同样可以是——最大化联合概率。原因是，联合概率正比于后验概率。图 2 所示为，二分类问题中，比较联合概率 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 两个曲面高度，可以获得相同的决策边界。

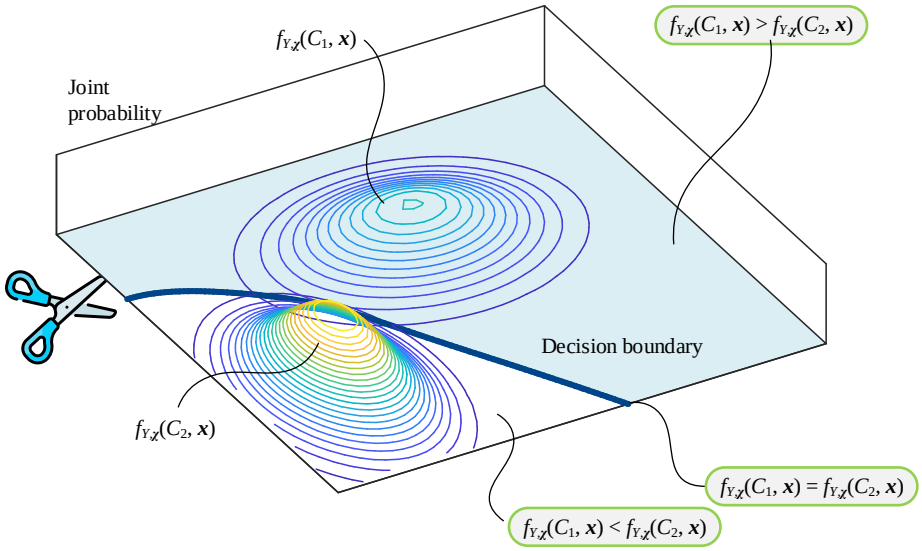


图 2. 二分类，比较联合概率大小，基于高斯分布

流程

图 3 所示为高斯朴素贝叶斯分类流程图。这一流程和上一章介绍的朴素贝叶斯分类流程完全一致。前文已经指出，高斯朴素贝叶斯分类器的特点是，估算特征条件边际分布时，高斯朴素贝叶斯分类采用高斯分布。

为方便大家学习，本章采用和上一章几乎一样结构。建议大家阅读本章时，平行对比上一章，对照边际分布曲线变化趋势，比较各种概率曲面特征，特别是对比决策边界形态。此外，本章帮助读者回顾高斯分布，让大家了解到在机器学习算法中如何引入高斯分布，以及明白高斯分布对决策边界形态有怎样的影响。

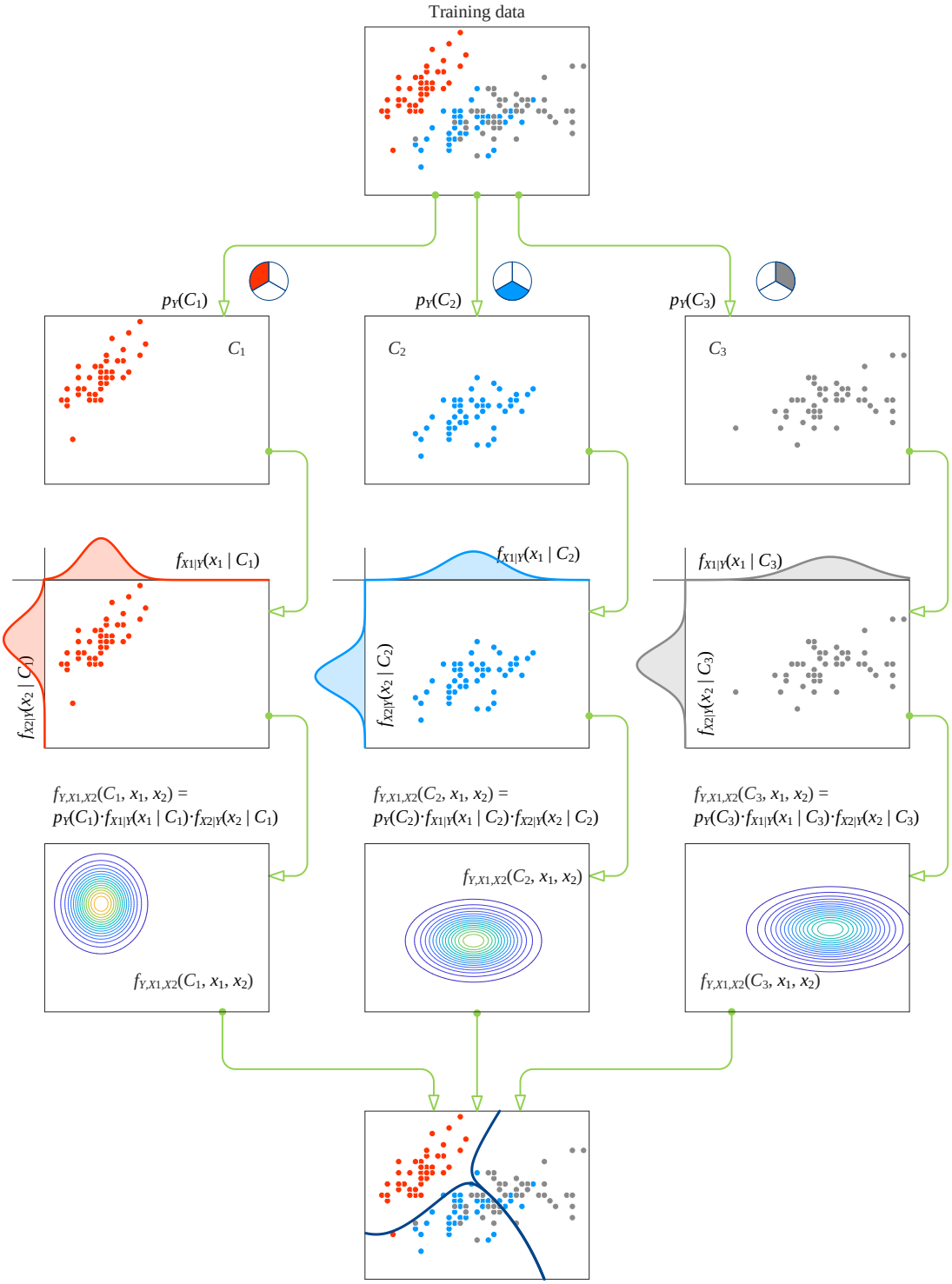


图 3. 高斯朴素贝叶斯分类过程

5.2 似然概率

朴素贝叶斯分类算法在估算似然概率时，假设特征之间条件独立：

$$\underbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}_{\text{Likelihood}}=\prod_{j=1}^{D}f_{X_j|Y}\left(x_j|C_k\right) \tag{1}$$

比如，下式计算 C_1 似然概率密度：

$$\underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_1\right)}_{\text{Likelihood}}=\underbrace{f_{X1|Y}\left(x_1|C_1\right)f_{X2|Y}\left(x_2|C_1\right)}_{\text{Conditional independence}} \tag{2}$$

引入高斯分布

高斯朴素贝叶斯中条件边际分布采用的是高斯分布估计。比如，上式中的 $f_{X1|Y}(x_1 \mid C_1)$ 和 $f_{X2|Y}(x_2 \mid C_1)$ 可以写成：

$$\begin{cases} f_{X1|Y}\left(x_1|C_1\right)=\dfrac{1}{\sqrt{2\pi}\sigma_{1|C1}}\exp\left(-\dfrac{1}{2}\left(\dfrac{x_1-\mu_{1|C1}}{\sigma_{1|C1}}\right)^2\right) \\ f_{X2|Y}\left(x_2|C_1\right)=\dfrac{1}{\sqrt{2\pi}\sigma_{2|C1}}\exp\left(-\dfrac{1}{2}\left(\dfrac{x_2-\mu_{2|C1}}{\sigma_{2|C1}}\right)^2\right) \end{cases} \tag{3}$$

对于鸢尾花数据，$\mu_{1\,|\,C1}$ 为标签为 C_1 数据在花萼长度 x_1 特征上的均值，$\sigma_{1\,|\,C1}$ 为 C_1 数据在 x_1 特征上标准差；$\mu_{2\,|\,C1}$ 为标签为 C_1 数据在花萼宽度 x_2 特征上的均值，$\sigma_{2\,|\,C1}$ 为 C_1 数据在 x_2 特征上标准差。

图 4 中给出 $f_{X1|Y}(x_1 \mid C_1)$ 和 $f_{X2|Y}(x_2 \mid C_1)$ 两个概率密度函数曲线，以及 $\mu_{1\,|\,C1}$ 和 $\mu_{2\,|\,C1}$ 所在位置。

将 (3) 代入 (2)，可以得到 $f_{X1,X2|Y}(x_1, x_2 \mid C_1)$：

$$\begin{aligned} f_{X1,X2|Y}\left(x_1,x_2|C_1\right)\cdot &= f_{X1|Y}\left(x_1|C_1\right)\cdot f_{X2|Y}\left(x_2|C_1\right) \\ &= \frac{\exp\left(-\dfrac{1}{2}\left(\dfrac{x_1-\mu_{1|C1}}{\sigma_{1|C1}}\right)^2\right)}{\sqrt{2\pi}\sigma_{1|C1}}\times\frac{\exp\left(-\dfrac{1}{2}\left(\dfrac{x_2-\mu_{2|C1}}{\sigma_{2|C1}}\right)^2\right)}{\sqrt{2\pi}\sigma_{2}} \\ &= \frac{\exp\left(-\dfrac{1}{2}\left(\dfrac{\left(x_1-\mu_{1|C1}\right)^2}{\sigma_{1|C1}^2}+\dfrac{\left(x_2-\mu_{2|C1}\right)^2}{\sigma_{2|C1}^2}\right)\right)}{\left(\sqrt{2\pi}\right)^2\sigma_{1|C1}\sigma_{2|C1}} \end{aligned} \tag{4}$$

图 4 中等高线便是 $f_{X1,X2|Y}(x_1, x_2| C_1)$ 曲面等高线。

大家可能已经发现，图 4 中等高线为正椭圆！对于本节情况，正椭圆说明特征条件独立。

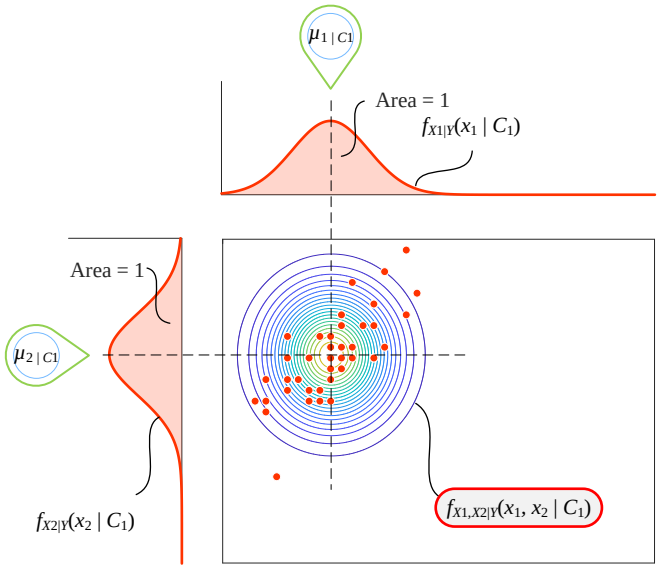


图 4. 分类 C_1 样本数据，假设鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率 $f_{X1,X2|Y}(x_1, x_2 | C_1)$，基于高斯分布

图 5 和图 6 所示为似然概率 $f_{X1,X2|Y}(x_1, x_2 | C_2)$ 和 $f_{X1,X2|Y}(x_1, x_2 | C_3)$ 结果。

⚠ 再次提醒大家注意，“特征条件独立”不同于“特征独立”。

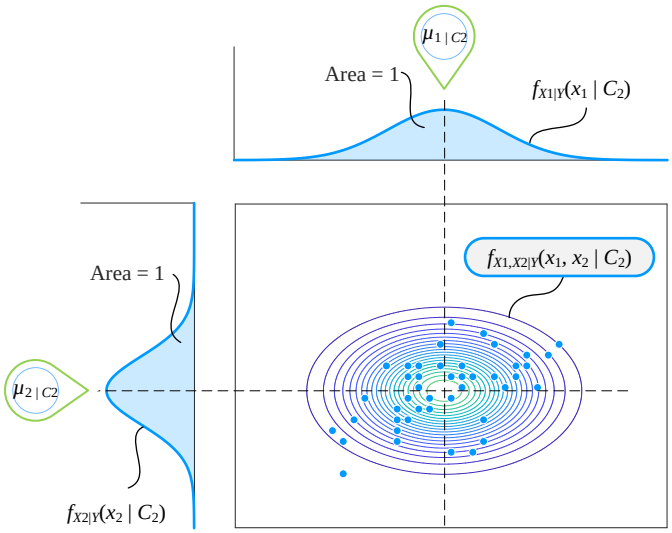


图 5. 分类 C_2 样本数据，假设鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率 $f_{X1,X2|Y}(x_1, x_2 | C_2)$，基于高斯分布

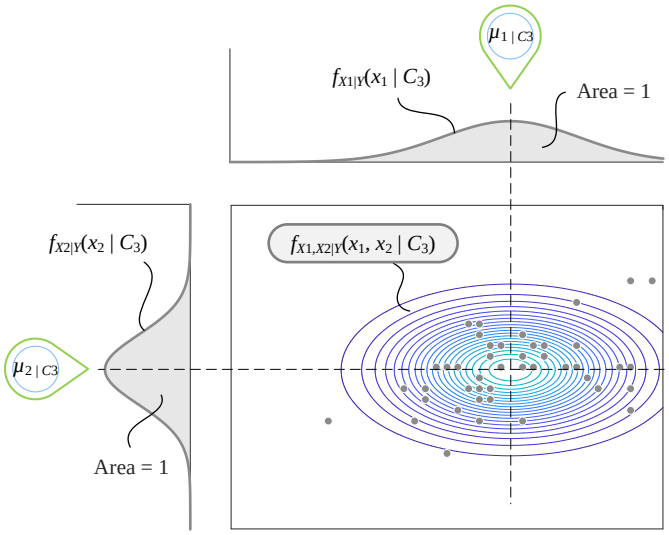


图 6. 分类 C_3 样本数据，假设鸢尾花花萼长度 x_1 和花萼宽度 x_2 条件独立，得到似然概率 $f_{X1,X2|Y}(x_1, x_2 \mid C_3)$，基于高斯分布

5.3 联合概率

这一节利用下式估算联合概率：

$$\underbrace{f_{\boldsymbol{X},Y}\left(\boldsymbol{x},C_k\right)}_{\text{Joint}}=\underbrace{p_Y\left(C_k\right)}_{\text{Prior}}\underbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}_{\text{Likelihood}}=\underbrace{p_Y\left(C_k\right)}_{\text{Prior}}\underbrace{\prod_{j=1}^{D}f_{X_j|Y}\left(x_j|C_k\right)}_{\text{Conditional independence}} \tag{5}$$

三分类问题

对于鸢尾花三分类 ($K = 3$) 问题，联合概率 $f_{X1,X2,Y}(x_1, x_2, C_k)$ ($k = 1, 2, 3$) 可以通过下式得到：

$$\underbrace{f_{X1,X2,Y}\left(x_1,x_2,C_k\right)}_{\text{Joint}}=\underbrace{f_{X1,X2|Y}\left(x_1,x_2|C_k\right)}_{\text{Likelihood}}\underbrace{p_Y\left(C_k\right)}_{\text{Prior}}=\underbrace{f_{X1|Y}\left(x_1|C_k\right)\cdot f_{X2|Y}\left(x_2|C_k\right)}_{\text{Conditional independence}}\underbrace{p_Y\left(C_k\right)}_{\text{Prior}} \tag{6}$$

再次注意，先验概率 $p_Y(C_k)$ 相当于一个缩放系数。

图 7 ~ 图 9 所示为 $f_{X1,X2,Y}(x_1, x_2, C_1)$、$f_{X1,X2,Y}(x_1, x_2, C_2)$ 和 $f_{X1,X2,Y}(x_1, x_2, C_3)$ 三个联合概率密度函数曲面。

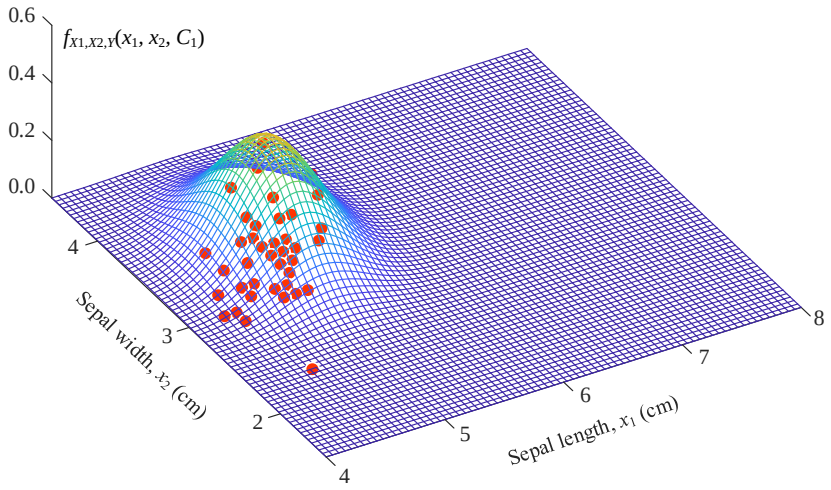


图 7. $f_{X1,X2,Y}(x_1, x_2, C_1)$ 概率密度曲面，基于高斯分布

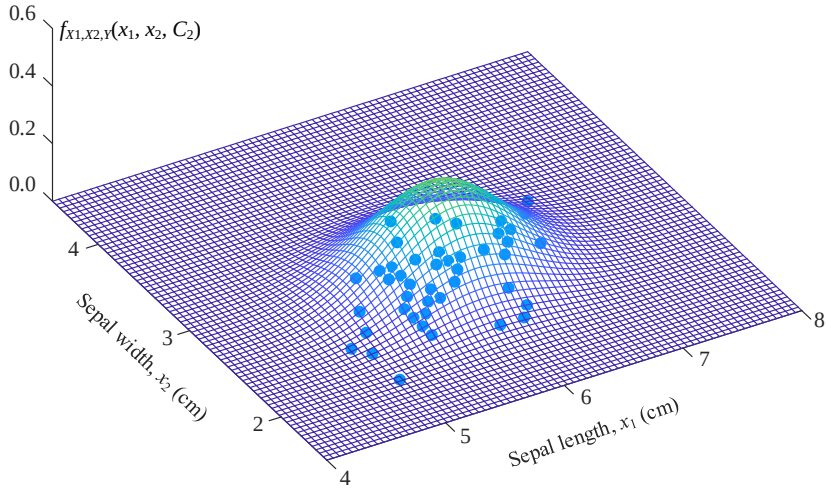


图 8. $f_{X1,X2,Y}(x_1, x_2, C_2)$ 概率密度曲面，基于高斯分布

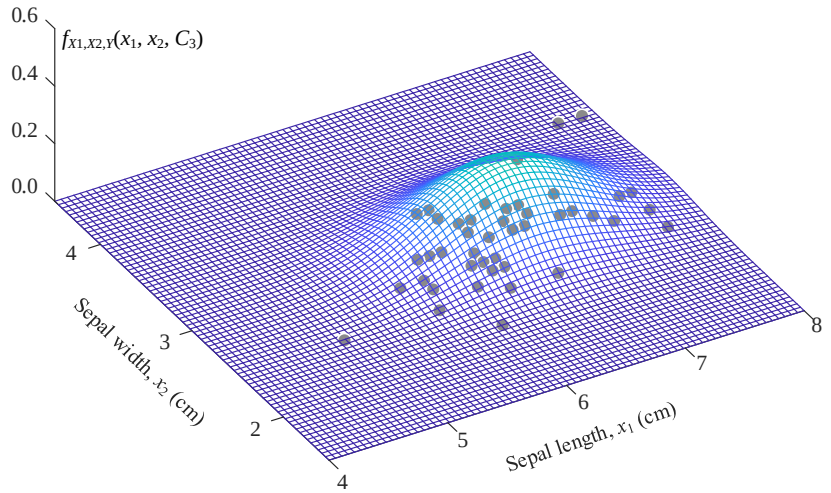


图 9. $f_{X1,X2,Y}(x_1, x_2, C_3)$ 概率密度曲面，基于高斯分布

分类依据：最大化联合概率

根据上一章介绍的高斯朴素贝叶斯优化目标之一——最大化联合概率；考虑到特征条件独立这一假设，高斯朴素贝叶斯目标函数为：

$$\hat{y} = \arg\max_{C_k} p_Y(C_k) \prod_{j=1}^{D} f_{X_j|Y}(x_j|C_k) \tag{7}$$

因此，比较图 7 ~ 图 9 三个曲面高度，可以进行鸢尾花分类预测。

sklearn 工具包高斯朴素贝叶斯分类算法的函数为 `sklearn.naive_bayes.GaussianNB`。同样，这个函数常用的 mehods 为 `fit(X, y)` 和 `predit(q)`。`fit(X, y)` 用来加载样本数据，`predit(q)` 用来预测查询点 $\boldsymbol{q}$ 的分类。

通过 sklearn 高斯朴素贝叶斯分类算法得到的分类预测和决策边界。比较上一章的决策边界，可以发现高斯朴素贝叶斯分类算法得到的决策边界，形态上更简洁。图 10 中决策边界实际上是二次曲线。也就是说，协方差矩阵为对角阵，即特征条件独立时，高斯判别分析算法得到的决策边界等同于高斯朴素贝叶斯。这一点，下一章讲详细讲解。

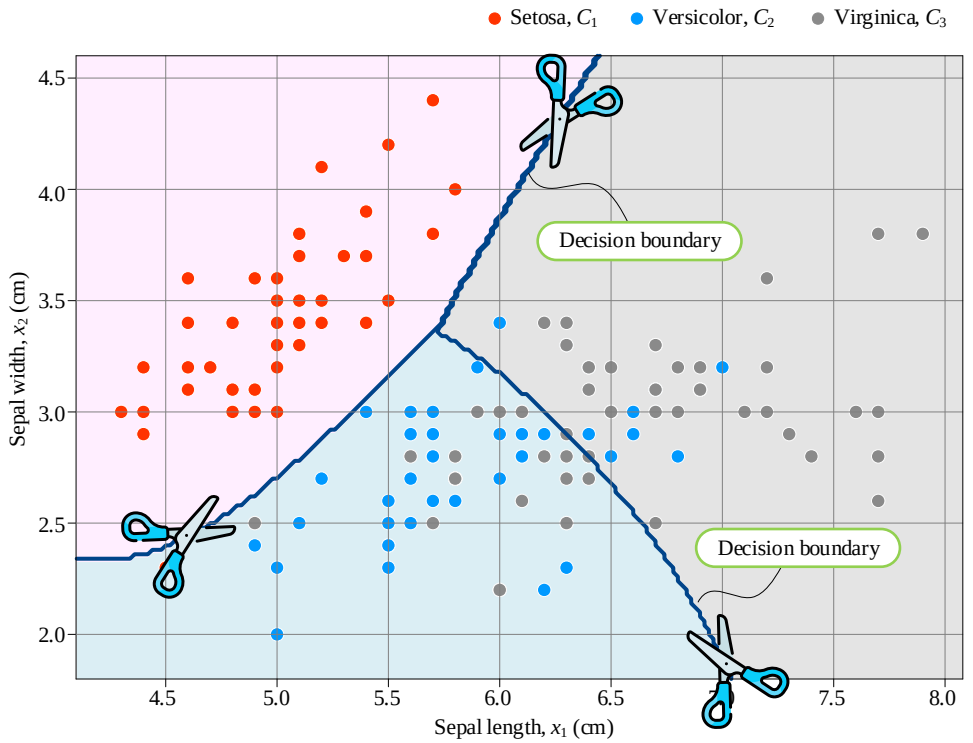


图 10. 鸢尾花分类预测，朴素贝叶斯决策边界，基于高斯分布

代码 Bk7_Ch05_01.py 利用高斯朴素贝叶斯分类鸢尾花数据集并绘制图 10。

5.4 证据因子：一种概率估算方法

根据全概率定理以及假设特征条件独立，证据因子 $f(\boldsymbol{x})$ 可以这样计算：

$$\underbrace{f_{\boldsymbol{X}}(\boldsymbol{x})}_{\text{Evidence}} = \sum_{k=1}^{K} \left\{ \underbrace{p_Y(C_k)}_{\text{Prior}} \underbrace{\prod_{j=1}^{D} f_{X_j|Y}(x_j|C_k)}_{\text{Conditional independence}} \right\} \tag{8}$$

上一章提到，上式本身是一种概率密度估算方法，具体如图 11 所示。

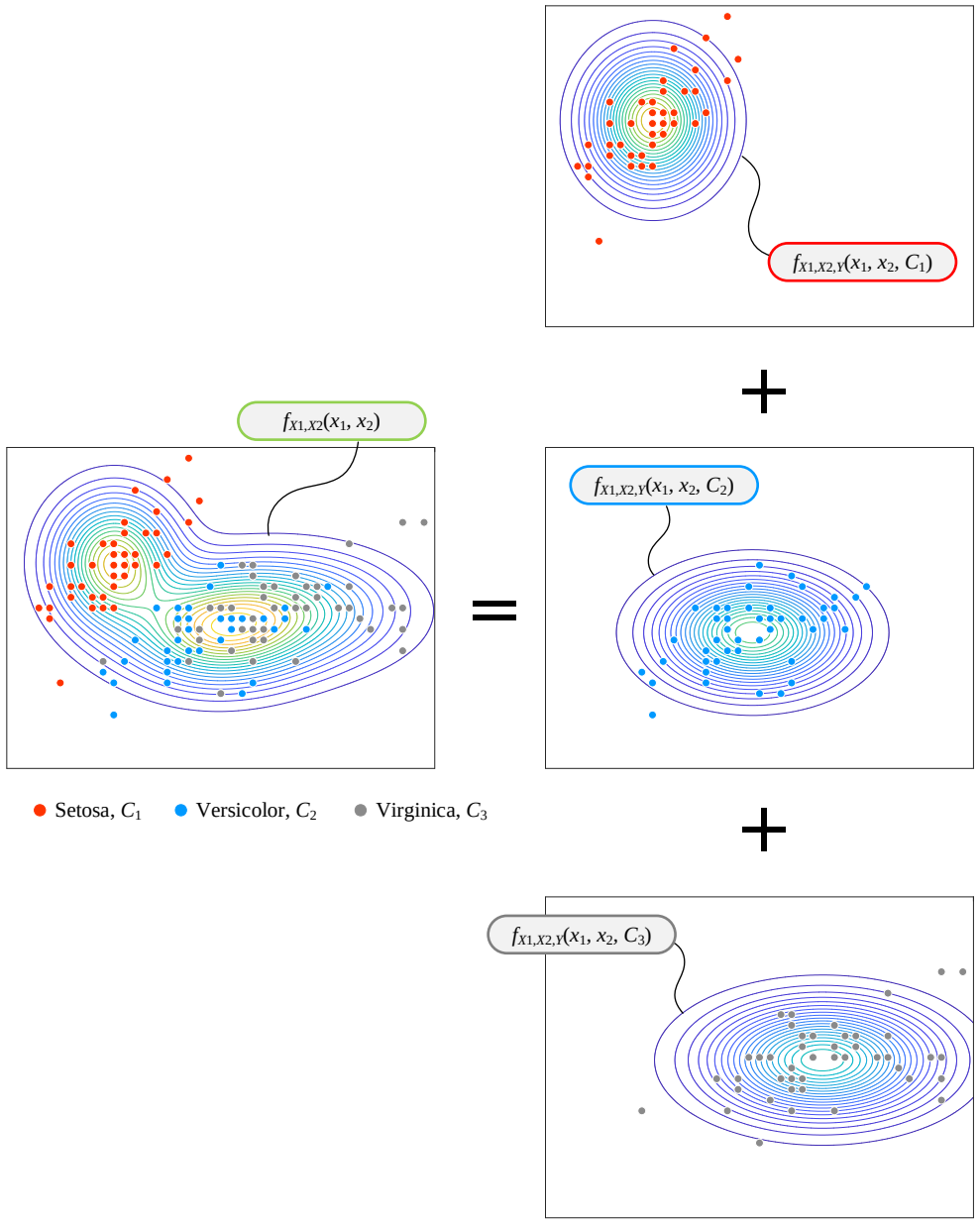


图 11. 估算证据因子概率密度，基于高斯分布

图 12 所示为估算得到的二元概率密度曲面 $f_{X1,X2}(x_1, x_2)$。注意，这个概率密度曲面主要基于以下两点：(1) 假设特征条件独立；(2) 条件边际概率通过一元高斯分布估计。

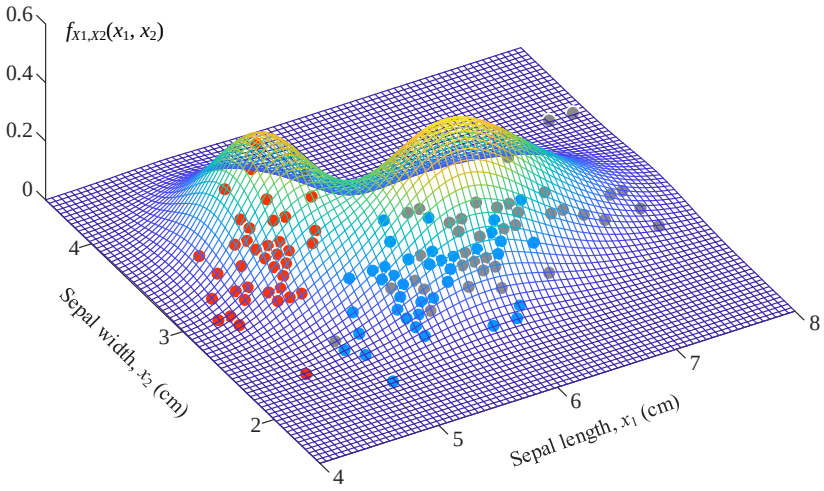


图 12. 估算得到的二元概率密度曲面，特征条件独立，基于高斯分布

5.5 后验概率：成员值

利用先验概率、似然概率和证据因子，根据贝叶斯定理计算得到后验概率，即成员值：

$$\underbrace{f_{Y|\boldsymbol{X}}\left(C_k|\boldsymbol{x}\right)}_{\text{Posterior}} = \frac{\overbrace{f_{\boldsymbol{X}|Y}\left(\boldsymbol{x}|C_k\right)}^{\text{Likelihood}}\overbrace{p_Y\left(C_k\right)}^{\text{Prior}}}{\underbrace{f_{\boldsymbol{X}}\left(\boldsymbol{x}\right)}_{\text{Evidence}}} \tag{9}$$

其中，假设分母中的证据因子不为 0。

如果假设“特征条件独立”，上式可以写成：

$$\underbrace{f_{Y|\boldsymbol{X}}\left(C_k|\boldsymbol{x}\right)}_{\text{Posterior}} = \frac{\overbrace{\prod_{j=1}^{D} f_{X_j|Y}\left(x_j|C_k\right)}^{\text{Likelihood}}\overbrace{p_Y\left(C_k\right)}^{\text{Prior}}}{\underbrace{f_{\boldsymbol{X}}\left(\boldsymbol{x}\right)}_{\text{Evidence}}} \tag{10}$$

上一章介绍过，朴素贝叶斯分类优化目标也可以是——最大化后验概率：

$$\hat{y} = \arg\max_{C_k} f_{Y|\boldsymbol{X}}\left(C_k|\boldsymbol{x}\right) \tag{11}$$

图 13 ~ 图 15 所示为 $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$、$f_{Y|X1,X2}(C_2 \mid x_1, x_2)$ 和 $f_{Y|X1,X2}(C_3 \mid x_1, x_2)$ 三个后验概率曲面。

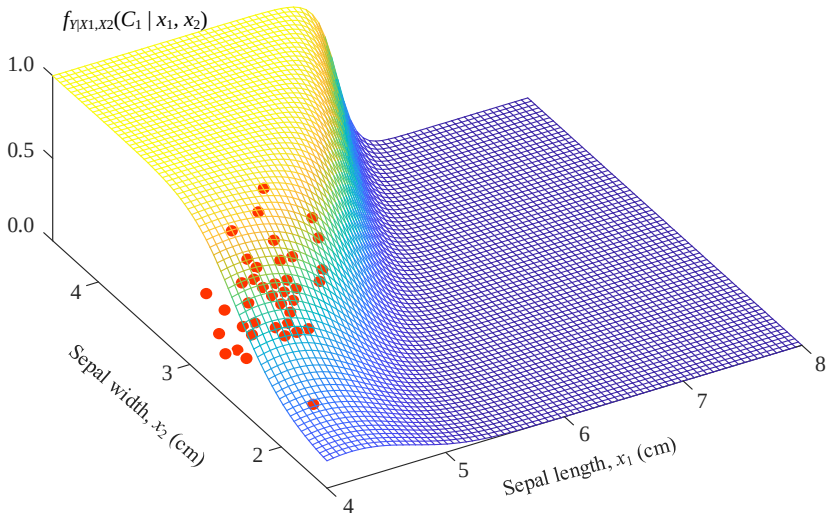


图 13. $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$ 后验概率曲面，基于高斯分布

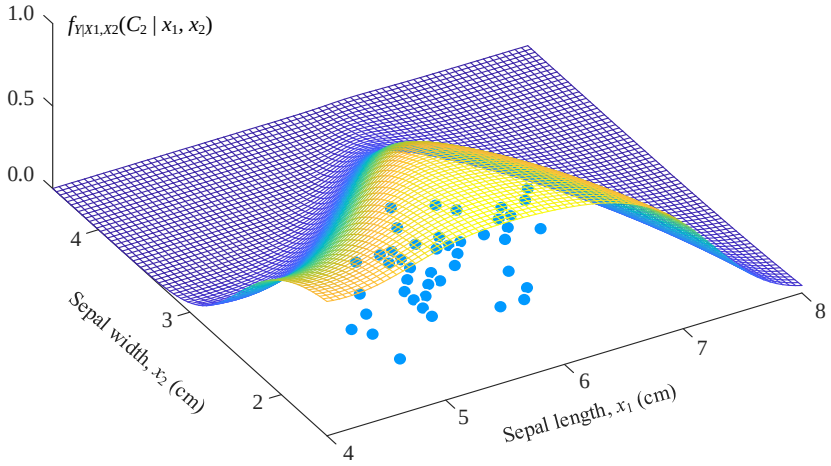


图 14. $f_{Y|X1,X2}(C_2 \mid x_1, x_2)$ 后验概率曲面，基于高斯分布

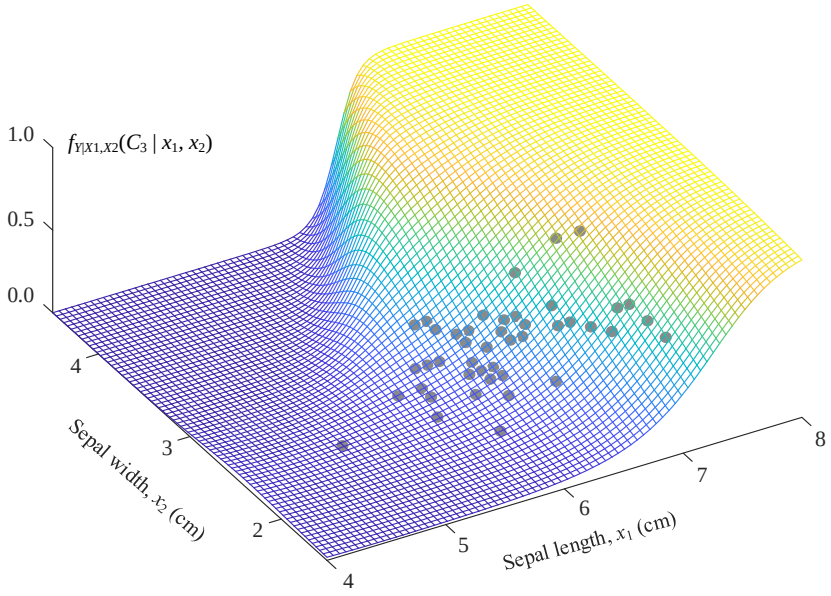


图 15. $f_{Y|X1,X2}(C_3 \mid x_1, x_2)$ 后验概率曲面，基于高斯分布

对于鸢尾花三分类问题，如图 16 所示，比较 $f_{Y|X1,X2}(C_1 \mid x_1, x_2)$、$f_{Y|X1,X2}(C_2 \mid x_1, x_2)$ 和 $f_{Y|X1,X2}(C_3 \mid x_1, x_2)$ 三个后验概率密度曲面高度，可以预测分类，并获得决策边界。

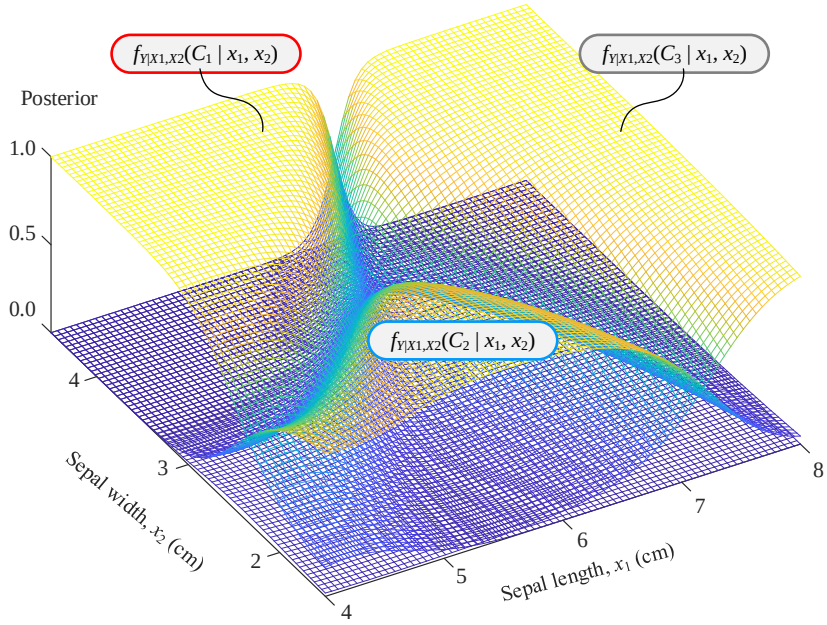


图 16. 比较三个后验概率曲面，基于高斯分布

贝叶斯定理是机器学习和深度学习中重要的概率论工具，广泛应用于分类、聚类、推荐系统等领域。本章和上一章介绍的朴素贝叶斯分类是贝叶斯定理的众多应用之一。我们在《统计至简》还介绍过贝叶斯统计推断，在《数据有道》聊过贝叶斯回归。

贝叶斯派思想强调我们对未知事物的认识应该是不断修正和更新的。它通过贝叶斯定理将已有的先验知识和新的实验数据结合起来，不断修正我们对未知事件的概率估计，实现对真实概率的逼近。贝叶斯派思想应用于机器学习和人工智能领域，可以用于推断和预测，解决实际问题，例如自然语言处理、图像识别、推荐系统等。贝叶斯派思想的优点是可以有效处理不确定性和噪声，具有广泛的应用前景。

6 高斯判别分析

Gaussian Discriminant Analysis

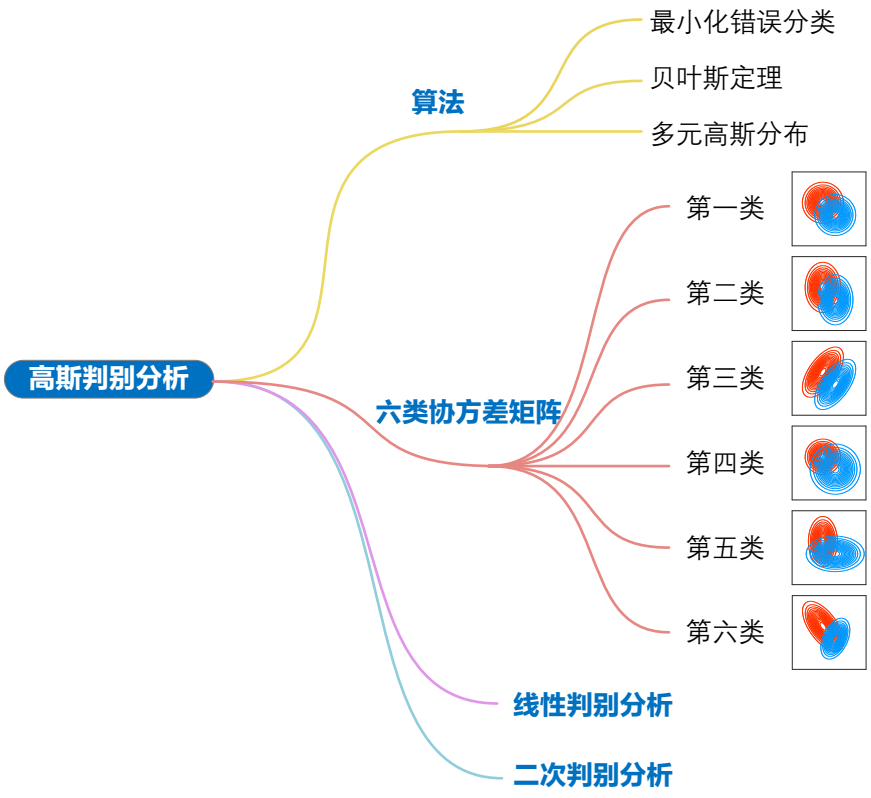
假设后验概率为高斯分布，最小化分类错误

唯有勇敢者，才能洞见科学的壮美。

The enchanting charms of this sublime science reveal themselves in all their beauty only to those who have the courage to go deeply into it.

—— 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

- `matplotlib.pyplot.contour()` 绘制等高线线图
- `matplotlib.pyplot.contourf()` 绘制填充等高线图
- `matplotlib.pyplot.scatter()` 绘制散点图
- `numpy.array()` 创建 array 数据类型
- `numpy.c_()` 按列叠加两个矩阵
- `numpy.linspace()`产生连续均匀向量数值
- `numpy.meshgrid()` 创建网格化数据
- `numpy.r_()` 按行叠加两个矩阵
- `numpy.ravel()` 将矩阵扁平化
- `seaborn.scatterplot()` 绘制散点图
- `sklearn.datasets.load_iris()` 加载鸢尾花数据集
- `sklearn.discriminant_analysis.LinearDiscriminantAnalysis` 线性判别分析函数
- `sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis` 二次判别分析函数



6.1 又见高斯

本章介绍**高斯判别分析** (Gaussian Discriminant Analysis, GDA)。高斯判别分析中，似然概率采用高斯多元分布估计，这便是其名称的由来。GDA 是一种监督学习算法，用于分类和判别问题。它基于假设，即每个类别的数据都服从高斯分布。具体来说，对于每个类别，GDA 通过估计该类别的均值和协方差矩阵来建模该类别的高斯分布。在训练过程中，算法学习这些参数，并使用它们来计算给定输入数据点属于哪个类别的后验概率。在测试时，算法使用这些后验概率来进行分类。

原理

图 1 所示为高斯判别分析的原理，大家可能已经发现六幅子图椭圆等高线呈现不同形态，这代表着高斯多元分布展现不同特点。

此外，图中决策边界包含一次函数和圆锥曲线。相信图 1 各种细节已经引起了大家好奇心，本章将为大家一一解密。

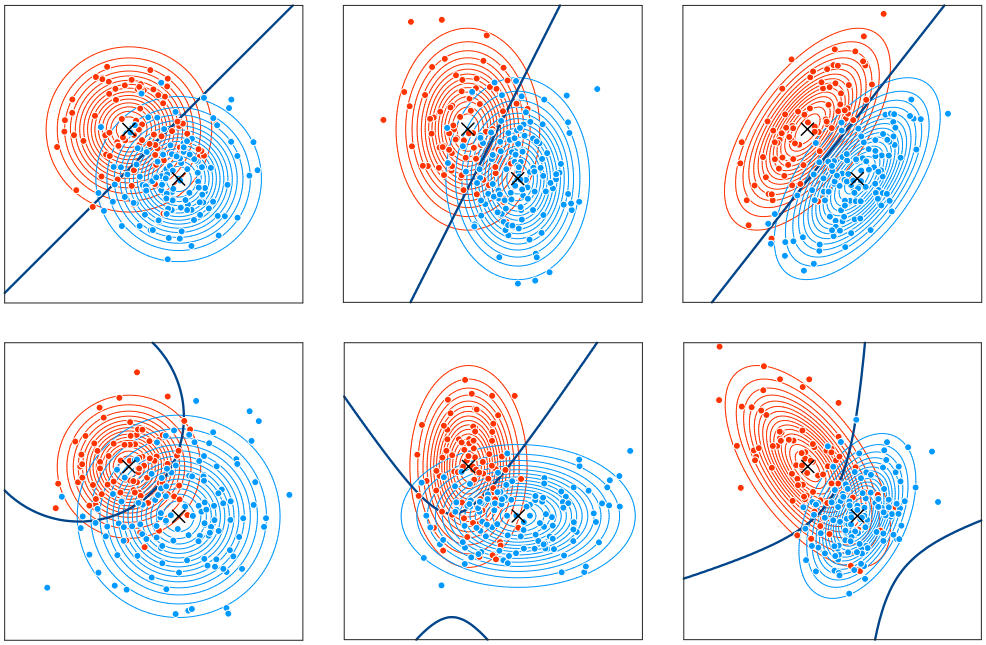


图 1. 高斯判别分析原理

分类

高斯判别分析，又细分为**线性判别分析** (Linear Discriminant Analysis, LDA) 和**二次判别分析** (Quadratic Discriminant Analysis, QDA)。

高斯判别分析算法得到的决策边界有解析解。从它们各自的名字上就可以看出，LDA 和 QDA 决策边界分别为线性式和二次式。

此外，二次判别分析 QDA 和上一章介绍的**高斯朴素贝叶斯** (Gaussian Naïve Bayes) 有着紧密关系。高斯判别分析分类算法和本书后续介绍的**高斯混合模型** (Gaussian Mixture Model, GMM) 也有千丝万缕的联系。

优化问题

高斯判别分析优化目标如下，**预测分类** (predicted classification) $\hat{y}$ 可以通过下式求得：

$$\hat{y} = \arg\min_{C_m} \sum_{k=1}^{K} f_{Y|\chi}\left(C_k|\boldsymbol{x}\right) \cdot c\left(C_m|C_k\right) \tag{1}$$

其中，K 为类别数量，m 和 k 均为类别序数——1, 2, …, K。

$f_{Y|\chi}\left(C_k|\boldsymbol{x}\right)$ 为任意一点 $\boldsymbol{x}$ 被预测分类为 C_k 类的**后验概率** (posterior)。

$c(C_m \mid C_k)$ 为惩罚因子，代表 $\boldsymbol{x}$ 正确分类为 C_k，但被预测分类为 C_m 对应的代价，具体计算如下：

$$c\left(C_m|C_k\right) = \begin{cases} 1 & m \neq k \\ 0 & m = k \end{cases} \tag{2}$$

$m \neq k$ 时，也就是当某一点真实类别为 C_k，但是却被错误地分类为 C_m 时，$c(C_m \mid C_k) = 1$。而当分类正确时，即 $m = k$，$c(C_m \mid C_k) = 0$。

(1) 中蕴含着高斯判别分析重要的思路——最小化错误分类。这个思路和朴素贝叶斯恰好相反。

计算后验概率

根据**贝叶斯定理** (Bayes theorem)，后验概率 $f_{Y|\chi}\left(C_k|\boldsymbol{x}\right)$ 可以通过下式计算获得：

$$\underbrace{f_{Y|\chi}\left(C_k|\boldsymbol{x}\right)}_{\text{Posterior}} = \frac{\overbrace{f_{\chi,Y}\left(\boldsymbol{x},C_k\right)}^{\text{Joint}}}{\underbrace{f_{\chi}\left(\boldsymbol{x}\right)}_{\text{Evidence}}} = \frac{\overbrace{f_{\chi|Y}\left(\boldsymbol{x}|C_k\right)}^{\text{Likelihood}}\overbrace{p_Y\left(C_k\right)}^{\text{Prior}}}{\underbrace{f_{\chi}\left(\boldsymbol{x}\right)}_{\text{Evidence}}} \tag{3}$$

证据因子 $f_{\chi}\left(\boldsymbol{x}\right)$ 可以通过下式求得：

$$f_{\chi}\left(\boldsymbol{x}\right) = \sum_{k=1}^{K} \overbrace{f_{\chi|Y}\left(\boldsymbol{x}|C_k\right)}^{\text{Likelihood}}\overbrace{p_Y\left(C_k\right)}^{\text{Prior}} \tag{4}$$

和本书之前介绍的朴素贝叶斯一样，证据因子 $f_{\chi}\left(\boldsymbol{x}\right)$ 也可以不求，后验 $\propto$ 似然 $\times$ 先验：

$$\underbrace{f_{Y|\chi}\left(C_k|\boldsymbol{x}\right)}_{\text{Posterior}} \propto \underbrace{f_{\chi,Y}\left(\boldsymbol{x},C_k\right)}_{\text{Joint}} \tag{5}$$

证据因子 $f_{\chi}\left(\boldsymbol{x}\right)$ 相当于对 $f_{\chi,Y}\left(\boldsymbol{x},C_k\right)$ 归一化处理。

引入多元高斯分布

高斯判别分析假设，似然概率 $f_{\chi|Y}\left(\boldsymbol{x}|C_k\right)$ 服从多元高斯分布，因此 $f_{\chi|Y}\left(\boldsymbol{x}|C_k\right)$ 具体表达式如下：

$$f_{\chi|Y}\left(\boldsymbol{x}|C_k\right)=\frac{\exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_k\right)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_k\right)\right)}{\sqrt{\left(2\pi\right)^{D}\left|\boldsymbol{\Sigma}_k\right|}} \tag{6}$$

其中，D 为特征数量，$\boldsymbol{x}$ 为列向量，$\boldsymbol{\mu}_k$ 为 C_k 类数据质心位置，$\boldsymbol{\Sigma}_k$ 为 C_k 类样本协方差矩阵。

可以说，此处便是高斯判别分析和高斯朴素贝叶斯分道扬镳之处！

6.2 六类协方差矩阵

高斯朴素贝叶斯，假设“特征条件独立”。

然而，高斯判别分析，根据 $\boldsymbol{\Sigma}_k$ 形态将算法分成六个类别。这六个类别中，有些特征条件独立，有些特征满足特殊条件。表 1 总结了六类高斯判别分析对应的协方差矩阵特点。

表 1. 根据协方差矩阵特点将高斯判别分析问题分成 6 类

	$\boldsymbol{\Sigma}_k$	特征方差 ($\boldsymbol{\Sigma}_k$ 对角线元素)	$\boldsymbol{\Sigma}_k$ 特点	似然概率 PDF 等高线	决策边界
第一类	相同	相同	对角阵 (特征条件独立)	正圆，形状相同	直线
第二类		不限制		正椭圆，形状相同	
第三类			非对角阵	任意椭圆，形状相同	
第四类	不同	相同	对角阵 (特征条件独立)	正圆	正圆
第五类		不限制		正椭圆	正圆锥曲线
第六类			非对角阵	任意椭圆	圆锥曲线

图 2 所示为六大类判别分析高斯分布椭圆形状。

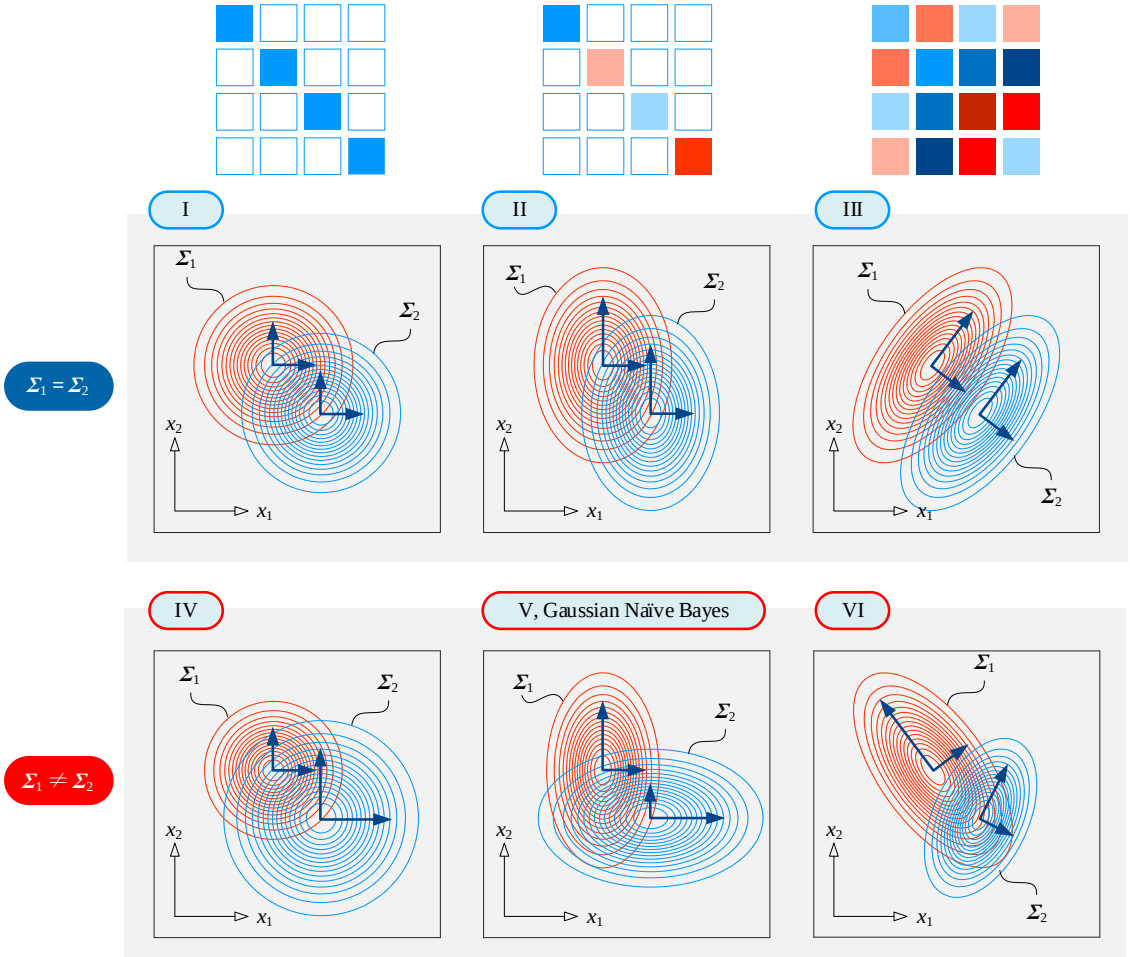


图 2. 六大类判别分析高斯分布椭圆形状，$K = 2$，$D = 2$

前三类：决策边界为直线

前三类 (I、II 和 III) GDA 有一个共同特点，假设各个类别协方差矩阵 Σ_k，完全一致；因此，这三类的决策边界为直线，因此它们被称作线性判别分析 LDA。这一点，本章后续将展开讲解；这里先给大家结论，希望读者学完本章回过头来再看一遍。

- 第一类 GDA 的重要特点是，Σ_k 为对角阵 (除主对角线之外元素为 0)，即特征之间“条件独立”。并且，Σ_k 对角线元素相同，即假设各个特征方差相同。因此，图 2 所示第一类 GDA 中，红色和蓝色 PDF 等高线为正圆，且大小相同。
- 第二类 GDA，Σ_k 为对角阵，特征条件独立；但是，对 Σ_k 对角线元素大小不做限制。如图 2 所示，红色和蓝色 PDF 等高线为大小相等的正椭圆。
- 第三类 GDA，仅仅假设各个类别协方差矩阵 Σ_k，完全一致。方差和条件独立不做任何限制。如图 2 所示，红色和蓝色 PDF 等高线为大小相等旋转椭圆。

后三类：决策边界为二次曲线

后三类 (IV、V 和 VI) GDA，各个类别协方差矩阵 Σ_k 不相同；这三类 GDA 决策边界为二次曲线，因此被称作二次判别分析 QDA。

- 第四类 GDA，Σ_k 为对角阵，也假设特征之间“条件独立”；同时假设每个类别协方差矩阵 Σ_k 对角线元素相同，即假设类别内样本数据各个特征方差相同。图 2 所示第四类 GDA 中，红色和蓝色 PDF 等高线为正圆，但是大小不同。
- 第五类 GDA，Σ_k 为对角阵，特征条件独立；不限制 Σ_k 对角线元素大小；如图 2 所示，红色和蓝色 PDF 等高线为正椭圆，但是大小不同。可以发现，第五类 (V) 对应高斯朴素贝叶斯分类。从几何图像上解释，高斯朴素贝叶斯中，条件概率曲面等高线为正椭圆。
- 第六类 GDA，对 Σ_k 不做任何限制。如图 2 所示，红色和蓝色 PDF 等高线旋转椭圆，大小不等。

此外，非监督学习中**高斯混合模型** (Gaussian Mixture Model, GMM) 也会使用到本章介绍的协方差矩阵特点和决策边界关系。

6.3 决策边界解析解

本节推导决策边界解析解一般形式。

判别函数

定义 C_k 类判别函数 $g_k(\boldsymbol{x})$ 如下：

$$
\begin{aligned}
g_k(\boldsymbol{x}) &= \ln\left(f_{\boldsymbol{X},Y}(\boldsymbol{x}, C_k)\right) = \ln\left(f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_k)\, p_Y(C_k)\right) \\
&= \ln\left(\frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right)}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}_k|}}\, p_Y(C_k)\right) \\
&= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_k| + \ln p_Y(C_k)
\end{aligned}
\tag{7}
$$

判别函数就是联合概率密度函数的自然对数。白话说，这个运算是为了去掉多元高斯分布中的 exp()。

两特征、两分类问题

为了方便讨论，本章以两个特征 ($D = 2$) 二分类 ($K = 2$) 为例。C_1 和 C_2 的判别函数分别为：

$$\begin{cases} g_1(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) - \frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_1| + \ln p_Y(C_1) \\ g_2(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^{\mathrm{T}}\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) - \frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_2| + \ln p_Y(C_2) \end{cases} \tag{8}$$

其中

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{9}$$

对于二分类 ($K = 2$) 问题，高斯判别分析的决策边界取决于如下等式：

$$g_1(\boldsymbol{x}) = g_2(\boldsymbol{x}) \tag{10}$$

将 (8) 代入 (10) 并整理得到决策边界对应的解析式：

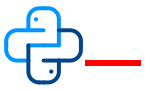
$$\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^{\mathrm{T}}\boldsymbol{\Sigma}_2^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) = \ln p_Y(C_1) - \ln p_Y(C_2) + \left(\frac{1}{2}\ln|\boldsymbol{\Sigma}_2| - \frac{1}{2}\ln|\boldsymbol{\Sigma}_1|\right) \tag{11}$$

容易发现上式为二次式，甚至是一次式。决策边界解析解次数和具体参数，和两个协方差矩阵 ($\boldsymbol{\Sigma}_1$ 和 $\boldsymbol{\Sigma}_2$) 取值直接相关。而先验概率 $p_Y(C_1)$ 和 $p_Y(C_2)$ 影响常数项。

决策边界形态

图 3 所示为各种高斯判别分析 QDA 二分类常见决策边界形态。观察图 3 可以发现，决策边界可以是直线、正圆、椭圆、抛物线、双曲线，以及各种蜕化二次曲线。

下一节开始讲逐个讲解各类 QDA。



代码 Bk7_Ch06_01.py 可视化二分类联合概率和决策边界。请读者学完本章后，自行修改簇质心位置 (代码中 `mu1` 和 `mu2`) 和簇协方差矩阵 (代码中 `sigma1` 和 `sigma2`)；大家可以交互式修改先验概率数值，并观察决策边界形态变化。

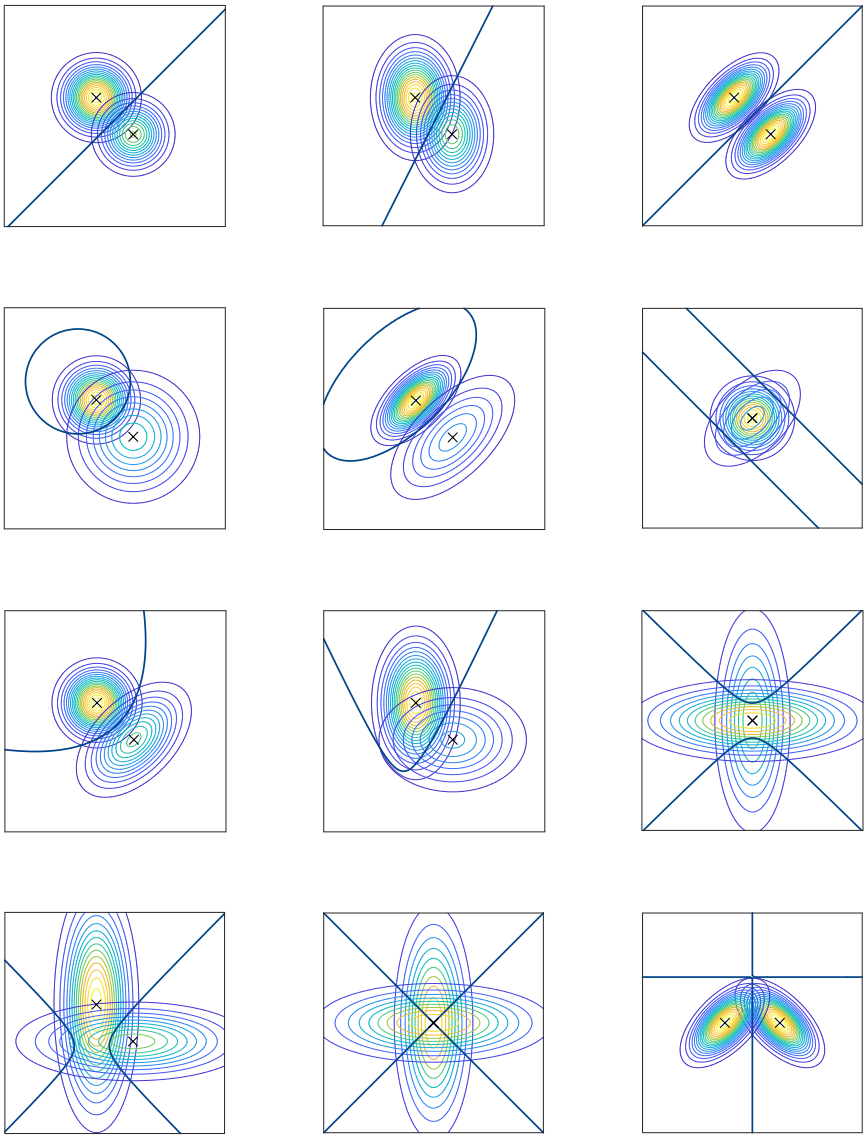


图 3.判别分析常见决策边界

6.4 第一类

第一类高斯判别分析，假设数据特征条件独立。协方差矩阵 $\boldsymbol{\Sigma}_k$ 为对角阵，即相关系数为 0，且假设 $\boldsymbol{\Sigma}_k$ 对角元素相同，即特征方差相同。

两特征、两分类

如下例 ($K = 2$ 且 $D = 2$)：

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \sigma^2 \boldsymbol{I} \tag{12}$$

两个协方差矩阵 ($\boldsymbol{\Sigma}_1$ 和 $\boldsymbol{\Sigma}_2$) 的逆矩阵如下：

$$\boldsymbol{\Sigma}_1^{-1} = \boldsymbol{\Sigma}_2^{-1} = \frac{1}{\sigma^2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{\boldsymbol{I}}{\sigma^2} \tag{13}$$

将 (13) 代入 (11) 得到：

$$\begin{aligned} &\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}} \frac{\boldsymbol{I}}{\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_1) - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^{\mathrm{T}} \frac{\boldsymbol{I}}{\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_2) = \ln p_Y(C_1) - \ln p_Y(C_2) \\ &\Rightarrow (\boldsymbol{x}-\boldsymbol{\mu}_1)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{\mu}_1) - (\boldsymbol{x}-\boldsymbol{\mu}_2)^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{\mu}_2) = 2\sigma^2\left(\ln p_Y(C_1) - \ln p_Y(C_2)\right) \end{aligned} \tag{14}$$

决策边界

整理得到的决策边界解析解如下：

$$(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{x} - \left[\sigma^2\left(\ln p_Y(C_1) - \ln p_Y(C_2)\right) + \frac{1}{2}\left(\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\mu}_1\right)\right] = 0 \tag{15}$$

回忆《矩阵力量》介绍的空间直线矩阵运算表达式：

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0 \tag{16}$$

$\boldsymbol{w}$ 为该直线法向量，也是梯度向量。比较 (15) 和 (16) 可以得到直线参数：

$$\begin{cases} \boldsymbol{w} = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \\ b = -\left[\sigma^2\left(\ln p_Y(C_1) - \ln p_Y(C_2)\right) + \frac{1}{2}\left(\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\mu}_1\right)\right] \end{cases} \tag{17}$$

先验概率

特别的，当 $p_Y(C_1) = p_Y(C_2)$，且 $\boldsymbol{\mu}_1 \neq \boldsymbol{\mu}_2$ 时，代入 (15) 可以得到：

$$\begin{aligned} &(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{x} - \frac{1}{2}\left(\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\mu}_1\right) = 0 \\ &\Rightarrow (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}(\boldsymbol{\mu}_2+\boldsymbol{\mu}_1) = 0 \\ &\Rightarrow (\boldsymbol{\mu}_2-\boldsymbol{\mu}_1)^{\mathrm{T}}\left[\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_2+\boldsymbol{\mu}_1)\right] = 0 \end{aligned} \tag{18}$$

特别提醒读者的是，式 (18) 中 $(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^{\mathrm{T}}$ 不能消去。

当 $p_Y(C_1) = p_Y(C_2)$ 时，观察 (18) 可以发现，决策边界直线通过 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 两点中点 $(\boldsymbol{\mu}_2 + \boldsymbol{\mu}_1)/2$，并垂直于两点连线，对应 $(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$ 向量。也就是说，决策边界为 C_1 和 C_2 类质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 中垂线。

再次注意，因为 $p_Y(C_1) = p_Y(C_2)$，所以决策边界距离 C_1 和 C_2 两类样本数据质心 ($\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$) 等距。给大家提一个小问题，如果 $p_Y(C_1) > p_Y(C_2)$，决策边界更靠近 C_1，还是 C_2?

举个例子

采用如下具体数值讨论第一类高斯判别分析：

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \quad p_Y(C_1) = 0.6, \quad p_Y(C_2) = 0.4, \quad \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{19}$$

图 4 直接比较 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 曲面高度，任意一点 $\boldsymbol{x}$，如果 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x}) > f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$，则该点分类可以被判定为 C_1；反之，$f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x}) < f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$，则该点分类可以被判定为 C_2。$f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x}) = f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 处便是决策边界。

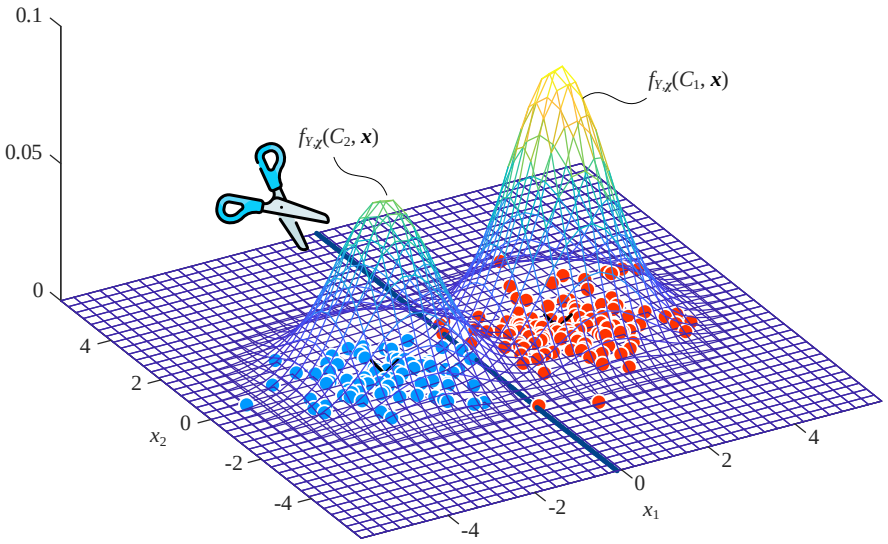


图 4. 第一类高斯判别分析，比较 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 曲面，$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = [1\ 0; 0\ 1]$，$p_Y(C_1) = 0.6$，$p_Y(C_2) = 0.4$

平面等高线更方便探讨高斯分布形状和决策边界。图 5 (a) 比较 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 两个曲面等高线；两个曲面交线便是决策边界 (图 5 (a) 中深蓝色线)。$p_Y(C_1)$ 大于 $p_Y(C_2)$，因此 C_1 类数据的影响更大，决策边界便远离质心 $\boldsymbol{\mu}_1$；也就是说 C_1“势力”更大。

观察图 5 (a) 等高线，发现 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 同心圆大小不同。再次，注意图 5 (a) 等高线为联合概率密度函数，而图 2 为似然概率。

由于每一类数据在每个特征上方差相同，且条件独立；因此 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 等高线为正圆。

图 5 (b) 比较后验概率 $f_{Y|\boldsymbol{X}}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|\boldsymbol{X}}(C_2 \mid \boldsymbol{x})$ 曲面。由于图 5 所示为二分类问题，因此只要 $f_{Y|\boldsymbol{X}}(C_1 \mid \boldsymbol{x}) > 0.5$，则可判定 $\boldsymbol{x}$ 分类为 C_1。

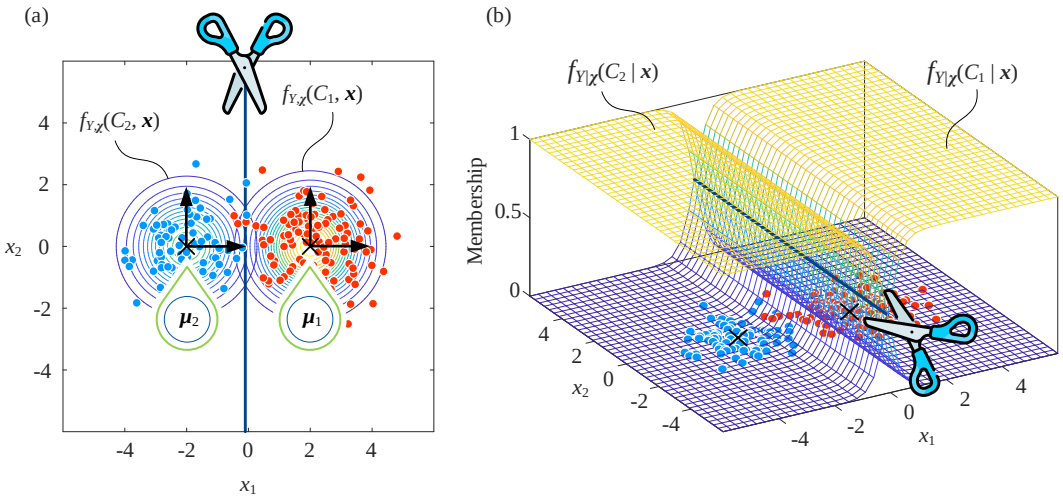


图 5. 第一类高斯判别分析，$\Sigma_1 = \Sigma_2 = [1\ 0; 0\ 1]$，$p_Y(C_1) = 0.6$，$p_Y(C_2) = 0.4$

6.5 第二类

第二类高斯判别分析，Σ_k相等且为对角阵 (协方差矩阵除主对角线外，其他元素为 0，即特征条件独立)；但是，主对角线元素不相等。

两特征、两分类

对于 $D = 2$，$K = 2$ 的情况，Σ_1 和 Σ_2 可以写成如下形式：

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \tag{20}$$

其中，$\sigma_1 \neq \sigma_2$。

由于 Σ_1 和 Σ_2 相等，代入 (11)，可以发现二次项消去；因此，确定第二类高斯判别分析的决策边界也是直线。

举个例子

下面，举个例子分析第二类高斯判别：

$$\mu_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \mu_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad p_Y(C_1) = 0.4, \quad p_Y(C_2) = 0.6, \quad \Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \tag{21}$$

图 6 比较 $f_{Y,X}(C_1, x)$ 和 $f_{Y,X}(C_2, x)$ 曲面，这两个曲面的交线为决策边界；$p_Y(C_2) > p_Y(C_1)$，$f_{Y,X}(C_2, x)$ 曲面高度高于 $f_{Y,X}(C_1, x)$。

观察图 7 (a)，发现 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 曲面等高线为形状相似正椭圆。图 7 (b) 所示为 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 后验概率曲面，以及两个曲面交线，即决策边界。

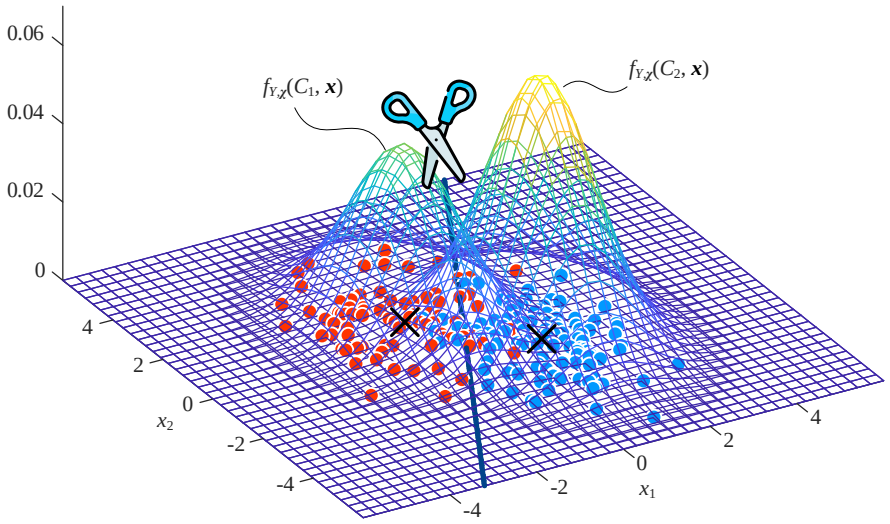


图 6. 比较 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 曲面；第二类高斯判别分析，$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = [1\ 0; 0\ 2]$，$p_Y(C_1) = 0.4$，$p_Y(C_2) = 0.6$

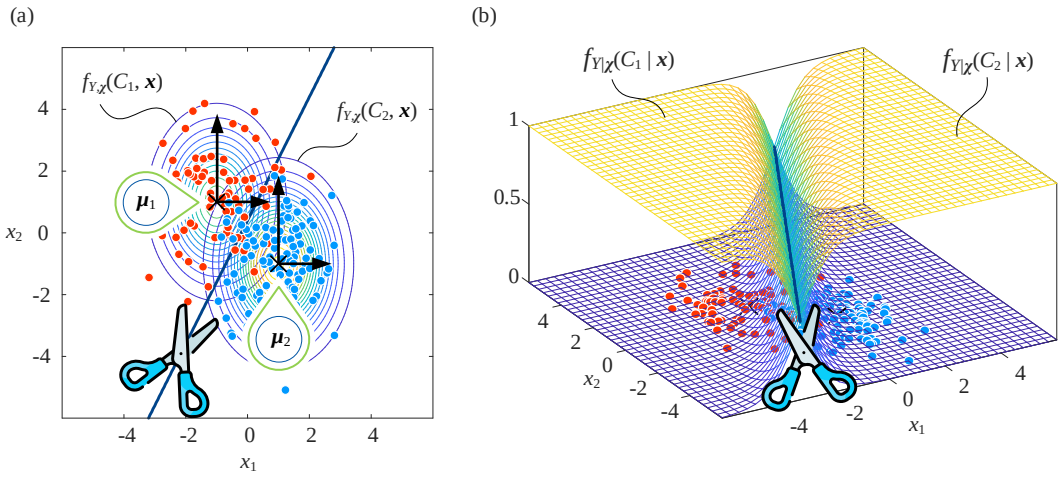


图 7. 第二类高斯判别分析，$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = [1\ 0; 0\ 2]$，$p_Y(C_1) = 0.4$，$p_Y(C_2) = 0.6$

6.6 第三类

第三类高斯判别分析特点是，仅假设类别协方差矩阵 $\boldsymbol{\Sigma}_k$ 完全一致；方差和条件独立不做任何限制。

举个例子

$$\boldsymbol{\mu}_1=\begin{bmatrix}-1\\1\end{bmatrix},\quad \boldsymbol{\mu}_2=\begin{bmatrix}1\\-1\end{bmatrix},\quad p_Y(C_1)=0.5,\quad p_Y(C_2)=0.5,\quad \boldsymbol{\Sigma}_1=\boldsymbol{\Sigma}_2=\begin{bmatrix}1 & 0.8\\0.8 & 2\end{bmatrix} \tag{22}$$

图 8 比较 $f_{Y,X}(C_1,\boldsymbol{x})$ 和 $f_{Y,X}(C_2,\boldsymbol{x})$。根据 $\boldsymbol{\Sigma}_1=\boldsymbol{\Sigma}_2$ 这个条件，可以判定决策边界为直线。由于假设 $p_Y(C_1)=p_Y(C_2)$，$f_{Y,X}(C_1,\boldsymbol{x})$ 曲面和 $f_{Y,X}(C_2,\boldsymbol{x})$ 高度相等。

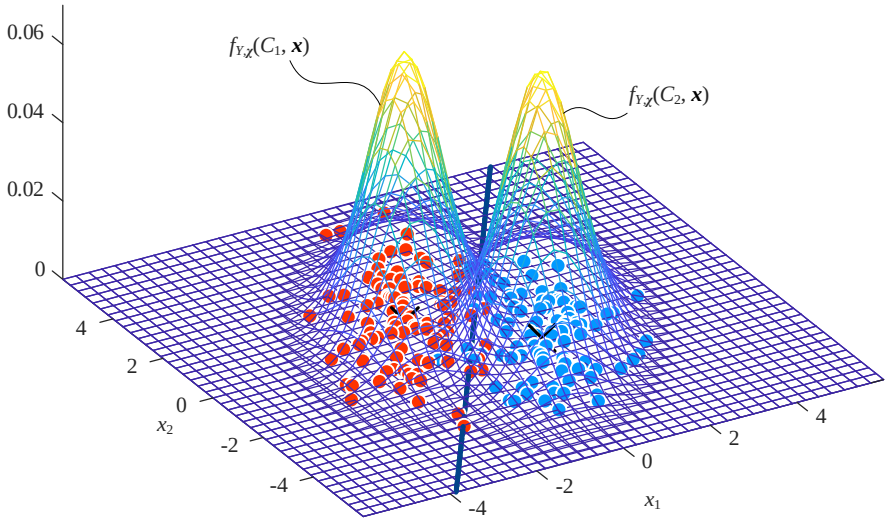


图 8. 比较 $f_{Y,X}(C_1,\boldsymbol{x})$ 和 $f_{Y,X}(C_2,\boldsymbol{x})$ 曲面；第三类高斯判别分析，$\boldsymbol{\Sigma}_1=\boldsymbol{\Sigma}_2$ = [1 0.8; 0.8 2]，$p_Y(C_1)$ = 0.5，$p_Y(C_2)$ = 0.5

观察图 9 (a) 可以发现，$f_{Y,X}(C_1,\boldsymbol{x})$ 和 $f_{Y,X}(C_2,\boldsymbol{x})$ 等高线为旋转椭圆，这是因为 $\boldsymbol{\Sigma}_1$ 和 $\boldsymbol{\Sigma}_2$ 两个矩阵协方差不为 0。图 9 (b) 所示为 $f_{Y|X}(C_1\,|\,\boldsymbol{x})$ 和 $f_{Y|X}(C_2\,|\,\boldsymbol{x})$ 两个后验概率曲面，以及决策边界。

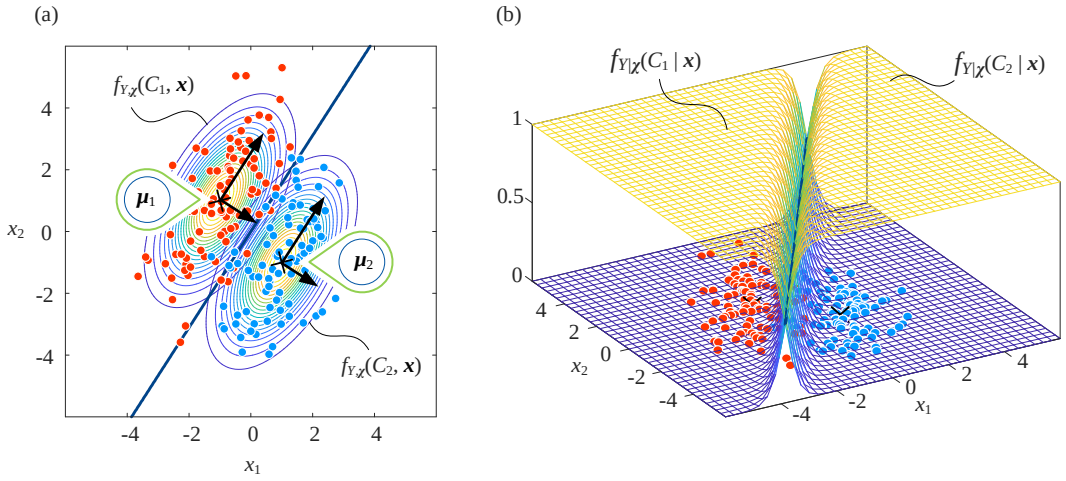


图 9. 第三类高斯判别分析，$\boldsymbol{\Sigma}_1=\boldsymbol{\Sigma}_2$ = [1 0.8; 0.8 2]，$p_Y(C_1)$ = 0.5，$p_Y(C_2)$ = 0.5

6.7 第四类

第四类高斯判别分析，Σ_k 为对角阵 (除主对角线之外元素为 0)，即类别内特征“条件独立”；并且，Σ_k 各自对角线元素相同。不同于第一类，第四类不同类别 Σ_k 不同。

举个例子

给定如下条件：

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad p_Y(C_1) = 0.3, \quad p_Y(C_2) = 0.7, \quad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \tag{23}$$

图 10 比较 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 曲面。投影在 x_1x_2 平面上，第四类高斯判别分析的决策边界为正圆，如图 11 (a) 所示。图 11 (b) 比较两个后验曲面高度。

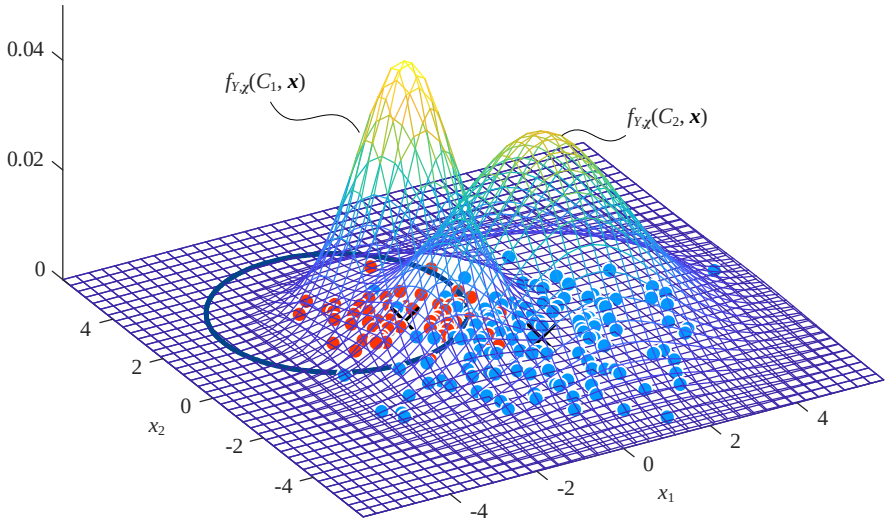


图 10. 比较 $f_{Y,\boldsymbol{X}}(C_1, \boldsymbol{x})$ 和 $f_{Y,\boldsymbol{X}}(C_2, \boldsymbol{x})$ 曲面；第四类高斯判别分析，Σ_1 = [1 0; 0 1], Σ_2 = [3 0; 0 3]，$p_Y(C_1)$ = 0.3，$p_Y(C_2)$ = 0.7

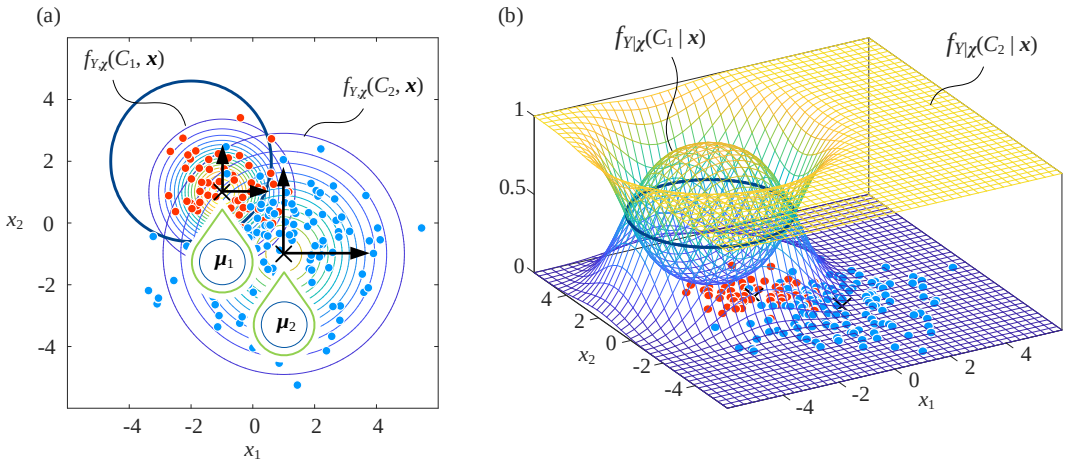


图 11. 第四类高斯判别分析，Σ_1 = [1 0; 0 1], Σ_2 = [3 0; 0 3]，$p_Y(C_1)$ = 0.3，$p_Y(C_2)$ = 0.7

6.8 第五类

第五类高斯判别分析，不同类别 Σ_k 不同；Σ_k 为对角阵，类别内特征条件独立。但是，第五类高斯判别分析对 Σ_k 对角线元素大小不做限制。

再次请大家注意，第五类高斯判别分析对应高斯朴素贝叶斯分类。大家可以自己推导第五类高斯判别分析决策边界一般式。

举个例子

下面举个例子方便可视化：

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad p_Y(C_1) = 0.4, \quad p_Y(C_2) = 0.6, \quad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \tag{24}$$

由于 Σ_1 和 Σ_2 均为对角阵，决策边界解析式中没有 x_1x_2 项。因此，决策边界为正圆锥曲线，如图 12 所示。图 13 (a) 比较 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 曲面等高线；图 13 (b) 比较后验曲面等高线。

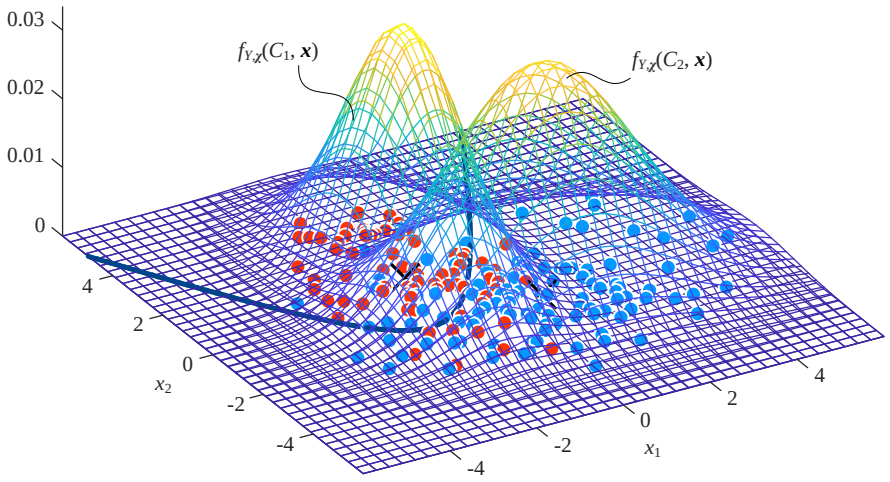


图 12. 比较 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 曲面；第五类高斯判别分析，Σ_1 = [1 0; 0 3], Σ_2 = [4 0; 0 2]，$p_Y(C_1)$ = 0.4，$p_Y(C_2)$ = 0.6

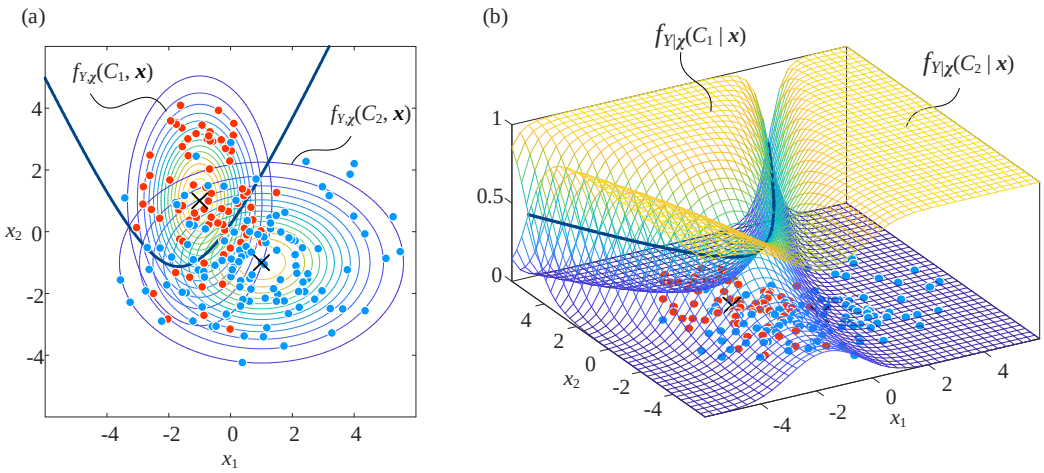


图 13. 第五类高斯判别分析，$\boldsymbol{\Sigma}_1$ = [1 0; 0 1], $\boldsymbol{\Sigma}_2$ = [3 0; 0 3]，$p_Y(C_1) = 0.4$，$p_Y(C_2) = 0.6$

6.9 第六类

第六类高斯判别分析对 $\boldsymbol{\Sigma}_k$ 不做任何限制。后验概率 PDF 等高线为任意椭圆。第六类高斯判别分析的决策边界可以是单条直线、平行直线、椭圆、双曲线、蜕化双曲线、抛物线等等。下面看几个例子。

椭圆

如下参数条件得到的决策边界为椭圆：

$$\boldsymbol{\mu}_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \begin{cases} p_Y(C_1) = 0.2 \\ p_Y(C_2) = 0.8 \end{cases}, \quad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 3 & 1.8 \\ 1.8 & 3 \end{bmatrix} \tag{25}$$

如图 14 所示，(25) 对应决策边界为椭圆。

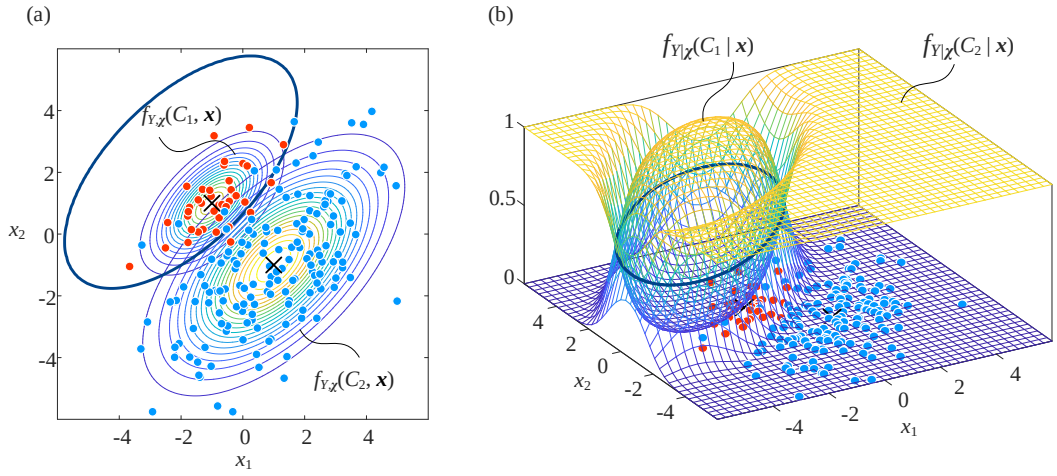


图 14. 决策边界为椭圆

双曲线

下例给出的参数条件得到双曲线决策边界，如图 15 所示：

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \begin{cases} p_Y(C_1) = 0.4 \\ p_Y(C_2) = 0.6 \end{cases}, \quad \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & -0.6 \\ -0.6 & 1 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2 = \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix} \tag{26}$$

有了以上铺垫，本章最后介绍线性判别分析和二次判别分析。

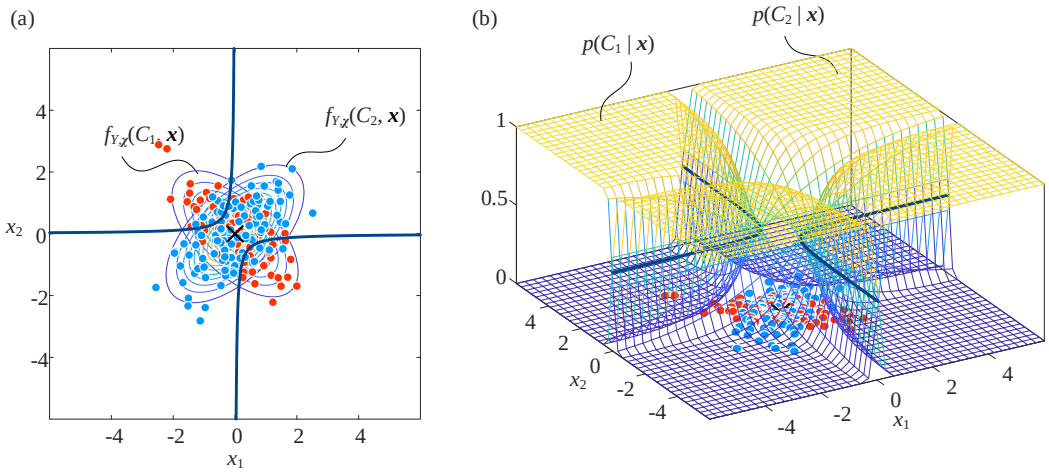


图 15. 决策边界为双曲线

读者可以利用代码 Bk7_Ch06_02.py，生成交互式绘图。请读者自行修改质心和协方差数据，产生交互式绘图。绘图窗口内，调节 $p_Y(C_1)$，观察等高线变化和决策边界变化。

6.10 线性和二次判别分析

线性判别分析 (Linear Discriminant Analysis, LDA) 是一种监督学习算法，用于分类和降维问题。它基于假设，即每个类别的数据都满足高斯分布，并且不同类别之间的协方差矩阵相等。具体来说，LDA 通过寻找一个投影方向，可以将数据从高维空间投影到低维空间，并最大程度地保留不同类别之间的差异，同时最小化同一类别内部的方差。

这个投影方向可以被认为是一条线，称为“判别线”，可以用于分类或降维。在训练过程中，算法学习这个判别线，并使用它来计算给定输入数据点属于哪个类别的后验概率。LDA 是 GDA 的一种特殊形式。

投影

本节先以“降维”这种思路讨论线性判别分析。

如图 16 所示，采用高斯分布描述 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ 这两个联合概率。四个参数刻画 $f_{Y,X}(C_1, \boldsymbol{x})$ 和 $f_{Y,X}(C_2, \boldsymbol{x})$ —— (1) C_1 质心位置 $\boldsymbol{\mu}_1$；(2) C_2 质心位置 $\boldsymbol{\mu}_2$；(3) C_1 形状 $\boldsymbol{\Sigma}_1$；(4) C_2 形状 $\boldsymbol{\Sigma}_2$。

决策边界的解析式为：

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0 \tag{27}$$

其中，$\boldsymbol{w}$ 为直线梯度向量，且为单位矩阵。

C_1 质心位置 $\boldsymbol{\mu}_1$ 和 C_2 质心位置 $\boldsymbol{\mu}_2$ 沿着决策边界方向投影，也就是向 $\boldsymbol{w}$ 投影，可以得到：

$$\mu_{1\_w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_1, \quad \mu_{2\_w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\mu}_2 \tag{28}$$

$\boldsymbol{\Sigma}_1$ 和 $\boldsymbol{\Sigma}_2$ 向 $\boldsymbol{w}$ 向量方向投影，得到：

$$\sigma^2_{1\_w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_1\boldsymbol{w}, \quad \sigma^2_{2\_w} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\Sigma}_2\boldsymbol{w} \tag{29}$$

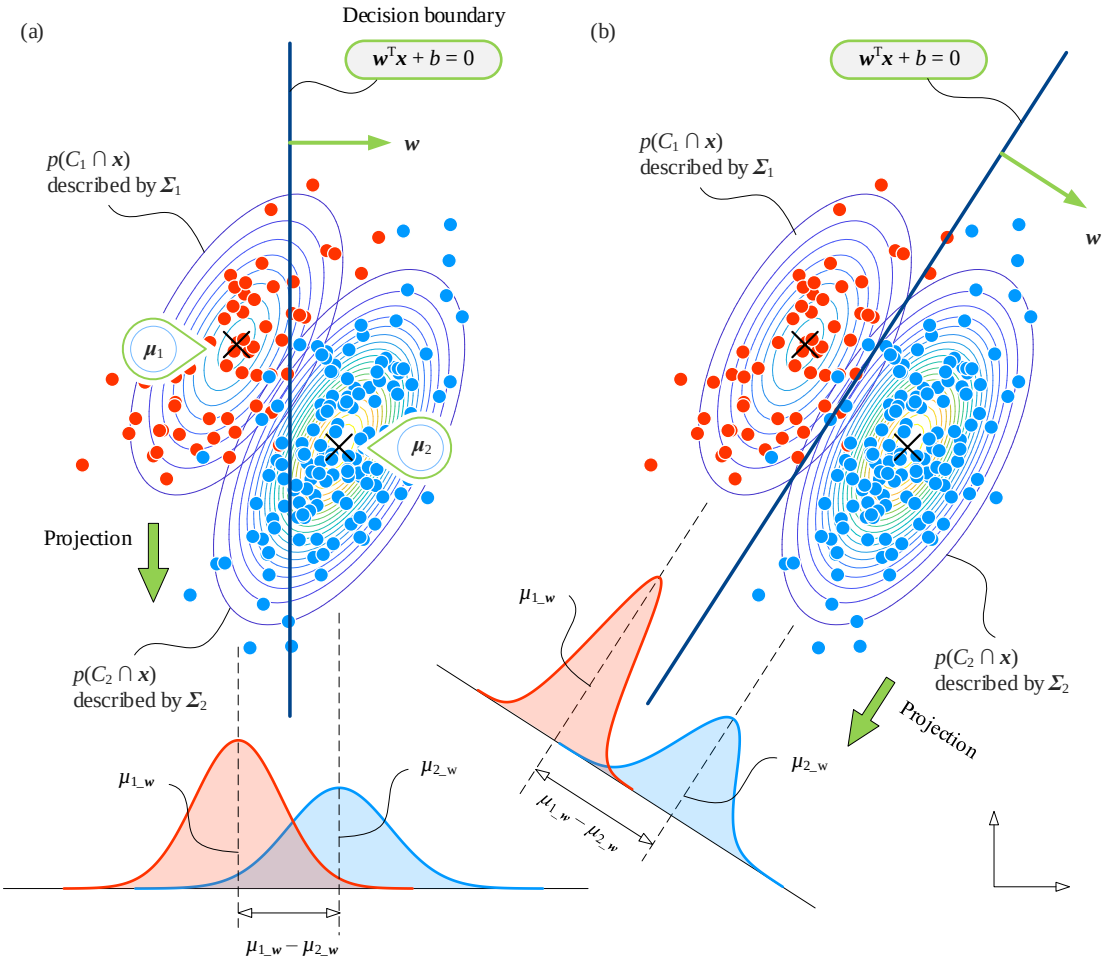


图 16. 从投影角度解释线性判别分析

比较图 16 两个子图，可以发现图 16 (b) 的分类效果更好。线性判别分析相当于样本数据投影后，最大化类间差异 $(\mu_{1\_w} - \mu_{2\_w})$，且最小化类内差异 $(\sigma^2_{1\_w} + \sigma^2_{2\_w})$。

目标函数

从投影角度，线性判别分析的目标函数为：

$$\arg\max_{w} \frac{\left(\mu_{1\_w} - \mu_{2\_w}\right)^2}{\sigma_{1\_w}^2 + \sigma_{2\_w}^2} \tag{30}$$

将 (28) 和 (29) 代入 (30)，得到：

$$\arg\max_{w} \frac{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^{\mathrm{T}}\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)\boldsymbol{w}} \tag{31}$$

(31) 分子描述的是类间距离，分母描述的是类内聚集程度。看到这个式子，大家是否眼前一亮？

《矩阵力量》从瑞利商、特征值分解、拉格朗日乘子法几个不同角度讲解过上式，建议大家回顾。

分类器函数

`sklearn.discriminant_analysis.LinearDiscriminantAnalysis` 为线性判别分析函数。这个函数采用本章第三类 GDA，即仅仅假设各个类别协方差矩阵 $\boldsymbol{\Sigma}_k$ 完全一致；方差和条件独立不做任何限制。

`sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis` 为二次判别分析算法函数。这个函数采用本章介绍的第六类 GDA，对 $\boldsymbol{\Sigma}_k$ 不做任何限制。

图 17 和图 18 分别展示采用线性判别分析 LDA 和二次判别分析 QDA 分类鸢尾花结果。简单来说，二次判别分析是一种监督学习算法，用于分类问题。它基于假设，即每个类别的数据都满足二次高斯分布，即每个类别的协方差矩阵都不相等。具体来说，QDA 通过估计每个类别的均值和协方差矩阵来建模该类别的二次高斯分布。

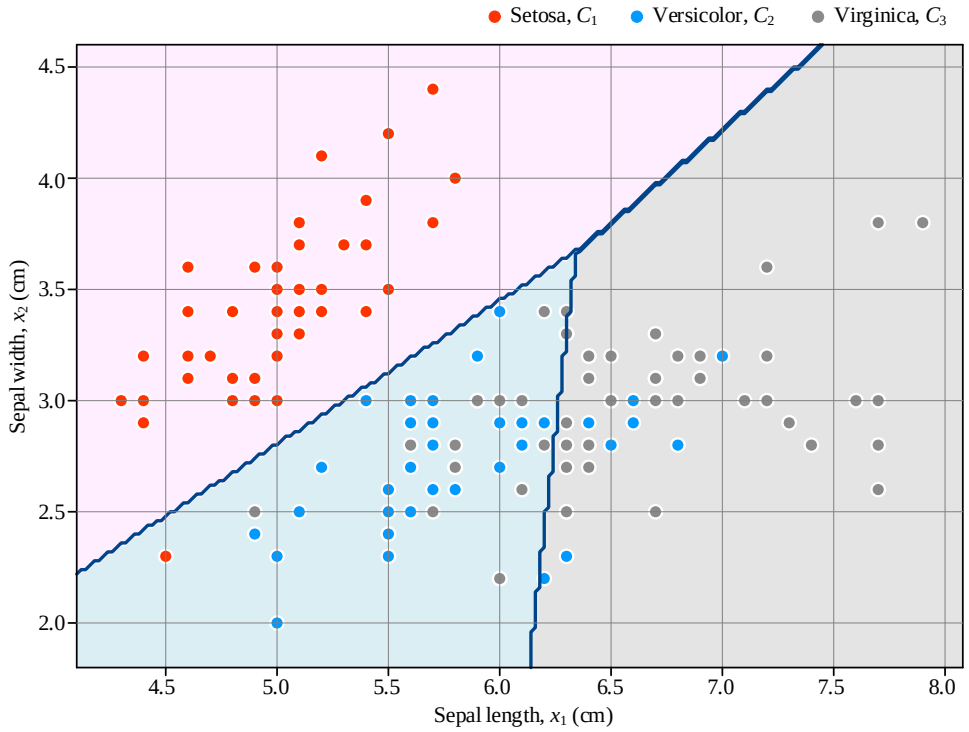


图 17. 线性判别分析分类鸢尾花

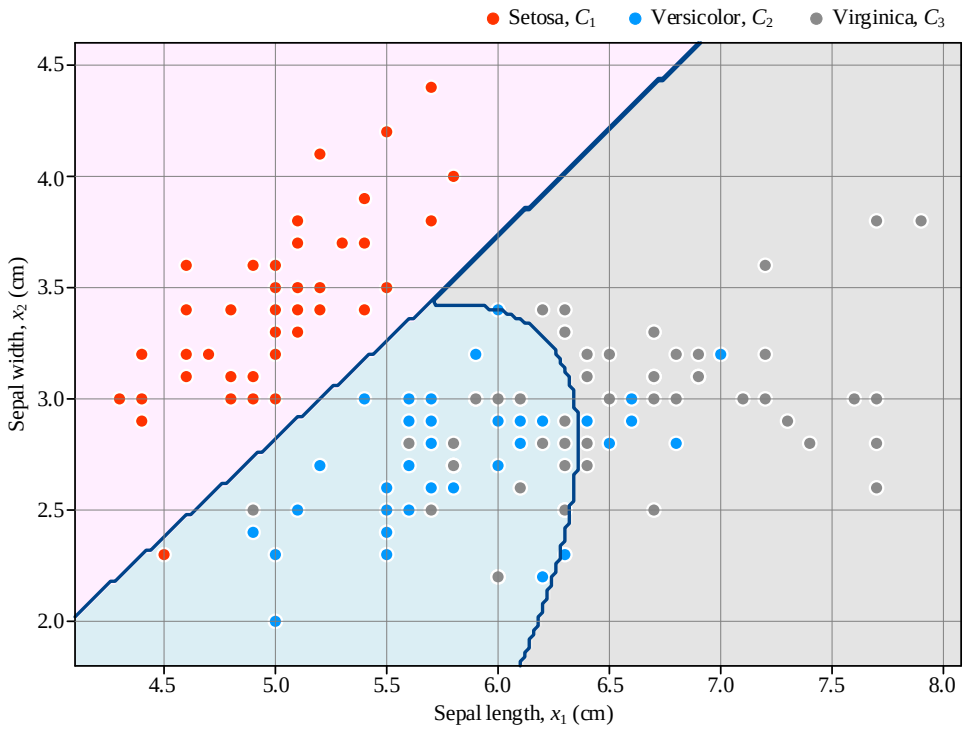
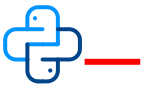
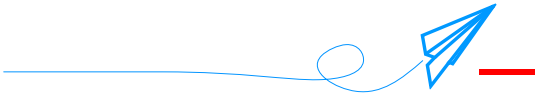


图 18. 二次判别分析分类鸢尾花



代码 Bk7_Ch06_03.py 利用判别分析分类器分类鸢尾花数据，并绘制图 17 和图 18。



高斯判别分析 GDA，一次判别分析 LDA、二次判别分析 QDA 是常见的监督学习算法，用于分类和判别问题。它们都基于不同的假设和数学模型。LDA、QDA 相当于 GDA 的特殊形式。

GDA 假设每个类别的数据都服从高斯分布，然后估计每个类别的均值和协方差矩阵来建模高斯分布，最后使用这些参数计算后验概率进行分类。

LDA 假设每个类别的数据都满足高斯分布，并且不同类别之间的协方差矩阵相等。然后，它寻找一个投影方向，可以将数据从高维空间投影到低维空间，并最大程度地保留不同类别之间的差异，同时最小化同一类别内部的方差。

QDA 假设每个类别的数据都满足二次高斯分布，即每个类别的协方差矩阵都不相等。然后，它估计每个类别的均值和协方差矩阵，最后使用这些参数计算后验概率进行分类。

建议大家研究下面这个官方示例。示例代码生成随机数，进行 LDA 和 QDA 分析，并绘制旋转椭圆来表达数据的协方差矩阵形状。

https://scikit-learn.org/stable/auto_examples/classification/plot_lda_qda.html

7 支持向量机

Support Vector Machine

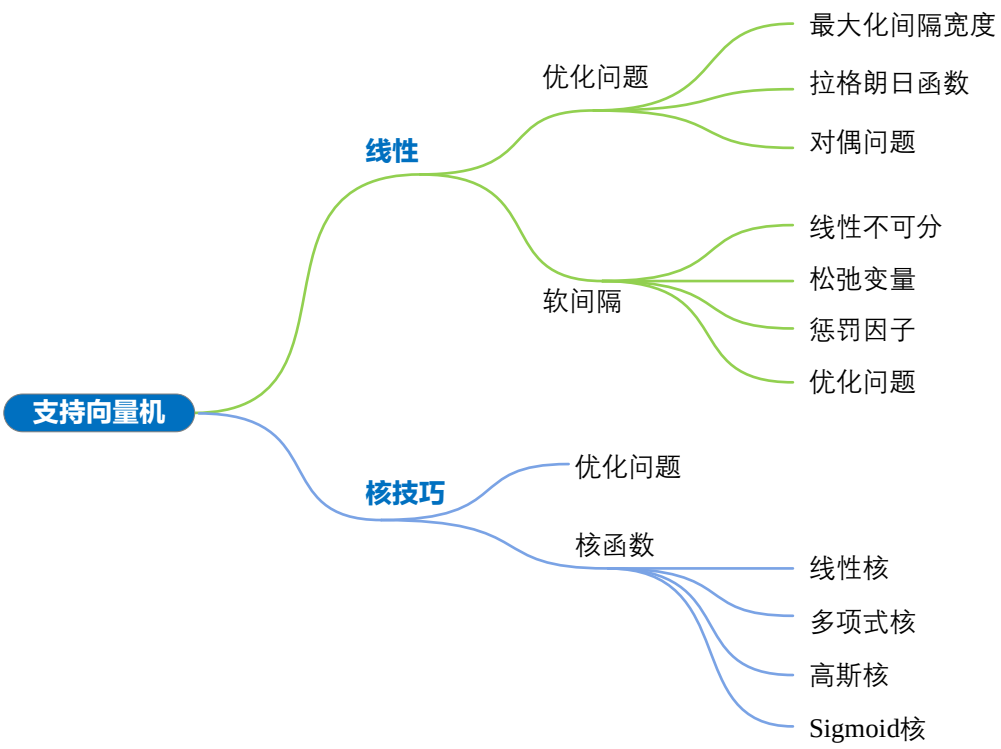
间隔最大化，支持向量确定决策边界

没有什么比精巧理论更实用的了。

Nothing is more practical than a good theory.

—— 弗拉基米尔·万普尼克 (Vladimir Vapnik) | 俄罗斯统计学家、数学家 | 1936 ~

- `numpy.hstack()` 水平方向将数组堆叠起来
- `numpy.vstack()` 竖直方向将数组堆叠起来
- `sklearn.svm.SVC` 支持向量机算法函数



7.1 支持向量机

弗拉基米尔 · 万普尼克 (Vladimir Vapnik) 和他的同事们发明并且完善了**支持向量机** (Support Vector Machine, SVM)。支持向量机 SVM 是一种用于分类和回归问题的监督学习算法。SVM 的主要思想是找到一个可以将不同类别分隔开的最优超平面，该超平面具有最大间隔，即离最近的数据点的距离最大。超平面可以被认为是一个决策边界，可以用于预测新的未知数据点的类别。

在实践中，SVM 使用内积核函数将原始输入数据映射到高维空间，从而能够处理非线性问题。一些常见的内积核函数包括线性核函数，多项式核函数和径向基函数核函数。

SVM 是一个非常强大的算法，因为它可以处理高维空间和非线性问题，并且能够有效地避免过拟合。SVM 的缺点是它对于大型数据集的计算成本很高，以及内积核函数的选择和调整需要一定的经验和技巧。

弗拉基米尔·万普尼克为机器学习发展奠定了大量理论基础，大家有兴趣的话可以翻看他的作品——*The Nature of Statistical Learning Theory*。

弗拉基米尔·万普尼克 (Vladimir Vapnik) | 俄罗斯统计学家、数学家 | 1936 ~
支持向量机发明者之一。关键词： ● 支持向量机 ● 核技巧

原理

图 1 所示为支持向量机核心思路。如图 1 所示，一片湖面左右散布着蓝色 ● 红色 ● 礁石，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。寻找一条路径，让该路径通过的皮划艇宽度最大。很明显，图 1 (b) 中规划的路径好于图 1 (a)。

图 1 (b) 中加黑圈 ○ 的五个点，就是所谓的**支持向量** (support vector)。

图 1 中深蓝色线，便是**决策边界**，也称**分离超平面** (separating hyperplane)。本书为了统一称呼，下文都使用决策边界。特别提醒大家注意一点，加黑圈 ○ 支持向量确定决策边界位置；其他数据并没有起到任何作用。因此，SVM 对于数据特征数量远高于数据样本量的情况也有效。

图 1 中两条虚线之间宽度叫做**间隔** (margin)。正如，本章副标题所言，支持向量机的优化目标为——间隔最大化。

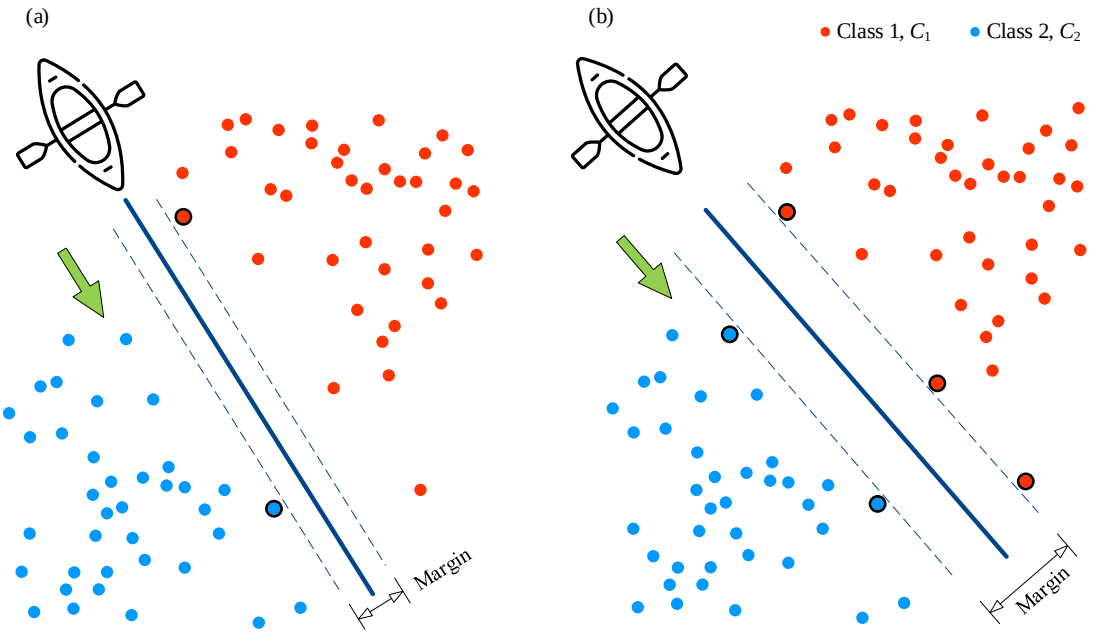


图 1. 支持向量机原理

线性可分、线性不可分

从数据角度，图 1 两类数据用一条直线便可以分割开来，这种数据叫做**线性可分** (linearly separable)。线性可分问题采用**硬间隔** (hard margin)；白话说，硬间隔指的是，间隔内没有数据点。

实践中，并不是所有数据都是线性可分。多数时候，数据**线性不可分** (non-linearly separable)。如图 2 所示，不能找到一条直线将蓝色 ● 红色 ● 数据分离。

对于线性不可分问题，就要引入两种方法——**软间隔** (soft margin) 和**核技巧** (kernel trick)。

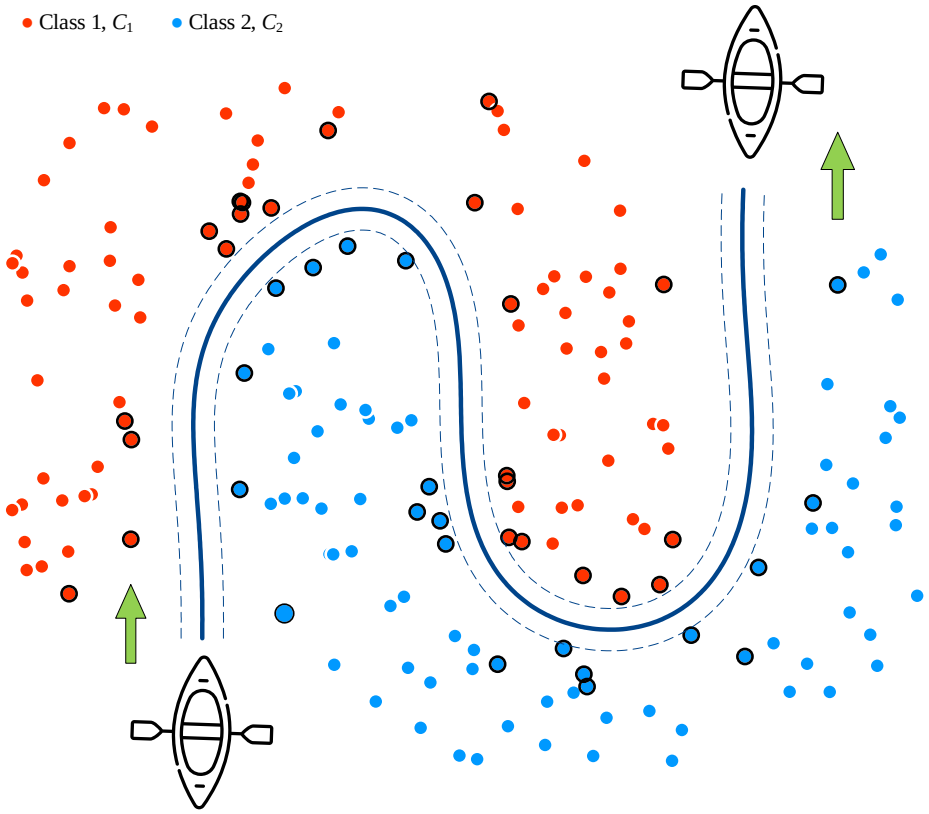


图 2. 线性不可分数据

软间隔

白话说，如图 3 所示，软间隔相当于一个**缓冲区** (buffer zone)。软间隔存在时，用决策边界分离数据时，有数据点侵入间隔，甚至超越间隔带。

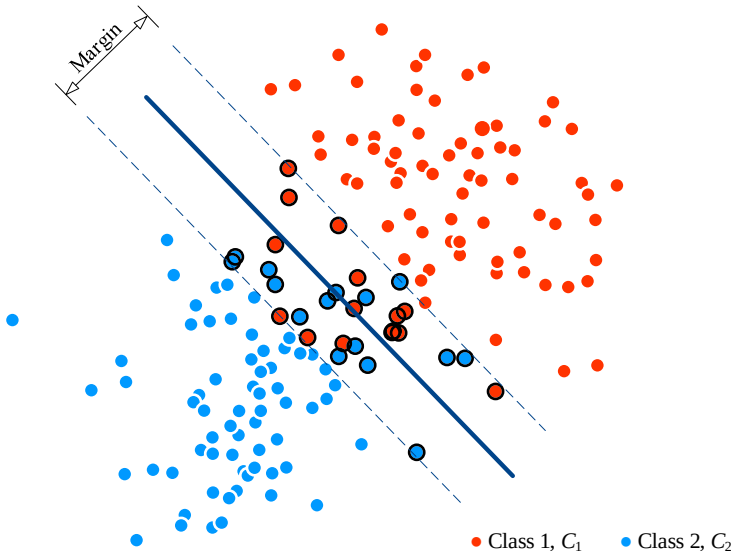


图 3. 软间隔

核技巧

核技巧将数据映射到高维特征空间，是一种数据升维。如图 4 所示，样本数据有两个特征，用平面可视化数据点位置。很明显图 4 给出的原始数据线性不可分。

采用核技巧，将图 4 二维数据，投射到三维核曲面上；很明显，在这个高维特征空间，容易找到某个水平面，将蓝色 ● 红色 ● 数据分离。利用核技巧，分离线性不可分数据变得更容易。

通常，采用支持向量机解决线性不可分问题，需要并用软间隔和核技巧。如图 5 所示，SVM 分类环形数据中，核技巧配合软间隔。

另外，支持向量机也可以用来处理回归问题，对应的方法为**支持向量回归** (Support Vector Regression, SVR)。本章将主要介绍硬间隔、支持向量和软间隔；下一章，将介绍核技巧。本章和下一章有一定比例的公式推导，这对理解支持向量机原理有帮助，希望大家耐心阅读。

《矩阵力量》第 19 章为本章提供大量数学工具，建议大家回顾。

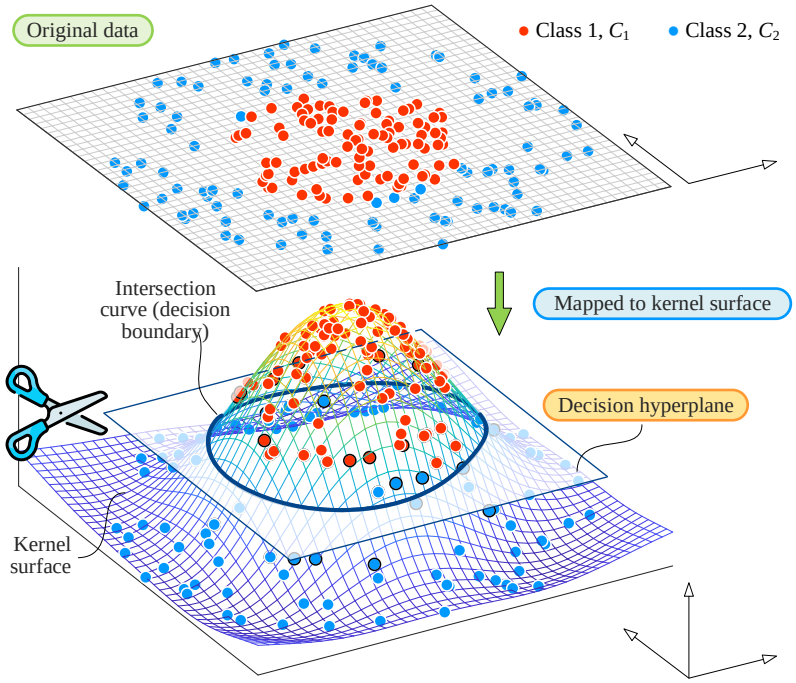


图 4. 核技巧原理

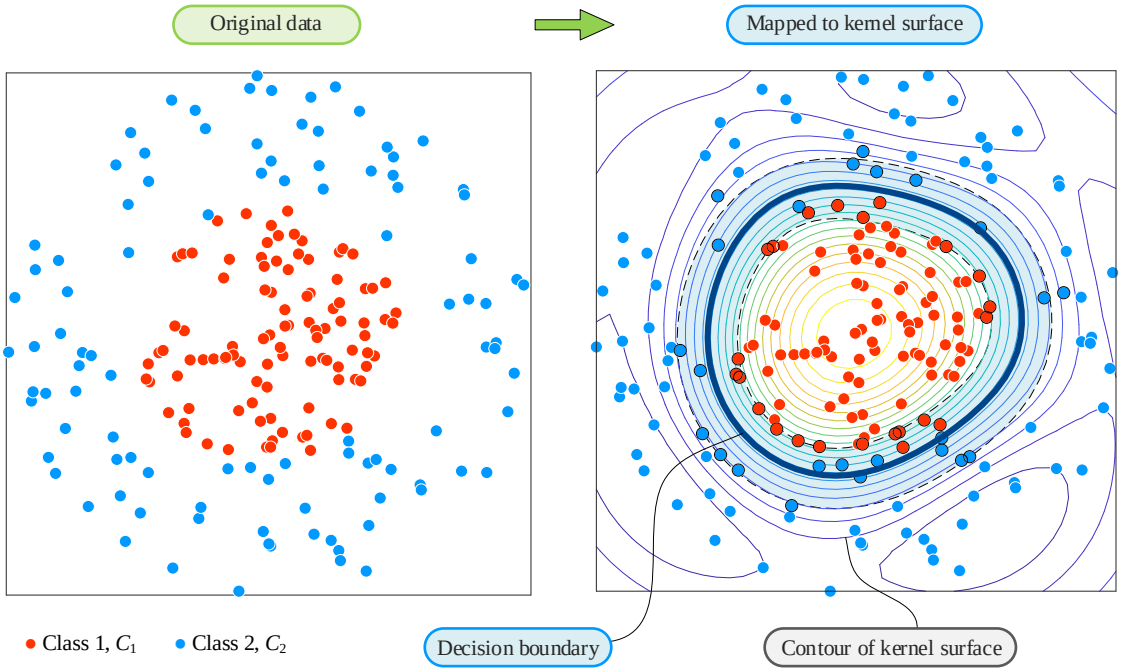


图 5. 核技巧配合软间隔

7.2 硬间隔：处理线性可分

支持向量机中硬间隔方法用来处理线性可分数据。利用《矩阵力量》一册讲解的向量几何知识，这一节将构造 SVM 中支持向量、决策边界、分类标签和间隔等元素之间的数学关系。

决策边界

如图 6 所示，决策边界定义如下：

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0 \tag{1}$$

其中，$\boldsymbol{w}$ 和 b 为模型参数；$\boldsymbol{w}$ 为 $f(\boldsymbol{w})$ 的梯度向量，形式为列向量。(1) 中，列向量 $\boldsymbol{w}$ 和 $\boldsymbol{x}$ 行数均为特征数 D。

很明显 (1) 为**超平面** (hyperplane)。注意，图 6 所示间隔宽度为 $2h$ ($h > 0$)。

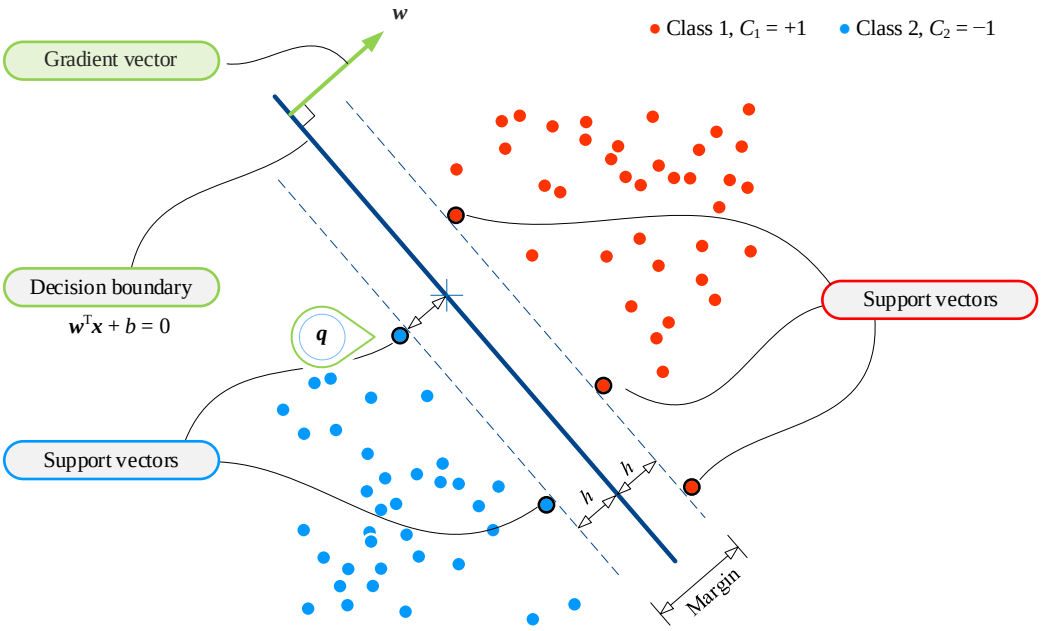


图 6. 硬间隔 SVM 处理二分类问题

(1) 可以展开为：

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b = 0 \tag{2}$$

特别地，对于 $D = 2$ 时，决策边界形式为：

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{3}$$

分类

对于二分类 ($K = 2$) 问题，决策边界“上方”的数据点满足：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b > 0 \tag{4}$$

展开 (4) 得到：

$$w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b > 0 \tag{5}$$

决策边界“下方”的数据点满足：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b < 0 \tag{6}$$

展开 (6) 得到：

$$w_1 x_1 + w_2 x_2 + \ldots + w_D x_D + b < 0 \tag{7}$$

准确地说，以 (1) 中 $f(\boldsymbol{x}) = 0$ 为基准，“上方”对应 $f(\boldsymbol{x}) > 0$；“下方”对应 $f(\boldsymbol{x}) < 0$。

决策函数

对任意查询点 $\boldsymbol{q}$，二分类决策函数 $p(\boldsymbol{q})$ 则可以表达为：

$$p(\boldsymbol{q}) = \text{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{q} + b\right) \tag{8}$$

其中，sign() 为**符号函数** (sign function)。

如图 6 所示，对于二分类 (K = 2) 问题，决策边界“上方”的数据点，预测分类为+1；决策边界“下方”的数据点，预测分类为−1。

支持向量到决策边界距离

图 6 中，某一支持向量坐标位置用列向量 $\boldsymbol{q}$ 表达。支持向量 $\boldsymbol{q}$ 到 (1) 对应的决策边界的距离为：

$$d = \frac{\left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{q} + b\right|}{\|\boldsymbol{w}\|} = \frac{\left|\boldsymbol{w}\cdot\boldsymbol{q} + b\right|}{\|\boldsymbol{w}\|} \tag{9}$$

对于上式陌生的读者，请回顾《矩阵力量》第 19 章第 6 节。

一般情况点线距离不考虑正负。但是，对于分类问题，考虑距离正负便于判断点和超平面关系。

(9) 分子去掉绝对值符号得到：

$$d = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{q} + b}{\|\boldsymbol{w}\|} = \frac{\boldsymbol{w}\cdot\boldsymbol{q} + b}{\|\boldsymbol{w}\|} \tag{10}$$

d 大于 0 时，点在超平面上方；d 小于 0 时，点在超平面下方。如图 7 所示，$\boldsymbol{q}_1$ 位于直线上方；而 $\boldsymbol{q}_2$ 位于直线下方。

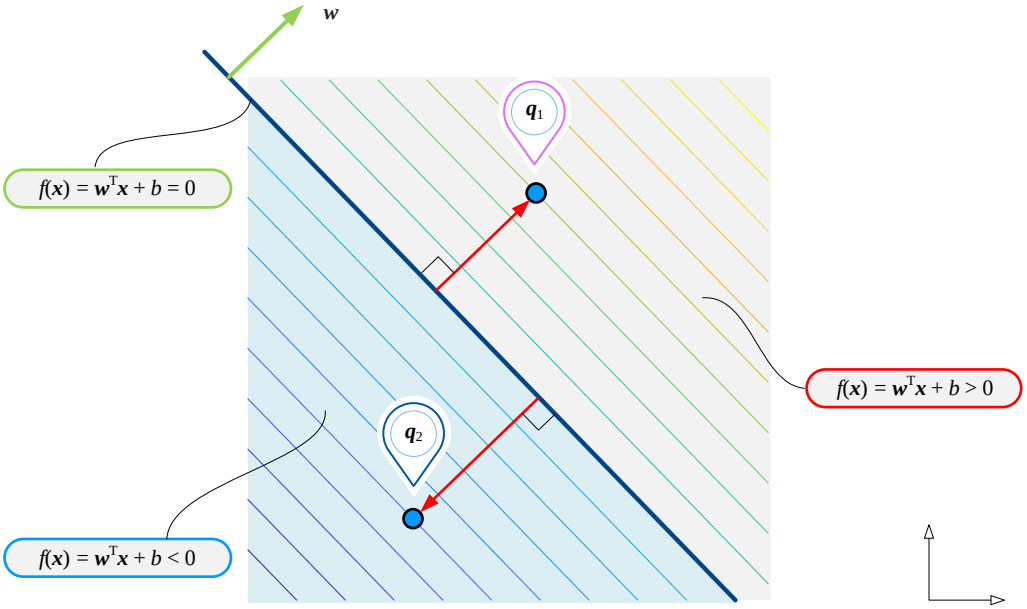


图 7. 直线外一点到直线距离，和平面外一点到平面距离

支持向量到硬间隔距离

如图 8 所示，硬间隔“下边界”为 l_1，l_1 到决策边界距离为$-h$。而支持向量 A、B 在 l_1 上，因此满足：

$$\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|}=-h \tag{11}$$

硬间隔“上边界”为 l_2，l_2 到决策边界为$+h$。支持向量 C 在 l_2 上，满足：

$$\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|}=+h \tag{12}$$

如图 8 所示，决策边界 (深蓝色线) 成功分离样本数据。距离决策边界大于等于 h 的样本点，标记为 $y=+1$；距离决策边界小于等于$-h$ 的样本点，标记为 $y=-1$，即：

$$\begin{cases}\dfrac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|}\geq+h, & y=+1\\ \dfrac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|}\leq-h, & y=-1\end{cases} \tag{13}$$

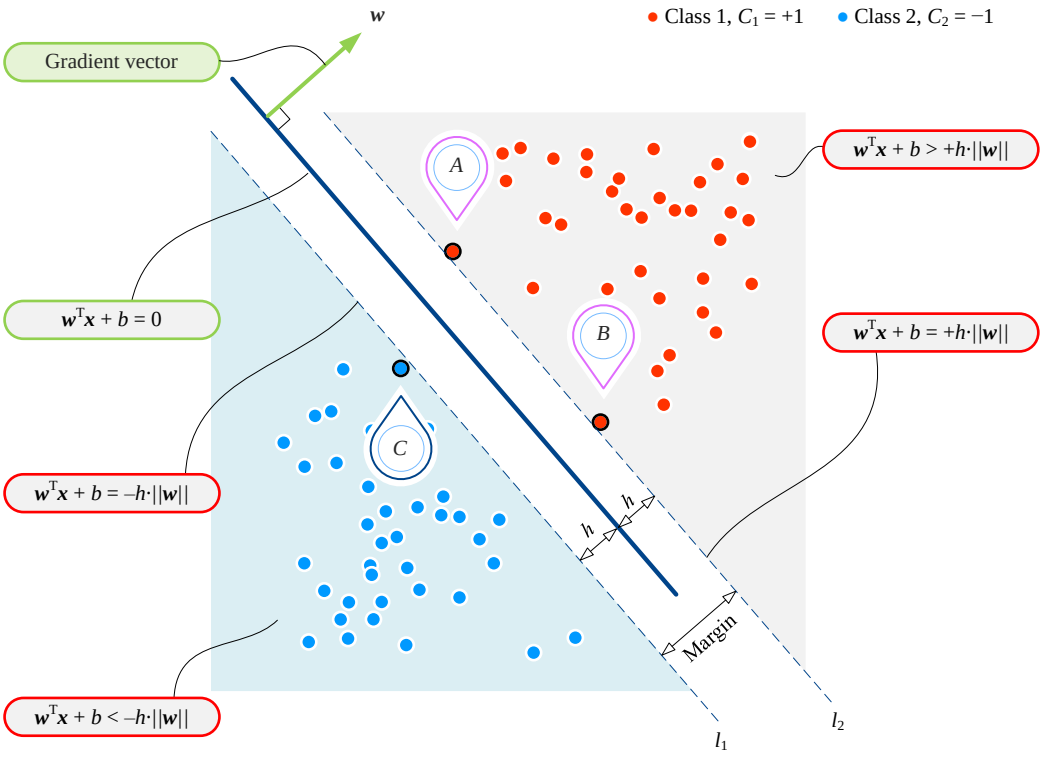


图 8. 硬间隔、决策边界和支持向量之间关系

整理 (13)，得到：

$$\begin{cases} \dfrac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|h} \geq +1, & y=+1 \\ \dfrac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}{\|\boldsymbol{w}\|h} \leq -1, & y=-1 \end{cases} \tag{14}$$

合并 (14) 两式可以得到：

$$\frac{\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)y}{\|\boldsymbol{w}\|h} \geq 1 \tag{15}$$

特别地，图 8 中三个支持向量点 A、B、C 满足下式：

$$\frac{\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)y}{\|\boldsymbol{w}\|h} = 1 \tag{16}$$

进一步简化运算

令：

$$\|\boldsymbol{w}\|h = 1 \tag{17}$$

(16) 可以简化为：

$$\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)y\geq 1 \tag{18}$$

利用内积来表达 (18)：

$$\left(\boldsymbol{w}\cdot\boldsymbol{x}+b\right)y\geq 1 \tag{19}$$

将 (17) 代入 (11) 和 (12)，可以得到间隔上下边界的解析式：

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=+1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=-1\end{cases} \tag{20}$$

根据 (18)，间隔宽度 $2h$ 可以用 $\boldsymbol{w}$ 表达：

$$2h=\frac{2}{\|\boldsymbol{w}\|} \tag{21}$$

7.3 构造优化问题

支持向量机的核心思想——最大化间隔。本节利用**拉格朗日乘子法** (method of Lagrange multipliers) 构造并求解支持向量机优化问题。本节内容相对来说“很不友好”，但是极其重要，建议大家耐心读完。

对拉格朗日乘子法感到陌生的话，请回顾《矩阵力量》第 18 章。

最大化间隔宽度

以 $\boldsymbol{w}$ 和 b 为优化变量，最大化 (21) 给出的间隔宽度：

$$\begin{aligned}&\arg\max_{\boldsymbol{w},b}\ \frac{2}{\|\boldsymbol{w}\|}\\ &\text{subject to}\ \left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)y^{(i)}\geq 1,\quad i=1,2,3,...,n\end{aligned} \tag{22}$$

其中，i 为样本数据点序数，i = 1,2, …, n。n 为样本数据数量。

最小化问题

(22) 等价于如下最小化问题：

$$\begin{aligned}&\arg\min_{\boldsymbol{w},b}\ \frac{\|\boldsymbol{w}\|^2}{2}=\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}{2}=\frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2}\\ &\text{subject to}\ \left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)y^{(i)}\geq 1,\quad i=1,2,3,...,n\end{aligned} \tag{23}$$

拉格朗日函数

构造**拉格朗日函数** (Lagrangian function) $L(\boldsymbol{w}, b, \boldsymbol{\lambda})$：

$$L(\boldsymbol{w},b,\boldsymbol{\lambda}) = \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2} + \sum_{i=1}^{n}\lambda_i\left(1-y^{(i)}\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)\right) \tag{24}$$

其中，$\boldsymbol{\lambda}$ 为拉格朗日乘子构造的列向量：

$$\boldsymbol{\lambda} = \begin{bmatrix}\lambda_1 & \lambda_2 & \cdots & \lambda_n\end{bmatrix}^{\mathrm{T}} \tag{25}$$

这样含不等式约束优化问题，转化为一个无约束优化问题。

偏导

$L(\boldsymbol{w}, b, \boldsymbol{\lambda})$ 对 $\boldsymbol{w}$ 和 b 偏导为 0，得到如下一系列等式：

$$\begin{cases}\dfrac{\partial L(\boldsymbol{w},b,\boldsymbol{\lambda})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)\mathrm{T}} = \boldsymbol{0} \\ \dfrac{\partial L(\boldsymbol{w},b,\boldsymbol{\lambda})}{\partial b} = \sum_{i=1}^{n}\lambda_i y^{(i)} = 0\end{cases} \tag{26}$$

这部分内容用到了《矩阵力量》第 17 章介绍的多元微分相关数学工具。

整理 (26) 可以得到：

$$\begin{cases}\boldsymbol{w} = \sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)\mathrm{T}} \\ \sum_{i=1}^{n}\lambda_i y^{(i)} = 0\end{cases} \tag{27}$$

注意，$\boldsymbol{w}$ 为列向量，而 $\boldsymbol{x}^{(i)}$ 为行向量。

简化拉格朗日函数

将上 (27) 带入 (24)，消去式中 $\boldsymbol{w}$ 和 b：

$$\begin{aligned}L(\boldsymbol{w},b,\boldsymbol{\lambda}) &= \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}}{2} + \sum_{i=1}^{n}\lambda_i\left(1-y^{(i)}\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)\right) \\ &= \frac{\left(\sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)}\right)^{\mathrm{T}}\left(\sum_{j=1}^{n}\lambda_j y^{(j)}\boldsymbol{x}^{(j)}\right)}{2} + \sum_{i=1}^{n}\lambda_i\left(1-y^{(i)}\left(\sum_{j=1}^{n}\lambda_j y^{(j)}\boldsymbol{x}^{(j)}\right)\cdot\boldsymbol{x}^{(i)} - y^{(i)}b\right) \\ &= \frac{\sum_{j=1}^{n}\sum_{i=1}^{n}\lambda_i\lambda_j y^{(i)}y^{(j)}\left(\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}^{(j)}\right)}{2} - \sum_{j=1}^{n}\sum_{i=1}^{n}\lambda_i\lambda_j y^{(i)}y^{(j)}\left(\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}^{(j)}\right) + \sum_{i=1}^{n}\lambda_i - b\sum_{i=1}^{n}\lambda_i y^{(i)} \\ &= \sum_{i=1}^{n}\lambda_i - \frac{\sum_{j=1}^{n}\sum_{i=1}^{n}\lambda_i\lambda_j y^{(i)}y^{(j)}\left(\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}^{(j)}\right)}{2}\end{aligned} \tag{28}$$

拉格朗日函数 $L(\boldsymbol{w}, b, \boldsymbol{\lambda})$ 简化为 $L(\boldsymbol{\lambda})$：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{\sum_{j=1}^{n}\sum_{i=1}^{n} \lambda_i \lambda_j y^{(i)} y^{(j)} \left(\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} \right)}{2} \tag{29}$$

对偶问题

利用拉格朗日乘子法，这样便将 (23) 优化问题转化成一个以 $\boldsymbol{\lambda}$ 为变量的优化问题：

$$\begin{aligned} &\arg\min_{\boldsymbol{\lambda}} \quad \sum_{i=1}^{n} \lambda_i - \frac{\sum_{j=1}^{n}\sum_{i=1}^{n} \lambda_i \lambda_j y^{(i)} y^{(j)} \left(\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)} \right)}{2} \\ &\text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \lambda_i y^{(i)} = 0 \\ \lambda_i \geq 0, \quad i, j = 1, 2, 3, ..., n \end{cases} \end{aligned} \tag{30}$$

这个优化问题常被称作**拉格朗日对偶问题** (Lagrange duality)，也称**对偶问题** (duality)。

发现二次型、格拉姆矩阵

大家是否发现 (29) 中的二次型？

对二次型陌生的读者，请回顾《矩阵力量》第 5 章。

举个例子，当 $n = 2$，即两个样本数据，(29) 可以展开为：

$$L(\boldsymbol{\lambda}) = (\lambda_1 + \lambda_2) - \frac{1}{2}\left(\lambda_1 \lambda_1 y^{(1)} y^{(1)} \left(\boldsymbol{x}^{(1)} \cdot \boldsymbol{x}^{(1)} \right) + 2\lambda_1 \lambda_2 y^{(1)} y^{(2)} \left(\boldsymbol{x}^{(1)} \cdot \boldsymbol{x}^{(2)} \right) + \lambda_2 \lambda_2 y^{(2)} y^{(2)} \left(\boldsymbol{x}^{(2)} \cdot \boldsymbol{x}^{(2)} \right) \right) \tag{31}$$

(31) 整理为如下二次型：

$$L(\boldsymbol{\lambda}) = (\lambda_1 + \lambda_2) - \frac{1}{2} \begin{bmatrix} \lambda_1 y^{(1)} & \lambda_2 y^{(2)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{(1)} \cdot \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(1)} \cdot \boldsymbol{x}^{(2)} \\ \boldsymbol{x}^{(2)} \cdot \boldsymbol{x}^{(1)} & \boldsymbol{x}^{(2)} \cdot \boldsymbol{x}^{(2)} \end{bmatrix} \begin{bmatrix} \lambda_1 y^{(1)} \\ \lambda_2 y^{(2)} \end{bmatrix} \tag{32}$$

类似地，(29) 可以整理为：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \begin{bmatrix} \lambda_1 y^{(1)} \\ \lambda_2 y^{(2)} \\ \vdots \\ \lambda_n y^{(n)} \end{bmatrix}^{\mathrm{T}} \underbrace{\begin{bmatrix} \left\langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)} \right\rangle \\ \left\langle \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(n)} \right\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \left\langle \boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(n)}, \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)} \right\rangle \end{bmatrix}}_{\text{Gram matrix}} \begin{bmatrix} \lambda_1 y^{(1)} \\ \lambda_2 y^{(2)} \\ \vdots \\ \lambda_n y^{(n)} \end{bmatrix} \tag{33}$$

相信大家已经在上式中看到了久违的**格拉姆矩阵** (Gram matrix)！

决策边界

利用 (27)，决策边界可以整理为：

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = \underbrace{\left(\sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)}\right)}_{\text{Coefficients}}\boldsymbol{x} + b = 0 \tag{34}$$

需要大家注意区分，行向量 $\boldsymbol{x}^{(i)}$ 为第 i 个数据点，$\boldsymbol{x}$ 为未知量构成的列向量。也就是说，$\sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)}$ 求和结果为行向量。

分类决策函数 $p(\boldsymbol{x})$ 则为：

$$p(\boldsymbol{x}) = \operatorname{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) = \operatorname{sign}\left(\underbrace{\left(\sum_{i=1}^{n}\lambda_i y^{(i)}\boldsymbol{x}^{(i)}\right)}_{\text{Coefficients}}\boldsymbol{x} + b\right) \tag{35}$$

7.4 支持向量机处理二分类问题

本节利用具体实例介绍如何实现硬间隔支持向量机算法。

实例

图 9 所示为 20 个样本数据，容易发现样本数据线性可分，下面利用支持向量机进行预测分类。

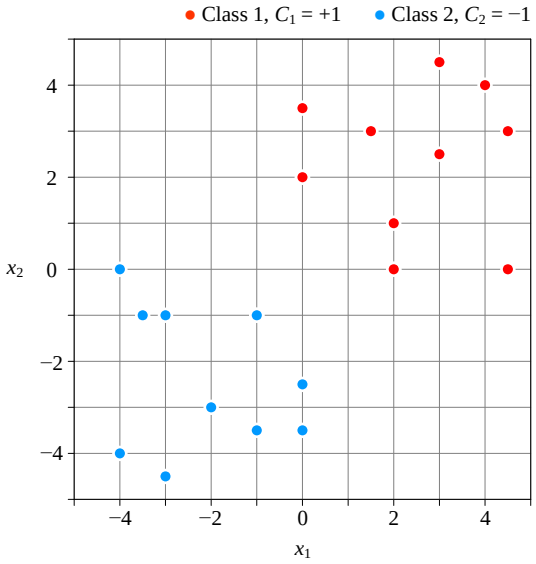


图 9. 20 个样本数据点平面位置

决策边界

对于 $D = 2$ 的情况，将 (1) 展开：

$$w_1 x_1 + w_2 x_2 + b = 0 \tag{36}$$

w_2 不等于 0 时，将 (36) 写成大家熟悉的一次函数形式：

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2} \tag{37}$$

硬间隔

根据 (20)，硬间隔“上边界”l_1 对应的函数为：

$$w_1 x_1 + w_2 x_2 + b = 1 \quad \Rightarrow \quad x_2 = -\frac{w_1}{w_2} x_1 - \frac{b-1}{w_2} \tag{38}$$

间隔“下边界” l_2 对应的函数为：

$$w_1 x_1 + w_2 x_2 + b = -1 \quad \Rightarrow \quad x_2 = -\frac{w_1}{w_2} x_1 - \frac{b+1}{w_2} \tag{39}$$

再次注意，因为 (37) 中 w_2 不能为 0，因此 (37) 存在局限性。这种表达方式仅为方便大家理解。

分类结果

图 10 为分类结果。容易发现，一共存在三个支持向量——A (0, 2)、B (2, 0) 和 C (−1, −1)。剩余 17 个样本数据对决策边界没有丝毫影响。

图 10 中深蓝色直线为决策边界，对应解析式：

$$\frac{x_1}{2} + \frac{x_2}{2} = 0 \quad \Rightarrow \quad x_1 + x_2 = 0 \quad \Rightarrow \quad x_2 = -x_1 \tag{40}$$

分类决策函数 $p(\boldsymbol{x})$ 为：

$$p(x_1, x_2) = \text{sign}(x_1 + x_2) \tag{41}$$

间隔“上”边界 l_1 对应的函数为：

$$\frac{x_1}{2} + \frac{x_2}{2} = 1 \quad \Rightarrow \quad x_2 = -x_1 + 2 \tag{42}$$

间隔“下”边界 l_2 对应的函数为：

$$\frac{x_1}{2}+\frac{x_2}{2}=-1 \quad \Rightarrow \quad x_2=-x_1-2 \tag{43}$$

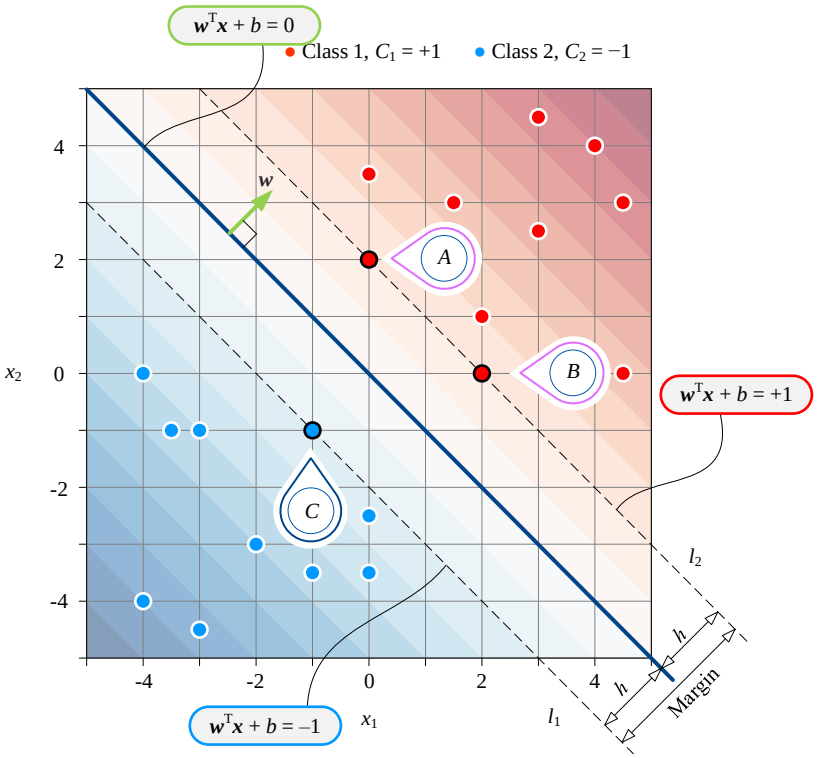


图 10. 硬间隔分类结果

预测分类

将 (4, 4) 代入 (41)，可以判断 (4, 4) 的预测分类为+1：

$$p(4,4)=\text{sign}(4+4)=+1 \tag{44}$$

将 (−2, −3) 代入 (41)，可以判断 (−2, −3) 的预测分类为−1：

$$p(-2,-3)=\text{sign}(-2-3)=-1 \tag{45}$$

将 (3, −3) 代入 (41)，结果为 0，可以判断 (3, −3) 位于决策边界上：

$$p(3,-3)=\text{sign}(3-3)=0 \tag{46}$$

支持向量影响决策边界

图 11 所示为删除点 A 后，支持向量变化，以及决策边界和间隔位置。再次强调，支持向量算法中，除支持向量之外的样本数据对决策边界没有影响。

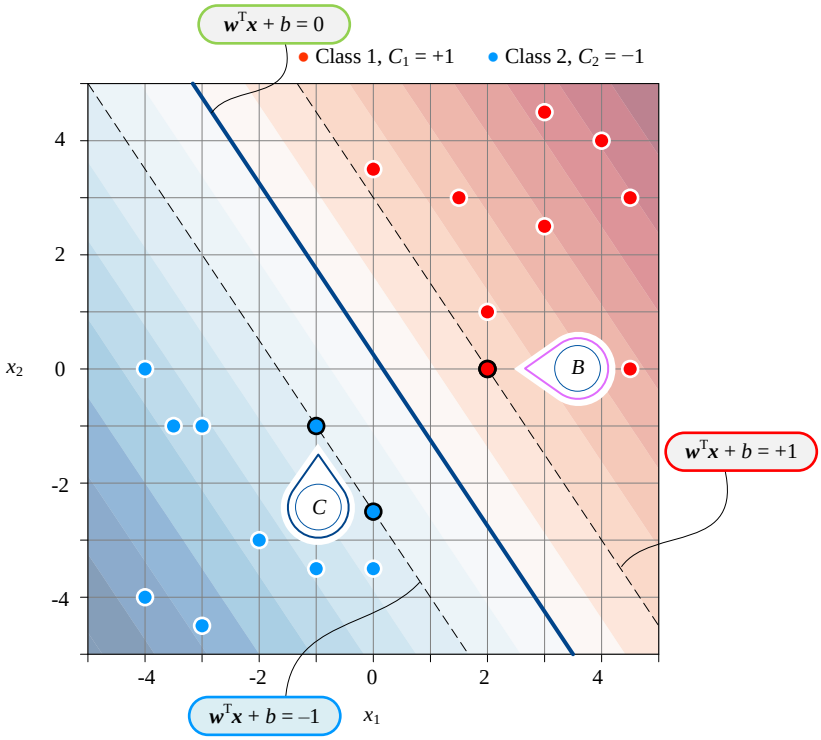


图 11. 删除点 A 后硬间隔 SVM 分类结果

7.5 软间隔：处理线性不可分

本章第一节提到，支持向量机可以采用**软间隔** (soft margin) 处理**线性不可分** (non-linearly separable data)。白话说，**硬间隔** (hard margin) 处理“泾渭分明”的分类数据，一条直线将样本数据彻底分离，如图 12 (a) 所示。而软间隔处理的数据呈现“你中有我，我中有你”，如图 12 (b) 所示。

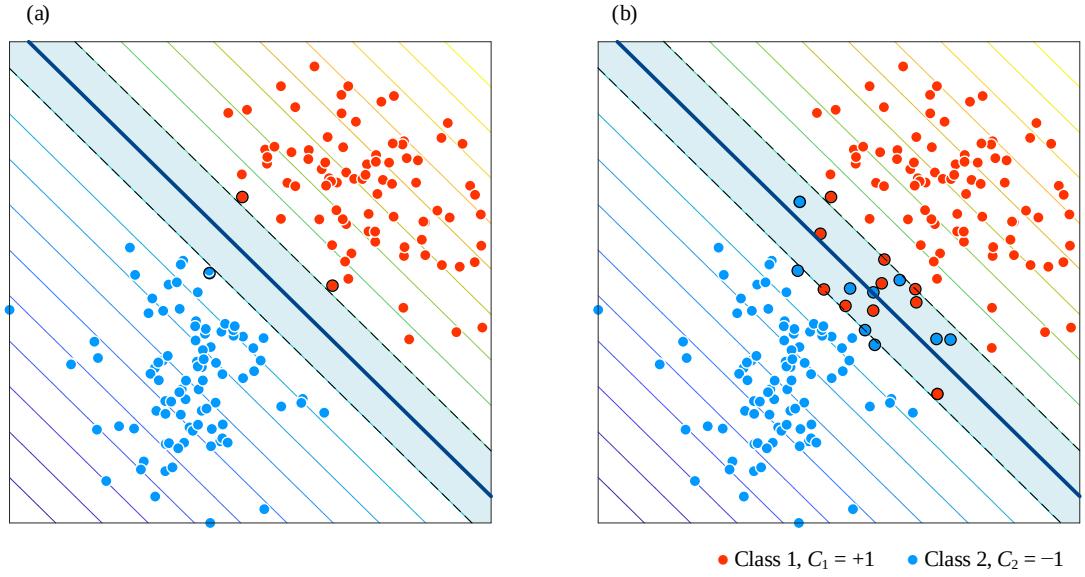


图 12. 比较硬间隔和软间隔

软间隔 SVM 方法的核心思想是牺牲部分数据点分类准确性，来换取更宽的间隔。

软间隔有两个重要参数：

- **松弛变量** (slack variable) ξ，一般读作 /ksaɪ/
- **惩罚因子** (penalty parameter) C

松弛变量

松弛变量用来模糊间隔边界，图 13 所示为原理图。

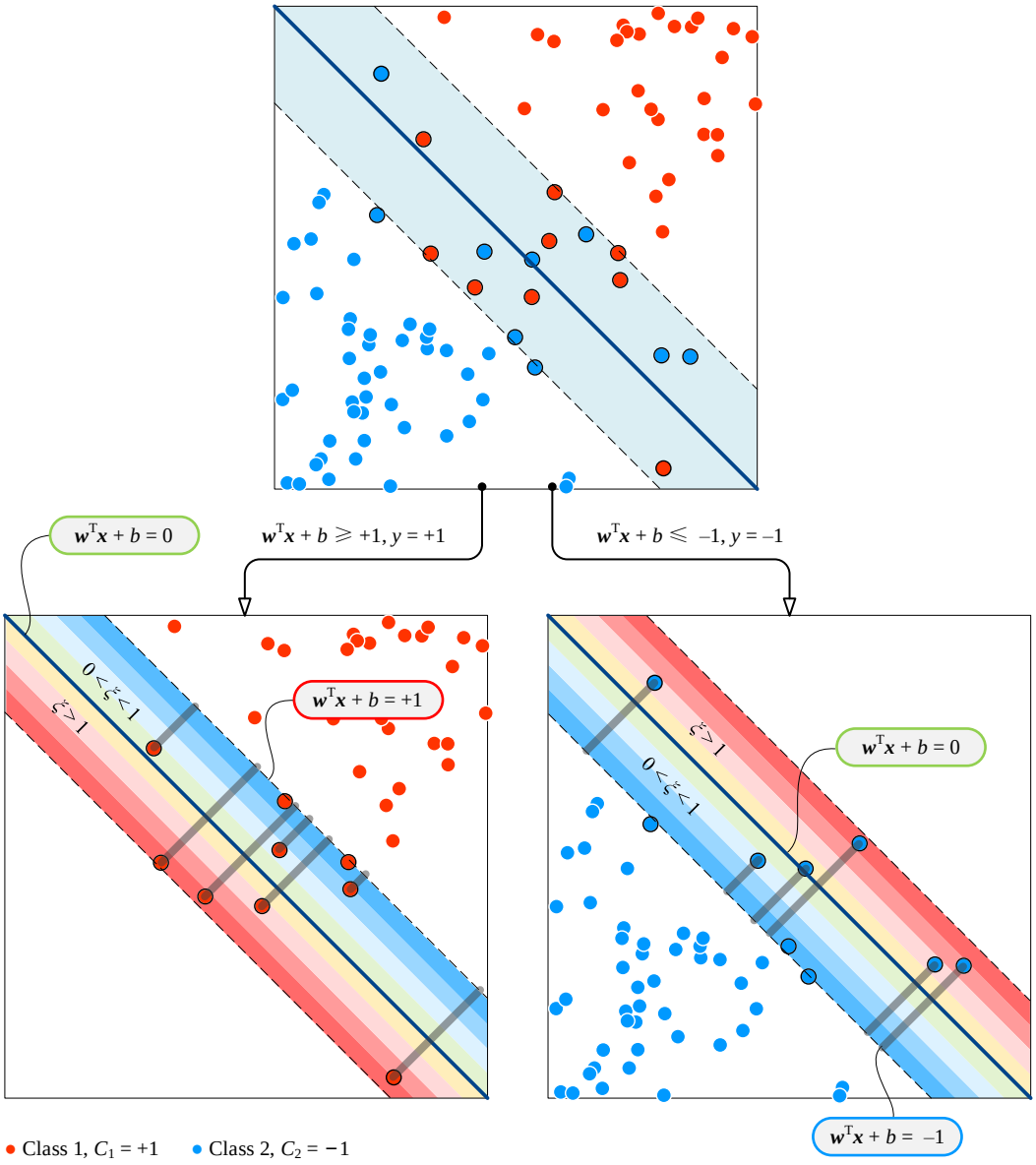


图 13. 软间隔中松弛变量作用

引入松弛变量 ξ，(19) 被改造为：

$$(\boldsymbol{w}\cdot\boldsymbol{x}+b)y\geq 1-\xi \tag{47}$$

当 $y=+1$，

$$(\boldsymbol{w}\cdot\boldsymbol{x}+b)\geq 1-\xi \tag{48}$$

当 $y=-1$

$$(\boldsymbol{w}\cdot\boldsymbol{x}+b)\leq -1+\xi \tag{49}$$

如图 13 所示，当 $\xi=0$，样本数据位于正确分类区域内或正确间隔边界上；当 $\xi>0$，样本数据位于软间隔范围之内，甚至在错误的分类区域内。图 13 中，红色带对应松弛变量 ξ 较大区域，蓝色带对应松弛变量 ξ 较小区域。

图 13 中，软间隔内任一数据点 $\boldsymbol{x}^{(i)}$ 距离各自边界距离为：

$$d_i=\frac{\xi_i}{\|\boldsymbol{w}\|} \tag{50}$$

优化问题

下面，在 (23) 基础上引入惩罚因子 C，构造软间隔 SVM 优化问题：

$$\begin{aligned}&\arg\min_{\boldsymbol{w},b,\xi}\ \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2}+C\sum_{i=1}^{n}\xi_i\\&\text{subject to}\ \begin{cases}y^{(i)}\left(\boldsymbol{x}^{(i)}\boldsymbol{w}+b\right)\geq 1-\xi_i,\quad i=1,2,3,...,n\\ \xi_i\geq 0\end{cases}\end{aligned} \tag{51}$$

惩罚因子 C 为用户设定参数，它调整松弛变量惩罚项的影响力。C 较大时，优化问题更在意分类准确性，牺牲间隔宽度；间隔可以窄一些，分类错误少犯一些。C 取值较小时，间隔更宽一些，间隔内的样本数据较多，分类错误可以多一点。

也可以采用 L^2 范数来构造松弛变量惩罚项，此时 (51) 被改造成：

$$\begin{aligned}&\arg\min_{\boldsymbol{w},b}\ \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2}+C\sum_{i=1}^{n}\xi_i^2\\&\text{subject to}\ \begin{cases}y^{(i)}\left(\boldsymbol{w}\cdot\boldsymbol{x}^{(i)}+b\right)\geq 1-\xi_i,\quad i=1,2,3,...,n\\ \xi_i\geq 0\end{cases}\end{aligned} \tag{52}$$

惩罚因子影响分类结果

图 14 所示为惩罚因子 C 取不同值时，支持变量、决策边界和间隔宽度变化。

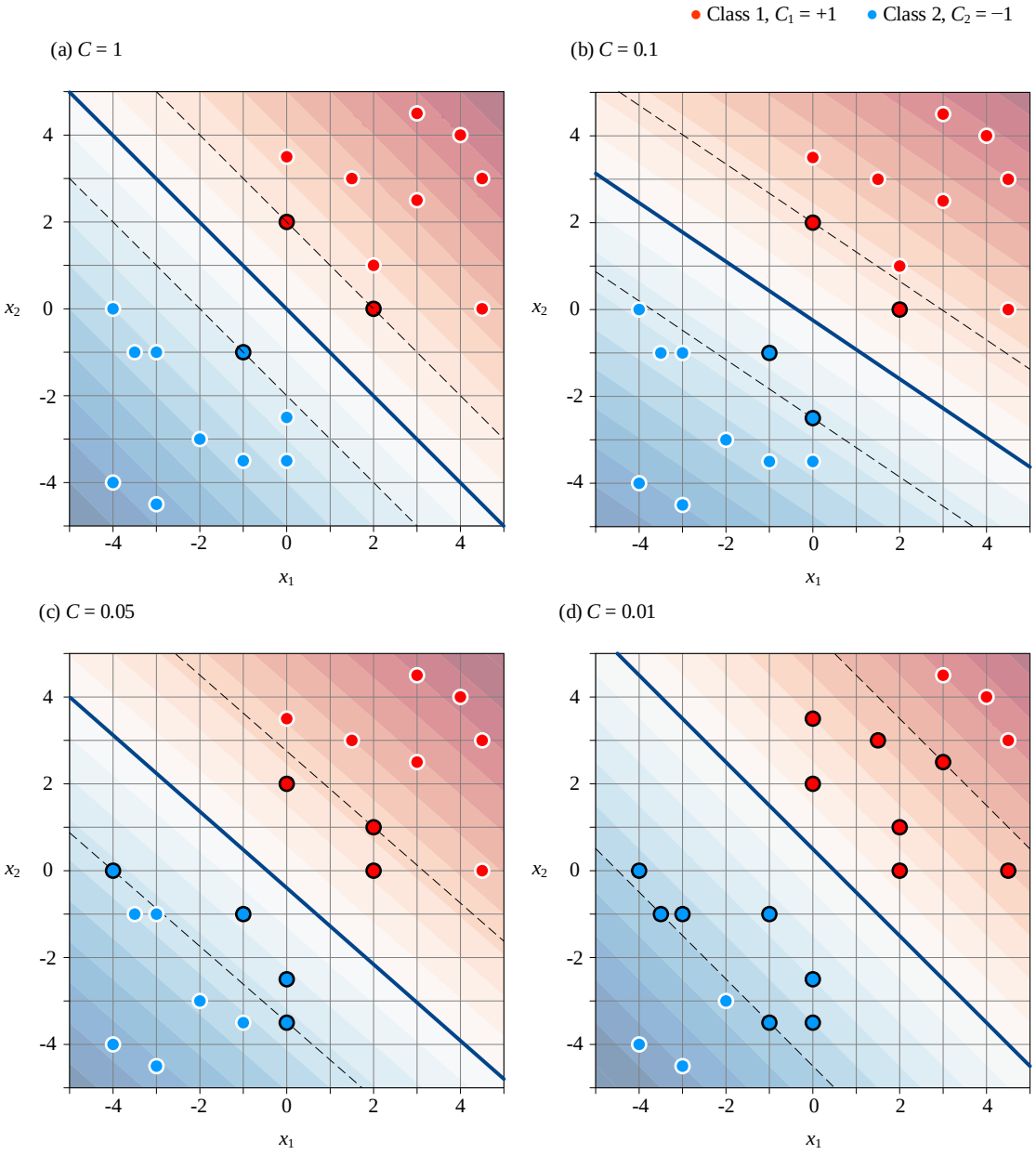
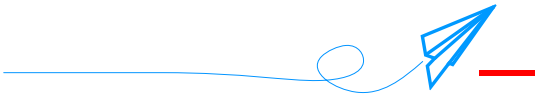


图 14. 惩罚因子对软间隔宽度和决策边界影响

代码 Bk7_Ch07_01.py 利用 SVM 实现分类，并绘制图 10、图 11 和图 14。



支持向量机目标是找到一个能够将两个类别线性分隔的最优超平面。SVM 通过优化一个约束条件下的目标函数来寻找最优超平面。优化问题分为硬间隔和软间隔两种情况，硬间隔要求数据能够完全被分隔，软间隔则允许一定程度的分类误差。优化目标函数可以转化为一个凸二次规划问题，可以通过拉格朗日乘子法来解决。

在实践中，SVM 使用核技巧将输入数据映射到高维空间，以便能够处理非线性问题。常用的核函数有线性核函数、多项式核函数和径向基函数核函数等。这种核技巧可以有效地提高 SVM 的性能和灵活性，因为它可以将低维输入空间中的非线性分类问题转化为高维空间中的线性分类问题。

SVM 是一种强大的分类模型，因为它可以处理高维空间和非线性问题，并且能够有效地避免过拟合。但是，SVM 的计算成本较高，选择和调整核函数也需要一定的经验和技巧。下一章将专门介绍 SVM 中的核函数。本章和下一章共用一个思维导图。

8 核技巧

Kernel Trick

将数据映射到高维特征空间

复杂理论，花拳绣腿；简单算法，立竿见影。

Complex theories do not work; simple algorithms do.

—— 弗拉基米尔·万普尼克 (Vladimir Vapnik) | 俄罗斯统计学家、数学家 | 1936 ~

- `sklearn.datasets.make_circles` 生成环形数据
- `sklearn.datasets.make_moons` 生成月牙形数据
- `sklearn.svm.SVC` 支持向量机函数

8.1 映射函数：实现升维

上一章简要介绍支持向量机**核技巧** (kernel trick) 原理——将样本数据映射到高维特征空间中，使数据在高维空间中线性可分。常用的核函数有线性核函数、多项式核函数和径向基函数核函数等。这种核技巧可以有效地提高 SVM 的性能和灵活性，因为它可以将低维输入空间中的非线性分类问题转化为高维空间中的线性分类问题。通过使用核技巧，SVM 能够处理非线性问题并且具有更好的泛化能力。

核技巧应用广泛，本章依托支持向量机，展开讲解核技巧及常用的几种核函数。

映射函数

首先，大家需要了解**映射函数** (mapping function) 这个概念。$\boldsymbol{x}$ 经过**特征映射** (feature map) 后得到 $\phi(\boldsymbol{x})$ 向量，$\phi()$ 叫映射函数。如图 1 所示，从 $\boldsymbol{x}$ 到 $\phi(\boldsymbol{x})$ 的过程便是一个升维过程。在原始数据特征空间线性不可分的数据，在新特征空间中变得线性可分。

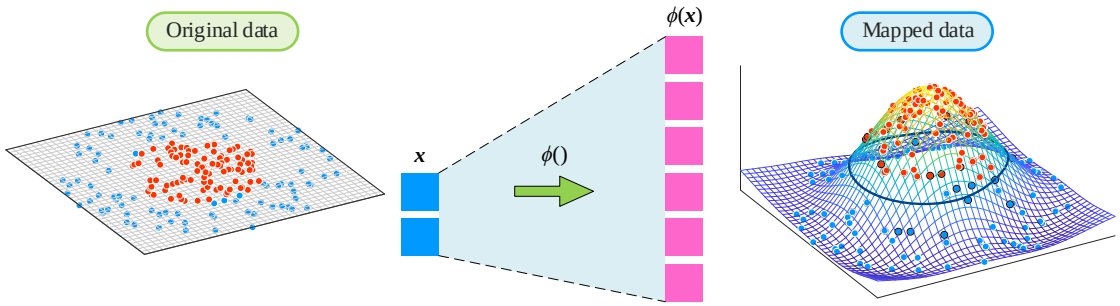


图 1. 映射原理示意图

从书前文一再提及**降维** (dimension reduction)，核技巧则采用升维解决分类问题；这一点，听着有点不可思议。下面举几个例子解释。

第一个例子

图 2 所示为两组单一特征数据。图 2 (a) 中，原始数据左右两侧各 4 个点标签为 ●；中间 9 个点标签为 ●。

在 x_1 这个单一维度上，样本数据不能直接线性分类；但是，经过类似二次函数映射后，在全新二维空间中，样本数据便很容易被分类。

$$\boldsymbol{x}=\begin{bmatrix}x_1\end{bmatrix}\xrightarrow{\phi(\,)}\phi(\boldsymbol{x})=\begin{bmatrix}x_1 & x_1^2\end{bmatrix}^{\mathrm{T}} \tag{1}$$

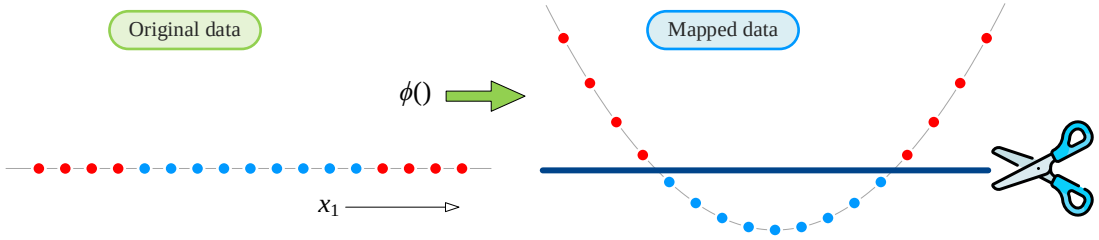


图 2. 核技巧，原始数据为单一特征

第二个例子

图 2 (a) 中原始数据标签交替出现。采用类似正弦函数映射到二维空间后，数据线性可分。

$$\boldsymbol{x}=\begin{bmatrix}x_1\end{bmatrix}\xrightarrow{\phi(\ )}\boldsymbol{\phi}(\boldsymbol{x})=\begin{bmatrix}x_1 & \sin(x_1)\end{bmatrix}^{\mathrm{T}} \tag{2}$$

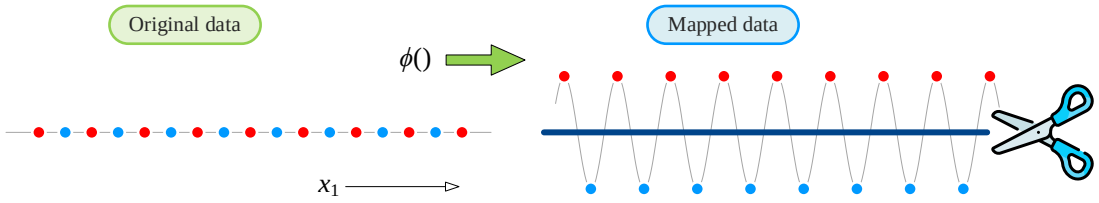


图 3. 核技巧，原始数据为单一特征

第三个例子

图 4 所示原始数据有两个特征，也是线性不可分。但是利用 XOR 函数映射之后，数据便容易分离。

$$\boldsymbol{x}=\begin{bmatrix}x_1 & x_2\end{bmatrix}^{\mathrm{T}}\xrightarrow{\phi(\ )}\boldsymbol{\phi}(\boldsymbol{x})=\begin{bmatrix}x_1 & x_2 & \mathrm{XOR}(x_1,x_2)\end{bmatrix}^{\mathrm{T}} \tag{3}$$

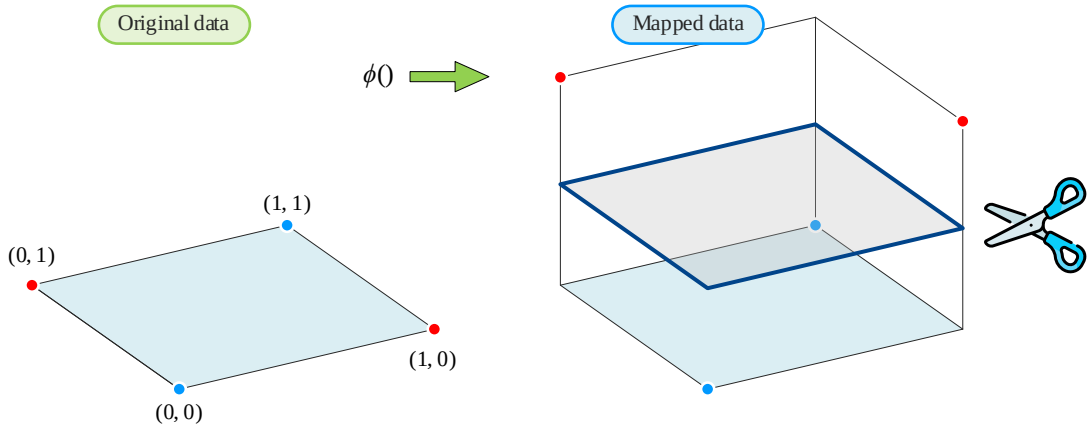


图 4. 核技巧，两特征数据

XOR 为**逻辑异或** (exclusive or) 函数，真值表如图 5。根据真值表，XOR(0, 1) = 1，XOR(1, 0) = 1；而，XOR(1, 1) = 0，XOR(0, 0) = 0。

		A	
	XOR(*A*, *B*)	False, 0	Truth, 1
B	False, 0	False, 0	Truth, 1
	Truth, 1	Truth, 1	False, 0

图 5. XOR 逻辑异或真值表

第四个例子

上一章已经展示过类似图 6 环形数据，这种数据线性不可分。但是按照如下规则映射，在新的特征空间中，数据变得线性可分。

$$\boldsymbol{x}=\begin{bmatrix}x_1 & x_2\end{bmatrix}^{\mathrm{T}} \xrightarrow{\phi(\,)} \phi(\boldsymbol{x})=\begin{bmatrix}x_1^2 & \sqrt{2}x_1x_2 & x_2^2\end{bmatrix}^{\mathrm{T}} \tag{4}$$

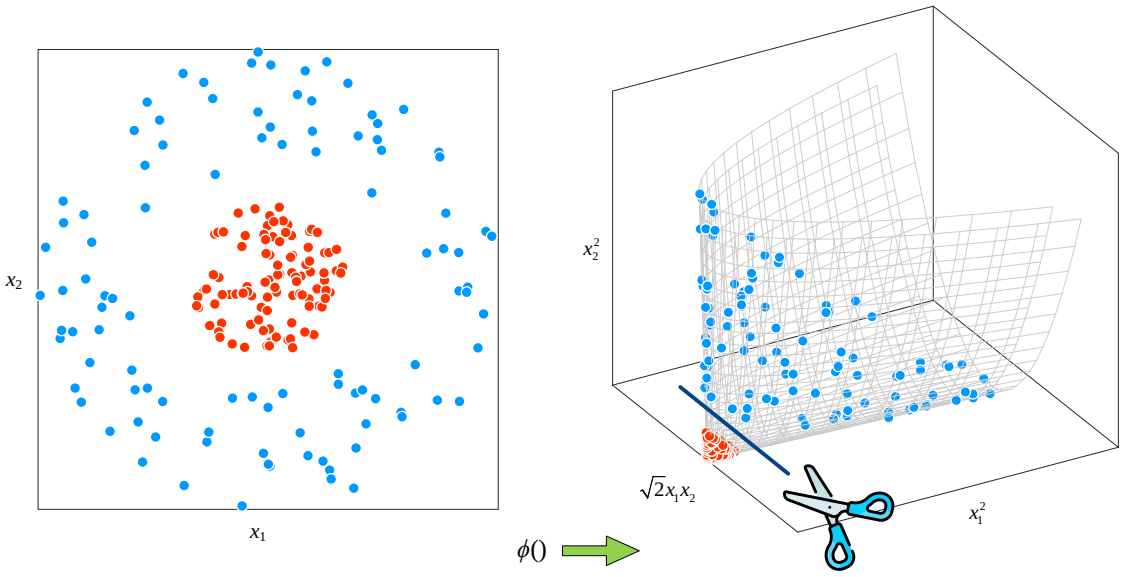


图 6. 环形数据映射到新特征空间，变得线性可分

改造支持向量机算法

通过上一章学习，我们知道 SVM 决策边界解析式为：

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0 \tag{5}$$

(5) 解析式代表超平面。

经过特征映射后的，决策边界如下：

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\underbrace{\phi(\boldsymbol{x})}_{\substack{\text{Mapping}\\ \text{function}}} + b = 0 \tag{6}$$

其中，$\boldsymbol{w}$ 和 b 为模型参数。决策边界具体为哪一类曲面，取决于 $\phi(\boldsymbol{x})$。注意，(5) 和 (6) 中向量 $\boldsymbol{w}$ 不同。(6) 中，列向量 $\boldsymbol{w}$ 和 $\phi(\boldsymbol{x})$ 行数一致。

一般情况，$\phi(\boldsymbol{x})$ 的特征数远多于原始数据 $\boldsymbol{x}$。因此，(6) 中，列向量 $\boldsymbol{w}$ 行数一般比 $\boldsymbol{x}$ 特征数多。极端情况，$\phi(\boldsymbol{x})$ 的特征数量可能为无穷。

需要大家格外注意的是，$\phi(\boldsymbol{x})$ 形式是并不重要!

有了映射函数的铺垫，下一节构造核技巧支持向量机优化问题。

8.2 核技巧 SVM 优化问题

在支持向量机中，核技巧将输入数据映射到高维空间中，使得原本的非线性问题转化为线性问题。这个转化的过程中，涉及到一个最优化问题，即要找到一个最优的决策函数，使得分类边界最优化。因此，通过核技巧转化后的 SVM 问题就是求解一个最优化问题。优化问题可以通过求解拉格朗日函数的最小值来解决。

优化问题

类似上一章硬间隔 SVM，构建核技巧 SVM 优化问题如下：

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} \quad \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2} \\ &\text{subject to } \; y^{(i)}\left(\boldsymbol{w}\cdot\underbrace{\phi\left(\boldsymbol{x}^{(i)}\right)}_{\substack{\text{Mapping}\\ \text{function}}} + b\right) \geq 1, \quad i = 1,2,3,\ldots,n \end{aligned} \tag{7}$$

拉格朗日函数

同样，构造拉格朗日函数 $L(\boldsymbol{w}, b, \boldsymbol{\lambda})$：

$$L(\boldsymbol{w},b,\boldsymbol{\lambda}) = \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2} + \sum_{i=1}^{n}\lambda_i\left(1 - y^{(i)}\left(\boldsymbol{w}\cdot\phi\left(\boldsymbol{x}^{(i)}\right) + b\right)\right) \tag{8}$$

偏导

$L(\boldsymbol{w}, b, \boldsymbol{\lambda})$ 对 $\boldsymbol{w}$ 和 b 偏导为 0，得到：

$$\begin{cases} \dfrac{\partial L(\boldsymbol{w},b,\boldsymbol{\lambda})}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \lambda_i y^{(i)} \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) = \boldsymbol{0} \\ \dfrac{\partial L(\boldsymbol{w},b,\boldsymbol{\lambda})}{\partial b} = \sum_{i=1}^{n} \lambda_i y^{(i)} = 0 \end{cases} \tag{9}$$

整理 (9) 得到：

$$\begin{cases} \boldsymbol{w} = \sum_{i=1}^{n} \lambda_i y^{(i)} \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) \\ \sum_{i=1}^{n} \lambda_i y^{(i)} = 0 \end{cases} \tag{10}$$

类似上一章推导过程，将 (10) 两式带入 (8)，消去 (8) 中 $\boldsymbol{w}$ 和 b：

$$\begin{aligned} L(\boldsymbol{w},b,\boldsymbol{\lambda}) &= \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2} + \sum_{i=1}^{n} \lambda_i \left(1 - y^{(i)}\left(\boldsymbol{w}\cdot\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right) + b\right)\right) \\ &= \sum_{i=1}^{n} \lambda_i - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i \lambda_j y^{(i)} y^{(j)} \overbrace{\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\cdot\boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right)}^{\text{Kernel function}}}{2} \end{aligned} \tag{11}$$

核函数

(11) 中 $\boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right)\cdot\boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)$ 中一项，可以记做：

$$\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right)\cdot\boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right) = \left\langle \boldsymbol{\phi}\left(\boldsymbol{x}^{(i)}\right), \boldsymbol{\phi}\left(\boldsymbol{x}^{(j)}\right)\right\rangle \tag{12}$$

其中，$\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right)$ 被称作**核函数** (kernel function)。白话说，核函数按照某种规则完成“向量 → 标量”运算。

根据内积运算原理，下式成立：

$$\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}\right) = \kappa\left(\boldsymbol{x}^{(j)}, \boldsymbol{x}^{(i)}\right) \tag{13}$$

简化拉格朗日函数

利用核函数记法，(11) 可以整理为 $L(\boldsymbol{\lambda})$：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n}\lambda_i - \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j y^{(i)}y^{(j)}\overbrace{\kappa\left(\boldsymbol{x}^{(i)},\boldsymbol{x}^{(j)}\right)}^{\text{Kernel}}}{2} \tag{14}$$

二次型、格拉姆矩阵

特别地，当 $n = 2$，即两个样本数据，(14) 可以展开为：

$$L(\boldsymbol{\lambda}) = (\lambda_1+\lambda_2) - \frac{1}{2}\left(\lambda_1\lambda_1 y^{(1)}y^{(1)}\kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(1)}\right) + 2\lambda_1\lambda_2 y^{(1)}y^{(2)}\kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}\right) + \lambda_2\lambda_2 y^{(2)}y^{(2)}\kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}\right)\right) \tag{15}$$

(15) 整理为如下二次型：

$$L(\boldsymbol{\lambda}) = (\lambda_1+\lambda_2) - \frac{1}{2}\begin{bmatrix}\lambda_1 y^{(1)} & \lambda_2 y^{(2)}\end{bmatrix}\begin{bmatrix}\kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}\right)\\ \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}\right)\end{bmatrix}\begin{bmatrix}\lambda_1 y^{(1)}\\ \lambda_2 y^{(2)}\end{bmatrix} \tag{16}$$

类似地，(14) 可以整理为：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\begin{bmatrix}\lambda_1 y^{(1)}\\ \lambda_2 y^{(2)}\\ \vdots\\ \lambda_n y^{(n)}\end{bmatrix}^{\mathrm{T}}\underbrace{\begin{bmatrix}\kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(n)}\right)\\ \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(n)}\right)\\ \vdots & \vdots & \ddots & \vdots\\ \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(n)}\right)\end{bmatrix}}_{\text{Gram matrix}}\begin{bmatrix}\lambda_1 y^{(1)}\\ \lambda_2 y^{(2)}\\ \vdots\\ \lambda_n y^{(n)}\end{bmatrix} \tag{17}$$

令**格拉姆矩阵** (Gram matrix) $\boldsymbol{K}$ 为：

$$\boldsymbol{K} = \underbrace{\begin{bmatrix}\kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(n)}\right)\\ \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(n)}\right)\\ \vdots & \vdots & \ddots & \vdots\\ \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(1)}\right) & \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(2)}\right) & \cdots & \kappa\left(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(n)}\right)\end{bmatrix}}_{\text{Gram matrix}} \tag{18}$$

(17) 可以整理为：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n}\lambda_i - \frac{1}{2}\begin{bmatrix}\lambda_1 y^{(1)}\\ \lambda_2 y^{(2)}\\ \vdots\\ \lambda_n y^{(n)}\end{bmatrix}^{\mathrm{T}}\boldsymbol{K}\begin{bmatrix}\lambda_1 y^{(1)}\\ \lambda_2 y^{(2)}\\ \vdots\\ \lambda_n y^{(n)}\end{bmatrix} \tag{19}$$

线性核：最简单的核函数

下式上一章获得 $L(\boldsymbol{\lambda})$ 函数：

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{\sum_{j=1}^{n}\sum_{i=1}^{n} \lambda_i \lambda_j y^{(i)} y^{(j)} \overbrace{\left(\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}\right)}^{\text{Linear kernel}}}{2} \tag{20}$$

对比 (14) 和 (20)，可以发现 $\left(\boldsymbol{x}^{(i)} \cdot \boldsymbol{x}^{(j)}\right)$ 实际上也是一种核函数——**线性核** (linear kernel) ——最简单的核函数。

对偶问题

至此，我们得到核技巧 SVM 优化问题的对偶问题：

$$\begin{aligned} &\arg\min_{\boldsymbol{\lambda}} \quad \sum_{i=1}^{n} \lambda_i - \frac{\sum_{j=1}^{n}\sum_{i=1}^{n} \lambda_i \lambda_j y^{(i)} y^{(j)} \overbrace{\phi\left(\boldsymbol{x}^{(i)}\right) \cdot \phi\left(\boldsymbol{x}^{(j)}\right)}^{\text{Kernel}}}{2} \\ &\text{subject to} \quad \begin{cases} \sum_{i=1}^{n} \lambda_i y^{(i)} = 0 \\ \lambda_i \geq 0, \quad i, j = 1, 2, 3, \ldots, n \end{cases} \end{aligned} \tag{21}$$

决策边界

整理得到核技巧 SVM 决策边界如下：

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w} \cdot \phi(\boldsymbol{x}) + b \\ &= \underbrace{\left(\sum_{i=1}^{n} \lambda_i y^{(i)} \phi\left(\boldsymbol{x}^{(i)}\right)\right)}_{\text{Coefficients}} \cdot \phi(\boldsymbol{x}) + b = 0 \end{aligned} \tag{22}$$

将 $\phi(\boldsymbol{x})$ 乘到求和符号 Σ 里，得到决策边界解析解为：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \left(\lambda_i y^{(i)} \underbrace{\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}\right)}_{\text{Kernel}} \right) + b = 0 \tag{23}$$

再次强调，$\boldsymbol{x}^{(i)}$ 代表一个已知样本点，而 $\boldsymbol{x}$ 代表未知量。注意，以上推导过程不再区分行、列向量。

使用核技巧，二分类决策函数 $p(\boldsymbol{x})$ 则可以表达为：

$$\begin{aligned} p(\boldsymbol{x}) &= \text{sign}\left(\boldsymbol{w} \cdot \phi(\boldsymbol{x}) + b\right) \\ &= \text{sign}\left(\sum_{i=1}^{n} \lambda_i y^{(i)} \underbrace{\kappa\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}\right)}_{\text{Kernel}} + b \right) \end{aligned} \tag{24}$$

对比线性核 SVM 分类决策函数：

$$\begin{aligned} p(\boldsymbol{x}) &= \operatorname{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+b) \\ &= \operatorname{sign}\left(\sum_{i=1}^{n}\lambda_i y^{(i)}\underbrace{\left(\boldsymbol{x}^{(i)}\boldsymbol{x}\right)}_{\text{Linear kernel}}+b\right) \end{aligned} \tag{25}$$

请读者注意，scikit-learn 中**决策函数** (decision function) 输出值指的是 (23) 结果。

软间隔 + 核技巧

引入惩罚因子 C，可以构造软间隔核技巧 SVM 优化问题：

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} \quad \frac{\boldsymbol{w}\cdot\boldsymbol{w}}{2}+C\sum_{i=1}^{n}\xi_i \\ &\text{subject to} \quad \begin{cases} y^{(i)}\left(\boldsymbol{w}\cdot\phi\left(\boldsymbol{x}^{(i)}\right)+b\right)\geq 1-\xi_i, \quad i=1,2,3,\ldots,n \\ \xi_i\geq 0 \end{cases} \end{aligned} \tag{26}$$

四种核函数

本章后面将逐一介绍四种核函数：

- **线性核** (linear kernel)
- **多项式核** (polynomial kernel)
- **高斯核** (Gaussian kernel)，也叫**径向基核 RBF** (radial basis function kernel)
- **sigmoid 核** (sigmoid kernel)

本章利用上述四种核函数求解图 7 所示三组数据——线性可分、月牙形和环形数据。请读者注意比较不同核函数优劣以及决策边界形状。

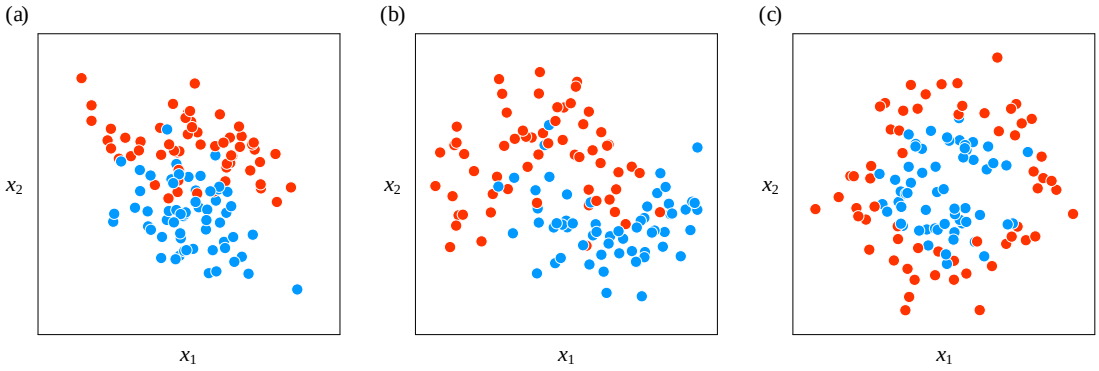


图 7. 三组数据——线性可分、月牙形和环形

8.3 线性核：最基本的核函数

线性核 (linear kernel) 是支持向量机 SVM 的标配！线性核函形式为：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \boldsymbol{x}\cdot\boldsymbol{q} = \langle \boldsymbol{x},\boldsymbol{q} \rangle \tag{27}$$

线性核 SVM 决策边界形式如下：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \left(\overbrace{\underbrace{\lambda_i y^{(i)} \boldsymbol{x}^{(i)}}_{\text{Coefficients}} \boldsymbol{x}}^{\text{Hyperplane}} \right) + b = 0 \tag{28}$$

其中 $(\boldsymbol{x}^{(i)}, y^{(i)})$ 为第 i 个样本数据点，λ_i 为求解得到拉格朗日乘子具体值。反复强调，$\boldsymbol{x}$ 为变量构成的列向量。

超平面叠加

定义函数 $f_i(\boldsymbol{x})$：

$$f_i(\boldsymbol{x}) = \overbrace{\lambda_i y^{(i)} \boldsymbol{x}^{(i)}}^{\text{Coefficients}} \boldsymbol{x} \tag{29}$$

此外，观察 (29)，给定样本数据 $(\boldsymbol{x}^{(i)}, y^{(i)})$，优化得到的拉格朗日乘子 λ_i 相当于权重。相信通过丛书之前数学内容学习，大家已经清楚，(29) 所示 $f_i(\boldsymbol{x})$ 空间形状为**超平面** (hyperplane)。

由此，(28) 可以记做：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(\boldsymbol{x}) + b = 0 \tag{30}$$

而 (30) 告诉我们，线性核 SVM 决策边界由 n 个超平面叠加而成，如图 8 所示。因此，线性核决策边界是超平面。

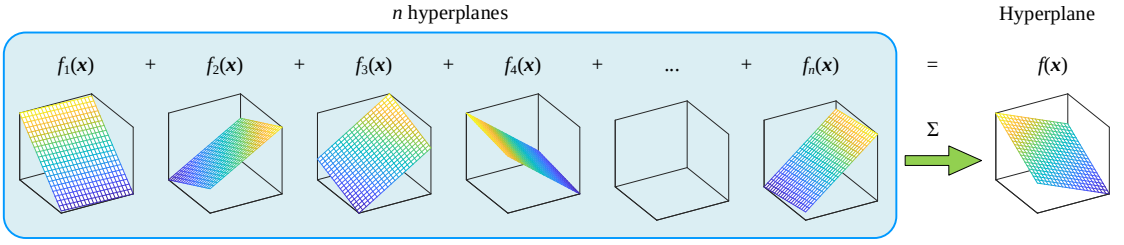


图 8. 线性核 SVM 决策超平面由 n 个超平面构造

决策边界

采用线性核预测图 7 所示三组形态不同数据分类。图 9、图 10 和图 11 分别为线性可分、月牙和环形数据预测分类结果。观察这三幅图，可以发现 $f(\boldsymbol{x})$ 对应几何形状均为平面，决策边界 $f(\boldsymbol{x}) = 0$ 为直线。

采用软间隔，线性核 SVM 尚可以分类图 9 所示样本数据。但是，对于图 10 和图 11 这种完全线性不可分数据，线性核 SVM 显得力不从心。

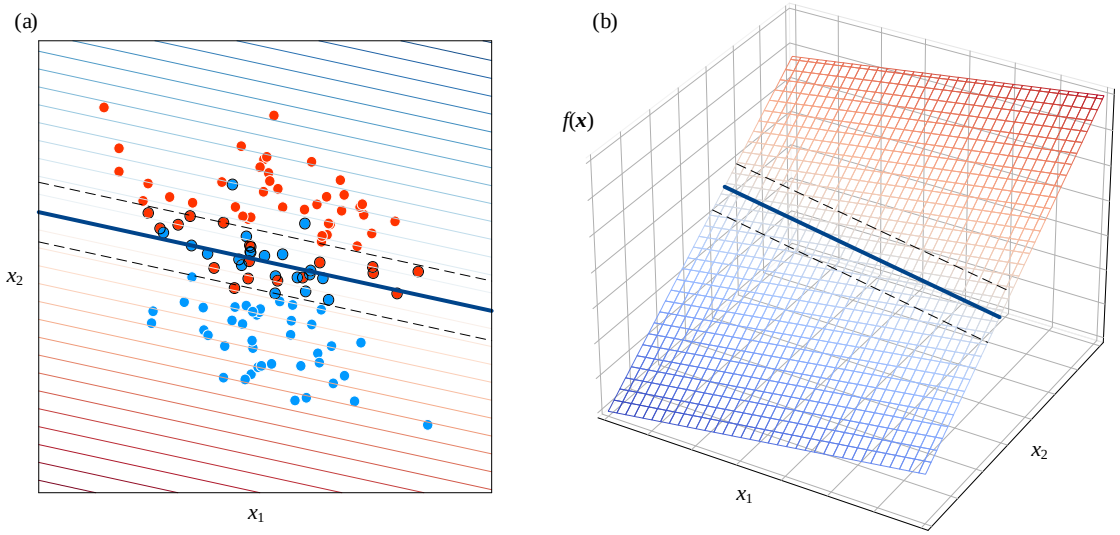


图 9. 线性可分数据，线性核 SVM

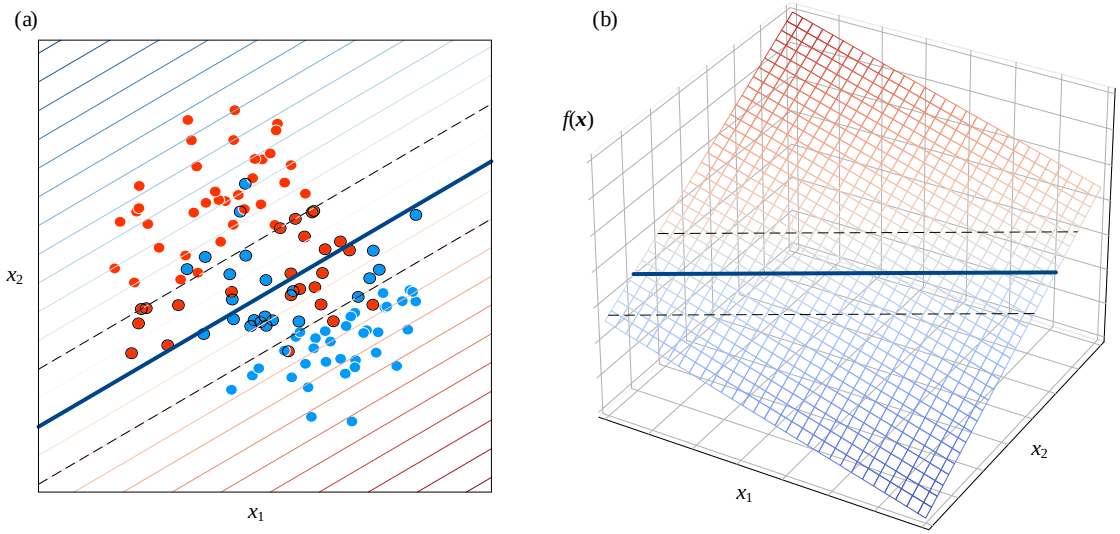


图 10. 月牙形数据，线性核 SVM

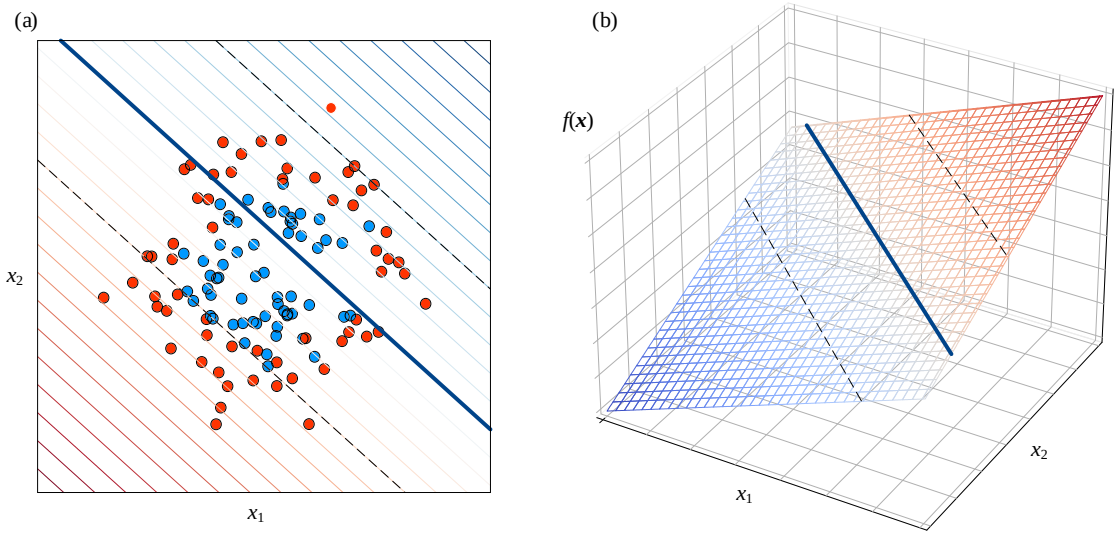


图 11. 环形数据，线性核 SVM

8.4 多项式核

多项式核 (polynomial kernel) 形式如下：

$$\kappa_{\text{poly}(d)}(\boldsymbol{x},\boldsymbol{q}) = (\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r)^d = (\gamma\langle \boldsymbol{x},\boldsymbol{q}\rangle + r)^d \tag{31}$$

其中，γ (gamma) 为系数，d 为多项式次数，r 为常数。

(31) 中 d 主导多项式核形态，因此最为重要。线性核是多项式核特例，即次数 $d = 1$。$d = 2$ 时，(31) 为二次核；$d = 3$ 时，(31) 为三次核。

以二次核为例

系数 $\gamma = 1$，常数 $r = 0$，次数 $d = 2$，特征数 $D = 2$ 条件下，(31) 可以写作：

$$\kappa_{\text{poly}(2)}(\boldsymbol{x},\boldsymbol{q}) = (\boldsymbol{x}\cdot\boldsymbol{q})^2 \tag{32}$$

其中，

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}} \quad \boldsymbol{q} = \begin{bmatrix} q_1 & q_2 \end{bmatrix}^{\mathrm{T}} \tag{33}$$

将 (33) 代入 (32)，整理得到：

$$\begin{aligned} \kappa_{\text{poly}(2)}(\boldsymbol{x},\boldsymbol{q}) &= (\boldsymbol{x}\cdot\boldsymbol{q})^2 = (x_1q_1 + x_2q_2)^2 \\ &= x_1^2q_1^2 + x_2^2q_2^2 + 2x_1x_2q_1q_2 = x_1^2\cdot q_1^2 + x_2^2\cdot q_2^2 + \sqrt{2}x_1x_2\cdot\sqrt{2}q_1q_2 \\ &= \phi(\boldsymbol{x})\cdot\phi(\boldsymbol{q}) \end{aligned} \tag{34}$$

其中，

$$\begin{aligned}\phi(\boldsymbol{x}) &= \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}x_1x_2 \end{bmatrix}^{\mathrm{T}} \\ \phi(\boldsymbol{q}) &= \begin{bmatrix} q_1^2 & q_2^2 & \sqrt{2}q_1q_2 \end{bmatrix}^{\mathrm{T}}\end{aligned} \tag{35}$$

(34) 还可以写作：

$$\begin{aligned}\kappa_{\text{poly}(2)}(\boldsymbol{x},\boldsymbol{q}) &= x_1^2 \cdot q_1^2 + x_2^2 \cdot q_2^2 + x_1x_2 \cdot q_1q_2 + x_2x_1 \cdot q_2q_1 \\ &= \phi(\boldsymbol{x}) \cdot \phi(\boldsymbol{q})\end{aligned} \tag{36}$$

其中，

$$\begin{aligned}\phi(\boldsymbol{x}) &= \begin{bmatrix} x_1^2 & x_2^2 & x_1x_2 & x_2x_1 \end{bmatrix}^{\mathrm{T}} \\ \phi(\boldsymbol{q}) &= \begin{bmatrix} q_1^2 & q_2^2 & q_1q_2 & q_2q_1 \end{bmatrix}^{\mathrm{T}}\end{aligned} \tag{37}$$

比较 (35) 和 (37)，可以发现 $\phi()$ 映射规则并不唯一。或者说，$\phi()$ 的具体形式并不重要，我们关心的是映射规则和标量结果。$\phi(\boldsymbol{x})$ 搭建起从 $\boldsymbol{x}$ 到核函数桥梁，如图 12 所示。

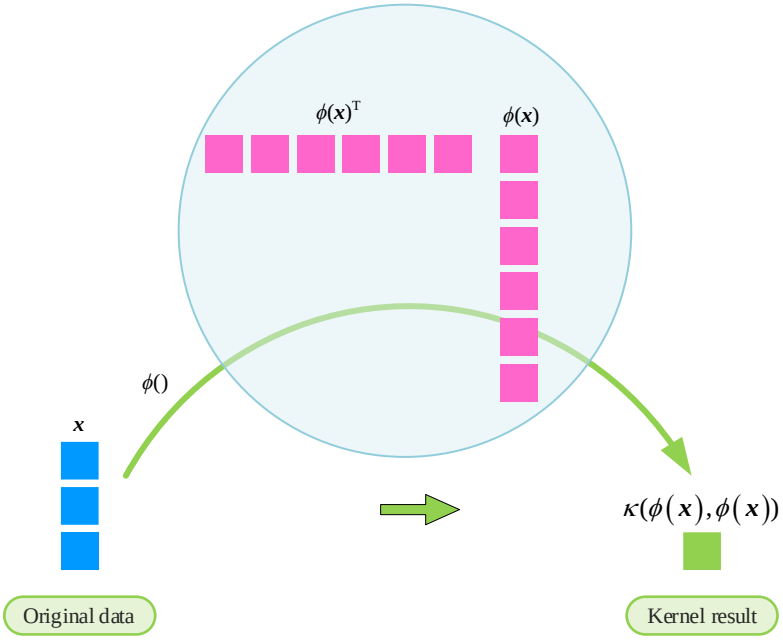


图 12. 映射原理

再看一个例子，系数 $\gamma = 1$，常数 $r = 1$，次数 $d = 2$，特征数 $D = 2$ 条件下，(31) 可以写作：

$$\kappa_{\text{poly}(2)}(\boldsymbol{x},\boldsymbol{q}) = (\boldsymbol{x} \cdot \boldsymbol{q} + 1)^2 \tag{38}$$

(38) 展开可以得到：

$$
\begin{aligned}
\kappa_{\text{poly}(2)}(\boldsymbol{x},\boldsymbol{q}) &= (\boldsymbol{x}\cdot\boldsymbol{q}+1)^2 \\
&= (x_1q_1+x_2q_2+1)^2 \\
&= 1+2x_1q_1+2x_2q_2+x_1^2q_1^2+x_2^2q_2^2+2x_1x_2q_1q_2 \\
&= 1\cdot 1+\sqrt{2}x_1\cdot\sqrt{2}q_1+\sqrt{2}x_2\cdot\sqrt{2}q_2+x_1^2\cdot q_1^2+x_2^2\cdot q_2^2+\sqrt{2}x_1x_2\cdot\sqrt{2}q_1q_2 \\
&= \phi(\boldsymbol{x})\cdot\phi(\boldsymbol{q})
\end{aligned} \tag{39}
$$

其中，

$$
\begin{aligned}
\phi(\boldsymbol{x}) &= \begin{bmatrix} 1 & \sqrt{2}x_1 & \sqrt{2}x_2 & x_1^2 & x_2^2 & \sqrt{2}x_1x_2 \end{bmatrix}^{\mathrm{T}} \\
\phi(\boldsymbol{q}) &= \begin{bmatrix} 1 & \sqrt{2}q_1 & \sqrt{2}q_2 & q_1^2 & q_2^2 & \sqrt{2}q_1q_2 \end{bmatrix}^{\mathrm{T}}
\end{aligned} \tag{40}
$$

多项式核 SVM 决策边界解析式为：

$$
f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(\lambda_i y^{(i)}\left(\gamma\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}+r\right)^d\right)+b=0 \tag{41}
$$

类似 (29)，可以用 n 个 $f_i(\boldsymbol{x})$ 函数构造决策边界解析式。

$$
f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(f_i(\boldsymbol{x})\right)+b=0 \tag{42}
$$

其中，

$$
f_i(\boldsymbol{x}) = \lambda_i y^{(i)}\left(\gamma\boldsymbol{x}^{(i)}\cdot\boldsymbol{x}+r\right)^d \tag{43}
$$

同理，(42) 可以解读为，多项式核**超曲面** (hypersurface) 相当有 n 个超曲面构造。图 13 所示为这一过程。下面，我们用两节内容，分别介绍如何用二次核和三次核分类图 7 所示三组形数据。

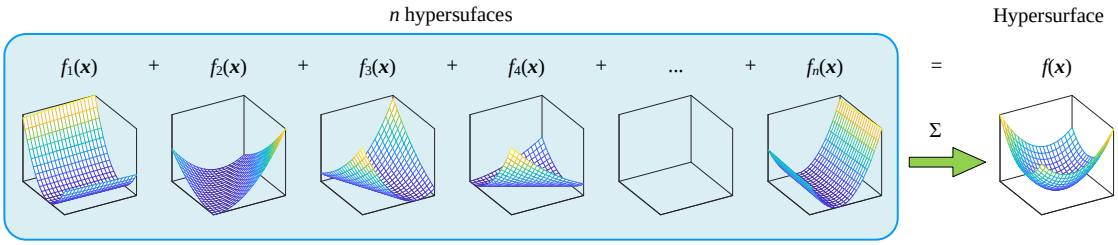


图 13. 多项式核超曲面相当有 n 个超曲面构造

8.5 二次核：二次曲面

二次核形式如下：

$$
\kappa_{\text{poly}(d)}(\boldsymbol{x},\boldsymbol{q}) = (\gamma\boldsymbol{x}\cdot\boldsymbol{q}+r)^2 = (\gamma\langle\boldsymbol{x},\boldsymbol{q}\rangle+r)^2 \tag{44}
$$

图 14、图 15 和图 16 所示为二次核 SVM 对线性可分、月牙形和环形数据预测分类结果。图 14 的决策面为双曲面；图 15 为椭圆抛物面；图 16 看似正圆抛物面。可以发现，二次核适用范围比较窄，分类效果一般。要想构造复杂曲面，就需要提高多项式核次数。

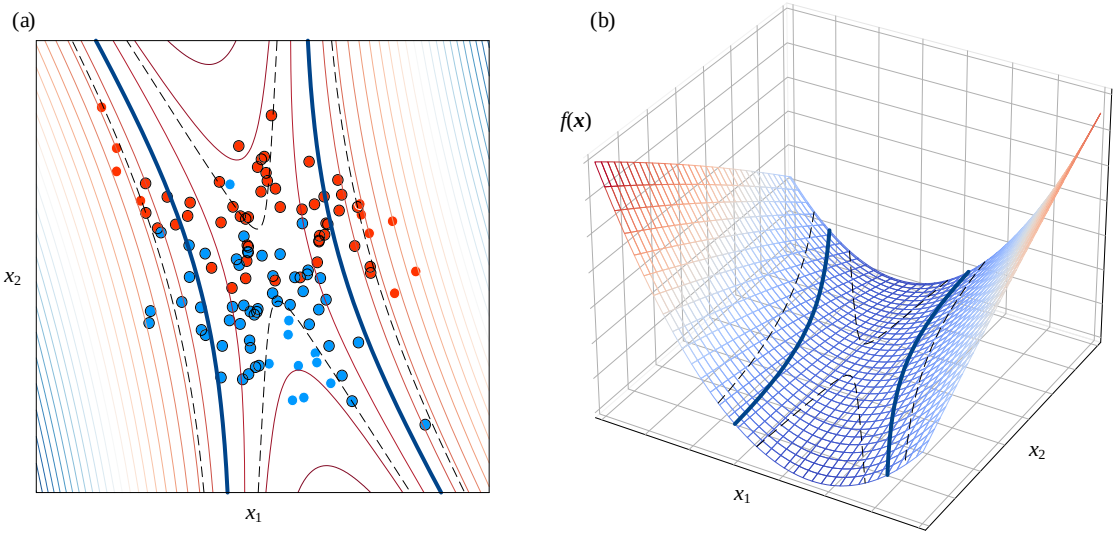


图 14. 线性可分数据，二次核 SVM

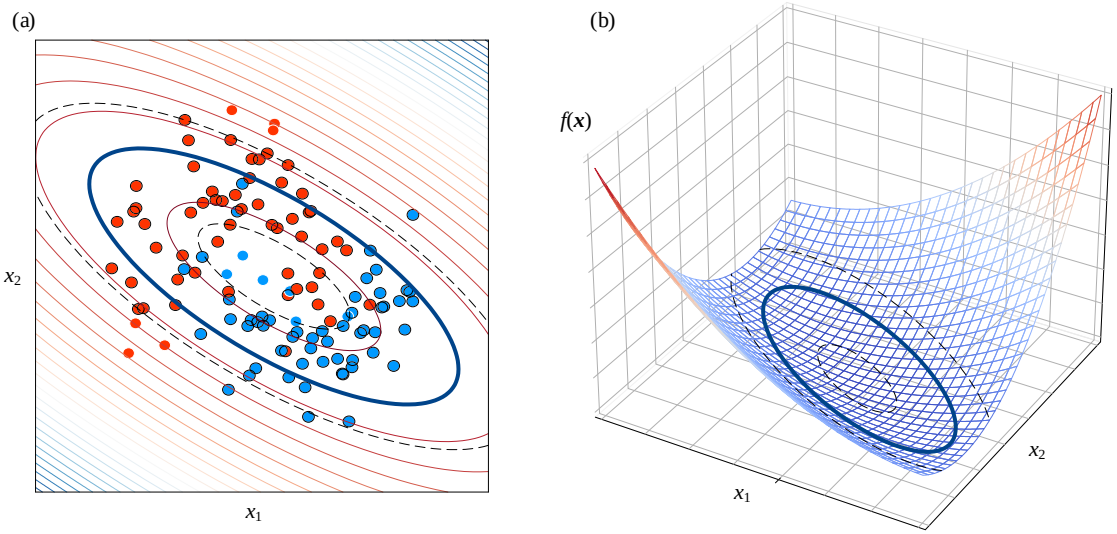


图 15. 月牙形数据，二次核 SVM

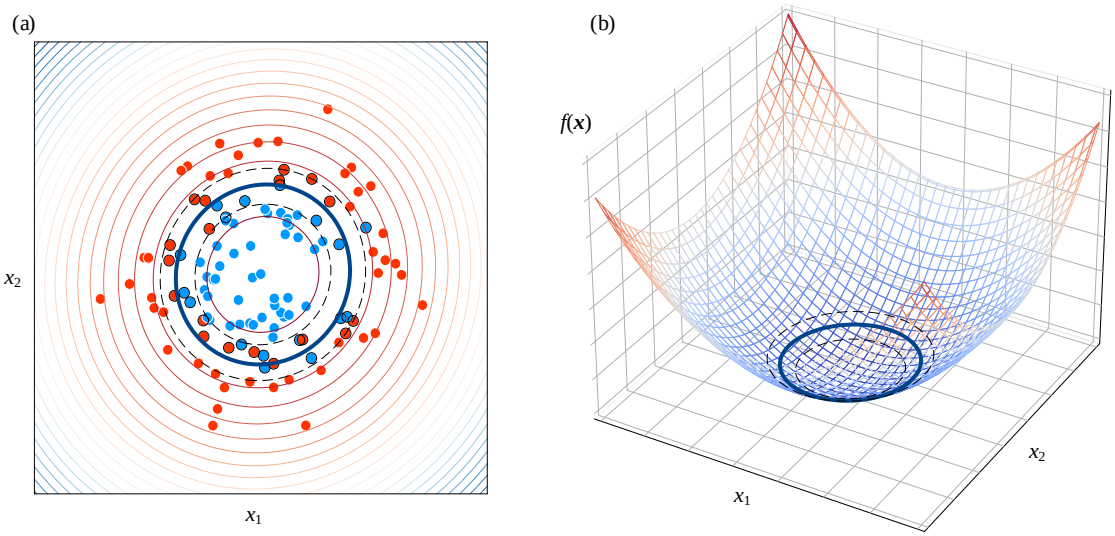


图 16. 环形数据，二次核 SVM

8.6 三次核：三次曲面

三次核形式如下：

$$\kappa_{\text{poly}(3)}\left(\boldsymbol{x},\boldsymbol{q}\right)=\left(\gamma\boldsymbol{x}\cdot\boldsymbol{q}+r\right)^3=\left(\gamma\left\langle\boldsymbol{x},\boldsymbol{q}\right\rangle+r\right)^3 \tag{45}$$

图 17、图 18 和图 19 所示为三次核 SVM 分类线性可分、月牙形和环形数据结果。对比二次核 SVM 结果，可以发现三次核分类结果远好于二次核。本章最后给出代码，请读者尝试不断提高多项式核次数，并观察不同次数多项式核预测分类结果，仔细观察决策边界形状。

多项式核次数不是越高越好；次数过高对带来过拟合 (overfitting)，泛化能力低的问题。过拟合指的是机器学习模型在训练数据上表现良好，但在测试数据上表现不佳的现象。这是因为模型过度拟合训练数据，学习了数据的噪声和细节，而忽略了数据的潜在模式。过拟合可能会导致模型的泛化性能下降，因此需要采用一些技术来避免过拟合。

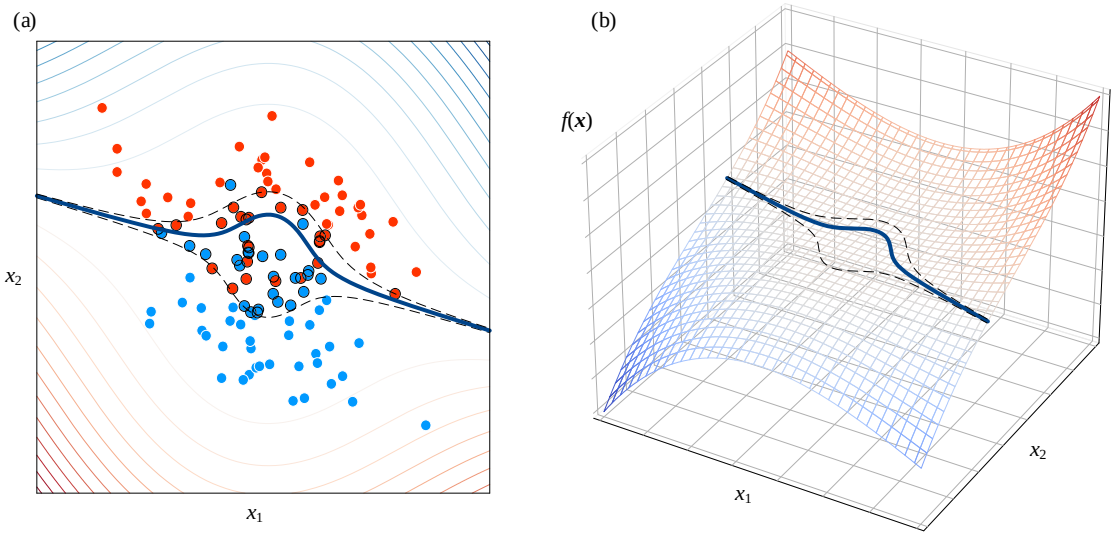


图 17. 线性可分数据，三次核 SVM

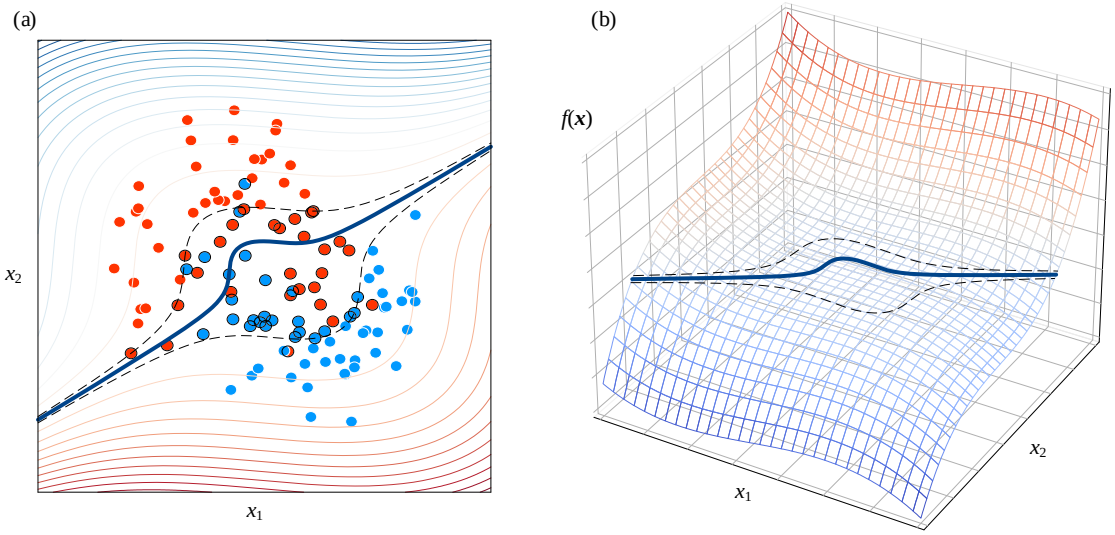


图 18. 月牙形数据，三次核 SVM

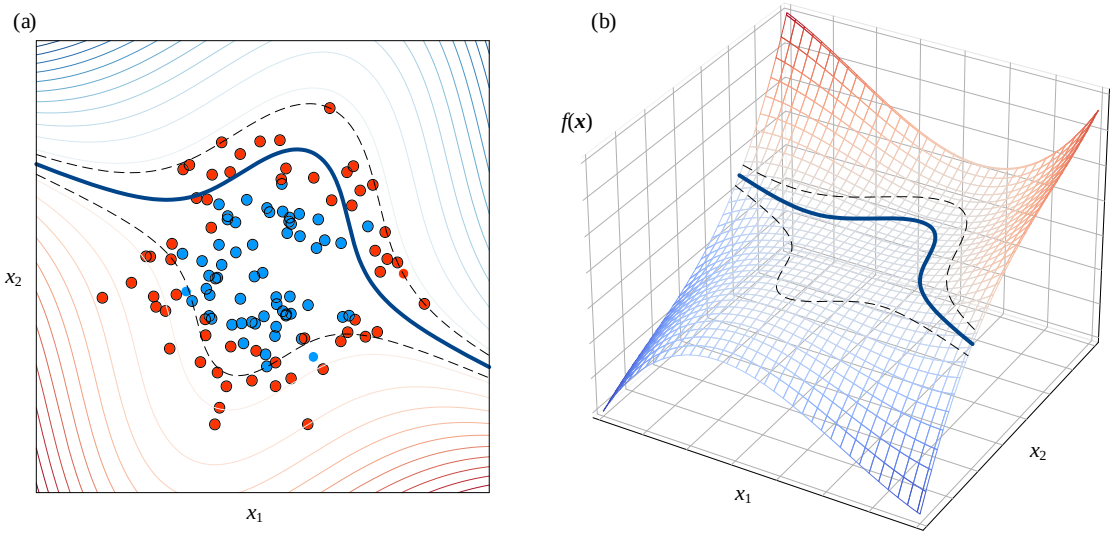


图 19. 环形数据，三次核 SVM

8.7 高斯核：基于径向基函数

对于刚接触支持向量机的读者，高斯核是谜一样的存在。它看上去那么简单，又那么神秘。本节就帮助大家揭开高斯核的面纱一角。

高斯核 (Gaussian kernel)，也叫**径向基核函数** (radial basis function kernel, RBF kernel)，具体形式为：

$$\kappa_{\text{RBF}}\left(\boldsymbol{x},\boldsymbol{q}\right)=\exp\left(-\gamma\left\|\boldsymbol{x}-\boldsymbol{q}\right\|^2\right) \tag{46}$$

(46) 有一个参数，γ ($\gamma>0$)；γ 决定高斯核函数曲面的开口大小。

形状

对于单特征 ($D=1$)，且 $q=0$ 时，(46) 可以写作：

$$\kappa_{\text{RBF}}\left(x,0\right)=\exp\left(-\gamma x^2\right) \tag{47}$$

图 20 给出 γ 如何影响高斯核曲线。

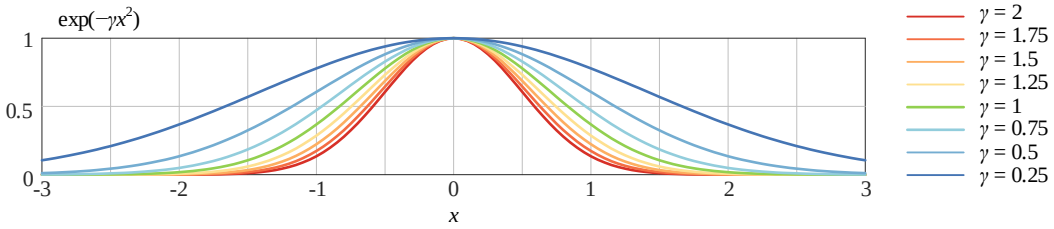


图 20. γ 决定高斯核函数曲面的开口大小

试着找到映射函数

$\gamma=1/2$ 时，

$$\begin{aligned}\kappa_{\text{RBF}}\left(\boldsymbol{x},\boldsymbol{q}\right)&=\exp\left(-\frac{1}{2}\left\|\boldsymbol{x}-\boldsymbol{q}\right\|^2\right)\\&=\exp\left(-\frac{1}{2}\left(\boldsymbol{x}-\boldsymbol{q}\right)\cdot\left(\boldsymbol{x}-\boldsymbol{q}\right)\right)\\&=\exp\left(-\frac{1}{2}\left(\left\|\boldsymbol{x}\right\|^2+\left\|\boldsymbol{q}\right\|^2-2\boldsymbol{x}\cdot\boldsymbol{q}\right)\right)\\&=\exp\left(-\frac{1}{2}\left(\left\|\boldsymbol{x}\right\|^2+\left\|\boldsymbol{q}\right\|^2\right)\right)\exp\left(\boldsymbol{x}\cdot\boldsymbol{q}\right)\end{aligned} \tag{48}$$

对 $\exp(\boldsymbol{x}\cdot\boldsymbol{q})$ 泰勒展开 (Taylor expansion)：

$$\begin{aligned}\exp(\boldsymbol{x}\cdot\boldsymbol{q}) &= \sum_{j=0}^{\infty}\frac{(\boldsymbol{x}\cdot\boldsymbol{q})^j}{j!}\\ &= 1\cdot 1+\boldsymbol{x}\cdot\boldsymbol{q}+\frac{(\boldsymbol{x}\cdot\boldsymbol{q})^2}{2}+\ldots\end{aligned} \tag{49}$$

可以发现 $\exp(\boldsymbol{x}\cdot\boldsymbol{q})$ 展开得到无限项。

$D = 2$ 时，仅考虑 (49) 前三项，$\exp(\boldsymbol{x}\cdot\boldsymbol{q})$ 可以展开并整理为：

$$\exp(\boldsymbol{x}\cdot\boldsymbol{q}) \approx 1\cdot 1+(x_1\cdot q_1+x_2\cdot q_2)+\frac{x_1^2\cdot q_1^2+x_2^2\cdot q_2^2+\sqrt{2}x_1x_2\cdot\sqrt{2}q_1q_2}{2} \tag{50}$$

将 (50) 代入 (48)，整理得到：

$$\begin{aligned}\kappa_{\text{RBF}}(\boldsymbol{x},\boldsymbol{q}) &\approx \exp\left(-\frac{1}{2}\|\boldsymbol{x}\|^2\right)\exp\left(-\frac{1}{2}\|\boldsymbol{q}\|^2\right)\left(1\cdot 1+(x_1\cdot q_1+x_2\cdot q_2)+\frac{x_1^2\cdot q_1^2+x_2^2\cdot q_2^2+\sqrt{2}x_1x_2\cdot\sqrt{2}q_1q_2}{2}\right)\\ &= \phi(\boldsymbol{x})\cdot\phi(\boldsymbol{q})\end{aligned} \tag{51}$$

其中，两个映射函数可以记做：

$$\begin{aligned}\phi(\boldsymbol{x}) &= \exp\left(-\frac{1}{2}\|\boldsymbol{x}\|^2\right)\begin{bmatrix}1 & x_1 & x_2 & x_1^2 & x_2^2 & \sqrt{2}x_1x_2\end{bmatrix}\\ \phi(\boldsymbol{q}) &= \exp\left(-\frac{1}{2}\|\boldsymbol{q}\|^2\right)\begin{bmatrix}1 & q_1 & q_2 & q_1^2 & q_2^2 & \sqrt{2}q_1q_2\end{bmatrix}\end{aligned} \tag{52}$$

如上文所述，高斯核经过泰勒展开后，实际上映射函数有无数项；但是，本章反复强调，映射函数的形式并不重要。

试着找到映射函数

高斯核 SVM 对应的决策边界解析式：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(\lambda_i y^{(i)}\exp\left(-\gamma\left\|\boldsymbol{x}^{(i)}-\boldsymbol{x}\right\|^2\right)\right)+b = 0 \tag{53}$$

和前文一样，用 n 个 $f_i(\boldsymbol{x})$ 函数构造 $f(\boldsymbol{x})$：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(f_i(\boldsymbol{x})\right)+b = 0 \tag{54}$$

其中，

$$f_i(\boldsymbol{x}) = \lambda_i y^{(i)}\exp\left(-\gamma\left\|\boldsymbol{x}^{(i)}-\boldsymbol{x}\right\|^2\right) \tag{55}$$

对于两特征 $D = 2$，(55) 可以展开得到：

$$f_i(\boldsymbol{x}) = \lambda_i y^{(i)} \exp\left(-\gamma\left(\left(x_1 - x_{i,1}\right)^2 + \left(x_2 - x_{i,2}\right)^2\right)\right) \tag{56}$$

其中，仅 x_1 和 x_2 为变量。$(x_{i,1}, x_{i,2})$ 决定曲面中心位置，γ 决定曲面“胖瘦”，$\lambda_i y^{(i)}$ 决定曲面高矮。

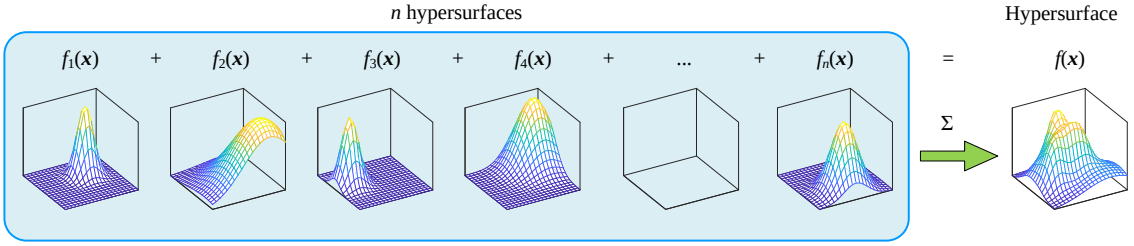


图 21. 高斯核 SVM 决策曲面相当有 n 个“高斯核曲面”构造

分类结果

图 22、图 23 和图 24 所示为高斯核 SVM 求解线性可分、月牙形和环形数据分类问题结果。可以发现高斯核 SVM 得到的决策边界形状丰富多样，分类预测结果要好于前文介绍的二次核和三次核。

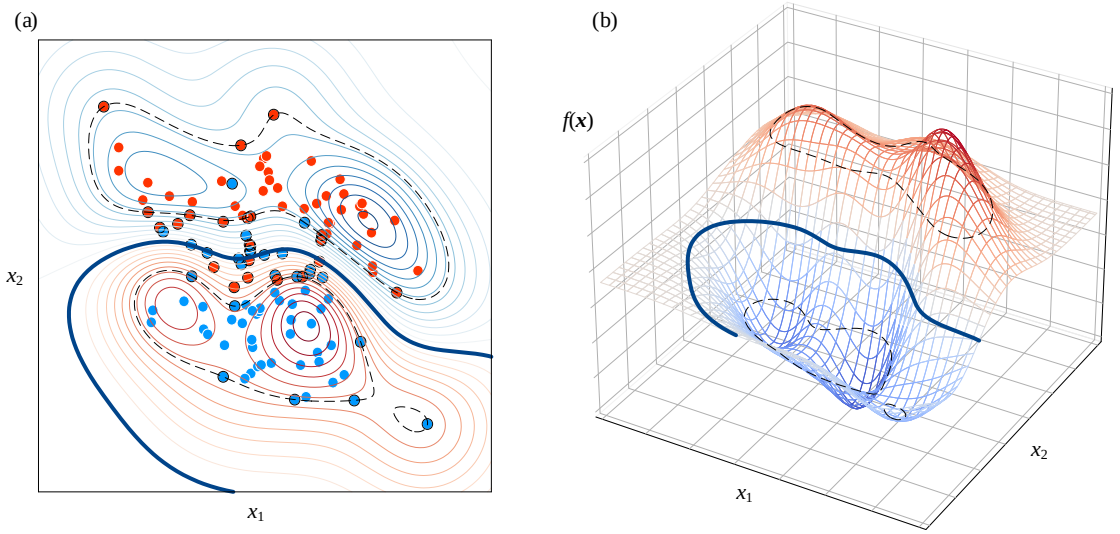


图 22. 线性可分数据，高斯核 SVM

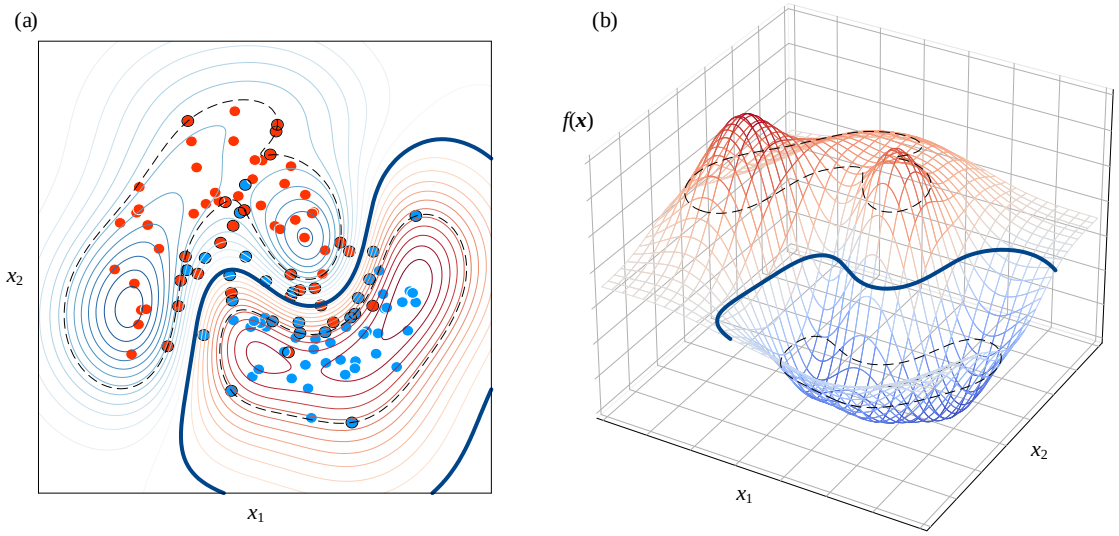


图 23. 月牙形数据，高斯核 SVM

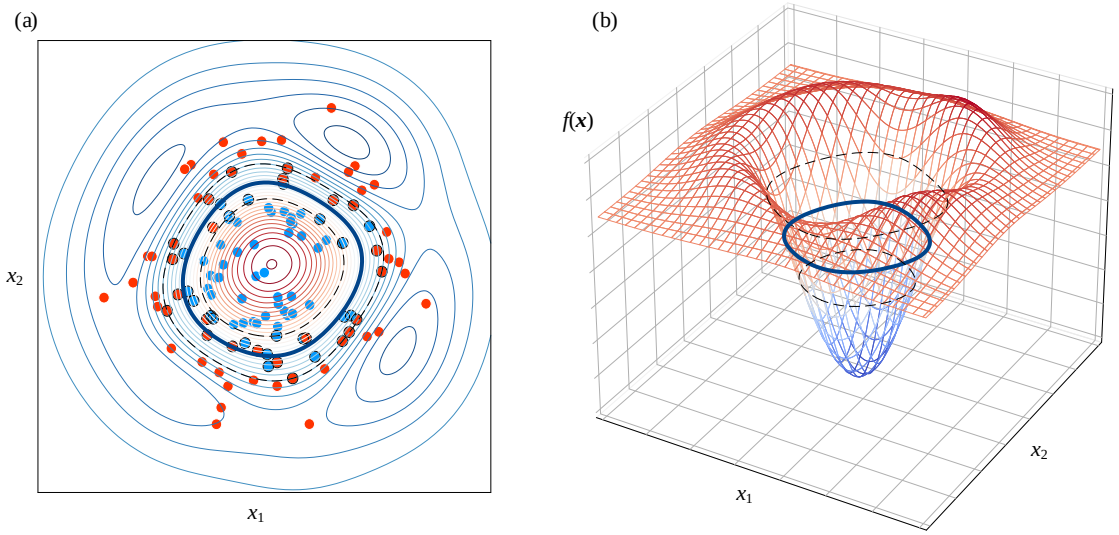


图 24. 环形数据，高斯核 SVM

8.8 Sigmoid 核

Sigmoid 核 (sigmoid kernel)，也叫 S 形核，形式如下：

$$\kappa_{\text{Sigmoid}}(\boldsymbol{x},\boldsymbol{q}) = \tanh\left(\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r\right) \tag{57}$$

其中，tanh() 为双曲正切函数 (hyperbolic tangent)。

形状

对于单特征 ($D = 1$)，且 $q = 1$、$r = 0$ 时，(57) 可以写作：

$$\kappa_{\text{Sigmoid}}(x,1) = \tanh(\gamma x) \tag{58}$$

丛书第一本数学部分介绍过双曲正切函数，图 25 给出 γ 如何影响 sigmoid 核曲线。tanh() 是机器学习的常客，比如在神经网络中 tanh()便是常用的激励函数之一。

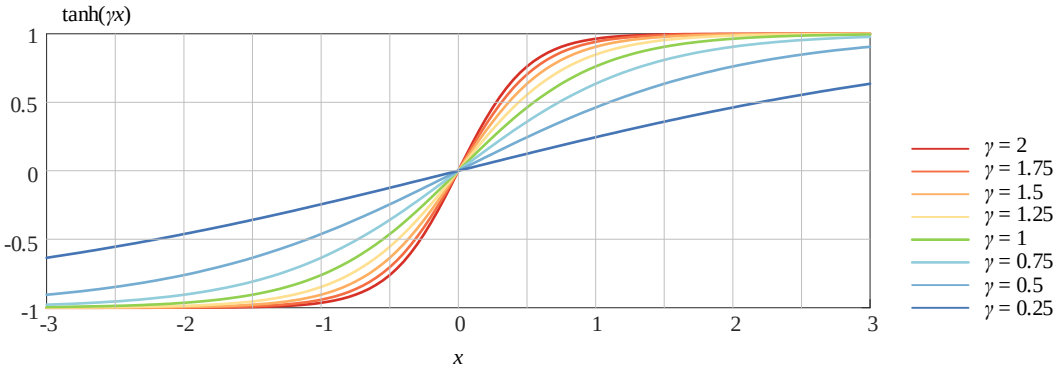


图 25. γ 影响双曲正切函数形状

决策边界

Sigmoid 核 SVM 决策边界解析式如下：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(\lambda_i y^{(i)} \tanh\left(\gamma \boldsymbol{x}^{(i)}\boldsymbol{x} + r\right)\right) + b = 0 \tag{59}$$

和前文一样，$f(\boldsymbol{x})$ 可以由 n 个 $f_i(\boldsymbol{x})$ 构造：

$$f(\boldsymbol{x}) = \sum_{i=1}^{n}\left(f_i(\boldsymbol{x})\right) + b = 0 \tag{60}$$

其中，

$$f_i(\boldsymbol{x}) = \lambda_i y^{(i)} \tanh\left(\gamma \boldsymbol{x}^{(i)} \cdot \boldsymbol{x} + r\right) \tag{61}$$

如图 26 所示，sigmoid 核曲面可以由若干 sigmoid 曲面构造得到。

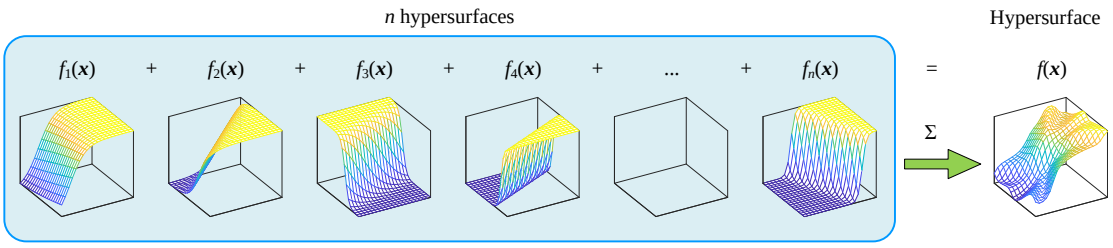


图 26. Sigmoid 核曲面可以由若干 sigmoid 曲面构造

分类结果

图 27、图 28 和图 29 所示为 sigmoid 核 SVM 解决三组样本数据结果。类似高斯核，sigmoid 核能够构造比较复杂的曲面形状。

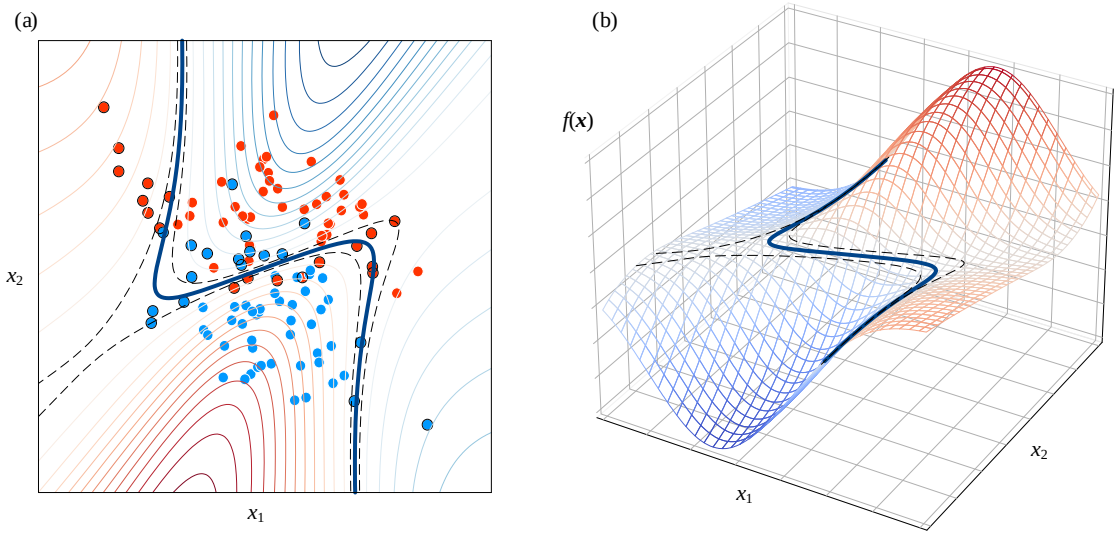


图 27. 线性可分数据，sigmoid 核 SVM

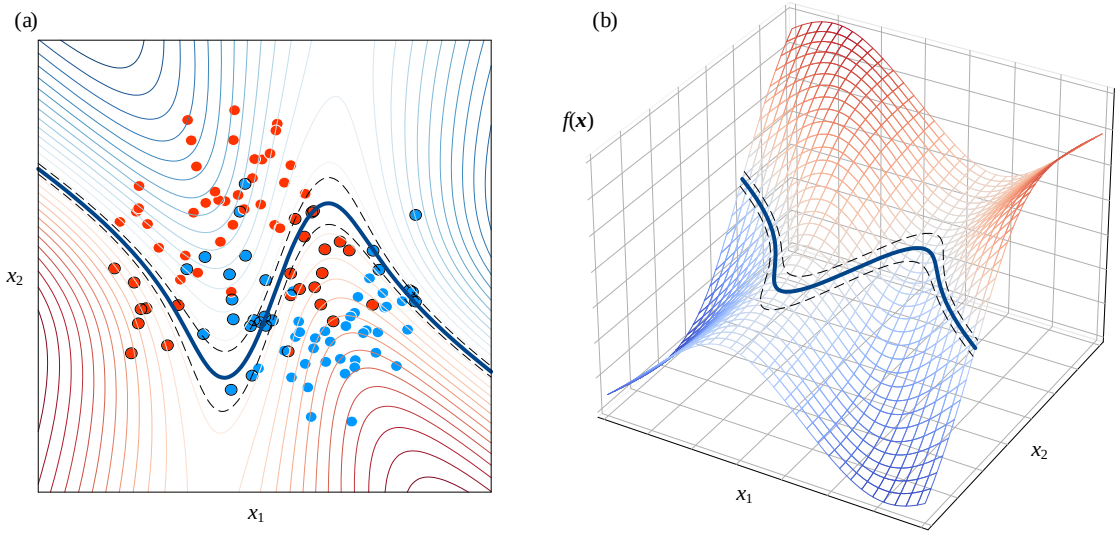


图 28. 月牙形数据，sigmoid 核 SVM

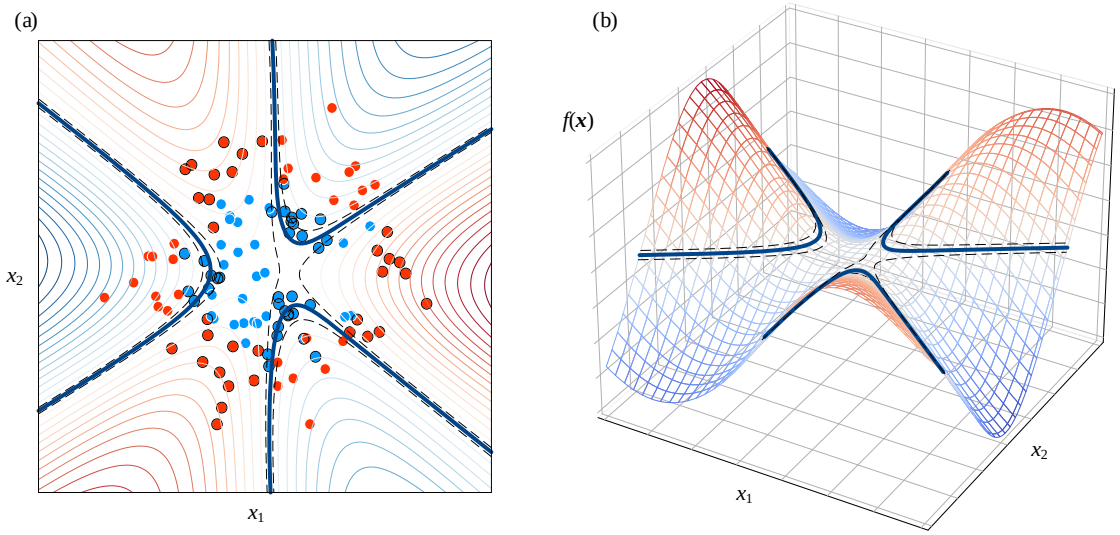


图 29. 环形数据，Sigmoid 核 SVM

代码 Bk7_Ch08_01.py 可以获得本章线性可分、月牙形和环形数据，并且利用线性核、多项式核、高斯核和 sigmoid 核求解这三类数据分类。

在 SVM 中，核技巧可以将输入数据映射到高维空间中，使得原本的非线性问题转化为线性问题，从而提高模型的分类性能。常用的核函数包括线性核函数、多项式核函数和径向基函数 RBF 核函数等。

线性核函数适用于线性可分的数据集，可以在低维空间中很好地工作，但在处理非线性问题时效果较差。多项式核函数可以处理一些简单的非线性问题，但需要调整多项式的次数。RBF 核函数是最常用的核函数之一，能够处理更加复杂的非线性问题。

不同的核函数适用于不同的数据集和问题类型。选择适当的核函数和调整参数是使用 SVM 的关键。在实践中，可以使用交叉验证等技术来选择最佳的核函数和参数。

9 Decision Tree 决策树

数据纯度越高，不确定度越低，信息熵越小

热力学两个基本定理是整个宇宙的基本规律：1. 宇宙能量守恒；2. 宇宙的熵不断增大。

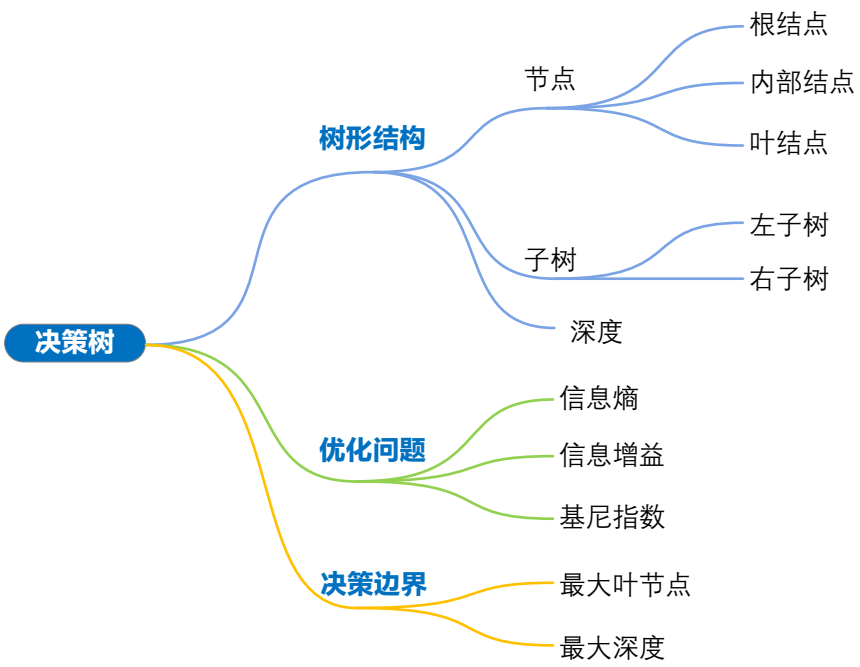
The fundamental laws of the universe which correspond to the two fundamental theorems of the mechanical theory of heat.

1. The energy of the universe is constant.

2. The entropy of the universe tends to a maximum.

—— 鲁道夫·克劳修斯 (Rudolf Clausius) | 德国物理学家 | 1822 ~ 1888

- `matplotlib.pyplot.contour()` 绘制等高线线图
- `matplotlib.pyplot.contourf()` 绘制填充等高线图
- `numpy.meshgrid()` 创建网格化数据
- `seaborn.scatterplot()` 绘制散点图
- `sklearn.datasets.load_iris()` 加载鸢尾花数据集
- `sklearn.tree.DecisionTreeClassifier` 决策树分类函数
- `sklearn.tree.plot_tree` 绘制决策树树形



9.1 决策树：可以分类，也可以回归

决策树分类算法是一种常用的机器学习算法，通过构建树形结构来对数据进行分类。决策树分类算法具有易解释、易理解和易实现的优点，但在处理复杂问题时可能会出现过拟合的问题。因此，可以采用剪枝等技术来提高决策树的泛化能力。

决策树结构

决策树 (decision tree) 类似《数学要素》第 20 章介绍的**二叉树** (binomial tree)。如图 1 所示，决策树结构主要由**结点** (node) 和**子树** (branch) 构成；结点又分为**根结点** (root node)、**内部结点** (internal node) 和**叶结点** (leaf node)。其中，内部节点又叫**母节点** (parent node)，叶节点又叫**子节点** (child node)。

每一个根节点和内部结点可以生长出一层二叉树，其中包括**左子树** (left branch) 和**右子树** (right branch)；构造子树的过程也是将结点数据划分为两个子集的过程。

图 1 所示树形结构有 4 个叶节点。请大家格外注意叶节点数目；决策树算法可以输入**最大叶节点数量** (maximum leaf nodes)，控制决策树大小，也称**剪枝** (pruning)。

此外，**深度** (depth) 也可以控制树形大小，所谓深度就是二叉树的层数。比如，图 1 二叉树有两层，所以深度为 2。深度也是决策树函数用户输入量之一。

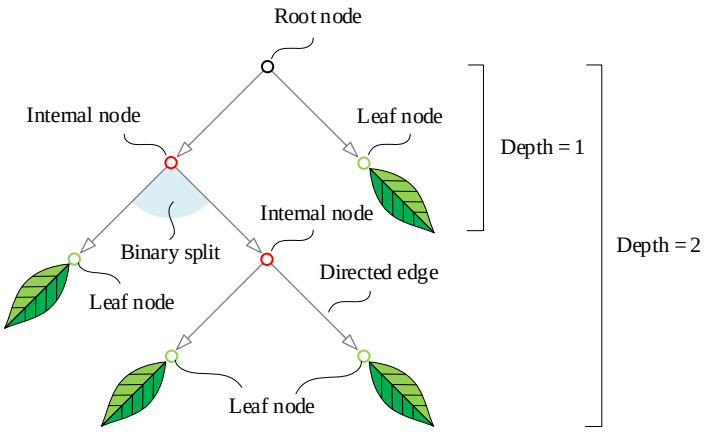


图 1. 决策树树形结构

如何用决策树分类

下面展开讲解决策树如何分类。

图 2 展示的决策树第一步划分：样本数据中 $x_1 \geq a$，被划分到右子树；样本数据中 $x_1 < a$，被划分到左子树。经过第一步二叉树划分，原始数据被划分为 A 和 B 两个区域。A 区域以红色 ● (C_1) 为主，B 区域以蓝色 ● (C_2) 为主。

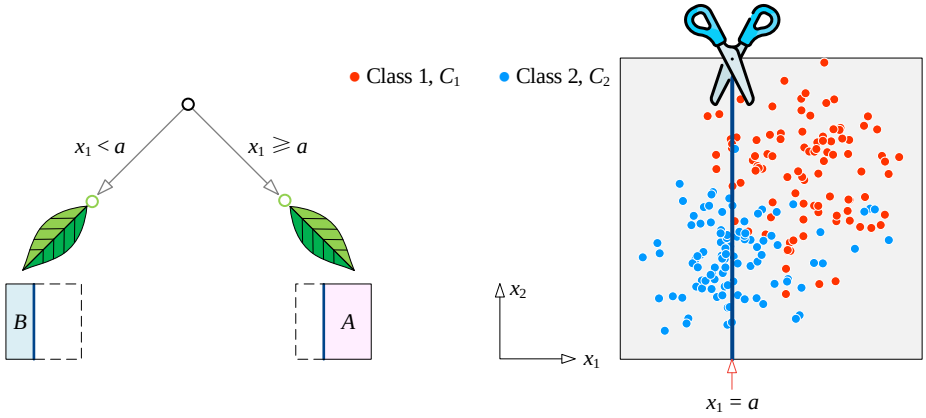


图 2. 决策树第一步划分

图 3 所示为图 2 右子树内部结点生长出一个新的二叉树。样本数据中 $x_2 \geq b$，被划分到右子树；样本数据中 $x_2 < b$，被划分到左子树。经过第二步二叉树划分，A 被划分为 C 和 D 两个区域。C 区域以红色 ● (C_1) 为主，D 区域以蓝色 ● (C_2) 为主。

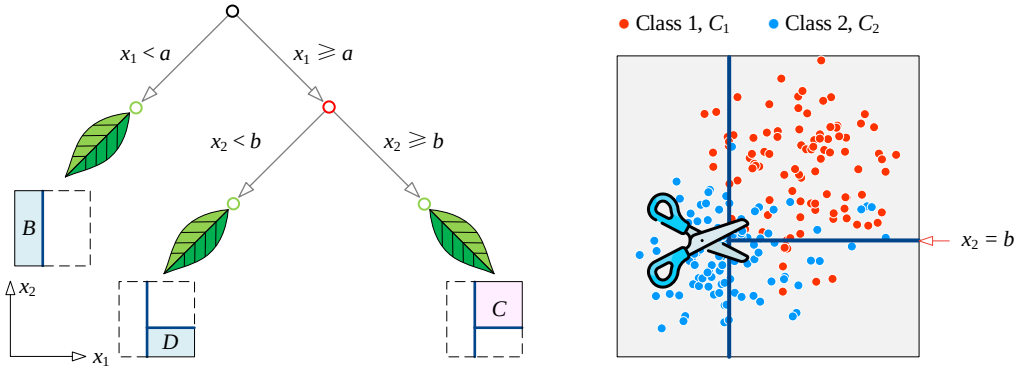


图 3. 决策树第二步划分

决策树分类算法有自己独特的优势。决策树的每个节点可以生长成一颗二叉树，这种基于某一特征的二分法很容易解释。此外，得到的决策树很方便可视化，本章后续将介绍如何可视化决策树树形结构。

如老子所言，“一生二，二生三，三生万物”，根据数据的复杂程度，决策树树形可以不断生长。数据结构越复杂，对应树形结构也就越复杂。但是，过于复杂的树形会导致过度拟合，模型泛化能力变弱。这种情况需要控制叶节点数量或者最大深度来控制树形规模，从而避免过度拟合。

有读者可能会问，依据什么标准选择划分的位置呢？比如图 2 中，a 应该选在什么位置？图 3 中的 b 又该选择什么位置？这就是下几节要回答的问题。

9.2 信息熵：不确定性度量

为了解决在决策树在哪划分节点的问题，需要介绍几个新概念：**信息熵** (information entropy) 、**信息增益** (information gain) 和**基尼指数** (Gini index)。本节首先介绍信息熵。

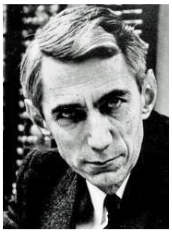
熵

熵 (entropy) 是物理系统混乱程度的度量。系统越混乱，熵越大；系统越有序，熵越小。熵这个概念起源热力学。1854 年，德国物理学家**鲁道夫·克劳修斯** (Rudolf Clausius) 引入熵这一概念。

维纳过程 (Wiener process) 的提出者——**诺伯特·维纳** (Norbert Wiener)，认为随着熵的增加，宇宙以及宇宙中所有封闭系统都会自然地退化，并失去其独特性。

信息熵

在**信息论** (information theory) 中，信息的作用是降低不确定性。**信息熵** (information entropy) 可以用来表示随机变量的不确定性度量。信息熵越大，不确定性越大。1948 年，**香农** (Claude Shannon) 提出信息熵这一概念，因此信息熵也常被称作**香农熵** (Shannon entropy)。



克劳德·香农 (Claude Shannon)
美国数学家、工程师、密码学家 | 1916 ~ 2001
信息论创始人。丛书关键词：● 信息熵 ● 信息增益

样本数据集合 Ω 的信息熵定义为：

$$\mathrm{Ent}(\Omega) = -\sum_{k=1}^{n} p_k \log_2 p_k \tag{1}$$

其中，p_k 为 Ω 中第 k 类样本所占比例，即概率值。由于 $\log_2 0$ 不存在，特别指定 $0 \times \log_2 0 = 0$。

举个例子

当样本数据集合 Ω 只有两类 $K = 2$，类别序数 $k = 1, 2$。

令

$$p_1 = p, \quad p_2 = 1 - p \tag{2}$$

其中，p 取值范围 [0, 1]。

这种情况下，Ω 的信息熵 Ent(Ω) 为：

$$\begin{aligned} \mathrm{Ent}(\Omega) &= -\sum_{k=1}^{2} p_k \log_2 p_k = -\left(p_1 \log_2 p_1 + p_2 \log_2 p_2\right) \\ &= -p \log_2 p - (1-p) \log_2 (1-p) \end{aligned} \tag{3}$$

其中，$p_1 = p$，$p_2 = 1 - p$。

观察 (3)，可以发现 Ent(Ω) 是以 p 为变量的函数。

图 6 告诉我们，在 A 和 C 点，当样本只属于某一特定类别时 ($p = 0$ 或 $p = 1$)，也就是数据纯度最高，不确定性最低，信息熵 Ent(Ω) 最小。

在 B 点，两类样本数据各占一半 ($p = 0.5$)，这时数据纯度最低，不确定性最高，信息熵 Ent(D) 最大。

从 A 到 B，信息熵不断增大；从 B 到 C，信息熵不断减小。

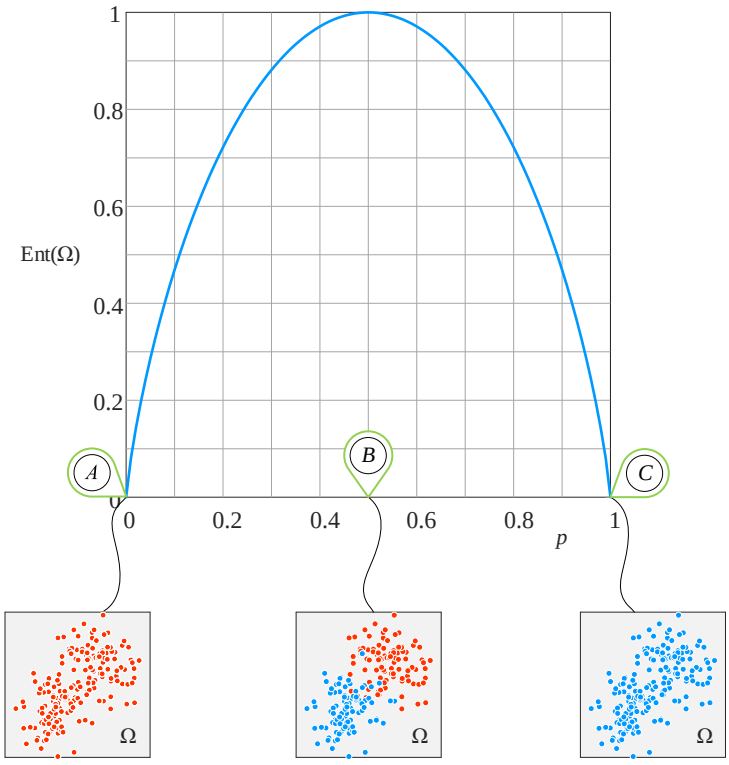


图 4. 信息熵 Ent(Ω) 随 p 变化趋势

K 类标签

如果样本数据集合 Ω 分为 K 类时，即 $\Omega = \{C_1, C_2, ..., C_K\}$，各类标签样本数量之和等于 Ω 中所有样本总数，即下式：

$$\sum_{k=1}^{K} \text{count}\left(C_k\right) = \text{count}\left(\Omega\right) \tag{4}$$

其中，count(C_k) 计算 C_k 类样本数量。

C_k 类样本概率 p_k 可以通过下式计算获得：

$$p_k = \frac{\text{count}\left(C_k\right)}{\text{count}\left(\Omega\right)} \tag{5}$$

将 (5) 代入 (1)，得到样本数据集合 Ω 的信息熵为：

$$\text{Ent}\left(\Omega\right) = -\sum_{k=1}^{K} p_k \log_2 p_k = -\sum_{k=1}^{K} \left\{ \frac{\text{count}\left(C_k\right)}{\text{count}\left(\Omega\right)} \log_2 \left(\frac{\text{count}\left(C_k\right)}{\text{count}\left(\Omega\right)} \right) \right\} \tag{6}$$

9.3 信息增益：通过划分，提高确定度

假设存在某个特征 a 将 Ω 划分为 m 个子集，即：

$$\Omega = \left\{\Omega_1, \Omega_2, ..., \Omega_m\right\} \tag{7}$$

而子集 Ω_j (j = 1, 2, ..., m) 中属于 C_k 类样本集合为 $\Omega_{j,k}$：

$$\Omega_{j,k} = \Omega_j \bigcap C_k \tag{8}$$

类别 C_k 元素在 Ω_j 中占比为：

$$p_{j,k} = \frac{\text{count}\left(\Omega_{j,k}\right)}{\text{count}\left(\Omega_j\right)} \tag{9}$$

计算子集 Ω_j 信息熵：

$$\text{Ent}\left(\Omega_j\right) = -\sum_{k=1}^{K} \left\{ \frac{\text{count}\left(\Omega_{j,k}\right)}{\text{count}\left(\Omega_j\right)} \log_2 \left(\frac{\text{count}\left(\Omega_{j,k}\right)}{\text{count}\left(\Omega_j\right)} \right) \right\} \tag{10}$$

而经过特征 a 划分后的集合 Ω 的信息熵为，m 个子集 Ω_j 信息熵的加权和：

$$\underbrace{\text{Ent}\left(\Omega|a\right)}_{\substack{\text{Weighted sum of} \\ \text{entropy after split}}} = \sum_{j=1}^{m} \left\{ \frac{\text{count}\left(\Omega_j\right)}{\text{count}\left(\Omega\right)} \text{Ent}\left(\Omega_j\right) \right\} \tag{11}$$

将 (10) 代入 (11)，得到：

$$\mathrm{Ent}\left(\Omega|a\right)=-\sum_{j=1}^{m}\left\{\frac{\mathrm{count}\left(\Omega_j\right)}{\mathrm{count}\left(\Omega\right)}\sum_{k=1}^{K}\left\{\frac{\mathrm{count}\left(\Omega_{j,k}\right)}{\mathrm{count}\left(\Omega_j\right)}\log_2\left(\frac{\mathrm{count}\left(\Omega_{j,k}\right)}{\mathrm{count}\left(\Omega_j\right)}\right)\right\}\right\} \tag{12}$$

经过特征 a 划分后的 Ω 信息熵减小，确定度提高。

举个例子

图 5 所示数据集 Ω 有两个标签，C_1 和 C_2。特征 a 将数据集 Ω 划分为 2 个子集——Ω_1、Ω_2。

根据 (9) 类别 C_1 元素在 Ω_1 中占比为：

$$p_{1,1}=\frac{\mathrm{count}\left(\Omega_{1,1}\right)}{\mathrm{count}\left(\Omega_1\right)} \tag{13}$$

子集 Ω_1 信息熵为：

$$\mathrm{Ent}\left(\Omega_1\right)=-\frac{\mathrm{count}\left(\Omega_{1,1}\right)}{\mathrm{count}\left(\Omega_1\right)}\log_2\left(\frac{\mathrm{count}\left(\Omega_{1,1}\right)}{\mathrm{count}\left(\Omega_1\right)}\right)-\frac{\mathrm{count}\left(\Omega_{1,2}\right)}{\mathrm{count}\left(\Omega_1\right)}\log_2\left(\frac{\mathrm{count}\left(\Omega_{1,2}\right)}{\mathrm{count}\left(\Omega_1\right)}\right) \tag{14}$$

同理，可以计算得到 Ω_2 子集信息熵。

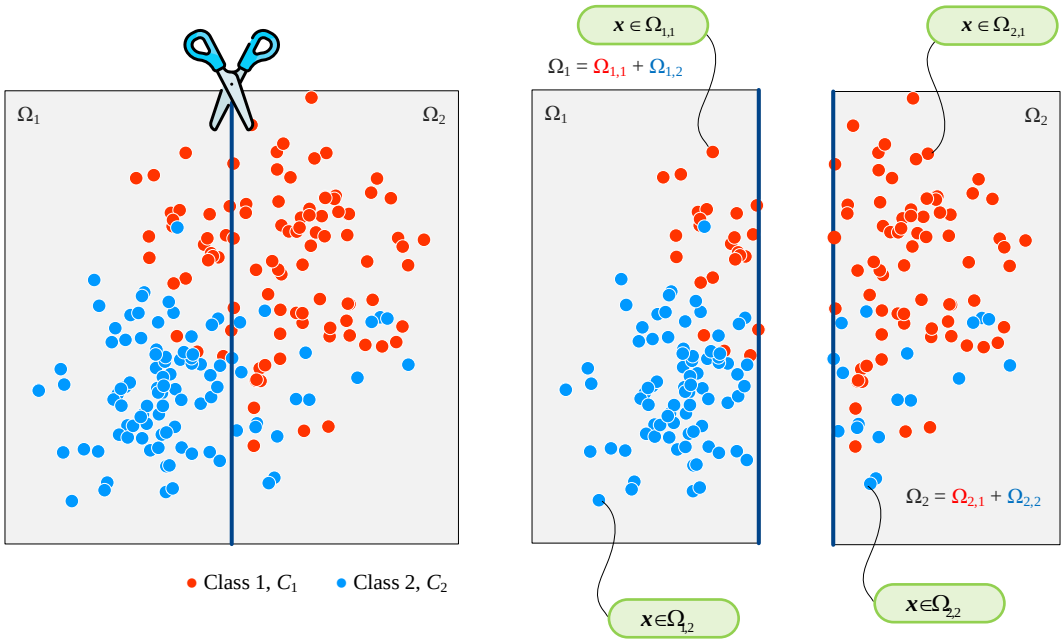


图 5. 数据集 Ω 划分为 2 个子集

信息增益

信息增益 (information gain) 量化划分前后信息熵变化：

$$\text{Gain}(D,a) = \underbrace{\text{Ent}(D)}_{\substack{\text{Entropy}\\ \text{before split}}} - \underbrace{\text{Ent}(D|a)}_{\substack{\text{Weighted sum of}\\ \text{entropy after split}}} \tag{15}$$

最佳划分 a 位置对应最大化信息增益：

$$\arg\max_{a} \text{Gain}(D,a) \tag{16}$$

9.4 基尼指数：指数越大，不确定性越高

类似信息熵，**基尼指数** (Gini index) 也可以用来表征样本数据集合 Ω 纯度。注意，这个基尼指数不同于衡量国家或地区收入差距的基尼指数。

基尼指数 Gini(Ω) 定义如下：

$$\text{Gini}(\Omega) = \sum_{i=1}^{n} p_i(1-p_i) = \sum_{i=1}^{n} p_i - \sum_{i=1}^{n} p_i^2 = 1 - \sum_{i=1}^{n} p_i^2 \tag{17}$$

类似上节，当样本数据集合 Ω 只有两类 $K = 2$。这种情况下，$p_1 = p$，$p_2 = 1 - p$。Ω 的信息熵 Gini(Ω) 为。

$$\begin{aligned} \text{Ent}(D) &= 1 - \sum_{i=1}^{n} p_i^2 = 1 - p_1^2 - p_2^2 \\ &= 1 - p^2 - (1-p)^2 = -2p^2 + 2p \end{aligned} \tag{18}$$

如图 6 (a) 所示，Gini(Ω) 越大，不确定性越高，数据纯度越低。Gini(Ω) 最大值为 1/2，对应图中 $p = 0.5$，也就是两类标签样本数据各占一半。图 6 (b) 比较 2 × Gini(Ω) 和 Ent(Ω) 两图形关系。

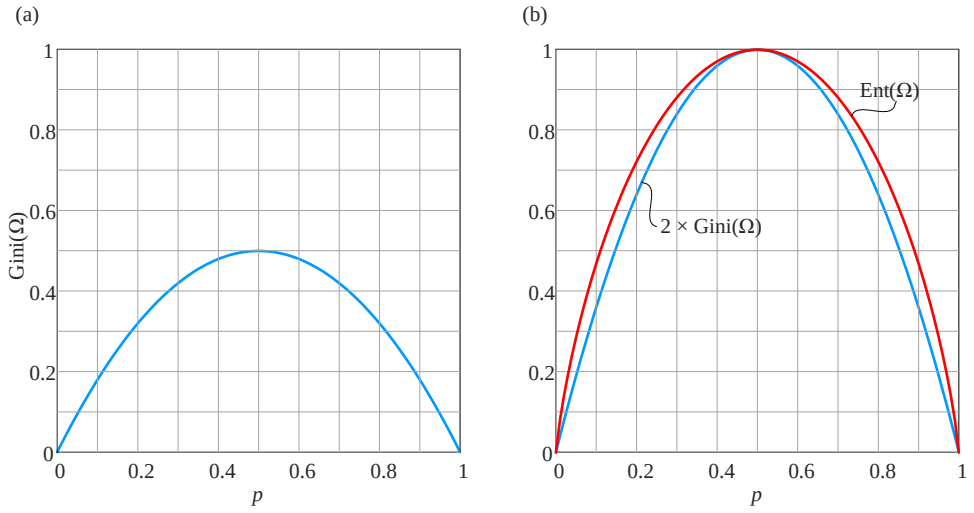


图 6. 比较信息熵和 Gini 指数图像

Scikit-learn 中决策树分类函数 `DecisionTreeClassifier`，就是默认采用 Gini 指数最大化作为分割依据。

9.5 最大叶节点：影响决策边界

本节利用决策树算法分类鸢尾花样本数据，并着重展示最大叶节点数分类影响。Scikit-learn 工具包决策树分类函数为 `sklearn.tree.DecisionTreeClassifier`；该函数可以用最大叶节点数 `max_leaf_nodes` 控制决策树树形大小。

同时，本节和下一节利用 `sklearn.tree.plot_tree` 绘制决策树。

最大叶节点数为 2

图 7 所示为当最大叶节点数 L 为 $L = 2$ 时，鸢尾花数据分类情况。图 7 (a) 所示，根据花萼长度 x_1 这一特征，特征平面被划分为两个区域——A 和 B。

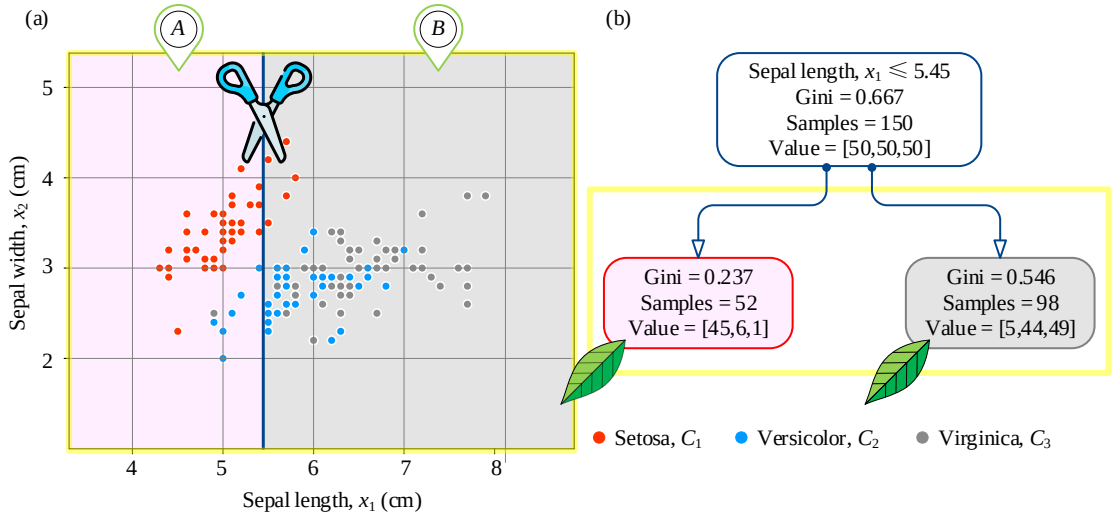


图 7. 最大叶节点数量为 2，（画出百分比，饼图）

图 7 (b) 树形图有大量重要信息。150 个样本数据 Gini 指数为 0.667。根绝 Gini 指数最大化原则，找到划分花萼长度 x_1 最佳位置为，$x_1 = 5.45$。$x_1 \leqslant 5.45$ 为区域 A；$x_1 > 5.45$ 为区域 B。

区域 A 中，样本数据 为 52；其中，● (C_1, $y = 0$) 为 45 个，● (C_2, $y = 1$) 为 6 个，● (C_3, $y = 2$) 为 1 个。显然，区域 A 预测分类为 C_1。区域 A 的 Gini 指数为 0.237。

区域 B 中，样本数据 为 98；其中，● (C_1, $y = 0$) 为 5 个，● (C_2, $y = 1$) 为 44 个，● (C_3, $y = 2$) 为 49 个。显然，区域 A 预测分类为 C_3。区域 B 的 Gini 指数为 0.546。

根据 (11)，可以计算得到特征 x_1 划分后信息熵：

$$\underbrace{\mathrm{Ent}\left(\Omega \middle| x_1 = 5.45\right)}_{\substack{\text{Weighted sum of} \\ \text{entropy after split}}} = \frac{52}{150} \times 0.237 + \frac{98}{150} \times 0.546 = 0.4389 \tag{19}$$

根据 (15) 信息增益为：

$$\mathrm{Gain}\left(D, a\right) = \underbrace{\mathrm{Ent}\left(D\right)}_{\substack{\text{Entropy} \\ \text{before split}}} - \underbrace{\mathrm{Ent}\left(\Omega \middle| x_1 = 5.45\right)}_{\substack{\text{Weighted sum of} \\ \text{entropy after split}}} = 0.667 - 0.4389 = 0.228 \tag{20}$$

最大叶节点数为 3

当最大叶节点数量 L 继续提高到 $L = 3$ 时，图 7 (b) 某一叶节点将会在某一特征基础上继续划分。图 8 所示为 $L = 3$，决策树分类鸢尾花结果。

观察图 8 (a)，可以发现图 7 (a) 中 B 区域沿着 x_1 方向进一步被划分为 C 和 D。划分的位置为 $x_1 = 6.15$。

区域 C 中，样本数据 为 43；其中，● (C_1, $y = 0$) 为 5 个，● (C_2, $y = 1$) 为 28 个，● (C_3, $y = 2$) 为 10 个。显然，区域 C 预测分类为 C_2。区域 C 的 Gini 指数为 0.508。

区域 D 中，样本数据 为 55；其中，● (C_1, $y = 0$) 为 0 个，● (C_2, $y = 1$) 为 16 个，● (C_3, $y = 2$) 为 39 个。显然，区域 A 预测分类为 C_3。区域 D 的 Gini 指数为 0.413。

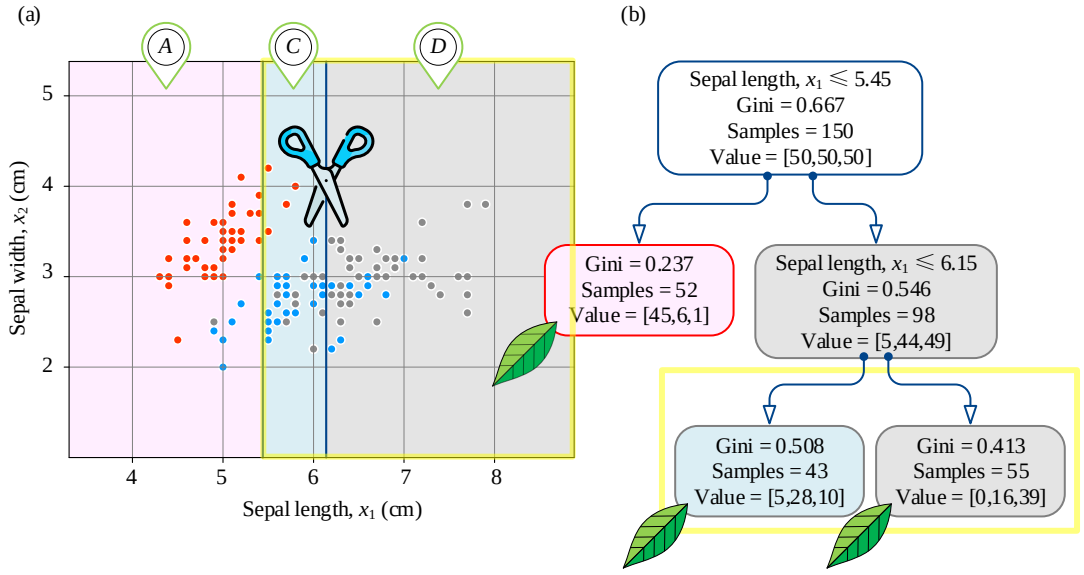


图 8. 最大叶节点数量为 3，（画出百分比，饼图）

最大叶节点数为 4

图 9 所示为最大叶节点数量 $L = 4$ 时，决策树分类结果和树形结构。可以发现图 8 中，A 区沿 x_2 方向被进一步划分为两个区域；其中一个区域 44 个 ● (C_1, $y = 0$)，1 个 ● (C_2, $y = 1$)，Gini 指数进一步降低到 0.043。请读者自行计算 Gini 指数变化。

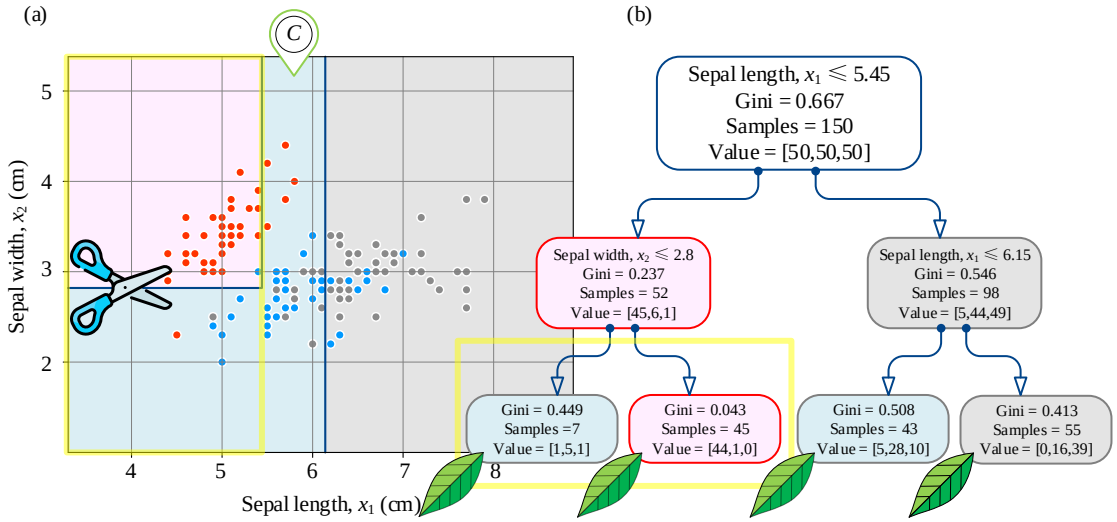


图 9. 最大叶节点数量为 4，（画出百分比，饼图）

最大叶节点数为 5

图 10 所示为最大叶节点数量 $L = 5$ 时，决策树分类结果和树形结构。比较图 10 和图 9，C 区沿 x_2 方向被进一步划分为两个区域，得到的一个区域全部样本数据为 ● (C_1, $y = 0$)；因此，该区域的 Gini 指数为 0，纯度最高。

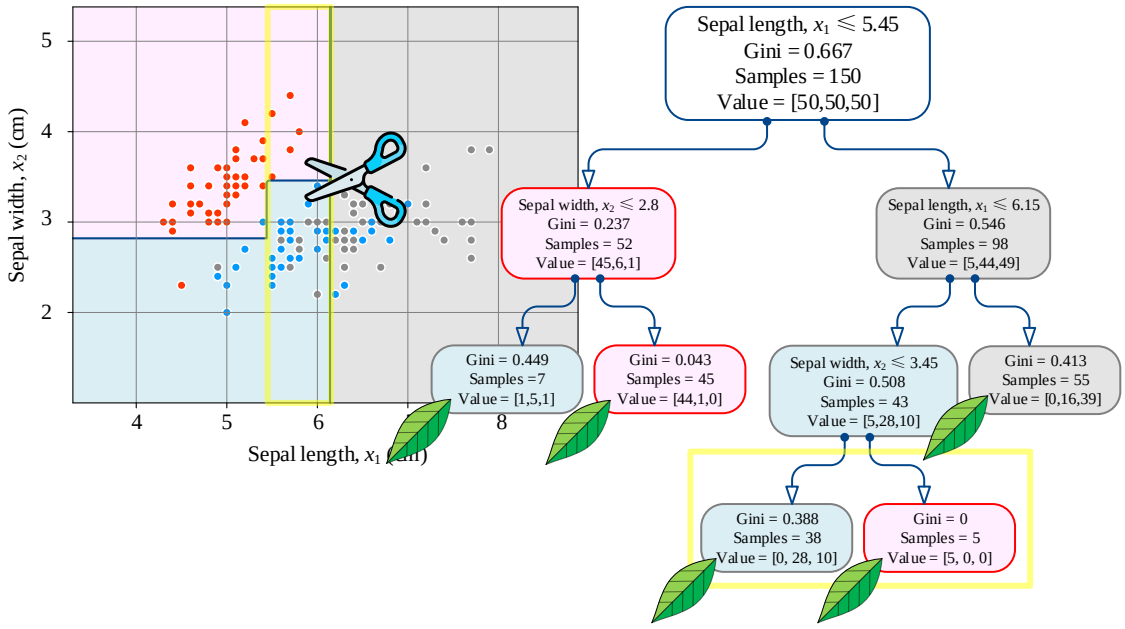


图 10. 最大叶节点数量为 5，（画出百分比，饼图）

下一节提供获得本节图像代码，代码中中最大叶节点数量包括 10、15 和 20 等更大数值。请大家自行设定最大叶节点数量，比较决策边界和树形结构变化。

9.6 最大深度：控制树形大小

类似最大叶节点数量，最大深度从二叉树层数角度控制树形大小。`sklearn.tree.DecisionTreeClassifier` 函数用 `max_depth` 改变最大深度。

图 11 所示为最大深度为 1 时，鸢尾花的分类结果和树形图。可以发现，图 11 和图 7 结果完全一致。图 12 所示为最大深度为 2 时，鸢尾花的分类结果和树形图。可以发现，图 12 和图 9 结果完全一致。

图 13 所示为最大深度为 3 时，鸢尾花分类结果。图 14 所示树形结构有 3 层二叉树。注意，当最大深度不断增大时，如果某一区域样本数据为单一样本；则该区域 Gini 指数为 0，无法进一步划分。图 14 中 8 个叶节点中，有 4 个纯度已经达到最高；

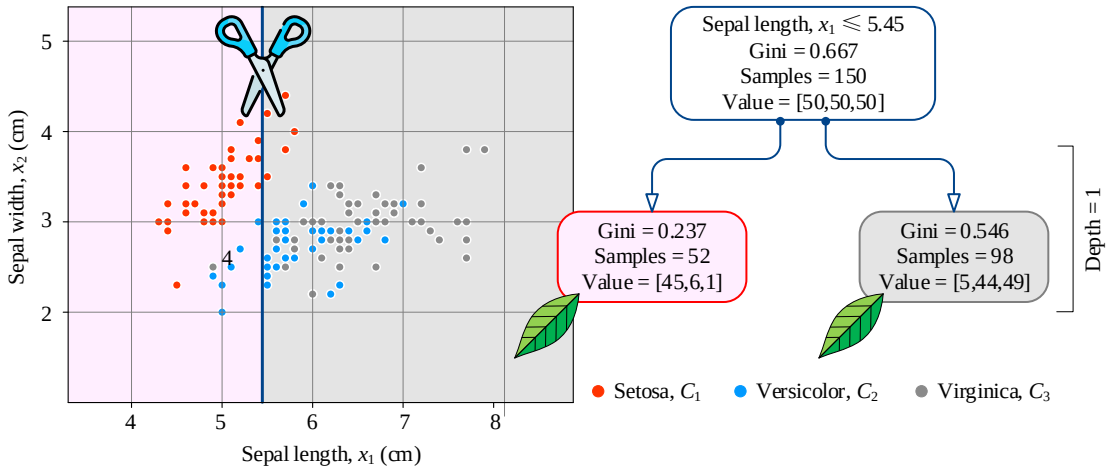


图 11. 最大深度为 1，（画出百分比，饼图）

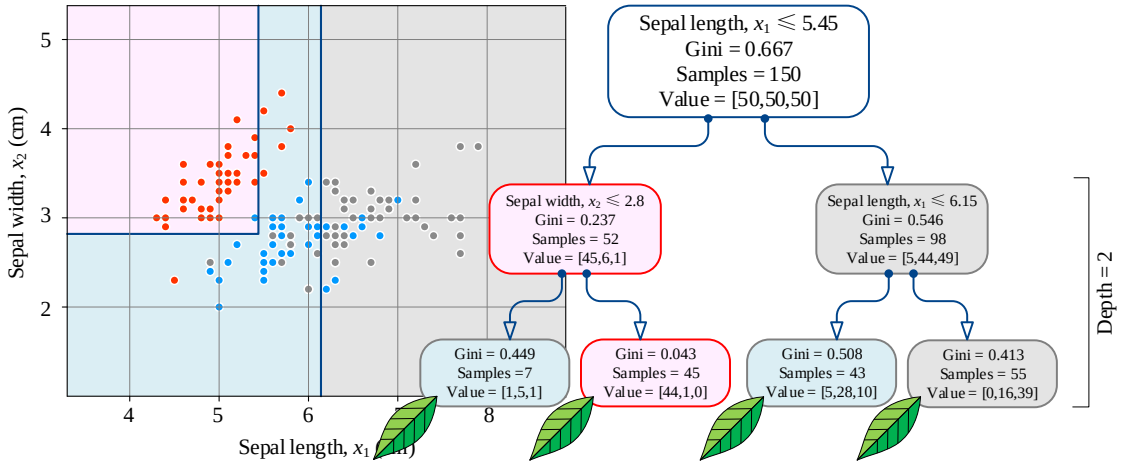


图 12. 最大深度为 2，（画出百分比，饼图）

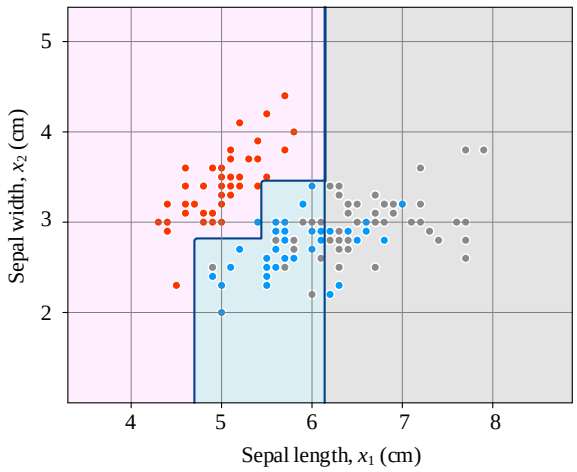


图 13. 最大深度为 3，分类结果

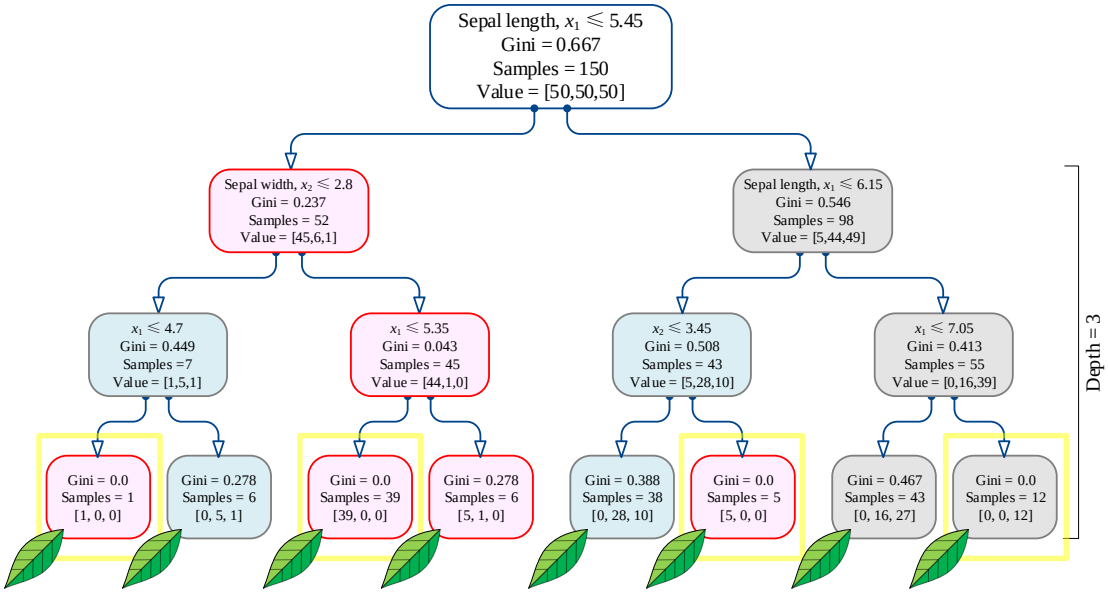


图 14. 最大深度为 3，树形结构，（画出百分比，饼图）

代码 Bk7_Ch09_01.py 利用决策树方法分类鸢尾花数据，并绘制本节和上一节图像。

决策树是一种基于树形结构的分类算法，其核心思想是通过一系列的问题来判断输入数据属于哪个类别。在构建决策树时，需要选择合适的划分特征和划分点。为了进行这些选择，常用的指标包括信息熵、信息增益和基尼指数。

信息熵是衡量样本纯度的指标，熵越高表示样本的混乱程度越高。信息增益是在使用某个特征进行分裂时，熵减少的程度，信息增益越大表示使用该特征进行划分所带来的纯度提升越大。基尼指数是另一种用于衡量样本纯度的指标，可以用来评估每个候选分裂点的优劣程度。

在构建决策树时，通常使用信息增益或基尼指数作为指标来选择最优的划分特征和划分点。不同的指标适用于不同的数据集和问题类型。在实践中，可以通过交叉验证等技术来选择最佳的指标。

10 Gaussian Process
高斯过程

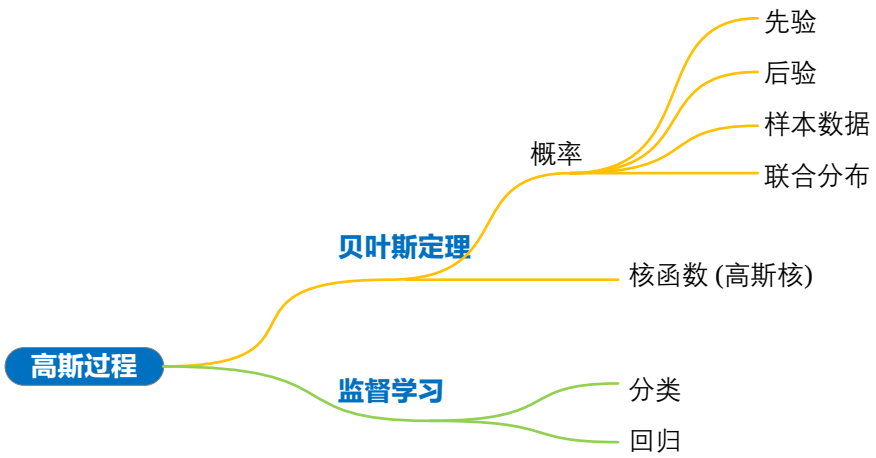
多元高斯分布的条件概率，协方差矩阵为核函数

生命就像一个永恒的春天，穿着崭新而绚丽的衣服站在我面前。

Life stands before me like an eternal spring with new and brilliant clothes.

—— 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

- `sklearn.gaussian_process.GaussianProcessRegressor()` 高斯过程回归函数
- `sklearn.gaussian_process.kernels.RBF()` 高斯过程高斯核函数
- `sklearn.gaussian_process.GaussianProcessClassifier()` 高斯过程分类函数



10.1 高斯过程原理

高斯过程 (Gaussian Process, GP) 既可以用来回归，又可以用来分类。高斯过程是一种概率模型，用于建模连续函数或实数值变量的概率分布。在高斯过程中，任意一组数据点都可以被视为多元高斯分布的样本，该分布的均值和协方差矩阵由先验信息和数据点间的相似度计算而得。通过高斯过程，可以对函数进行预测并对其不确定性进行量化，这使得其在机器学习、优化和贝叶斯推断等领域中被广泛应用。

在使用高斯过程进行预测时，通常使用条件高斯分布来表示先验和后验分布。通过先验分布和数据点的观测，可以计算后验分布，并通过该分布来预测新数据点的值。在高斯过程中，协方差函数或核函数起着重要的作用，它定义了数据点间的相似性，不同的核函数也适用于不同的应用场景。一些常见的核函数包括线性核、多项式核、高斯核、拉普拉斯核等。

高斯过程具有许多优点，如对噪声和异常值具有鲁棒性、能够对预测结果的不确定性进行量化、对于小样本也能够进行有效的预测等。然而，高斯过程的计算复杂度较高，通常需要通过一些技巧来提高其效率。

想要理解高斯过程，必须对多元高斯分布、条件概率、协方差矩阵、贝叶斯推断等数学工具烂熟于心。

《统计至简》第 11 章讲解多元高斯分布，第 12 章将讲解条件高斯分布，第 13 章介绍协方差矩阵，第 20、21 两章介绍贝叶斯推断，建议大家回顾。

先验

$\boldsymbol{x}_2$ 为一系列需要预测的点，$\boldsymbol{y}_2 = \mathrm{GP}(\boldsymbol{x}_2)$ 对应高斯过程预测结果。

高斯过程的先验为：

$$\boldsymbol{y}_2 \sim N\left(\boldsymbol{\mu}_2, \boldsymbol{K}_{22}\right) \tag{1}$$

其中，$\boldsymbol{\mu}_2$ 为高斯过程的均值，$\boldsymbol{K}_{22}$ 为协方差矩阵。之所以写成 $\boldsymbol{K}_{22}$ 这种形式，是因为高斯过程的协方差矩阵通过核函数定义。

本章主要利用高斯核 (Gaussian kernel)，也叫径向基核 RBF。

在 Scikit-learn 中，高斯核的定义为：

$$\kappa\left(x_i, x_j\right) = \exp\left(-\frac{\left(x_i - x_j\right)^2}{2l^2}\right) \tag{2}$$

当输入为多元的情况，上式分子为向量差的欧氏距离平方，即 $\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2$。图 1 所示为 $l = 1$ 时，先验协方差矩阵的热图。

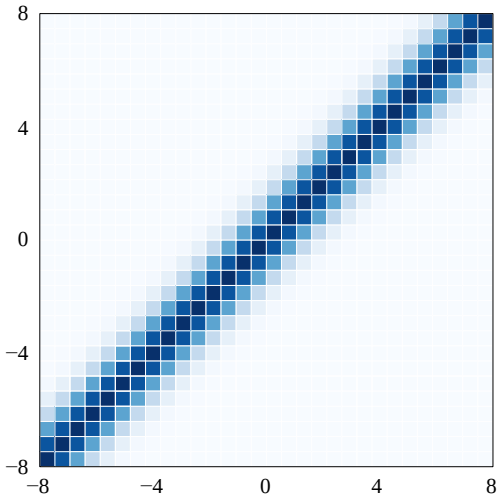


图 1. 高斯过程的先验协方差矩阵，高斯核

图 2 所示为参数 l 对先验协方差矩阵的影响。

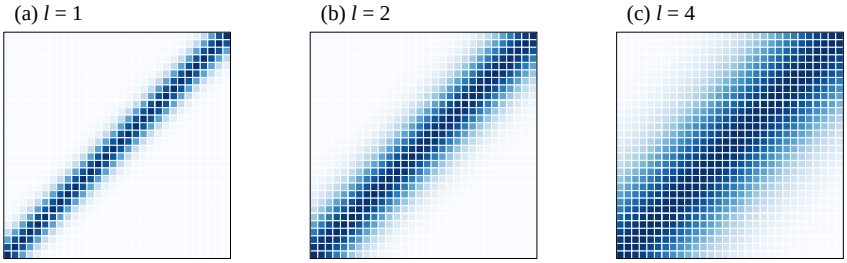


图 2. 先验协方差矩阵随着参数 l 变化，高斯核

很多其他文献上中高斯核定义为：

$$\kappa\left(x_i, x_j\right)=\sigma^2 \exp\left(-\frac{\left(x_i-x_j\right)^2}{2l^2}\right) \tag{3}$$

本章采用 Scikit-learn 中高斯核的定义。

本书前文介绍过，核函数本身实际上可以看做一个格拉姆矩阵。协方差矩阵也是格拉姆矩阵的一种特例。

⚠ 注意，高斯过程可以选择的核函数有很多。此外，不同核函数还可以叠加组合。

图 3 所示每一条代表一个根据当前均值、协方差的函数采样。打个比方，在没有引入数据之前，图 3 的曲线可以看成是一捆没有扎紧的丝带，随着微风飘动。

图 3 中的红线为高斯过程的均值，本章假设均值为 0。

《统计至简》第 9 章介绍过“68-95-99.7 法则”，图 3 中$\pm 2\sigma$ 对应约 95%。即约 95%样本位于距平均值 1 正负 2 个标准差之内。

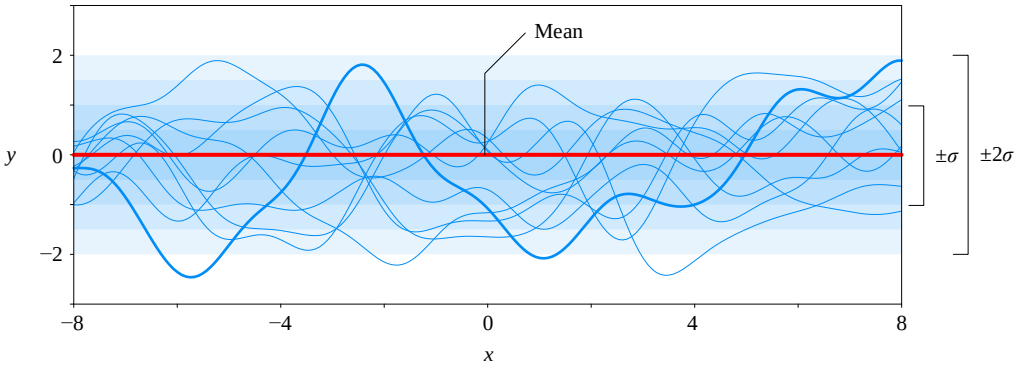


图 3. 高斯过程的采样函数，高斯核

样本

观测到的样本数据为 $(\boldsymbol{x}_1, \boldsymbol{y}_1)$。图 4 给出两个样本点，它们相当于扎紧丝带的两个节点。

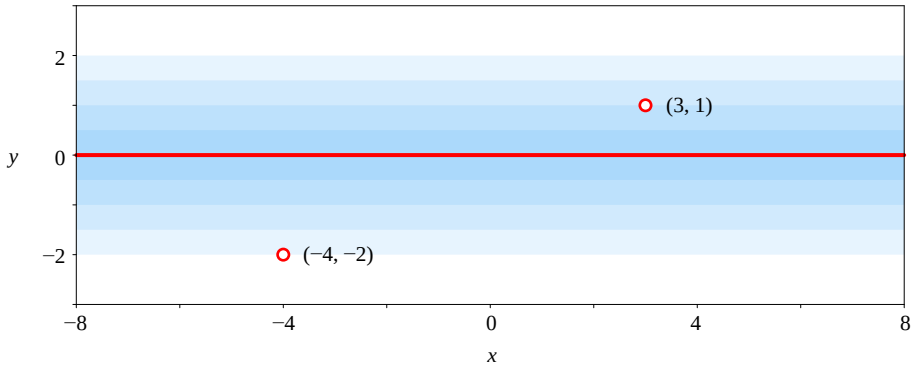


图 4. 给定样本数据 {(−4, −2), (3, 1)}

联合分布

假设样本数据 $\boldsymbol{y}_1$ 和预测值 $\boldsymbol{y}_2$ 服从联合高斯分布：

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} \end{bmatrix} \right) \tag{4}$$

简单来说，高斯过程对应的分布可以看成是无限多个随机变量的联合分布。图 5 中的协方差矩阵来自 $[\boldsymbol{x}_1, \boldsymbol{x}_2]$ 的核函数。

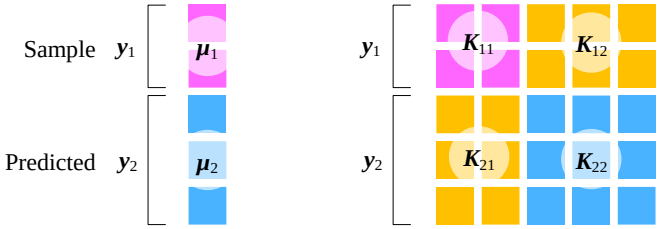


图 5. 样本数据 y_1 和预测值 y_2 服从联合高斯分布

后验

后验分布为：

$$f\left(\boldsymbol{y}_2 \mid \boldsymbol{y}_1\right) \sim N\left(\underbrace{\boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\left(\boldsymbol{y}_1-\boldsymbol{\mu}_1\right)+\boldsymbol{\mu}_2}_{\text{Expectation}}, \underbrace{\boldsymbol{K}_{22}-\boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\boldsymbol{K}_{12}}_{\text{Covariance matrix}}\right) \tag{5}$$

图 6 所示为引入样本数据 {(−4, −2), (3, 1)} 后，后验协方差矩阵的热图。

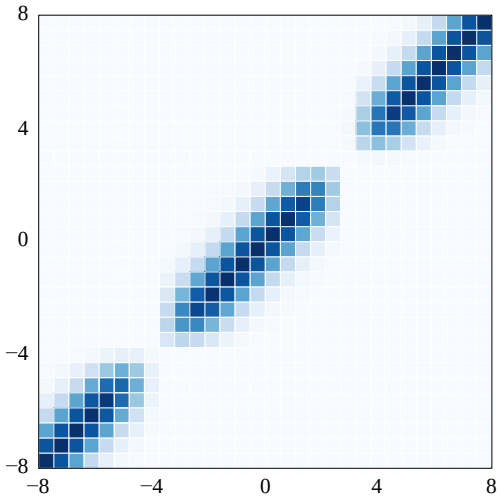


图 6. 高斯过程的后验协方差矩阵，高斯核

如图 7 所示，样本点位置丝带被锁紧，而其余部分丝带仍然舞动。

图 7 中红色曲线对应后验分布的均值：

$$\boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\left(\boldsymbol{y}_1-\boldsymbol{\mu}_1\right)+\boldsymbol{\mu}_2 \tag{6}$$

图 7 中带宽对应一系列标准差，它们的方差为：

$$\operatorname{diag}\left(\boldsymbol{K}_{22}-\boldsymbol{K}_{21}\boldsymbol{K}_{11}^{-1}\boldsymbol{K}_{12}\right) \tag{7}$$

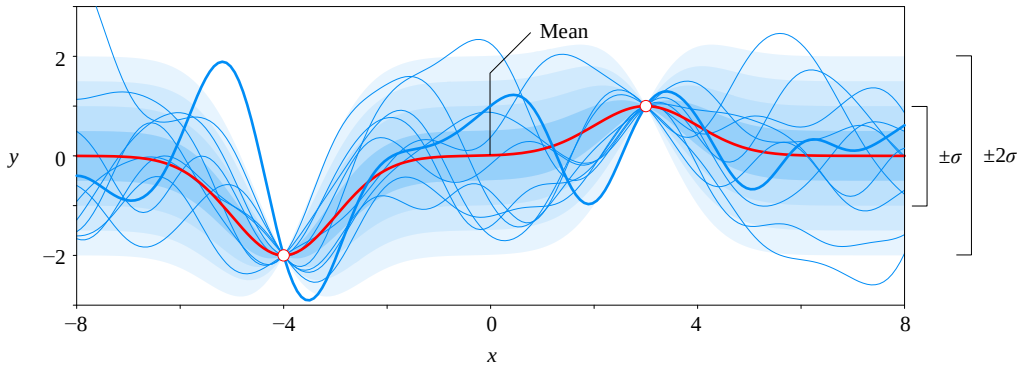


图 7. 高斯过程后验分布的采样函数，高斯核

其他几组情况

图 8 所示为随着样本不断增加，后验概率分布的协方差矩阵热图、高斯过程后验分布采样函数变化情况。

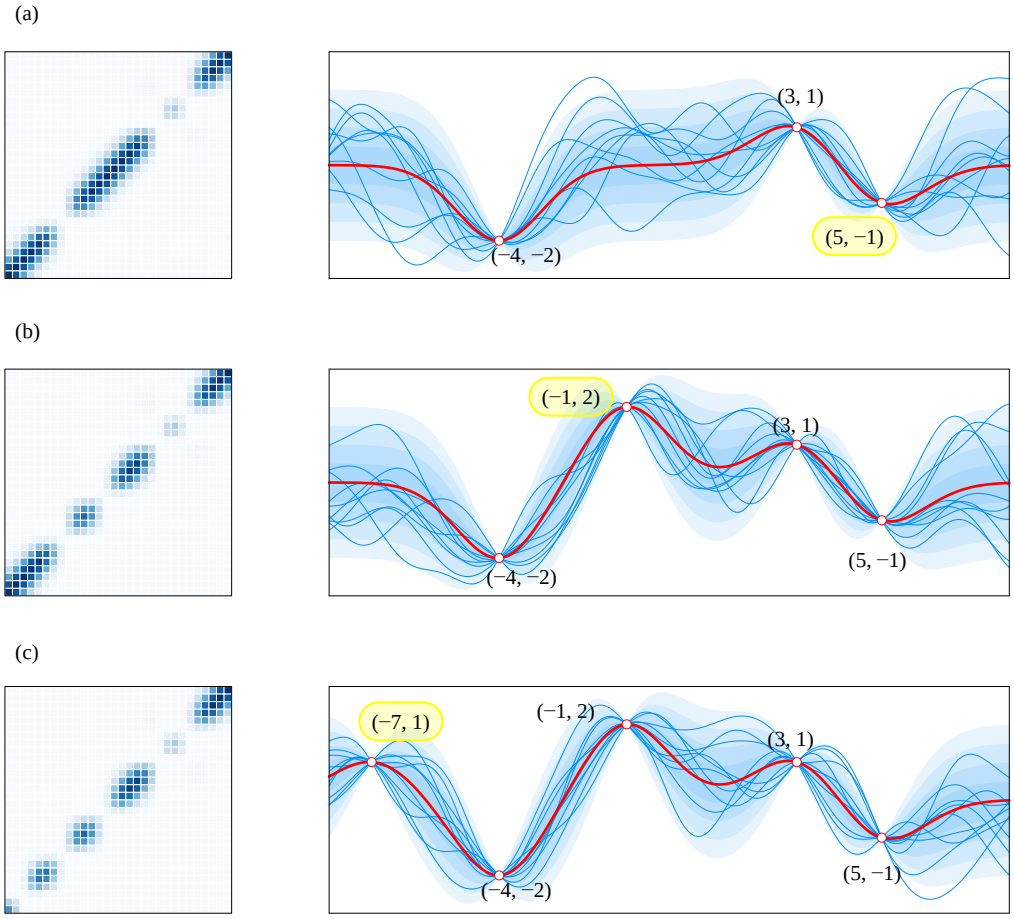


图 8. 样本数据不断增加

10.2 解决回归问题

Scikit-learn 解决回归问题的函数为 sklearn.gaussian_process.GaussianProcessRegressor()。

图 9 (a) 中蓝色曲线为真实曲线，对应函数为 $f(x) = x\sin(x)$。图 9 (a) 中红色点为样本点，蓝色曲线为高斯过程回归曲线，浅蓝色宽带为 95%置信区间。

图 9 (b) 所示为在样本点上加上噪音后的回归结果。

这个例子中使用的是高斯核函数，对应 sklearn.gaussian_process.kernels.RBF()。请大家条件参数，观察对回归结果的影响。此外，请大家试着使用其他核函数，并比较回归曲线。

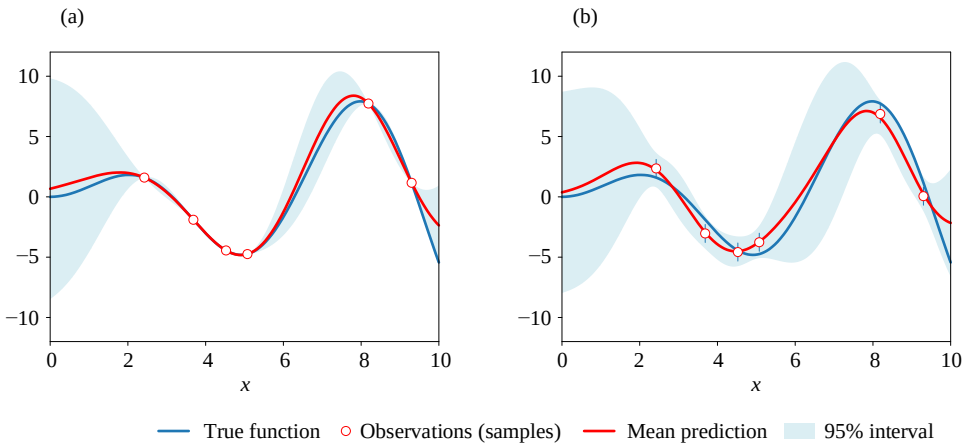


图 9. 使用高斯过程完成回归

本节高斯过程回归参考如下 Scikit-learn 示例，请大家自行学习：

https://scikit-learn.org/stable/auto_examples/gaussian_process/plot_gpr_noisy_targets.html

10.3 解决分类问题

sklearn.gaussian_process.GaussianProcessClassifier() 是 Scikit-learn 中专门用来解决高斯过程分类的函数。本例利用这函数根据花萼长度、花萼宽度分类鸢尾花。图 10 所示为采用高斯过程分类得到的决策边界。图 11 所示为三个后验曲面三维曲面和平面等高线。

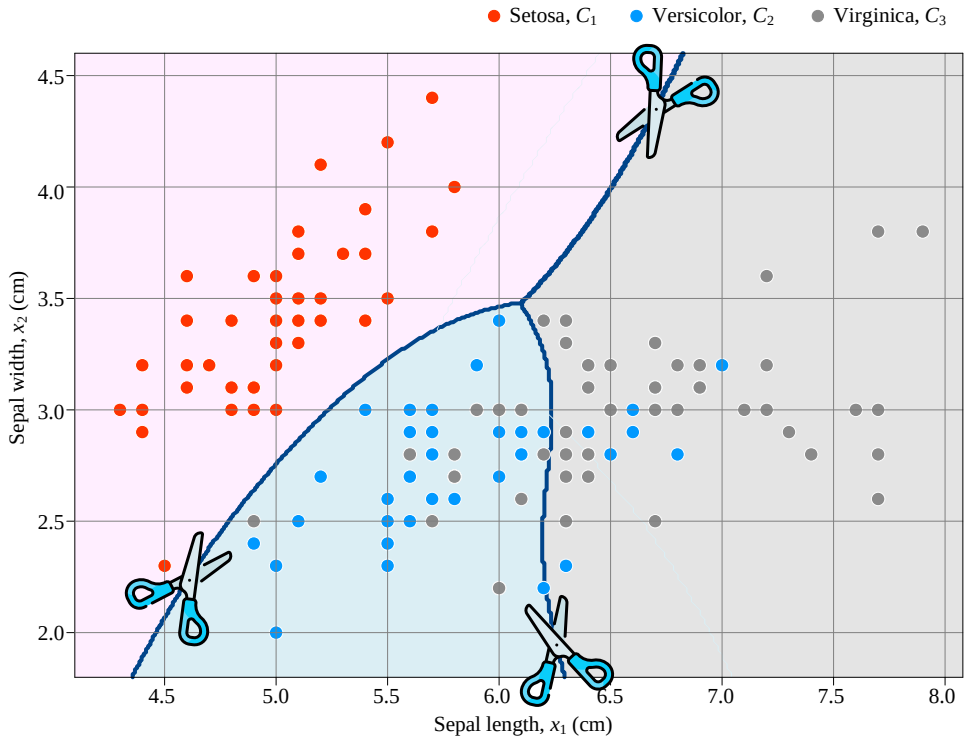


图 10. 使用高斯过程完成鸢尾花分类

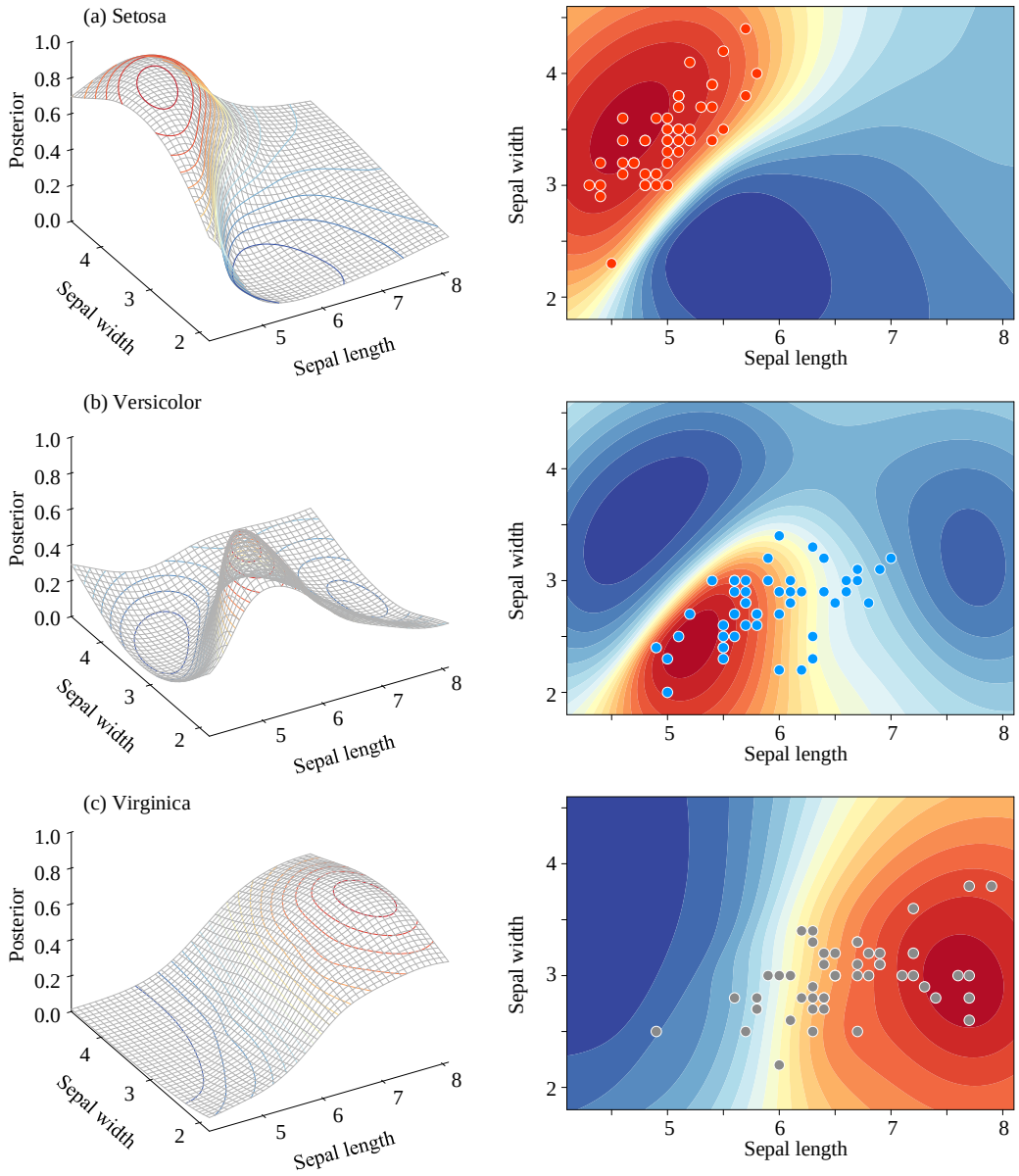
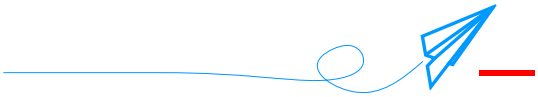


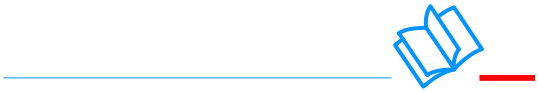
图 11. 后验概率曲面

Bk7_Ch10_01.py 利用高斯过程完成鸢尾花分类，并绘制图 10 和图 11。



高斯过程是一种基于概率论的非参数模型，可以用于建模连续函数或实数值变量的概率分布。在高斯过程中，先验分布通过核函数和一些超参数来定义，数据点的观测可以通过似然函数与先验分布相结合，计算后验分布。高斯过程中的核函数通常定义了数据点之间的相似性，超参数可以通过最大化似然或最大化边缘似然来优化。

在回归问题中，高斯过程可以用于预测连续变量的值，并估计预测值的不确定性。在分类问题中，高斯过程分类器可以将先验分布定义为高斯分布，通过后验分布进行分类预测。高斯过程的优点包括模型具有灵活性、能够对预测结果的不确定性进行量化、对噪声和异常值具有鲁棒性等。但高斯过程的计算复杂度较高，需要进行计算优化和近似方法。



大家想要深入学习高斯过程，请参考如下开源图书 *Gaussian Processes for Machine Learning*：

https://gaussianprocess.org/gpml/

这篇博士论文中专门介绍了不同核函数的叠加：

https://www.cs.toronto.edu/~duvenaud/thesis.pdf

作者认为下面这篇文章解释高斯过程做的交互设计最佳，这篇文章给了作者很多可视化方面的启发：

https://distill.pub/2019/visual-exploration-gaussian-processes/

11 Regression 回归

鸢尾花书有关回归算法模型的综述

卓越从来都不是偶然。卓越永远都是志存高远、百折不挠、有勇有谋的结果；它代表了明智之选。选择，而不是机会，决定了你的命运。

Excellence is never an accident. It is always the result of high intention, sincere effort, and intelligent execution; it represents the wise choice of many alternatives. Choice, not chance, determines your destiny.

—— 亚里士多德 (Aristotle) | 古希腊哲学家 | 384 ~ 322 BC

- `sklearn.linear_model.ElasticNet()` 求解弹性网络回归问题
- `sklearn.linear_model.lars_path()` 生成 Lasso 回归参数轨迹图
- `sklearn.linear_model.Lasso()` 求解套索回归问题
- `sklearn.linear_model.LinearRegression()` 最小二乘法回归
- `sklearn.linear_model.LogisticRegression()` 逻辑回归函数，也可以用来分类
- `sklearn.linear_model.Ridge()` 求解岭回归问题
- `sklearn.neighbors.NearestCentroid` 最近质心分类算法函数
- `sklearn.preprocessing.PolynomialFeatures()` 建模过程中构造多项式特征
- `statsmodels.api.add_constant()` 线性回归增加一列常数 1
- `statsmodels.api.OLS()` 最小二乘法函数

11.1 一张“回归”版图

机器学习中的回归 (regression) 是一种利用已知数据来预测未知数据的方法，常用于预测数值型的结果。回归模型通过分析数据中的变量之间的关系，得到一种数学模型，可以对未知数据进行预测。回归模型可以分为线性回归和非线性回归两种，其中线性回归模型假设变量之间存在线性关系，而非线性回归模型则允许变量之间存在更为复杂的关系。

图 1 总结本系列丛书介绍的各种回归算法。本章就按照这个“回归”版图展开讲解。这幅图也就是本章的思维导图。

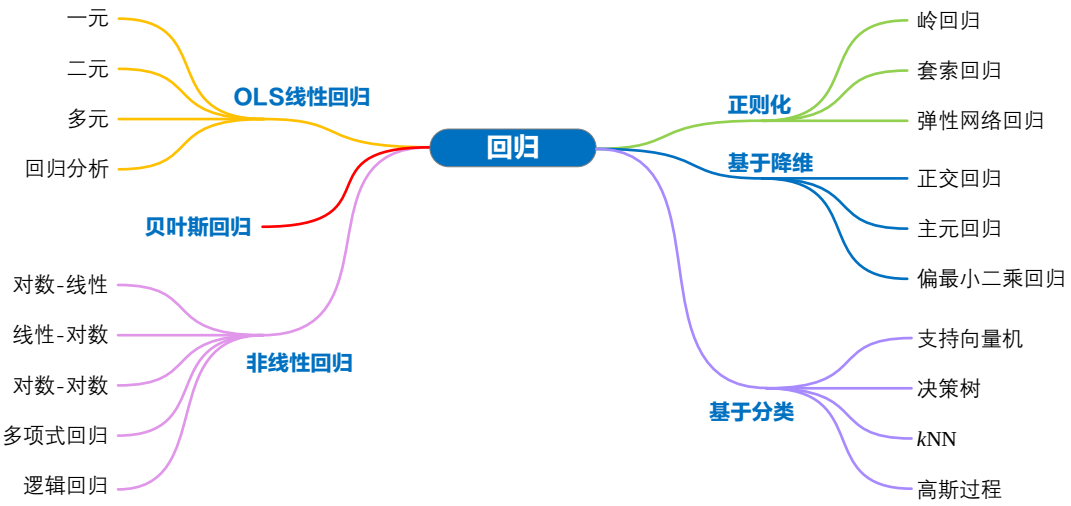


图 1. 回归方法分类

最小二乘算法

相信大家对于最小二乘 (Ordinary Least Squares, OLS) 线性回归已经烂熟于心。本章想强调如下几点。

首先，希望大家能够从多重视角理解 OLS 线性回归，比如优化 (图 2)、条件概率 (图 3)、几何 (图 4)、投影 (图 5)、数据、线性组合、SVD 分解、QR 分解、最大似然 MLE、最大后验 MAP 等视角。

此外，回归模型不能拿来就用，需要通过严格的回归分析。

再提到 OLS 线性回归时，希望大家闭上眼睛，脑中不仅仅浮现各种多彩的图像，而且能够用 OLS 线性回归把代数、几何、线性代数、概率统计、优化等数学板块有机地联结起来！

丛书讲解 OLS 线性回归时可谓抽丝剥茧、层层叠叠。对于这些视角感到生疏的话，请回归《数学要素》第 24 章、《矩阵力量》第 9、25 两章、《统计至简》第 24 章、《数据有道》第 10、11 两章。

线性回归常用来预测。代码 Bk7_Ch11_01.py 给出一个例子，根据股票日收益率协方差矩阵，推断如果股票指数上涨或下跌，其他股票价格变动情况。这个回归分析是单变量，但是多输出，也就是说一个 x、多个 y。请大家自行学习。

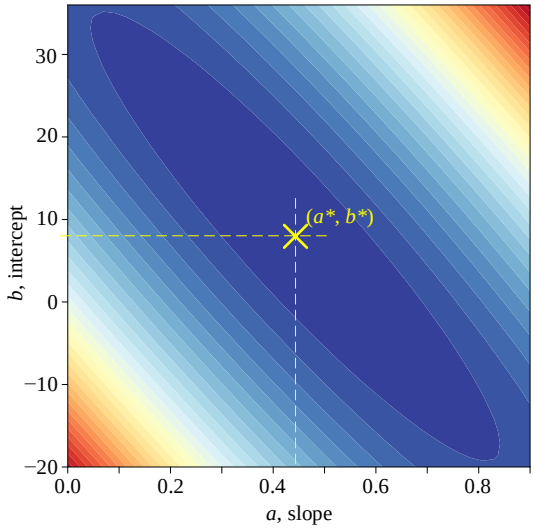


图 2. 一元 OLS 回归目标函数，图片来自《数学要素》第 24 章

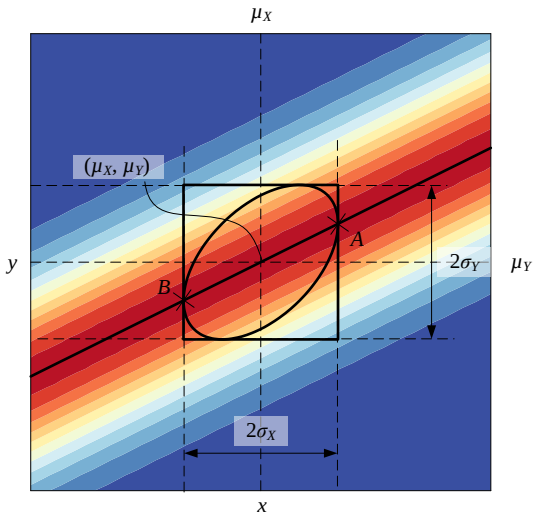


图 3. 条件期望视角看 OLS 线性回归，图片来自《统计至简》第 12 章

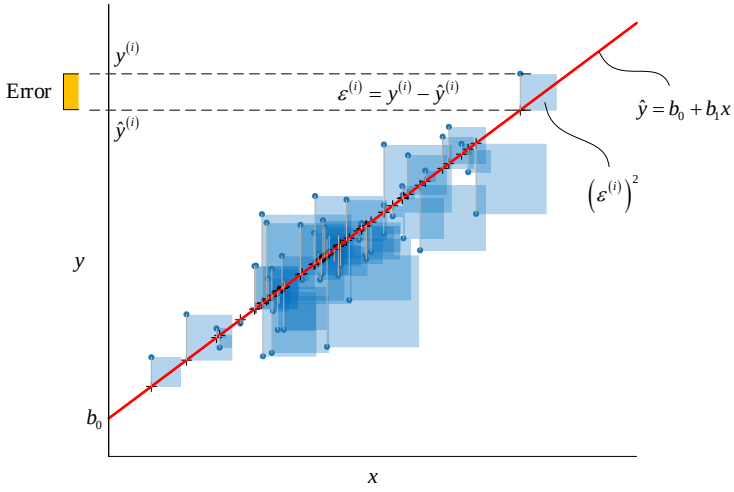


图 4. 残差平方和的几何意义，图片来自《统计至简》第 24 章

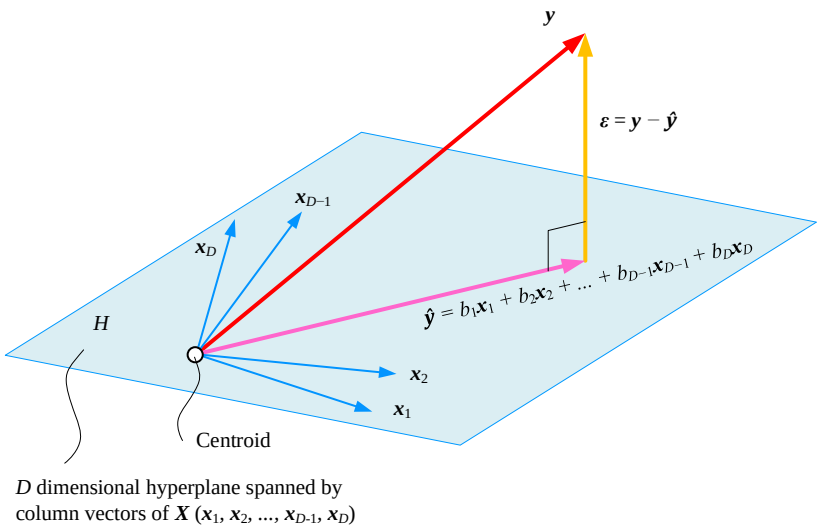


图 5. 投影角度解释多元最小二乘法线性回归，图片来自《数据有道》第 11 章

贝叶斯回归

贝叶斯回归基于贝叶斯推断 (Bayesian inference)。

贝叶斯推断把模型参数看作随机变量。根据主观经验和既有知识给出未知参数的概率分布，称为先验分布。从总体中得到样本数据后，根据贝叶斯定理，基于给定的样本数据，得出模型参数的后验分布。

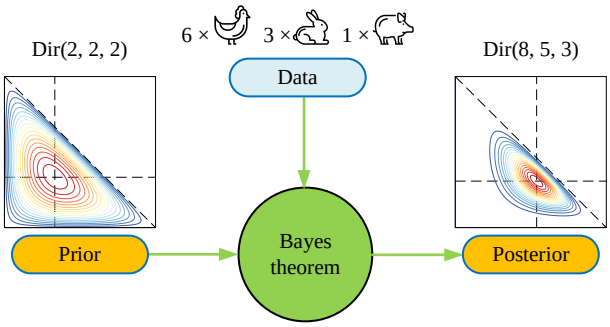


图 6. 先验 Dir(2, 2, 2) + 样本 → 后验 Dir(8, 5, 3)，图片来自《统计至简》第 22 章

贝叶斯回归的优化问题对应最大后验 MAP。贝叶斯推断中，后验 ∝ 似然 × 先验，是最重要的关系，希望大家牢记。希望大家掌握利用 PyMC3 完成贝叶斯回归参数模拟。

欢迎大家回顾《统计至简》第 20、21、22 三章有关贝叶斯推断的内容。此外，请大家回顾《数据有道》第 13 章有关贝叶斯回归。

非线性回归

非线性回归 (nonlinear regression) 寻找因变量和自变量之间关系的非线性模型的方法。

《数据有道》第 14、15 章专门介绍非线性回归。

请大家特别注意，逻辑回归不但可以用来回归，也可以用来分类。

此外，大家会发现 *k*-NN、高斯过程算法完成的回归也都可以归类为非线性回归。

正则化

正则化是多元回归的重要技术之一。正则化能够降低多重共线性，有效解决模型过拟合问题，提高泛化能力。请大家注意正则化和范数的关系。

对这部分内容感到生疏的话，请回顾《数据有道》第 2 章。此外，《数据有道》第 13 章还介绍如何从贝叶斯推断角度理解正则化。

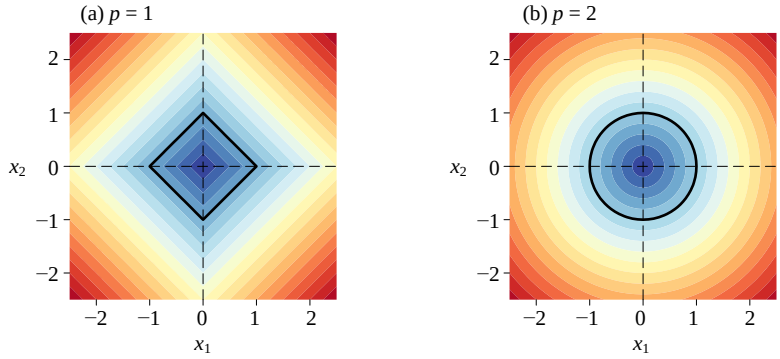


图 7. 两个范数

基于降维算法的回归

《数据有道》第 18、19 章特别介绍两种基于主成分分析的回归方法——正交回归、主元回归。

平面上，最小二乘法线性回归 OLS 仅考虑纵坐标方向上误差，如图 8 (a) 所示；而正交回归 TLS 同时考虑横纵两个方向误差，如图 8 (b) 所示。

主元回归的因变量则来自于主成分分析结果。

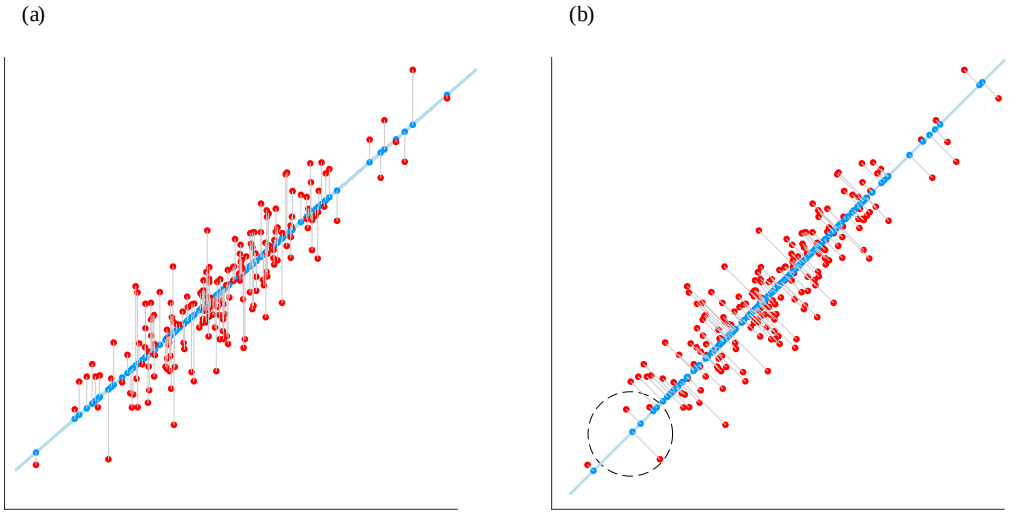


图 8. 对比 OLS 和 TLS 线性回归，图片来自《数据有道》第 18 章

基于分类算法的回归

实际上，监督学习的很多算法都兼顾分类、回归两项任务，比如逻辑回归、*k*-NN、支持向量机、高斯过程等等。本章下一节给出一个 *k*-NN 回归的例子。

11.2 *k*-NN 回归：非参数回归

本书第 2 章介绍的 *k*-NN 分类算法针对离散标签，比如 C_1 (红色 ●) 和 C_2 (蓝色 ●)。当输出值 y 为连续数据时，监督学习便是回归问题。本节讲解如何利用 *k*-NN 求解回归问题。

对分类问题，一个查询点的标签预测是由它附近 k 个近邻中占多数的标签决定；同样，某个查询点的回归值，也是由其附近 k 个近邻的输出值决定。

采用等权重条件下，查询点 $\boldsymbol{q}$ 回归值 $\hat{y}$ 可以通过下式计算获得：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{k} \sum_{i \in kNN(\boldsymbol{q})} y^{(i)} \tag{1}$$

其中，$k\text{NN}(\boldsymbol{q})$ 为查询点 $\boldsymbol{q}$ 的 k 个近邻构成的集合。

举个例子

如图 9 所示，当 $k = 3$ 时，查询点 Q 附近三个近邻 $\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$ 和 $\boldsymbol{x}^{(3)}$ 标记为蓝色 ●。这三个点对应的连续输出值分别为 $y^{(1)}$、$y^{(2)}$ 和 $y^{(3)}$。根据 (1) 计算 $y^{(1)}$、$y^{(2)}$ 和 $y^{(3)}$ 平均值，得到查询点回归预测值 $\hat{y}$：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{3}\left(y^{(1)} + y^{(2)} + y^{(3)}\right) = \frac{1}{3}(5+3+4) = 4 \tag{2}$$

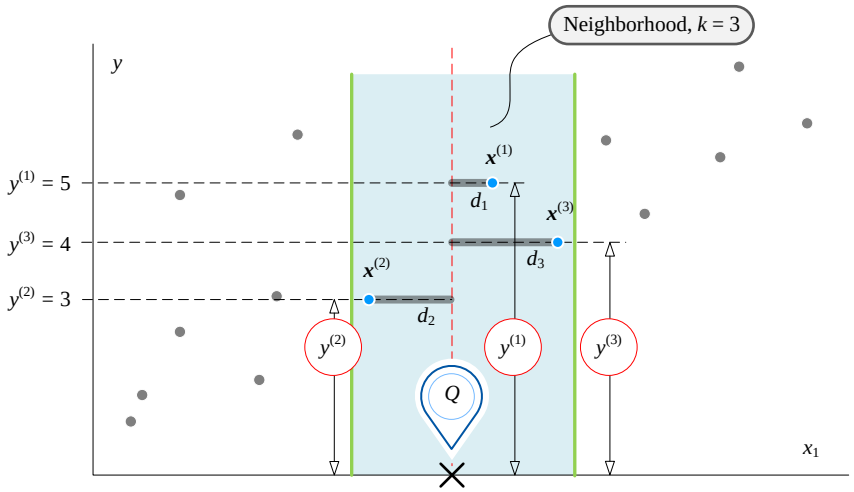


图 9. *k*-NN 回归算法原理

函数

`sklearn.neighbors.KNeighborsRegressor` 函数完成 *k*-NN 回归问题求解。默认等权重投票，`weights = 'uniform'`。

如果 *k*-NN 回归中考虑近邻投票权重，查询点 $\boldsymbol{q}$ 回归值 $\hat{y}$ 可以通过下式计算获得：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{\sum_{i \in kNN(\boldsymbol{q})} w_i} \sum_{i \in kNN(\boldsymbol{q})} w_i y^{(i)} \tag{3}$$

类似 *k*-NN 分类，`weights = 'distance'`设置样本数据权重与到查询点距离成反比。

图 10 所示为利用 *k*-NN 回归得到的不同种类鸢尾花花萼长度 x_1 和花萼宽度 x_2 回归关系。花萼宽度 x_2 相当于 (3) 中 y。图 10 (a) 采用等权重投票，图 10 (b) 中投票权重与查询点距离成反比。

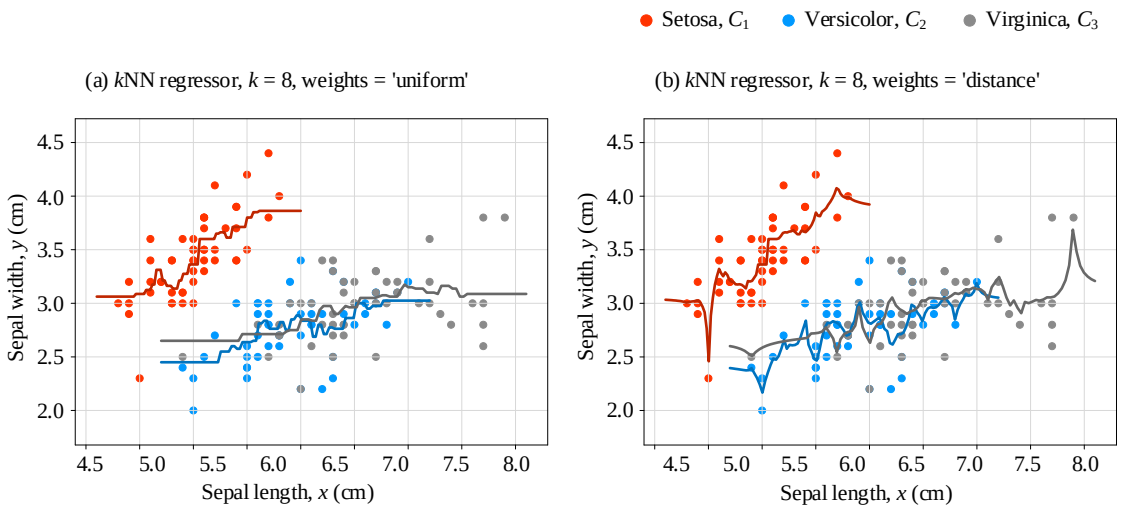


图 10. *k*-NN 回归，不同种类鸢尾花花萼长度和花萼宽度回归关系

代码 Bk7_Ch11_02.py 完成 *k*-NN 回归，并绘制图 10 两幅图像。

回归算法是一种机器学习技术，用于预测一个或多个连续型变量的值。其中 OLS 线性回归是最常见的回归算法之一，它基于最小二乘法来建立一个线性方程，使得预测值和实际值之间的差异最小化。

贝叶斯回归是一种基于贝叶斯定理的概率模型，能够通过考虑先验知识来提高预测准确性。非线性回归能够建立非线性模型来更准确地预测目标变量。正则化方法包括 L1 和 L2 正则化，可以在模型训练时加入惩罚项来避免过拟合。

基于 PCA 回归是一种使用主成分分析来降维和建立回归模型的技术。基于分类的回归方法如支持向量机、*k*NN 和决策树，通过将回归问题转化为分类问题来预测目标变量的值。每种回归算法都有其优缺点和适用场景，需要根据具体情况选择合适的方法。

利用支持向量机完成回归，请参考：

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html

利用决策树完成回归，请参考：

https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

12 K均值聚类

K-Means Clustering

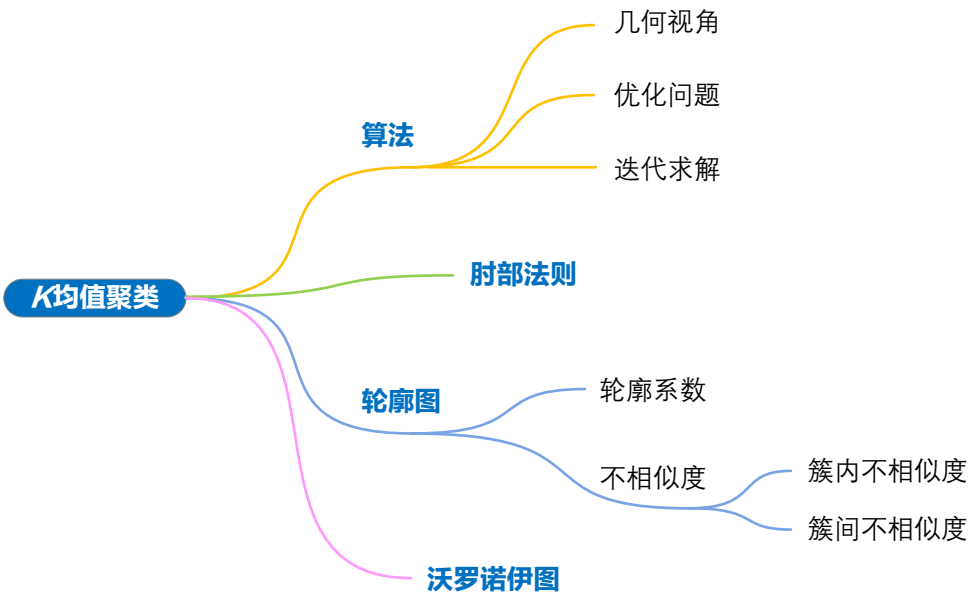
簇内距离和最小，迭代求解

几何是万物美的本原。

Geometry is the archetype of the beauty of the world.

—— 约翰内斯·开普勒 (Johannes Kepler) | 德国天文学家、数学家 | 1571 ~ 1630

- `numpy.cov()` 计算协方差矩阵
- `pandas.DataFrame.cov()` 计算数据帧协方差矩阵
- `scipy.spatial.Voronoi` 函数获得沃罗诺伊图相关数据
- `scipy.spatial.voronoi_plot_2d` 函数绘制沃罗诺伊图
- `sklearn.cluster.KMeans()` K 均值聚类算法函数；`model.fit()` 拟合数据，`model.predict()` 预测聚类标签，`model.cluster_centers_`，输出簇质心位置，`model.inertia_`输出簇 SSE 之和
- `sklearn.metrics.silhouette_score` 计算轮廓系数
- `yellowbrick.cluster.SilhouetteVisualizer` 函数绘制轮廓图



12.1 K均值聚类

K均值聚类 (*K*-means clustering) 的 K 不同于 k 近邻中的 k。

本书第 2 章介绍的 **k近邻算法** (*k*-Nearest Neighbors, *k*-NN) 是有监督学习分类算法，样本数据有标签，它的 k 是指设定的近邻数量。

而 K 均值聚类则是无监督学习聚类算法，样本数据无标签，K 是指将给定样本集 Ω 划分成 K 簇 $C = \{C_1, C_2, ... C_K\}$。

原理

图 1 所示为 K 均值算法原理图。K 均值聚类的每一簇样本数据用**簇质心** (cluster centroid) 来描述。比如，二聚类问题有两个簇质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$。如果以欧氏距离为距离度量，距离质心 $\boldsymbol{\mu}_1$ 更近的点，被划分为 C_1 簇；而距离质心 $\boldsymbol{\mu}_2$ 更近的点，被划分为 C_2 簇。

比如，图 1 中 A 点明显距离 $\boldsymbol{\mu}_1$ 更近，A 点被划分为 C_1 簇，C 点距离 $\boldsymbol{\mu}_2$ 更近，因此 C 点划分到 C_2 簇。B 点距离 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 相等，因此 B 点位于决策边界。很明显，决策边界为 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ **中垂线** (perpendicular bisector)。

建议大家回顾《矩阵力量》第 19 章讲解的有关中垂线内容。

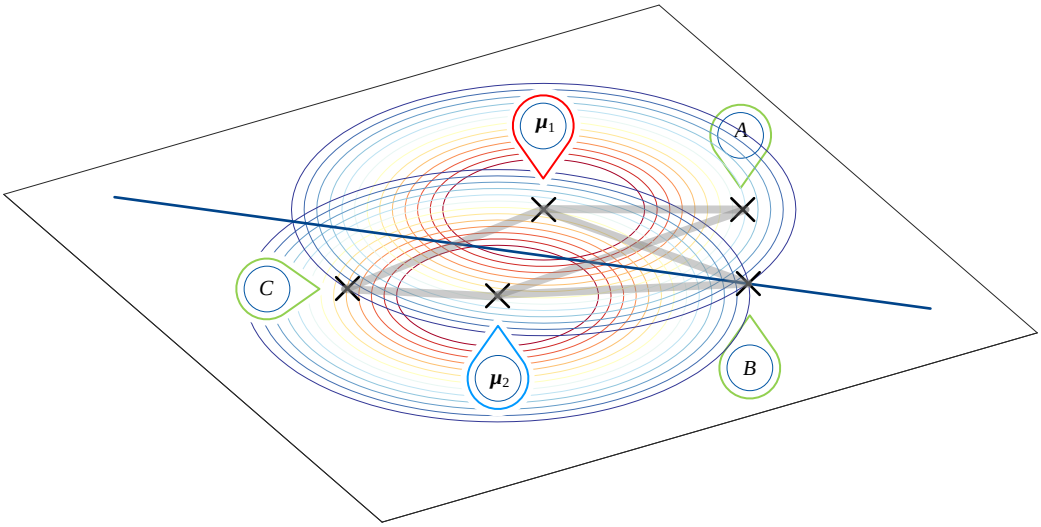


图 1. K 均值算法原理

由于采用欧氏距离，图 1 中簇质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 等高线为两组同心圆；同心圆颜色相同，代表距离簇质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 距离相同。因此，同色同心圆的交点位于决策边界上。

12.2 优化问题

K 均值聚类算法的优化目标是，将所有给定样本点划分 K 簇，并使得簇内距离平方和最小。采用最简单的欧拉距离，以上优化目标记做：

$$\arg\min_{C} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}_k \right\|^2 \tag{1}$$

其中，$\boldsymbol{x}$ 为样本数据任意一点，形式为列向量；$\boldsymbol{\mu}_k$ 为任意一簇 C_k 样本数据的质心。

实际上，(1) 中簇内距离平方和相当于**残差平方和** (Sum of Squared Error, SSE)。SSE 度量样本数据的聚集程度；为了方便读者理解，下一节专门讲解质心、协方差矩阵、残差平方和等描述簇数据的数学工具。

任意一点 $\boldsymbol{x}$ 和质心 $\boldsymbol{\mu}_k$ 欧氏距离平方，可以通过下式计算得到：

$$d^2 = \text{dist}\left(\boldsymbol{x}, \boldsymbol{\mu}_k\right)^2 = \left\| \boldsymbol{x} - \boldsymbol{\mu}_k \right\|^2 = \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right)^{\mathrm{T}} \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right) \tag{2}$$

其中，

$$\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_D \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{\mu}_k = \begin{bmatrix} \mu_{k,1} & \mu_{k,2} & \cdots & \mu_{k,D} \end{bmatrix}^{\mathrm{T}} \tag{3}$$

两特征聚类

当特征数为 $D = 2$ 时，欧氏距离平方和展开为下式：

$$\begin{aligned} d^2 &= \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right)^{\mathrm{T}} \left(\boldsymbol{x} - \boldsymbol{\mu}_k\right) \\ &= \left(x_1 - \mu_{k,1}\right)^2 + \left(x_2 - \mu_{k,2}\right)^2 \end{aligned} \tag{4}$$

丛书反复介绍，当 d 取某一定值时，(4) 所示解析式是以 $(\mu_{k,1}, \mu_{k,2})$ 为圆心的正圆，d 为半径。如图 1 所示，d 取不同值时，(4) 的几何表达为以 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 为中心得到两组同心正圆。

对于二分类问题，决策边界满足：

$$\left(\boldsymbol{x} - \boldsymbol{\mu}_1\right)^{\mathrm{T}} \left(\boldsymbol{x} - \boldsymbol{\mu}_1\right) = \left(\boldsymbol{x} - \boldsymbol{\mu}_2\right)^{\mathrm{T}} \left(\boldsymbol{x} - \boldsymbol{\mu}_2\right) \tag{5}$$

整理得到决策边界解析式。=：

$$\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right)^{\mathrm{T}} \left(\boldsymbol{x} - \frac{\left(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2\right)}{2} \right) = 0 \tag{6}$$

发现 K 均值聚类决策边界为一超平面，超平面通过 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 中点，并垂直于 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 连线，即垂直于$(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)$。

三聚类

图 2 所示为三聚类问题簇数据和质心位置。根据这三个质心位置，可以绘制两两质心的中垂线。决策边界在这三条中垂线上。图 2 实际上便是**沃罗诺伊图** (Voronoi diagram)。本章最后一节介绍沃罗诺伊图。

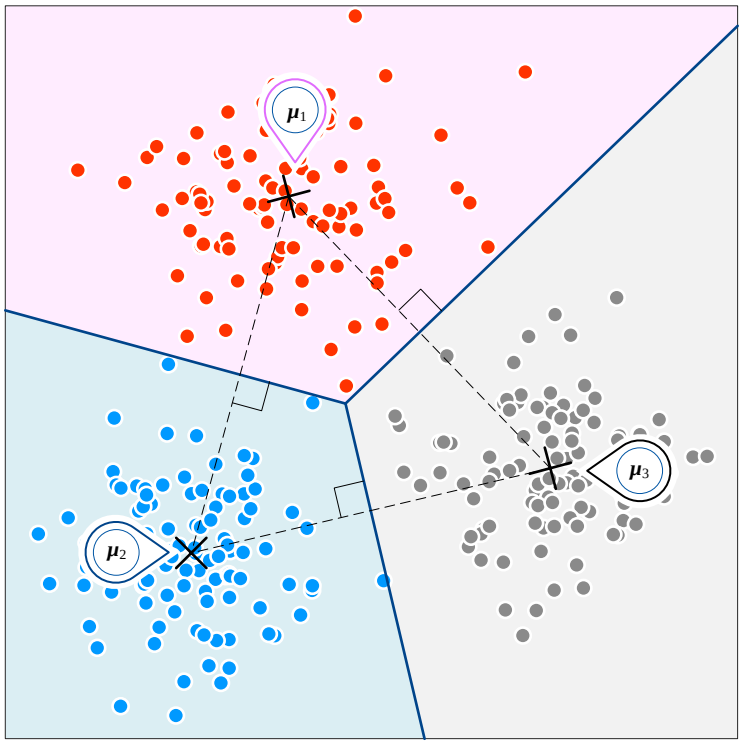


图 2. 三聚类问题簇质心、决策边界和区域划分

图 3 中，z 轴高度代表三聚类预测标签。

容易发现，在确定决策边界位置上，*K* 均值聚类原理和本书第 2 章介绍的**最近质心分类器** (Nearest Centroid Classifier) 很相似。

不同的是，*K* 均值聚类算法采用迭代方式找到簇质心位置，这是下一节要介绍的内容。

采用欧氏距离的 *K* 均值聚类相当于**高斯混合模型** (Gaussian Mixture Model, GMM) 的一个特例。这一点，下一章会详细介绍。

目前 Scikit-learn 中 *K* 均值聚类算法距离度量仅支持欧氏距离；MATLAB 中 *K* 均值聚类算法函数还支持城市街区距离、余弦距离、相关系数距离等。

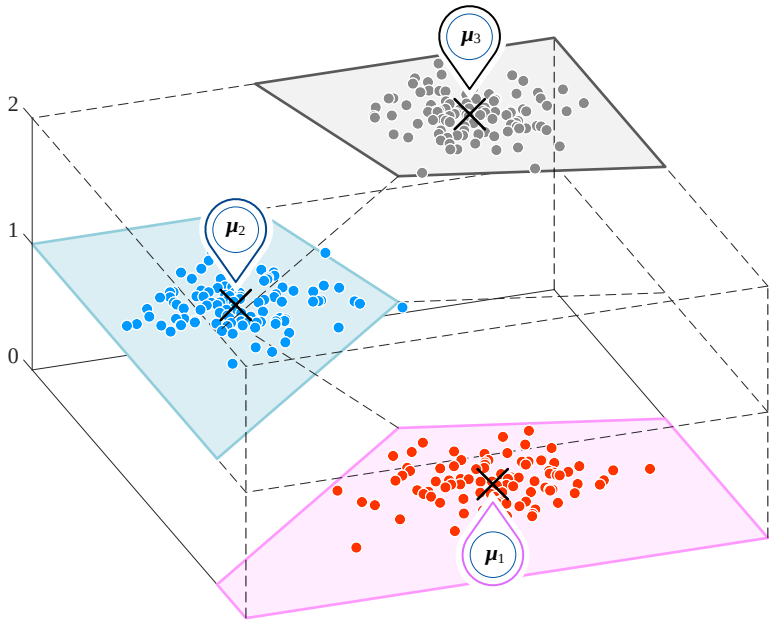


图 3. 三聚类预测标签

12.3 迭代过程

本节以二聚类为例介绍 K 均值聚类流程图。

流程

流程输入为样本数据和聚类簇数 (比如 2)。然后，从样本中随机选取 2 个数据作为初始簇质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$。然后进入如下迭代循环：

- 计算每一个样本和均值向量 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 距离；
- 比较每个样本和 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 距离，确定簇划分；
- 根据当前簇，计算并更新均值向量 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$

直到均值向量 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$ 满足迭代停止条件，得到簇划分。

图 4 所示为一鸢尾花数据为例 K 均值算法迭代过程。随机选取三个样本点 (黄色高亮) 作为初始簇质心 $\boldsymbol{\mu}_1$、$\boldsymbol{\mu}_2$ 和 $\boldsymbol{\mu}_3$，经过 10 次迭代，簇质心位置不断连续变化，最终收敛。

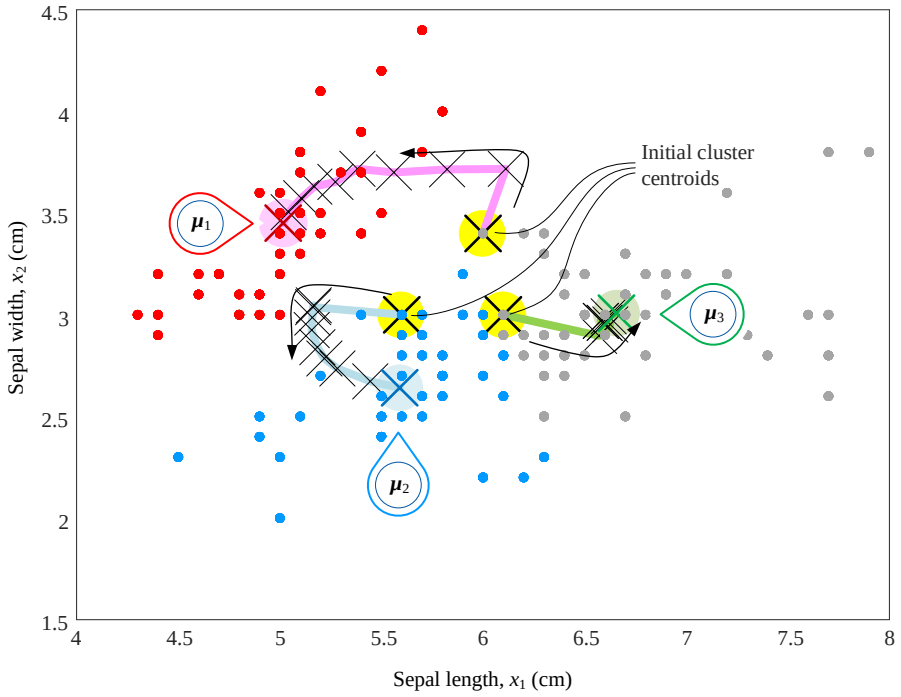


图 4. *K* 均值算法迭代过程，鸢尾花数据为例

鸢尾花数据聚类

图 5 所示为 *K* 均值算法聚类鸢尾花数据。Scikit-learn 工具包用来 *K* 均值聚类算法函数为 `sklearn.cluster.KMeans()`。利用 `model.fit()` 拟合数据后，`model.predict()` 预测聚类标签，`model.cluster_centers_`，输出簇质心位置。

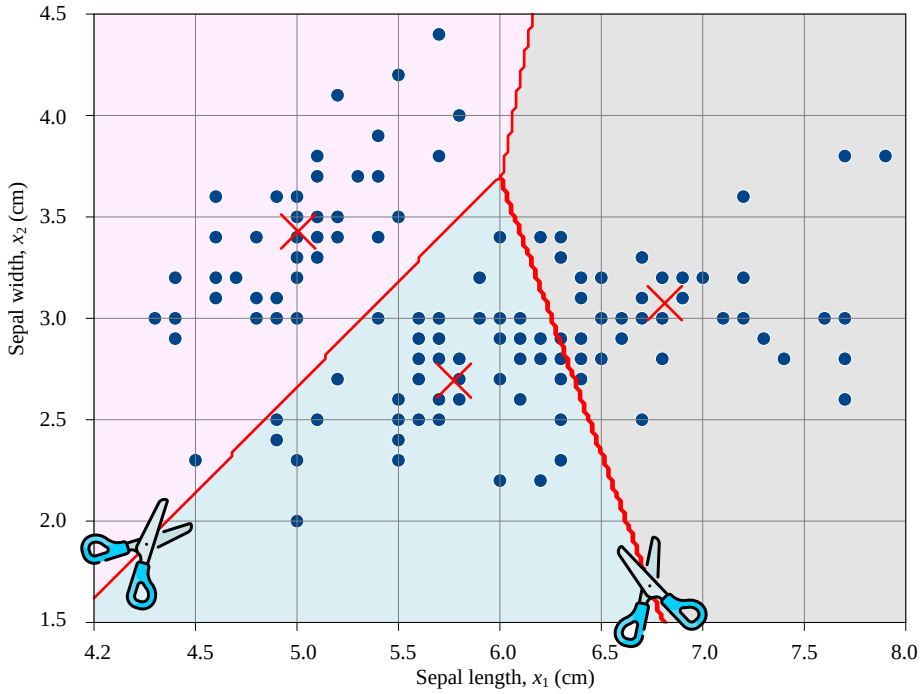
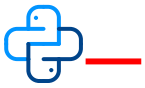


图 5. K 均值算法聚类鸢尾花数据



代码 Bk7_Ch11_01.py 绘制图 5。

12.4 肘部法则：选定聚类簇值

手肘法则 (elbow method) 可以用来判断合适的聚类簇值 K。手肘法则的关键指标是误差平方和 SSE：

$$\mathrm{SSE}\left(\boldsymbol{X}\,|K\right)=\sum_{k=1}^{K}\mathrm{SSE}\left(C_k\right)=\sum_{k=1}^{K}\sum_{\boldsymbol{x}\in C_k}\left\|\boldsymbol{x}-\boldsymbol{\mu}_k\right\|^2 \tag{7}$$

随着聚类簇数 K 不断增大，均值聚类算法对样本数据划分会逐渐变得更加精细；因此，随着 K 不断增大，每个簇的聚合程度会逐渐提高，SSE 会逐渐变小。

极端情况，当 $K = n$，也就是每个样本数据自成一簇，SSE = 0。

K 不断增大，SSE 不断减小的过程如图 6 所示。观察此图发现一个有意思的现象，当 K 小于“合适”聚类数时，K 增大时，会导致 SSE 大幅下降；但是，K 大于“合适”聚类数时，K 再增大，SSE 下降幅度不断变缓。

这就是为什么图 6 呈现出“手肘”形状，也便是手肘法则的名称来由。理想的聚类簇数 K 便是“肘”拐点的位置。K 均值聚类算法函数输出值 `model.inertia_` 便是当前 SSE 值。

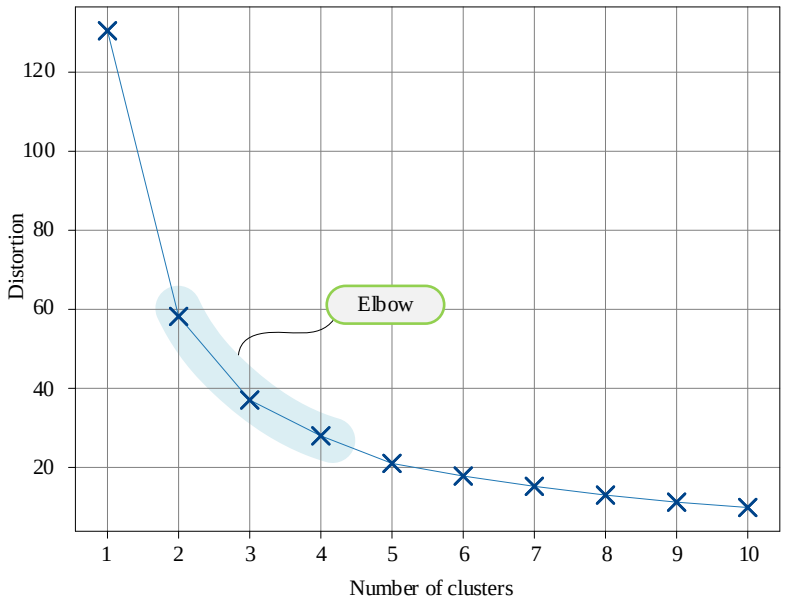


图 6. K 均值算法聚类鸢尾花数据

代码 Bk7_Ch11_02.py 绘制图 6。

12.5 轮廓图：选定聚类簇值

轮廓图 (silhouette plot) 也常用来选定聚类簇值 K。

轮廓图上每一条线代表的是**轮廓系数** (silhouette coefficient)，s_i，可以通过下式计算获得：

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}} \tag{8}$$

其中，a_i 为簇内不相似度，b_i 为簇间不相似度。

簇内不相似度

如图 7 (a) 所示，簇内不相似度 a_i 代表样本 i ($i \in C_k$) 到同簇其他样本 j ($j \in C_k$, $i \neq j$) 距离平均值：

$$a_i = \frac{1}{\text{count}(C_k) - 1} \sum_{j \in C_k, i \neq j} d_{i,j} \tag{9}$$

其中，$d_{i,j}$ 为样本 i 和 j 之间距离。a_i 越小，说明样本 i 越应该被划分到 C_k 簇。

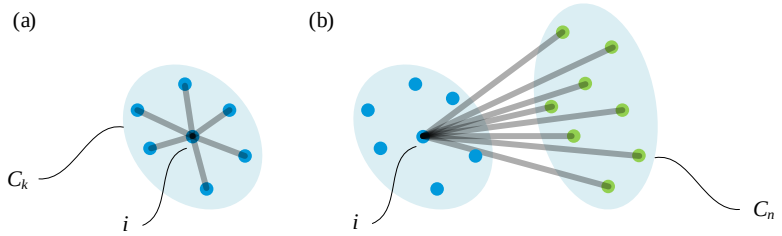


图 7. a_i 为簇内不相似度，和 b_i 为簇间不相似度

簇间不相似度

如图 7 (b) 所示，簇间不相似度 b_i 代表样本 i ($i \in C_k$) 到其他簇 (C_m) 样本 j ($j \in C_m$, $C_m \neq C_k$) 距离平均值的最小值：

$$b_i = \min \frac{1}{\text{count}(C_m)} \sum_{j \in C_m} d_{i,j} \tag{10}$$

b_i越大，说明样本 i 越不应该被划分到其他簇。

注意，当簇数超过 2 时，b_i需要在不同簇之间取最小值。

以鸢尾花数据为例

轮廓系数 s_i的取值在 [−1, 1] 区间。s_i越趋向于 1，说明样本 i 分类越正确；s_i越趋向于−1，说明样本 i 分类越错误。当 s_i在 0 附近时，样本 i 靠近聚类边界。

图 8、图 9 和图 10 所示为 K 分别取 3、4 和 5 时，聚类边界和轮廓图。理想的聚类结果是，簇内尽量紧密，簇间尽量远离。轮廓系数平均值越高，说明分类越合理。比较图 8、图 9 和图 10，K = 3 时，轮廓系数较高，并且轮廓图簇宽度均匀。轮廓图结合肘部法则判断聚类簇数更合适。

计算轮廓系数的函数为 `sklearn.metrics.silhouette_score`。

`yellowbrick.cluster.SilhouetteVisualizer` 函数绘制轮廓图。

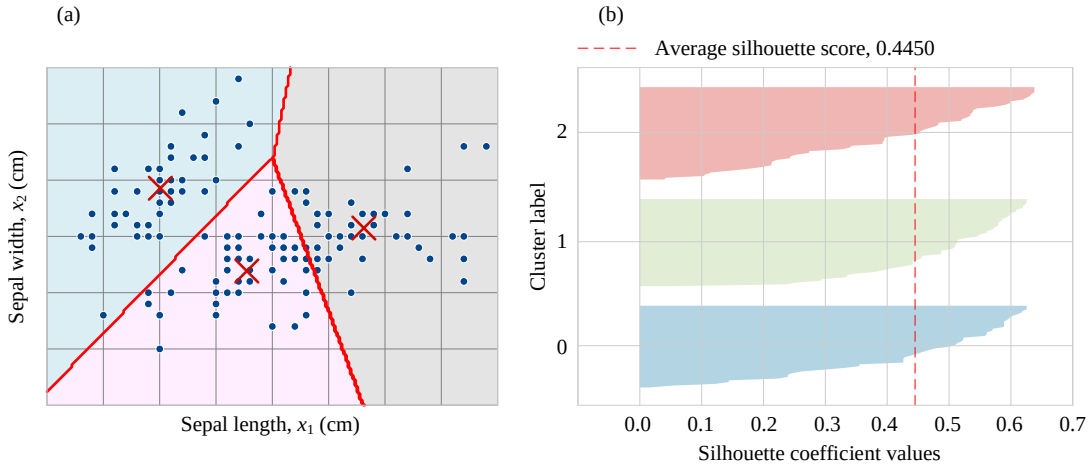


图 8. K 均值算法聚类鸢尾花数据和轮廓图，K = 3

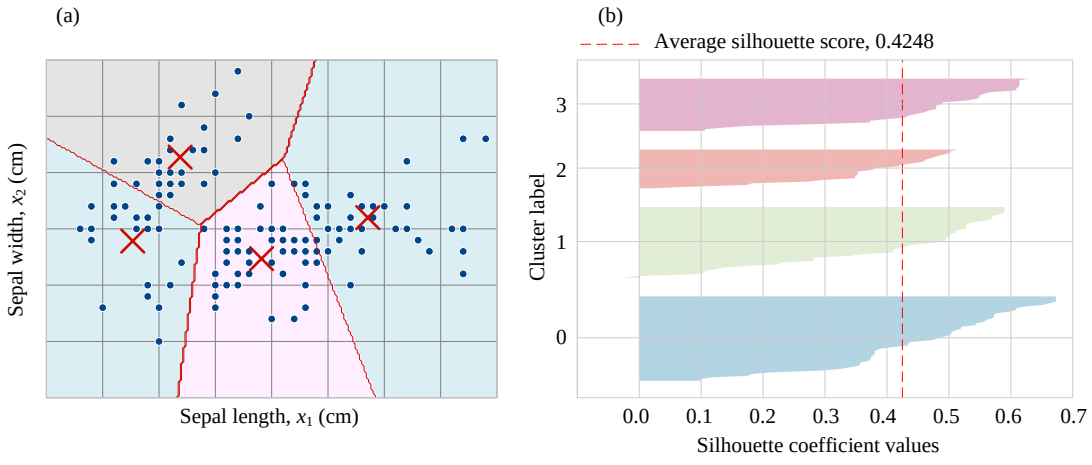


图 9. K 均值算法聚类鸢尾花数据和轮廓图，K = 4

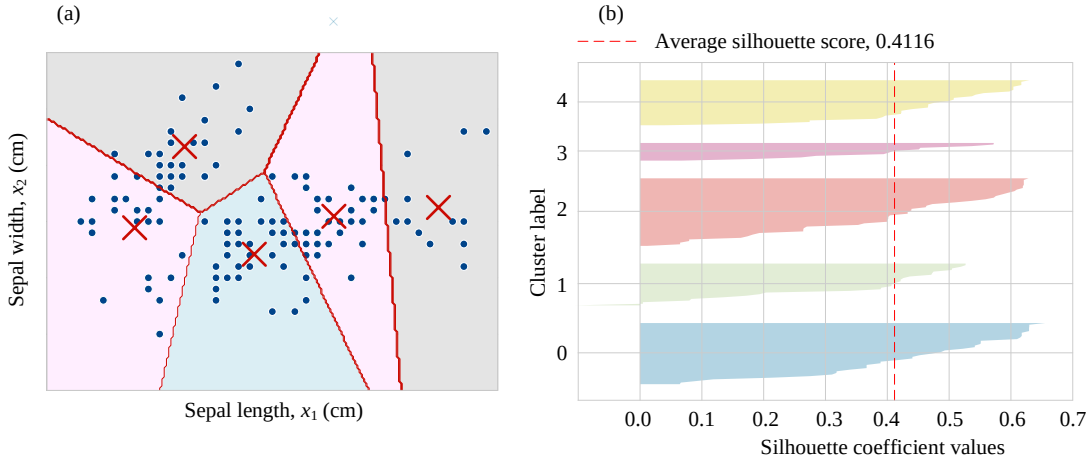


图 10. K 均值算法聚类鸢尾花数据和轮廓图，$K = 5$

代码 Bk7_Ch11_03.py 绘制图 8、图 9 和图 10。

12.6 沃罗诺伊图

沃罗诺伊图 (Voronoi diagram)，是由俄国数学家格**奥尔吉·沃罗诺伊** (Georgy Voronoy) 发明的空间分割算法。本章介绍的 K 均值聚类，本书前文介绍的**最近质心分类器** (Nearest Centroid Classifier)，实际上都依赖沃罗诺伊图确定决策边界。

图 11 所示为平面 4 点构造的沃罗诺伊图。距离较近的两点连线，绘制中垂线；若干中垂线便是分割平面区域的边界线。

`scipy.spatial.Voronoi` 函数获得沃罗诺伊图相关数据。`scipy.spatial.voronoi_plot_2d` 函数绘制沃罗诺伊图。图 12 所示为随机生成平面 30 个点，以及它们构造的沃罗诺伊图。

K 均值聚类，相当于在利用圆圈 (欧氏距离) 描述每个簇质心；而实际上，描述簇数据更好的形状可能是正椭圆，甚至旋转椭圆。这就是下一章高斯混合模型 GMM 需要解决的问题。

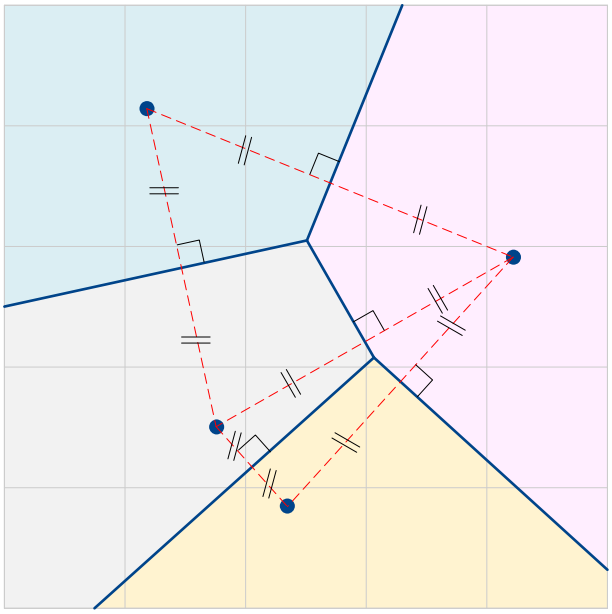


图 11. 4 点平面沃罗诺伊图

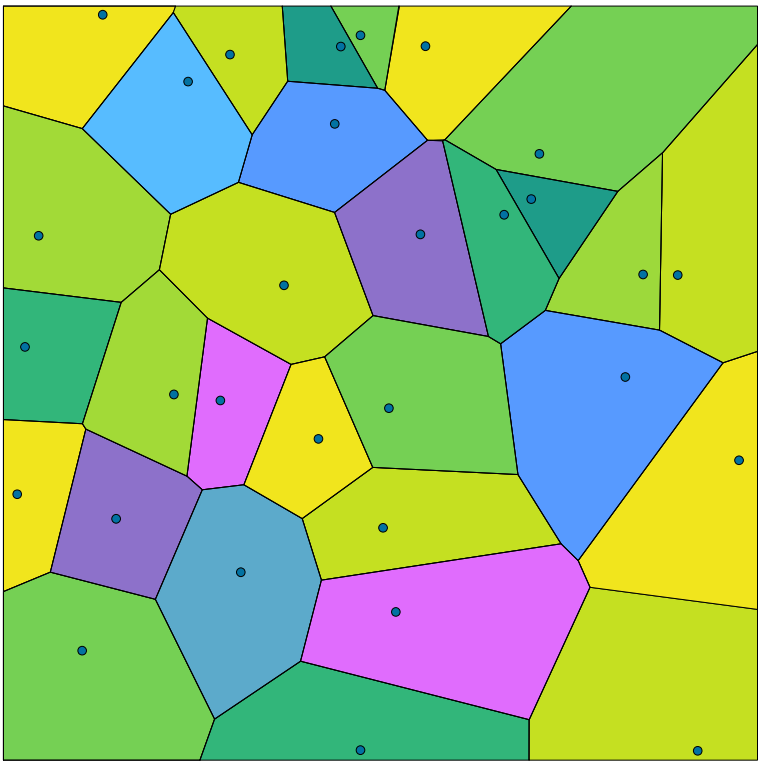
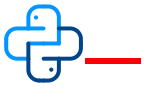
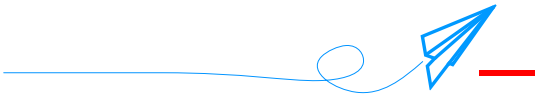


图 12. 30 点平面沃罗诺伊图



代码 Bk7_Ch11_04.py 绘制图 12。



K 均值聚类是一种无监督的机器学习技术，用于将数据集分为 *K* 个不同的簇。该算法首先需要随机初始化 *K* 个聚类中心，然后根据数据点和聚类中心的距离将数据点划分到最近的簇中。接着更新聚类中心，并重复以上步骤，直到聚类中心不再发生变化或达到预设的迭代次数。

该算法的优化问题是最小化数据点与其所属聚类中心之间的距离和，可以使用梯度下降等方法来求解。肘部法则是一种确定最佳 *K* 值的方法，它基于聚类中心数量 *K* 与聚类误差平方和之间的关系。当 *K* 值增大时，SSE 逐渐减小，但减小速度会逐渐变慢，当 *K* 达到某个值时，SSE 的下降速度会急剧减缓，这个 *K* 值对应的点就是肘部。轮廓图是一种衡量聚类结果质量的方法，它基于数据点与其所属簇的紧密度和分离度之间的平衡。

注意，*K*-means 聚类结果的簇质心并不是从样本数据点挑选出来的；如果从样本数据点所在位置挑选合适的位置作为簇质心的话，这种方法叫做 *k* 中心聚类 (*k*-medoids clustering)。请大家参考下例，这个例子还使用不同距离度量。

https://scikit-learn-extra.readthedocs.io/en/latest/auto_examples/cluster/plot_kmedoids_digits.html

13 Gaussian Mixture Model
高斯混合模型

组合若干高斯分布，期望最大化

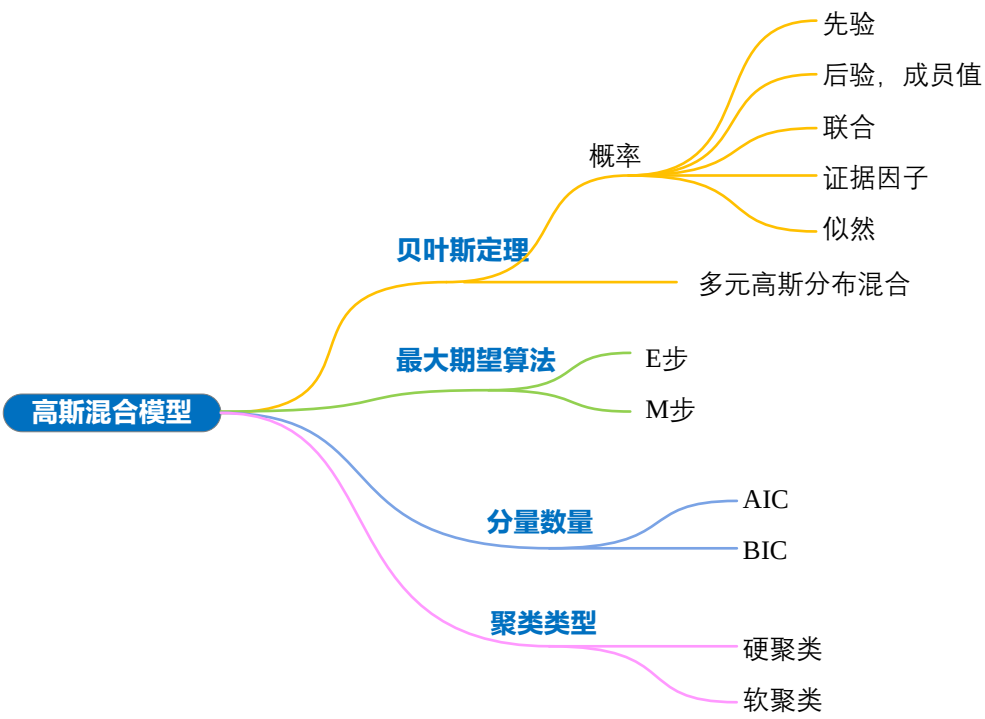
每当竭力厘清某一数学话题后，我便径直离开，投身另一处昏暗角落；

孜孜以求的人如此奇怪，求解一个问题后，他不会自我陶醉、故步自封，而是踏上新的旅程。

When I have clarified and exhausted a subject, then I turn away from it, in order to go into darkness again; the never satisfied man is so strange if he has completed a structure, then it is not in order to dwell in it peacefully, but in order to begin another.

—— 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

- `matplotlib.patches.Ellipse()` 绘制椭圆
- `numpy.arctan2()` 输入正切值分子分母两个数，输出为反正切，值域为 `[-pi, pi]`
- `numpy.linalg.eigh ()` 返回实对称矩阵的特征值和特征向量
- `numpy.linalg.norm()` 默认 L2 范数
- `numpy.linalg.svd()` SVD 分解函数
- `plt.quiver()` 绘制箭头图
- `seaborn.barplot()` 绘制直方图
- `sklearn.mixture.GaussianMixture` 高斯混合模型聚类函数



13.1 高斯混合模型

高斯混合模型 (Gaussian Mixture Model, GMM) 是一种常用的无监督机器学习算法，它的核心思维是——用多个高斯密度函数估计样本数据分布。高斯混合模型是一种概率模型，它假定所有数据点都是由有限个参数未知的高斯分布混合产生。

某种意义上讲，高斯混合模型是 K 均值聚类的推广。高斯混合模型和 K 均值聚类都是采用迭代方法求解优化问题。K 均值利用簇质心，最小化簇内残差平方和 SSE；而高斯混合模型利用簇质心和协方差，最大化对数似然函数。

此外，高斯混合模型和本书监督学习部分讲解的贝叶斯分类和高斯判别分析联系紧密。

前文说过，高斯混合模型是若干个高斯分布混合；下面分别一元和二元高斯分布来介绍这以思想。

一元高斯分布混合

大家对一元高斯分布概率密度函数 $f_X(x)$ 再熟悉不过：

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{1}$$

其中，μ 为均值/期望值，σ 为标准差。对于一元高斯分布，给定 μ 和 σ 就确定分布形状。

对于单一特征样本数据，高斯混合模型的意义就是利用若干一元高斯分布来描述样本分布。

图 1 给出的是鸢尾花样本数据花萼长度和花萼宽度两个特征。分别观察这两个特征，可以发现用单一高斯分布都不能准确描述数据的边际分布，但是组合三个高斯分布却可以描述数据特征分布。

注意，对于无监督学习，样本数据没有标签，即并不知道样本数据的类别。高斯混合模型算法通过一系列运算估计预测样本数据类别。

通常，称高斯混合模型 GMM 每一高斯分布为一个**分量** (component)。对于一元高斯分布，高斯混合分布算法难点就是确定每个分量各自的参数，μ 和 σ。

本书前文在贝叶斯分类部分介绍过，根据全概率定理，$C_1, C_2, \ldots, C_K$ 为一组不相容分类，对样本空间 Ω 形成分割，下式成立：

$$f_X(x) = \sum_{k=1}^{K} \underbrace{f_{Y,X}(C_k, x)}_{\text{Joint}} \tag{2}$$

根据贝叶斯定理，证据因子、后验概率和联合概率存在如下关系：

$$\underbrace{f_{Y,X}(C_k, X)}_{\text{Joint}} = \underbrace{f_{X|Y}(x|C_k)}_{\text{Likelihood}} \underbrace{p_Y(C_k)}_{\text{Prior}} \tag{3}$$

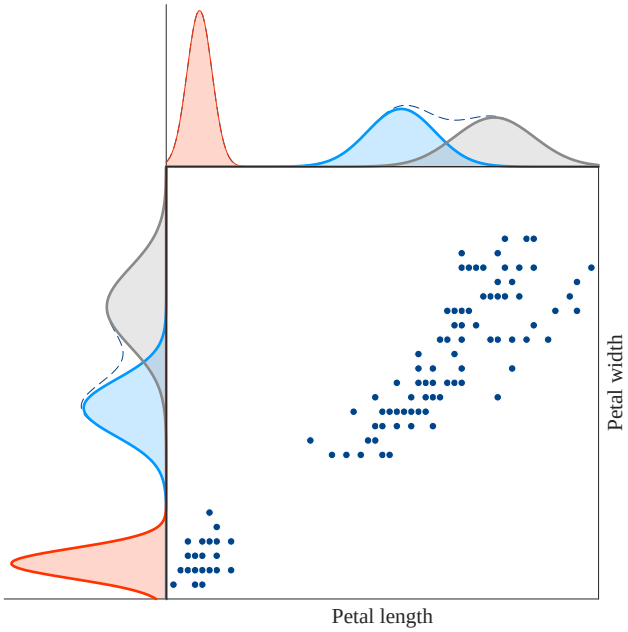


图 1. 用三个高斯一元分布描述样本数据边际分布

将 (3) 代入 (2) 得到：

$$f_X(x)=\sum_{k=1}^{K}\underbrace{f_{X|Y}\left(x|C_k\right)}_{\text{Likelihood}}\underbrace{p_Y\left(C_k\right)}_{\text{Prior}} \tag{4}$$

对于高斯混合模型 GMM，$f_{X|Y}\left(x|C_k\right)$ 为**似然概率** (likelihood) 用高斯分布描述，$p_Y\left(C_k\right)$ 为**先验概率** (prior)，表达样本集合中 C_k 类样本占比。

对于无监督学习，样本数据标签未知；因此，高斯混合模型 GMM 迭代过程中，似然概率 $f_{X|Y}\left(x|C_k\right)$ 和先验概率 $p_Y\left(C_k\right)$ 不断估算更新，直到满足迭代停止条件。

而对于有监督学习，样本标签数据已知，即 C_k 确定；比如，高斯朴素贝叶斯算法，直接就可以估算似然概率 $f_{X|Y}\left(x|C_k\right)$ 和先验概率 $p_Y\left(C_k\right)$。

对于单一特征样本数据，且 $K=3$，图 2 对应的边际分布 $p_Y\left(C_k\right)$ 可以用三个一元高斯分布叠加获得：

$$\begin{aligned}
f_X(x) &= \underbrace{p_Y\left(C_1\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x|C_1\right)}_{\text{Likelihood}}+\underbrace{p_Y\left(C_2\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x|C_2\right)}_{\text{Likelihood}}+\underbrace{p_Y\left(C_3\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x|C_3\right)}_{\text{Likelihood}} \\
&= \alpha_1 N\left(x,\mu_1,\sigma_1\right)+\alpha_2 N\left(x,\mu_2,\sigma_2\right)+\alpha_3 N\left(x,\mu_3,\sigma_3\right) \\
&= \underbrace{\alpha_1\frac{\exp\left(-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2\right)}{\sigma_1\sqrt{2\pi}}}_{C_1}+\underbrace{\alpha_2\frac{\exp\left(-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2\right)}{\sigma_2\sqrt{2\pi}}}_{C_2}+\underbrace{\alpha_3\frac{\exp\left(-\frac{1}{2}\left(\frac{x-\mu_3}{\sigma_3}\right)^2\right)}{\sigma_3\sqrt{2\pi}}}_{C_3}
\end{aligned} \tag{5}$$

如图 2 所示，μ_1、μ_2 和 μ_3 为期望值，描述三个正态分布质心位置；σ_1、σ_2 和 σ_3 为标准差，刻画三个正态分布的离散程度；而先验概率 α_1、α_2 和 α_3 给出三个正态分布对 $f_X(x)$ 的贡献。

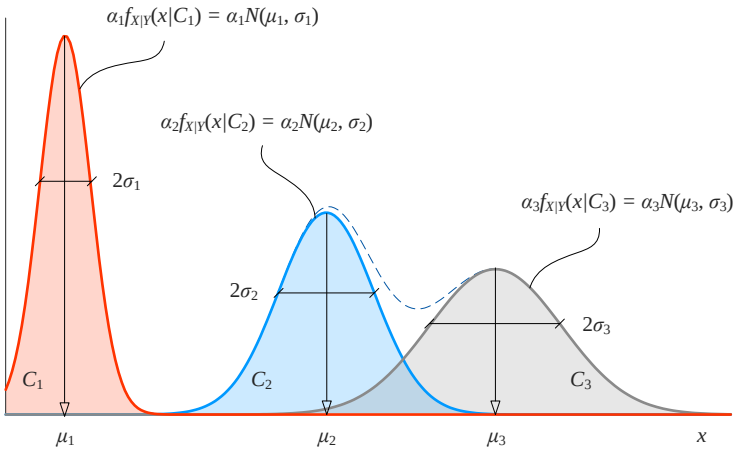


图 2. 三个一元高斯分布重要统计描述量

令：

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \mu_1 & \mu_2 & \mu_3 & \sigma_1 & \sigma_2 & \sigma_3 \end{bmatrix} \tag{6}$$

三个一元高斯分布叠加产生的高斯混合分布记做 $f_X(x \mid \boldsymbol{\theta})$。

$$f(x \mid \boldsymbol{\theta}) = p_Y(C_1) f_{X|Y}(x|C_1, \boldsymbol{\theta}) + p_Y(C_2) f_{X|Y}(x|C_2, \boldsymbol{\theta}) + p_Y(C_3) f_{X|Y}(x|C_3, \boldsymbol{\theta}) \tag{7}$$

多元高斯分布混合

下面考虑样本数据多特征情况。C_k 类数据条件概率 $f_{X|Y}(x|C_k)$ 分布服从多元高斯分布：

$$f_{X|Y}(x|C_k) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_k)\right)}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} \tag{8}$$

其中，D 为特征数量，即多元高斯分布维数；$\boldsymbol{x}$ 为列向量，$\boldsymbol{\mu}_k$ 为 C_k 类簇质心位置，即期望值/均值；$\boldsymbol{\Sigma}_k$ 为 C_k 类数据协方差矩阵，刻画正态分布离散程度和相关性。

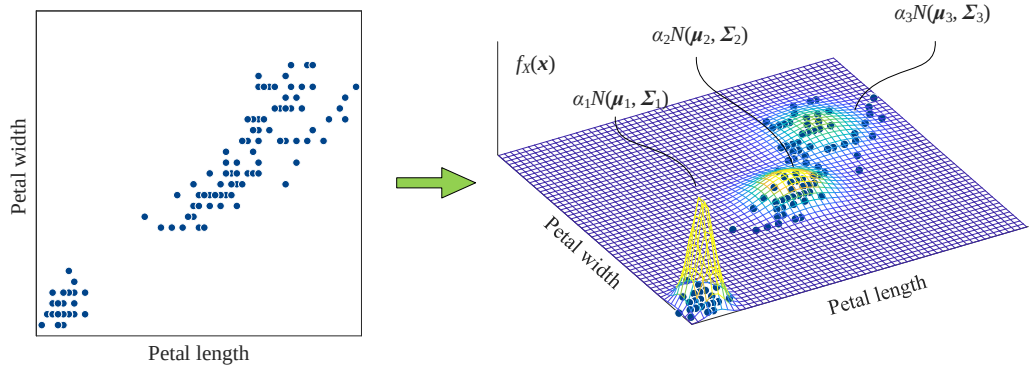


图 3. 三个二元高斯分布叠加描述鸢尾花数据分布

图 3 展示的鸢尾花花萼长度和宽度样本数据分布。显然，样本数据不适合用一个二元高斯分布，也不能用两个二元高斯分布叠加得。但是，每个高斯分布描述一簇数据，采用三个高斯分布叠加就可以比较准确的描述数据分布情况：

$$\begin{aligned} f(\boldsymbol{x}\,|\,\boldsymbol{\theta}) &= p_Y(C_1)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_1,\boldsymbol{\theta}) + p_Y(C_2)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_2,\boldsymbol{\theta}) + p_Y(C_3)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_3,\boldsymbol{\theta}) \\ &= \alpha_1 \underbrace{N(\boldsymbol{x},\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}_{C_1} + \alpha_2 \underbrace{N(\boldsymbol{x},\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)}_{C_2} + \alpha_3 \underbrace{N(\boldsymbol{x},\boldsymbol{\mu}_3,\boldsymbol{\Sigma}_3)}_{C_3} \end{aligned} \tag{9}$$

定义参数 $\boldsymbol{\theta}$ 为：

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \boldsymbol{\mu}_1 & \boldsymbol{\mu}_2 & \boldsymbol{\mu}_3 & \boldsymbol{\Sigma}_1 & \boldsymbol{\Sigma}_2 & \boldsymbol{\Sigma}_3 \end{bmatrix} \tag{10}$$

再次强调，作为无监督学习，样本数据标签未知；高斯混合模型 GMM 通过迭代求解优化问题，迭代过程，参数 $\boldsymbol{\theta}$ 不断更新。当迭代收敛时，参数 $\boldsymbol{\theta}$ 更新变化平缓。因此，(10) 定义的 $\boldsymbol{\theta}$，实际上是某一轮迭代时参数估计的快照。

后验概率

根据贝叶斯定理，计算后验概率：

$$f_{Y|\boldsymbol{X}}(C_k\,|\,\boldsymbol{x},\boldsymbol{\theta}) = \frac{p_Y(C_k)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_k,\boldsymbol{\theta})}{f_{\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{\theta})} = \frac{p_Y(C_k)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_k,\boldsymbol{\theta})}{\sum_{k=1}^{K} p_Y(C_k)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_k,\boldsymbol{\theta})} \tag{11}$$

由 K 个高斯分布构造的混合分布函数如下所示：

$$\begin{aligned} f_{\boldsymbol{X}}(\boldsymbol{x},\boldsymbol{\theta}) &= \sum_{k=1}^{K} p_Y(C_k)\, f_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_k,\boldsymbol{\theta}) \\ &= \sum_{k=1}^{K} \alpha_k N(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \end{aligned} \tag{12}$$

其中，第 i 个高斯分布参数有两个，分别是均值向量 $\boldsymbol{\mu}_k$ 和协方差矩阵 $\boldsymbol{\Sigma}_k$。α_k 为混合系数，是混合成分的后验概率，$\alpha_i > 0$。

参数 $\boldsymbol{\theta}$ 定义为：

$$\boldsymbol{\theta} = \left\{ \alpha_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k \right\} \quad k = 1, 2, ..., K \tag{13}$$

K 个混合系数之和为 1：

$$\sum_{k=1}^{K} \alpha_k = 1 \tag{14}$$

高斯混合模型，分量数量 K 是一个用户输入值。本章后续会介绍如何选取合适分量数量 K。

三聚类

假设数据聚类为 C_1、C_2 和 C_3 三类，后验概率 $f_{\chi|Y}\left(\boldsymbol{x}|C_1,\boldsymbol{\theta}\right)$、$f_{\chi|Y}\left(\boldsymbol{x}|C_2,\boldsymbol{\theta}\right)$ 和 $f_{\chi|Y}\left(\boldsymbol{x}|C_3,\boldsymbol{\theta}\right)$ 可以通过下式获得：

$$\begin{cases} f_{Y|\chi}\left(C_1|\boldsymbol{x},\boldsymbol{\theta}\right) = \dfrac{p_Y\left(C_1\right) f_{\chi|Y}\left(\boldsymbol{x}|C_1,\boldsymbol{\theta}\right)}{f_{\chi}\left(\boldsymbol{x},\boldsymbol{\theta}\right)} \\ f_{Y|\chi}\left(C_2|\boldsymbol{x},\boldsymbol{\theta}\right) = \dfrac{p_Y\left(C_2\right) f_{\chi|Y}\left(\boldsymbol{x}|C_2,\boldsymbol{\theta}\right)}{f_{\chi}\left(\boldsymbol{x},\boldsymbol{\theta}\right)} \\ f_{Y|\chi}\left(C_3|\boldsymbol{x},\boldsymbol{\theta}\right) = \dfrac{p_Y\left(C_3\right) f_{\chi|Y}\left(\boldsymbol{x}|C_3,\boldsymbol{\theta}\right)}{f_{\chi}\left(\boldsymbol{x},\boldsymbol{\theta}\right)} \end{cases} \tag{15}$$

其中，

$$f_{\chi}\left(\boldsymbol{x},\boldsymbol{\theta}\right) = p_Y\left(C_1\right) f_{\chi|Y}\left(\boldsymbol{x}|C_1,\boldsymbol{\theta}\right) + p_Y\left(C_2\right) f_{\chi|Y}\left(\boldsymbol{x}|C_2,\boldsymbol{\theta}\right) + p_Y\left(C_3\right) f_{\chi|Y}\left(\boldsymbol{x}|C_3,\boldsymbol{\theta}\right) \tag{16}$$

图 4 所示后验概率 $f_{Y|\chi}\left(C_1|\boldsymbol{x},\boldsymbol{\theta}\right)$、$f_{Y|\chi}\left(C_2|\boldsymbol{x},\boldsymbol{\theta}\right)$ 和 $f_{Y|\chi}\left(C_3|\boldsymbol{x},\boldsymbol{\theta}\right)$ 三曲面。比较这三个曲面高度便可以确定预测聚类区域。高斯混合模型迭代过程，$f_{Y|\chi}\left(C_1|\boldsymbol{x},\boldsymbol{\theta}\right)$、$f_{Y|\chi}\left(C_2|\boldsymbol{x},\boldsymbol{\theta}\right)$ 和 $f_{Y|\chi}\left(C_3|\boldsymbol{x},\boldsymbol{\theta}\right)$ 三曲面形状不断变化，决策边界也不断变化。

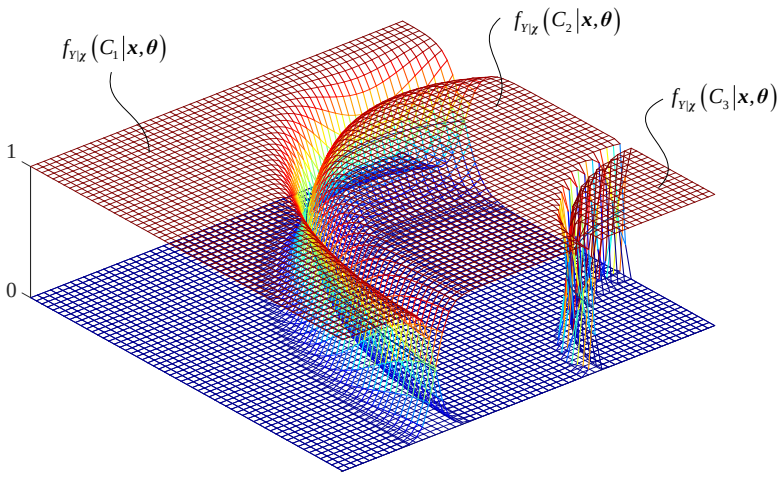


图 4. GMM 模型下后验概率曲面

给定无标记样本数据，可以采用高斯混合模型对数据进行聚类；类似贝叶斯分类，后验概率可以判定聚类决策边界。因此高斯混合模型聚类这个问题的优化目标，便是找到满足条件的参数 $\boldsymbol{\theta}$。下一节，我们将采用**最大期望算法** (expectation maximization, EM)，简称 **EM 算法**，解决这一问题。而下一章将专门讲解最大期望算法。

13.2 四类协方差矩阵

多元高斯分布用来刻画 C_k 类数据条件概率 $f_{X|Y}(\boldsymbol{x} \mid C_k)$；而多元高斯分布中，协方差矩阵 $\boldsymbol{\Sigma}_k$ 决定高斯分布的形状。本书前文在高斯判别分析 GDA 中介绍过六类 GDA，这六类 GDA 中协方差矩阵 $\boldsymbol{\Sigma}_k$ 各有特点。

如表 1 总结，scikit-learn 工具包中 `sklearn.mixture` 高斯混合模型支持四种协方差矩阵——`tied` （平移）、`spherical` （球面）、`diag` （对角）和 `full` (完全)。

`tied` 指的是，所有分量共享一个非对角协方差矩阵 $\boldsymbol{\Sigma}$；`tied` 类似第三类高斯判别分析。每个分量 PDF 等高线为大小相等旋转椭圆。根据本书前文分析，由于不同分量协方差相同，决策边界解析式二次项消去；因此 `tied` 对应的决策边界为直线。

`spherical` 指的是，每个分量协方差矩阵 $\boldsymbol{\Sigma}_k$ 不同，但是每个分量 $\boldsymbol{\Sigma}_k$ 均为对角阵；且 $\boldsymbol{\Sigma}_k$ 对角元素相同，即特征方差相同；`spherical` 类似第四类高斯判别分析。每个分量 PDF 等高线为正圆。`spherical` 对应的决策边界为圆形弧线。

`diag` 指每个分量有各自独立的对角协方差矩阵，也就是 $\boldsymbol{\Sigma}_k$ 为对角阵，特征条件独立；但是对 $\boldsymbol{\Sigma}_k$ 对角线元素大小不做限制。`diag` 对应第五类高斯判别分析。每个分量 PDF 等高线正椭圆，`diag` 对应的决策边界为正圆锥曲线。

`full` 指每个分量有各自独立协方差矩阵，即对 $\boldsymbol{\Sigma}_k$ 不做任何限制。`full` 对应第六类高斯判别分析。`full` 对应的决策边界为任意圆锥曲线。

表 1. 根据方差-协方差矩阵特点将高斯混合模型分为 4 类

参数设置	$\boldsymbol{\Sigma}_i$	$\boldsymbol{\Sigma}_i$ 特点	PDF 等高线	决策边界
tied (第三类)	相同	非对角阵	任意椭圆	直线
spherical (第四类)	不相同	对角阵，对角线元素等值	正圆	正圆
diag (第五类)		对角阵	正椭圆	正圆锥曲线
full (第六类)		非对角阵	任意椭圆	圆锥曲线

和 K 均值聚类算法一样，高斯混合模型 GMM 也需要指定 K 值；高斯混合模型也是利用迭代求解优化问题。不同的是，GMM 利用协方差矩阵，可以估算后验概率/成员值。GMM 的协方差矩阵有四种类型，每种类型对应不同假设，获得不同决策边界类型。

K 均值聚类可以看作是高斯混合模型一个特例。如图 5 所示，K 均值聚类对应的 GMM 特点是，各簇协方差矩阵 $\boldsymbol{\Sigma}_k$ 相同，$\boldsymbol{\Sigma}_k$ 为对角阵，并且 $\boldsymbol{\Sigma}_k$ 主对角线元素相等。

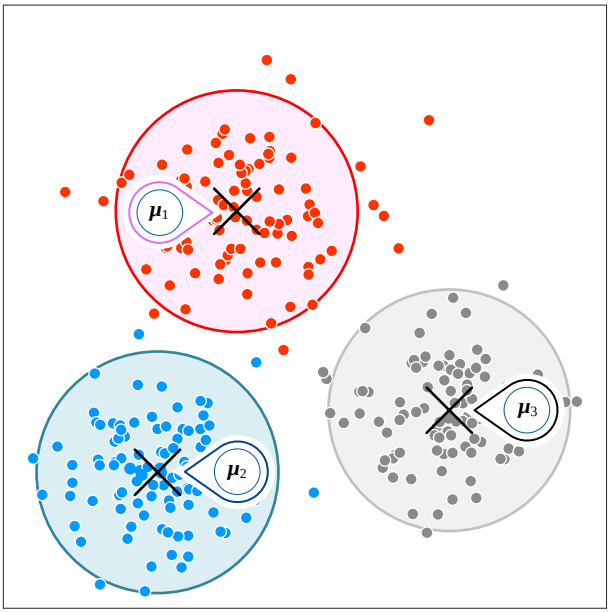


图 5. K 均值聚类可以看作是高斯混合模型一个特例

以鸢尾花数据为例

下面，我们分别利用 `sklearn.mixture` 四种协方差矩阵设置，比较鸢尾花数据的聚类结果。图 6 中，GMM 的协方差矩阵设置为 `tied`；容易发现获得的决策边界为直线，这是因为所有分量公用一个非对角协方差矩阵。图 7 中，GMM 的协方差矩阵设置为 `spherical`；对应的决策边界显然为三段圆弧构造。图 8 中，GMM 的协方差矩阵设置为 `diag`；图 8 中椭圆弧线长度较短，不容易直接判断它们对应的椭圆是否为正圆锥曲线。图 9 中，GMM 的协方差矩阵设置为 `full`；决策边界为任意圆锥曲线。读者可以回顾本书高斯判别分析中有关决策边界形态内容。

另外，图 6~图 9 这四幅图中，还给出高斯分布椭圆等高线的半长轴和半短轴向量指向。

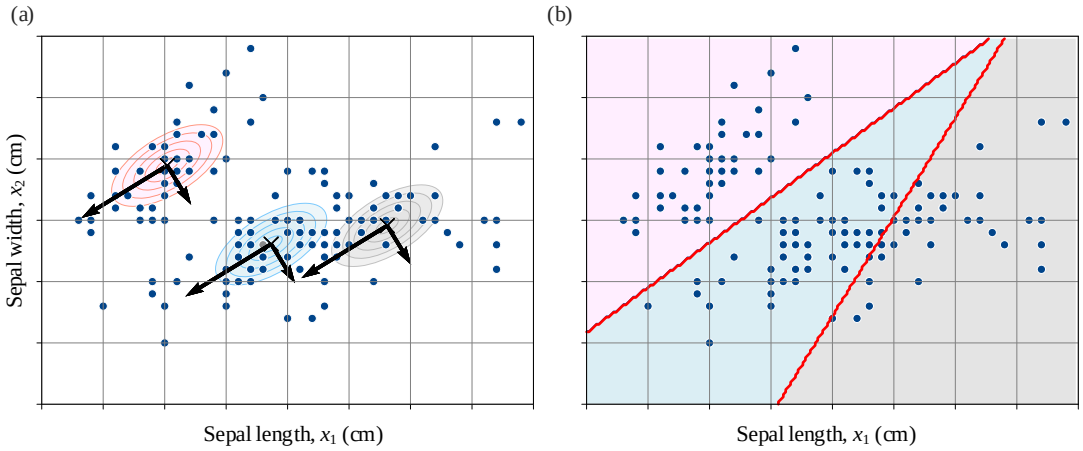


图 6. GMM 的协方差矩阵设置为 `tied`

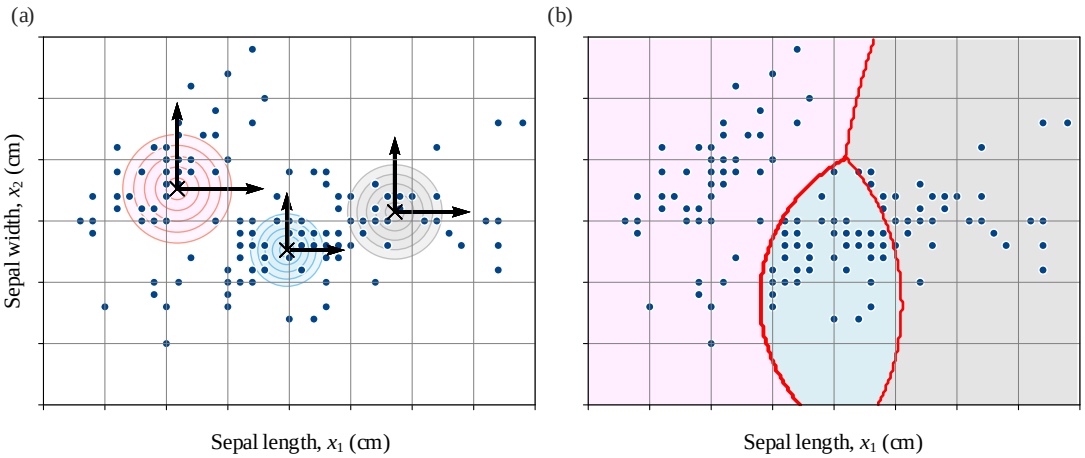


图 7. GMM 的协方差矩阵设置为 `spherical`

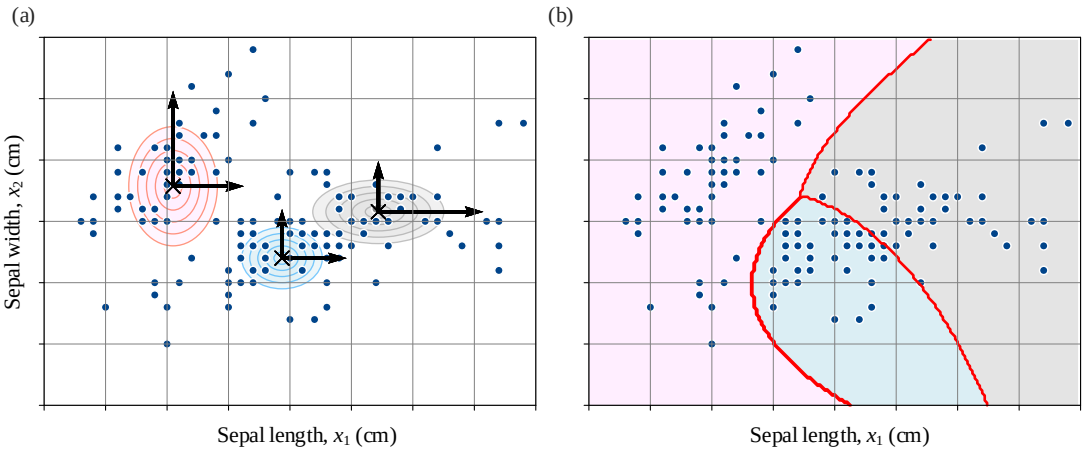


图 8. GMM 的协方差矩阵设置为 `diag`

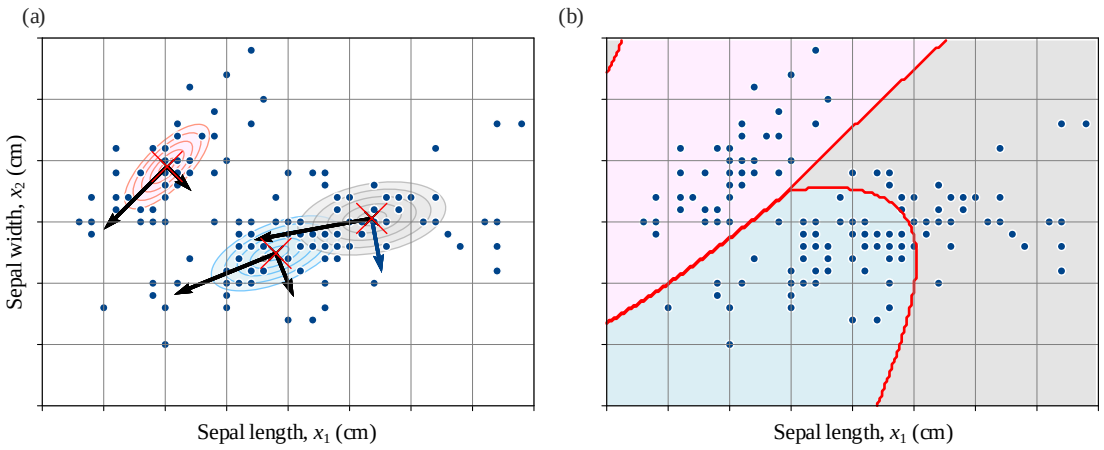
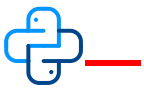


图 9. GMM 的协方差矩阵设置为 `full`



代码 Bk7_Ch12_01.py 完成本节分类问题。

13.3 分量数量

前文介绍，高斯混合模型 GMM 的分量数量 K 是用户输入值。选取合适 K 值，对于 GMM 聚类效果至关重要。本节介绍采用 AIC 和 BIC 选择高斯混合模型分量数量。

赤池信息量准则

丛书读者对 AIC 和 BIC 并不陌生，在《数据科学》一册回归相关内容，我们已经了解过这两个指标。AIC 为**赤池信息量准则** (Akaike information criterion, AIC)，定义如下：

$$\mathrm{AIC} = \underbrace{2K}_{\text{Penalty}} - 2\ln(L) \tag{17}$$

其中，K 是分量数量，即聚类数量；L 是似然函数。

Scikit-learn 工具包中 AIC 计算形式稍有不同。AIC 鼓励数据拟合的优良性；但是，尽量避免出现过度拟合。(17) 中 $2K$ 项为**惩罚项** (penalty)。

贝叶斯信息准则

贝叶斯信息准则 (Bayesian Information Criterion, BIC) 也称**施瓦茨信息准则** (Schwarz information criterion, SIC)，定义如下：

$$\mathrm{BIC} = \underbrace{K\ln(n)}_{\text{Penalty}} - 2\ln(L) \tag{18}$$

其中，n 为样本数据数量。BIC 的惩罚项比 AIC 大。

图 10 所示为三簇数据构成的样本数据。采用高斯混合模型聚类算法，K 取不同值 (K = 1, 2, …, 6)，协方差矩阵分别采用前文介绍的四种设置——`tied` （平移）、`spherical` （球面）、`diag` （对角）和 `full` (完全)。对于这 24 种组合，我们取出对应模型 AIC 和 BIC 结果。

图 11 所示为 AIC 随协方差形状和分量数变化直方图。图 12 所示为 BIC 随协方差形状和分量数变化直方图。可以发现，24 中设置中，`spherical` (球面) 和 K = 3 参数组合对图 10 所示样本数据聚类效果最好。

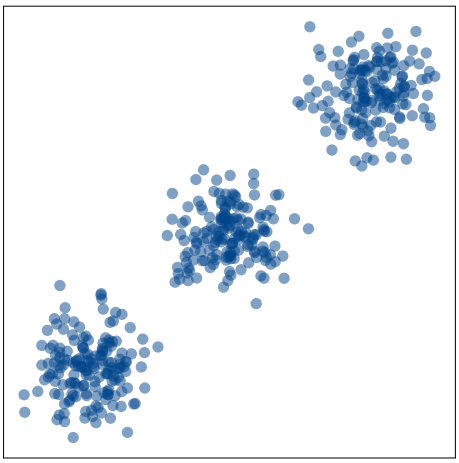


图 10. 三簇数据构成的样本数据

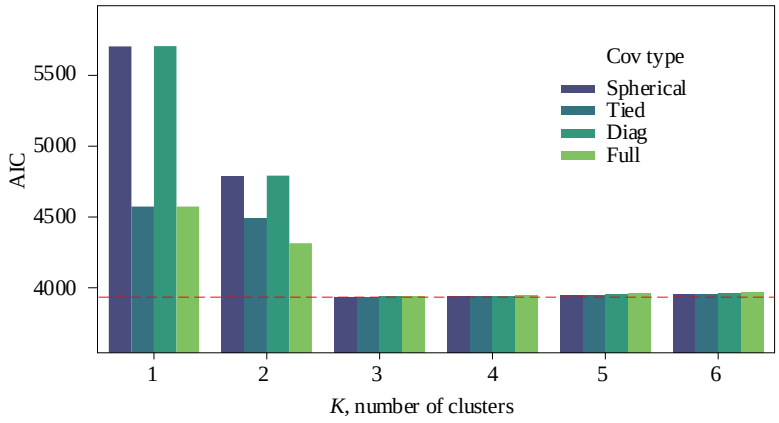


图 11. AIC 随协方差形状和分量数变化

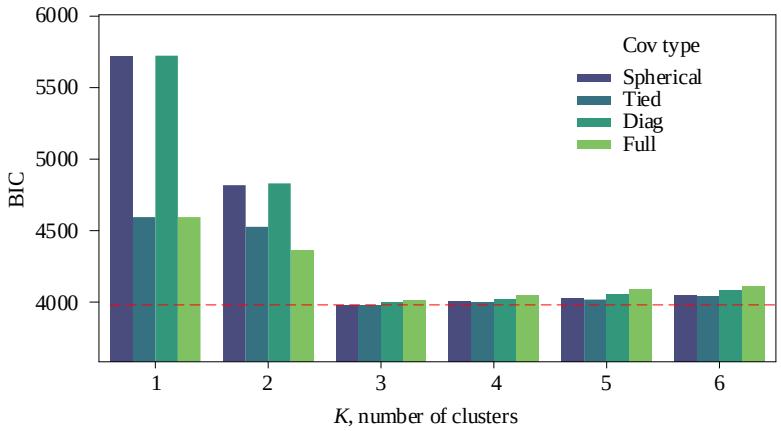


图 12. BIC 随协方差形状和分量数变化

代码 Bk7_Ch12_02.py 绘制本节图像。

13.4 硬聚类和软聚类

本书朴素贝叶斯分类算法中提到，后验概率相当于成员值。**硬聚类** (hard clustering) 指的根据成员值大小，决策边界清楚划定；但是**软聚类** (soft clustering) 则设定缓冲带，当后验概率/成员值在这个缓冲带内，样本数据没有明确的聚类。这样，软聚类的决策边界不再“泾渭分明”，而变成了一条宽带。

给定如图 13 所示 450 个样本数据。利用高斯混合模型算法获得 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 两后验概率曲面，如图 14 所示。

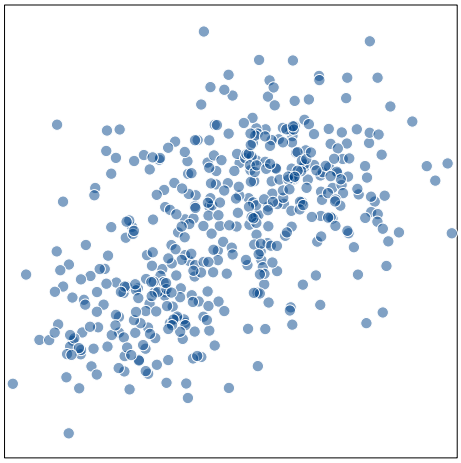


图 13. 样本数据

如图 14 所示，以成员值大小排列这 450 个样本数据；对于二聚类问题，硬聚类以后验概率 0.5 为分界线。当 $f_{Y|X}(C_1|\boldsymbol{x}) = 0.5$ 对应着决策边界；当 $f_{Y|X}(C_1|\boldsymbol{x}) > 0.5$，$\boldsymbol{x}$ 被聚类到 C_1 簇；当 $f_{Y|X}(C_1|\boldsymbol{x}) < 0.5$，$\boldsymbol{x}$ 被聚类到 C_2 簇。

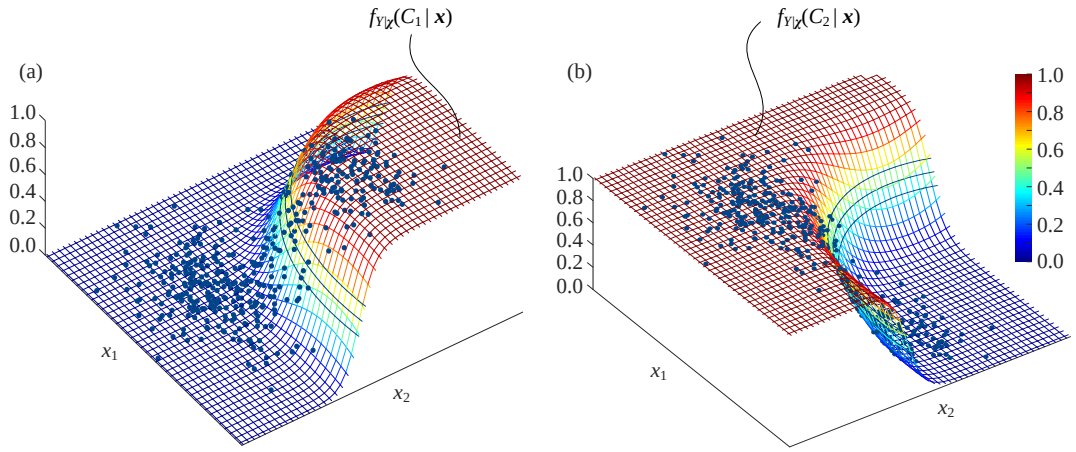


图 14. $f_{Y|X}(C_1|\boldsymbol{x})$ 和 $f_{Y|X}(C_2|\boldsymbol{x})$ 两后验概率曲面

软聚类

而对于软聚类，后验概率在一段阈值内，比如 [0.3, 0.7]，数据没有明确的分类。图 16 所示为聚类结果，加黑圈的样本数据，位于“决策带”之内，没有明确预测分类。

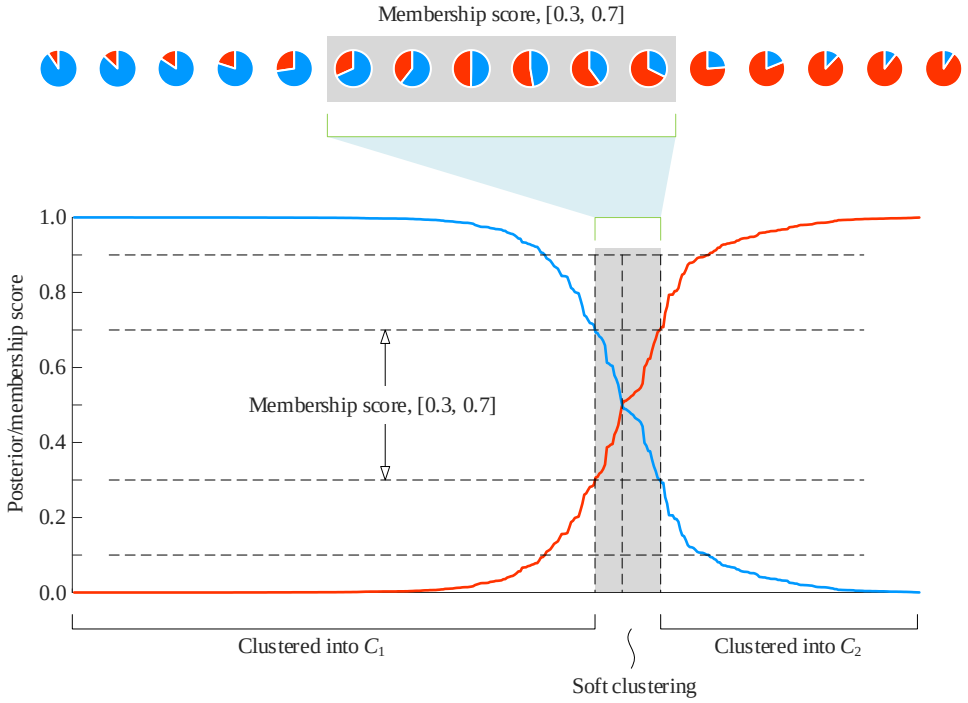


图 15. 成员值与软聚类

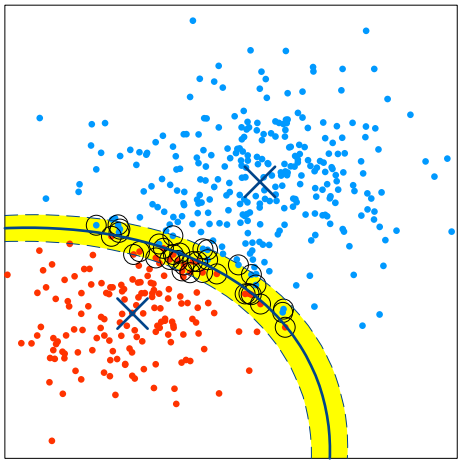


图 16. 软聚类分区和决策带

高斯混合模型 (Gaussian Mixture Model，GMM) 是一种概率模型，用于对多维数据进行建模和聚类。它将一个数据集看作由多个多元高斯分布的线性组合构成，每个多元高斯分布代表着一个簇，而簇的个数是由用户指定的。GMM 通过最大化似然函数来估计参数，其中参数包括每个高斯分布的均值、方差和系数 (即每个高斯分布在总分布中的占比)。在训练结束后，GMM 可以用于聚类、密度估计和生成新的数据点。与 k-means 算法相比，GMM 具有更强的建模能力和更大的灵活性，但其计算复杂度更高。

GMM 的参数估计通常使用最大期望算法 EM 完成，下一章专门介绍 EM。本章和下一章共用一个思维导图。

14 最大期望算法

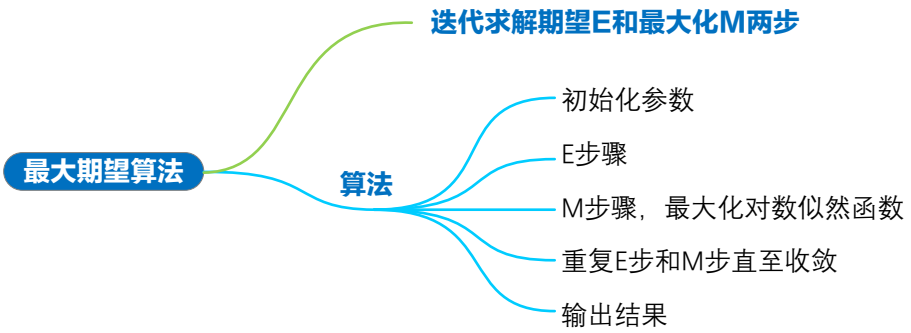
Expectation Maximization

迭代优化两步走：E 步，M 步；最大化对数似然函数

我解决的每个问题，都变成了定理法则；它们都被拿去解决更多的问题。

Each problem that I solved became a rule, which served afterwards to solve other problems.

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



14.1 最大期望

求解高斯混合模型 (Gaussian Mixture Model, GMM) 绕不开 **EM 算法**，即**最大期望算法** (Expectation Maximization, EM)。EM 算法是一种迭代算法，其核心思想是在不完全观测的情况下，通过已知的观测数据来估计模型参数。

上一章介绍的高斯混合模型核心思想是，叠加若干高斯分布来描述样本数据分布。一元高斯分布有两个重要参数，均值和均方差；而多元高斯分布则通过质心和协方差来描述。除此，我们还需要知道每个高斯分布分量的贡献，即先验概率值。遗憾的是，这几个参数不能通过解析方法求解。

本章介绍的最大期望算法正是求解高斯混合模型参数的方法。

E 步、M 步

EM 算法是一个收敛迭代过程。EM 算法两个步骤交替进行迭代：

- 第一步 (即所谓 E 步)，利用当前参数 $\boldsymbol{\theta}$ 计算期望值，并计算对数似然函数 $L(\boldsymbol{\theta})$；根据当前参数估计值计算每个数据点属于每个高斯分布的后验概率，即每个数据点在每个簇中的权重。
- 第二步 (即所谓 M 步)，在第一步基础上最大化，并更新参数 $\boldsymbol{\theta}$；根据上一步中计算得到的后验概率重新估计每个高斯分布的均值、方差和系数，并更新参数估计值。

EM 算法不断迭代这两个步骤，直到收敛为止。在 GMM 中，EM 算法的收敛条件可以是参数变化的阈值或者似然函数的收敛。

14.2 E 步：最大化期望

本节以单一特征样本数据为例，可视化最大期望算法迭代过程。观察发现数据应该被分为两簇，设定 $K = 2$。

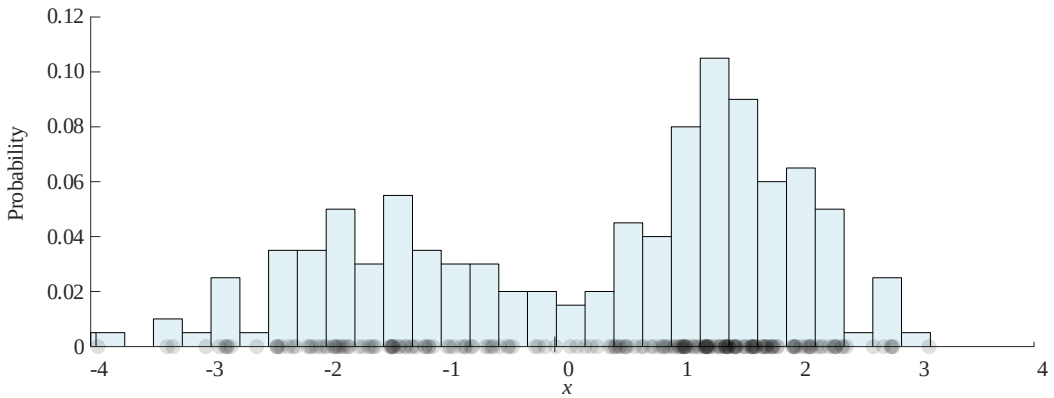


图 1. 一维样本待聚类样本数据

初始化

利用一元高斯分布叠加，首先初始化参数 $\boldsymbol{\theta}$：

$$\boldsymbol{\theta}^{(0)} = \left\{\alpha_1^{(0)}, \alpha_2^{(0)}, \mu_1^{(0)}, \mu_2^{(0)}, \sigma_1^{(0)}, \sigma_2^{(0)}\right\} \tag{1}$$

上角标 $^{(i)}$ 代表当前迭代次数，$^{(0)}$ 代表迭代初始。

选定初始化参数 $\boldsymbol{\theta}$ 具体数值如下：

$$\begin{cases} \alpha_1^{(0)} = p_Y\left(C_1, \boldsymbol{\theta}^{(0)}\right) = 0.5, \quad \alpha_2^{(0)} = p_Y\left(C_2, \boldsymbol{\theta}^{(0)}\right) = 0.5 \\ \mu_1^{(0)} = -0.05, \quad \mu_2^{(0)} = 0.05 \\ \sigma_1^{(0)} = \sigma_2^{(0)} = 1 \end{cases} \tag{2}$$

证据因子 α_1 和 α_2 给出两个不同高斯分布对 $f_X(x)$ 的贡献。

μ_1 和 μ_2 为期望值，描述两个正态分布质心位置。

σ_1 和 σ_2 为标准差，刻画正态分布离散程度。

似然概率

通过 (2) 给出六个参数，利用高斯分布估算得到 $f_{X|Y}(x \mid C_1, \boldsymbol{\theta}^{(0)})$ 和 $f_{X|Y}(x \mid C_2, \boldsymbol{\theta}^{(0)})$ 的两个似然概率 PDF，具体如下：

$$\begin{cases} f_{X|Y}\left(x \middle| C_1, \boldsymbol{\theta}^{(0)}\right) = \dfrac{\exp\left(-\dfrac{1}{2}\left(\dfrac{x-\mu_1}{\sigma_1}\right)^2\right)}{\sigma_1\sqrt{2\pi}} = \dfrac{\exp\left(-\dfrac{1}{2}(x+0.05)^2\right)}{\sqrt{2\pi}} \\ f_{X|Y}\left(x \middle| C_2, \boldsymbol{\theta}^{(0)}\right) = \dfrac{\exp\left(-\dfrac{1}{2}\left(\dfrac{x-\mu_2}{\sigma_2}\right)^2\right)}{\sigma_2\sqrt{2\pi}} = \dfrac{\exp\left(-\dfrac{1}{2}(x-0.05)^2\right)}{\sqrt{2\pi}} \end{cases} \tag{3}$$

图 2 所示为初始化参数对应的初始化参数对应的 $f_{X|Y}(x \mid C_1)$ 和 $f_{X|Y}(x \mid C_2)$ 图像。

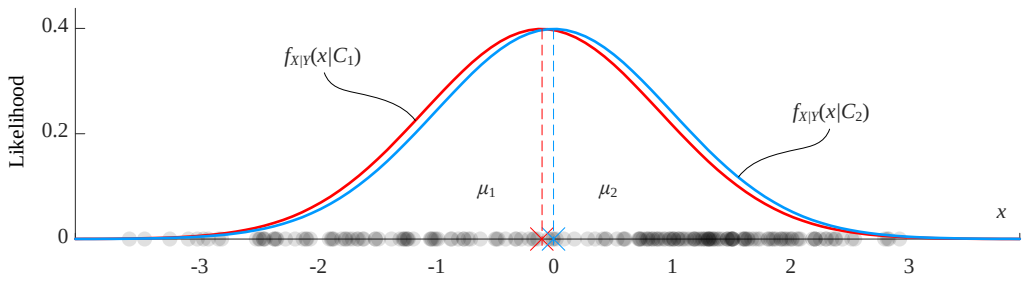


图 2. 初始化参数 $\boldsymbol{\theta}^{(0)}$ 对应的 $f_{X|Y}(x \mid C_1)$ 和 $f_{X|Y}(x \mid C_2)$ 图像

证据因子

下一步，估算概率密度函数 $f_X(x \mid \boldsymbol{\theta}^{(0)})$：

$$
\begin{aligned}
f_X\left(x\middle|\boldsymbol{\theta}^{(0)}\right) &= f_{X,Y}\left(x, C_1, \boldsymbol{\theta}^{(0)}\right) + f_{X,Y}\left(x, C_2, \boldsymbol{\theta}^{(0)}\right) \\
&= p_Y\left(C_1, \boldsymbol{\theta}^{(0)}\right) f_{X|Y}\left(x\middle|C_1, \boldsymbol{\theta}^{(0)}\right) + p_Y\left(C_2, \boldsymbol{\theta}^{(0)}\right) f_{X|Y}\left(x\middle|C_2, \boldsymbol{\theta}^{(0)}\right)
\end{aligned} \tag{4}
$$

将 (2) 和 (3) 代入 (4)，整理得到：

$$
f_X\left(x\middle|\boldsymbol{\theta}^{(0)}\right) = = \frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x+0.05)^2\right)}{\sqrt{2\pi}} + \frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x-0.05)^2\right)}{\sqrt{2\pi}} \tag{5}
$$

图 3 展示的是这一轮迭代 $f_{X,Y}(x, C_1)$、$f_{X,Y}(x, C_2)$ 和 $f_X(x)$ 结果图像。根据本书第 4 章有关朴素贝叶斯分类介绍的内容，图 3 所示 $f_{X,Y}(x, C_1)$、$f_{X,Y}(x, C_2)$ 曲线高度可以判断当前条件下数据聚类结果。图 3 中横轴数据点颜色代表本轮预测聚类结果。

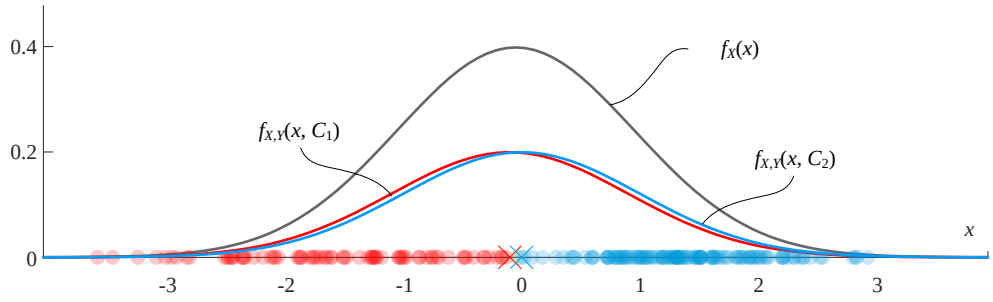


图 3. 初始化参数计算得到 $f_{X,Y}(x, C_1)$、$f_{X,Y}(x, C_2)$ 和 $f_X(x)$

后验概率

根据贝叶斯定理，计算后验概率 $f_{Y|X}(C_1 \mid x, \boldsymbol{\theta}^{(0)})$ 和 $f_{Y|X}(C_2 \mid x, \boldsymbol{\theta}^{(0)})$：

$$
\begin{cases}
f_{Y|X}\left(C_1\middle|x, \boldsymbol{\theta}^{(0)}\right) = \dfrac{p_Y\left(C_1, \boldsymbol{\theta}^{(0)}\right) f_{X|Y}\left(x\middle|C_1, \boldsymbol{\theta}^{(0)}\right)}{f_X\left(x\middle|\boldsymbol{\theta}^{(0)}\right)} = \dfrac{\frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x+0.05)^2\right)}{\sqrt{2\pi}}}{\frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x+0.05)^2\right)}{\sqrt{2\pi}} + \frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x-0.05)^2\right)}{\sqrt{2\pi}}} \\
f_{Y|X}\left(C_2\middle|x, \boldsymbol{\theta}^{(0)}\right) = \dfrac{p_Y\left(C_2, \boldsymbol{\theta}^{(0)}\right) f_{X|Y}\left(x\middle|C_2, \boldsymbol{\theta}^{(0)}\right)}{f_X\left(x\middle|\boldsymbol{\theta}^{(0)}\right)} = \dfrac{\frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x-0.05)^2\right)}{\sqrt{2\pi}}}{\frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x+0.05)^2\right)}{\sqrt{2\pi}} + \frac{1}{2} \times \frac{\exp\left(-\frac{1}{2}(x-0.05)^2\right)}{\sqrt{2\pi}}}
\end{cases} \tag{6}
$$

图 4 给出初始参数条件下后验概率 $f_{Y|X}(C_1 \mid x)$ 和 $f_{Y|X}(C_2 \mid x)$ 随 x 变化。对于任意一点 x，下式成立：

$$f_{Y|X}\left(C_1|x\right)+f_{Y|X}\left(C_2|x\right)=1 \tag{7}$$

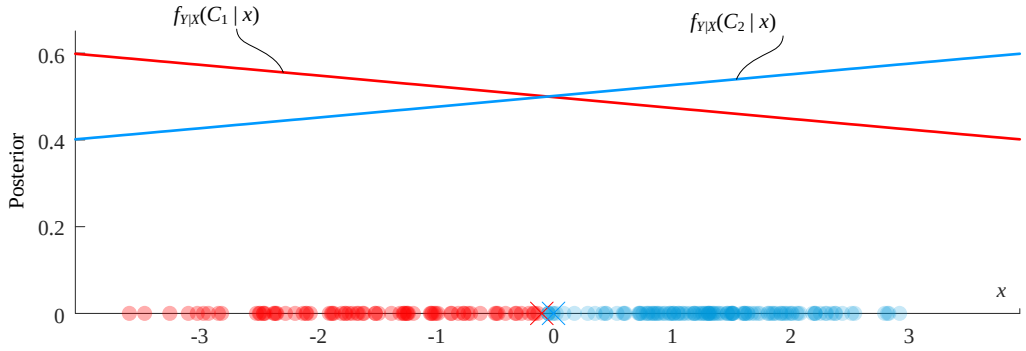


图 4. 初始化参数计算得到后验概率 $f_{Y|X}(C_1 \mid x)$ 和 $f_{Y|X}(C_2 \mid x)$

后验概率大小代表成员值，某一点不同簇后验值区分越大，分类才越有理有据。如果不同簇后验值区分不大，据此得到的分类预测则显得很牵强。因此，迭代优化还需要继续。

14.3 M 步：最大化似然概率

下一步是 EM 算法中非常重要的环节——更新参数、最大化似然概率。对于迭代 EM 算法，这便是 M 步。

先验概率

更新参数 α_1 和 α_2：

$$\begin{cases} \alpha_1^{(1)} = \dfrac{\sum_{i=1}^{n} f_{Y|X}\left(C_1|x^{(i)},\boldsymbol{\theta}^{(0)}\right)}{n} = 0.49379 \\ \alpha_2^{(1)} = \dfrac{\sum_{i=1}^{n} f_{Y|X}\left(C_2|x^{(i)},\boldsymbol{\theta}^{(0)}\right)}{n} = 0.50621 \end{cases} \tag{8}$$

α_1 和 α_2 相当于数据聚类比例。可以这样理解上式，一共有 n 个数据点，每个点有 $1/n$ 的投票权。对二聚类问题，$1/n$ 要分成两份，分别给 C_1 和 C_2。每个点的后验概率决定比例分配。

整理 (8) 可以得到如下等式：

$$
\begin{cases} n\alpha_1^{(1)} = \sum_{i=1}^{n} f_{Y|X}\left(C_1 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right) \\ n\alpha_2^{(1)} = \sum_{i=1}^{n} f_{Y|X}\left(C_2 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right) \end{cases} \tag{9}
$$

均值

利用当前每个样本数据估算得到的后验概率/成员值，更新 μ_1 和 μ_2：

$$
\begin{cases} \mu_1^{(1)} = \dfrac{\sum_{i=1}^{n} \left\{ \underbrace{f_{Y|X}\left(C_1 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)}_{\text{Membership score}} \cdot x^{(i)} \right\}}{\sum_{i=1}^{n} f_{Y|X}\left(C_1 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)} = \dfrac{\sum_{i=1}^{n} \left\{ f_{Y|X}\left(C_1 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right) \cdot x_i \right\}}{n\alpha_1^{(1)}} = 0.11073 \\ \mu_2^{(1)} = \dfrac{\sum_{i=1}^{n} \left\{ \underbrace{f_{Y|X}\left(C_2 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)}_{\text{Membership score}} \cdot x^{(i)} \right\}}{\sum_{i=1}^{n} f_{Y|X}\left(C_2 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)} = \dfrac{\sum_{i=1}^{n} \left\{ f_{Y|X}\left(C_2 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right) \cdot x^{(i)} \right\}}{n\alpha_2^{(1)}} = 0.38248 \end{cases} \tag{10}
$$

上式相当于求加权均值。后验概率/成员值相当于样本数据从属于不同聚类的权重。

标准差

同理，求加权方法，更新 σ_1 和 σ_2：

$$
\begin{cases} \sigma_1^{(1)} = \sqrt{\dfrac{\sum_{i=1}^{n} \left\{ \underbrace{f_{Y|X}\left(C_1 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)}_{\text{Membership score}} \cdot \left(x^{(i)} - \mu_1^{(1)}\right)^2 \right\}}{N\alpha_1^{(1)}}} = 2.8303 \\ \sigma_2^{(1)} = \sqrt{\dfrac{\sum_{i=1}^{n} \left\{ \underbrace{f_{Y|X}\left(C_2 \middle| x^{(i)}, \boldsymbol{\theta}^{(0)}\right)}_{\text{Membership score}} \cdot \left(x^{(i)} - \mu_2^{(1)}\right)^2 \right\}}{N\alpha_2^{(1)}}} = 2.5922 \end{cases} \tag{11}
$$

全新参数

这样，我们便得到了一组全新的参数 $\boldsymbol{\theta}^{(1)}$：

$$
\begin{cases} \alpha_1^{(1)} = p_Y(C_1) = 0.49379, \quad \alpha_2^{(1)} = p_Y(C_2) = 0.50621 \\ \mu_1^{(1)} = 0.11073, \quad \mu_2^{(1)} = 0.38248 \\ \sigma_1^{(1)} = 2.8303, \quad \sigma_2^{(1)} = 2.5922 \end{cases} \tag{12}
$$

证据因子

根据全概率公式，第 i 个数据点证据因子 $f_X(x^{(i)}, \boldsymbol{\theta})$ 可以通过叠加联合概率得到：

$$\begin{aligned}\underbrace{f_X\left(x^{(i)},\boldsymbol{\theta}\right)}_{\text{Evidence}} &= \sum_{k=1}^{K}\underbrace{f_{X,Y}\left(x^{(i)},C_k,\boldsymbol{\theta}\right)}_{\text{Joint}} \\ &= \sum_{k=1}^{K}\underbrace{p_Y\left(C_k,\boldsymbol{\theta}\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x^{(i)}|C_k,\boldsymbol{\theta}\right)}_{\text{Likelihood}}\end{aligned} \tag{13}$$

对数似然函数

构造对数似然函数 (log likelihood function) $L(\boldsymbol{\theta})$，如下：

$$L(\boldsymbol{\theta}) = \overbrace{\ln\left(\underbrace{\prod_{i=1}^{n} f_X\left(x^{(i)},\boldsymbol{\theta}\right)}_{\text{Likelihood function}}\right)}^{\text{Log likelihood function}} = \sum_{i=1}^{n}\left[\ln f_X\left(x^{(i)},\boldsymbol{\theta}\right)\right] \tag{14}$$

对数似然函数 $L(\boldsymbol{\theta})$ 就是样本数据证据因子之积，再求对数。

取对数的叫做对数似然函数，而不做对数处理的叫做似然函数 (likelihood function)。白话说，这里的“似然”指的是“可能性”。

对于似然函数陌生的同学可以参考《统计至简》第 16、20 章。

不管是似然函数，还是对数似然函数，反映的都是在特定参数 $\boldsymbol{\theta}$ 取值下，当前样本集合的可能性。

将 (13) 代入 (14) 可以得到：

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n}\left\{\ln\left[\sum_{k=1}^{K}\underbrace{p_Y\left(C_k,\boldsymbol{\theta}\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x^{(i)}|C_k,\boldsymbol{\theta}\right)}_{\text{Likelihood}}\right]\right\} \tag{15}$$

对于本例二聚类问题，对数似然函数值可以通过下式计算获得：

$$L\left(\boldsymbol{\theta}^{(1)}\right) = \sum_{i=1}^{n}\left\{\ln\left[\underbrace{p_Y\left(C_1,\boldsymbol{\theta}^{(1)}\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x^{(i)}|C_1,\boldsymbol{\theta}^{(1)}\right)}_{\text{Likelihood}} + \underbrace{p_Y\left(C_2,\boldsymbol{\theta}^{(1)}\right)}_{\text{Prior}}\underbrace{f_{X|Y}\left(x^{(i)}|C_2,\boldsymbol{\theta}^{(1)}\right)}_{\text{Likelihood}}\right]\right\} \tag{16}$$

代入 (12) 列出的本轮参数以及样本数据，得到 $L(\boldsymbol{\theta}^{(1)}) = -1.9104$。

下面便是重复 E 步和 M 步，直到满足收敛条件。

14.4 迭代过程

12 轮迭代

经过 12 轮迭代，参数 $\boldsymbol{\theta}$ 如下：

$$\begin{cases} \alpha_1^{(12)} = 0.49105, \quad \alpha_2^{(12)} = 0.50895 \\ \mu_1^{(12)} = -0.81597, \quad \mu_2^{(12)} = 1.5396 \\ \sigma_1^{(12)} = 2.4602, \quad \sigma_2^{(12)} = 0.49993 \end{cases} \tag{17}$$

图 5 到图 7 给出第 12 轮迭代结果。本轮对数似然函数值 $L(\boldsymbol{\theta}^{(12)}) = -1.7344$。

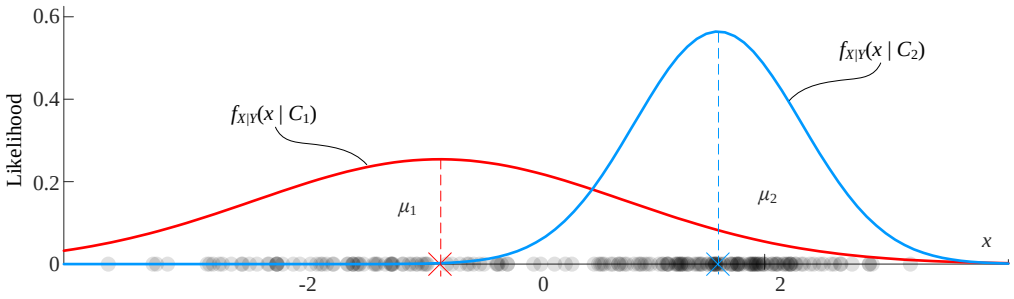


图 5. 经过 12 轮迭代参数对应的似然概率 $f_{X|Y}(x \mid C_1)$ 和 $f_{X|Y}(x \mid C_2)$

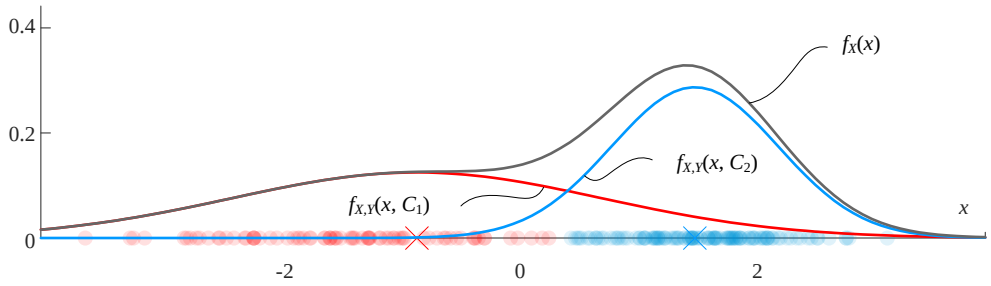


图 6. 经过 12 轮迭代参数对应的 $f_{X,Y}(x, C_1)$、$f_{X,Y}(x, C_2)$ 和 $f_X(x)$

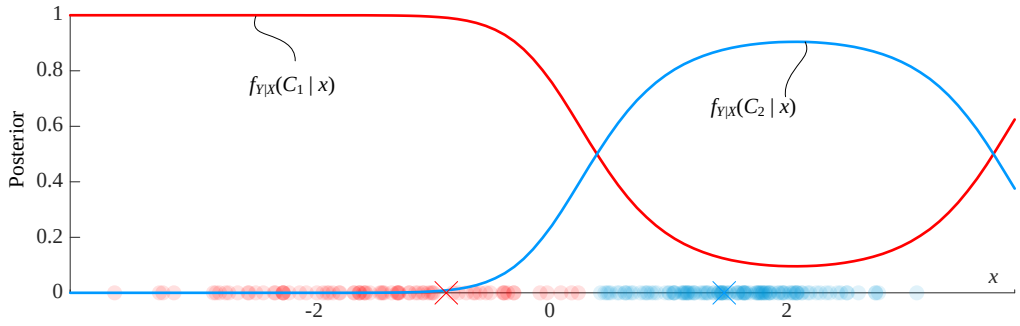


图 7. 经过 12 轮迭代参数对应的后验概率 $f_{Y|X}(C_1 \mid x)$ 和 $f_{Y|X}(C_2 \mid x)$

36 轮迭代

经过 36 轮迭代，得到的参数 $\boldsymbol{\theta}$ 如下：

$$\begin{cases} \alpha_1^{(36)} = 0.410, \quad \alpha_2^{(36)} = 0.590 \\ \mu_1^{(36)} = -1.325, \quad \mu_2^{(36)} = 1.493 \\ \sigma_1^{(36)} = 1.329, \quad \sigma_2^{(36)} = 0.364 \end{cases} \tag{18}$$

图 8 到图 10 所示为经过 36 轮迭代得到的结果。本轮对数似然函数值 $L(\boldsymbol{\theta}^{(36)}) = -1.7232$。

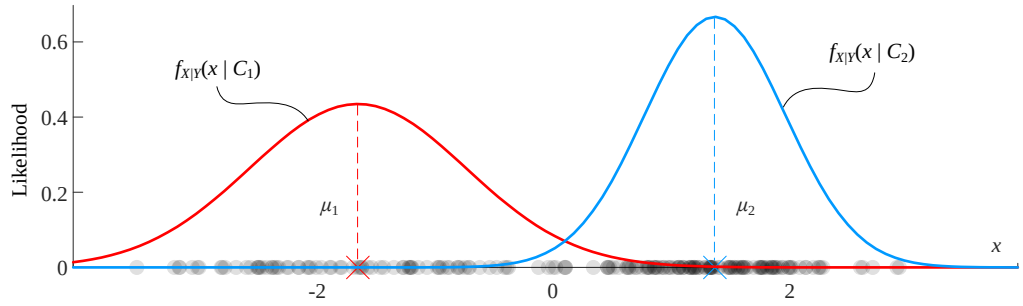


图 8. 经过 36 轮迭代参数对应的似然概率 $f_{X|Y}(x \mid C_1)$ 和 $f_{X|Y}(x \mid C_2)$

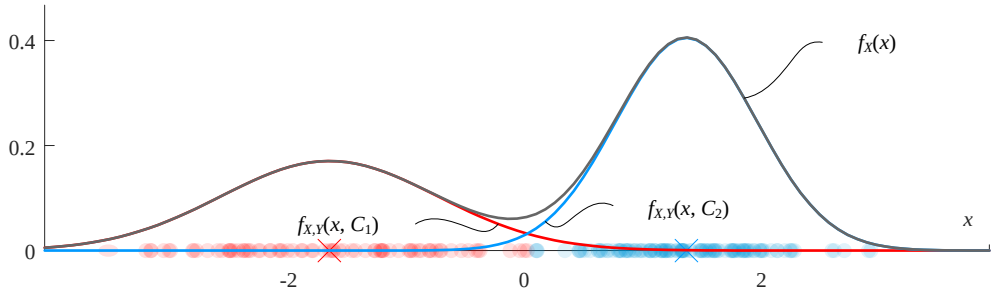


图 9. 经过 36 轮迭代参数对应的 $f_{X,Y}(x, C_1)$、$f_{X,Y}(x, C_2)$ 和 $f_X(x)$

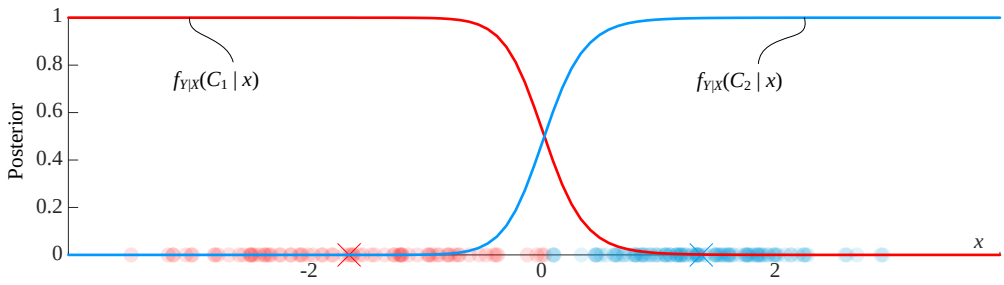


图 10. 经过 36 轮迭代参数对应的 $f_{Y|X}(C_1 \mid x)$ 和 $f_{Y|X}(C_2 \mid x)$

本例设置的迭代截止条件是，要么和上一轮相比对数似然函数 $L(\boldsymbol{\theta})$ 值变化小于 0.00001，要么迭代次数超过 50 次；最先满足两者之一，则迭代停止。

迭代收敛过程

图 11 所示为 36 次迭代，对数似然函数 $L(\boldsymbol{\theta})$ 不断收敛过程。第 15 轮迭代之后，对数似然函数 $L(\boldsymbol{\theta})$ 值便趋于稳定。

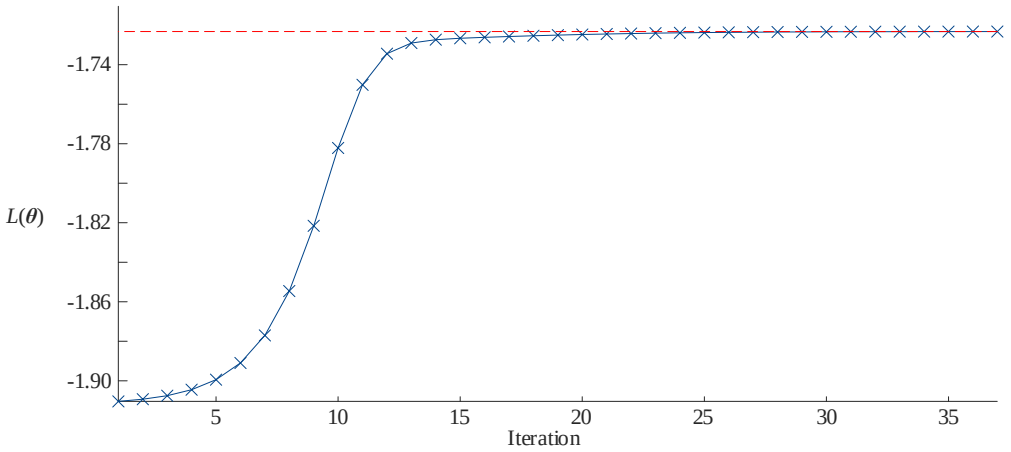


图 11. 经过 36 次迭代，对数似然函数 $L(\boldsymbol{\theta})$ 不断收敛过程

本例是单特征、二聚类问题，因此 $\boldsymbol{\theta}$ 共有 6 个参数；在迭代过程中，这 6 个参数数值也在不断收敛。图 12 展示参数 α_1 和 α_2 不断收敛过程；图 13 所示为参数 μ_1 和 μ_2 不断收敛过程；图 14 为参数 μ_1 和 μ_2 不断收敛过程。

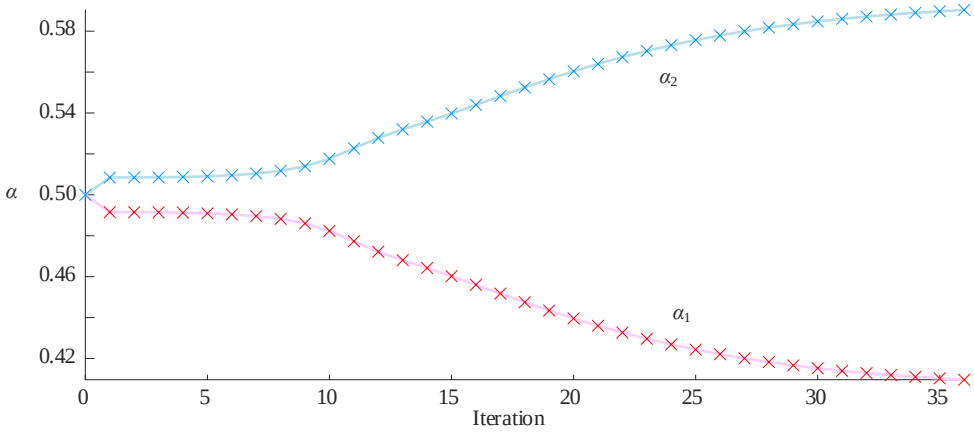


图 12. 经过 36 次迭代，参数 α_1 和 α_2 不断收敛过程

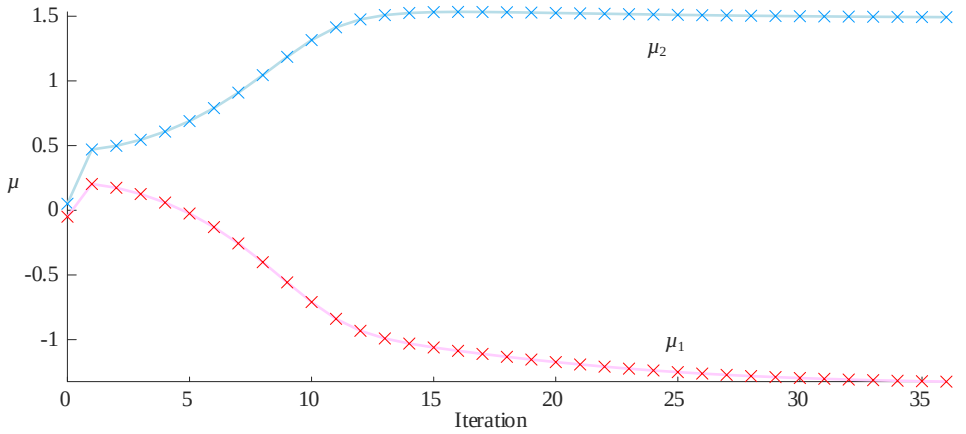


图 13. 经过 36 次迭代，参数 μ_1 和 μ_2 不断收敛过程

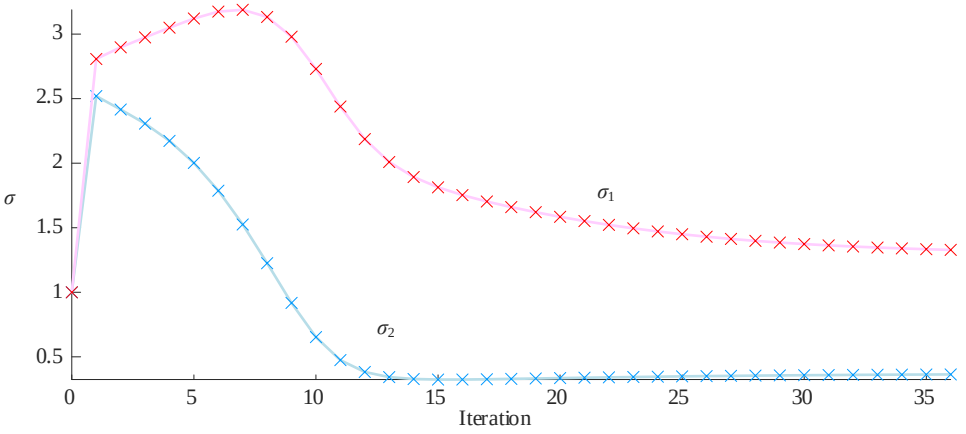


图 14. 经过 36 次迭代，参数 σ_1 和 σ_2 不断收敛过程

EM 算法的迭代过程便是随着参数不断迭代更新，对数似然函数 $L(\boldsymbol{\theta})$ 数值不断增大过程，直到满足收敛条件。EM 算法不仅仅是针对 $L(\boldsymbol{\theta})$ 收敛过程，也是对于参数 $\boldsymbol{\theta}$ 的收敛过程。

14.5 多元 GMM 迭代

多元 EM 算法和本章前文介绍的一元 EM 算法思路完全一致。多元 EM 算法引入大量矩阵运算。本节以二元样本数据聚类为例逐步介绍多元 EM 算法。

图 15 所示为两特征样本数据分布及直方图。

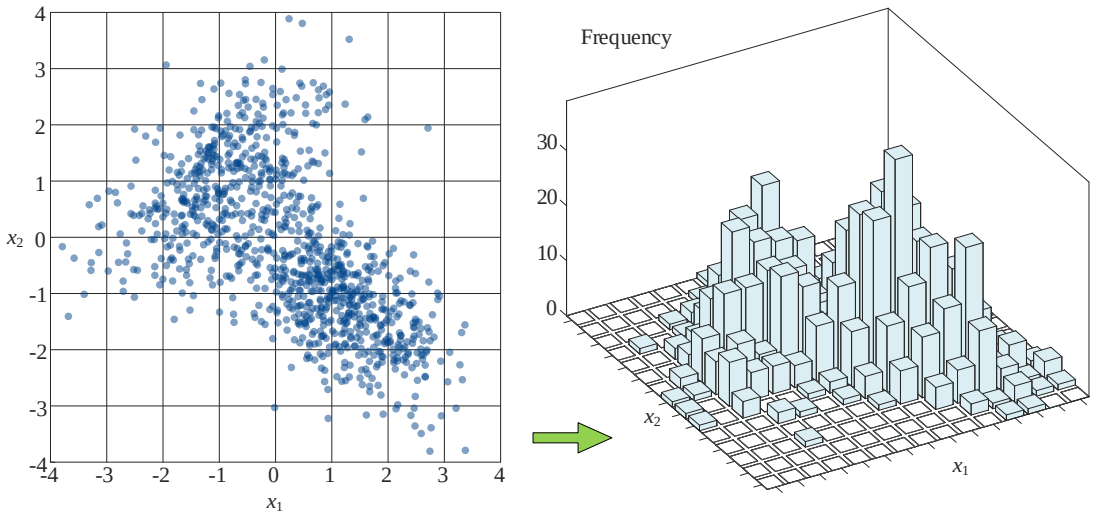


图 15. 两特征样本数据分布

初始化

首先初始化参数 $\boldsymbol{\theta}$：

$$\boldsymbol{\theta}^{(0)} = \left\{\alpha_1^{(0)}, \alpha_2^{(0)}, \boldsymbol{\mu}_1^{(0)}, \boldsymbol{\mu}_2^{(0)}, \boldsymbol{\Sigma}_1^{(0)}, \boldsymbol{\Sigma}_2^{(0)}\right\} \tag{19}$$

初始化参数 $\boldsymbol{\theta}$ 具体数值如下：

$$\begin{cases} \alpha_1^{(0)} = \mathrm{P}\left(C_1, \boldsymbol{\theta}^{(0)}\right) = 0.5, \quad \alpha_2^{(0)} = \mathrm{P}\left(C_2, \boldsymbol{\theta}^{(0)}\right) = 0.5 \\ \boldsymbol{\mu}_1^{(0)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{\mu}_2^{(0)} = \begin{bmatrix} -1 & 0 \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{\Sigma}_1^{(0)} = \boldsymbol{\Sigma}_2^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{cases} \tag{20}$$

似然概率

假设 $f_{\boldsymbol{X}|Y}(\boldsymbol{x} \mid C_1, \boldsymbol{\theta}^{(0)})$ 和 $f_{\boldsymbol{X}|Y}(\boldsymbol{x} \mid C_2, \boldsymbol{\theta}^{(0)})$ 的概率密度函数 PDF 均为正态分布，具体如下：

$$
\begin{cases}
f_{\chi|Y}\left(\boldsymbol{x}\middle|C_1,\boldsymbol{\theta}^{(0)}\right)=\dfrac{\exp\left(-\dfrac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_1^{(0)}\right)^{\mathrm{T}}\left(\boldsymbol{\Sigma}_1^{(0)}\right)^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_1^{(0)}\right)\right)}{\sqrt{(2\pi)^2\left|\boldsymbol{\Sigma}_1^{(0)}\right|}} \\
f_{\chi|Y}\left(\boldsymbol{x}\middle|C_2,\boldsymbol{\theta}^{(0)}\right)=\dfrac{\exp\left(-\dfrac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_2^{(0)}\right)^{\mathrm{T}}\left(\boldsymbol{\Sigma}_2^{(0)}\right)^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_2^{(0)}\right)\right)}{\sqrt{(2\pi)^2\left|\boldsymbol{\Sigma}_2^{(0)}\right|}}
\end{cases} \tag{21}
$$

证据因子

下一步，估算证据因子概率密度函数 $f_{\chi}(\boldsymbol{x},\boldsymbol{\theta}^{(0)})$：

$$
\begin{aligned}
f_{\chi}(\boldsymbol{x},\boldsymbol{\theta}^{(0)}) &= f_{\chi,Y}\left(\boldsymbol{x},C_1,\boldsymbol{\theta}^{(0)}\right)+f_{\chi,Y}\left(\boldsymbol{x}\cap C_2,\boldsymbol{\theta}^{(0)}\right) \\
&= p_Y\left(C_1,\boldsymbol{\theta}^{(0)}\right)f_{\chi|Y}\left(\boldsymbol{x}\middle|C_1,\boldsymbol{\theta}^{(0)}\right)+p_Y\left(C_2,\boldsymbol{\theta}^{(0)}\right)f_{\chi|Y}\left(\boldsymbol{x}\middle|C_2,\boldsymbol{\theta}^{(0)}\right) \\
&= \frac{1}{2}\times\frac{\exp\left(-\dfrac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_1^{(0)}\right)^{\mathrm{T}}\left(\boldsymbol{\Sigma}_1^{(0)}\right)^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_1^{(0)}\right)\right)}{\sqrt{(2\pi)^2\left|\boldsymbol{\Sigma}_1^{(0)}\right|}}+\frac{1}{2}\times\frac{\exp\left(-\dfrac{1}{2}\left(\boldsymbol{x}-\boldsymbol{\mu}_2^{(0)}\right)^{\mathrm{T}}\left(\boldsymbol{\Sigma}_2^{(0)}\right)^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}_2^{(0)}\right)\right)}{\sqrt{(2\pi)^2\left|\boldsymbol{\Sigma}_2^{(0)}\right|}}
\end{aligned} \tag{22}
$$

图 16 (a) 展示初始化参数 $\boldsymbol{\theta}^{(0)}$对应的 $f_{\chi|Y}(\boldsymbol{x}\mid C_1)$ 和 $f_{\chi|Y}(\boldsymbol{x}\mid C_2)$ 等高线；图 16 (b) 展示 $f_{\chi}(\boldsymbol{x})$ 等高线图。

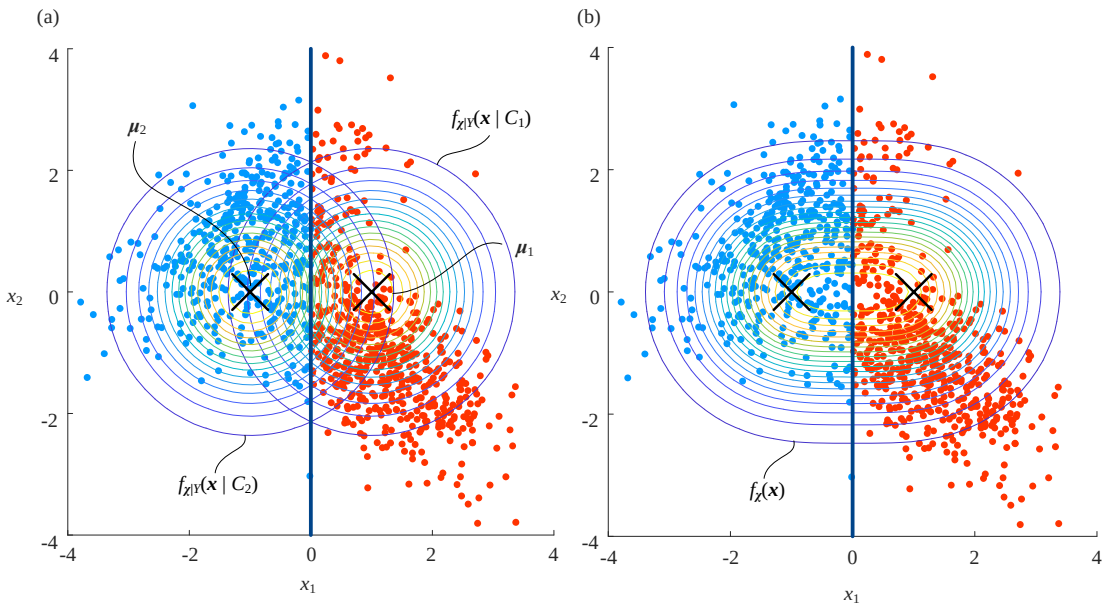


图 16. 初始化参数 $\boldsymbol{\theta}^{(0)}$对应的 $f_{\chi|Y}(\boldsymbol{x}\mid C_1)$ 和 $f_{\chi|Y}(\boldsymbol{x}\mid C_2)$ 等高线，以及 $f_{\chi}(\boldsymbol{x})$ 等高线图

后验概率

根据贝叶斯定理，计算后验概率 $f_{Y|\chi}(C_1\mid\boldsymbol{x},\boldsymbol{\theta}^{(0)})$ 和 $f_{Y|\chi}(C_2\mid\boldsymbol{x},\boldsymbol{\theta}^{(0)})$：

$$\begin{cases} f_{X|Y}\left(\boldsymbol{x}\middle|C_1,\boldsymbol{\theta}^{(0)}\right)=\dfrac{p_Y\left(C_1,\boldsymbol{\theta}^{(0)}\right)f_{X|Y}\left(\boldsymbol{x}\middle|C_1,\boldsymbol{\theta}^{(0)}\right)}{f_X\left(\boldsymbol{x},\boldsymbol{\theta}^{(0)}\right)} \\ f_{X|Y}\left(\boldsymbol{x}\middle|C_2,\boldsymbol{\theta}^{(0)}\right)=\dfrac{p_Y\left(C_2,\boldsymbol{\theta}^{(0)}\right)f_{X|Y}\left(\boldsymbol{x}\middle|C_2,\boldsymbol{\theta}^{(0)}\right)}{f_X\left(\boldsymbol{x},\boldsymbol{\theta}^{(0)}\right)} \end{cases} \tag{23}$$

图 17 所示为初始化参数 $\boldsymbol{\theta}^{(0)}$计算得到后验概率曲面。

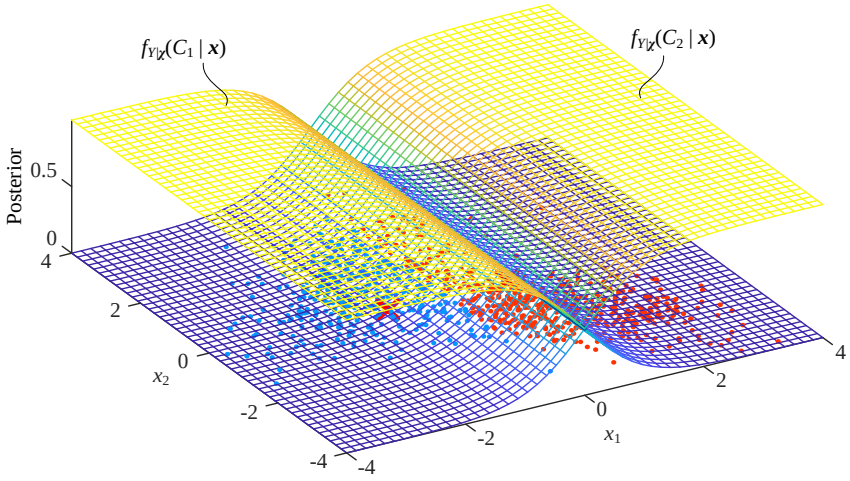


图 17. 初始化参数 $\boldsymbol{\theta}^{(0)}$计算得到 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 曲面

更新参数

下一步进行 EM 算法中 M 步，更新参数。

更新参数 α_1 和 α_2：

$$\begin{cases} \alpha_1^{(1)}=\dfrac{\sum_{i=1}^{n} f_{Y|X}\left(C_1\middle|\boldsymbol{x}^{(i)},\boldsymbol{\theta}^{(0)}\right)}{n}=0.56019 \\ \alpha_2^{(1)}=\dfrac{\sum_{i=1}^{n} f_{Y|X}\left(C_2\middle|\boldsymbol{x}^{(i)},\boldsymbol{\theta}^{(0)}\right)}{n}=0.43981 \end{cases} \tag{24}$$

更新簇质心 $\boldsymbol{\mu}_1$ 和 $\boldsymbol{\mu}_2$：

$$\begin{cases} \boldsymbol{\mu}_1^{(1)}=\dfrac{\sum_{i=1}^{n}\left\{f_{Y|X}\left(C_1\middle|\boldsymbol{x}^{(i)},\boldsymbol{\theta}^{(0)}\right)\boldsymbol{x}^{(i)}\right\}}{n\alpha_1^{(1)}}=\begin{bmatrix}1.098\\-0.764\end{bmatrix} \\ \boldsymbol{\mu}_2^{(1)}=\dfrac{\sum_{i=1}^{n}\left\{f_{Y|X}\left(C_2\middle|\boldsymbol{x}^{(i)},\boldsymbol{\theta}^{(0)}\right)\boldsymbol{x}^{(i)}\right\}}{n\alpha_2^{(1)}}=\begin{bmatrix}-0.8924\\0.4627\end{bmatrix} \end{cases} \tag{25}$$

为了方便运算默认 $\boldsymbol{x}^{(i)}$ 为列向量。

更新簇协方差矩阵 $\boldsymbol{\Sigma}_1$ 和 $\boldsymbol{\Sigma}_2$：

$$
\begin{cases}
\boldsymbol{\Sigma}_1^{(1)} = \dfrac{\sum_{i=1}^{n}\left\{ f_{Y|X}\left(C_1 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(0)}\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_1\right)^{\mathrm{T}}\right\}}{n\alpha_1^{(1)}} = \begin{bmatrix} 0.9346 & -0.7809 \\ -0.7809 & 1.787 \end{bmatrix} \\
\boldsymbol{\Sigma}_2^{(1)} = \dfrac{\sum_{i=1}^{n}\left\{ f_{Y|X}\left(C_2 \middle| \boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(0)}\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_2\right)\left(\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_2\right)^{\mathrm{T}}\right\}}{n\alpha_2^{(1)}} = \begin{bmatrix} 1.034 & -0.1588 \\ -0.1588 & 1.213 \end{bmatrix}
\end{cases} \tag{26}
$$

这样，我们便得到了一组全新的参数 $\boldsymbol{\theta}^{(1)}$。

对数似然函数

构造对数似然函数 $L(\boldsymbol{\theta})$，如下：

$$
L\left(\boldsymbol{\theta}^{(1)}\right) = \ln\left(\prod_{i=1}^{n} f_X\left(\boldsymbol{x}^{(i)}, \boldsymbol{\theta}^{(1)}\right)\right) \tag{27}
$$

代入 (24) 、 (25) 和 (26) 中更新得到的参数，计算得到对数似然值 $L(\boldsymbol{\theta}^{(1)}) = -3.213045$。

图 18 和图 19 展示 $\boldsymbol{\theta}^{(1)}$参数对应的概率曲面。分别比较图 16 和图 17，可以发现图 18 和图 19 所示聚类决策边界已经发生显著变化。

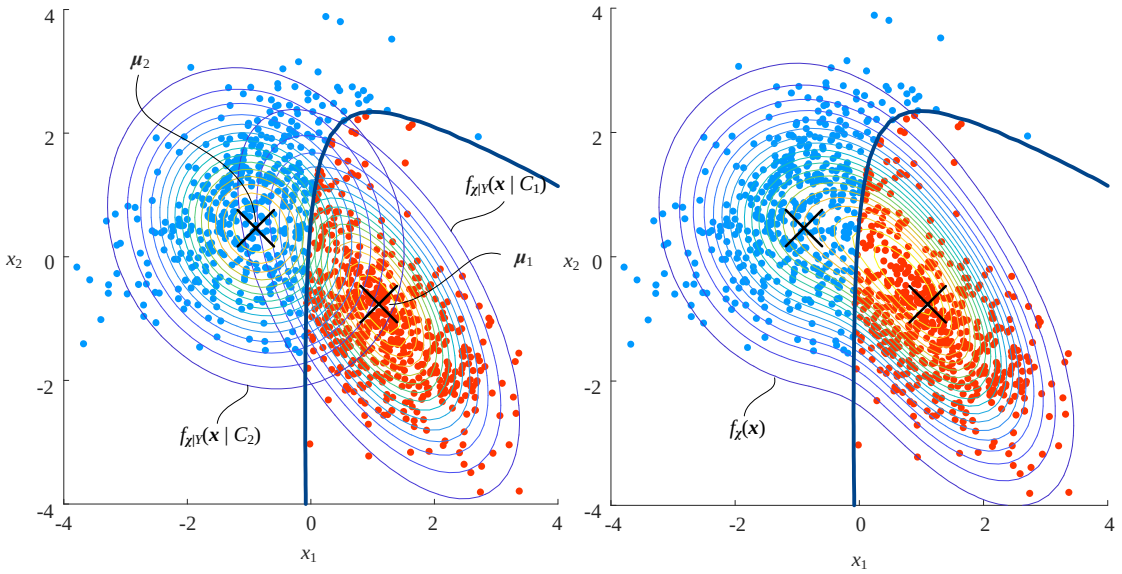


图 18. 参数 $\boldsymbol{\theta}^{(1)}$对应的 $f_{X|Y}(\boldsymbol{x} \mid C_1)$ 和 $f_{X|Y}(\boldsymbol{x} \mid C_2)$ 等高线，以及 $f_X(\boldsymbol{x})$ 等高线图

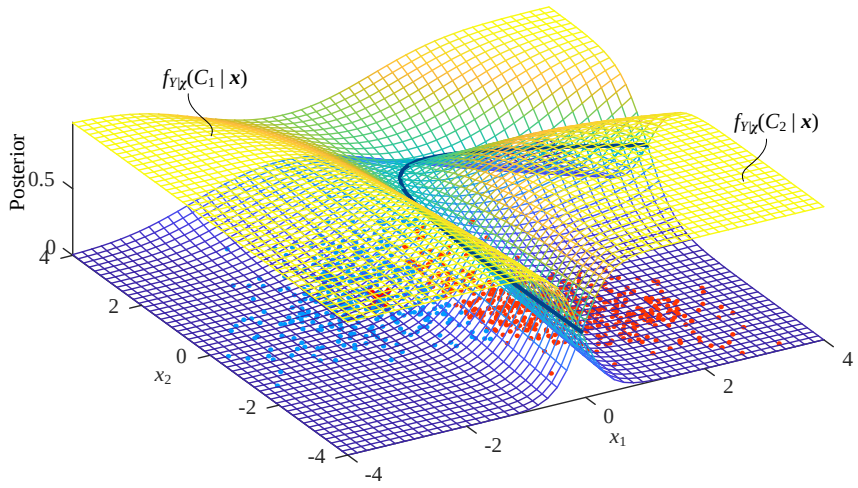


图 19. 参数 $\boldsymbol{\theta}^{(1)}$计算得到 $f_{Y|X}(C_1 \mid \boldsymbol{x})$ 和 $f_{Y|X}(C_2 \mid \boldsymbol{x})$ 曲面

第二轮迭代

进入第 2 轮迭代，更新参数 $\boldsymbol{\theta}^{(2)}$:

$$
\begin{cases}
\alpha_1^{(2)} = \mathrm{P}\left(C_1, \boldsymbol{\theta}^{(2)}\right) = 0.56481, \quad \alpha_2^{(2)} = \mathrm{P}\left(C_2, \boldsymbol{\theta}^{(2)}\right) = 0.43519 \\
\boldsymbol{\mu}_1^{(2)} = \begin{bmatrix} 1.097 & -0.84 \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{\mu}_2^{(2)} = \begin{bmatrix} -0.9121 & 0.5744 \end{bmatrix}^{\mathrm{T}} \\
\boldsymbol{\Sigma}_1^{(2)} = \begin{bmatrix} 0.9179 & -0.7818 \\ -0.7818 & 1.614 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2^{(2)} = \begin{bmatrix} 1.02 & 0.07167 \\ 0.07167 & 1.153 \end{bmatrix}
\end{cases} \tag{28}
$$

第 11 轮迭代

经过 11 轮迭代，满足优化结束条件，并获得更新参数 $\boldsymbol{\theta}^{(11)}$:

$$
\begin{cases}
\alpha_1^{(11)} = \mathrm{P}\left(C_1, \boldsymbol{\theta}^{(11)}\right) = 0.57516, \quad \alpha_2^{(11)} = \mathrm{P}\left(C_2, \boldsymbol{\theta}^{(11)}\right) = 0.42484 \\
\boldsymbol{\mu}_1^{(11)} = \begin{bmatrix} 1.096 & -1.114 \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{\mu}_2^{(11)} = \begin{bmatrix} -0.9589 & 0.9795 \end{bmatrix}^{\mathrm{T}} \\
\boldsymbol{\Sigma}_1^{(11)} = \begin{bmatrix} 0.8938 & -0.4735 \\ -0.4735 & 0.7659 \end{bmatrix}, \quad \boldsymbol{\Sigma}_2^{(11)} = \begin{bmatrix} 0.9627 & 0.5045 \\ 0.5045 & 0.9269 \end{bmatrix}
\end{cases} \tag{29}
$$

图 20 和图 21 展示完成迭代后曲面等高线结果。

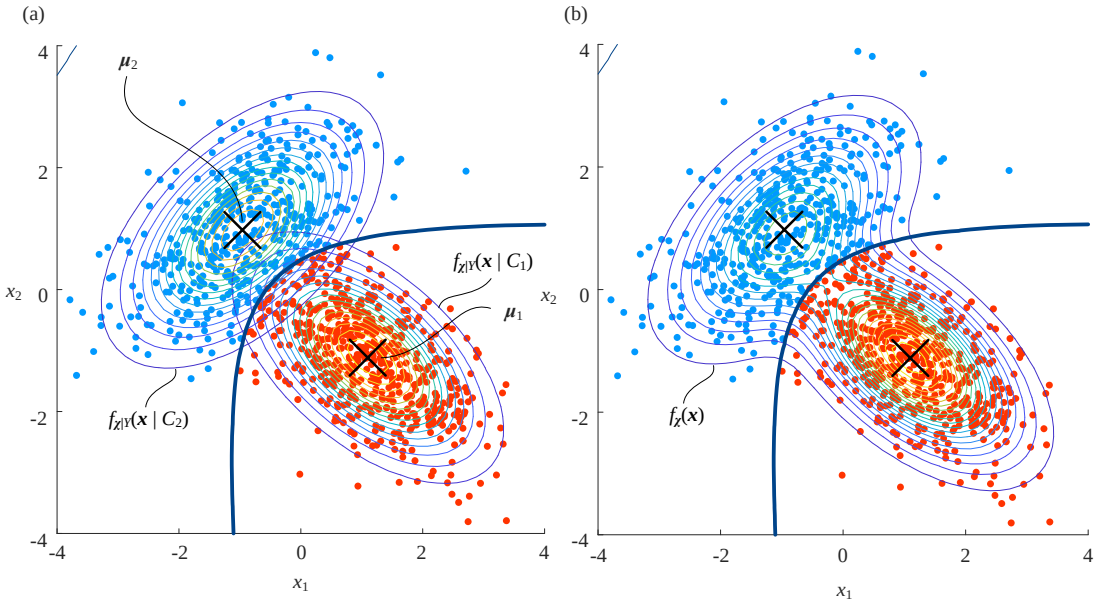


图 20. 参数 $\theta^{(11)}$对应的 $f_{X|Y}(x \mid C_1)$ 和 $f_{X|Y}(x \mid C_2)$ 等高线，以及 $f_X(x)$ 等高线图

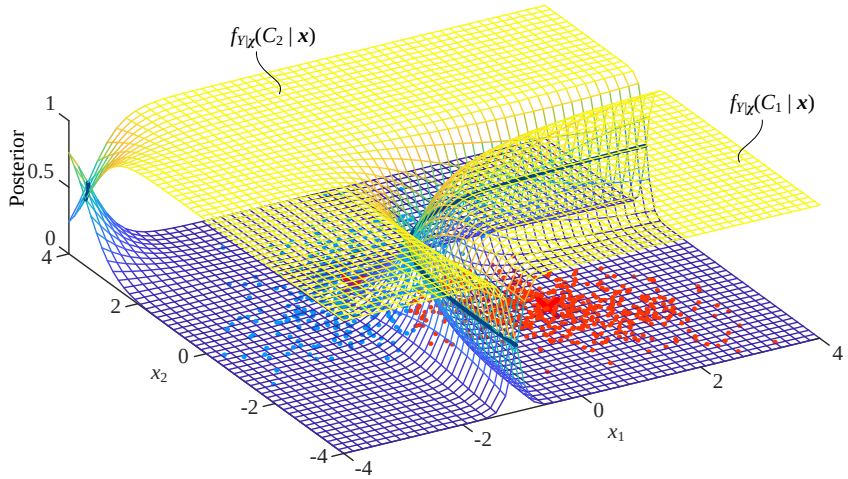


图 21. 参数 $\theta^{(11)}$计算得到 $f_{Y|X}(C_1 \mid x)$ 和 $f_{Y|X}(C_2 \mid x)$ 曲面

迭代收敛过程

图 22 展示的是经过 11 次迭代 $L(\theta)$ 递增收敛过程。相信大家看过图 11 和图 22 这两幅图，便明白为什么对数似然函数 $L(\theta)$ 是参数 θ 的函数了。参数 θ 相当于未知数，由于不存在解析解，只能通过迭代优化求解参数 θ。整个过程就是找到描述样本数据集合最佳参数 θ。

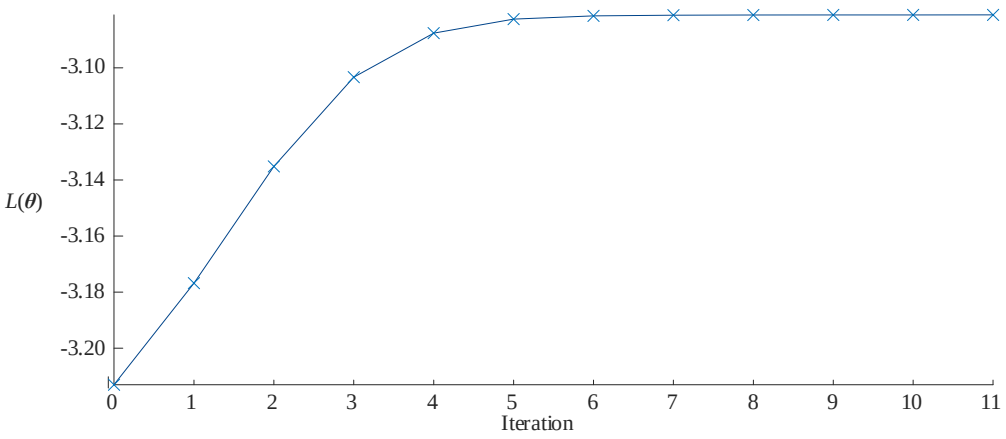


图 22. 经过 11 次迭代，似然函数 $L(\boldsymbol{\theta})$ 不断收敛过程

图 23 所示为经过 11 次迭代，参数 α_1 和 α_2 不断收敛过程。参数 α_1 和 α_2 也代表着分别划分到 C_1 和 C_2 样本数据的比例。

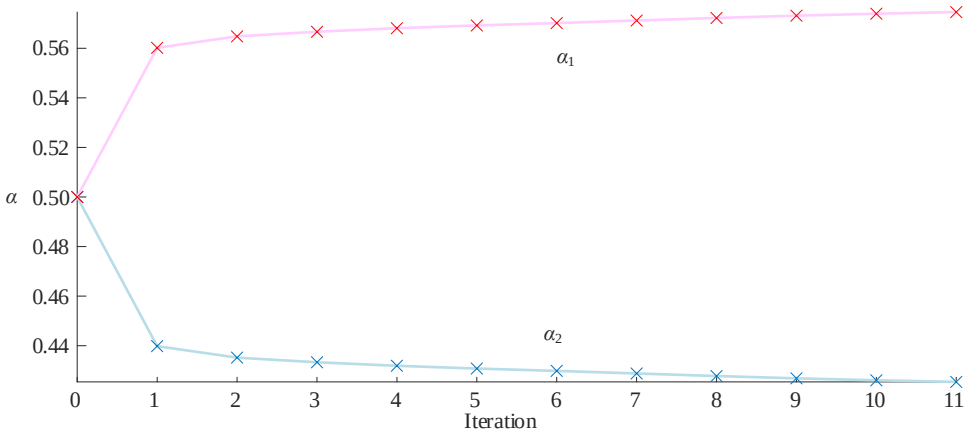


图 23. 11 次迭代，参数 α_1 和 α_2 不断收敛过程

为了更好地可视化二元高斯分布参数——质心和协方差——变化过程，我们利用椭圆来表达协方差，而椭圆中心所在位置便是簇质心。图 24 很好地展示 11 次迭代，两个二元高斯分布质心和协方差不断变化过程。图 25 则展示决策边界随着迭代不断变化。

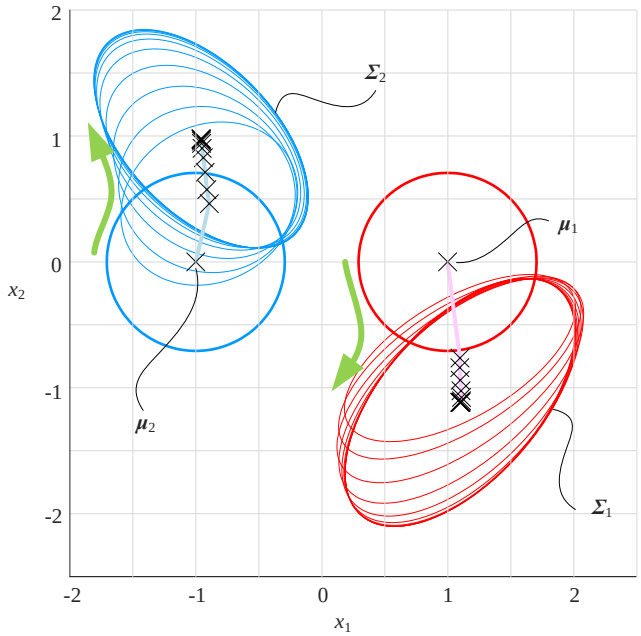


图 24. 11 次迭代，二元高斯分布质心和协方差不断变化过程

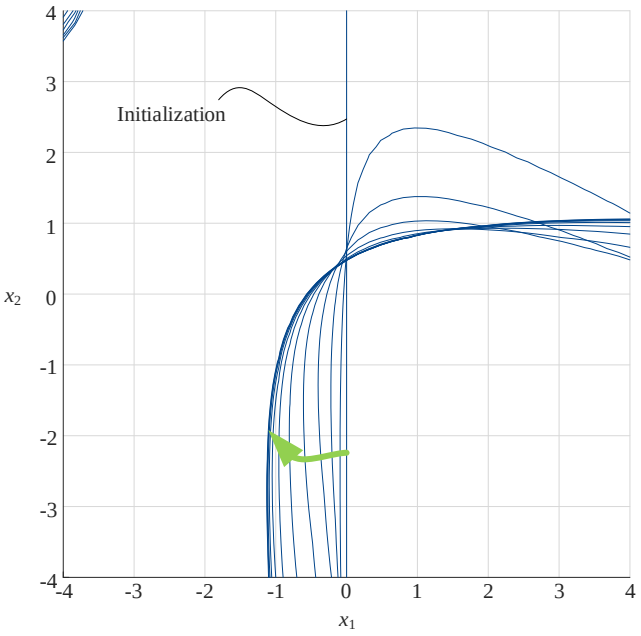


图 25. 11 次迭代，决策边界不断变化过程

EM 算法很有可能迭代收敛在局部极大值处，而非全局最大值；常用的解决办法是，选取不同初始值进行迭代优化；比较对数似然函数 $L(\boldsymbol{\theta})$ 收敛值，从不同优化解中选取理想解。

EM 算法是一种迭代算法，用于在不完全观测的情况下，通过已知的观测数据来估计模型参数。其核心思想是通过不断迭代，利用已知数据计算未知参数的最大似然估计。EM 算法的迭代包括两个步骤：E 步骤和 M 步骤，其中 E 步骤计算隐变量的后验概率，M 步骤利用后验概率重新估计参数。EM 算法通常用于处理混合模型、隐马尔可夫模型等问题，具有广泛的应用，如聚类、密度估计、图像处理等领域。

15 层次聚类

Hierarchical Clustering

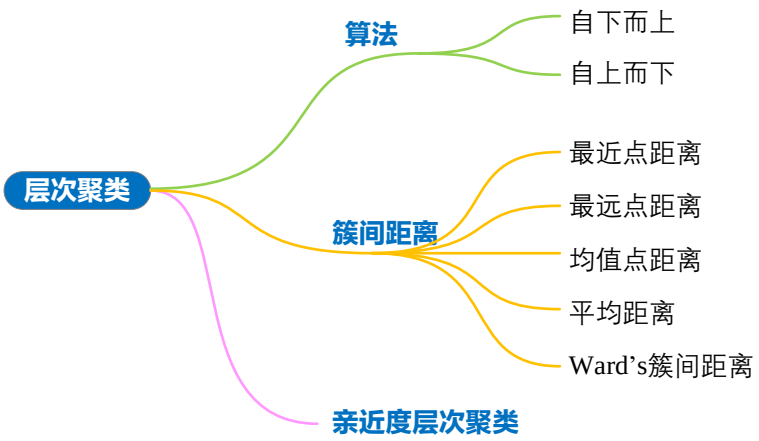
基于数据之间距离，自下而上聚合，或自上而下分裂

如果不能简单地解释某个理论，说明你并没有真正理解它。

If you can't explain it simply, you don't understand it well enough.

—— 阿尔伯特·爱因斯坦 (Albert Einstein) | 理论物理学家 | 1879 ~ 1955

- `numpy.triu()` 提取上三角矩阵
- `scipy.cluster.hierarchy.dendrogram()` 绘制树形图
- `scipy.cluster.hierarchy.linkage()` 计算簇间距离
- `seaborn.clustermap()` 绘制树形图和热图
- `seaborn.heatmap()` 绘制热图
- `sklearn.cluster.AgglomerativeClustering()` 层次聚类函数
- `sklearn.metrics.pairwise.rbf_kernel()` 计算 RBF 核成对亲近度矩阵



15.1 层次聚类

层次聚类 (hierarchical clustering) 算法是一种聚类分析算法。层次聚类依据数据之间的距离远近，或者亲近度大小，将样本数据划分为簇。层次聚类可以通过**自下而上** (agglomerative) 合并，或者**自上而下** (divisive) 分割来构造分层结构聚类。

图 1 所示为根据鸢尾花样本数据前两个特征——花萼长度和宽度——获得的层次聚类**树形图** (dendrogram)。

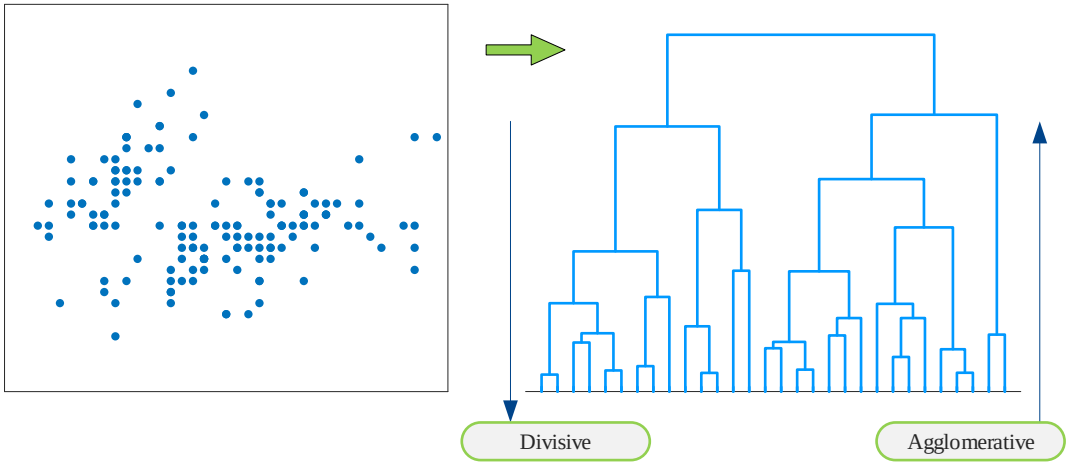


图 1. 区分“自上而下”和“自下而上”层次聚类

自下而上合并

图 2 所示为自下而上合并原理。整个过程有点像“搭积木”，首先以每个数据点本身作为一簇，每次迭代合并“距离”较近或亲近度大的类别，直到最后只剩一簇为止。这个过程可以使用的距离度量或亲和度也是多种多样的。请大家回顾本书第 3 章有关距离度量和亲近度内容。

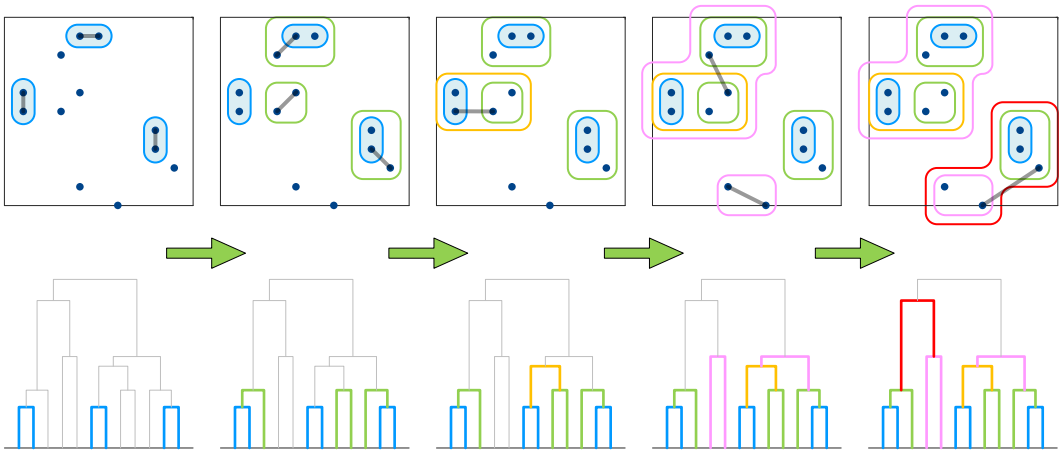


图 2. 层次聚类原理

本章下面首先介绍如何一步步通过自下而上合并获得树形图。大家可能已经注意到，图 2 中不仅仅要考虑，“点”与“点”之间距离，还需要考虑“簇”与“簇”之间的距离。“簇”与“簇之间的距离，也是本章要探讨的核心内容之一。

15.2 树形图

图 3 给出 12 个样本数据在平面上的位置。相信大家还记得**成对距离矩阵** (pairwise distance matrix) 这个概念。图 4 所示 12 个样本数据成对欧氏距离矩阵的热图。

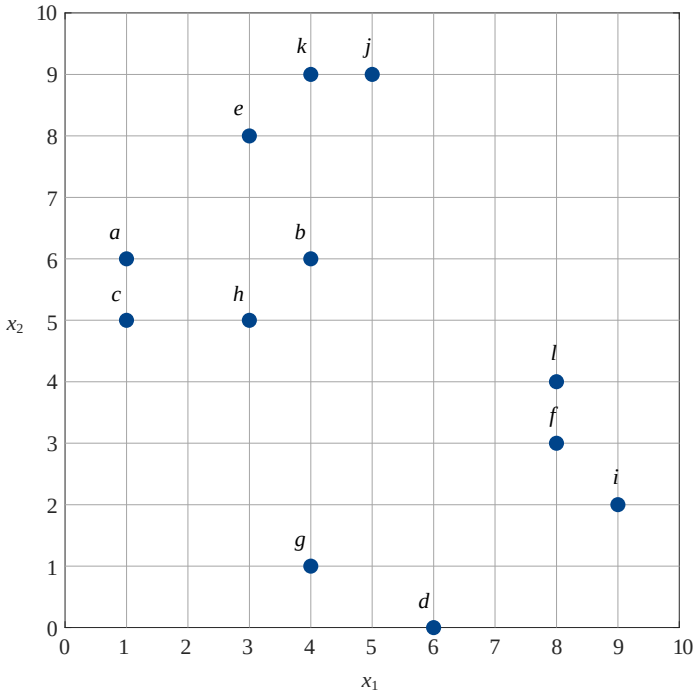


图 3. 12 个样本数据

有了图 4 所示成对欧氏距离矩阵，便可以得到如图 5 所示树形图。树形图横轴对应样本数据编号，纵轴对应数据点间距离和簇间欧氏距离。

观察图 5 树形图，在距离值 2.5 处切一刀，可以将图 3 所示数据分成 3 簇；如果在距离值为 4 处切一刀，可以将图 3 所示数据分成 2 簇。下面，我们一步步介绍如何自下而上构造如图 5 所示树形图。

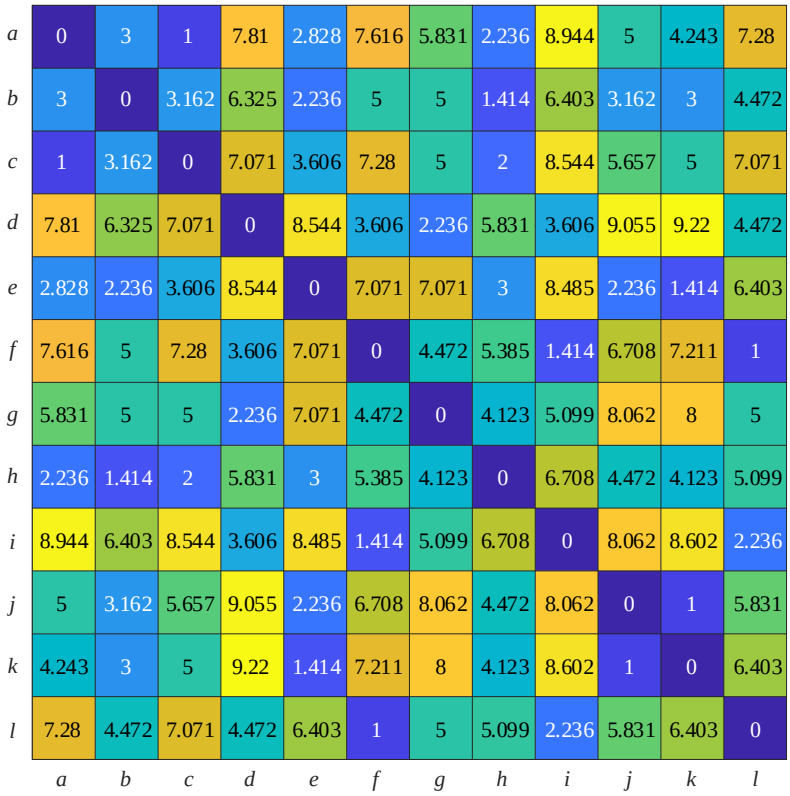


图 4. 12 个样本数据成对距离构成的方阵热图

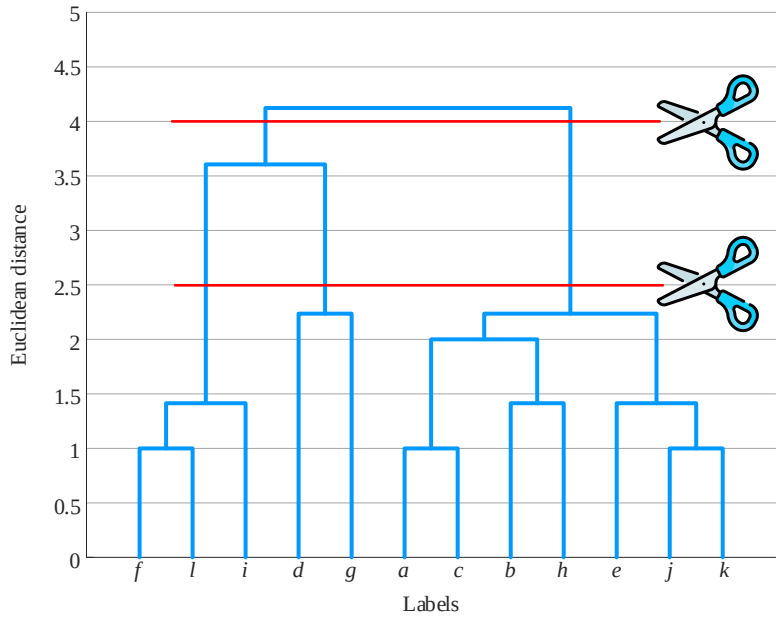


图 5. 数据树形图

第一层

图 6 所示，首先发现 a 和 c、k 和 j、f 和 l 成对距离最短，均为 1；这样我们便构造树形图最底层。这三个成对距离在热图位置如图 8 (a) 所示。

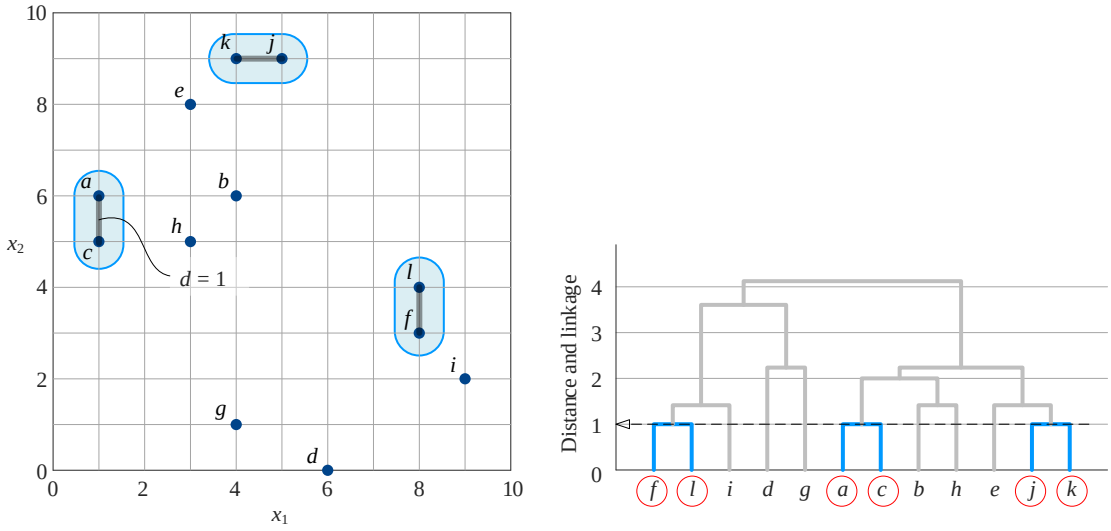


图 6. 构建树形图，第一层

第二层

构造树形图第二层时，遇到一个麻烦——簇间距离如何定义。这里，我们首先采用最简单的**最近点距离** (single linkage 或 nearest neighbor)。最近点距离指的是两个簇样本数据成对距离最近值。

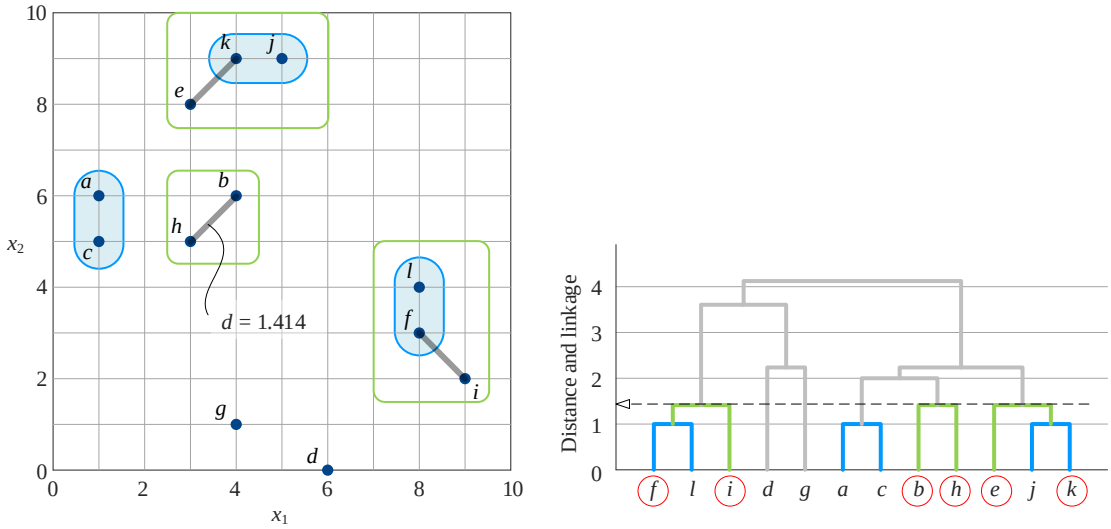


图 7. 构建树形图，第二层

k 和 j、f 和 l 已经分别“成团”；e 距离 k 更近，而 i 距离 f 更近。因此树形图第二层的距离值定为 sqrt(2)，也就是约 1.414。同样，b 和 h 的距离也是 1.414。这样我们便构造得到了如图 7 所示的树形图第二层。这三个“簇间”/“点间”距离在热图位置如图 8 (b) 所示。

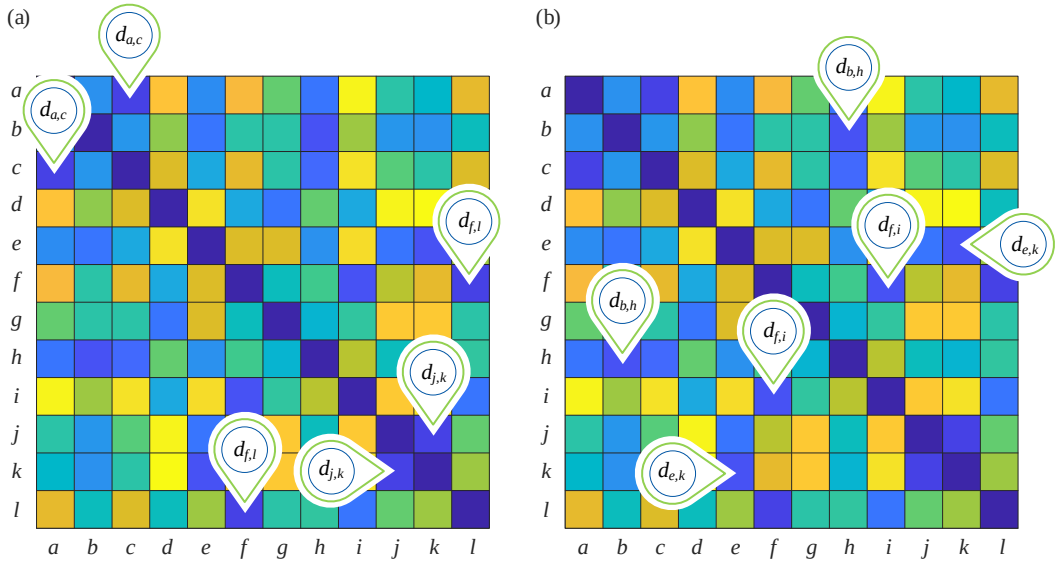


图 8. 成对距离矩阵热图，第一层和第二层距离位置

第三层

再向上一层，利用簇 a 和 c (第一层)、簇 b 和 h (第二层)之间簇间距离 2，从而得到树形图第三层。这个距离在热图上的位置如图 11 (a) 所示。

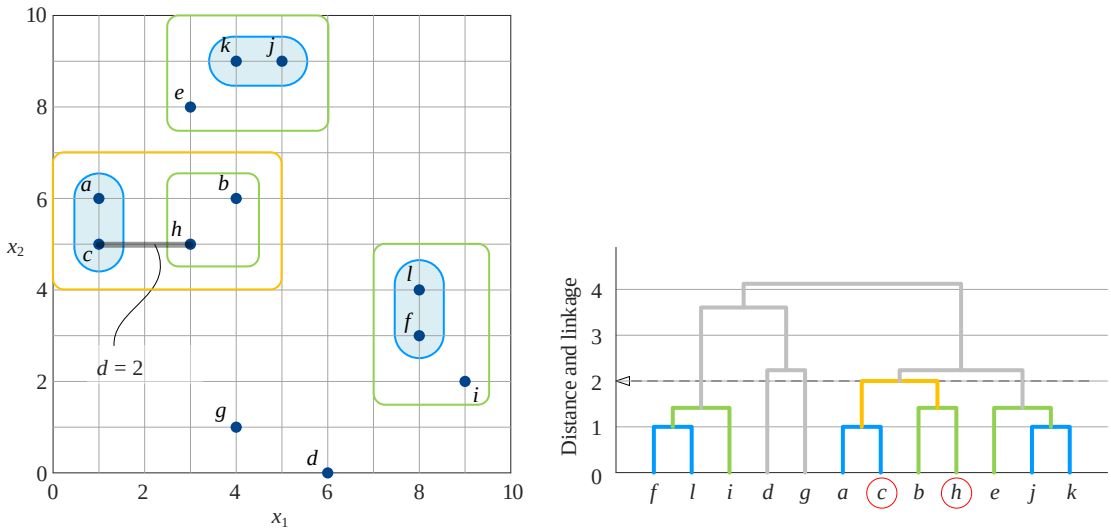


图 9. 构建树形图，第三步

第四层

树形图的第四层采用的距离值为 sqrt(5)，约 2.236。图 10 所示为树形图第四层位置。可以发现此时，所有的数据点均参与聚类，形成 3 簇。

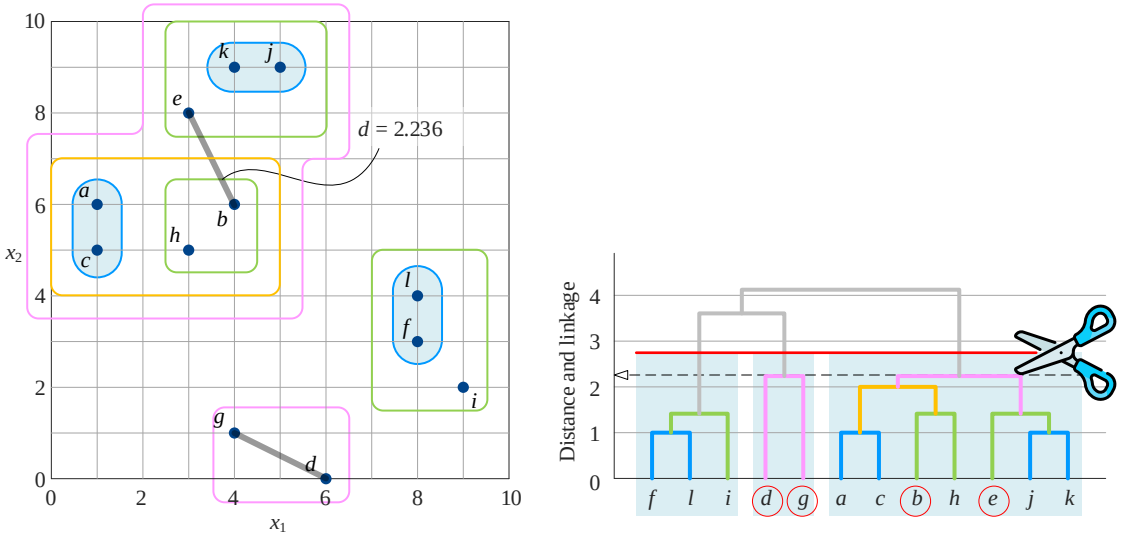


图 10. 构建树形图，第四层

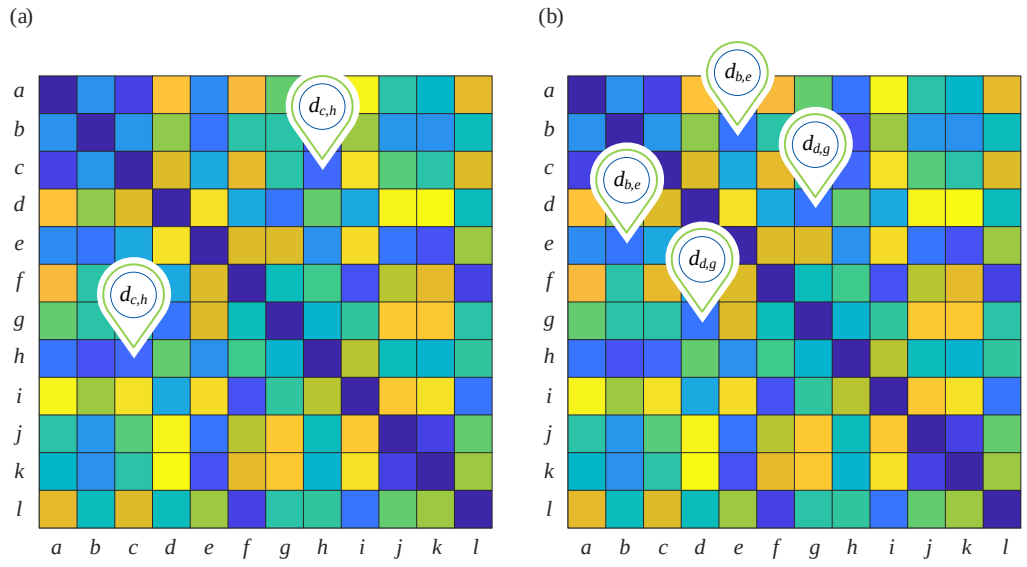


图 11. 成对距离矩阵热图，第三层和第四层距离位置

第五层

图 12 所示为树形图第五层位置。在第五层，样本数据被划分为两簇；再加一层，整个树形图便封顶。

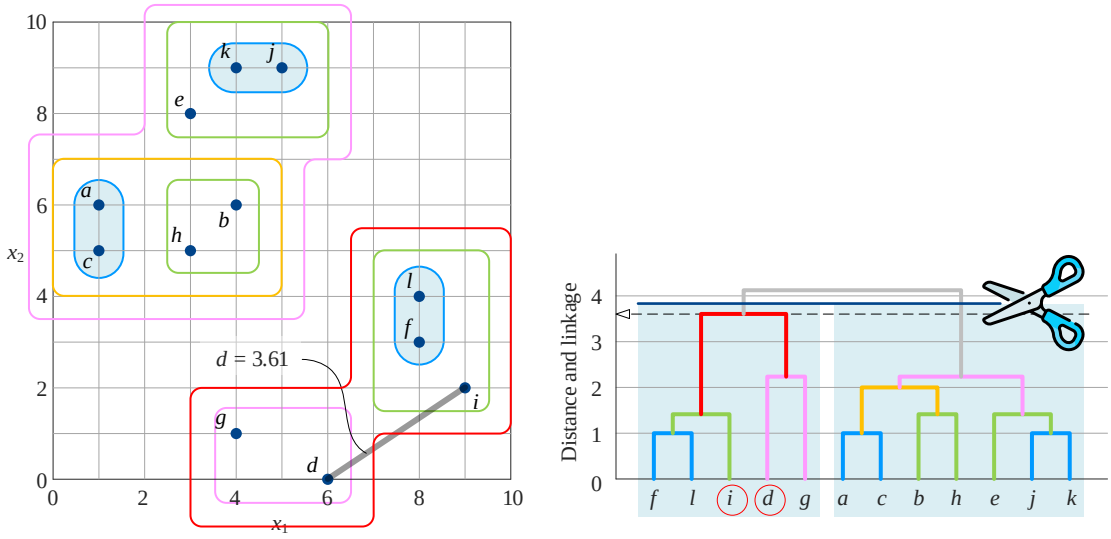


图 12. 构建树形图，第五层

重新排序

树形结构把数据序号重新排列。根据这个顺序，可以得到一个全新的热图，如图 13 所示。根据颜色，图 13 所示热图很容易分为两个区域，对应数据划分为两簇。这便是层次聚类的思路。

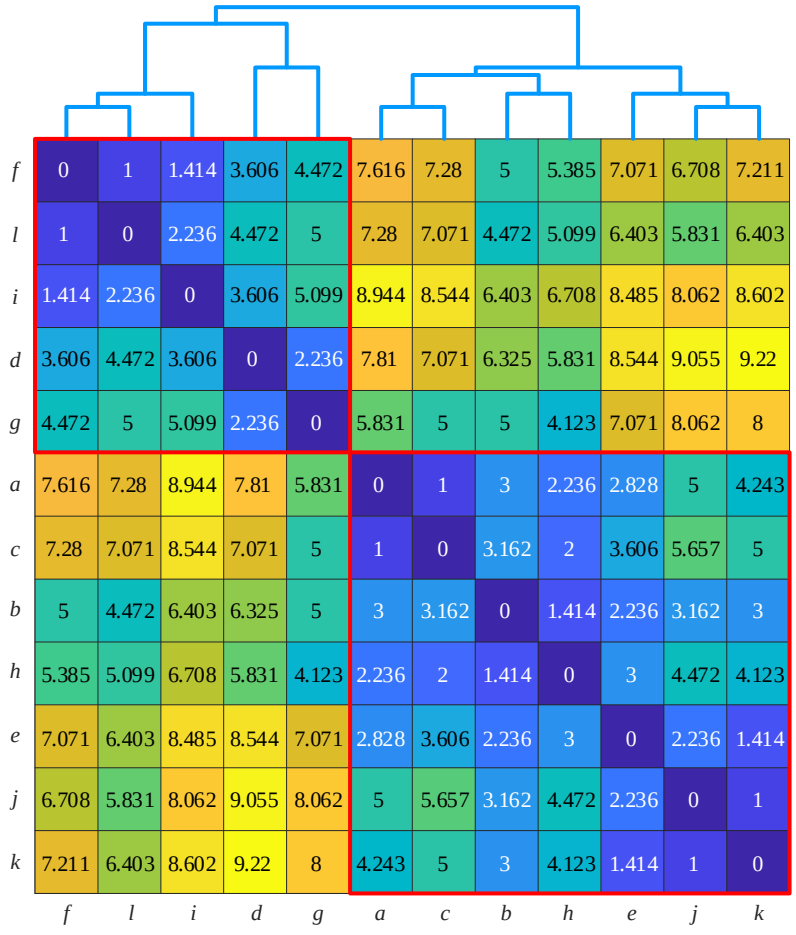


图 13. 按树形图重组数据树形图

15.3 簇间距离

上一节提到，两簇之间的距离可以采用最近点距离；当然，簇间距离也有其他定义。本节介绍常用的几种簇间距离。

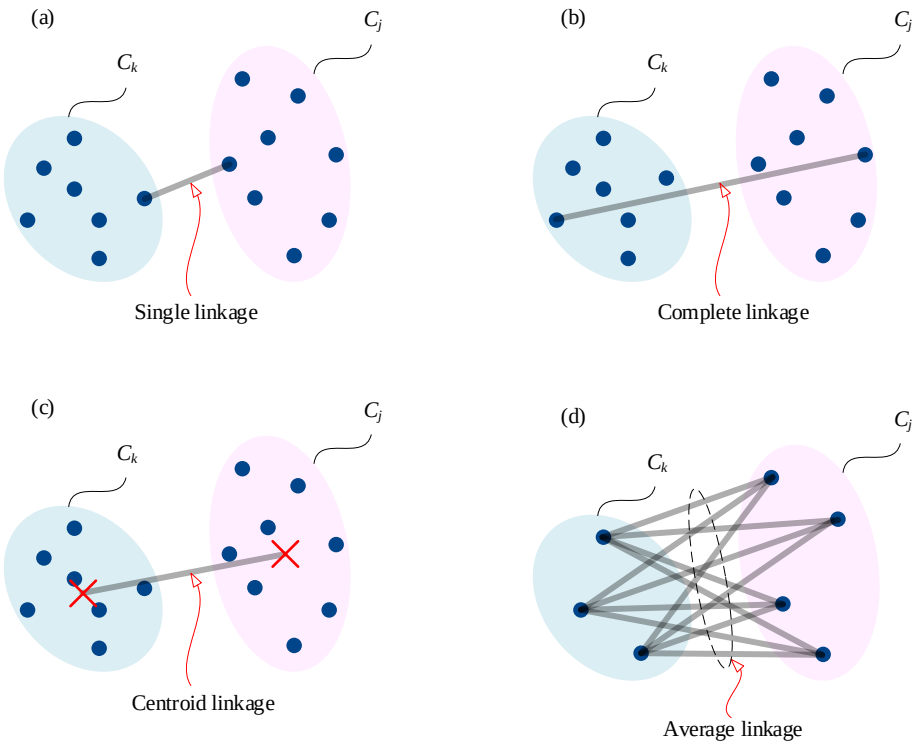


图 14. 簇间距离定义

最近点距离

簇间距离，也叫做**距离值** (linkage distance 或者 linkage)。如图 14 (a) 所示，**最近点距离** (single linkage 或 nearest neighbor)，代号为 `'single'`，指的是两个簇样本数据成对距离最近值：

$$d\left(C_k, C_j\right) = \min_{\boldsymbol{x}\in C_k,\ \boldsymbol{z}\in C_j}\left(\text{dist}\left(\boldsymbol{x},\boldsymbol{z}\right)\right) \tag{1}$$

图 15 所示为，采用 `'single'` 层次聚类得到的树形图和鸢尾花数据聚类结果。可以发现，树形图分支并不均衡，聚类结果不理想。

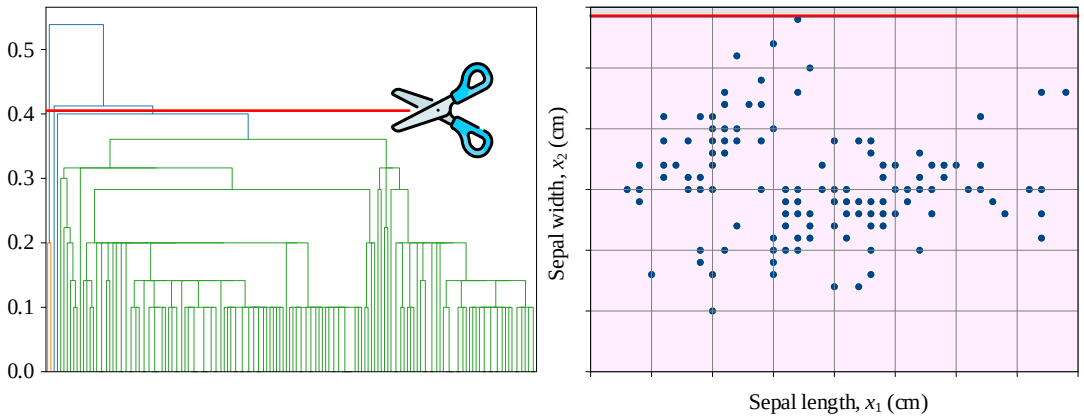


图 15. 鸢尾花聚类结果，'single' 层次聚类

最远点距离

如图 14 (b) 所示，**最远点距离** (complete linkage 或 farthest neighbor) 定义为，两簇样本数据成对距离最远值：

$$d\left(C_k, C_j\right) = \max_{\boldsymbol{x}\in C_k,\ \boldsymbol{z}\in C_j}\left(\text{dist}\left(\boldsymbol{x},\boldsymbol{z}\right)\right) \tag{2}$$

最远点距离代号为 `'complete'`。图 16 所示为，采用 `'complete'` 层次聚类得到的树形图和鸢尾花数据聚类结果。最远点距离对于离群点/噪音点敏感。

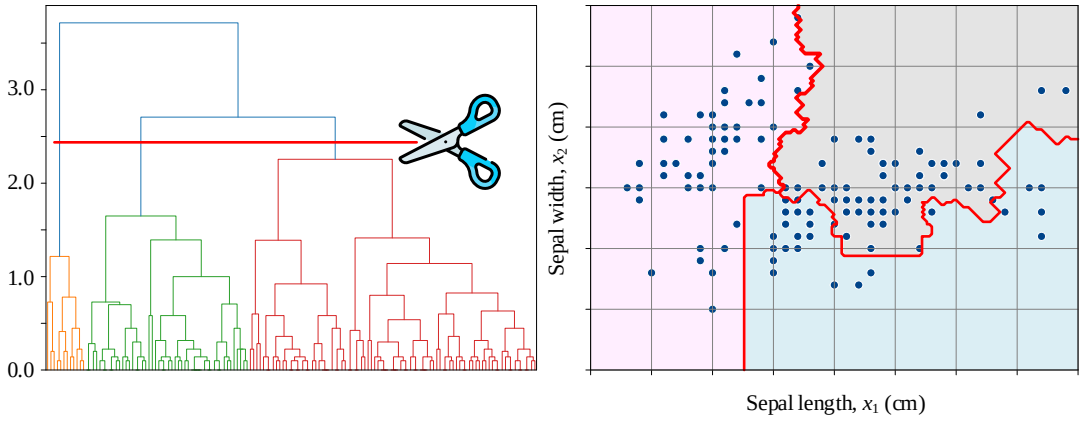


图 16. 鸢尾花聚类结果，'complete' 层次聚类

均值点距离

如图 14 (c) 所示，**均值点距离** (centroid linkage) 采用两个簇样本数据均值点之间的距离：

$$d\left(C_i, C_j\right) = d\left(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j\right) \tag{3}$$

其中，$\boldsymbol{\mu}_i$和$\boldsymbol{\mu}_j$分别为C_i和C_j的质心点。目前 Scikit-learn 中的层次聚类函数并不支持均值点距离；但是`scipy.cluster.hierarchy.linkage`支持均值点距离，代号为`'centroid'`。

平均距离

如图 14 (d) 所示，**平均距离** (average linkage) 采用两簇样本数据成对点之间距离取平均值：

$$d\left(C_k, C_j\right) = \underset{\boldsymbol{x}\in C_k,\ \boldsymbol{z}\in C_j}{\text{mean}}\left(\text{dist}\left(\boldsymbol{x},\boldsymbol{z}\right)\right) = \frac{\sum_{\boldsymbol{x}\in C_k,\ \boldsymbol{z}\in C_j}\text{dist}\left(\boldsymbol{x},\boldsymbol{z}\right)}{\text{count}\left(C_k\right)\cdot\text{count}\left(C_j\right)} \tag{4}$$

平均距离代号为`'average'`。图 17 所示为，采用`'average'`层次聚类得到的树形图和鸢尾花数据聚类结果。

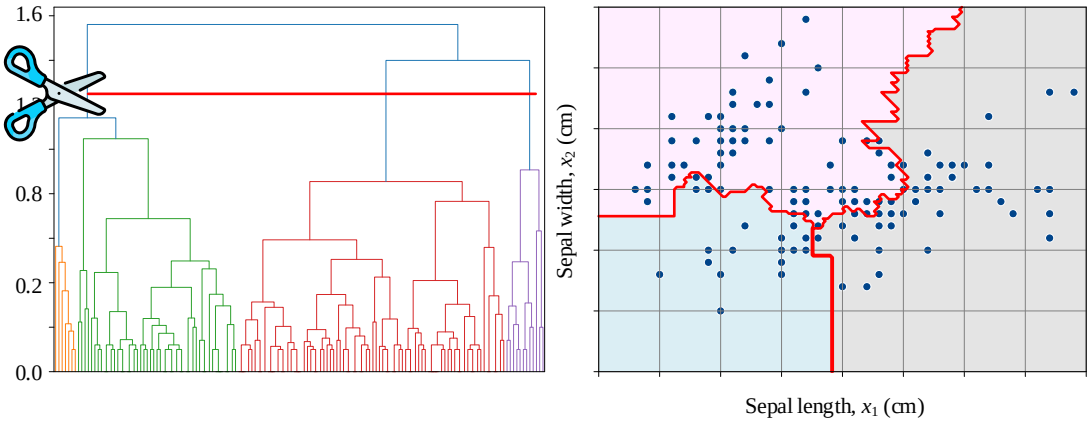


图 17. 鸢尾花聚类结果，'average' 层次聚类

Ward's 簇间距离

Ward's 簇间距离的定义如下：

$$\begin{aligned} d\left(C_k, C_j\right) &= \sqrt{2\times\left(\underbrace{\sum_{\boldsymbol{x}\in C_k\cup C_j}\text{dist}\left(\boldsymbol{x},\boldsymbol{\mu}_{C_k\cup C_j}\right)^2}_{\text{After merge}} - \underbrace{\left(\sum_{\boldsymbol{x}\in C_k}\text{dist}\left(\boldsymbol{x},\boldsymbol{\mu}_{C_k}\right)^2 + \sum_{\boldsymbol{x}\in C_j}\text{dist}\left(\boldsymbol{x},\boldsymbol{\mu}_{C_j}\right)^2\right)}_{\text{Before merge}}\right)} \\ &= \sqrt{\frac{2\cdot\text{count}\left(C_k\right)\cdot\text{count}\left(C_j\right)}{\text{count}\left(C_k\right)+\text{count}\left(C_j\right)}}\cdot\text{dist}\left(\boldsymbol{\mu}_{C_k},\boldsymbol{\mu}_{C_j}\right) \end{aligned} \tag{5}$$

观察 (5)，可以发现它等价于：

$$d\left(C_k, C_j\right) = \sqrt{2\times\left(\underbrace{\text{SST}\left(C_k\cup C_j\right)}_{\text{After merge}} - \underbrace{\left(\text{SST}\left(C_k\right)+\text{SST}\left(C_j\right)\right)}_{\text{Before merge}}\right)} \tag{6}$$

其中，SSE 为丛书前文介绍的**总离差平方和** (Sum of Squares for Total, SST)。

SSE 便是本书前文介绍的“**簇惯性** (cluster inertia)”，也就是：

$$
\begin{cases}
\mathrm{SST}\left(C_k \cup C_j\right) = \sum\limits_{\boldsymbol{x} \in C_k \cup C_j} \mathrm{dist}\left(\boldsymbol{x}, \boldsymbol{\mu}_{C_k \cup C_j}\right)^2 = \sum\limits_{\boldsymbol{x} \in C_k \cup C_j} \left\| \boldsymbol{x} - \boldsymbol{\mu}_{C_k \cup C_j} \right\|^2 \\
\mathrm{SST}\left(C_j\right) = \sum\limits_{\boldsymbol{x} \in C_j} \mathrm{dist}\left(\boldsymbol{x}, \boldsymbol{\mu}_{C_j}\right)^2 = \sum\limits_{\boldsymbol{x} \in C_j} \left\| \boldsymbol{x} - \boldsymbol{\mu}_{C_j} \right\|^2 \\
\mathrm{SST}\left(C_k\right) = \sum\limits_{\boldsymbol{x} \in C_k} \mathrm{dist}\left(\boldsymbol{x}, \boldsymbol{\mu}_{C_k}\right)^2 = \sum\limits_{\boldsymbol{x} \in C_k} \left\| \boldsymbol{x} - \boldsymbol{\mu}_{C_k} \right\|^2
\end{cases} \tag{7}
$$

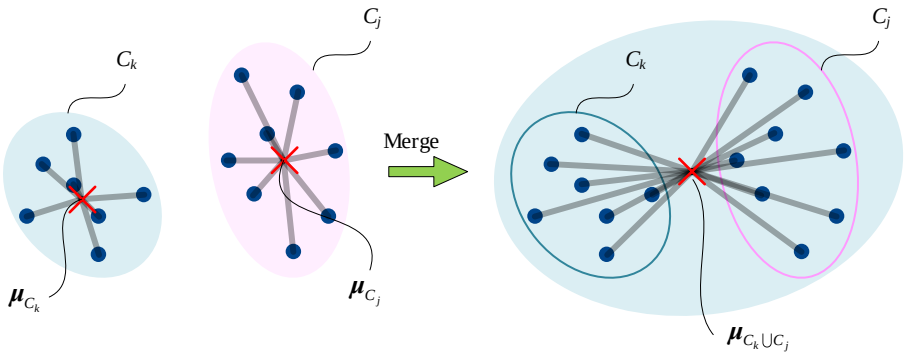


图 18. 鸢尾花聚类结果，Ward’s 簇间距离

Ward’s 簇间距离定义看着复杂，实际上背后的思想很简单——计算**合并后** (after merge)、**合并前** (before merge) 残差平方和 SSE 的差值。原理如图 18 所示。这个差值，也就是一种簇数据“合并”的“代价”。

平均距离代号为 `'ward'`。`'ward'`为 Scikit-learn 默认簇间距离。图 19 所示为，采用 `'ward'` 层次聚类得到的树形图和鸢尾花数据聚类结果。

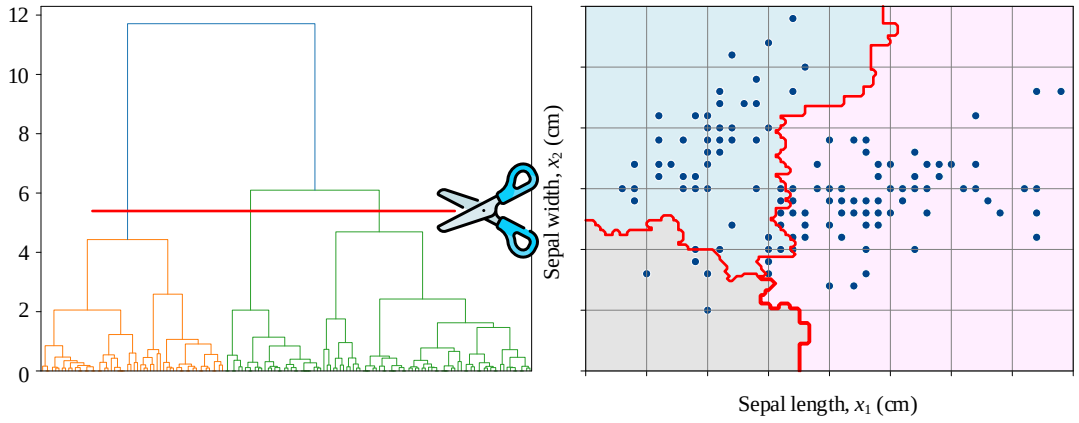
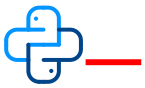


图 19. 鸢尾花聚类结果，'ward' 层次聚类



代码 Bk7_Ch14_01.py 可以绘制图 15、图 16、图 17 和图 19。

15.4 亲近度层次聚类

本章前文介绍的是采用欧氏距离构造树形图，以便进行层次聚类；其实，亲近度也可以用来构造树形图，从而聚类。回顾**高斯核** (Gaussian kernel) 亲近度定义：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma\left\|\boldsymbol{x}-\boldsymbol{q}\right\|^2\right) \tag{8}$$

图 20 左图是鸢尾花数据高斯核亲近度成对矩阵热图。利用 `seaborn.clustermap()` 函数可以绘制基于亲近度矩阵的树形图，以及相应热图，具体如图 20 右图所示。

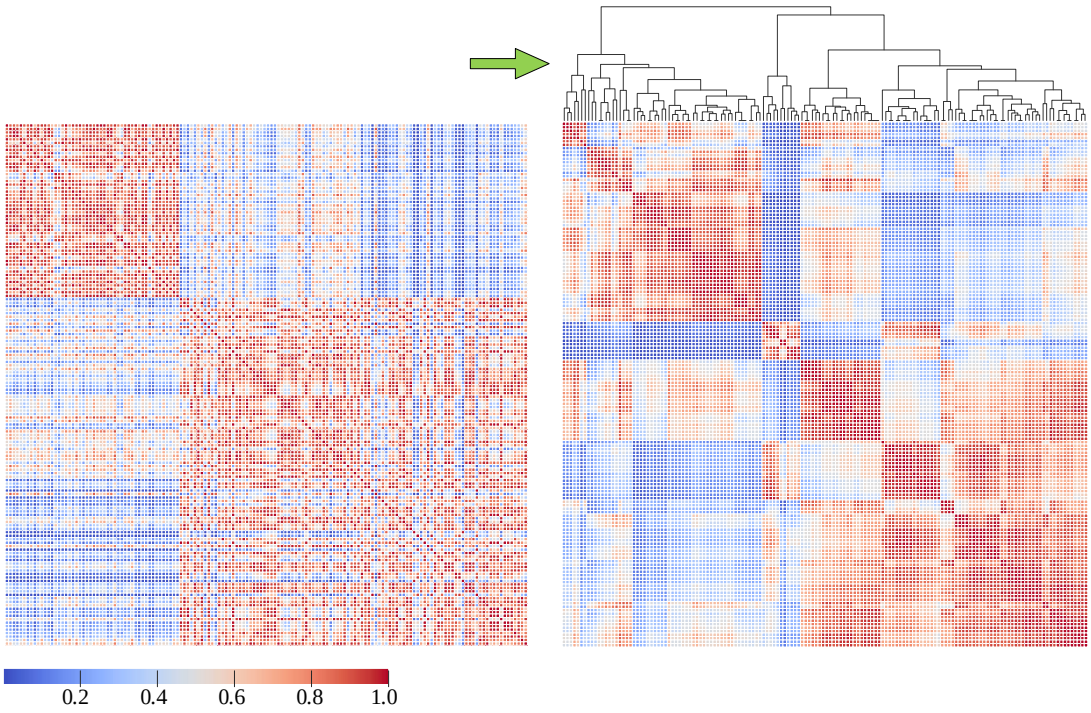
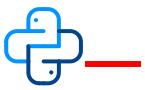
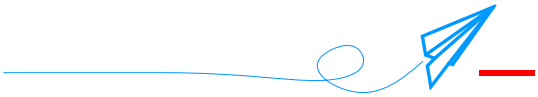


图 20. 鸢尾花花萼两个特征构造的亲近度矩阵，以及树形结构和重排额的亲近度矩阵



代码 Bk7_Ch14_02.py 可以绘制图 20 两幅子图。



层次聚类是一种无需预先指定聚类簇数的聚类方法，其输出结果以树形图的形式呈现。在层次聚类中，不同簇之间的距离可以通过不同的距离度量方法计算，如欧几里得距离、曼哈顿距离等。层次聚类可以分为凝聚聚类和分裂聚类两种方法，其中凝聚聚类是一种从下往上的方法，从每个数据点开始，逐步合并簇，形成树形图；分裂聚类是一种从上往下的方法，将所有数据点放在一个簇中，然后逐步分裂簇，形成树形图。请大家格外注意不同簇间距离定义方式。

在亲近度层次聚类中，距离度量方法可以用相似度度量方法代替，如相关系数、余弦相似度等。亲近度层次聚类通常应用于文本聚类、图像聚类等领域。

16 Density-Based Clustering
密度聚类

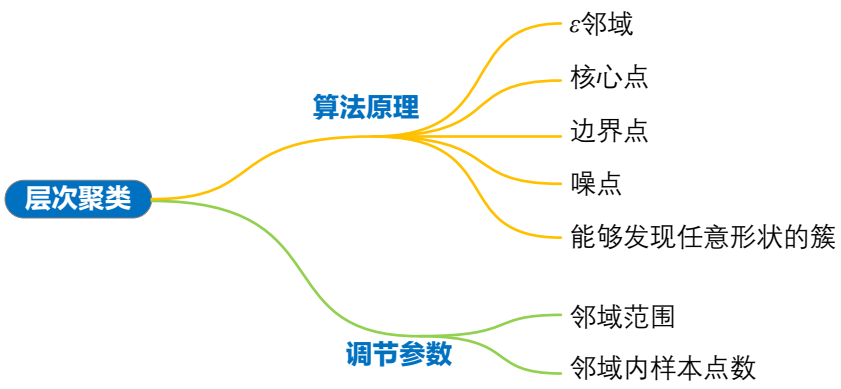
利用数据分布紧密程度聚类

实验是科学向自然提出的问题，测量是对自然回答的记录。

An experiment is a question which science poses to Nature, and a measurement is the recording of Nature's answer.

—— 马克斯·普朗克 (Max Planck) | 德国物理学家，量子力学的创始人 | 1858 ~ 1947

- `itertools.cycle()` 把一组数据循环取出
- `itertools.islice()` 返回一个迭代器
- `numpy.random.seed()` 设置随机数种子可以使每一次生成随机数据的时候结果相同
- `sklearn.cluster.DBSCAN()` DBSCAN 聚类函数
- `sklearn.cluster.OPTICS()` OPTICS 聚类函数
- `sklearn.datasets.make_circles()` 创建环形样本数据
- `sklearn.preprocessing.StandardScaler().fit_transform()` 标准化数据；通过减去均值然后除以标准差，处理后数据符合标准正态分布



16.1 DBSCAN 聚类

密度聚类是一种基于数据点密度的聚类方法，其核心思想是将高密度区域作为聚类中心，并将低密度区域作为聚类边界。常用的密度聚类算法有 DBSCAN、OPTICS、DENCLUE 等。

DBSCAN 通过设定邻域半径和最小密度等参数，将具有足够密度的数据点聚成一个簇；OPTICS 在 DBSCAN 的基础上，通过建立可达距离图来优化聚类结果；DENCLUE 则采用高斯核函数来建模数据点的密度，通过求解梯度的方式来寻找密度峰值，进而进行聚类。密度聚类方法对于数据分布的形态没有特殊要求，对于噪声和离群点的鲁棒性较强，具有广泛的应用价值。

DBSCAN 聚类算法全称为 Density-Based Spatial Clustering of Applications with Noise，它是一种基于密度的聚类方法，是本章要重点介绍的算法。为了方便大家理解 DBSCAN 聚类算法，下面打个比方。

原理

如图 1 所示，限定距离范围内 (即圆圈圈定领域)，粉丝超过一定数量的点就是 UP 主 (头顶皇冠者)；DBSCAN 聚类算法和核心是，如果任意两个 UP 主互粉 (在对方的圆圈范围之内)，则两个 UP 主及各自粉丝可以被划分为一簇。

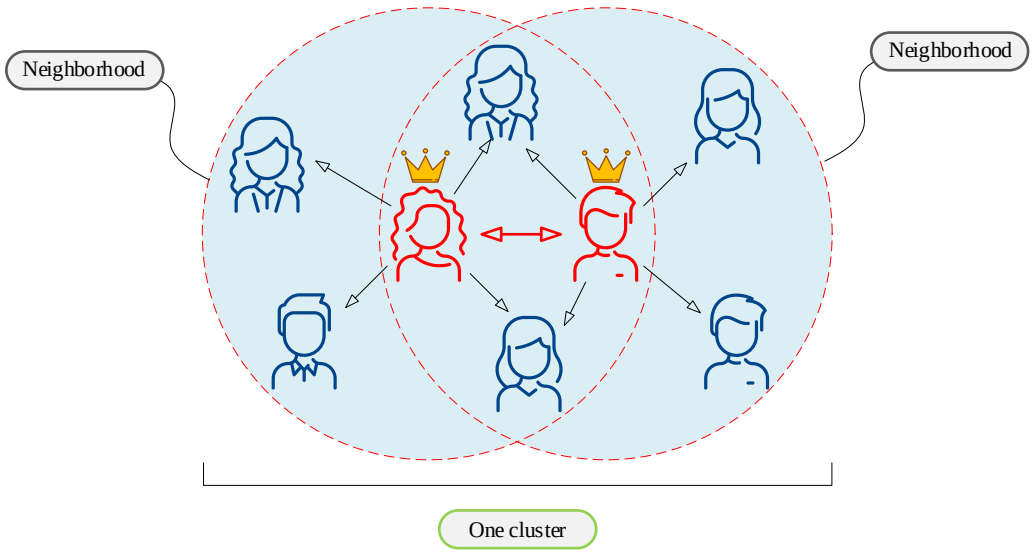


图 1. DBSCAN 算法原理

几个概念

下面介绍 DBSCAN 涉及到的几个概念。

ε 邻域 (ε neighborhood, epsilon neighborhood) `esp` 限定领域范围，`esp` 对应图 1 中的圆圈半径。准确地说，ε 邻域指的是以某样本数据点为中心、`esp` 为半径的区域。

以空间某点为中心，`esp` 为半径邻域内包含至少 `min_samples` 数量的数据点，则称该点为**核心点** (core point)，即前文所说的 UP 主。

核心点 ε 邻域内的点，被称之为**边界点** (border point)。核心点相当于图 1 中 UP 主；边界点，相当于粉丝。特别需要读者注意的是，`min_samples` 为核心点和边界点数量之和。

样本数据点可以是核心点，也可以是边界点，甚至身兼两者角色；如果数据点既不是核心点，也不是边界点，该数据点被称作**噪点** (noise point)，即**离群数据** (outlier)。

聚类

图 2 给出平面内 8 个样本数据点；以每个数据点为中心，ε 为半径扫描整个平面，且定义 `min_samples` = 4。

发现只有样本点 $\boldsymbol{x}^{(5)}$的 ε 邻域内有 4 个样本点 (包括自 $\boldsymbol{x}^{(5)}$身)；因此，$\boldsymbol{x}_5$ 为核心点，$\boldsymbol{x}^{(2)}$、$\boldsymbol{x}^{(4)}$和 $\boldsymbol{x}^{(7)}$为边界点，剩余其他数据点为噪点。

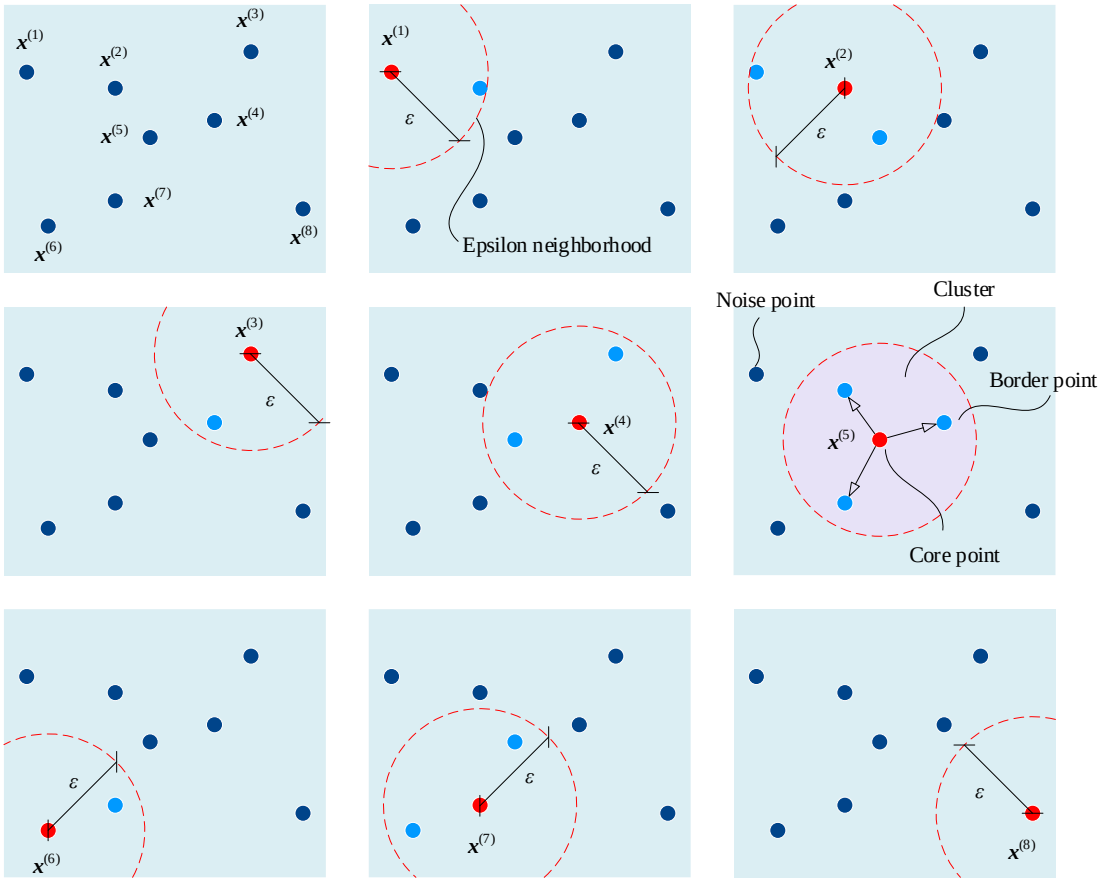


图 2. DBSCAN 算法扫描 8 个样本数据点

如图 3 所示，通过 DBSCAN 算法，空间数据被分为 3 簇。图 3 中，红色数据点为核心点 (即 UP 主)。UP 主的最低要求是在以自己为中心的 ε 邻域内包括含自己在内有 4 名成员；浅蓝色数据点为边界点，深蓝色为噪点。C_1 自成一簇；三个 UP 主互粉，三个 ε 邻域相互连接，构成 C_2；两个 UP 主互粉，两个 ε 邻域相互连接，构造图 3 中所示 C_3。

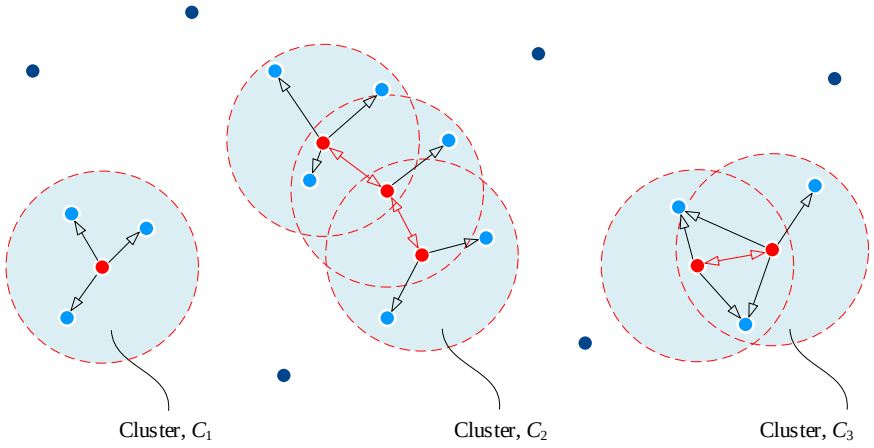


图 3. 通过 DBSCAN 算法，数据被分为 3 簇

16.2 调节参数

邻域范围

`eps` 控制邻域范围大小。`eps` 值选取过大，会导致整个数据集被分为一簇；但是 eps 取值过小，会导致簇过多且分散，并且标记过多噪音点。

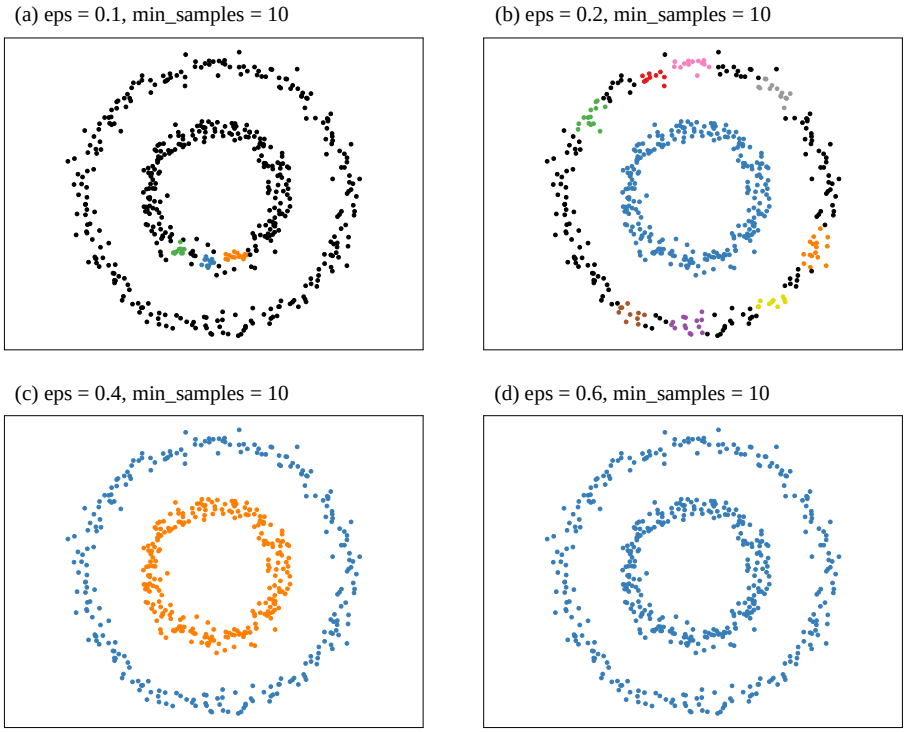


图 4. eps 对聚类结果影响

图 4 (a) 所示，当 `eps` = 0.1 时，环形样本数据多数被标记为噪音 (黑色点)。当 `eps` 增大到 0.2 时，被标记为噪音点减少，且小环被划分为一簇 (蓝色点)，如图 4 (b) 所示。当 `eps` = 0.4 时，环形样本数据被正确地分类为两簇，如图 4 (c) 所示。当 `eps` 增大到 0.6 时，所有样本数据被划分为一簇。请读者注意，ε 邻域半径 `eps` 这个距离，未必是欧氏距离。

邻域内样本点数

`min_samples` 调节 DBSCAN 算法对噪声的容忍度；当数据噪音过大时，应该适当提高 `min_samples`。

k 均值和 GMM 聚类算法需要预先声明聚类数量；但是，DBSCAN 则不需要。DBSCAN 聚类不需要预设分布类型，不受数据分布影响，且可以分辨离群数据。但是，DBSCAN 算法对 `eps` 和 `min_samples` 这两个初始参数都很敏感；协同调节 `eps` 和 `min_samples` 两个参数显得非常重要。

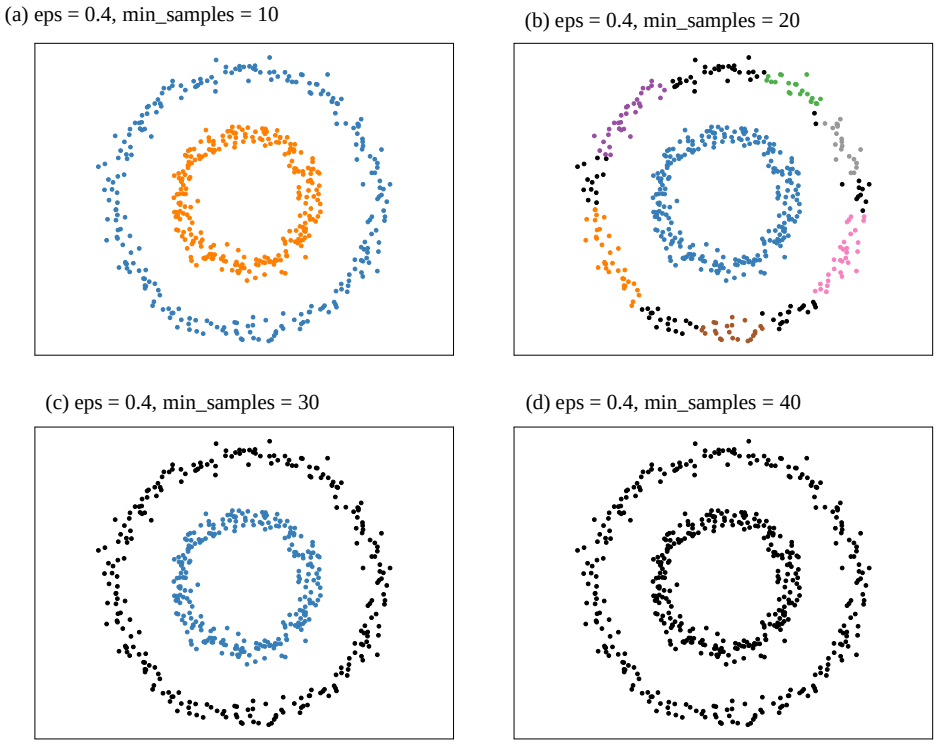


图 5. min_samples 对聚类结果影响

代码 Bk7_Ch15_01.py 绘制图 4、图 5。

DBSCAN 是一种基于密度的聚类算法，其特点是可以自动识别出任意形状的簇，并将离群点视为噪声数据。DBSCAN 将密度定义为在给定半径内的数据点数量，利用这一度量将数据点分为核心点、边界点和噪声点三类。

在聚类过程中，DBSCAN 通过不断扩展核心点的密度直到达到最大密度，将核心点和边界点划分到同一簇中。优点是不需要事先设定聚类数量，鲁棒性强，可以处理不同形状、大小和密度的簇。缺点是对于密度分布较为均匀的数据集，可能出现聚类失效的情况。

OPTICS (Ordering Points To Identify the Clustering Structure) 聚类算法和 DSCAN 非常相似。不同的是，需要用户输入 `eps` 和 `min_samples` 两个参数；而 OPTICS 虽然也需要输入这两个参数，但是对 `eps` 不敏感。请读者自行学习下例。

https://scikit-learn.org/stable/auto_examples/cluster/plot_optics.html

17 Spectral Clustering
谱聚类

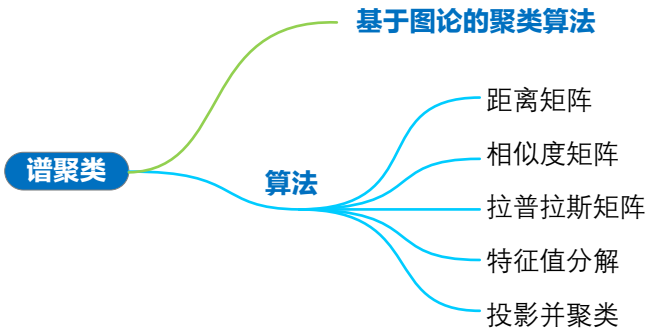
构造无向图，距离远的两点，权重值低；降维聚类

生命中最重要的问题，几乎都是概率问题。

The most important questions of life are indeed, for the most part, really only problems of probability.

—— 皮埃尔-西蒙·拉普拉斯 (Pierre-Simon Laplace) | 法国著名天文学家和数学家 | 1749 ~ 1827

- `sklearn.cluster.SpectralClustering()` 谱聚类算法
- `sklearn.datasets.make_circles()` 创建环形样本数据
- `sklearn.preprocessing.StandardScaler().fit_transform()` 标准化数据；通过减去均值然后除以标准差，处理后数据符合标准正态分布



17.1 谱聚类

谱聚类 (spectral clustering) 是一种基于图论的聚类算法，其特点是能够处理高维数据和非凸数据簇，并且对于数据分布的形态没有特殊要求。优点是可以在任意维度上进行聚类，并且不会受到噪声的影响。缺点是需要进行谱分解计算，计算量较大。

具体来说，谱聚类的思路是将样本数据看做是空间**节点** (node)，这些节点之间用**边** (edge) 连构成的**无向图** (undirected graph)，也叫**加权图**。无向图中，距离远的数据点，边的权重值低；距离近的数据点，在无向图中，边的权重值高。

用无向图聚类的过程很简单，切断无向图中权重值低的边，得到一系列子图。子图内部节点之间边的权重尽可能高，子图之间边权重尽可能低。将节点之间的相似度构成的矩阵称为邻接矩阵，通过对邻接矩阵进行谱分解，得到数据点的特征向量，进而将其映射到低维空间进行聚类。

流程

这个思路虽然简单，但是实际操作需要一系列矩阵运算。

首先，需要计算数据 $\boldsymbol{X}$ 之间的两两距离，并构造成距离矩阵 $\boldsymbol{D}$。然后，将距离转换成权重值，即**相似度** (similarity)，构造**相似度矩阵** (similarity matrix) $\boldsymbol{S}$，利用 $\boldsymbol{S}$ 可以绘制无向图。

之后，将相似度矩阵转化成**拉普拉斯矩阵** (Laplacian matrix) $\boldsymbol{L}$。最后，**特征值分解** (eigen decomposition) $\boldsymbol{L}$，相当于将 $\boldsymbol{L}$ 投影在一个低维度正交空间。在这个低维度空间中，用简单聚类方法对投影数据进行聚类，并得到原始数据聚类。

下面通过实例，我们一一讨论谱聚类这些步骤所涉及的技术细节。

17.2 距离矩阵

图 1 给出 12 个样本点在平面上位置。计算数据**两两距离** (pairwise distance)，$\boldsymbol{x}^{(i)}$和 $\boldsymbol{x}^{(j)}$两个点之间欧氏距离 $d_{i,j}$:

$$d_{i,j} = \sqrt{\left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right)^{\mathrm{T}} \left(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right)} = \left\| \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)} \right\| \tag{1}$$

其中，约定 $\boldsymbol{x}^{(i)}$ 和 $\boldsymbol{x}^{(j)}$ 均为列向量。

图 2 所示为热图描绘的 12 个样本点两两欧氏距离构造的对称矩阵 $\boldsymbol{D}$；注意，$\boldsymbol{D}$ 的对角线元素均为 0，这是因为观察点和自身之间距离为 0。色块颜色越浅，说明距离越近；色块颜色越深，说明距离越远。

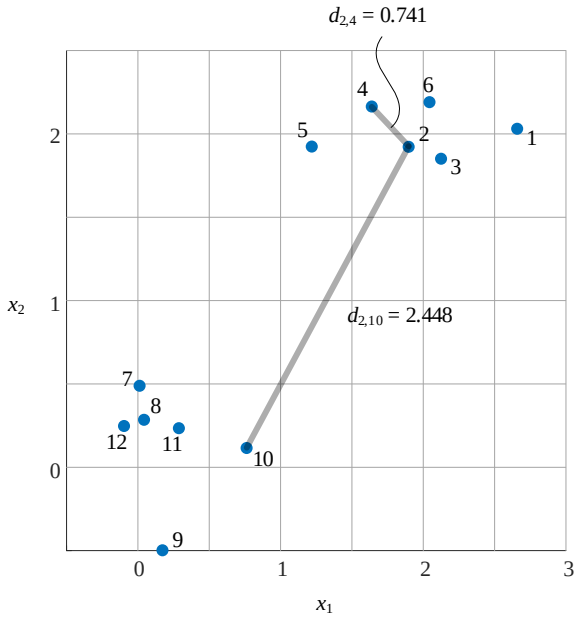


图 1. 12 个样本点平面位置

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0.3845	0.8692	0.7926	0.1449	0.2649	2.891	2.131	2.586	2.74	2.605	3.282
2	0.3845	0	0.4894	0.7412	0.2494	0.3749	2.629	1.929	2.356	2.448	2.333	2.975
3	0.8692	0.4894	0	0.851	0.7387	0.8378	2.439	1.865	2.228	2.207	2.134	2.695
4	0.7926	0.7412	0.851	0	0.7935	0.9998	3.276	2.638	3.039	3.054	2.972	3.546
5	0.1449	0.2494	0.7387	0.7935	0	0.2063	2.763	2.017	2.466	2.605	2.474	3.144
6	0.2649	0.3749	0.8378	0.9998	0.2063	0	2.65	1.875	2.335	2.514	2.369	3.063
7	2.891	2.629	2.439	3.276	2.763	2.65	0	0.867	0.4055	0.3493	0.3054	0.6349
8	2.131	1.929	1.865	2.638	2.017	1.875	0.867	0	0.4841	0.9092	0.6789	1.443
9	2.586	2.356	2.228	3.039	2.466	2.335	0.4055	0.4841	0	0.5782	0.354	1.028
10	2.74	2.448	2.207	3.054	2.605	2.514	0.3493	0.9092	0.5782	0	0.2384	0.5628
11	2.605	2.333	2.134	2.972	2.474	2.369	0.3054	0.6789	0.354	0.2384	0	0.7683
12	3.282	2.975	2.695	3.546	3.144	3.063	0.6349	1.443	1.028	0.5628	0.7683	0

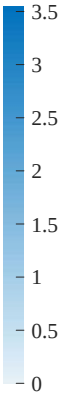


图 2. 12 个样本点两两欧氏距离构造的成对距离矩阵 ***D***

17.3 相似度

然后利用 $d_{i,j}$ 计算 i 和 j 两点的相似度 $s_{i,j}$，“距离 → 相似度”的转换采用高斯核函数：

$$s_{i,j} = \exp\left(-\left(\frac{d_{i,j}}{\sigma}\right)^2\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|^2}{\sigma^2}\right) \tag{2}$$

相似度取值区间为 (0, 1]。$\boldsymbol{x}^{(i)}$ 和 $\boldsymbol{x}^{(j)}$两个点距离越近，它们的相似性越高。任意点和自身的距离为 0，因此对应的相似度最大为 1。

$\sigma = 1$ 时，两两距离 $d_{i,j}$和相似度 $s_{i,j}$两者关系如图 3 所示。

图 1 中，点 $\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(10)}$之间欧氏距离为 $d_{2,10} = 2.448$，点 $\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(4)}$之间欧氏距离为 $d_{2,4} = 0.741$。利用上式，可以计算得到，点 $\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(10)}$之间相似度 $s_{2,10} = 0.0025$，点 $\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(4)}$之间欧氏距离为 $s_{2,4} = 0.577$。

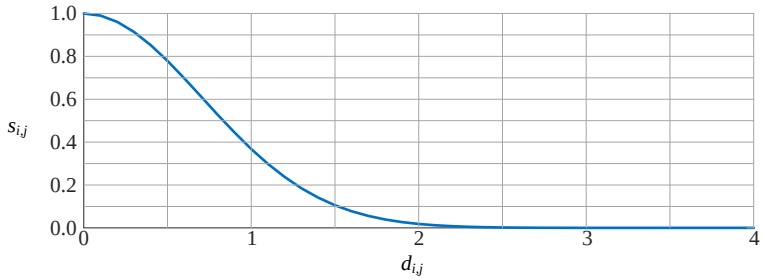


图 3. 欧氏距离和相似度关系

图 2 所示成对距离矩阵可以转化为图 4 所示**相似度矩阵** (similarity matrix) $\boldsymbol{S}$。$\boldsymbol{S}$ 也叫**邻接矩阵** (adjacency matrix)。相似度矩阵 $\boldsymbol{S}$ 的每个元素均大于 0。请大家注意，一些教材将两两距离矩阵 $\boldsymbol{D}$ 叫做相似度矩阵。从图 4 一眼就可以看出数据可以划分为两簇。

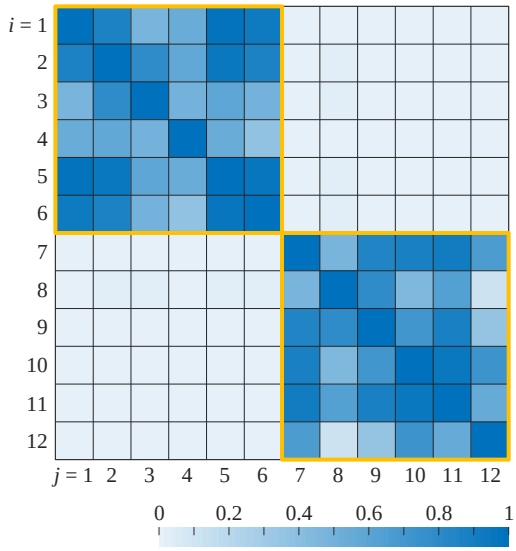


图 4. 12 个样本点两两相似度矩阵 $\boldsymbol{S}$

17.4 无向图

图 5 为相似度矩阵 $\boldsymbol{S}$ 无向图。图中绿色线越粗，表明两点之间的相似度越高，也就是两点距离越近。

切断相似度小于 0.001 两两元素之间的联系得到无向图图 6。图 7 为，切断相似度小于 0.005 两两元素之间的联系得到无向图。观察图 8 可以知道，当切断相似度小于 0.031 两两元素之间的联系，可以将原始数据划分为两簇。

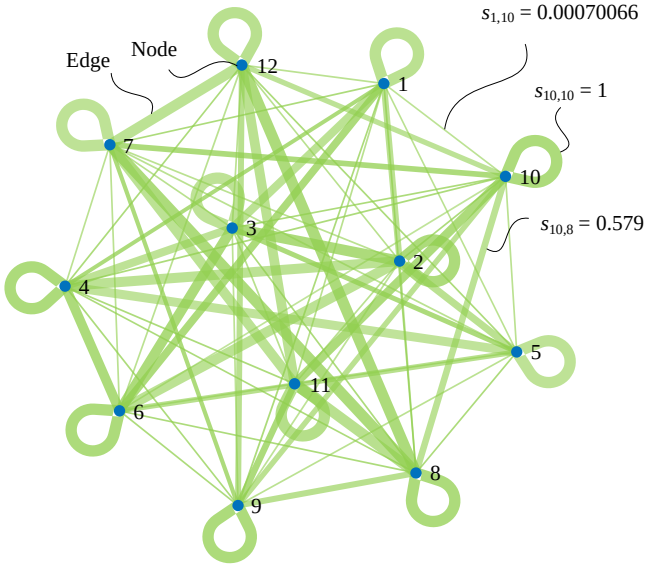


图 5. 相似度对称矩阵 S 无向图

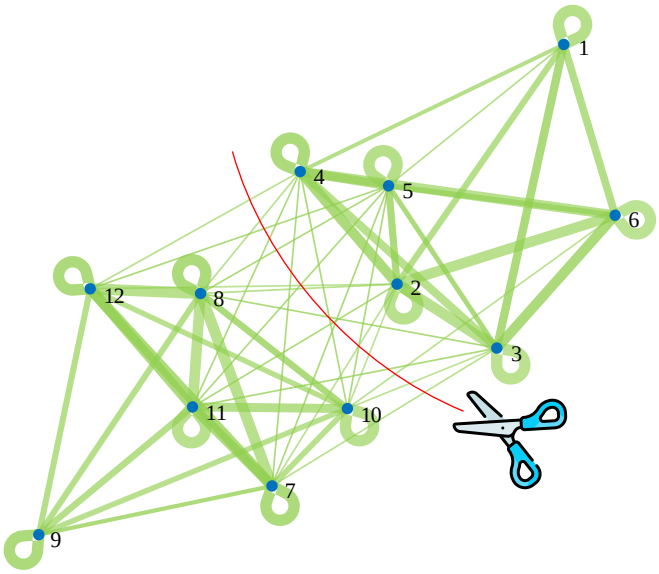


图 6. 当切断相似度小于 0.001 两两元素之间的联系得到无向图

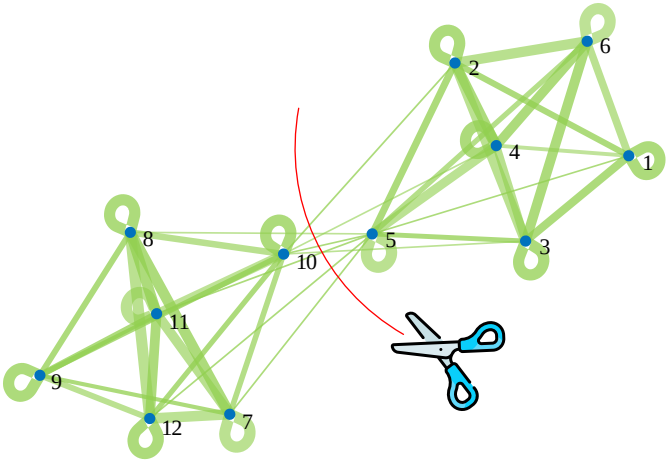


图 7. 当切断相似度小于 0.005 两两元素之间的联系得到无向图

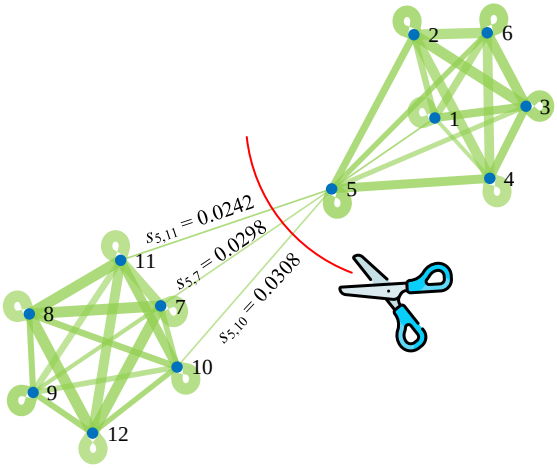
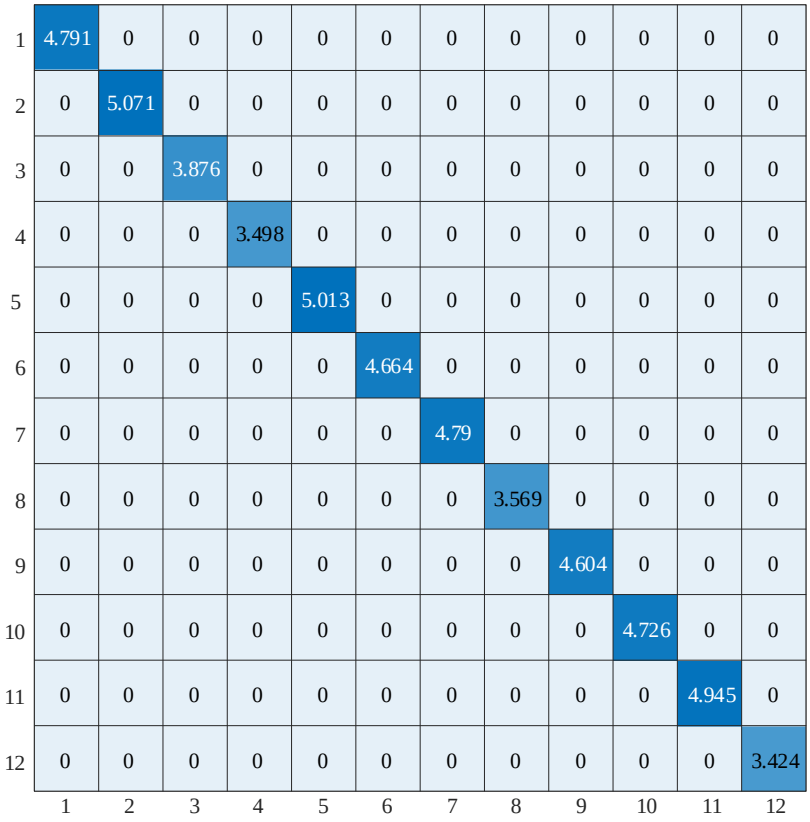


图 8. 当切断相似度小于 0.02 两两元素之间的联系得到无向图

17.5 拉普拉斯矩阵

度矩阵 (degree matrix) $\boldsymbol{G}$ 是一个对角阵。$\boldsymbol{G}$ 的对角线元素是对应相似度矩阵 $\boldsymbol{S}$ 对应列元素之和，即：

$$\boldsymbol{G}_{i,i} = \sum_{j=1}^{n} s_{i,j} = \operatorname{diag}\left(\boldsymbol{1}^{\mathrm{T}} \boldsymbol{S}\right) \tag{3}$$



	1	2	3	4	5	6	7	8	9	10	11	12
1	4.791	0	0	0	0	0	0	0	0	0	0	0
2	0	5.071	0	0	0	0	0	0	0	0	0	0
3	0	0	3.876	0	0	0	0	0	0	0	0	0
4	0	0	0	3.498	0	0	0	0	0	0	0	0
5	0	0	0	0	5.013	0	0	0	0	0	0	0
6	0	0	0	0	0	4.664	0	0	0	0	0	0
7	0	0	0	0	0	0	4.79	0	0	0	0	0
8	0	0	0	0	0	0	0	3.569	0	0	0	0
9	0	0	0	0	0	0	0	0	4.604	0	0	0
10	0	0	0	0	0	0	0	0	0	4.726	0	0
11	0	0	0	0	0	0	0	0	0	0	4.945	0
12	0	0	0	0	0	0	0	0	0	0	0	3.424

图 9. 12 个样本点两两相似度构造的度矩 $\boldsymbol{G}$

拉普拉斯矩阵

然后构造**拉普拉斯矩阵** (Laplacian matrix) $\boldsymbol{L}$。有三种方法构造拉普拉斯矩阵。

第一种叫做**未归一化拉普拉斯矩阵** (unnormalized Laplacian matrix)，具体定义如下：

$$\boldsymbol{L} = \boldsymbol{G} - \boldsymbol{S} \tag{4}$$

第二种叫做**归一化随机漫步拉普拉斯矩阵** (normalized random-walk Laplacian matrix)，也叫 Shi-Malik 矩阵，定义如下：

$$\boldsymbol{L}_{rw} = \boldsymbol{G}^{-1}\left(\boldsymbol{G} - \boldsymbol{S}\right) \tag{5}$$

第三种叫做**归一化对称拉普拉斯矩阵** (normalized symmetric Laplacian matrix)，也叫做 Ng-Jordan-Weiss 矩阵，如下：

$$\boldsymbol{L}_{s} = \boldsymbol{G}^{-1/2}\left(\boldsymbol{G} - \boldsymbol{S}\right)\boldsymbol{G}^{-1/2} \tag{6}$$

采用第一种方法获得拉普拉斯矩阵 $\boldsymbol{L}$，热图如图 10 所示。

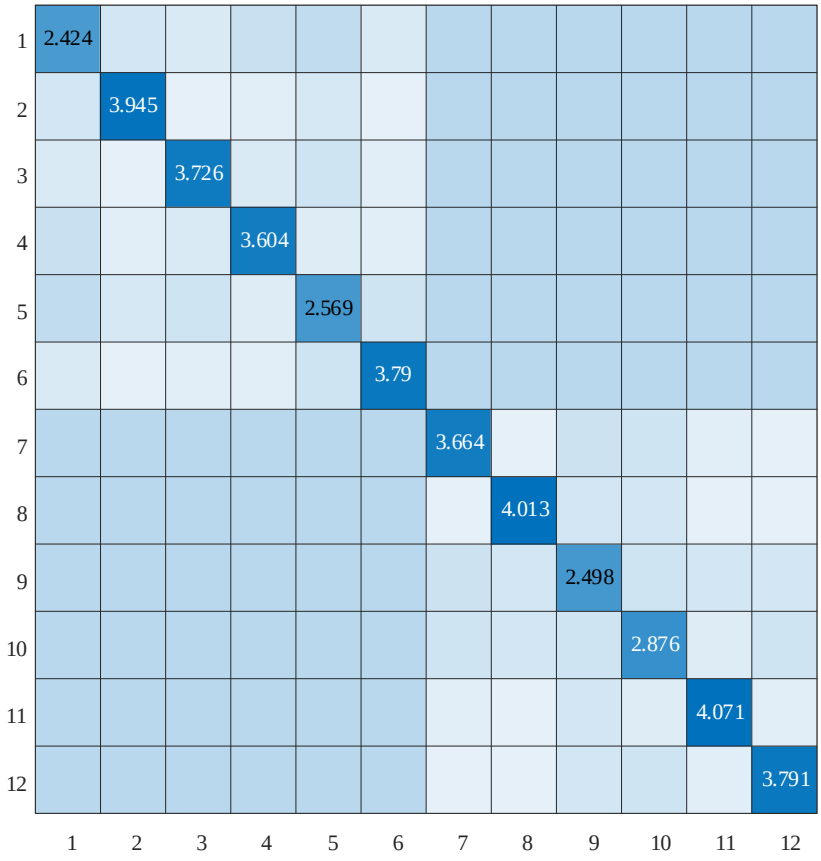


图 10. 12 个样本点两两相似度构造未归一化拉普拉斯矩阵 $\boldsymbol{L}$

17.6 特征值分解

对拉普拉斯矩阵 $\boldsymbol{L}$ 进行特征值分解：

$$\boldsymbol{L} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1} \tag{7}$$

其中

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{12} \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \ldots & \boldsymbol{v}_1 \end{bmatrix} \tag{8}$$

图 11 所示为拉普拉斯矩阵 $\boldsymbol{L}$ 特征值分解得到的特征值从小到大排序。按从小到大排列 λ 值，对应第 2 个，$\lambda_2 = 0.01285$，对应的特征向量 $\boldsymbol{v}_2 = [-0.300, -0.295, -0.297, -0.294, -0.275, -0.298, 0.283, 0.285, 0.288, 0.278, 0.284, 0.286]$。

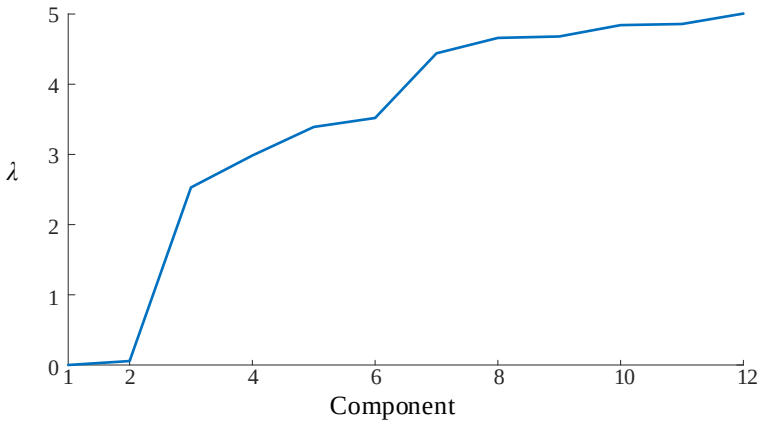


图 11. 拉普拉斯矩阵 $\boldsymbol{L}$ 特征值分解得到的特征值从小到大排序

图 12 和图 13 分别展示前两个特征向量的结果。相当于将拉普拉斯矩阵 $\boldsymbol{L}$ 投影到一个二维空间，具体如图 14 所示。在图 14 所示平面内，可以很容易将数据划分为两簇。

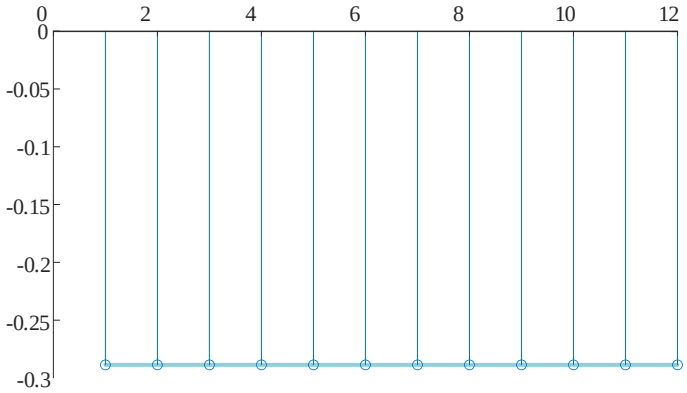


图 12. 特征向量 $\boldsymbol{v}_1$ 结果

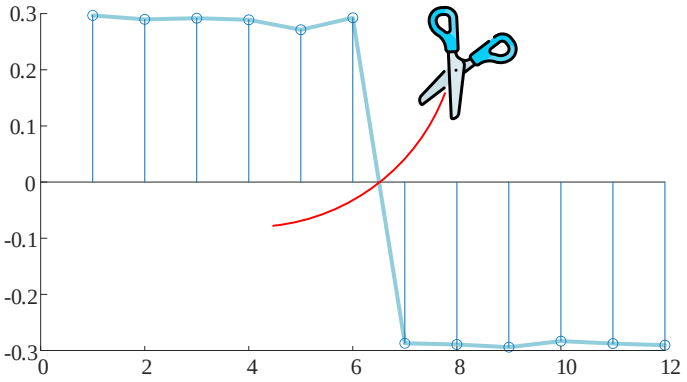


图 13. 特征向量 $\boldsymbol{v}_2$ 结果

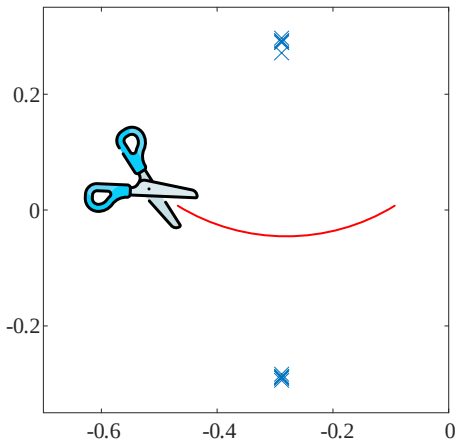


图 14. 矩阵 $\boldsymbol{L}$ 投影到低维度正交空间结果

图 15 所示为采用谱聚类算法对环形样本数据聚类结果。谱聚类的可调节参数包括：相似度矩阵可以使用不同的相似度度量方式。拉普拉斯矩阵可以采用不同类型。特征向量数量可以影响聚类效果。最终的聚类可以选择不同算法。

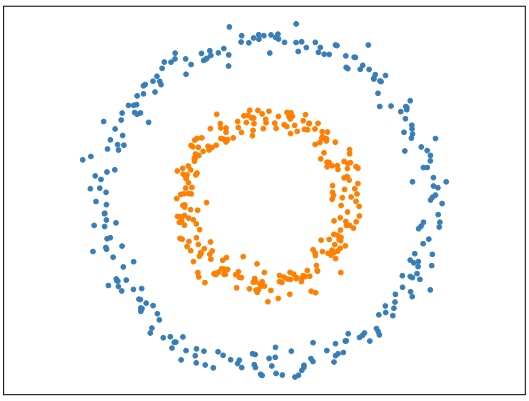
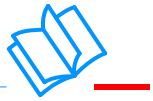


图 15. 环形样本数据聚类结果

代码 Bk7_Ch16_01.py 可以获得图 15。

谱聚类是一种基于图论的聚类算法，其特点是能够处理高维数据和非凸数据簇，并且对于数据分布的形态没有特殊要求。谱聚类通过将数据点看作图中的节点，将它们之间的相似度构成的矩阵称为邻接矩阵，通过对邻接矩阵进行谱分解，得到数据点的特征向量，进而将其映射到低维

空间进行聚类。优点是可以在任意维度上进行聚类，并且不会受到噪声的影响。缺点是需要进行谱分解计算，计算量较大。



请大家注意，拉普拉斯矩阵 $\boldsymbol{L}$ 为**半正定矩阵** (positive semi-definite matrix)。证明过程请参考 Ulrike von Luxburg 创作的 *A Tutorial on Spectral Clustering*。

18 降维

Dimensionality Reduction

鸢尾花书有关降维算法模型的综述

人类的历史，本质上是思想的历史。

Human history is, in essence, a history of ideas.

—— 赫伯特·乔治·威尔斯 (Herbert George Wells) | 英国小说家和历史学家 | 1866 ~ 1946

- `sklearn.decomposition.PCA()` 主成分分析函数
- `sklearn.decomposition.TruncatedSVD()` 截断奇异值分解
- `sklearn.decomposition.FastICA()` 独立成分分析
- `sklearn.decomposition.IncrementalPCA()` 增量主成分分析

18.1 一张“降维”版图

机器学习中的降维 (dimensionality reduction) 是指将高维数据转换为低维数据的过程，即将包含大量特征的数据集通过某种方式转化为特征较少但仍能保留原有信息的数据集。降维的目的是减少特征数量，降低计算复杂度，并提高模型的准确性和泛化能力。降维算法在尽可能地保留数据的重要信息的同时，将高维数据映射到低维空间中。图 1 总结几种常见降维的算法。

相信大家对下面这几种算法已经熟悉：主成分分析 (Principal Component Analysis, PCA)、典型相关分析 (Canonical Correlation Analysis, CCA)。这幅图也是本章的思维导图。

本书前文介绍的线性判别分析 (Linear Discriminant Analysis, LDA) 也可以视作一种降维方法。本章还要简单介绍核主成分分析 (Kernel Principal Component Analysis, KPCA)、独立成分分析 (Independent Component Analysis)、流形学习 (Manifold Learning) 这几种方法。

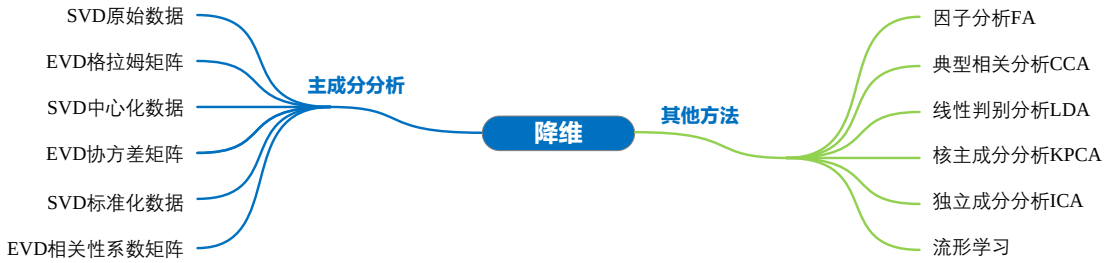


图 1. 降维方法分类

主成分分析

鸢尾花书对主成分分析着墨颇多。主成分分析是一种常用的数据降维方法，通过将高维数据投影到低维空间中，尽可能保留原始数据的重要信息。PCA 将原始数据的特征转换为新的特征，这些新特征按照重要性递减排列。通过选取前面的几个主成分，可以实现对数据的压缩和可视化。主成分分析常用于数据预处理、数据可视化和特征提取等领域。它能够剔除冗余的特征信息，简化数据模型，提高模型的效率和准确性，是机器学习中非常重要的技术之一。

和 OLS 线性回归类似，主成分分析也可以从几何 (图 2)、投影、数据、线性组合、特征值分解、SVD 分解、优化、概率统计等视角来理解。

《数据有道》第 18、19 两章还介绍利用主成分分析进行回归的两种方法：正交回归、主元回归。

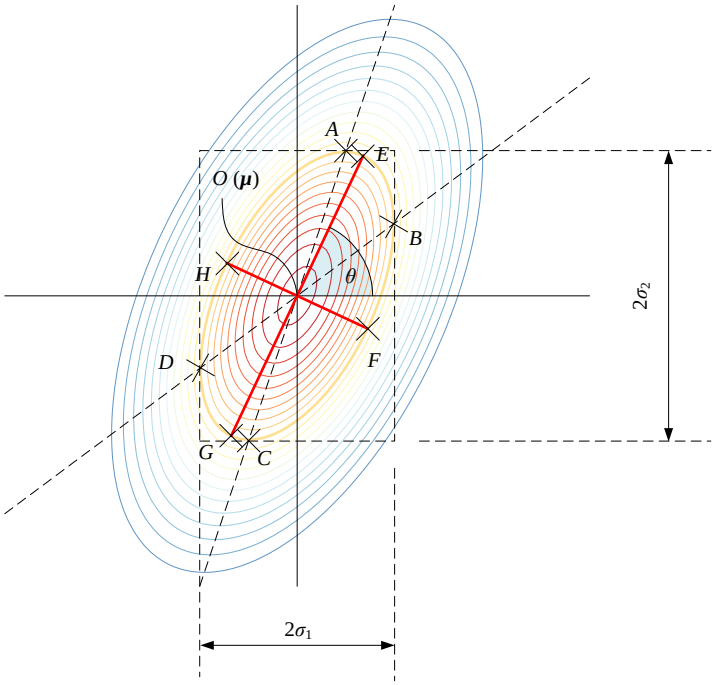


图 2. 主成分分析和椭圆的关系，图片来自《统计至简》第 25 章

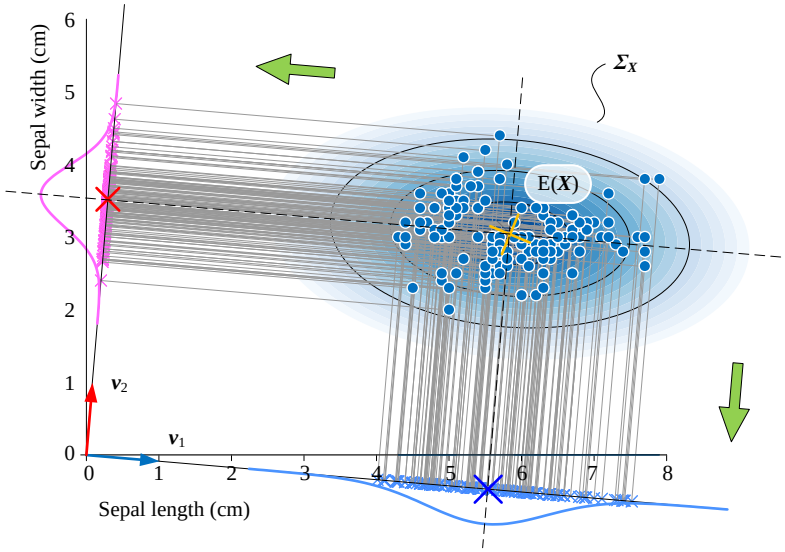


图 3. 投影视角看 PCA，图片来自《统计至简》第 14 章

此外，《数据有道》第 17 章还专门比较过主成分分析的六条技术路线，如表 1 所示。

表 1. 六条 PCA 技术路线，来自《矩阵分解》第 25 章

对象	方法	结果

原始数据矩阵 $\boldsymbol{X}$	奇异值分解	$\boldsymbol{X} = \boldsymbol{U_X S_X V_X}^{\mathrm{T}}$
格拉姆矩阵 $\boldsymbol{G} = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}$ 本章中用“修正”的格拉姆矩阵 $\boldsymbol{G} = \frac{\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}}{n-1}$	特征值分解	$\boldsymbol{G} = \boldsymbol{V_X \Lambda_X V_X}^{\mathrm{T}}$
中心化数据矩阵 $\boldsymbol{X}_c = \boldsymbol{X} - \mathrm{E}(\boldsymbol{X})$	奇异值分解	$\boldsymbol{X}_c = \boldsymbol{U_c S_c V_c}^{\mathrm{T}}$
协方差矩阵 $\boldsymbol{\Sigma} = \frac{(\boldsymbol{X} - \mathrm{E}(\boldsymbol{X}))^{\mathrm{T}}(\boldsymbol{X} - \mathrm{E}(\boldsymbol{X}))}{n-1}$	特征值分解	$\boldsymbol{\Sigma} = \boldsymbol{V_c \Lambda_c V_c}^{\mathrm{T}}$
标准化数据 (z 分数) $\boldsymbol{Z}_{\boldsymbol{X}} = (\boldsymbol{X} - \mathrm{E}(\boldsymbol{X}))\boldsymbol{D}^{-1}$ $\boldsymbol{D} = \mathrm{diag}(\mathrm{diag}(\boldsymbol{\Sigma}))^{\frac{1}{2}}$	奇异值分解	$\boldsymbol{Z_X} = \boldsymbol{U_Z S_Z V_Z}^{\mathrm{T}}$
相关性系数矩阵 $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{\Sigma}\boldsymbol{D}^{-1}$ $\boldsymbol{D} = \mathrm{diag}(\mathrm{diag}(\boldsymbol{\Sigma}))^{\frac{1}{2}}$	特征值分解	$\boldsymbol{P} = \boldsymbol{V_Z \Lambda_Z V_Z}^{\mathrm{T}}$

奇异值分解

表 1 中前两种 PCA 方法，又叫截断奇异值 (truncated SVD)。

sklearn.decomposition.TruncatedSVD() 这个函数支持这两种技术路线。

《矩阵力量》第 16 章介绍了四种奇异值分解，图 4 ~ 图 7 展示了它们之间的关系。此外，请大家回顾《矩阵力量》第 6 章有关分块矩阵乘法相关内容。

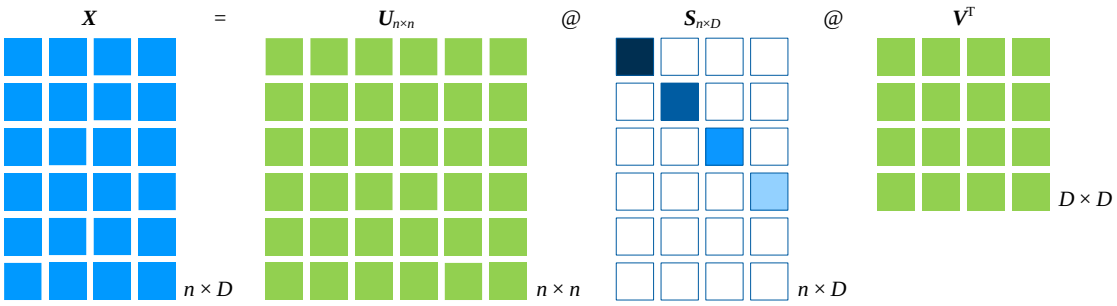


图 4. 完全型 SVD 分解

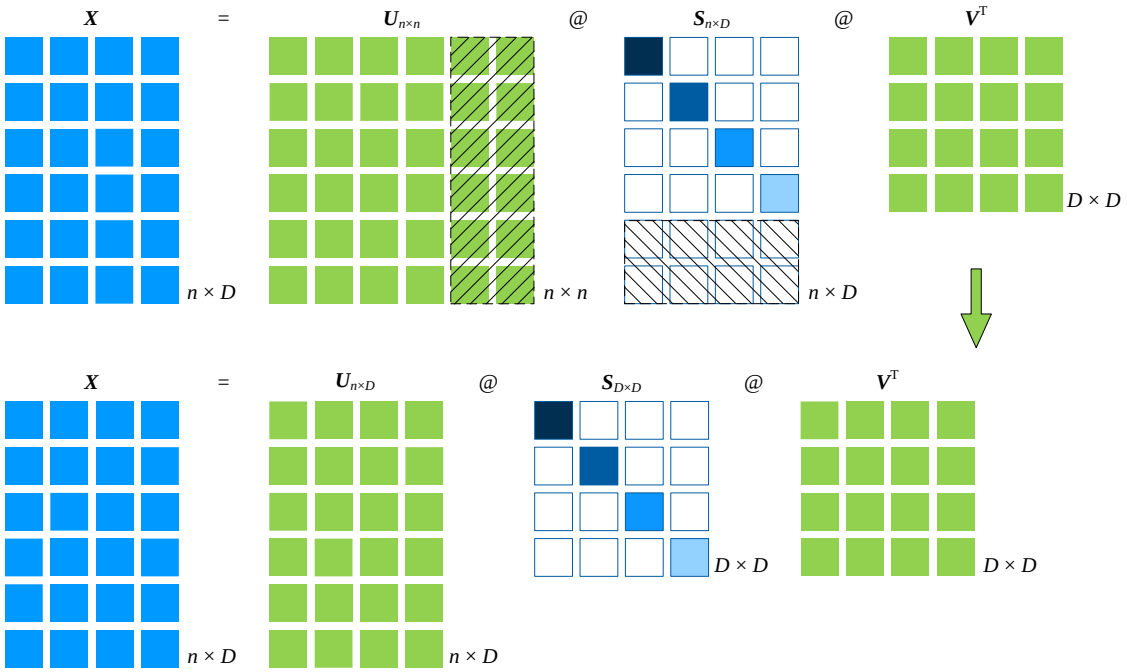


图 5. 从完全型到经济型

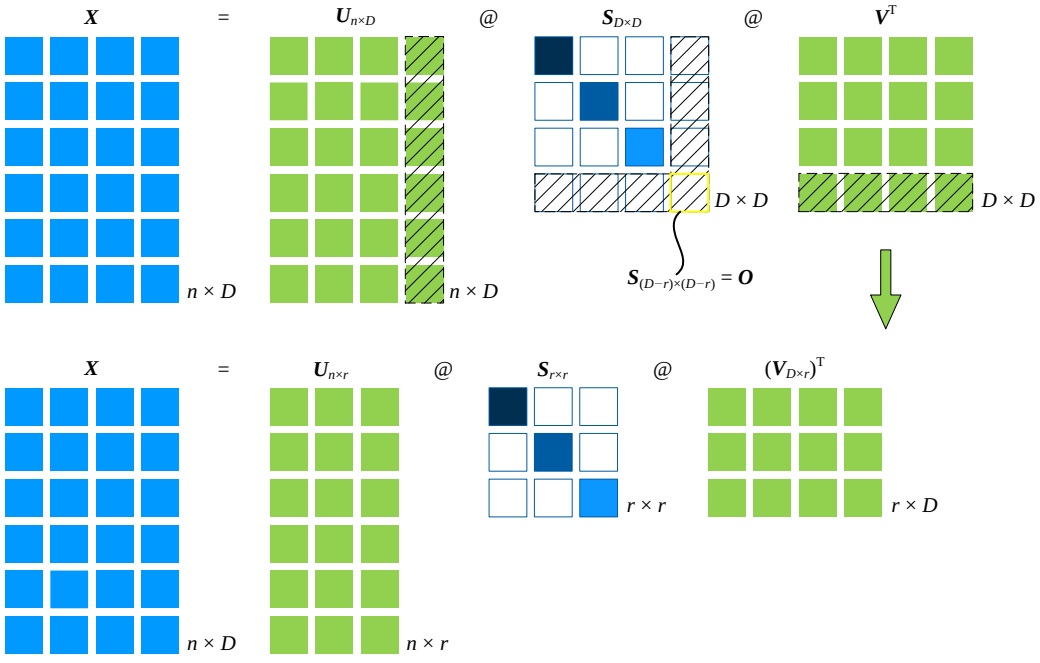


图 6. 从经济型到紧缩型

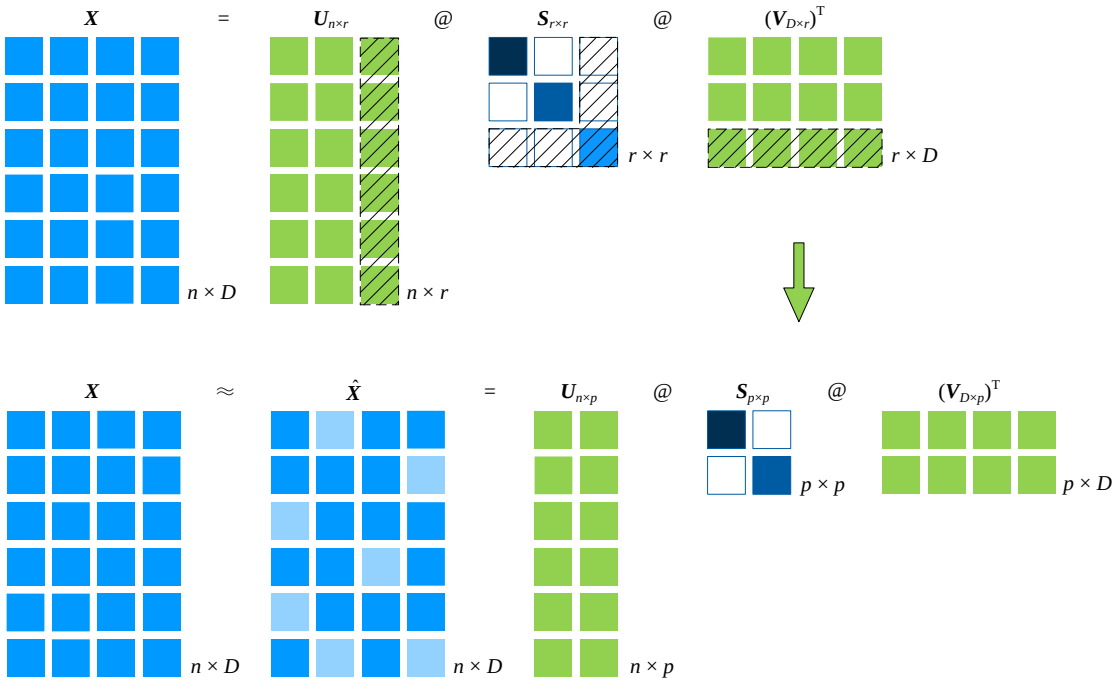


图 7. 从紧缩型到截断型

增量 PCA

当 PCA 需要处理的数据矩阵过大，以至于内存无法支持，可以使用增量主成分分析 (Incremental PCA, IPCA) 替代主成分分析。IPCA 分批处理输入数据，以便节省内存使用。Scikit-learn 中专门做增量 PCA 的函数为 sklearn.decomposition.IncrementalPCA()。

有关增量 PCA，大家可以参考下例：

https://scikit-learn.org/stable/auto_examples/decomposition/plot_incremental_pca.html

典型相关分析 CCA

典型相关分析也可以视作一种降维算法。典型相关分析是一种用于探究两组变量之间相关关系的统计方法，通常用于多个变量之间的关系分析。典型相关分析可以找出两组变量中最相关的线性组合，从而找到它们之间的相关性。典型相关分析的目的是提取出两组变量之间的共性信息，用于预测和解释数据。CCA 也可以从几何、数据、优化、线性组合、统计几个不同视角来理解。

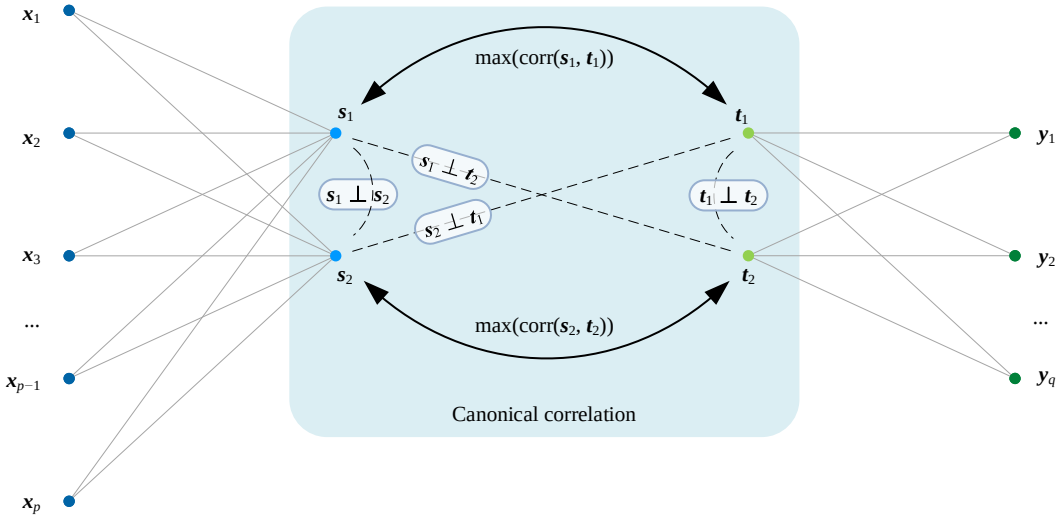


图 8. 线性组合角度看 CCA，图片来自《数据有道》第 20 章

下面这个例子比较偏最小二乘法 PLS、CCA，请大家参考：

https://scikit-learn.org/stable/auto_examples/cross_decomposition/plot_compare_cross_decomposition.html

18.2 核主成分分析

核主成分分析 (Kernel PCA) 是一种非线性的主成分分析方法，它通过使用核技巧将高维数据映射到低维空间中，从而提取出数据中的主要特征。与传统的 PCA 相比，Kernel PCA 可以更好地处理非线性数据，更准确地保留数据中的非线性结构。

可以这样理解，PCA 是 Kernel PCA 的特列。PCA 中用到的格拉姆矩阵、协方差矩阵、相关性系数矩阵都可以看成是不同线性核。

图 9 (a) 所示数据线性不可分，我们先用非线性映射把数据映射到高维空间，使其线性可分。利用 KPCA 之后的结果如图 9 (b)。这一点和支持向量机中的核技巧颇为类似。

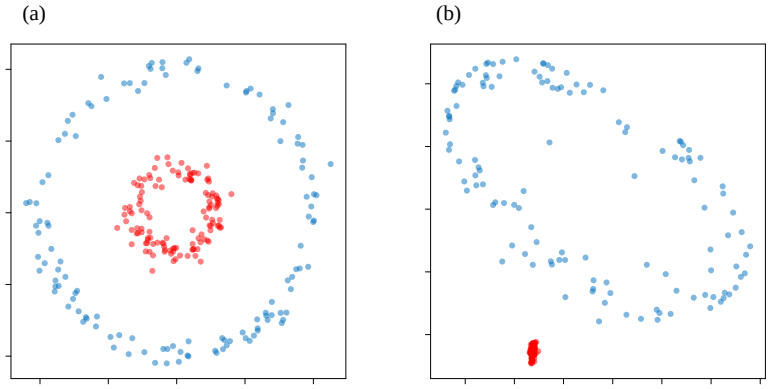


图 9. 核主成分分析

参考自如下示例，请大家自行学习：

https://scikit-learn.org/stable/auto_examples/decomposition/plot_kernel_pca.html

核主成分分析算法介绍，请参考：

https://people.eecs.berkeley.edu/~wainwrig/stat241b/scholkopf_kernel.pdf

18.3 独立成分分析

独立成分分析是一种用于从混合信号中恢复原始信号的数学方法。ICA 通过将混合信号映射到独立的成分空间中，从而恢复原始信号。独立成分分析将一个多元信号分解成独立性最强的可加子成分。因此，独立成分分析常用来分离叠加信号。

图 10 比较 PCA 和 CCA 对同一组数据的分解结果。与 PCA 不同的是，ICA 假设原始信号是独立的，而 PCA 假设它们是正交关系。

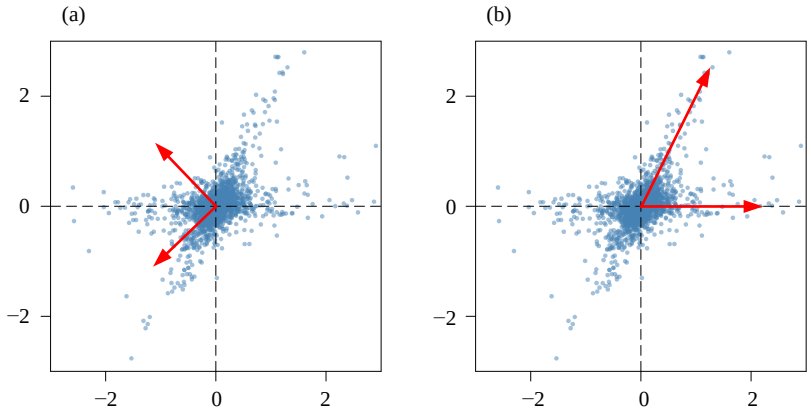


图 10. 比较 PCA 和 ICA

参考自如下示例，请大家自行学习：

https://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_vs_pca.html

有关独立成分分析算法原理，请大家参考：

https://www.emerald.com/insight/content/doi/10.1016/j.aci.2018.08.006/full/html

18.4 流形学习

空间的数据可能是按照某种规则“卷曲”，度量点与点之间的“距离”要遵循这种卷曲的趋势。换一种思路，我们可以像展开“卷轴”一样，将数据展开并投影到一个平面上，得到的数据如图 12 所示。在图 12 所示平面上，A 和 B 两点的“欧氏距离”更好地描述了两点的距离度量，因为这个距离考虑了数据的“卷曲”。

流形学习 (manifold learning) 核心思想类似图 11 和图 12 所示展开“卷轴”的思想。流形学习用于发现高维数据中的低维结构，也是非线性降维的一种方法。于 PCA 不同的是，流形学习可以更好地处理非线性数据和局部结构，具有更好的可视化效果和数据解释性。

在 scikit-learn 中，流形学习的函数是 sklearn.manifold 模块中的 Isomap、LocallyLinearEmbedding、SpectralEmbedding 和 TSNE 等。其中，Isomap 使用测地线距离来保留流形上的全局结构，LocallyLinearEmbedding 使用局部线性嵌入来保留局部结构，SpectralEmbedding 使用谱分解来发现流形的嵌入表示，TSNE 使用高斯分布来优化样本的嵌入表示，用于可视化高维数据。这些函数提供了一种方便、高效、易于使用的流形学习工具，可帮助大家更好地理解数据结构和特征。本书不展开讲解流形学习。

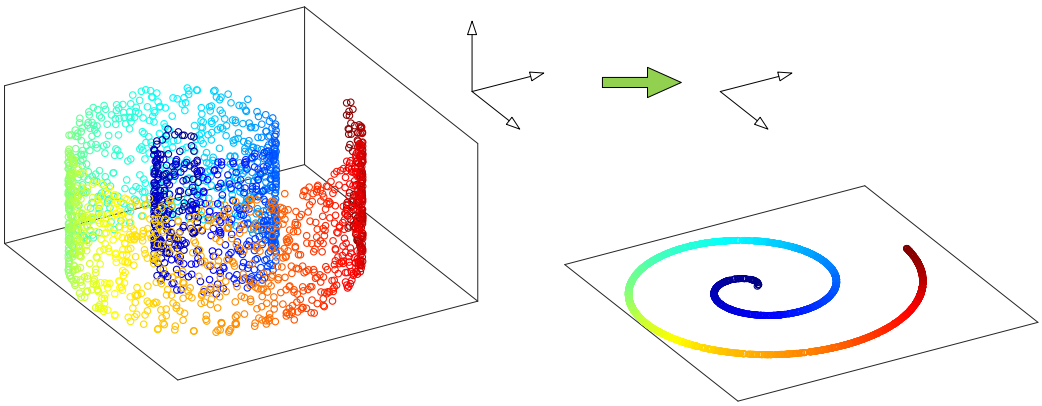


图 11. “卷曲”的数据

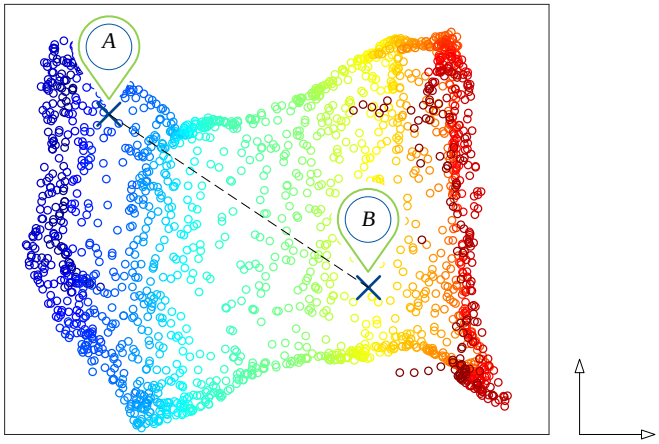


图 12. 展开“卷曲”的数据

机器学习中的降维是一种数据预处理技术，旨在通过减少特征数量来提高模型的训练效率和泛化能力。其中，主成分分析 PCA 是最常见的线性降维技术，前者通过将数据投影到最大方差方向上，后者则试图解释数据背后的因素。典型相关分析 CCA，是一种用于研究两个数据集之间的相关性的线性降维技术。它将两个数据集中的每个变量对应地进行线性组合，以使得这两个新的变量集之间的相关性最大。内核主成分分析 KPCA 是一种非线性降维方法，能够将数据映射到高维特征空间中，从而在新空间中找到数据的低维表示。独立成分分析 ICA 则试图从混合信号中恢复原始信号，假设原始信号是独立的。这些降维技术在机器学习、计算机视觉、信号处理、神经科学等领域都有广泛的应用，是提高数据处理和模型性能的重要工具。

Scikit-learn 中更多有关降维工具，请大家参考：

https://scikit-learn.org/stable/modules/decomposition.html

想要深入了解 Scikit-learn 中的流形学习工具，请大家参考：

https://scikit-learn.org/stable/modules/manifold.html

如下这篇文献介绍了流形学习的数学基础，请大家参考：

https://arxiv.org/pdf/2011.01307.pdf

“鸢尾花书”的整体布局

数学

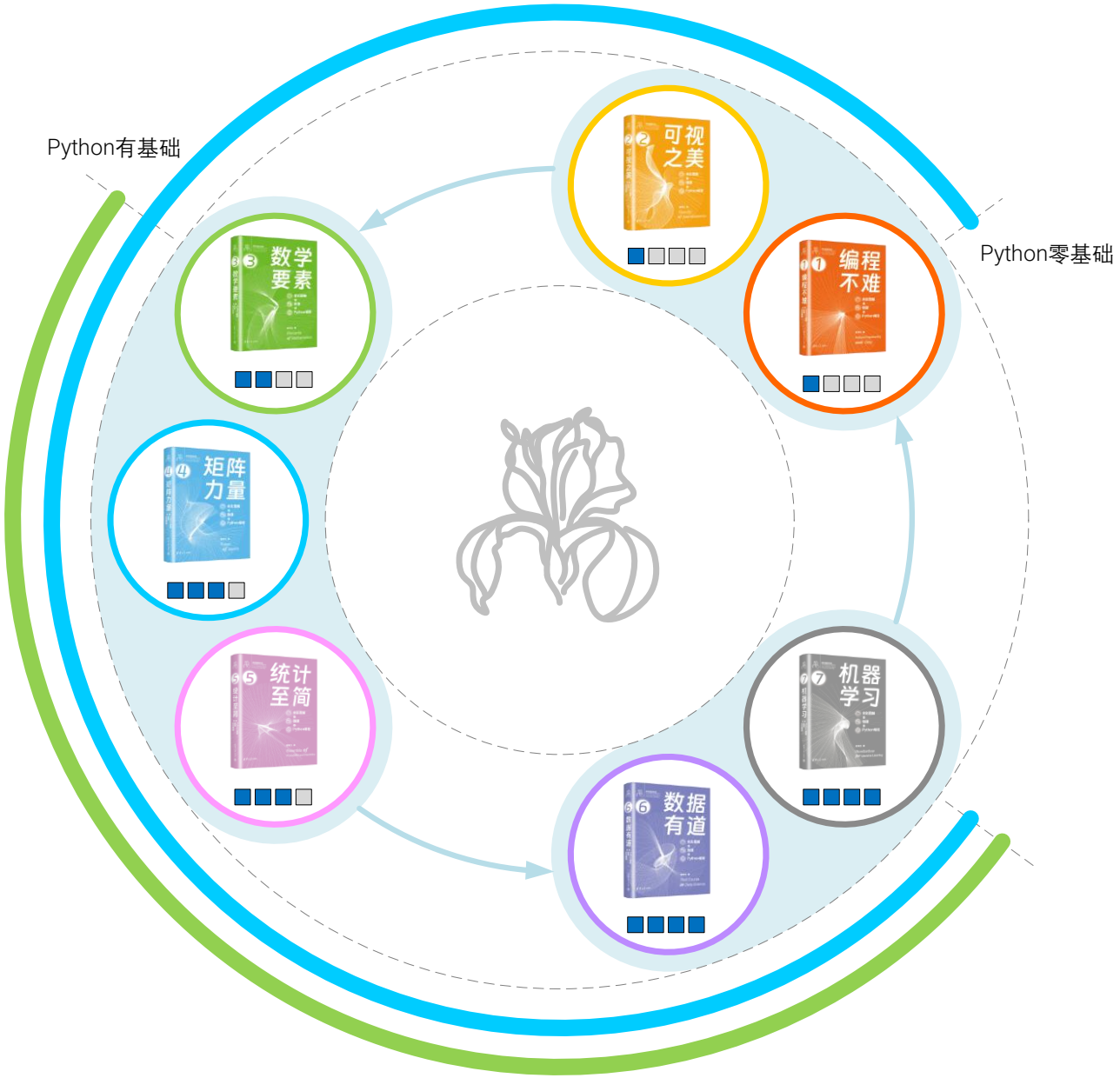
数学基础　线性代数　概率统计

Python编程　数据可视化　回归、降维　分类、聚类

工具　实践

数学 + Python编程 + 可视化 + 机器学习实践

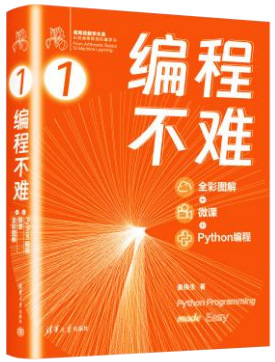
“鸢尾花书”的学习顺序



分册进度状态

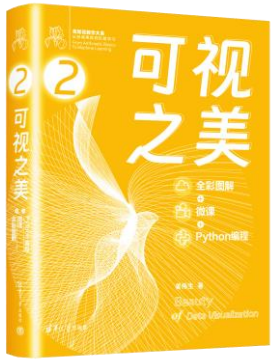
	草稿、Python	打磨、视频	清华社五审五校	上架
1 编程不难 	~50%			
2 可视之美 	~100%	2023, 10		
3 数学要素 	100%	完成	完成	完成
4 矩阵力量 	100%	完成	完成	完成
5 统计至简 	100%	2023, 07	2023, 08	2023, 09
6 数据有道 	80%	2024年初		
7 机器学习 	80%	2024年初		

Book 1《编程不难》



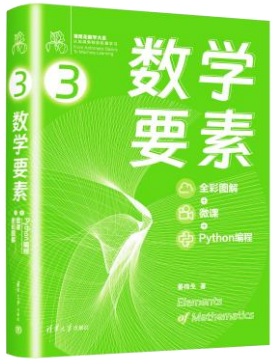
- 编程不难
  - 预备
    - 介绍
    - JupyterLab
    - Spyder
  - 语法
    - 基础
    - 数据类型
    - 运算
    - 循环
    - 函数
    - 面向对象编程
  - 绘图
    - 基础
    - 二维
    - 三维
  - 数组
    - 基础
    - 索引和切片
    - 常见运算
    - 变形
    - 数组操作
    - 线性代数
  - 符号数学
    - 代数
    - 微积分
    - 线性代数进阶
  - 数值计算
    - 代数进阶
    - 概率统计
    - 算法
  - 数据科学
    - 数据帧
    - 索引和切片
    - 常见运算
    - 规整
    - 时间序列
    - 统计分析
  - 机器学习
    - 综述
    - 回归
    - 降维
    - 分类
    - 聚类
    - 神经网络

Book 2《可视之美》



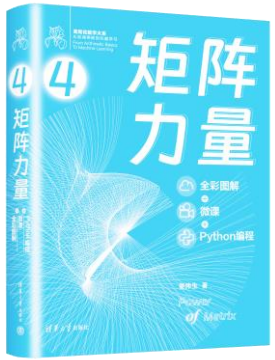
- 《可视之美》
  - 图说
    - 数学 + 艺术
    - 说图
  - 美化
    - 布局
    - 装饰
    - 风格
    - 媒介
  - 彩色
    - 色彩
    - 色彩空间
    - 色谱
  - 二维
    - 平面散点
    - 平面线图
    - 平面填充
    - 极坐标
    - 各种网格
    - 平面等高线
    - 热图
    - 非矢量图片
  - 三维
    - 三维散点
    - 三维线图
    - 网格曲面
    - 三维等高线
    - 箭头图
    - 三角网格
  - 数学之美
    - 平面几何
    - 立体几何
    - 距离
    - 平面几何变换
    - 立体几何变换
    - 贝塞尔曲线
    - 函数
    - 二次型
    - 复数
    - 分形
    - 网络图
    - 数据
    - 模式 + 随机

Book 3《数学要素》



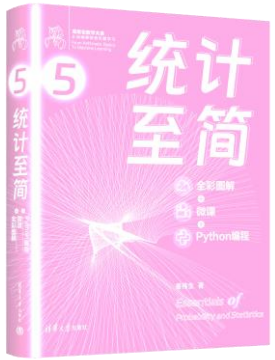
- 数学要素
  - 基础
    - 算数
    - 几何
    - 代数
  - 坐标系
    - 二维
    - 三维
  - 解析几何
    - 距离
    - 圆锥曲线
  - 函数
    - 可视化
    - 代数函数
    - 超越函数
    - 二元函数
    - 数列
  - 微积分
    - 导数
    - 偏导数
    - 微分
    - 积分
    - 优化入门
  - 概率统计
    - 概率
    - 统计
  - 线性代数
    - 向量
    - 鸡兔同笼三部曲

Book 4《矩阵力量》



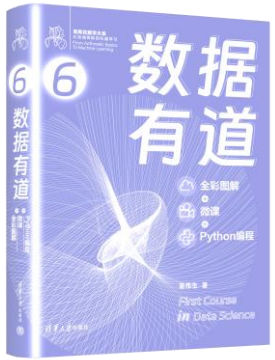
- 矩阵力量
  - 向量
    - 定义
    - 运算
    - 范数
  - 矩阵
    - 定义
    - 运算
    - 分块矩阵
  - 向量空间
    - 定义
    - 几何变换
    - 投影
    - 数据投影
  - 矩阵分解
    - 各种分解
    - Cholesky分解
    - 特征值分解
    - 奇异值分解
  - 微积分
    - 多元微分
    - 拉格朗日乘子法
  - 空间几何
    - 直线到超平面
    - 曲线
    - 曲面
  - 数据
    - 多元统计入门
    - 数据空间
    - 数据分解
    - 数据应用

Book 5《统计至简》



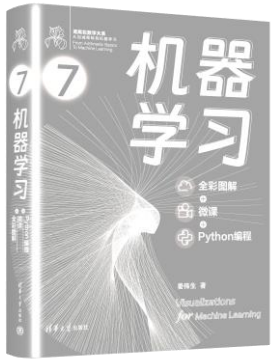
- 统计至简
  - 统计
    - 概率统计全景
    - 统计描述
  - 概率
    - 古典概率模型
    - 离散随机变量
    - 离散分布
    - 连续随机变量
    - 连续分布
    - 条件概率
  - 高斯
    - 一元
    - 二元
    - 多元
    - 条件
    - 协方差矩阵
  - 随机
    - 随机变量的函数
    - 蒙特卡洛模拟
  - 频率派
    - 频率统计推断
    - 概率密度估计
  - 贝叶斯派
    - 贝叶斯分类
    - 贝叶斯统计推断
    - 马尔科夫链
  - 椭圆
    - 马氏距离
    - 线性回归
    - 主成分分析

Book 6《数据有道》



- 数据有道
  - 数据处理
    - 缺失值
    - 离群值
    - 数据转换
    - 插值
  - 时间数据
    - 时间数据
    - 移动窗口
    - 随机过程入门
  - 回归
    - 回归分析
    - 多元线性回归
    - 正则化
    - 贝叶斯回归
    - 非线性回归
  - 降维
    - 主成分分析
    - 正交回归
    - 主元回归
    - 典型相关分析

Book 7《机器学习》


7
机器学习
全彩图解
微课
Python编程
Visualizations for Machine Learning


机器学习
有监督学习
k近邻
距离
朴素贝叶斯
高斯朴素贝叶斯
高斯判别分析
支持向量机
核技巧
决策树
高斯过程
回归
无监督学习
k均值聚类
高斯混合模型
最大期望算法
层次聚类
密度聚类
谱聚类
降维

开源资源

PDF书稿、代码：https://github.com/Visualize-ML

微课视频：https://space.bilibili.com/513194466

信息发布：https://www.zhihu.com/people/jamestong-xue

专属邮箱：jiang.visualize.ml@gmail.com