8 *k*-nearest neighbors algorithm

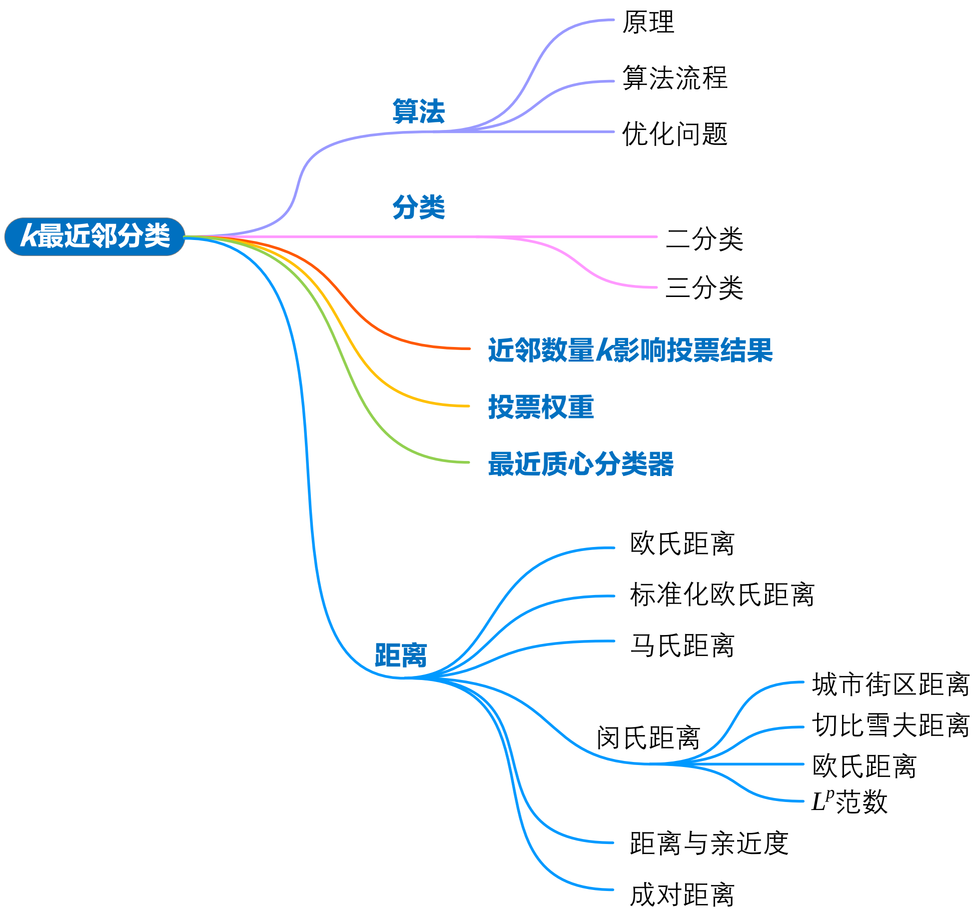
k 最近邻分类

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
- `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.array()` 创建 `array` 数据类型
- `numpy.c_()` 按列叠加两个矩阵
- `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
- `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
- `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

- `seaborn.scatterplot()` 绘制散点图
- `sklearn.datasets.load_iris()` 加载鸢尾花数据集
- `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
- `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


*k*最近邻分类
算法
原理
算法流程
优化问题
分类
二分类
三分类
近邻数量*k*影响投票结果
投票权重
最近质心分类器
距离
欧氏距离
标准化欧氏距离
马氏距离
闵氏距离
城市街区距离
切比雪夫距离
欧氏距离
L^p范数
距离与亲近度
成对距离

8.1 *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。

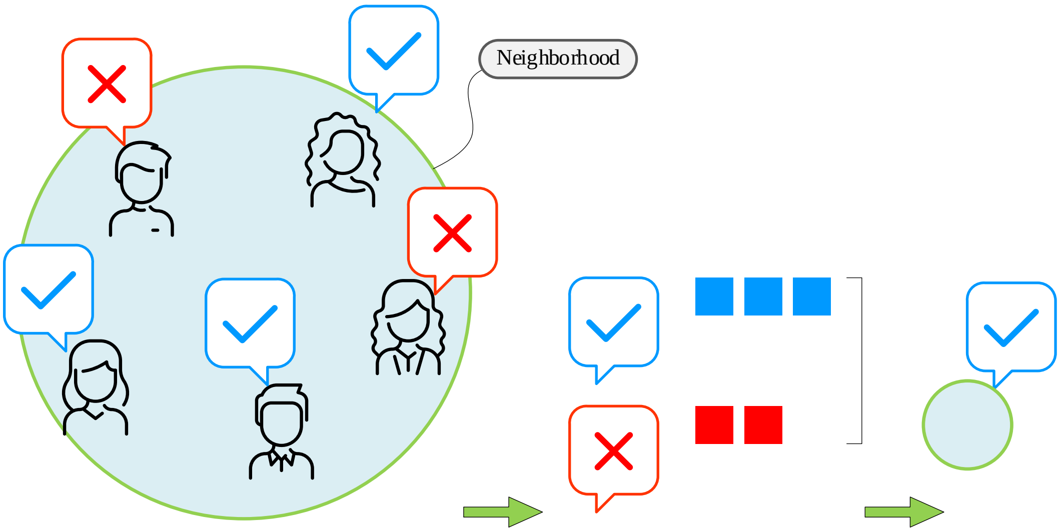


图 1. k 近邻分类核心思想——小范围投票，少数服从多数

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) $\hat{y}$：

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

其中，$k\text{NN}(\boldsymbol{q})$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。

8.2 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 ●)。

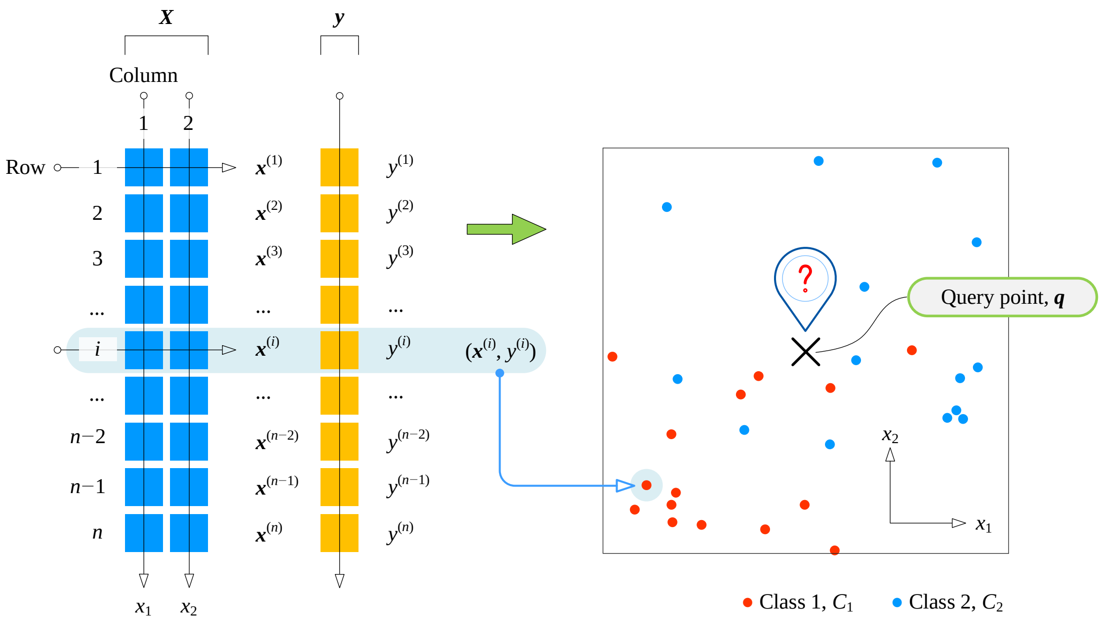


图 2. 两特征 ($D = 2$) 含标签样本数据可视化

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：

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

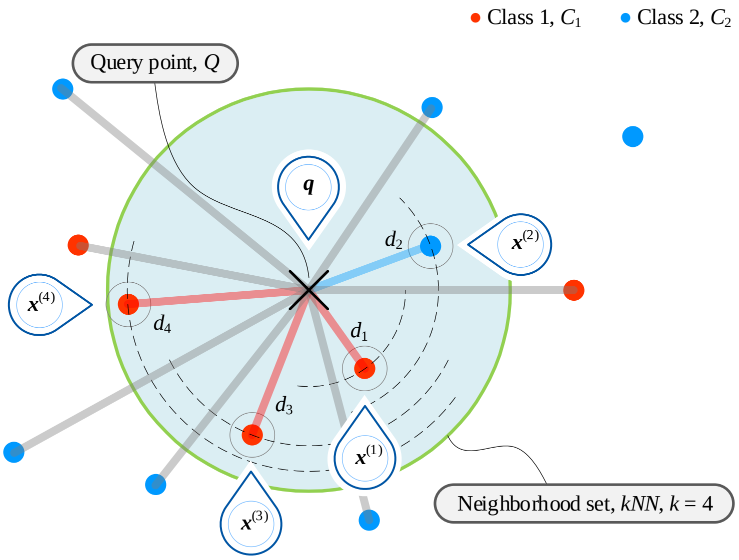


图 3. *k* 近邻原理

将具体分类标签带入 (2)，可以得到：

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

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。

8.3 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。

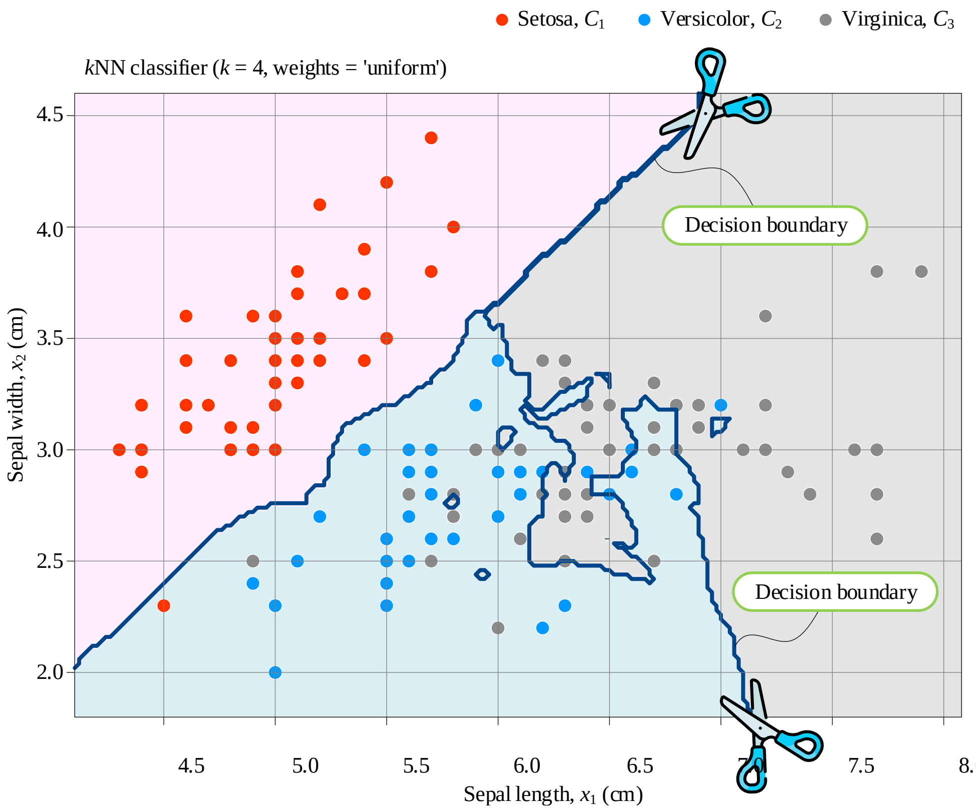


图 4. k 近邻分类，$k = 4$，采用 2 个特征 (花萼长度 x_1，和花萼宽度 x_2) 分类三种鸢尾花

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_Ch08_01.ipynb 可以用来实现本节分类问题，并绘制图 4。下面聊聊其中关键语句。

```
    # 近邻数量
a   k_neighbors = 4

    # kNN分类器
b   clf = neighbors.KNeighborsClassifier(k_neighbors)

    # 拟合数据
c   clf.fit(X, y)

    # 查询点
d   q = np.c_[xx1.ravel(), xx2.ravel()];

    # 预测
e   y_predict = clf.predict(q)

    # 规整形状
f   y_predict = y_predict.reshape(xx1.shape)
```

代码 1. 用 sklearn.neighbors.KNeighborsClassifier()分类 | Bk7_Ch08_01.ipynb

ⓐ 定义近邻的数量为 4，请大家尝试其他近邻数量。

ⓑ 用 sklearn.neighbors.KNeighborsClassifier()创建 kNN 分类对象。

ⓒ 调用 kNN 分类对象，并拟合数据。

ⓓ 这句话将网格坐标转化为二维数组。

ⓔ 对网格数据进行分类预测。

ⓕ 将预测结果规整为和网格数据相同形状，以便于后续可视化。

8.4 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 (d) 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。

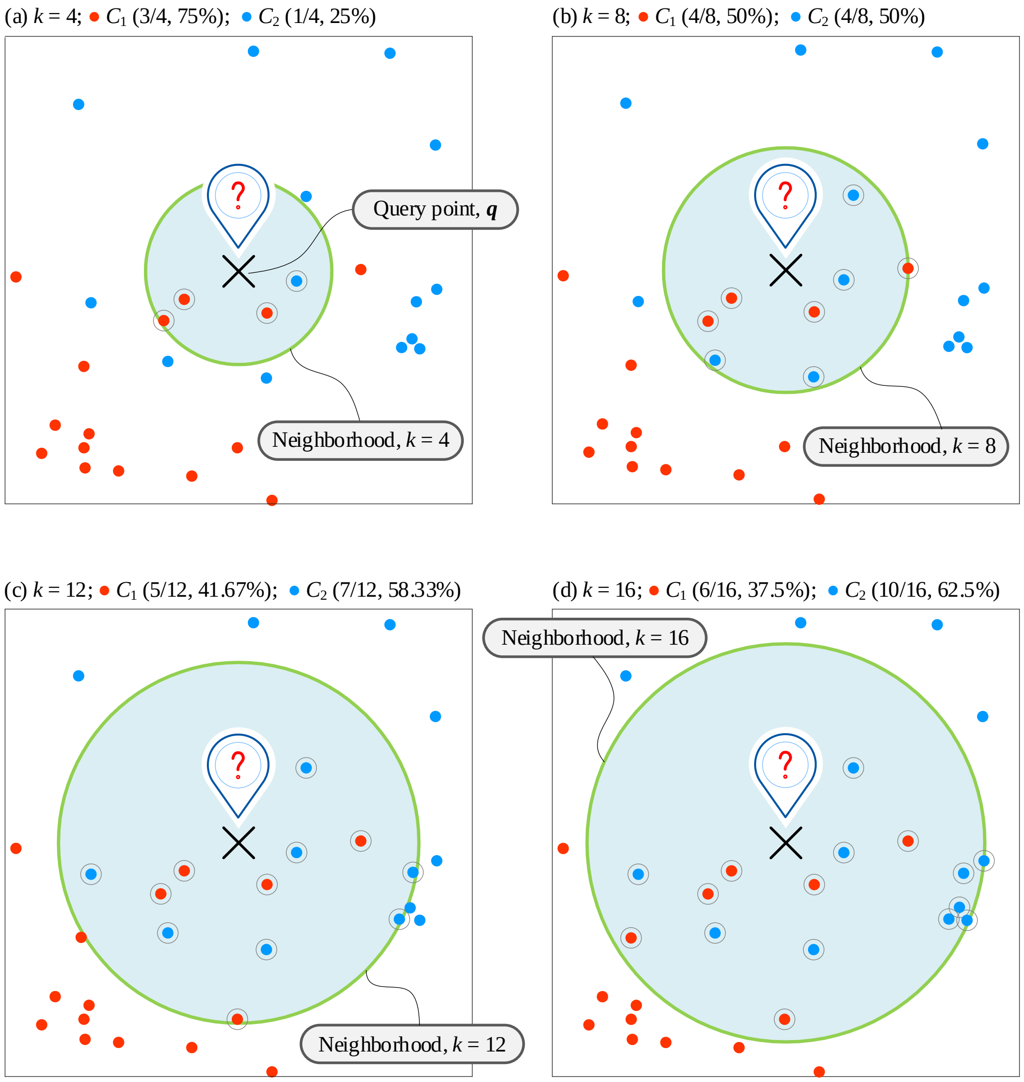


图 5. 近邻数量 k 值影响查询点的分类结果

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 Bk7_Ch08_04 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。

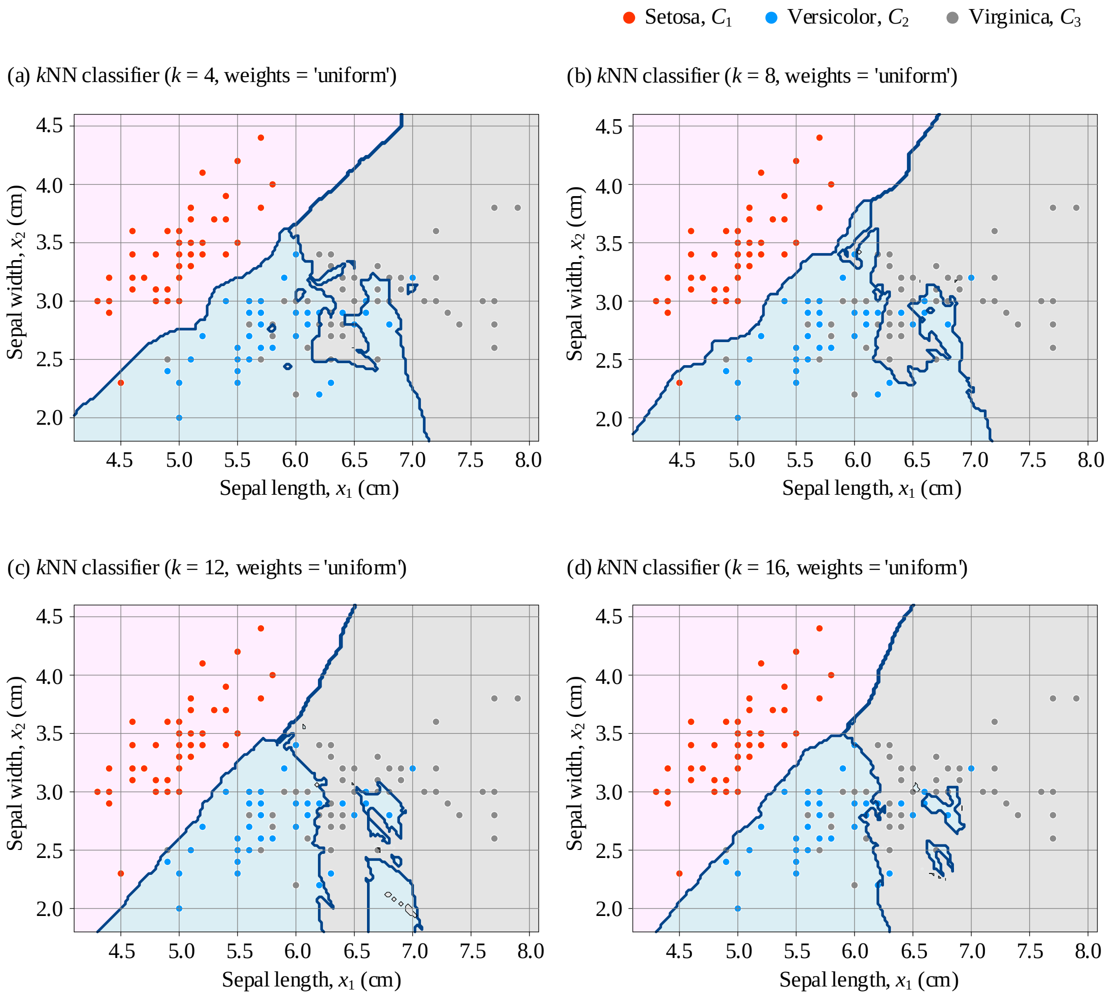


图 6. k-NN，k 选取不同值时对鸢尾花分类影响

图 7 所示为用 Streamlit 搭建的 App 展示 k 对 kNN 聚类结果影响。

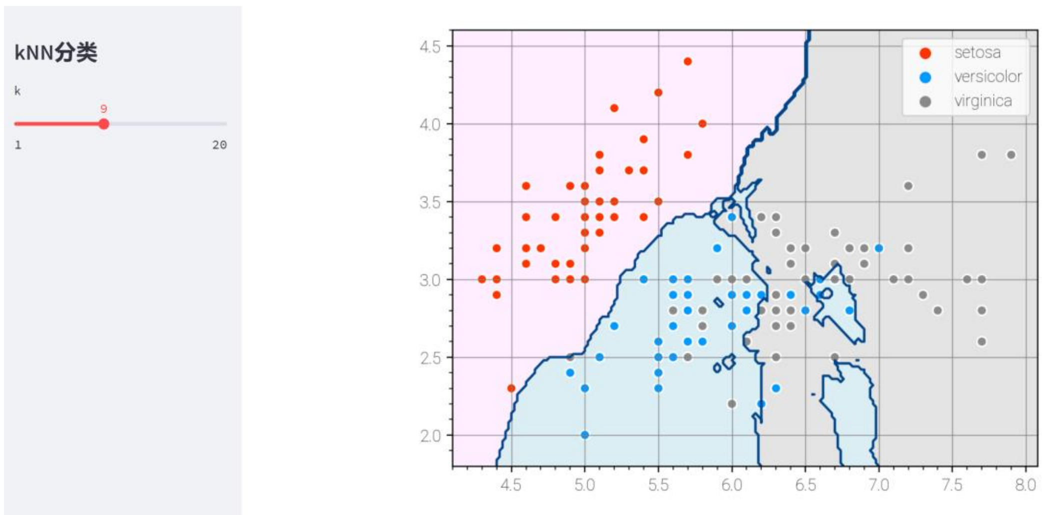


图 7．展示 k 对 kNN 聚类结果影响的 App，Streamlit 搭建 | Streamlit_Bk7_Ch08_02.py

代码 Streamlit_Bk7_Ch08_02.py 搭建图 7 所示 App，请大家自行学习。

8.5 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：

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

`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'`。

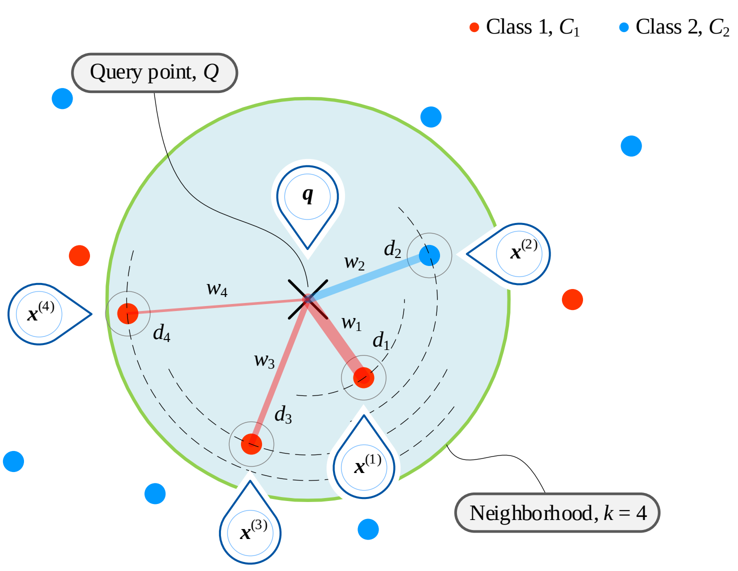


图 8. k 近邻原理，加权投票

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 *k*NN 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。

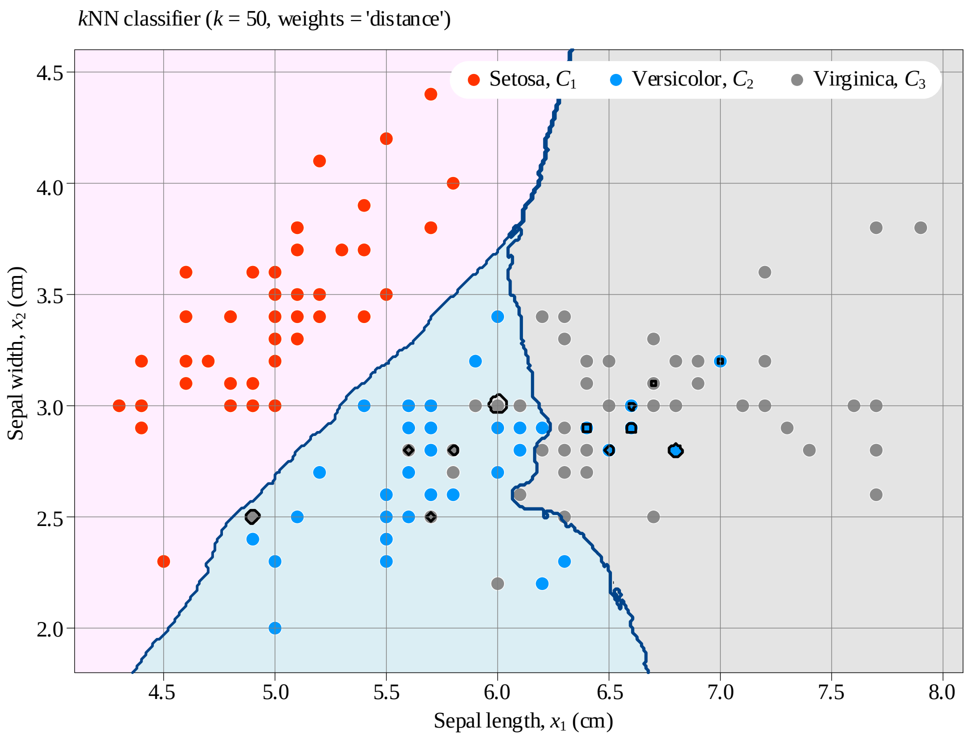


图 9. $k = 50$ 时，鸢尾花分类决策边界，投票权重与查询点距离成反比

8.6 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 $\boldsymbol{\mu}_m$ ($m = 1, 2, \ldots, K$)；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：

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

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。

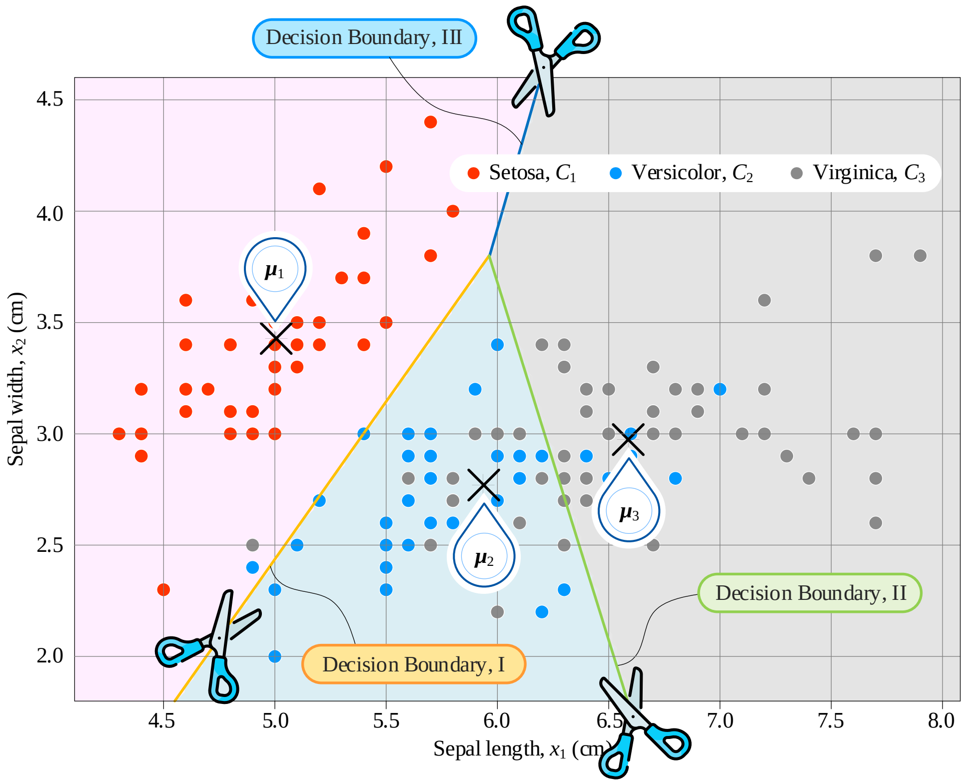


图 10. 鸢尾花分类决策边界，最近质心分类

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。

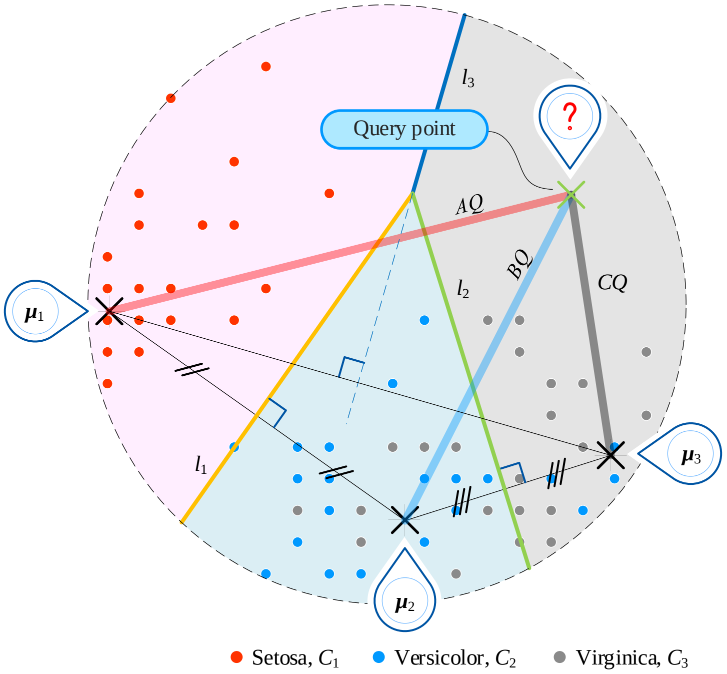


图 11. 最近质心分类决策边界原理

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，根据收缩阈值大小，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_Ch08_03.ipynb 绘制图 12 所示四幅图像。

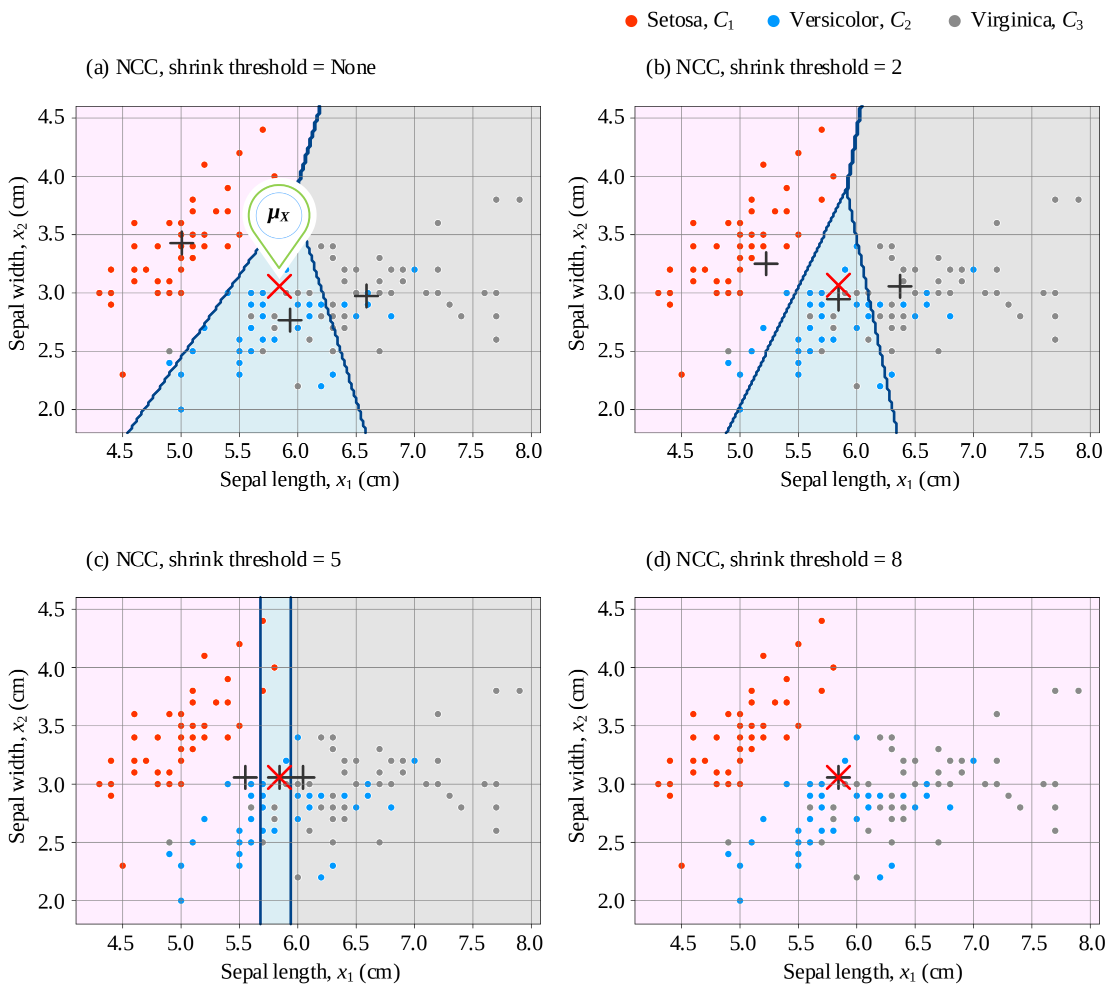


图 12. 收缩阈值增大对决策边界影响

8.7 k-NN 回归：非参数回归

本章前文的 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：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{k} \sum_{i \in kNN(\boldsymbol{q})} y^{(i)} \tag{8}$$

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 13 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 (8) 计算 $y^{(1)}$、$y^{(2)}$ 和 $y^{(3)}$ 平均值，得到查询点回归预测值 $\hat{y}$：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{3}\left(y^{(1)} + y^{(2)} + y^{(3)}\right) = \frac{1}{3}(5+3+4) = 4 \tag{9}$$

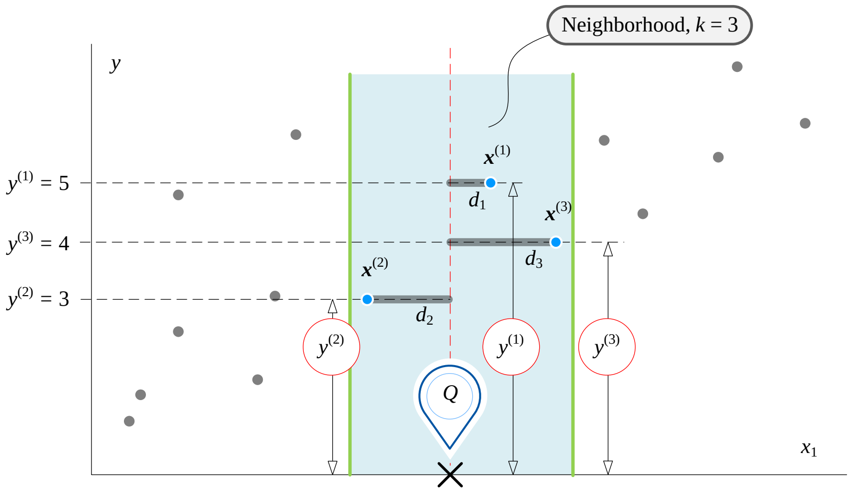


图 13. *k*-NN 回归算法原理

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：

$$\hat{y}(\boldsymbol{q}) = \frac{1}{\sum_{i \in kNN(\boldsymbol{q})} w_i} \sum_{i \in kNN(\boldsymbol{q})} w_i y^{(i)} \tag{10}$$

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 14 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 (10) 中 y。图 14 (a) 采用等权重投票，图 14 (b) 中投票权重与查询点距离成反比。

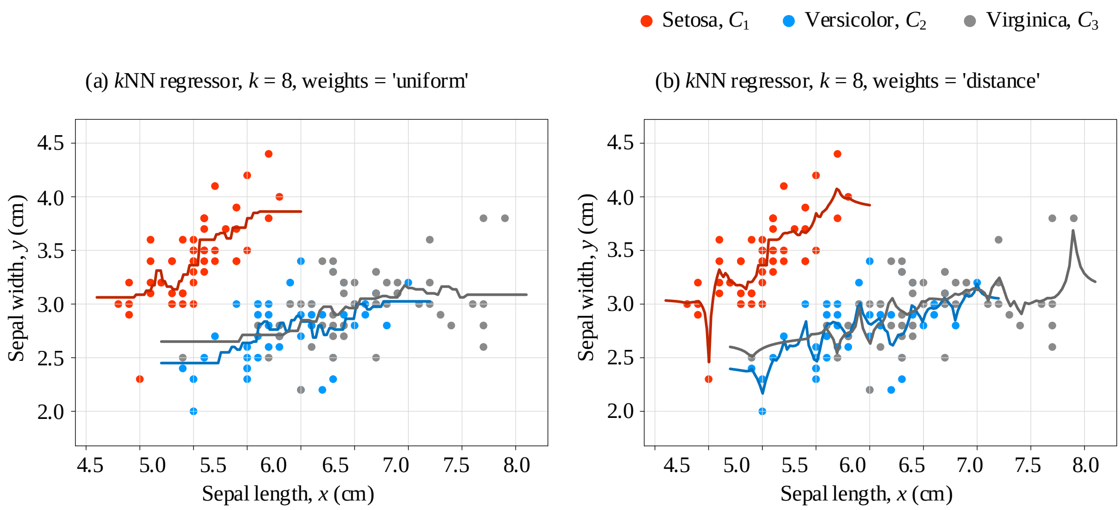


图 14. *k*-NN 回归，不同种类鸢尾花花萼长度和花萼宽度回归关系

代码 Bk7_Ch08_04.ipynb 完成 *k*-NN 回归，并绘制图 14 两幅图像。

8.8 各种距离度量

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。

本章后续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。

8.9 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：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\| = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{q})} \tag{11}$$

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：

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{(x_1-q_1)^2+(x_2-q_2)^2+\dots+(x_D-q_D)^2} \tag{12}$$

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：

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{(x_1-q_1)^2+(x_2-q_2)^2} \tag{13}$$

举个例子

如果查询点 $\boldsymbol{q}$ 有两个特征，并位于原点，即：

$$\boldsymbol{q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{14}$$

如图 15 所示，三个样本点 $\boldsymbol{x}^{(1)}$、$\boldsymbol{x}^{(2)}$和 $\boldsymbol{x}^{(3)}$的位置如下：

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -5 & 0 \end{bmatrix},\quad \boldsymbol{x}^{(3)} = \begin{bmatrix} 4 & 3 \end{bmatrix},\quad \boldsymbol{x}^{(3)} = \begin{bmatrix} 3 & -4 \end{bmatrix} \tag{15}$$

根据 (11) 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：

$$\begin{cases} d_1 = \sqrt{\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} -5 & 0 \end{bmatrix}\right)\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} -5 & 0 \end{bmatrix}\right)^{\mathrm{T}}} = \sqrt{\begin{bmatrix} 5 & 0 \end{bmatrix}\begin{bmatrix} 5 & 0 \end{bmatrix}^{\mathrm{T}}} = \sqrt{25+0} = 5 \\ d_2 = \sqrt{\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 4 & 3 \end{bmatrix}\right)\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 4 & 3 \end{bmatrix}\right)^{\mathrm{T}}} = \sqrt{\begin{bmatrix} -4 & -3 \end{bmatrix}\begin{bmatrix} -4 & -3 \end{bmatrix}^{\mathrm{T}}} = \sqrt{16+9} = 5 \\ d_3 = \sqrt{\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 3 & -4 \end{bmatrix}\right)\left(\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} 3 & -4 \end{bmatrix}\right)^{\mathrm{T}}} = \sqrt{\begin{bmatrix} -3 & 4 \end{bmatrix}\begin{bmatrix} -3 & 4 \end{bmatrix}^{\mathrm{T}}} = \sqrt{9+16} = 5 \end{cases} \tag{16}$$

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

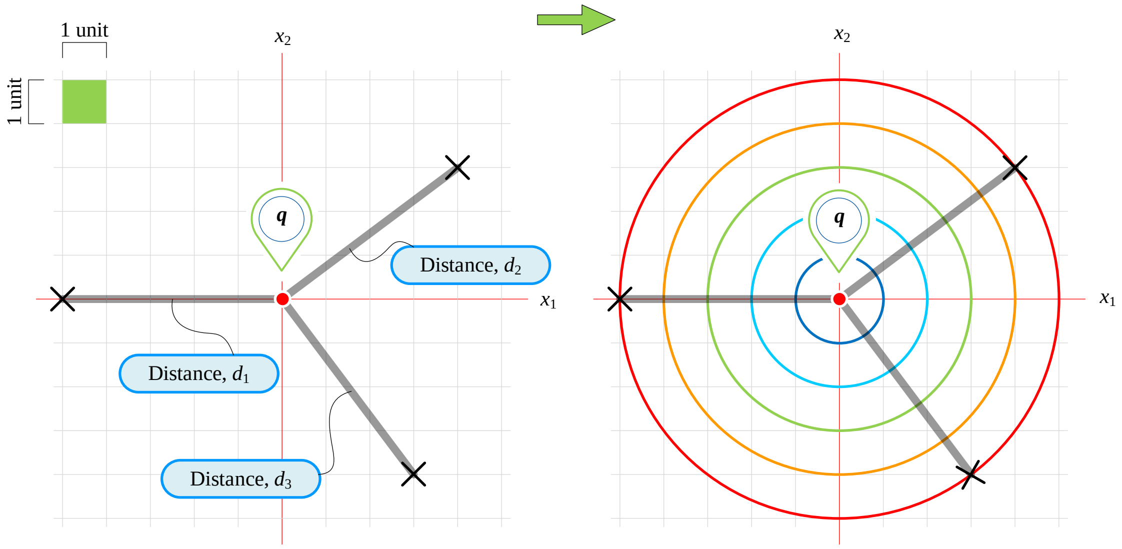


图 15. 2 特征 (D = 2) 欧几里得距离

代码 Bk7_Ch08_05.ipynb 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 15 所示，三个样本点 $x^{(1)}$、$x^{(2)}$和 $x^{(3)}$ 之间也存在两两距离，我们管它们叫做**成对距离** (pairwise distance)。图 16 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。

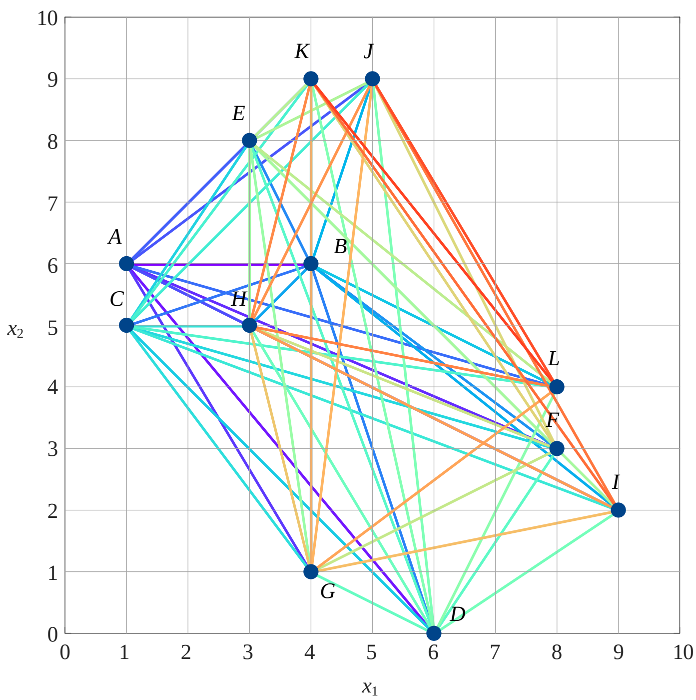


图 16. 平面上 12 个点，成对距离，来自鸢尾花书《数学要素》

代码 Bk7_Ch08_06.ipynb 计算图 15 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。

8.10 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。

$$d\left(\boldsymbol{x},\boldsymbol{q}\right)=\sqrt{\left(\boldsymbol{x}-\boldsymbol{q}\right)^{\mathrm{T}}\boldsymbol{D}^{-1}\boldsymbol{D}^{-1}\left(\boldsymbol{x}-\boldsymbol{q}\right)} \tag{17}$$

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

$$\boldsymbol{D}=\operatorname{diag}\left(\operatorname{diag}\left(\boldsymbol{\Sigma}\right)\right)^{\frac{1}{2}}=\operatorname{diag}\left(\operatorname{diag}\begin{bmatrix}\sigma_1^2 & \rho_{1,2}\sigma_1\sigma_2 & \cdots & \rho_{1,D}\sigma_1\sigma_D\\ \rho_{1,2}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2,D}\sigma_2\sigma_D\\ \vdots & \vdots & \ddots & \vdots\\ \rho_{1,D}\sigma_1\sigma_D & \rho_{2,D}\sigma_2\sigma_D & \cdots & \sigma_D^2\end{bmatrix}\right)^{\frac{1}{2}}=\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_D\end{bmatrix} \tag{18}$$

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 (18) 带入 (17) 得到：

$$\begin{aligned}d\left(\boldsymbol{x},\boldsymbol{q}\right)&=\sqrt{\begin{bmatrix}x_1-q_1 & x_2-q_2 & \cdots & x_D-q_D\end{bmatrix}\begin{bmatrix}\sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_D^2\end{bmatrix}^{-1}\begin{bmatrix}x_1-q_1 & x_2-q_2 & \cdots & x_D-q_D\end{bmatrix}^{\mathrm{T}}}\\ &=\sqrt{\frac{\left(x_1-q_1\right)^2}{\sigma_1^2}+\frac{\left(x_2-q_2\right)^2}{\sigma_2^2}+\dots+\frac{\left(x_D-q_D\right)^2}{\sigma_D^2}}=\sqrt{\sum_{j=1}^{D}\left(\frac{x_j-q_j}{\sigma_j}\right)^2}\end{aligned} \tag{19}$$

(19) 可以记做：

$$d\left(\boldsymbol{x},\boldsymbol{q}\right)=\sqrt{z_1^2+z_2^2+\dots+z_D^2}=\sqrt{\sum_{j=1}^{D}z_j^2} \tag{20}$$

其中，z_j 为：

$$z_j = \frac{x_j - q_j}{\sigma_j} \tag{21}$$

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：

$$d^2 = \frac{(x_1 - q_1)^2}{\sigma_1^2} + \frac{(x_2 - q_2)^2}{\sigma_2^2} \tag{22}$$

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 (22)，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 17 中，网格的坐标已经转化为"标准差"，而标准化欧氏距离等距线为正椭圆。

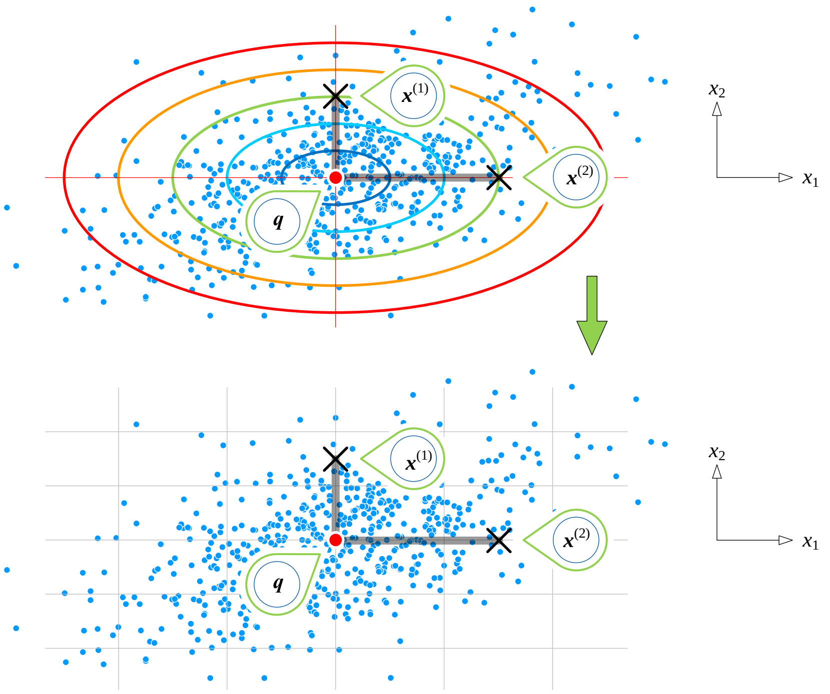


图 17. 2 特征 ($D = 2$) 标准化欧氏距离

几何变换视角

如图 18 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。

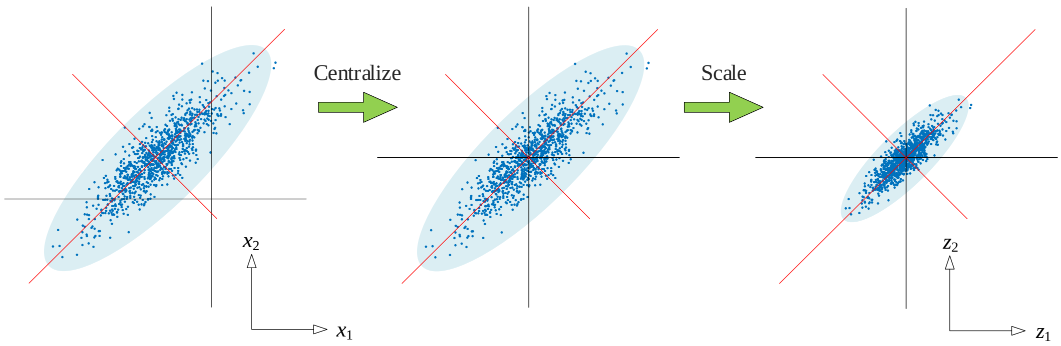


图 18. 标准化欧氏距离运算过程

计算标准化欧氏距离的函数为 `scipy.spatial.distance.seuclidean()`。代码 Bk7_Ch08_07.ipynb 计算本节标准化欧氏距离。

8.11 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：

$$d(\boldsymbol{x},\boldsymbol{q}) = \sqrt{(\boldsymbol{x}-\boldsymbol{q})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{q})} \tag{23}$$

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：

$$\boldsymbol{\Sigma}=\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}} \tag{24}$$

其中，$\boldsymbol{V}$ 为正交矩阵。

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

$$\boldsymbol{\Sigma}^{-1}=\left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}}\right)^{-1}=\left(\boldsymbol{V}^{\mathrm{T}}\right)^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^{-1}=\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^{\mathrm{T}} \tag{25}$$

将 (25) 代入 (23) 得到：

$$d\left(\boldsymbol{x},\boldsymbol{\mu}\right)=\left\|\underbrace{\boldsymbol{\Lambda}^{\frac{-1}{2}}}_{\text{Scale}}\underbrace{\boldsymbol{V}^{\mathrm{T}}}_{\text{Rotate}}\underbrace{\left(\boldsymbol{x}-\boldsymbol{\mu}\right)}_{\text{Centralize}}\right\| \tag{26}$$

其中，$\boldsymbol{\mu}$ 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 19 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。

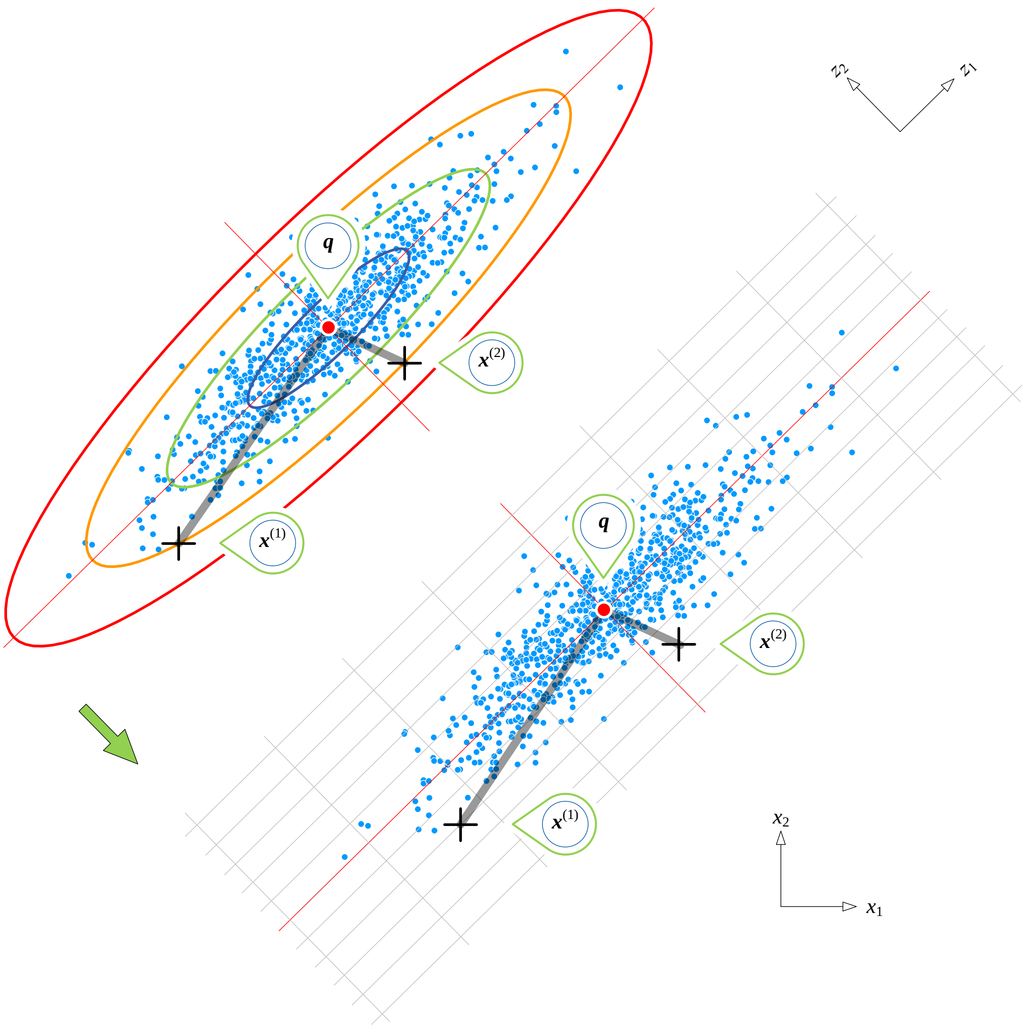


图 19. 2 特征 (D = 2) 马氏距离

代码 Bk7_Ch08_08.ipynb 计算图 19 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 $\boldsymbol{\mu} = [0, 0]^{\mathrm{T}}$。两个样本点的坐标分别为。

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -3.5 & -4 \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{x}^{(2)} = \begin{bmatrix} 2.75 & -1.5 \end{bmatrix}^{\mathrm{T}} \tag{27}$$

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。

$$\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \tag{28}$$

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。

$$\begin{aligned} d_1 &= \sqrt{\left(\begin{bmatrix} -3.5 & -4 \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\right)\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1}\left(\begin{bmatrix} -3.5 & -4 \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\right)^{\mathrm{T}}} \\ &= \sqrt{\begin{bmatrix} -3.5 & -4 \end{bmatrix} \cdot \frac{1}{3} \cdot \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} -3.5 & -4 \end{bmatrix}^{\mathrm{T}}} = 3.08 \\ d_2 &= \sqrt{\left(\begin{bmatrix} 2.75 & -1.5 \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\right)\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}^{-1}\left(\begin{bmatrix} 2.75 & -1.5 \end{bmatrix} - \begin{bmatrix} 0 & 0 \end{bmatrix}\right)^{\mathrm{T}}} \\ &= \sqrt{\begin{bmatrix} 2.75 & -1.5 \end{bmatrix} \cdot \frac{1}{3} \cdot \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}\begin{bmatrix} 2.75 & -1.5 \end{bmatrix}^{\mathrm{T}}} = 3.05 \end{aligned} \tag{29}$$

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

8.12 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：

$$d(\boldsymbol{x}, \boldsymbol{q}) = \|\boldsymbol{x} - \boldsymbol{q}\|_1 = \sum_{j=1}^{D} |x_j - q_j| \tag{30}$$

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 (30) 展开得到下式：

$$d(\boldsymbol{x}, \boldsymbol{q}) = |x_1 - q_1| + |x_2 - q_2| + \ldots + |x_D - q_D| \tag{31}$$

特别地，当 $D = 2$ 时，城市街区距离为：

$$d(\boldsymbol{x}, \boldsymbol{q}) = |x_1 - q_1| + |x_2 - q_2| \tag{32}$$

旋转正方形

如图 20 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。

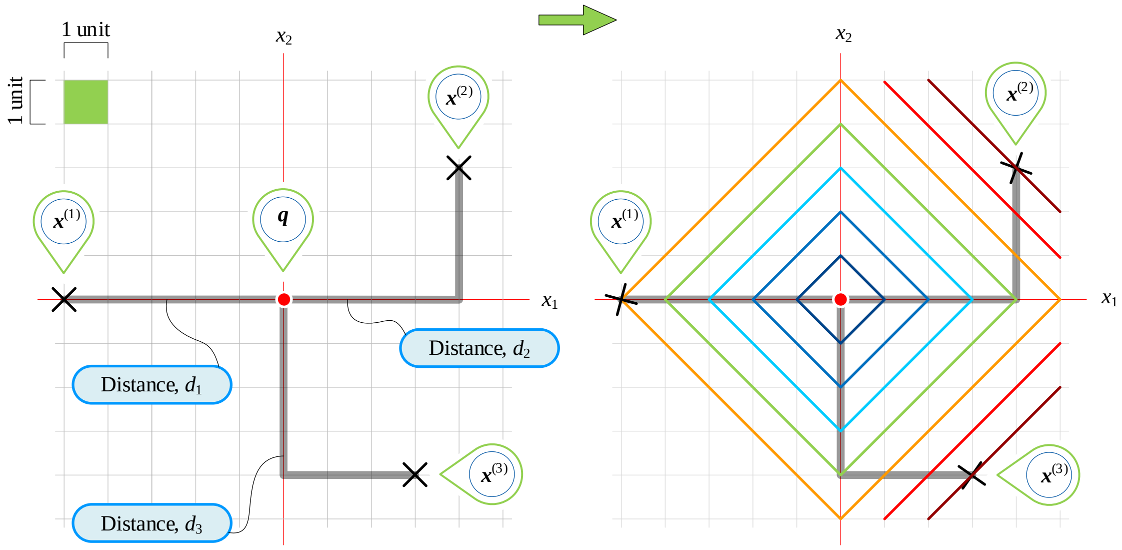


图 20. 2 特征 (D = 2) 城市街区距离

代码 Bk7_Ch08_09.ipynb 给出两种方法计算得到图 20 所示城市街区距离。

8.13 切比雪夫距离：L^∞范数

切比雪夫距离 (Chebyshev distance)，具体如下：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\|_\infty = \max_j \left\{ |x_j - q_j| \right\} \tag{33}$$

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 (33) 展开得到下式：

$$d(\boldsymbol{x},\boldsymbol{q}) = \max\left\{ |x_1 - q_1|,\ \ |x_2 - q_2|,\ \ \dots,\ \ |x_D - q_D| \right\} \tag{34}$$

特别地，当 D = 2 时，切比雪夫距离为：

$$d(\boldsymbol{x},\boldsymbol{q}) = \max\left\{ |x_1 - q_1|,\ \ |x_2 - q_2| \right\} \tag{35}$$

正方形

如图 21 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。

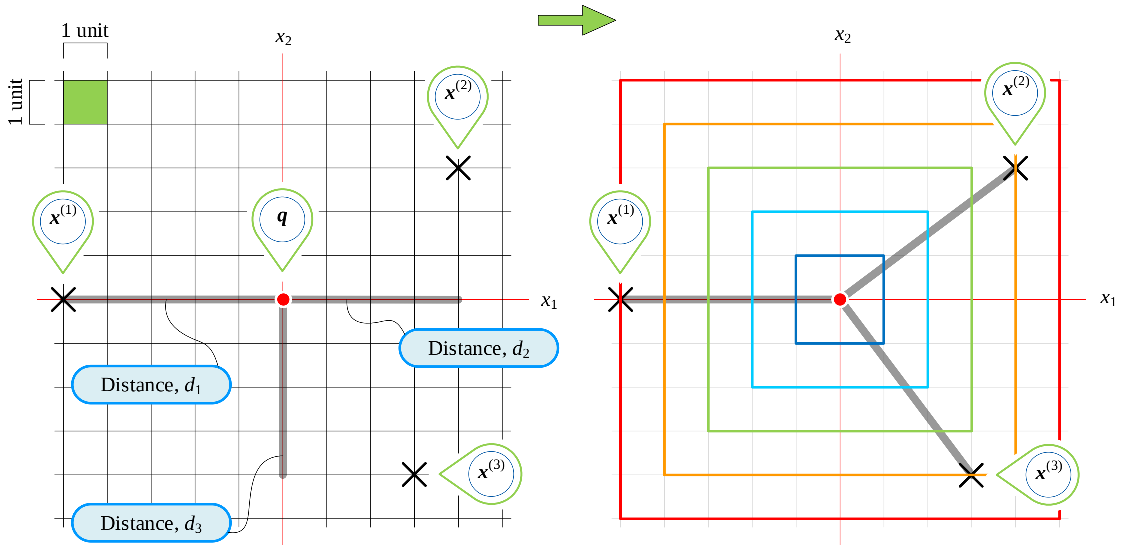


图 21. 2 特征 ($D = 2$) 切比雪夫距离

代码 Bk7_Ch08_10.ipynb 计算图 21 所示切比雪夫距离。

8.14 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：

$$d(\boldsymbol{x},\boldsymbol{q}) = \|\boldsymbol{x}-\boldsymbol{q}\|_p = \left(\sum_{j=1}^{D}\left|x_j - q_j\right|^p\right)^{1/p} \tag{36}$$

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 22 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。

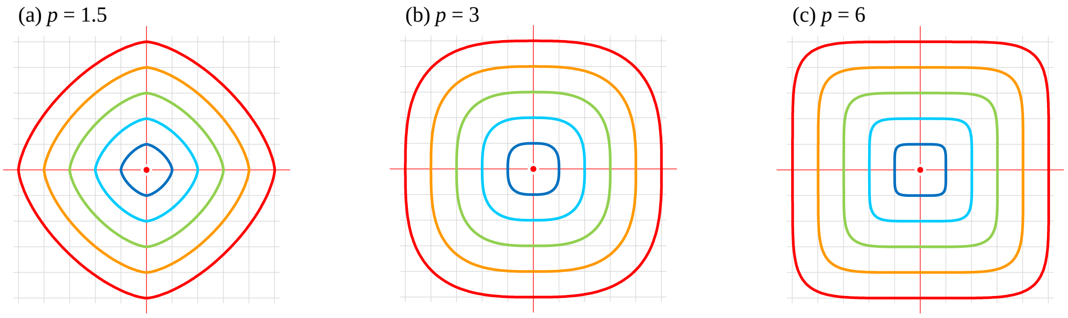


图 22. 闵氏距离 (D = 2)，p 取不同值

8.15 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：

$$k(\boldsymbol{x},\boldsymbol{q}) = \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} = \frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} \tag{37}$$

如图 23 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。

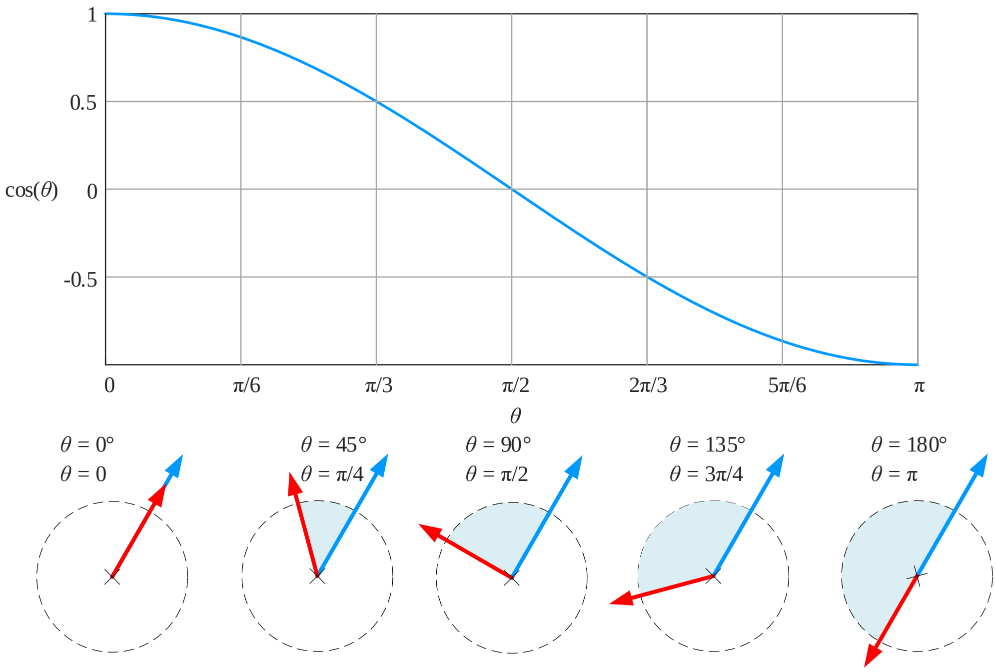


图 23. 余弦相似度

举个例子

给定如下两个向量具体值：

$$
\boldsymbol{x} = \begin{bmatrix} 8 & 2 \end{bmatrix}^{\mathrm{T}}, \quad \boldsymbol{q} = \begin{bmatrix} 7 & 9 \end{bmatrix}^{\mathrm{T}} \tag{38}
$$

将 (38) 代入 (37) 得到：

$$
k(\boldsymbol{x},\boldsymbol{q}) = \frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} = \frac{8\times 7+2\times 9}{\sqrt{8^2+2^2}\times\sqrt{7^2+9^2}} = \frac{74}{\sqrt{68}\times\sqrt{130}} = 0.7871 \tag{39}
$$

代码 Bk7_Ch08_11.ipynb 得到和 (39) 一致结果。

余弦距离

余弦距离 (cosine distance) 的定义如下：

$$
d(\boldsymbol{x},\boldsymbol{q}) = 1 - k(\boldsymbol{x},\boldsymbol{q}) = 1 - \frac{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} = 1 - \frac{\boldsymbol{x}\cdot\boldsymbol{q}}{\|\boldsymbol{x}\|\|\boldsymbol{q}\|} \tag{40}
$$

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 (38) 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：

$$k(\boldsymbol{x},\boldsymbol{q})=\frac{(\boldsymbol{x}-\bar{x})^{\mathrm{T}}(\boldsymbol{q}-\bar{q})}{\|\boldsymbol{x}-\bar{x}\|\|\boldsymbol{q}-\bar{q}\|}=\frac{(\boldsymbol{x}-\bar{x})\cdot(\boldsymbol{q}-\bar{q})}{\|\boldsymbol{x}-\bar{x}\|\|\boldsymbol{q}-\bar{q}\|} \tag{41}$$

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 (41)，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 (38) 为例，计算 $\boldsymbol{x}$ 和 $\boldsymbol{q}$ 两个向量的相关系数相似度。将 (38) 代入 (41) 可以得到：

$$\begin{aligned}k(\boldsymbol{x},\boldsymbol{q})&=\frac{\left(\begin{bmatrix}8 & 2\end{bmatrix}^{\mathrm{T}}-\frac{8+2}{2}\right)\cdot\left(\begin{bmatrix}7 & 9\end{bmatrix}^{\mathrm{T}}-\frac{7+9}{2}\right)}{\|\boldsymbol{x}-\bar{\boldsymbol{x}}\|\|\boldsymbol{q}-\bar{\boldsymbol{q}}\|}\\&=\frac{\begin{bmatrix}3 & -3\end{bmatrix}^{\mathrm{T}}\cdot\begin{bmatrix}-1 & 1\end{bmatrix}^{\mathrm{T}}}{\left\|\begin{bmatrix}3 & -3\end{bmatrix}^{\mathrm{T}}\right\|\left\|\begin{bmatrix}-1 & 1\end{bmatrix}^{\mathrm{T}}\right\|}=\frac{-6}{6}=-1\end{aligned} \tag{42}$$

代码 Bk7_Ch08_13.ipynb 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：

$$\kappa(\boldsymbol{x},\boldsymbol{q})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{q}=\boldsymbol{x}\cdot\boldsymbol{q} \tag{43}$$

对比 (37) 和 (43)，(37) 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 (38) 代入 (43)，得到线性核亲近度为：

$$\kappa(\boldsymbol{x},\boldsymbol{q})=8\times7+2\times9=74 \tag{44}$$

多项式核 (polynomial kernel) 亲近度定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \left(\gamma \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q} + r\right)^{d} = \left(\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r\right)^{d} \tag{45}$$

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：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \tanh\left(\gamma \boldsymbol{x}^{\mathrm{T}}\boldsymbol{q} + r\right) = \tanh\left(\gamma \boldsymbol{x}\cdot\boldsymbol{q} + r\right) \tag{46}$$

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)：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma \left\|\boldsymbol{x}-\boldsymbol{q}\right\|^{2}\right) \tag{47}$$

(47) 中 $\left\|\boldsymbol{x}-\boldsymbol{q}\right\|^{2}$ 为欧氏距离的平方，(47) 也可以写作：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma d^{2}\right) \tag{48}$$

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 24 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。

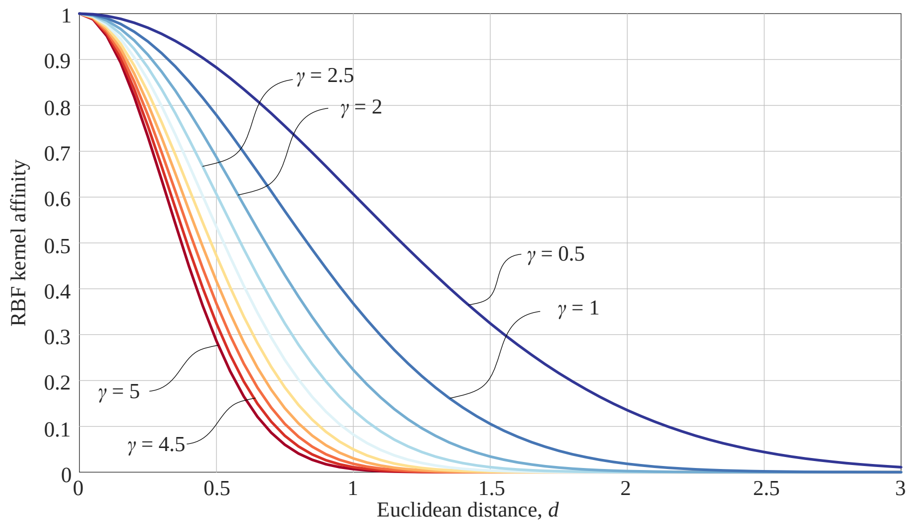


图 24. 高斯核亲近度随欧氏距离变化

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：

$$f_{\chi}(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}} = \frac{\exp\left(-\frac{1}{2}d^2\right)}{(2\pi)^{\frac{D}{2}}\left|\boldsymbol{\Sigma}\right|^{\frac{1}{2}}} \tag{49}$$

拉普拉斯核 (Laplacian kernel) 亲近度，定义如下：

$$\kappa(\boldsymbol{x},\boldsymbol{q}) = \exp\left(-\gamma\left\|\boldsymbol{x}-\boldsymbol{q}\right\|_1\right) \tag{50}$$

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

图 25 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`。

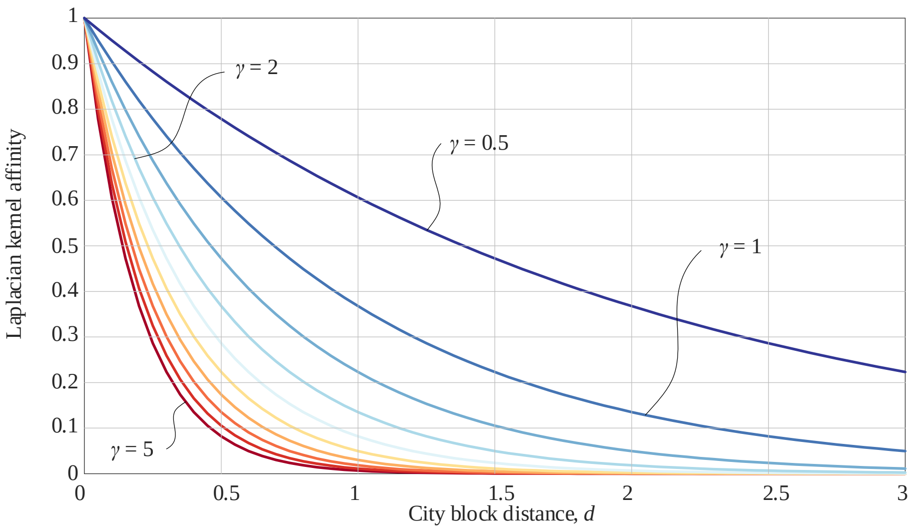


图 25. 拉普拉斯核亲近度随距离变化

8.16 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：

$$\boldsymbol{X}_{n\times D} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(n)} \end{bmatrix} \tag{51}$$

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

$$\boldsymbol{X}_{n\times D} = \begin{bmatrix} \boldsymbol{x}^{(1)\mathrm{T}} \\ \boldsymbol{x}^{(2)\mathrm{T}} \\ \vdots \\ \boldsymbol{x}^{(n)\mathrm{T}} \end{bmatrix} \tag{52}$$

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

$$\boldsymbol{D}_{n\times n} = \begin{bmatrix} 0 & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ d_{2,1} & 0 & d_{2,3} & \cdots & d_{2,n} \\ d_{3,1} & d_{3,2} & 0 & \cdots & d_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & d_{n,3} & \cdots & 0 \end{bmatrix} \tag{53}$$

每个样本数据点和自身的距离为 0，因此 (53) 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 26 给定 12 个样本数据点坐标点。

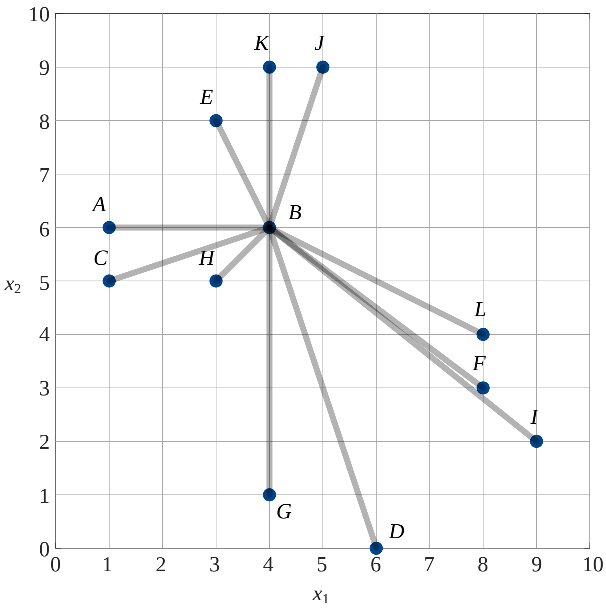


图 26. 样本数据散点图和成对距离

利用 `sklearn.metrics.pairwise.euclidean_distances`，我们可以计算图 26 数据点的成对欧氏距离矩阵。图 27 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：

$$C_n^2 = \frac{n(n-1)}{2} \tag{54}$$

也就是说，(53) 中不含对角线的下三角矩阵包含的信息足够使用。

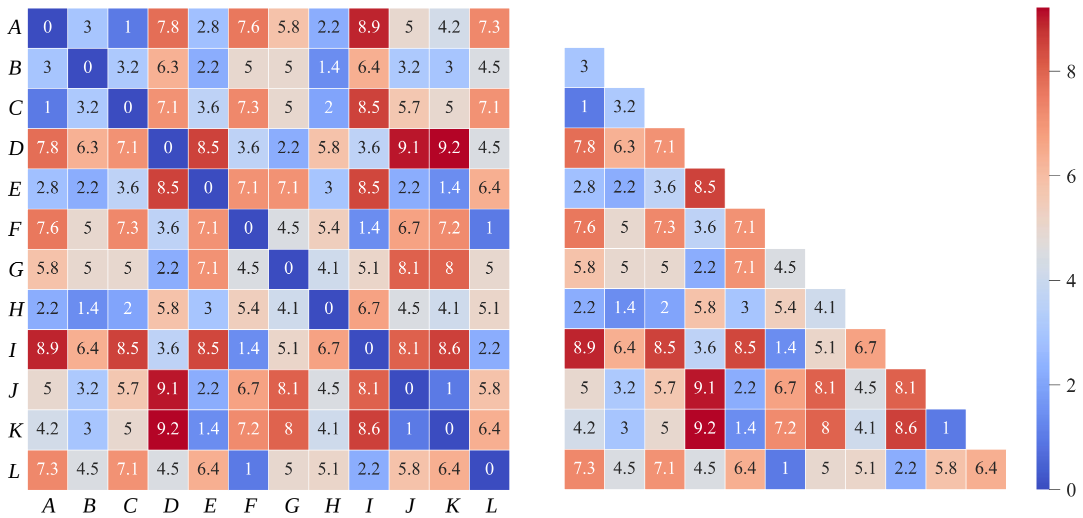


图 27. 样本数据成对距离矩阵热图

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_Ch08_14.ipynb 可以绘制图 26、图 27。

树形图

图 27 数据矩阵是很多机器学习算法的起点；看似杂乱无章的图 27，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 28 所示。计算日对数回报率，然后估算相关系数矩阵，如图 29 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。

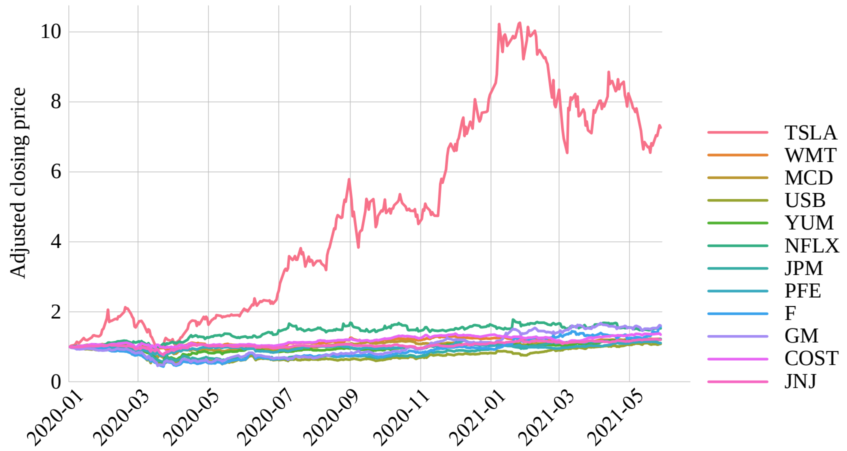


图 28. 12 只股票股价水平，初始股价归一化

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 30 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 30 树形图将股票按照相似度重新排列顺序。图 30 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。

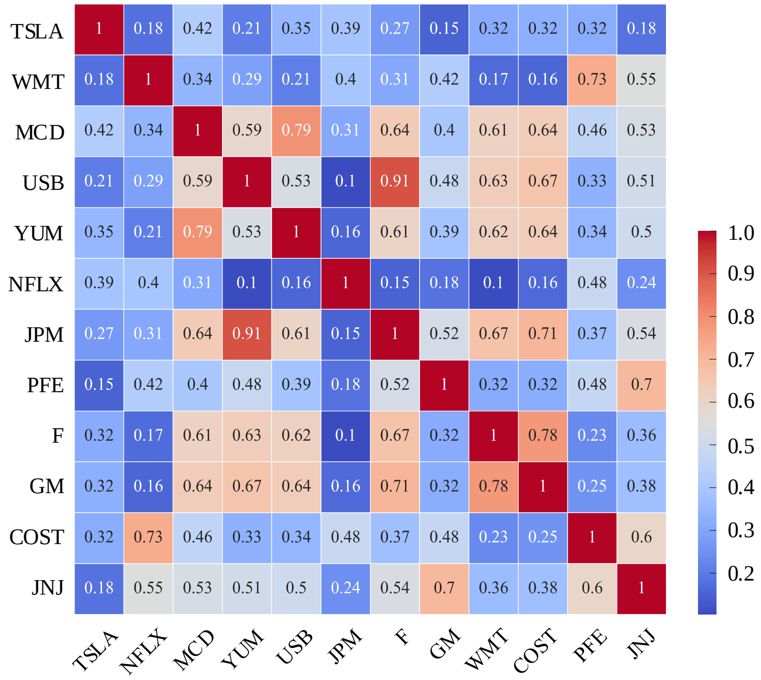


图 29. 12 只股票相关性热图

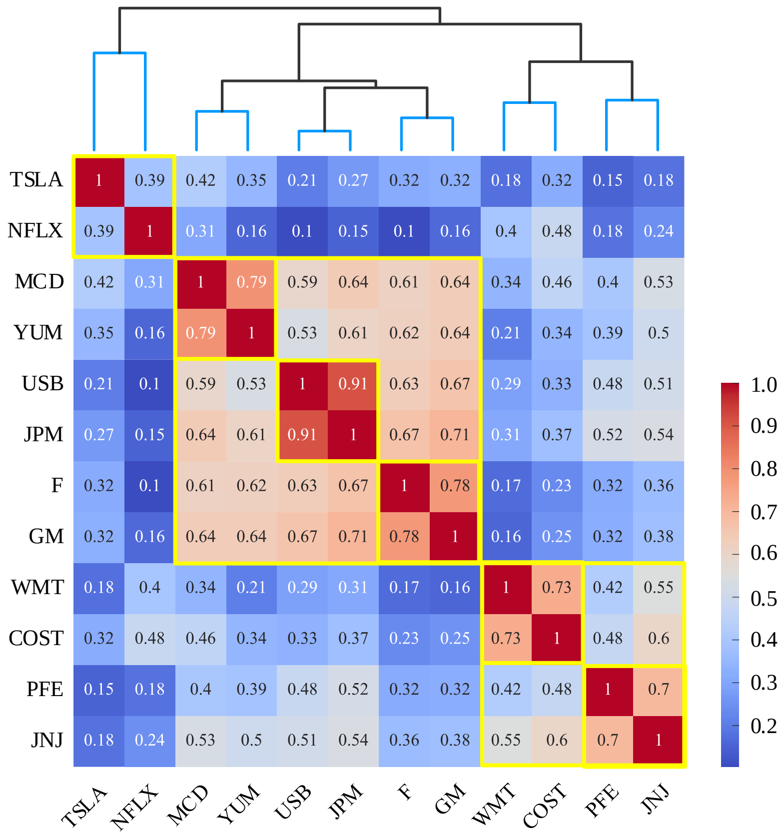


图 30. 根据树形图重组相关性热图

代码 Bk7_Ch08_15.ipynb 绘制图 28、图 29 和图 30。

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。

因此，本章还总结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。