

作者	生姜 DrGinger
脚本	生姜 DrGinger
视频	崔崔 CuiCui
开源学习资源	https://github.com/Visualize-ML
平台	https://www.youtube.com/@DrGinger_Jiang https://space.bilibili.com/3546865719052873 https://space.bilibili.com/513194466

02 Matrices 矩阵

不仅仅是个表格

2.1 矩阵



本节你将掌握的核心技能：

- ▶ 矩阵结构：二维数组，由行和列构成，用于数据存储、线性变换。
- ▶ 行列索引规则：行从上到下编号，列从左到右编号，NumPy 索引从 0 开始。
- ▶ 矩阵索引、分块：熟悉用 NumPy 获取行、列向量与单个元素的方法。
- ▶ 主对角线：主对角线元素位于行列索引相同位置。
- ▶ 矩阵加减法：两个形状相同的矩阵可逐元素加减，满足交换律与结合律。
- ▶ 标量乘法：矩阵与标量相乘是对每个元素放大或缩小。
- ▶ 张量：标量是 0 阶张量，向量是 1 阶，矩阵是 2 阶，彩色图像和视频为高阶张量。

矩阵是一种按照行、列排列的数值表格，广泛应用于数学、数据分析、机器学习、人工智能等领域。本节中，我们定义了矩阵的行、列结构，并通过实例说明了如何用矩阵表示数据。接着，本节讨论了矩阵的加法和减法，即对形状相同的矩阵进行逐元素相加或相减。随后，我们介绍了矩阵的标量乘法，即用一个数乘以矩阵中的每个元素。最后，本节还从张量角度把标量、向量、矩阵等概念联系起来。

什么是矩阵？

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

版权归清华大学出版社所有，请勿商用，引用请注明出处。

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

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

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

矩阵 (matrix) 是一个按照行、列排列的二维数组。形象来说，矩阵就是个充满数字的表格。

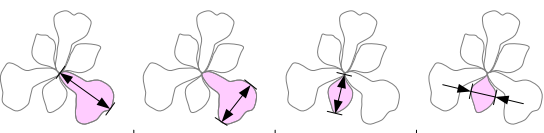
矩阵不仅仅是数学里的概念，还广泛出现在我们日常生活的各种数据之中。

比如，我们可以用矩阵来表示一张学生的成绩单

$$\begin{bmatrix} 98 & 90 & 78 \\ 88 & 96 & 80 \\ \vdots & \vdots & \vdots \\ 68 & 89 & 88 \end{bmatrix} \quad (1)$$

这张表格的每一**行** (row) 代表一个学生，每一**列** (column) 代表一个科目，矩阵中的数值为学生的成绩。

再如，图 1 中的鸢尾花数据集也可以看作是一个数据矩阵。



Index	Sepal length X_1	Sepal width X_2	Petal length X_3	Petal width X_4	Species C
1	5.1	3.5	1.4	0.2	Setosa C_1
2	4.9	3	1.4	0.2	
3	4.7	3.2	1.3	0.2	
...	
49	5.3	3.7	1.5	0.2	
50	5	3.3	1.4	0.2	Versicolor C_2
51	7	3.2	4.7	1.4	
52	6.4	3.2	4.5	1.5	
53	6.9	3.1	4.9	1.5	
...	
99	5.1	2.5	3	1.1	Virginica C_3
100	5.7	2.8	4.1	1.3	
101	6.3	3.3	6	2.5	
102	5.8	2.7	5.1	1.9	
103	7.1	3	5.9	2.1	
...	
149	6.2	3.4	5.4	2.3	
150	5.9	3	5.1	1.8	

图 1. 鸢尾花数据，数值数据单位为厘米 (cm)，图片来自《编程不难》

数据集的全称为**安德森鸢尾花数据集** (Anderson's Iris data set)，是植物学家**埃德加·安德森** (Edgar Anderson) 在加拿大魁北克加斯帕半岛上的采集的鸢尾花样本数据。

数据集中每一行代表一朵鸢尾花样本，前 4 列代表鸢尾花的四个特征：**萼片长度** (sepal length)、**萼片宽度** (sepal width)、**花瓣长度** (petal length)、**花瓣宽度** (petal width)。

图 1 中最后一列为鸢尾花品种。鸢尾花数据集共有三个不同品种——**山鸢尾** (Setosa)、**变色鸢尾** (Versicolor)、**维吉尼亚鸢尾** (Virginica)。数据集一共有 150 个样本，每个品种 50 个样本。

图 2 是作者拍摄的一张鸢尾花照片。将这张照片转换为黑白模式后，它可以表示为一个大小为 2990×2714 的矩阵，即 2990 行、2714 列，其中每个元素代表一个像素的灰度值。

图 2 这张照片显然不是矢量图。

如果我们不断放大，会发现图像的边缘逐渐变得模糊。继续放大，就能看到照片实际上是由一个个灰度像素点组成的热图。

再进一步，若从中提取 4 个像素点，它们对应于矩阵中的 4 个元素，这样我们就得到了一个 2×2 的实数矩阵。

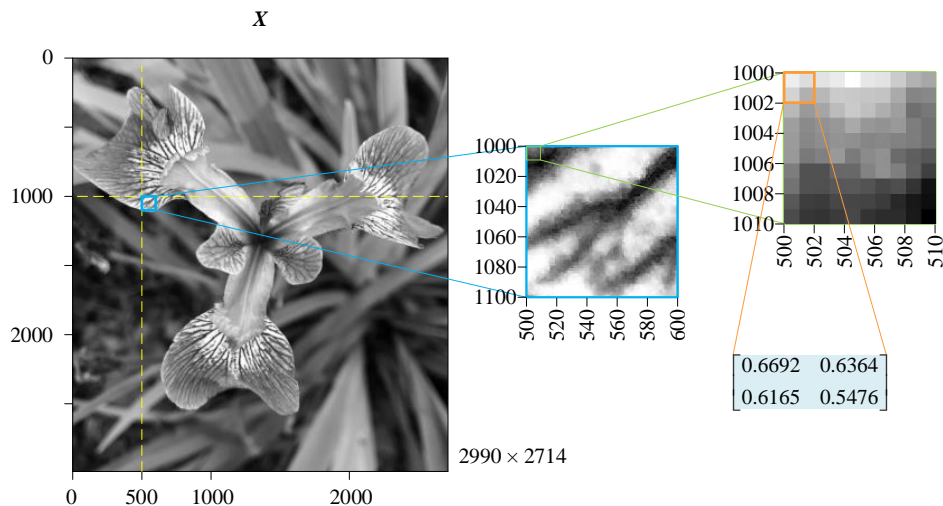


图 2. 照片也是数据矩阵，图片来源《矩阵力量》

但矩阵不仅仅是存放数据的普通表格，这一点大家很快就会看到。

矩阵的形状

本书中，如果一个矩阵 A 的形状标记为 $m \times p$ ，其中 m 代表**行数** (number of rows)， p 代表**列数** (number of columns)。

这个矩阵也常记做 $A_{m \times p}$ ，形状在下角标中。 $m \times p$ 的乘积就是**矩阵元素** (entry, component, element) 的数量。

如图 3 所示，本书中矩阵每个元素都会用方方正正的格子可视化呈现。

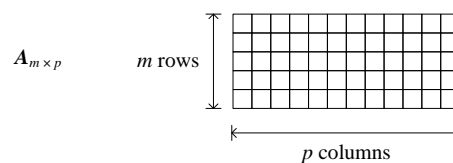


图 3. 矩阵 A 的形状

打个比方，如下 2×3 矩阵 A 就像一座 2 层楼高、每层 3 个房间的数字大厦；而 3×2 矩阵 B 则有 3 层楼高，每层 2 个房间。

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \quad (2)$$

⚠ 注意，先行后列这个顺序。这种标记方式暗示了矩阵的“生长方向”。

上一章提过，**行向量** (row vector)、**列向量** (column vector) 都是矩阵的特殊形状。

行向量是一行、多列的矩阵，列向量则是多行、一列的矩阵。

方阵 (square matrix) 则是行数、列数相同的矩阵。

行

行索引 (row index) 用来定位矩阵行的位置。

如图 4 所示，我们用 i 来表达矩阵 \mathbf{A} 的行索引； i 从上到下递增，取值范围是 1 到 m 的正整数。

本书中，矩阵 \mathbf{A} 的第 1 行记作 $\mathbf{a}^{(1)}$ ，第 2 行记作 $\mathbf{a}^{(2)}$ ，第 i 行记作 $\mathbf{a}^{(i)}$ 。

⚠ 再次强调，行索引从上到下递增。

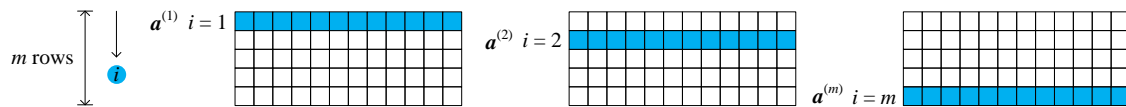


图 4. 矩阵 \mathbf{A} 的行索引

换个角度来看，如图 5 所示，矩阵可以看作由一组有序行向量构成。

比如，矩阵 \mathbf{A} 可以写成

$$\mathbf{A}_{m \times p} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,p} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,p} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,p} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,p} \end{bmatrix} \\ \begin{bmatrix} a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,p} \end{bmatrix} \\ \begin{bmatrix} a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,p} \end{bmatrix} \\ \vdots \\ \begin{bmatrix} a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,p} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \\ \mathbf{a}^{(3)} \\ \vdots \\ \mathbf{a}^{(m)} \end{bmatrix} \quad (3)$$

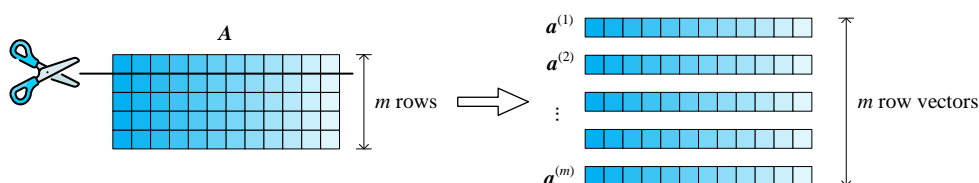


图 5. 把矩阵 \mathbf{A} 切成一组行向量

比如，把矩阵 2×3 矩阵 A 、 3×2 矩阵 B 分别写成一组行向量

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} [1 & 2 & 3] \\ [4 & 5 & 6] \end{bmatrix} = \begin{bmatrix} \mathbf{a}^{(1)} \\ \mathbf{a}^{(2)} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} [1 & 4] \\ [2 & 5] \\ [3 & 6] \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \mathbf{b}^{(3)} \end{bmatrix} \quad (4)$$

这实际上体现的是**分块矩阵** (block matrix, partitioned matrix) 思想。分块矩阵是将一个大矩阵按行列划分成多个小矩阵的形式，每个小矩阵称为**块** (block)，或**子矩阵** (submatrix)。本章后续会展开讲解分块矩阵。

代码 1 创建二维数组，相当于矩阵；然后提取如图 6 所示的行向量。

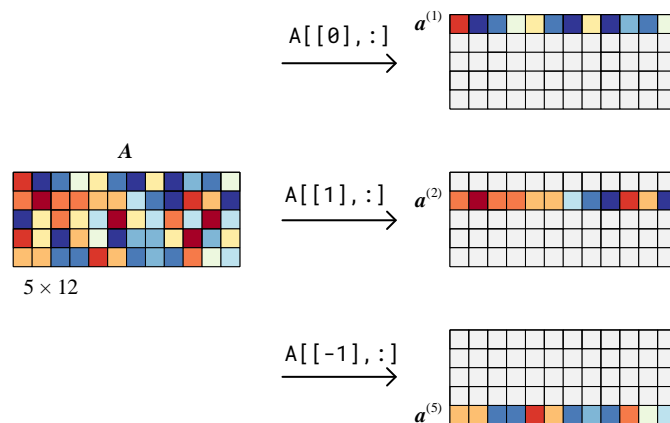


图 6. 取出行向量

然我们聊聊代码 1 的核心语句。

- a** 用 `numpy.random.seed(88)` 的作用是固定随机数的生成，使得每次运行代码时产生的随机数相同。88 是种子值，可以是任何整数，不同的种子会导致不同的随机数序列。
- b** 用 `numpy.random.randint(0, 10, (5, 12))` 生成一个 5×12 的整数矩阵，矩阵中的数值是 0 到 9 之间的随机整数（注意，不包括 10）。(5, 12) 指定了矩阵的形状，其中 5 代表行数，12 代表列数。
- c** 用 `seaborn.heatmap()` 绘制热图可视化矩阵。
参数 `cmap = 'RdYlBu_r'` 设置颜色映射；'RdYlBu_r' 代表从红色到黄色再到蓝色，并进行了反转 (r)。
- 参数 `square = True` 使得每个单元格保持正方形。
- 参数 `linecolor='w'` 设置单元格之间的网格线颜色为白色。
- 参数 `linewidths=0.25` 设定网格线的宽度为 0.25。
- d** 提取矩阵 A 的第一行，并赋值给变量 `a_row_1`。索引 [0] 代表第一行，: 代表所有列。注意 `A[[0], :]` 返回的是一个 2D 数组，形状是 (1, 12)，而不是 1D 数组。
- e** `A[0, :]` 同样获取第一行的所有列数据，但返回的是一个 1D 数组，形状是 (12,)。
- f** 提取矩阵 A 的第二行，并赋值给 `a_row_2`，返回的也是一个 2D 数组。
- g** 用负索引 -1 代表最后一行，因此这行代码提取矩阵 A 的最后一行，并赋值给 `a_row_5`，返回的也是一个形状为 (1,12) 的二维数组。

代码 1. 二维数组提取行 | LA_02_01_01.ipynb

```

## 初始化
import numpy as np
import seaborn as sns

## 创建矩阵
a np.random.seed(88)
b A = np.random.randint(0, 10, (5, 12))

## 可视化
c sns.heatmap(A, cmap = 'RdYlBu_r', square = True,
               linecolor = 'w', linewidths = 0.25)

## 行索引
d a_row_1 = A[[0], :]
e A[0, :]

f a_row_2 = A[[1], :]

g a_row_5 = A[[-1], :]

```

列

类似地，**列索引** (column index) 用来定位矩阵列的位置。

如图 7 所示，用 k 来表达矩阵 A 的列索引， k 从左到右递增，取值范围是 1 到 p 的正整数。

本书中，矩阵 A 的第 1 列记作 a_1 ，第 2 列记作 a_2 ，第 k 列记作 a_k 。

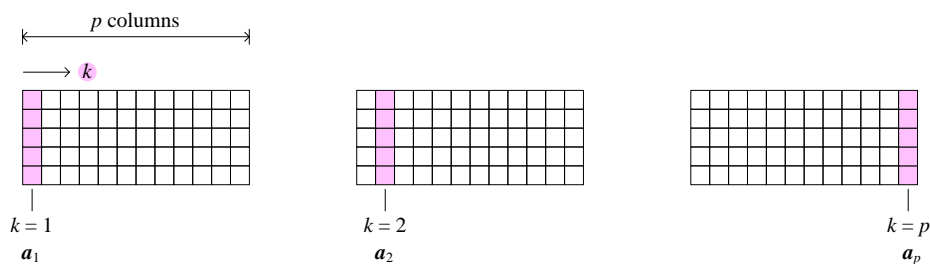
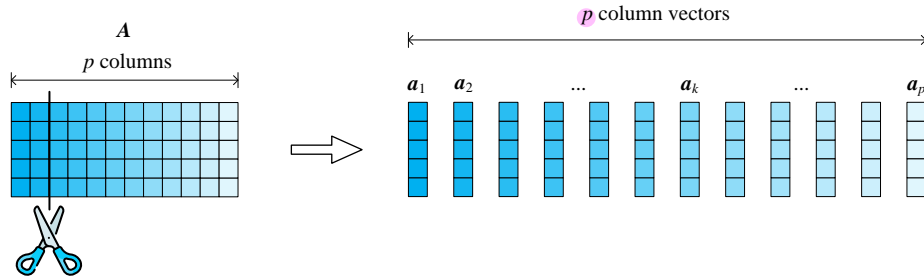


图 7. 矩阵 A 的列索引

⚠ 再次强调，在数学的世界里，矩阵的行列索引从 1 开始依次递增。然而，使用 NumPy 这样的科学计算库时，索引的起点却悄然发生了变化——它们从 0 开始计数！

同样地，如图 8 所示，矩阵可以看作由一组有序行向量构成。比如，矩阵 A 可以写成

$$A_{m \times p} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,p} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,p} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,p} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \\ \vdots \\ a_{m,1} \end{bmatrix} & \begin{bmatrix} a_{1,2} \\ a_{2,2} \\ a_{3,2} \\ \vdots \\ a_{m,2} \end{bmatrix} & \begin{bmatrix} a_{1,3} \\ a_{2,3} \\ a_{3,3} \\ \vdots \\ a_{m,3} \end{bmatrix} & \cdots & \begin{bmatrix} a_{1,p} \\ a_{2,p} \\ a_{3,p} \\ \vdots \\ a_{m,p} \end{bmatrix} \end{bmatrix} = [a_1 \ a_2 \ a_3 \ \cdots \ a_p] \quad (5)$$

图 8. 把矩阵 A 切成一组列向量

举个例子，把矩阵 2×3 矩阵 A 、 3×2 矩阵 B 分别写成一组列向量

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} & \begin{bmatrix} 2 \\ 5 \end{bmatrix} & \begin{bmatrix} 3 \\ 6 \end{bmatrix} \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} & \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} \end{bmatrix} \quad (6)$$

这体现的也是矩阵分块的思想。



LA_02_01_01.ipynb 给出了如何提取矩阵 A 的列向量，具体如图 9 所示。

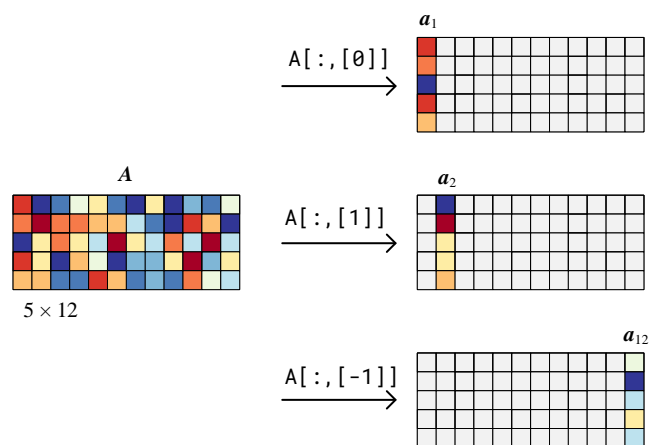


图 9. 取出列向量

矩阵的元素

有了行、列索引，我们就可以轻松标记矩阵元素的位置。

一般来说矩阵 A 第 i 行、第 k 列元素记做 $a_{i,k}$ ；这样的话，矩阵 A 可以写成

$$A_{m \times p} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,p} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,p} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,p} \end{bmatrix} \quad (7)$$

比如说，矩阵 A 第 1 行、第 1 列的元素记作 $a_{1,1}$ ；再如，矩阵 A 第 2 行、第 3 列元素记作 $a_{2,3}$ 。



LA_02_01_01.ipynb 还给出了如何获取矩阵 A 的元素，具体如图 10 所示。

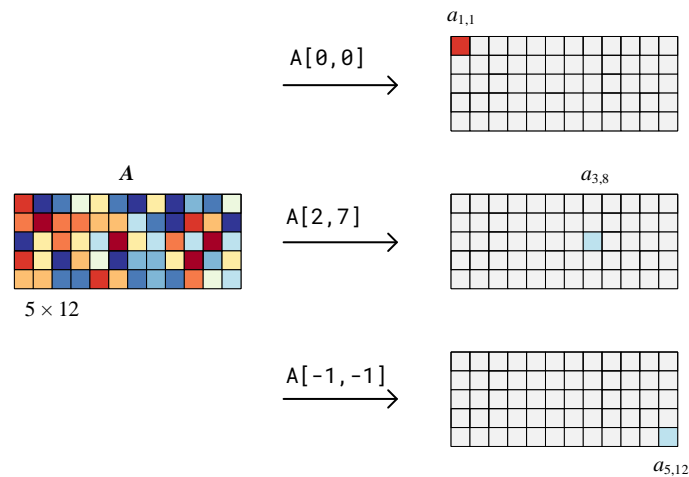


图 10. 取出元素

主对角元、主对角线

任何一个矩阵都有一组特殊元素——**主对角元** (diagonal element)——它们的行、列索引相同。

比如，图 11 中的蓝色元素对应矩阵 A 中的主对角元 $a_{1,1}$ 、 $a_{2,2}$ 、 $a_{3,3}$ 等等。

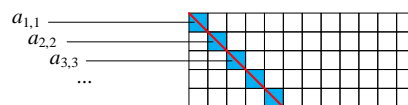


图 11. 矩阵 A 的主对角元

主对角元的连线叫做**主对角线** (diagonal, principal diagonal, main diagonal, leading diagonal)，即图 11 中的红线。

几何角度来看，如果把矩阵的每个元素看作一个大小相同的正方形格子，那么**主对角线**就是从矩阵左上角的第一个格子出发，以 45 度的倾角向右下方延伸的一条线段。

主对角线依次穿过每个行索引、列索引相同的格子，就像一条贯穿矩阵的整齐阶梯。

无论矩阵的形状如何变化，主对角线始终存在。图 12 中红色线为细高矩阵、扁平矩阵、方阵、行向量、列向量中的主对角线。

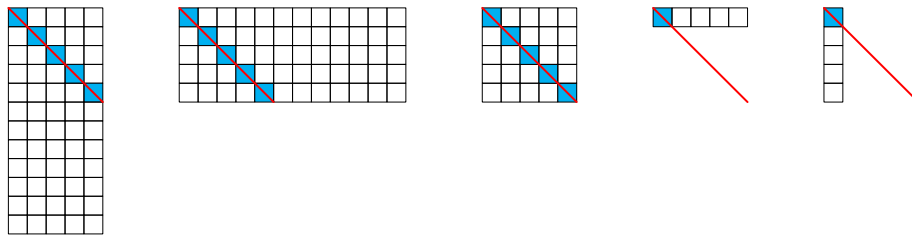


图 12. 各种形状的矩阵都有自己的主对角线

如图 13 所示，函数 `numpy.diag()` 可以用来提取矩阵 A 的对角线元素，结果为一维数组。

函数 `numpy.diag()` 还可以用于创建**对角方阵** (square diagonal matrix)。当传入一个一维数组时，它会生成一个以该数组元素为对角线元素的方阵，其他元素填充为零。

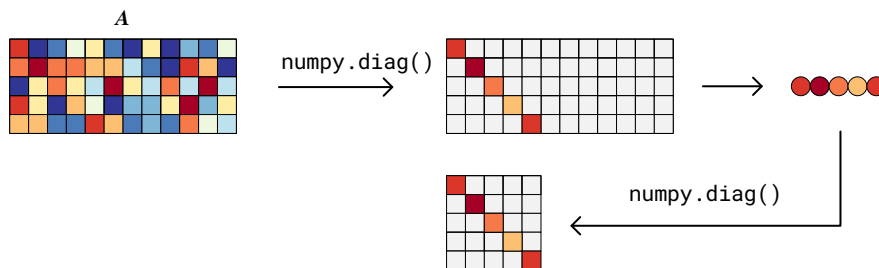


图 13. 提取对角线元素，结果为一维数组

矩阵加减

两个形状相同的矩阵 A 、 B 相加，即对它们对应位置的元素逐一相加，具体如下：

$$A_{m \times p} + B_{m \times p} = \begin{bmatrix} a_{1,1} + b_{1,1} & a_{1,2} + b_{1,2} & \dots & a_{1,p} + b_{1,p} \\ a_{2,1} + b_{2,1} & a_{2,2} + b_{2,2} & \dots & a_{2,p} + b_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} + b_{m,1} & a_{m,2} + b_{m,2} & \dots & a_{m,p} + b_{m,p} \end{bmatrix}_{m \times p} \quad (8)$$

矩阵**加法交换律** (commutative property) 指的是对于两个形状相同的矩阵 A 、 B ，它们的加法满足以下关系：

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

版权归清华大学出版社所有，请勿商用，引用请注明出处。

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

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

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

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} \quad (9)$$

即，无论先加哪个矩阵，结果始终相同。

矩阵**加法结合律** (associative property) 指的是，对于相同大小的三个矩阵 \mathbf{A} 、 \mathbf{B} 、 \mathbf{C} ，它们的加法满足以下关系：

$$\mathbf{A} + \mathbf{B} + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C} \quad (10)$$

即，先对任意两个矩阵求和，再与第三个矩阵相加，结果始终相同。

上述规则也适用于矩阵减法，本书不再赘述。

本书用 \mathbf{O} 表示元素全为 0 的矩阵，即**零矩阵** (zero matrix)。

零矩阵具有以下性质：

$$\begin{aligned} \mathbf{A} + \mathbf{O} &= \mathbf{O} + \mathbf{A} = \mathbf{A} \\ \mathbf{A} - \mathbf{A} &= \mathbf{O} \end{aligned} \quad (11)$$

上式中， \mathbf{A} 和 \mathbf{O} 形状相同。即，任意矩阵与和其形状相同的零矩阵相加，结果仍然是原矩阵。

矩阵标量乘法

矩阵的**标量乘法** (scalar multiplication) 指的是用一个标量乘以矩阵中的每一个元素。

比如，标量 k 和矩阵 \mathbf{A} 的乘积记作 $k\mathbf{A}$ ：

$$k\mathbf{A} = \begin{bmatrix} k \cdot a_{1,1} & k \cdot a_{1,2} & \cdots & k \cdot a_{1,p} \\ k \cdot a_{2,1} & k \cdot a_{2,2} & \cdots & k \cdot a_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ k \cdot a_{m,1} & k \cdot a_{m,2} & \cdots & k \cdot a_{m,p} \end{bmatrix} \quad (12)$$

矩阵 \mathbf{A} 的形状为 $m \times p$ ，标量乘法的结果的形状还是 $m \times p$ ；结果中的每个元素都被放大或缩小。

特别地，当 $k = 0$ 时，上式的结果为零矩阵 \mathbf{O} ，形状为 $m \times p$ 。



LA_02_01_02.ipynb 介绍如何用 NumPy 完成矩阵加减、数乘运算，代码很简单，请大家自行学习。

张量

张量是数学中的多维数组，描述了多种维度上的数值数据。

根据张量的**阶** (order)，它可以表达不同复杂程度的信息：

► **0 阶张量** (zero-order tensor) 便是**标量** (scalar)。

- ▶ **1 阶张量** (first-order tensor) 就是**向量** (vector)。
- ▶ **2 阶张量** (second-order tensor) 对应**矩阵** (matrix)。
- ▶ 3 阶及以上张量为多维数组。

还是用前文照片那个例子。0 阶张量可以是黑白照片中一个像素的灰度值，比如 0.5。

1 阶张量的例子可以是一个 RGB 颜色，用列向量可以表示为 $[R, G, B]^T$ 。

一个 D 维度空间可以表达为 \mathbb{R}^D 。比如，二维平面可以记做 \mathbb{R}^2 ，三维空间可以记做 \mathbb{R}^3 。

鸢尾花数据集 4 个特征所在的空间记作 \mathbb{R}^4 ，150 个样本所在的空间为 \mathbb{R}^{150} 。

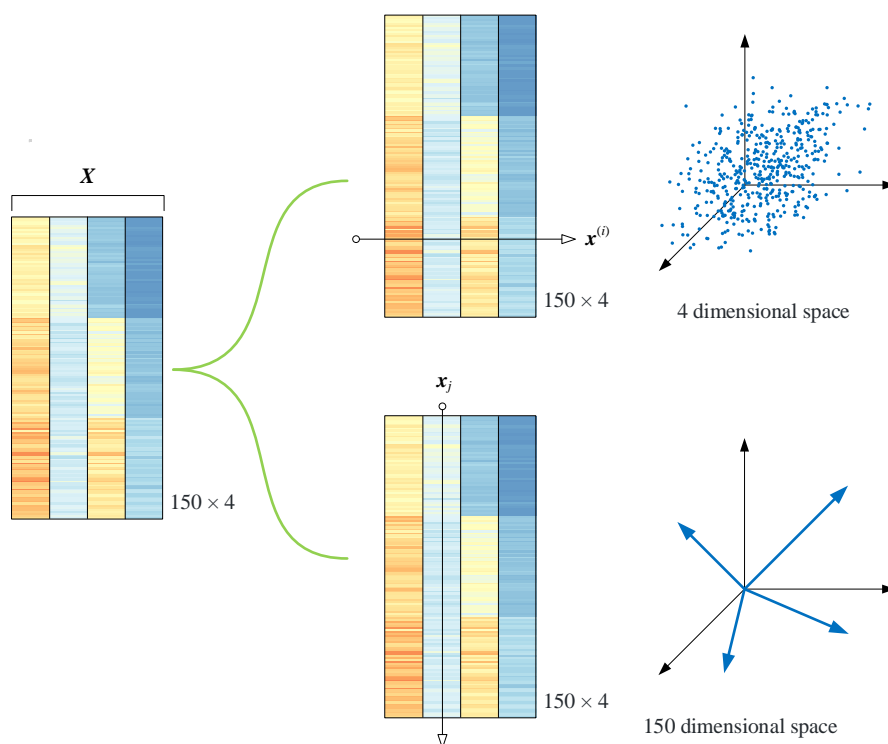


图 14. 从行、列视角看鸢尾花数据，图片来自《矩阵力量》

举个例子，黑白照片的数据便是一个 2 阶张量，两个维度是照片的高度 H 和宽度 W ，可以记做 $\mathbb{R}^{H \times W}$ 。再如，一个 2 行 3 列矩阵则属于 $\mathbb{R}^{2 \times 3}$ 。

如图 15 所示，一张彩色照片则包含了 R、G、B 三个通道的信息，这便是 3 阶张量。相比黑白照片，彩色照片在高度 H 和宽度 W ，增加了一个颜色通道 C ，可以记做 $\mathbb{R}^{C \times H \times W}$ 。彩色照片每个颜色切片则是一个 2 阶张量。

而视频可以看作连续彩色图像的时间序列，是一个 4 阶张量，进一步增加了一个时间 (帧) 维度 T ，可以记作 $\mathbb{R}^{T \times C \times H \times W}$ 。

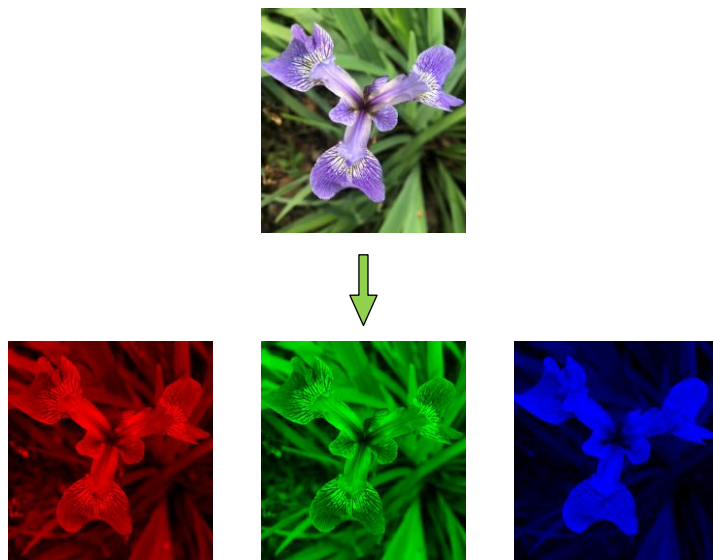


图 15. 鸢尾花彩色照片（3 阶张量）分解成红、绿、蓝三个通道（分别是 2 阶张量）

张量是现代数据分析和机器学习的核心工具，其多维数据的表示能力为模型构建和复杂计算奠定了坚实的基础。作为标量、向量、矩阵的自然扩展，张量能够灵活地描述任意维度的数据结构，从单一数值到多通道图像、视频序列甚至多模态数据。机器学习中的许多核心要素，如神经网络的权重矩阵、图像的像素通道表示、自然语言处理中的词嵌入向量等，都是以张量的形式存在的。

通过张量，复杂的数学操作如矩阵乘法、卷积运算得以高效实现，并能在 GPU 等硬件上进行并行化计算，从而显著提高处理大规模数据的能力。无论是在计算机视觉、自然语言处理还是时间序列分析中，张量都以其强大的表达能力和计算性能，推动了机器学习从理论到实际的广泛应用，成为构建现代人工智能系统不可或缺的基础工具。



请大家用 DeepSeek/ChatGPT 等工具完成本节如下习题。

Q1. 学习使用 `seaborn.heatmap()` 绘制热图。

<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

Q2. 什么是 Pandas 数据帧 (DataFrame)? 和 NumPy 数组有什么区别联系?

Q3. 如何用 NumPy 二维数组构造 Pandas DataFrame?

Q4. 用随机数发生器生成一个 8 行、18 列的数组，用 `seaborn.heatmap()` 对其可视化。

Q5. 请提取 Q4 数组的

- ▶ 第 1 行 (结果为二维数组);
- ▶ 第 5 行 (结果为一维数组);
- ▶ 最后一行 (结果为二维数组);
- ▶ 第 1 列 (结果为一维数组);
- ▶ 第 5 列 (结果为二维数组);

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

版权归清华大学出版社所有，请勿商用，引用请注明出处。

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

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

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

- ▶ 最后一列 (结果为二维数组);
- ▶ 第 1 行、第 1 列元素;
- ▶ 第 2 行、第 2 列元素;
- ▶ 第 3 行、第 3 列元素;
- ▶ 第 8 行、第 8 列元素;
- ▶ 所有对角线元素;

Q6. 用随机数发生器生成两个 3 行、8 列的数组，并求解两者和、差。

Q7. 自学 NumPy 的广播机制 (broadcasting)。

<https://numpy.org/doc/stable/user/basics.broadcasting.html>

Q8. 用随机数发生器生成一个 8 行、8 列的数组，计算每列的最大值、最小值、平均值。

Q9. 请学习使用以下几个 NumPy 函数，并解释每个函数的作用。

- ▶ `numpy.flatten()`
- ▶ `numpy.column_stack()`
- ▶ `numpy.row_stack()`
- ▶ `numpy.flip()`
- ▶ `numpy.fliplr()`
- ▶ `numpy.flipud()`
- ▶ `numpy.reshape()`

Q10. 请自学矩阵逐项积 (Hadamard product, element-wise product)。