4.1 聊聊 NumPy

Fundamentals of NumPy

本节的核心是用 NumPy 产生不同类型数组

重要的不是生命的长度，而是深度。

It is not the length of life, but the depth.

—— 拉尔夫·沃尔多·爱默生 (Ralph Waldo Emerson) | 美国思想家、文学家 | 1942 ~ 2018

什么是 Numpy?

NumPy 是 Python 科学计算中非常重要的一个库，它提供了快速、高效的多维数组对象及其操作方法，是众多其他科学计算库的基础。

NumPy 最重要的功能之一是提供了高效的多维数组对象 ndarray，可以用来表示向量、矩阵和更高维的数组。它是 Python 中最重要的科学计算数据结构，支持广泛的数值运算和数学函数操作。

此外，如果大家需要处理有标签、多维数组数据的话，推荐使用 xarray。xarray 可以看作是在 NumPy 的基础上，增加了标签和元数据的功能。xarray 可以对多个数组进行向量化计算，避免了循环操作，提高了计算效率。xarray 提供了多种统计分析函数，可以方便地对多维数组数据进行统计分析。本书将不会展开讲解 xarray。

NumPy 提供了多种数组操作方法，包括数组索引、切片、迭代、转置、变形、合并等，以及广播 (broadcasting) 机制，使得数组操作更加方便、高效。这些话题是本书后续要展开讲解的内容。本书后文会专门讲解广播。

NumPy 提供了丰富的数学函数库，包括三角函数、指数函数、对数函数、逻辑函数、统计函数、随机函数等，能够满足大多数科学计算需要。

“鸢尾花书”中《数学要素》一册将大量使用这些函数库来可视化常见函数。

NumPy 支持多种文件格式的读写操作，包括文本文件、二进制文件、CSV 文件等。NumPy 基于 C 语言实现，因此可以利用底层硬件优化计算速度，同时还支持多线程、并行计算和向量化操作，使得计算更加高效。

NumPy 提供了丰富的线性代数操作方法，包括矩阵乘法、求逆矩阵、特征值分解、奇异值分解等，可以方便地解决线性代数问题。

本书中会简要介绍这些常见线性代数操作，详细讲解请大家参考“鸢尾花书”中的《矩阵力量》一册。

NumPy 可以于 Matplotlib 库集成使用，方便地生成各种图表，如线图、散点图、柱状图等。相信大家在本书前文已经看到基于 NumPy 数据绘制的平面、三维图像。

NumPy 提供了一些常用的数据处理方法，如排序、去重、聚合、统计等，方便对数据进行预处理。即便如此，“鸢尾花书”中我们更常用 Pandas 处理数据，本书后续将专门介绍 Pandas。

Python 中许多数据分析和机器学习的库都是基于 NumPy 创建。Scikit-learn 是一个流行的机器学习库，它基于 NumPy、SciPy 和 Matplotlib 创建，提供了各种机器学习算法和工具，如分类、回归、聚类、降维等。PyTorch 是一个开源的机器学习框架，它基于 NumPy 创建，提供了张量计算和动态计算图等功能，可以用于构建神经网络和其他机器学习算法。TensorFlow 是一个深度学习框架，它基于 NumPy 创建，提供了各种神经网络算法和工具，包括卷积神经网络、循环神经网络等。

“鸢尾花书”中的《数据有道》专门讲解回归、降维这两类机器学习算法，而《机器学习》一册则侧重分类、聚类。

从 numpy.array() 说起

我们可以利用 **numpy.array**() 手动生成一维、二维、三维等数组。下面首先介绍如何使用 **numpy.array**() 这个函数。

numpy.array(object, dtype)

这个函数的重要输入参数：

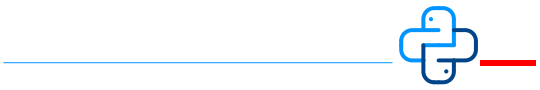
- object 转换为数组的输入数据，可以是列表、元组、其他数组或类似序列的对象。
- dtype 参数用于指定数组的数据类型。如果不指定 dtype 参数，则 NumPy 会自动推断数组的数据类型。

请大家在 JupyterLab 中自行学习下例。

```
import numpy as np

# 从列表中创建一维数组
arr1 = np.array([1, 2, 3, 4])

# 指定数组的数据类型
arr2 = np.array([1, 2, 3, 4], dtype=float)

# 从元组中创建二维数组
arr3 = np.array([(1, 2, 3), (4, 5, 6)])

# 指定最小维度
arr4 = np.array([1, 2, 3, 4], ndmin=2)
```

NumPy 中的 array 是什么？

在 NumPy 中，array 是一种多维数组对象，它可以用于表示和操作向量、矩阵和张量等数据结构。array 是 NumPy 中最重要的数据结构之一，它支持高效的数值计算和广播操作，可以用于处理大规模数据集和科学计算。与 Python 中的列表不同，array 是一个固定类型、固定大小的数据结构，它可以支持多维数组操作和高性能数值计算。array 的每个元素都是相同类型的，通常是浮点数、整数或布尔值等基本数据类型。在创建 array 时，用户需要指定数组的维度和类型。例如，可以使用 numpy.array() 函数创建一个一维数组或二维数组，也可以使用 numpy.zeros() 函数或 numpy.ones() 函数创建指定大小的全 0 或全 1 数组，还可以使用 numpy.random 模块生成随机数组等。除了基本操作之外，NumPy 还提供了许多高级的数组操作，例如数组切片、数组索引、数组重塑、数组转置、数组拼接和分裂等。



本节配套的 Jupyter Notebook 文件 BK_2_Topic_4.01_1.ipynb，请大家边读正文边在 Notebook 中探究学习。

手动生成一维数组

在 NumPy 中，一维数组是最基本的数组类型，也被称为一维 ndarray。它只有一个维度，并且可以包含多个元素，其中每个元素都是相同的数据类型。

图 1 所示为利用 **numpy.arange**() 生成的一维数组。这个数组的形状为 (7,)，长度为 7，维度为 1。和本书前文介绍的 list 一样，NumPy 数组的索引也是从 0 开始。下一话题专门讲解 NumPy 数组索引和切片。再次强调，如图 1 所示，本书可视化一维数组时用圆形。

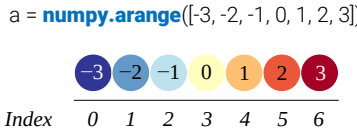


图 1. 手动生成一维数组

下面区分一下形状、长度、维度、大小这四个特征：

- 形状：可以使用 **shape** 属性来获取数组的形状，即每个维度上的大小，例如，如果数组 arr 是一个二维数组，则可以使用 arr.**shape** 来获取其形状。
- 长度：可以使用 **len**()函数来获取数组的长度，例如，如果数组 arr 是一个一维数组，则可以使用 **len**(arr)来获取其长度。
- 维数：可以使用 **ndim** 属性来获取数组的维数，例如，如果数组 arr 是一个二维数组，则可以使用 arr.**ndim** 来获取其维数。
- 大小：可以使用 **size** 属性来获取数组的大小，即所有元素的个数，例如，如果数组 arr 是一个二维数组，则可以使用 arr.**size** 来获取其大小。

手动生成二维数组

图 2 所示为利用 **numpy.array**() 生成的二维数组。利用 V 方法，大家可以发现图 2 中数组的维度都是 2。此外，**numpy.matrix**() 专门用来生成二维矩阵，请大家自行学习。

⚠ 请大家注意图 2 中中括号 [] 的数量。特别强调，本书中，行向量、列向量都被视作特殊的二维数组。也就是说，行向量是一行多列矩阵，而列向量是多行一列矩阵。

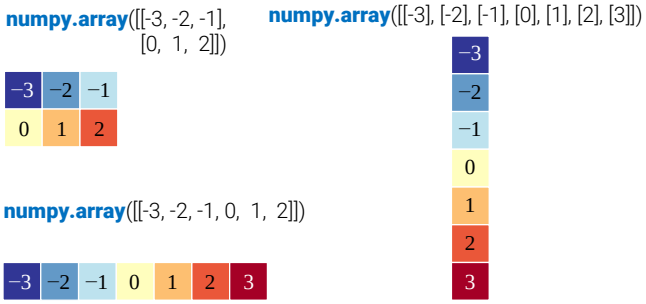


图 2. 手动生成二维数组

手动生成三维数组

图 3 所示为利用 **numpy.array**() 生成的三维数组，这个数组的形状为 (2, 3, 4)，也就是 2 页、3 行、4 列。Jupyter Notebook 文件展示如何获取三维数组的第 0 页和第 1 页。

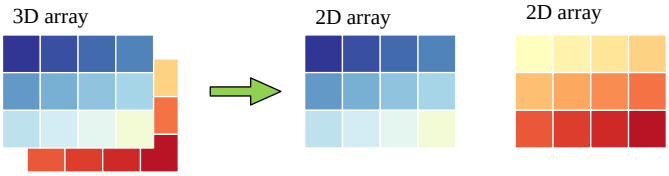


图 3. 手动生成三维数组

数列

在 NumPy 中我们常用以下三个函数生成数列：

- **numpy.arange**(start, stop, step)。生成等差数列，从起始值 start 开始，以步长 step 递增，直到结束值 stop (不包含 stop)。例如，**numpy.arange**(1, 11, 2) 将生成一个等差数列 [1, 3, 5, 7, 9]。实际上，**numpy.arange**() 和前文介绍的 **range**() 函数颇为相似。
- **numpy.linspace**(start, stop, num, endpoint)。生成等间距数列，从起始值 start 开始，到结束值 stop 结束，num 指定数列的长度 (元素的个数)，默认为 50。endpoint 参数指定是否包含结束值。例如，numpy.linspace(1, 10, 6) 生成一个等间距数列 [1, 3.25, 5.5, 7.75, 10]。
- **numpy.logspace**(start, stop, num, endpoint, base)：生成等比数列，从 base 的 start 次幂开始，到 base 的 stop 次幂结束，num 指定数列的长度，默认为 50。endpoint 和 dtype 参数与 **numpy.linspace**() 函数相同。例如，**numpy.logspace**(0, 4, 5, base=2) 将生成一个等比数列 [1, 2, 4, 8, 16]。

相信大家对 **numpy.linspace**() 函数已经不陌生，本书前文在讲可视化时已经介绍过这个函数。我们经常会在二维可视化中用到 **numpy.linspace**()。

什么是数列？

数列是指一列按照一定规律排列的数，它通常用一个公式来表示，也可以用递推关系式来定义。数列中的每个数称为数列的项，用 a_n 来表示第 n 项。数列在数学中具有广泛的应用，它是许多数学分支的基础，如数学分析、概率论、统计学、离散数学和计算机科学等。在数学中，数列是一种有序的集合，通常用于研究数学对象的性质和行为，例如函数、级数、微积分和代数等。数列可以分为等差数列、等比数列和通项公式不规则数列等几种类型。等差数列的项之间的差是固定的，比如 1、2、3、4 … 100。等比数列的相邻项之间的比是固定的，比如 2、4、8、16 … 2048。

网格数据

本书前文提过 **numpy.meshgrid**() 函数。**numpy.meshgrid**() 可以生成多维网格数据，它可以将多个一维数组组合成一个 N 维数组，并且可以方便地对这个 N 维数组进行计算和可视化。

在科学计算中，常常需要对多维数据进行可视化，比如绘制 3D 曲面图、等高线图等。**numpy.meshgrid**() 可以方便地生成网格数据，使得我们可以对多维数据进行可视化。

例如，如图 4 所示，对于二元函数 $f(x_1, x_2)$，我们可以使用 **numpy.meshgrid**() 生成横坐标和纵坐标的网格点，然后计算每个网格点的函数值，最后将网格点和对应的函数值作为输入，绘制出如图 5 所示的 3D 曲面图。

《可视之美》将介绍如何生成图 5 这幅图。

如图 6 所示，**numpy.meshgrid()** 还可以用来生成三维网格数据。在《可视之美》一册中，大家可以看到大量利用三维网格数据完成的可视化方案。

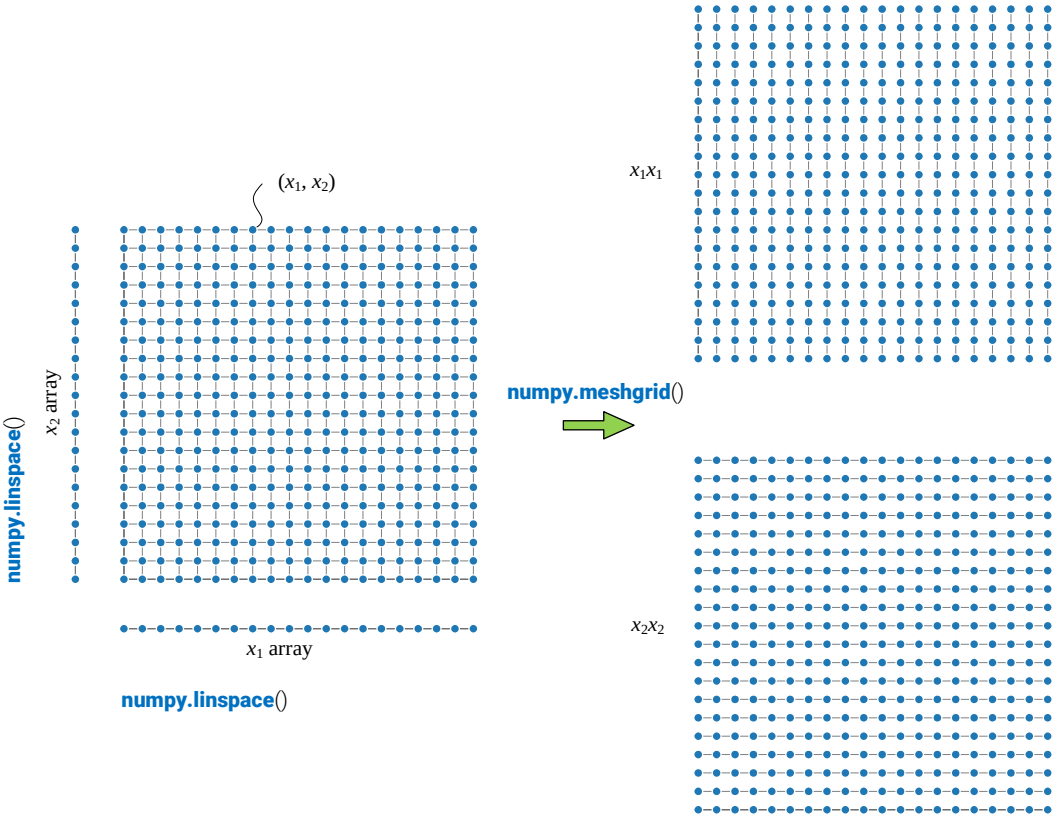


图 4. 用 numpy.meshgrid() 生成二维网络状坐标，x_1x_1 代表散点的横轴坐标，x_2x_2 代表散点的纵轴坐标

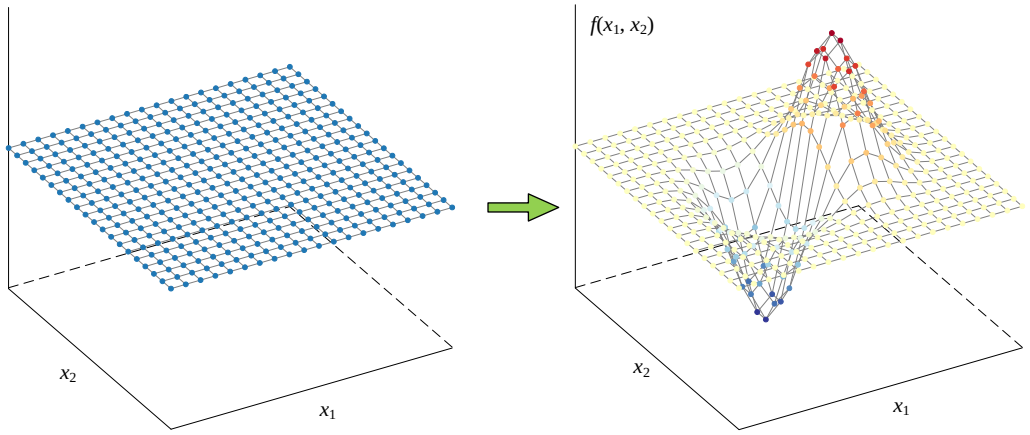


图 5. 三维空间看二维网络状坐标

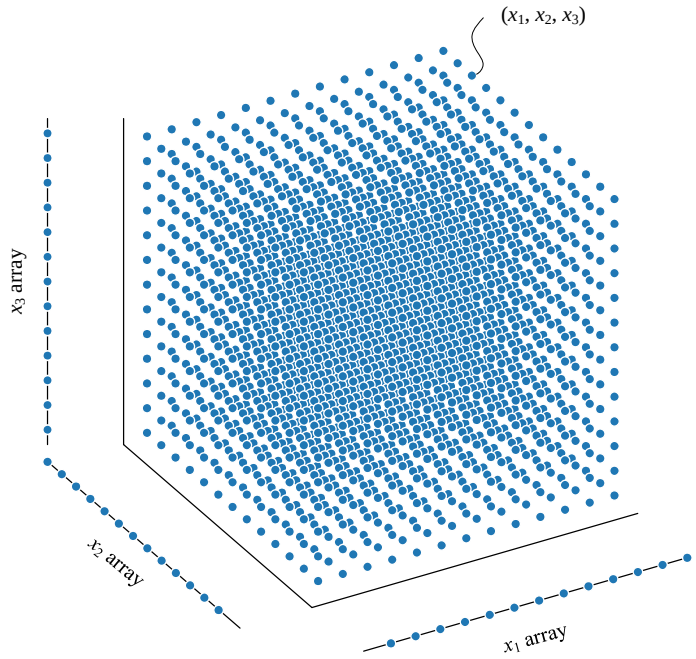


图 6. 三维网格

特殊数组

表 1 总结 NumPy 中常用来生成特殊数组的函数、用途、示例。表 1 第二列都是由 ChatGPT 生成的答案。请大家在 JupyterLab 中练习使用这些函数。

表 1. 用 NumPy 函数生成特殊数组

函数	用途	代码示例
numpy.empty()	numpy.empty()是一个用于创建一个指定大小的、未初始化的数组的函数。它返回一个数组对象，其元素的值是随机的，取决于数组在内存中的位置。因此，使用 numpy.empty()创建的数组的值是不确定的。	`import numpy as np` `np.empty([4,4])`

numpy.empty_like()	numpy.empty_like()是一个用于创建与给定数组具有相同形状和数据类型的未初始化数组的函数。它返回一个新的数组对象，其元素的值是随机的，取决于数组在内存中的位置。因此，使用 numpy.empty_like()创建的数组的值是不确定的。	`import numpy as np`
		`A = np.array([[1, 2, 3],`
		`[4, 5, 6]])`
		`np.empty_like(A)`

numpy.eye()	numpy.eye()是一个用于创建一个二维数组，表示单位矩阵的函数。它返回一个 $N \times N$ 的矩阵，其中对角线上的元素为 1，其他元素为 0。可以通过指定参数 N，来指定矩阵的大小。	`import numpy as np` `np.eye(5)`
numpy.full()	numpy.full()是一个用于创建一个指定大小和给定值的数组的函数。它返回一个数组对象，其所有元素都初始化为指定的值。可以通过指定参数来指定数组的大小和数据类型，以及所填充的值。	`import numpy as np` `np.full((3,3), np.inf)`

numpy.full_like()	numpy.full_like()是一个用于创建与给定数组具有相同形状和数据类型，且所有元素都是指定值的数组的函数。它返回一个新的数组对象，其所有元素都初始化为指定的值。可以通过指定参数来指定所填充的值。	`import numpy as np`
		`A = np.array([[1, 2, 3],`
		`[4, 5, 6]])`
		`np.full_like(A, 100)`

numpy.ones()	numpy.ones()是一个用于创建一个指定大小的全 1 数组的函数。它返回一个数组对象，其所有元素都是 1。可以通过指定参数来指定数组的大小和数据类型。	`import numpy as np` `np.ones((5,5))`

numpy.ones_like()	numpy.ones_like()是一个用于创建与给定数组具有相同形状和数据类型，且所有元素都是 1 的数组的函数。它返回一个新的数组对象，其所有元素都是 1。可以通过指定参数来指定所创建数组的数据类型。	`import numpy as np`
		`A = np.array([[1, 2, 3],`
		`[4, 5, 6]])`
		`np.ones_like(A)`

numpy.zeros()	numpy.zeros()是一个用于创建一个指定大小的全 0 数组的函数。它返回一个数组对象，其所有元素都是 0。可以通过指定参数来指定数组的大小和数据类型。	`import numpy as np` `np.zeros((5,5))`

numpy.zeros_like()	numpy.zeros_like()是一个用于创建与给定数组具有相同形状和数据类型，且所有元素都是 0 的数组的函数。它返回一个新的数组对象，其所有元素都是 0。可以通过指定参数来指定所创建数组的数据类型。	`import numpy as np`
		`A = np.array([[1, 2, 3],`
		`[4, 5, 6]])`
		`np.zeros_like(A)`

随机数

NumPy 中还有大量产生随机数的函数。图 7 所示为满足二元连续均匀分布、二元高斯分布的随机数。请大家翻阅帮助文档了解这些函数的用法，并在 JupyterLab 中动手实践。表 2 总结 NumPy 中和随机数有关的常用函数

“鸢尾花书”《统计至简》一册将专门讲解各种常用概率分布。

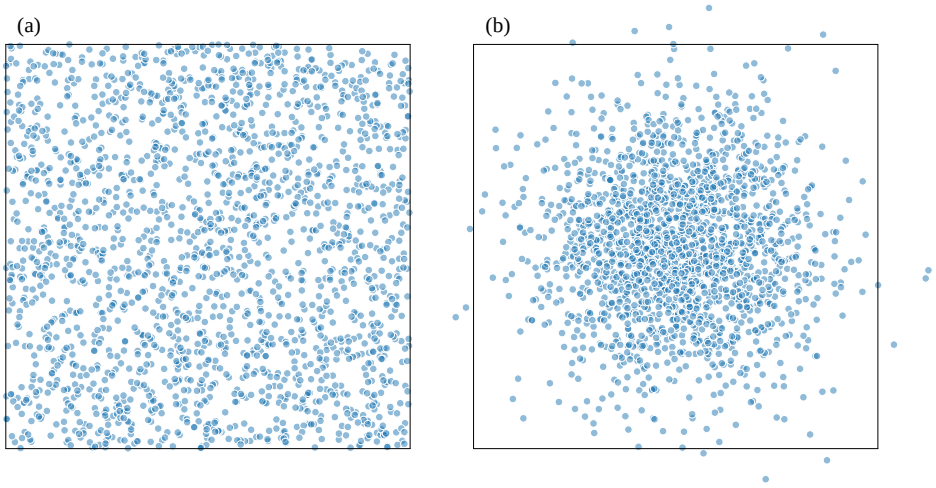


图 7. 分别满足二元连续均匀分布、二元高斯分布的随机数

表 2. NumPy 中和随机数有关的常见函数

函数名称	函数介绍
numpy.random.beta()	生成指定形状参数的贝塔分布随机数
numpy.random.binomial()	返回给定形状的随机二项分布数组。
numpy.random.chisquare()	生成指定自由度的卡方分布随机数
numpy.random.choice()	随机从给定的数组中选择元素。
numpy.random.dirichlet()	生成指定参数的狄利克雷分布随机数
numpy.random.exponential()	生成指定尺度的指数分布随机数
numpy.random.gamma()	生成指定形状和尺度的伽马分布随机数
numpy.random.lognormal()	生成指定均值和标准差的对数正态分布随机数
numpy.random.multivariate_normal()	生成多元正态分布随机数
numpy.random.normal()	生成指定均值和标准差的正态分布随机数
numpy.random.poisson()	生成指定均值的泊松分布随机数
numpy.random.power()	返回给定形状的随机幂律分布数组。
numpy.random.rand()	返回一个给定形状的随机浮点数数组，值在 0 到 1 之间。
numpy.random.randint()	返回一个给定形状的随机整数数组，值在给定范围之间。
numpy.random.randn()	返回一个给定形状的随机浮点数数组，值遵循标准正态分布。
numpy.random.random()	生成[0, 1)之间的随机数
numpy.random.seed()	设置随机数生成器的种子，确保随机数生成的可重复性。
numpy.random.shuffle()	随机打乱给定的数组。
numpy.random.uniform()	生成指定范围内的均匀分布随机数

概率统计中，随机是什么意思？

在概率统计中，随机指的是一个事件的结果是不确定的，而且每种可能的结果出现的概率是可以计算的。随机事件是由各种随机变量所描述的，随机变量是一个具有不确定结果的数学变量，其值取决于随机事件的结果。概率统计学家使用随机变量和概率分布来描述随机事件的结果和出现的概率。随机事件的结果可能是离散的，例如掷骰子的结果是 1、2、3、4、5 或 6，也可能是连续的，例如衡量人的身高或重量。概率统计学家使用各种数学方法和技术，例如概率、期望值和方差等，来分析和理解随机事件和随机变量的性质和行为。概率统计的研究在现代科学和工程中有着广泛的应用，例如金融、生物学、医学、物理学等领域。

什么是随机数发生器？

随机数生成器是一种用于生成随机数的计算机程序或硬件设备。随机数生成器可分为真随机数生成器和伪随机数生成器两种。真随机数生成器的输出完全基于物理过程，如大气噪声、放射性衰变或者热噪声等，其生成的随机数序列是完全随机且不可预测的。真随机数生成器通常需要专门的硬件设备支持。伪随机数生成器则使用计算机算法生成伪随机数，其看似随机，但是实际上是可预测的，因为它们是由固定的算法和种子值生成的。伪随机数生成器通常使用伪随机数序列和随机种子，以便在需要时生成随机数。随机数生成器在计算机科学、加密学、模拟实验、游戏设计、统计分析等领域中被广泛使用。在加密学中，随机数生成器通常用于生成安全密钥和初始化向量等关键数据，以保证加密算法的强度和安全性。在模拟实验和游戏设计中，随机数生成器用于模拟不可预测的因素，如掷骰子、扑克牌等。

从 CSV 文件中导出、导入

numpy.savetxt() 可以把 numpy array 写成 txt、CSV 文件。numpy.genfromtxt() 可以用来读入 txt、CSV 文件。图 8 所示为鸢尾花表格和热图。大家在本书后文，特别是在《矩阵力量》一册中会看到，我们大量使用热图可视化矩阵运算。

5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5.0	3.4	1.5	0.2
4.4	2.9	1.4	0.2
4.9	3.1	1.5	0.1
5.4	3.7	1.5	0.2
4.8	3.4	1.6	0.2
4.8	3.0	1.4	0.1
4.3	3.0	1.1	0.1
...	...	...	...
5.9	3.0	5.1	1.8

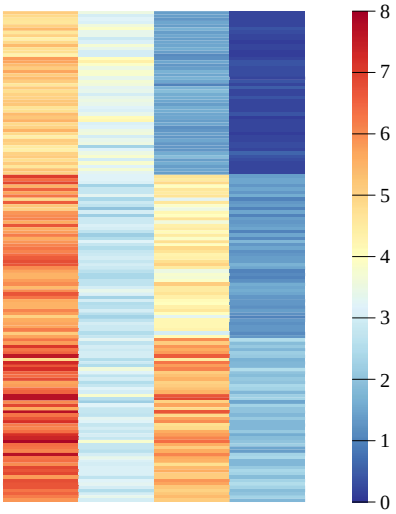


图 8. 鸢尾花数据表格和热图