3.2 Numpy 的一维数组

2023 年 9 月 8 日

Numpy 的一维数组

1. Numpy 的数组介绍

在学习 python 基础编程时，我们已经掌握了列表（Lists）的创建和使用。Numpy 中的有一种数据结构叫：数组（Arrays）。它和 Lists 有什么不同呢?

- Numpy 数组比列表运行速度更快，更省计算机资源。
- 作为 NumPy 中主要的数据结构，数组是一些值组成的网格，它包含关于原始数据的信息，以及如何定位元素，意思是可以以各种方式对其进行索引。这些元素都是相同的类型，称为数组 dtype。

数组可以由非负整数、元组、布尔值、另一个数组组成。数组的形状是一个整数元组，给出了数组的每个维度的大小。

初始化 NumPy 数组的一种方法是使用 Python 列表，对二维或高维数据使用嵌套列表。

举例，

[1]:
```
import numpy as np
a = np.array([1, 2, 3, 4, 5, 6])
a
```

[1]:
```
array([1, 2, 3, 4, 5, 6])
```

[2]:
```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
a
```

[2]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

1.1 关于数组的更多信息

您可能偶尔会听到一个数组被称为“ndarray”，它是“n 维数组”的简写，d 是单词 dimension 的缩写。n 维数组就是具有任意维数的数组。你可能还听说过一维数组，二维数组，等等。NumPy ndarray 用于表示矩阵和向量。向量是一维的数组 (行向量和列向量没有区别)，而矩阵是二维的数组。对于三维或更高维度的数组，张量这个术语也常用。

数组通常是固定大小的容器，包含相同类型的项。数组中的维数和项数由数组的形状定义。数组的形状是由非负整数组成的元组，指定每个维度的大小。

在 NumPy 中，维度称为轴。这意味着如果你有一个像这样的 2D 数组:

```
[[0., 0., 0.],
 [1., 1., 1.]]
```

数组有两个轴。第一个轴的长度是 2，第二个轴的长度是 3。

2. 如何创建一个数组

[3]:
```
import numpy as np
np.array([1, 2, 3])
```

[3]:
```
array([1, 2, 3])
```

用一种可视化方式来理解:

创建数组的时候，可以指定其中元素的类型，使用参数 dtype。dtype 可以等于 np.int64, np.float64, np.str 等等。

虽然默认的数据类型是浮点数 (np.float64)，但您可以使用 dtype 关键字显式指定想要的数据类型。

[4]:
```
a = np.array([1, 2, 3, 4, 5, 6], dtype=np.float64)
a
```

[4]:
```
array([1., 2., 3., 4., 5., 6.])
```

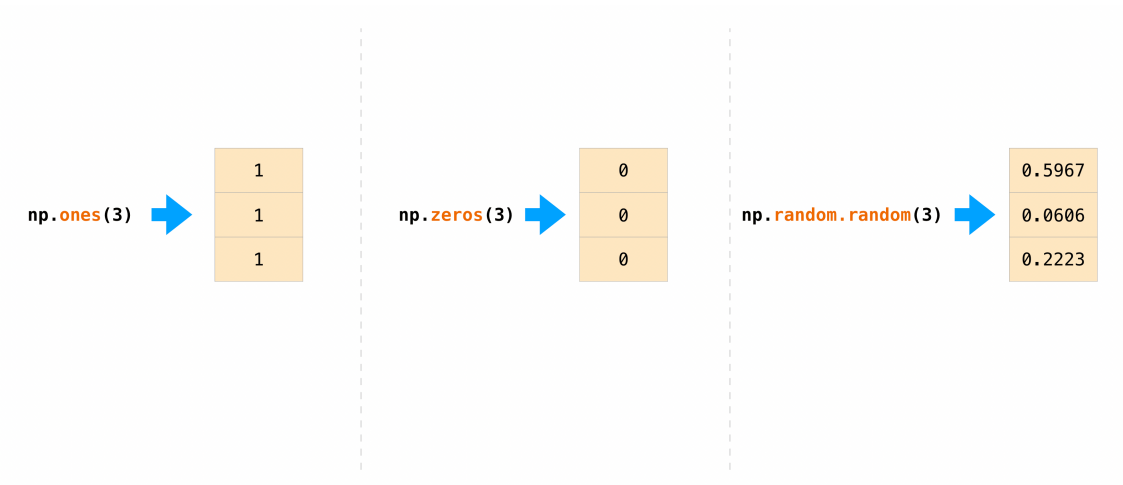
练习

请创建一个包含以下元素的一维数组：3.1, 5.2, 2.9, 6.6, 3.4, 5.5, 4.7。

[ ]:

3. 几种生成常见数组的方法

除了指定每个元素来创建数组之外，你还可以轻松创建一个由 0 填充的数组，或者由 1 填充的数组，或者是创建一组随机数。



包含指定范围的数组：

[5]:
```
np.arange(4)
```

[5]:
```
array([0, 1, 2, 3])
```

甚至包含等距间隔的数组。为此，您需要指定第一个数字、最后一个数字和步长。下面这个例子，就是从 2 开始，到 9 结束，间隔为 2。

[6]:
```
np.arange(2, 9, 2) #step=2
```

[6]:
```
array([2, 4, 6, 8])
```

你也可以使用 np.linspace() 来创建一个数组，该数组的值按指定的间隔线性排列：

[7]:
```
np.linspace(0, 10, num=5)
```

[7]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

练习

生成以一个等差数列：19, 16, 13, 10, 7, 4, 1

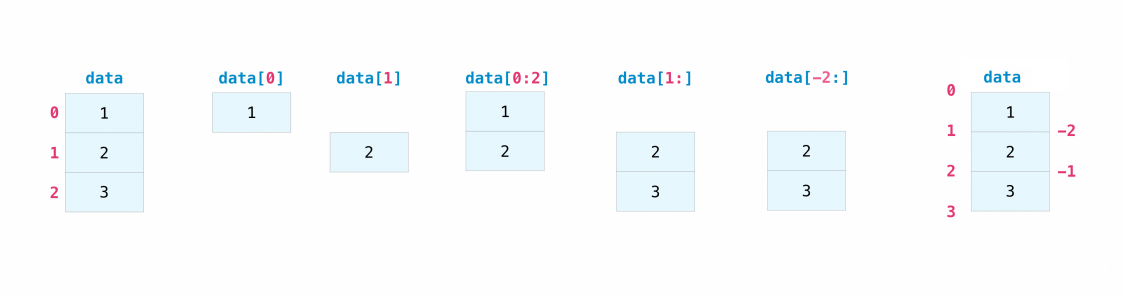
[ ]:

4. 索引和切片

您可能希望获取数组的一部分或特定数组元素，以便在进一步的分析或其他操作中使用。要做到这一点，您需要对数组进行子集、切片或索引。

就像使用列表 (lists) 一样，你可以用同样的方式索引和切片，选择数组中的部分元素。

下面用一种可视化的方式：



练习

给定下列数组 [52, 11, 26, 35, 41, 71]，请选取出部分元素:[26, 35], 再倒序选取 [35,41]。

[ ]:

5. 数组的操作方法

5.1 排序数组元素

使用 np.sort() 对元素进行排序很简单。您可以在调用函数时指定轴、类型和顺序。

[8]:
```
import numpy as np
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])  # 使用 np.array 创建一个 1 维无序数组
```

你可以按照升序进行排序:

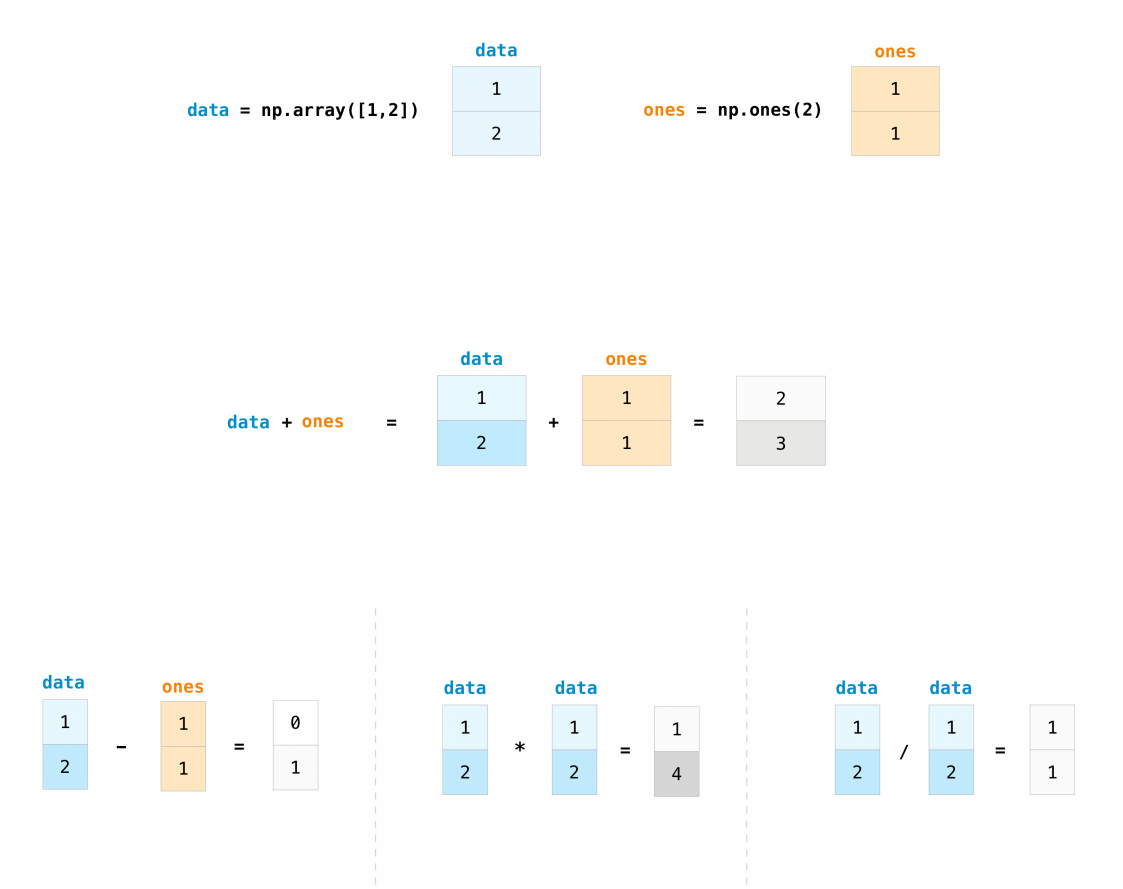
[9]:
```
np.sort(arr)
```

[9]: array([1, 2, 3, 4, 5, 6, 7, 8])

除了 sort(返回数组的已排序副本)，您还可以使用: - argsort，它是沿着指定轴进行间接排序，- lexsort，是对多个键的间接稳定排序，- searchsorted，它将在已排序的数组中查找元素 - partition，这是一种部分排序。

5.2 数学运算

一旦创建了数组，就可以开始使用它们了。比如说，你创建了两个数组，一个叫 data，一个叫 ones。



[10]:
```
data = np.array([1, 2])
ones = np.ones(2, dtype=int)
data + ones
```

[10]: array([2, 3])

[11]:
```
data - ones
```

[11]:
```
array([0, 1])
```

[12]:
```
data * data
```

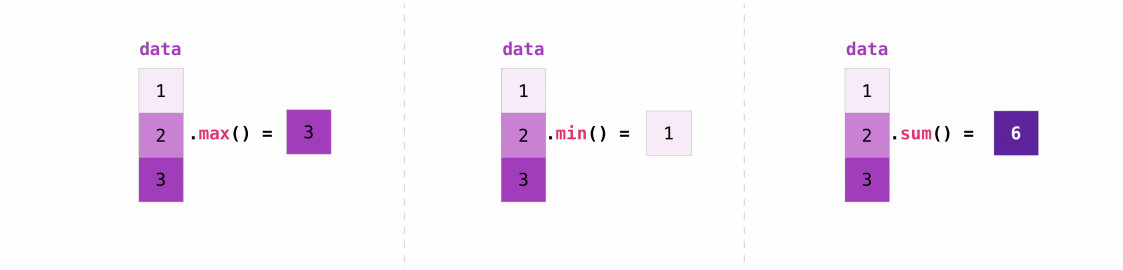
[12]:
```
array([1, 4])
```

[13]:
```
data / data
```

[13]:
```
array([1., 1.])
```

5.3 求和、最大值、最小值

NumPy 还执行聚合函数。除了 min、max 和 sum 之外，您还可以轻松地运行 mean 以得到平均值，prod 以得到元素相乘的结果，std 以得到标准差，等等。



[14]:
```
data = np.array([1, 2, 3])
print('序列 [1,2,3]\n最大值:\t%i\n最小值:\t%i\n总和:\t%i' %(np.max(data), np.
  ↪min(data), np.sum(data)))
```

```
序列 [1,2,3]
最大值:    3
最小值:    1
总和:    6
```

[15]:
```
data = np.array([1, 2, 3])
print('序列 [1,2,3]\n最大值:\t%i\n最小值:\t%i\n总和:\t%i' %(data.max(), data.
  ↪min(), data.sum()))
```

```
序列 [1,2,3]
最大值:    3
最小值:    1
总和:    6
```

6. 一个例子

实现在矩阵和向量上的数学公式是 NumPy 的一个关键用处，这也是为什么 NumPy 是 python 科学计算领域的宠儿。

例如，均方误差公式是解决回归问题的有监督机器学习模型的一个关键。

$$MeanSquareError = \frac{1}{n}\sum_{i=1}^{n}(Y\_prediction_i - Y_i)^2$$

用 NumPy 来实现是一件轻而易举的事：

```
error = (1/n) * np.sum(np.square(predictions - labels))
```

优雅之处在于 numpy 不关心 predictions 和 labels 的容量是 1 还是几百个值（只要它们有同样的容量）。我们可以通过如下四个步骤来对这行代码进行一个序列解读：

```
                                  predictions  labels
error = (1/3) * np.sum(np.square(     1          1    ))
                                      1      -   2
                                      1          3
```

predictions 和 labels 向量都有 3 个值，也就是说 n = 3, 计算完减法后，我们得到如下的公式：

```
error = (1/3) * np.sum(np.square(     0    ))
                                     -1
                                     -2
```

```
error = (1/3) * np.sum(     0     )
                            1
                            4
```

然后对这个向量求平方操作：

```
error = (1/3) * 5
```

现在，我们对三个数进行求和：

error 中的值就是模型预测的误差。

练习

给定 Y（或称为 label）序列值为 [10.1, 9.8, 10.5, 10.0, 10.3], 其预测的结果，也就是 predictions 为 [10, 10, 10, 10, 10] 时，MSE 的值是多少?

[ ]: