

3.3 Numpy 的二维数组

2023 年 9 月 8 日

Numpy 的二维数组

1. 创建二维数组

如何创建一个二维数组呢？你可以传递 Python 列表来创建一个 2d 数组 (或“矩阵”)。

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

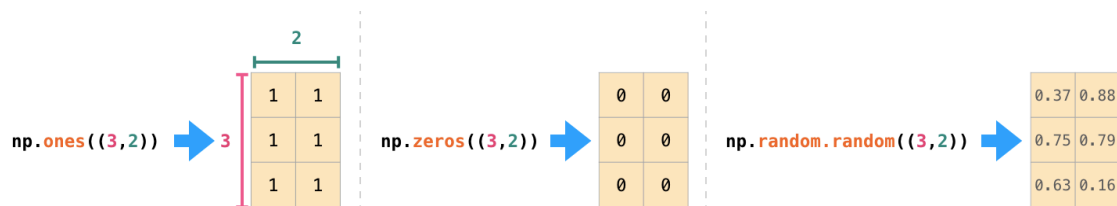
```
[1]: array([[1, 2],
           [3, 4],
           [5, 6]])
```

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



1	2
3	4
5	6

在一维数组的创建中，我们已经学会 `ones()`, `zeros()` 和 `random()`，当然也可以使用 `ones()`, `zeros()` 和 `random()` 来创建一个 2D 数组，如果你给他们一个元组来描述矩阵的维数：



1.1 数组的形状和尺寸

`ndarray.ndim` 会告诉你数组的轴数，或者说是维度。

`ndarray.size` 会告诉你数组元素的总数。这是数组的形状元素的乘积。

`ndarray.shape` 将显示一个整数元组，该元组指示数组每个维度上存储的元素数量。

```
[1]: # array_example 是一个 3 维数组
import numpy as np
array_example = np.array([[[0, 1, 2, 3],
                           [4, 5, 6, 7]],
                          [[0, 1, 2, 3],
                           [4, 5, 6, 7]],
                          [[0, 1, 2, 3],
                           [4, 5, 6, 7]]])
```

```
[2]: print('维度:', array_example.ndim)
      print('尺寸:', array_example.size)
      print('形状:', array_example.shape)
```

维度: 3

尺寸: 24

形状: (3, 2, 4)

猜猜看以下数组的 `ndim`、`size`、`shape`

```
[5]: # 2 维数组
array_example = np.array([[0, 1, 2, 3],
                           [4, 5, 6, 7],
                           [0, 1, 2, 3],
                           [4, 5, 6, 7],
                           [0, 1, 2, 3],
                           [4, 5, 6, 7]])
```

```
[6]: print('维度:', array_example.ndim)
      print('尺寸:', array_example.size)
      print('形状:', array_example.shape)
```

维度: 2

尺寸: 24

形状: (6, 4)

2. 索引和切片

当你在操作矩阵时, 索引和切片操作很有用:

data			data[0,1]			data[1:3]			data[0:2,0]		
	0	1		0	1		0	1		0	1
0	1	2		1	2		1	2		1	2
1	3	4		3	4		3	4		3	4
2	5	6		5	6		5	6		5	6

```
[6]: data = np.array([[1, 2], [3, 4], [5, 6]])  
data[0, 1]
```

```
[6]: 2
```

```
[7]: data[1:3]
```

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

```
[8]: data[0:2, 0]
```

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

2.1 满足特定条件的切片

如果想从数组中选择满足特定条件的值, 使用 NumPy 很简单。

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

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

从 a 中选取小于 10 的子集:

```
[10]: below_five = a[a < 10]
      below_five
```

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

或者是选取可以被 2 整除的子集:

```
[11]: divisible_by_2 = a[a%2==0]          # % 整除, 求余数
      divisible_by_2
```

```
[11]: array([ 2,  4,  6,  8, 10, 12])
```

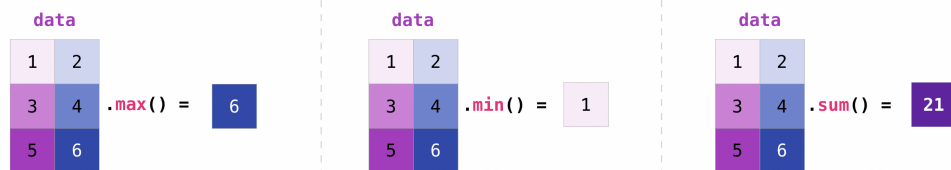
选取满足多组条件下的子集:

```
[12]: a[(a > 2) & (a < 11)]
```

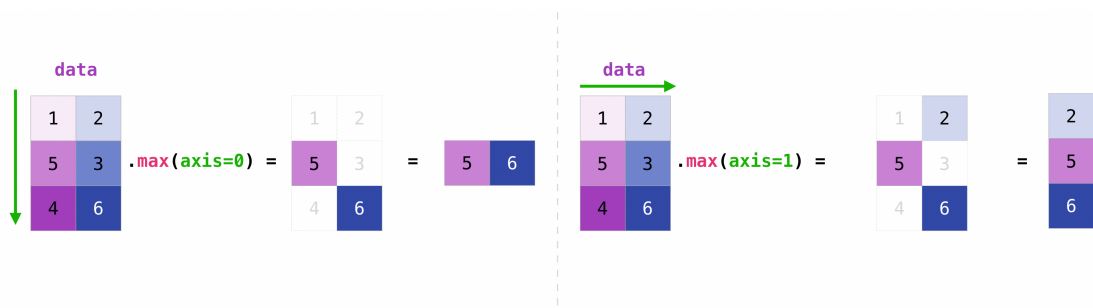
```
[12]: array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

3. 求和、最大值、最小值

和一维数组一样，二维数组的求最大值、最小值和总和，如下:



你可以在一个矩阵中聚合所有的值，你可以使用轴参数跨列或跨行聚合它们:



```
[16]: import numpy as np
data = np.array([[1, 2], [5, 3], [4, 6]])
data.max(axis=0)
```

```
[16]: array([5, 6])
```

```
[17]: data.max(axis=1)
```

```
[17]: array([2, 5, 6])
```

一旦你创建了你的矩阵，如果你有两个相同大小的矩阵，你可以使用算术运算符对它们进行相加和相乘

$$\text{data} + \text{ones} = \begin{array}{|c|c|} \hline \text{data} & \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{ones} & \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 2 & 3 \\ \hline 4 & 5 \\ \hline \end{array}$$

你可以在不同大小的矩阵上做这些算术运算，但只有当一个矩阵只有一列或一行。在本例中，NumPy 将使用它的广播规则进行操作。

$$\text{data} + \text{ones_row} = \begin{array}{|c|c|} \hline \text{data} & \\ \hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline \end{array} + \begin{array}{|c|c|} \hline \text{ones_row} & \\ \hline 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & 4 \\ \hline 5 & 6 \\ \hline \end{array} + \begin{array}{|c|c|} \hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 2 & 3 \\ \hline 4 & 5 \\ \hline 6 & 7 \\ \hline \end{array}$$

```
[19]: data = np.array([[1, 2], [3, 4], [5, 6]])
ones_row = np.array([[1, 1]])
data + ones_row
```

```
[19]: array([[2, 3],
           [4, 5],
           [6, 7]])
```

4. 转置

通常需要对 2 维数组进行转置。NumPy 数组有一个属性 `T`，它允许转置

data		data.T		
1	2	1	3	5
3	4	2	4	6
5	6			

```
[20]: data
```

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

```
[21]: data.T
```

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

5. 变形

你可能还需要变换矩阵的维数。例如，当您有一个模型，它需要一个与您的数据集不同的输入形状时，就会发生这种情况。这就是重塑方法可以发挥作用的地方。你只需要传递你想要的矩阵的新维度。

data

1
2
3
4
5
6

data.reshape(2,3)

1	2	3
4	5	6

data.reshape(3,2)

1	2
3	4
5	6

The diagram illustrates the process of reshaping a 1D array into 2D arrays. The original array 'data' contains the values 1 through 6. The first reshaping operation, 'data.reshape(2,3)', creates a 2x3 array where the first row contains [1, 2, 3] and the second row contains [4, 5, 6]. The second reshaping operation, 'data.reshape(3,2)', creates a 3x2 array where the first two rows contain [1, 2] and [3, 4], and the third row contains [5, 6]. Red vertical brackets indicate the number of rows (2 and 3 respectively), and purple horizontal brackets indicate the number of columns (3 and 2 respectively).

如果想把 2 维数组变成 1 维，该如何操作呢？

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

```
[23]: x.flatten()
```

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

6. 连结两个数组

```
[24]: a = np.array([1, 2, 3, 4])  
      b = np.array([5, 6, 7, 8])
```

可以使用 `np.concatenate()` 将它们连接起来。

```
[25]: np.concatenate((a, b))      # 或者使用 np.concatenate([a, b])
```

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

或者是另一个方向，

```
[26]: x = np.array([[1, 2], [3, 4]])  
      y = np.array([[5, 6]])
```

```
[27]: x
```

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

```
[28]: y
```

```
[28]: array([[5, 6]])
```

你把它它们拼接在一起

```
[29]: np.concatenate((x, y), axis=0)
```

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

[5, 6]])

7. 参考:

1. NumPy: the absolute basics for beginners, https://numpy.org/doc/stable/user/absolute_beginners.html
2. A Visual Intro to NumPy and Data Representation, <https://jalammar.github.io/visual-numpy/>

[]: