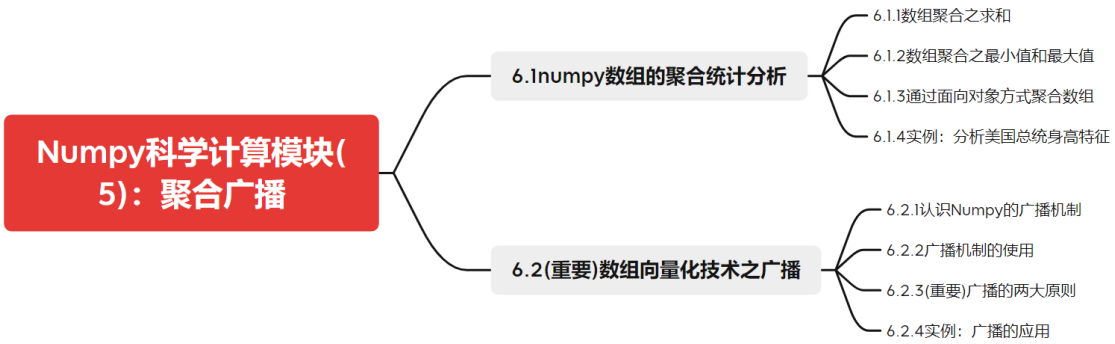
Numpy科学计算模块（五）：聚合广播

思维导图



6.1numpy数组的聚合统计分析

当面对大量的数据时，首先需要计算数据的**相关统计值。**

常见统计值有均值和标准差，通过这两个值可快速了解数据的基本特征，但是其他一些形式的聚合统计方式：求和、乘积、中位数、最小值和最大值、分位数等等。

任务：认识numpy数组的聚合

> 知识点：经常会在numpy中听到**聚合**术语，那么何谓聚合？
>
> - **聚合**：指对数组进行某种操作或运算(求和、极值等)，得到了一个**对原数组信息概括的结果数组。**而结果数组的维度`ndim`被压缩了。换言之，**原数组的维度**`ndim`**大于结果数组的维度**`ndim`**。**
> - 聚合作用：结果数组**按某种操作聚合(提炼)了**原数组的信息；结果数组概括了原数组某些方面的统计信息，比如求和、均值、方差等。
> - 原数组：numpy聚合函数的输入数组。
> - 结果数组：numpy聚合函数的返回数组。

习题：

- 了解术语：原数组、结果数组、聚合的概念。

```
原数组：numpy聚合函数的输入数组。

结果数组：numpy聚合函数的返回数组。

聚合：指对数组进行某种操作或运算(求和、极值等)，得到了一个对原数组信息概括的结果数组。而结果数组的维度`ndim`被压缩了。换言之，原数组的维度`ndim`大于结果数组的维度`ndim`。
```

6.1.1数组聚合之求和

应用广播：数组归一化处理

> 知识点：
>
> - NumPy向量化技术包括：通用函数和广播技术。
> - 通用函数让NumPy用户免于低效的Python循环。
> - 广播技术扩展了NumPy向量化适用范围。

> 知识点：在数据预处理中，通常都需要对**数据进行归一化。**那么为何需要数据归一化？

> - 数据由样本组成，而样本是由特征所组成。
> - 通常，数组的行代表样本，列代表特征。(联想Excel表格)。
> - 例如，现在有一个身高体重的数据集，那么样本代表一条记录，每个样本都是有2个特征(身高和体重)组成。
> - 由于每个特征的量纲(单位)不一样，如果直接使用原始数据处理，将会对某些数值较小的特征造成偏见。为此，需要使用**数据归一化技术，将每个特征都进行无量纲化。**

假设数据集有10个样本，每个样本包含3个特征(属性或数值)，那么可以使用10×3的数组存放该数据。

```
# 随机生成10×3的数据数组
# 数组每行为1个样本，每个列为数据的1个特征。
X = np.random.random((10, 3))
print(X)

# 调用通用函数mean()沿第0轴(样本)计算均值(聚合)
Xmean = X.mean(axis=0)
print('mean:', Xmean)
```

```
# 中心化处理：对数组每个行加去特征矩阵
# 思考这里是如何进行广播的？
X_centered = X - Xmean

# 输出中心化后，每个特征的均值
# 查看归一化的数组的均值是否接近0。
# 计算机是
X_centered.mean(0)
```

计算机是对浮点数表示存在误差。在机器精度范围内，上述均值为 0。

应用广播：生成二维函数

下面将通过广播技术，生成一个二维函数，并绘制该函数图像。

定义函数, 和的定义域为 `[0, 50)`。这里将`x`作为第1维坐标，`y`作为第2维坐标。使用`np.linspace()`生成数据。

```
# x和y表示0~5区间50个步长的序列
# 二维数组的列向量(axis=0)
x = np.linspace(0, 5, 50)[:, np.newaxis]

# 一维数组，广播规则1，等价于
# 二维数组的行向量(axis=1)
# np.linspace(0, 5, 50)[np.newaxis,:]
y = np.linspace(0, 5, 50)
```

```
# 根据z的二元函数，生成函数值
# 思考：z的`.shape`是多少？
# 思考广播技术是如何进行的？
z = np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
```

```
# 使用matplotlib模块绘制函数图像
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set()  # 设置绘图风格

# 绘制函数等高线
fig, ax = plt.subplots()
CS = plt.contour(z)
# 设置等高线上的文本
ax.clabel(CS, inline=1, fontsize=10)
```

习题：

1、题目要求

- (概念题) 结合原数组和结果数组的`shape`，分析`np.sum()`的`axis`参数的含义。

代码如下：

```
x = np.random.randint(10, size=(4,7))
print(x)
print('sum(x)', sum(x))
print('np.add.reduce()', np.add.reduce(x))
print('np.sum(x)', np.sum(x))
print('np.sum(x, axis=0)', np.sum(x, axis=0))
print('np.sum(x, axis=1)', np.sum(x, axis=1))
```

运行结果：

```
[[2 1 9 1 0 8 4]
 [2 3 0 3 0 6 0]
 [3 9 9 4 4 3 6]
 [9 3 1 0 8 3 4]]
sum(x) [16 16 19  8 12 20 14]
np.add.reduce() [16 16 19  8 12 20 14]
np.sum(x) 105
np.sum(x, axis=0) [16 16 19  8 12 20 14]
np.sum(x, axis=1) [25 14 38 28]
```

2、题目要求

- 随机生成一个`size=(4,7)`的二维整形数组。
    - 使用`np.sum()`，设置不同的`axis`参数，对该数组进行求和聚合，输出结果数组的`shape`、`ndim`等信息。
    - 使用`sum()`、`np.add.reduce()`函数，输出结果数组，然后对比`np.sum()`的结果。

代码如下：

```
x = np.random.randint(10, size=(4,7,8,10))
print(x)
print('sum(x)', sum(x))
print('np.add.reduce()', np.add.reduce(x))
print('np.sum(x)', np.sum(x))
print('np.sum(x, axis=0)', np.sum(x, axis=0))
print('np.sum(x, axis=1)', np.sum(x, axis=1))
```

运行结果：

```
[[[[1 0 7 ... 5 4 4]
   [7 9 8 ... 0 2 2]
   [3 0 3 ... 5 7 5]
   ...
   [7 0 6 ... 9 1 5]
   [0 2 4 ... 1 5 8]
   [0 8 4 ... 2 1 6]]

  [[8 5 9 ... 4 5 3]
   [6 0 7 ... 5 7 3]
   [8 4 7 ... 2 0 8]
   ...
   [1 0 3 ... 5 1 3]
   [7 0 4 ... 7 2 8]
   [7 4 8 ... 4 9 1]]

  [[7 6 4 ... 8 2 0]
   [5 0 9 ... 3 2 6]
   [2 6 1 ... 1 3 1]
   ...
   [0 2 6 ... 4 4 2]
   [7 4 2 ... 9 1 5]
   [7 6 5 ... 6 4 7]]

  ...

  [20 35 37 41 33 33 28 39 46 32]
  [38 34 30 36 29 37 35 25 25 30]
  [24 30 27 29 38 32 26 36 36 35]
  [26 15 28 34 16 19 29 17 36 26]
  [31 39 35 36 42 32 34 32 25 29]]]
```

6.1.2数组聚合之最小值和最大值

任务：`np.min()`和`np.max()`数组的最小值和最大值

> 知识点：`np.min()`和`np.max()`分别获取numpy数组的最小值和最大值。由于用法与`np.sum()`类似，这里不具体展开。
>
> - 与`np.sum()`类似，`np.min()`和`np.max()`也可以通过参数`axis`指明对哪个`axis`轴元素进行聚合。
> - 最小值方法语法：`numpy.min(a, axis=None, out=None)`。
> - 最大值方法语法：`numpy.max(a, axis=None, out=None)`。
> - python也提供了内置`min()`和`max()`方法，但是该功能**仅限于一维数组，不能用于高维数组**。

> - 新numpy版本中，`np.min()`的别名方法`np.amin()`；`np.max()`的别名方法`np.amax()`。功能等价。

```
# 对大数组进行运算
big_array = np.random.rand(1000000)
%time min(big_array)
%time np.min(big_array)
```

```
x = np.random.randint(10, size=(3,4))
print(x)

#python内置的min不能用于高维数组
#print('min(x)', min(x))

print('np.min(x)', np.min(x))
# 对二维数组，axis=0，沿列求最小值
print('np.min(x, axis=0)', np.min(x, axis=0))
# 对二维数组，axis=1，沿行求最小值
print('np.min(x, axis=1)', np.min(x, axis=1))
```

习题

题目要求：

- 随机生成一个`size=(4,7,2)`的三维整形数组。
- - 使用`np.min()`和`np.max()`，设置不同的`axis`参数，对该数组进行聚合，输出结果数组的`shape`、`ndim`等信息。
  - 尝试着使用python内置函数，输出结果数组。

代码如下：

```
x = np.random.randint(10, size=(4,7,2))
print(x)
print('np.min(x)', np.min(x))
print('np.min(x, axis=0)', np.min(x, axis=0))
print('np.min(x, axis=1)', np.min(x, axis=1))

print('np.max(x)', np.max(x))
print('np.max(x, axis=0)', np.max(x, axis=0))
print('np.max(x, axis=1)', np.max(x, axis=1))
```

运行结果：

```
[[[3 7]
  [5 1]
  [8 0]
  [4 7]
  [4 6]
  [3 4]
  [5 8]]
```

```
 [[4 9]
  [3 8]
  [5 8]
  [7 4]
  [4 6]
  [4 2]
  [1 8]]

 [[3 2]
  [1 8]
  [1 2]
  [7 5]
  [6 3]
  [1 8]
  [4 6]]

 [[5 2]
  [0 1]
  [5 5]
  [2 0]
  [4 6]
  [4 3]
  [6 0]]]
np.min(x) 0
np.min(x, axis=0) [[3 2]
 [0 1]
 [1 0]
 [2 0]
 [4 3]
 [1 2]
 [1 0]]
np.min(x, axis=1) [[3 0]
 [1 2]
 [1 2]
 [0 0]]
np.max(x) 9
np.max(x, axis=0) [[5 9]
 [5 8]
 [8 8]
 [7 7]
 [6 6]
 [4 8]
 [6 8]]
np.max(x, axis=1) [[8 8]
 [7 9]
 [7 8]
 [6 6]]
```

6.1.3通过面向对象方式聚合数组

任务：通过对象方法的方式对数组进行聚合

> 知识点：
>
> - `np.min()`、`np.max()`、`np.sum()`在数组对象`x`中，可以通过更简洁的**面向对象方法**被调用。
> - 调用方式：`x.min()`、`x.max()`、`x.sum()`。其中`x`为数组对象。
> - `np.min()`第一个参数为数组。然而，在`x.min()`中，通过对象`x`调用`min()`方法，`x`被隐式传递给了`min()`方法。因此，不再需要额外传递数组参数。
> - 通过对象来调用聚合方法，可实现数组的求和、最小值和最大值等功能。
> - `axis`参数功能类似，可以实现多维数组的聚合功能。

> 知识点：可以通过`axis`参数，指明聚合函数是针对数组的哪个`axis`轴进行操作的。

> 知识点：如果缺省`axis`参数，那么聚合函数将会返回对整个数组的聚合结果。

```
x = np.random.randint(10, size=(3,4))
print(x)

# 缺省`axis`参数，返回整个数组元素的求和
print('x.sum()', x.sum())

# 对二维数组，axis=0，沿列求和
print('x.sum(axis=0)', x.sum(axis=0))

# 对二维数组，axis=1，沿行求和
print('x.sum(axis=1)', x.sum(axis=1))
```

习题

题目要求：

- 随机生成一个`size=(4,7,4)`的三维整形数组，使用`x.min()`、`x.max()`对原数组进行聚合，分析不同`axis`参数对结果的影响，并输出结果数组的`shape`和`ndim`。

代码如下：

```
x = np.random.randint(10, size=(4,7,4))
print(x)
print('np.min(x)', np.min(x))
print('np.min(x, axis=0)', np.min(x, axis=0))
print('np.min(x, axis=1)', np.min(x, axis=1))

print('np.max(x)', np.max(x))
print('np.max(x, axis=0)', np.max(x, axis=0))
print('np.max(x, axis=1)', np.max(x, axis=1))
```

运行结果：

```
[[[5 3 6 5]
  [5 7 2 4]
  [9 9 4 2]
  [7 2 9 8]
  [5 6 6 4]
  [7 3 6 6]
  [6 1 6 2]]

 [[6 5 2 5]
  [5 3 0 1]
  [2 0 9 2]
  [6 4 8 9]
  [8 4 3 4]
  [5 8 6 1]
  [7 4 2 9]]

 [[4 6 7 2]
  [7 1 1 7]
  [3 3 2 8]
  [6 7 3 8]
  [0 3 1 1]
  [5 5 5 4]
  [9 2 7 1]]

 [[2 5 4 4]
  [5 9 9 3]
  [1 5 7 7]
  [1 2 2 7]
  [1 3 0 0]
  [4 8 2 0]
  [4 2 5 5]]]
np.min(x) 0
np.min(x, axis=0) [[2 3 2 2]
 [5 1 0 1]
 [1 0 2 2]
 [1 2 2 7]
 [0 3 0 0]
 [4 3 2 0]
 [4 1 2 1]]
np.min(x, axis=1) [[5 1 2 2]
 [2 0 0 1]
 [0 1 1 1]
 [1 2 0 0]]
np.max(x) 9
np.max(x, axis=0) [[6 6 7 5]
 [7 9 9 7]
 [9 9 9 8]
 [7 7 9 9]
 [8 6 6 4]
 [7 8 6 6]
 [9 4 7 9]]
np.max(x, axis=1) [[9 9 9 8]
 [8 8 9 9]
 [9 7 7 8]
 [5 9 9 7]]
```

任务：numpy数组的聚合函数清单

numpy提供了很多聚合函数，下表提供了聚合函数的清单。NaN安全版本表示，数组含有缺失值时的聚合操作，后面小节会涉及到缺失值数组。

表格最右边栏不能完全显示，可以拖动滚动条显示结果。

函数名称	NaN安全版本	描述
`np.sum`	`np.nansum`	计算元素的和
`np.prod`	`np.nanprod`	计算元素的积
`np.mean`	`np.nanmean`	计算元素的平均值
`np.std`	`np.nanstd`	计算元素的标准差
`np.var`	`np.nanvar`	计算元素的方差
`np.min`	`np.nanmin`	找出最小值
`np.max`	`np.nanmax`	找出最大值
`np.argmin`	`np.nanargmin`	找出最小值的索引
`np.argmax`	`np.nanargmax`	找出最大值的索引
`np.median`	`np.nanmedian`	计算元素的中位数
`np.percentile`	`np.nanpercentile`	计算基于元素排序的统计值
`np.any`	`N/A`	验证任何一个元素是否为真
`np.all`	`N/A`	验证所有元素是否为真

习题

题目要求：

- 随机生成一个`size=(4,7)`的二维整形数组。任选2种聚合方法(之前没学过的)，设置不同`axis`参数对数组进行聚合，并分析结果。

代码如下：

```
x = np.random.randint(10, size=(4,7))
print(x)
print('x.prod()', x.prod())
print('x.prod(axis=0)', x.prod(axis=0))
print('x.prod(axis=1)', x.prod(axis=1))

print('x.mean()', x.mean())
print('x.mean(axis=0)', x.mean(axis=0))
print('x.mean(axis=1)', x.mean(axis=1))
```

运行结果：

```
[[8 6 4 1 5 0 0]
 [4 2 3 0 6 0 5]
 [5 5 6 4 8 8 8]
 [8 9 4 6 4 0 1]]
x.prod() 0
x.prod(axis=0) [1280  540  288    0  960    0    0]
x.prod(axis=1) [     0      0 307200      0]
x.mean() 4.285714285714286
x.mean(axis=0) [6.25 5.5  4.25 2.75 5.75 2.   3.5 ]
x.mean(axis=1) [3.42857143 2.85714286 6.28571429 4.57142857]
```

- 随机生成一个`size=(4,7)`的二维整形数组。任选2种聚合方法(之前没学过的)，设置不同`axis`参数对数组进行聚合，并分析结果。

6.1.4实例：分析美国总统身高特征

任务：分析美国总统身高特征

用numpy的聚合功能来概括一组数据的信息是非常有用的。下面将通过一个简单的例子了解这项功能在实际问题中的应用——计算所有美国总统的身高。这个数据在president_heights.csv文件中. 需要用Pandas模块来读文件并抽取身高信息。(请注意，身高的计量单位是厘米)。后续章节会详细解释Pandas。

> 知识点：如果发现模块不存在，安装`pandas`、`matplotlib`和`seaborn`模块。在控制台中，运行`conda install pandas matplotlib seaborn`命令进行安装(并选中`Y`)。

任务：通过pandas模块导入数据

```
import pandas as pd
data = pd.read_csv('president_heights.csv')
heights = np.array(data['height(cm)'])
print(heights)
```

任务：通过聚合方法概括数组信息

针对身高数组`heights`，使用聚合函数获得各种概括的统计信息。

```
print("Mean height:       ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height:    ", heights.min())
print("Maximum height:    ", heights.max())
```

在这个例子中，聚合函数将整个数组缩减到一个值(思考为何是一个值而不是数组)，通过这些统计信息值，可以掌握`heights`数组的分布信息。下面获得分位数统计信息。

```
print("25th percentile:   ", np.percentile(heights, 25))
print("Median:            ", np.median(heights))
print("75th percentile:   ", np.percentile(heights, 75))
```

任务：可视化结果

可以看到，美国总统的身高中位数是 182cm，或者说不到 6 英尺。当然，有些时候将数据可视化更有用。这时可以先进行一个快速的可视化，通过Matplotlib模块(第4章将详细讨论该工具)建立可视化结果。

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set()  # 设置绘图风格

plt.hist(heights)
plt.title('Height Distribution of US Presidents')
plt.xlabel('height (cm)')
plt.ylabel('number');
```

6.2（重要）数组向量化技术之广播

6.2.1认识NumPy的广播机制

任务：认识NumPy的广播机制

> 知识点：之前学习了NumPy是如何通过**向量化**技术(通用函数)来避免缓慢的Python循环。二元通用函数的向量化运算，要求两个参与运算的数组`.shape`**必须相同。**
>
> - 因为需要对两个运算数组的相应元素进行逐个计算。
> - 如果`.shape`不匹配，必定有某些数组元素无法参与运算，致使程序出错。
> - 为解决数组`.shape`不匹配问题，NumPy专门设计了一套广播机制(broadcasting)。
> - **在满足一定条件下**，NumPy在底层通过广播机制，修正运算的数组`.shape`匹配。进而允许`.shape`不匹配的数组，可以参与向量化运算。
> - 广播技术堪称NumPy数组向量化的黑科技，它为NumPy通用函数带来了高效和便捷。

```
import numpy as np
a = np.array([0, 1, 2])
print('a:', a)
b = np.array([5, 5, 5])
print('b:', b)
print('a+b:', a + b)
```

> 知识点：
>
> - 在满足一定规则下，广播(broadcasting)允许二元通用函数输入**不同大小的数组。**
> - 标量可认为是**零维数组**，它的`.ndim`为0；`.shape`为`()`。

6.2.2广播机制的使用

任务：广播之标量和数组运算

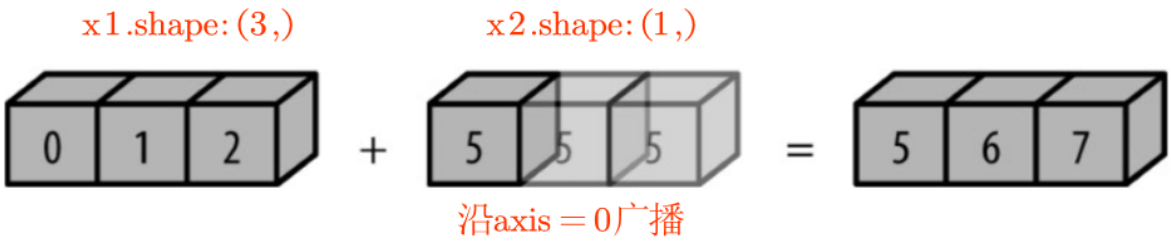
下面给出标量和一维数组的通用函数运算(相加)。

```
# 输出一维数组的维度和shape
x1 = np.arange(3)
print(x1, x1.shape, x1.ndim)

# 输出标量的维度和shape
x2 = np.array(5)
print(x2, x2.shape, x2.ndim)

# 调用二元通用函数(相加)进行运算，底层调用广播机制
print(x1 + x2)
```

> 知识点：广播机制的理解。
>
> - 注意：`x2`是标量`.shape`为`()`，**可等价视为一维数组**，即`.shape`为`(1,)`。下面对`x2`的分析都采用一维数组方式。
> - `x1`的`.shape`为`(3,)`和`x2`的`.shape`为`(1,)`。
> - 参与二元通用函数运算的`x1`和`x2`不匹配。
> - 广播机制直观理解(结合下图)：
> - - 首先NumPy对标量`x2`**采用元素复制的方式**，将它扩展为`[5, 5, 5]`一维向量。
>   - 此时新的`x2`的`.shape`为`(3,)`，和`x1`匹配，满足通用函数运算条件。
>   - 然后，执行二元通用函数(相加)运算。
> - 上面仅是广播机制的直观解释，实质上NumPy并非直接通过复制元素的方式，来匹配运算数组的`.shape`，而是采用了一种更高效的方式(这里就不展开讨论)。



浅色的盒子表示广播的值。同样需要注意的是，这个额外的内存并没有在实际操作中进行分配，但是这样的想象方式更方便我们从概念上理解。

任务：广播之一维和二维数组运算

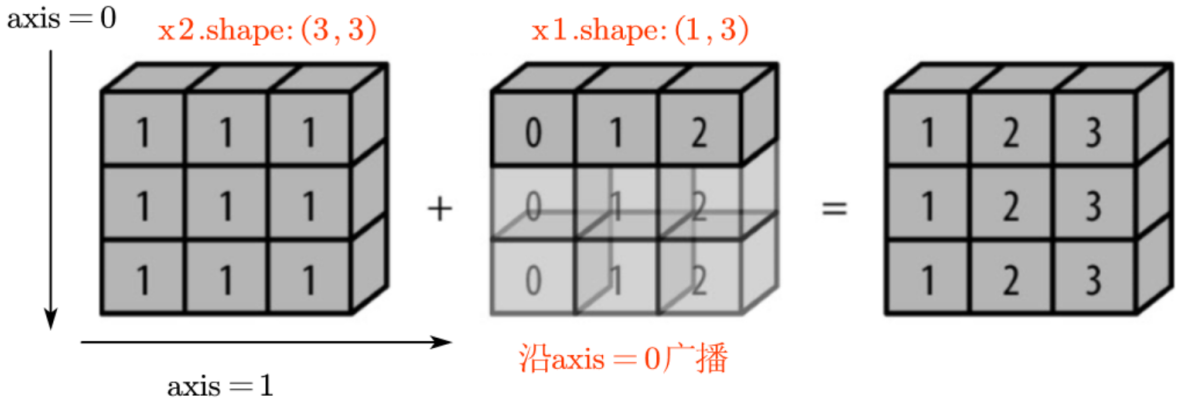
广播机制同样适用于更高维数组之间的运算。下面给出一维数组和二维数组的通用函数运算(相加)。

```
# 输出一维数组的维度和shape
x1 = np.arange(3)
print(x1, x1.shape, x1.ndim)

# 输出二维数组的维度和shape
x2 = np.ones((3, 3))
print(x2, x2.shape, x2.ndim)

# 调用二元通用函数(相加)进行运算，底层调用广播机制
print(x1 + x2)
```

> 知识点：广播机制的理解。
>
> - 注意：`x1`是一维向量`.shape`为`(3,)`，**可等价视为二维数组(行向量)**，即`.shape`为`(1,3)`。下面对`x1`的分析都采用行向量。
> - `x1`的`.shape`为`(1,3)`和`x2`的`.shape`为`(3,3)`。
> - 参与二元通用函数运算的`x1`和`x2`不匹配。
> - 广播机制直观理解(结合下图理解)：
> - - 观察可知，`x1`在第1维度(`axis=0`)的`.shape`和`x2`不匹配。
>   - 那么NumPy对`x1`将沿着第1维度扩展至`.shape`为`(3,3)`的二维数组(类似复制元素的方式)。
>   - 此时新的`x1`的`.shape`为`(3,3)`，和`x2`匹配，满足通用函数运算条件。
>   - 然后，执行二元通用函数(相加)运算。



浅色的盒子表示广播的值。

任务：两个数组同时广播

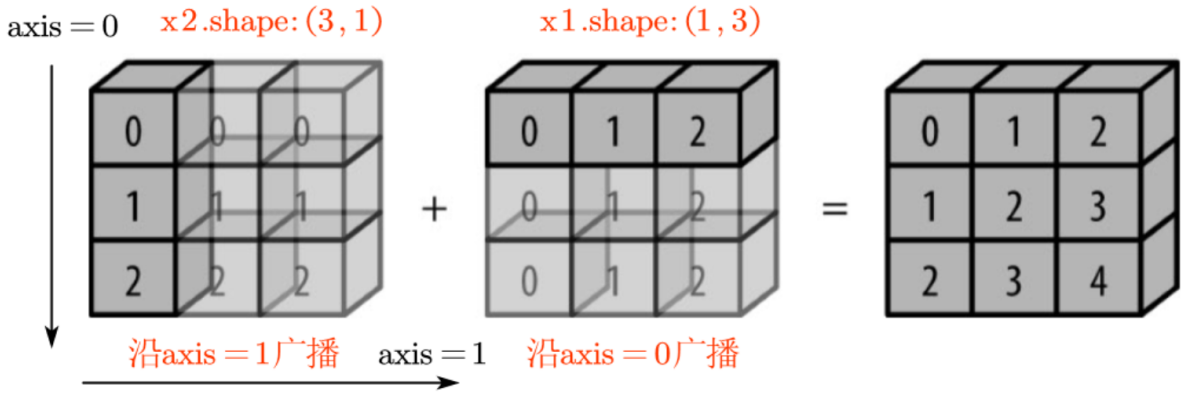
上面两个例子都是对其中的一个数组进行广播。下面，将学习更复杂的两个数组同时广播。

```python
# 输出一维数组的维度和shape
x1 = np.arange(3)
print(x1, x1.shape, x1.ndim)

# 将x1通过np.newaxis升维为二维数组(列向量)
# np.newaxis如何升维请自行回顾(之前知识点)
x2 = x1[:, np.newaxis]
# 输出二维数组(列向量)的维度和shape
print(x2, x2.shape, x2.ndim)

# 调用二元通用函数(相加)进行运算，底层调用广播机制
print(x1 + x2)
```

> 知识点：广播机制的理解。
>
> - 注意：`x1`是一维向量`.shape`为`(3,)`，**在NumPy中可等价视为二维数组(行向量)**，即`.shape`为`(1,3)`。下面对`x1`的分析都采用行向量。
> - `x1`的`.shape`为`(1,3)`和`x2`的`.shape`为`(3,1)`。
> - 参与二元通用函数运算的`x1`和`x2`不匹配。
> - 广播机制直观理解(结合下图理解)：
> -
>   - 观察可知，`x1`和`x2`在两个维度上都不匹配。
>   - 首先，NumPy对`x1`将沿着第1维度扩展至`.shape`为`(3,3)`的二维数组(类似复制元素的方式)。
>   - 然后，NumPy对`x2`将沿着第2维度扩展至`.shape`为`(3,3)`的二维数组(类似复制元素的方式)。
>   - 此时新的`x1`的`.shape`为`(3,3)`，和`x2`的`.shape`为`(3,3)`，它们相互匹配，满足通用函数运算条件。
>   - 然后，执行二元通用函数(相加)运算。



浅色的盒子表示广播的值。

6.2.3 (重要)广播的两大规则

任务：掌握广播两大规则

> 知识点：并非所有的数组都能使用NumPy的广播功能。NumPy设定了一组严格规则，只有满足了才能进行数组广播。
>
> - 规则1(`.ndim`不匹配)：如果两个数组`.ndim`维度数不相同，那么较小`.ndim`的数组`.shape`将会在**最左边补1**。
> - 规则2(`.shape`不匹配)：如果两个数组`.shape`存在不匹配。检查不匹配的那维`.shape`**小的值**是否等于1，如果是，那么就按大的`.shape`值进行广播；如果为否，就会出现异常。
> - 口诀：先看`.ndim`；后看`.shape`，检查不匹配维度是否有1；都不满足，那么就会异常。

任务：一个数组的广播

二维数组与一维数组相加。

```
x1 = np.ones((2, 3))
print(x1.ndim, x1.shape,'\n', x1)

x2 = np.arange(3)
print(x2.ndim, x2.shape,'\n', x2)

print(x1 + x2)
```

请仔细看下面的广播规则分析，掌握2个规则的适用场合。

> 广播规则分析：
>
> - `x1`的`.ndim: 2`，`.shape: (2,3)`。
> - `x2`的`.ndim: 1`，`.shape: (3,)`。
> - `x2`应用规则1：由于`x2`的`.ndim`小于`x1`，将`x2`的`.shape`左边补1为`(1,3)`。
> - `x2`应用规则2：由于新`x2`的`.shape: (1,3)`，和`x1`在第1维上不匹配，且`x2`在该维的`.shape`为1，满足规则2。因此，将`x2`广播为`.shape: (2,3)`。
> - 最终两个数组`.shape: (2,3)`，满足通用函数运算要求。

任务：两个数组的广播

行向量和列向量相加。

```
x1 = np.arange(3).reshape((3, 1))
print(x1.ndim, x1.shape,'\n', x1)

x2 = np.arange(3)
print(x2.ndim, x2.shape,'\n', x2)

print(x1 + x2)
```

> 广播规则分析：

- `x1`的`.ndim: 2`，`.shape: (3,1)`。
- `x2`的`.ndim: 1`，`.shape: (3,)`。
- `x2`应用规则1：由于`x2`的`.ndim`小于`x1`，将`x2`的`.shape`左边补1为`(1,3)`。
- `x2`应用规则2：由于新`x2`的`.shape: (1,3)`，和`x1`在第1维上不匹配，且`x2`在该维的`.shape`为1，满足规则2。因此，将`x2`广播为`.shape: (3,3)`。
- `x1`应用规则2：由于新`x2`的`.shape: (3,3)`，和`x1`在第2维上不匹配，且`x1`在该维的`.shape`为1，满足规则2。因此，将`x1`广播为`.shape: (3,3)`。
- 最终两个数组`.shape: (3,3)`，满足通用函数运算要求。

任务：广播异常处理

两个数组广播如果都不满足上述两条规则，那么就会发生异常。

```
x1 = np.ones((3, 2))
print(x1.ndim, x1.shape,'\n', x1)

x2 = np.arange(3)
print(x2.ndim, x2.shape,'\n', x2)

print(x1 + x2)
```

广播规则分析：

- `x1`的`.ndim: 2`，`.shape: (3,2)`。
- `x2`的`.ndim: 1`，`.shape: (3,)`。
- `x2`应用规则1：由于`x2`的`.ndim`小于`x1`，将`x2`的`.shape`左边补1为`(1,3)`。
- `x2`应用规则2：由于新`x2`的`.shape: (1,3)`，和`x1`在第1维上不匹配，且`x2`在该维的`.shape`为1，满足规则2。因此，将`x2`广播为`.shape: (3,3)`。
- `x1`应用规则2：由于新`x2`的`.shape: (3,3)`，和`x1`在第2维上不匹配，且`x1`在该维的`.shape`为2，**不满足规则2**(小的`.shape`必需要为1)。
- 程序将会报错(Traceback指出广播异常)。不满足通用函数运算要求。

任务：手动升维来满足广播条件

注意：

- 如果能在`x1`数组的右边补1，而不是在左边补1，那么`x1`和`x2`后续的`.shape`就会变得兼容。
- **但是这不被广播的规则所允许，因为不满足规则1(必需左边补1)。**
- 如果逻辑上允许右边补1，那么可以数组变形函数`reshape()`或`np.newaxis`的方式手动升维。

```
x1 = np.ones((3, 2))
print(x1.ndim, x1.shape,'\n', x1)

# 使用np.newaxis在第2维上升维
# 可合并为x2 = np.arange(3)[:, np.newaxis]
x2 = np.arange(3)
x2 = x2[:, np.newaxis]
print(x2.ndim, x2.shape,'\n', x2)

print(x1 + x2)
```

6.2.4实例：广播的应用

应用广播：数组归一化处理

> 知识点：
>
> - NumPy向量化技术包括：通用函数和广播技术。
> - 通用函数让NumPy用户免于低效的Python循环。
> - 广播技术扩展了NumPy向量化适用范围。

> 知识点：在数据预处理中，通常都需要对**数据进行归一化。**那么为何需要数据归一化?
>
> - 数据由样本组成，而样本是由特征所组成。
> - 通常，数组的行代表样本，列代表特征。(联想Excel表格)。
> - 例如，现在有一个身高体重的数据集，那么样本代表一条记录，每个样本都是有2个特征(身高和体重)组成。
> - 由于每个特征的量纲(单位)不一样，如果直接使用原始数据处理，将会对某些数值较小的特征造成偏见。为此，需要使用**数据归一化技术，将每个特征都进行无量纲化。**

假设数据集有10个样本，每个样本包含3个特征(属性或数值)，那么可以使用10×3的数组存放该数据。

```
# 随机生成10×3的数据数组
# 数组每行为1个样本，每个列为数据的1个特征。
X = np.random.random((10, 3))
print(X)

# 调用通用函数mean()沿第0轴(样本)计算均值(聚合)
Xmean = X.mean(axis=0)
print('mean:', Xmean)
```

```
# 中心化处理：对数组每个行加去特征矩阵
# 思考这里是如何进行广播的？
X_centered = X - Xmean

# 输出中心化后，每个特征的均值
# 查看归一化的数组的均值是否接近0。
# 计算机是
X_centered.mean(0)
```

计算机是对浮点数表示存在误差。在机器精度范围内，上述均值为 0。

应用广播：生成二维函数

下面将通过广播技术，生成一个二维函数，并绘制该函数图像。

定义函数, 和的定义域为`[0, 50)`。这里将`x`作为第1维坐标，`y`作为第2维坐标。使用`np.linspace()`生成数据。

```
# x和y表示0~5区间50个步长的序列
# 二维数组的列向量(axis=0)
x = np.linspace(0, 5, 50)[:, np.newaxis]

# 一维数组，广播规则1，等价于
# 二维数组的行向量(axis=1)
# np.linspace(0, 5, 50)[np.newaxis,:]
y = np.linspace(0, 5, 50)
```

```
# 根据z的二元函数，生成函数值
# 思考：z的`.shape`是多少？
# 思考广播技术是如何进行的？
z = np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)
```

```
# 使用matplotlib模块绘制函数图像
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn; seaborn.set()  # 设置绘图风格

# 绘制函数等高线
fig, ax = plt.subplots()
CS = plt.contour(z)
# 设置等高线上的文本
ax.clabel(CS, inline=1, fontsize=10)
```