Numpy

第一章 量的定义

> 定义n维数组，并且在数组上进行简单的变换与操作。

1 名称

- 标量，单个数据，零阶张量，
- 向量，一维数组，一阶张量，
- 矩阵，二维数组，二阶张量
- 张量，高维数组，张量，

2 关系

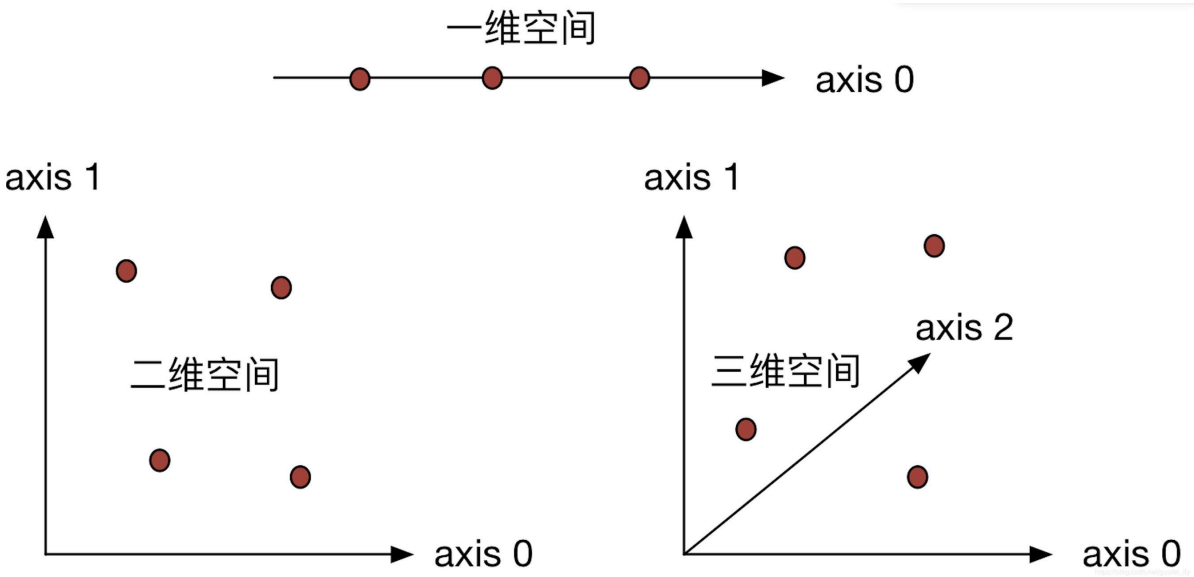
- 可以使用向量，来定义n维线性空间、n维向量空间。向量，以数学的方式描述n维线性空间。标量、向量、矩阵、张量是对n维线性空间的暴力展开。
- 维数：数组总共有多少个维度。3维
- 维度：数组每个维的长度是多少。维度是(2,3,4)
- 范数：用来衡量数组的特征。F1范数，绝对值之和。F2范数，平方和。
- 列表和张量不同。列表的低维的维度可以不同。[[1],[1,2]]。张量，相同维的维度必须一致。
- 对于张量的描述可以使三阶，一阶二维，二阶三维，三阶四维。
- 对于数组的描述，应该是三维数组，一维维度是2，二维维度是3，三维维度是4
- 向量描述n维线性空间，向量的维度描述线性空间维度的个数，向量的数据描述每个维度的大小。
- 高维是外层的，低维是内层的。高维包含多个低维。低维能锁定更精确的数据。高维可以索引低维。

3 运算

- 同维度的四则运算，对应位运算。
- 不同维度的四则运算，进行广播。
- 点乘，同维度，同位置相乘相加。
  点乘，不同维度，

4 维度说明

ndarray(多维数组)是Numpy处理的数据类型。多维数组的维度即为对应数据所在的空间维度，1维可以理解为直线空间，2维可以理解为平面空间，3维可以理解为立方体空间。



- 轴是用来对多维数组所在空间进行定义、描述的一组正交化的直线，根据数学惯例可以用i,j,ki, j ,ki,j,k来表示。
- 在一维空间中，用一个轴就可以表示清楚，numpy中规定为axis 0，空间内的数可以理解为直线空间上的离散点 (x i,)。
- 在二维空间中，需要用两个轴表示，numpy中规定为axis 0和axis 1，空间内的数可以理解为平面空间上的离散点（x i，y j）。
- 在三维空间中，需要用三个轴才能表示清楚，在二维空间的基础上numpy中又增加了axis 2，空间内的数可以理解为立方体空间上的离散点（x i，y j，z k）。

> Python中可以用numpy中的ndim和shape来分别查看维度，以及在对应维度上的长度。直观上可以根据符号“[ ]”的层数来判断，有m层即为m维，最外面1层对应axis0， 依次为axis1，axis2…

第二章 数据类型

1 ndarray对象

- N 维数组对象 ndarray，它是一系列同类型数据的集合，以 0 下标为开始进行集合中元素的索引。
- ndarray 对象是用于存放同类型元素的多维数组。
- ndarray 中的每个元素在内存中都有相同存储大小的区域。

2 ndarray定义

```
1    numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)
```

名称	描述
object	数组或嵌套的数列
dtype	数组元素的数据类型，可选
copy	对象是否需要复制，可选
order	创建数组的样式，C为行方向，F为列方向，A为任意方向（默认）
subok	默认返回一个与基类类型一致的数组
ndmin	指定生成数组的最小维度

3 ndarray数据类型

名称	描述
bool_	布尔型数据类型（True 或者 False）
int_	默认的整数类型（类似于 C 语言中的 long，int32 或 int64）
intc	与 C 的 int 类型一样，一般是 int32 或 int 64
intp	用于索引的整数类型（类似于 C 的 ssize_t，一般情况下仍然是 int32 或 int64）
int8	字节（-128 to 127）
int16	整数（-32768 to 32767）
int32	整数（-2147483648 to 2147483647）
int64	整数（-9223372036854775808 to 9223372036854775807）
uint8	无符号整数（0 to 255）
uint16	无符号整数（0 to 65535）
uint32	无符号整数（0 to 4294967295）
uint64	无符号整数（0 to 18446744073709551615）
float_	float64 类型的简写
float16	半精度浮点数，包括：1 个符号位，5 个指数位，10 个尾数位
float32	单精度浮点数，包括：1 个符号位，8 个指数位，23 个尾数位
float64	双精度浮点数，包括：1 个符号位，11 个指数位，52 个尾数位
complex_	complex128 类型的简写，即 128 位复数
complex64	复数，表示双 32 位浮点数（实数部分和虚数部分）
complex128	复数，表示双 64 位浮点数（实数部分和虚数部分）

4 ndarray数据类型操作

方法	描述
can_cast (from_, to[, casting])	如果根据强制转换规则可以在数据类型之间进行强制转换，则返回True。
promote_types (type1, type2)	返回Type1和Type2都可以安全强制转换为的最小大小和最小标量种类的数据类型。
min_scalar_type (a)	对于标量a，返回具有最小大小和可以保存其值的最小标量种类的数据类型。
result_type (*arrays_and_dtypes)	返回将NumPy类型提升规则应用于参数而得到的类型。
common_type (*arrays)	返回输入数组通用的标量类型。
obj2sctype (rep[, default])	返回对象的Python类型的标量dtype或NumPy等效值。

5 创建数据类型

方法	描述
dtype (obj[, align, copy])	创建数据类型对象。
format_parser (formats, names, titles[, …])	类将格式、名称、标题说明转换为dtype。

6 数据类型信息

方法	描述
finfo (dtype)	浮点类型的机器限制。
iinfo (type)	整数类型的机器限制。
MachAr ([float_conv, int_conv, …])	诊断机器参数。

7 数据类型测试

方法	描述
issctype (rep)	确定给定对象是否表示标量数据类型。
issubdtype (arg1, arg2)	如果第一个参数是类型层次结构中较低/等于的类型码，则返回True。
issubsctype (arg1, arg2)	确定第一个参数是否是第二个参数的子类。
issubclass_ (arg1, arg2)	确定一个类是否是第二个类的子类。
find_common_type (array_types, scalar_types)	按照标准强制规则确定常见类型。

8 杂项

方法	描述
typename (char)	返回给定数据类型代码的说明。
sctype2char (sctype)	返回标量dtype的字符串表示形式。
mintypecode (typechars[, typeset, default])	返回给定类型可以安全强制转换到的最小大小类型的字符。
maximum_sctype (t)	返回与输入类型相同精度最高的标量类型。

第三章 数组属性

1 基本概念

NumPy 数组的维数称为秩（rank），秩就是轴的数量，即数组的维数，一维数组的秩为 1，二维数组的秩为 2，以此类推。

很多时候可以声明 axis。axis=0，表示沿着第 0 轴进行操作，即对每一列进行操作；axis=1，表示沿着第1轴进行操作，即对每一行进行操作。

- 轴=秩=维数
- 第一维是高维，最后一维是低维。

2 数组属性

属性	说明
`ndarray.ndim`	秩，即轴的数量或维度的数量
`ndarray.shape`	数组的维度，对于矩阵，n 行 m 列
`ndarray.size`	数组元素的总个数，相当于 .shape 中 n*m 的值
`ndarray.dtype`	ndarray 对象的元素类型
`ndarray.itemsize`	ndarray 对象中每个元素的大小，以字节为单位
`ndarray.flags`	ndarray 对象的内存信息
`ndarray.real`	ndarray元素的实部
`ndarray.imag`	ndarray 元素的虚部
`ndarray.data`	包含实际数组元素的缓冲区，由于一般通过数组的索引获取元素，所以通常不需要使用这个属性。

第四章 创建数组

1 填充创建

方法	描述
empty (shape[, dtype, order])	返回给定形状和类型的新数组，而无需初始化条目。
empty_like (prototype[, dtype, order, subok, …])	返回形状和类型与给定数组相同的新数组。
eye (N[, M, k, dtype, order])	返回一个二维数组，对角线上有一个，其他地方为零。
identity (n[, dtype])	返回标识数组。
ones (shape[, dtype, order])	返回给定形状和类型的新数组，并填充为1。
ones_like (a[, dtype, order, subok, shape])	返回形状与类型与给定数组相同的数组。
zeros (shape[, dtype, order])	返回给定形状和类型的新数组，并用零填充。
zeros_like (a[, dtype, order, subok, shape])	返回形状与类型与给定数组相同的零数组。
full (shape, fill_value[, dtype, order])	返回给定形状和类型的新数组，并用fill_value填充。
full_like (a, fill_value[, dtype, order, …])	返回形状和类型与给定数组相同的完整数组。

2 从现有数据创建

方法	描述
array (object[, dtype, copy, order, subok, ndmin])	创建一个数组。
asarray (a[, dtype, order])	将输入转换为数组。
asanyarray (a[, dtype, order])	将输入转换为ndarray，但通过ndarray子类。
ascontiguousarray (a[, dtype])	返回内存中的连续数组（ndim > = 1）（C顺序）。
asmatrix (data[, dtype])	将输入解释为矩阵。
copy (a[, order])	返回给定对象的数组副本。
frombuffer (buffer[, dtype, count, offset])	将缓冲区解释为一维数组。
fromfile (file[, dtype, count, sep, offset])	根据文本或二进制文件中的数据构造一个数组。
fromfunction (function, shape, **kwargs)	通过在每个坐标上执行一个函数来构造一个数组。
fromiter (iterable, dtype[, count])	从可迭代对象创建一个新的一维数组。
fromstring (string[, dtype, count, sep])	从字符串中的文本数据初始化的新一维数组。
loadtxt (fname[, dtype, comments, delimiter, …])	从文本文件加载数据。

3 创建记录的数组

方法	描述
core.records.array (obj[, dtype, shape, …])	从各种各样的对象构造一个记录数组。
core.records.fromarrays (arrayList[, dtype, …])	从（平面）数组列表创建记录数组
core.records.fromrecords (recList[, dtype, …])	从文本格式的记录列表创建一个rearray
core.records.fromstring (datastring[, dtype, …])	根据字符串中包含的二进制数据创建（只读）记录数组
core.records.fromfile (fd[, dtype, shape, …])	根据二进制文件数据创建数组

4 创建字符数组

方法	描述
core.defchararray.array (obj[, itemsize, …])	创建一个chararray。
core.defchararray.asarray (obj[, itemsize, …])	将输入转换为chararray，仅在必要时复制数据。

5 数值范围

方法	描述
arange ([start,] stop[, step,][, dtype])	返回给定间隔内的均匀间隔的值。
np.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None)	返回指定间隔内的等间隔数字。
logspace (start, stop[, num, endpoint, base, …])	返回数以对数刻度均匀分布。
geomspace (start, stop[, num, endpoint, …])	返回数字以对数刻度（几何级数）均匀分布。
meshgrid (*xi, **kwargs)	从坐标向量返回坐标矩阵。
mgrid	nd_grid实例，它返回一个密集的多维“ meshgrid”。
ogrid	nd_grid实例，它返回一个开放的多维“ meshgrid”。

6 创建矩阵

方法	描述
diag (v[, k])	提取对角线或构造对角线数组。
diagflat (v[, k])	使用展平的输入作为对角线创建二维数组。
tri (N[, M, k, dtype])	在给定对角线处及以下且在其他位置为零的数组。
tril (m[, k])	数组的下三角。
triu (m[, k])	数组的上三角。
vander (x[, N, increasing])	生成范德蒙矩阵。

7 定义矩阵

方法	描述
mat (data[, dtype])	将输入解释为矩阵。
bmat (obj[, ldict, gdict])	从字符串，嵌套序列或数组构建矩阵对象。

第五章 索引迭代

1 索引

- 有三种可用的索引方法类型： 字段访问，基本切片和高级索引
- 所有的索引方式都是在方括号内。
  - :表示切片
  - ,表示高维度索引
  - [][]表示递归索引，对索引结果再次索引。

2 字段访问

ndarray对象的内容可以通过索引或切片来访问和修改，与 Python 中 list 的切片操作一样。

ndarray 数组可以基于 0 - n 的下标进行索引。

```
import numpy as np

a = np.arange(10)
s = slice(2,7,2)    # 从索引 2 开始到索引 7 停止，间隔为2
print (a[s])
```

3 基本切片

切片对象可以通过内置的 slice 函数，并设置 start, stop 及 step 参数进行，从原数组中切割出一个新数组。

- 基本切片语法是 i:j:k，其中 i 是起始索引，j 是停止索引，k 是步骤（k\neq0）。这将选择具有索引值（在相应的维度中）i, i+k, ..., i+(m-1) k 的 m 个元素，

```
import numpy as np

a = np.arange(10)
b = a[2:7:2]    # 从索引 2 开始到索引 7 停止，间隔为 2
print(b)
```

- 负 i 和 j 被解释为 n + i 和 n + j，其中 n 是相应维度中的元素数量。负 k 使得踩踏指向更小的指数。

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

- 切片还可以包括省略号 …，来使选择元组的长度与数组的维度相同。 如果在行位置使用省略号，它将返回包含行中元素的 ndarray。Ellipsis扩展:为选择元组索引所有维度所需的对象数。在大多数情况下，这意味着扩展选择元组的长度是x.ndim。可能只存在一个省略号。

```
import numpy as np

a = np.array([[1,2,3],[3,4,5],[4,5,6]])
print (a[...,1])   # 第2列元素
print (a[1,...])   # 第2行元素
print (a[...,1:])  # 第2列及剩下的所有元素
```

4 高级索引

数组可以由整数数组索引、布尔索引及花式索引。

4.1 整数数组索引

当索引包含尽可能多的整数数组时，索引的数组具有维度，索引是直接的，但与切片不同。

- 高级索引始终作为 一个 整体进行 广播 和迭代：

```
result[i_1, ..., i_M] == x[ind_1[i_1, ..., i_M], ind_2[i_1, ..., i_M],..., ind_N[i_1, ..., i_M]]
```

- 应从每一行中选择特定的元素。 行索引只是[0, 1, 2]，列索引指定要为相应行选择的元素，这里是[0, 1, 0]。 将两者结合使用，可以使用高级索引解决任务

```
import numpy as np

x = np.array([[1,  2],  [3,  4],  [5,  6]])
y = x[[0,1,2],  [0,1,0]]
print (y)

>>> [1,4,5]
```

4.2 布尔索引

我们可以通过一个布尔数组来索引目标数组。

布尔索引通过布尔运算（如：比较运算符）来获取符合指定条件的元素的数组。

```
import numpy as np

x = np.array([[  0,  1,  2],[  3,  4,  5],[  6,  7,  8],[  9,  10,  11]])
print ('我们的数组是：')
print (x)
print ('\n')
# 现在我们会打印出大于 5 的元素
print  ('大于 5 的元素是：')
print (x[x >  5])

我们的数组是：
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]

大于 5 的元素是：
[ 6  7  8  9 10 11]
```

4.3 花式索引

花式索引指的是利用整数数组进行索引。

- 花式索引根据索引数组的值作为目标数组的某个轴的下标来取值。对于使用一维整型数组作为索引，如果目标是一维数组，那么索引的结果就是对应位置的元素；如果目标是二维数组，那么就是对应下标的行。花式索引跟切片不一样，它总是将数据复制到新数组中。
- 传入顺序索引数组

```
import numpy as np

x=np.arange(32).reshape((8,4))
print (x[[4,2,1,7]])
输出结果为：

[[16 17 18 19]
 [ 8  9 10 11]
 [ 4  5  6  7]
 [28 29 30 31]]
```

- 传入倒序索引数组

```
import numpy as np

x=np.arange(32).reshape((8,4))
print (x[[-4,-2,-1,-7]])
输出结果为：

[[16 17 18 19]
 [24 25 26 27]
 [28 29 30 31]
 [ 4  5  6  7]]
```

5 迭代数组

5.1 迭代器

NumPy 迭代器对象 numpy.nditer 提供了一种灵活访问一个或者多个数组元素的方式。

```
import numpy as np

a = np.arange(6).reshape(2,3)
print ('原始数组是：')
print (a)
print ('\n')
print ('迭代输出元素：')
for x in np.nditer(a):
      print (x, end=", " )
print ('\n')
```

5.2 按顺序迭代

这反映了默认情况下只需访问每个元素，而无需考虑其特定顺序。我们可以通过迭代上述数组的转置来看到这一点，并与以 C 顺序访问数组转置的 copy 方式做对比，如下实例：

```
import numpy as np

a = np.arange(6).reshape(2,3)
for x in np.nditer(a.T):
      print (x, end=", " )
print ('\n')

for x in np.nditer(a.T.copy(order='C')):
      print (x, end=", " )
print ('\n')
```

控制遍历顺序

`for x in np.nditer(a, order='F'):Fortran order`，即是列序优先；
`for x in np.nditer(a.T, order='C'):C order`，即是行序优先

```
import numpy as np

a = np.arange(0,60,5)
a = a.reshape(3,4)
print ('原始数组是：')
print (a)
print ('\n')
print ('原始数组的转置是：')
b = a.T
print (b)
print ('\n')
print ('以 C 风格顺序排序：')
c = b.copy(order='C')
print (c)
for x in np.nditer(c):
      print (x, end=", " )
print  ('\n')
print  ('以 F 风格顺序排序：')
c = b.copy(order='F')
print (c)
for x in np.nditer(c):
      print (x, end=", " )
输出结果为：

原始数组是：
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


原始数组的转置是：
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]


以 C 风格顺序排序：
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]
0, 20, 40, 5, 25, 45, 10, 30, 50, 15, 35, 55,

以 F 风格顺序排序：
[[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]
0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
```

5.3 修改迭代的值

修改数组中元素的值
nditer 对象有另一个可选参数 op_flags。 默认情况下，nditer 将视待迭代遍历的数组为只读对象（read-only），为了在遍历数组的同时，实现对数组元素值得修改，必须指定 read-write 或者 write-only 的模式。

```
import numpy as np

a = np.arange(0,60,5)
a = a.reshape(3,4)
print ('原始数组是：')
print (a)
print ('\n')
for x in np.nditer(a, op_flags=['readwrite']):
      x[...]=2*x
print ('修改后的数组是：')
print (a)
输出结果为：

原始数组是：
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


修改后的数组是：
[[  0  10  20  30]
 [ 40  50  60  70]
 [ 80  90 100 110]]
```

5.4 广播迭代

广播迭代
如果两个数组是可广播的，nditer 组合对象能够同时迭代它们。 假设数组 a 的维度为 3X4，数组 b 的维度为 1X4，则使用以下迭代器（数组 b 被广播到 a 的大小）。

```
import numpy as np

a = np.arange(0,60,5)
a = a.reshape(3,4)
print  ('第一个数组为：')
print (a)
print  ('\n')
print ('第二个数组为：')
b = np.array([1,  2,  3,  4], dtype =  int)
print (b)
print ('\n')
print ('修改后的数组为：')
for x,y in np.nditer([a,b]):
      print ("%d:%d"  %  (x,y), end=", " )
输出结果为：

第一个数组为：
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]
```

```
21
22
23    第二个数组为：
24    [1 2 3 4]
25
26
27    修改后的数组为：
28    0:1, 5:2, 10:3, 15:4, 20:1, 25:2, 30:3, 35:4, 40:1, 45:2, 50:3, 55:4,
```

第六章 数组操作

> 主要对数组本身的进行更改。并不是进行运算。

1 复制数组

方法	描述
copyto (dst, src[, casting, where])	将值从一个数组复制到另一个数组，并根据需要进行广播。

2 改变形状

方法	描述
reshape (a, newshape[, order])	在不更改数据的情况下为数组赋予新的形状。
ravel (a[, order])	返回一个连续的扁平数组。
ndarray.flat	数组上的一维迭代器。
ndarray.flatten ([order])	返回折叠成一维的数组副本。

3 转置数组

方法	描述
moveaxis (a, source, destination)	将数组的轴移到新位置。
rollaxis (a, axis[, start])	向后滚动指定的轴，直到其位于给定的位置。
swapaxes (a, axis1, axis2)	互换数组的两个轴。
ndarray.T	转置数组。
transpose (a[, axes])	排列数组的尺寸。

4 更改维度数

方法	描述
atleast_1d (*arys)	将输入转换为至少一维的数组。
atleast_2d (*arys)	将输入视为至少具有二维的数组。
atleast_3d (*arys)	以至少三个维度的数组形式查看输入。
broadcast	产生模仿广播的对象。
broadcast_to (array, shape[, subok])	将数组广播为新形状。
broadcast_arrays (*args, **kwargs)	互相广播任意数量的阵列。
expand_dims (a, axis)	扩展数组的形状。
squeeze (a[, axis])	从数组形状中删除一维条目。

5 改变种类

方法	描述
asarray (a[, dtype, order])	将输入转换为数组。
asanyarray (a[, dtype, order])	将输入转换为ndarray，但通过ndarray子类。
asmatrix (data[, dtype])	将输入解释为矩阵。
asfarray (a[, dtype])	返回转换为浮点类型的数组。
asfortranarray (a[, dtype])	返回以Fortran顺序排列在内存中的数组（ndim> = 1）。
ascontiguousarray (a[, dtype])	返回内存中的连续数组（ndim> = 1）（C顺序）。
asarray_chkfinite (a[, dtype, order])	将输入转换为数组，检查NaN或Infs。
asscalar (a)	将大小为1的数组转换为其等效的标量。
require (a[, dtype, requirements])	返回提供的类型满足要求的ndarray。

6 组合数组

方法	描述
concatenate ((a1, a2, …)	沿现有轴连接一系列数组。
stack (arrays[, axis, out])	沿新轴连接一系列数组。
column_stack (tup)	将一维数组作为列堆叠到二维数组中。
dstack (tup)	沿深度方向（沿第三轴）按顺序堆叠数组。
hstack (tup)	水平（按列）顺序堆叠数组。
vstack (tup)	垂直（行）按顺序堆叠数组。
block (arrays)	从块的嵌套列表中组装一个nd数组。

7 拆分数组

方法	描述
split (ary, indices_or_sections[, axis])	将数组拆分为多个子数组，作为ary的视图。
array_split (ary, indices_or_sections[, axis])	将一个数组拆分为多个子数组。
dsplit (ary, indices_or_sections)	沿第3轴（深度）将数组拆分为多个子数组。
hsplit (ary, indices_or_sections)	水平（按列）将一个数组拆分为多个子数组。
vsplit (ary, indices_or_sections)	垂直（行）将数组拆分为多个子数组。

8 平铺数组

方法	描述
tile (A, reps)	通过重复A代表次数来构造一个数组。
repeat (a, repeats[, axis])	重复数组的元素。

9 添加和删除元素

方法	描述
delete (arr, obj[, axis])	返回一个新的数组，该数组具有沿删除的轴的子数组。
insert (arr, obj, values[, axis])	沿给定轴在给定索引之前插入值。
append (arr, values[, axis])	将值附加到数组的末尾。
resize (a, new_shape)	返回具有指定形状的新数组。
trim_zeros s(filt[, trim])	修剪一维数组或序列中的前导和/或尾随零。
unique (ar[, return_index, return_inverse, …])	查找数组的唯一元素。

10 重新排列元素

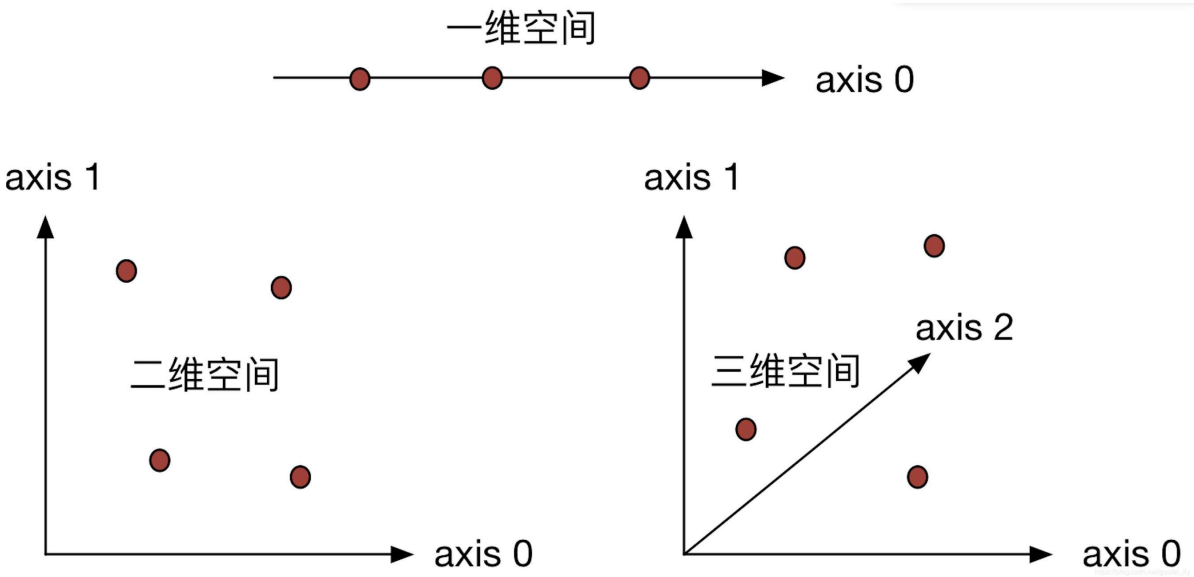
方法	描述
flip (m[, axis])	沿给定轴颠倒数组中元素的顺序。
fliplr (m)	左右翻转数组。
flipud (m)	上下翻转阵列。
reshape (a, newshape[, order])	在不更改数据的情况下为数组赋予新的形状。
roll (a, shift[, axis])	沿给定轴滚动数组元素。
rot90 (m[, k, axes])	在轴指定的平面中将阵列旋转90度。

11 数组拼接

concatenate	提供了axis参数，用于指定拼接方向
append	默认先ravel再拼接成一维数组，也可指定axis
stack	提供了axis参数，用于生成新的维度
hstack	水平拼接，沿着行的方向，对列进行拼接
vstack	垂直拼接，沿着列的方向，对行进行拼接
dstack	沿着第三个轴（深度方向）进行拼接
column_stack	水平拼接，沿着行的方向，对列进行拼接
row_stack	垂直拼接，沿着列的方向，对行进行拼接
r_	垂直拼接，沿着列的方向，对行进行拼接
c_	水平拼接，沿着行的方向，对列进行拼接

11.1 维度和轴

在正确理解Numpy中的数组拼接、合并操作之前，有必要认识下维度和轴的概念：
ndarray(多维数组)是Numpy处理的数据类型。多维数组的维度即为对应数据所在的空间维度，1维可以理解为直线空间，2维可以理解为平面空间，3维可以理解为立方体空间。



轴是用来对多维数组所在空间进行定义、描述的一组正交化的直线，根据数学惯例可以用i,j,ki, j ,ki,j,k来表示。

- 在一维空间中，用一个轴就可以表示清楚，numpy中规定为axis 0，空间内的数可以理解为直线空间上的离散点 (x iii,)。
- 在二维空间中，需要用两个轴表示，numpy中规定为axis 0和axis 1，空间内的数可以理解为平面空间上的离散点（x iii，y jjj）。
- 在三维空间中，需要用三个轴才能表示清楚，在二维空间的基础上numpy中又增加了axis 2，空间内的数可以理解为立方体空间上的离散点（x iii，y jjj，z kkk）。

Python中可以用numpy中的ndim和shape来分别查看维度，以及在对应维度上的长度。直观上可以根据符号“[ ]”的层数来判断，有m层即为m维，最外面1层对应axis0， 依次为axis1，axis2...

```
>>> a = np.array([1,2,3])
>>> a.ndim    # 一维数组
1
>>> a.shape    # 在这个维度上的长度为3
(3,)

>>> b = np.array([[1,2,3], [4,5,6]])
>>> b.ndim    # 二维数组
2
>>> b.shape    # 在axis 0 上的长度为2， 在axis 1上的长度为3.或者可以感性的理解为2行3列
(2, 3)

>>> c = np.array([[[1,2,3], [4,5,6]]])
>>> c.ndim    # 三维数组
3
>>> c.shape    # 在axis 0 上的长度为1，在axis 1上的长度为2, 在axis 2上的长度为3. 或者可以感性的理解为1层2行3列
(1, 2, 3)
```

11.2 np.concatenate()

```
concatenate(a_tuple, axis=0, out=None)
"""
参数说明：
a_tuple:对需要合并的数组用元组的形式给出
axis: 沿指定的轴进行拼接，默认0，即第一个轴
"""
```

示例

```
>>> import numpy as np
>>> ar1 = np.array([[1,2,3], [4,5,6]])
>>> ar2 = np.array([[7,8,9], [11,12,13]])
>>> ar1
array([[1, 2, 3],
       [4, 5, 6]])
>>> ar2
array([[ 7, 8, 9],
       [11, 12, 13]])

>>> np.concatenate((ar1, ar2))   # 这里的第一轴(axis 0)是行方向
array([[ 1, 2, 3],
       [ 4, 5, 6],
       [ 7, 8, 9],
       [11, 12, 13]])

>>> np.concatenate((ar1, ar2),axis=1)   # 这里沿第二个轴，即列方向进行拼接
array([[ 1, 2, 3, 7, 8, 9],
       [ 4, 5, 6, 11, 12, 13]])

>>> ar3 = np.array([[14,15,16]]) # shape为（1，3）的2维数组
>>> np.concatenate((ar1, ar3))   # 一般进行concatenate操作的array的shape需要一致，当然如果array在拼接axis方向的size不一样，也可以完成
>>> np.concatenate((ar1, ar3)) # ar3虽然在axis0方向的长度不一致，但axis1方向上一致，所以沿axis0可以拼接
array([[ 1, 2, 3],
       [ 4, 5, 6],
       [14, 15, 16]])
>>> np.concatenate((ar1, ar3), axis=1)   # ar3和ar1在axis0方向的长度不一致，所以报错
```

11.3 np.append()

```
append(arr, values, axis=None)
"""
参数说明：
arr：array_like的数据
values: array_like的数据，若axis为None，则先将arr和values进行ravel扁平化,再拼接；否则values应当与arr的shape一致，或至多
    在拼接axis的方向不一致
axis：进行append操作的axis的方向，默认无
"""
```

示例

```
>>> np.append(ar1, ar2)   # 先ravel扁平化再拼接，所以返回值为一个1维数组
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 11, 12, 13])

>>> np.append(ar1, ar2, axis=0)    # 沿第一个轴拼接，这里为行的方向
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [11, 12, 13]])

>>> np.append(ar1, ar2, axis=1)    # 沿第二个轴拼接，这里为列的方向
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 11, 12, 13]])
Python客栈送红包、纸质书
```

11.4 np.stack()

```
stack(arrays, axis=0, out=None)
"""
沿着指定的axis对arrays(每个array的shape必须一样)进行拼接，返回值的维度比原arrays的维度高1
axis：默认为0，即第一个轴，若为-1即为第二个轴
"""
```

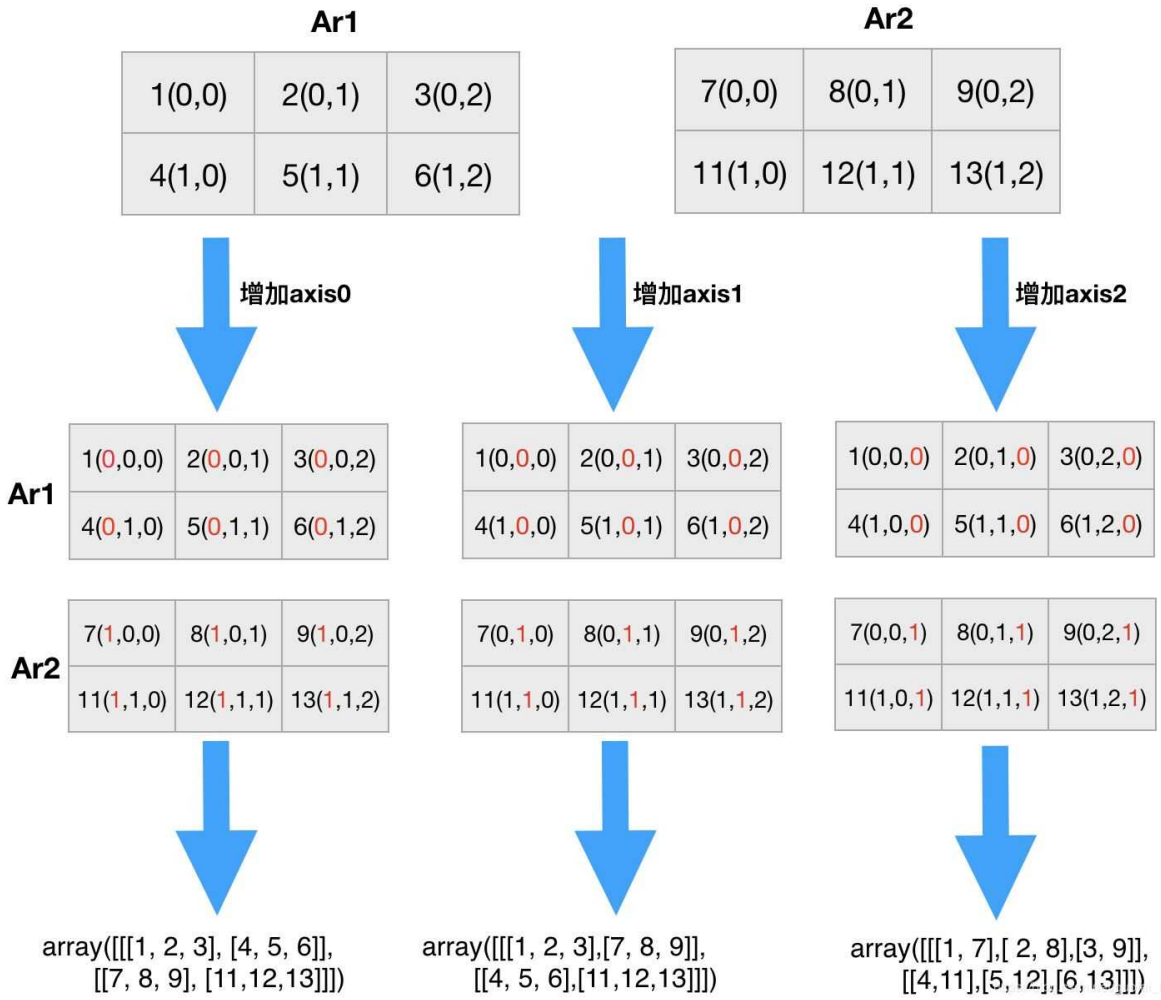
示例

```
>>> np.stack((ar1, ar2))    # 增加第一个维度（axis0，之后的axis向后顺延：0—>1，1—>2）
array([[[ 1,  2,  3],
        [ 4,  5,  6]],
       [[ 7,  8,  9],
        [11, 12, 13]]])

>>> np.stack((ar1, ar2), axis=1)    # 增加第二个维度（axis1，之后的axis向后顺延，1—>2）
array([[[ 1,  2,  3],
        [ 7,  8,  9]],
       [[ 4,  5,  6],
        [11, 12, 13]]])

>>> np.stack((ar1, ar2), axis=2)    # 增加第三个维度（axis2，和axis=-1的效果一样，原来的axis0和axis1保持不变）
array([[[ 1,  7],
        [ 2,  8],
        [ 3,  9]],
       [[ 4, 11],
        [ 5, 12],
        [ 6, 13]]])
```

关于维度增加的一种理解方式



11.5 hstack、vstack和vstack

```
>>> np.hstack((ar1,ar2))   # 水平拼接，沿着行的方向，对列进行拼接
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 11, 12, 13]])

>>> np.vstack((ar1,ar2))   # 垂直拼接，沿着列的方向，对行进行拼接
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [11, 12, 13]])

>>> np.dstack((ar1,ar2))   # 对于2维数组来说，沿着第三轴（深度方向）进行拼接，效果相当于stack(axis=-1)
array([[[ 1,  7],
        [ 2,  8],
        [ 3,  9]],
       [[ 4, 11],
        [ 5, 12],
        [ 6, 13]]])
```

11.6 column_stack和row_stack

```
>>> np.column_stack((ar1,ar2))   # 水平拼接，沿着行的方向，对列进行拼接
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 11, 12, 13]])

>>> np.row_stack((ar1,ar2))   # 垂直拼接，沿着列的方向，对行进行拼接
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [11, 12, 13]])
```

11.7 np.r_ 和np.c_

常用于快速生成ndarray数据

```
>>> np.r_[ar1,ar2]    # 垂直拼接，沿着列的方向，对行进行拼接
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [11, 12, 13]])

>>> np.c_[ar1,ar2]   # 水平拼接，沿着行的方向，对列进行拼接
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 11, 12, 13]])
```

总结

对于两个shape一样的二维array来说:

增加行（对行进行拼接）的方法有:

```
np.concatenate((ar1, ar2),axis=0)
np.append(ar1, ar2, axis=0)
np.vstack((ar1,ar2))
np.row_stack((ar1,ar2))
np.r_[ar1,ar2]
```

增加列（对列进行拼接）的方法有:

```
np.concatenate((ar1, ar2),axis=1)
np.append(ar1, ar2, axis=1)
np.hstack((ar1,ar2))
np.column_stack((ar1,ar2))
np.c_[ar1,ar2]
```

第七章 数学运算

1 三角函数

method	description
sin (x, /[, out, where, casting, order, …])	正弦函数, element-wise.
cos (x, /[, out, where, casting, order, …])	余弦函数 element-wise.
tan (x, /[, out, where, casting, order, …])	正切函数, element-wise.
arcsin (x, /[, out, where, casting, order, …])	反正弦函数, element-wise.
arccos (x, /[, out, where, casting, order, …])	反余弦函数, element-wise.
arctan (x, /[, out, where, casting, order, …])	反正切函数, element-wise.
hypot (x1, x2, /[, out, where, casting, …])	传入直角三角形的“直角边”，返回其斜边。
arctan2 (x1, x2, /[, out, where, casting, …])	x1 / x2的 Element-wise 反正切线正确选择象限。
degrees (x, /[, out, where, casting, order, …])	将角度从 **弧度** 转换为度。
radians(x, /[, out, where, casting, order, …])	将角度从度转换为弧度。
unwrap (p[, discont, axis])	通过将值之间的增量更改为2 * pi来展开。
deg2rad (x, /[, out, where, casting, order, …])	将角度从度转换为弧度。
rad2deg (x, /[, out, where, casting, order, …])	将角度从弧度转换为度。

2 双曲函数

method	description
sinh (x, /[, out, where, casting, order, …])	双曲正弦, element-wise.
cosh (x, /[, out, where, casting, order, …])	双曲余弦, element-wise.
tanh (x, /[, out, where, casting, order, …])	计算双曲正切 element-wise.
arcsinh (x, /[, out, where, casting, order, …])	反双曲正弦 element-wise.
arccosh (x, /[, out, where, casting, order, …])	反双曲余弦, element-wise.
arctanh (x, /[, out, where, casting, order, …])	反双曲正切 element-wise.

3 四舍五入

method	description
around (a[, decimals, out])	平均舍入到给定的小数位数。
round_ (a[, decimals, out])	将数组舍入到给定的小数位数。
rint (x, /[, out, where, casting, order, …])	将数组的元素四舍五入到最接近的整数。
fix (x[, out])	四舍五入为零。
floor (x, /[, out, where, casting, order, …])	返回输入的底限, element-wise.
ceil (x, /[, out, where, casting, order, …])	返回输入的上限, element-wise.
trunc (x, /[, out, where, casting, order, …])	返回输入的截断值, element-wise.

4 加法函数, 乘法函数, 减法函数

method	description
prod (a[, axis, dtype, out, keepdims, …])	返回给定轴上数组元素的乘积。
sum (a[, axis, dtype, out, keepdims, …])	给定轴上的数组元素的总和。
nanprod (a[, axis, dtype, out, keepdims])	返回数组元素在给定轴上的乘积，将非数字（NaNs）视为一个。
nansum (a[, axis, dtype, out, keepdims])	返回给定轴上的数组元素的总和，将非数字（NaNs）视为零。
cumprod (a[, axis, dtype, out])	返回沿给定轴的元素的累加乘积。
cumsum (a[, axis, dtype, out])	返回沿给定轴的元素的累加和。
nancumprod (a[, axis, dtype, out])	返回数组元素在给定轴上的累积乘积，将非数字（NaNs）视为一个。
nancumsum (a[, axis, dtype, out])	返回在给定轴上将非数字（NaNs）视为零的数组元素的累积总和。
diff (a[, n, axis, prepend, append])	计算沿给定轴的第n个离散差。
ediff1d (ary[, to_end, to_begin])	数组的连续元素之间的差值。
gradient (f, *varargs, **kwargs)	返回N维数组的梯度。
cross (a, b[, axisa, axisb, axisc, axis])	返回两个（数组）向量的叉积。
trapz (y[, x, dx, axis])	使用复合梯形规则沿给定轴积分。

5 指数和对数

method	description
exp (x, /[, out, where, casting, order, …])	计算输入数组中所有元素的指数。
expm1 (x, /[, out, where, casting, order, …])	为数组中的所有元素计算exp（x）-1。
exp2 (x, /[, out, where, casting, order, …])	为输入数组中的所有p计算2 ** p。
log (x, /[, out, where, casting, order, …])	自然对数, element-wise.
log10 (x, /[, out, where, casting, order, …])	返回输入数组的以10为底的对数, element-wise.
log2 (x, /[, out, where, casting, order, …])	x的以2为底的对数。
log1p (x, /[, out, where, casting, order, …])	返回元素加一个输入数组的自然对数。
logaddexp (x1, x2, /[, out, where, casting, …])	输入取幂之和的对数。
logaddexp2 (x1, x2, /[, out, where, casting, …])	以2为底的输入的幂和的对数。

6 其他特殊函数

method	description
i0 (x)	第一种修改的Bessel函数，阶数为0。
sinc (x)	返回sinc函数。

7 浮点

method	description
signbit (x, /[, out, where, casting, order, …])	在设置了符号位（小于零）的情况下返回 element-wise True。
copysign (x1, x2, /[, out, where, casting, …])	将x1的符号更改为x2的符号, element-wise.
frexp (x[, out1, out2], / [[, out, where, …])	将x的元素分解为尾数和二进制指数。
ldexp (x1, x2, /[, out, where, casting, …])	返回x1 * 2 ** x2, element-wise.
nextafter (x1, x2, /[, out, where, casting, …])	向x2返回x1之后的下一个浮点值, element-wise.
spacing (x, /[, out, where, casting, order, …])	返回x与最近的相邻数字之间的距离。

8 理性例程

method	description
lcm (x1, x2, /[, out, where, casting, order, …])	返回1和x2的最小公倍数
gcd (x1, x2, /[, out, where, casting, order, …])	返回x1和x2的最大公约数

9 算术运算

method	description
add (x1, x2, /[, out, where, casting, order, …])	按元素添加参数。
reciprocal (x, /[, out, where, casting, …])	以元素为单位返回参数的倒数。
positive (x, /[, out, where, casting, order, …])	数值正, element-wise.
negative (x, /[, out, where, casting, order, …])	数值负数, element-wise.
multiply (x1, x2, /[, out, where, casting, …])	逐个乘以参数。
divide (x1, x2, /[, out, where, casting, …])	返回输入的真实除法, element-wise.
power (x1, x2, /[, out, where, casting, …])	第一阵列元素从第二阵列提升为幂, element-wise.
subtract (x1, x2, /[, out, where, casting, …])	逐个元素地减去参数。
true_divide (x1, x2, /[, out, where, …])	返回输入的真实除法, element-wise.
floor_divide (x1, x2, /[, out, where, …])	返回小于或等于输入的除法的最大整数。
float_power (x1, x2, /[, out, where, …])	第一阵列元素从第二阵列提升为幂, element-wise.
fmod (x1, x2, /[, out, where, casting, …])	返回元素的除法 remainder 。
mod (x1, x2, /[, out, where, casting, order, …])	返回元素的除法余数。
modf (x[, out1, out2], / [[, out, where, …])	返回数组的分数和整数部分, element-wise.
remainder(x1, x2, /[, out, where, casting, …])	返回元素的除法余数。
divmod (x1, x2[, out1, out2], / [[, out, …])	同时返回按元素商和余数。

10 处理复数

method	description
angle (z[, deg])	返回复杂参数的角度。
real (val)	返回复杂参数的实部。
imag (val)	返回复杂参数的虚部。
conj (x, /[, out, where, casting, order, …])	返回 complex conjugate , element-wise.
conjugate (x, /[, out, where, casting, …])	返回复共轭, element-wise.

11 杂项

method	description
convolve (a, v[, mode])	返回两个一维序列的离散线性卷积。
clip (a, a_min, a_max[, out])	裁剪（限制）数组中的值。
sqrt (x, /[, out, where, casting, order, …])	返回数组的非负 **平方** 根, element-wise.
cbrt (x, /[, out, where, casting, order, …])	返回数组的立方根, element-wise.
square (x, /[, out, where, casting, order, …])	返回输入的元素方平方。
absolute (x, /[, out, where, casting, order, …])	计算绝对值 element-wise.
fabs (x, /[, out, where, casting, order, …])	计算绝对值 element-wise.
sign (x, /[, out, where, casting, order, …])	返回数字符号的逐元素指示。
heaviside (x1, x2, /[, out, where, casting, …])	计算Heaviside阶跃函数。
maximum (x1, x2, /[, out, where, casting, …])	数组元素的逐元素最大值。
minimum (x1, x2, /[, out, where, casting, …])	数组元素的按元素最小值。
fmax (x1, x2, /[, out, where, casting, …])	数组元素的逐元素最大值。
fmin (x1, x2, /[, out, where, casting, …])	数组元素的按元素最小值。
nan_to_num (x[, copy, nan, posinf, neginf])	用较大的有限数字（默认行为）或使用用户定义的nan，posinf和/或neginf关键字定义的数字将NaN替换为零和无穷大。
real_if_close (a[, tol])	如果复杂输入接近实数，则返回复杂数组。
interp (x, xp, fp[, left, right, period])	一维线性插值。

第八章 统计运算

1 顺序统计

method	description
amin (a[, axis, out, keepdims, initial, where])	返回数组的最小值或沿轴的最小值。
amax (a[, axis, out, keepdims, initial, where])	返回数组的最大值或沿轴的最大值。
nanmin (a[, axis, out, keepdims])	返回数组的最小值或沿轴的最小值，忽略任何 NaN。
nanmax (a[, axis, out, keepdims])	返回数组的最大值或沿轴的最大值，忽略任何 NaN。
ptp (a[, axis, out, keepdims])	返回数组的范围（即最大值-最小值）。
percentile (a, q[, axis, out, …])	沿指定轴计算数据的第 q 个百分位数。
nanpercentile (a, q[, axis, out, …])	沿指定轴计算数据的第 q个百分位数，同时忽略 nan 值。
quantile (a, q[, axis, out, overwrite_input, …])	沿指定轴计算数据的第 q 个分位数。
nanquantile (a, q[, axis, out, …])	沿指定轴计算数据的第 q 个分位数，同时忽略 nan 值。

2 平均值和方差

method	description
median (a[, axis, out, overwrite_input, keepdims])	计算沿指定轴的中位数。
average (a[, axis, weights, returned])	计算沿指定轴的加权平均值。
mean (a[, axis, dtype, out, keepdims])	沿指定轴计算算术平均值。
std (a[, axis, dtype, out, ddof, keepdims])	计算沿指定轴的标准偏差。
var (a[, axis, dtype, out, ddof, keepdims])	计算沿指定轴的方差。
nanmedian (a[, axis, out, overwrite_input, …])	沿指定轴计算中位数，同时忽略 NaN。
nanmean (a[, axis, dtype, out, keepdims])	沿指定轴计算算术平均值，忽略 NaNs。
nanstd (a[, axis, dtype, out, ddof, keepdims])	沿指定轴计算标准偏差，同时忽略 NaNs。
nanvar (a[, axis, dtype, out, ddof, keepdims])	计算沿指定轴的方差，同时忽略 NaNs。

3 相关统计

method	description
corrcoef (x[, y, rowvar, bias, ddof])	返回 Pearson 积矩相关系数。
correlate (a, v[, mode])	两个一维序列的互相关。
cov (m[, y, rowvar, bias, ddof, fweights, …])	估计给定数据和权重的协方差矩阵。

4 直方图统计

method	description
histogram (a[, bins, range, normed, weights, …])	计算一组数据的直方图。
histogram2d (x, y[, bins, range, normed, …])	计算两个数据样本的二维直方图。
histogramdd (sample[, bins, range, normed, …])	计算某些数据的多维直方图。
bincount (x[, weights, minlength])	计算非负整数数组中每个值的出现次数。
histogram_bin_edges (a[, bins, range, weights])	仅计算直方图函数使用的条柱边的函数。
digitize (x, bins[, right])	返回输入数组中每个值所属的条柱的索引。

第九章 线性代数

1 矩阵和向量积

方法	描述
dot (a, b[, out])	两个数组的点积。
linalg.multi_dot (arrays)	在单个函数调用中计算两个或更多数组的点积，同时自动选择最快的求值顺序。
vdot (a, b)	返回两个向量的点积。
inner (a, b)	两个数组的内积。
outer (a, b[, out])	计算两个向量的外积。
matmul (x1, x2, /[, out, casting, order, …])	两个数组的矩阵乘积。
tensordot (a, b[, axes])	沿指定轴计算张量点积。
einsum (subscripts, *operands[, out, dtype, …])	计算操作数上的爱因斯坦求和约定。
einsum_path (subscripts, *operands[, optimize])	通过考虑中间数组的创建，计算einsum表达式的最低成本压缩顺序。
linalg.matrix_power (a, n)	将方阵提升为(整数)n次方。
kron (a, b)	两个数组的Kronecker乘积。

2 分解

方法	描述
linalg.cholesky (a)	Cholesky分解
linalg.qr (a[, mode])	计算矩阵的QR分解。
linalg.svd (a[, full_matrices, compute_uv, …])	奇异值分解

3 矩阵特征值

方法	描述
linalg.eig (a)	计算方阵的特征值和右特征向量。
linalg.eigh (a[, UPLO])	返回复数Hermitian（共轭对称）或实对称矩阵的特征值和特征向量。
linalg.eigvals (a)	计算通用矩阵的特征值。
linalg.eigvalsh (a[, UPLO])	计算复杂的Hermitian或实对称矩阵的特征值。

4 范数和其他数字

方法	描述
linalg.norm (x[, ord, axis, keepdims])	矩阵或向量范数。
linalg.cond (x[, p])	计算矩阵的条件数。
linalg.det (a)	计算数组的行列式。
linalg.matrix_rank (M[, tol, hermitian])	使用SVD方法返回数组的矩阵的rank
linalg.slogdet (a)	计算数组行列式的符号和（自然）对数。
trace (a[, offset, axis1, axis2, dtype, out])	返回数组对角线的和。

5 解方程和逆矩阵

方法	描述
linalg.solve (a, b)	求解线性矩阵方程或线性标量方程组。
linalg.tensorsolve (a, b[, axes])	对x求解张量方程a x = b。
linalg.lstsq (a, b[, rcond])	返回线性矩阵方程的最小二乘解。
linalg.inv (a)	计算矩阵的（乘法）逆。
linalg.pinv (a[, rcond, hermitian])	计算矩阵的（Moore-Penrose）伪逆。
linalg.tensorinv (a[, ind])	计算N维数组的“逆”。

6 例外

方法	描述
linalg.LinAlgError	泛型Python-linalg函数引发的异常派生对象。

7 一次在多个矩阵上的线性代数

*1.8.0*版中的新功能

上面列出的几个线性代数例程能够一次计算几个矩阵的结果，如果它们堆叠在同一数组中的话。

这在文档中通过输入参数规范（如 `a : (..., M, M) array_like` ）表示。
这意味着，例如，如果给定输入数组 `a.shape == (N, M, M)` ，则将其解释为N个矩阵的“堆栈”，
每个矩阵的大小为M×M。类似的规范也适用于返回值，
例如行列式 `det : (...)` 。并且在这种情况下将返回形状 `det(a).shape == (N,)` 的数组。
这推广到对高维数组的线性代数操作：多维数组的最后1或2维被解释为向量或矩阵，视每个操作而定。

第十章 随机数

numpy中的random模块包含了很多方法可以用来产生随机数，这篇文章将对random中的一些常用方法做一个总结。

1、 numpy.random.rand(d0, d1, ..., dn)

- 作用：产生一个给定形状的数组（其实应该是ndarray对象或者是一个单值），数组中的值服从[0, 1)之间的均匀分布。
- 参数：d0, d, ..., dn : int，可选。如果没有参数则返回一个float型的随机数，该随机数服从[0, 1)之间的均匀分布。
- 返回值：ndarray对象或者一个float型的值
  例子：

```
# [0, 1)之间均匀分布的随机数，3行2列
a = np.random.rand(3, 2)
print(a)
# 不提供形状
b = np.random.rand()
print(b)
输出：

[[0.26054323 0.28184468]
 [0.7783674  0.71733674]
 [0.90302256 0.49303252]]
0.6022098740124009
```

2、 numpy.random.uniform(low=0.0, high=1.0, size=None)

- 作用：返回一个在区间[low, high)中均匀分布的数组，size指定形状。
- 参数：
  - low, high：float型或者float型的类数组对象。指定抽样区间为[low, high)，low的默认值为0.0，hign的默认值为1.0

- size：int型或int型元组。指定形状，如果不提供size，则返回一个服从该分布的随机数。
  例子：

```
# 在[1, 10)之间均匀抽样，数组形状为3行2列
a = np.random.uniform(1, 10, (3, 2))
print(a)
# 不提供size
b = np.random.uniform(1, 10)
print(b)
输出：

[[5.16545387 6.3769087 ]
 [9.98964899 7.88833885]
 [1.37173855 4.19855075]]
3.899250175275188
```

3、numpy.random.randn(d0, d1, ..., dn)

- 作用：返回一个指定形状的数组，数组中的值服从标准正态分布（均值为0，方差为1）。
- 参数：d0, d, ..., dn : int，可选。如果没有参数，则返回一个服从标准正态分布的float型随机数。
- 返回值：ndarray对象或者float
  例子：

```
# 3行2列
a = np.random.randn(3, 2)
print(a)
# 不提供形状
b = np.random.randn()
print(b)
输出：

[[-1.46605527  0.35434705]
 [ 0.43408199  0.02689309]
 [ 0.48041554  1.62665755]]
-0.6291254375915813
```

4、numpy.random.normal(loc=0.0, scale=1.0, size=None)

- 作用：返回一个由size指定形状的数组，数组中的值服从 μ=loc,σ=scale 的正态分布。
- 参数：
  - loc : float型或者float型的类数组对象，指定均值 μ
  - scale : float型或者float型的类数组对象，指定标准差 σ
  - size : int型或者int型的元组，指定了数组的形状。如果不提供size，且loc和scale为标量（不是类数组对象），则返回一个服从该分布的随机数。
- 输出：ndarray对象或者一个标量
  例子：

```
# 标准正态分布，3行2列
a = np.random.normal(0, 1, (3, 2))
print(a)
# 均值为1，标准差为3
b = np.random.normal(1, 3)
print(b)
输出：

[[ 0.34912031 -0.08757564]
 [-0.99753101  0.37441719]
 [ 2.68072286 -1.03663963]]
5.770831320998463
```

5、numpy.random.randint(low, high=None, size=None, dtype='l')

- 作用：返回一个在区间[low, high)中离散均匀抽样的数组，size指定形状，dtype指定数据类型。
- 参数：
  - low, high：int型，指定抽样区间[low, high)
  - size：int型或int型的元组，指定形状
  - dypte：可选参数，指定数据类型，比如int,int64等，默认是np.int
- 返回值：如果指定了size，则返回一个int型的ndarray对象，否则返回一个服从该分布的int型随机数。
  例子：

```
# 在[1, 10)之间离散均匀抽样，数组形状为3行2列
a = np.random.randint(1, 10, (3, 2))
print(a)
# 不提供size
b = np.random.randint(1, 10)
print(b)
# 指定dtype
c = np.random.randint(1, 10, dtype=np.int64)
print(c)
type(c)
输出：

[[3 1]
 [3 3]
 [5 8]]
6
2
numpy.int64
```

6、numpy.random.random(size=None)

- 作用：返回从[0, 1)之间均匀抽样的数组，size指定形状。
- 参数：
  - size：int型或int型的元组，如果不提供则返回一个服从该分布的随机数
- 返回值：float型或者float型的ndarray对象
- 例子：

```
# [0, 1)之间的均匀抽样，3行2列
a = np.random.random((3, 2))
print(a)
# 不指定size
b = np.random.random()
print(b)
输出：

[[0.80136714 0.63129059]
 [0.04556679 0.04433006]
 [0.09643599 0.53312761]]
0.32828505898057136
```