第7章 NumPy库

第7章 NumPy库

- 7.1 ndarray多维数组
- 7.2 数组元素的索引、切片和选择
- 7.3 随机数数组
- 7.4 数组的运算
- 7.5 读写数据文件

第7章 NumPy库

NumPy是用Python语言进行科学计算和数据分析的扩展库，是很多数据分析扩展库的基础库。NumPy主要功能有：可以构建具有矢量运算和复杂广播能力的快速且节省空间的多维数组，含有大量对数组数据进行快速运算的数学函数，以及线性代数、随机数生成等功能。

第7章 NumPy库

- ✓ 7.1 ndarray多维数组
- ☐ 7.2 数组元素的索引、切片和选择
- ☐ 7.3 随机数数组
- ☐ 7.4 数组的运算
- ☐ 7.5 读写数据文件

7.1 ndarray多维数组

NumPy库的基础是N维数组对象（即ndarray对象），该对象由两部分组成，一部分是实际的数据（同种类型），另一部分是描述这些数据的元数据。

7.1.1 创建ndarray数组

1. 创建ndarray数组最简单的方式是使用numpy的array()函数。

```
>>> import numpy as np        #本章中出现的np默认均是这个含义
>>> a=np.array([1,2,3])       #以列表作为参数创建一维数组
>>> a
array([1, 2, 3])
>>> b = np.array([[1,2],[3,4]])  #以列表作为参数创建2×2的二维数组
>>> b
array([[1, 2],
       [3, 4]])
```

7.1 ndarray多维数组

7.1.1 创建ndarray数组

2. 使用ndarray()函数创建ndarray数组，其常用参数如下表所示

参数	类型	作用
shape	int型tuple	指定数组的维数和每维的大小
dtype	int型、float型等	指定数组中元素的类型
buffer	ndarray	用于初始化数组的数组
offset	int	buffer中用于初始化数组的首个数据的偏移，是数组元素在内存所占字节数的整数倍
strides	int型tuple	每个轴的下标增加1时，数据指针在内存中增加的字节数
order	'C' 或者 ’F’	'C'表示行优先，buffer中的数据按行的顺序存入将要创建的数组中；'F'表示列优先

7.1 ndarray多维数组

7.1.1 创建ndarray数组

#buffer中的数据按行的顺序存入将要创建的数组e中

>>> e=np.ndarray(shape=(2,3), dtype=int, buffer= np.array([0, 1,2,3,4,5,6,7,9]), offset=0, order="C")

```
>>> e
array([[0,  1,  2],
       [3,  4,  5]])
```

#buffer中的数据按列的顺序存入将要创建的数组e中

```
>>>f=np. ndarray(shape=(2, 3), dtype=int, buffer=
np. array([0, 1, 2, 3, 4, 5, 6, 7, 9]), offset=0, order="F")
>>> f
array([[0,  2,  4],
       [1,  3,  5]])
```

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

（1）使用ones()函数创建一个元素全为1的数组

ones(shape, dtype=None)

参数说明：

shape：int或int类型序列，表示矩阵形状。

dtype：用来指定所要生成的数组元素的数据类型。

返回值：返回按给定要求生成的全1的数组。

```
>>> a = np.ones(shape = (3, 3))        #通过shape指定生成3行3列
```

的全为1的数组 默认数据类型float64

```
>>> a
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

（2）使用zeros()函数创建一个元素全为0的数组

（3）使用empty()函数创建一个随机数组

```
>>> c = np.zeros(shape =(2,3),dtype=int)
>>>array([[ 0,      0,      0],
        [0, 0,   0]])

>>> c = np.empty(shape =(2,3),dtype=int)
>>> c
array([[         308,            0,            0],
        [1713398048, 1952673397,  745434985]])
```

7.1.2 创建特殊的ndarray数组

（4）使用arange()函数创建数组

arange([start,]stop[,step,],dtype=None)

参数说明：

start：可选参数，起始值，默认值为0。

stop：结尾值，所生成的数组不包括该值。

step：可选参数，步长（数组中相邻两个元素的间隔值），默认值是1，如果step被指定，要求start也必须被指定。

dtype：用来指定所要生成的数组元素的数据类型。

返回值：返回在指定数值区间的均匀间隔的数值序列所组成的ndarray数组。

```
>>> np.arange(1,10)
array([1, 2, 3, 4, 5, 6, 7, 8, 9])   #不包括10
```

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

（5）使用linspace()函数创建数组

linspace(start,stop,num=50,endpoint=True,retstep=False,dtype=None)

返回值：返回指定数值区间内均匀间隔的数字序列所构成的一维数组。

参数说明：

start：序列的起始值。

stop：若endpoint不被设置为False，stop为所生成的序列的结尾值。

num：可选参数，指定所要生成的序列包含的样本数，默认值为50。

endpoint：可选参数，如果设置为True，stop为所生成序列的最后一个值，否则，stop不被作为最后一个值，endpoint默认值为True。

retstep：可选参数，如果设置为True，函数返回一个元组（所生成的数组，数组中相邻两个元素的间隔值）。

dtype：可选参数，指定数组元素的数据类型。

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

linspace(start,stop,num=50,endpoint=True,retstep=False,dtype=None)

返回值：返回指定数值区间内均匀间隔的数字序列（等差数列）所构成的一维数组。

参数说明：

start：序列的起始值。

stop：若endpoint不被设置为False，stop为所生成的序列的结尾值。

num：可选参数，指定所要生成的序列包含的样本数，默认值为50。

```
>>> np.linspace(1,10,10)
array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
>>> np.linspace(0,10,6,dtype=int)
array([ 0,  2,  4,  6,  8, 10])
```

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

（6）使用logspace()函数创建数组

logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None)

返回值：返回一个以base**start开始以base**stop结束的包含num个数的等比数列构成的一维数组。

start：base**start是序列的起始值。

stop：base**stop是序列的结尾值。

num：生成的数组包含的元素个数，默认值为50。

base：指定生成等比数列数组的元素的幂的底数，默认值为10.0。

```
>>> np.logspace(1, 3, num=3)
array([   10.,   100.,  1000.])
>>> np.logspace(1, 3, num=3,base=2)
array([ 2.,  4.,  8.])
```

7.1 ndarray多维数组

7.1.2 创建特殊的ndarray数组

（7）eye()函数

eye(N, M=None, k=0)

返回值：返回一个对角线全1，其

时，全1对角线为主对角线；k>0

时，全1对角线向左下方偏移。

参数说明：

N：行数；

M：可选参数，列数，默认值为N；

k：k=0时，全1对角线为主对角线；k>0时，全1对角线向上移动相应的位置；k<0时，全1对角线向下移动相应的位置。

```
>>> np.eye(3, k = 0)
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
>>> np.eye(3, k = 1)
array([[ 0.,  1.,  0.],
       [ 0.,  0.,  1.],
       [ 0.,  0.,  0.]])
```

7.1.2 创建特殊的ndarray数组

（8）使用identity()函数创建数组

identity(n, dtype=None)

返回值：返回n*n单位矩阵，主对角线上元素都为1，其他元素都为0。

参数说明：

n：单位矩阵的行数（也是列数）；

dtype：指定矩阵元素的数据类型。

```
>>> np.identity(3,dtype=int)
array([[1,  0,  0],
       [0,  1,  0],
       [0,  0,  1]])
```

7.1.2 创建特殊的ndarray数组

（9）使用full()函数创建数组

full(shape, fill_value, dtype=None, order='C')

返回值：返回由固定值填充的数组。

参数说明：

shape：int或int类型序列，表示矩阵形状；

fill_value：填充值；

dtype：可选参数，指定元素的数据类型；

order：可选参数，取值'C'或者'F'，表示数组在内存的存放次序是以行(C)为主还是以列(F)为主，默认值为'C'。

```
>>> np.full((2, 3), 10)
array([[10, 10, 10],
       [10, 10, 10]])
```

7.1.2 创建特殊的ndarray数组

（9）使用full()函数创建数组

full(shape, fill_value, dtype=None, order='C')

返回值：返回由固定值填充的数组。

参数说明：

shape：int或int类型序列，表示矩阵形状；

fill_value：填充值；

dtype：可选参数，指定元素的数据类型；

order：可选参数，取值'C'或者'F'，表示数组在内存的存放次序是以行(C)为主还是以列(F)为主，默认值为'C'。

```
>>> np.full((2, 3), 10)
array([[10, 10, 10],
       [10, 10, 10]])
```

7.1 ndarray多维数组

7.1.4 ndarray对象的属性

ndarray数组对象的属性	属性说明
T	ndarray.T返回数组的转置
dtype	描述数组元素的类型
shape	返回数组的型，以元组表示的数组的各维的大小
ndim	返回数组的维度
size	ndarray.size返回数组中元素的个数
itemsize	返回数组中的单个元素在内存所占字节数
flat	返回一个数组的迭代器，对flat赋值将导致整个数组的元素被覆盖
real/imag	给出复数数组的实部/虚部
nbytes	返回数组的所有元素占用的存储空间

7.1 ndarray多维数组

7.1.4 ndarray对象的属性

```
>>> a = np.array([[ 0, 1, 2, 3, 4],[ 5, 6, 7, 8, 9],[10, 11, 12, 13, 14]])
>>> a.T                #返回数组的转置
array([[ 0,  5, 10],
       [ 1,  6, 11],
       [ 2,  7, 12],
       [ 3,  8, 13],
       [ 4,  9, 14]])
>>> a.size             #返回数
15
>>> a.shape            #返回数组的型
(3, 5)
```

```
>>> a.ndim
2
```

```
>>> a.flat             #返回数组的迭代器
<numpy.flatiter object at
0x0000000003728C20>
>>> for x in a.flat:
        print(x,end=',')

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,
```

第7章 NumPy库

- 7.1 ndarray多维数组
- ✓ 7.2 数组元素的索引、切片和选择
- 7.3 随机数数组
- 7.4 数组的运算
- 7.5 读写数据文件

7.2 数组元素的索引、切片和选择

7.2.1 索引和切片

- 数组索引机制指的是用方括号[]加序号的形式抽取元素、选取数组的某些元素、以及为索引处的元素重新赋值。
- 数组的切片操作是用来抽取数组的一部分元素生成新数组。切片是把用冒号“:”隔开的数字置于方括号“[]”里。
- 需要注意的是：对Python列表进行切片操作得到的列表是原列表的拷贝，而NumPy对ndarray数组进行切片操作得到的数组是指向相同缓冲区的视图，对所得切片数组元素的改变就是对原数组元素的改变。

```
>>> x = np.arange(10)  #生成0到9
>>> x[5]    #获取第i个值
5
```

```
>>> x[2:6]     #切片
array([2, 3, 4, 5])
```

7.2 数组元素的索引、切片和选择

7.2.1 索引和切片

2. 使用列表索引数组

```
>>> x = np.arange(10,1,-1)
>>> x
array([10, 9, 8, 7, 6, 5, 4, 3, 2])
#用列表[2,2,1,6]取出x中的第2,2,1,6的四个元素组成一个数组
>>> x[[2, 2, 1, 6]]
 array([8, 8, 9, 4])
```

3. 布尔值索引数组

```
>>> y = np.arange(30)
>>> b = y>20
>>> y[b]
array([21, 22, 23, 24, 25, 26, 27, 28, 29])
```

（4）使用arange()函数创建数组

arange([start,]stop[,step,],dtype=None)

参数说明：

start：可选参数，起始值，默认值为0。

stop：结尾值，所生成的数组不包括该值。

7.2 数组元素的索引、切片和选择

7.2.2 选择数组元素的方法

(1) ndarray.take(indices[,axis=None,out=None, mode='raise'])

根据指定的索引indices从数组对象ndarray中获取对应元素，并构成一个新的数组返回。

axis：用来指定选择元

入数组，即把数组当

out：是ndarray对象，

与函数返回值的shape

mode：指定越界索引

```
>>> x
array([ 0,  2,  4,  6,  8, 10, 12,
14, 16, 18])
#获取0,2,4索引处的元素
>>> x.take([0,2,4]) array([0,  4,
8])
>>> x.take([[2,  5], [3,6]])  #返回
数组的形状与索引的形状相同
array([[ 4,  10],
```

7.2 数组元素的索引、切片和选择

7.2.2 选择数组元素的方法

（2）ndarray.put(indices, values[, mode])

将数组中索引indices指定的位置处的元素值设置为values中对应的元素值。

```
>>> x = np.arange(0,20,2)
>> x
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
>>>  x.put([0, 1], [1, 3]) #将x中索引[0,1]处的值设置为列表[1,3]中对应的值
>>> x
array([ 1,  3,  4,  6,  8, 10, 12, 14, 16, 18])
```

7.2 数组元素的索引、切片和选择

7.2.2 选择数组元素的方法

（3）ndarray.searchsorted(v, side='left', sorter=None)

将v插入到当前有序的数组中，返回插入的位置索引。

```
>>> w = np.array([1, 2, 3, 3, 3, 3, 6, 7, 9, 10, 12])
>>> w.searchsorted(3)
2
>>> w.searchsorted(3,side='right')          # side='right'
6
```

7.2 数组元素的索引、切片和选择

7.2.2 选择数组元素的方法

（8）ndarray.itemset(*args)：修改数组中某个元素的值。

```
>>> a.itemset(3, 33)      #将序号为3的元素修改为33
>>> a
array([[ 0,  1,  2],
       [33,  4,  5],
       [ 6,  7,  8]])
>>> a.itemset((1, 2), 12)  #将a数组中(1, 2)处的元素修改为12
>>> a     #第2行，第3列
array([[ 0,  1,  2],
       [33,  4, 12],
       [ 6,  7,  8]])
```

7.2 数组元素的索引、切片和选择

7.2.3 ndarray数组的形状变换

ndarray数组对象提供了一些方法用来改变数组的形状，常用的改变数组形状的方法如表所示。

改变数组形状的方法	功能
ndarray.reshape(shape,order)	返回一个具有相同数据域，但shape不一样的视图
ndarray.resize(new_shape,orefcheck)	原地修改数组的形状，需要保持元素个数前后相同
ndarray.transpose(*axes)	返回数组针对某一轴进行转置后的数组，对于二维ndarray，transpose在不指定参数时默认是矩阵转置
ndarray.swapaxes(axis1,asix2)	返回数组axis1轴与axis2轴互换的视图
ndarray.flatten(order)	返回将原数组展平后的一维数组的拷贝（全新的数组）
ndarray.ravel(order='C')	返回将原数组展平后的一维数组的视图，order='C'表示行序优先，order='F'表示列序优先

7.2 数组元素的索引、切片和选择

7.2.3 ndarray数组

```
import numpy as np
A = np.array([1, 2, 3, 4, 5,
print("A:\n", A)
A_resize = A.resize((2, 3))
print("A_resize:\n", A_re
print("A(after resize):\n"
print('-'*10)
B = np.array([1, 2, 3, 4, 5,
print("B:\n", B)
B_reshape = B.reshape((2,
print("B_reshape:\n", B_r
print("B(after reshape):\n
```

```
E:\MySoftware\Anaconda\envs\tensorflow-gpu\python.exe E:/MySoftwa
A:
 [1 2 3 4 5 6]
A_resize:
 None
A(after resize):
 [[1 2 3]
 [4 5 6]]
----------
B:
 [1 2 3 4 5 6]
B_reshape:
 [[1 2 3]
 [4 5 6]]
B(after reshape):
 [1 2 3 4 5 6]
```

resize:

返回值为None

原数组发生了改变

reshape:

返回值为reshape后的数组

原数组不变

区别一：

resize **无返回值**（返回值为None），**会改变**原数组。

reshape **有返回值**，返回值是被reshape后的数组，**不会改变**原数组。

7.2 数组元素的索引、切片和选择

7.2.3 ndarray

```
import numpy as np
A = np.array([1, 2, 3
print("A:\n", A)
A_resize = A.resize((
print("A_resize:\n",
print("A(after resize
A_resize = A.resize((
print("A_resize:\n",
print("A(after resize
print('-'*10)
B = np.array([1, 2, 3
print("B:\n", B)
```

```
A:
 [1 2 3 4 5 6]
A_resize:
 None
A(after resize):
 [[1 2 3 4]
 [5 6 0 0]
 [0 0 0 0]]
A_resize:
 None
A(after resize):
 [[1 2]
 [3 4]]
----------
```

resize:

放大：用0补全

缩小：舍弃多余元素

resize 可以放大或者缩小原数组的形状：**放**大时，会用0补全剩余元素；**缩**小时，直接丢弃多余元素。

reshape 要求reshape前后元素个数相同，否则**会报错**，无法运行。

7.2 数组元素的索引、切片和选择

7.2.3 ndarray数组的形状变换

a = np.arange(24).reshape((2,3,4))

```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
In [177]: a.shape
Out[177]: (2, 3, 4)

In [178]: a.swapaxes(0,1).shape
Out[178]: (3, 2, 4)

In [179]: a.swapaxes(0,2).shape
Out[179]: (4, 3, 2)
```

7.2 数组元素的索引、切片和选择

7.2.3 ndarray数组的形状变换

```
>>> a=array([[1,2],[3,4],[5,6]])
>>> a
array([[1, 2],
    [3, 4],
    [5, 6]])
>>> a.flatten() #默认按行的方向降维
array([1, 2, 3, 4, 5, 6])
>>> a.flatten('F') #按列降维
array([1, 3, 5, 2, 4, 6])
>>> a.flatten('A') #按行降维
array([1, 2, 3, 4, 5, 6])
```

第7章 NumPy库

- 7.1 ndarray多维数组
- 7.2 数组元素的索引、切片和选择
- ✓ 7.3 随机数数组
- 7.4 数组的运算
- 7.5 读写数据文件

7.3 随机数数组

7.3.1 简单随机数

numpy.random模块中常用的生成简单随机数的函数如表所示

rand(d0,d1,...,dn)	生成一个(d0,d1,...,dn)维的数组，数组的元素取自[0,1)上的均匀分布。
randn(d0, d1, …, dn)	生成一个(d0,d1,...,dn)维的数组，数组元素是标准正态分布随机数，若没有参数，返回单个数据
randint(low,high=None, size=None, dtype='I')	生成size个随机整数，取值区间为[low, high)，若没有输入参数high，则取值区间为[0,low)
random_sample(size=None)	生成一个[0,1)之间的随机数或指定维的随机数组
random(size=None)	同random_sample(size=None)
sample([size]))	同random_sample([size])
choice(a,size=None, replace=True, p=None)	从a（数组）中选取size个（维度）随机数，replace=True表示可重复抽取，p是a中每个数出现的概率，p的长度和a的长度必须相同，且p中元素之和为1，否则报错；若a是整数，则a代表的数组是np.range(a)

7.3 随机数数组

7.3.2 随机分布

numpy.random模块中常用的随机分布函数如所示

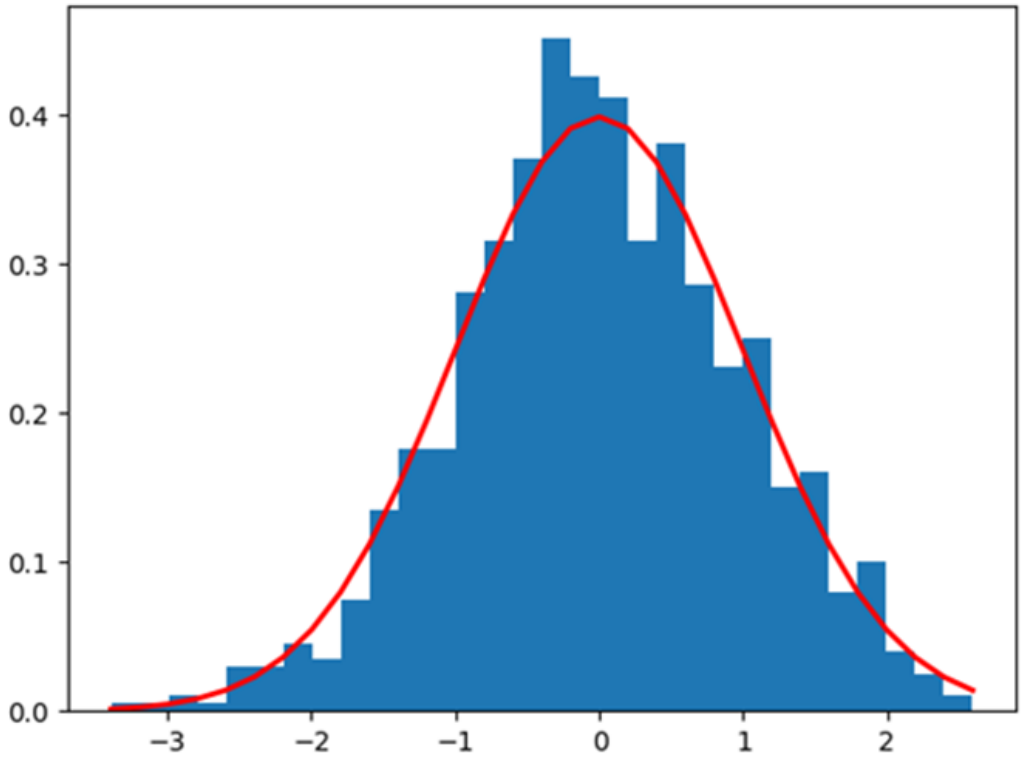
随机分布函数	函数功能
binomial(n, p, size=None)	产生size个二项分布的样本值，n表示n次试验，p表示每次实验发生（成功）的概率。函数的返回值表示n次实验中发生（成功）的次数
exponential(scale=1.0, size=None)	产生size个指数分布的样本值，这里的scale是β，为标准差，β=1/λ，λ为单位时间内事件发生的次数
normal(loc=0.0, scale=1.0, size=None)	产生size个正态（高斯）分布的样本值，loc为正态分布的均值，scale为正态分布的标准差，size缺省，则采样一个
poisson(lam=1.0, size=None)	从泊松分布中生成随机数，lam是单位时间内事件的平均发生次数
uniform(low=0.0, high=1.0, size=None)	产生size个均匀分布[low,high)的样本值，size为int或元组类型，如size = (m, n, k)，则输出m*n*k个样本，缺省时输出1个值

7.3 随机数数组

7.3.2 随机分布　demo

（2）normal(loc=0.0, scale=1.0, size=None)：产生size个正态（高斯）分布的样本值。

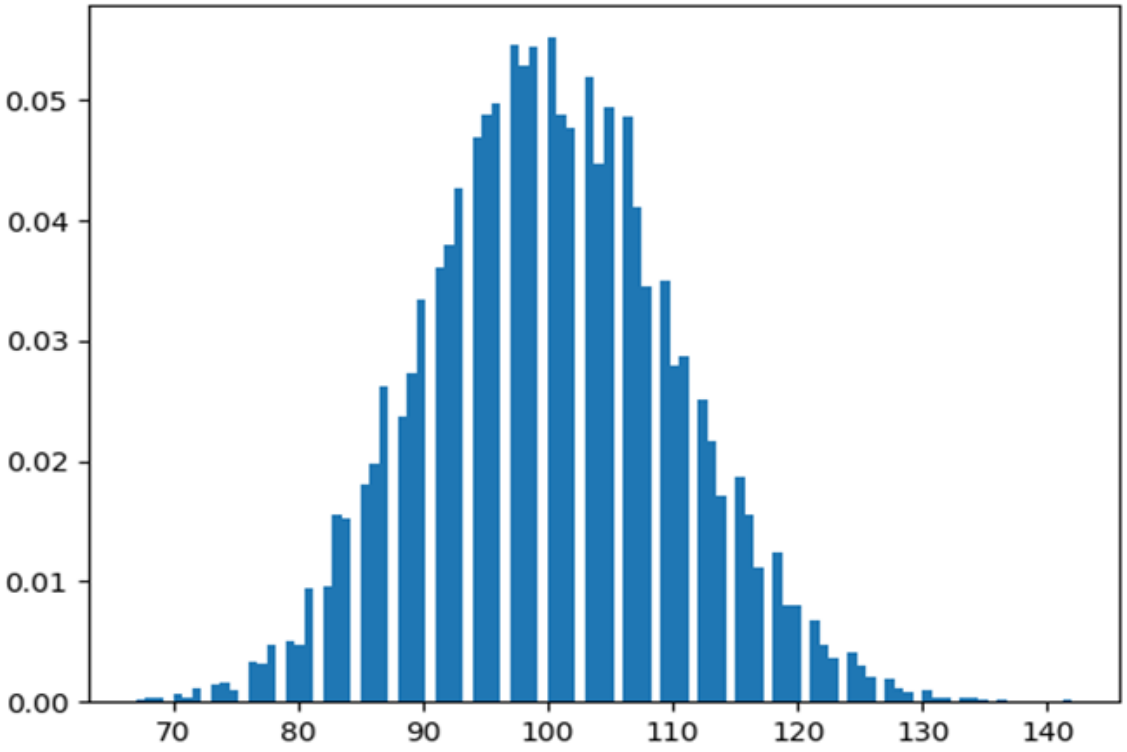
```
>>> from numpy import rand
>>> import numpy as np
>>> mu, sigma = 0, 1
>>> s = random.normal(loc=m
#绘制样本的直方图，以及概率
>>> import matplotlib.pyplot
>>> count, bins, patches = plt.
>>> plt.plot(bins, 1/(sigma * n
sigma**2)), linewidth=2, color
>>> plt.show()  #绘制的直方图以及概率密度函数曲线如图所示
```



7.3.2 随机分布

（3）poisson(lam=1.0, size=None)：从泊松分布中生成随机数，lam是单位时间内事件的平均发生次数。

```
>>> import nur
>>> import ma
>>> s=np.rand
>>> count,bins,
>>> plt.show()
```



直方图

7.3 随机数数组

7.3.3 随机排列

numpy.random模块中用于对数组对象随机排列的函数

函数名称	函数功能
shuffle(x)	打乱对象x（多维数组按照第一维打乱），直接在原来的数组上进行操作，改变原来数组元素的顺序，无返回值，x可以是数组或者列表
permutation(x)	打乱并返回新对象（多维数组按照第一维打乱），不直接在原数组上进行操作，而是返回一个新的打乱元素顺序的数组，并不改变原来的数组，x可以是整数或者列表，如果是整数k，那就随机打乱numpy.arange(k)

7.3 随机数数组

7.3.3 随机排列

numpy.random模块中用于对数组对象随机排列的函数

（1）shuffle(x)：打乱对象x（多维数组按照第一维打乱），直接在原来的数组上进行操作，改变原来数组元素的顺序，无返回值，x可以是数组或者列表。

```
>>> import numpy as np
>>> arr1=np.arange(9).reshape((3, 3))
>>> arr1
array([[0,  1,  2],
       [3,  4,  5],
       [6,  7,  8]])
>>> np.random.shuffle(arr1)
>>> arr1
array([[6,  7,  8],
       [0,  1,  2],
       [3,  4,  5]])
```

7.3 随机数数组

7.3.4 随机数生成器

随机数是由随机种子根据一定的计算方法计算出来的数值，所以，只要计算方法一定，随机种子一定，那么产生的随机数就不会变。

numpy.random.seed([seed])：用来确定随机数生成器的种子，如果在seed()中传入的数字相同，那么接下来使用random()、rand()、permutation()等方法所生成的随机数都是相同的。

7.3 随机数数组

7.3.4 随机数生成器 demo1

下面给出seed([seed])函数的应用举例。

```
import numpy as np
for i in range(3):
    np.random.seed(10)  #每次seed()函数的参数值都是10
    perm = np.random.permutation(10)
    print(perm)
```

上述代码在IDLE中运行的结果如下：

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

第7章 NumPy库

- ☐ 7.1 ndarray多维数组
- ☐ 7.2 数组元素的索引、切片和选择
- ☐ 7.3 随机数数组
- ✓ 7.4 数组的运算
- ☐ 7.5 读写数据文件

7.4 数组的运算

7.4.1 算术运算与函数运算

数组最常用的运算是算术运算，可以为数组的每个元素加上或乘以某个数值，可以让两个数组对应元素之间做加、减、乘、除运算等。

```
>>> import numpy as np
>>> a=np.array([[1,2,3],[4,5,6]])
>>> a
array([[1,  2,  3],
       [4,  5,  6]])
>>> a + 6   #数组a中的每个元素+6
array([[ 7,   8,   9],
       [10,  11,  12]])
```

7.4 数组的运算

7.4.1 算术运算与函数运算

通过在数组和数字之间使用条件运算符，比如大于号，将会得到由布尔值组成的数组，对于原数组中满足条件的元素，布尔数组中处于同等位置的元素为True。

```
>>> a
array([[0.51691317, 0.20933602, 0.22460165],
       [0.91598144, 0.43223077, 0.59339314]])
>>> a>0.5
array([[ True, False, False],
       [ True, False,  True]])
```

直接把a>0.5的条件表达式置于方括号中，能抽取所有大于0.5的元素，组成一个新数组。

```
>>> b=a[a>0.5]
>>> b
array([0.51691317, 0.91598144, 0.59339314])
```

7.4 数组的运算

7.4.1 算术运算与函数运算

numpy模块中对两个数组的元素进行算术运算的二元函数。

函数	说明
add(a,b)	将两个数组中对应的元素相加，a、b为两个ndarray数组
subtract(a,b)	将两个数组中对应的元素相减
multiply(a,b)	两个数组中对应的元素相乘
power(a,b)	对第一个数组中的元素x，第二个数组中的对应位置的元素y，计算x的y次方
greate、greate_equal、less、less_equal、equal、not_equal	将两个数组中对应的元素进行比较运算，最终产生布尔型的数组。相当于运算符>、>=、<、<=、==、!=

7.4 数组的运算

7.4.2 统计计算

ndarray数组对象的很多统计计算方法都有一个axis参数，它有如下作用：

1. 当axis=None（默认）时，数组被当成一个一维数组，对数组的计算操作是对整个数组进行的，比如sum方法，就是求数组中所有元素的和；

2. 当axis被指定为一个int整数时，对数组的计算操作是以提供的axis轴进行的，axis=0表示对column（列）进行操作，axis=1表示对row（行）进行操作。

7.4 数组的运算

7.4.2 统计计算

ndarray数组对象的常用统计计算方法	功能
ndarray.max(axis=None, out=None)	返回根据指定的axis计算最大值，axis=0表示求各column的最大值，axis=1表示求各row的最大值
ndarray.argmax(axis=None, out)	返回根据指定axis计算最大值的索引，out是ndarray对象，用来存放函数返回值，要求其shape必须与函数返回值的shape一致
ndarray.min(axis=None, out=None)	返回根据指定的axis计算最小值
ndarray.argmin(axis=None, out=None)	返回指定axis最小元素的索引
ndarray.ptp(axis, out)	返回根据指定axis计算最大值与最小值的差
ndarray.clip(min, max, out)	返回数组元素限制在[min, max]之间的新数组，小于min的转为min，大于max的转为max

7.4 数组的运算

7.4.2 统计计算

ndarray.trace(offset=0, axis1=0, axis2=1, dtype=None, out=None)	返回数组的迹（对角线元素的和）, offset表示离开主对角线的偏移量
ndarray.sum(axis=None, dtype=None, out=None)	返回指定axis的所有元素的和，默认求所有元素的和
ndarray.cumsum(axis=None, dtype=None, out=None)	按照所给定的轴参数返回元素的累计和
ndarray.mean(axis=None, dtype=None, out=None)	返回指定axis的数组元素均值
ndarray.var(axis=None, dtype=None, out=None, ddof=0)	根据指定的axis计算数组的方差
ndarray.std(axis=None, dtype=None, out=None, ddof=0)	根据指定axis计算数组的标准差
ndarray.prod(axis=None, dtype=None, out=None)	返回指定轴的所有元素乘积
ndarray.cumprod(axis=None, dtype=None, out=None)	返回指定轴的累积

7.4 数组的运算

7.4.2 统计计算

（1）ndarray.max(axis=None, out=None)：返回根据指定的axis计算最大值，axis=0表示求各column的最大值，axis=1表示求各row的最大值，out是ndarray对象，用来存放函数返回值，要求其shape必须与函数返回值的shape一致。

```
>>> import numpy as np
>>> a=np.array([[2, 3, 4, 9],[8, 7, 6, 5],[4, 3, 5, 8]])
>>> o=np.ndarray(shape=3)
>>> a.max(axis=1,out=o)      # axis=1表示求各row的最大值
array([9., 8., 8.])
>>> print(o)
[9. 8. 8.]
```

7.4 数组的运算

7.4.2 统计计算

（4）ndarray.clip(min，max，out)：返回数组元素限制在[min, max]之间的新数组（小于min的转为min，大于max的转为max）。

#返回数组元素限制在[5，　8]之间的新数组，小于5的转为5，大于8的转为8

```
>>> a=np.array([[2, 3, 4, 9],[8, 7, 6, 5],[4, 3, 5, 8]])
>>> a.clip(5, 8)
array([[5, 5, 5, 8],
       [8, 7, 6, 5],
       [5, 5, 5, 8]])
```

7.4 数组的运算

7.4.2 统计计算

（6）ndarray.cumsum(axis=None, dtype=None, out=None)：按照所给定的轴参数返回元素的累计和。

```
>>> a=np.array([[2, 3, 4, 9],[8, 7, 6, 5],[4, 3, 5, 8]])
>>> a.cumsum(axis=1)                    #按行求累计和
array([[ 2,  5,  9, 18],
       [ 8, 15, 21, 26],
       [ 4,  7, 12, 20]], dtype=int32)
```

7.4 数组的运算

7.4.3 线性代数运算

在NumPy中，用“*”进行两个数组相乘是两个数组对应位置上的元素相乘。NumPy用dot()函数执行一般意义上的矩阵（用ndarray数组表示）乘积，矩阵A、B乘积运算时要求矩阵A的列数必须等于矩阵B的行数，A、B乘积结果是一个矩阵，其第m行第n列的元素等于矩阵A的第m行的元素与矩阵B的第n列对应元素乘积之和。

```
>>> A=np.arange(1,10).reshape((3,3))
>>> A
array([[1,  2,  3],
       [4,  5,  6],
       [7,  8,  9]])
```

7.4 数组的运算

7.4.3 线性代数运算

```
>>> B=np.ones(shape=(3,3),dtype=int)
>>> B
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]])
>>> np.dot(A,B) #执行两个
array([[ 6,  6,  6],
       [15, 15, 15],
       [24, 24, 24]])
```

矩阵乘积的另外一种写法是把dot()函数当作其中一个矩阵对象的方法。

```
>>> A.dot(B)
array([[ 6,  6,  6],
       [15, 15, 15],
       [24, 24, 24]])
```

7.4 数组的运算

7.4.3 线性代数运算

numpy的linalg模块提供了一些进行线性代数运算的函数，如表所示

函数	描述
det	计算矩阵行列式
eig(A)	计算方阵A的特征值和特征向量
inv(A)	计算方阵A的逆
svd (A, full_matrices=1, compute_uv=1)	对矩阵进行奇异值分解，该函数返回3个矩阵——U、Sigma和V，其中U和V是正交矩阵，Sigma是输入矩阵的奇异值
solve(a, b)	求解形如AX= B的线性方程组，其中A是一个N*N的二维数组，而B是一个长度为N的一维数组，数组X是待求解的线性方程组的解

7.4.3 线性代数运算

（4）奇异值分解

奇异值分解(简称SVD)不仅可用于降维算法中的特征分解，还可以用于推荐系统，以及自然语言处理等领域。

假设矩阵A是一个m×n的矩阵，矩阵A的SVD为：

$$A=U\Sigma V^T$$

其中U是一个m×m的矩阵，Σ是一个m×n的矩阵，除了主对角线上的元素以外全为0，V是一个n×n的矩阵。U和V的列分别叫做A的左奇异向量和右奇异向量，Σ的□角□上的值叫做A的奇异值。

7.4 数组的运算

7.4.3 线性代数运算

（4）奇异值分解：$A=U\Sigma V^T$

```
>>> A
array([[0, 1],
       [1, 1],
       [1, 0]])
'''使用svd函数分解矩阵，返回U
构成的一维数组，可使用diag函数
>>> U,Sigma,V  = np.linalg.svd(A)
>>> U
array([[-0.40824829,  0.70710678,  0.57735027],
       [-0.81649658,  0.        , -0.57735027],
       [-0.40824829, -0.70710678,  0.57735027]])
```

```
>>> V
array([[-0.70710678, -0.70710678],
       [-0.70710678,  0.70710678]])
>>> Sigma
array([1.73205081, 1.        ])
#使用diag函数生成完整的奇异值矩阵，忽略全为0的行
>>>          np.diag(Sigma)
array([[1.73205081, 0.        ],
       [0.        , 1.        ]])
```

7.4 数组的运算

7.4.4 排序

ndarray数组对象的排序元素的常用方法如表所示。

排序元素的方法	方法功能
ndarray.sort(axis=-1, kind='quicksort', order=None)	原地对数组元素进行排序，即排序后改变了原数组，axis指定排序沿着数组的（轴）方向，0表示按行，1表示按列，axis默认值为-1，表示沿最后的轴排序；kind指定排序的算法，其取值集合为{‘quicksort’, ‘mergesort’, ‘heapsort’}，order指定元素的排列顺序，默认升序
ndarray.argsort(axis=-1, kind='quicksort', order=None)	返回对数组进行升序排序之后的数组元素在原数组中的索引

7.4 数组的运算

7.4.4 排序

（1）ndarray.sort(axis=-1, kind='quicksort', order=None)：原地对数组元素进行排序。

```
>>> y1=np.array([[0,15,10,5],[25, 22, 3, 2],[55, 45, 59, 50]])
>>> y1
array([[ 0, 15, 10,  5],
       [25, 22,  3,  2],
       [55, 45, 59, 50]])
>>> y1.sort()
>>> y1
array([[ 0,  5, 10, 15],
       [ 2,  3, 22, 25],
       [45, 50, 55, 59]])
```

7.4 数组的运算

7.4.5 数组拼接与切分

有时候需要用已有的数组创建新数组，有时候需要切分已有数组创建新数组。numpy提供了一些函数来实现数组的拼接与切分。

1）垂直拼接

vstack(tup)用来将列数相同的数组序列tup中的数组进行竖直方向的拼接，即把数组序列tup中后一个数组作为行追加到前一个数组的下边，数组向竖直方向上生长。参数tup的类型可以是元组，列表，或者numpy数组，返回结果为numpy数组。

```
>>> a = np.array([1, 2, 3])
>>> b = np.array([2, 3, 4])
>>> np.vstack((a,b))
array([[1, 2, 3],
       [2, 3, 4]])
```

7.4.5 数组拼接与切分

2）水平拼接

hstack(tup)用来将行数相同的数组序列tup中的数组进行水平方向的拼接，即把数组序列tup中后一个数组作为列追加到前一个数组的右边，数组向水平方向上生长。参数tup的类型可以是元组，列表，或者numpy数组，返回结果为numpy的数组。

```
>>> a = np.array([1, 2, 3]).reshape(3,1)
>>> b = np.array([4, 5, 6]).reshape(3,1)
>>> a
array([[1],
       [2],
       [3]])
>>> b
array([[4],
       [5],
       [6]])
>>> np.hstack((a,b))
array([[1, 4],
       [2, 5],
       [3, 6]])
```

7.4 数组的运算

7.4.5 数组拼接与切分

1）水平切分

numpy提供了函数hsplit()用来水平

按照宽度切分为两部分，例如把4

hsplit(ary, indices_or_sections)

ary：表示待切分的数组

indices_or_sections：可以是一个整

等分，否则会报错，也可是一个数

表示要将ary分成ary[:2]、ary[2:3

```
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

```
#返回三个子数组
>>> x,y,z=np.hsplit(a,[2,3])
>>> x
array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]])
>>> y
array([[ 2],
       [ 6],
       [10],
       [14]])
>>> z
array([[ 3],
       [ 7],
       [11],
       [15]])
```

7.4 数组的运算

7.4.5 数组拼接与切分

2）竖直切分

numpy提供了函数vsplit(ary, indices_or_sections)用来竖直方向上切分数组。

```
>>> a=np.arange(16).reshape(4,4)
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
>>> np.vsplit(a, 2)
[array([[0, 1, 2, 3],
       [4, 5, 6, 7]]), array([[ 8,  9, 10, 11],
       [12, 13, 14, 15]])]
```

第7章 NumPy库

- 7.1 ndarray多维数组
- 7.2 数组元素的索引、切片和选择
- 7.3 随机数数组
- 7.4 数组的运算
- ✓ 7.5 读写数据文件

7.5 读写数据文件

7.5.1 读写二进制文件

numpy中的save()函数以二进制格式保存数组到一个文件中，文件的扩展名为“.npy”，该扩展名是由系统自动添加的。numpy中的load()函数从二进制文件中读取数据。save()函数的语法格式如下：

numpy.save(file, arr)

参数说明：

file：用来保存数组的文件名或文件路径，是字符串类型。

arr：要保存的数组。

```
>>> import numpy as np
>>> A = np.arange(16).reshape(2,8)
>>> A
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15]])
>>> np.save("C:/workspace/Python/A", A)#默认存储 .npy数据
```

7.5 读写数据文件

7.5.2 读写文本文件

numpy中的savetxt()函数用于将数

savetxt()的语法格式如下：

numpy.savetxt(filename, arr, fmt=

filename：存放数据的文件名。

arr：要保存的数组。

fmt：指定数据存入的格式。

delimiter：数据列之间的分隔符，

newline：数据行之间的分隔符。

```
>>> a
array([[0,  1,  2,  3,  4],
       [5,  6,  7,  8,  9]])
#以空格分隔将数组a存放到文本文
>>> np.savetxt("C:/workspace/Python/a.txt",a)
```

numpy.loadtxt()函数用于从文本文件中读取数据到数组中，其语法格式如下。

numpy.loadtxt(fname, dtype=<class 'float'>, delimiter=None, converters=None)

参数说明：

fname：文件名/文件路径。

dtype：要读取的数据类型。

delimiter：读取数据时的数据列之间的分隔符，数据类型为字符串。

converters：读取数据时的数据行之间的分隔符。

注意：根据numpy.savetxt()定制的保存格式，相应的加载数据的函数numpy.loadtxt()也得相应变化。

7.5 读写数据文件

7.5.2 读写文本文件

```
>>> np.loadtxt("C:/workspace/Python/a.txt")
array([[ 0.,  1.,  2.,  3.,  4.],
       [ 5.,  6.,  7.,  8.,  9.]])
>>> b=np.arange(0,10,0.5).reshape(2,10)
#将数组元素保存为浮点数，以逗号分隔
>>> np.savetxt("C:/workspace/Python/b.txt",b,fmt="%f",delimiter=",")
#load时也要指定以逗号分隔，指定要读取的数据类型为浮点型
>>> np.loadtxt("C:/workspace/Python/b.txt",dtype="f",delimiter=",")
array([[0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5],
           [5. , 5.5, 6. , 6.5, 7. , 7.5, 8. , 8.5, 9. , 9.5]], dtype=float32)
```

作业

1.导入numpy库并简写为 np
2.创建一个值域范围从10到49的向量
3.反转一个向量(第一个元素变为最后一个)
4.创建一个 3x3 并且值从0到8的矩阵
5.找到数组[1,2,0,0,4,0]中非0元素的位置索引
6.创建一个 3x3 的单位矩阵
7.创建一个 3x3x3的随机数组
8.创建一个 10x10 的随机数组并找到它的最大值和最小值
9.创建一个长度为30的随机向量并找到它的平均值
10.创建一个长度为10的随机向量，并将其排序

THE END