NumPy 数值计算基础

2019/9/1

目录

创建数组对象

1．数组属性：ndarray（数组）是存储单一数据类型的多维数组。

属性	说明
ndim	返回 int。表示数组的维数
shape	返回 tuple。表示数组的尺寸，对于 n 行 m 列的矩阵，形状为(n,m)
size	返回 int。表示数组的元素总数，等于数组形状的乘积
dtype	返回 data-type。描述数组中元素的类型
itemsize	返回 int。表示数组的每个元素的大小（以字节为单位）。

创建数组对象

2．数组创建

*numpy.**array**(object, dtype=None, copy=True, order='K',subok=False, ndmin=0)*

参数名称	说明
object	接收array。表示想要创建的数组。无默认。
dtype	接收data-type。表示数组所需的数据类型。如果未给定，则选择保存对象所需的最小类型。默认为None。
ndmin	接收int。指定生成数组应该具有的最小维数。默认为None。

创建数组对象

- 创建数组并查看数组属性

In[1]:	import numpy as np #导入NumPy库arr1 = np.array([1, 2, 3, 4]) #创建一维数组
	print('创建的数组为：',arr1)

Out[1]:	创建的数组为： [1 2 3 4]

In[2]:	arr2 = np.array([[1, 2, 3, 4],[4, 5, 6, 7], [7, 8, 9, 10]]) #创建二维数组
	print('创建的数组为：\n',arr2)

Out[2]:	创建的数组为： [[1 2 3 4] [4 5 6 7] [7 8 9 10]]
In[3]:	print('数组维度为：',arr2.shape) #查看数组结构
Out[3]:	数组维度为： (3, 4)

In[4]:	print('数组维度为：',arr2.dtype) #查看数组类型
Out[4]:	数组维度为： int32
In[5]:	print('数组元素个数为：',arr2.size) #查看数组元素个数
Out[5]:	数组元素个数为： 12
In[6]:	print('数组每个元素大小为：',arr2.itemsize) #查看数组每个元素大小
Out[6]:	数组每个元素大小为： 4

创建数组对象

- 重新设置数组的 shape 属性

In[7]:	arr2.shape = 4,3 #重新设置 shape
	print('重新设置 shape 后的 arr2 为：',arr2)

Out[7]:	重新设置shape维度后的arr2为： [[1 2 3] [4 4 5] [6 7 7] [8 9 10]]

- 使用 arange 函数创建数组

In[8]:	print(' 使 用 arange 函 数 创 建 的 数 组 为 ：\n',np.arange(0,1,0.1))
Out[8]:	使用arange函数创建的数组为： [0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]

创建数组对象

- 使用 linspace 函数创建数组

```
In[9]:    print('使用 linspace 函数创建的数组为：',np.linspace(0, 1, 12))
```

```
Out[9]:   使用linspace函数创建的数组为： [ 0.         0.09090909  …  1.        ]
```

- 使用 logspace 函数创建等比数列

```
In[10]:   print('使用logspace函数创建的数组为：',np.logspace(0, 2, 20))
```

```
Out[10]:  使用logspace函数创建的数组为： [   1.          1.27427499
          1.62377674 ...,   61.58482111   78.47599704  100.        ]
```

创建数组对象

- 使用zeros函数创建数组

```
In[11]:    print('使用zeros函数创建的数组为：',np.zeros((2,3)))
```

```
Out[11]: 使用zeros函数创建的数组为：
[[ 0.  0.  0.]
 [ 0.  0.  0.]]
```

- 使用eye函数创建数组

```
In[12]:    print('使用eye函数创建的数组为：',np.eye(3))
```

```
Out[12]: 使用eye函数创建的数组为：
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
```

创建数组对象

- 使用diag函数创建数组

```
In[13]:  print('使用diag函数创建的数组为：',np.diag([1,2,3,4]))
```

```
Out[13]: 使用diag函数创建的数组为：
         [[1 0 0 0]
          [0 2 0 0]
          [0 0 3 0]
          [0 0 0 4]]
```

- 使用ones函数创建数组

```
In[14]:  print('使用ones函数创建的数组为：',np.ones((5,3)))
```

```
Out[14]: 使用ones函数创建的数组为：
         [[ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]
          [ 1.  1.  1.]]
```

创建数组对象

3．数组数据类型

NumPy基本数据类型与其取值范围（只展示一部分）

类型	描述
bool	用一位存储的布尔类型（值为TRUE或FALSE）
inti	由所在平台决定其精度的整数（一般为int32或int64）
int8	整数，范围为−128至127
int16	整数，范围为−32768至32767
int32	整数，范围为-2^{31}至$2^{32}-1$
……	……

创建数组对象

数组数据类型转换

In[15]:	print('转换结果为：',np.float64(42)) #整型转换为浮点型
Out[15]:	转换结果为： 42.0
In[16]:	print('转换结果为：',np.int8(42.0)) #浮点型转换为整型
Out[16]:	转换结果为： 42
In[17]:	print('转换结果为：',np.bool(42)) #整型转换为布尔型
Out[17]:	转换结果为： True

In[18]:	print('转换结果为：',np.bool(0)) #整型转换为布尔型
Out[18]:	转换结果为： False
In[19]:	print('转换结果为：',np.float(True)) #布尔型转换为浮点型
Out[19]:	转换结果为： 1.0
In[20]:	print('转换结果为：',np.float(False)) #布尔型转换为浮点型
Out[20]:	转换结果为： 0.0

创建数组对象

创建一个存储餐饮企业库存信息的数据类型。其中，用一个长度为40个字符的字符串来记录商品的名称，用一个64位的整数来记录商品的库存数量，最后用一个64位的单精度浮点数来记录商品的价格，具体步骤如下。

- 创建数据类型

In[21]:	df = np.dtype([("name", np.str_, 40), ("numitems", np.int64), ("price",np.float64)]) print('数据类型为：',df)
Out[21]:	数据类型为： [('name', '<U40'), ('numitems', '<i8'), ('price', '<f8')]

创建数组对象

- 查看数据类型，可以直接查看或者使用numpy.dtype函数查看。

In[22]:	print('数据类型为：',df["name"])
Out[22]:	数据类型为： <U40
In[23]:	print('数据类型为：',np.dtype(df["name"]))
Out[23]:	数据类型为： <U40

创建数组对象

在使用array函数创建数组时，数组的数据类型默认是浮点型。自定义数组数据，则可以预先指定数据类型

```
In[24]:  itemz = np.array([("tomatoes", 42, 4.14), ("cabbages", 13, 1.72)],
         dtype=df)
         print('自定义数据为：',itemz)
```

```
Out[24]: 自定义数据为： [('tomatoes', 42,  4.14) ('cabbages', 13,  1.72)]
```

生成随机数

- 无约束条件下生成随机数

```
In[25]:    print('生成的随机数组为：',np.random.random(100))
```

```
Out[25]:  生成的随机数组为： [ 0.15343184 0.51581585
0.07228451 ... 0.24418316
0.92510545 0.57507965]
```

- 生成服从均匀分布的随机数

```
In[26]:    print('生成的随机数组为：\n',np.random.rand(10,5))
```

```
Out[26]:  生成的随机数组为：
[[ 0.39830491 0.94011394 0.59974923 0.44453894
0.65451838]
...
[ 0.1468544 0.82972989 0.58011115 0.45157667
0.32422895]]
```

生成随机数

- 生成服从正态分布的随机数

In[27]:	print('生成的随机数组为：\n',np.random.randn(10,5))
Out[27]:	生成的随机数组为： [[-0.60571968 0.39034908 -1.63315513 0.02783885 -1.84139301] ..., [-0.27500487 1.41711262 0.6635967 0.35486644 -0.26700703]]

- 生成给定上下范围的随机数，如创建一个最小值不低于 2、最大值不高于 10 的 2 行 5 列数组

In[28]:	print('生成的随机数组为：',np.random.randint(2,10,size = [2,5]))
Out[28]:	生成的随机数组为： [[6 6 6 6 8] [9 6 6 8 4]]

生成随机数

random模块常用随机数生成函数

函数	说明
seed	确定随机数生成器的种子。
permutation	返回一个序列的随机排列或返回一个随机排列的范围。
shuffle	对一个序列进行随机排序。
binomial	产生二项分布的随机数。
normal	产生正态（高斯）分布的随机数。
beta	产生beta分布的随机数。
chisquare	产生卡方分布的随机数。
gamma	产生gamma分布的随机数。
uniform	产生在[0,1)中均匀分布的随机数。

通过索引访问数组

1．一维数组的索引

In[29]:	arr = np.arange(10) print('索引结果为：',arr[5]) #用整数作为下标可以获取数组中的某个元素
Out[29]:	索引结果为： 5
In[30]:	print('索引结果为：',arr[3:5]) #用范围作为下标获取数组的一个切片，包括arr[3]不包括arr[5]
Out[30]:	索引结果为： [3 4]
In[31]:	print('索引结果为：',arr[:5]) #省略开始下标，表示从arr[0]开始
Out[31]:	索引结果为： [0 1 2 3 4]
In[32]:	print('索引结果为：',arr[-1]) #下标可以使用负数，-1表示从数组后往前数的第一个元素
Out[32]:	索引结果为： 9

通过索引访问数组

1．一维数组的索引

续表

In[33]:	arr[2:4] = 100,101 print('索引结果为：',arr) #下标还可以用来修改元素的值
Out[33]:	索引结果为：[0 1 100 101 4 5 6 7 8 9]

In[34]:	#范围中的第三个参数表示步长，2表示隔一个元素取一个元素
	print('索引结果为：',arr[1:-1:2])

Out[34]:	索引结果为：[1 101 5 7]
In[35]:	print('索引结果为：',arr[5:1:-2]) #步长为负数时，开始下标必须大于结束下标
Out[35]:	索引结果为：[5 101]

通过索引访问数组

2．多维数组的索引

```
In[36]:  arr = np.array([[1, 2, 3, 4, 5],[4, 5, 6, 7, 8], [7, 8, 9, 10, 11]])
         print('创建的二维数组为：',arr)
```

```
Out[36]: 创建的二维数组为： [[ 1  2  3  4  5]
          [ 4  5  6  7  8]
          [ 7  8  9 10 11]]
```

```
In[37]:  print('索引结果为：',arr[0,3:5])  #索引第0行中第3和4列的元素
```

```
Out[37]: 索引结果为： [4 5]
```

通过索引访问数组

2．多维数组的索引

续表

In[38]:	#索引第2和3行中第3～5列的元素
	print('索引结果为：',arr[1:,2:])

Out[38]:	索引结果为： [[6 7 8] [9 10 11]]
In[39]:	print('索引结果为：',arr[:,2]) #索引第2列的元素
Out[39]:	索引结果为： [3 6 9]

通过索引访问数组

2．多维数组的索引（使用整数和布尔值索引访问数据）

In[40]:	#从两个序列的对应位置取出两个整数来组成下标：arr[0,1]，arr[1,2]，arr[2,3]
	print('索引结果为：',arr[[(0,1,2),(1,2,3)]])

Out[40]:	索引结果为： [2 6 10]
In[41]:	print('索引结果为：',arr[1:,(0,2,3)]) #索引第2、3行中第0、2、3列的元素
Out[41]:	索引结果为： [[4 6 7] [7 9 10]]

In[42]:	mask = np.array([1,0,1],dtype = np.bool)
	#mask是一个布尔数组，它索引第1、3行中第2列的元素
	print('索引结果为：',arr[mask,2])

Out[42]:	索引结果为： [3 9]

变换数组的形态

- 改变数组形状

In[43]:	arr = np.arange(12) #创建一维数组
	print('创建的一维数组为：',arr)

Out[43]:	创建的一维数组为： [0 1 2 3 4 5 6 7 8 9 10 11]
In[44]:	print('新的一维数组为：',arr.reshape(3,4)) #设置数组的形状
Out[44]:	新的一维数组为： [[0 1 2 3] [4 5 6 7] [8 9 10 11]]
In[45]:	print('数组维度为：',arr.reshape(3,4).ndim) #查看数组维度
Out[45]:	数组维度为： 2

变换数组的形态

- 使用ravel函数展平数组

In[46]:	arr = np.arange(12).reshape(3,4) print('创建的二维数组为：',arr)
Out[46]:	创建的二维数组为： [[0 1 2 3] [4 5 6 7] [8 9 10 11]]
In[47]:	print('数组展平后为：',arr.ravel())
Out[47]:	数组展平后为： [0 1 2 3 4 5 6 7 8 9 10 11]

变换数组的形态

- 使用flatten函数展平数组

In[48]:	print('数组展平为：',arr.flatten()) #横向展平
Out[48]:	数组展平为： [0 1 2 3 4 5 6 7 8 9 10 11]
In[49]:	print('数组展平为：',arr.flatten('F')) #纵向展平
Out[49]:	数组展平为： [0 4 8 1 5 9 2 6 10 3 7 11]

变换数组的形态

组合数组

- 使用hstack函数实现数组横向组合：np.hstack((arr1,arr2))
- 使用vstack函数实现数组纵向组合：np.vstack((arr1,arr2))
- 使用concatenate函数实现数组横向组合：np.concatenate((arr1,arr2),axis = 1))
- 使用concatenate函数实现数组纵向组合：np.concatenate((arr1,arr2),axis = 0))

变换数组的形态

切割数组

- 使用hsplit函数实现数组横向分割： np.hsplit(arr1, 2)
- 使用vsplit函数实现数组纵向分割： np.vsplit(arr, 2)
- 使用split函数实现数组横向分割： np.split(arr, 2, axis=1)
- 使用split函数实现数组纵向分割： np.split(arr, 2, axis=0)

目录

1 掌握 NumPy 数组对象 ndarray

2 掌握 NumPy 矩阵与通用函数

3 利用 NumPy 进行统计分析

创建NumPy矩阵

创建与组合矩阵

- 使用mat函数创建矩阵： matr1 = np.mat("1 2 3;4 5 6;7 8 9")
- 使用matrix函数创建矩阵：matr2 = np.matrix([[123],[456],[789]])
- 使用bmat函数合成矩阵：np.bmat("arr1 arr2; arr1 arr2")

创建NumPy矩阵

矩阵的运算

- 矩阵与数相乘：matr1*3
- 矩阵相加减：matr1±matr2
- 矩阵相乘：matr1*matr2
- 矩阵对应元素相乘：np.multiply(matr1,matr2)
- 矩阵特有属性：

属性	说明
T	返回自身的转置
H	返回自身的共轭转置
I	返回自身的逆矩阵
A	返回自身数据的2维数组的一个视图

认识ufunc函数

全称通用函数（universal function），是一种能够对数组中所有元素进行操作的函数。

- 四则运算：加（+）、减（-）、乘（*）、除（/）、幂（**）。数组间的四则运算表示对每个数组中的元素分别进行四则运算，所以形状必须相同。
- 比较运算：>、<、==、>=、<=、!=。比较运算返回的结果是一个布尔数组，每个元素为每个数组对应元素的比较结果。
- 逻辑运算：np.any函数表示逻辑“or”，np.all函数表示逻辑“and”。运算结果返回布尔值。

认识ufunc函数

ufunc函数的广播机制

广播（broadcasting）是指不同形状的数组之间执行算术运算的方式。需要遵循4个原则。

- 让所有输入数组都向其中shape最长的数组看齐，shape中不足的部分都通过在前面加1补齐。
- 输出数组的shape是输入数组shape的各个轴上的最大值。
- 如果输入数组的某个轴和输出数组的对应轴的长度相同或者其长度为1时，这个数组能够用来计算，否则出错。
- 当输入数组的某个轴的长度为1时，沿着此轴运算时都用此轴上的第一组值。

认识ufunc函数

ufunc函数的广播机制

- 一维数组的广播机制

$$\begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + [1\ 2\ 3] \rightarrow \begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + \begin{matrix}[1\ 2\ 3]\\ [1\ 2\ 3]\\ [1\ 2\ 3]\\ [1\ 2\ 3]\end{matrix} = \begin{matrix}[1\ 2\ 3]\\ [2\ 3\ 4]\\ [3\ 4\ 5]\\ [4\ 5\ 6]\end{matrix}$$

认识ufunc函数

ufunc函数的广播机制

- 二维数组的广播机制

$$\begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + \begin{matrix}[1]\\ [2]\\ [3]\\ [4]\end{matrix} \rightarrow \begin{matrix}[0\ 0\ 0]\\ [1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\end{matrix} + \begin{matrix}[1\ 1\ 1]\\ [2\ 2\ 2]\\ [3\ 3\ 3]\\ [4\ 4\ 4]\end{matrix} = \begin{matrix}[1\ 1\ 1]\\ [3\ 3\ 3]\\ [5\ 5\ 5]\\ [7\ 7\ 7]\end{matrix}$$

目录

1 掌握 NumPy 数组对象 ndarray

2 掌握 NumPy 矩阵与通用函数

3 利用 NumPy 进行统计分析

读写文件

NumPy文件读写主要有二进制的文件读写和文件列表形式的数据读写两种形式

- save函数是以二进制的格式保存数据。 np.save("../tmp/save_arr",arr)
- load函数是从二进制的文件中读取数据。 np.load("../tmp/save_arr.npy")
- savez函数可以将多个数组保存到一个文件中。 np.savez('../tmp/savez_arr',arr1,arr2)
- 存储时可以省略扩展名，但读取时不能省略扩展名。

读写文件

读取文本格式的数据

- savetxt函数是将数组写到某种分隔符隔开的文本文件中。

```
np.savetxt("../tmp/arr.txt", arr, fmt="%d", delimiter=",")
```

- loadtxt函数执行的是把文件加载到一个二维数组中。

```
np.loadtxt("../tmp/arr.txt",delimiter=",")
```

- genfromtxt函数面向的是结构化数组和缺失数据。

```
np.genfromtxt("../tmp/arr.txt", delimiter = ",")
```

使用数组进行简单统计分析

直接排序

- sort函数是最常用的排序方法。 arr.sort()
- sort函数也可以指定一个axis参数，使得sort函数可以沿着指定轴对数据集进行排序。axis=1为沿横轴排序；axis=0为沿纵轴排序。

使用数组进行简单统计分析

间接排序

- argsort函数返回值为重新排序值的下标。 arr.argsort()
- lexsort函数返回值是按照最后一个传入数据排序的。 np.lexsort((a,b,c))

使用数组进行简单统计分析

去重与重复数据

- 通过unique函数可以找出数组中的唯一值并返回已排序的结果。
- tile函数主要有两个参数，参数“A”指定重复的数组，参数“reps”指定重复的次数。

np.tile(A，reps)

- repeat函数主要有三个参数，参数“a”是需要重复的数组元素，参数“repeats”是重复次数，参数“axis”指定沿着哪个轴进行重复，axis = 0表示按行进行元素重复；axis = 1表示按列进行元素重复。

numpy.repeat(a, repeats, axis=None)

- 这两个函数的主要区别在于，tile函数是对数组进行重复操作，repeat函数是对数组中的每个元素进行重复操作。

常用的统计函数

当axis=0时，表示沿着纵轴计算。当axis=1时，表示沿着横轴计算。默认时计算一个总值。

函数	说明
sum	计算数组的和
mean	计算数组均值
std	计算数组标准差
var	计算数组方差
min	计算数组最小值
max	计算数组最大值
argmin	返回数组最小元素的索引
argmax	返回数组最小元素的索引
cumsum	计算所有元素的累计和
cumprod	计算所有元素的累计积

任务实现

读取iris数据集中的花萼长度数据（已保存为csv格式），并对其进行排序、去重，并求出和、累积和、均值、标准差、方差、最小值、最大值

Thank you!