

PYTHON Numpy

Numpy



NumPy
Base N-dimensional
array package



SciPy library
Fundamental
library for scientific
computing



Matplotlib
Comprehensive 2D
Plotting



IP[y]:
IPython
Enhanced
Interactive Console



Sympy
Symbolic
mathematics



pandas
Data structures &
analysis

Numpy

NumPy是在1995年诞生的Python库Numeric的基础上建立起来的。但真正促使NumPy的发行的是Python的SciPy库。

SciPy是2001年发行的一个类似于Matlab，Maple，Mathematica等数学计算软件的Python库，它实现里面的大多数功能。

但SciPy中并没有合适的类似于Numeric中的对于基础的数据对象处理的功能。于是，SciPy的开发者将SciPy中的一部分和Numeric的设计思想结合，在2005年发行了NumPy。

Numpy

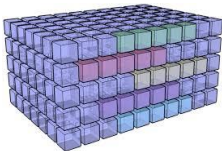
标准的Python中用list（列表）保存值，可以当做数组使用，但因为列表中的元素可以是任何对象，所以浪费了CPU运算时间和内存。

NumPy诞生为了弥补这些缺陷。它提供了两种基本的对象：
ndarray：全称（n-dimensional array object）是储存单一数据类型的多维数组。
ufunc：全称（universal function object）它是一种能够对数组进行处理的函数。

Numpy

NumPy是Python的一种开源的数值计算扩展库。它包含很多功能：

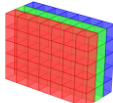
- 创建n维数组（矩阵）
- 对数组进行函数运算
- 数值积分
- 线性代数运算
- 傅里叶变换
- 随机数产生



Numpy

- NumPy中的核心对象是**ndarray**。
- ndarray可以看成数组，存放**同类元素**
- NumPy里面所有的函数都是围绕ndarray展开的

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$



colour image as N × M × 3-array

- ndarray 内部由以下内容组成：
- 一个指向数据（内存或内存映射文件中的一块数据）的指针。
 - 数据类型或 dtype，描述在数组中的固定大小值的格子。
 - 一个表示数组形状（shape）的元组，表示各维度大小的元组。

Numpy 数据类型

numpy 支持的数据类型比 Python 内置的类型要多很多，基本上可以和 C 语言的数据类型对应上。

- 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)
- 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

7

Numpy 数组的属性

属性	说明
<code>ndarray.ndim</code>	秩，即轴的数量或维度的数量
<code>ndarray.shape</code>	数组的维度，对于矩阵，n 行 m 列
<code>ndarray.size</code>	数组元素的总个数，相当于 <code>.shape</code> 中 <code>n*m</code> 的值
<code>ndarray.dtype</code>	ndarray 对象的元素类型
<code>ndarray.itemsize</code>	ndarray 对象中每个元素的大小，以字节为单位
<code>ndarray.flags</code>	ndarray 对象的内存信息
<code>ndarray.real</code>	ndarray 元素的实部
<code>ndarray.imag</code>	ndarray 元素的虚部
<code>ndarray.data</code>	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。

8

Numpy 创建

```
a = np.array([1, 2, 3, 4])      a = np.array([1, 2, 3, 4], dtype=np.int32)
b = np.array((5, 6, 7, 8))
c = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
```

ndarray 对维数没有限制。

[] 从内到外分别为第 0 轴，第 1 轴，第 2 轴。
c 第 0 轴长度为 2，第 1 轴长度为 4。

9

Numpy 创建

`numpy.arange(start, stop, step, dtype)`: `start` : `step` : **stop** # 尾部不一定能取到。

`np.linspace(start, stop, num=50, endpoint=True, dtype=None)`: 创建给定长度的等差数列

`numpy.empty((2,3), dtype = float)` : 未初始化的数组

`numpy.zeros((3,3), dtype = float)`: 数组元素以 0 来填充

`numpy.ones(shape, dtype = float)`: 数组元素以 1 来填充

`numpy.full((3,3), 0, dtype = float)`: 数组元素以 3 来填充

`numpy.eye(N, M=None, dtype = float)`: 矩阵对角线上元素为 1，其他为 0

`numpy.diag(v)`: 矩阵对角线的元素，其他为 0

10

Numpy 索引和切片

```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])      # list of list
a[0]: 指的是第一行
a[1, 2]: 全下标定位单个元素
```

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，设置 `start`, `stop` 及 `step` 参数

```
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
```

通过切片的对 ndarray 中的元素进行更改。

```
a[0, 2:4] = 100, 101
```

11

Numpy 索引和切片

切片的内存共享

切片产生一个新的数组 b，b 和 a 共享同一块数据存储空间。引用

```
a = np.arange(10)
b = a[3:7]
b[2] = 99
a = ?
```

如果想改变这种情况，我们可以用列表对数组元素切片。

```
b = a[ [3, 7] ]
```

12

Numpy 切片和索引

多维数组

NumPy的多维数组和一维数组类似。多维数组有多个轴。
我们前面已经提到从内到外分别是第0轴，第1轴...

```
array([[ 0,  1,  2,  3,  4,  5],
       [10, 11, 12, 13, 14, 15],
       [20, 21, 22, 23, 24, 25],
       [30, 31, 32, 33, 34, 35],
       [40, 41, 42, 43, 44, 45],
       [50, 51, 52, 53, 54, 55]])
```

a[0, 3:5]	a[4:, 4:]	a[2::2, ::2]
-----	-----	-----
[3, 4]	[[44, 45],	[[20, 22, 24],
	[54, 55]]	[40, 42, 44]]

共享原数组的存储空间。

13

Numpy 切片和索引

整数数组索引

如何获取数组中(0,0)，(1,1)和(2,0)位置处的元素？

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

```
x = np.array([[ 0,  1,  2],[ 3,  4,  5],[ 6,  7,  8],[ 9, 10, 11]])
rows = np.array( [ [0,0],[3,3] ] )
cols = np.array( [ [0,2],[0,2] ] )
y = x[rows,cols]
```

14

Numpy 切片和索引

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

```
x = np.array([[ 0,  1,  2],[ 3,  4,  5],[ 6,  7,  8],[ 9, 10, 11]])
print ('大于 5 的元素是: ')
print (x[x > 5])
```

```
array([[False, False, False],
       [False, False, False],
       [ True,  True,  True],
       [ True,  True,  True]])
```

```
array([ 6,  7,  8,  9, 10, 11])
```

15

Numpy 切片和索引

元素查找定位

numpy.where (condition[, x, y])

1. np.where(condition, x, y): 满足条件(condition)，输出x，不满足输出y。
2. np.where(condition): 输出满足条件（即非0）元素的坐标

```
aa = np.arange(10)
np.where(aa,1,-1)
```

```
np.where([True,False], [True,True],
        [[1,2],[3,4]],
        [[9,8],[7,6]])
```

```
a = np.array([2,4,6,8,10,3]).reshape(2,3)
c = np.where(a > 5) # return the indexes
a[c] # get the elements
c[0]: array([0, 1, 1], dtype=int32)
c[1]: array([2, 0, 1], dtype=int32)
```

16

Numpy 切片和索引

元素删除

```
a = np.array([1,2,3])
a = np.array([[1,2,3], [4,5,6]])
del(a[0]) ?
```

np.delete(arr, obj, axis=None)

- 第一个参数：要处理的矩阵，
- 第二个参数，处理的位置，下标
- 第三个参数，0表示按照行删除，1表示按照列删除
- 返回值为，删除后的剩余元素构成的矩阵

```
a = np.arange(10)
b = np.delete(a, [0,1,2,3])
```

```
arr = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
np.delete(arr, 1, 0) # np.delete(arr, [0]) ?
```

```
array([[ 1,  2,  3,  4],
       [ 9, 10, 11, 12]])
```

17

Numpy ufunc 函数

ufunc是universal function的简称，种能对数组每个元素进行运算的函数。
NumPy的许多ufunc函数都是用C语言实现的，因此它们的运算速度非常快。

```
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)
```

NumPy提供了许多ufunc函数，它们和相应的运算符运算结果相同

- np.add(a, b) # 加法
- np.subtract(a, b) # 减法
- np.multiply(a, b) # 按元素乘法
- np.divide(a, b) # 按元素除法
- np.power(a, b) # 乘方

18

比较运算和布尔运算

两个数组比较，返回一个布尔数组，每一个元素都是对应元素的比较结果。

```
np.array([1, 2, 3]) < np.array([3, 2, 1])
array([ True, False, False], dtype=bool)
```

布尔运算在NumPy中也有对应的ufunc函数。

表达式	ufunc函数
$y=x1==x2$	<code>equal(x1,x2[,y])</code>
$y=x1!=x2$	<code>not_equal(x1,x2[,y])</code>
$y=x1<x2$	<code>less(x1,x2[,y])</code>
$y=x1<=x2$	<code>not_equal(x1,x2[,y])</code>
$y=x1>x2$	<code>greater(x1,x2[,y])</code>
$y=x1>=x2$	<code>gerater_equal(x1,x2[,y])</code>

19

练习

How to create the matrix?

```
a = np.arange(1,13).reshape(3,4)
```

How to get the second row?

```
a[1]
```

How to get block of `[[7 8], [11 12]]`?

```
a[1:3, 2:4]
```

How to delete the second column?

```
np.delete(a, 1, axis=1)
```

How to swap the first two rows?

```
a[[0, 1]] = a[[1, 0]]
```

Use `np.where` to find 3的倍数?

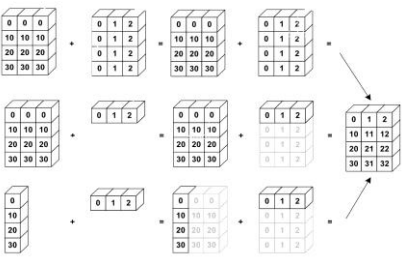
```
a[np.where(a % 3 == 0)]
```

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

20

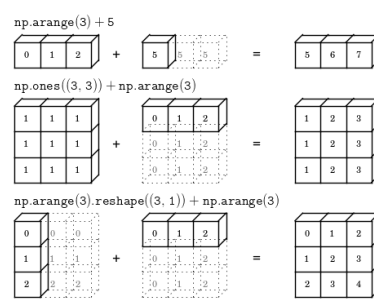
Numpy 广播

如果数组的形状不相同，就会进行广播处理。简而言之，就是向两个数组每一维度上的最大值看齐。



21

Numpy 广播



22

Numpy 数组拼接

多维数组可以进行连接，分段等多种操作。

- `e = np.concatenate((a,b), axis=0)` # 行增加了;
- `f = np.concatenate((a,b), axis=1)` # 列增加了;

`numpy.hstack((a, b))` #行连接: 等价于 `np.concatenate((a,b),axis = 1)`

`d = np.vstack((a,b))`: 列连接: 等价于 `np.concatenate((a,b),axis = 0)`

23

Numpy 数组拼接

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

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

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

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

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

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

```
a = np.array([[1], [2], [3]])
b = np.array([[4], [5], [6]])
c = np.vstack((a,b))

c.shape: (6, 1)
```

24

Numpy 数组拼接

numpy.tile(A, reps): Construct an array by repeating A the number of times given by reps.

```
a = np.array([0, 1, 2])
np.tile(a, 2)

array([0, 1, 2, 0, 1, 2])

np.tile(a, (2, 2))

array([[0, 1, 2, 0, 1, 2],
       [0, 1, 2, 0, 1, 2]])
```

```
b = np.array([[1, 2], [3, 4]])
np.tile(b, (2, 1))

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

25

Numpy 数组分割

1. 水平分割: np.split(arr,n,axis=1) 或 np.hsplit(arr,n): Split an array into multiple sub-arrays column-wise. 按列分成几份。返回一个list
2. 垂直分割: np.split(arr,n,axis=0) 或 np.vsplit(arr,n): split array into multiple sub-arrays row-wise.按行分成几份, 返回一个list

numpy.split(ary, indices_or_sections, axis=0)

If indices_or_sections is an integer, N, the array will be divided into N equal arrays along axis. If such a split is not possible, an error is raised.

If indices_or_sections is a 1-D array of sorted integers, the entries indicate where along axis the array is split.

For example, [2, 3] would, for axis=0, result in ary[:2], ary[2:3], ary[3:]

26

Numpy 数组分割

```
x = np.arange(9.0)
np.split(x, 3)

[array([0., 1., 2.]), array([3., 4., 5.]), array([6., 7., 8.])]
```

```
x = np.arange(8.0)
np.split(x, [3, 5, 6, 10])

[array([0., 1., 2.]),
 array([3., 4.]),
 array([5.]),
 array([6., 7.]),
 array([], dtype=float64)]
```

27

Numpy 维度变换

- x.reshape(shape): 不改变原数组元素, 返回一个新的shape维度的数组(维度变换)
- np.swapaxes(x, ax1,ax2) 将两个维度调换 OR x.swapaxes(ax1, ax2)
- np.transpose(x): 矩阵的转置 OR x.transpose()
- np.flipud(x): 上下翻转
- np.fliplr(x): 左右翻转
- x.flatten(): 数组降维成一维数组, 原数组不变。[np.ravel()]
- tolist(): 将N维数组转换成列表(维度变换)

28

Numpy 维度变换

ndarray.reshape(shape): Returns an array containing the same data with a new shape.

```
a = np.arange(20).reshape([4,5])

array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

29

Numpy 维度变换

ndarray.swapaxes(axis1, axis2): Return a view of the array with axis1 and axis2 interchanged. 将两个维度调换, 就是把对应的下标换个位置

```
a = np.arange(20).reshape(4,5)
b = a.swapaxes(1,0)

array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])

array([[ 0,  5, 10, 15],
       [ 1,  6, 11, 16],
       [ 2,  7, 12, 17],
       [ 3,  8, 13, 18],
       [ 4,  9, 14, 19]])
```

30

Numpy 维度变换

ndarray.tolist()

Return a copy of the array data as a (nested) Python list. Data items are converted to the nearest compatible builtin Python type, via the item function.

```
a = np.array([[1, 2], [3, 4]])
a.tolist()
```

```
[[1, 2], [3, 4]]
```

31

Numpy IO

numpy.save(file, array): 保存一个数组到一个二进制的文件中,保存格式是.npy

arr = numpy.load(file): 读取numpy 文件到内存

numpy.savetxt(fname, X, fmt='%.18e', delimiter=' '): 保存到文本文件

np.savetxt('test1.txt', x,fmt='%1.4e')

arr = numpy.loadtxt(fname, delimiter=None)

np.loadtxt('test2.out', delimiter=',')

32

Numpy 数值计算

1. 求和, 平均值, 方差
2. 大小与排序
3. 统计函数
4. 随机数
5. 多项式函数

33

Numpy数值计算

统计运算

函数名	功能
sum	求和
average	加权平均数
var	方差
mean	期望
std	标准差
product	连乘积

34

Numpy数值计算

a	np.sum(a, axis=1)	np.sum(a, axis=0)
-----	-----	-----
[[6, 3, 7, 4, 6], [9, 2, 6, 7, 4], [3, 7, 7, 2, 5], [4, 1, 7, 5, 1]]	[26, 28, 24, 18]	[22, 13, 27, 18, 16]

其他类似

35

Numpy数值计算

最大最小值

min, max, median,

argmin, argmax: 最小大值的下标

np.min(a, axis=1)

np.max(a, axis=0)

np.median(a, axis=1)

array([[10, 11, 12],
[13, 14, 15]])

np.argmax(a, axis=0)
array([0, 0, 0])

np.argmin(a, axis=1)
array([0, 0])

36

Numpy数值计算

排序

sort, argsort: 排序, 排序后的下标

```
a = [3,2,1,4]
b = np.sort(a)
b2 = np.argsort(a)

array([2, 1, 0, 3], dtype=int32)
```

How to get the n largest values of an array?

```
Z = np.arange(10000)
np.random.shuffle(Z)
Z[np.argsort(Z)[-n:]]
```

37

Numpy数值计算

np.unique(a): 得到不重复的元素

np.histogram: 统计

38

Numpy数值计算

rand	0到1之间的随机数	normal	正太分布的随机数
randint	制定范围内的随机整数	uniform	均匀分布
randn	标准正太的随机数	poisson	泊松分布
choice	随机抽取样本	shuffle	随机打乱顺序

```
np.random.rand(3,2)

array([[ 0.14022471,  0.96360618],
       [ 0.37601032,  0.25528411],
       [ 0.49313049,  0.94909878]])
```

```
arr = np.arange(9).reshape((3, 3))
np.random.shuffle(arr)
```

```
np.random.randint(2, size=10)

array([1, 0, 0, 0, 1, 1, 0, 0, 1, 0])

np.random.randint(5, size=(2, 4))

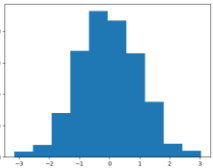
array([[4, 0, 2, 1],
       [3, 2, 2, 0]])
```

39

Numpy数值计算

```
x = np.random.randn(1000,1)

plt.hist(x)
```



40

Numpy数值计算

两个一维向量的内积

```
a = np.array([1,2,3])
b = np.dot(a,a)
b2 = np.vdot(a,a)
```

两个矩阵的内积

```
a = np.array([[1,2],[3,4]])
b = np.array([[1,12],[13,14]])

# vdot 将数组展开计算内积
print (np.vdot(a,b))
```

41

Numpy数值计算

两个矩阵相乘

```
a = np.array([[1,0],[0,1]])
b = np.array([[4,1],[2,2]])
print(np.matmul(a,b))
```

矩阵的行列式

```
a = np.array([[1,2],[3,4]])
print (np.linalg.det(a))
```

矩阵求逆

```
x = np.array([[1,2],[3,4]])
y = np.linalg.inv(x)
```

42

Numpy数值计算

求解方程组

$$\begin{cases} x + y + z = 6 \\ 2y + 5z = -4 \\ 2x + 5y - z = 27 \end{cases} \quad \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 5 \\ 2 & 5 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 6 \\ -4 \\ 27 \end{bmatrix}$$

```
a = np.array([[1,1,1],[0,2,5],[2,5,-1]])
b = np.array([[6],[-4],[27]])
x = np.matmul(np.linalg.inv(a), b)
x2 = np.linalg.solve(a,b)
```

43

Numpy数值计算

矩阵的特征值

```
A = np.random.randint(-10,10,(4,4))
C = np.dot(A.T, A)
vals, vecs = np.linalg.eig(C)
```

反向验证

```
C = vecs * np.diag(vals) * vecs.T

C2 = np.matmul(vecs, np.diag(vals))
C3 = np.matmul(C2, vecs.T)
```

44

Numpy数值计算

np.linalg.matrix_rank(A): 矩阵的秩
np.trace(A): 矩阵的迹

45

Numpy数值计算

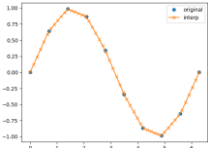
插值运算

numpy.interp(x, xp, fp, left=None, right=None, period=None)

- x - 表示将要计算的插值点x坐标
- xp - 表示已有的xp数组
- fp - 表示对应于已有的xp数组的值

```
x = np.linspace(0, 2*np.pi, 10)
y = np.sin(x)

xvals = np.linspace(0, 2*np.pi, 50)
yinterp = np.interp(xvals, x, y)
```



46

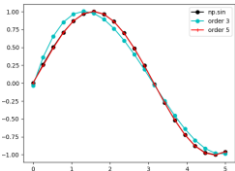
Numpy数值计算

曲线拟合

numpy.polyfit(x, y, deg): Least squares polynomial fit.

$$E = \sum_{j=0}^k |p(x_j) - y_j|^2$$

```
x = np.array([0.0, 1.0, 2.0, 3.0, 4.0, 5.0])
y = np.array([0.0, 0.8, 0.9, 0.1, -0.8, -1.0])
z = np.polyfit(x, y, 3)
z2 = np.polyfit(x, y, 5)
```



47