浅谈Python数据分析-Numpy

IDE: Pycharm Professional

OS: Deepin Linux

Python Version: 3.7

写在前面

Numpy是一个Python中非常重要的第三方库，而且还是 SciPy、Pandas 等数据科学的基础库。它所提供的数据结构比 Python 自身的“更高级、更高效”，可以这么说，NumPy 所提供的数据结构是 Python 数据分析的基础。

为什么要用 NumPy 数组结构而不是 Python 本身的列表 list？这是因为列表 list 的元素在系统内存中是分散存储的，而 NumPy 数组存储在一个均匀连续的内存块中。这样数组计算遍历所有的元素，不像列表 list 还需要对内存地址进行查找，从而节省了计算资源。

另外在内存访问模式中，缓存会直接把字节块从 RAM 加载到 CPU 寄存器中。因为数据连续的存储在内存中，NumPy 直接利用现代 CPU 的矢量化指令计算，加载寄存器中的多个连续浮点数。另外 NumPy 中的矩阵计算可以采用多线程的方式，充分利用多核 CPU 计算资源，大大提升了计算效率。

在 NumPy 里有两个重要的对象：**ndarray**（N-dimensional array object）解决了多维数组问题，而 **ufunc**（universal function object）则是解决对数组进行处理的函数。

ndarray 对象

ndarray 实际上是多维数组的含义。在 NumPy 数组中，维数称为秩（rank），一维数组的秩为 1，二维数组的秩为 2，以此类推。在 NumPy 中，每一个线性的数组称为一个轴（axes），其实秩就是描述轴的数量。

如下是创建数组的操作

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b[1,1]=10

print(a.shape)
print(b.shape)
print(a.dtype)
print(b)
```

运行结果

```
hwx@Neuromancer:~$ python3 demo.py
(3,)
(3, 3)
int64
[[ 1  2  3]
 [ 4 10  6]
 [ 7  8  9]]
```

创建数组前，要引用 NumPy 库，可以直接通过 **array** 函数创建数组，如果是多重数组，比如示例里的 b，可以先把一个数组作为一个元素，然后嵌套起来，比如示例 b 中的[1,2,3]就是一个元素，然后[4,5,6][7,8,9]也是作为元素，然后把三个元素再放到[]数组里，赋值给变量 b。

数组也是有属性的，可以通过函数 **shape** 属性获得数组的大小，通过 **dtype** 获得元素的属性。如果想对数组里的数值进行修改的话，直接赋值即可，注意下标是从 0 开始计的，所以如果想对 b 数组，九宫格里的中间元素进行修改的话，下标应该是[1,1]。

结构数组

如果想统计一个班级里面学生的姓名、年龄，以及语文、英语、数学成绩，可以用数组的下标来代表不同的字段，比如下标为 0 的是姓名、下标为 1 的是年龄等，但是这样不显性。

实际上在 C 语言里，可以定义结构数组，也就是通过 struct 定义结构类型，结构中的字段占据连续的内存空间，每个结构体占用的内存大小都相同，在 NumPy 中是怎样操作的呢？

```
import numpy as np

persontype = np.dtype({
    'names':['name', 'age', 'chinese', 'math', 'english'],
    'formats':['S32','i', 'i', 'i', 'f']})
peoples = np.array([("ZhangFei",32,75,100, 90),("GuanYu",24,85,96,88.5),
       ("ZhaoYun",28,85,92,96.5),("HuangZhong",29,65,85,100)],
    dtype=persontype)
ages = peoples[:]['age']
chineses = peoples[:]['chinese']
maths = peoples[:]['math']
englishs = peoples[:]['english']
print(np.mean(ages))
print(np.mean(chineses))
print(np.mean(maths))
print(np.mean(englishs))
```

```
hwx@Neuromancer:~$ python3 demo.py
28.25
77.5
93.25
93.75
```

首先在 NumPy 中是用 **dtype** 定义的结构类型，然后在定义数组的时候，用 array 中指定了结构数组的类型 **dtype=persontype**，这样你就可以自由地使用自定义的 persontype 了。比如想知道每个人的语文成绩，就可以用 chineses = peoples[:]['chinese']， NumPy 中还有一些自带的数学运算，比如计算平均值使用 **np.mean**。

ufunc 是 universal function 的缩写，听起来功能就很强大。它能对数组中每个元素进行函数操作。NumPy 中很多 ufunc 函数计算速度非常快，因为都是采用 C 语言实现的。

连续数组的创建

NumPy 可以很方便地创建连续数组，比如使用 arange 或 linspace 函数进行创建：

```
x1 = np.arange(1,11,2)
print(x1)
x2 = np.linspace(1,9,5)
print(x2)
```

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
[1 3 5 7 9]
[1. 3. 5. 7. 9.]
```

np.arange 和 **np.linspace** 起到的作用是一样的，都是创建等差数组。这两个数组的结果 x1,x2 都是[1 3 5 7 9]。结果相同，但是能看出来创建的方式是不同的。

arange 类似内置函数 range()，通过指定初始值、终值、步长来创建等差数列的一维数组，默认是不包括终值的。

linspace 是 linear space 的缩写，代表线性等分向量的含义。linspace() 通过指定初始值、终值、元素个数来创建等差数列的一维数组，默认是包括终值的。

算术运算

通过 NumPy 可以自由地创建等差数组，同时也可以进行加、减、乘、除、求 n 次方和取余数。

```python
import numpy as np

x1 = np.arange(1,11,2)
x2 = np.linspace(1,9,5)
print(np.add(x1, x2)) # 将两个数列相加
print(np.subtract(x1, x2)) # 将两个数列相减
print(np.multiply(x1, x2)) # 将两个数列相乘
print(np.divide(x1, x2)) # 将两个数列相除
print(np.power(x1, x2)) # 以x1中数字为基数，x2中数字为次数，做乘方运算
print(np.remainder(x1, x2)) # 取余
```

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
[ 2.  6. 10. 14. 18.]
[0. 0. 0. 0. 0.]
[ 1.  9. 25. 49. 81.]
[1. 1. 1. 1. 1.]
[1.00000000e+00 2.70000000e+01 3.12500000e+03 8.23543000e+05
 3.87420489e+08]
[0. 0. 0. 0. 0.]
```

统计函数

如果你想要对一堆数据有更清晰的认识，就需要对这些数据进行描述性的统计分析，比如了解这些数据中的最大值、最小值、平均值，是否符合正态分布，方差、标准差多少等等。

计数组 / 矩阵中的最大值函数 amax()，最小值函数 amin()

```python
import numpy as np

a = np.array([[1,2,3], [4,5,6], [7,8,9]]) # 创建二维数组
print(np.amin(a))    # 求整个二维数组中最小值
print(np.amin(a, 0)) # 求延着 axis=0 轴的最小值
print(np.amin(a, 1)) # 求延着 axis=1 轴的最小值
print(np.amax(a))    # 求整个二维数组中最大值
print(np.amax(a, 0)) # 求延着 axis=0 轴的最大值
print(np.amax(a, 1)) # 求延着 axis=1 轴的最大值
```

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
1
[1 2 3]
[1 4 7]
9
[7 8 9]
[3 6 9]
```

axis标识了横纵向，当axis=1为横向，axis=0为纵向。

使用axis其实将二维数组按行或列分组了

amin() 用于计算数组中的元素沿指定轴的最小值。对于一个二维数组 a，amin(a) 指的是数组中全部元素的最小值，amin(a,0) 是延着 axis=0 轴的最小值，即按列看整个二维数组，把元素看成了[1,4,7], [2,5,8], [3,6,9]三个元素，所以最小值为[1,2,3]。

amin(a,1) 是延着 axis=1 轴的最小值，axis=1 轴是把元素看成了[1,2,3], [4,5,6], [7,8,9]三个元素，所以最小值为[1,4,7]。同理 amax() 是计算数组中元素沿指定轴的最大值。

统计最大值与最小值之差 ptp()

```
import numpy as np

a = np.array([[1,2,3], [4,5,6], [7,8,9]])
print(np.ptp(a))
print(np.ptp(a,0))
print(np.ptp(a,1))
```

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
8
[6 6 6]
[2 2 2]
```

对于相同的数组 a，np.ptp(a) 可以统计数组中最大值与最小值的差，即 9-1=8。同样 ptp(a,0) 统计的是沿着 axis=0 轴的最大值与最小值之差，即 7-1=6（当然 8-2=6,9-3=6，第三行减去第一行的 ptp 差均为 6），ptp(a,1) 统计的是沿着 axis=1 轴的最大值与最小值之差，即 3-1=2（当然 6-4=2, 9-7=2，即第三列与第一列的 ptp 差均为 2）。

统计数组的百分位数 percentile()

首先得知道一下什么是百分位数: 百分位数_百度百科

如果将一组数据从小到大排序，并计算相应的累计百分位，则某一百分位所对应数据的值就称为这一百分位的百分位数。可表示为：一组n个观测值按数值大小排列。如，处于p%位置的值称第p百分位数。

百分位通常用第几百分位来表示，如第五百分位，它表示在所有测量数据中，测量值的累计频次达5%。以身高为例，身高分布的第五百分位表示有5%的人的身高小于此测量值，95%的身高大于此测量值。

值得注意的是，用99个数值或者是99个点来按大小排列出来之后，划分为100个等分，而这99个数值或者99个点就是百分位数。中位数是第五十个百分位数。

下面的步骤来说明如何计算第p百分位数

第1步：以递增顺序排列原始数据（即从小到大排列）。

第2步：计算指数i=np%

第3步：

l）若 i 不是整数，将 i 向上取整。大于i的毗邻整数即为第p百分位数的位置。

2. 若i是整数，则第p百分位数是第i项与第(i+l)项数据的平均值。

除了以上方法，再介绍另外一种方法，这种方法是SPSS所用方法，也是SAS所用方法之一。

第一步：将n个变量值从小到大排列，X(j)表示此数列中第j个数。

第二步：计算指数，设(n+1)P%=j+g，j为整数部分，g为小数部分。

第三步：1)当g=0时：P百分位数=X(j);

2)当g≠0时：P百分位数=g*X(j+1)+(1-g)*X(j)=X(j)+g*[X(j+1)-X(j)]。

分位数是用于衡量数据的位置的量度，但它所衡量的，不一定是中心位置。百分位数提供了有关各数据项如何在最小值与最大值之间分布的信息。对于无大量重复的数据，第p百分位数将它分为两个部分。大约有p%的数据项的值比第p百分位数小；而大约有(100-p)%的数据项的值比第p百分位数大。对第p百分位数，严格的定义如下：

第p百分位数是这样一个值，它使得至少有p%的数据项小于或等于这个值，且至少有(100-p)%的数据项大于或等于这个值。

```
import numpy as np

a = np.array([[1,2,3], [4,5,6], [7,8,9]])
print(np.percentile(a, 0))
print(np.percentile(a, 100))
print(np.percentile(a, 50))
print(np.percentile(a, 50, axis=0))
print(np.percentile(a, 50, axis=1))
```

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
1.0
9.0
5.0
3.4
[4. 5. 6.]
[2. 5. 8.]
```

同样，percentile() 代表着第 p 个百分位数，这里 p 的取值范围是 0-100，如果 p=0，那么就是取最小值，p=100则求得最大值，p=50则求得中位数。

统计数组中的中位数 median()、平均数 mean()

```
import numpy as np

a = np.array([[1,2,3], [4,5,6], [7,8,9]])
#求中位数
print(np.median(a))
print(np.median(a, axis=0))
print(np.median(a, axis=1))
#求平均数
print(np.mean(a))
print(np.mean(a, axis=0))
print(np.mean(a, axis=1))
```

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
5.0
[4. 5. 6.]
[2. 5. 8.]
5.0
[4. 5. 6.]
[2. 5. 8.]
```

可以用 median() 和 mean() 求数组的中位数、平均值，同样也可以求得在 axis=0 和 1 两个轴上的中位数、平均值。

统计数组中的加权平均值 average()

```
import numpy as np

a = np.array([1,2,3,4])
wts = np.array([1,2,3,4])
print(np.average(a))
print(np.average(a, weights=wts))
```

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
2.5
3.0
```

average() 函数可以求加权平均，加权平均的意思就是每个元素可以设置个权重，默认情况下每个元素的权重是相同的，所以 np.average(a)=(1+2+3+4)/4=2.5

也可以指定权重数组 wts=[1,2,3,4]，这样加权平均 np.average(a,weights=wts)=(1×1+2×2+3×3+4×4)/(1+2+3+4)=3.0

统计数组中的标准差 std()、方差 var()

```
import numpy as np

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

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
1.118033988749895
1.25
```

方差的计算是指每个数值与平均值之差的平方求和的平均值，即 mean((x - x.mean())** 2)。标准差是方差的算术平方根。在数学意义上，代表的是一组数据离平均值的分散程度。所以 np.var(a)=1.25, np.std(a)=1.118033988749895。

NumPy 排序

```
import numpy as np

a = np.array([[4,3,2],[2,4,1]])
print(np.sort(a))
print(np.sort(a, axis=None))
print(np.sort(a, axis=0))
print(np.sort(a, axis=1))
```

运行结果

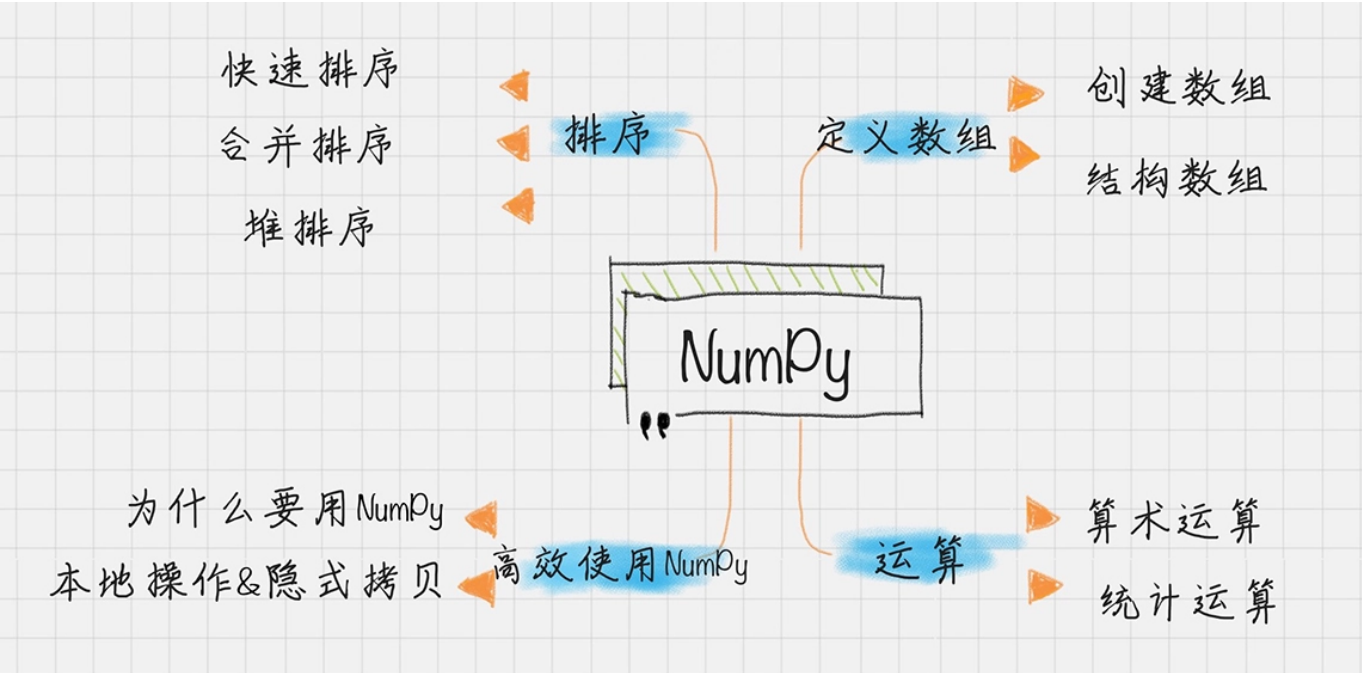
```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
[[2 3 4]
 [1 2 4]]
[1 2 2 3 4 4]
[[2 3 1]
 [4 4 2]]
[[2 3 4]
 [1 2 4]]
```

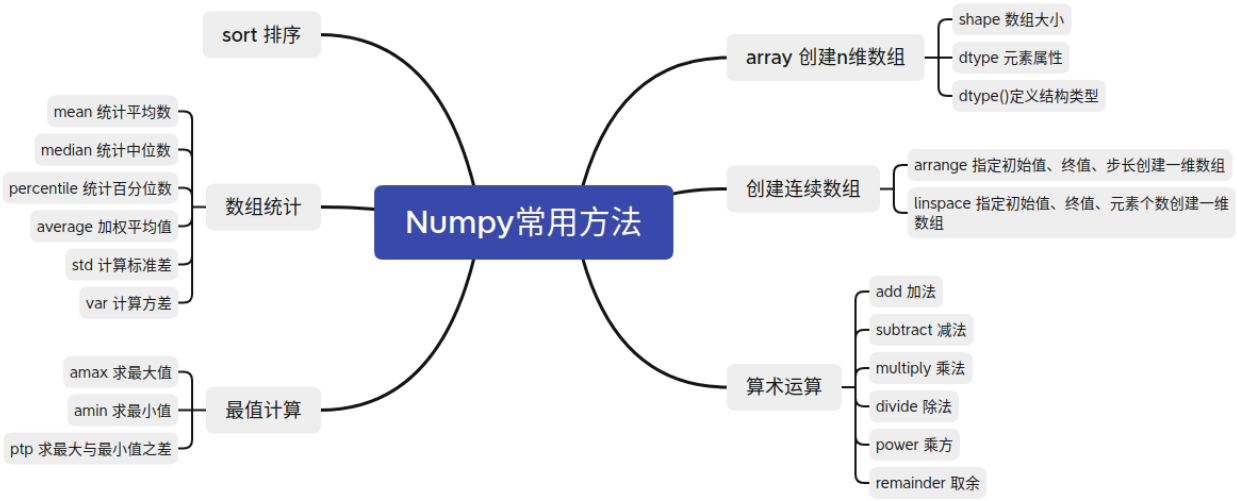
这里可以使用 sort 函数，sort(a, axis=-1, kind='quicksort', order=None)，默认情况下使用的是快速排序。

在 kind 里，可以指定 quicksort、mergesort、heapsort 分别表示快速排序、合并排序、堆排序。同样 axis 默认是 -1，即沿着数组的最后一个轴进行排序，也可以取不同的 axis 轴，或者 axis=None 代表采用扁平化的方式作为一个向量进行排序。

另外 order 字段，对于结构化的数组可以指定按照某个字段进行排序。

总结





案例

假设一个团队里有 5 名学员，成绩如下表所示。用 NumPy 统计下这些人在语文、英语、数学中的平均成绩、最小成绩、最大成绩、方差、标准差。然后把这些人的总成绩排序，得出名次进行成绩输出。

姓名	语文	英语	数学
张飞	66	65	30
关羽	95	85	98
赵云	93	92	96
黄忠	90	88	77
典韦	80	90	90

代码如下

```python
import numpy as np

# 自定义结构
persontype = np.dtype({
    'names': ['name', 'chinese', 'math', 'english'],
    'formats': ['S32','i', 'i', 'i']})
# 初始化数据
peoples = np.array([('ZhangFei', 66, 65, 30),
                    ('GuanYu', 95, 85, 98), ('ZhaoYun', 93, 92, 96),
                    ('HuangZhong', 90, 88, 77), ('DianWei', 80, 90, 90)],
                   dtype=persontype)

chineses = peoples[:]['chinese']  # 语文分数数组
maths = peoples[:]['math']        # 数学分数数组
englishs = peoples[:]['english']  # 英语分数数组

print(f"语文 最高分:{np.amax(chineses)} 最低分:{np.amin(chineses)} 平均分:
{np.mean(chineses)} 标准差:{np.std(chineses)} 方差:{np.var(chineses)}")
print(f"数学 最高分:{np.amax(maths)} 最低分:{np.amin(maths)} 平均分:
{np.mean(maths)} 标准差:{np.std(maths)} 方差:{np.var(maths)}")
print(f"英语 最高分:{np.amax(englishs)} 最低分:{np.amin(englishs)} 平均分:
{np.mean(englishs)} 标准差:{np.std(englishs)} 方差:{np.var(englishs)}")
ranking = sorted(peoples, key=lambda x: x[1]+x[2]+x[3], reverse=True) # 按总分排
序
for item in ranking:
    print(item)
```

运行结果

```
/home/hwx/PycharmProjects/testpy/venv/bin/python /home/hwx/PycharmProjects/testpy/main.py
语文 最高分:95 最低分:66 平均分:84.8 标准差:10.721940122944169 方差:114.96000000000001
数学 最高分:92 最低分:65 平均分:84.0 标准差:9.777525249264253 方差:95.6
英语 最高分:98 最低分:30 平均分:78.2 标准差:25.19047439013406 方差:634.56
(b'ZhaoYun', 93, 92, 96)
(b'GuanYu', 95, 85, 98)
(b'DianWei', 80, 90, 90)
(b'HuangZhong', 90, 88, 77)
(b'ZhangFei', 66, 65, 30)
```