数据分析 lib

for learning

Ethan[1]

[1]Peking University

最初创建于：2024 年 08 月 20 日

最后更新于：2024 年 08 月 23 日

目录

章节 1. numpy

1.1. 生成矩阵

1.1.1. 头

```
import numpy as np
```

1.1.2. 生成矩阵

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

1.1.3. 性质

```
print(arr)
print('dimensions: ',arr.ndim) # 维度
print('shape: ',arr.shape)     # 各维度的长度
print('size: ', arr.size)      # 总元素数
```

输出结果:

```
[[1 2 3]
 [4 5 6]]
dimensions:  2
shape:  (2, 3)
size:  6
```

1.1.4. 生成特殊矩阵

用 np.zeros()生成全零矩阵

```
arr_zeros=np.zeros( (2,3) )
```

用 np.ones()生成全一矩阵

```
arr_ones=np.ones( (2,3) )
```

生成随机矩阵(需要 import random)

```
np.random.random()
arr_random=np.random.random((2,3))
```

用 np.arange()生成数列

```
arr=np.arange(6,12)
```

用 np.arange().reshape()将数列转成矩阵

```
arr=np.arange(6,12).reshape( (2,3) )
```

用 np.linspace(开始，结束，多少个点平分线段),同样也可以用 reshape()

```
arr_1=np.linspace(1,5,3)
print(arr_1)
arr_2=np.linspace(1,13,3)
print(arr_2)
# 输出结果：
[1. 3. 5.]
[ 1.  7. 13.]
```

1.2. 运算

```
arr1 = np.array([1,2,3,6])
arr2 = np.arange(4)
```

1.2.1. 矩阵加/减法

```
arr_sum = arr1 + arr2
arr_sub = arr1 - arr2
```

1.2.2. 矩阵乘法

```
arr_multi = arr1**3 # 求每个元素的立方

arr_multi = arr1*arr2 # 元素逐个相乘(非矩阵乘法)
#[1,2,3,6] * [0   = [1,2,3,6  * [0,0,0,0  = [0,0,0,0
#            1       1,2,3,6     1,1,1,1     1,2,3,6
#            2       1,2,3,6     2,2,2,2     2,4,6,12
#            3]      1,2,3,6]    3,3,3,3]    3,6,9,18]
# 对应元素相乘,不足的补齐

arr_multi = np.dot(arr1, arr2.reshape((4,1))) # 维度 1*4 和 4*1 矩阵相乘
arr_multi = arr1.dot(arr2.reshape((4,1))) # 也可以使用矩阵名.dot(矩阵名)
```

两个数组的点积。具体说来：

1. 如果 a 和 b 都是一维数组，则它是向量的内积（无复共轭）。
2. 如果 a 和 b 都是二维数组，则为矩阵乘法， 但首选使用 OR。`a \@ b`
3. 如果 a 或 b 为 0-D（标量），则它等价于 和 使用 或 是 首选。`numpy.multiply(a, b)a \* b`
4. 如果 a 是 N 维数组，b 是一维数组，则它是 A 和 B 的最后一个轴。
5. 如果 a 是 N 维数组，b 是 M-D 数组 （where ），则它是 A 的最后一个轴和 B 的倒数第二个轴上的乘积：`M>=2`

1.2.3. 三角运算

```
arr_sin=np.sin(arr1)  #np.sin()/np.cos()/np.tan()
```

1.2.4. 逻辑运算

```
print(arr1<3)  # 查看 arr1 矩阵中哪些元素小于 3，返回[ True True False False]
```

1.2.5. 矩阵求和

```
np.sum(arr1) # 矩阵求和
np.sum(arr1,axis=0) # 矩阵每列求和
np.sum(arr1,axis=1).reshape(2,1) # 矩阵每行求和，reshape 是为了以列向量表示，非必需
```

这里的 **axis** 即轴,**0** 即沿着行的轴计算得到列的性质,依次类推。

1.2.6. 矩阵最大最小值

```
np.min(arr1) # 矩阵最小值
np.max(arr1) # 矩阵最大值
np.min(arr1,axis=0) # 矩阵每列最小值
np.min(arr1,axis=1) # 矩阵每行最小值
```

1.2.7. 矩阵平均值与中位数

```
print(np.mean(arr1)) # 输出矩阵平均值，也可以用 arr1.mean()
print(np.median(arr1)) # 输出矩阵中位数
```

1.2.8. 输出矩阵某些值的位置

```
arr1=np.arange(2,14).reshape((3,4))
print(arr1)

print(np.argmin(arr1)) # 输出矩阵最小值的位置，0
print(np.argmax(arr1)) # 输出矩阵最大值的位置，11

print(np.cumsum(arr1)) # 返回行向量:[前一个数的和，前两个数的和...]
print(np.diff(arr1)) # 输出相邻两个数的差值

arr_zeros=np.zeros((3,4))
print(np.nonzero(arr_zeros)) #输出矩阵非零元素位置,返回多个行向量,第 i 个行向量表示第 i 个维度
```

1.2.9. 矩阵操作

```
print(np.sort(arr1)) # 矩阵逐行排序
print(np.transpose(arr1)) # 矩阵转置，也可以用 *arr1.T*
print(np.clip(arr1,x,y)) # 将矩阵中小于 x 的数置 x，大于 y 的数置 y
```

1.3. 索引

```
arr1=np.array([1,2,3,6])
arr2=np.arange(2,8).reshape(2,3)
```

1.3.1. 访问

```
print(arr1[0]) # 索引从 0 开始计数
print(arr2[0][2]) # arr[行][列], 也可以用 arr[行,列]
print(arr2[0,:]) # 用:来代表所有元素的意思, 这里即第 0 行的所有元素
print(arr2[0,0:3]) # 表示输出第 0 行, 从第 0 列到第 2 列所有元素
                   # 注意 python 索引一般是左闭右开
```

1.3.2. 遍历

通过 for 循环每次输出矩阵的一行

```
for row in arr2:
    print(row)
```

如果需要输出每列,转置矩阵即 arr2.T

压成一行逐个输出

```
arr2_flat=arr2.flatten()
print(arr2_flat) # 输出的是行向量

for i in arr2.flat: # 也可以用 arr2.flatten()
    print(i) # 一行只输出一个元素
```

1.4. 合并与分割

1.4.1. 矩阵合并

```
arr_hor=np.hstack((arr1,arr2)) # 水平合并, horizontal
arr_ver=np.vstack((arr1,arr2)) # 垂直合并, vertical
```

1.4.2. 矩阵分割

```
print(np.split(arr3,4,axis=1)) # 将矩阵按列均分成 4 块
print(np.split(arr3,2,axis=0)) # 将矩阵按行均分成 2 块
print(np.hsplit(arr3,4)) # 将矩阵按列均分成 4 块
print(np.vsplit(arr3,2)) # 将矩阵按行均分成 2 块
print(np.array_split(arr3,3,axis=1)) # 将矩阵进行不均等划分,倾向于前面数量更多,例如 4 列分三组, 为 2: 1: 1
```

1.5. 复制

1.5.1. 浅复制(指向同一个地址)

```
arr1=np.array([3,1,2,3])
a1=arr1
b1=a1
```

此时 id(a1)==id(b1)==id(arr1)
通过三个名称里的任何一个名称改变矩阵，另外两个矩阵同时改变
本质上是同一个矩阵

1.5.2. 深复制()

```
b2=arr2.copy() # 深复制，此时 b2 拥有不同于 arr2 的空间
a2=b2.copy()
```

三者 id 不同，指向不同地址
改变值互不影响

1.6. 线性代数相关运算

1.6.1. 求范数

```
a=np.array([5,12])
b=np.linalg.norm(a) # norm 表示范数，默认求 2 范数，ord=1 求 1 范数，ord=np.inf 求无穷范数
```

1.6.2. 求矩阵的迹、行列式、秩、特征值、特征向量

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

```
c=np.trace(b)  # 求矩阵的*迹*（主对角线上各个元素的总和）

c=np.linalg.det(b)    # 求矩阵的*行列式值*，这里是一个很小的值 6.66133814775e-16
# 如果希望输出为 0，使用 round(c, 2)，四舍五入保留小数点后两位
# 不过对精度要求高可以使用 decimal 模块

c=np.linalg.matrix_rank(b) # 求矩阵的*秩*

u,v=np.linalg.eig(b) # u 为特征值组成的矢量，v 为特征向量组成的矩阵，每一列为一个特征向量
```

1.6.3. 矩阵分解

```
# Cholesky 分解并重建
```

```
d = np.array([
    [2, 1],
    [1, 2]
])

l = np.linalg.cholesky(d)
print(l) # 得到下三角矩阵
e=np.dot(l, l.T)
print(e) # 重建得到矩阵 d
```

```
# 对不正定矩阵，进行 SVD 分解并重建
U, s, V = np.linalg.svd(d)

S = np.array([
    [s[0], 0],
    [0, s[1]]
])

print(np.dot(U, np.dot(S, V))) # 重建得到矩阵 d
```

章节 2. matplotlib

2.1. 绘制折线图

2.1.1. 头

```
from matplotlib import pyplot as plt ->导入 pyplot
import random
import matplotlib
```

2.1.2. 设置中文字体

```
font = {'family' : 'MicroSoft YaHei',
        'weight' : 'bold'   }
matplotlib.rc("font",**font)
```

2.1.3. 调整窗口大小以及绘图清晰度

```
fig = plt.figure(figsize=(10,8),dpi=80)
#figsize=(长, 宽)
#dpi=像素点个数/单位
```

2.1.4. x 与 y 轴

```
x = range(2,26,2)
y = [15,13,14,5,17,20,25,26,26,24,22,18,15]
```

2.1.5. 设置 x 轴的刻度

```
plt.xticks()
```

2.1.6. 设置 y 轴的刻度

```
plt.yticks()
```

2.1.7. 绘图

```
plt.plot(x,y)
#color 线条颜色
#linestyle 线条风格
#linewidth 线条粗细
#alpha 透明度
#label 指定显示的图例内容
```

2.1.8. 保存

```
plt.savefig("./figures/line.png")
```

2.1.9. 展示图形

```
plt.show()
```

2.1.10. 调整 x 轴的刻度(字符串)

```
_x = list(x)
_xtick_labels = ["10点{}分".format(i) for i in range(60)]
_xtick_labels += ["11点{}分".format(i) for i in range(60)]
plt.xticks(_x[::3],_xtick_labels[::3],rotation = 45)
#rotation 为顺时针旋转的角度
```

2.1.11. 添加描述信息

```
plt.xlabel("时间")
plt.ylabel("温度 单位(℃)")
plt.title("10点到12点每分钟的气温变化情况")
```

2.1.12. 绘制网格

```
plt.grid(alpha = 0.4)
#alpha 调整透明度
```

2.1.13. 添加图例

```
plt.legend()
#loc 调整标签位置 默认右上角
```

2.2. 绘制散点图

2.2.1. 绘制

```
plt.scatter(x,y)
```

2.3. 绘制条形图

2.3.1. 绘制

```
plt.bar(x,y)
```

可传参数 width 代表其宽度

2.3.2. 绘制横着的条形图

```
plt.barh(x,y)
```

此时需要注意设置条形图的宽度不再用 width,而是 height

2.4. 绘制直方图

2.4.1. 计算组数

```
d = # 组距
num_bins = (max(a)-min(a))//d+1
```

2.4.2. 绘制

```
plt.hist(a,num_bins,range = (min(a),min(a)+d*num_bins))
```

此时为了网格与 x 轴对齐,我们手动传入 range 实际最小值和绘图最大值作为绘图总区间
此时绘图最大值可能与实际最大值不相等,但实际最大值包含在最后一个区间内

2.4.3. 设置 x 轴的刻度

```
plt.xticks(range(min(a),min(a)+d*num_bins+d,d))
```

2.4.4. 两种代码的对比图

第一种,如图 2.1

```
from matplotlib import pyplot as plt

a = [11,11,17,17,17,21,21,21,21,30,30,30,30,30]
d = 5
num_bins = (max(a)-min(a))//d+1
plt.hist(a,num_bins,range = (min(a),min(a)+d*num_bins))
plt.xticks(range(min(a),min(a)+d*num_bins+d,d))
plt.grid(alpha = 0.4)
plt.show()
```

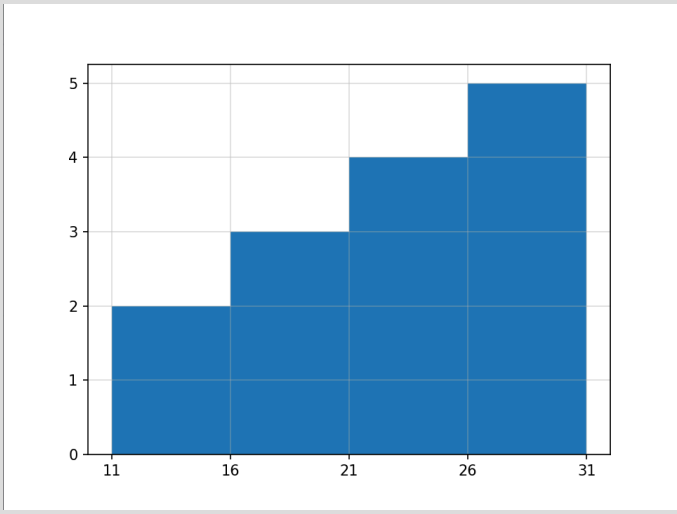

图 2.1　对齐的直方图

第二种，如图 2.2

```
from matplotlib import pyplot as plt

a = [11,11,17,17,17,21,21,21,21,30,30,30,30,30]
d = 5
num_bins = (max(a)-min(a))//d+1
plt.hist(a,num_bins)
plt.xticks(range(min(a),max(a)+d,d))
plt.grid(alpha = 0.4)
plt.show()
```

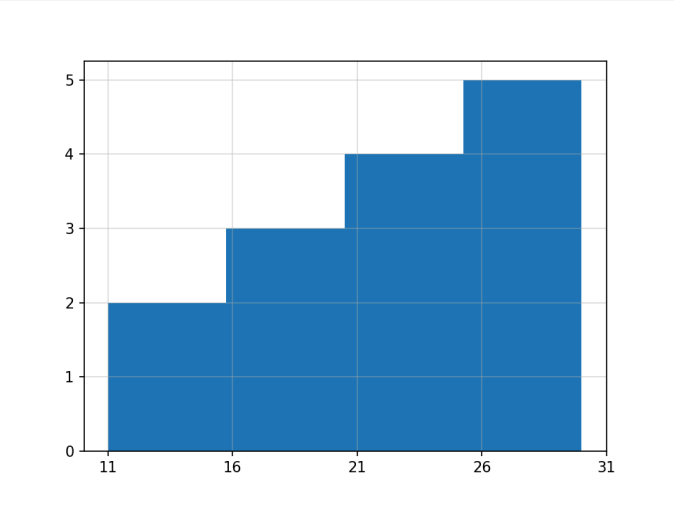

图 2.2　未对齐的直方图

2.5. 支持的其他图形

请访问 ： https://matplotlib.org/stable/gallery/index.html

参考文献