大连理工大学
未来技术学院/人工智能学院
SCHOOL OF FUTURE TECHNOLOGY, SCHOOL OF ARTIFICIAL INTELLIGENCE, DUT

第10章 Python机器学习常用工具1

AI程序设计课程组

目录页

Numpy介绍

科学计算

- 科学计算是解决科学和工程中的数学问题利用计算机进行的数值计算，它不仅是科学家在运算自然规律时所使用的计算工具，更是普通人提升专业化程度的必要手段。

- 开展基本的科学计算需要两个步骤：组织数据和展示数据。

- 组织数据是运算的基础，也是将客观世界数字化的必要手段；

- 展示数据是体现运算结果的重要方式，也是展示结论的有力武器。

Numpy介绍

- NumPy（Numerical Python）是Python中一个重要的数值计算库，它提供了用于处理大型多维数组和矩阵的数据结构，以及用于执行各种数学、逻辑、线性代数和统计操作的函数。

- NumPy是数据科学和科学计算中的关键工具，因为它能够高效地处理大数据集，提供了广泛的数学和统计函数，以及优化的数组操作。

Numpy介绍

- NumPy 库处理的最基础数据类型是由同种元素构成的多维数组（ndarray），简称“数组”。
- 数组中所有元素的类型必须相同，数组中元素可以用整数索引，序号从0开始。
- ndarray 类型的维度(dimensions)叫做轴(axes)，轴的个数叫做秩(rank)。一维数组的秩为1，二维数组的秩为2，二维数组相当于由两个一维数组构成。

基本数组数据类型

NumPy提供的多种基本数据数组类型。

- **整数类型**

NumPy提供了多种整数类型，包括：int8、int16、int32、int64，分别表示8、16、32和64位的有符号整数；还有无符号整数类型，如uint8、uint16、uint32、uint64

- **浮点数类型**

NumPy包括不同精度的浮点数类型，如float16、float32、float64，分别表示16位、32位和64位的浮点数。

基本数组数据类型

NumPy提供的多种基本数据数组类型。

- **复数类型**

NumPy支持复数类型，如complex64和complex128，分别表示64位和128位的复数。这对于处理复数数学运算非常有用。

- **布尔类型**

NumPy提供bool类型，用于存储布尔值（True或False）。

总结：

这些数据类型允许你在NumPy数组中存储各种不同类型的数据，同时也提供了广泛的数学和逻辑操作，能够对这些数组进行高效的处理和计算。这些数据类型的选择取决于你的具体应用需求和所处理的数据类型。

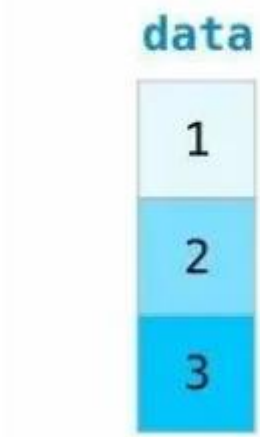
常用数据操作

NumPy提供了丰富的数组操作方法，包括创建数组、增加元素、删除元素、修改元素和查询元素等。

例如：创建数组

```python
import numpy as np

# 从列表创建一维数组
arr = np.array([1, 2, 3, 4, 5])

# 从嵌套列表创建二维数组
matrix = np.array([[1, 2, 3], [4, 5, 6]])

# 创建全零数组
zeros = np.zeros((2, 3))  # 2行3列的全零数组

# 创建全一数组
ones = np.ones((3, 2))  # 3行2列的全一数组
```

```python
data = np.array([1,2,3])
```

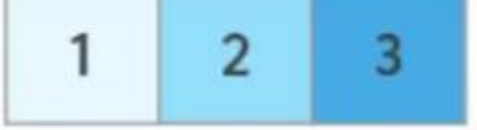


常用数据操作

（1）创建一个新数组

a =

```
np.array([1, 2, 3])
```



（2）np.zeros_like(a)：可以创建一个新数组，其形状和类型与给定数组a相同，但是所有元素都被设置为 0。

```
np.zeros_like(a)
```

（3）np.ones_like(a)：可以创建一个新数组，其形状和类型与给定数组a相同，但是所有元素都被设置为 1。

（4）np.empty_like(a)：返回一个与给定数组a具有相同形状和类型的新数组。

在数组末尾添加元素

numpy.append(arr, values, axis=None)

```
arr = np.array([1, 2, 3])
new_arr = np.append(arr, 4)  # 在arr末尾添加4
```

numpy.append函数用来向数组的末尾追加值。

它接收三个参数：arr、values和axis。

(1)arr是要追加值的数组；

(2)values是要追加到arr数组末尾的数组或者值；

(3)axis参数是可选的，用来指定沿着哪个轴向进行追加操作

在数组末尾添加元素

numpy. concatenate((a1, a2, …), axis=0)

- 其中，a1,a2....是数组类型的参数，传入的数组必须具有相同的形状。
- axis 指定拼接方向，默认axis = 0 (逐行拼接) (纵向的拼接沿着axis= 1方向)
- 注:一般axis = 0，就是对该轴问的数组进行操作，操作方向是另外一个轴，即axis=1。

- concatenate()比append()效率更高，适合大规模的数据拼接，能够一次完成多个数组的拼接。append () 不能进行三个及以上数组的直接拼接。

在数组末尾添加元素

numpy. concatenate((a1, a2, …), axis=0)

```
In [27]: import  numpy as np
In [28]: a = np.array([1, 2])
In [29]: b = np.array([3, 4])
In [30]: c = np.array([5, 6])
In [31]: np.concatenate((a,b,c))
Out[31]: array([1, 2, 3, 4, 5, 6])
```

多个数组同时拼接

```
In [33]: import  numpy as np
In [34]: a = np.array([[1, 2], [3, 4]])
In [35]: b = np.array([[5, 6], [7, 8]])
In [36]: np.concatenate((a, b), axis=0)
Out[36]:
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
In [37]: np.concatenate((a, b), axis=1)
Out[37]:
array([[1, 2, 5, 6],
       [3, 4, 7, 8]])
```

沿着不同维度进行拼接

删除数组中的元素

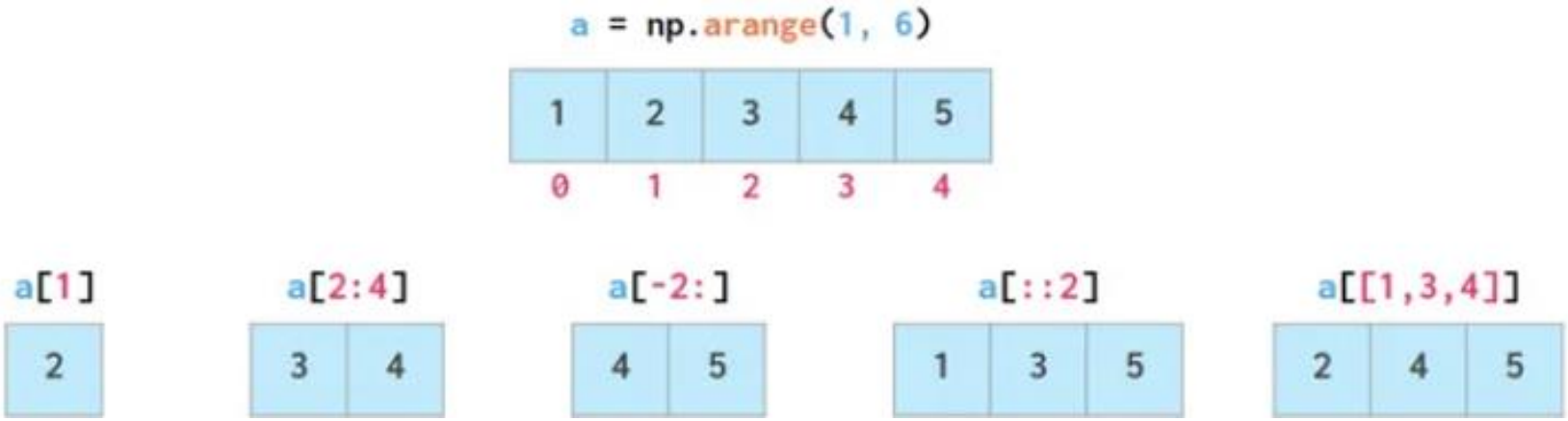
```
arr = np.array([1, 2, 3, 4, 5])
new_arr = np.delete(arr, [1, 3])  # 删除索引为1和3的元素
```

numpy.delete(arr, obj, axis)
(1)arr：需要处理的矩阵
(2)obj：要删除的索引位置
(3)axis：这是一个可选参数，axis = None，1，0
axis=None：arr会先按行展开，然后按照obj，删除第obj-1（从0开始）位置的数，返回一个行矩阵。
axis = 0：arr按行删除
axis = 1：arr按列删除

修改数组中的元素

```
arr = np.array([1, 2, 3, 4, 5])
arr[2] = 10   # 将索引为2的元素修改为10
```

numpy索引



查询元素

获取数组的形状

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
shape = arr.shape  # 返回(2, 3)，表示2行3列的数组
```

访问数组元素

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
element = arr[1, 2]  # 获取第二行第三列的元素，值为6
```

Numpy中基本科学计算函数

Numpy可以用于进行数学运算、统计分析、线性代数和其他科学计算任务。它提供了许多基本的科学计算函数。

1. 数学运算函数

1.1、数学函数

NumPy提供了许多常见的数学函数，如三角函数（sin、cos、tan）、指数函数（exp）、对数函数（log）、平方根函数（sqrt）等。

1.1、数学函数

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

# 三角函数
sin_x = np.sin(x)
cos_x = np.cos(x)
tan_x = np.tan(x)

# 指数函数
exp_x = np.exp(x)

# 对数函数
log_x = np.log(x)

# 平方根函数
sqrt_x = np.sqrt(x)
```

Numpy中基本科学计算函数

Numpy可以用于进行数学运算、统计分析、线性代数和其他科学计算任务。它提供了许多基本的科学计算函数。

1.2、数组运算

NumPy支持数组之间的逐元素运算，包括加法、减法、乘法、除法等。

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

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
```

数组之间的加减乘除

```
1   import numpy as np
2   a = np.array([1,2])
```

- 数组加法

a为数组，直接+3表示对数组中每个元素都加3

```
print(a+3)
```

→

```
[4 5]
```

- 数组乘法

a为数组，直接*3表示对数组中每个元素都乘3

```
print(a*3)
```

→

```
[3 6]
```

Numpy中基本科学计算函数

对数组的每个元素取整

```
import numpy as np
a = np.array([1.1,1.5,1.9,2.5])
```

- floor函数（其功能是“向下取整”）

```
print(np.floor(a))
```

→

```
[1. 1. 1. 2.]
```

- ceil函数（返回数字的上入整数）

```
print(np.ceil(a))
```

→

```
[2. 2. 2. 3.]
```

- round函数（返回数字的四舍五入整数）

```
print(np.round(a))
```

→

```
[1. 2. 2. 2.]
```

求数组的最大、最小数以及均值

```
import numpy as np
a = np.array([1,2,3])
```

- max函数（返回给定参数的最大值）

```
print(np.max(a))
```

```
3
```

- min函数（返回给定参数的最小值）

```
print(a.min())
```

```
1
```

- mean函数（计算给定数组沿指定轴的算术平均值）

```
print(a.mean())
```

```
2.0
```

求数组的每个数求解 三角函数

- 求三角函数

np.sin(	np.pi	np.pi/2	)	=	0.	1.
np.arcsin(	0.	1.	)	=	0.	1.57

sin	arcsin	sinh	arcsinh
cos	arccos	cosh	arccosh
tan	arctan	tanh	arctanh

2、统计分析函数

2.1 基本统计函数

NumPy提供了一系列基本统计函数，如mean（平均值）、median（中位数）、std（标准差）、var（方差）等。

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

mean_value = np.mean(data)
median_value = np.median(data)
std_dev = np.std(data)
variance = np.var(data)
```

3、线性代数函数

3.1. 矩阵操作

NumPy支持矩阵操作，包括矩阵乘法、求逆、行列式计算等

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

# 矩阵乘法
matrix_product = np.dot(A, B)

# 求逆矩阵
inverse_A = np.linalg.inv(A)

# 计算行列式
determinant_A = np.linalg.det(A)
```

Numpy中基本科学计算函数

np.dot()函数主要有两个功能，向量点积和矩阵乘法，这里简单列举了三种最常用到的情况

（1）np.dot(a, b)，其中a为一维向量，b为一维向量，此时为向量点积

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
b = np.array([6, 7, 8, 9, 10])
print(np.dot(a, b))

output:
130
[Finished in 0.2s]
```

Numpy中基本科学计算函数

（2）np.dot(a, b)，其中a为二维矩阵，b为一维向量，这时b会被当做一维矩阵进行计算

```
import numpy as np

a = np.random.randint(0,10, size = (5,5))
b = np.array([1,2,3,4,5])
print("the shape of a is " + str(a.shape))
print("the shape of b is " + str(b.shape))
print(np.dot(a, b))


output:
the shape of a is (5, 5)
the shape of b is (5,)
[42 85 50 81 76]
```

Numpy中基本科学计算函数

（3）np.dot(a ,b)，其中a和b都是二维矩阵，此时dot进行的是矩阵乘法运算

```
import numpy as np

a = np.random.randint(0, 10, size = (5, 5))
b = np.random.randint(0, 10, size = (5, 3))
print("the shape of a is " + str(a.shape))
print("the shape of b is " + str(b.shape))
print(np.dot(a, b))


output:
the shape of a is (5, 5)
the shape of b is (5, 3)
[[ 66  80  98]
 [ 53  60  60]
 [ 65  84  85]
 [ 25 113 101]
 [ 42  78  77]]
```

4. 随机数生成函数

NumPy还提供了用于生成随机数的函数，包括均匀分布、正态分布、随机整数等。

```
# 生成随机整数
random_int = np.random.randint(1, 10, size=(3, 3))

# 生成随机样本，符合标准正态分布
random_normal = np.random.normal(0, 1, size=(3, 3))
```

```
numpy.random.randint(low, high=None, size=None, dtype='l')
```

参数解释:

low：为最小值
high: 为最大值
size: 为数组维度大小
dtype: 为数据类型，默认的数据类型是np.int

返回值:
返回随机整数或整型数组，范围区间为[low,high)，包含low，不包含high; high没有填写时，默认生成随机数的范围是[0，low)

Numpy中基本科学计算函数

np.random.normal(loc=0.0, scale=1.0, size=None)

参数解释：

（1）loc：概率分布的均值（对应着整个分布的中心）
（2）scale：概率分布的标准差（对应于分布的宽度，scale越大，图形越矮胖；scale越小，图形越瘦高。
（3）size：输出的shape。默认为None，只输出一个值

```
In [9]: np.random.normal(0, 1)
Out[9]: -0.5658362038344488
In [10]: np.random.normal(0, 1, 3)
Out[10]: array([-0.85138676,  0.52032297,  2.04450606])
```

Numpy中数据加载与保存

Numpy可以读写文本数据或二进制数据。

NumPy 为ndarray 对象引入了一个简单的文件格式：.npy，npy文件用于存储重建ndarray所需的数据、图形、dtype和其他信息。

- load() 和 save() 函数是读写文件数组数据的两个主要函数。默认情况下，数组是以原始二进制格式保存在扩展名为.npy 的文件中。
- loadtxt() 和 savetxt() 函数处理正常的文本文件(.txt等)

Numpy中数据加载与保存

numpy.save()函数：将数组保存到以 .npy 为扩展名的文件中。

numpy.save(file, arr, allow_pickle=True, fix_imports=True)

参数说明：

- file：要保存的文件，扩展名为 .npy，如果文件路径末尾没有扩展名 .npy，该扩展名会被自动加上。
- arr：要保存的数组

```
import numpy as np

a = np.array([1, 2, 3, 4, 5])
# 保存到 outfile.npy 文件上
np.save( file: 'outfile.npy', a)
```

数据是 Numpy 专用的二进制格式后的数据，因此直接打开会看到是乱码。可以用numpy.load函数读取数据。

Numpy中数据加载与保存

numpy.load()函数：将数组保存到以 .npy 为扩展名的文件中。

np.load(file, mmap mode=None, allow pickle=False, fix imports=True, encoding='ASCII")

参数说明：

- file：要读取的数据文件，文件扩展名为 .npy 或 .npz。

```
b = np.load('outfile.npy')
print(b)
```

```
[1 2 3 4 5]

Process finished with exit code 0
```

Numpy中数据加载与保存

numpy.savez()函数：将多个数组保存到以 npz 为扩展名的文件

numpy.savez(file, *args, **kwds)

file：要保存的文件，扩展名为 .npz，如果文件路径末尾没有扩展名 .npz，该扩展名会被自动加上。

•args：要保存的数组，可以使用关键字参数为数组起一个名字，非关键字参数传递的数组会自动起名为 arr_0, arr_1, … 。

•kwds：要保存的数组使用关键字名称。

```
a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.arange(0, 1.0, 0.2)
np.savez("res.npz", a, b)
```

```
r = np.load("res.npz")    # 读取数据
print(r.files)  # 查看各个数组名称
print(r["arr_0"])  # 数组 a
print(r["arr_1"])  # 数组 b
```

```
['arr_0', 'arr_1']
[[1 2 3]
 [4 5 6]]
[0.  0.2 0.4 0.6 0.8]
```

Numpy中数据加载与保存

savetxt()函数：

savetxt() 函数是以简单的文本文件格式存储数据，对应的使用 loadtxt() 函数来获取数据。

保存：np.savetxt(file, arr, fmt="%d", delimiter=",")
读取：np.loadtxt(file, dtype=int, delimiter=' ')

file：要保存的文件，扩展名为 .txt，
arr：要保存的数组
delimiter：可指定各种分隔符、针对特定列的转换器函数、需要跳过的行数等

```
import numpy as np
a = np.array([1, 2, 3, 4, 5])
np.savetxt('out.txt', a)
b = np.loadtxt('out.txt')
print(b)
```

```
[1. 2. 3. 4. 5.]
```

Matplotlib介绍

Matplotlib介绍

Matplotlib是一个用于绘制高质量图形的Python库，它提供了丰富的绘图功能，允许用户创建各种类型的图表，包括线图、散点图、直方图、柱状图、饼图等。Matplotlib还允许用户自定义图形的外观、添加标签和标题，以及保存图形为各种图像格式。

Matplotlib中有两个核心概念，即图（Figure）和子图（Axes）

什么是图

图是Matplotlib中的最顶层容器，它表示整个绘图区域。在一个Figure对象内，可以包含一个或多个子图（Axes），以便在同一绘图区域上创建多个子图。

可以使用plt.figure()函数创建一个新的Figure对象。

```
import matplotlib.pyplot as plt

fig = plt.figure()  # 创建一个新的Figure对象
```

什么是子图

子图（Axes）是Figure内的绘图区域，它用于绘制具体的图表元素，如折线图、散点图等。

Axes对象包括了绘图区域的坐标轴、标签、标题和所有图表元素。可以使用fig.add_subplot()方法或plt.subplots()函数来创建一个或多个子图。

具体例子见后面。

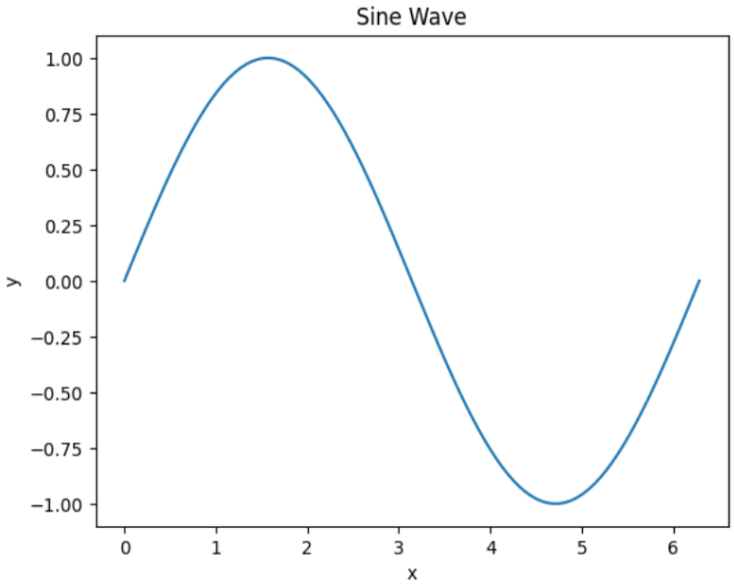
```
# 创建一个包含一个子图的Figure
fig, ax = plt.subplots()

# 创建一个包含多个子图的Figure（2x2的子图布局）
fig, axs = plt.subplots(2, 2)
```

Matplotlib绘图函数

Matplotlib提供了多种绘图函数，用于可视化数据。这些绘图函数通常接受NumPy数组或类似的数据结构作为输入。

1、绘制折线图 - plot()

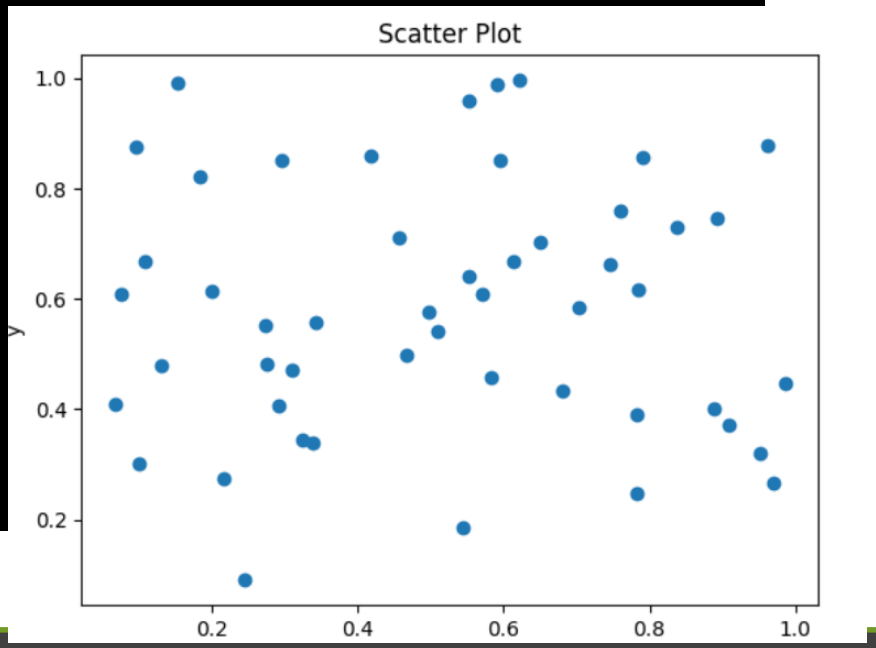
```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2 * np.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.title('Sine Wave')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



2、绘制散点图 - scatter()

scatter()函数用于创建散点图，通常用于显示数据点的分布。

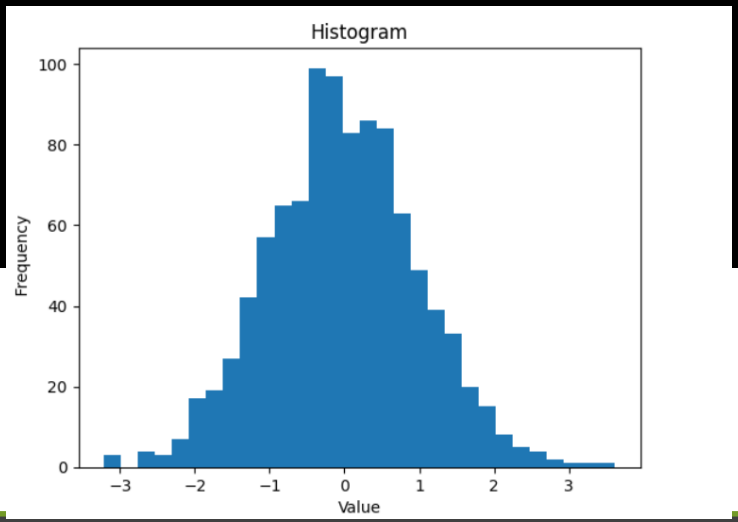
```
data = np.random.rand(50, 2)  # 50个随机数据点，每个点有两个值

x = data[:, 0]  # 提取第一列数据
y = data[:, 1]  # 提取第二列数据

plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



3、绘制直方图 - hist()

hist()函数用于绘制直方图，用于显示数据的分布和频率。

```
data = np.random.normal(0, 1, 1000)  # 生成1000个符合标准正态分布的随机数

plt.hist(data, bins=30)  # 使用30个bins绘制直方图
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

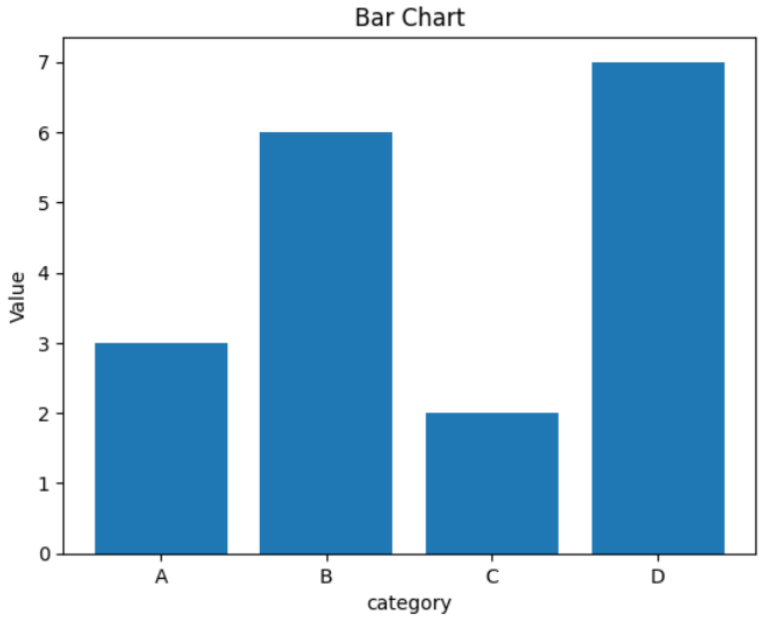


4、绘制柱状图 - bar()

bar()函数用于绘制柱状图，通常用于比较不同类别的数据。

例子：创建柱状图，显示不同类别的值。

```
categories = ['A', 'B', 'C', 'D']
values = np.array([3, 6, 2, 7])

plt.bar(categories, values)
plt.title('Bar Chart')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```



利用 subplot（）函数同时绘制多个子图

- subplot(nrows, ncols, index, **kwargs)

该函数将整个绘图区域分成 nrows 行和 ncols 列。然后从左到右，从上到下的顺序对每个子区域进行编号 1...N ，左上的子区域的编号为 1、右下的区域编号为 N，编号可以通过参数 index 来设置。

设置 numRows = 1，numCols = 2，就是将图表绘制成 1行2列 的图片区域

利用 subplot（）函数同时绘制多个子图

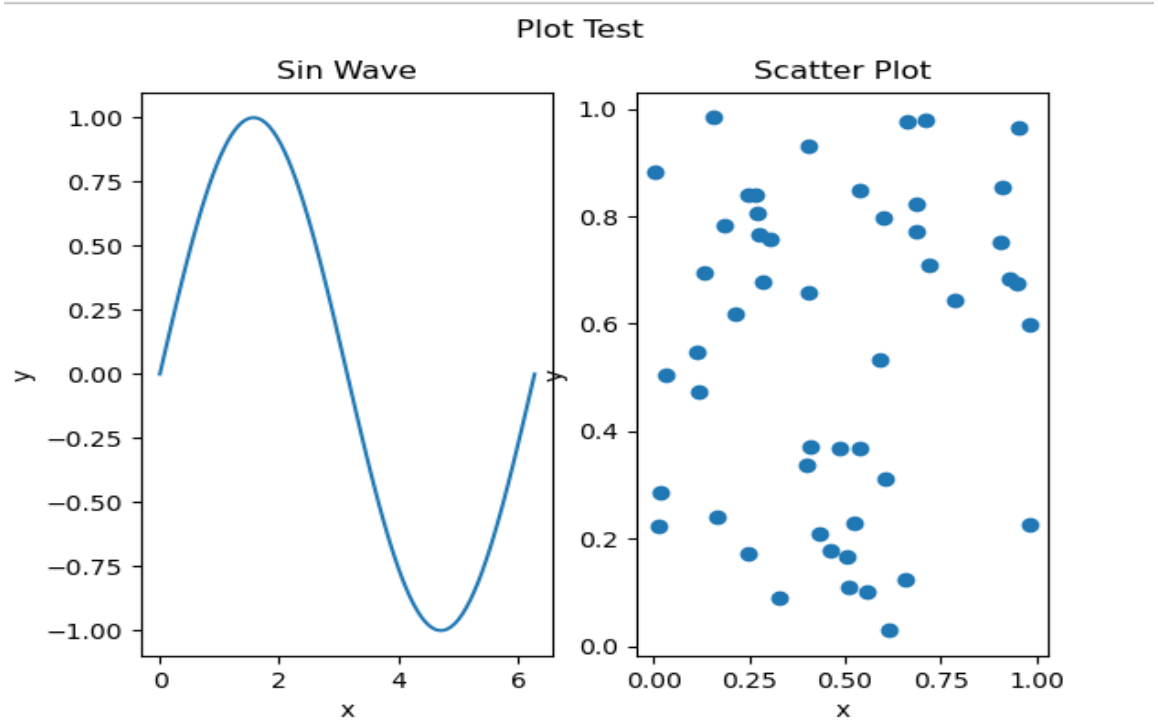
- subplot(nrows, ncols, index, **kwargs)

```
import numpy as np
import matplotlib.pyplot as plt

### 第1个子图为 Sin曲线
x = np.linspace( start: 0, 2 * np.pi, num: 100)
y = np.sin(x)

# 创建包含2个子图的图，排版为1行2列布局，并绘制第 1 个子图
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: x, y)
plt.title('Sin Wave')
plt.xlabel('x')
plt.ylabel('y')
```

```
### 第2个子图为 散点图
data = np.random.rand(50, 2)
x = data[:, 0]
y = data[:, 1]

# 绘制第 2 个子图
plt.subplot( *args: 1, 2, 2)
plt.scatter(x, y)
plt.title('Scatter Plot')
plt.xlabel('x')
plt.ylabel('y')

plt.suptitle("Plot Test")    # 给图加一个总的title
plt.show()
```

利用 **subplot**（）函数同时绘制多个子图

- subplot(nrows, ncols, index, **kwargs)



Scipy介绍

Scipy

SciPy是一个开源的Python科学计算库，构建在NumPy之上，提供了许多高级的科学和工程计算功能。

物理常量和函数

SciPy可以提供一些常数和特殊函数，如物理常量，常用单位，常用函数等等

```
from scipy import constants as C

print("光速: ", C.c)
print('普朗克常数: ', C.h)
print('一英里: ', C.mile)
print('一英寸', C.inch)
```

```
光速:  299792458.0
普朗克常数:  6.62607015e-34
一英里:  1609.3439999999998
一英寸 0.0254
```

special函数库

Scipy中的special模块是一个非常完整的函数库，其中包含了基本数学函数，特殊数学函数以及numpy中所出现的所有函数。伽马函数是概率统计学中经常出现的一个特殊函数，它的计算如下：

```
from scipy import special as S

print(S.gamma(4))
```

```
6.0

进程已结束,退出代码0
```

scipy.linalg包含了多种线性代数中的常用函数

- scipy.linalg.issymmetric(a): 判断方阵a是否对称矩
- scipy.linalg.inv(a): 求方阵a的逆矩阵，和np.linalg.inv(a)类似
- scipy.linalg.norm: 求矩阵或向量范数
- scipy.linalg.det(a): 求方阵的行列式
- scipy.linalg.solve(a.b): 求解ax=b的根，a为方阵。与np.linalg.solve(a,b)类似
- scipy.linalg.solve_triangular(a,b): 求解ax=b方程。a为三角方阵

例子1：利用 scipy.linalg求解线性方程组

假设要求解右侧方程组：

$$
\begin{aligned}
x + 3y + 5z &= 10 \\
2x + 5y + z &= 8 \\
2x + 3y + 8z &= 3
\end{aligned}
$$

```python
import numpy as np
from scipy import linalg

A = np.array([[1, 3, 5], [2, 5, 1], [2, 3, 8]])
B = np.array([10, 8, 3])
x = linalg.solve(A, B)
print(x)
```

```
[-9.28  5.16  0.76]

进程已结束,退出代码0
```

例子2：利用 scipy.linalg计算给定矩阵的特征值和特征向量

```
import scipy.linalg as la
import numpy as np

# 创建一个矩阵
A = np.array([[2, -1, 0], [-1, 2, -1], [0, -1, 2]])

# 计算特征值和特征向量
eigenvalues, eigenvectors = la.eig(A)
print("特征值：", eigenvalues)
print("特征向量：", eigenvectors)
```

```
特征值：[3.41421356+0.j 2.        +0.j 0.58578644+0.j]
特征向量：[[-5.00000000e-01 -7.07106781e-01  5.00000000e-01]
 [ 7.07106781e-01  4.05405432e-16  7.07106781e-01]
 [-5.00000000e-01  7.07106781e-01  5.00000000e-01]]
```

例子3：利用scipy.integrate 求函数积分

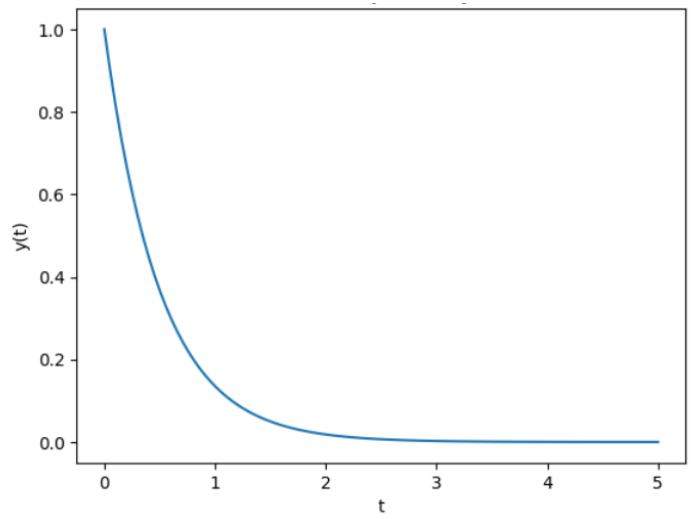
例：对函数x^2在[0,1]区间上求积分

```python
import scipy.integrate as spi

# 定义被积函数
def func(x):
    return x**2

# 进行定积分
result, error = spi.quad(func, 0, 1)
print(f"定积分结果: {result}, 估计误差: {error}")
```

例子4：利用scipy.integrate 求解微分方程

例：求解微分方程dy/dt = -y

```python
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

# 定义微分方程
def dydt(y, t):
    return -2 * y

# 定义初值和时间点
y0 = 1.0
t = np.linspace(0, 5, 100)

# 求解微分方程
y = spi.odeint(dydt, y0, t)

# 绘制结果
plt.plot(t, y)
plt.xlabel('t')
plt.ylabel('y(t)')
plt.title('解微分方程 dy/dt = -2y')
plt.show()
```



scipy.io

scipy.io包提供了多种功能来解决不同格式的文件的输入和输出。

- scipy.io.savemat(file, data)

将数据保存在文件中
file: 文件路径
data: 要保存的数据

```
from scipy import io
import numpy as np
# np.mat()用来创建一个矩阵
B = np.mat([[12], [-2], [10]])
x = [1, 2, 3]
y = [4, 5, 6]
z = [7, 8, 9]
# 保存到mat文件中
io.savemat("a.mat", {"text": B, "x": x, "y": y, "z": z})
```

scipy.io

- scipy. io.loadmat(file) 函数：加载数据
  其中，file为要加载的数据文件

```python
from scipy import io
import numpy as np
# np.mat()用来创建一个矩阵
B = np.mat([[12], [-2], [10]])
x = [1, 2, 3]
y = [4, 5, 6]
z = [7, 8, 9]
# 保存到mat文件中
io.savemat("a.mat", {"text": B, "x": x, "y": y, "z": z})
# 运用loadmat载入数据
data = io.loadmat("a.mat")
print(data["text"])
```

```
[[12]
 [-2]
 [10]]

进程已结束,退出代码0
```

本章小结

- Numpy介绍
- Matplotlib介绍
- Scipy介绍