Python 程序设计

第 9 章 Python 科学计算与可视化

第9章 Python科学计算与可视化

- 9.1 Scipy 科学与数值运算
- 9.2 Numpy 数学函数与计算
- 9.3 Pandas 数据分析与处理
- 9.4 模块7: matplotlib 库的使用
- 9.5 实例13:
- 9.6 实例14:

第9章 Python科学计算与可视化

方法论

- **从Python角度理解的计算和数据呈现**

实践能力

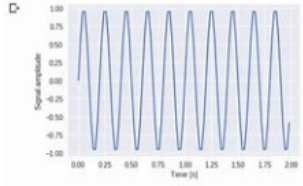
- **Python科学计算与可视化库的使用**

SciPy 科学与数值运算

https://scipy.org/scipylib/

SciPy in Python **Tutorial:**

SciPy in Python is an **open-source library used for solving mathematical, scientific, engineering, and technical problems**. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extention.



安装Python包的方法通常有三种方法：

- 通过pip来进行安装
- 通过conda来进行安装
- 通过whl文件进行

SciPy – Installation and Environment Setup

You can also install SciPy in Windows via pip

```
Python3 -m pip install --user numpy scipy
```

Install Scipy on Linux

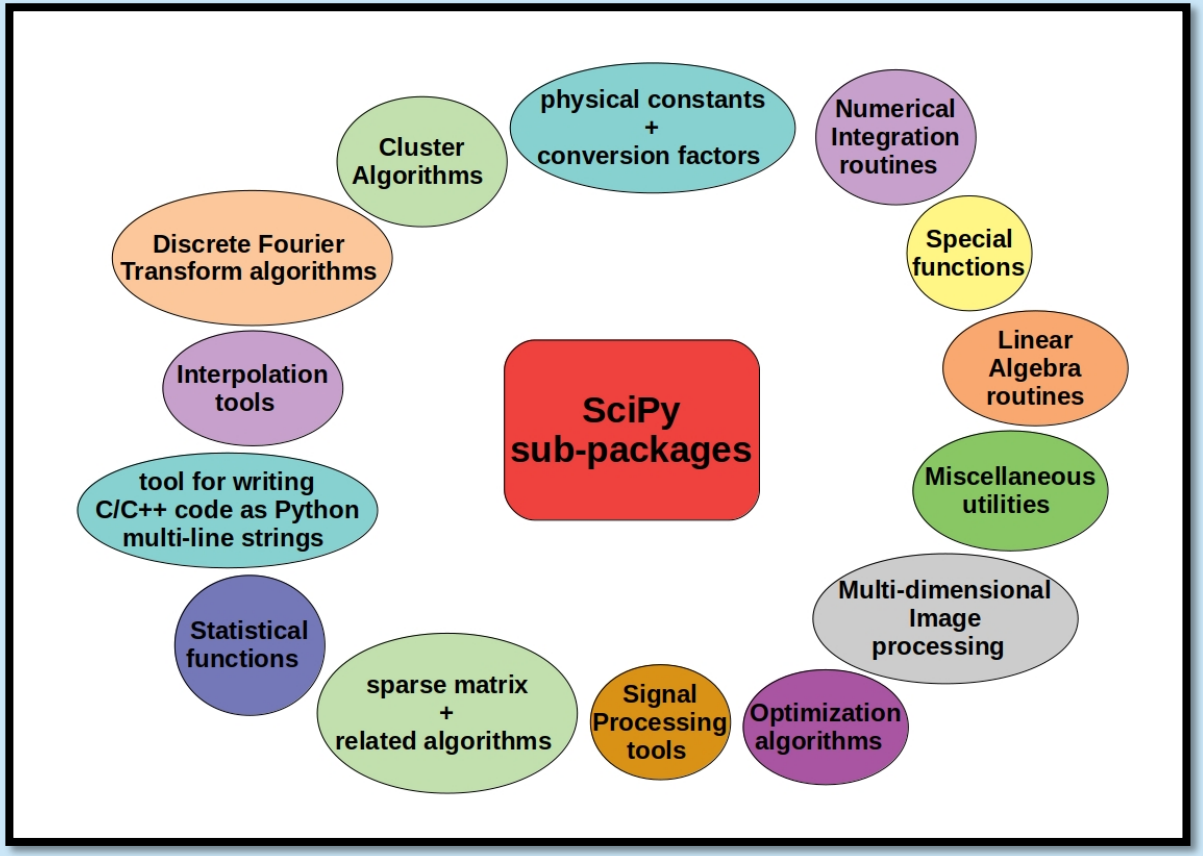
```
sudo apt-get install  python-scipy python-numpy
```

Install SciPy in Mac

```
sudo port install py35-scipy py35-numpy
```

§9 Python科学计算与可视化

- File input/output – scipy.io
- Special Function – scipy.special
- Linear Algebra Operation – scipy.linalg
- Interpolation – scipy.interpolate
- Optimization and fit – scipy.optimize
- Statistics and random numbers – scipy.stats
- Numerical Integration – scipy.integrate
- Fast Fourier transforms – scipy.fftpack
- Signal Processing – scipy.signal
- Image manipulation – scipy.ndimage



§9 Python科学计算与可视化

9.1 SciPy 科学与数值运算

9.1.1 SciPy 的安装与配置

9.1.2 SciPy 的子包

9.1.3 SciPy VS NumPy

模块名	功能
scipy.cluster	向量量化
scipy.constants	数学常量
scipy.fft	快速傅里叶变换
scipy.integrate	积分
scipy.interpolate	插值
scipy.io	数据输入输出
scipy.linalg	线性代数
scipy.misc	图像处理
scipy.ndimage	N 维图像
scipy.odr	正交距离回归
scipy.optimize	优化算法
scipy.signal	信号处理
scipy.sparse	稀疏矩阵
scipy.spatial	空间数据结构和算法
scipy.special	特殊数学函数
scipy.stats	统计函数

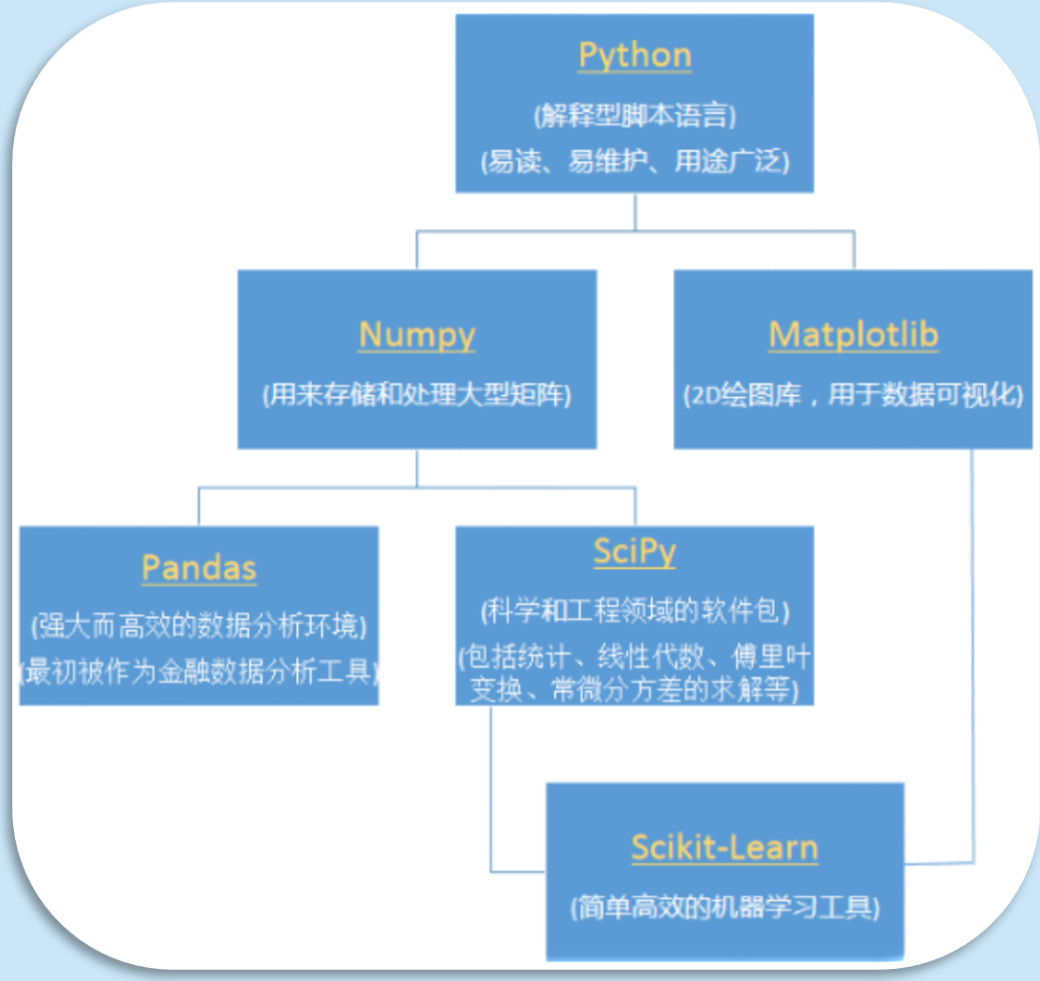
§9 Python科学计算与可视化

SciPy基于NumPy

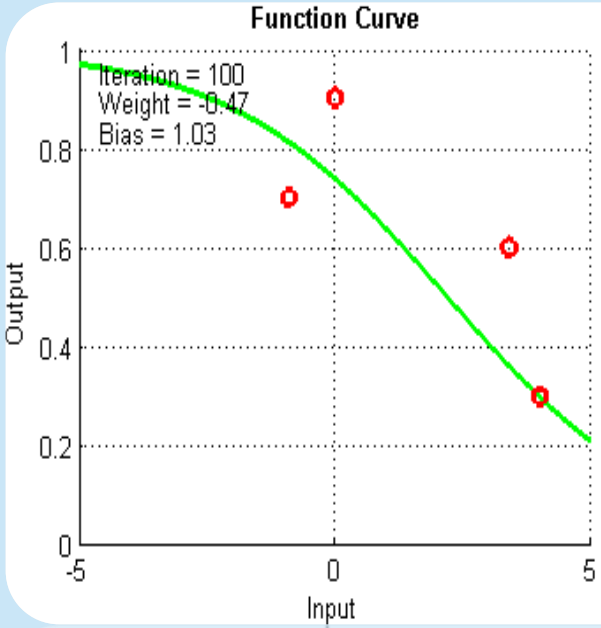
- 科学计算库
- 高阶抽象
- 物理模型

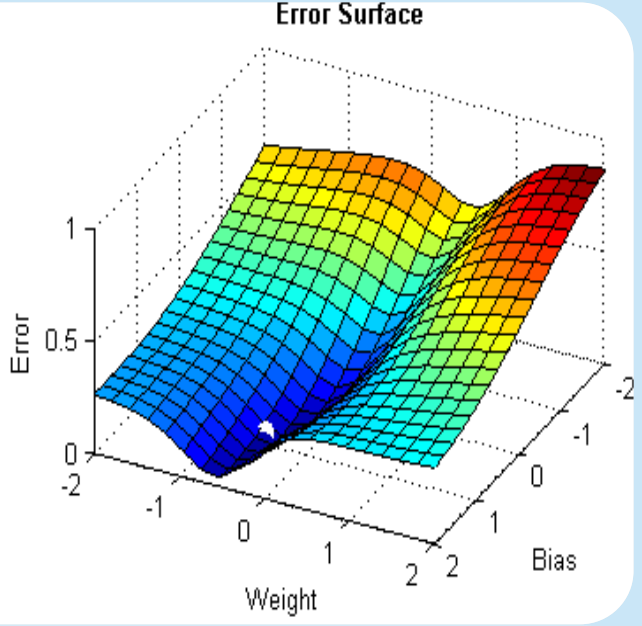
以傅立叶变换为例：

纯数学 NumPy
滤波器 SciPy。



§9 Python科学计算与可视化





- 拟合与优化 **optimize**

用SciPy库中的优化函数求解**非线性方程组**

- optimize.leastsq
- curve_fit

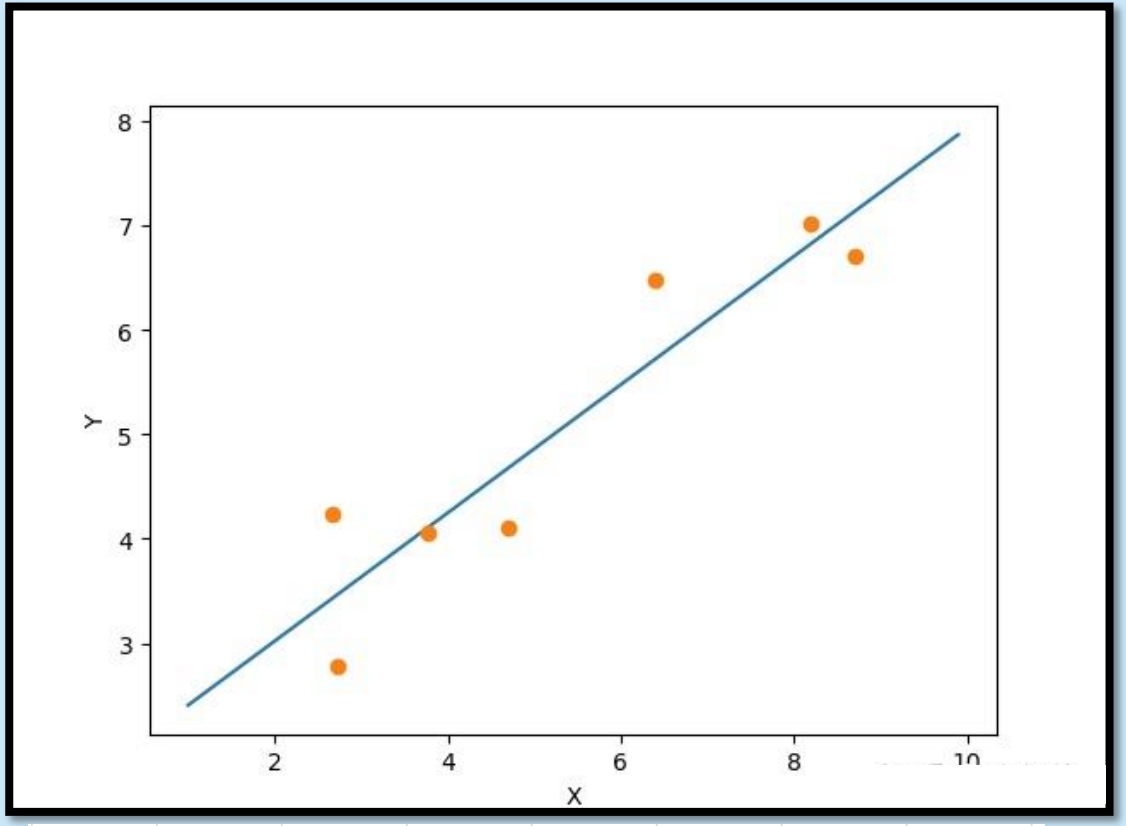
X	8.19	2.72	6.39	8.71	4.7	2.66	3.78
Y	7.01	2.78	6.47	6.71	4.1	4.23	4.05

最小二乘法：线性拟合

- 拟合与优化 **optimize**

用SciPy库中的优化函数求解**非线性方程组**

  - optimize.leastsq
  - curve_fit



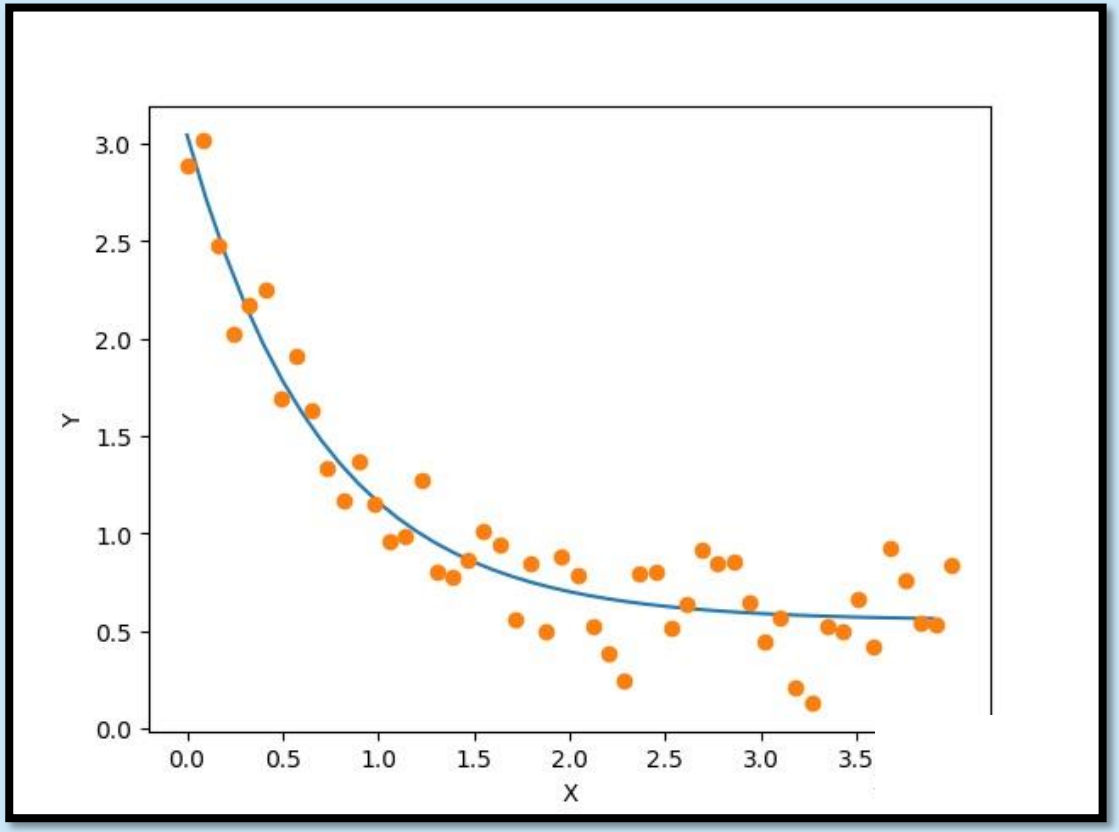
X	8.19	2.72	6.39	8.71	4.7	2.66	3.78
Y	7.01	2.78	6.47	6.71	4.1	4.23	4.05

最小二乘法拟合：指数函数

- 拟合与优化 **optimize**

用SciPy库中的优化函数求解**非线性方程组**

- optimize.leastsq
- curve_fit

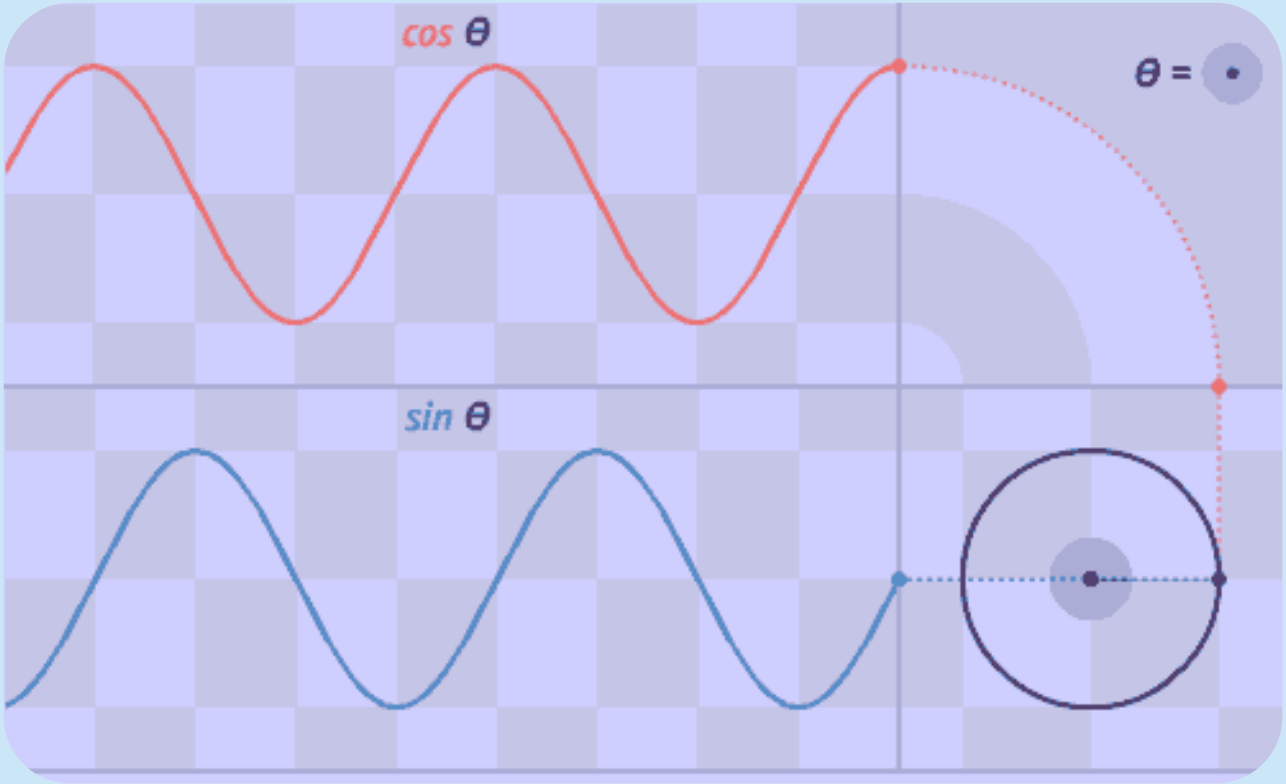


- 拟合与优化 **optimize**

$$5x_1 + 3 = 0$$
$$4x_0 - 2\sin(x_1 x_2) = 0$$
$$x_1 x_2 - 1.5 = 0$$

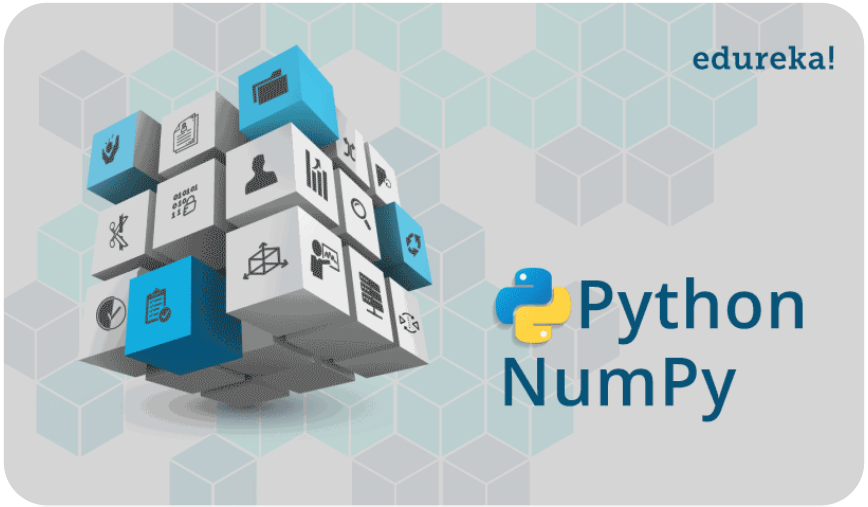
```
from math import sin, cos
from scipy import optimize
####定义非线性方程组
def f(x):
  ########tolist的作用是将list转化为浮点型的list
   x0, xl, x2 = x.tolist()
   return [
         5*xl+3,
         4*x0*x0 - 2*sin(xl*x2),
         xl*x2 - 1.5
         ]
# f 计算方程组的误差，[ 1 ,1 ,1 ]是未知数的初始值
result = optimize.fsolve(f, [1,1,1])
#######输出x0 x1 x2
print (result)
######检验求解的结果的正确性
print (f(result))
```

§9 Python科学计算与可视化



Python 语言程序设计

NumPy 数学函数与计算



NumPy

Module for Python

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project.

§9 Python科学计算与可视化

9.1 SciPy 科学与数值运算

9.2 NumPy数学函数与计算

- NumPy 的前身 Numeric 最早是由 Jim Hugunin 与其它协作者共同开发，2005 年，Travis Oliphant 在 Numeric 中结合了另一个同性质的程序库 Numarray 的特色，并加入了其它扩展而开发了 NumPy。NumPy 为开放源代码并且由许多协作者共同维护开发。

- NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。

NumPy 是一个运行速度非常快的数学库，主要用于数组计算，包含：

- 一个强大的N维数组对象 ndarray
- 广播功能函数
- 整合 C/C++/Fortran 代码的工具
- 线性代数、傅里叶变换、随机数生成等功能

9.2.1 Numpy数组

NumPy库中处理的最基础数据类型是同种元素构成的数组（ndarray）。NumPy数组是一个多维数组对象，称为ndarray。

NumPy数组的下标从0开始。
同一个NumPy数组中所有元素的类型必须是相同的。

9.2.1 Numpy数组

2．创建Numpy数组

可以使用array函数从常规的Python列表和元组创造数组。

```
>>> from numpy import *
>>> a = array( [2,3,4] )
```

可使用双重序列来表示二维的数组，三重序列表示三维数组，以此类推。

```
>>> b = array( [ (1.5,2,3), (4,5,6) ] )
>>> b
  array([[ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
```

9.2.1 Numpy数组

用函数zeros可创建一个全是0的数组，用函数ones可创建一个全为1的数组，函数empty创建一个内容随机的数组。

NumPy提供2个类似range的函数返回一个数列形式的数组。

（1）arange函数

```
>>>import numpy as np
>>>np.arange(0, 1, 0.1)     #步长0.1
array([ 0. ,0.1,0.2,          0.3,0.4,      0.5,0.6,      0.7,0.8,      0.9])
```

（2）linspace函数

通过指定开始值、终值和元素个数（默认为50）来创建一维数组，可以通过endpoint关键字指定是否包括终值，缺省设置是包括终值:

```
>>> np.linspace(0, 1, 5)
array([ 0. , 0.25, 0.5 , 0.75, 1. ])
```

9.2.1 Numpy数组

Numpy 库有一般 math 库函数的数组实现；如sin，cos，log。基本函数（三角、对数、平方和立方等）的使用就是在函数前加上np.这样就能实现数组的函数计算。

```
>>>x=np.arange(0,np.pi/2,0.1)
>>> x
array([0. ,0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ,1.1, 1.2, 1.3, 1.4, 1.5])
>>>y=np.sin(x)
array([ 0.          ,0.09983342,0.19866933,          0.29552021,0.38941834,
0.47942554,         0.56464247,0.64421769,          0.71735609,0.78332691,
0.84147098,         0.89120736,0.93203909,          0.96355819,0.98544973,
0.99749499])
```

9.2.1 Numpy数组

3．NumPy中的数据类型

名称	描述
bool	用一个字节存储的布尔类型（True 或 False）
inti	由所在平台决定其大小的整数（一般为 int32 或 int64）
int8	一个字节大小，-128 至 127
int16	整数，-32768 至 32767
int32	整数，-2^{31} 至 $2^{32}-1$
int64	整数，-2^{63} 至 $2^{63}-1$
uint8	无符号整数，0 至 255
uint16	无符号整数，0 至 65535
uint32	无符号整数，0 至 $2^{32}-1$
uint64	无符号整数，0 至 $2^{64}-1$
float16	半精度浮点数：16 位，正负号 1 位，指数 5 位，精度 10 位
float32	单精度浮点数：32 位，正负号 1 位，指数 8 位，精度 23 位
float64 或 float	双精度浮点数：64 位，正负号 1 位，指数 11 位，精度 52 位
complex64	复数，分别用两个 32 位浮点数表示实部和虚部
complex128 或 complex	复数，分别用两个 64 位浮点数表示实部和虚部

9.2.1 Numpy数组

4．Numpy数组中的元素访问

数组切片得到的是原始数组的视图，所有修改都会直接反映到源数组。如果需要得到Numpy数组（ndanay）切片的一份副本，需要进行复制操作，比如b[5:8].copy()。

访问	描述
X[i]	索引第 i 个元素
X[-i]	从后向前索引第 i 个元素
X[n:m]	切片，默认步长为 1，从前往后索引，不包含 m
X[-m,-n]	切片，默认步长为 1，从后往前索引，不包含 n
X[n,m,i]	切片，指定 i 步长的由 n 到 m 的索引

9.2.2 Numpy数组的算术运算

Numpy数组的算术运算是按元素逐个运算。Numpy数组运算后将创建包含运算结果的新数组。

```
>>>import numpy as np
>>> a= np.array([20,30,40,50])
>>> b= np.arange( 4)   #相当于np.arange(0, 4)
>>> b
输出：array([0, 1, 2, 3])
>>> c= a-b
>>> c
输出：array([20, 29, 38, 47])
```

9.2.2 Numpy数组的算术运算

与其他矩阵语言不同，NumPy中的乘法运算符*按元素逐个计算，矩阵乘法可以使用dot函数或创建矩阵对象实现。

```
>>>import numpy as np
>>> A= np.array([[1,1],  [0,1]])
>>> B= np.array([[2,0],  [3,4]])
>>> A*B                  # 逐个元素相乘
array([[2, 0],
    [0, 4]])
>>> np.dot(A,B)              # 矩阵相乘
array([[5, 4],
    [3, 4]])
```

9.2.3 Numpy数组的形状操作

1．数组的形状

数组的形状取决于其每个轴上的元素个数。

```
>>> a=np.int32(100*np.random.random((3,4)))  #3*4整数数组
>>> a.shape                    #(3, 4)
```

2．更改数组的形状

可以用多种方式修改数组的形状：

reshape() shape()、resize()和ravel()

```
>>> a.ravel()                  # 平坦化数组
array([26, 11,  0, 41, 48,  9, 93, 38, 73, 55,  8, 81])
>>> a.resize((2,6))
```

9.2.4 文件存取数组内容

NumPy提供了多种文件操作函数方便我们存取数组内容。
使用数组的方法函数tofile可以方便地将数组中数据以二进制的格式写进文件。tofile输出的数据没有格式，因此用numpy.fromfile读回来的时候需要自己格式化数据：

```
>>> a = np.arange(0,12)
>>> a.shape = 3,4    # 改成3*4数组
>>> a
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>> a.tofile("a.bin")
>>> b = np.fromfile("a.bin", dtype=np.int32) # 按照int32类型读入数据
>>> b # 数据是一维的
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
>>> b.shape = 3, 4 # 按照a的shape修改b的shape
```

9.2.5 Numpy中图像数组

1．获取图像数组

当载入图像时，我们通过调用array()方法将图像转换成NumPy的数组对象。

对于图像数据，下面的例子阐述了这一点：

```
im = array(Image.open('d:\\test.jpg'))          #彩色图像
print (im.shape, im.dtype)
im = array(Image.open('d:\\test.jpg').convert('L'),'f')  #灰度化处理
print (im.shape, im.dtype)
```

输出结果如下所示：

```
(800, 569, 3) uint8
(800, 569) float32
```

2．利用图像数组进行灰度变换

将图像读入NumPy 数组对象后，我们可以对它们执行任意数学操作。

```
from PIL import Image
from numpy import *
im = array(Image.open('empire.jpg').convert('L'))
im2 = 255 - im                    # 对图像进行反相处理
im3 = (100.0/255) * im + 100      # 将图像像素值变换到
100...200 区间
im4 = 255.0 * (im/255.0)**2       # 对图像像素值求平方后得
到的图像
```

3．图像缩放

NumPy的数组对象是我们处理图像和数据的主要工具。想要对图像进行缩放处理没有现成简单的方法。我们可以使用之前PIL对图像对象转换的操作，写一个简单的用于图像缩放的函数。

```
def imresize(im,sz):
        """ 使用PIL 对象重新定义图像数组的大小
"""
        pil_im = Image.fromarray(uint8(im))
        return array(pil_im.resize(sz))
```

4．直方图均衡化

直方图均衡化是指将一幅图像的灰度直方图变平，使变换后的图像中每个灰度值的分布概率都相同。在对图像做进一步处理之前，直方图均衡化通常是对图像灰度值进行归一化的一个非常好的方法，并且可以增强图像的对比度。

9.3 Matplotlib绘图可视化

Matplotlib实际上是一套面向对象的绘图库，它所绘制的图表中的每个绘图元素，例如线条Line2D、文字Text、刻度等都有一个对象与之对应。

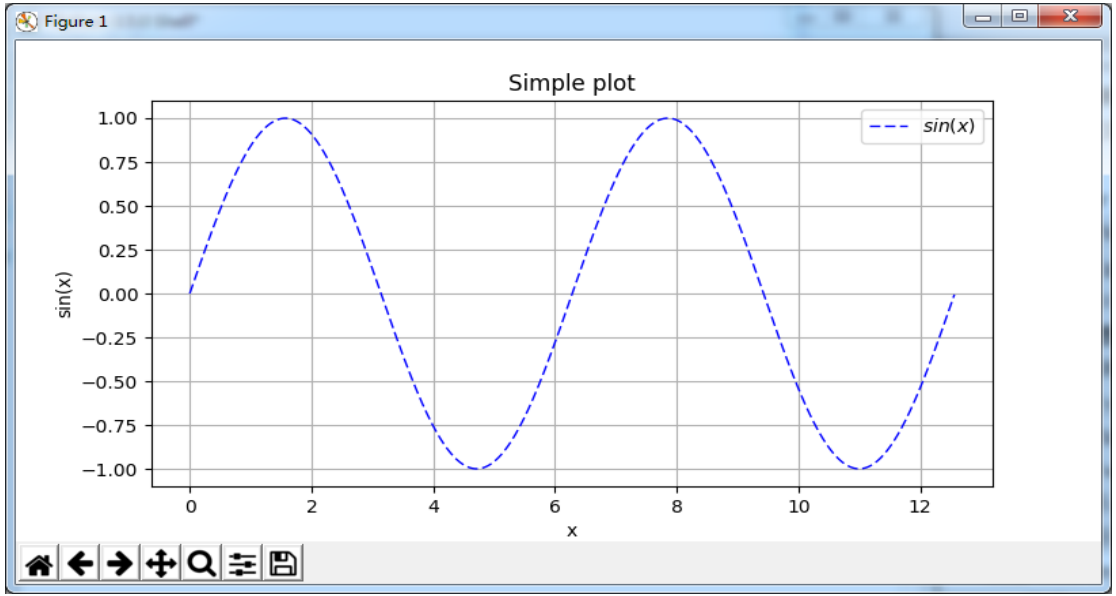
为了方便快速绘图，Matplotlib通过pyplot模块提供了一套和MATLAB类似的绘图API，将众多绘图对象所构成的复杂结构隐藏在这套API内部。我们只需要调用pyplot模块所提供的函数就可以实现快速绘图以及设置图表的各种细节。

9.3.1 Matplotlib.pyplot模块

•简单的绘制正弦三角函数y=sin(x)的例子

```
# plot a sine wave from 0 to 4pi
  import matplotlib.pyplot as plt
  from numpy import *                    #也可以使用 from pylab import *
  plt.figure(figsize=(8,4))              #创建一个绘图对象，大小为800*400
  x_values = arange(0.0, math.pi * 4, 0.01)    #步长0.01，初始值0.0，终值4π
  y_values = sin(x_values)
  plt.plot(x_values, y_values, 'b--', linewidth=1.0, label('$sin(x)$') #进行绘图
  plt. xlabel('x ')                      #设置X轴的文字
  plt. ylabel('sin(x)')                  #设置Y轴的文字
  plt.ylim(-1, 1)                        #设置Y轴的范围
  plt. title('Simple plot')              #设置图表的标题
  plt.legend()                           #显示图例(legend)
  plt.grid(True)
  plt.savefig("sin.png")
  plt.show()
```

9.3.1 Matplotlib.pyplot模块



9.3.1 Matplotlib.pyplot模块

1.调用figure创建一个绘图对象
2.通过调用plot函数在当前的绘图对象中进行绘图
3.设置绘图对象的各个属性
xlabel、ylabel：分别设置X、Y轴的标题文字。
title：设置图的标题。
xlim、ylim：分别设置X、Y轴的显示范围。
legend()：显示图例，即图中表示每条曲线的标签(label)和样式的矩形区域。
4．清空plt绘制的内容

```
plt.cla()                    #清空plt绘制的内容
plt.close(0)                  # 关闭0号图
plt.close('all')               # 关闭所有图
```

9.3.1 Matplotlib.pyplot模块

5．图形保存和输出设置

plt.savefig("test.png",dpi=120)

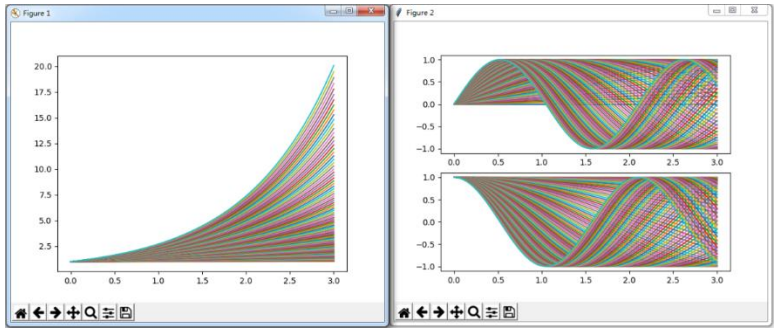
6. 绘制多子图

可以使用subplot()快速绘制包含多个子图的图表，它的调用形式如下：

subplot(numRows, numCols, plotNum)

7．调节轴之间的间距和轴与边框之间的距离

8．绘制多幅图表



9.3.2 绘制条形图、饼状图、散点图等

matplotlib是一个Python的绘图库，使用其绘制出来的图形效果和MATLAB下绘制的图形类似。pyplot模块提供了14个用于绘制“基础图表”的常用函数。

表 14-5 plt 库中绘制基础图表函数

函数	功能
plt.polt(x, y, label, color, width)	根据 x、y 数组绘制点、直线或曲线
plt.boxplot(data, notch, position)	绘制一个箱型图(Box-plot)
plt.bar(left, height, width, bottom)	绘制一个条形图
plt.barh(bottom, width, height, left)	绘制一个横向条形图
plt.polar(theta, r)	绘制极坐标图
plt.pie(data,explode)	绘制饼图
plt.psd(x, NFFT=256, pad_to, Fs)	绘制功率谱密度图
plt.specgram(x, NFFT=256, pad_to, F)	绘制谱图
plt.cohere (x, y, NFFT=256, Fs)	绘制 X-Y 的相关性函数
plt.scatter()	绘制散点图(x、y 是长度相同的序列)
plt.step(x, y, where)	绘制步阶医
plt.hist(x, bins, normed),	绘制直方图
plt.contour(X, Y, Z, N)	绘制等值线
pit.vlines()	绘制垂直线
plt.stem(x, y, linefmt, markerfmt, basefmt)	绘制曲线每个点到水平轴线的垂线
plt.plot_date()	绘制数据日期
plt.plothle()	绘制数据后写入文件

9.3.2 绘制条形图、饼状图、散点图等

plt库提供了3个区域填充函数，对绘图区域填充颜色

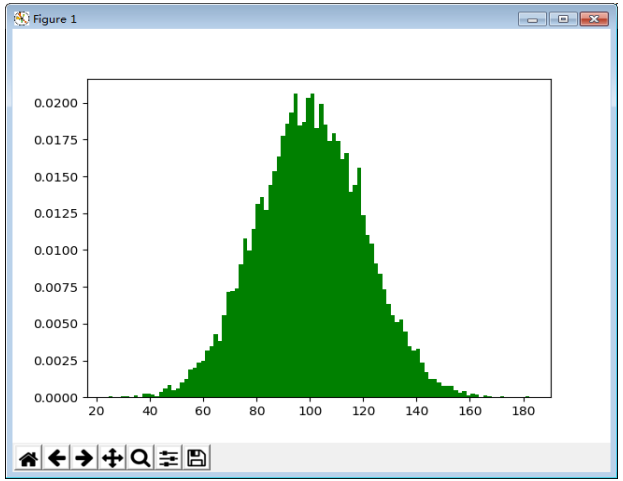
表 14-6 plt 库的区域填充函数

函数	功能
fill(x,y,c,color)	填充多边形
fill_between(x,y1,y2,where,color)	填充两条曲线围成的多边形
fill_betweenx(y,x1,x2,where,hold)	填充两条水平线之间的区域

1．直方图（histograms）

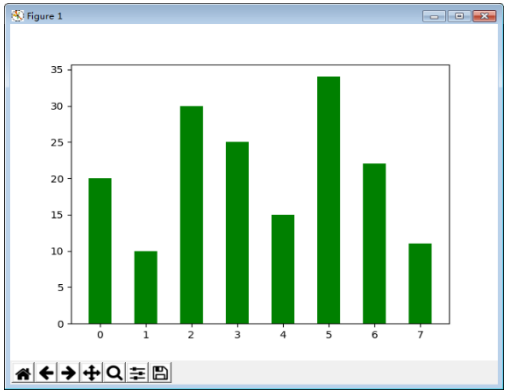
直方图(histogram)又称质量分布图。是一种统计报告图，由一系列高度不等的纵向条纹或线段表示数据分布的情况。 一般用横轴表示数据类型，纵轴表示分布情况。直方图的绘制通过pyplot中的hist()来实现。

pyplot.hist(x, bins=10, color=None, range=None, rwidth=None, normed=False, orientation=u'vertical', **kwargs)

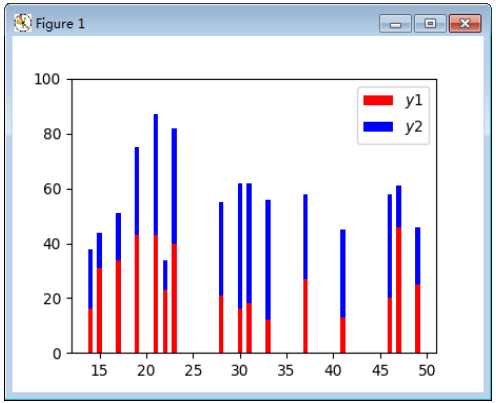


2．条形图（bar）

条形统计图是用一个单位长度表示一定的数量，根据数量的多少画成长短不同的直条，然后把这些直条按一定的顺序排列起来。从条形统计图中很容易看出各种数量的多少。条形图的绘制通过pyplot中的bar()或者是barh()来实现。bar默认是绘制竖直方向的条形图，也可以通过设置 orientation = "horizontal" 参数来绘制水平方向的。barh()就是绘制水平方向的条形图。



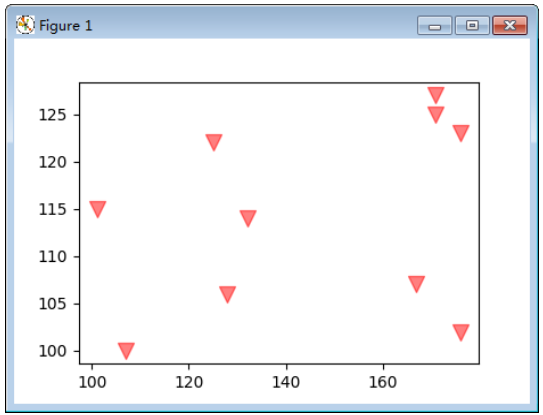
•条形图实例



•层叠的条形图实例

3．散点图（scatter）

散点图（scatter diagram），在回归分析中是数据点在直角坐标系平面上的分布图。一般用两组数据构成多个坐标点，考察坐标点的分布，判断两变量之间是否存在某种关联或总结坐标点的分布模式。使用pyplot中的scatter ()绘制散点图。



4．饼状图（Pie Graph）

饼状图（Sector Graph又名Pie Graph）显示一个数据系列中各项的大小与各项总和的比例，饼状图中的数据点显示为整个饼状图的百分比。使用pyplot中的pie()绘制饼状图。

9.3.3 交互式标注

有时用户需要和某些应用交互，例如在一幅图像中标记一些点，或者标注一些训练数据。matplotlib.pyplot库中的ginput() 函数就可以实现交互式标注。

```
#交互式标注
from PIL import Image
from numpy import *
import matplotlib.pyplot as plt
im = array(Image.open('d:\\test.jpg'))
plt.imshow(im)
print ('Please click 3 points')
x = plt.ginput(3)
print ('you clicked:',x  )
plt.show()
```

•等待用户在绘图窗口的图像区域点击三次。程序将这些点击的坐标[x, y]自动保存在x列表里。