数据科学之路：02-2 Jupyter Notebook和Numpy

Python 中的科学计算工具

课程简介

用 Python 进行科学计算时有很多常用的工具，比如Jupyter Notebooks、 Numpy、 Scipy、 pandas、 Matplotlib和sklearn等，本节对其中将要用到的部分进行介绍，为之后学习作准备。

学习目标

- 学习 Jupyter Notebooks 的基础使用方式
- 了解 Numpy 数组的创建和基础操作
- 了解 Scipy 稀疏矩阵的创建和格式转换
- 了解 pandas对数据进行处理的基本方法
- 了解 matplotlib 基本绘图操作
- 了解 scikit—learn基本操作，学会应用机器学习算法解决实际问题

Jupyter Notebooks

Jupyter Notebook（此前被称为 IPython notebook）是一个交互式笔记本，支持运行 40 多种编程语言。在本文中，我们将介绍 Jupyter notebook 的主要特性，以及为什么对于希望编写漂亮的交互式文档的人来说是一个强大工具。

优点

- 基于过程
- Python
- Hackable:计算+文档的混合体

缺点

- 计算语法过于复杂

访问文档

```
# help()方法
help(len)

# 利用?
len?

# L = [1,2,3]

# L.insert?

# L?
```

```
def square(a):
    """Return the square of a."""
    return a ** 2

square?

# square??
```

```
# TAB键
L.

# L.app

# L._

# import

# from numpy import ar
```

```
# *
*Warning?
```

```
# str.*find*?
```

快捷键

见Help下的Keyboard Shortcuts

魔术指令

- %run 运行py文件
- %timeit 计算代码运行时间
- %%timeit 计算单元格运行时间
- ……

%run

```
# file:myscript.py
def square(x):
    """square a number"""
    return x ** 2

for N in range(1, 4):
    print(N, "squared is", square(N))
```

```
%run myscript.py
```

%timeit

```
%time
```

%%timeit

```
%%timeit
L = []
```

```
for n in range(1000):
    L.append(n ** 2)
```

```
%timeit?
```

In和Out

- In:输出当前回话前所有输入
- Out:输出当前回话前所有输出

```
import math
```

```
math.sin(2)
```

```
math.pi
```

```
print(In)

# print(Out)
```

Shell命令

调用

```
!ls
```

```
!pwd
```

```
!echo "printing from the shell"
```

赋值

```
contents = !ls
print(contents)
```

```
directory = !pwd
print(directory)
```

```
ls
```

扩展

入门 | 数据科学家效率提升必备技巧之Jupyter Notebook篇

值得推荐的五个Jupyter Notebook扩展

Numpy 数组

NumPy(Numeric Python)系统是Python的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比Python自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix））。

```
import numpy as np
np.__version__
```

```
np?
```

数组

```
import numpy as np

# 创建数组
np.array([1, 4, 2, 5, 3])
```

```
# 多维数组
np.array([range(i, i + 3) for i in [2, 4, 6]])
```

常用数组

```
# 全零
np.zeros(10, dtype=int)
# 全一
np.ones((3, 5), dtype=float)

# 单位
np.eye(3)

# 固定
np.full((3, 5), 3.14)

# 线性序列
np.arange(0, 20, 2)

# 均匀间隔
np.linspace(0, 1, 5)

# 0到1随机
np.random.random((3, 3))

# 标准正太分布随机
np.random.normal(0, 1, (3, 3))

# 随机整数
np.random.randint(0, 10, (3, 3))
```

数组基础

```
import numpy as np
np.random.seed(0)

x1 = np.random.randint(10, size=6)  # 一唯数组
```

```
x2 = np.random.randint(10, size=(3, 4))  # 二维数组
x3 = np.random.randint(10, size=(3, 4, 5))  # 三维数组
```

```
print("x3 ndim: ", x3.ndim)  # 维数
print("x3 shape:", x3.shape) # 维度
print("x3 size: ", x3.size)  # 元素个数
print("dtype:", x3.dtype)    # 数据类型
print("itemsize:", x3.itemsize, "bytes")  # 元素大小(字节)
print("nbytes:", x3.nbytes, "bytes")      # 数组大小(字节)
```

索引

```
x1[1]
```

```
x1[0] = 3.12
```

切片

格式：x[开始：结束：步长]

```
x = np.arange(10)

x[5:]

# x[4:7]

# x[::2]

# x[1::2]

# x[::-1]

# x[5::-2]
```

多维子序列

```
x2

x2[:2, :3]  #两行三列

x2[:3, ::2] #三行、隔一列

x2[::-1, ::-1]
```

访问数组

```
print(x2[:, 0])

print(x2[0, :])

print(x2[0])
```

复制数组

```
# 改变子数组，会改变数组
print(x2)

x2_sub = x2[:2, :2]
print(x2_sub)

x2_sub[0, 0] = 99
print(x2_sub)

print(x2)
```

```
# 通过np.copy()或.copy()方法复制
x2 = np.random.randint(10, size=(3, 4))
print(x2)

x2_sub = np.copy(x2[:2, :2])
# x2_sub_copy = x2[:2, :2].copy()
print(x2_sub)
```

```
x2_sub[0, 0] = 99
print(x2_sub)

print(x2)
```

数组链接

```
# np.concatenate()
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])

grid = np.array([[1, 2, 3],
                 [4, 5, 6]])
np.concatenate([grid, grid])

np.concatenate([grid, grid], axis=1)
```

```
# np.vstack()垂直链接
x = np.array([1, 2, 3])
grid = np.array([[9, 8, 7],
                 [6, 5, 4]])
np.vstack([x, grid])
```

```
# np.hstack()水平链接
y = np.array([[99],
              [99]])
np.hstack([grid, y])
```

数组分割

```
# np.split()
x = [1, 2, 3, 99, 99, 3, 2, 1,0]
# print(np.split(x, 3))
```

```
x1, x2, x3 = np.split(x, [1,5])
print(x1,x2,x3)
```

```
# np.vsplit()垂直分割
grid = np.arange(16).reshape((4, 4))
print(grid)

upper, lower = np.vsplit(grid, [2])
print(upper)
print(lower)
```

```
# np.hsplit()水平分割
left, right = np.hsplit(grid, [2])
print(left)
print(right)
```

数组运算

```
import numpy as np
np.random.seed(0)

def compute_reciprocals(values):
    output = np.empty(len(values))
    for i in range(len(values)):
        output[i] = 1.0 / values[i]
    return output

values = np.random.randint(1, 10, size=5)
compute_reciprocals(values)
```

统计

- np.sum():求和
- np.min():最小值
- np.max():最大值
- np.mean():均值

```
# 对数组进行操作
import numpy as np
x = np.array([[5, 10, 15],
              [20, 25, 30],
              [35, 40, 45]])

x.sum()    # 输出'225'
x.max()    # 输出'45'
x.min()    # 输出'5'
x.mean()   # 输出'25.0'

# 对行列进行操作
x.sum(axis = 1)   #输出‘array([ 30,  75, 120])’

x.sum(axis = 0)   #输出'array([60, 75, 90])'
```

矩阵运算

```
import numpy as np
a = np.array([[1, 2],
              [3, 4]])

b = np.array([[5, 6],
              [7, 8]])

# 对应位置元素相乘
print(a*b)
print('---------------')
# 矩阵乘法
print (a.dot(b))
print('---------------')
# 矩阵乘法，同上
print (np.dot(a, b))
print('---------------')
# 矩阵转置
c = np.array([[1,2,3,4],[5,6,7,8]])
print(c)
print('---------------')
print(c.T)
```

广播

```
import numpy as np
a = np.array([0, 1, 2])
b = np.array([5, 5, 5])
a + b
```

```
a = np.arange(3)
b = np.arange(3)[:, np.newaxis]

print(a)
print(b)
```

```
a + b
```

排序

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

# x.sort()
# print(x)
```

```
# np.argsort()返回索引
x = np.array([2, 1, 4, 3, 5])
i = np.argsort(x)
print(i)

x[i]
```

按行或列排序

```
rand = np.random.RandomState(42) # 伪随机数
X = rand.randint(0, 10, (4, 6))
print(X)

# 按列排序
np.sort(X, axis=0)

# # 按行排序
# np.sort(X, axis=1)
```

源代码