《Python与数据挖掘》

第5章 Python实用模块

讲师：武永亮

目录

模块的作用

- 从这一章开始，我们开始讨论Python模块并详细介绍多个Python必须掌握的模块。通过前面章节的学习，理论上我们已经可以几乎能够使用Python做任何事情。
- 但是如果让你们写一段实现矩阵的工业代码，我想这也是一件不容易的事情，而这段代码已经有人完整地写好了，我们可以在别人的基础上进行深一层的研究，这就需要模块。
- 通过引入模块，我们能够调用别人写好的函数和类，我们没有必要重新做别人已经做好的东西。Python的模块使得程序员能够轻松地分享自己的成果，这也是Python这门开源语言的一个亮点。

模块的含义

- 模块是最高级别的程序组织单元，它能够将程序代码和数据封装以便重用。
- 模块往往对应了Python的脚本文件（.py），包含了所有编写该模块的程序员定义的函数和变量。
- 模块可以被别的程序导入，以使用该模块的函数等功能，这也是使用Python标准库的方法。
- 导入模块后，在模块文件定义的所有变量名都会以被导入的模块对象的成员的形式被调用。换言之，模块文件的全局作用域变成了模块对象的局部作用域。

模块的导入

- 因此模块能够划分系统命名空间，避免了不同文件的变量重名的问题。Python的模块使得独立的文件连接成了一个巨大的程序系统。
- 模块的导入是通过import语句,下面是三种import语句的格式

a) import numpy : 直接导入NumPy模块

b) import numpy as np: 导入NumPy模块后并将其改名为np

c) from numpy import array: 从NumPy模块中导入其中的array方法

目录


1
模块的概述
2
NumPy
3
Pandas
4
SciPy
5
Scikit-learn
6
其他Python常用模块

NumPy的含义

- NumPy是一个Python科学计算的基础模块。
- NumPy不但能够完成科学计算的任务，也能够被用作有效的多维数据容器，可用于存储和处理大型矩阵。
- NumPy的数据容器能够保存任意类型的数据，这使得NumPy可以无缝并快速地整合各种数据。
- 在性能上NumPy比起Python自身的嵌套列表结构要高效得多。
- Python在科学计算的其他模块大多数都是在NumPy的基础上编写的。

创建数组

- NumPy有多种方法去创建数组，例如通过元组和列表。

```
import NumPy as np   #导入模块
print '''创建数组'''
arr1 = np.array([2,3,4])  # 通过列表创建数组
arr2 = np.array([(1.3,9,2.0),(7,6,1)]) # 通过元组创建数组
arr3 = np.zeros((2,3))  # 通过元组(2,3)生成零矩阵(矩阵也是数组的一种)
arr4 = np.identity(3)    # 生成3维的单位矩阵
arr5 = np.random.random(size = (2,3)) # 生成每个元素都在[0,1]之间
                                      的随机矩阵
print arr1
```

访问数组

- 创建数组后，NumPy有很多方法接口去访问数组的属性。在科学计算时，我们需要频繁访问数组元素，通过NumPy索引,切片和迭代器方法能够快速灵活地访问数组。

```
def f(x,y):# 通过索引和切片访问数组元素
    return 10*x+y
arr8 = np.fromfunction(f,(4,3),dtype = int)
print arr8
for row in arr8:# 通过迭代器访问数组元素
    print row
for element in arr8.flat:# 输出矩阵全部元素
    print element
```

数组的运算

- NumPy的运算是相当方便高效的，其运算符都是针对整个数组，比起使用for循环，使用NumPy的运算方法速度上要优秀得多。如果NumPy数组是一个矩阵，还支持矩阵求逆，转置等操作。

```
print '''数组的运算'''
arr9 = np.array([[2,1],[1,2]])
arr10 = np.array([[1,2],[3,4]])
print arr9 - arr10
print arr9**2
print arr9*arr10  #普通乘法
print np.dot(arr9,arr10)  #矩阵乘法
print arr10.T  #转置
print np.linalg.inv(arr10) #返回逆矩阵
print arr10.sum()  #数组元素求和
print arr10.max()  #返回数组最大元素
print arr10.cumsum(axis = 1)  #沿行累计总和
```

NumPy通用函数

- 许多数学上的函数，如sin,cos等在NumPy都有重新的实现。在NumPy中，这些函数称为通用函数（universal functions)。通用函数是针对整个NumPy数组的，因此我们不需要对数组的每一个元素都进行一次操作，它们都是以NumPy数组作为输出。

```
    print '''NumPy通用函数'''
#指数函数
    print np.exp(arr9)
#正弦函数（弧度制）
     print np.sin(arr9)
#和arr9+arr10效果一样
     print np.add(arr9,arr10)
#开方函数
     print np.sqrt(arr9)
```

数组的合并和分割

- 下面介绍如何通过方法接口对数组进行合并和分割：

```
print '''数组合并与分割' ''  # 合并
```

- #纵向合并数组，由于与堆栈类似，故命名为vstack

```
arr11 = np.vstack((arr9,arr10)) print arr11
```

- #横向合并数组

```
arr12 = np.hstack((arr9,arr10))
print arr12
```

- # 数组纵向分为2部分

```
print np.vsplit(arr11,2)
```

- # 将数组横向分为2部分

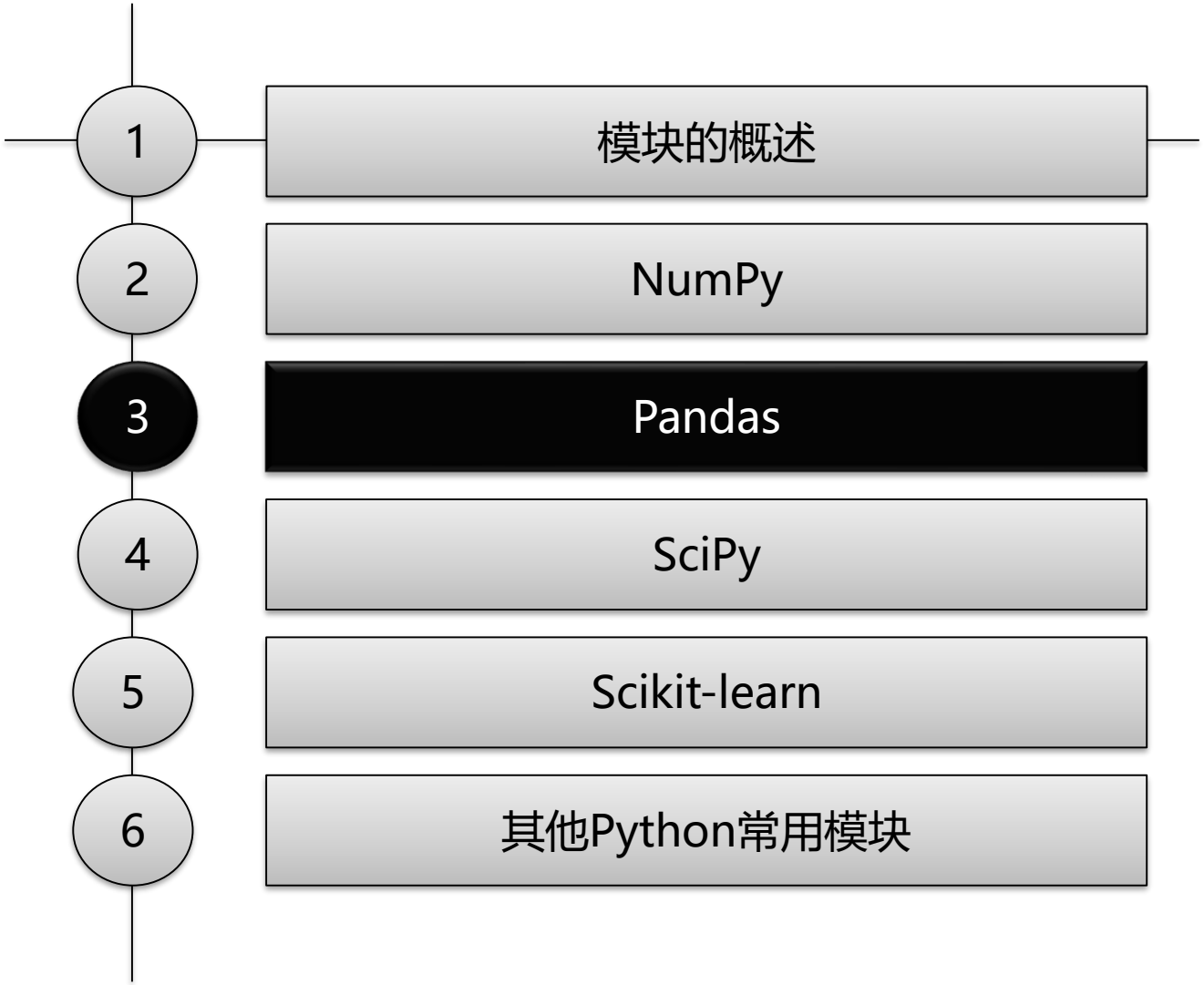
```
print np.hsplit(arr12,2)
```

其他NumPy常用方法

- 上面未能将NumPy的所有方法逐一介绍，以下附一张NumPy上面未涉及但却常用的方法清单，了解NumPy的功能。

方法	效果或用途	返回类型
np.empty	返回一个给定规模的数组	NumPy数组
np.all	测试数组元素是否均为True	True或False
np.any	测试数组元素是否有至少一个为True	True或Fasle
np.average	计算加权平均值	NumPy数组
np.nonzero	返回数组非0元素的位置	记录位置元组
np.sort	对数组元素进行排序	NumPy数组
np.var	计算方差	NumPy数组
np.where	返回数组满足条件的元素	NumPy数组
np.reshape	转换数组的规模但不更改其中的数据	NumPy数组
np.reshape	转换数组的规模	NumPy数组
np.eye	生成单位矩阵	NumPy数组
np.transpose	矩阵转置，与.T效果相同	NumPy数组
np.std	计算标准差	NumPy数组
np.cov	给定数据和权重计算协方差矩阵	NumPy数组

目录


1
模块的概述
2
NumPy
3
Pandas
4
SciPy
5
Scikit-learn
6
其他Python常用模块

Pandas

- Pandas模块是一个强大的数据分析和处理工具。它提供快速、灵活、富有表现力的数据结构，能为复杂情形下的数据提供坚实的基础分析功能。所谓复杂情形，可能有以下的三种：
  a) 数据库表或EXCEL表，包含了多列不同数据类型的数据（如数字，文字）
  b) 时间序列类型的数据，包括有序和无序的情形，甚至是频率不固定的情形
  c) 任意的矩阵型/二维表/观测统计数据，允许独立的行或列带有标签
- 对于数据科学家，和数据打交道的流程可以分为几个阶段：清洗数据、分析和建模、组织分析的结果并以图表的形式展示出来。
- 官方提倡的模块导入语法为：import pandas as pd

Pandas中的高级数据结构

- 为了开始使用Pandas，你需要熟悉两个重要的数据结构： 系列（Series）和数据框（DataFrame）。
- 有了它们，你可以利用Pandas在计算机内存中构建一个虚拟的数据库。

数据框（DataFrame）

- 我们首先介绍数据框，它的结构与矩阵神似，但与矩阵不同。数据框中每列表示一个变量，每行则是一个观测，行列交汇的某个单元格，对应该变量的某次具体的观测值，如下图：
- 数据框有行和列的索引（index），能让你快速地按索引访问数据框的某几行或某几列，在DataFrame里的面向行和面向列的操作大致是对称的。

	age	cash	id
0	18	10.53	Jack
1	35	500.70	Sarah
2	20	13.60	Mike

创建数据框

- 有很多方法来创建一个数据框，但最常用的是用一个包含相等长度列表的字典或NumPy数组。需要注意的是：数据框创建时会根据内置的多种规则对数据进行排序，导致结果的行列位置可能不一样，但数据的对应关系不会出现任何错位。

```
import pandas as pd    # 为pandas取一个别名pd
data = {'id': ['Jack', 'Sarah', 'Mike'],
   'age': [18, 35, 20],
   'cash': [10.53, 500.7, 13.6]}
df = pd.DataFrame（data)    # 调用构造函数并将结果赋值给df
print df
```

创建数据框

- 上述代码的输出可以观察到：由于没有显式声明，行索引自动分配，并且对列名（列索引）进行了排序。
- 而下面的例子应用了pd.DataFrame()中更高级的参数设置，显式地声明了列名排序方式和行索引。

```
df2 = pd.DataFrame(data, columns=['id', 'age',  'cash'],index=['one', 'two', 'three'])
print df2
```

系列（Series）

- 系列是对同一个属性进行多次观测之后得到的一列结果。
- 用统计学的语言说，它们服从某种分布。我们可以认为，系列是一种退化的数据框，也可以认为它是一种广义的一维数组。
- 在默认情况下，系列的索引是自增的非负整数列（0,1,2,3,...）。
- 值得注意的是，同个系列的数据共享一个列名，而数组不要求。在时间序列（Time Series）的相关问题中，系列（Series）这一数据结构有宝贵的价值。
- 创建序列：

```
s = pd.Series({'a': 4, 'b': 9, 'c': 16}, name='number')
print s
```

基础数据处理方法

- 系列可以被认为是数据框的一个子集。因此，应首先关注系列的基础操作。
- 按下标访问（call-by-index）：

```
print s[:3]
print s[0]
```

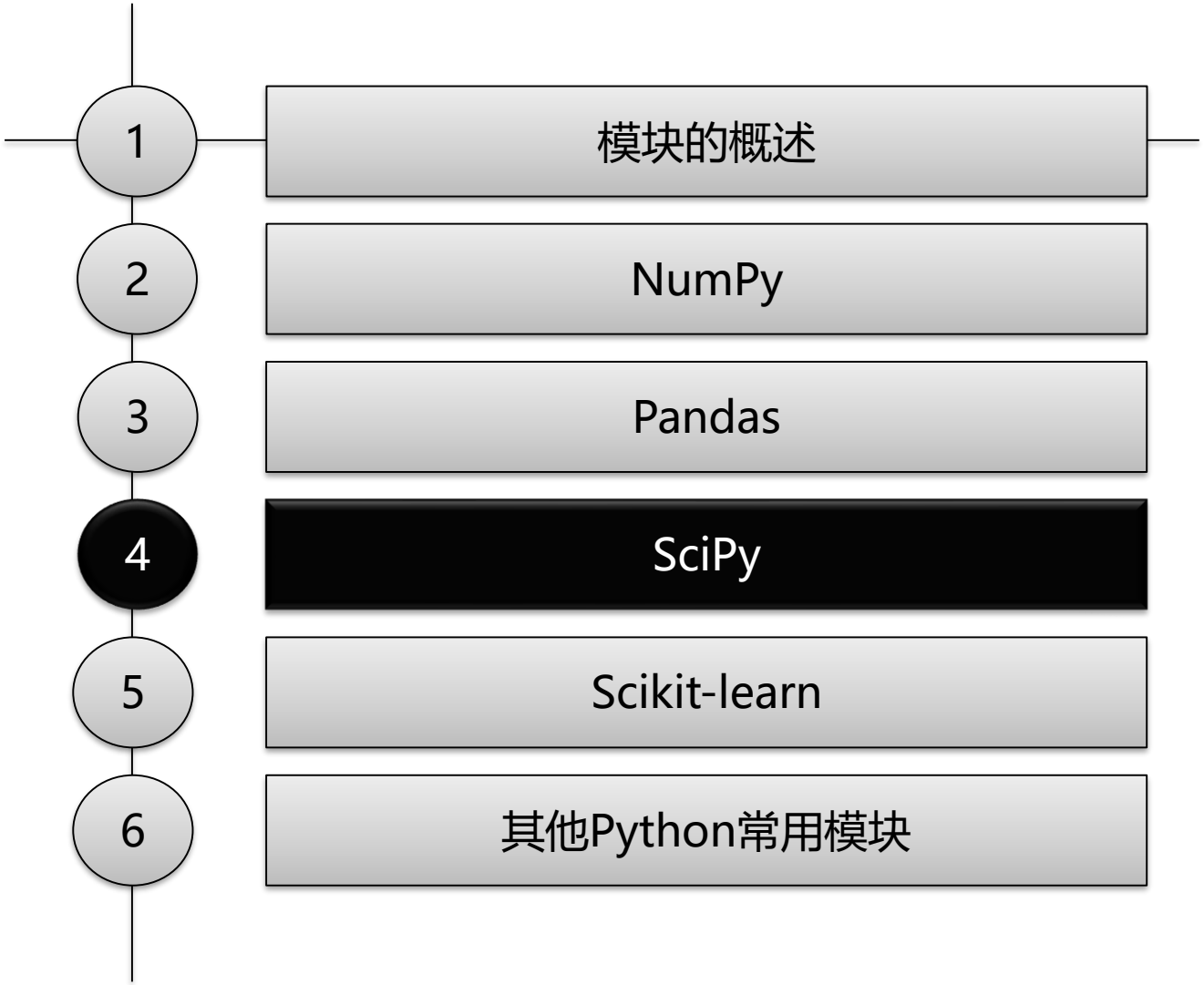
- 类似于数组，系列支持按索引访问内容。更有趣的是，系列还支持类似字典的访问方式——按键值（列名）访问。
- 同时，作为一种高级数据结构，系列同样支持向量化操作。也就是说，我们能够同时对一个系列的所有取值执行同样的操作，一致地应用某种方法。

Pandas常用方法

- 前面介绍基本的概念和用法。以下附一张Pandas常用方法清单，掌握利用Pandas进行数据分析和处理的基本要领。

方法名称	效果或用途	返回类型
pd.read_csv()	将.csv文件中的数据读入内存，快速构建数据框	数据框
pd.concat()	按横向或纵向方向合并两个Pandas数据结构	系列或数据框
pd.get_dummies()	将类别变量转变为哑变量（One-hot Encoding）	数据框
Series.isnull()	判断某个系列中是否含有空值	同维的0-1系列
Series.is_unique	判断某个系列中的所有值是否存在重复	布尔值
Series.value_counts()	统计某个系列中所有取值出现的次数	统计所得的系列
DataFrame.mean()	按行或按列分别计算平均值	系列或数据框
DataFrame.dropna()	删除所有缺失数据的行或列	数据框
DataFrame.drop_duplicates()	删除所有重复的行	数据框
DataFrame.head()	默认返回数据框中的前五行，以验证数据样式	数据框
Dataframe.tail()	默认返回数据框中的最后五行	数据框

目录


1
模块的概述
2
NumPy
3
Pandas
4
SciPy
5
Scikit-learn
6
其他Python常用模块

SciPy

- 在Python的科学计算栈中，SciPy为数学、物理、工程等方面涉及的科学计算提供无可替代的支持。
- 它是一个基于NumPy的高级模块，在符号计算、信号处理、数值优化等任务中有突出表现，覆盖了绝大部分科学计算领域。SciPy主要子模块汇总表如下表：

子模块名称	相关领域/用途描述
scipy.cluster	主流的聚类算法
scipy.constants	数学和物理常数
scipy.fftpack	快速傅里叶变换
scipy.integrate	求解积分和常微分方程
scipy.linalg	线性代数
scipy.ndimage	n维图像处理
scipy.signal	信号处理
scipy.spatial	空间数据结构和算法
scipy.stats	统计分布及相关函数

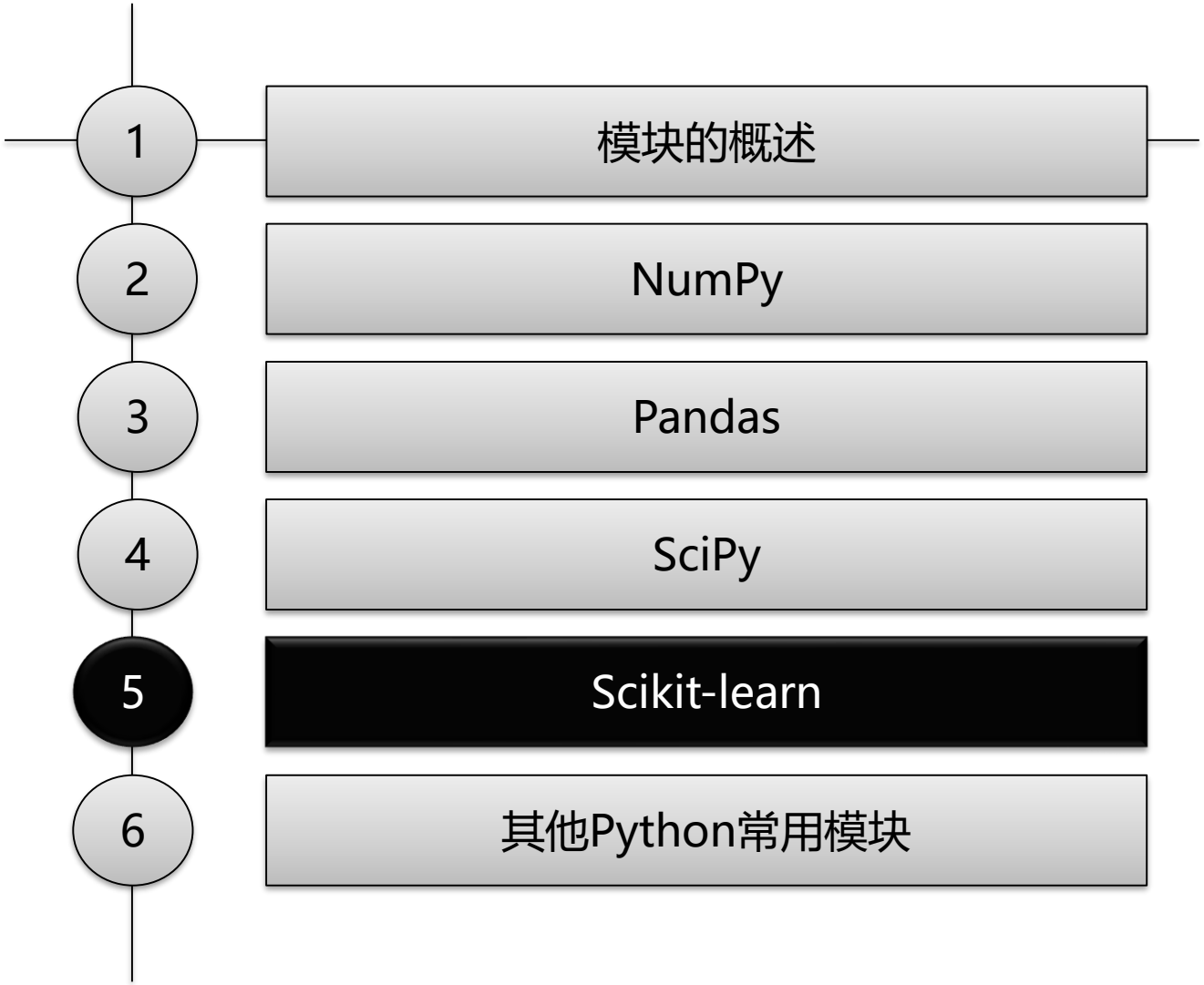
符号计算

- 众所周知，程序中使用的变量仅代表一个空间，真正参与运算的是这个空间中存放的内容或取值。也就是说，数学中最常见的代数表达式，如x^2+x+1，在程序中是没有意义的。
- 但这就是SciPy特别之处，它能够支持符号计算。我们有两种等价的方式去处理一元n次多项式，从而可以不加赋值的进行符号计算。其中一种方式就是使用NumPy中的plot1d类。它可以通过多项式系数或者多项式的根显示地声明一个多项式，并进行加、减、乘、除、积分、求导等操作。

函数向量化

- 为了增强程序的健壮性，通常的做法是使函数接受向量形式的参数传入，以达到高效的运算或处理效率。
- Python无法彻底地支持这一点，但SciPy很好地弥补了这个缺憾。有一个很特别的用法，便是将函数本身作为参数，传递给vectorize()函数作为其参数，经过处理返回一个能接受向量化输入的函数。
- 当你使用SciPy模块时，你很可能需要优化、信号处理、函数变换等功能。
- 如果你打算自己实现这些功能以应对特殊的需求，那么这个特别的函数将使你的工作量大大减少。同时，你的代码将更加优雅。

目录

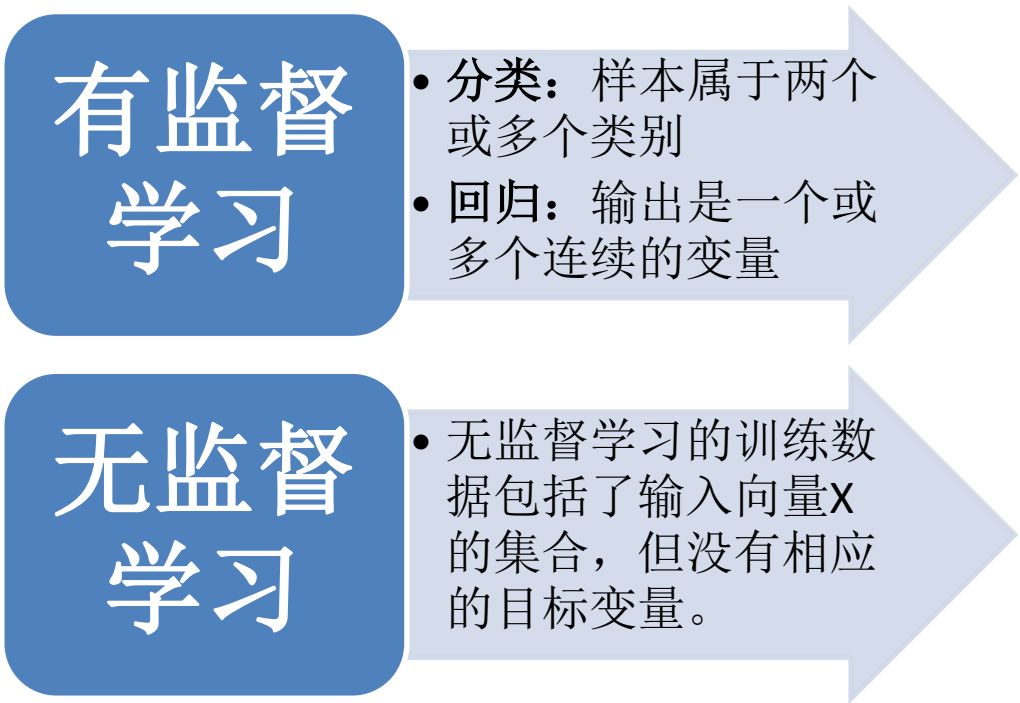

1
模块的概述
2
NumPy
3
Pandas
4
SciPy
5
Scikit-learn
6
其他Python常用模块

scikit-learn

- 本节介绍的是Python在机器学习方面一个非常非常强力的模块，scikit-learn。
- scikit-learn是在NumPy，SciPy和matplotlib三个模块上编写的，是数据挖掘和数据分析的一个简单而有效的工具。
- 在其官方网站上我们可以看到scikit-learn有6大功能如下：

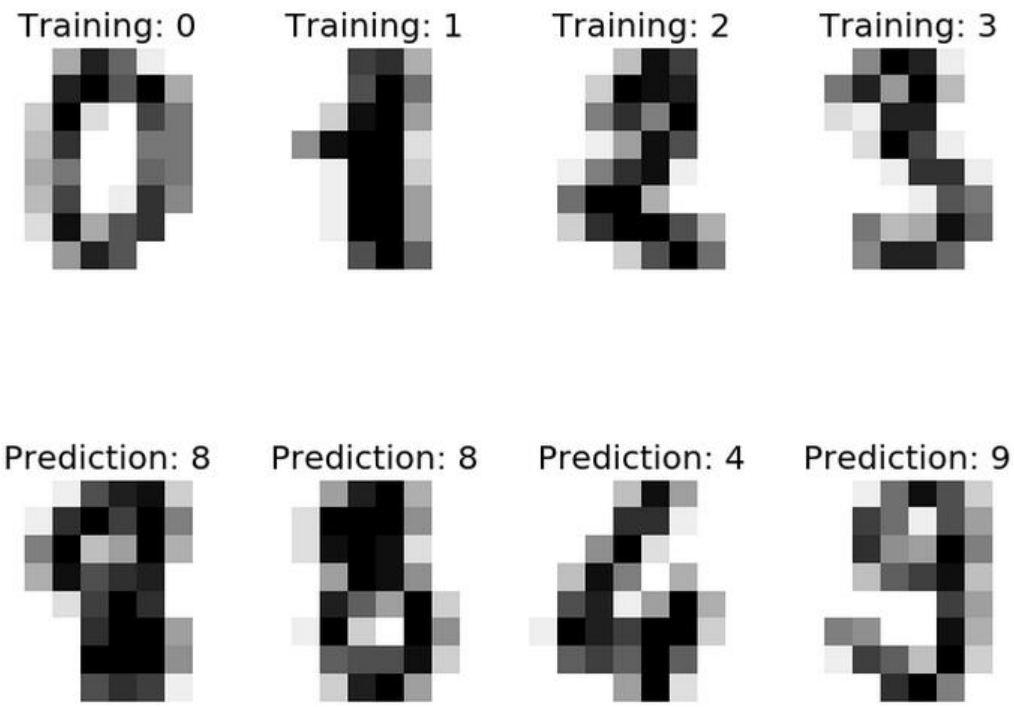
机器学习

- 一般来说，我们可以这样理解机器学习问题：我们n个样本（sample）的数据集，想要预测未知数据的属性。如果样本的数据是多维的，那么我们就说样本具有多个属性或特征。
- 我们可以将学习问题分为以下几类：



scikit-learn的数据集

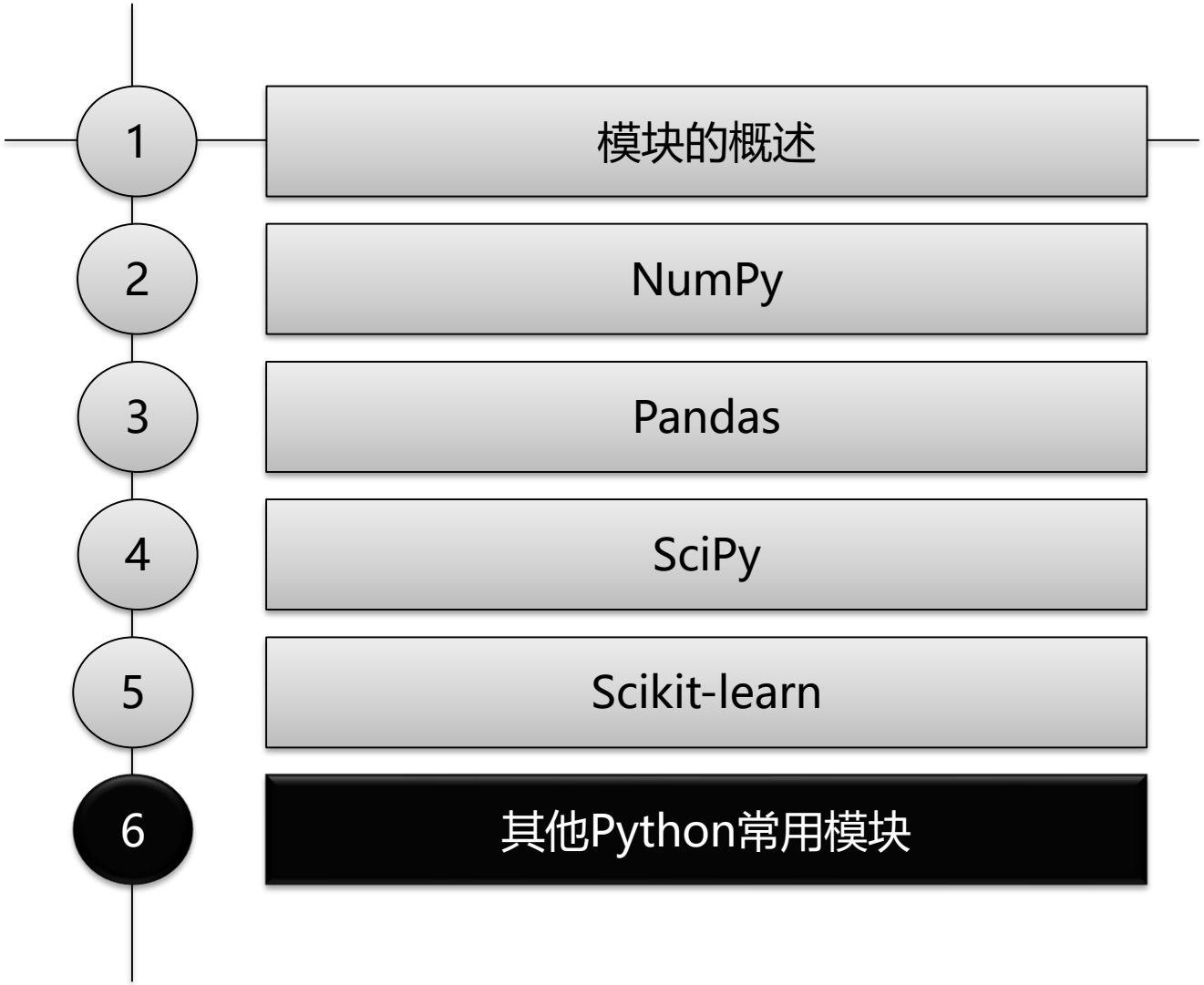
- scikit-learn有一些标准数据集，比如分类的iris和digits数据集和用于回归的波士顿房价（boston house prices）数据集。
- 针对digits数据集的任务是给定一个8*8像素数组，程序能够预测这64个像素代表哪个数字，如下图 所示：



scikit-learn的训练和预测

- 接着上面的例子，我们的任务是给定一副像素图案，预测其表示的数字。这是一个有监督学习的分类问题，总共有10个可能的分类（数字0到9）。我们将训练一个预测器（estimator）来预测（predict）未知样本所属分类。
- 在scikit-learn中，分类的预测器是一个Python对象，具有方法fit(X,y)和predict(test)方法。

目录


1
模块的概述
2
NumPy
3
Pandas
4
SciPy
5
Scikit-learn
6
其他Python常用模块

其他Python常用模块.

- Python处理数据挖掘的模块有很多，下面表格对Python处理数据挖掘常用模块进行简单介绍。

模块名称	用途
Theano	Theano 是一个 Python 库，用来定义、优化和模拟数学表达式计算，用于高效的解决多维数组的计算问题以及深度学习框架。
Keras	Keras是基于Theano的深度学习库，主要用于搭建人工神经网络，自编码器，卷积神经网络等深度学习模型。
Gensim	Gensim是Python的自然语言处理模块，包括了自然语言主题模型，用于文本的主题挖掘。
StatsModels	StatsModels是注重数据统计建模分析的数据处理模块，它与pandas结合，是当前Python的一个强大数据挖掘组合。
Pygame	Pygame是专为电子游戏设计的Python模块
NLTK	NLTK（Natural Language Toolkit）是Python的自然语言处理模块，包括一系列的字符处理和语言统计模型。NLTK 常用于学术研究和教学，应用的领域有语言学、认知科学、人工智能、信息检索、机器学习等。
Mlpy	Mlpy是基于NumPy和SciPy的机器学习模块，是CPython的拓展应用。
PyBrain	PyBrain是Python的一个机器学习模块，主要用于处理神经网络、强化学习、无监督学习、进化算法。
Milk	Milk是Python的一个机器学习工具箱，其重点是提高监督分类法与几种有效的分类分析：SVMs（基于libsvm），kNN，随机森林和决策树。

Thank You!