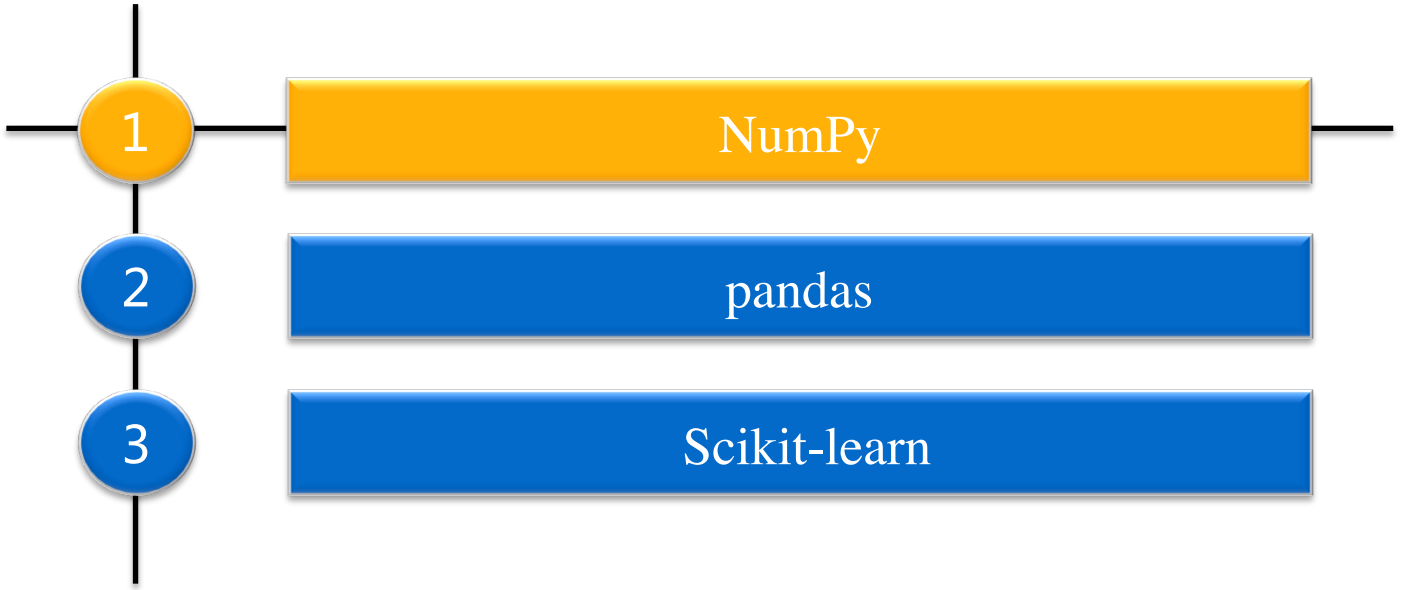
科学计算与机器学习库

目录



NumPy

简介

数值计算的需求

- 2006年，NumPy v1.0

ndarrary：NumPy库的心脏

- ndarray：多维数组，具有矢量运算能力，且快速、节省空间
- 可对整组数据进行快速运算的标准数学函数、线性代数、随机数生成等功能
- import numpy as np

NumPy

N维数组对象

- np.array(collection)，collection可以是列表或元组
- d=((1,2,3),(2,4,1))
- d=np.array(d)
- np.zeros() 生成零矩阵（矩阵属于数组）
- np.ones() 生成全1矩阵
- np.identity() 生成单位矩阵
- print(d[1,1]) #索引
- print(d[1:2,0:2]) #切片

创建数组

arange

- 使用格式：np.arange(开始值, 终值, 步长)
- 类似Python的range()，注意不包括终值

linspace

- 使用格式：np.linspace(开始值, 终值, 元素个数)
- 默认包括终值，可以使用endpoint设置是否包括终值

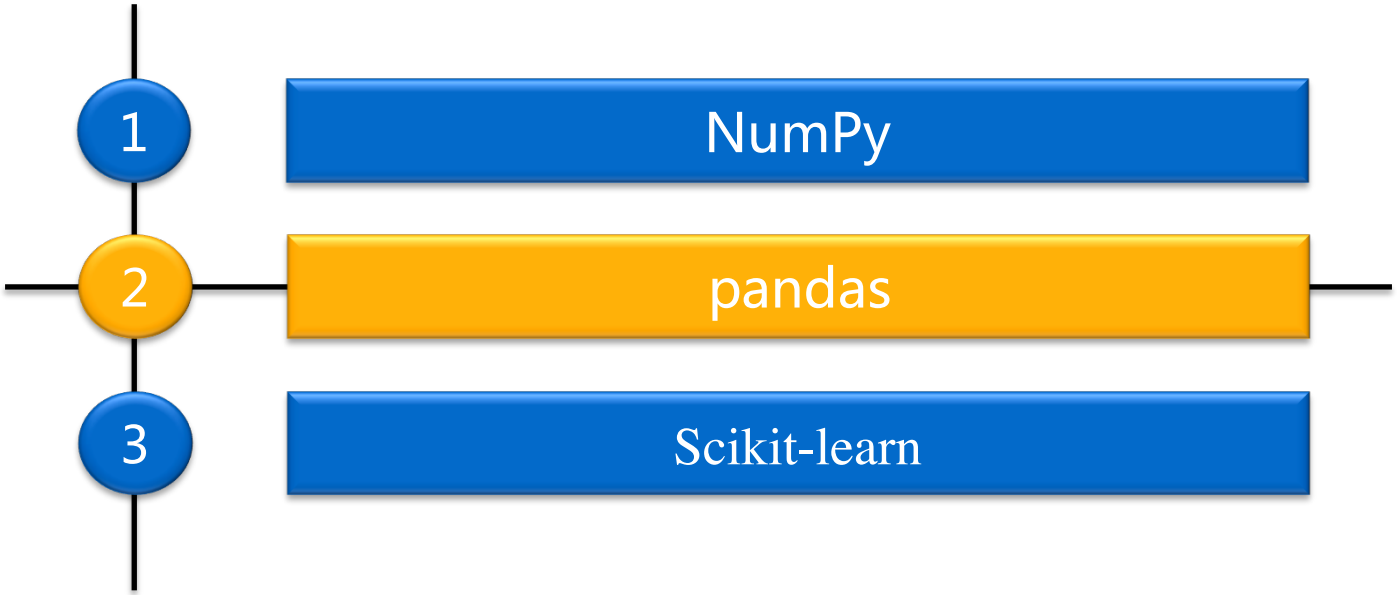
reshape

- 使用格式：arr.reshape(shape)
- 转换数组的规模但不更改其中的数据，常搭配arange或lispace使用

访问数组：索引与切片

- 索引：选取数组的元素，以[]表示
- 切片：带有冒号（ ： ）的索引表示切片，用来访问一定范围内的元素，冒号前表示起始索引，冒号后表示终止索引，但不包含终止索引所对应的元素。
- 如：arr[0:2] 表示从第一个元素开始，直到但不包含第三个元素
- 注意：数组的切片返回的是原始数组的视图，即对切片的任何修改，都会反映到原始数组上
- 一维数组的索引与Python的列表索引类似

目录



pandas

pandas：强大的数据分析和处理工具。

- 快速、灵活、富有表现力的数据结构（Series系列和DataFrame数据框）
- 支持类似SQL的数据增、删、查、改
- 带有丰富的数据处理函数
- 支持时间序列分析功能
- 支持灵活处理缺失数据

pandas 文件读写

读文件

- read_table
- read_csv
- read_excel
- read_hdf
- read_sql
- read_json
- read_msgpack (experimental)
- read_html
- read_gbq (experimental)
- read_clipboard

pandas 文件读写

写文件：

- to_csv
- to_excel
- to_hdf
- to_sql
- to_json
- to_msgpack (experimental)
- to_html
- to_gbq (experimental)
- to_stata
- to_clipboard

pandas

pandas：强大的数据分析和处理工具。

核心函数

- Series()
- DataFrame()

导入方式：

- import pandas as pd
- import numpy as np
- from pandas import DataFrame,Series

Series()

- 一维序列，类比列表
- 每个元素具有名称

- ser = Series([1,2,'a'],index=['a','b','c'])
- ser = Series({'a':[1,2,3],'b':['1','2','3']})

数据框：DataFrame()

数据框的构造：

- d=[[1.3,2.0,3,4],[2,4,1,4],[2,5,1.9,7],[3,1,0,11]]
- df = DataFrame(d,index=['a','b','c','d'],columns=list('ABCD'))
- DataFrame(index=['1','2'],columns=['b','c'])　#生成缺失值矩阵
- DataFrame(0,index=['1','2'],columns=['b','c']) #生成全零矩阵
- d={'color':['blue','green','yellow','red','white'],
  'object':['ball','pen','pencil','paper','mug'],
  'price':[1.2,1.0,0.6,0.9,1.7]}
  frame=DataFrame(d,index=['a','b','c','d','e'])

➢ 注意行列名称构造

索引

- 数据框的索引
- 通过标签索引
- 通过位置索引
- 布尔索引
- 其他索引方式
- df[:2]
- df[:'c'] #行
- df['A'] #列
- df[['A','C']]

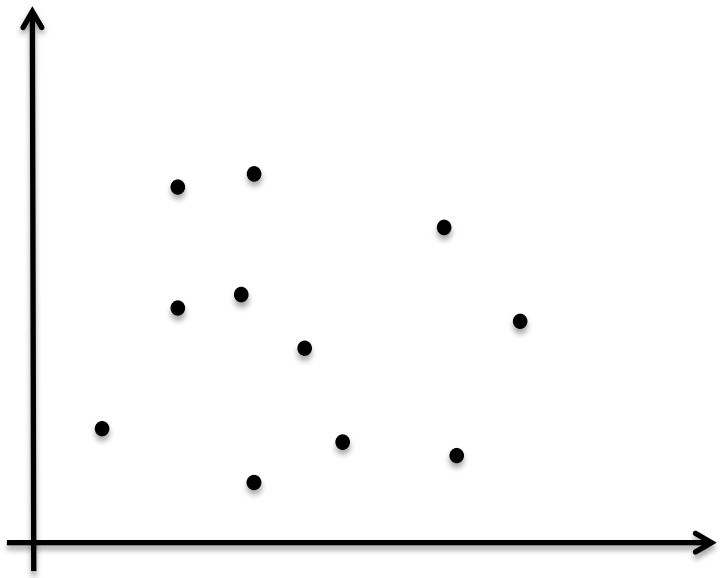
数据框的索引

- #ix既能通过行列名称也能通过位置索引
- df.ix[:,['A','B']]
- df.ix[3:5,0:2]
- df.ix[3,2]
- df.ix[[0,2],1] = 1
- df = DataFrame(index=['0','1','2','3'],columns=['b','c']) #生成缺失值矩阵
- df.ix[[True,False,True,True],1] = 1 #逻辑索引

索引小结

- Pandas的索引可归纳为3种：
- loc：标签索引
- iloc：位置索引
- ix：标签与位置混合索引
- 注意：
- DataFrame索引操作时可将其看作ndarray操作
- 标签的切片索引是包含终止索引的

练习3：平面上有100个点，求任意2点间的距离，并将其保存。



	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5									
6									
7									
8									
9									

练习4：数据索引

练习4：数据索引

1. 将iris.data与iris.target合并成一个数据框，要求列名分别为：‘sepal length’、‘sepal width’、‘petal length’、‘petal width’；并将第5列前50、中间50、最后50个元素分别赋值为：'setosa'、'versicolor'、'virginica'
2. 找出iris数据集中virginica种类的样本数据
3. 找出iris数据集中Sepal.Width大于4的样本数据

综合练习：K-means算法实现

- 对平面上的100个点进行聚类，要求聚为2类，其横纵坐标都为0到99；
- K-means算法流程：

1、随机选取k个样本作为初始类中心；

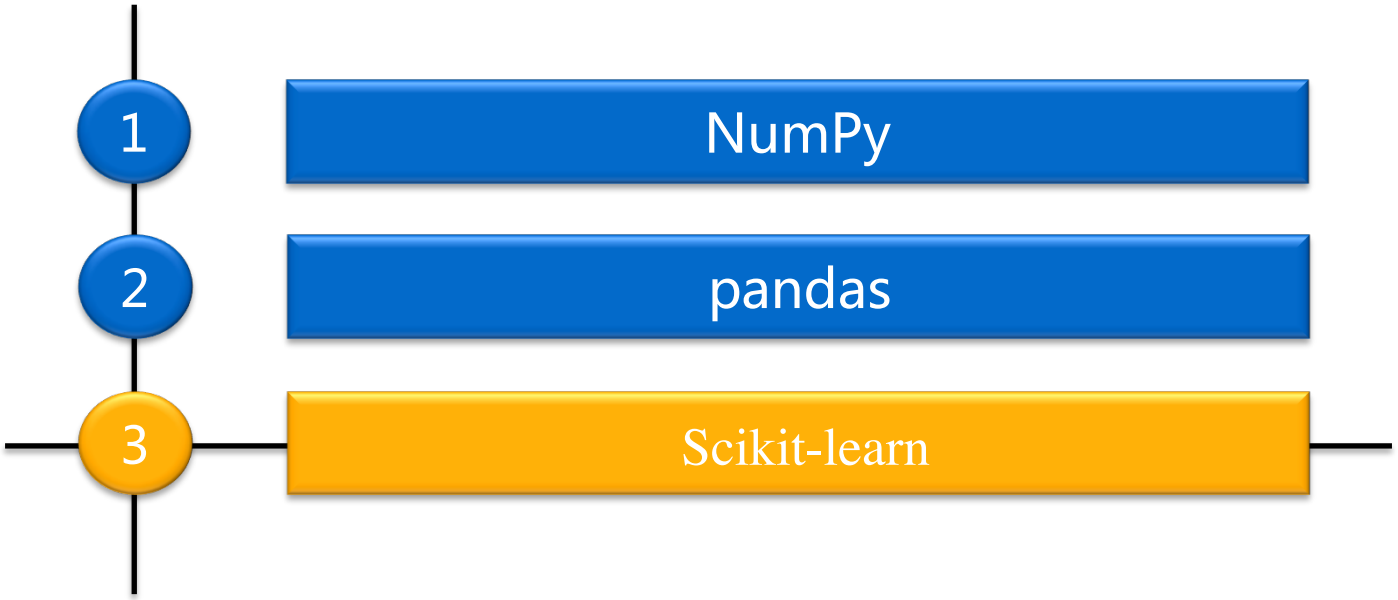
2、求各样本至各类中心的距离；

3、将样本归为距离其最近的类中心所属类；

4、计算各类样本均值作为新类中心；

5、判断类中心有无变换，有则跳至第2步，若无则结束算法。

目录



Scikit-learn

机器学习算法库scikit-learn简介

- 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：
- 学习问题主要可以归为2类：

有监督学习

分类：样本属于两个或多个类别
回归：输出是一个或多个连续的变量

无监督学习

无监督学习的训练数据包括了输入向量X的集合，但没有相应的目标变量

Scikit-learn

安装和使用

- pip3 install sklearn
- **from** sklearn **import** …

API文档：

- http://scikit-learn.org/stable/modules/classes.html

Scikit-learn

主要函数库

- sklearn.datasets: Datasets（数据集）
- sklearn.tree: Decision Trees（决策树）
- sklearn.neural_network: Neural network models（神经网络）
- sklearn.svm: Support Vector Machines（支持向量机）
- sklearn.cluster: Clustering（聚类分析）
- sklearn.naive_bayes: Naive Bayes（朴素贝叶斯）
- sklearn.neighbors：KNeighborsClassifier（KNN）
- sklearn.covariance: Covariance Estimators（协方差估计）
- sklearn.model_selection: Model Selection（模型选择）

神经网络

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	class
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
7	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.3	3.3	6	2.5	virginica
5.8	2.7	5.1	1.9	virginica
6.5	3	5.8	2.2	?
6.2	2.9	4.3	1.3	?

Scikit-learn

神经网络

代码实现（sklearn）

- Net = MLPClassifier(hidden_layer_sizes=10,max_iter=1000).fit(tr_data.ix[:,0:6],tr_data.ix[:,6])
- res = Net.predict(te_data.ix[:,0:6])

决策树

代码实现（sklearn）

- modle = DecisionTreeClassifier(criterion='gini').fit(tr_data.ix[:,0:6],tr_data.ix[:,6]) #模型训练
- res = modle.predict(te_data.ix[:,0:6])

Thank you!

泰迪科技：www.tipdm.com
热线电话：40068-40020