Python与数据分析

第八节：scikit-learn

主讲人：江琳

scikit-learn简介

- scikit-learn是基于Python的机器学习模块
- 这个项目早在2007年发起，目前是由社区志愿者进行维护。
- scikit-learn的基本功能主要分为六部分
  - 分类、回归、聚类、数据降解、数据预处理、和模型选择

scikit-learn简介

- Github： https://github.com/scikit-learn/scikit-learn
- 官网： http://scikit-learn.org/stable/documentation.html

scikit-learn的安装

Installing scikit-learn

Note: If you wish to contribute to the project, it's recommended you install the latest development version.

Installing the latest release

Scikit-learn requires:

- Python (>= 2.6 or >= 3.3),
- NumPy (>= 1.6.1),
- SciPy (>= 0.9).

If you already have a working installation of numpy and scipy, the easiest way to install scikit-learn is using `pip`

```
pip install -U scikit-learn
```

or `conda`:

```
conda install scikit-learn
```

scikit-learn的API

powered by Google

API Reference

This is the class and function reference of scikit-learn. Please refer to the full user guide for further details, as the class and function raw specifications may not be enough to give full guidelines on their uses.

`sklearn.base` : Base classes and utility functions

Base classes for all estimators.

Base classes

`base.BaseEstimator`	Base class for all estimators in scikit-learn
`base.ClassifierMixin`	Mixin class for all classifiers in scikit-learn.
`base.ClusterMixin`	Mixin class for all cluster estimators in scikit-learn.
`base.RegressorMixin`	Mixin class for all regression estimators in scikit-learn.
`base.TransformerMixin`	Mixin class for all transformers in scikit-learn.

Functions

`base.clone` (estimator[, safe])	Constructs a new estimator with the same parameters.

`sklearn.cluster` : Clustering

The `sklearn.cluster` module gathers popular unsupervised clustering algorithms.

User guide: See the Clustering section for further details.

Classes

`cluster.AffinityPropagation` ([damping, ...])	Perform Affinity Propagation Clustering of data.
`cluster.AgglomerativeClustering` ([...])	Agglomerative Clustering
`cluster.Birch` ([threshold, branching_factor, ...])	Implements the Birch clustering algorithm

数据集模块

sklearn.datasets : Datasets

The `sklearn.datasets` module includes utilities to load datasets, including methods to load and fetch popular reference datasets. It also features some artificial data generators.

User guide: See the Dataset loading utilities section for further details.

Loaders

`datasets.clear_data_home` ([data_home])	Delete all the content of the data home cache.
`datasets.get_data_home` ([data_home])	Return the path of the scikit-learn data dir.
`datasets.fetch_20newsgroups` ([data_home, ...])	Load the filenames and data from the 20 newsgroups dataset.
`datasets.fetch_20newsgroups_vectorized` ([...])	Load the 20 newsgroups dataset and transform it into tf-idf vectors.
`datasets.load_boston` ([return_X_y])	Load and return the boston house-prices dataset (regression).
`datasets.load_breast_cancer` ([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).
`datasets.load_diabetes` ([return_X_y])	Load and return the diabetes dataset (regression).
`datasets.load_digits` ([n_class, return_X_y])	Load and return the digits dataset (classification).
`datasets.load_files` (container_path[, ...])	Load text files with categories as subfolder names.
`datasets.load_iris` ([return_X_y])	Load and return the iris dataset (classification).
`datasets.fetch_lfw_pairs` ([subset, ...])	Loader for the Labeled Faces in the Wild (LFW) pairs dataset
`datasets.fetch_lfw_people` ([data_home, ...])	Loader for the Labeled Faces in the Wild (LFW) people dataset
`datasets.load_linnerud` ([return_X_y])	Load and return the linnerud dataset (multivariate regression).
`datasets.mldata_filename` (dataname)	Convert a raw name for a data set in a mldata.org filename.
`datasets.fetch_mldata` (dataname[, ...])	Fetch an mldata.org data set
`datasets.fetch_olivetti_faces` ([data_home, ...])	Loader for the Olivetti faces data-set from AT&T.
`datasets.fetch_california_housing` ([...])	Loader for the California housing dataset from StatLib.
`datasets.fetch_covtype` ([data_home, ...])	Load the covertype dataset, downloading it if necessary.
`datasets.fetch_kddcup99` ([subset, shuffle, ...])	Load and return the kddcup 99 dataset (classification).
`datasets.fetch_rcv1` ([data_home, subset, ...])	Load the RCV1 multilabel dataset, downloading it if necessary.
`datasets.load_mlcomp` (name_or_id[, set_, ...])	Load a datasets as downloaded from http://mlcomp.org
`datasets.load_sample_image` (image_name)	Load the numpy array of a single sample image
`datasets.load_sample_images` ()	Load sample images for image manipulation.
`datasets.fetch_species_distributions` ([...])	Loader for species distribution dataset from Phillips et.

scikit-learn的DataSets模块包含有很多数据集，可以用来测试

iris数据集

sklearn.datasets.load_iris

`sklearn.datasets.` **`load_iris`** (*return_X_y=False*) [source]

Load and return the iris dataset (classification).

The iris dataset is a classic and very easy multi-class classification dataset.

Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

Read more in the User Guide.

Parameters: **return_X_y** : boolean, default=False.

If True, returns `(data, target)` instead of a Bunch object. See below for more information about the data and target object.

New in version 0.18.

Returns: **data** : Bunch

Dictionary-like object, the interesting attributes are: 'data', the data to learn, 'target', the classification labels, 'target_names', the meaning of the labels, 'feature_names', the meaning of the features, and 'DESCR', the full description of the dataset.

(data, target) : tuple if `return_X_y` is True

New in version 0.18.

鸢尾花数据集

- 鸢尾花数据集是20世纪30年代的经典数据集。
- 数据特征：
  - 花萼长度
  - 花萼宽度
  - 花瓣长度
  - 花瓣宽度

鸢尾花数据集

- 分类：
  - 山鸢尾花
  - 变色鸢尾花
  - 维吉尼亚鸢尾花

导入iris数据集

```
In [1]: from sklearn.datasets import load_iris

In [2]: dataset = load_iris()
```

1. 使用from … import … 导入load_iris数据集方法
2. 使用load_iris()获得数据集

查看iris里的数据内容

```
In [5]: dataset.data
Out[5]:
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
```

查看数据的分类向量

```
In [6]: dataset.target
Out[6]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

dataset.target来查看iris数据集的分类向量，可以看出这一共有三种分类

查看数据的分类名称

```
In [7]: dataset.target_names
Out[7]:
array(['setosa', 'versicolor', 'virginica'],
      dtype='<U10')
```

.target_names用来获取分类名称，这里一共有三个分类

练习

1. 查看iris数据集的形状
2. 查看iris分类的样本大小

朴素贝叶斯

`sklearn.naive_bayes` : Naive Bayes

The `sklearn.naive_bayes` module implements Naive Bayes algorithms. These are supervised learning methods based on applying Bayes' theorem with strong (naive) feature independence assumptions.

User guide: See the Naive Bayes section for further details.

`naive_bayes.GaussianNB` ([priors])	Gaussian Naive Bayes (GaussianNB)
`naive_bayes.MultinomialNB` ([alpha, ...])	Naive Bayes classifier for multinomial models
`naive_bayes.BernoulliNB` ([alpha, binarize, ...])	Naive Bayes classifier for multivariate Bernoulli models.

1. scikit-learn包含朴素贝叶斯算法模块
2. 其中包含三种实现方式：
    1. 高斯朴素贝叶斯
    2. 多项式朴素贝叶斯
    3. 伯努利朴素贝叶斯

伯努利朴素贝叶斯

- 伯努利模型中，对于一个样本来说，其特征用的是全局的特征。
- 在伯努利模型中，每个特征的取值是布尔型的，即true和false，或者1和0。在文本分类中，就是一个特征有没有在一个文档中出现。

伯努利朴素贝叶斯

sklearn.naive_bayes.BernoulliNB

class `sklearn.naive_bayes.` **`BernoulliNB`** (*alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None*) [source]

Naive Bayes classifier for multivariate Bernoulli models.

Like MultinomialNB, this classifier is suitable for discrete data. The difference is that while MultinomialNB works with occurrence counts, BernoulliNB is designed for binary/boolean features.

Read more in the User Guide.

Parameters:

alpha : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

binarize : float or None, optional (default=0.0)

Threshold for binarizing (mapping to booleans) of sample features. If None, input is presumed to already consist of binary vectors.

fit_prior : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

class_prior : array-like, size=[n_classes,], optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

使用方法

Examples

```
>>> import numpy as np
>>> X = np.random.randint(2, size=(6, 100))
>>> Y = np.array([1, 2, 3, 4, 4, 5])
>>> from sklearn.naive_bayes import BernoulliNB
>>> clf = BernoulliNB()
>>> clf.fit(X, Y)
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
>>> print(clf.predict(X[2:3]))
[3]
```

Methods

`fit` (X, y[, sample_weight])	Fit Naive Bayes classifier according to X, y
`get_params` ([deep])	Get parameters for this estimator.
`partial_fit` (X, y[, classes, sample_weight])	Incremental fit on a batch of samples.
`predict` (X)	Perform classification on an array of test vectors X.
`predict_log_proba` (X)	Return log-probability estimates for the test vector X.
`predict_proba` (X)	Return probability estimates for the test vector X.
`score` (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params` (**params)	Set the parameters of this estimator.

`__init__` (*alpha=1.0, binarize=0.0, fit_prior=True, class_prior=None*) [source]

`fit` (*X, y, sample_weight=None*) [source]

Fit Naive Bayes classifier according to X, y

Parameters: **X** : {array-like, sparse matrix}, shape = [n_samples, n_features]

Training vectors, where n_samples is the number of samples and n_features is the number of features.

使用方法

- 导入
  - **from sklearn.naive_bayes import BernoulliNB**
- 创建对象
  - **clf = BernoulliNB()**
- 训练样本
  - **clf.fit(x, y)**
  - **fit方法用来训练函数，x为训练样本数据，y为训练样本分类**

使用方法

- 预测数据
- clf.predict()
- predict方法来预测数据

使用方法

```
In [1]: from sklearn.naive_bayes import BernoulliNB

In [2]: clf = BernoulliNB()

In [3]: import numpy as np

In [4]: x = np.random.randint(2, size=(6,100))

In [5]: x
Out[5]:
array([[1, 1, 1, ..., 0, 1, 0],
       [1, 0, 0, ..., 1, 1, 1],
       [0, 0, 0, ..., 1, 1, 0],
       [1, 0, 1, ..., 1, 1, 0],
       [0, 0, 0, ..., 1, 0, 1],
       [1, 1, 0, ..., 1, 0, 0]])

In [6]: y = np.array([1,2,3,4,4,5])

In [7]: clf.fit(x,y)
Out[7]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)

In [8]: result = clf.predict(x)

In [9]: result
Out[9]: array([1, 2, 3, 4, 4, 5])
```

随机创建一个训练样本

多项式朴素贝叶斯

- 多项式模型会考虑样本元素出现的次数，而不是只考虑是否出现

多项式朴素贝叶斯

sklearn.naive_bayes.MultinomialNB

class `sklearn.naive_bayes.` **`MultinomialNB`** (*alpha=1.0, fit_prior=True, class_prior=None*) [source]

Naive Bayes classifier for multinomial models

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Read more in the User Guide.

Parameters:

alpha : float, optional (default=1.0)

Additive (Laplace/Lidstone) smoothing parameter (0 for no smoothing).

fit_prior : boolean, optional (default=True)

Whether to learn class prior probabilities or not. If false, a uniform prior will be used.

class_prior : array-like, size (n_classes,), optional (default=None)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

Attributes:

class_log_prior_ : array, shape (n_classes,)

Smoothed empirical log probability for each class.

多项式朴素贝叶斯

Examples

```
>>> import numpy as np
>>> X = np.random.randint(5, size=(6, 100))
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> from sklearn.naive_bayes import MultinomialNB
>>> clf = MultinomialNB()
>>> clf.fit(X, y)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
>>> print(clf.predict(X[2:3]))
[3]
```

Methods

`fit` (X, y[, sample_weight])	Fit Naive Bayes classifier according to X, y
`get_params` ([deep])	Get parameters for this estimator.
`partial_fit` (X, y[, classes, sample_weight])	Incremental fit on a batch of samples.
`predict` (X)	Perform classification on an array of test vectors X.
`predict_log_proba` (X)	Return log-probability estimates for the test vector X.
`predict_proba` (X)	Return probability estimates for the test vector X.
`score` (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params` (**params)	Set the parameters of this estimator.

高斯朴素贝叶斯

- 有些特征可能是连续型变量，比如说人的身高，物体的长度，这些特征可以转换成离散型的值，比如如果身高在160cm以下，特征值为1；在160cm和170cm之间，特征值为2；在170cm之上，特征值为3。也可以这样转换，将身高转换为3个特征，分别是f1、f2、f3，如果身高是160cm以下，这三个特征的值分别是1、0、0，若身高在170cm之上，这三个特征的值分别是0、0、1。不过这些方式都不够细腻，高斯模型可以解决这个问题。

- 高斯模型假设这些一个特征的所有属于某个类别的观测值符合高斯分布

高斯朴素贝叶斯

sklearn.naive_bayes.GaussianNB

class `sklearn.naive_bayes.` **`GaussianNB`** (*priors=None*) [source]

Gaussian Naive Bayes (GaussianNB)

Can perform online updates to model parameters via partial_fit method. For details on algorithm used to update feature means and variance online, see Stanford CS tech report STAN-CS-79-773 by Chan, Golub, and LeVeque:

http://i.stanford.edu/pub/cstr/reports/cs/tr/79/773/CS-TR-79-773.pdf

Read more in the User Guide.

Parameters:	**priors** : array-like, shape (n_classes,) Prior probabilities of the classes. If specified the priors are not adjusted according to the data.
Attributes:	**class_prior_** : array, shape (n_classes,) probability of each class. **class_count_** : array, shape (n_classes,) number of training samples observed in each class. **theta_** : array, shape (n_classes, n_features) mean of each feature per class

高斯朴素贝叶斯

Examples

```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> Y = np.array([1, 1, 1, 2, 2, 2])
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
GaussianNB(priors=None)
>>> print(clf.predict([[-0.8, -1]]))
[1]
>>> clf_pf = GaussianNB()
>>> clf_pf.partial_fit(X, Y, np.unique(Y))
GaussianNB(priors=None)
>>> print(clf_pf.predict([[-0.8, -1]]))
[1]
```

Methods

`fit` (X, y[, sample_weight])	Fit Gaussian Naive Bayes according to X, y
`get_params` ([deep])	Get parameters for this estimator.
`partial_fit` (X, y[, classes, sample_weight])	Incremental fit on a batch of samples.
`predict` (X)	Perform classification on an array of test vectors X.
`predict_log_proba` (X)	Return log-probability estimates for the test vector X.
`predict_proba` (X)	Return probability estimates for the test vector X.
`score` (X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
`set_params` (**params)	Set the parameters of this estimator.

使用iris数据集

```
In [2]: from sklearn.datasets import load_iris

In [3]: iris = load_iris()

In [4]: iris.feature_names
Out[4]:
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

特征名称

训练和预测iris数据

```
In [5]: from sklearn.naive_bayes import GaussianNB

In [6]: clf = GaussianNB()

In [7]: clf.fit(iris.data, iris.target)
Out[7]: GaussianNB(priors=None)

In [8]: result = clf.predict(iris.data)
```

结果

```
In [15]: result
Out[15]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

In [16]: iris.target
Out[16]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

练习

1. 计算用iris数据集进行高斯朴素贝叶斯的正确率
2. iris数据集作为训练集，使用scikit-learn的k-近邻算法模块

Questions?