4.聚类算法

# 1 实验目的

1.理解sklearn.cluster模块中K-均值聚类算法、聚合聚类算法的实现原理；

2.了解如何使用sklearn.cluster模块中的常用算法完成机器学习任务。

# 2 实验要求

本次实验后，要求同学们能：

1.掌握k-均值聚类算法和聚合聚类算法的实现原理；

2.学会使用Scikit-learn实现常用的聚类算法；

3.了解sklearn.cluster模块中各个常用算法的特点，了解各个超参数的含义。

# 3 实验内容

## 3.1 聚类算法简介

聚类算法与分类算法类似，都是将数据划分为几个类或者“簇”，但是分类算法是有监督学习，而聚类算法是无监督学习。

聚类算法主要是考察样本之间的相似度，将相似的样本归为一类，不相似的样本分散在不同类中。可以通过聚类算法发现没有标注的原始数据中的特点或者进行数据预处理，常用在数据挖掘、模式识别等领域。

scikit-learn库中sklearn.cluster模块提供了不同的聚类算法模型，下面介绍最常用的两种聚类算法：k均值聚类和层次聚类中的聚合聚类。

## 3.2 k均值聚类Kmeans

### 3.2.1原理

k均值聚类又称k-means聚类算法，与k邻近算法有很多相似之处。

k均值聚类步骤如下：①事先指定好要聚出的簇数k，通过随机或其他方式指定k个初始的质心，将各个样本分派到离它最近的质心所代表的簇中。②对于上一步中聚出的k个簇，计算各个簇中所有样本的均值得到新的质心，更新k个质心。③重复迭代，根据新的质心重新将样本分簇，然后再根据新的簇更新质心，再重新分簇……直到质心位置不再变化，说明算法以及收敛，即可得到聚类结果。

k均值算法随着不断迭代能够收敛，但不能保证得到全局最优解。另外，k均值聚类算法中，初始质心的选择会对算法产生很大的影响。

### 3.2.2 实验

（1）导入依赖并加载数据

用Scikit-learn自带的鸢尾花数据集作为实验数据，数据集一共含有150个样本（每一类各50个样本），每一行对应一朵花，列代表每朵花的四个测量数据，分别是：花瓣的长度，宽度，花萼的长度、宽度。最后一项表示鸢尾花的类别，共三类，分别是山鸢尾（0表示）、色鸢尾（1表示）、维吉尼亚鸢尾（2表示）

选用鸢尾花花萼的长度和宽度两个特征进行聚类，划分训练集和测试集。由于数据集中花萼长度和宽度的尺度一致，不需要标准化。数据预处理代码如下：

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

# 加载数据

iris\_data = load\_iris()

# 从花萼的长度和宽度角度聚类

data = iris\_data.data[:, :2]

target = iris\_data.target

（2）创建算法模型实例

使用sklearn.cluster模块的Kmeans类可以创建k-means聚类算法实例。其构造器重要参数及其含义如下：

**algorithm：选择K-means聚类计算距离的算法**

**n\_clusters：，表示要聚类形成的簇数，也是K-means聚类的质心数，默认为8。**

**init：选择确定初始质心的方法，默认为“k-means++”表示用一种更合适的方法选择初始质心以加速收敛；选择“random”则表示随机确定质心；也可以提供一个自定义的**ndarray 数组作为质心**。**

**random\_state：如果选择随机生成质心，可以通过给random\_state传入一个整数将随机性确定下来。**

**max\_iter：单次运行k-means聚类算法的最大迭代次数。**

**tol：用于设置算法收敛的阈值，即可以接受的连续两次迭代中质心的最大差值。当实际差值小于tol时视为聚类已完成收敛，与最大迭代次数共同作为聚类的停止条件。算法以F范式计算最大差值。**

**copy\_x：是否在原矩阵上进行计算，默认为True，若设置为False会覆盖原矩阵。**

**用默认值创建一个能得到出3个簇的k-means聚类算法实例，代码如下。**

# 创建k-均值聚类算法实例

km = KMeans(n\_clusters=3)

（3）训练并测试

该算法实例提供以下方法：

**transform（X, weight）：**将X转换为X中个样本到各个质心的距离组成的矩阵。返回X\_new，X\_new为K列的矩阵（K即簇数），X\_new的第i行第j列表示X中的第i个样本点到第j个质心的距离。

此外，fit()、predict()、get\_params()、get\_params()、score()等方法的用法不再赘述。

# 训练数据

km.fit(data)

print("前两条数据(X)\n", data[:2])

print("前两条数据在簇距离空间中的坐标(X\_new)\n", km.transform(data)[:2])

print("质心坐标\n", km.cluster\_centers\_)

print("前两条数据所属类别",km.labels\_[:2])

（4）查看聚类后的参数

可以用parameter\_（参数名加下划线）的形式直接访问训练得到的以下参数：

**cluster\_centers\_：**查看算法得到的质心坐标。

**labels\_：**查看每一个样本点的标签（对应的质心序号）。

**inertia\_：**查看所有样本到距离各自最近的质心的距离平方之和。

**n\_iter\_：**查看算法运行的迭代次数。

**代码示例如下：**

**# 得到聚类结果**

**labels = km.labels\_ # 每个点的聚类结果标签**

**centers = km.cluster\_centers\_ # 聚类完成后各个簇的质心**

**# 查看参数**

**print("\ncluster\_centers\_\n", centers)**

**print("labels\_\n", labels)**

**print("inertia\_\n", km.inertia\_)**

**print("n\_iter\_\n", km.n\_iter\_)**

**（5）数据可视化**

**绘图表示出150朵鸢尾花在花萼长度、宽度这两个维度上的分布情况。然后画出聚类结果，并与实际分类情况做比较。**

**# 打开画布，定义画布大小**

**plt.figure(num=1, figsize=(12, 3))**

**# 调节子图边距**

**plt.subplots\_adjust(wspace=0.5,hspace=0.5)**

**# 用花萼特征聚类前的分布情况**

**plt.subplot(1, 3, 1)**

**plt.scatter(data[:, 0], data[:, 1], s=30)**

**plt.xlabel('Calyx Lenth')**

**plt.ylabel('Calyx Width')**

**plt.title("Raw Data of Calyx")**

**# 用花萼特征聚类后**

**plt.subplot(1, 3, 2)**

**plt.scatter(data[:, 0], data[:, 1], c=labels, s=30, cmap='cool')**

**# 绘制质心点**

**plt.scatter(centers[:,0], centers[:,1], c='red', marker='o', s=60)**

**plt.xlabel('Calyx Lenth')**

**plt.ylabel('Calyx Width')**

**plt.title("K-Means by Calyx Features")**

**# 实际的类别**

**plt.subplot(1, 3, 3)**

**plt.scatter(data[:, 0], data[:, 1], c=target, s=30, cmap='summer')**

**plt.xlabel('Calyx Lenth')**

**plt.ylabel('Calyx Width')**

**plt.title("Real Label")**

**plt.show()**

### 3.2.3 实验结果

（1）用transform方法转换前两条数据的结果

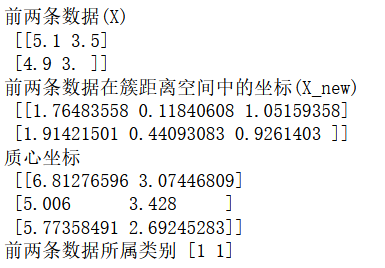


图1 transform方法尝试

拿第一条数据举例，原样本、样本在簇距离空间的坐标（即，样本到各个质心的距离）、质心坐标的关系满足

由K-means算法原理可得，样本距离哪个质心最近就属于哪个质心对应的簇。X\_new中第一行第二个维度最小，说明X中的第一个样本属于第1簇（簇号从0开始计数）。X\_new中第二行第二个维度最小，说明X中第二个样本也属于第1簇。

（2）查看训练出的模型的各个参数

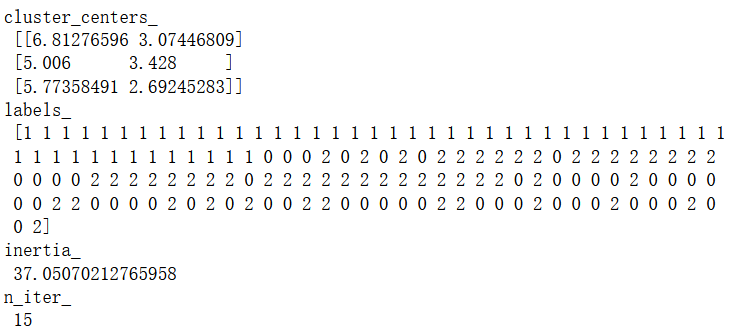


图2 查看参数

（3）可视化

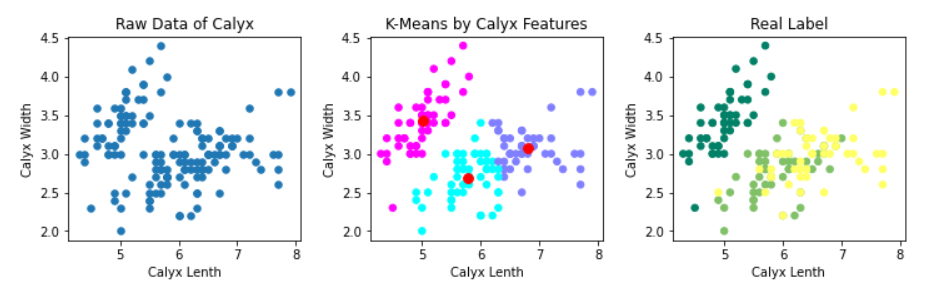


图3 数据可视化

可以看到，只用花萼的长度、宽度两个维度进行K-means聚类，聚类效果良好。

## 3.3 聚合聚类AgglomerativeClustering

### 3.3.1 原理

聚合聚类是层次聚类的一种，它在一开始将每个样本各自分为一类；之后按照一定的**合并规则**，将**相似**的两个类进行合并；反复进行合并操作，每次合并会减少一个类，直到满足**停止条件**。

所以聚合聚类主要需要确定以下三个要素：

（1）距离或相似度度量：可以使用明科夫斯基距离（欧氏距离、曼哈顿距离等）、马氏距离、相关系数、夹角余弦等。

（2）合并规则：类与类之间的距离也称为连接（linkage）时，每次将连接值最小的两个类进行合并。

连接的计算可以使用①**最短距离**，又称单链接（single linkage），用类A所有样本到类B所有样本的所有距离中，最短的距离作为两个类之间的距离；②**最长距离**，又称全连接（complete linkage）， 用类A所有样本到类B所有样本的所有距离中，最长的距离作为两个类之间的距离；③**中心距离**，用类A的中心到类B的中心（比如用整个类所有样本的均值作为中心）的距离作为两个类之间的距离；④**平均距离**，将类A和类B之间任意两个样本之间的距离的平均值作为两个类之间的距离。

（3）停止条件：可以是类的个数达到阈值，或者类的大小超过阈值。

### 3.3.2 实验

（1）数据准备

还是使用之前的鸢尾花花萼长度、宽度信息进行聚类。

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.cluster import AgglomerativeClustering

# 加载数据集

iris\_data = load\_iris()

data = iris\_data.data[:, :2]

target = iris\_data.target

（2）训练模型并可视化

使用sklearn.cluster模块的AgglomerativeClustering类可以创建层次聚类算法模型实例，其构造器的重要参数及含义如下：

**n\_clusters：**设置要聚类的簇数，作为聚合聚类的停止条件，与参数distance\_threshold只能二选一（为其中一个设置了值，另一个就必须为None）。

**affinity：**设置距离或相似度度量方式，可以选择“euclidean”、“manhattan”、“L1”、“L2”、“cosine”等。如果连接方式设置为“ward”，则距离度量只能为“euclidean”。

**linkage：用于设置合并规则中连接的计算方式，可以设置为“**single**”单链接、“**complete**”全连接、“**average**”平均距离连接、“ward”最小方差连接。**

**下面创建三种连接计算方式的聚合聚类算法模型实例，分别进行训练、可视化，比较它们在鸢尾花数据上聚类效果的差异。**

# 创建三种聚合聚类算法实例

agcs = [AgglomerativeClustering(n\_clusters=3, affinity="euclidean",

linkage=criterion) for criterion in ["single", "complete", "average"]]

index = 1 # 画布序号

# 打开画布，定义画布大小

plt.figure(num=1, figsize=(12, 3))

# 调节子图边距

plt.subplots\_adjust(wspace=0.3,hspace=0.3)

# 对三种模型分别训练并可视化

for agc in agcs:

# 训练并得到结果

agc.fit(data)

labels = agc.labels\_

# 可视化

plt.subplot(1, 3, index)

index += 1

plt.scatter(data[:, 0], data[:, 1], c=labels, s=30, cmap='cool')

plt.xlabel('Calyx Lenth')

plt.ylabel('Calyx Width')

plt.title(agc.linkage)

plt.show()

### 3.3.3 实验结果

用三种类不同的间距离计算规则进行聚合聚类，聚类结果如下：

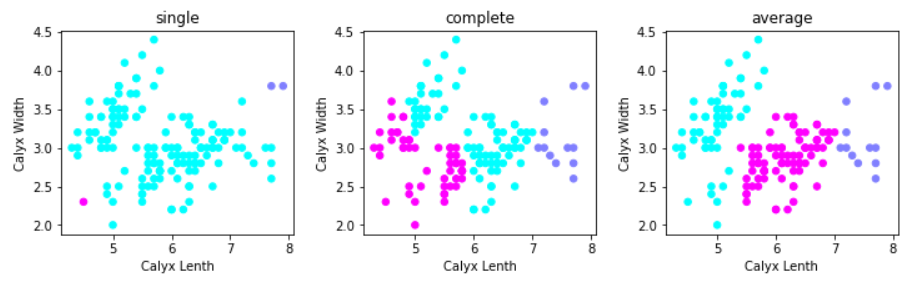


图4 不同合并规则的聚合聚类结果