【华子】机器学习023-使用层次聚类算法构建模型 -

(本文所使用的Python库和版本号: Python 3.6, Numpy 1.14, scikit-learn 0.19, matplotlib 2.2 )

聚类的算法有很多种，前面我们讲解了k-means算法和均值漂移算法，此处我们继续讲解层次聚类算法。

k-means是一种分散性聚类算法，以空间中K个点为中心进行聚类，将最靠近他们的样本收归门下。**k-means的优势**在于简单快速，对于大数据集，该算法仍然可以保持可伸缩性和高效率，对于密集型的数据集，效果非常好。

**缺点也很明显**：必须事先给出K值，而且对初值敏感，对于不同的初始值，可能会产生不同结果；对于非密集型数据集，比如大小差别很大的簇，可能不太适用；而且k-means对噪声和孤立点数据比较敏感。

正是因为k-means有上述优点，所以它才能够经久不衰，也正是因为它有上述缺点，所以它才没有一统天下。

# 1. 层次（hierarchy ）聚类算法简介

层次聚类，顾名思义，就是一层一层的进行聚类，通常，我们可以把整个数据集看成一颗树，每一个数据点就是该树的叶子，而一个节点就是该节点下的叶子总数。故而，对这个数据集进行聚类分析，就相当于找到各种叶子对应的节点，这种寻找方式就是**层次聚类算法。**

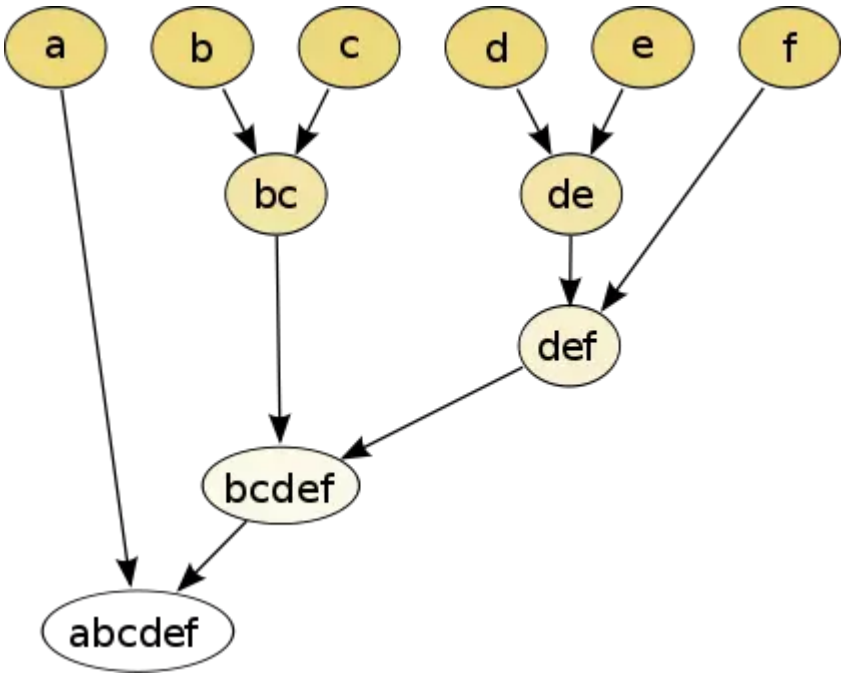
一般的，层次聚类算法有两种：**自下而上的算法**和**自上而下的算法。**

在自下而上的算法中，刚开始每个数据点（即每个叶子）都被看成一个单独的集群，然后将这些集群不断的合并，直到所有的集群都合并成一个巨型集群，这种自下而上的合并算法也叫做**凝聚层次聚类算法。**Agglomerative （类似细胞融合 ）

而相反的，在自上而下的算法中，刚开始所有的叶子被当做一个巨型集群，然后对这个集群进行不断的分解，直到所有的集群都变成一个个单独的数据点，即巨型集群被分解成单独的叶子节点，这种自上而下的的分解算法也叫做**分裂层次聚类算法。**Divisive （类似细胞分裂）

目前应用最多的还是凝聚层次聚类算法，故而下面只对该算法进行介绍。

!



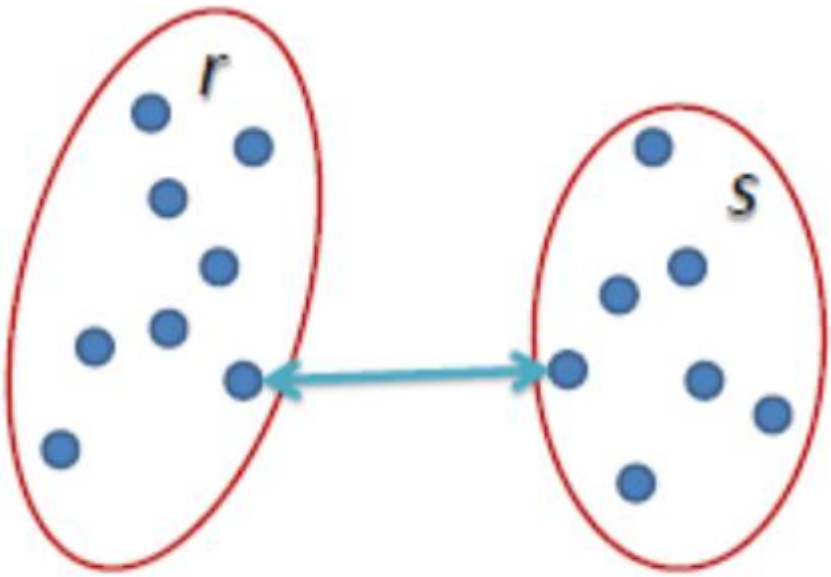
凝聚层次聚类算法的核心思想如上图所示，由叶子逐步合并成根节点。

凝聚法（Agglomerative ）的具体计算过程可以描述为：

1，将数据集中的所有的数据点都当做一个独立的集群 2，计算两两之间的距离，找到距离最小的两个集群，并合并这两个集群为一个集群，认为距离越小，两者之间的相似度越大，越有可能是一个集群。 3，重复上面的步骤2，直到聚类的数目达到设定的条件，表示聚类过程完成。

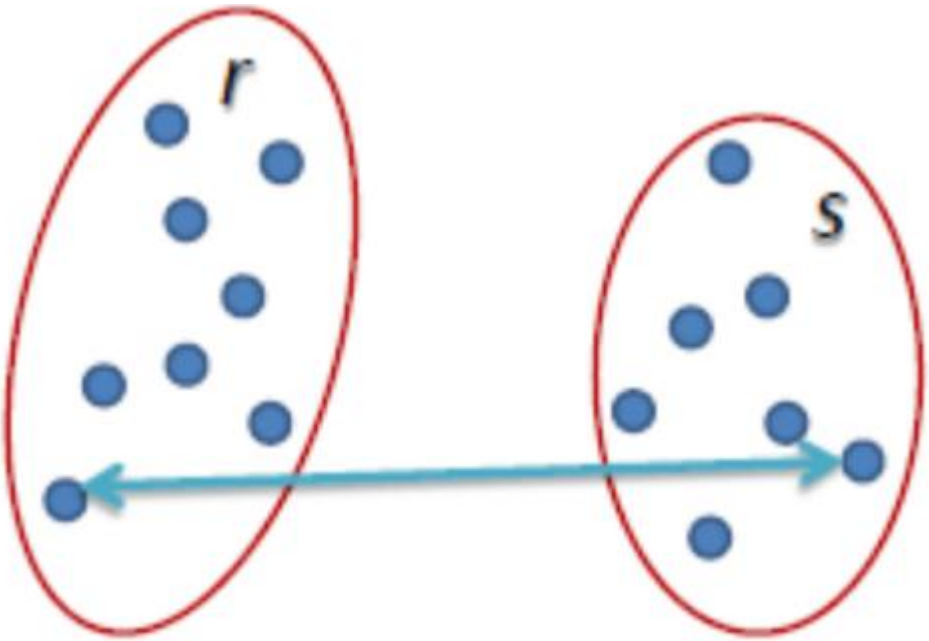
依据对相似度（距离）的不同定义，将Agglomerative Clustering的聚类方法分为三种：

1，SingleLinkage: 又叫做nearest-neighbor，其本质就是取两个集群中距离最近的两个样本的距离作为这两个集群的距离。这种方式有一个缺点，即明明两个集群相距甚远，但由于其中个别点的距离比较近而把他们计算成距离比较近。



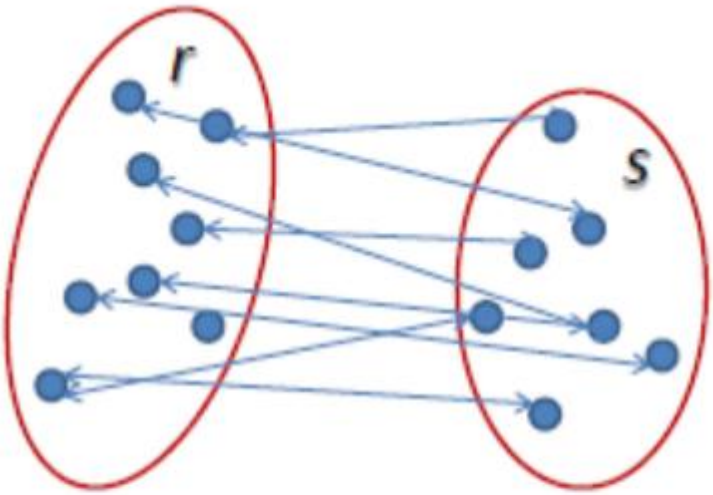

$$L(r,s)=\min(D(x_{ri},x_{sj}))$$

2，CompleteLinkage: 这种计算方式是上面的SingleLinkage算法的反面，即取两个集群中距离最远的两个点的距离作为这两个集群的距离，所以它的缺点也和上面的算法类似。




$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

3，AverageLinkage: 由于上面两种算法都存在一定的问题，都会被离群点带到沟里去，所以本算法就考虑整体的平均值，即把两个集群中的点两两的距离都计算出来后求平均值，作为两个集群的距离。（有的时候，并不是计算平均值，而是取中值，其背后的逻辑也是类似的，但取中值更加能够避免个别离群数据点对结果的干扰。）

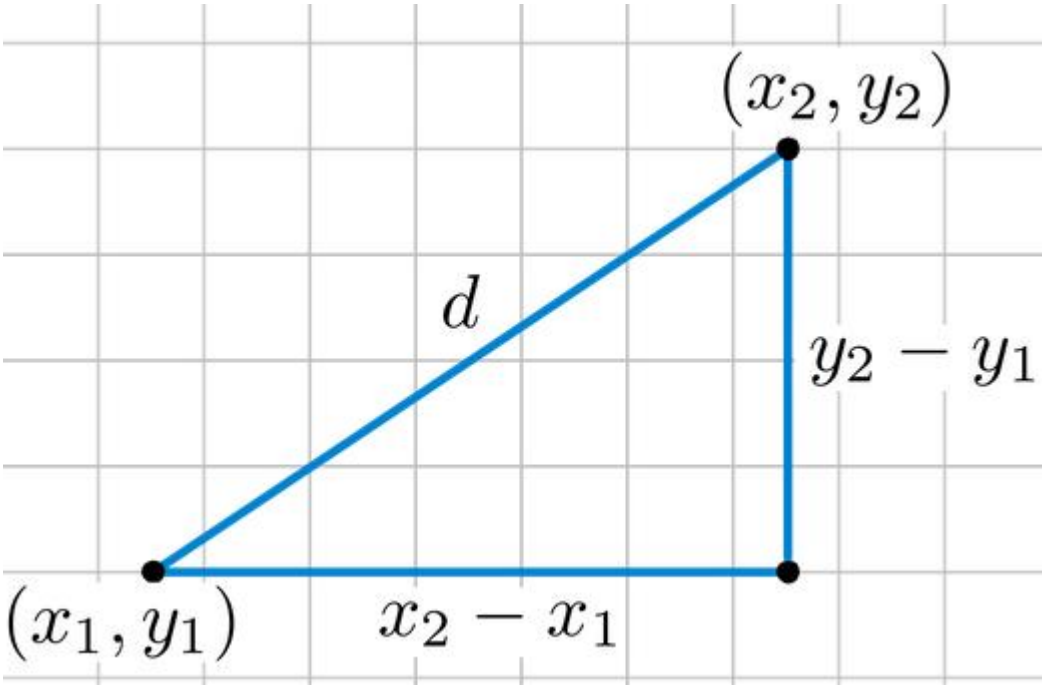



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

# 常见距离计算方式

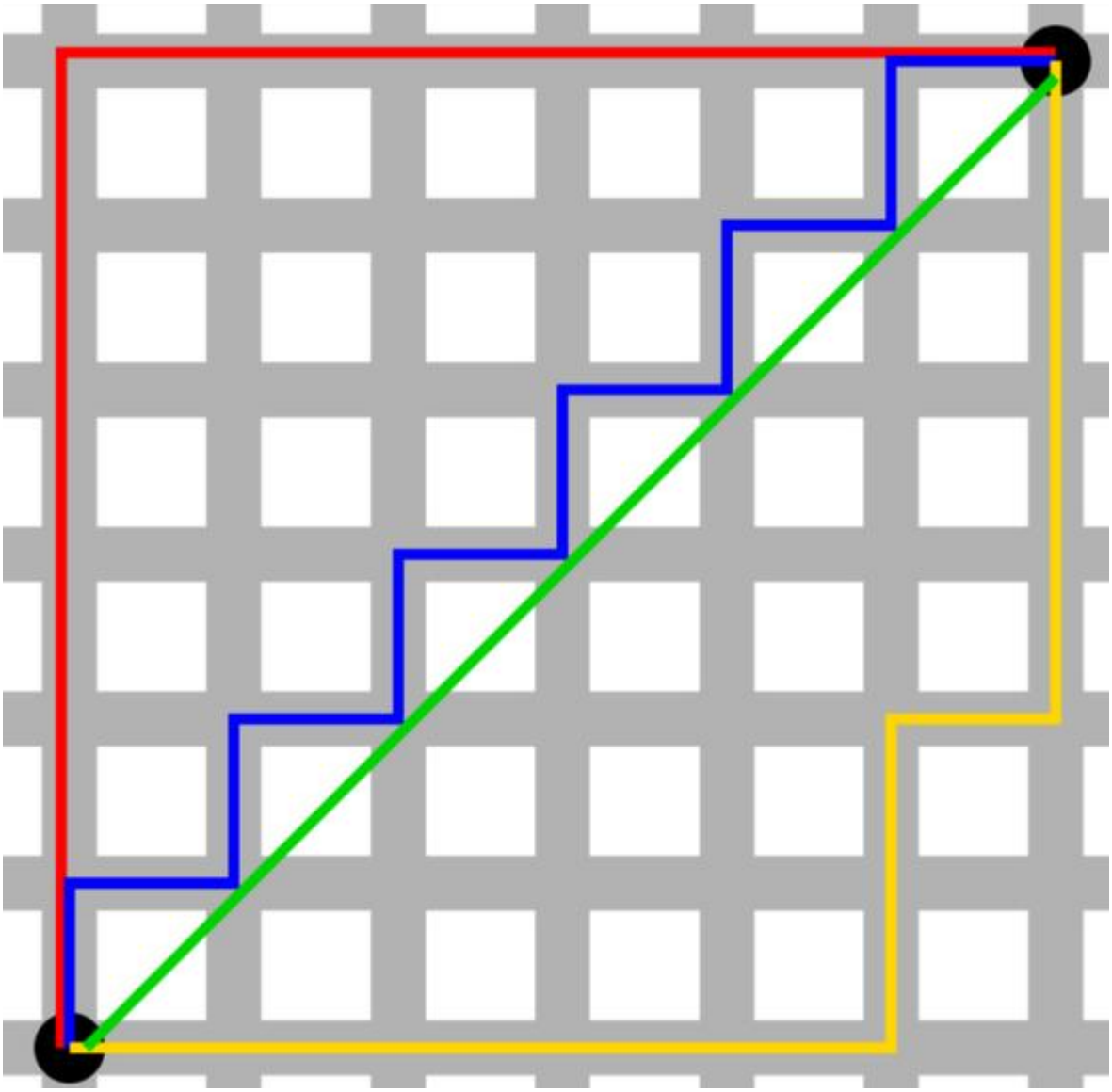
**欧几里德距离**

两点之间的最短距离。例如，如果x =（a，b）和y =（c，d），则x和y之间的欧几里德距离为√（（a-c）²+（b-d）²）



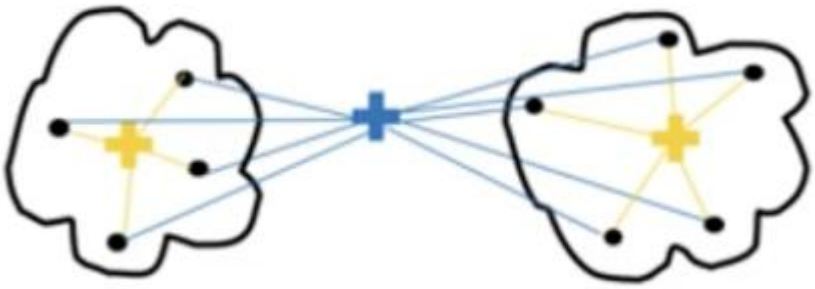

**曼哈顿距离**

想象一下，你在一个大城市的市中心，你想要从A点到达B点。你将无法跨越建筑物，而是你必须沿着各条街道行走。例如，如果x =（a，b）和y =（c，d），则x和y之间的曼哈顿距离是| a-c | + | b-d |。



**Ward Linkage（病房）**

聚类之间的距离是所有聚类内的平方差的总和

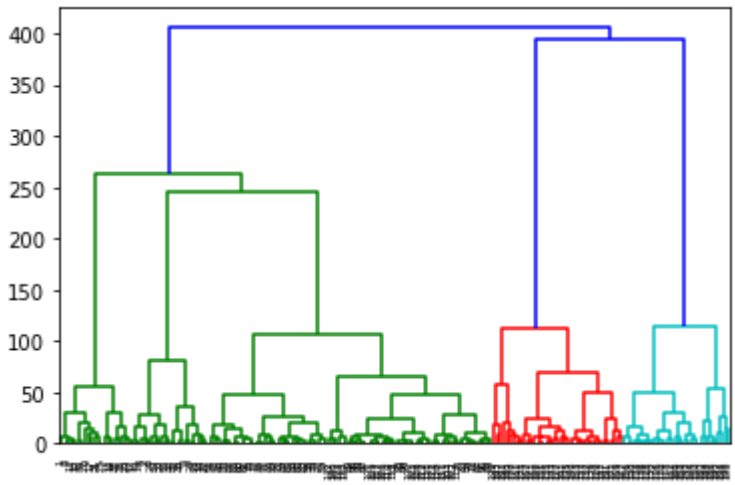


## 2. 准备数据集

```
#Numpy: 基础的数学计算模块,以矩阵为主,纯数学。
```

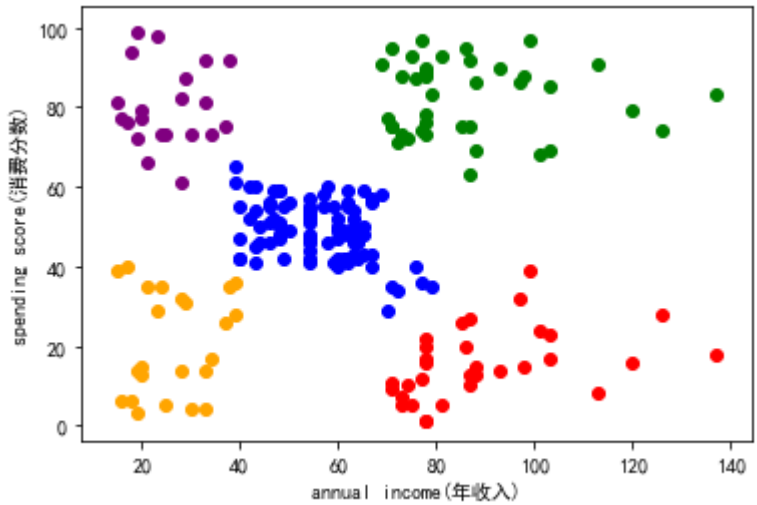
```
#SciPy: 基于Numpy,提供方法(函数库)直接计算结果,封装了一些高阶抽象和物理模型。
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering # 凝聚层次聚类
import scipy.cluster.hierarchy as sch # 层次模块，用于绘图
%matplotlib inline

#数据集： gender，age，annual income（年收入） 和 spending score（支出分数）
dataset = pd.read_csv('./023-Mall_Customers.csv')
print(dataset.info())
# 此处只采用两个变量（annual income 和spending score）
X = dataset.iloc[:, [3, 4]].values
# 指定类间链接算法
tree = sch.linkage(y=X, method='ward') #y：要计算距离的矩阵，method:计算类间距离的方法
# 生成树状图（dendrogram：树图）
dendrogram = sch.dendrogram(tree)
```




## 3. 使用凝聚层次聚类算法构建模型

```
#affinity：计算距离，取值“euclidean”，“manhattan”，如果linkage是“病房ward”，则只接受“欧几里德”。
#linkage：指定层次聚类判断相似度的方法，有以下几种：{“ward”，“complete”，“average”，“single”}，可选（默认=“ward”）
model = AgglomerativeClustering(n_clusters=4, affinity='euclidean', linkage='ward')
model.fit(X)
labels = model.labels_  #查看标签
# 数据可视化
plt.scatter(X[labels==0, 0], X[labels==0, 1], marker='o', color='red')
plt.scatter(X[labels==1, 0], X[labels==1, 1], marker='o', color='blue')
plt.scatter(X[labels==2, 0], X[labels==2, 1], marker='o', color='green')
plt.scatter(X[labels==3, 0], X[labels==3, 1], marker='o', color='purple')
plt.scatter(X[labels==4, 0], X[labels==4, 1], marker='o', color='orange')
plt.xlabel('annual income(年收入)')
plt.ylabel('spending score(消费分数)')
plt.show()
```




##########**小**********结**##########

**1，凝聚层次聚类算法的构建和训练比较简单，和其他聚类算法没有太大区别，直接构建AgglomerativeClustering对象并训练即可。**

**2，模型参数**

**3，树状图分析**

##########

参考资料:

1, Python机器学习经典实例，Prateek Joshi著，陶俊杰，陈小莉译