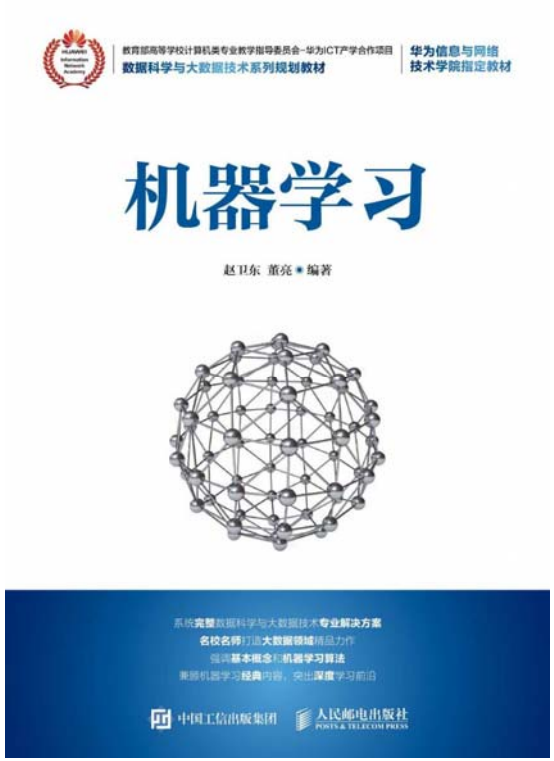


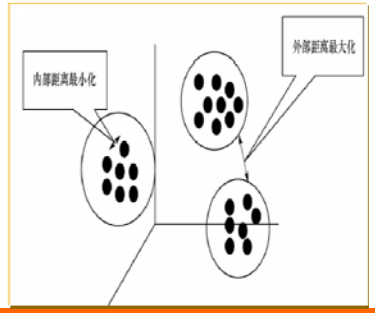
机器学习
聚类分析

复旦大学 **赵卫东** 博士

wdzhao@fudan.edu.cn

章节介绍

- 聚类分析是一种典型的无监督学习， 用于对未知类别的样本进行划分，将它们按照一定的规则划分成若干个类族，把相似(距高相近)的样本聚在同一个类簇中， 把不相似的样本分为不同类簇，从而揭示样本之间内在的性质以及相互之间的联系规律
- 聚类算法在银行、零售、保险、医学、军事等诸多领域有着广泛的应用
- 本章主要内容包括聚类分析基础、聚类效果评价指标、聚类实现方法，重点介绍基于划分的方法、基于密度的方法、基于层次的方法等方法，并结合实例讲解聚类算法的应用



章节结构

- 聚类分析概念
  - 聚类方法分类
  - 良好聚类算法的特征
- 聚类分析的度量
  - 外部指标
  - 内部指标
- 基于划分的聚类
  - k-均值算法、k-medoids算法、k-prototype算法
- 基于密度聚类
  - DBSCAN算法、OPTICS算法、DENCLUE算法
- 基于模型的聚类
  - 模糊聚类、Kohonen神经网络聚类

聚类分析概念

- 将未标记的样本自动划分成多个类簇
- 在销售领域，利用聚类分析对客户历史数据进行分析，对客户划分类别，刻画不同客户群体的特征，从而深入挖掘客户潜在需求，改善服务质量，增强客户黏性
- 在医学领域，对图像进行分析，挖掘疾病的不同临床特征，辅助医生进行临床诊断。聚类算法被用于图像分割，把原始图像分成若干个特定的、具有独特性质的区域并提取目标
- 在生物领域，将聚类算法用于推导动植物分类，以往对动植物的认知往往是基于外表和习性，应用聚类分析按照功能对基因聚类，获取不同种类物种之间的基因关联

聚类方法分类

- 基于划分的聚类
- 基于层次的聚类
- 基于密度的聚类
- 基于网格的聚类
- 基于模型的聚类

良好聚类算法的特征

- 良好的可伸缩性
- 处理不同类型数据的能力
- 处理噪声数据的能力
- 对样本顺序的不敏感性
- 约束条件下的表现
- 易解释性和易用性

聚类分析的度量

- 聚类分析的度量指标用于对聚类结果进行评判，分为内部指标和外部指标两大类
  - 外部指标指用事先指定的聚类模型作为参考来评判聚类结果的好坏
  - 内部指标是指不借助任何外部参考，只用参与聚类的样本评判聚类结果好坏
- 聚类的目标是得到较高的簇内相似度和较低的簇间相似度，使得簇间的距离尽可能大，簇内样本与簇中心的距离尽可能小
- 聚类得到的簇可以用聚类中心、簇大小、簇密度和簇描述等来表示
  - 聚类中心是一个簇中所有样本点的均值(质心)
  - 簇大小表示簇中所含样本的数量
  - 簇密度表示簇中样本点的紧密程度
  - 簇描述是簇中样本的业务特征

外部指标

- 对于含有n个样本点的数据集S，其中的两个不同样本点(x_i, x_j)，假设C是聚类算法给出的簇划分结果，P是外部参考模型给出的簇划分结果。那么对于样本点x_i, x_j来说，存在以下四种关系：
  - SS：x_i, x_j在C和P中属于相同的簇。
  - SD：x_i, x_j在C中属于相同的簇，在P中属于不同的簇。
  - DS：x_i, x_j在C中属于不同的簇，在P中属于相同的簇。
  - DD：x_i, x_j在C和P中属于不同的簇。
- 令a, b, c, d分别表示SS, SD, DS, DD所对应的关系数目，由于x_i, x_j之间的关系必定存在于四种关系中的一种，且仅能存在一种关系

外部指标

- Rand统计量（Rand Statistic）
  - $R = \frac{a+d}{a+b+c+d}$
- F值（F-measure）
  - $P = \frac{a}{a+b}$ $R = \frac{a}{a+c}$
  - P表示准确率，R表示召回率。
  - $F = \frac{(\beta^2+1)PR}{\beta^2 P+R}$
  - β是参数，当$\beta = 1$时，就是最常见的$F1 - measure$

外部指标

- Jaccard系数（Jaccard Coefficient）
  - $J = \frac{a}{a+b+c}$
- FM指数（Fowlkes and Mallows Index）
  - $FM = \sqrt{\frac{a}{a+b} \cdot \frac{a}{a+c}}$
- 以上四个度量指标的值越大，表明聚类结果和参考模型直接的划分结果越吻合，聚类结果就越好

内部指标

- 内部指标不借助外部参考模型，利用样本点和聚类中心之间的距离来衡量聚类结果的好坏。在聚类分析中，对于两个m维样本$x_i = (x_{i1}, x_{i2}, \dots, x_{im})$和$x_j = (x_{j1}, x_{j2}, \dots, x_{jm})$
- 常用的距离度量有欧式距离、曼哈顿距离、切比雪夫距离和明可夫斯基距离等
- 欧式距离（Euclidean Distance）是计算欧式空间中两点之间的距离，是最容易理解的距离计算方法，其计算公式如下：
  - $dist_{ed} = \sqrt{\sum_{k=1}^{m}(x_{ik} - x_{jk})^2}$

内部指标

- 曼哈顿距离（Manhattan Distance）也称城市街区距离，欧式距离表明了空间中两点间的直线距离，但是在城市中，两个地点之间的实际距离是要沿着道路行驶的距离，而不能计算直接穿过大楼的直线距离，曼哈顿距离就用于度量这样的实际行驶距离
  - $dist_{mand} = \sum_{k=1}^{m} |x_{ik} - x_{jk}|$
- 切比雪夫距离（Chebyshev Distance）是向量空间中的一种度量，将空间坐标中两个点的距离定义为其各坐标数值差绝对值的最大值。切比雪夫距离在国际象棋棋盘中，表示国王从一个格子移动到此外一个格子所走的步数
  - $dist_{cd} = \lim_{t \to \infty} \left(\sum_{k=1}^{m} |x_{ik} - x_{jk}|^t \right)^{\frac{1}{t}}$

内部指标

- 明可夫斯基距离（Minkowski Distance）是欧式空间的一种测度，是一组距离的定义，被看作是欧式距离和曼哈顿距离的一种推广
- $dist_{mind} = \sqrt[p]{\sum_{k=1}^{m}|x_{ik} - x_{jk}|^p}$
- 其中p是一个可变的参数，根据p取值的不同，明可夫斯基距离可以表示一类距离。当$p = 1$时，明可夫斯基距离就变成了曼哈顿距离；当$p = 2$时，明可夫斯基距离就变成了欧式距离；当$p \to \infty$时，明可夫斯基距离就变成了切比雪夫距离

内部指标

- 根据空间中点的距离度量，可以得出以下聚类性能度量内部指标
- 紧密度（Compactness）是每个簇中的样本点到聚类中心的平均距离。对于有n个样本点的簇C来说，该簇的紧密度为：$CP_c = \frac{1}{n}\sum_{i=1}^{n}\|x_i - C\|$，其中$w_c$为簇$c$的聚类中心
- 对于聚类结果，需要使用所有簇紧密度的平均值来衡量聚类结果的好坏，假设总共有k个簇：$CP = \frac{1}{k}\sum_{i=1}^{k} CP_i$，紧密度的值越小，表示簇内样本点的距离越近，即簇内样本的相似度越高

内部指标

- 分隔度（Seperation）是各簇的聚类中心c_i、c_j两两之间的平均距离，其计算公式如下：
- $SP = \frac{2}{k^2-k}\sum_{i=1}^{k}\sum_{j=i+1}^{k}\|c_i - c_j\|$， 分隔度的值越大，表示各聚类中心相互之间的距离越远，即簇间相似度越低

内部指标

- 戴维森堡丁指数（Davies-Bouldin Index，DBI）衡量任意两个簇的簇内距离之和与簇间距离之比，求最大值。首先定义簇中n个m维样本点之间的平均距离avg
- $$avg = \frac{2}{n(n-1)} \sum_{1 \le i < j \le n} \sqrt{\sum_{t=1}^{m} \left(x_{it} - x_{jt}\right)^2}$$
- 根据两个簇内样本间的平均距离，可以得出戴维森堡丁指数的计算公式如下，其中c_i、c_j表示簇C_i、C_j的聚类中心
- $$DBI = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{avg(C_i) + avg(C_j)}{\|c_i - c_j\|_2} \right)$$
- DBI的值越小，表示簇内样本之间的距离越小，同时簇间距离越大，即簇内相似度高，簇间相似度低，说明聚类结果越好

内部指标

- 邓恩指数（Dunn Validity Index，DVI）是计算任意两个簇的样本点的最短距离与任意簇中样本点的最大距离之商。假设聚类结果中有k个簇，其计算公式如下：

- $$DVI = \frac{\min\limits_{0<m\neq n<k}\left\{\min\limits_{\forall x_i\in C_m, x_j\in C_n}\{\|x_i-x_j\|\}\right\}}{\max\limits_{0<n\leq k}\max\limits_{\forall x_i,x_j\in C_n}\{\|x_i-x_j\|\}}$$

- DVI的值越大，表示簇间样本距离越远，簇内样本距离越近，即簇间相似度低，簇内相似度高，聚类结果越好。

基于划分的方法

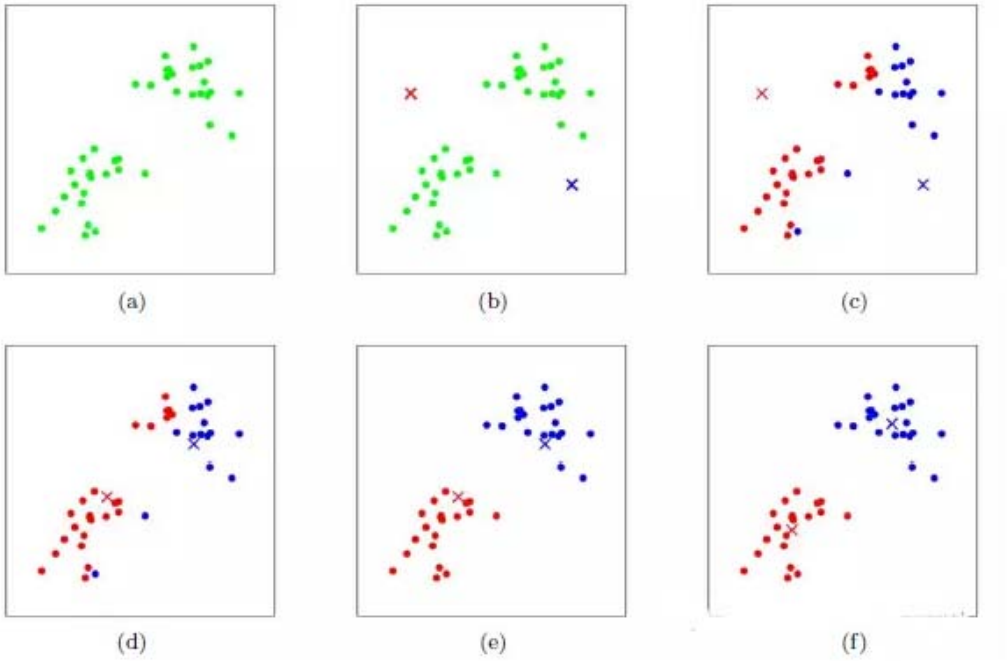
- 基于划分的方法是简单、常用的一种聚类方法
- 通过将对象划分为互斥的簇进行聚类， 每个对象属于且仅属于一个簇
- 划分结果旨在使簇之间的相似性低，簇内部的相似度高
- 基于划分的方法常用算法有k均值、k-medoids、k-prototype等

k-均值算法

- k-均值聚类是基于划分的聚类算法，计算样本点与类簇质心的距离，与类簇质心相近的样本点划分为同一类簇。k-均值通过样本间的距离来衡量它们之间的相似度，两个样本距离越远，则相似度越低，否则相似度越高
- k-均值算法聚类步骤如下：
  - 首先选取k个类簇（k需要用户进行指定）的质心，通常是随机选取。
  - 对剩余的每个样本点，计算它们到各个质心的欧式距离，并将其归入到相互间距离最小的质心所在的簇。计算各个新簇的质心。
  - 在所有样本点都划分完毕后，根据划分情况重新计算各个簇的质心所在位置，然后迭代计算各个样本点到各簇质心的距离，对所有样本点重新进行划分。
  - 重复第（2）步和第（3）步, 直到迭代计算后，所有样本点的划分情况保持不变，此时说明k-均值算法已经得到了最优解，将运行结果返回

k-均值算法

- k-均值聚类算法过程



k-均值算法

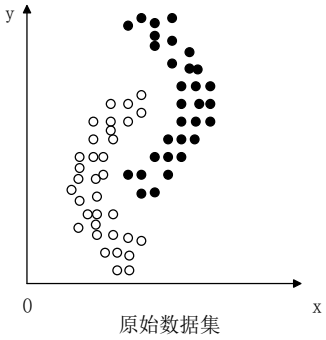
- k-均值算法原理简单，容易实现，且运行效率比较高
- k-均值算法聚类结果容易解释，适用于高维数据的聚类
- k-均值算法采用贪心策略，导致容易局部收敛，在大规模数据集上求解较慢
- k-均值算法对离群点和噪声点非常敏感，少量的离群点和噪声点可能对算法求平均值产生极大影响，从而影响聚类结果
- k-均值算法中初始聚类中心的选取也对算法结果影响很大，不同的初始中心可能会导致不同的聚类结果。对此，研究人员提出k-均值++算法，其思想是使初始的聚类中心之间的相互距离尽可能远

k-均值算法

- k-均值++算法步骤如下：
  - 从样本集χ中随机选择一个样本点c_1作为第1个聚类中心；
  - 计算其它样本点x到最近的聚类中心的距离$d(x)$；
  - 以概率$\frac{d(x)^2}{\sum_{x\in\chi} d(x)^2}$选择一个新样本点$c_i$加入聚类中心点集合中，其中距离值$d(x)$越大，被选中的可能性越高；
  - 重复步骤(2)和(3)选定k个聚类中心；
  - 基于这k个聚类中心进行k-均值运算

k-均值算法

- k-均值算法不适用于非凸面形状（非球形）的数据集，例如图中例子，k-均值算法的聚类结果就与初始目标有非常大的差别



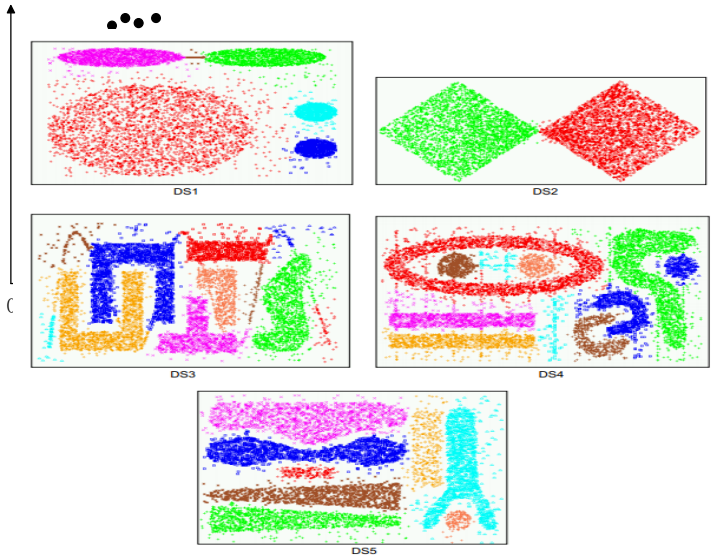
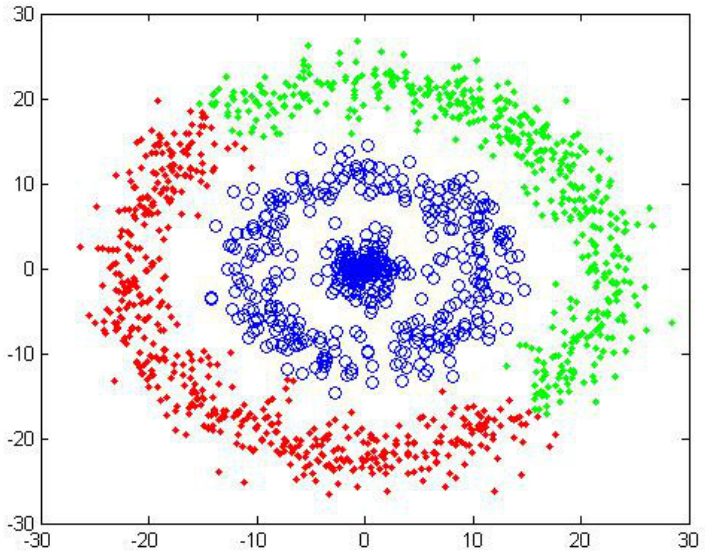


Figure 10: The clusters discovered by CHAMELEON for the five data sets.



k-均值算法

- 使k-均值聚类时，需要注意如下问题：
  - 模型的输入数据为数值型数据（如果是离散变量，需要作哑变量处理
  - 需要将原始数据作标准化处理（防止不同量纲对聚类产生影响）
- 对k值的选取，主要有以下几种：
  - 与层次聚类算法结合，先通过层次聚类算法得出大致的聚类数目，并且获得一个初始聚类结果，然后再通过k-均值算法改进聚类结果
  - 基于系统演化的方法，将数据集视为伪热力学系统，在分裂和合并过程中，将系统演化到稳定平衡状态从而确定k值

k-均值算法

- 利用sklearn库应用k-均值聚类算法实现对Iris数据集进行聚类。首先引用相应的库，其中sklearn.cluster为sklearn中已经实现的聚类算法工具包，代码如下

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
from sklearn import datasets
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
```

k-均值算法

- 首先，从IRIS数据集中加载鸢尾花样本信息到X和y两个变量中，其中，X存放花瓣长宽等特征，y存放花的类别标签。构造并初始化K-均值模型，设置类簇数量为3类，调用fit方法执行聚类，代码如下

```
np.random.seed(5)
iris = datasets.load_iris()
X = iris.data
y = iris.target
est = KMeans(n_clusters=3)
est.fit(X)
labels = est.labels_
```

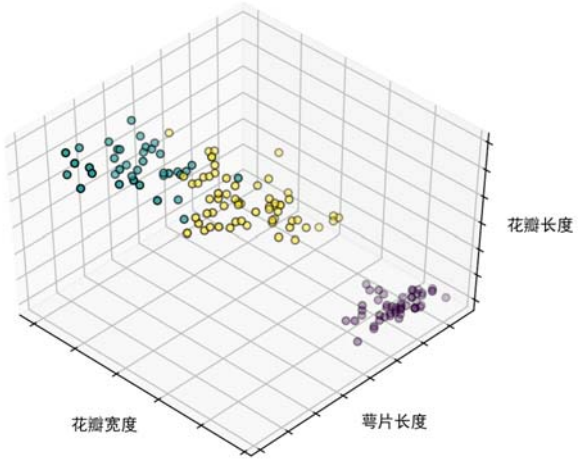
k-均值算法

- 接下来，对聚类的结果可视化显示，使用Axes3D将其显示在3维空间中，其中花瓣宽度、萼片长度、花瓣长度分别作为x,y,z三个维度

```
fig = plt.figure(1, figsize=(4, 3))
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=48, azim=134)
ax.scatter(X[:, 3], X[:, 0], X[:, 2],c=labels.astype(np.float), edgecolor='k')
ax.w_xaxis.set_ticklabels([])
ax.w_yaxis.set_ticklabels([])
ax.w_zaxis.set_ticklabels([])
ax.set_xlabel('花瓣宽度')
ax.set_ylabel('萼片长度')
ax.set_zlabel('花瓣长度')
ax.set_title("3类")
ax.dist = 12
plt.show()
```

k-均值算法

- k-均值对IRIS数据集聚类的效果



k-medoids算法

- k-均值算法簇的聚类中心选取受到噪声点的影响很大，因为噪声点与其他样本点的距离远，在计算距离时会严重影响簇的中心。k-medoids 算法克服了k-均值算法的这一缺点， k -medoids算法不通过计算簇中所有样本的平均值得到簇的中心，而是通过选取原有样本中的样本点作为代表对象代表这个簇，计算剩下的样本点与代表对象的距离，将样本点划分到与其距离最近的代表对象所在的簇中
- 距离计算过程与k均值算法的计算过程类似，只是将距离度量中的中心替换为代表对象，绝对误差标准如下
- $E = \sum_{i=1}^{k} \sum_{x^{(j)} \in C_i} \left\| x^{(j)} - o_c(i) \right\|^2$，式中$o_c(i)$表示第$i$簇$C_i$的中心，$x^{(j)}$表示$C_i$簇中的点，$E$最小表示最小化所有簇中点与点之间距离

k-medoids算法

- 围绕中心点划分(Partitioning Around Mediods, PAM) 算法是k -medoids聚类的一种典型实现。PAM 算法中簇的中心点是一个真实的样本点而不是通过距离计算出来的中心。PAM算法与k均值一样，使用贪心策略来处理聚类过程
- k-均值迭代计算簇的中心的过程，在PAM算法中对应计算是否替代对象o'比原来的代表对象o能够具有更好的聚类结果，替换后对所有样本点进行重新计算各自代表样本的绝对误差标准。若替换后，替换总代价小于0，即绝对误差标准减小，则说明替换后能够得到更好的聚类结果，若替换总代价大于0，则不能得到更好的聚类结果，原有代表对象不进行替换。在替换过程中，尝试所有可能的替换情况，用其他对象迭代替换代表对象，直到聚类的质量不能再被提高为止

k-prototype 算法

- k-prototype算法的聚类过程与k-均值算法相同，只是在聚类过程中引入参数γ来控制数值属性和分类属性的权重。对于m维样本$x_i = (x_{i1}^r, x_{i2}^r, \dots, x_{ip}^r, x_{i(p+1)}^c, \dots, x_{im}^c)$,其中标号为1至$p$下标的属性为数值型，$p+1$到$m$下标的属性为分类型。
- 定义样本与簇的距离为:$d(x_i, Q_l) = \sum_{j=1}^{p}(x_{ij}^r - q_{lj}^r)^2 + \gamma_l \sum_{j=p+1}^{m} \delta(x_{ij}^c, q_{lj}^c)$
- 其中x_{ij}^r和x_{ij}^c分别是x_i第j个属性的数值属性取值和分类属性取值，q_{lj}^r和q_{lj}^c分别是聚类l的原型Q_l的数值属性取值和分类属性取值，δ为符号函数

基于密度聚类

- 基于划分聚类和基于层次聚类的方法在聚类过程中根据距离来划分类簇，因此只能够用于挖掘球状簇。为了解决这一缺陷，基于密度聚类算法利用密度思想，将样本中的高密度区域(即样本点分布稠密的区域)划分为簇，将簇看作是样本空间中被稀疏区域(噪声)分隔开的稠密区域。这一算法的主要目的是过滤样本空间中的稀疏区域，获取稠密区域作为簇
- 基于密度的聚类算法是根据密度而不是距离来计算样本相似度，所以基于密度的聚类算法能够用于挖掘任意形状的簇，并且能够有效过滤掉噪声样本对于聚类结果的影响
- 常见的基于密度的聚类算法有DBSCAN、OPTICS和DENCLUE等。其中，OPTICS 对DBSCAN算法进行了改进，降低了对输入参数的敏感程度。DENCLUE算法综合了基于划分、基于层次的方法

DBSCAN 算法

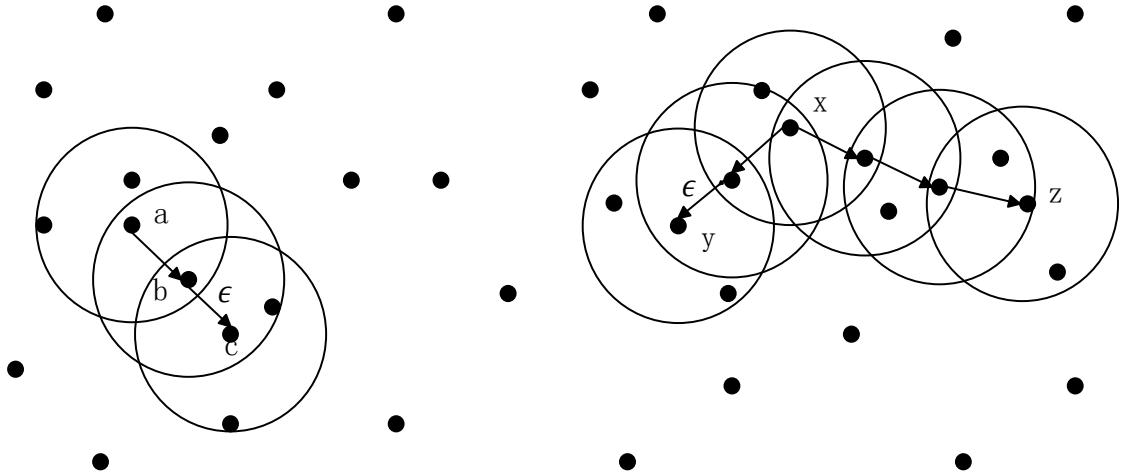
- DBSCAN采用基于中心的密度定义，样本的密度通过核心对象在ϵ半径内的样本点个数（包括自身）来估计。DBSCAN算法基于领域来描述样本的密度，输入样本集$S=\{x_1,x_2,\ldots,x_m\}$和参数$(\varepsilon,MinPts)$刻画邻域的样本分布密度。其中，ε表示样本的邻域距离阈值，$MinPts$表示对于某一样本p，其ϵ-邻域中样本个数的阈值。下面给出DBSCAN中的几个重要概念。
- **ϵ-邻域**：给定对象x_i，在半径ε内的区域称为x_i的ε-邻域。在该区域中，S的子样本集$N_\epsilon(x_i)=\{x_j\in S|distance(x_i,x_j)\leq\varepsilon\}$。
- 核心对象（core object）：如果对象$x_i\in S$，其ϵ-邻域对应的子样本集$N_\epsilon(x_i)$至少包含$MinPts$个样本，$|N_\epsilon(x_i)|\geq MinPts$,那么$x_i$为核心对象。

DBSCAN 算法

- 直接密度可达（directly density-reachable）：对于对象x_i和x_j，如果x_i是一个核心对象，且x_j在x_i的ε-邻域内，那么对象x_j是从x_i直接密度可达的。
- 密度可达（density-reachable）：对于对象x_i和x_j，若存在一个对象链$p_1, p_2, \ldots, p_n$，使得$p_1 = x_i, p_n = x_j$，并且对于$p_i(1 \leq i \leq n)$,p_{i+1}从p_i关于$(\varepsilon, MinPts)$直接密度可达，那么x_j是从x_i密度可达的。
- 密度相连（density-connected）：对于对象x_i和x_j，若存在x_k使得x_i和x_j是从x_k关于$(\varepsilon, MinPts)$密度可达，那么x_i和x_j是密度相连的。

DBSCAN 算法

- 在下图中，若$MinPts = 3$，则a、b、c和x、y、z都是核心对象，因为在各自的ε-邻域中，都至少包含3个对象。对象c是从对象b直接密度可达的，对象b是从对象a直接密度可达的，则对象c是从对象a密度可达的。对象y是从对象x密度可达的，对象z是从对象x密度可达的，则对象y和z是密度相连的



DBSCAN 算法

- DBSCAN算法根据密度可达关系求出所有密度相连样本的最大集合，将这些样本点作为同一个簇。DBSCAN算法任意选取一个核心对象作为“种子”，然后从“种子”出发寻找所有密度可达的其他核心对象，并且包含每个核心对象的ε-邻域的非核心对象，将这些核心对象和非核心对象作为一个簇。当寻找完成一个簇之后，选择还没有簇标记的其他核心对象，得到一个新的簇，反复执行这个过程，直到所有的核心对象都属于某一个簇为止。

DBSCAN 算法

- DBSCAN可以用于对任意形状的稠密数据集进行聚类，DBSCAN算法对输入顺序不敏感。DBSCAN能够在聚类的过程中发现数据集中的噪声点，且算法本身对噪声不敏感。当数据集分布为非球型时，使用DBSCAN算法效果较好
- DBSCAN算法要对数据集中的每个对象进行邻域检查，当数据集较大时， 聚类收敛时间长，需要较大的内存支持，I/O 消耗也很大，此时可以采用KD树或球树对算法进行改进，快速搜索最近邻，帮助算法快速收敛。此外，当空间聚类的密度不均匀，聚类间距离相差很大时，聚类的质量较差
- DBSCAN算法的聚类结果受到邻域参数$(\varepsilon, MinPts)$的影响较大，不同的输入参数对聚类结果有很大的影响，邻域参数也需要人工输入，调参时需要对两个参数联合调参，比较复杂

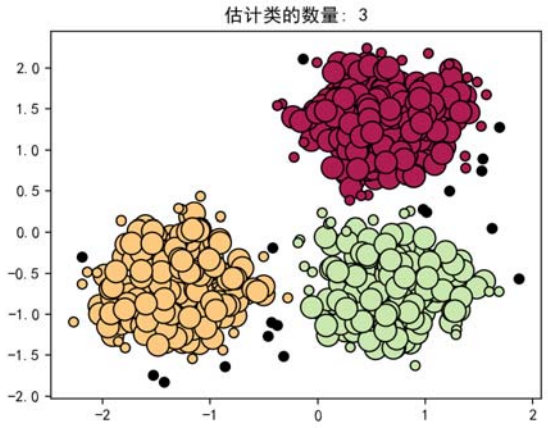
DBSCAN 算法

- 应用sklearn库中DBSCAN算法实现聚类。DBSCAN算法位于sklearn.cluster库中，数据源是用make_blobs方法随机生成的，数量为750条，有3个类簇。数据经过StandardScaler().fit_transform()对数据进行标准化处理，保证每个维度的方差为1，均值为0，使预测结果不会被某些维度过大的特征值而主导

```
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
centers = [[1, 1], [-1, -1], [1, -1]]
X,ltrue=make_blobs(n_samples=750,centers=centers,cluster_std=0.4,random_state=0)
X = StandardScaler().fit_transform(X)
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
```

DBSCAN 算法

- DBSCAN算法聚类效果

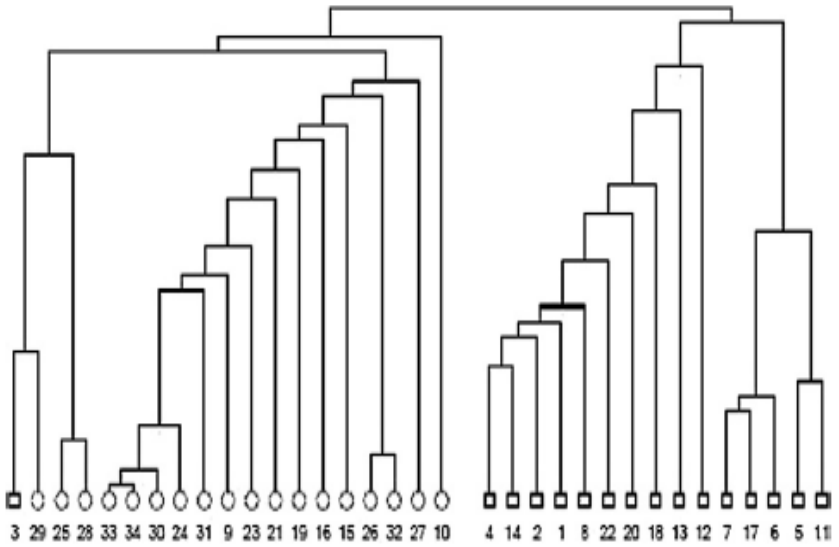


基于层次聚类

- 层次聚类的应用广泛程度仅次于基于划分的聚类，核心思想就是通过对数据集按照层次，把数据划分到不同层的簇，从而形成一个树形的聚类结构。层次聚类算法可以揭示数据的分层结构，在树形结构上不同层次进行划分，可以得到不同粒度的聚类结果。按照层次聚类的过程分为自底向上的聚合聚类和自顶向下的分裂聚类。聚合聚类以AGNES、BIRCH、ROCK等算法为代表，分裂聚类以DIANA算法为代表。
- 自底向上的聚合聚类将每个样本看作一个簇，初始状态下簇的数目等于样本的数目，然后根据算法的规则对样本进行合并，直到满足算法的终止条件。自顶向下的分裂聚类先将所有样本看作属于同一个簇，然后逐渐分裂成更小的簇，直到满足算法终止条件为止。目前大多数是自底向上的聚合聚类，自顶向下的分裂聚类比较少

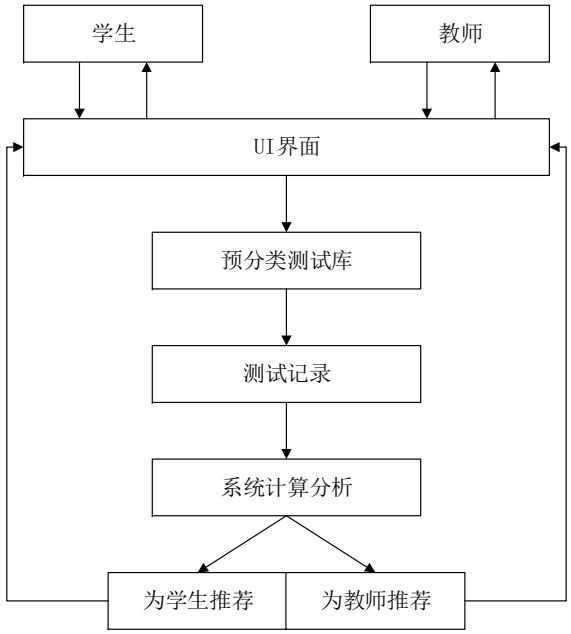
基于层次聚类

- 层次聚类（hierarchical clustering）方法把数据组织成若干簇，并形成一个相应的树状图进行聚类。
- 聚合层次聚类采用自底向上的策略，首先把每个对象单独作为一类，然后根据一定的规则，例如把簇间距离最小的相似簇合并成为越来越大的簇，直到所有样本凝聚成一个大的簇，针对给定应用选择最好结果的聚类层次。
- 与聚合型方法相反，分裂聚类采用自顶向下的方法，先把所有的对象都看成一个簇，然后不断分解直至满足一定的条件。



层次型聚类算法应用案例

基于层次聚类的个性化教学



层次型聚类算法应用案例

单个学生错题情况

问题	知识点			
	1	2	3	…
1		1		…
2	1			…
3		1		…
4			0	…
5		1		…
.	1			…
.			1	…
.	0			
总数	2	3	1	…

所有学生答题错误表

学生编号	知识点																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	7	18	19
1	1	0	2	2	2	2	1	1	1	1	1	2	1	1	2	1	3	2	2
2	2	1	2	2	3	1	2	2	1	1	2	3	2	1	1	1	1	1	1
3	1	2	3	2	2	2	2	2	2	3	1	2	3	1	3	1	2	1	2
4	2	2	2	2	2	1	2	1	3	2	1	2	2	2	2	1	2	0	1
5	1	3	3	0	2	2	2	2	1	2	1	2	2	2	1	1	1	1	1
6	1	2	2	2	1	1	3	3	1	2	1	1	2	2	1	1	1	1	2
7	2	0	2	2	2	3	2	2	2	2	2	2	0	0	2	1	1	1	2
8	1	2	3	2	2	1	2	2	1	0	2	2	1	2	1	2	1	2	2
9	1	2	2	2	2	1	2	3	3	2	2	2	1	0	2	1	1	1	1
10	2	1	1	3	2	2	3	2	1	2	2	2	1	3	1	3	2	1	1
11	1	1	1	2	2	3	2	2	1	2	1	1	2	1	3	2	2	1	1
12	2	0	3	3	2	0	1	2	2	2	2	2	1	2	1	1	2	1	1
13	1	2	2	2	1	1	1	2	1	2	1	2	1	1	2	3	1	2	2
14	1	3	2	0	1	2	2	2	1	1	0	1	1	2	2	3	1	1	3
15	1	3	2	2	2	1	2	3	2	2	2	2	2	0	1	1	1	2	1
16	0	1	1	1	1	1	1	1	3	2	2	3	1	0	2	2	1	2	2
17	1	2	1	2	2	1	2	2	2	3	1	1	2	2	2	2	0	0	1
18	2	1	2	2	0	1	2	2	2	2	1	2	0	2	1	2	1	0	1
19	1	2	2	1	1	2	2	1	1	3	1	1	1	2	2	2	2	2	1
20	1	2	2	2	2	2	1	2	2	2	3	2	1	2	2	1	2	1	1
21	1	1	3	2	2	2	2	2	2	1	2	1	3	1	2	1	3	2	1
22	3	2	1	2	1	2	1	0	2	2	2	2	3	2	2	1	1	2	0
23	2	2	2	2	1	2	1	1	2	2	2	2	2	2	2	1	1	1	1
24	2	1	2	2	2	3	2	2	2	2	2	1	1	1	2	2	2	1	1
25	2	1	1	2	2	0	3	2	1	1	1	1	1	1	1	2	2	2	1
错误总数	36	39	48	44	41	40	45	46	43	45	41	43	35	38	42	36	37	31	32

基于层次聚类的个性化教学

χ Clusters of students

Cluster	Students number
1	93002
	93011
	93025
	93034
	93045
2	93003
	93007
	93010
	93023
	93033
	93035
	93029
3	93006
	93012
	93018
;	;
;	
;	93009

层次型聚类算法应用案例

单个学生错题情况

问题	知识点			
	1	2	3	…
1		1		…
2	1			…
3		1		…
4			0	…
5		1		…
.	1			…
.			1	…
.	0			
总数	2	3	1	…

所有学生答题错误表

学生编号	知识点																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	7	18	19
1	1	0	2	2	2	2	1	1	1	1	1	2	1	1	2	1	3	2	2
2	2	1	2	2	3	1	2	2	1	1	2	3	2	1	1	1	1	1	1
3	1	2	3	2	2	2	2	2	2	3	1	2	3	1	3	1	2	1	2
4	2	2	2	2	2	1	2	1	3	2	1	2	2	2	2	1	2	0	1
5	1	3	3	0	2	2	2	2	1	2	1	2	2	2	1	1	1	1	1
6	1	2	2	2	1	1	3	3	1	2	1	1	2	2	1	1	1	1	2
7	2	0	2	2	2	3	2	2	2	2	2	2	0	0	2	1	1	1	2
8	1	2	3	2	2	1	2	2	1	0	2	2	1	2	1	2	1	2	2
9	1	2	2	2	2	1	2	3	3	2	2	2	1	0	2	1	1	1	1
10	2	1	1	3	2	2	3	2	1	2	2	2	1	3	1	3	2	1	1
11	1	1	1	2	2	3	2	2	1	2	1	1	2	1	3	2	2	1	1
12	2	0	3	3	2	0	1	2	2	2	2	2	1	2	1	1	2	1	1
13	1	2	2	2	1	1	1	2	1	2	1	2	1	1	2	3	1	2	2
14	1	3	2	0	1	2	2	2	1	1	0	1	1	2	2	3	1	1	3
15	1	3	2	2	2	1	2	3	2	2	2	2	2	0	1	1	1	2	1
16	0	1	1	1	1	1	1	1	3	2	2	3	1	0	2	2	1	2	2
17	1	2	1	2	2	1	2	2	2	3	1	1	2	2	2	2	0	0	1
18	2	1	2	2	0	1	2	2	2	2	1	2	0	2	1	2	1	0	1
19	1	2	2	1	1	2	2	1	1	3	1	1	1	2	2	2	2	2	1
20	1	2	2	2	2	2	1	2	2	2	3	2	1	2	2	1	2	1	1
21	1	1	3	2	2	2	2	2	2	1	2	1	3	1	2	1	3	2	1
22	3	2	1	2	1	2	1	0	2	2	2	2	3	2	2	1	1	2	0
23	2	2	2	2	1	2	1	1	2	2	2	2	2	2	2	1	1	1	1
24	2	1	2	2	2	3	2	2	2	2	2	1	1	1	2	2	2	1	1
25	2	1	1	2	2	0	3	2	1	1	1	1	1	1	1	2	2	2	1
错误总数	36	39	48	44	41	40	45	46	43	45	41	43	35	38	42	36	37	31	32

基于层次聚类的个性化教学

χ Clusters of students

Cluster	Students number
1	93002
	93011
	93025
	93034
	93045
2	93003
	93007
	93010
	93023
	93033
	93035
	93029
3	93006
	93012
	93018
;	;
;	
;	93009

层次型聚类算法应用案例

单个学生错题情况

问题	知识点			
	1	2	3	…
1		1		…
2	1			…
3		1		…
4			0	…
5		1		…
.	1			…
.			1	…
.	0			
总数	2	3	1	…

所有学生答题错误表

学生编号	知识点																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	7	18	19
1	1	0	2	2	2	2	1	1	1	1	1	2	1	1	2	1	3	2	2
2	2	1	2	2	3	1	2	2	1	1	2	3	2	1	1	1	1	1	1
3	1	2	3	2	2	2	2	2	2	3	1	2	3	1	3	1	2	1	2
4	2	2	2	2	2	1	2	1	3	2	1	2	2	2	2	1	2	0	1
5	1	3	3	0	2	2	2	2	1	2	1	2	2	2	1	1	1	1	1
6	1	2	2	2	1	1	3	3	1	2	1	1	2	2	1	1	1	1	2
7	2	0	2	2	2	3	2	2	2	2	2	2	0	0	2	1	1	1	2
8	1	2	3	2	2	1	2	2	1	0	2	2	1	2	1	2	1	2	2
9	1	2	2	2	2	1	2	3	3	2	2	2	1	0	2	1	1	1	1
10	2	1	1	3	2	2	3	2	1	2	2	2	1	3	1	3	2	1	1
11	1	1	1	2	2	3	2	2	1	2	1	1	2	1	3	2	2	1	1
12	2	0	3	3	2	0	1	2	2	2	2	2	1	2	1	1	2	1	1
13	1	2	2	2	1	1	1	2	1	2	1	2	1	1	2	3	1	2	2
14	1	3	2	0	1	2	2	2	1	1	0	1	1	2	2	3	1	1	3
15	1	3	2	2	2	1	2	3	2	2	2	2	2	0	1	1	1	2	1
16	0	1	1	1	1	1	1	1	3	2	2	3	1	0	2	2	1	2	2
17	1	2	1	2	2	1	2	2	2	3	1	1	2	2	2	2	0	0	1
18	2	1	2	2	0	1	2	2	2	2	1	2	0	2	1	2	1	0	1
19	1	2	2	1	1	2	2	1	1	3	1	1	1	2	2	2	2	2	1
20	1	2	2	2	2	2	1	2	2	2	3	2	1	2	2	1	2	1	1
21	1	1	3	2	2	2	2	2	2	1	2	1	3	1	2	1	3	2	1
22	3	2	1	2	1	2	1	0	2	2	2	2	3	2	2	1	1	2	0
23	2	2	2	2	1	2	1	1	2	2	2	2	2	2	2	1	1	1	1
24	2	1	2	2	2	3	2	2	2	2	2	1	1	1	2	2	2	1	1
25	2	1	1	2	2	0	3	2	1	1	1	1	1	1	1	2	2	2	1
错误总数	36	39	48	44	41	40	45	46	43	45	41	43	35	38	42	36	37	31	32

基于层次聚类的个性化教学

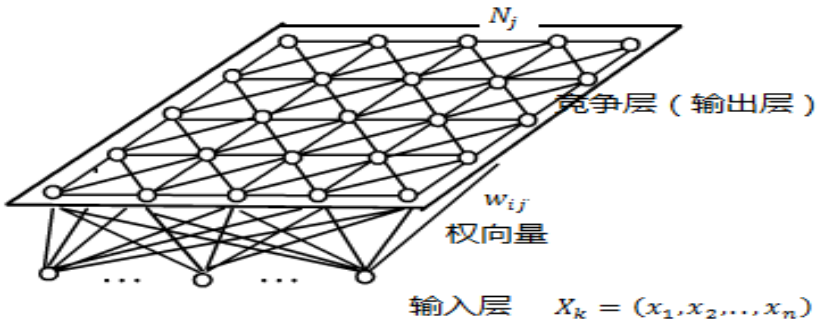
χ Clusters of students

Cluster	Students number
1	93002
	93011
	93025
	93034
	93045
2	93003
	93007
	93010
	93023
	93033
	93035
	93029
3	93006
	93012
	93018
;	;
;	
;	93009

基于模型的聚类

- 基于神经网络模型的聚类
- 基于概率模型的聚类

Kohonen神经网络聚类



- 在Kohonen神经网络的运行过程中，匹配竞争胜出的神经元及其邻近的神经元与相应输入层神经元之间的权向量朝着样本输入（特征）向量方向更新，如此经过多次迭代，这些权向量就可以对样本进行自动聚类，完成自组织学习(映射)过程。

Kohonen神经网络聚类

- Kohonen神经网络的具体执行过程如下：
  - 网络初始化。对输入层到输出层所有神经元的连接权值ω_{ij}赋予随机的小数作为初始值。
  - 计算连接向量。对网络的输入样本$X_k = (x_1^k, x_2^k, \dots, x_n^k)$，计算$X_k$与所有输出神经网络节点连接向量的距离。

$$d_j = \sum_{i=1}^{n} (x_i^k - \omega_{ij})^2, i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,m\}$$

  - 定义获胜单元。根据第（2）步距离计算结果，找到与输入向量距离最小的输出节点j^*，即$d_{j^*} = \min_{j \in \{1,2,\dots,m\}} \{d_j\}$
  - 在获胜单元的邻近区域，调整权重使其向输入向量靠拢。调整输出节点j^*连接的权值以及邻域NE_{j^*}内输出节点的连接权值$\Delta\omega_{ij} = \eta(x_i^k - \omega_{ij})$，$j \in NE_{j^*}$，其中$\eta$是学习因子，随着学习的进行，利用$\eta$逐渐减少权值调整的幅度。
  - 提供新样本进行训练，收缩邻域半径，减小学习率。重复上述步骤，直到小于阈值，输出聚类结果。

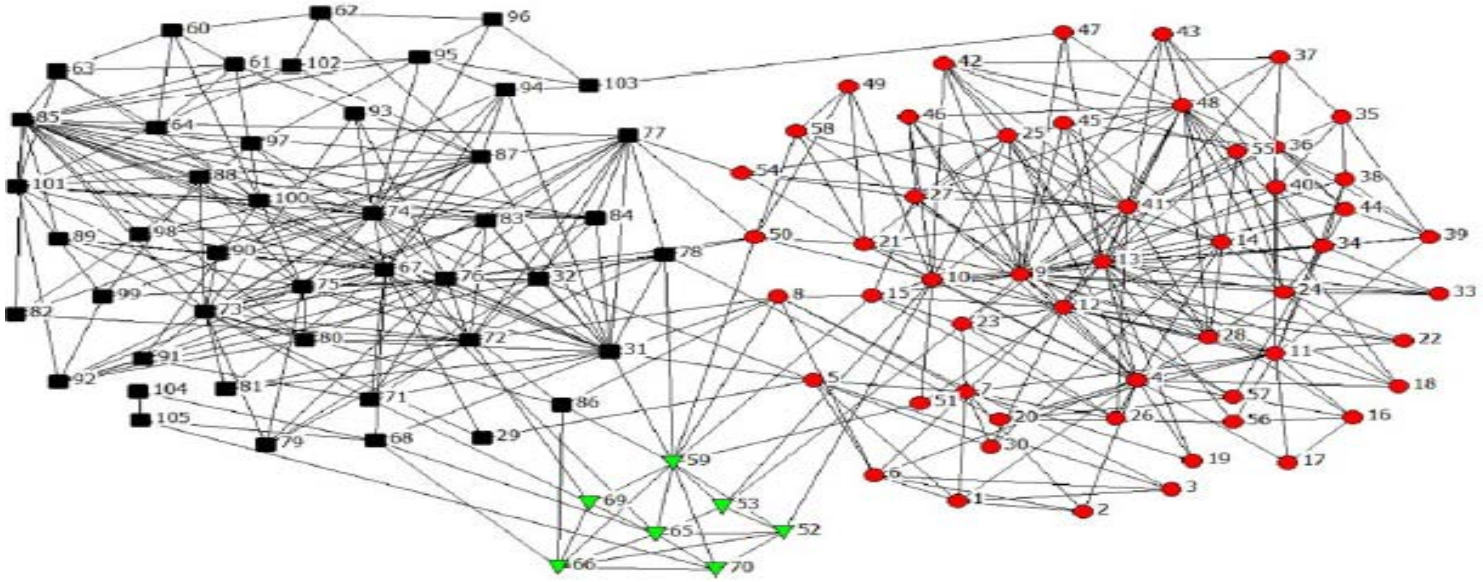
Kohonen神经网络聚类

- Kohonen神经网络的具体执行过程如下：
  - 网络初始化。对输入层到输出层所有神经元的连接权值ω_{ij}赋予随机的小数作为初始值。
  - 计算连接向量。对网络的输入样本$X_k = (x_1^k, x_2^k, \dots, x_n^k)$，计算$X_k$与所有输出神经网络节点连接向量的距离。

$$d_j = \sum_{i=1}^{n} (x_i^k - \omega_{ij})^2, i \in \{1,2,\dots,n\}, j \in \{1,2,\dots,m\}$$

  - 定义获胜单元。根据第（2）步距离计算结果，找到与输入向量距离最小的输出节点j^*，即$d_{j^*} = \min_{j \in \{1,2,\dots,m\}} \{d_j\}$
  - 在获胜单元的邻近区域，调整权重使其向输入向量靠拢。调整输出节点j^*连接的权值以及邻域NE_{j^*}内输出节点的连接权值$\Delta\omega_{ij} = \eta(x_i^k - \omega_{ij})$，$j \in NE_{j^*}$，其中$\eta$是学习因子，随着学习的进行，利用$\eta$逐渐减少权值调整的幅度。
  - 提供新样本进行训练，收缩邻域半径，减小学习率。重复上述步骤，直到小于阈值，输出聚类结果。

聚类是社会网络中的应用




THANK
YOU