第3章 聚类

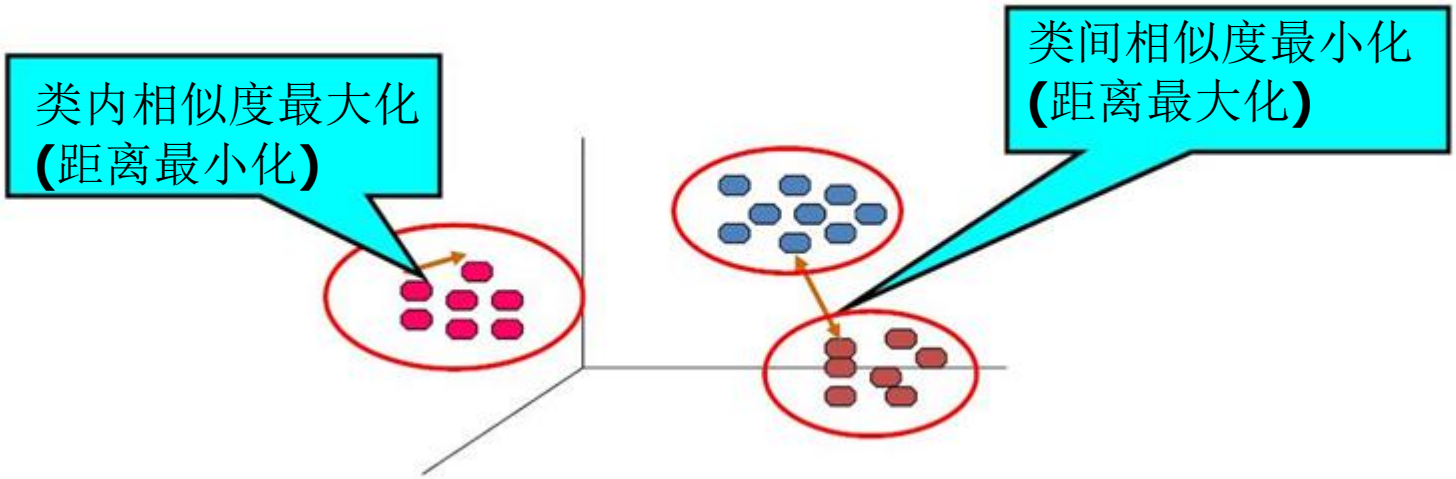
主要内容

1. 聚类分析概述
2. 相似性计算方法
3. 常用聚类方法
   - 3.1 划分方法
     - k-means算法（k-均值算法）
     - k-medoids算法（k-中心算法）
   - 3.2 层次方法
     - AGNES算法（合并聚类法）
     - DIANA算法（分裂聚类法）
   - 3.3 基于密度的聚类
     - DBSCAN算法

3.1聚类分析概述

在“无监督学习”任务中研究最多、应用最广

简单地描述，聚类(Clustering)是将数据集划分为若干相似对象组成的多个组(group)或簇(cluster)的过程，使得同一组中对象间的相似度最大化，不同组中对象间的相似度最小化。或者说一个簇(cluster)就是由彼此相似的一组对象所构成的集合，不同簇中的对象通常不相似或相似度很低。



聚类分析的定义

– 聚类分析(Cluster Analysis)是一个将数据集中的所有数据，按照相似性划分为多个类别（Cluster, 簇）的过程；

- 簇是相似数据的集合。

– 聚类分析是一种无监督(Unsupervised Learning)分类方法：数据集中的数据没有预定义的类别标号（无训练集和训练的过程）。

– 要求：聚类分析之后，应尽可能保证类别相同的数据之间具有较高的相似性，而类别不同的数据之间具有较低的相似性。

聚类分析在数据挖掘中的作用：

– 作为一个独立的工具来获得数据集中数据的分布情况；

– 作为其他数据挖掘算法的预处理步骤。

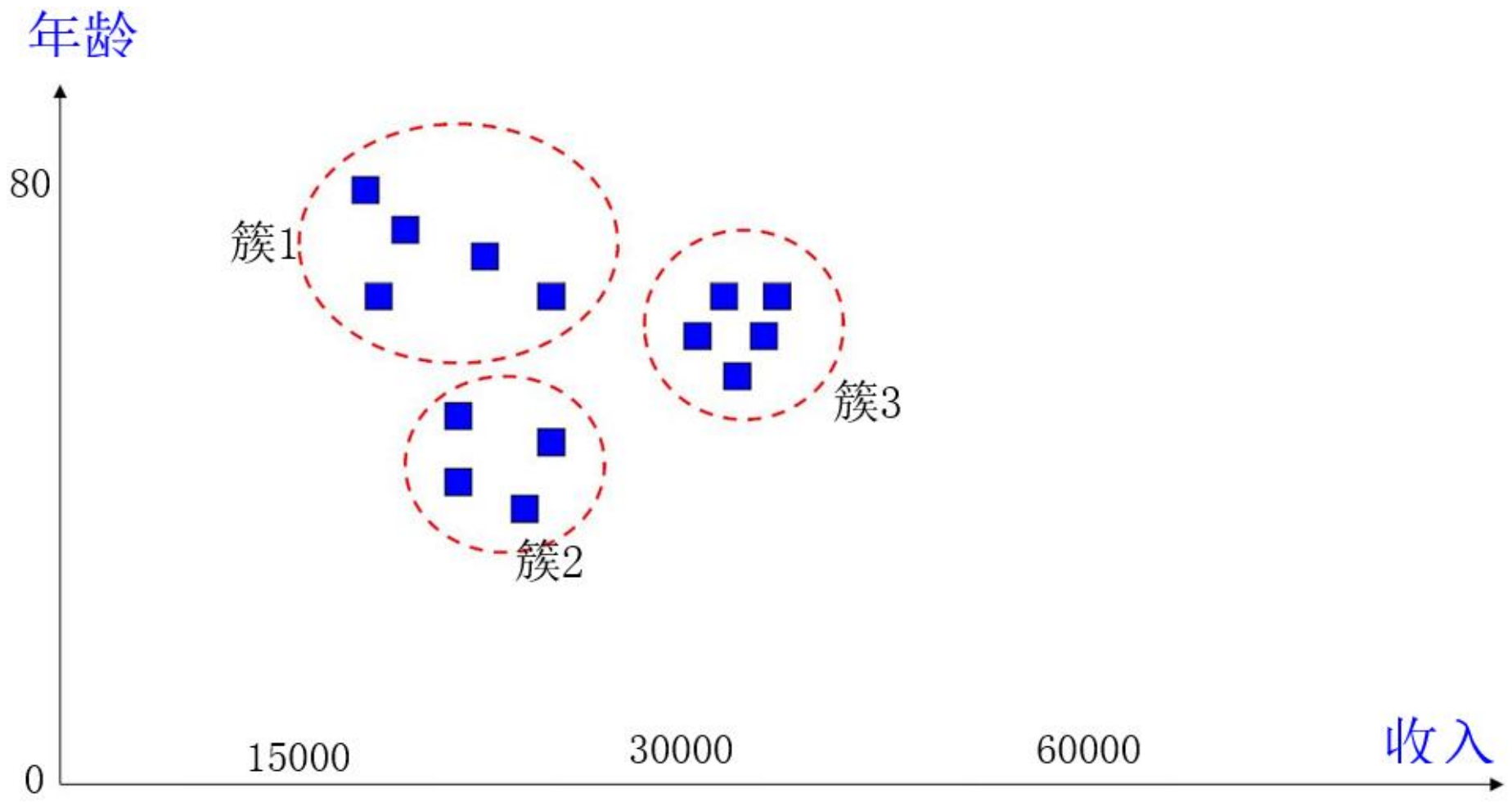
- **常用的聚类分析方法：**

  – 划分法（Partitioning Methods）：以距离作为数据集中不同数据间的相似性度量，将数据集划分成多个簇。

  – 层次法（Hierarchical Methods）：对给定的数据集进行层次分解，形成一个树形的聚类结果。

    - 属于这样的聚类方法有：自顶向下法、自底向上法。

划分示例



3.2相似性计算方法

在聚类分析中，样本之间的相似性通常采用样本之间的距离来表示。

– 两个样本之间的距离越大，表示两个样本越不相似性，差异性越大；

–两个样本之间的距离越小，表示两个样本越相似性，差异性越小。

– 特例：当两个样本之间的距离为零时，表示两个样本完全一样，无差异。

– 在不同应用领域，样本的描述属性的类型可能不同，因此相似性的计算方法也不尽相同。

- 连续型属性(如：重量、高度、年龄等)
- 二值离散型属性(如：性别、考试是否通过等)
- 多值离散型属性(如：收入分为高、中、低等)
- 混合类型属性(上述类型的属性至少同时存在两种)

1、连续型属性的相似性计算方法

假设两个样本X 和X 分别表示成如下形式：

– *Xi=(xi1, xi2, …, xid)*

– *Xj=(xj1, xj2, …, xjd)*

它们都是*d*维的特征向量，并且每维特征都是一个连续型数值。对于连续型属性，样本之间的相似性通常采用如下三种距离公式进行计算。

- 欧氏距离（Euclidean distance）

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2}$$

q=2

- 曼哈顿距离（Manhattan distance）

$$d(x_i, x_j) = \sum_{k=1}^{d} |x_{ik} - x_{jk}|$$

q=1

- 闵可夫斯基距离（Minkowski distance）

$$d(x_i, x_j) = (\sum_{k=1}^{d} |x_{ik} - x_{jk}|^q)^{1/q}$$

2、二值离散型属性的相似性计算方法

二值离散型属性只有0和1两个取值。

- 其中：0表示该属性为空，1表示该属性存在。
  - 例如：描述病人的是否抽烟的属性(smoker)，取值为1表示病人抽烟，取值0表示病人不抽烟。

假设两个样本Xi和Xj分别表示成如下形式：

– Xi=(xi1, xi2, …, xip)

– Xj=(xj1, xj2, …, xjp)

– 它们都是p维的特征向量，并且每维特征都是一个二值离散型数值。

假设二值离散型属性的两个取值具有相同的权重，则可以得到一个两行两列的可能性矩阵。

		X_j		
		1	0	*sum*
X_i	1	*a*	*b*	*a*+*b*
	0	*c*	*d*	*c*+*d*
	sum	*a*+*c*	*b*+*d*	*p*

– a = the number of attributes where Xi was 1 and Xj was 1;

– b = the number of attributes where Xi was 1 and Xj was 0;

– c = the number of attributes where Xi was 0 and Xj was 1;

– d = the number of attributes where Xi was 0 and Xj was 0.

如果样本的属性都是对称的二值离散型属性，则样本间的距离可用简单匹配系数(Simple Matching Coefficients, SMC)计算

$$SMC = (b + c) / (a + b + c + d)$$

– 其中：对称的二值离散型属性是指属性取值为1或者0同等重要。

– 例如：性别就是一个对称的二值离散型属性，即：用1表示男性，用0表示女性；或者用0表示男性，用1表示女性是等价的，属性的两个取值没有主次之分。

如果样本的属性都是不对称的二值离散型属性，则样本间的距离可用Jaccard系数计算(Jaccard Coefficients, JC)：

$$JC = (b + c) / (a + b + c)$$

– 其中：不对称的二值离散型属性是指属性取值为1或者0不是同等重要。

– 例如：血液的检查结果是不对称的二值离散型属性，阳性结果的重要程度高于阴性结果，因此通常用1来表示阳性结果，而用0来表示阴性结果。

例：已知两个样本p=[1 0 0 0 0 0 0 0 0 0]和q= [0 0 0 0 0 0 1 0 0 1]

– a = 0 (the number of attributes where p was 1 and q was 1)
– b = 1 (the number of attributes where p was 1 and q was 0)
– c = 2 (the number of attributes where p was 0 and q was 1)
– d = 7 (the number of attributes where p was 0 and q was 0)

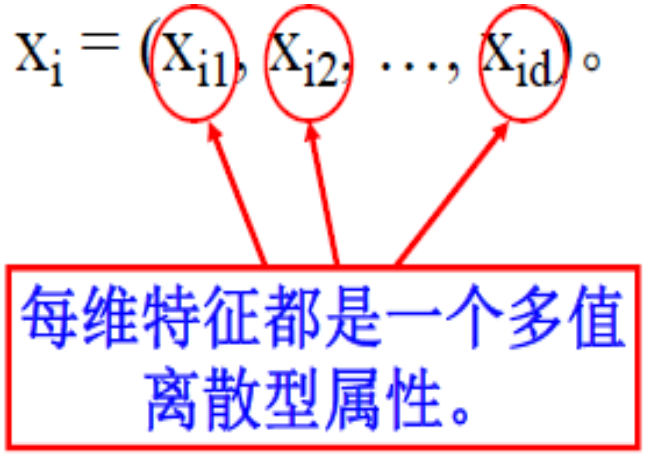
SMC = (b + c) / (a + b + c + d)
= (1+2) / (0+1+2+7)
= 0.3

JC = (b+c) / (a + b + c)
= (1+2) / (0+1+2)
= 1

3、多值离散型属性的相似性计算方法

- 多值离散型属性是指取值个数大于2的离散型属性。
  - 例如：成绩可以分为优、良、中、差。

- 假设一个多值离散型属性的取值个数为N，给定数据集X={xi | i=1,2,…,total}。
  - 其中：每个样本xi可用一个d维特征向量描述，并且每维特征都是一个多值离散型属性，即：



问题：给定两个样本xi = (xi1, xi2, …, xid)和xj= (xj1, xj2, …, xjd)，如何计算它们之间的距离？

- **方法一：简单匹配方法。**
  - 距离计算公式如下

$$d(x_i, x_j) = \frac{d - u}{d}$$

其中 d为数据集中的属性个数，u为样本xi和xj取值相同的属性个数。

样本序号	年龄段	学历	收入
x_1	青年	研究生	高
x_2	青年	本科	低
x_3	老年	本科以下	中
x_4	中年	研究生	高

– d(x1, x2) = (3-1)/3 ≈ 0.667

– d(x1, x3) = (3-0)/3 = 1

– d(x1, x4) = (3-2)/3 ≈ 0.333

方法二：先将多值离散型属性转换成多个二值离散型属性，然后再使用Jaccard系数计算样本之间的距离。

对有N个取值的多值离散型属性，可依据该属性的每种取值分别创建一个新的二值离散型属性，这样可将多值离散型属性转换成多个二值离散型属性。

样本序号	青年	中年	老年	本科以下	本科	研究生	高	中	低
x_1	0	0	1	0	0	1	1	0	0
x_2	0	0	1	0	1	0	0	0	1
x_3	1	0	0	1	0	0	0	1	0
x_4	0	1	0	0	0	1	1	0	0

4、混合类型属性的相似性计算方法

在实际中，数据集中数据的描述属性通常不只一种类型，而是各种类型的混合体。

- 连续型属性
- 二值离散型属性
- 多值离散型属性

问题：对于包含混合类型属性的数据集，如何计算样本之间的相似性？

方法：将混合类型属性放在一起处理，进行一次聚类分析。

– 假设给定的数据集X={xi | i=1,2,…,total}，每个样本用d个描述属性A1, A2, …, Ad来表示，属性Aj(1≤j≤d)包含多种类型。

– 在聚类之前，对样本的属性值进行预处理：

• 对连续型属性，将其各种取值进行规范化处理，使得属性值规范化到区间[0.0, 1.0]；

• 对多值离散型属性，根据属性的每种取值将其转换成多个二值离散型属性。

• 预处理之后，样本中只包含连续型属性和二值离散型属性

给定两个样本xi = (xi1, xi2, …, xid)和xj = (xj1,xj2, …, xjd) ，它们之间的距离为：

$$d(x_i, x_j) = \frac{\sum_{k=1}^{d} \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^{d} \delta_{ij}^{(k)}}$$

– 其中：$dij^{(k)}$表示xi和xj在第k个属性上的距离。

- 当第k个属性为连续型时，使用如下公式来计算$dij^{(k)}$：

$$dij^{(k)}=|xik-xjk|$$

- 当第k个属性为二值离散型时，如果xik=xjk，则$dij^{(k)}$ = 0;否则，$dij^{(k)}$ = 1。

$\delta_{ij}^{(k)}$表示第k个属性对计算xi和xj距离的影响.

（1）如果xik或xjk缺失（即：样本xi或样本xj没有第k个属性的度量值），则：$\delta_{ij}^{(k)}=0$。

（2）如果xik=xjk=0，且第k个属性是不对称的二值离散型，则$\delta_{ij}^{(k)}=0$。

（3）除了上述（1）和（2）之外的其他情况下，则$\delta_{ij}^{(k)}=1$。

3.3常用聚类方法

1、K-means

初始随机给定K个簇中心，按照最邻近原则把待分类样本点分到各个簇。然后按平均法重新计算各个簇的质心，从而确定新的簇心。一直迭代，直到簇心的移动距离小于某个给定的值。

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

Q1：K是什么？A1：k是聚类算法当中类的个数。
Q2：means是什么？A2：means是均值算法。
Summary：Kmeans是用均值算法把数据分成K个类的算法！

习题1:对下表中二维数据，使用k-means算法将其划分为2个簇，假设初始簇中心选为P7(4，5)，P10(5，5)。k-means聚类过程示例数据集:

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
x	3	3	7	4	3	8	4	4	7	5
y	4	6	3	7	8	5	5	1	4	5

解：(1)根据题目，假设划分的两个簇分别为C1和C2，中心分别为(4，5)和(5，5)，下面计算10个样本到这2个簇中心的距离，并将10个样本指派到与其最近的簇。

(2)第一轮迭代结果如下：

属于簇C1的样本有：{P7，P1，P2，P4，P5，P8}

属于簇C2的样本有：{P10，P3，P6，P9}

重新计算新的簇的中心，有：C1的中心为(3.5，5.167)，C2的中心为(6.75，4.25)

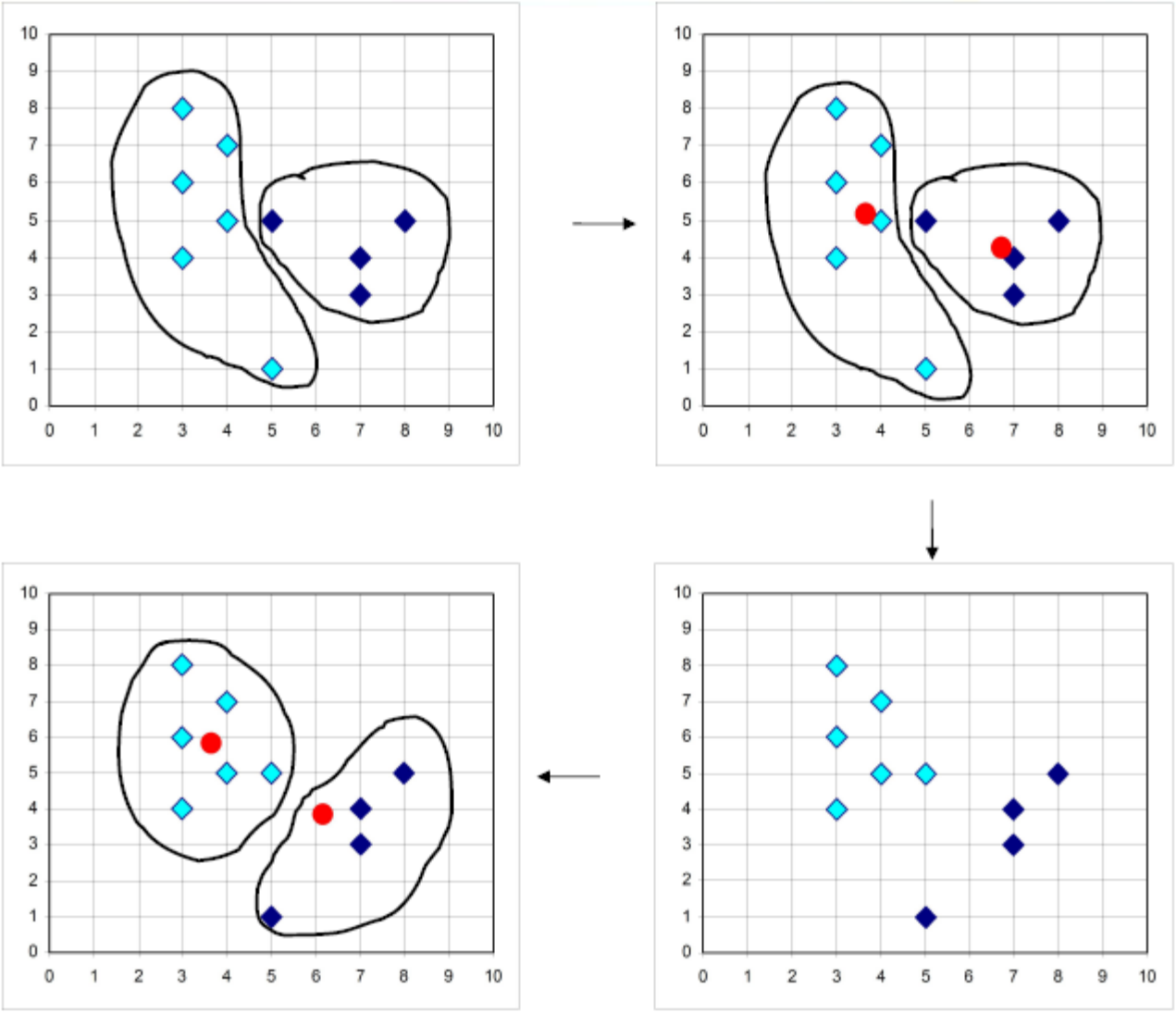
(3)继续计算10个样本到新的簇的中心的距离，重新分配到新的簇中，第二轮迭代结果如下：

属于簇C1的样本有：{ P1，P2，P4，P5，P7，P10}

属于簇C2的样本有：{ P3，P6，P8，P9}

重新计算新的簇的中心，有：C1的中心为(3.67，5.83)，C2的中心为(6.5，3.25)

(4)继续计算10个样本到新的簇的中心的距离，重新分配到新的簇中，发现簇中心不再发生变化，算法终止。



选择P1和P6进行计算，结果是否一样？

k-means算法的评价准则：误差平方和准则

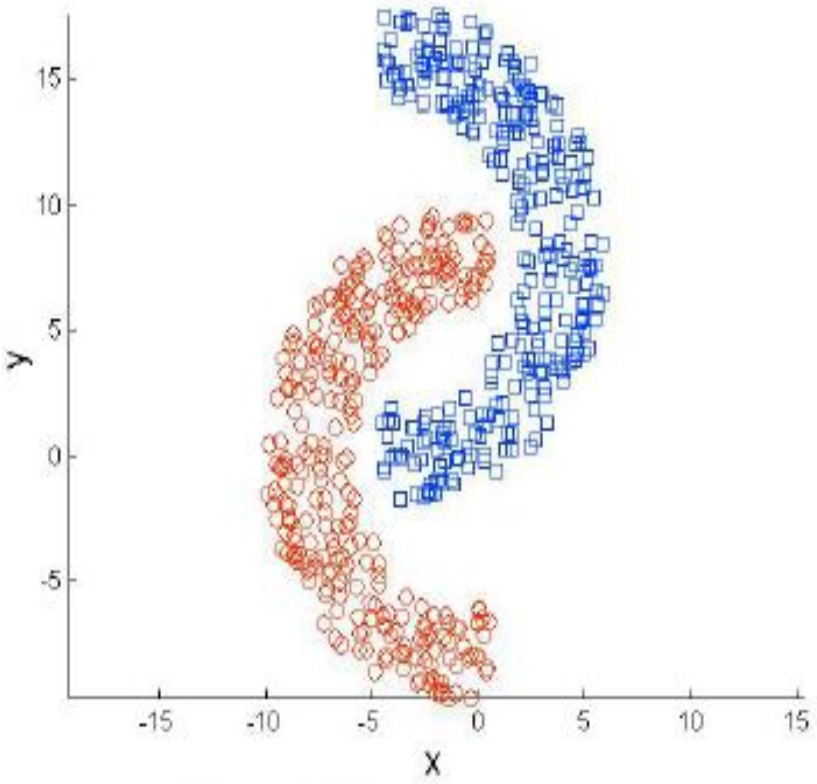
$$E = \sum_{i=1}^{K} \sum_{p \in C_i} | p - m_i |^2$$

– 误差平方和达到最优（小）时，可以使各聚类的类内尽可能紧凑，而使各聚类之间尽可能分开。

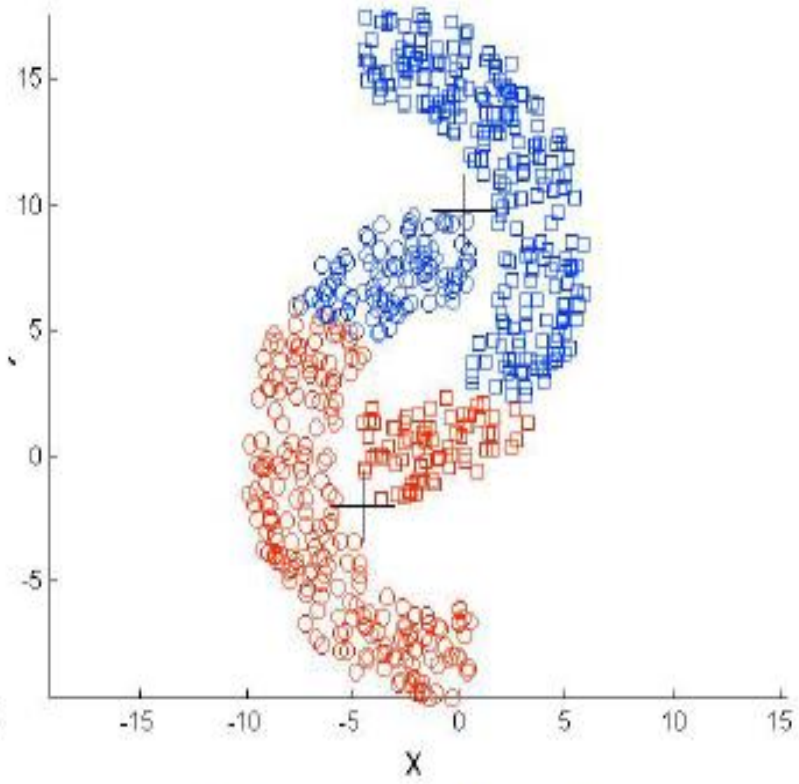
– 对于同一个数据集，由于k-means算法对初始选取的聚类中心敏感，因此可用该准则评价聚类结果的优劣。

– 通常，对于任意一个数据集，k-means算法无法达到全局最优，只能达到局部最优。

K-means算法无法探测凹型簇



Original Points

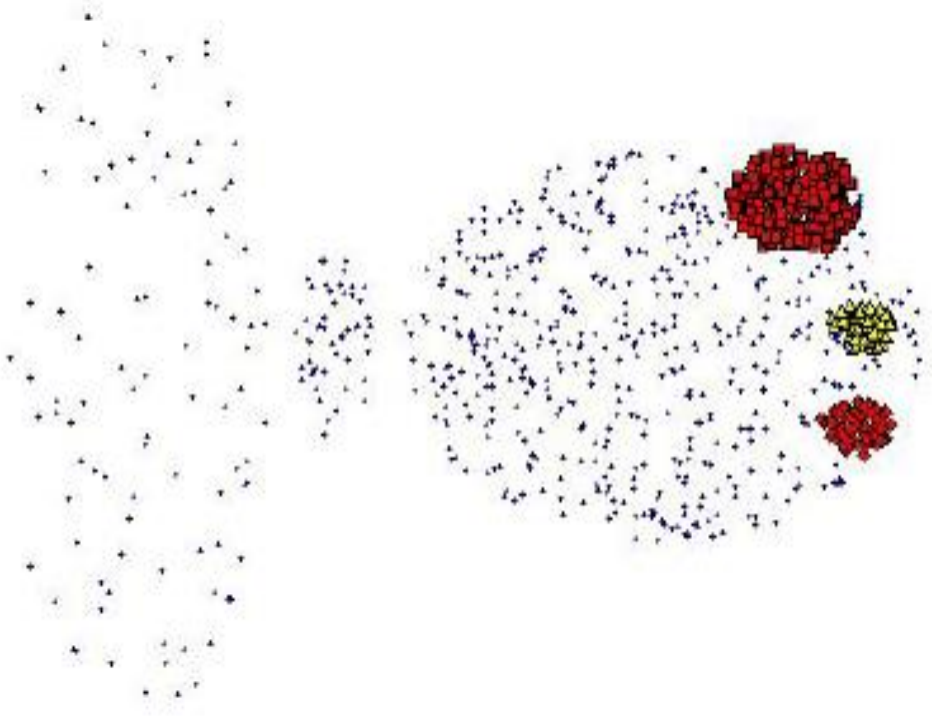


K-means (2 Clusters)

- 优点：
  - 可扩展性较好，算法复杂度为*O(nkt)*。
    - 其中：n为样本个数，k是簇的个数，t是迭代次数。
- 缺点：
  - 簇数目k需要事先给定，但非常难以选定；
  - 初始聚类中心的选择对聚类结果有较大的影响；
  - 不适合于发现非球状簇；
  - 对噪声和离群点数据敏感。

2、DBSCAN

DBSCAN是一个基于密度的聚类算法.(他聚类方法大都是基于对象之间的距离进行聚类，聚类结果是球状的簇)，
基于密度的聚类是寻找被低密度区域分离的高密度区域。

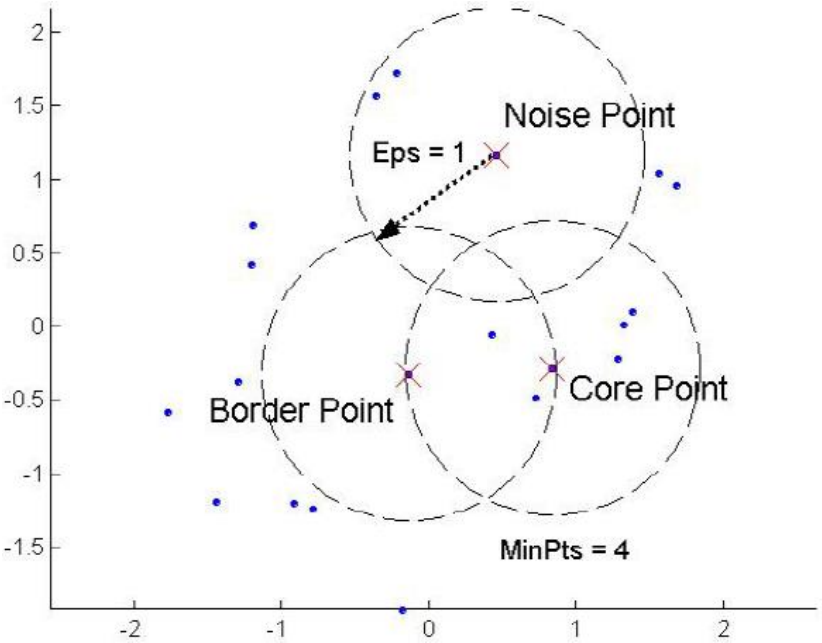


DBSCAN基于密度定义，我们将点分为：

– 稠密区域内部的点(核心点)：
在半径Eps内含有超过MinPts数目的点，则**该点**为核心点，这些点都是在簇内的

– 稠密区域边缘上的点(边界点)：
在半径Eps内点的数量小于MinPts，但是在核心点的邻居

– 稀疏区域中的点(噪声或背景点)：
任何不是核心点或边界点的点..



DBSCAN算法概念

Eps邻域：给定对象半径Eps内的邻域称为该对象的Eps邻域，我们用$N_{EPS}(p)$ 表示点p的Eps-半径内的点的集合，即:
$N_{EPS}(p)=\{q|q$在数据集D中，$distance(p,q)\leq Eps\}$

核心对象：如果对象的Eps邻域至少包含最小数目MinPts的对象，则称该对象为核心对象。

边界点：边界点不是核心点，但落在某个核心点的邻域内。

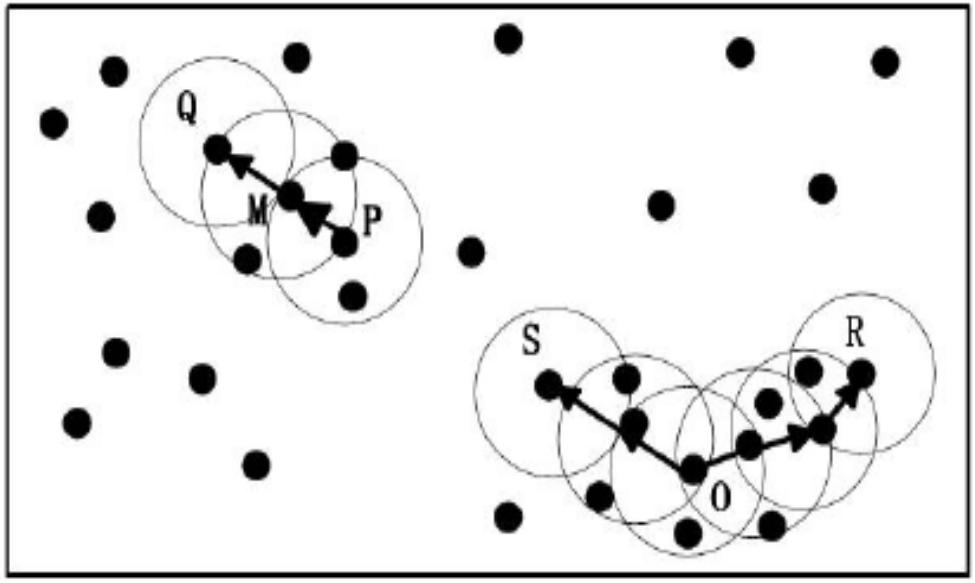
噪音点：既不是核心点，也不是边界点的任何点

直接密度可达：给定一个对象集合D，如果p在q的Eps邻域内，而q是一个核心对象，则称对象p从对象q出发时是直接密度可达的(directly density-reachable)。

密度可达：如果存在一个对象链p1,p2,…pn，p1=q，pn=p，对于pi∈D(1≤i≤n)，pi+1是从pi关于Eps和MinPts直接密度可达的，则对象p是从对象q关于Eps和MinPts密度可达的(density-reachable)。

密度相连：如果存在对象O∈D，使对象p和q都是从O关于Eps和MinPts密度可达的，那么对象p到q是关于Eps和MinPts密度相连的(density connected).

如图所示，Eps用一个相应的半径表示，设MinPts=3，请分析Q,M,P,S,O,R这5个样本点之间的关系。



解答：根据以上概念知道：由于有标记的各点M、P、O和R的Eps近邻均包含3个以上的点，因此它们都是核对象；M是从P“直接密度可达”；而Q则是从M“直接密度可达”；基于上述结果，Q是从P“密度可达”；但P从Q无法“密度可达”(非对称)。类似地，S和R从O是“密度可达”的；O、R和S均是“密度相连”的。

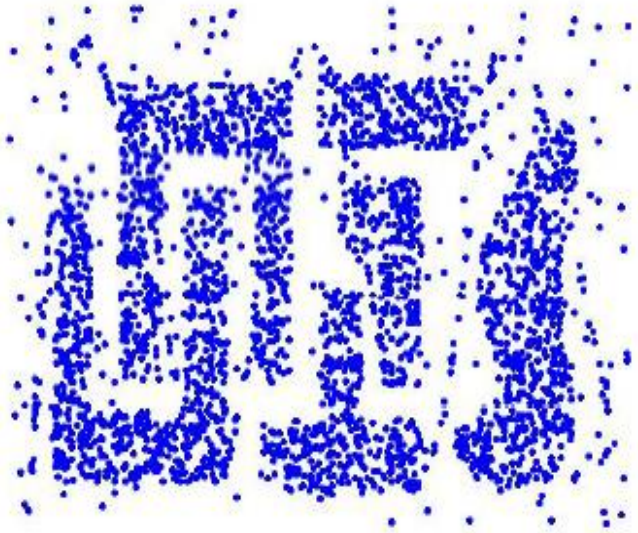
DBSCAN算法原理

DBSCAN通过检查数据集中每点的Eps邻域来搜索簇，如果点p的Eps邻域包含的点多于MinPts个，则创建一个以p为核心对象的簇。

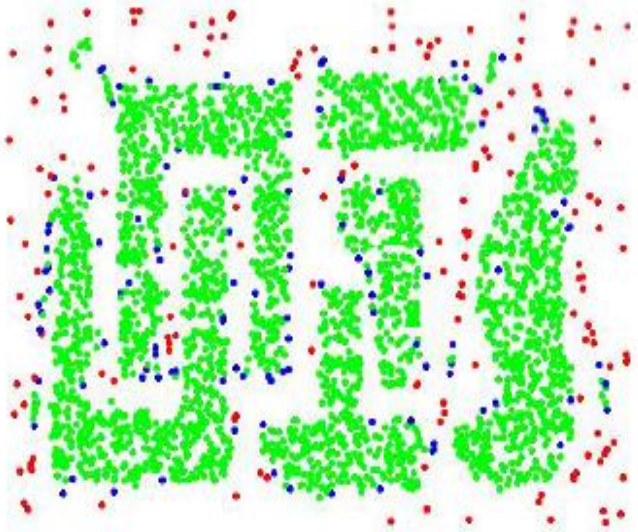
然后，DBSCAN迭代地聚集从这些核心对象直接密度可达的对象，这个过程可能涉及一些密度可达簇的合并。

当没有新的点添加到任何簇时，该过程结束。

DBSCAN运行效果



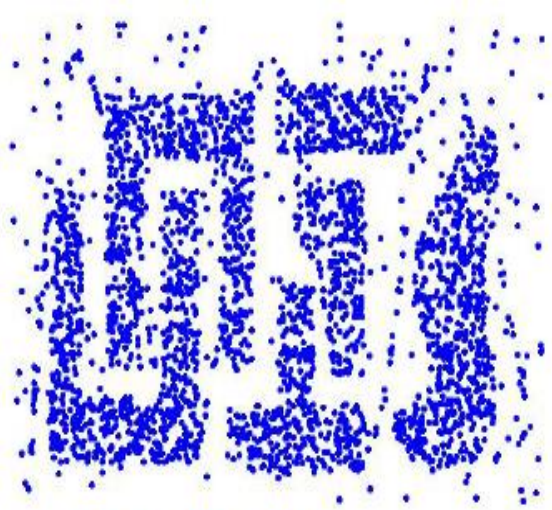
Original Points



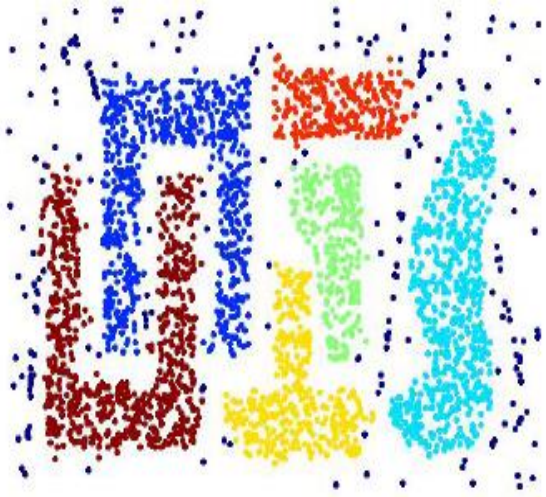
Point types: core, border and noise

Eps = 10, MinPts = 4

DBSCAN运行效果好的时候



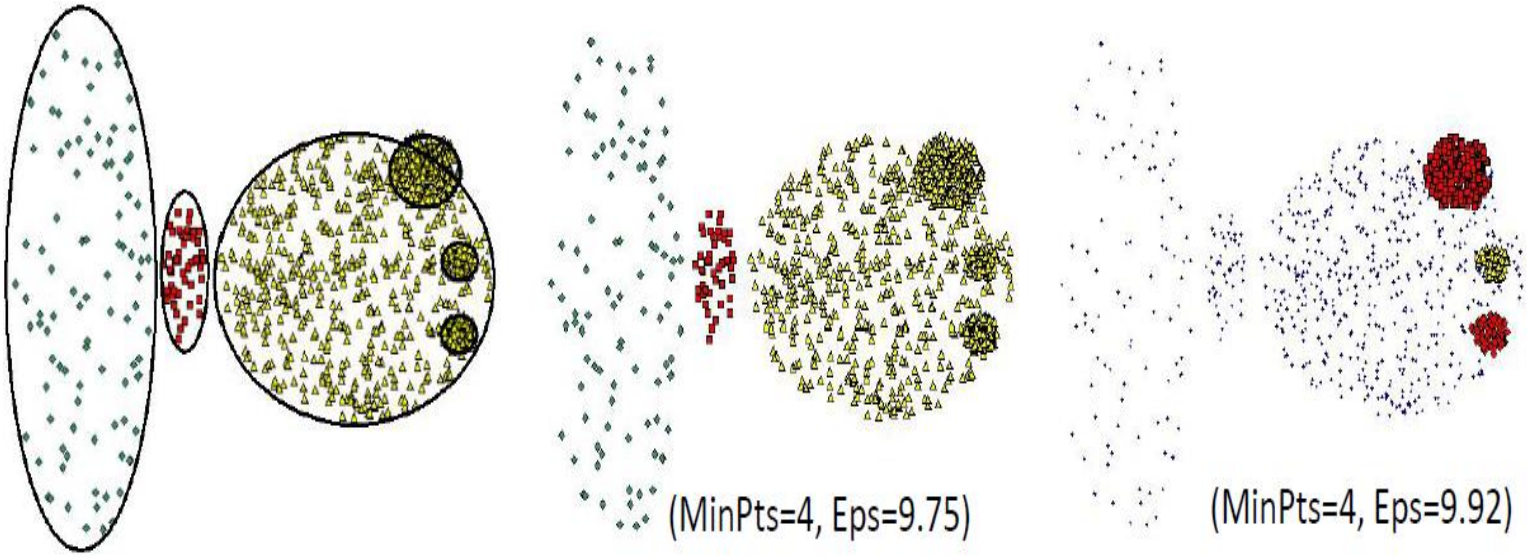
Original Points



Clusters

对噪音不敏感；可以处理不同形状和大小的数据

DBSCAN运行不好的效果：密度变化的数据；高维数据



DBSCAN的其它问题

时间复杂度

– DBSCAN的基本时间复杂度是 O(N*找出Eps领域中的点所需要的时间), N是点的个数。最坏情况下时间复杂度是$O(N^2)$

– 在低维空间数据中,有一些数据结构如KD树，使得可以有效的检索特定点给定距离内的所有点，时间复杂度可以降低到O(NlogN)

空间复杂度

– 低维或高维数据中，其空间都是O(N)，对于每个点它只需要维持少量数据，即簇标号和每个点的标识(核心点或边界点或噪音点)

如何合适选取EPS和MinPts

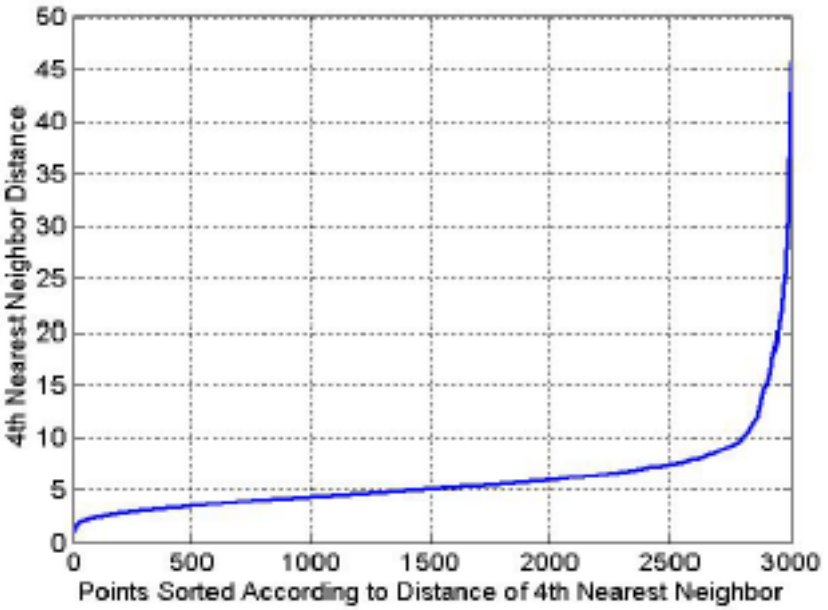
思想是这样的对于在一个类中的所有点,它们的第k个最近邻大概距离是一样的

- 噪声点的第k个最近邻的距离比较远
- 所以尝试根据每个点和它的第k个最近邻之间的距离来选取

然后

– Eps取什么？

– MinPts取什么？



- 优点
  - 基于密度定义，相对抗噪音，能处理任意形状和大小的簇
- 缺点
  - 当簇的密度变化太大时，会有麻烦
  - 对于高维问题，密度定义是个比较麻烦的问题