回顾

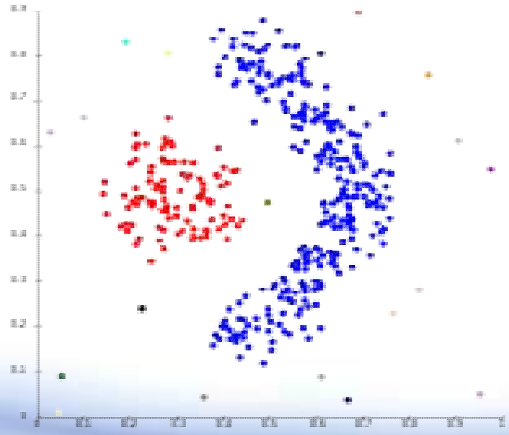
聚类

聚类(clustering)也称为聚类分析,指将样本分到不同的组中使得同一组中的样本差异尽可能的小，而不同组中的样本差异尽可能的大。

聚类得到的不同的组称为簇(cluster)。

一个好的聚类方法将产生以下的聚类

- 最大化类中的相似性
- 最小化类间的相似性



回顾

聚类的分类：

- 划分聚类方法
- 层次聚类方法
- 密度聚类方法
- 网格聚类方法
- 模型聚类方法

划分聚类方法

在基于划分的聚类中，任务就是将数据划分成K个不相交的点集，使每个子集中的点尽可能同质。

基于划分的方法，其代表算法有 **k-means**算法、K-medoids等

k-means 算法

- k-means 算法基本步骤

1. 从 n个数据对象任意选择 k 个对象作为初始聚类中心；
2. 根据每个聚类对象的均值(中心对象)，计算每个对象与这些中心对象的距离；并根据最小距离重新对相应对象进行划分；
3. 重新计算每个(有变化)聚类的均值(中心对象)；
4. 计算标准测度函数，当满足一定条件，如函数收敛时，则算法终止；如果条件不满足则回到步骤2。

k-means优缺点

- 主要优点：
  - 是解决聚类问题的一种经典算法，简单、快速。
  - 对处理大数据集，该算法是相对可伸缩和高效率的。
  - 当结果簇是密集的，它的效果较好。
- 主要缺点
  - 在簇的平均值被定义的情况下才能使用。
  - 必须事先给出k（要生成的簇的数目），而且对初值敏感，对于不同的初始值，可能会导致不同结果。
  - 不适合于发现非凸面形状的簇或者大小差别很大的簇。而且，它对于“躁声”和孤立点数据是敏感的。

层次聚类方法

- 层次聚类方法对给定的数据集进行层次的分解，直到某种条件满足为止。具体又可分为：

  凝聚的层次聚类：一种自底向上的策略，首先将每个对象作为一个簇，然后合并这些原子簇为越来越大的簇，直到某个终结条件被满足。

  分裂的层次聚类：采用自顶向下的策略，它首先将所有对象置于一个簇中，然后逐渐细分为越来越小的簇，直到达到了某个终结条件。

- 层次凝聚的代表是AGNES算法。层次分裂的代表是DIANA算法。

层次聚类优缺点

- 层次聚类方法是不可逆的，也就是说，当通过凝聚式的方法将两组合并后，无法通过分裂式的办法再将其分离到之前的状态，反之亦然。
- 另外，层次聚类过程中调查者必须决定聚类在什么时候停止，以得到某个数量的分类。
- 在不必要的情况下应该小心使用层次聚类方法。

密度聚类方法

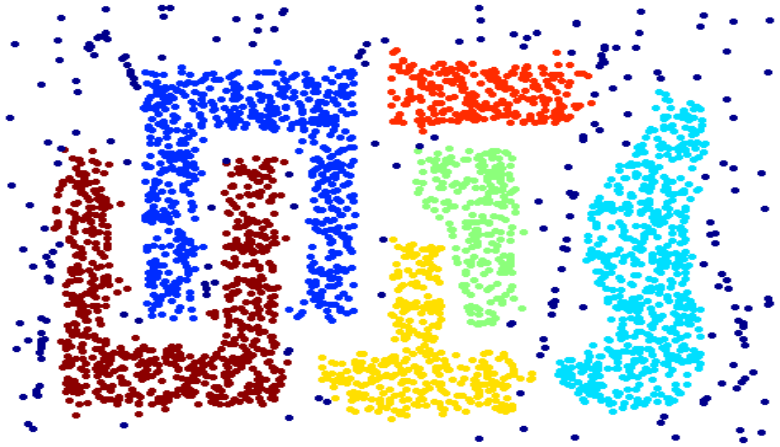
- 划分聚类方法
- 层次聚类方法
- 密度聚类方法 ：基于密度的聚类方法以数据集在空间分布上的稠密程度为依据进行聚类，无需预先设定簇的数量，因此特别适合对于未知内容的数据集进行聚类。
- 网格聚类方法
- 模型聚类方法

基于密度方法的聚类

- 密度聚类方法的指导思想是，只要一个区域中的点的密度大于某个域值，就把它加到与之相近的聚类中去。对于簇中每个对象，在给定的半径ε的邻域中至少要包含最小数数目（MinPts）个对象。
- 这类算法能克服基于距离的算法只能发现“类圆形”的聚类的缺点，可发现任意形状的聚类，且对噪声数据不敏感。
- 代表算法有：DBSCAN、OPTICS、DENCLUE算法等。

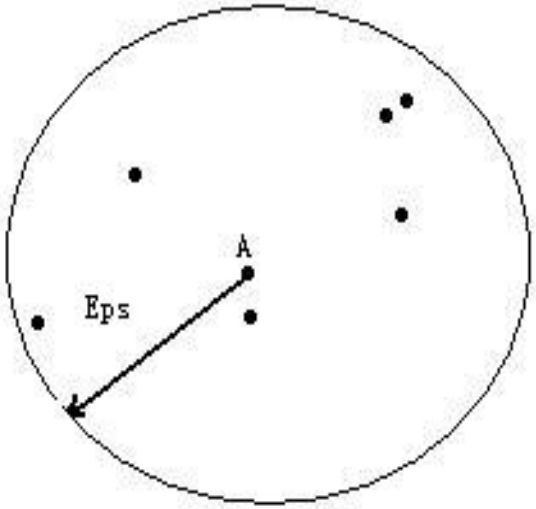
基于密度方法的聚类- DBSCAN

- DBSCAN（Density-Based Spatial Clustering of Applications with Noise）一个比较有代表性的基于密度的聚类算法。与层次聚类方法不同，它将簇定义为密度相连的点的最大集合，能够把具有足够高密度的区域划分为簇，并可在有“噪声”的空间数据库中发现任意形状的聚类。



传统的密度定义：基于中心的方法

- 传统基于中心的密度定义为：
  数据集中特定点的密度通过该点ε半径之内的点计数(包括本身)来估计。
  显然，密度依赖于半径。



基于密度方法的聚类- DBSCAN 所用到的基本术语

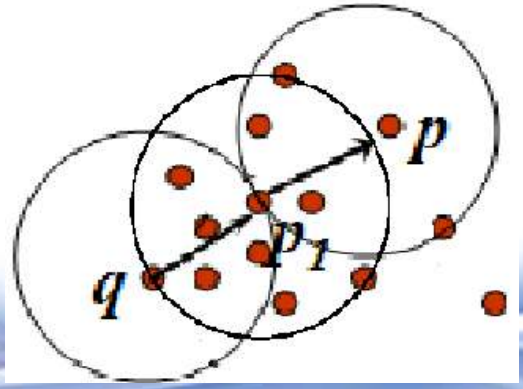
定义 对象的ε-邻域：给定对象在半径ε内的区域。

定义 核心对象：如果一个对象的ε-邻域至少包含最小数目MinPts个对象，则称该对象为核心对象。

例 下图中，ε=1cm，MinPts=5，q是一个核心对象。

定义 直接密度可达：给定一个对象集合D，如果p是在q的ε-邻域内，而q是一个核心对象，我们说对象p从对象q出发是直接密度可达的。

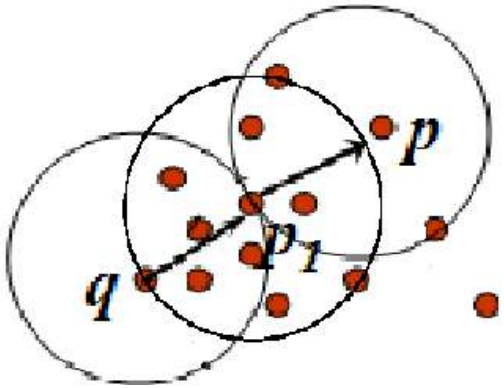
例 在下图中，ε=1cm，MinPts=5，q是一个核心对象，对象$p1$从对象q出发是直接密度可达的。



基于密度方法的聚类- DBSCAN 所用到的基本术语

定义 密度可达的：如果存在一个对象链$p1$，$p2$，…，pn，$p1=q$，$pn=p$，对$pi \in D$，（$1<=i<=n$），$pi+1$是从pi关于ε和MitPts直接密度可达的，则对象p是从对象q关于ε和MinPts密度可达的。

例　在下图中，ε=1cm，MinPts=5，q是一个核心对象，$p1$是从q关于ε和MitPts直接密度可达，p是从$p1$关于ε和MitPts直接密度可达，则对象p从对象q关于ε和MinPts密度可达的



密度可达

基于密度方法的聚类- DBSCAN 所用到的基本术语

定义 密度相连的： 如果对象集合*D*中存在一个对象*o*，使得对象*p*和*q*是从*o*关于ε和MinPts密度可达的，那么对象*p*和*q*是关于ε和MinPts密度相连的。

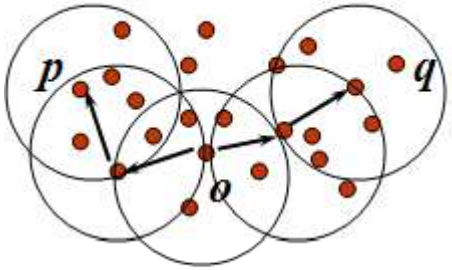


图 密度相连

定义 噪声: 一个基于密度的簇是基于密度可达性的最大的密度相连对象的集合。不包含在任何簇中的对象被认为是“噪声”。

边界点：**边界点不是核心点，但落在某个核心点的邻域内。**

噪声就是那些既不是边界点也不是核心点的对象

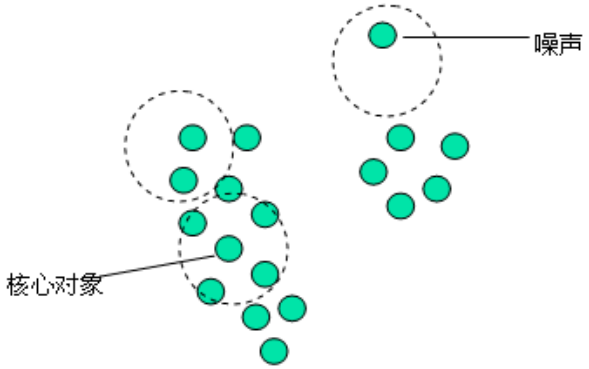
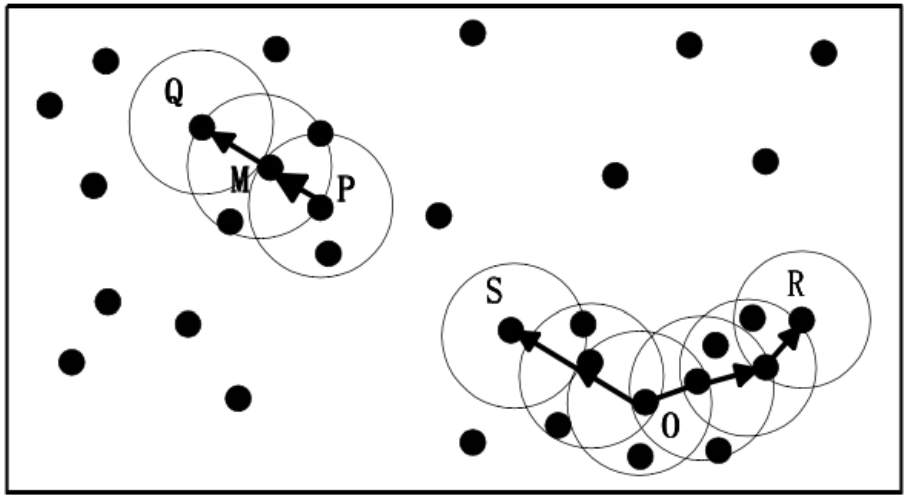


图 噪声

DBSCAN算法概念示例

- 如图所示，ε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。



“直接密度可达”和“密度可达”概念示意描述

解答：根据以上概念知道：由于有标记的各点-M、P、O和R的ε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。

基于密度方法的聚类- DBSCAN

- DBSCAN 算法根据以上的定义在数据库中发现簇和噪声。簇可等价于集合D中，这个簇核心对象密度可达的所有对象的集合。

- DBSCAN通过检查数据集中每个对象的ε-邻域来寻找聚类。如果一个点p的ε-邻域包含多于MinPts个对象，则创建一个p作为核心对象的新簇C。然后，DBSCAN从C中寻找未被处理对象q的ε-邻域，如果q的ε-邻域包含多MinPts个对象，则还未包含在C中的q的邻点被加入到簇中，并且这些点的ε-邻域将在下一步中进行检测。这个过程反复执行，当没有新的点可以被添加到任何簇时，该过程结束。具体如下：

基于密度方法的聚类- DBSCAN

DBSCAN算法描述：

- **输入：包含*n*个对象的数据库，半径*ε*，最少数目MinPts。**
- **输出：所有生成的簇，达到密度要求。**
- **1. REPEAT**
- **2. 从数据库中抽取一个未处理过的点；**
- **3. IF 抽出的点是核心点 THEN找出所有从该点密度可达的对象，形成一个簇**
- **4. ELSE 抽出的点是边缘点(非核心对象)，跳出本次循环，寻找下一点；**
- **5. UNTIL 所有点都被处理；**

DBSCAN算法步骤

输入：数据集D，参数MinPts, ε 输出：簇集合

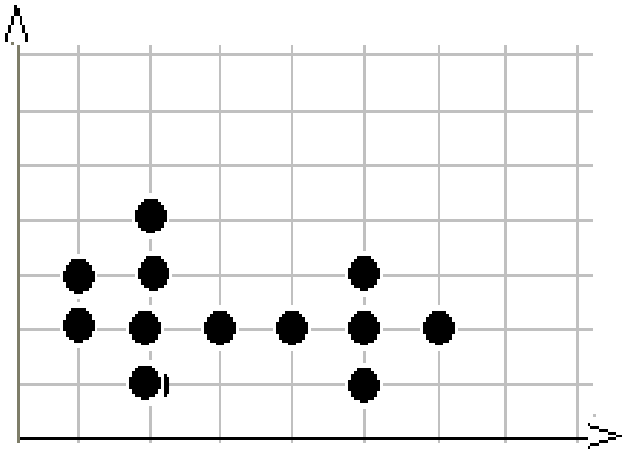
(1) 首先将数据集*D*中的所有对象标记unvisited ；
(2) do
(3)　从D中随机选取一个unvisited对象p，并将p标记为visited ；
(4)　if　p的 ε 邻域 包含的对象数至少为MinPts个
(5)　　创建新簇C ，并把p添加到c中；
(6)　　令N为 p的 ε 邻域 中对象的集合；
(7)　　for N 中每个点pi
(8)　　　if pi 是unvisited
(9)　　　　标记pi 为visited；
(10)　　　　if pi 的 ε 邻域 至少有MinPts个 对象，把这些对象添加到N ；
(11)　　　if pi 还不是任何簇的对象。将 pi 添加到 簇C中 ；
(12)　　end for
(13)　　输出C
(14) Else 标记p 为噪声
(15) Untill 没有标记为unvisited 的对象

基于密度方法的聚类- DBSCAN

下面给出一个样本事务数据库（见下表），对它实施DBSCAN算法。
根据所给的数据通过对其进行DBSCAN算法，以下为算法的步骤（设 n=12，用户输入ε=1，MinPts=4）

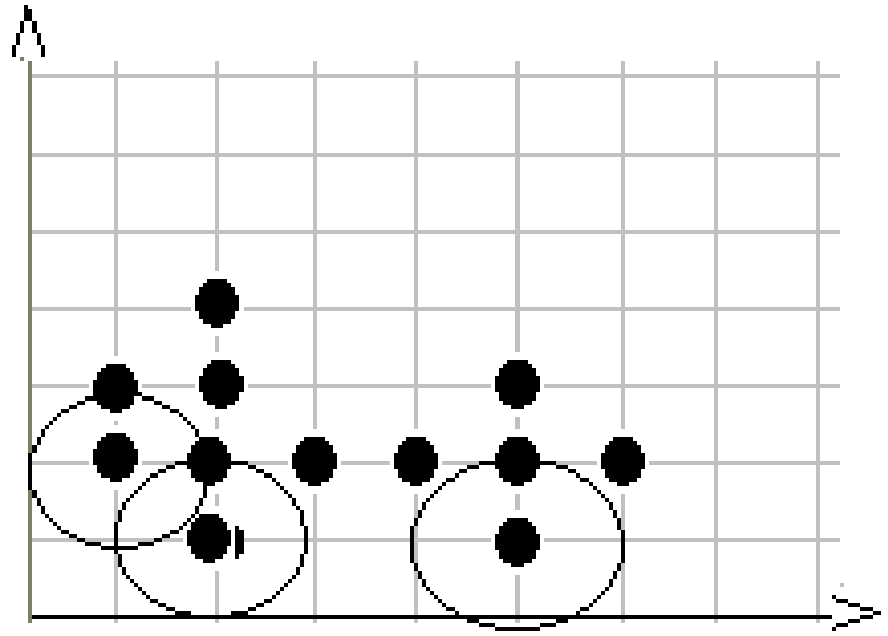
样本事务数据库

序号	属性 1	属性 2
1	2	1
2	5	1
3	1	2
4	2	2
5	3	2
6	4	2
7	5	2
8	6	2
9	1	3
10	2	3
11	5	3
12	2	4



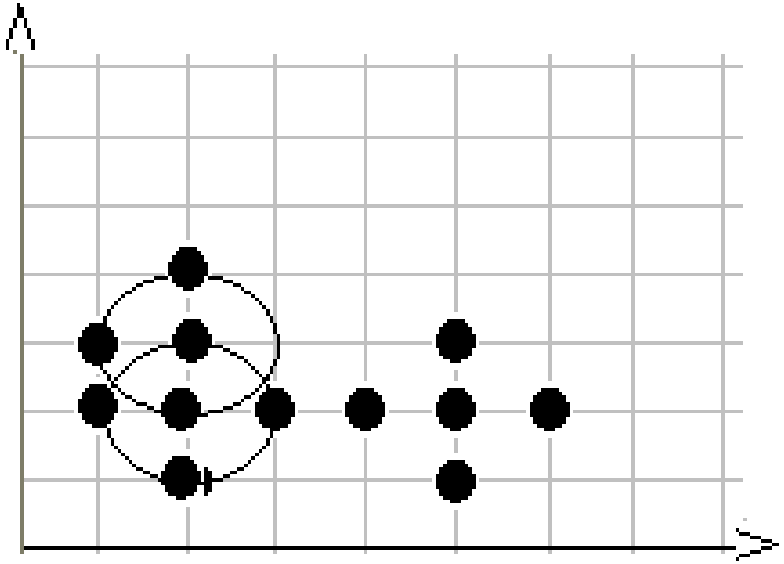
DBSCAN聚类过程

- 第1步，在数据库中选择一点1，由于在以它为圆心的，以1为半径的圆内包含2个点（小于4），因此它不是核心点，选择下一个点。
- 第2步，在数据库中选择一点2，由于在以它为圆心的，以1为半径的圆内包含2个点，因此它不是核心点，选择下一个点。
- 第3步，在数据库中选择一点3，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。



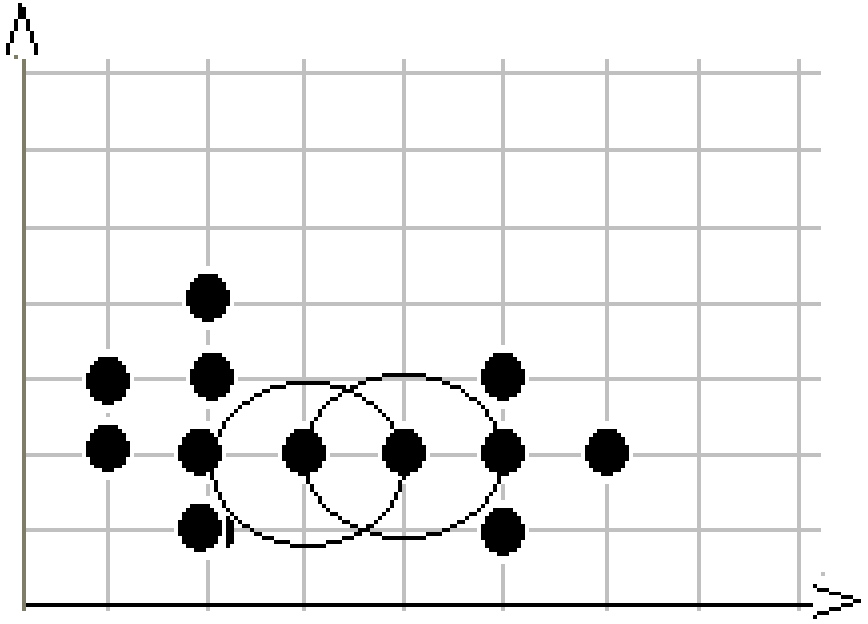
DBSCAN聚类过程

- 第4步，在数据库中选择一点4，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点（直接可达4个，间接可达3个），聚出的新类{1，3，4，5，9，10，12}，选择下一个点。



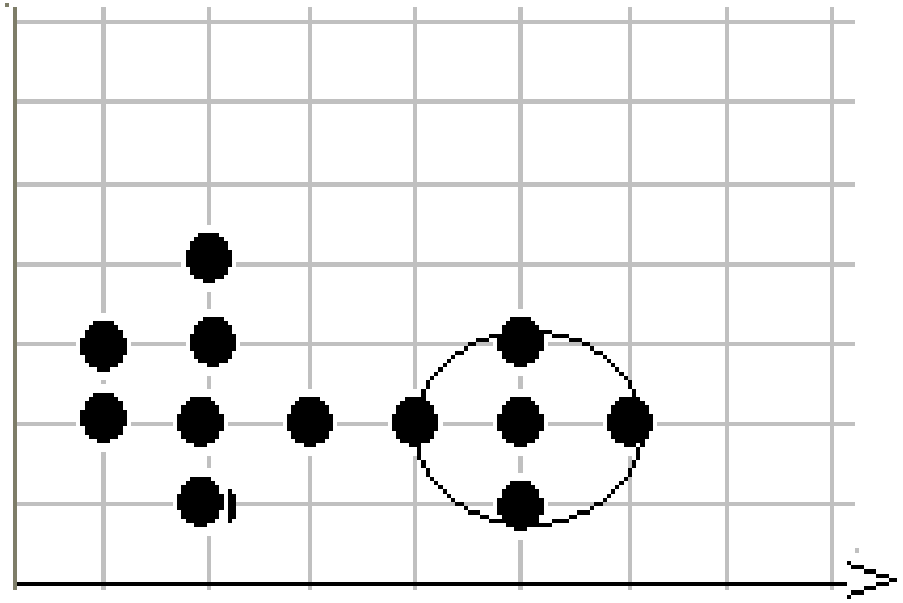
DBSCAN聚类过程

- 第5步，在数据库中选择一点5，已经在簇1中，选择下一个点。
- 第6步，在数据库中选择一点6，由于在以它为圆心的，以1为半径的圆内包含3个点，因此它不是核心点，选择下一个点。



DBSCAN聚类过程

- 第7步，在数据库中选择一点7，由于在以它为圆心的，以1为半径的圆内包含5个点，因此它是核心点，寻找从它出发可达的点，聚出的新类{2，6，7，8，11}，选择下一个点。



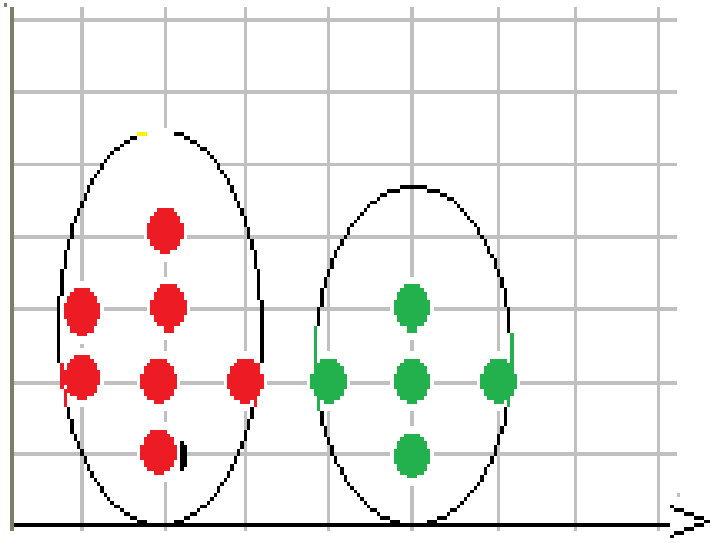
DBSCAN聚类过程

- 第8步，在数据库中选择一点8，已经在簇2中，选择下一个点。
- 第9步，在数据库中选择一点9，已经在簇1中，选择下一个点。
- 第10步，在数据库中选择一点10，已经在簇1中，选择下一个点。
- 第11步，在数据库中选择一点11，已经在簇2中，选择下一个点。
- 第12步，选择12点，已经在簇1中，由于这已经是最后一点所有点都以处理，程序终止。

基于密度方法的聚类- DBSCAN

算法执行过程：

步骤	选择的点	在ε中点的个数	通过计算可达点而找到的新簇
1	1	2	无
2	2	2	无
3	3	3	无
4	4	5	簇C_1：{1，3，4，5，9，10，12}
5	5	3	已在一个簇C_1中
6	6	3	无
7	7	5	簇C_2：{2，6，7，8，11}
8	8	2	已在一个簇C_2中
9	9	3	已在一个簇C_1中
10	10	4	已在一个簇C_1中，
11	11	2	已在一个簇C_2中
12	12	2	已在一个簇C_1中



DBSCAN的时间复杂性

- 时间复杂度
  - DBSCAN算法要对每个数据对象进行邻域检查时间性能较低。
  - 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)

DBSCAM的空间复杂性

- 空间复杂度

在聚类过程中，DBSCAN一旦找到一个核心对象，即以该核心对象为中心向外扩展．此过程中核心对象将不断增多，未处理的对象被保留在内存中．若数据库中存在庞大的聚类，将需要很大的存来存储核心对象信息，其需求难以预料．

当数据量增大时，要求较大的内存支持 I/0 消耗也很大;

低维或高维数据中，其空间都是O(N)

基于密度方法的聚类

- 优点

能克服基于距离的算法只能发现“类圆形”的聚类的缺点，可发现任意形状的聚类，有效地处理数据集中的噪声数据，数据输入顺序不敏感

- 缺点

输入参数敏感．确定参数ε，MinPts困难，若选取不当，将造成聚类质量下降．

由于在DBSCAN算法中，变量ε，MinPts是全局惟一的，当空间聚类的密度不均匀、聚类间距离相差很大时，聚类质量较差。

计算密度单元的计算复杂度大，需要建立空间索引来降低计算量，且对数据维数的伸缩性较差。这类方法需要扫描整个数据库，每个数据对象都可能引起一次查询，因此当数据量大时会造成频繁的*I/O*操作。。