一种基于密度的发现集群的算法在有噪声大空间数据库中的应用

### Martin Ester，Hans-Peter Kriegel，Jorg Sander，徐小卫

摘要

聚类算法对空间数据库中的类别识别任务很有吸引力。 然而，对大型空间数据库的应用对聚类算法提出了以下要求：确定输入参数的领域知识需求最少，在大数据库中发现具有任意形状的聚类和高效率。 众所周知的聚类算法不能解决这些要求的组合问题。 在本文中，我们提出了一种新的聚类算法DBSCAN，它依赖于基于密度的聚类概念，旨在覆盖任意形状的聚类。 DBSCAN只需要一个输入参数，并支持用户为其确定适当的值。 我们使用合成数据和SEQUOIA 2000基准实测数据对DBSCAN的有效性和效率进行了实验评估。 我们的实验结果表明：（1）DBSCAN在发现任意形状的聚类方面比众所周知的算法CLAR A NS有效得多，并且（2）DBSCAN在CLARANS方面的表现优于超过100的CLARANS效率。

关键词：聚类算法，簇的任意形状，大型空间数据库效率，处理Nlj4-275oise。

1. 介绍

许多应用需要管理空间数据，即与空间有关的数据。 空间数据库系统（SDBS）（1994年国际会议）是用于管理空间数据的数据库系统。 从卫星图像，X射线晶体学或其他自动设备获得越来越多的数据。 因此，在空间数据库中，自动化知识领域发现变得越来越重要。

文献中已经定义了数据库中知识发现的几项任务（KDD）（Matheus，Chan＆Piatetsky-Shapiro 1993）。本文考虑的任务是类标识，即将数据库的对象分组为有意义的子类。例如，在地球观测数据库中，我们可能想要发现沿着某条河流的房屋类别。

聚类算法对于类别识别的任务很有吸引力。 但是，对大型空间数据库的应用对聚类算法提出了以下要求：

1. 在处理大型数据时往往不能提前知道合适的参数取值，领域知识的最低要求决定了输入参数。
2. 发现任意形状的簇，因为在空间数据库中簇的形状不规则。
3. 在大型数据库中具有很高效率。

一些著名的算法没有提出同时解决这些问题的方法，本文提出一种新的聚类算法DBSCAN, 它只需要一个输入参数，并支持用户为其确定合适的值.它能够发现任意形状的簇。最后，即使对于大型空间数据库，DBSCAN也具有很高效率。本文的其余部分组织如下。在第二节我们讨论了聚类算法并按照上述要求对其进行评估。第三节我们基于数据库密度的概念提出了簇的概念。第四节我们介绍了DBSCAN算法，它能够在空间数据库中发现簇。在第5节中，我们使用SEQUOIA 2000基准的合成数据和数据对DBSCAN的有效性和效率进行了实验评估。 第6节总结了未来研究的总结和一些方向。

1. 聚类算法

有两种基本类型的聚类算法（Kaufman＆Rousseeuw 1990）：分区和分层算法。 分区算法将n个对象的数据库D划分为一组k个集群。 k是这些算法的输入参数，即某些领域知识是必需的，遗憾的是许多应用不可用。 分区算法通常从D的初始分区开始，然后使用迭代控制策略来优化目标函数。 每个聚类由聚类的重心（k-means算法）或位于其中心附近的聚类中的一个对象（k-medoid算法）表示。 因此，分区算法使用两步过程。 首先，确定最小化目标函数的k代表。 其次，将每个对象以其代表“对象”标记给所考虑的对象。 第二步意味着一个分区相当于一个voronoi图，每个集群包含在一个voronoi单元中。 因此，所有的形状由分区算法发现的聚类是凸的，这是非常严格的。

Ng＆Han（1994）在空间数据库中探索KDD的分区算法。 引入了一种称为CLARANS（基于RANdomized Search的聚类大应用）算法，该算法是一种改进的k-medoid方法。 与以前的k-medoid算法相比，CLARANS更有效，更高效。 实验评估表明，CLARANS能够在数千个对象的数据库上高效运行。 Ng＆Han（1994）也讨论了确定数据库中“簇”的“自然”数目的方法。 他们建议从2到n的每个k运行一次CLARANS。 对于每个发现的聚类，计算silhouette系数（Kaufman＆Rousseeuw 1990），最后选择具有最大silhou ette系数的聚类作为“自然”聚类。 不幸的是，这种方法的运行时间对于大n来说是禁止的，因为它意味着O（n）调用CLARANS。

**C**LARANS假定所有要聚集的对象可以同时在主内存中重新访问，这对于大型数据库不适用。 而且，CLARANS的运行时间对于大型数据库而言是禁止的。 因此，Ester，Kriegel＆Xu（1995）提出了几种聚焦技术，通过将聚类过程聚焦在数据库的相关部分来解决这两个问题。 首先，焦点小到足以成为内存驻留，其次，CLARANS在焦点对象上的运行时间明显小于整个数据库上的运行时间。

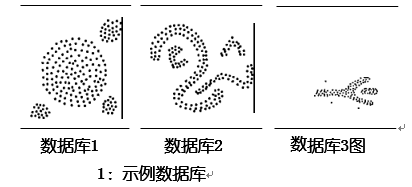
*分*层算法创建D的分层分解。分层分解由delldrogram表示，该分层分解树是将D反复分割成更小的子集，直到每个子集仅由一个对象组成的树。 在这样的层次结构中，树的每个节点都代表D的一个簇。树状图可以通过合并或划分聚类来创建，从叶子到根（凝聚方法），或者从根到树叶（分裂方法） ters在每一步。 与分区算法相比，高级存档算法不需要k作为输入。 但是，必须定义一个最终条件，指明合并或分割过程何时应该终止。 集聚方法中终止条件的一个例子是Q的所有集群之间的临界距离Dmin。

到目前为止，层次聚类的主要问题。算法一直是为终止条件推导适当参数的困难，例如Dmin的值足够小以便分离所有“自然”群集，并且同时足够大以至于没有群集被分成两部分。 最近，在信号处理领域，提出了一种高级算法Ejcluster（Garcfa，Fdez-Valdivia，Cortijo＆Molina 1994），自动推导出终止条件。 它的关键思想是，如果您可以通过“足够小”的步骤从第一个点走到第二个点，那么两个点对于同一个簇是长的。 Ejcluster遵循分裂的方法。 它不需要任何领域知识的输入。 此外，实验表明它在发现非凸集群方面非常有效。 但是，Ejcluster的计算成本是O（n2）。对于字符识别这样的应用这是可接受的，但是对于大型数据库的应用程序而言是过分的。

Jain（1988）探索了一种基于密度的方法来识别k维点集中的聚类。 数据集被分成许多不重叠的单元格，并构建直方图。 具有相对较高频点计数的单元是潜在的聚类中心，并且聚类之间的边界落在直方图的“谷”中。 该方法具有识别任何形状的簇的能力。 然而，用于存储和搜索多维直方图的空间和运行时间要求可能是巨大的。 即使空间和运行时间要求得到优化，这种方法的性能关键取决于单元的大小。

1. 基于密度的集群概念

当查看图1所示的样本点集合时，我们可以轻松而明确地检测到不属于任何这些聚类的点和噪声点集群。



我们认识这些聚类的主要原因在于，在每个聚类中，我们有一个典型的点密度，这个密度远高于聚类外。 此外，噪音区域内的密度低于任何群集中的密度。

在下文中，我们尝试在一些k维空间S的点D的数据库D中形式化这种直观的“集群”和“噪声”概念。注意，我们的集群概念和算法DBSCAN都适用于二维或三维欧几何空间关于一些高维特征空间。 关键的思想是，对于一个集群的每个点，给定半径的邻居罩必须包含至少最小数量的点，即邻域中的密度必须超过某个阈值。 一个邻域的形状由两个点p和q的距离函数的选择来确定，用dist（p，q）表示。 例如，当在二维空间中使用曼哈顿距离时，相邻罩的形状是矩形的。 请注意，我们的方法适用于任何距离函数，因此可以为某个给定的应用程序选择适当的函数。 为了适当的缩小化，所有示例都将在使用欧几里德距离的二维空间中进行。

1. ϵ邻域：对于xj∈D，其ϵ邻域包含样本集D中与xj的距离不大于ϵ的子样本集，即,这个样本集的个数记为
2. 核心对象：对于任一样本xj∈D，如果其ϵ邻域对应的至少包含MinPts个样本，即如果，则xj是核心对象。
3. 密度直达：如果且,则称p由q密度直达。当p和q都是核心对象时，密度直达满足对称性。

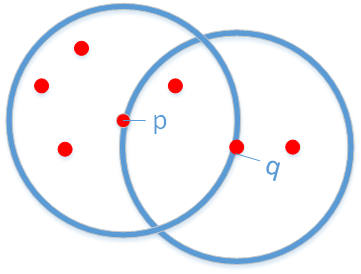


图2

图2.1中，假设MinPts=5,则p是核心对象，q是噪声点，从p到q密度直达，而从q到p不是密度直达

1. 密度可达：对于xi和xj,如果存在样本序列p1,p2,…,pT满足p1=x1,pT=xj,且pt+1由pt密度直达，则称xj由xi密度可达。即密度可达满足传递性，但不满足对称性。
2. 密度相连：对于xi和xj,如果存在核心对象样本xk,使得xi和xj均由xk密度可达，则称xi和xj密度相连。密度相连满足对称性。
3. 簇：簇是样本集的一个子集，当样本点满足下列条件时称样本点包含于簇C。
4. :如果p∈C且q由p密度可达，则q∈C
5. :p,q密度相连
6. 噪声点：设C1,C2,…,Ck是样本集D中的簇，则定义噪声点为不属于任何簇的数据集，即

**引**理1：令p是D和INEps（p）I MinPts中的一个点。 然后集合O = {o I o ED并且o从p wrt密度可达。 Eps和MinPts}是一个群集wrt。 Eps和MinPts。

C群并不明显。 Eps和MinPts由其任何核心点独一无二地决定。 但是，C中的每个点都可以从C的任何核心点密度可达，因此，C群恰好包含从C的任意核心点密度可达的点。

**引**理2：设C是一个簇。 Eps和MinPts，并让p为INEps（p）I MinPts中的任意一点。 那么C等于集合O = {o I o是密度可达p wrt。 Eps和MinPts}。

1. DBSCAN：基于密度的带噪声空间聚类

在本节中，我们提出了DBSCAN（基于密度的空间应用噪声空间聚类）算法，该算法旨在根据定义5和6发现空间数据库中的聚类和噪声。理想情况下，我们必须知道适当的参数每个群集的Eps和MinPts以及来自各个群集的至少一个点。 然后，我们可以使用正确的参数检索从给定点密度可达的所有点。但没有简单的方法可以事先为数据库的所有群集获取此信息。 然而，有一个简单而有效的启发式方法（在4.2节中介绍）来确定数据库中“最薄”（即最小稠密）簇的参数Eps和MinPts。 因此，DBSCAN使用Eps和MinPts的全局值，即所有群集的相同值。 “最薄”簇的密度参数是指定不被认为是nmse的最低密度的这些全局参数值的良好候选者。

* 1. 算法

为了找到一个集群，DBSCAN以任意点p开始，并从p个密钥中获取所有密度可达的点。 Eps和MinPts。 如果p是核心点，则此过程会生成一个聚类。 Eps和MinPts（见引理2）。 如果pis是一个边界点，则不可以从p到密度可达的点，并且DBSCAN访问数据库的下一个点。

由于我们使用Eps和MinPts的全局值，因此如果两个不同密度的聚类彼此“接近”，DBSCAN可能会根据定义5将两个聚类合并为一个聚类。 设两组点S1和S2之间的距离定义为dist（S1，S2）= min {dist（p，q）Ip E S1，q E S2}，然后，两组至少具有密度的点，只有当两组之间的距离大于Eps时，最薄的组才会彼此分离。 因此，对于MinPts具有较高值的​​检测集群，可能需要递归调用DBSCAN。 然而，这不是缺点，因为DBSCAN的递归应用产生了一个优雅而高效的基本算法。 此外，只有在易于检测到的情况下，群集的点的递归聚类才是必需的。

下面，我们介绍DBSCAN的基本版本，省略数据类型的详细信息并生成有关群集的附加信息：

DBSCAN（SetOfPoints，Eps，MinPts）

*II* SetOfPoints是UNCLASSIFIED Clusterid：= nextid（NOISE）;

我从1到SetOfPoints.size DO Point：= SetOfPoints.get（i）;

IF Point.Clid = UNCLASSIFIED THEN

IF ExpandCluster（SetOfPoints，Point，Clusterid，Eps，MinPts）THEN

Clusterid .- nextid（Clusterid）END IF

END IF END FOR

SetOfPoints可以是整个数据库，也可以是上一次运行的Dis覆盖的群集。Eps和MinPts是手动确定的全局密度参数，或根据第4.2节中的启发式确定的。函数SetOf Points.get(i)返回SetOf Points的第i个元素。最重要的功能DBSCAN使用的是存在的ExpandCluster.

ExpandCluster（SetOfPoints，Point，Clid，Eps，MinPts）：Boolean;

seeds：= SetOfPoints.regionQuery（Point，EPS）;

IF seeds.size < MinPts THEN //没有核心点

SetOfPoint.changeClid（Point，NOISE）;

Return false;

ELSE //种子中的所有点都是密度可达 -

Point SetOfPoints.changeClids（seeds，Clid）;

seeds.delete（Point）;

WHILE seeds <> Empty DO

currentP：= seeds.first（）;

result：= SetOfPoints.regionQuery（currentP，EPS）;

IF result.size>= MinPts THEN

FOR i FROM 1 TO result.size DO

resultP：= result.get（i）;

IF resultP.Clid IN {UNCLASSIFIED，NOISE} THEN

IF resultP.Clid = UNCLASSIFIED THEN

seeds.append（resultP）;

END IF;

SetOfPoints.changeClid（resultP，CLID）;

END IF; //未分类或噪音

END FOR;

END IF; //result.size> = MinPts

seeds.delete（currentP）;

END WHILE; //种子<>空

RETURN True

END IF

END //ExpandCluster

SetOfPoints.regionQue-ry（Point，Eps）的调用将SetOf Points中的Eps-邻域作为点列表返回。 区域查询可以通过空间访问方法有效地支持，如R \* - 树（贝克曼等人，1990），假设这些空间访问方法在SDBS中可用于高效处理几种类型的空间查询（Brinkhoff et al。1994）。 在最坏的情况下，对于一个有n个点的数据库，R \*树的高度为O（log n），具有“小”查询区域的查询只能遍历R \*树中有限数量的路径。 由于Eps邻域与整个数据空间的大小相比预计较小，因此单个区域查询的平均运行时间复杂度为O（log n）。 对于数据库的n个点中的每一个，我们最多只有一个区域查询。 因此，DBSCAN的平均运行时间复杂度是O（n \* log n）。

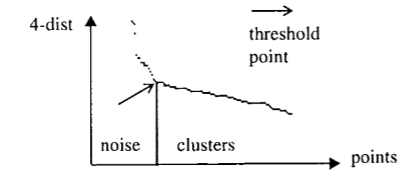
已标记为NOISE的点的Id Id（clusterId）可稍后更改，如果它们可从数据库的其他某个点进行密度达到。 这发生在群集的边界点上。 这些点不会添加到种子列表中，因为我们已经知道具有NOISE Clld的点不是核心点。 将这些点添加到种子只会导致额外的区域查询，这将不会产生新的答案。

如果两个簇C1和C2彼此非常接近，则可能发生某点p属于C1和C2两者。 那么p必须是两个簇中的边界点，因为其他明智的C1将等于C2，因为我们使用全局参数。在这种情况下，点p将被分配到首先展开的集群。 除了这些罕见的情况外，由于引理2，DBSCAN的结果与访问数据库点的顺序无关。

4.2 确定参数Eps和MinPts

在本节中，我们开发了一种简单但有效的启发式方法来确定数据库中“最薄”集群的参数Eps和MinPts。 该启发式基于以下观察。 设d是点p到其第k个最近邻点的距离，则p的d邻域几乎包含所有点p的ex + k + I点。 只有当几个点与p有完全相同的距离d时，p的d-邻域才会包含超过k + 1个点，这相当不可能。 Furthennore指出，在集群中改变k点并不会导致d的巨大变化。 这种情况只有在p叉=第1,3，...的第k个最近邻居大致位于一条直线上时才会发生，这条直线对于一个簇中的一个点通常不是正确的。

对于给定的k，我们定义从数据库D到实数的函数k-dist，将每个点映射到距其第k个最近邻居的距离。 按照k-dist值的降序对数据库中的点进行排序时，该函数的图形给出了关于数据库中密度分布的一些提示。 我们称这个图为排序的k-dist图。 如果我们选择一个任意点p，将参数Eps设置为k-dist（p）并将参数MinPts设置为k，那么具有相等或更小的k-dist值的所有点将成为核心点。 如果我们可以在D的“最薄”簇中找到具有最大k距离值的阈值点，那么我们将具有期望的参数值。 阈值点是排序k-dist图的第一个“谷”中的第一个点（见图4）。 具有较高k-dist值（阈值左边）的所有点被认为是噪声，所有其他点（阈值的右边）被分配给某个群集。

****

**图4：样本数据库的排序4-dist图**

一般而言，首先自动检测第一个“谷”是非常困难的，但对于用户以图形表示来看这个谷是相对简单的。 因此，我们建议采用交互式方法来确定阈值。

DBSCAN需要两个参数Eps和MinPts。 实际上，我们的实验表明，k-dist图fork> 4与4-dist图显着不同，而且它们需要更多的计算。 因此，我们通过将所有数据库（对于二维数据）设置为4来消除MinPts参数。 我们提出以下交互方法来确定DBSCAN的Eps参数：

1. 系统计算并显示数据库的4-dist图。
2. 如果用户可以估计噪音的百分比，则输入这个百分比，系统从中得出阈值点的建议。
3. 用户接受建议的阈值或选择另一个点作为阈值点。 阈值点的4-dist值用作DBSCAN的Eps值。
4. 效果评估

在本节中，我们评估DBSCAN的性能。 我们将它与CLARANS的性能进行比较，因为这是为KDD设计的第一个也是唯一的聚类算法。 在我们未来的研究中，我们将与经典的基于密度的聚类算法进行比较。 基于R \* -tree的实现，我们已经在C ++中实现了DBSCAN（Beckmann et al。1990）。 所有实验都在HP 735/100工作站上运行。 我们使用了合成样本数据库和SEQUOIA 2000基准数据库。

为了将DBSCAN与CLARANS在效率（准确性）方面进行比较，我们使用了图1中描述的三个合成样本数据库。由于DBSCAN和CLARANS是不同类型的聚类算法，因此它们没有常见的分类精度定量度量。 因此，我们通过目测来评估两种算法的准确性。 在样品数据库1中，有四个尺寸明显不同的球形团簇。 示例数据库2包含四个非凸形状的集群。 在样本数据库3中，有四个不同形状和大小的群集以及额外的噪声。 为了显示两种聚类算法的结果，我们用不同的颜色对每个聚类进行可视化（请参阅第6节后的www可用性）。 为了给CLARANS一些优势，我们为这些样本数据库设置参数k为4。 图5描述了CLARANS发现的聚类。

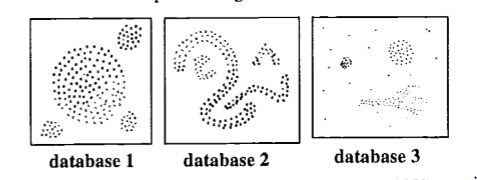


图5 CLARANS发现的簇

对于DBSCAN，我们将样本数据库I和2的噪声百分比设置为0％，对于样本数据库3，将噪声百分比设置为10％。 DBSCAN发现的聚类图见图6。

DBSCAN发现所有群集（根据定义5），并从所有样本数据库中检测噪声点（根据定义6）。 然而，如果CLARANS相对较大或者它们靠近其他某个群集，它们会将聚类分裂。 此外，CLARANS没有明确的噪音​​。 相反，所有点都被分配到他们最近的medoid。

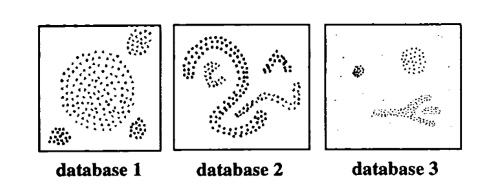


图6. DBSCAN发现的簇

为了测试DBSCAN和CLARANS的效率，我们使用SEQUOIA 2000基准数据。 SEQUOIA 2000基准数据库（Stonebraker et al。1993）使用代表地球科学任务的真实数据集。 数据库中有四种类型的数据：栅格数据，点数据，多边形数据和有向图数据。 点数据集包含美国地质调查局地理名称信息系统提供的62,584个加利福尼亚地标名称及其位置。 点数据集约占2.1 M字节。 由于CLAR ANS在整个数据集上的运行时间非常长，我们已经提取了包含整个集合的2％至20％代表的SEQUIOA 2000点数据集的一系列子集。 表1列出了DBSCAN和CLARANS在这些数据库上的运行时间比较。

**表1：以秒为单位的运行时间**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 点数 | 1252 | 2503 | 3910 | 5213 | 6256 |
| DBSCAN | 3.1 | 6.7 | 11.3 | 16.0 | 17.8 |
| CLAR- ANS | 758 | 3026 | 6845 | 11745 | 18029 |
| 点数 | 7820 | 8937 | 10426 | 12512 |  |
| DBSCAN | 24.5 | 28.2 | 32.7 | 41.7 |  |
| CLAR- ANS | 29826 | 39265 | 60540 | 80638 |  |

我们的实验结果表明，DBSCAN的运行时间略高于线性点的数量。 然而，CLARANS的运行时间接近点的数量。 结果表明，DB SCAN的性能优于CLARANS的250到1900之间，随着数据库大小的增加而增长。

1. 结论

聚类算法对空间数据库中的类别识别任务很有吸引力。 然而，当应用于大型空间数据库时，众所周知的算法存在严重的缺点。 在本文中，我们提出了聚类算法DBSCAN，它依赖于基于密度的聚类概念。 它只需要一个输入参数，并支持用户为其确定合适的值。 我们对合成数据进行了绩效评估以及SEQUOIA 2000基准测试的实际数据。 这些实验的结果表明，DBSCAN在发现任意形状的聚类方面比知名算法CLARANS更有效。 此外，实验表明，就效率而言，DBSCAN超过了至少100倍的CLARANS。

未来的研究将不得不考虑以下问题。 首先，我们只考虑了点对象。 但是，空间数据库也可能包含扩展对象，例如多边形。 为了推广DBSCAN，我们必须在多边形数据库的Bps邻域中定义密度。 其次，应该研究DBSCAN在高维特征空间中的应用。 特别是，这种应用中的k-dist图的形状必须进行探索。

参考文献

[1] Schneider R, Seeger B, Beckmann N, et al. The R\*-tree: an efficient and robust access method for points and rectangles[C]//Proc. ACM SIGMOD Int. Conf. Manage. Data. 1990: 322-331.

[2] Brinkhoff T, Kriegel H P, Schneider R, et al. Multi-step processing of spatial joins[M]. ACM, 1994.

[3] Ester M, Kriegel H P, Xu X. A database interface for clustering in large spatial databases[C]//KDD. 1995: 94-99.

[4] García J A, Fdez-Valdivia J, Cortijo F J, et al. A dynamic approach for clustering data[J]. Signal Processing, 1995, 44(2): 181-196.

[5] Güting R H. An introduction to spatial database systems[J]. The VLDB Journal—The International Journal on Very Large Data Bases, 1994, 3(4): 357-399.*VLDB Journal 3（4）：357-399。*

[6]J Kaufman L, Rousseeuw P J. Finding groups in data: an introduction to cluster analysis[M]. John Wiley & Sons, 2009.

[7] Matheus C J, Chan P K, Piatetsky-Shapiro G. Systems for knowledge discovery in databases[J]. IEEE Transactions on knowledge and data engineering, 1993, 5(6): 903-913.

[8] Ng R T, Han J. CLARANS: A method for clustering objects for spatial data mining[J]. IEEE transactions on knowledge and data engineering, 2002, 14(5): 1003-1016.

[9] Stonebraker M, Frew J, Gardels K, et al. The Sequoia 2000 storage benchmark[C]//ACM SIGMOD Record. ACM, 1993, 22(2): 2-11.