一种改进的基于密度的抽样聚类算法

胡彩平 秦小麟
(南京航空航天大学信息科学与技术学院，南京 210016)

摘 要 基于密度的聚类算法 DBSCAN 是一种有效的空间聚类算法，它能够发现任意形状的聚类并且有效地处理噪声。然而，DBSCAN 算法也有一些缺点，例如，①在聚类时只考虑空间属性没有考虑非空间属性；②在对大规模空间数据库进行聚类分析时需要较大的内存支持和 I/O 消耗。为此，在分析 DBSCAN 算法不足的基础上，提出了一种改进的基于密度的抽样聚类(improved density-based spatial clustering algorithm with sampling, IDBSCAS)算法，使之能够有效地处理大规模空间数据库，并且它不仅考虑了空间属性也考虑了非空间属性。2 维空间数据的测试结果表明，该算法是可行、有效的。

关键词 空间数据挖掘 空间聚类 密度 种子 非空间属性
中图法分类号：TP301.6 **文献标识码**：A **文章编号**：1006-8961(2007)11-2031-06

An Improved Density-based Spatial Clustering Algorithm with Sampling

HU Cai-ping, QIN Xiao-lin
(*College of Information Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing* 210016)

Abstract DBSCAN is one of the effective spatial clustering algorithms, which can discover clusters of any arbitrary shape and handle the noise effectively. However, it has also several disadvantages. First, it is based on only spatial attributes without considering non-spatial attributes in the databases. Second, when DBSCAN handles large-scale spatial databases, it requires large volume of memory support and I/O cost. In this paper, an improved density-based spatial clustering algorithm with sampling(IDBSCAS) is developed, which not only clusters large-scale spatial databases effectively, but also considers spatial attributes and non-spatial attributes. Experimental results of 2-D spatial datasets show that the new algorithm is feasible and efficient.

Keywords spatial data mining, spatial clustering, density, seeds, non-spatial attributes

1 引 言

近年来，数据挖掘的研究已从关系型和事务型数据库扩展到空间数据库。由于雷达、红外、光电、卫星、电视摄像、电子显微成像、CT 成像等各种宏观与微观传感器的使用，空间数据的数量、大小和复杂性都在飞快地增长，已经远远超出了人们的解释能力。终端用户不可能详细地分析所有的这些数据，并提取感兴趣的空间知识，致使“空间数据爆炸，但知识贫乏”。因此，利用空间数据挖掘(spatial data mining, SDM)从空间数据库中自动或半自动地挖掘事先未知却潜在有用的空间模式已变得十分必要。所谓空间数据挖掘指的是从空间数据库中抽取隐含的知识、空间关系或非显式地存储在空间数据库中有意义的特征或模式。

聚类分析是数据挖掘领域中的一项重要的研究课题。所谓聚类，就是根据相似性对数据对象进行分组，通过发现空间数据的分布特征，使得每一个聚类中的数据有非常高的相似性，而不同聚类中的数据尽可能不同。迄今为止，人们已经提出了许多聚类算法。聚类分析算法通常有以下 5 类[1]：

基金项目：国家自然科学基金项目(60673127)；江苏省自然科学基金项目(BK2001045)
收稿日期：2006-04-13；改回日期：2006-08-22
第一作者简介：胡彩平(1977～)，男。博士研究生。研究方向为空间数据挖掘等。E-mail：hucaiping@nuaa.edu.cn

(1)基于划分的方法[2],如 K-means 算法、K-medoids 算法、PAM 算法、CLARA 算法、CLARANS 算法和 EM 算法;(2)基于层次的方法[3,4],如 CURE 算法和 BIRCH 算法;(3)基于密度的方法[5~7],如 DBSCAN 算法、OPTICS 算法、DENCLUE 算法和 GDBSCAN 算法;(4)基于网格的方法[8,9],如 STING 算法、CLIQUE 算法和 WaveCluster 算法;(5)基于模型的方法,如 COBWEB 算法。其中,DBSCAN 算法是一种颇具代表性的基于密度的空间聚类分析算法,它具有可以发现任意形状的聚类和有效屏蔽噪声数据干扰的优点。本文在分析 DBSCAN 算法不足的基础上,提出了一种改进的基于密度的抽样聚类算法(improved density-based spatial clustering algorithm with sampling,IDBSCAS)。该算法是通过减少区域查询的次数来降低聚类时间和 I/O 消耗。另外,由于本文算法不仅考虑了空间属性,还考虑了非空间属性,从而提高了聚类的质量。

2 关于 DBSCAN 算法

2.1 DBSCAN 算法的思想

DBSCAN 算法是第 1 个基于密度的空间聚类算法,其基本思想是:对于一个聚类中的每一个对象,在其给定半径 ε 的邻域中包含的对象不能少于某一给定的最小数目 N_{min},然后对具有密度连接特性的对象进行聚类。

对于对象集 D,给定距离函数 d,参数 $\varepsilon \geq 0$ 和 $N_{min} \geq 0$。下面是基于密度的聚类算法 DBSCAN 的一些定义[5]。

定义 1(ε-邻域) 给定 ε 和对象 $p \in D$,以对象 p 为中心,以 ε 为半径的区域是对象 p 的 ε-邻域,用 $N_\varepsilon(p)$ 来表示,即 $N_\varepsilon(p) = \{q \in D \mid d(p,q) \leq \varepsilon\}$。

定义 2(核心对象) 给定 ε 和 N_{min},若对象 p 的 ε-邻域 $N_\varepsilon(p)$ 包含的对象个数 $|N_\varepsilon(p)| \geq N_{min}$,则称 p 为核心对象。

定义 3(直接密度可达) 给定 ε 和 N_{min},则对象 q 是从对象 p 出发直接密度可达的,当

(1) $q \in N_\varepsilon(p)$

(2) $|N_\varepsilon(p)| \geq N_{min}$

定义 4(密度可达) 给定对象集合 D,当存在一个对象链 $p_1, p_2, \cdots, p_n; p_1 = q, p_n = p$,对 $\forall p_i \in D (i = 1,2,\cdots,n)$,且 p_{i+1} 是从 p_i 关于 ε 和 N_{min} 直接密度可达的,则称对象 p 从对象 q 关于 ε 和 N_{min} 密度可达(非对称)。

定义 5(密度相连) 如果对象集合 D 中存在一个对象 o,并使得对象 p 和 q 是从 o 关于 ε 和 N_{min} 密度可达的,那么对象 p 和 q 关于 ε 和 N_{min} 密度相连(对称)。

定义 6(聚类和噪声) 基于密度可达性最大的密度相连对象的集合称为聚类,不在任何聚类中的对象被认为是噪声。

为了发现一个聚类,DBSCAN 算法先开始从数据库中任意选取一个对象 p 开始执行一个区域查询,即寻找 p 的邻域对象,如果这个邻域的对象个数小于给定的阈值,则 p 为噪声,否则,p 邻域的所有对象形成一个聚类;然后这个邻域中的每个对象也执行同样的操作,同时看这些邻域中的对象是否能加到这个聚类中,重复执行同样的操作,直到这个聚类不能扩大为止。如果这个聚类不能被进一步扩大,那么 DBSCAN 算法就另外选取一个没有被标为某个聚类或噪声的对象,重复执行同样的操作。这个操作被重复执行,直到在数据库中的所有对象都被标为某个聚类或噪声。对于一个有 n 个对象的空间数据库,由于要执行 n 次区域查询,所以该算法的时间复杂性为 $O(n^2)$;在空间索引(如 R^*-树)的支持下[10],其复杂性为 $O(n\log n)$。

2.2 DBSCAN 算法的局限性及改进

实际上,DBSCAN 算法进行聚类的过程就是一个不断执行区域查询的过程,聚类过程的大部分时间都用在区域查询操作上。DBSCAN 算法对核心对象的邻域中包含的所有对象都执行区域查询来扩展聚类,显然这样时间花费很高。假设对聚类 C 中的某一给定核心对象 p 来说,可以想象它的邻域中所包含的所有对象的邻域将会互相覆盖。假定 q 是 p 邻域中的一个对象,如果它的邻域被 p 邻域中的其他对象的邻域所覆盖,则表明对 q 的区域查询是可以省掉的。这是因为 q 的邻域中所包含的对象可以通过对覆盖它的其他对象执行区域查询得到。也就是说,q 没有必要作为种子对象用于聚类扩展。实际上,对于稠密的数据集来说,在一个核心对象的邻域中有相当多的对象可以不用作为聚类扩展用的种子对象。这样从加速 DBSCAN 算法来讲,应当选择核心对象邻域中的部分代表对象,而不是像 DBSCAN 算法那样选择所有对象,作为种子对象用于聚类的扩展。这样就可以减少区域查询的次数和提高查询效率。

另外,DBSCAN 算法没有考虑非空间属性,如果非空间属性在聚类中起一个很大的作用的话,那么它就不适用,因为空间数据库不仅保存了空间物体的空间属性,也保存了它的非空间属性。空间属性表现为空间物体的方位信息,非空间属性表现为一个空间物体的其他属性,例如在一个地理信息系统中,一个社区的失业率就是非空间属性,社区的方位是空间属性。为了使聚类具有更高的质量,在聚类时不仅要考虑空间属性,也要考虑非空间属性。

3　一种改进的基于密度的抽样聚类算法

本文提出了一种改进的基于密度的抽样聚类算法。为了能够处理非空间属性,本文利用了纯度的概念,纯度是指邻域内相同非空间属性的对象个数和邻域内所有对象个数的比值。

3.1　基本概念

对于对象集 D,给定距离函数 $dist$,参数 $\varepsilon \geqslant 0$、$MinPts \geqslant 0$ 和 $MinPur \geqslant 0$,用 $attr$ 来表示对象的一个非空间属性(很容易推广到多个非空间属性)。下面是改进的基于密度的抽样聚类算法的一些定义。

定义 7(ε-匹配邻域)　给定 ε 和对象 $p \in D$,对象 p 的匹配邻域用 $\hat{N}_\varepsilon(p)$ 来表示,它被定义为 $\hat{N}_\varepsilon(p) = \{q \in D \mid dist(p,q) \leqslant \varepsilon \text{ and } p.attr = q.attr\}$。

定义 8(纯核心对象)　给定 ε,$MinPts$,$MinPur$,若对象 p 的 ε-匹配邻域 $\hat{N}_\varepsilon(p)$ 包含的对象个数 $|\hat{N}_\varepsilon(p)| \geqslant MinPts$ 并且 $|\hat{N}_\varepsilon(p)|/|N_\varepsilon(p)| \geqslant MinPur$,则称 p 为纯核心对象。

定义 9(纯直接密度可达)　给定 ε,$MinPts$ 和 $MinPur$,如果满足以下的两个条件,则对象 q 是从对象 p 出发纯直接密度可达的,可用 $p \Rightarrow q$ 来表示。

(1) $q \in \hat{N}_\varepsilon(p)$

(2) $|\hat{N}_\varepsilon(p)| \geqslant MinPts$ 并且 $|\hat{N}_\varepsilon(p)|/|N_\varepsilon(p)| \geqslant MinPur$

定义 10(纯密度相连)　在对象集 D 中,当给定 ε,$MinPts$ 和 $MinPur$,则纯密度相连关系⇔满足如下两个条件:

(1) $\forall p \in D, p \Leftrightarrow p$

(2) 当存在一个对象链 $p_1, p_2, \cdots, p_n; p_1 = q$,$p_n = p$,对 $\forall p_i \in D (i = 1,2,\cdots,n)$,如果 $\forall p_{i+1} \Rightarrow p_i$ 或 $p_i \Rightarrow p_{i+1}$,则对象 $p \Leftrightarrow q$

从这个定义,就能推出如下的定理。

定理　在对象集 D 中,对于给定 ε,$MinPts$ 和 $MinPur$ 的纯密度相连关系⇔是一个等价关系。

理由很明显,证明略。

由于纯密度相连关系⇔是一个等价关系,所以纯密度相连关系⇔能在对象集 D 上确定它的一个划分,每一个划分就是对象集 D 的一个聚类或噪声。

定义 11(聚类)　在对象集 D 中的一个非空子集 C 是一个聚类,它必须满足以下两个条件:

(1) $\forall p, q \in D$,如果 $p \in C$ 并且 $p \Leftrightarrow q$,则 $q \in C$

(2) $\forall p, q \in C, p \Leftrightarrow q$

3.2　种子对象的选取

种子对象的选取非常重要,种子对象不能太多,亦不能太少。若太多,就难以发挥算法的效率,反之,如果太少,则种子对象邻域难以比较完全地覆盖其他对象的邻域,从而造成对象丢失,并影响到聚类质量和效率。所以有两个问题需要解决:(1)种子对象应该选多少;(2)如何选择种子对象。

在这里仅考虑 2 维空间数据,但这个方法也很容易推广到 2 维以上的空间数据。对于给定的 ε 和 $MinPts$,先假设对象 p 是核心对象,然后以对象 p 为中心,并以 ε 为半径画圆(如图 1 所示)。在这个圆上,先以对象 p 为原点画坐标系交圆周于 A、B、C 和 D 4 个点,然后画两条分别与 x 轴成 45°和 135°角的两条直径交圆周于 E、F、G 和 H 4 个点。本文把这些点称为代表边界对象(representative boundary objects, RBO)。在每一个象限中有 3 个 RBO,例如在右上象限中有 B、E 和 C,在右下象限中有 A、H 和 B。

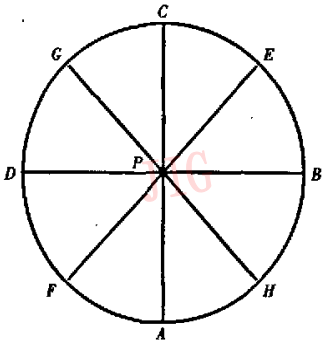


图 1　种子的选取

Fig. 1　Selection of seeds

下面给出了一种从核心对象的邻域中选择种子对象的算法。其基本思想是:对每一个 RBO,可在

它所在的象限内确定与它最近的对象，并把这个对象选为种子，即使一个种子对象到两个以上的 RBO 都是最近的，它也只被选一次。因此在 2 维空间范围内对任一对象的被选种子数最多是 8 个。一般情况下，对 d 维空间，由于有 3^d-1 个 RBO 和 2^d 个象限，因此被选种子数最多为 3^d-1 个，该算法的伪码如下：

```
Finding_seeds(Object,ε,Seeds)
  Initialize Seeds to NULL;
  Identify the RBOs about Object and ε;
  for each RBO do
    MinDist = ∞ ;
  Identify the Quadrant in which the RBO falls;
    for each unclassified object p in the Quadrant do
      distance = dist(RBO,p);
      if(distance≤minDist)
        minDist = distance;
        minObject = p;
      endfor
    Seeds = Seeds∪{minObject};
  endfor
end
```

3.3 算法描述

算法名称：一种改进的基于密度的抽样聚类算法——IDBSCAS 算法（an improved density-based spatial clustering algorithm with sampling）。

输入：2 维平面上的空间数据库（DB），邻域半径 $\varepsilon\geqslant 0$，最小对象个数阈值 $MinPts\geqslant 0$，最小纯度阈值 $MinPur\geqslant 0$.

输出：聚类 Cluster.

算法伪码如下：

```
Algorithm IDBSCAS(DB,ε,MinPts,MinPur)
  ClusterId = nextId(NOISE);
  for each object o in DB do
    if o.ClId = UNCLASSIFIED
        if Expanding_cluster(o,ε,DB,MinPts,MinPur,
ClusterId)
          ClusterId = nextId(ClusterId);
  end
Expanding_cluster(Object,ε,DB,MinPts,MinPur,
ClId):Boolean
  Neighbours = DB.matchingNeighbours(Object,ε);
  if(Neighbours.size < MinPts) or
    (Neighbours.pur < MinPur)
      //Object 为噪声
      DB.changClId(Object,NOISE);
      return false;
  else//Object 为纯核心对象
    DB.changClId(Neighbours,ClId);
    Finding_Seeds(Object,ε,Seeds);
    Seed_list = Seed_list.add(Seeds);
    While Seed_list≠NULL do
    currentP = Seed_list.first();
      Neighbours = DB.matchingNeighbours(currentP,ε);
      if(Neighbours.size ≥ MinPts) and(Neighbours.pur ≥
        MinPur)
      //currentP 为纯核心对象
  Finding_Seeds(currentP,ε,Seeds);
    Seed_list = Seed_list.add(Seeds);
      for each object p in Neighbours do
        if p.ClId = UNCLASSIFIED or NOISE
DB.changClId(p,ClId);
          else
            ClId = p.ClId;//把两个纯密度相连的聚类合并
成一个聚类
      Seed_list.delete(currentP);
        endif
      endwhile
    return true;
  endif
end
```

在 IDBSCAS 算法中，当第 1 个纯核心对象找到后，就可通过 Finding_seeds 函数找出它所对应的种子对象作为聚类扩展用。在随后的聚类扩展中，新种子不断增加到种子集合 Seed_list 中，用于后续聚类的扩展。如此循环执行下去，直到 Seed_list 为空，这表明该聚类扩展完毕；然后 IDBSCAS 算法就另外选取一个没有被标为某个聚类或噪声的对象，重复执行同样的操作。这个操作被重复执行，直到在数据库中的所有对象被标为某个聚类或噪声。与 DBSCAN 算法相比，IDBSCAS 算法主要在以下 3 个方面进行了改进：(1) 可以处理非空间属性；(2) 在 Expanding_cluster 函数中，增加了 ClId = p.ClId 语句，通过它可以把两个纯密度相连的聚类进行合并；(3) 在 Expanding_cluster 函数中加入了 Finding_seeds 函数，用它在纯核心对象邻域中寻找种子对象执行聚类扩展，不需要再对邻域中的每个对象都执行区域查询，这样就大大节省了执行时间。

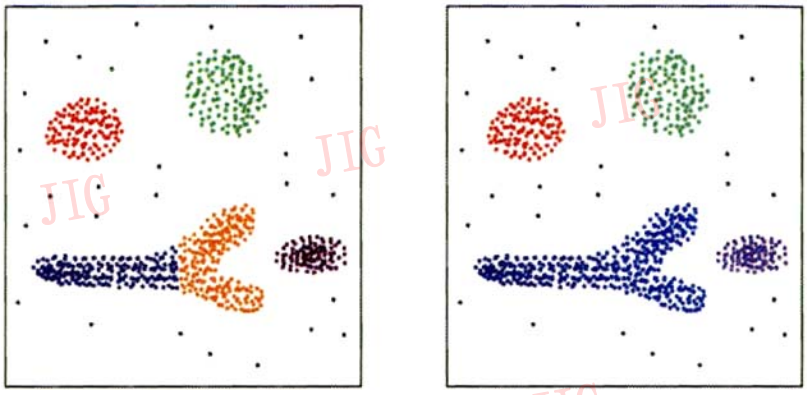
4 实验与分析

本文测试所用的数据为模拟数据和 DBSCAN

算法所使用的测试数据集,测试的硬件环境是:P4-1.5GHz的CPU,主存为256M,硬盘为40G。软件环境是:操作系统为 Microsoft Windows 2000 Professional,算法实现的语言为 C++。

4.1　聚类效果的比较

为了比较聚类的效果,本实验采用了文献[5]中的有203个点的数据集作为测试数据(如图2(a)所示)。在这个数据集中,用不同的颜色表示非空间属性。在其中的"Y"形点集中,它左右两部分的颜色不同,也就是说,它的非空间属性不同。IDBSCAS算法成功地确定了它的5个聚类和噪声,聚类的准确率为100%(如图2(a)所示)。但DBSCAN算法由于没有考虑非空间属性,它把"Y"点集确定为一个聚类,所以它得到的是4个聚类和噪声(如图2(b)所示)。可见,DBSCAN算法聚类的准确率明显比IDBSCAS算法聚类的准确率低。



(a) IDBSCAS 算法的聚类结果　　(b) DBSCAN 算法的聚类结果

图2　测试数据集

Fig. 2　Testing datasets

4.2　执行效率的比较

为了比较IDBSCAS算法和DBSCAN算法的执行效率,本文实验采用了一个对象个数从20 000到100 000个的模拟数据集。从图3可以看出,IDBSCAS算法的效率明显高于DBSCAN算法。在空间索引如 R^*-树的支持下,IDBSCAS算法的时间复杂性和DBSCAN算法一样,也为 $O(n\log n)$。但IDBSCAS算法在执行区域查询的时候采用了抽样技术,由于不需要对核心对象邻域中的每个对象都执行区域查询,所以它比DBSCAN算法节省了不少时间。虽然IDBSCAS算法比DBSCAN算法多用了一个函数 Finding_seeds,但并没有全面提高IDBSCAS算法的时间复杂性。假设邻域中对象个数为 m,对象的维数为 d,则函数 Finding_seeds 的时间复杂性为 $O(md)$,但邻域中对象个数 m 和对象的维数 d 与数据库中对象的个数 n 相比是非常小的。

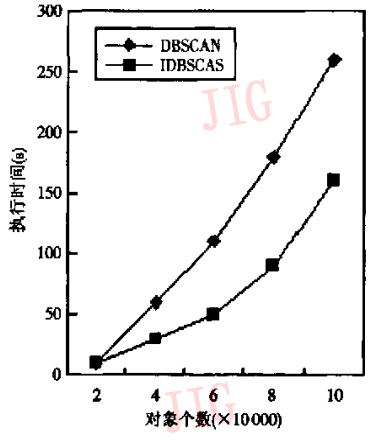


图3　性能比较

Fig. 3　Performance comparison

5　结　论

本文提出了一种改进的基于密度的抽样聚类算法,该算法保持了基于密度聚类算法可以发现任意形状的聚类和对噪声不敏感的优点,而且由于该算法采用了抽样技术,从而使聚类速度有了显著的提高;另外,由于该算法通过引入纯度的概念,在聚类时,不仅考虑了空间属性,而且也考虑了非空间属

性,从而提高了聚类的质量。

由于空间数据不断积累,空间数据库的规模不断扩大,因此如何提高聚类算法的执行效率,改进聚类结果的质量,仍然是有待进一步研究的内容。

参考文献(References)

1 Han Jiawei, Kamber Micheline (Fan Ming, Meng Xiao-feng, *et al* Translate). Data Mining Concepts and Techniques[M]. Beijing: China Machine Press, 2001. [Han Jiawei, Kamber Micheline(范明, 孟小峰等译). 数据挖掘概念和技术[M]. 北京: 机械工业出版社, 2001.]

2 Ng R T, Han Jiawei. CLARANS: A method for clustering objects for spatial data mining[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(5): 1003 ~ 1016.

3 Guha S, Rastogi R, Shim K. CURE: An efficient clustering algorithm for large databases[A]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data[C], Seattle, WA, USA, 1998: 73 ~ 84.

4 Zhang T, Ramakrishna R, Livny M. BIRCH: An efficient data clustering method for very large databases[A]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data [C], Montreal, Canada, 1996: 103 ~ 114.

5 Ester M, Kriegel H, Sander J, *et al*. A density-based algorithm for discovering clusters in large dpatial databases with noise[A]. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining[C], Portland, Maine, USA, 1996: 226 ~ 231.

6 Ankerst M, Breunig M, Kriegel H, *et al*. OPTICS: Ordering points to identify the clustering structure[A]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data[C], Philadelphia, PA, USA, 1999: 49 ~ 60.

7 Sander J, Ester M, Kriegel H, *et al*. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications[J]. Data Mining and Knowledge Discovery, 1998, 2(2): 169 ~ 194.

8 Wang W, Yang J, Muntz R. STING: An statistical information grid approach to spatial data mining[A]. In: Proceedings of the 23th International Conference on Very Large Data Bases[C], Athens, Greece, 1997: 186 ~ 195.

9 Sheikholeslami G, Chatterjee S, Zhang A. WaveCluster: A multi resolution clustering approach for very large spatial databases[A]. In: Proceedings of the 24th International Conference on Very Large Data Bases[C], New York, USA, 1998: 428 ~ 439.

10 Beckmann N, Kriegel H P, Schneider R, *et al*. The R*-Tree: An efficient and robust access method for points and rectangles[A]. In: Proceedings of the ACM SIGMOD International Conference on Management of Data [C], Atlantic City, NJ, USA, 1990: 322 ~ 331.