

一种改进的基于密度的聚类算法

许虎寅, 王治和

(西北师范大学 数学与信息科学学院, 甘肃 兰州 730070)

摘 要: 聚类是数据挖掘领域中的一个重要研究方向, 在基于密度的聚类算法 DBSCAN 的基础上, 提出了一种改进的基于密度的聚类算法, 该算法在核心点的邻域扩展中不再将邻域内的点作为种子点, 而是按顺序选择一个邻域外未被标记的点作为种子点, 然后分不同情况进行相应的聚类扩展, 此算法可以有效减少聚类中核心点邻域重叠区域查询的次数和运行的时间, 实验测试结果也表明该算法聚类的效率和质量明显优于 DBSCAN 算法。

关键词: DBSCAN; 聚类; 密度; 核心点; 邻域

中图分类号: TP301

文献标识码: A

文章编号: 1000-7180(2012)02-0044-04

An Improved Density-based Clustering Algorithm

XU Hu-yin, WANG Zhi-he

(College of Mathematics and Information Science, North West Normal University, Lanzhou 730070, China)

Abstract: Clustering is an important field of research for data mining, an improved density-based clustering algorithm was presented based on DBSCAN. This algorithm chooses unlabelled points outside a core object neighborhood but not inside the neighborhood as seeds to expand clusters. Then, according to the different conditions for the corresponding clustering expand so that execution frequency of region queries could be decreased, and consequently the time cost was diminished. Experimental results also show that the time efficiency and the clustering quality of the new algorithm are greatly superior to those of the original DBSCAN.

Key words: DBSCAN; clustering; density; core; neighborhood

1 引言

聚类(Clustering)是数据挖掘领域中最为活跃的研究分支之一, 在统计学、模式识别、图像处理、机器学习、生物学、市场营销等许多领域有广泛的应用。聚类就是将物理或抽象对象的集合分组成为有类似的对象组成的多个类的过程, 使得同一组内的数据尽可能相似而不同组内的数据尽可能不同。聚类分析不仅可以作为独立的工具来获得数据的分布情况, 而且也可以作为其他算法的预处理步骤。在数据挖掘领域中, 目前研究工作主要集中在为大型数据库的有效和实际的聚类分析寻找适当的方法, 比较活跃的研究主题主要在聚类方法的可伸缩性、对

聚类复杂形状和类型的数据有效性、高维聚类分析技术、以及针对大型数据库中混合数值和分类数据的聚类方法等方面。目前已有的聚类算法有基于划分的聚类算法(KMEANS^[1])、基于层次的聚类算法(BIRCH^[2])、基于密度的聚类算法(DBSCAN^[3])、基于网格的聚类算法(STING^[4])等。以上算法都在试图寻找一种很好的途径来实现对大规模数据库的有效聚类, 但总体来说, 都还存在自身一定的局限性。其中基于密度的聚类算法可以发现任意形状的簇且能够较好地处理噪声数据, 越来越受到广泛的关注。本文在算法 DBSCAN 的基础上, 提出一种改进的基于密度的聚类算法 IDBSCAN, 能够有效地提高聚类的效率和质量。

收稿日期: 2011-05-18; **修回日期:** 2011-07-21

基金项目: 美国福特基金会资助项目(1085-1398); 国家自然科学基金资助项目(10771171); 甘肃省科技攻关基金项目(2GS035-A052-011)

2 DBSCAN 算法

首先给出算法所涉及的一些概念^[5],其中以 Eps 表示空间数据点的邻域半径,MinPts 表示密度阈值, $\text{dist}(p, q)$ 表示点 p 和 q 之间的距离, D 表示数据集.

定义 1 邻域 (Neighborhood): 数据空间中的任意一点 p 的邻域是以该点为圆心,以 Eps 为半径的圆区域内包含的点的集合,记作 $N_{\text{Eps}}(p) = \{q \in D \mid \text{dist}(p, q) \leq \text{Eps}\}$. 将数据空间中某一点 p 的邻域内所包含的点的数目称为该点 p 的密度,对于给定的密度阈值 MinPts,如果点 p 的密度大于 MinPts,则称该点为核心点,反之称为非核心点.

定义 2 直接密度可达 (directly density reachable): 若点 p 和 q 满足: (1) p 在 q 的邻域中,即 $p \in N_{\text{Eps}}(q)$; (2) q 是核心点,即 $|N_{\text{Eps}}(q)| \geq \text{MinPts}$,则称点 p 从点 q 直接密度可达.

定义 3 密度可达 (density reachable): 存在一串对象 $p_1, p_2, \dots, p_n$, 其中从 p_i 可以直接密度可达 p_{i+1} 其中 $p_1 = p, p_n = q$, 则称 q 从 p 密度可达.

定义 4 密度连接 (density connected): 若存在点 o , 使得点 p 和点 q 都从 o 密度可达, 则称 p 和 q 是密度连接的.

定义 5 类 (Cluster): 数据集 D 的非空集合 C 是一个类, 当且仅当 C 满足条件: (1) 对于空间中的任意点 p 和 q , 若 $p \in C$, 且 q 从 p 密度可达, 则 $q \in C$; (2) 对于空间中的任意点 p 和 q , 有 $p \in C$ 和 $q \in C$, 则 p 和 q 是密度连接的.

定义 6 噪声 (Noise): 将数据集 D 中不属于任何类的点称为噪声点.

DBSCAN 算法将类看作是数据空间中被低密度区域分割开的高密度对象区域. 其基本思想是: 按某种顺序考察数据集 D 中的某一个点 p , 若 p 是核心点, 则通过区域查询得到该点的邻域, 如果邻域中的点和 p 同属于一个类, 那么这些点将作为下一轮的考察对象 (也即种子点), 并通过不断地对种子点进行区域查询来扩展它们所在的类, 直到找到一个完整的类, 然后, 依此过程寻找其他的类. 最后剩下的不属于任何类的点为噪声点. DBSCAN 算法可以挖掘任意形状的聚类, 对数据输入顺序不敏感, 并且具有处理异常数据 (噪音) 的能力. 对具有 N 个样本的数据库, 该算法的时间复杂度为 $O(N^2)$; 在空间索引如 R^* 一树的支持下, 其复杂度为 $O(N \log N)$. 需要指出的是, DBSCAN 算法没有考虑建立空间索

引的时间, 而建立索引通常需要消耗大量的时间. 因此需要寻求一种更可靠有效的方法来解决此问题.

3 改进的基于密度的聚类算法 (IDBSCAN)

3.1 算法的思想

在 DBSCAN 算法中, 核心点的邻域扩展中每查询完一个核心点的邻域后, 然后选择该点邻域内的每一个点作为种子点继续扩展查询, 但是许多核心点的邻域是相互重叠的, 这样的话就没必要对邻域中的每一个点都要进行邻域查询, 所以在新算法中, 不再将邻域内的点作为种子点, 而是按顺序将邻域外的一个距离核心点最近的未标记的点作为种子点继续考察.

IDBSACN 算法的基本思想是: 首先对样本集按某一维排序, 然后从排序索引最小的点 p 开始搜索它的邻域 (以下均以图 1 为例说明), 如果 p 是核心点, 则建立新类, 并将 p 邻域内的所有点用 p 的类标号标记, 接着再按顺序在 p 的邻域外找一个未作类标记的点 q 进行邻域扩展, 分以下几种类型:

(1) 如果 q 不是核心点, 且 q 的邻域与 p 的邻域不重叠或重叠区域不存在核心点, 则暂时将 q 标为噪声点;

(2) 如果 q 不是核心点, 且 q 的邻域与 p 的邻域重叠区域存在核心点, 则用 p 的类标号将 q 标记;

(3) 如果 q 是核心点, 且 q 的邻域与已知类 p 的邻域没有重叠, 则建立新类 q , 将 q 邻域内的所有点用 q 的类标号标记;

(4) 如果 q 是核心点, 且 q 的邻域与 p 的邻域有重叠且重叠区域存在核心点, 则用 p 的类标号将 q 邻域内的所有点标记, 同时如果 q 与多个已作不同标号的核心点的邻域重叠且在重叠区域内存在核心点, 则将它们合并为同一类;

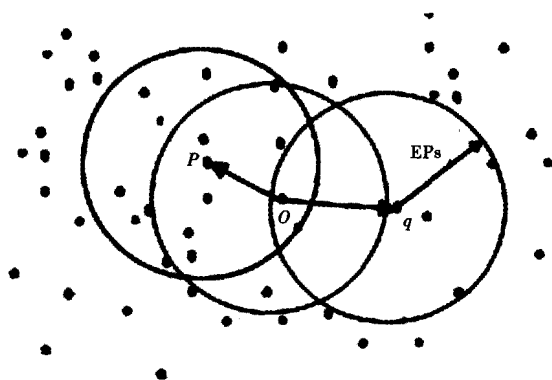


图1 核心点邻域重叠

(5) 如果 q 是核心点, 且 q 的邻域与 p 邻域的重叠区域内存在点且都是非核心点, 建立新类 q , 将重叠区域内离 q 的距离小于离 p 中最近核心点距离的点用 q 的类标号标记;

重复上述过程, 直到所有的点都被处理. 算法的部分伪代码如下:

```
IDBSCAN(Points, Eps, MinPts)
{ Ordering(Points);
  ClusterId=NOISE;
  For(i from 1 to Points.size)
  { Point=Points.get(i);
    IF Point.CId==UNCLASSIFIED {
      seeds=Points.RegionQuery(Point, Eps);
      IF (seeds.size < MinPts) // q 不是核心点
      { OldClusterId=getfirstId(seeds);
        IF OldClusterId<>UNCLASSIFIED
        { Seeds1=getfirstseeds(seeds);
          IF Points.IsCore(seeds1, Eps, MinPts)
          Points.ChangeCId(Point, OldClusterId);
        }
      }
      ELSE Points.ChangeCId(Point, NOISE);
      // q 是噪声点
    }
    ELSE // q 是核心点
    { OldClusterId=getfirstId(seeds);
      IF OldClusterId==UNCLASSIFIED // 没有重叠
      { ClusterId=nextId(ClusterId);
        Points.ChangeCId(seeds, ClusterId);
      }
      Else // 有重叠
      { Seeds2=getfirstseeds(seeds)
        IF(Points.IsCore(seeds2, Eps, MinPts))
        Points.MergeClu(seeds, OldClusterId);
        ELSE ClusterId=nextId(ClusterId);
          Points.changeCId(seeds, ClusterId);
        While(seeds2<>empty)
        { cp=seeds2.first();
          Points.changeCId(cp, ClusterId, OldClusterId);
          Seeds2.delete(currenttp);
        }
      }
    }
  }
}
```

其中函数 Ordering() 对样本集按指定维排序; getfirstId() 返回某一集合内已标记的索引最小的类标号, 如果该集合内所有点都未标记则返回 UNCLASSIFIED; getfirstseeds() 返回某一集合内已标记的点的集合; 函数 Arrange() 最终整理数据聚类结果; 函数 Points.IsCore() 用来判断指定的数据集中是否存在核心点, 函数 Points.MergeClu() 进行子类合并, 其他的函数均与 DBSCAN 算法的函数

一样, 在 DBSCAN 算法中, 要求用户指定一个全局参量 Eps, 这导致不能较好地处理类密度不均一的样本集. 为了解决这一问题, 我们采用核聚类^[6]的思想对样本集进行非线性变换使样本集的分布尽可能地均匀, 然后再用全局参量 Eps 聚类, 同时, 在算法的邻域扩展中, 上述类型(5)的方法能够有效的动态适应密度不均匀的聚类, 可以得到较好的结果, 测试实验也证明了这一点.

3.2 算法的性能分析

基于以上思想, 算法 IDBSCAN 的主要运行时间是执行函数 Points.regionQuery(), 如果采用了 R⁺ 树建立空间索引结构, 则函数执行一次的时间度是 $O(\log N)$. 由于新算法中邻域需要搜索查询的点并不多, 则执行区域查询函数的次数远小于 N , 因此邻域查询的时间复杂度远小于 $O(N \log N)$. 函数 Points.IsCorePoint() 的作用也只是判断是否存在核心点, 而不需要搜索整个邻域, 并且只是判断两个邻域的重叠区域, 所以其时间消耗可以不计, 因此总的复杂度要远小于 $O(N \log N)$, 在下面的实验中也证明了 IDBSACN 的运算效率要远高于算法 DBSCAN.

4 算法测试

对算法 IDBSCAN 的性能进行测试, 并将测试结果与 DBSCAN 以及周水庚等人提出的 FDBSCAN 做了比较. 算法测试是在一台普通 PC 机上进行, 其配置为 P4 2.4 GHz CPU、512 MB 内存、80 GB 硬盘, 算法用 MATLAB 7.5 实现. 测试数据集同时使用了真实数据和自行构造的模拟数据, 真实数据用的是 SEQUOIA2000 数据库, 该数据库也被文献[3]等用于对 DBSCAN 算法的性能测试, 典型的测试结果分别如图 2 所示.

图 2 中同时将 DBSCAN 算法和 FDBSCAN 算

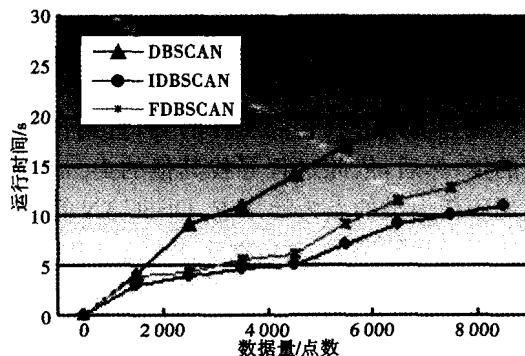
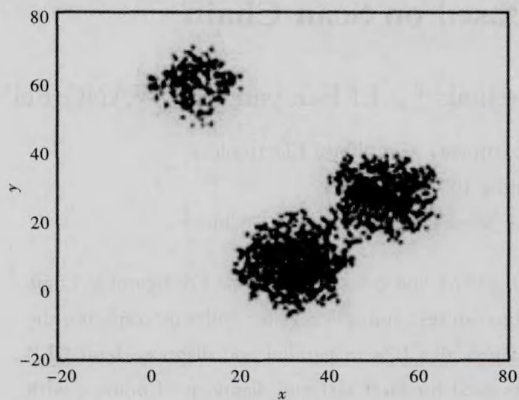
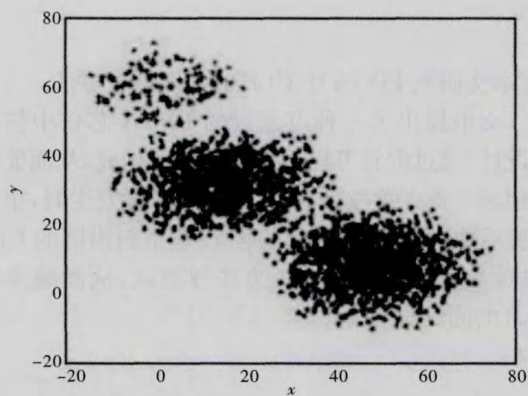


图 2 三种算法针对不同大小数据量的聚类时间

法在 SEQUOIA2000 数据库上针对不同数据量样本的执行时间和 IDBSCAN 算法的执行时间做了比较,从图 2 中可以看到,算法 IDBSCAN 比上述两种算法执行的速度快,而且可以看出随着样本数据量的增大,这种速度上的差距也越来越明显,这也表明算法 IDBSACN 对样本集的大小具有很好的可扩展性.图 3 和图 4 所示为 DBSCAN 算法和 IDBSCAN 算法对自行构造的两个数据集针对不同参数 Eps 以及 MinPts 的聚类结果,每个数据集都包含三个子类,图 3 是算法 DBSCAN 的聚类结果,可以看出罗近的子类并没有被很好的分开,聚类质量较差,图 4 是 IDBSCAN 算法聚类分析后的结果,数据集被正确地分为三个子类.测试结果说明采用 IDBSACN 算法能显著地提高聚类质量,另外,针对密度分布不均匀的数据集聚类,IDBSCAN 算法表现出了更好的动态适应性.测试结果说明采用 IDBSACN 算法能显著地提高聚类质量,另外,针对密度分布不均匀的数据集聚类^[7],IDBSCAN 算法表现出了更好的动态适应性.

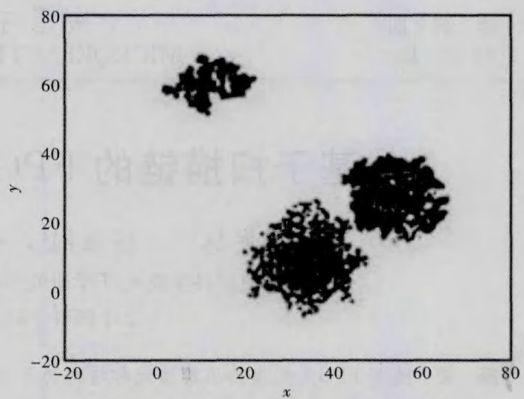


(a) Eps=0.02, MinPts=4

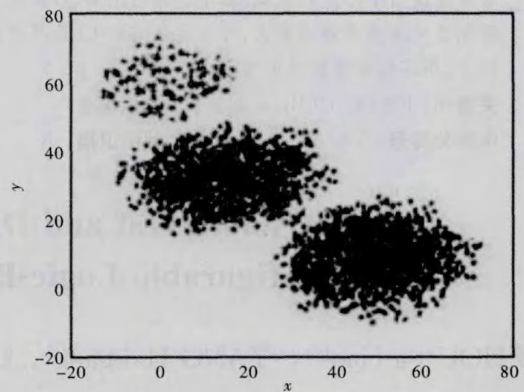


(b) Eps=0.03, MinPts=4

图 3 算法 DBSCAN 的聚类结果



(a) Eps=0.02, MinPts=4



(b) Eps=0.03, MinPts=4

图 4 算法 IDBSACN 的聚类结果

5 结束语

本文提出了一种改进的基于密度的聚类算法 IDBSCAN,该算法在很大程度上减少了核心点邻域查询次数、降低了聚类分析的时间复杂度、提高了聚类的准确度,并能很好的适应密度分布不均匀的数据集聚类.下一步的工作是研究如何进一步减少聚类参数 Eps 以及 MinPts 对聚类结果的敏感性,以降低算法对参数的依赖性.

参考文献:

- [1] Parsons L, Haque E, Liu H. Subspace clustering for high dimensional data: a review [J]. SIGKDD Explorations, 2004, 6 (1): 90-105.
- [2] Guhas, Rastogir, Shim K. Cure: an efficient clustering algorithm for large databases[C]//Proceedings of the ACM SIGMOD international conference on Management of data. New York: ACM Press, 1998:73-84.

(下转第 53 页)

需求,能够在一个TS中实现所有CLB的并行测试,相比于当前的BIST方法,TS数目减少了一半以上,减少了CLB的测试时间.实验结果表明,本文方法可以处理多个CLB故障的情况,并可以在故障测试的同时完成对故障CLB单元的定位,文中采用的CLB模型,在固定故障模型下,只需4次配置,就能达到100%的故障覆盖率.

参考文献:

- [1] Xilinx, Inc. Virtex platform FPGAs: complete data sheet (V3.4) [EB/OL]. [2010-07-04]. <http://www.xilinx.com>.
- [2] Altera Inc. Cyclone device handbook [EB/OL]. [2010-08-20]. <http://www.altera.com>.
- [3] Toutounchi S, Lai A. FPGA test and coverage[C]//ITC International Test Conference. Baltimore, 2002: 599-607.
- [4] Tang Ming, Zhang Guoping. A testing approach for xilinx FPGA's CLB[C]//ICSICT'06, 8th International Conference on Solid-State and Integrated Circuit Technology. Shanghai, 2006: 2158-2160.
- [5] Miron Abramovici, Stroud C. BIST-based test and diagnosis of FPGA logic blocks[J]. IEEE Transactions on Very Large Scale Integration Systems, 2001, 9(1): 159-172.
- [6] Armin Alaghi, Mahnaz Sadoughi Yarandi, Zainalabedin Navabi. An optimum ORA BIST for multiple fault FPGA look-up yable testing[C]//15th Asian Test Symposium. Calcutta, 2006: 12-40.
- [7] Lu Shyuekung, Wu Hungchin. Testing and diagnosis techniques for LUT-based FPGA's[C]//Proceedings of the 13th Asian Test Symposium. Kenting, 2004: 12-34.
- [8] Renovell M, Faure P, Portal J M. IS-FPGA: a new symmetric FPGA architecture with implicit scan[C]//International Test Conference(ITC). Baltimore, 2001: 924-931.

作者简介:

周发标 男,(1984—),博士研究生.研究方向为大规模集成电路可测性设计.

杨海钢 男,(1960—),博士,研究员,博士生导师.研究方向为高速可编程逻辑芯片设计技术、数模混合信号SOC设计技术.

秋小强 男,(1973—),博士研究生.研究方向为大规模集成电路设计.

李凡阳 男,(1982—),博士研究生.研究方向为大规模集成电路设计.

王飞 男,(1982—),助理研究员.研究方向为大规模集成电路设计.

(上接第47页)

- [3] Ester M, Hans2peter K R Iegel, Sander J, et al. A density based algorithm for discovering clusters in large spatial databases with noise[C]//Proceeding the 2nd international conference on Knowledge discovery and data mining(KDD). Portland, 1996. 226-231.
- [4] 马帅,王腾蛟,唐世渭,等.一种基于参考点和密度的快速聚类算法[J].软件学报,2003,14(6):1089-1095.
- [5] 赵杰,杨柳.聚类分析算法dBScan的改进与实现[J].微电子学与计算机,2009,26(11):189-192.
- [6] 张莉,周伟达,焦李成.核聚类算法[J].计算机学报,2002,25(6):587-590.
- [7] 孙凌燕,杨明,任建斌.一种基于相对密度的快速聚类算法[J].微电子学与计算机,2009,12(26):109-111.

作者简介:

许虎寅 男,(1986—),硕士研究生.研究方向为数据库技术及应用(数据挖掘).

王治和 男,(1965—),教授,CCF高级会员.研究方向为数据库技术及应用(数据挖掘).