一种基于参考点和密度的快速聚类算法*

马 帅[1+], 王腾蛟[1], 唐世渭[1,2], 杨冬青[1], 高 军[1]

[1](北京大学 计算机科学技术系,北京 100871)

[2](北京大学 视觉与听觉信息处理国家重点实验室,北京 100871)

A Fast Clustering Algorithm Based on Reference and Density

MA Shuai[1+], WANG Teng-Jiao[1], TANG Shi-Wei[1,2], YANG Dong-Qing[1], GAO Jun[1]

[1](Department of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](National Laboratory on Machine Perception, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: 86-10-62756374, E-mail: mashuai@db.pku.edu.cn; mashuai@cis.pku.edu.cn

http://www.pku.edu.cn

Received 2002-04-19; Accepted 2002-07-02

Ma S, Wang TJ, Tang SW, Yang DQ, Gao J. A fast clustering algorithm based on reference and density. *Journal of Software*, 2003,14(6):1089~1095.

http://www.jos.org.cn/1000-9825/14/1089.htm

Abstract: The efficiency of data mining algorithms is strongly needed with data becoming larger and larger. Density-Based clustering analysis is one kind of clustering analysis methods that can discover clusters with arbitrary shape and is insensitive to noise data. In this paper, a new kind of clustering algorithm that is called CURD (clustering using references and density) is presented. The creativity of CURD is capturing the shape and extent of a cluster by references, and then analyzes the data based on the references. CURD keeps the ability of density based clustering method's good features, and it can reach high efficiency because of its linear time complexity, so it can be used in mining very large databases. Both theory analysis and experimental results confirm that CURD can discover clusters with arbitrary shape and is insensitive to noise data. In the meanwhile, its executing efficiency is much higher than traditional DBSCAN algorithm based on R*-tree.

Key words: clustering; density; high dimension; reference; data mining

摘 要: 数据的规模越来越大,要求数据挖掘算法有很高的执行效率.基于密度的聚类是聚类分析中的一种,其主要优点是发现任意形状的聚类和对噪音数据不敏感.提出了一种新的基于参考点和密度的 CURD(clustering using references and density)聚类算法,其创新点在于,通过参考点来准确地反映数据的空间几何特征,然后基于参考点对数据进行分析处理.CURD 算法保持了基于密度的聚类算法的上述优点,而且 CURD 算法具有近似线性的时间复杂性,因此 CURD 算法适合对大规模数据的挖掘.理论分析和实验结果也证明了 CURD 算法具有处

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA483440 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.G1999032705 (国家重点基础研究发展规划(973)); the Foundation of the Innovation Research Institute of PKU-IBM of China (北京大学-IBM 创新研究院项目)

第一作者简介: 马帅(1975－),男,山东潍坊人,博士生,主要研究领域为数据库,信息系统.

理任意形状的聚类、对噪音数据不敏感的特点,并且其执行效率明显高于传统的基于 R*-树的 DBSCAN 算法.

关键词: 聚类;密度;高维;参考点;数据挖掘

中图法分类号: TP181 **文献标识码**: A

聚类分析是数据挖掘领域中的一项重要的研究课题.它既可以作为一个单独的工具以发现数据库中数据分布的一些深入的信息,也可以作为其他数据挖掘分析算法的一个预处理步骤.聚类分析同时也是一个具有很强挑战性的领域,它的一些潜在应用对分析算法提出了特别的要求.下面是关于聚类的一些典型的要求[1]:可扩展性、处理不同数据类型的能力、发现具有任意形状的聚类的能力、输入参数对领域知识的最小限度的依赖性、能够处理异常数据的能力、数据输入顺序对聚类结果的不敏感性、处理高维数据的能力、基于约束的聚类以及聚类结果的可解释性和可用性.迄今为止,人们已经提出了许多聚类算法,如 K-MEANS[2],DBSCAN[3],CURE[4],CLIQUE[5]和 C^2P[6],所有这些算法都试图通过不同的途径实现对大规模数据库的有效聚类,但总的来说,都没有取得理想的效果.可以说,对大规模、高维数据库的高效聚类分析仍然是一个有待研究的开放问题.

(1) 相关工作

K-MEANS 是一种基于划分的聚类算法,它试图找出满足某一特定标准的 k 个划分(聚类),通常采用平方差的标准.K-MEANS 算法的时间复杂度为 $O(tkn)$,其中 n 是数据的总数,k 是聚类的个数,t 是算法循环的次数,通常有 $k,t<<n$,因此算法的效率很高.其缺点是:需要输入最终结果的聚类个数 k,而判断一个未知数据集的划分个数通常是很困难的;k 个初始点的选择对最终的聚类结果影响很大;基于划分的聚类算法为了减少平方差会将一个大的聚类分裂为几个小的聚类[4].

CURE 是一种自底向上的层次聚类算法,首先将输入的每个点作为一个聚类,然后合并相似的聚类,直到聚类的个数为 k 时为止.在 CURE 中指出,基于中心点的方法和所有的点的距离计算方法都不适合非球形或任意形状的聚类,因此 CURE 采用了折衷的方法,即用固定数目的点表示一个聚类,从而提高了算法挖掘任意形状的聚类的能力.CURE 算法的时间复杂性为 $O(n\cdot n)$(低维数据)和 $O(n\cdot n\cdot \log n)$(高维数据),算法在处理大量数据时必须基于抽样、划分等技术.

DBSCAN 是一种基于密度的聚类算法,其基本思想是:对于一个聚类中的每一个对象,在其给定半径的邻域中包含的对象不能少于某一给定的最小数目,然后对具有密度连接特性的对象进行聚类.在该算法中,发现一个聚类的过程是基于这样的事实:一个聚类能够被其中的任意一个核心对象所确定.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.该算法的时间复杂性为 $O(n\cdot n)$;在空间索引如 R*-树的支持下,其复杂性为 $O(n\cdot \log n)$.需要指出的是,DBSCAN 算法没有考虑建立索引的时间,而建立索引通常需要消耗大量的时间.

CLIQUE 是一种基于网格和密度的聚类算法,具有网格类算法效率高的优点,并且可以处理高维的数据.但是不可避免地具有网格聚类算法的缺点,即在划分网格时没有或者很少考虑数据的分布,而且用一个网格内的统计信息来代替该网格内的所有点,从而导致了聚类质量的降低.

(2) 本文的工作

本文提出了一种基于参考点和密度的快速聚类算法 CURD(clustering using references and density).CURD 算法受 CURE 算法的启发,采用一定数目的参考点(references)来有效地表示一个聚类区域和形状.与 CURE 算法不同的是,参考点是虚拟的点,不是实际输入数据的点,而且一个聚类中的参考点数目是不固定的.CURD 算法采用密度的方法屏蔽异常数据(噪音)对算法的影响,与 DBSCAN 算法在处理任意形状聚类方面的能力接近.CURD 算法具有和 K-MEANS 算法相同的时间复杂性,因此效率很高.此外,采用距离的方法可以将对高维数据的处理转换到一维空间[7,8],在这个意义上,CURD 算法可以处理高维数据,同时 CURD 算法的参考点充分考虑了输入数据的空间几何特征,因此聚类的质量要高于网格类算法.多维空间与二维空间的距离计算相似,为了方便地描述算法,我们在本文中以二维空间为例来分析 CURD 算法.

本文首先介绍 CURD 算法,然后从多个角度具体分析 CURD 算法的性能,最后进行讨论.

1 CURD 聚类算法

1.1 概 念

定义 1(点的密度). 对空间中任意点 p 和距离 *Radius*,以 p 点为中心,半径为 *Radius* 的区域内的点的个数称为点 p 基于距离 *Radius* 的密度(density),记作 *Density*(p,*Radius*).

定义 2(参考点). 对空间中任意点 p、距离 *Radius* 和阈值 t,如果满足 *Density*(p,*Radius*)>=t,则称 p 为参考点(reference),同时我们称 t 为密度阈值(density threshold).

参考点不是实际输入数据中的点,而是虚拟的点或称为假想点.

定义 3(代表区域). 每个参考点代表了以该点为圆心、半径为 *Radius* 的圆形区域,我们称该区域为参考点的代表区域(representing region).

定义 4(邻接参考点). 给定距离 *Radius* 和阈值 t,如果参考点 p,q 满足 *Dist*(p,q)<=2·*Radius*,即 p 和 q 之间的距离小于或等于 2 倍于 *Radius*,则称 p,q 为邻接参考点(neighboring references).

实际上,如果两个参考点的代表区域相切、相交或重合(圆心距离小于或等于半径之和),那么这两个参考点就是邻接参考点.

1.2 CURD算法

在本节中,我们将详细介绍 CURD 算法,算法的整体框架如图 1 所示.CURD 算法首先寻找可以准确反映输入数据空间几何特征的参考点;接着建立参考点与其代表区域内点的映射;然后对参考点进行分类,每一类参考点构成了一个聚类的基本信息;属于同一类的参考点代表区域内的点的集合,最终构成了一个聚类.

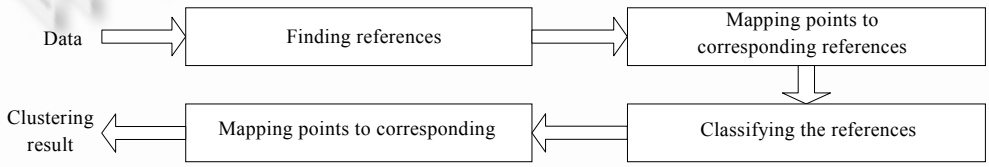


Fig.1 CURD algorithm

图 1 CURD 算法

1.2.1 数据结构

基于参考点的重要性,我们首先介绍参考点的数据结构.每个参考点包含 4 部分信息:参考点的 X 坐标 X_m 和 Y 坐标 Y_m;参考点代表区域内的所有点的 X 坐标之和 X_s 和 Y 坐标之和 Y_s;参考点的密度(参考点代表区域内点的总数)N_s;参考点代表区域内的点的集合 P_s.

1.2.2 寻找参考点

对参考点的寻找分为两步:第 1 步,寻找候选参考点集;第 2 步,将不符合密度阈值条件的候选参考点筛选过滤掉,剩下的候选参考点是参考点.对候选参考点的筛选非常简单,因此我们主要描述寻找候选参考点的过程(如图 2 所示).从图 2 可以看出,生成候选参考点的过程实际上是对 Single_Scan 过程(如图 3 所示)和 Regenerate_Candidate_References 过程的反复调用,因此我们将首先加以介绍.

```
Finding_Candidate_References (PointSet, dRadius, Iterate){
//Input: data set: PointSet, distance: dRadius, number of iterate times: Iterate
//Output: candidate reference set: CandidateReferenceSet
Begin
Step 1. CandidateReferenceSet=∅,I=1
Step 2. While (I<Iterate) {
Step 3. Single_Scan (PointSet, dRadius, CandidateReferenceSet)
Step 4. Regenerate_Candidate_References (CandidateReferenceSet)
Step 5. I++}
Step 6. Single_Scan (PointSet, dRadius, CandidateReferenceSet)
End
```

Fig.2 Finding_Candidate_References procedure

图 2 Finding_Candidate_References 过程

每次调用 Single_Scan 过程都会产生新的候选参考点集,其主要操作是将点 p 的坐标信息加入到候选参考点集中.如果点 p 与所有的候选参考点的距离都大于 *Radius*,则生成一个新的候选参考点;否则将点 p 的坐标信息加入到所有与点 p 距离小于等于 *Radius* 的候选参考点,如图 3 所示.

```
Single_Scan (PointSet, dRadius, CandidateReferenceSet)
//Input: data set: PointSet, distance: dRadius, candidate reference set: CandidateReferenceSet
//Output: candidate reference set: CandidateReferenceSet
Begin
Step 1. I=1
Step 2. While (I<=PointSet.Size) {
Step 3. For each point Pi:
        Adding Pi's information to every candidate reference R, whose distance with p is equal to
        or less than the distance dRadius: R.Xs=R.Xs+Pi.X, R.Ys=R.Ys+Pi.Y, R.Ns=R.Ns+1
        If the distances between Pi and all candidate references are larger than the distance
        dRadius, a new candidate reference R is generated: R.Xm=Pi.X, R.Ym=Pi.Y, R.Xs=Pi.X,
        R.Ys=Pi.Y, R.Ns=1, which is added to CandidateReferenceSet
Step 4. I++}
End
```

Fig.3 Single_Scan procedure

图 3 Single_Scan 过程

过程 Regenerate_Candidate_References 生成新的候选参考点,将候选参考点代表区域内的点的中心作为新的候选参考点.在 Single_Scan 过程中已经保存了候选参考点代表区域内所有点的 X 坐标和、Y 坐标和以及参考点的密度.对于所有候选参考点 R,用新的候选参考点 R'代替,其中 $R'.X_m=R.X_s/R.N_s$,$R'.Y_m=R.Y_s/R.N_s$,$R'.X_s=0$,$R'.Y_s=0$,$R'.N_s=0$.

寻找候选参考点的过程需要多次调用 Single_Scan 过程和 Regenerate_Candidate_References 过程,每次调用都趋向于产生更加能够准确表示输入数据空间几何特征的候选参考点.这样,经过一系列的优化,使得产生的候选参考点就可以粗略地反映输入点集的空间几何特征,然后筛选候选参考点,将密度小于密度阈值 t 的候选参考点过滤掉,实际上是将代表噪音数据的候选参考点过滤掉,这样得到的参考点集就可以准确地反映输入数据的空间几何特征.

1.2.3 建立参考点与对应数据的映射

定理 1. 在 CURD 算法中,点 p 具体与哪个参考点之间建立映射不会对聚类的结果产生影响,只需满足点 p 与该参考点的距离小于等于 *Radius*.

证明:如果参考点 R_1 和 R_2 与点 p 的距离都小于等于 *Radius*,由三角形边长定理:任意一边的边长小于其余两边的边长之和可知,参考点 R_1 和 R_2 之间的距离小于 2 倍的 *Radius*,因此参考点 R_1 和 R_2 是相邻参考点.在多维空间中,由于任意不在同一直线上的 3 个点形成一个平面,因此上面的三角形定理仍然适用.当点 p,R_1 和 R_2 在同一直线时,R_1 和 R_2 之间的距离等于 2 倍的 *Radius*,R_1 和 R_2 是邻接参考点.CURD 算法中相邻参考点属于同一个聚类的基本信息,因此点 p 无论与参考点 R_1 还是 R_2 建立映射,点 p 最终都属于同一个聚类,定理 1 得证. □

Mapping_References_and_Points 过程建立参考点与其代表区域内输入数据的对应关系.理论上,对于任意点 p 将其与所有参考点中距离最近的参考点建立映射比较合理,由定理 1 可知,点 p 具体与哪个参考点之间建立映射不会对聚类的结果产生影响,只要点 p 与该参考点的距离小于等于 *Radius*.因此,在 Mapping_References_and_Points 过程中,点 p 顺序地与参考点集中的参考点进行比较,如果两者之间的距离小于等于 *Radius*,则在该参考点和点 p 之间建立映射,这样的处理可以提高算法的效率,如图 4 所示.

1.2.4 对参考点进行分类

在给定距离 *Radius* 和阈值 t 产生的参考点集中,如果两个参考点 R_1 和 R_2 的距离小于等于 2 倍的 *Radius*,则 R_1 和 R_2 是邻接参考点.这样我们就可以用无向图来描述生成的参考点集,图的顶点是参考点,相邻参考点之间有一条边,处于同一个连通子图中的参考点组成一类.利用图的广度优先搜索算法(BFS)就可以将处于同一连通子图的顶点找出来,因此我们对参考点进行分类的过程不再详细地加以描述.处于同一个分类中的参考点组成了一个聚类的基本信息,可以准确地表示该聚类的空间几何特征.

```
Mapping_References_and_Points (PointSet, ReferenceSet, t, dRadius)
//Input: data set: PointSet, reference set: ReferenceSet, density threshold: t, distance: dRadius
//Output: reference set: ReferenceSet
Begin
Step 1. Each reference, R.Ps=∅, I=1
Step 2. While (I<=PointSet.Size) {
Step 3. For each point Pi:
        If the distance between Pi and some reference R is equal to or less than dRadius,
        R.Ps=R.Ps∪{Pi}.
        Otherwise, the distances between Pi and all the references are larger than dRadius, Pi
        is considered as noise and is discarded.
Step 4. I++}
End
```

Fig.4 Mapping_References_and_Points procedure

图 4 Mapping_References_and_Points 过程

1.2.5 建立聚类与对应数据的映射

由于参考点和数据之间已经建立了映射,在对参考点进行分类过程的同时实际上已经建立了聚类和参考点之间的映射,属于同一个分类中的参考点代表区域内的输入数据的集合就是一个聚类,因此聚类和数据之间的映射非常简单.

1.2.6 算法的时空复杂性分析

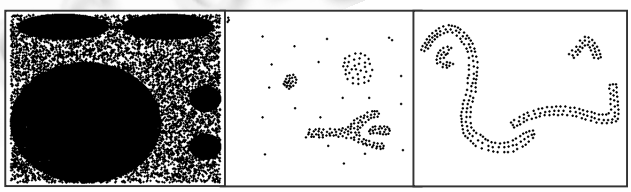
我们假定数据量为 N,在生成候选参考点集的过程中,候选参考点的数目最大为 K,循环次数为 I,参考点的数目为 M,聚类的个数为 C.

容易知道,Single_Scan过程的时间复杂性为 $O(K\cdot N)$,Regenerate_Candidate_References过程的时间复杂性为 $O(K)$,因此寻找候选参考点的过程的时间复杂性为 $O(I\cdot K\cdot N+(I-1)\cdot K)$,对候选参考点的筛选过滤可以在 $O(K)$ 内完成,这样,整个寻找参考点的过程的时间复杂性为 $O(I\cdot K\cdot N+(I-1)\cdot K)+O(K)$.每个点可以在 $O(M)$时间内找到对应的参考点,因此建立参考点与对应数据的映射的时间复杂性为 $O(M\cdot N)$.采用图的广度优先算法对参考点进行分类的时间复杂性为 $O(M\cdot M)$.聚类和数据之间的映射实际上在对参考点进行分类后就已经建立了.从上面的分析可以看出,CURD 聚类算法的时间复杂性为 $O(I\cdot K\cdot N+(I-1)\cdot K)+O(K)+O(M\cdot N)+O(M\cdot M)$,通常地,$K,I,M<<N$,因此,CURD 聚类算法的时间复杂性可以近似为 $O(I\cdot K\cdot N+M\cdot N)$,具有和 K-MEANS 算法相同的线性时间复杂性,算法的执行效率很高.

容易知道,算法的空间复杂性为 $O(N)+O(K)+O(C)$.

2 算法的性能分析

我们的测试数据集来自 CURE 算法中的 data set 1(DS1)、DBSCAN 算法中的 database 3(DS2)和采用 Dr. Jörg Sander 提供的 DBSCAN 程序仿照 DBSCAN 中 database 2 生成的数据集(DS3),数据量分别为 100000,203,306(如图 5 所示).我们实验的硬件环境是 IBM Netfinity 5500:两个 X86 Family 6 Model 10 Stepping 1 GenuineIntel~700Mhz 的 CPU、主存为 512M、硬盘为 20G;软件环境是:操作系统为 Microsoft Windows 2000 Server,算法采用 Java 编写.



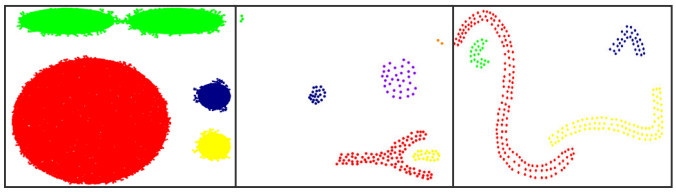
(a) DS1 (b) DS2 (c) DS3

Fig.5 Data sets

图 5 数据集

2.1 聚类效果的比较

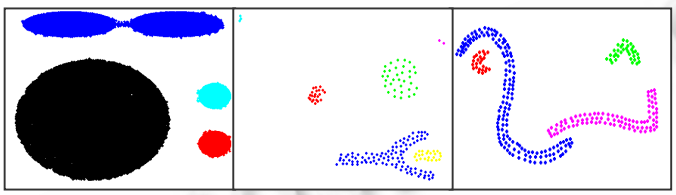
CURD 算法的参数 *Radius* 和 t 与 DBSCAN 算法的 *Eps* 和 *MinPts* 有着相似的特点,此外,DBSCAN 算法也是一种基于密度的聚类算法,因此选择该算法与 CURD 算法作比较.图 6、图 7 分别是 DBSCAN 算法和 CURD 算法对数据集 DS1,DS2 和 DS3 的聚类结果.图 8、图 9 是 CURD 算法发现的候选参考点和参考点.



(a) DS1 (b) DS2 (c) DS3

Fig.6 Clustering results of DBSCAN

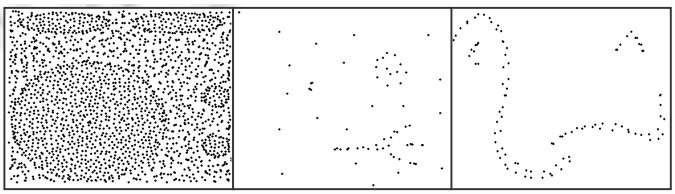
图 6 DBSCAN 算法聚类结果



(a) *Radius*=0.5, *t*=60 (b) *Radius*=2, *t*=2 (c) *Radius*=2, *t*=2

Fig.7 Clustering results of CURD

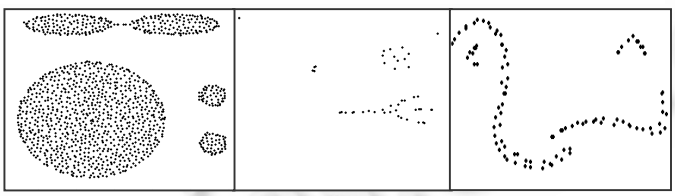
图 7 CURD 算法聚类结果



(a) DS1 (b) DS2 (c) DS3

Fig.8 Candidate references

图 8 候选参考点



(a) DS1 (b) DS2 (c) DS3

Fig.9 References

图 9 参考点

可以看出,CURD 算法可以很好地处理任意形状的数据集,同时也较好地屏蔽了异常数据的影响.比较图 6 和图 7 也可以发现,CURD 算法和 DBSCAN 算法聚类的结果非常接近.从图 8 可以看出,候选参考点基本上反映了输入数据的几何空间特征,但是仍然受到噪音数据的影响;从图 9 可以看出,经过对候选参考点的筛选过滤,得到的参考点准确地反映了原有数据的空间几何特征.

2.2 输入参数的设定

CURD 算法有 3 个主要参数:距离 *Radius*、密度阈值 *t* 和循环次数 *Iterate*.参数的选择会影响算法的聚类结果和执行效率.

在实验中我们发现,如果距离 *Radius*,密度阈值 *t* 与 DBSCAN 算法的参数 *Eps*,Min*Pts* 取值相同,则两个算法会产生类似的结果.在 DBSCAN 算法中,参数 *Eps*,Min*Pts* 的取值通常难以确定,但是在 CURD 算法中,距离 *Radius* 和密度阈值 *t* 的选取可以近似得到.距离 *Radius* 需要考虑的是整个数据的空间,距离 *Radius* 越小,聚类的结果越好,但是参考点的数目也就越多,算法的执行效率就会变低.我们经过实验还发现,在 *Radius* 取整个数据

空间的 1/50~1/100 时就可以取得较好的效果,如:DS1 是 30×30 的二维空间,取距离为 0.5;DS2 和 DS3 是 100×100 的二维数据空间,取距离为 2.同时也发现,使用数据总量与候选参考点的数目比值,即候选参考点的平均密度作为密度阈值可以取得较好的效果.至于循环次数 *Iterate*,需要指出的是,我们所有的实验结果都是在 *Iterate*=4 的情况下得到的,在条件允许的情况下,*Iterate* 的值越大,聚类的结果就越好,同时所需时间也就越长.

2.3 执行效率的比较

我们选择基于 R*-树的 DBSCAN 算法与 CURD 算法进行算法的比较,有 3 个主要原因:(1) 两个算法都是基于密度的聚类算法;(2) 两个算法的输入参数有相似性;(3) 基于 R*-树的 DBSCAN 算法的时间复杂性为 $O(n\cdot\log n)$,因此,DBSCAN 算法是一种效率比较高的聚类算法.

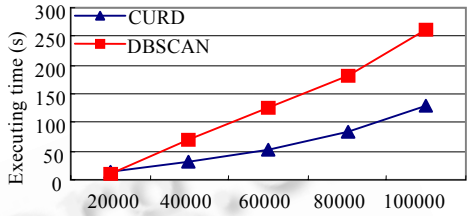


Fig.10 Performance comparison
图 10 性能比较

从图 10 中可以看出,CURD 算法的效率明显高于 DBSCAN 算法,因此 CURD 算法是一种效率非常高的聚类算法.此外,由于基于 R*-树的 DBSCAN 算法需要建立索引,而建立索引的时间代价是很大的;如果把建立索引的时间也考虑进去,则 CURD 算法将会大大优于 DBSCAN 算法.

3 讨 论

在本文中我们提出了一种使用参考点来描述数据空间几何特征的方法,并提出了一种基于参考点和密度的快速聚类算法 CURD,CURD 算法保持了基于密度的聚类算法可以发现任意形状的聚类和对噪音数据不敏感的优点;保持了 K-MEANS 算法的高效性,具有接近线性的时间复杂性,适合对大规模数据的挖掘.我们的理论分析和实验结果也证明了上述结论.

采用距离的方法可以将对高维数据的处理转换到一维空间[7,8],从这个意义上看,CURD 算法可以处理高维数据.我们下一步的工作是测试 CURD 算法在高维情况下的性能.此外,本文的算法找到的参考点可以准确地反映数据的几何空间特征,因此,我们正在考虑如何利用 CURD 算法对数据进行有效的抽样处理.

致谢 IBM 信息基础设施技术组陈捷博士后、宋国杰、遇辉等博士生对本文的完成给予了很大帮助,DBSCAN 算法的作者之一 Dr.Jörg Sander 先生提供了 DBSCAN 程序和相应的信息,在此一并表示感谢.

References:

[1] Han JW, Kambr M. Data Mining Concepts and Techniques. Beijing: Higher Education Press, 2001. 145~176.

[2] Kaufan L, Rousseeuw PJ. Finding Groups in Data: an Introduction to Cluster Analysis. New York: John Wiley & Sons, 1990.

[3] Ester M, Kriegel HP, Sander J, Xu X. A density based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han JW, Fayyad UM, eds. Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 226~231.

[4] Guha S, Rastogi R, Shim K. CURE: an efficient clustering algorithm for large databases. In: Haas LM, Tiwary A, eds. Proceedings of the ACM SIGMOD International Conference on Management of Data. Seattle: ACM Press, 1998. 73~84.

[5] Agrawal R, Gehrke J, Gunopolos D, Raghavan P. Automatic subspace clustering of high dimensional data for data mining application. In: Haas LM, Tiwary A, eds. Proceedings of the ACM SIGMOD International Conference on Management of Data. Seattle: ACM Press, 1998. 94~105.

[6] Alexandros N, Yannis T,Yannis M. C^2P: clustering based on closest pairs. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proceedings of the 27th International Conference on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 331~340.

[7] Berchtold S, Bohm C, Kriegel H-P. The pyramid-technique: towards breaking the curse of dimensionality. In: Haas LM, Tiwary A, eds. Proceedings of the ACM SIGMOD International Conference on Management of Data. Seattle: ACM Press, 1998. 142~153.

[8] Yu C, Ooi BC, Tan K-L, Jagadish HV. Indexing the distance: an efficient method to KNN processing. In: Apers PMG, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proceedings of the 27th International Conference on Very Large Data Bases. Roma: Morgan Kaufmann Publishers, 2001. 421~430.