文章编号:1001 - 2486(2006)05 - 0073 - 07

基于相对密度的增量式聚类算法*

刘青宝,侯东风,邓 苏,张维明
(国防科技大学 信息系统与管理学院,湖南 长沙 410073)

摘 要:基于聚类的相对性原则:簇内对象具有较高的相似度,而簇间对象则相反,提出一种基于相对密度的增量式聚类算法,它继承了基于绝对密度聚类算法的抗噪声能力强、能发现任意形状簇等优点[1],并有效解决了聚类结果对参数设置过于敏感、参数值难以确定以及高密度簇完全被相连的低密度簇所包含等问题。同时,通过定义新增对象的影响集和种子集能够有效支持增量式聚类。

关键词:增量式聚类;K 近邻;聚类参数;相对密度
中图分类号:TP391 **文献标识码**:A

Relative Density Based Incremental Clustering Algorithm

LIU Qing-bao, HOU Dong-feng, DENG Su, ZHANG Wei-ming
(College of Information System and Management, National University of Defense Technology, Changsha 410073, China)

Abstract: A new incremental clustering algorithm is proposed in this paper based on the relativity principle, which means that the similarities of objects in the same cluster is higher than those among different clusters. This approach not only inherits the advantages of absolute density based algorithms which can discover arbitrary shape clusters and are insensitive to noises[1], but also efficiently solves the following common problems: clustering results are very sensitive to the user-defined parameters, reasonable parameters are hard to be determined, and high density clusters are contained fully in coterminous low density clusters. With this approach, incremental clustering can also be supported effectively by defining the affected sets and seed sets of the updating objects in this approach.

Key words: incremental clustering; K-nearest neighbors; clustering parameter; relative density

聚类分析已经广泛地应用在许多领域,包括模式识别、数据分析、图像处理以及市场研究。目前文献中存在大量聚类分析算法,它们多数侧重于如何提高算法效率,而往往忽视了算法的有效性问题。聚类算法的有效性问题主要表现在三个方面:其一,聚类算法大多要求用户输入一定的参数,例如希望产生的簇的数目,而这些参数通常难以确定,特别是针对高维空间中稀疏分布的实际应用数据集,用户几乎无法给出合适的算法参数,因此非专业用户需要与数据分析专家密切配合才能保证获得理想的聚类结果,导致算法的使用极为不便;其二,聚类结果对于输入的参数值过于敏感[2],往往参数值的一些轻微变化却产生聚类结果的很大差异;其三,对于高维的实际应用数据集,其数据分布往往是稀疏的、杂乱的,很难为算法选择全局的参数进行准确的聚类分析,使得聚类的质量难以保证。本文提出的基于相对密度的增量式聚类算法,能适应增量式数据加载环境,并继承了基于密度的聚类算法抗噪声能力强、能发现任意形状的簇等优点。同时,有效地解决聚类结果对参数值过于敏感、参数值难以设置以及高密度簇完全被相连的低密度簇所包含等问题。

1 相关研究

第一个增量式聚类挖掘方法是文献[3]提出的增量式 DBSCAN 算法,是在 DBSCAN 算法[1]基础上,针对数据仓库环境中增量式数据加载要求而改造的。DBSCAN 算法依赖两个参数实现聚类:对象的邻域半径 r 和邻域内的最少对象数 MinPts。聚类结果对这两个参数的取值非常敏感,细微的设置偏差则

* 收稿日期:2006 - 05 - 08
基金项目:国家自然科学基金资助项目(60172012)
作者简介:刘青宝(1967—),男,副教授,博士生。

可能导致差别较大的聚类结果,而且参数的取值无参考区间,用户难以确定[2]。同时,由于DBSCAN算法采用全局一致的绝对密度作为聚类标准,导致了高密度簇完全被相连的低密度簇所包含[4]。如图1所示,在合适的参数设置时,可区分A、B、C三个类。

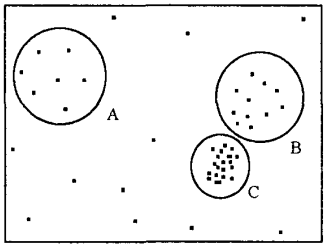

图1 参数值合适的聚类结果
Fig.1 Clusters when parameters are set properly

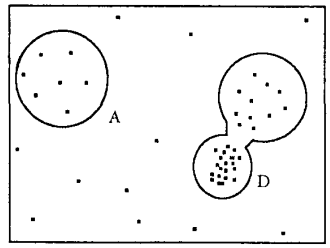

图2 绝对密度设置较低时的聚类结果
Fig.2 Clusters when parameter of density is set low

但当所选的MinPts较大,r较小,即绝对密度设置较高的时候,就会把A、B两个类全部判别成孤立点。反之,当绝对密度设置较低时,就会把B、C两个类合并为一个类D,如图2所示。

在数据仓库环境中,由于数据集的增量式加载,数据集固有的密度在不断改变,用户根本无法确定未来整个数据集的绝对密度参数,所以在DBSCAN算法基础上改造的增量式聚类算法很难有效使用。为此,本文从聚类的本质出发(即同一个簇中的对象之间具有较高的相似度,而不同簇中的对象差别较大)提出一种基于相对密度的增量式聚类算法RDBI Clustering(Relative Density Based Incremental Clustering),它能很好地发现任意形状、不同密度的类,并且能有效地解决聚类结果对参数值过于敏感、参数值难以设置以及高密度簇完全被相连的低密度簇所包含等问题。同时,由于相对密度的特点,RDBIClustering算法中对象的插入删除操作仅影响该对象的邻居所在的类。

2 有关概念

首先引入一些有关概念:

定义1 对象p的k近邻距离[5]。

对于任何正整数k和集合D,对象p的k近邻距离定义为p到它的第k个最近邻居o的距离;其中$p,o\in D$。对象p的k近邻距离用k-distance(p)表示。

定义2 对象p的k近邻邻居。

D为给定数据集,对象p的k近邻邻居,记为$N_k(p)$,定义为下面的集合:

$N_k(p)=\{q\in D\setminus\{p\}\mid d(p,q)<=\text{k-distance}(p)\}$;也称为$p$的$k$-近邻集合。

定义3 对象p的k近邻平均距离[6]。

给定集合D,$p\in D$,$o\in N_k(p)$;p的近邻平均距离(Near Neighbors Average Distance)记为$nnad_k(p)$,定义如下:

$$nnad_k(p)=\left(\frac{\sum_{o\in N_k(p)}d(p,o)}{|N_k(p)|}\right)$$

定义4 对象p关于k近邻的相对密度。

给定集合D,$p\in D$,p关于其k近邻$N_k(p)$的相对密度(Relative Density)记为$rd_k(p)$,定义如下:

$$rd_k(p)=\frac{\min_{q\in N_k(p)}\{nnad_k(q),nnad_k(p)\}}{\max_{q\in N_k(p)}\{nnad_k(q),nnad_k(p)\}}$$

当$rd_k(p)$接近1时,说明对象p与其邻居在分布密度上十分接近,能很好地融为一体。

定义5 核心对象。

给定阈值 $\delta>0$ 和集合 D,$p\in D$,若$[1-rd_k(p)]<\delta$,则称该对象 p 为核心对象。

定义 6 核心对象 p 的核心集合。

核心对象 p 的 k 近邻邻居中,由所有核心对象加 p 本身构成的子集称为核心对象 p 的核心集合,记为 $Core_k(p)$。若 p 非核心对象,则其核心集合无定义。

定义 7 核心对象 p 初始类。

设 D 是数据集,$p\in D$,且 p 为核心对象,p 的初始类 C:

$$C=\{q\in D\mid \exists\, o\in Core_k(p),\ 使得\ q\in N_k(o)\}$$

显然,核心对象 p 的初始类 C 非空,且有 $N_k(p)\subseteq C$。

定义 8 对象 p 关于类 C 的相对密度。

给定集合 D,$C\subseteq D$,$p\in D$,p 关于类 C 的相对密度(Relative Density)记为 $rd_C(p)$,定义如下:

$$rd_C(p)=\frac{\min\limits_{q\in C}\{nnad_k(q),nnad_k(p)\}}{\max\limits_{q\in C}\{nnad_k(q),nnad_k(p)\}}$$

定义 9 对象 p 的影响集。

设 D 是数据集,D 中受对象 p 的插入删除操作所影响的对象集合,称为对象 p 的影响集,记为 $Affected_D(p)$,有:

$$Affected_D(p)=N_k(p)\cup\{q\mid q\in\cup_{o\in N_k(p)}N_k(o)\}$$

定义 10 对象 p 的种子集。

由对象 p 的影响集中所有的核心对象组成的集合,称为对象 p 的种子集,记为 $Seed(p)$。

3 基于相对密度的聚类算法

基于相对密度的聚类算法首先在数据集 D 中找到任意一个核心对象 p,求出 p 的核心集合,得到初始类 C;然后由初始类 C 开始进行类的扩展,直至没有任何对象可以归入该类;重新在 D 中寻找任意一个未归类的核心对象 q,重复上述过程,直至没有任何对象可以归入任何类,算法结束。

由初始类 C 的扩展过程分两步进行:首先,对 p 的核心集合进行扩展,得到类 C 的扩展核心集合;然后,根据关于扩展核心集合中核心对象的密度可达这一条件,对类 C 进行扩展,详细扩展方法见过程 ExpandCluster 中的伪码描述。

基于相对密度的聚类算法 RDBClustering (Relative Density Based Clustering)伪码描述如下:

```
RDBClustering(Set Setofpoint, int k, real δ)
//Setofpoint 是数据集,k 为最少的最近邻数,
//δ 为大于零的阈值
BEGIN
  REPEAT
    point = GetCorePoint(Setofpoint, k, δ);
//在未标识对象中找一个核心对象,
//若无核心对象,则 point 为 NULL
    IF point < > NULL THEN
      Coreset = GetCoreSet(Setofpoint, point, k, δ);
//在未标识对象中得到 point 的核心集合
      clusterId = NewClusterId( );
//得到新类标识号
      C = GetInitCluster(Setofpoint, point, CoreSet, k, clusterID);
//从未标识对象中得到 point 的初始类 C,
//并标识为 clusterID
```

```
        ExpandCluster(Setofpoint, C,CoreSet, k, δ);
  //对类 C 进行扩展
      END IF
    UNTIL 没有任何类可以进行扩展;
  END RDBClustering.
```

其中,ExpandCluster 过程伪码具体描述如下:

```
  ExpandCluster (Set Setofpoint, Cluster C, Set CoreSet, int k, real δ)
  BEGIN
    WHILE NOT CoreSet.empty() DO
      point = GetOutPoint(CoreSet );
  //从 CoreSet 中取出一核心对象 point
      NewCoreset = GetCoreSet(Setofpoint, point, k, δ);
  //在未标识的对象中得到 point 的核心集合
      FOR i FROM 1 TO NewCoreset.size DO
        object = NewCoreset.get(i);
        IF (1 - rd_CoreSet(object)) < δ THEN
  //对密度渐变所产生累加效应进行阈值检查
        CoreSet = CoreSet∪{object};
  //object 加入类 C 的核心集合
      END IF;
    END FOR;
    C = C∪N_k(point);
  //把核心对象 point 的 k 近邻邻居扩展进簇 C
    END WHILE;
  END ExpandCluster.
```

过程 ExpandCluster 中条件"$(1\text{-}rd_{CoreSet}(object)) < \delta$"用以对密度渐变所产生累加效应进行阈值检查,使得在数据密度连续渐变情况下,也能区分不同密度等级的类。

4 基于相对密度的增量式聚类算法

增量式聚类算法要求利用前次的结果来加速本次聚类过程,而不是简单地将算法重新作用于整个数据集,从而提高数据聚类过程的效率。数据对象集的更新分为对象插入和删除操作,限于篇幅这里只讨论对象插入操作的处理。

对象 p 插入操作,处理分为五种情况,如图 3 所示:

(1)p 是孤立点:p 不属于任何一个核心对象的 k 近邻;

(2)创建新类:p 和其它新插入的对象以及原孤立点形成新类,如图 3(a)所示;

(3)吸收:若 p 只被一个类所标识,如图 3(b)所示;

(4)合并:由于对象 p 以及其他新对象的插入造成了两个或多个类的合并,如图 3(c)所示;

(5)拆分:由于对象 p 以及其他新对象的插入,一个类的不同部分的相对密度超过阈值,导致该类裂分为两个或多个类,如图 3(d)所示。

已知原数据集 D,存在类 $C1, C2, \cdots, Cn$,孤立点集合 O,新增数据集 D'。首先对 D' 中的对象 p 求其在数据集 $D \cup D'$ 中的影响集 $Affected_{D\cup D'}(p)$,从影响集 $Affected_{D\cup D'}(p)$ 出发进行增量式聚类。

基于相对密度的增量式聚类算法 RDBIClustering (Relative Density Based Incremental Clustering)伪码如下:

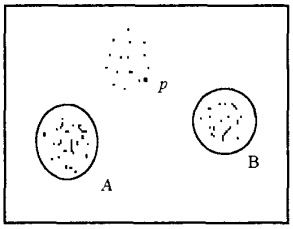

图 3(a) 形成新类
Fig.3(a) Forming a new cluster

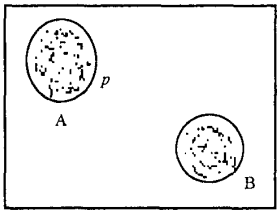

图 3(b) 被已存在类 A 吸收
Fig.3(b) Sorbed by the former cluster A

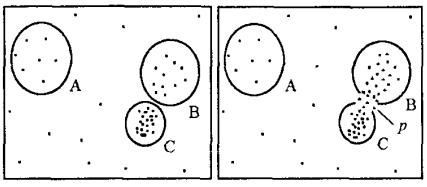

图 3(c) 使已存在类 B、C 合并
Fig.3(c) Bring the merging of two former clusters B and C

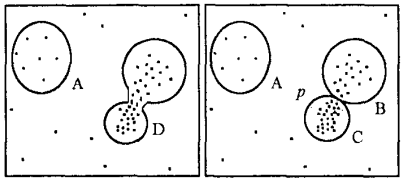

图 3(d) 使已存在类 D 分裂为类 B、C
Fig.3(d) Making the former cluster D split into cluster B and cluster C

```
RDBIClustering(ClusterSet CluSet,PointSet D, PointSet D', int k, real δ)
//原聚类结果集 CluSet = {Ci},原数据集 D,
//新到数据集 D'
BEGIN
  FOR i FROM 1 TO D'.size DO
    p = GetOutPoint(D');
    IF p 已标识 THEN NEXT
//若 p 已标识,则进入下一次循环
    Affected_{D∪D'}(p) = GetAffectSet(p);
    IncreCluster(D∪D', Affected_{D∪D'}(p), CluSet, k, δ);
//从影响集 Affected_{D∪D'}(p)出发增量式聚类
  END FOR;
END RDBIClustering.
```

其中,IncreCluster 过程伪码具体描述如下:

```
IncreCluster(Set Setofpoint, Set AffectedofP, ClusterSet CluSet, int k, real δ)
BEGIN
  SeedSet = GetSeedSet(AffectedofP, k, δ);
//从影响集中得到种子集
  IF SeedSet = = NULL THEN RETURN
//这时插入对象 p 为孤立点,IncreCluster 结束;
  FOR i FROM 1 TO SeedSet.size DO
    point = GetOutPoint(SeedSet);
    cluId = GetClusterId(point);
//得到 point 的类标识,若 point 未标识,cluId 为 0
  C = GetSet(CluSet,cluId);
  IF C < > NULL THEN
```

```
    IF (1 - rd_C(point)) < δ THEN// point 仍属于类 C
      C = C∪N_k(point);
      NEXT;
    END IF;
  END IF;
  cluId = NewClusterId( );
  IExpClu(Setofpoint, point, cluId, CluSet, k, δ);
//不受原 CluSet 的限制,从 point 出发进行扩展,
//并对最后形成的扩展集赋以类标识 cluId
  END FOR;
END IncreCluster.
```

5 算法对比分析

这里从处理时间、参数选择和聚类质量三个指标分析基于相对密度的聚类算法的性能。

5.1 处理时间

算法 RDBClustering 运行过程中需多次使用对象 p 的 k 近邻平均距离 nnad,实现时采用中间数据表 M 对其进行保存,做到一次计算多次查询。整个算法的时间主要由 k 近邻查询的时间和对中间数据表 M 的扫描时间两部分组成。k 近邻查询的时间复杂度为 $O(\log_2 n)$[8],对中间数据表 M 的扫描时间复杂度为 $O(n)$,总体时间复杂度为 $O(n\ \log_2 n)$,与 DBSCAN 算法的时间复杂度 $O(n\ \log_2 n)$[2,8] 相比属于同阶的,没有明显的时间差异。

算法 RDBIClustering 的时间复杂度分析比较困难,它与新增对象集和种子集的分布情况以及原有聚类结果密切相关。设新增对象数为 m,种子集的平均对象数为 a,最坏情况是每个新增对象的所有 a 个种子都"未标识",这时,需进行 ma 次 k 近邻查询,总体时间复杂度为 $O(ma\log_2 n)$。由此可见,当 $m \ll n$, $a \ll n$ 时,算法 RDBIClustering 的效率比算法 RDBClustering 将大大提高。

5.2 参数选择

定理 设 C 是一个对象集,用 dist-min 表示 C 中对象的最小 k 近邻距离,用 dist-max 表示 C 中对象的最大 k 近邻距离。令 ϵ = dist-min/dist-max,对于 C 中的所有对象 p,若 $N_k(p) \subseteq C$,而且 $\forall q \in N_k(p)$,也都有 $N_k(q) \subseteq C$,则 $\epsilon \leqslant rd_k(p) < 1$。

定理的直观解释如下:C 对应于一个类,考虑在类内深处对象 p,即 p 任一 k 近邻邻居 q 都在 C 中,并且,q 所有的 k 近邻邻居也在 C 中,对这样的类 C 深处对象 p,p 的 $rd_k(p)$ 值是有一定范围的。该定理为算法参数 δ 的选择限定了取值范围,从而不像 DBSCAN 算法参数的取值具有太多的盲目性和试探性。

5.3 聚类质量

基于相对密度的聚类算法与算法 DBSCAN 一样可以在有噪声的数据集中发现任意形状的簇,并且有效地解决了算法 DBSCAN 所存在的问题:高密度的聚类结果被完全包含在相连的低密度的聚类结果中。对于如图 4 所示数据集,算法 RDBClustering 和 RDBIClustering 可以聚类出如图 5 所示结果,如实地反映了数据集的数据分布情况。

6 小 结

基于相对密度的增量式聚类算法 RDBIClustering 有效地解决了参数的设置和算法对于参数过于敏感的问题,同时成功地解决了传统的基于密度聚类算法所存在的高密度聚类结果被包含在相连的低密度聚类结果中的问题,能区分不同密度等级的簇。而且在新增数据集相对已有数据集规模较小时,算法效率高。

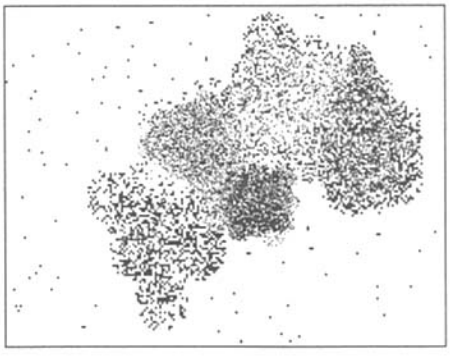

图 4 实验数据集
Fig.4 The test data set

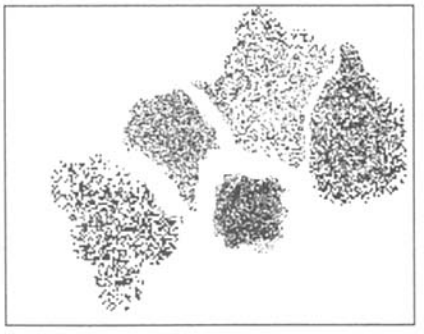

图 5 算法 RDBClustering 的聚类结果
Fig.5 The clustering result of RDBClustering

算法的不足之处:必须用簇中的所有点来表示聚类形成的任意形状,这在内存有限情况下对动态数据集进行增量式聚类是难以适用的。如何在有限内存情况下对动态数据集进行增量式聚类是进一步研究工作的方向。

参 考 文 献:

[1] Ester M, Kriegel H P, Sander J, et al. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise[A]. In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining[C], Portland, OR, 1996:226 - 231.

[2] Han J W, Kamber M, Fan M, et al. Data Ming: Conception and Technology[M]. Beijin: Machine Press, 2001.

[3] Ester M, Kriegel H P, Sander J, et al. Incremental Clustering for Mining in a Data Warehousing Environment[A]. In: Gupta A, Shmueli O, Widom J, eds., the 24th International Conference on Very Large Data Bases[C], New York, Morgan Kaufmann Publishers Inc., 1998:323 - 333.

[4] Ankerst M, Breunig M, Kriegel H P, et al. OPTICS: Ordering Points To Identify the Clustering Structure[A]. In: Proc. ACM SIGMOD'99, Int. Conf. on Management of Data[C], Philadelphia, PA, 1999.

[5] Breunig M, Kriegel H P, Ng R T, et al. LOF: Identifying Density-based Local Outliers[A]. In: Proc. ACM SIGMOD 2000 Int. Conf. On Management of Data[C], Dalles, TX, 2000.

[6] Liu Q B, Deng Su, Lu C H, et al. Relative Density Based K-nearest Neighbors Clustering Algorithm[A]. In: Proc. 2003 Int. Conf. on Machine Learning and Cybernetics[C], Xi'an, China, 2003, 133 - 137.

[7] Tang J, Chen Z X, et al. A Robust Outlier Detection Scheme for Large Data Sets[EB/OL]. In: http://www.cs.panam.edu/chen/papers.html.

[8] 邵峰晶,于忠清. 数据挖掘:原理与算法[M]. 北京:中国水利水电出版社, 2004.