软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(5):1113−1128 [doi: 10.13328/j.cnki.jos.004717]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向海量数据流的基于密度的簇结构挖掘算法*

于彦伟[1], 王 欢[2], 王 沁[3], 赵金东[1]

[1](烟台大学 计算机与控制工程学院,山东 烟台 264005)

[2](Department of Computer Science, University of California, San Diego, USA)

[3](北京科技大学 计算机与通信工程学院,北京 100083)

通讯作者: 赵金东, E-mail: zjd@ytu.edu.cn, http://www.ytu.edu.cn

摘 要: 提出一种基于密度的簇结构挖掘算法(mining density-based clustering structure over data streams,简称MCluStream),以解决数据流密度聚类中输入参数选择困难和重叠簇识别等问题.首先,设计了一种树拓扑 CR-Tree 索引结构,将直接核心可达的一对数据点映射成树结构中的父子关系,蕴含了数据点依赖关系的 CR-Tree 涵盖了一系列 *subEps* 参数下的基于密度的簇结构;其次,MCluStream 算法采用滑动窗口的方式更新 CR-Tree,在线维护当前窗口上的簇结构,实现了对海量数据流的快速演化聚类分析;再次,设计了一种快速从 CR-Tree 提取簇结构的方法,根据可视化的簇结构,选择合理的聚类结果;最后,在真实和合成海量数据上的实验验证了 MCluStream 算法具有有效的挖掘效果、较高的聚类效率和较小的空间开销.MCluStream 可适用于海量数据流应用中自适应的密度聚类演化分析.

关键词: 聚类分析;密度聚类;簇结构;数据流;滑动窗口

中图法分类号: TP311

中文引用格式:于彦伟,王欢,王沁,赵金东.面向海量数据流的基于密度的簇结构挖掘算法.软件学报,2015,26(5):1113−1128. http://www.jos.org.cn/1000-9825/4717.htm

英文引用格式: Yu YW, Wang H, Wang Q, Zhao JD. Density-Based cluster structure mining algorithm for high-volume data streams. Ruan Jian Xue Bao/Journal of Software, 2015,26(5):1113−1128 (in Chinese). http://www.jos.org.cn/1000-9825/4717.htm

Density-Based Cluster Structure Mining Algorithm for High-Volume Data Streams

YU Yan-Wei[1], WANG Huan[2], WANG Qin[3], ZHAO Jin-Dong[1]

[1](School of Computer and Control Engineering, Yantai University, Yantai 264005, China)

[2](Department of Computer Science, University of California, San Diego, USA)

[3](School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract: This paper proposes a mining algorithm of density-based cluster-structure, named MCluStream, to resolve the problems of input parameter selection and overlapping cluster identification in evolving data stream. First, a tree topology index, named CR-Tree, is designed to map a pair of data points with directly core reachable into relationship of father and child node. The CR-Tree that record relationships among points represents cluster-structure under a series of *subEps* settings. Second, the online update of cluster-structure on CR-Tree is completed by MCluStream under sliding window environments, which effectively maintains clusters over massive evolving data streams. Third, a fast cluster-structure extraction method is implemented from the CR-Tree. Users can easily select reasonable clustering results according to the visualized cluster-structure. Finally, experimental evaluations on massive-scale real and synthetic data demonstrate the effective mining result and better performance of the proposed algorithm compared against state-of-the-art methods. MCluStream is desirable to be applied to self-adaptive density-based clustering over high-volume data streams.

* 基金项目: 国家自然科学基金(61403328, 61302065, 61172049); 山东省自然科学基金(ZR2013FM011); 山东省高等学校科技计划(J14LN24); 吉林大学符号计算与知识工程教育部重点实验室开放基金(93K172014K13)

收稿时间: 2014-05-16; 修改时间: 2014-08-15; 定稿时间: 2014-09-12

Key words: cluster analysis; density-based clustering; cluster structure; data stream; sliding window

数据流聚类分析已经广泛应用于网络数据监测、金融数据分析、视频监控、移动计算、实时交通管理等大量实时应用[1].通过对数据流实时聚类,可发现应用中有效的目标分类,快速提取海量数据流蕴含的有用信息,以支持实时决策或实时查询处理.与传统聚类算法相比,数据流聚类算法具有以下特点:第一,在有限的内存及存储空间中聚类无限的流式数据;第二,由于I/O代价昂贵,对海量流数据只能进行线性扫描,因此,数据流聚类算法应尽量对数据进行一遍扫描,且可以增量式处理;第三,算法应该可以对记录进行实时处理.由于数据流中数据变化速度较快,对算法的响应速度要求很高,因而数据流聚类算法应该能够实现高效的在线处理.然而对于基于密度的数据流聚类,还存在一个重要的挑战——算法对输入参数的过度依赖问题.

目前,大多数据流聚类算法,如CluStream[2]以及基于微聚类架构的ACluStream[3],DenStream[4]和DStream[5]都需要输入密度参数.密度参数的选择一般依赖于处理数据的特征,但由于无法提前获取数据流中的全部数据特征,因此密度数据流聚类算法输入参数的选择非常困难.Lee等人[6]提出了一种使用熵理论的方法确定最优输入参数,但该方法仍然需要根据数据特征确定输入参数.由于数据流的无限性、变化性和不确定性,在应用场景无法预知的前提下,该方法并不适用于流数据聚类方法的密度参数选择.此外,使用全局统一输入参数的数据流聚类算法还有另外一个缺陷:无法对"密度重叠簇"进行识别,即,无法识别密度不同的簇结构.

Kranen等人[7]提出了一种根据数据流的更新速度自适应地调整聚类速度的方法,从而充分利用时间来完善聚类效果.之后,他们又设计了一种压缩自适应索引结构CluTree[8],以在线维护数据流中聚类簇概要信息,用于提高聚类效率和效果.Xu等人[9]提出了一种自适应的进化聚类框架,该框架采用一种收缩的方式为静态聚类算法(包括层次聚类、*K*-Means和光谱聚类)获得最优的输入参数,并将这些静态聚类算法扩展成进化聚类算法.类似对静态算法输入参数自适应选择的研究还有很多,如对DBSCAN[10]参数自适应优化方法[11,12].Yang等人[13,14]在滑动窗口环境下针对多查询处理,提出了一种基于预测视角成长理论和层次聚类结构的多聚类查询共享执行策略Chandi,虽然能够解决多个不同输入参数下的并发聚类查询,但仍然不能解决输入参数的自适应选择问题.为了解决DBSCAN的参数选择及全局参数问题,Ankerst等人[15]提出一种基于排序数据点的簇结构识别方法OPTICS,该算法并不显式地产生一个聚类簇结果,而是通过螺旋式方式计算数据点的可达距离并排序产生一个数据点序列,这个序列代表了所有数据在一系列输入参数下基于密度的聚类簇结构.OPTICS只针对静态数据库聚类,因此仍然不能解决数据流环境下密度聚类的参数选择问题.

表1总结与分析了几种典型的密度数据流聚类算法对输入参数的要求及提出的选择方法.

Table 1 Input parameter analysis of density-based data stream clustering algorithms

表1 基于密度的数据流聚类算法输入参数分析

Algorithm	Window models	Number of parameter	Selected method
CluTree[8]	Damped	3	No
Chandi[14]	Sliding window	4	No
OLDStream[16]	Landmark	2	Sample and entropy theory
D-Stream[5]	Damped	5	No
Den-Stream[4]	Damped	4	DBSCAN and real application knowledge
ACluStream[3]	Landmark	3	STICKY count

本文针对密度数据流聚类算法输入参数选择困难和过度依赖的问题,提出一种基于滑动窗口的密度簇结构挖掘算法MCluStream(mining density-based clustering structure over data streams),以自适应地发现海量进化数据流中任意密度的聚类结果.首先,我们采用可达距离和核心可达等概念,设计了一种树拓扑CR-Tree(core reachable tree)索引目标数据点,将直接核心可达的一对数据点表示成树结构中的父子关系,蕴含数据点间可达距离的CR-Tree涵盖了一系列*subEps*参数下的簇结构;然后,分析了新数据点和过期数据点对本地簇结构的影响,采用滑动窗口的方式实现了对海量数据流中当前窗口的CR-Tree的在线更新;进而提出了一种从CR-Tree中快速提取簇结构的方法,可通过简单转化提取出当前窗口的簇结构;最后,在实际和合成数据集上进行实验验证,综合评估了MCluStream的演化聚类效果、聚类效率、空间开销和可伸缩性,并且分析了算法的参数敏感性.

1 问题定义

1.1 相关概念

本节在 DBSCAN[10]和 OPTICS[15]的基础上进一步对相关术语给出了定义.以下定义涉及到两个参数:数据点邻域范围半径 *Eps* 和最小邻居点数目阈值 *MinPts*.设定当前数据点集合为 D, $N_{Eps}^{D}(p)$ 表示数据点 p 在 *Eps* 邻域范围内的邻居点集合,其数量标记为 $|N_{Eps}^{D}(p)|$.

定义 1(核心距离). 对于 $p\in D$,设定 *MinPts-dist*(p)为数据点 p 到它的第 *MinPts* 个邻居的距离,数据点 p 的核心距离 *core-dist*(p)定义如公式(1):

$$core\text{-}dist(p)=\begin{cases}MinPts\text{-}dist(p), & |N_{Eps}^{D}(p)|\geqslant MinPts\\ Undefined, & \text{otherwise}\end{cases} \tag{1}$$

由定义可知:核心距离只对核心点有效,它表示该数据点满足核心点条件时最小的邻域半径阈值,即,该点到其第 *MinPts* 个邻居的距离.非核心点不存在核心距离,定义为 *Undefined*.如图 1 所示,若 *MinPts*=3,数据点 o 的核心距离为点 o 到点 p_2 的距离,这是因为点 p_2 是点 o 的第 3 个邻居点;数据点 p_3 的核心距离为点 p_3 到点 o 的距离.

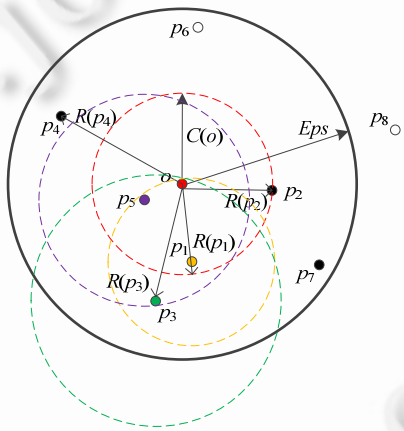


Fig.1 An example of data points distribution

图 1 数据点分布示例

定义 2(可达距离). 对 $p\in D, \forall o\in N_{Eps}^{D}(p)$,数据点 p 相对于点 o 的可达距离 *reach-dist*(p,o)的定义如公式(2):

$$reach\text{-}dist(p,o)=\begin{cases}Undefined, & |N_{Eps}^{D}(o)|<MinPts\\ \max(dist(o,p),core\text{-}dist(o)), & \text{otherwise}\end{cases} \tag{2}$$

直观来看,点 p 相对于点 o 的可达距离只有当点 o 是核心点时才有效,否则为 *Undefined*.并且点 p 的可达距离依赖于它所对应的数据点,也就是说,同一个数据点相对于不同的核心点,其可达距离可不相同.可达距离表示为保证点 p 直接密度可达到点 o 的最小 *Eps* 阈值,若 *Eps* 小于该值,则点 p 不可能直接密度可达到点 o.如在图 1 中,*MinPts*=3,点 p_1 相对于点 o 的可达距离为点 o 的核心距离,这是因为点 p_1 到点 o 的距离小于点 o 的核心距离,若 *Eps* 小于该值,则不能保证点 o 是核心点;点 p_3 相对于点 o 的可达距离为点 p_3 到点 o 的距离,这是因为它们间的距离大于点 o 的核心距离,若 *Eps* 小于该值,则不能保证点 p_3 在点 o 的邻域范围内.

定义 3(直接核心可达). 针对数据点 $p,o\in D$,如果对于 $\forall o'\in N_{Eps}^{D}(p)$ 满足如下条件:

$$reach\text{-}dist(p,o)\leqslant reach\text{-}dist(p,o'),$$

则称点 p 直接核心可达于数据点 o.可达距离 *reach-dist*(p,o')称为点 p 的核心可达距离.

对于直接密度可达的一对核心点,其关系是对称的.但是对于直接核心可达的一对核心点,其直接核心可达关系则是非对称的,这是由于点 p 相对于点 o 的可达距离最小,但点 o 相对于点 p 的可达距离并非是最小的.如在图 1 中,点 p_1 直接核心可达于点 o,它的核心可达距离为点 o 的核心距离.这是因为点 p_1 相对于点 o 的可达距离最小.直接核心可达的概念给出了在最小 *Eps* 参数下的密度可达关系.

定义 4(核心可达). 如果数据点 p 核心可达于数据点 q,当且仅当存在一系列数据点 $p_1,p_2,\ldots,p_n$ 满足 p_i 直接核心可达 $p_{i+1}(i=1,2,\ldots,n-1)$,并且 $p=p_1,q=p_n$.

由定义可知,核心可达关系是传递的直接核心可达关系的延伸.因此,核心可达关系是可传递的.

1.2 密度簇结构

OPTICS 引入了一种密度簇结构的概念,如图 1 中数据集合示例,根据 OPTICS,随机从数据点 o 开始处理,设定 o 的可达距离为 *Undefined*,然后计算它的核心距离 $C(o)$.设定 *Eps* 如图中所示,*MinPts*=3.点 o 的核心距离 $C(o)$为点 o 到点 p_2 的距离.然后,根据可达距离定义,相对于核心点 o,p_1 和 p_2 的可达距离都为 $C(o)$,分别记为 $R(p_1)$,$R(p_2)$;而 p_3 和 p_4 的可达距离则为它们到点 o 的实际距离,因此,按照可达距离递增排序,*temp* 列表顺序为 $p_1<p_2<p_3<p_4$.同时,因为点 o 已经处理完毕,所以将它加入到排序点列表 L 中.然后,按照 *temp* 列表中的顺序依次执行以上的操作,直到将所有处理过的数据加入到 L 中.最后可得出排序点列表 $L=(o,p_1,p_3,p_5,p_2,p_7,p_4,p_6,p_8)$,如图 2 所示,纵坐标表示数据点的可达距离,其中,$p_i(p_j)$表示点 p_i 选择的可达距离为相对于点 p_j 的可达距离.根据该排序点列表,我们可得出一系列的聚类结构.如设定 *subEps* 如实线线条纵坐标值,则在线条以下的连续的数据点构成一个聚类簇,而在线条以上的数据则被认定为噪声.若 *subEps* 的值设定为虚线线条纵坐标,则该示例的所有数据点都被划为同一聚类簇.

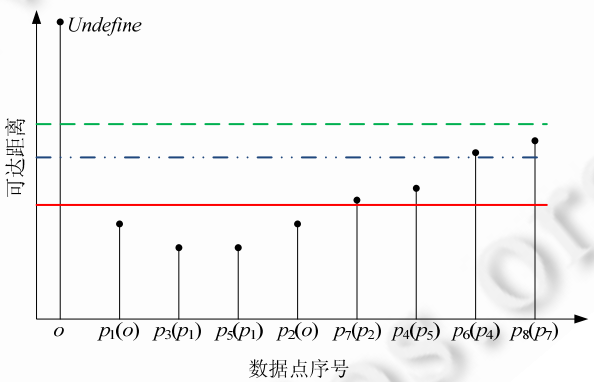


Fig.2 Order list of reach distance

图 2 可达距离排序列表

基于密度的簇结构有效地表示了数据集中一系列 *subEps* 参数下的聚类结构,并且可将复杂的数据分布以线性列表形式可视化,便于用户选择合适的聚类结果.

但是,数据流是一个局部可见、快速更新的无限数据集合,各数据点的核心距离、可达距离等属性随数据流不断变化,而呈现出的簇结构也在不断演化.因此,本文着重于研究如何从海量数据流中提取基于密度的簇结构,以解决数据流密度聚类算法对输入参数的过度依赖和敏感性问题.

2 基于滑动窗口的密度簇结构挖掘框架

本节将详细介绍一种滑动窗口环境下的密度簇结构挖掘算法 MCluStream.首先,提出了一种映射数据点间核心可达关系的树存储结构 CR-Tree;然后,分析了数据流中新数据和过期数据对簇结构的影响;最后,实现了一种高效的基于滑动窗口的数据流密度簇结构挖掘算法.

2.1 CR-Tree结构

MCluStream 维护了一个核心可达树 CR-Tree 索引结构,其树节点关系定义如下:

父亲节点. 如果数据点 p 是另一数据点 q 的父亲节点,当且仅当满足:q 直接核心可达于点 p,并且点 q 相对于点 p 获取它的可达距离.

祖先节点. 如果数据点 p 是另一数据点 q 的祖先节点,当且仅当满足:q 核心可达于点 p.

由上述定义可知:CR-Tree 中的父子关系是由直接核心可达概念产生;祖先与子孙节点是由核心可达概念产生.

图 3 展示了一个数据集合 D,包含 15 个数据点:$p_1,p_2,\ldots,p_{11},o_1,\ldots,o_4$,其中,$o_1,o_2,o_3$ 和 o_4 密度稍高于周围数据点.设定 *MinPts*=2,并且 *Eps* 足够大,可以满足将 D 中所有的数据点都包含在任意数据点的 *Eps* 邻域内.根据直接核心可达和父亲节点的定义,如果点 m 直接核心可达点 n,那么设定一个方向箭头,由 m 指向 n,其中,n 为父亲节点.依次类推,我们可得到一个有向拓扑结构,如图 4 所示.该有向拓扑描述了数据集 D 中数据点间的直接核心可达的密度关系.被箭头连接起来的数据点相对于 *Eps* 都相互密度相连.对于没有包含任何箭头链中数据点,它在 *Eps* 和 *MinPts* 条件下是一个噪声点.

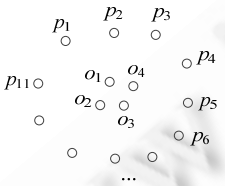


Fig.3 A sample of a dataset D

图 3 数据点集合 D 示例

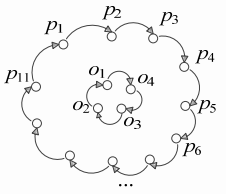


Fig.4 Digraph with cycle topology structure of dataset D

图 4 数据集 D 的有向环拓扑结构

在利用直接核心可达关系得到的有向拓扑结构中,可能存在孤立的循环链结构,如图 4 所示,$p_1,p_2,\ldots,p_{11}$ 建立了一个闭环,而 $o_1,\ldots,o_4$ 同样构成一个闭环.这是由于每个数据点都直接指向其直接核心可达的数据点,当数据集中有较高聚集密度的数据点或者存在密度差异较大的多个簇时,高密度聚集的数据点被链接成独立的循环链结构.但是,这种独立的循环链结构隔离了高密度数据集合与其他数据间的密度关系,进而被处理为独立的聚类簇,切断了在松弛参数条件下的聚类结构.为了避免这种现象造成的影响,应尽量避免产生循环链结构,以保证算法维护具有更宽参数条件下的簇结构.

本文首先设计了一种 CR-Tree 构建算法,以避免循环链结构的产生.其主要思想是:算法维护一个栈 S,任意从 D 中取出某个未被处理过的数据点 p.首先,找到点 p 直接核心可达的数据点 o(点 o 不在栈内,若已被压栈,则取比点 o 次小的相对核心点代替点 o),如果点 o 不在 CR-Tree 上,则将点 p 压到栈 S,先处理点 o,等待点 o 加入 CR-Tree 后,将点 o 更新为点 p 的父亲节点(如算法 1 第 7 行~第 11 行描述).然后,取栈顶元素进行处理,若栈 S 为空,则从集合 D 中取出下一个未被处理的数据点进行处理.其伪代码如算法 1 所示,其中,$p\propto 0$ 表示点 p 直接核心可达于点 o.在没有索引的情况下,所有数据点计算核心距离的时间复杂度为 $O(n^2)$,每个数据点计算直接核心距离的时间复杂度为 $O(n)$.总的来说,算法 1 的时间复杂度为 $O(n^2)$.

算法1. BulidCRTree (bulid CR-tree for D).

```
1. foreach p ∈ D
2.     if p.pro == false
3.         Process(p);
 Process(p)
4. find o (o ∉ S) that p ∝ o;
5. if (o == null)
6.     CR-Tree.root ← p; p.pro = true; return;
```

7. if ($o \notin CR\text{-}Tree$)
8. $S.push(p)$;
9. $Process(o)$;
10. $S.pop()$;
11. $p.father \leftarrow o$; $p.pro$ = true;

按照 CR-Tree 构建策略,可将数据集 D 转换成树状拓扑结构,如图 5 中右图所示.如何对数据流环境下的 CR-Tree 中的数据点的核心可达距离进行更新,将在下节详细描述.

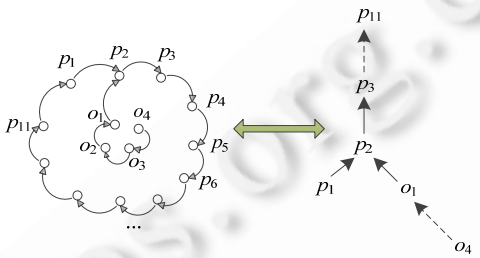


Fig.5 CR-Tree structure corresponding to the dataset D

图 5 数据集 D 对应的 CR-Tree 结构

2.2 数据流密度簇结构挖掘算法

本文采用 CQL 使用的周期滑动窗口概念[17]来定义无限海量数据流序列中的子数据流.给定一个数据流 S,任何一个子序列 $W=\{p_{i-w+1}p_{i-w+2}\ldots p_i\}$被称作一个滑动窗口 W,其中,w 为窗口长度.每个窗口包括一个开始时间戳 $W.T_{start}$和一个结束时间戳 $W.T_{end}=W.T_{start}+w-1$.结束时间 $W.T_{end}$等于当前时间点的窗口称为当前窗口.对于当前窗口 W,p_i 为新到达数据点,p_{i-w} 为过期数据点.

2.2.1 新数据点的影响及处理方法

当新数据点 p 到达后,对本地的核心可达关系产生了影响,因此仅可能影响到本地区域的簇结构,对远离其 Eps 邻域外的聚类簇不会产生影响.

根据 Eps 领域的定义,一个新数据点能够直接影响 Eps 半径为其内的数据点的属性.但由于簇结构引入了核心距离和可达距离的概念,被影响的 Eps 邻居将可能传递地影响它们的 Eps 邻居.因而数据点的属性变化有两种:(1) 核心距离发生变化;(2) 可达距离发生变化.而被新数据点 p 影响的数据可分为以下 3 类:

(a) 新增的核心点:新增加的核心点集合标记为

$NewC(p).NewC(p)=\{q \mid NewC(p).NewC(p)=\{q \parallel N_{Eps}^{D-p}(q)|<MinPts$并且$|N_{Eps}^{D}(q)|\geqslant MinPts\}$ 并且$\}$.

若新点 p 是核心点,则 $p \in NewC(p)$;

(b) 核心距离减小的核心点:点 p 邻域内核心距离发生变化的 Eps-邻居点集合,标记为 $ReduceCDist(p)$:

$$ReduceCDist(p)=\{q \parallel N_{Eps}^{D-p}(q)|\geqslant MinPts, q \in N_{Eps}^{D}(p), dist(q,p)<core\text{-}dist_{old}(q)\}.$$

核心距离减小的核心点一定是新点 p 的 Eps 邻域点,且和新点 p 的距离小于之前的核心距离;

(c) 核心可达距离减小的点:该数据点集合标记为 $ReduceRDist(p)$,则

$$ReduceRDist(p)=\{q \mid q \in N_{Eps}^{D}(ReduceCDist(p) \cup NewC(p))且reach\text{-}dist_{new}(q)<reach\text{-}dist_{old}(q)\}.$$

$ReduceRDist(p)$的核心可达距离的减小是由于一些数据点的核心距离减小而被影响的.因此,(a)类和(b)类数据点都属于核心距离属性变化的数据点,(c)类数据属于可达距离属性发生变化的数据点.

图 6 为二维空间下新数据点对周围数据影响的实例,设定点 p 是新加入的点.实线圆圈表示 Eps 邻域,虚线圆圈表示核心点的核心距离邻域.假设 $MinPts$=4.由于点 p 的加入,q_1 由边界点变成了核心点;o_1 的核心距离减小了,其 Eps-邻居的可达距离也将可能发生变化.由于点 p 也是核心点,q_2 的核心可达距离减小了(由相对于 o_1 到相对于 p);同时,q_1 的核心可达距离也发生了变化(由相对于 o_2 到相对于 p).

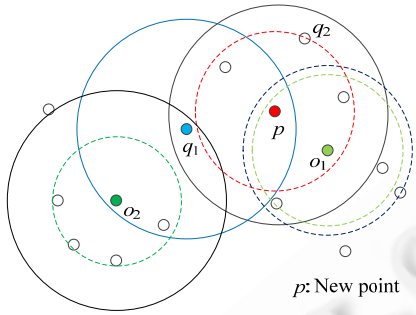


Fig.6 Effect of new point on neighboring points

图 6 新数据点对周围数据的影响

由于 CR-Tree 中数据点间的关系依赖于数据点间的核心可达关系,所以,(a)类和(b)类两类数据点并不影响它们在 CR-Tree 中的结构.但是前两类数据点会将其影响传递给其 *Eps* 邻域范围内的数据点(即,(c)类被影响的数据点).而对于核心可达距离减小的数据点,其在 CR-Tree 的结构的位置将可能发生改变.

由于 *NewC*(*p*)和 *ReduceCDist*(*p*)中的数据点会对其 *Eps* 邻居造成核心可达距离减小的影响,因此,我们在这两类集合的基础上对核心可达距离减小的数据点做进一步扩展处理.对新点 *p* 的扩展处理方法分为两步:

步骤 1. 搜索 *NewC*(*p*)和 *ReduceCDist*(*p*)两个集合.

根据新数点 *p*,找出其邻域内的新增核心点集合 *NewC*(*p*)和核心距离减小的数据集合 *ReduceCDist*(*p*).

步骤 2. 根据步骤 1 搜索结果,扩展处理 *ReduceRDist*(*p*).

由步骤 1 的结果,可确定核心可达距离减小的数据点集合 *ReduceRDist*(*p*)的范围,即,*NewC*(*p*)和 *ReduceCDist*(*p*)的 *Eps* 邻域范围.对于核心可达距离减小的数据点 *q*,若其父亲节点没有发生变化,则 *q* 在 CR-Tree 中的位置不发生变化;否则,将点 *q* 指向新的父亲节点.仅通过对这些数据点的父亲节点的修改,级联地修正了新点 *p* 对整个窗口内数据的簇结构的影响.

2.2.2 过期数据的影响及处理方法

由于过期数据点将从滑动窗口中删除,因此,过期数据的删除同样会对本地的簇结构产生影响.同新数据点对 *Eps* 邻域的影响一样,被影响的数据点的属性变化也是两种:核心距离发生变化和可达距离发生变化.过期数据点 *p* 对本地数据点的影响也可分为 3 类:

(a) 退化的核心点:由于点 *p* 的删除,导致原来的一些核心点退化为边界点或噪声的数据点,该类数据集合标记为 *ExpiredC*(*p*):

$$ExpiredC(p)=\{q \mid N_{Eps}^{D}(q)\geqslant \mathrm{Min}Pts\text{并且}\mid N_{Eps}^{D-p}(q)\mid<\mathrm{Min}Pts\}.$$

如果过期数据点 *p* 在过期之前是核心点,则 $p\in ExpiredC(p)$.

(b) 核心距离增大的核心点:新点 *p* 邻域内核心距离增大的数据点,该类数据集合标记为 *IncreCDist*(*p*):

$$IncreCDist(p)=\{q \mid q\in N_{Eps}^{D}(p),\mid N_{Eps}^{D-p}(q)\mid\geqslant \mathrm{Min}Pts, dist(q,p)\leqslant core\text{-}dist_{old}(q)\}.$$

(c) 核心可达距离增大的数据点:该类数据的集合标记为 *IncreRDist*(*p*),则

$$IncreRDist(p)=\{q \mid q\in N_{Eps}^{D}(IncreCDist(p)\cup ExpiredC(p))\text{且}reach\text{-}dist_{new}(q)>reach\text{-}dist_{old}(q)\}.$$

因此,对过期数据点的删除可看作是加入新数据点的一个逆过程.如图 7 所示,若点 *p* 是过期数据点,由于点 *p* 的删除,q_1 由核心点退化成边界点,o_2 的核心距离也增大了.同时,q_1 的核心可达距离也增大了,由相对于点 *p* 到相对于点 o_2;点 q_2 的核心可达距离也增大了,由相对于点 *p* 转变为相对于点 o_1.

相应地,对过期数据的处理与对新数据的处理过程类似,也可以通过两个步骤实现:步骤 1 找出退化的核心点集合 *ExpiredC*(*p*)和核心距离增大的数据集合 *IncreCDist*(*p*);步骤 2 在这两个集合基础上进一步扩展处理 *IncreRDist*(*p*)集合中的每个元素.

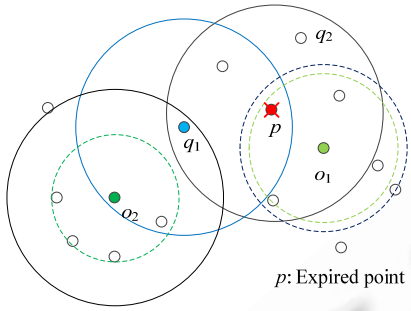


Fig.7 Effect of expired point on neighboring points

图 7 过期数据点对周围数据的影响

2.3 滑动窗口挖掘算法实现

基于滑动窗口的密度簇结构挖掘算法 MCluStream 伪代码如算法 2 所示.当新的数据到达当前窗口 W_c 时,首先采用 *InsertNewPoint* 处理新到达数据点,然后采用 *DeleteExpiredPoint* 处理过期数据.

算法2. MCluStream (mining clustering struture in data stream).

Parameter: *Eps* and Min*Pts*

Input: W_c and *CR-Tree*.

Output: updated *CR-Tree*.

Algorithm:

```
1: for each p ∈ W_c.New do
2:    InsertNewPoint(p);
3: for each p ∈ W_c.Expired do
4:    DeleteExpiredPoint(p);
InsertNewPoint(point p)
1:  list ← NewC(p);
2:  list ← list ∪ ReduceCDist(p);
3:  get corePt ← getCorePoint(p);  //nearst core point
4:  setReachDist(p,corePt);
5:  p.father ← corePt;
6:  foreach (q in list)
7:    foreach (d in N_Eps^D(q))
8:      expandUpdated(q,d);
    expandUpdated(corePt,d)
9:  j ← corePt.core-dist;
10: dist ← dist(corePt,d);
11: t ← (dist < j)? j : dist;
12: if (t < d.reach-dist)
13:   if (d is not ancestor of corePt)
14:     d.reach-dist ← t;
15:     d.father ← corePt;
16:   else if (reach-dist(corePt,d.father) < d.reach-dist)
17:     corePt.father ← d.father;
18:     corePt.reach-dist ← reach-dist(corePt,d.father)
```

19: $d.father \leftarrow corePt$;
20: $d.reach\text{-}dist \leftarrow t$;
DeleteExpiredPoint (*point p*)
1: $list1 \leftarrow ExpiredC(p)$;
2: $list2 \leftarrow IncreCDist(p)$;
3: foreach (q in $list1 \cup list2$)
4: foreach (d in $N_{Eps}^{D}(q)$)
5: exp*andUpdated*($q,d,list1,list2$);
exp*andUpdated*($q,d,list1,list2$)
6: get $corePt \leftarrow d.father$;
7: if ($corePt \in list1 \cup list2$)
8: get $corePt \leftarrow getCorePoint(d)$;
9: if (d is not ancestor of $corePt$)
10: $d.reach\text{-}dist \leftarrow reach\text{-}dist(d,corePt)$;
11: $d.father \leftarrow corePt$;
12: else if ($reach\text{-}dist(corePt,d.father) < d.reach\text{-}dist$)
13: $corePt.father \leftarrow d.father$;
14: $corePt.reach\text{-}dist \leftarrow reach\text{-}dist(corePt,d.father)$
15: $d.father \leftarrow corePt$;
16: $d.reach\text{-}dist \leftarrow reach\text{-}dist(d,corePt)$;
17: detele p from *CR-Tree*;

在 *InsertNewPoint* 中,首先执行步骤 1,如第 1 行、第 2 行,本文实现的搜索过程都采用了 grid 索引,详见文献[18].接着处理新点 p,如第 3 行~第 5 行.首先找到点 p 直接核心可达的数据点 *corePt*,由于点 p 一定不在 CR-Tree 上,所以直接将点 p 指向点 *corePt*.然后,依次对核心可达距离减小的数据点进行处理,该处理过程如第 9 行~第 18 行所示.当数据点 d 核心可达距离减小,且其直接核心可达的相对核心点也发生变化时,首先判断点 d 是否是它的新相对核心点的祖先节点,即,是否会产生循环链:若不是,则直接修改点 d 的父亲节点;若是,则进行第 16 行~第 20 行的操作,以避免 CR-Tree 中循环链的产生.第 16 行~第 20 行代码表示:如点 d 是 *corePt* 的祖先节点,如果 *corePt* 相对于 *d.fathter* 的可达距离小于点 d 的可达距离,则将 *d.father* 更新为 *corePt* 的父亲节点,将 *corePt* 更新为点 d 的新父亲节点;否则,保持原来状态不变.

过期数据的处理方与新数据点的处理方法相似,在 *DeleteExpiredPoint* 函数中,同样先执行步骤 1,如第 1 行、第 2 行.然后,依次处理过期数据影响到的数据点.处理过程如第 7 行~第 16 行所示,对于核心可达距离增大的数据点 d,找到它新的相对核心点,重新定位它的父亲节点,这个过程与更新核心可达距离减小的数据点的方法相同.最后,将过期数据点 p 从 CR-Tree 中删除.

- 时间复杂度

假设滑动窗口内的数据服从正态分布 $N(\mu,\delta^2)$,随机选择两个数据点 s_1 和 s_2,这两个点的距离小于等于 *Eps* 的概率表示为 $p=p(|s_1-s_2|\leqslant Eps)$,由于点 s_1 和 s_2 都服从正态分布 $N(\mu,\delta^2)$,设定 $z_1=(s_1-\mu)/\delta, z_2=(s_2-\mu)/\delta$,则 z_1 和 z_2 服从正态分布 $N(0,1)$.

$$\begin{aligned} p(|s_1-s_2|\leqslant Eps) &\Leftrightarrow p(s_1-Eps\leqslant s_2\leqslant s_1+Eps)\\ &\Leftrightarrow p((s_1-\mu)/\delta-Eps/\delta\leqslant(s_2-\mu)/\delta\leqslant(s_1-\mu)/\delta+Eps/\delta)\\ &\Leftrightarrow P(Z\leqslant z_1+Eps/\delta)-P(Z\leqslant z_1-Eps/\delta). \end{aligned}$$

对于正态分布的数据点,其 *Eps* 邻居点数量为$(n-1)\times p$.

根据统计学理论,在$\mu\pm3\times\delta$范围内的数据点被认为是聚类簇中的数据点[19].若设定 $MinPts=\alpha\times n$,其中,α为 *MinPts* 占总数据点 n 的比值,取值范围为(0,1),当 $s_1\leqslant\mu-3\times\delta$,即 $z_1\leqslant-3.0$,则

$$p(s_1-Eps\leqslant S\leqslant s_1+Eps)\leqslant\alpha\Leftrightarrow P(Z\leqslant z+Eps/\delta)-P(Z\leqslant z-Eps/\delta)\leqslant\alpha.$$

由此可推出 Eps 是α的函数,设为 $Eps=f(\alpha)\times\delta$.也就是说,当 Eps 满足这个条件时,滑动窗口内的数据点才能满足 99.72%(即 $3\times\delta$以内的数据点)的点不是异常点.因此,

$$p=P(Z\leqslant z_1+Eps/\delta)-P(Z\leqslant z_1-Eps/\delta)<P(Z\leqslant 0+f(\alpha))-P(Z\leqslant 0-f(\alpha))=2\times\varphi(f(\alpha))-1.$$

对于标准正态分布的数据来说,点 z_1 越靠近均值 0,邻居点数量越多.

在最坏的情况下,新点或过期点的邻居点的核心可达距离都发生了变化,更新时间的消耗为 $O(2\times(n-1)\times p\times(n-1)\times p)$,则时间复杂度为 $O(p^2\times n^2)$.由上述分析,在正态分布下,MCluStream 算法的时间复杂度的上界为 $O((2\times\varphi(f(\alpha))-1)^2\times n^2)$.因此在滑动窗口下,数据点数目 n 不变,算法在单个窗口内聚类时间基本稳定,这与第 4.3 节的实验测试结果基本一致.

- 空间复杂度

设定窗口内数据点数量为 n,算法 2 中每个数据点构成一个树节点,设定每个树节点占用空间为 k,则 CR-Tree 的空间复杂度为 $O(k\times n)$.需要更新的核心点的个数设为 l 个,则算法 2 的空间复杂度为 $O(k\times n+l)$.

3 簇结构提取算法

MCluStream 算法在线维护数据流中当前窗口数据的 CR-Tree 结构,以保持对数据流中簇结构的更新和演化分析.CR-Tree 蕴含了数据点间核心可达的密度关系,当有用户请求查看当前窗口的簇结构时,可通过简单转化提取出体现密度簇结构的数据排序列表,提取方法如算法 3 所示.

算法3. ExtraClu (extract clustering structure).

Input: W_c and *CR-Tree*;

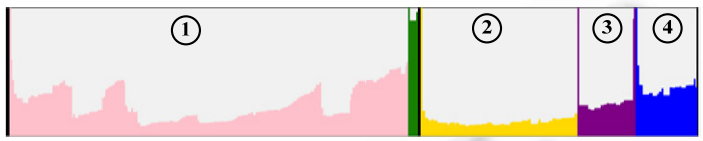
Output: order points list *L*.

Algorithm:

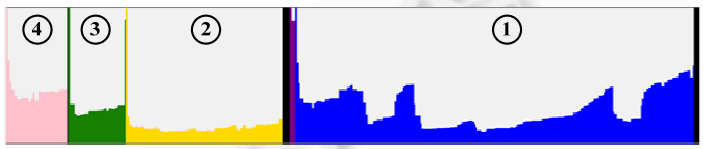
```
1 : foreach (dp in CR-Tree)
2 :   if (dp.extracted is false)
3 :     transform(dp);
    transform(dp)
4 : if (dp.father is null || L.contains(dp.father))
5 :   L.add(dp.father,dp);
6 : else
7 :   transform(dp.father)
    L.add(corePt, p)
8 : if (corePt==null)  L.addAtEnd(p);
9 : else
10:   while (++(i ← order_D(corePt)) < L.count)
11:     if (L[i].reach-dist ⩾ p.reach-dist)
12:       L.insertAt(i, p); break;
13:   if (i == L.count)
14:     L.addAtEnd(p);
```

随机从 CR-Tree 中任意一个数据点 *dp* 开始,若点 *dp* 没有被提取过(未被插入到排序点列表 *L*),则开始提取 *dp* (第 1 行~第 3 行描述).若点 *dp* 的父亲节点为空,则直接将点 *dp* 插入到列表 *L* 的尾端(第 8 行描述).若点 *dp* 的父亲节点在 *L* 中,则从其父亲节点在列表 *L* 的序号位置之后开始,比较点 *dp* 与列表 *L* 中数据点的核心可达距离值,直到点 *dp* 的可达距离大于或者等于列表 *L* 中某点的可达距离,然后将点 *dp* 插入到该位置(第 10 行~第 12 行);若直到 *L* 结尾都大于点 *dp* 的核心可达距离,则将点 *dp* 插入到列表尾端,如第 13 行、第 14 行描述.若点 *dp* 的父亲节点不在列表 *L* 中,则先嵌套处理 *dp* 的父亲节点,如第 6 行、第 7 行代码描述.

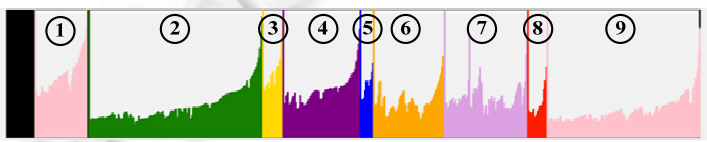
ExtraClu 提取尽量仿照 OPTICS 的排序过程,但是 ExtraClu 仅从 CR-Tree 中通过比较部分数据的核心可达距离排序数据,提取过程非常迅速.得到的排序列表 L 可视化显示后的效果如图 8 所示,选择一个较大的 *subEps*,可以可视化地显示当前窗口所有数据的簇结构.根据可视化的簇结构,用户可自主选择 *subEps*,以获取不同 *Eps* 参数下的聚类结果.



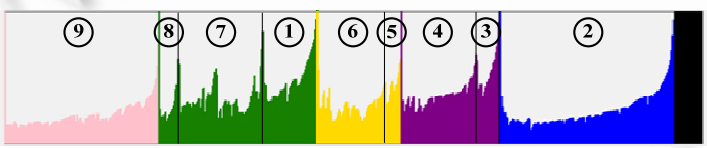
(a) MCluStream在SEQ上挖掘的簇结构



(b) OPTICS在SEQ上挖掘的簇结构



(c) MCluStream在SSD上挖掘的簇结构



(d) OPTICS在SSD上挖掘的簇结构

Fig.8 Effectiveness comparison of MCluStream and OPTICS

图 8 MCluStream 与 OPTICS 的挖掘效果比较

4 实验测试及分析

本节对 MCluStream 算法的挖掘效果、聚类效率、参数的敏感度进行了综合实验评估,并与 OPTICS, OLDStream[16]和 Chandi[14]进行了对比.其中,OPTICS 包括两种实现方法:一是 OPTICS 原始算法,二是使用了 grid 索引[16]优化的 OPTICS-G 算法.

4.1 实验数据和测试方法

实验平台采用 2.4GHz i5 处理器,3G 内存,Windows 7 操作系统.

测试数据. 真实数据 SEQ 是公开的 SEQUOIA 2000 数据的一个子集,同时包含多个任意形状的、不同密度的数据簇以及重叠子簇.合成数据 SSD 是一个采用 Matlab 合成的数据集,包含 4 413 个分布于 1000×1000 的正方形区域的数据点,其中约 300 个噪点.真实数据 GEO 采用微软亚洲研究院的 Geolife 项目采集的 GPS 轨迹数据[18],该数据集收集了 178 位用户的 2 360 多万个位置信息.合成数据 VLD 是一个大规模合成数据集,它包含了分布于 5000×5000 范围内的 119 万多个空间数据点.

测试方法. 在 CHAOS[20]流数据分析平台上连续输入数据集,模拟海量高速数据流.为便于识别算法的聚类效果,在两个小规模数据 SEQ 和 SSD 上测试了算法的挖掘效果.在大规模数据集 GEO 和 VLD 上评估了算法的效率和参数敏感性,测量了数据流最关注的处理时间指标(CPU 时间)和空间开销指标(内存).

4.2 效果测试

在 SEQ 和 SSD 数据上评估了 MCluStream 的挖掘效果、重叠簇识别和聚类演化分析,评估方法采用与 OPTICS 的聚类结果相比较的方式.两算法在相同数据集上采用相同参数:SEQ:*Eps*=20,*MinPts*=2;SSD:*Eps*=50, *MinPts*=5.

为了将 SEQ 和 SSD 内所有数据都放入窗口,设定窗口长度 *w*=5k,得到的排序列表如图 8 所示,横坐标为数据点序列,纵坐标为各数据点的核心可达距离.图 8(a)和图 8(c)为 MCluStream 挖掘结构,图 8(b)和图 8(d)为 OPTICS 的簇结构.为了便于分辨,我们选取了一个较大的 *subEps*,并采用不同的颜色标示簇结果,并将两种算法对应的簇用相同的序号标示出来,提取出的排序列表可将簇结构清晰地可视化展现出来.数据点的可达距离值反映了数据集合的密度,较低可达距离的数据点集具有较高的密度,如图中同一颜色的数据点集中,处于低洼的数据点集相对于周围的数据点具有较高的聚集密度.从图中相同序号簇结构的对比可以看出,MCluStream 可发现和 OPTICS 相似的簇结构.由于两种算法对数据的处理方式不同,MCluStream 通过更新 CR-Tree 在线维护每个数据点的最短可达距离,通过转换算法提取簇结构,而 OPTICS 则从某点开始螺旋式处理排序数据,它所获得的排序列表中,数据的可达距离不一定是最短可达距离.两种方法挖掘出的簇结构中的簇排序并不相同,但独立的簇结果可相互对应,且内部簇结构基本相似的.

在 SEQ 数据集上,我们对 MCluStream 识别簇重叠的能力进行了评估.如图 9 所示,图 9(a)~图 9(c)为 MCluStream 在选择不同 *subEps* 下的簇结果,图 9(d)~图 9(f)为 OPTICS 分别在对应参数下的聚类结果.在同一簇结构中,MCluStream 选择不同的 *subEps* 可以获取到不同密度的簇结果.通过减小 *subEps* 可获取高密度的子簇,以实现对簇重叠的识别.与 OPTICS 的对比,可以充分验证 MCluStream 能够正确地挖掘出数据流中的簇结构,且具有相同的簇重叠识别能力.

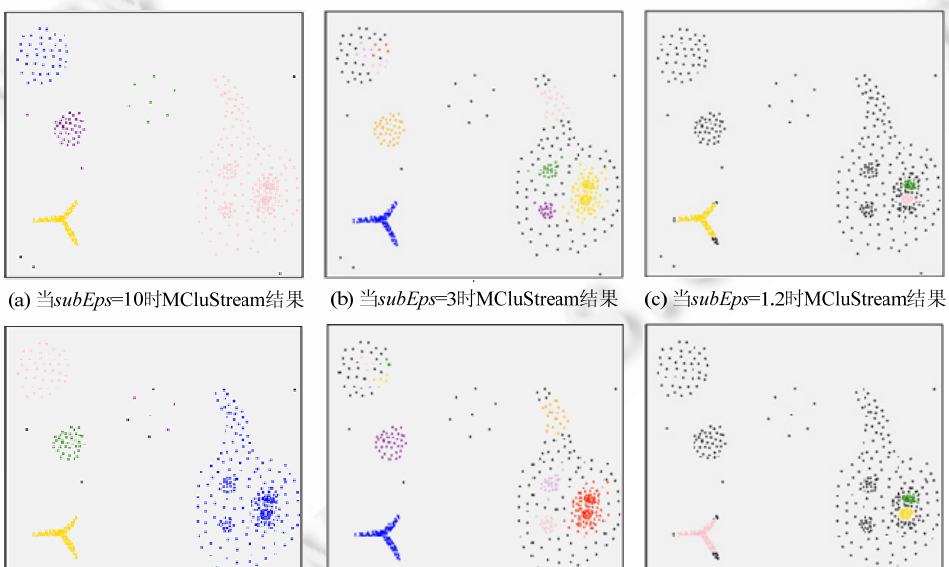
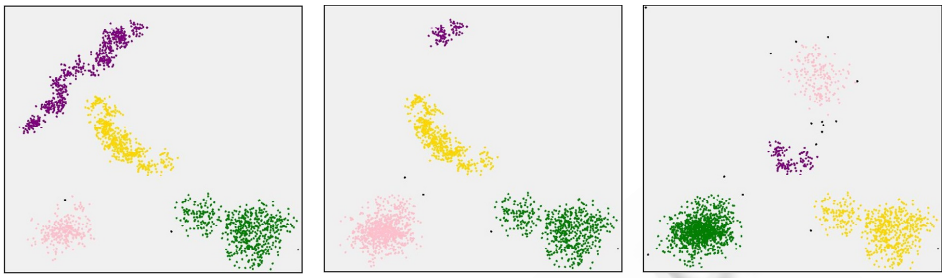


Fig.9 Comparison of identifying overlapping clusters

图 9 簇重叠识别性能对比

此外,我们还在 SSD 数据集验证了 MCluStream 对聚类结果的演化分析.如图 10 所示,设定窗口长度 *w*=2k,我们在 *t*=3k,*t*=3.5k 和 *t*=4k 时检测了当前窗口的聚类结果,取 *subEps*=30,聚类结果分别如图 10(a)~图 10(c)所示.从图中结果可以清楚地看到,MCluStream 能够对数据流进行实时演化聚类分析.



(a) t=3k 聚类结果 (b) t=3.5k 聚类结果 (c) t=4k 聚类结果

Fig.10 Evolving clustering results

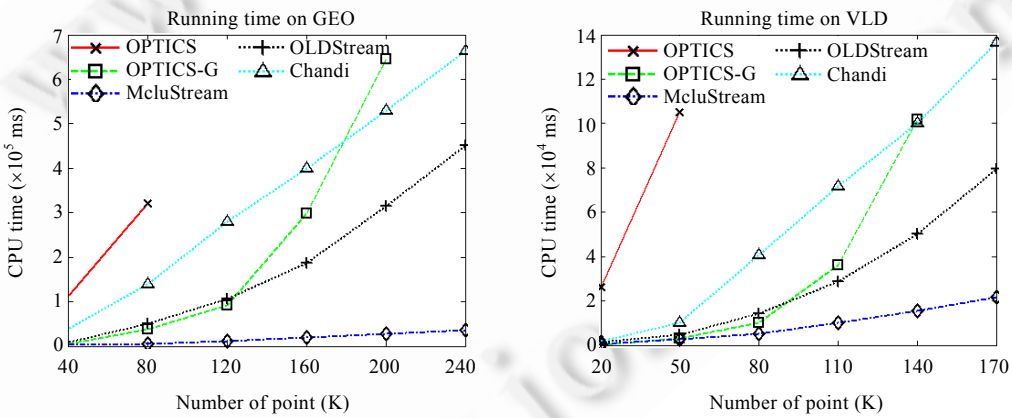
图 10 演化聚类结果

4.3 性能测试

本节在大规模数据 GEO 和 VLD 上测试了 MCluStream 的挖掘效率和空间上的内存消耗,并与 OPTICS, OPTICS-G,OLDStream 和 Chandi 效率进行了比较.各算法在同一数据集上设置相同的参数:

- GEO:*Eps*=50,*MinPts*=5;
- VLD:*Eps*=100,*MinPts*=10.

Chandi 设定 *MinPts* 不变,同时执行 10 个聚类查询.在 GEO 上,*Eps* 从 20 到 50 变化;在 VLD 上,*Eps* 从 50 到 100 变化.设定窗口长度 w=50k.随着数据流,各算法的运行时间如图 11 所示,其中,MCluStream,OLDStream 和 Chandi 为累积时间,OPTICS 为运行时间.



(a) 在 GEO 上的运行时间比较 (b) 在 VLD 上的运行时间比较

Fig.11 Efficiency comparison of algorithms

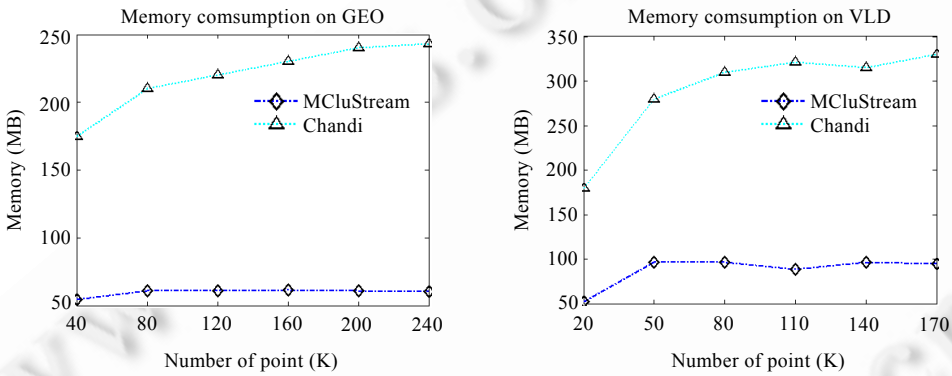
图 11 算法效率对比分析

在 GEO 和 VLD 数据集上,MCluStream 都表现出了巨大的效率优势;而且随着数据流的处理,优势越来越明显.OPTICS 消耗最多的 CPU 时间,且随着数据量的增大,时间消耗快速增长,并远远超过其他算法.Chandi 累积时间虽然大于 OLDStream,但由于同时执行 10 个聚类查询,平均效率高于 OLDStream,在 GEO 上平均处理时间为 3ms/点.MCluStream 累积时间基本保持线性增长,统计结果显示,在 w=50k 时,平均处理速度为 0.13ms/点.在 GEO 上,当数据量为 200K 时,MCluStream 的处理速度比 OLDStream,Chandi 和 OPTICS-G 分别提高了 12 倍、20 倍和 24 倍.

OPTICS 需要维护一个巨大的排序列表,并对每个数据点进行邻居查询、计算可达距离,并对数据排序.大量

计算导致了巨大的时间花销,尤其是当处理大规模数据集时.OPTICS-G 虽然采用 grid 索引降低区域邻居查询的时间消耗,但是当数据量较大且分布比较紧密时,OPTICS-G 仍需消耗大量时间处理数据的排序问题.Chandi 算法采用共享执行策略可同时执行多个聚类查询,在滑动窗口环境下平均处理时间稳定,但当执行 10 个查询时,CUP 消耗已远大于 MCluStream.而 MCluStream 采用 CR-Tree 方法维护密度簇结构,数据点间通过父子关系、祖先与子孙关系表示直接核心可达和核心可达的密度关系.当数据点之间的关系发生变化时,只需通过更新父亲节点来改变直接核心可达关系,进而同步更改级联的核心可达关系,极大地降低了时间成本.同时由于数据影响的局部性,MCluStream 算法能够保证线性的时间复杂度,从而保证具有较高的可伸缩性.

MCluStream 和 Chandi 的随窗口滑动时的内存消耗测量结果如图 12 所示.在两个数据集上,MCluStream 同样具有较大的空间开销优势,与 Chandi 相比,分别平均节省了 72.5%和 70.8%的内存消耗.这是因为 MCluStream 仅维护一个 CR-Tree 结构,而 Chandi 需要维护 10 个聚类集合的层次结构.MCluStream 随着窗口滑动,其内存消耗基本保持不变,这与空间复杂度的分析结果基本一致.



(a) 在 GEO 上的内存消耗比较 (b) 在 VLD 上的内存消耗比较

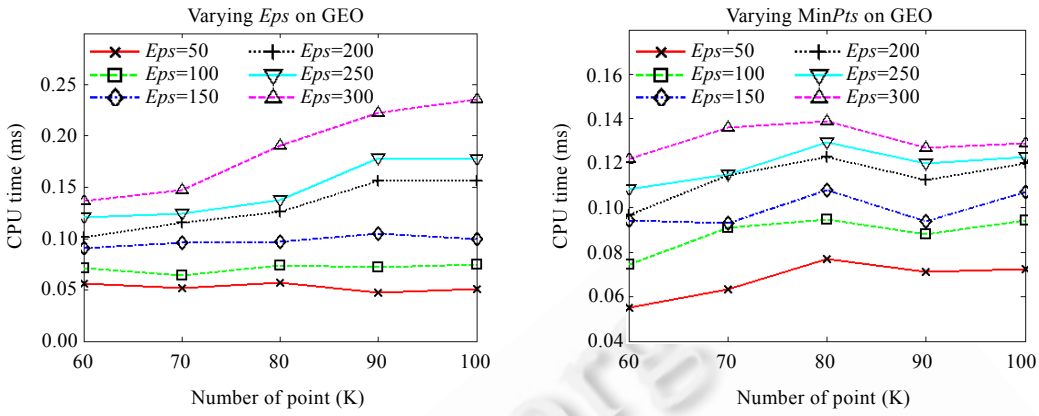
Fig.12 Memory comparison of algorithms

图 12 算法内存对比

4.4 敏感度分析

MCluStream 可以从数据流中挖掘密度簇结构以降低数据流密度聚类算法对输入参数选择的困难,但仍需要两个较为松弛的参数输入:*Eps* 和 *MinPts*.本节在 GEO 上评估了 MCluStream 聚类效率对输入参数的敏感度,窗口长度 w=50k,每次滑动 1 个数据点.实验结果如图 13 所示,图 13(a)为 MCluStream 相对于 *Eps* 随数据流每 10k 个窗口的单个窗口的平均消耗 CPU 时间,图 13(b)为 MCluStream 相对于不同的 *MinPts* 的平均 CPU 时间.可以看出,*Eps* 和 *MinPts* 对 MCluStream 的运行效率都没有明显的影响:

- 当设定 *MinPts*=10 时,随着 *Eps* 的增加,MCluStream 的聚类时间也相应地增加,但 *Eps* 在 150 以内时,算法单个窗口的 CPU 时间基本平稳;在 200 以上时,CPU 时间稍有增加.这是由于 MCluStream 采用 grid 索引优化算法的区域邻居查询,因此当 *Eps* 增长时,查询区域的增大造成搜索邻居的时间增加.但是,MCluStream 获取的簇结构并不直接依赖于 *Eps*,当 *Eps* 达到一定大小后,进一步地增加 *Eps* 并不会带来算法挖掘效果的显著提升.
- 当 *Eps* 固定为 100 时,*MinPts* 的变化对 MCluStream 的运行时间影响较小,这是由于 *MinPts* 仅影响数据点的核心距离和可达距离.随着 *MinPts* 的增长,单个窗口的 CPU 消耗有缓慢增加,但随着数据流,滑动窗口的平均消耗时间非常平稳.



(a) 相对于 *Eps* 的效率 (b) 相对于 *MinPts* 的效率

Fig.13 Efficiency of MCluStream with respective to parameters

图 13 MCluStream 效率相对于输入参数分析

5 总 结

本文针对数据流密度聚类问题,实现了一种从海量数据流中提取密度簇结构挖掘算法 MCluStream,以解决数据流密度聚类算法对输入参数的选择和重叠簇识别问题.首先,设计了一种 CR-Tree 索引结构维护数据流中数据点的密度关系;其次,采用滑动窗口的方式处理新到达数据和过期数据,在线更新当前窗口内数据的 CR-Tree 簇结构;此外,在离线情况下,用户可实时请求从当前窗口的 CR-Tree 中提取密度簇结构,并根据可视化的簇结构,合理选择 *subEps* 生成具体聚类结果;最后,综合评估实验验证了算法具有较好聚类效果和演化分析能力、较高的效率及可伸缩性.下一步,我们将重点研究适应于 *MinPts* 选择的密度簇结构挖掘算法和滑动窗口环境下的数据优化处理策略.

致谢 在此,特别感谢伍斯特理工学院的 Elke Rundensteiner 教授为本文提供了 Chandi 等相关算法资料;同时,对本文工作给予支持和建议的同行及评审老师深表感谢.

References:

[1] Han JW, Kamber M, Pei J. Data Mining: Concepts and Techniques. 3rd ed., Morgan Kaufmann Publishers, 2011. 444−476.

[2] Aggarwal CC, Han JW, Wang JY, Yu PS. A framework for clustering evolving data streams. In: Proc. of the 29th Very Large Data Bases (VLDB) Conf. Berlin: VLDB Endowment, 2003. 81−92.

[3] Zhu WH, Yin J, Xie YH. Arbitrary shape cluster algorithm for clustering data stream. Ruan Jian Xue Bao/Journal of Software, 2006,17(3):379−387 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/379.htm [doi: 10.1360/jos170379]

[4] Cao F, Ester M, Qian WN, Zhou AY. Density-Based clustering over an evolving data stream with noise. In: Proc. of the 2006 SIAM Conf. on Data Mining. Bethesda: SIAM Press, 2006. 326−337.

[5] Chen YX, Tu L. Density-Based clustering for real-time stream data. In: Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2007. 133−142. [doi: 10.1145/1281192.1281210]

[6] Lee JG, Han JW, Whang KY. Trajectory clustering: A partition-and-group framework. In: Proc. of the SIGMOD Conf. Beijing, 2007. 593−604. [doi: 10.1145/1247480.1247546]

[7] Kranen P, Assent I, Baldauf C, Seidl T. Self-Adaptive anytime stream clustering. In: Proc. of 2009 the 9th IEEE Int'l Conf. on Data Mining (ICDM). 2009. 249−258. [doi: 10.1109/ICDM.2009.47]

[8] Kranen P, Assent I, Baldauf C, Seidl T. The clustree: Indexing micro-clusters for anytime stream mining. Knowledge and Information Systems, 2011,29(2):249−272. [doi: 10.1007/s10115-010-0342-8]

[9] Xu KS, Kliger M, Hero III AO. Adaptive evolutionary clustering. Data Mining and Knowledge Discovery, 2014,28(2):304−336. [doi: 10.1007/s10618-012-0302-x]

[10] Ester M, Kriegel HP, Sander J, Xu XW. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the KDD. Portland: AAAI Press, 1996. 226−231.

[11] Xia LN, Jing JW. SA-DBSCAN: A self-adaptive density-based clustering algorithm. Journal of the Graduate School of the Chinese Academic of Sciences, 2009,26(4):530−538 (in Chinese with English abstract).

[12] Cai YK, Xie KQ, Ma XJ. An improved DBSCAN algorithm which is insensitive to input parameters. Acta Scicentiarum Naturalum Universitis Pekinesis, 2004,40(3):480−486 (in Chinese with English abstract).

[13] Yang D, Rundensteiner EA, Ward MO. A shared execution strategy for multiple pattern mining requests over streaming data. In: Proc. of the VLDB 2009. 2009. [doi: 10.14778/1687627.1687726]

[14] Yang D, Rundensteiner EA, Ward MO. Shared execution strategy for neighbor-based pattern mining requests over streaming windows. ACM Trans. on Database System, 2012,37(1):Article No.5. [doi: 10.1145/2109196.2109201]

[15] Ankerst M, Breunig M, Kriegel HP, Sander J. OPTICS: Ordering points to identify the clustering structure. In: Proc. of the ACM SIGMOD'99. 1999. [doi: 10.1145/304182.304187]

[16] Yu YW, Kuang J, Wang Q, He J. An on-line density-based clustering algorithm for spatial data stream. Acta Automatica Sinica, 2012,38(6):1051−1059 (in Chinese with English abstract).

[17] Arasu A, Babu S, Widom J. The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal, 2006,15(2):121−142. [doi: 10.1007/s00778-004-0147-z]

[18] Zheng Y, Xie X, Ma WY. GeoLife: A collaborative social networking service among user, location and trajectory. IEEE Data Engineering Bulletin, 2010,33(2):32−40.

[19] Freedman DA, Pisani R, Purves RA. Statistics. 4th ed., New York: W. W. Norton & Company, Inc., 2007.

[20] Gupta C, Wang S, Ari I, Hao M. CHAOS: A data stream analysis architecture for enterprise applications. In: Proc. of the Commerce and Enterprise Computing 2009. IEEE, 2009. 33−40. [doi: 10.1109/CEC.2009.74]

附中文参考文献:

[3] 朱蔚恒,印鉴,谢益煌.基于数据流的任意形状聚类算法.软件学报,2006,17(3):379−387. http://www.jos.org.cn/1000-9825/17/379.htm [doi: 10.1360/jos170379]

[11] 夏鲁宁,荆继武.SA-DBSCAN:一种自适应基于密度聚类算法.中国科学院研究生院学报,2009,26(4):530−538.

[12] 蔡颖琨,谢昆青,马修军.屏蔽了输入参数敏感性的 DBSCAN 改进算法.北京大学学报(自然科学版),2004,40(3):480−486.

[16] 于彦伟,邝俊,王沁,何杰.一种基于密度的空间数据流在线聚类算法.自动化学报,2012,38(6):1051−1059.

于彦伟(1986－),男,山东菏泽人,博士,讲师,CCF 会员,主要研究领域为数据挖掘,无线传感器网络.

王沁(1961－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为无线传感器网络,嵌入式系统,数据处理与分析.

王欢(1990－),男,硕士生,主要研究领域为机器学习,数据挖掘.

赵金东(1974－),男,博士,副教授,主要研究领域为智能数据处理,无线传感器网络.