ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.5, May 2010, pp.1031−1041
doi: 10.3724/SP.J.1001.2010.03470

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

一种基于时态密度的倾斜分布数据流聚类算法*

杨 宁+, 唐常杰, 王 悦, 陈 瑜, 郑皎凌

(四川大学 计算机学院,四川 成都 610065)

Clustering Algorithm on Data Stream with Skew Distribution Based on Temporal Density

YANG Ning+, TANG Chang-Jie, WANG Yue, CHEN Yu, ZHENG Jiao-Ling

(College of Computer Science, Sichuan University, Chengdu 610065, China)

+ Corresponding author: E-mail: yneversky@gmail.com, http://cs.scu.edu.cn/~yangning

Yang N, Tang CJ, Wang Y, Chen Y, Zheng JL. Clustering algorithm on data stream with skew distribution based on temporal density. *Journal of Software*, 2010,21(5):1031−1041. http://www.jos.org.cn/1000-9825/3470.htm

Abstract: To solve the problem of clustering this paper proposes a concept of temporal density, which reveals a set of mathematical properties, especially the incremental computation. A clustering algorithm named TDCA (temporal density based clustering algorithm) with time complexity of $O(c\times m\times \lg m)$ is created with a tree structure implemented for both storage and retrieve efficiency. TDCA is capable of capturing the temporal features of a data stream with skew data distribution either in real time or on demand. The experimental results show that TDCA is functionable and scalable.

Key words: data stream clustering; temporal density; skew distribution

摘 要: 为解决倾斜分布的数据流聚类这一难题,提出了时态密度概念,给出其度量,揭示了其包括可增量计算在内的一系列数学性质;设计了时态密度树结构,提高了聚类时的存储和检索效率;设计了能够以实时或异步方式捕捉数据倾斜分布的数据流时态特征的聚类算法 TDCA(temporal density based clustering algorithm),其时间复杂度为 $O(c\times m\times \lg m)$.实验结果表明,该算法不仅有较强的功能,而且具有较好的规模可伸缩性.

关键词: 数据流聚类;时态密度;倾斜分布

中图法分类号: TP311 **文献标识码**: A

数据流日益广泛地出现在传感器网络、网络监测、电话通话记录、Web 点击事件以及股票交易数据处理中.面向数据流的知识发现已经成为数据挖掘领域的重要研究内容[1−6].

社会热点监测和 Web Usage 挖掘是数据流聚类的两个典型应用.在社会热点监测中,不同时期人群关心的热点事件会随时间而演变,同一时期不同人群关心的事件也是不同的,有些事件可能是潜在的热点事件(人群密度还比较稀疏,但是已经具备聚集的特点).在 Web Usage 挖掘中,用户群对不同 Web 页面的兴趣也会随时间而发生变化,离当前时刻越近的访问数据越有意义,而且不同用户群有不同的兴趣,用户群数量是不同的,在分布上

* Supported by the National Natural Science Foundation of China under Grant No.600773169 (国家自然科学基金); the National Key Technology R&D Program in the 11th Five-Year Plan of China under Grant No.2006BAI05A01 (国家"十一五"科技支撑计划)

Received 2008-02-25; Accepted 2008-10-07; Published online 2009-04-10

是倾斜的.

上述应用对数据流聚类算法提出了挑战,集中表现为:(1) 难以获取关于簇个数和形状的先验知识;(2) 要求较高的灵活性:要能同时处理数据流聚类的时态特征和倾斜分布特征,即要求能够实时捕捉空间中局部密度不同的簇,以及簇的个数、密度、形状等性质随时间的变化情况;(3) 需要时态权重,即数据对于聚类的贡献随时间衰减,离现在越近的数据,对聚类贡献越大;(4) 要求较高的时空效率:无法保存数据流全局的和历史的数据,只能访问一个特定窗口内的数据,而且只能进行单遍扫描,所以要尽可能地节省存储空间和处理时间.

本文提出了一种数据流聚类算法 TDCA(temporal density based clustering algorithm).该算法不需要关于簇个数和形状的先验知识,既能捕捉数据流时态特征,又能同时处理数据倾斜分布的问题.主要贡献包括:(1) 提出了时态密度(temporal density)概念,定义了具有时间和空间自适应特性的时态密度阈值函数,证明了它们的重要数学性质.时态密度和阈值函数既能体现数据的时态特征,又能反映空间倾斜分布;(2) 提出了基于时态密度的聚类概念;(3) 设计了时态密度树(temporal density tree)来压缩存储密度图,提高了数据流聚类时的存储效率和搜索效率;(4) 提出了基于时态密度的数据流聚类算法 TDCA.

1 相关工作

文献[7,8]提出了基于聚类数 k 的 STREAM 算法;文献[9]提出了数据流聚类的 CluStream 算法.这些文献虽然对传统聚类算法进行了扩展,但仍然有以下不足:(a) 其基本思想是基于距离的,难以发现任意形状的聚类;(b) 需要关于簇个数等先验知识,不能动态识别数据流聚类的变化.文献[10−13]提出了基于密度的算法,克服了一般算法无法发现任意形状的聚类的缺点,但仅局限于处理静态数据,无法直接应用于数据流聚类.

文献[14]将基于密度的方法引入 CluStream 算法,提出了 ACluStream 算法,能够识别任意形状的聚类.但是无法实时处理分布倾斜的数据.文献[15]研究了基于密度和格(cell)的数据流聚类算法,提出了 CDS-Tree 算法.这些算法的主要不足是:(a) 密度阈值是空间上全局一致的,因此不能处理空间分布倾斜的情况,无法区分局部密度不同的聚类.以图 1 为例,当阈值较小时,它们会把 $c2$ 和 $c3$ 识别为同一个聚类;当阈值较大时,它们只能识别 $c2$,不能识别 $c3$;(b) 密度阈值是时间上固定不变的,无法捕捉因为密度随时间变化而造成的聚类变化.因此在图 1 中,这些算法将无法意识到 $c3$ 的出现;(c) 数据没有时间权重,难以发现聚类结构的时间特征.体现在图 1 中,这些算法将无法识别由于部分数据的老化而引起的 $c1$ 到 $c2$ 的变化.

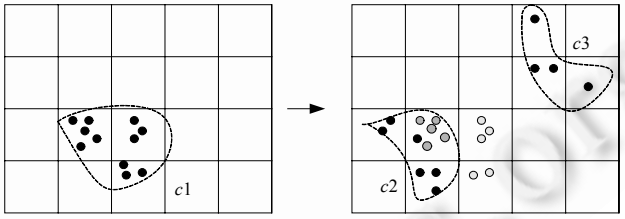


Fig.1 Temporal feature of clusters in data stream

图 1 数据流聚类的时态特征

已有算法不能实时处理倾斜数据的主要原因是,算法中的密度阈值是一个完全依赖于先验知识且事先设定的常量.与已有算法不同,本文在定义了时态密度以记录数据时间权重的基础上,定义密度阈值为一个时间的函数,同时考虑了不同局部区域密度的历史状态,历史上较为稀疏(稠密)的区域,现在的密度阈值就比较小(大),即密度阈值是空间和时间自适应的,因此能够区分数据空间不同区域的密度,并能识别各个局部密度随时间的变化情况.

2 时态密度的基本概念

k 维数据空间 $D=D_1\times D_2\times\ldots\times D_k$,$D_i$ 为第 i 维的定义域,$i=1,\ldots,k$.设 D_i 被划分成 s_i 个区间,则空间 D 被划分成 $M=s_1\times s_2\times\ldots\times s_k$ 个单元格(cell).一个单元格记为 $c(c_1,c_2,\ldots,c_k)$,其中,c_i 是单元格 c 在第 i 维上的区间号坐标.一个 k

维 D 点记为 $r(x_1,x_2,\ldots,x_k)$,其中,x_i 为点 r 在第 i 维上的坐标.如果一个点 r 落入某个单元格 c,则称点 r 属于单元格 c,记为 $r\in c$.

称 k 维数据流中的一个记录 r 为一个数据点,它对应于 k 维数据空间 D 中的一个点,其到达时间记为 t_r.

例 1:考察图 1 中所示流数据与时俱进的分布情况.在 t_1 时刻,数据流只有一个聚类 $c1$.在 t_2 时刻,出现了两个聚类 $c2$ 和 $c3$,其中,$c3$ 的密度明显小于 $c2$.此外,部分数据虽然继续存在,但其对于聚类的作用随着时间的流逝发生了不同程度的退化(如图中灰度不同的点所示),有些单元格因数据退化程度较大而失去参与聚类的意义.在这个过程中,不仅聚类的个数发生了改变,而且聚类的形状和密度都发生了改变.同时,数据的分布是倾斜的,表现为不同区域的簇的局部密度不同.

例 1 表明:(1) 数据点对于聚类的贡献随着时间变化,离当前时刻越近的数据点贡献程度越大;(2) 个体对于聚类的贡献,既与个体本身有关,也与簇中其他个体(尤其是新个体)有关;(3) 不同区域的簇的密度不同,而且也随时间而变化,在一大批应用中,这一过程近似于热体自然降温,即近似于指数衰减规律.

把上述观察形式化,我们得到:

定义 1(时态权重和时态密度). 设 r 是数据流中的一个数据点,c 是空间中的单元格.(1) r 在 t 时刻的时态权重是一个函数 $W(r,t)=2^{-\lambda(t-t_r)}$,其中,$\lambda\geq 1$ 是权重衰减速度,t_r 为数据点 r 到达的时刻;(2) c 中数据点在时刻 t 的时态权重之和 $D(c,t)=\sum_{r\in c}W(r,t)$ 称为 c 在时刻 t 的时态密度.

直观上,时态权重描述了数据点 r 在 t 时刻对聚类的贡献程度.新数据点的初始时态权重为 1,然后随时间呈指数衰减.如果一个单元格总有新的数据点到达,则单元格的时态密度将逐步增大;否则,将逐步减小.由此可以反映数据空间不同区域的密度随时间的变化情况.

命题 1. 给定时刻 t_0 和 t_1,$t_0>t_1$,如果 Δn 是从 t_0 到 t_1 时刻之间到达单元格 c 的数据点数,则有 c 在 t_1 时刻的时态密度 $D(c,t_1)=2^{-\lambda(t_1-t_0)}D(c,t_0)+\Delta n$.

证明:设在 t_0 时刻 c 中有 n_0 个数据点.由定义 1(2)可知,在 t_0 时刻,$D(c,t_0)=\sum_{i=1}^{n_0}W(r_i,t_0)$.由定义 1(1)可知,对于任意数据点 $r_i\in c$ 在 t_1 时刻的时态权重 $W(r_i,t_1)=2^{-\lambda(t_1-t_{r_i})}$,$i\in[1,n_0]$,则有

$$W(r_i,t_1)=2^{-\lambda(t_1-t_{r_i})}=2^{-\lambda(t_1-t_0)}2^{-\lambda(t_0-t_{r_i})}=2^{-\lambda(t_1-t_0)}W(r_i,t_0) \tag{1}$$

又 t_1 时刻到达的新数据点数量为 Δn,则由定义 1(2)和式(1)可得:

$$\begin{aligned}D(c,t_1)&=\sum_{i=1}^{n_0+\Delta n}W(r_i,t_1)\\&=\sum_{i=1}^{n_0}W(r_i,t_1)+\sum_{j=n_0+1}^{n_0+\Delta n}W(r_j,t_1)\\&=\sum_{i=1}^{n_0}[2^{-\lambda(t_1-t_0)}W(r_i,t_0)]+\sum_{j=n_0+1}^{n_0+\Delta n}2^{-\lambda(t_1-t_{r_j})}\\&=2^{-\lambda(t_1-t_0)}\sum_{i=1}^{n_0}W(r_i,t_0)+\sum_{j=n_0+1}^{n_0+\Delta n}2^{-\lambda(t_1-t_1)}\\&=2^{-\lambda(t_1-t_0)}\sum_{i=1}^{n_0}W(r_i,t_0)+\Delta n\\&=2^{-\lambda(t_1-t_0)}D(c,t_0)+\Delta n.\end{aligned}$$

结论成立. □

命题 1 表明,任意时刻单元格的时态密度由两部分构成:一部分是原有数据点衰减后的密度;另一部分是新到达的数据点的密度.命题 1 同时表明,单元格时态密度的更新可以按照增量的方式进行计算,避免每次计算单元格中所有点的时态密度,提高了计算效率.

命题 2. 设 R_t 是 t 时刻数据空间中所有点的集合,v 是数据流的平均速率,记 S 为 R_t 中所有点的时态权重之和,$S=\sum_{r\in R_t}W(r,t)$,则 $S\leq v/(1-2^{-\lambda})$.

证明:由定义 1 可知,

$$\begin{aligned}S &= \sum_{r\in R_t} W(r,t)\\ &= \sum_{r\in R_t} 2^{-\lambda(t-t_r)}\\ &= v2^{-\lambda(t-0)} + v2^{-\lambda(t-1)} + \ldots + v2^{-\lambda(t-t)}\\ &= v\sum_{\tau=0}^{\tau=t} 2^{-\lambda(t-\tau)}\\ &= v[(1-2^{-\lambda(t+1)})/(1-2^{-\lambda})]\\ &\le v[1/(1-2^{-\lambda})].\end{aligned}$$

结论成立. □

由命题1和命题2易知,任何时刻数据空间中的所有单元格的时态密度之和不大于 $v/(1-2^{-\lambda})$,全部单元格的时态密度平均值不超过 $v/[M(1-2^{-\lambda})]$.

定义 2(时态密度阈值). 设单元格 c 的时态密度最近一次被更新的时刻为 t_u,则单元格 c 在时刻 t 的时态密度阈值规定为 $\sigma(c,t,t_u)=[D(c,t_u)/M]\sum_{i=0}^{t-t_u}2^{-\lambda i}=D(c,t_u)[1-2^{-\lambda(t-t_u+1)}]/[M(1-2^{-\lambda})]$.

注意,定义2规定的密度阈值函数具有时间和空间的自适应性.首先,定义2考虑了密度随时间变化的因素,体现了数据流聚类的时态特征;其次,通过因子 $D(c,t_u)$ 反映了不同空间区域密度稀疏不同对阈值的影响,越稀疏的区域阈值越小,从而能够处理具有倾斜分布特征的聚类.

命题 3. 如果 $t\ge t_a\ge t_b$,则时态密度阈值满足不等式 $\sigma(c,t,t_b)\ge\sigma(c,t,t_a)$.

证明:由定义2可得:

$$\begin{aligned}\sigma(c,t,t_b) &= [D(c,t_b)/M]\sum_{i=0}^{t-t_b}2^{-\lambda i}\\ &= [D(c,t_b)/M]\sum_{i=0}^{t-t_a+(t_a-t_b)}2^{-\lambda i}\\ &= [D(c,t_b)/M]\sum_{i=0}^{t-t_a}2^{-\lambda i} + [D(c,t_b)/M]\sum_{i=t-t_a+1}^{t-t_b}2^{-\lambda i}\\ &\ge [2^{\lambda(t_a-t_b)}D(c,t_a)/M]\sum_{i=0}^{t-t_a}2^{-\lambda i}\\ &\ge [D(c,t_a)/M]\sum_{i=0}^{t-t_a}2^{-\lambda i}\\ &= \sigma(c,t,t_a).\end{aligned}$$

结论成立. □

命题3表明,同一单元格在不同的时间起点 t_u 对应的时态密度阈值函数是不同的.

3 基于时态密度的聚类

k 维数据空间可以看作是一张无向图,一个单元格就是图中的一个结点.称这个图为时态密度图,简称为密度图,其中每个结点存储了对应单元格的时态密度.

定义 3(密度图). k 维数据空间的密度图是一个二元组 $\langle V,E\rangle$,其中:(1) V 是结点的集合,每个结点记录了对应数据空间中的对应单元格的坐标和时态密度;(2) E 是边的集合,当且仅当两个结点 $c_a(c_{a1},c_{a2},\ldots,c_{ak})$ 和 $c_b(c_{b1},c_{b2},\ldots,c_{bk})$ 满足 $D(c_a,t)>0, D(c_b,t)>0$,且存在 $j(1\le j\le k)$,使(a) $c_{ai}=c_{bi}, i=1,\ldots,j-1,j+1,\ldots,k$;(b) $|c_{aj}-c_{bj}|=1$ 同时成立,则结点 c_a 和 c_b 之间存在边 $\langle c_a,c_b\rangle$,此时称这两个结点是相邻的.

密度图中的连通性遵循图论中连通性的一般理论.下面,我们在图论中连通分量的概念基础上给出密度连通分量的定义.

定义 4(密度连通分量). (1) 如果一个结点 c 在时刻 t 的时态密度 $D(c,t)\ge\sigma(c,t,t_u)$,则称该结点是稠密结点.(2) 如果两个结点 c_a 和 c_b 是稠密结点且是相邻的,则称它们是稠密相邻的.(3) 如果一个连通分量中的任意两个相邻结点都是稠密相邻的,则称该连通分量为密度连通分量.

定义 5(时态密度簇). 一个时态密度簇就是密度图中的一个密度连通分量.

命题 4. 密度图中的任何一个结点只属于一个时态密度簇.

证明:用反证法.图论中连通分量定义为图中的最大连通子图.假设某结点属于多个聚类,由定义 5 可知,这些簇都是连通分量,则这些连通分量通过该结点相邻,这与连通分量的定义矛盾.原命题成立. □

定义 5 表明,查找聚类得到的簇,就是找出密度图的各个密度连通分量.由于密度图的稀疏特性,为了高效完成密度连通分量的查找,需要设计一个既能保存拓扑信息和时态密度信息,又能紧凑存储稀疏密度图的数据结构.本文在文献[15]提出的索引树结构基础上加以扩展,设计了时态密度树,使其能够存储单元格的时态密度和时间戳,并加入额外的链接指针以便灵活地遍历叶子结点.

定义 6(时态密度树). k 维数据空间 $D=D_1\times D_2\times\ldots\times D_k$ 的时态密度树是一个三元组$\langle DimNodes,DenNodes,Edges\rangle$,其中:

(1) *Edges* 是时态密度树中边的集合;

(2) *DimNodes* 是维结点集合.维结点的结构为$\langle i_1,p_{i1},i_2,p_{i2},\ldots,i_n,p_{in},s\rangle$,其中,$i_m(1\le m\le k)$是第 i 维中数据不为空的区间的编号,且 $i_1<i_2<\ldots<i_k$;p_{im} 是指向下一维关联区间的指针;s 是指向兄弟结点的指针;

(3) *DenNodes* 是稠密结点集合.稠密结点的结构为$\langle c,e,cN,tU,s\rangle$,其中,$c$ 是该结点对应单元格的坐标,e 为其时态密度,cN 是该单元格所属簇的编号,tU 是该结点所对应的单元格的时态密度最近一次被更新的时间,s 是指向下一个兄弟或者堂兄弟结点的指针.

维结点分布在 0~k 层.第 i 层的维结点代表第 i 维 D_i 中有数据的分区.稠密结点位于时态密度树的叶子层,即第 k+1 层.一个稠密结点存储一个单元格的时态密度值.所有的稠密结点通过 s 指针链接成一个单链表,并用根结点的 s 指针指向第一个稠密结点.由于叶子结点都出现在同一层,所以时态密度树是一棵平衡树.

图 2 是一个 2 维数据空间的时态密度树.在图 2(a)中,单元格 $c(1,4),c(2,3),c(2,4),c(4,3),c(4,4)$的时态密度大于 0,图 2(b)是其对应的时态密度树.第 1 层根结点是对 D_1 维的非空数据区间的索引,第 2 层结点是对 D_2 维的非空区间的索引.k 维数据空间的时态密度树高度为 k+1.从图 2 可以看出:(1) 时态密度树只存储时态密度不等于 0 的单元格;(2) 时态密度树保存了单元格间的位置关系.由于稠密结点保存了单元格的坐标,因此可以迅速判断两个单元格是否相邻,从而加速查找密度连通分量.

一个稠密结点对应一个单元格,为方便描述,下文的叙述中认为二者是等价的.

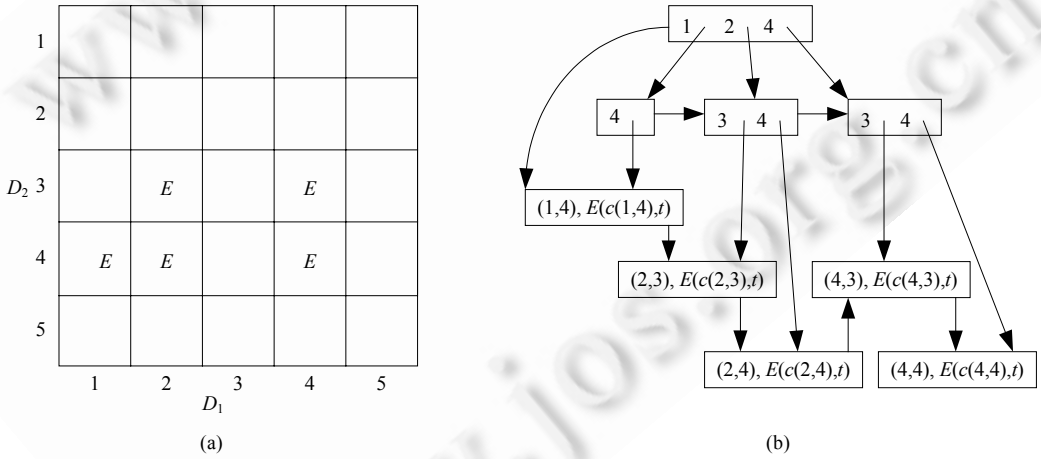


Fig.2 Examples of temporal density tree

图 2 时态密度树示例

4 基于时态密度的聚类算法 TDCA

基于时态密度的数据流聚类算法 TDCA 借鉴了 CluStream 算法[9]的处理方式,分为在线维护和离线聚类两个部分.在线维护部分周期性地更新时态密度树,离线聚类部分完成聚类任务.TDCA 算法采用时态密度树结构,实现密度图的高效存储和结点的检索.算法 1 描述了 TDCA 算法的主要步骤.

算法 1. *TDCA*(*Period*,*k*,*v*,*M*).

输入:处理周期 *Period*,维数 *k*,数据流平均速率 *v*,单元格数 *M*.

输出:基于时态密度的聚类.

```
(1)  initialize the root of temporal density tree;  //初始化时态密度树根结点 Root
(2)  initialize array Stream[];  //初始化数据流记录数组 Stream[]
(3)  while (1) {
(5)    delay the main thread Period seconds;  //延迟 Period 秒
(6)    read the new data records into Stream[];
(7)    Size=the number of new data;
(8)    if (Size==0) goto (10);
(9)    update the temporal density tree;  //更新时态密度树
(10)   if (demand of cluster){
(11)     clustering in a new thread;}}  //在新的线程中完成聚类
```

注意,在每个处理周期,如果有新的数据到达,则在线更新时态密度树(语句(9)).当收到用户聚类请求时,算法在新的线程中完成聚类处理(语句(11)),这个处理过程和算法的其余部分之间是异步执行的,实现了离线的聚类处理.

算法 2 描述了更新密度树的过程.

算法 2. *UpdateDenTree*(*Root*,*Stream*[],*Size*,*k*).

输入:时态密度树根结点 *Root*,数据 *Stream*[],数据数量 *Size*,维数 *k*.

输出:更新后的时态密度树 *Root*.

```
(1)  t=current time;
(2)  for each point in Stream[]{
(3)    C[]=the coordinates of Stream[i];  //数据点 Stream[i]的 k 维坐标
(4)    for (int j=0;j<k;j++){  //处理 0~k–1 维坐标
(5)      if (at level j, no nodes containing C[j]){  //第 j 层中还没有包含 C[j]的结点
(6)        generate a new DimNode or DenNode to index C[j];}}  //生成相应的 DimNode 或 DenNode
(7)    save the old value of temporal density of the density node;  //保存稠密结点中的旧密度值
(8)    update the temporal density of the density node;  //根据命题 1 更新时态密度
(9)    if (the node is non-density) delete the node;}  //根据定义 4(1)删除非稠密结点
```

当新的数据到达时,如果此时还没有对应的稠密结点,则算法 2 最终将包含该数据的稠密结点;如果已经有对应的稠密结点,则算法 2 在命题 1 的指导下(语句(8))更新该稠密结点记录的时态密度.易知,算法 2 的时间复杂度为 $O(vk)$,其中,v 是在一个处理周期内到达的数据点的数量,k 是维数.

算法 3 给出了离线聚类算法.

算法 3. *GetEvolDenCluster*(*Root*,*v*,*M*).

输入:时态密度树根结点 *Root*,数据流平均速率 *v*,单元格数 *M*.

输出:更新后的时态密度树,各个稠密结点所属簇的编号 *ClusterNo* 被标记.

```
(1)  D=Root→sibling;
(2)  while (D!=NULL){
(3)    if (D→clusterNo!=0) continue;
(4)    SearchDenConnSubgraph(D);
(5)    D=D→sibling;}
```

算法 3 对时态密度树中的所有稠密结点所属簇的编号进行标记.命题 4 保证了运行算法 3 以后,所有稠密

结点属于而且只属于一个簇,且所有簇都被找到.算法 3 中语句(4)调用 *SearchDenConnSubgraph* 函数,实现从一个稠密结点出发,标记同一个簇中的所有结点,其过程由算法 4 给出.

算法 4. *SearchDenConnSubgraph*(*D*,*v*,*M*,*k*).

输入:稠密结点 *D*,数据流平均速率 *v*,单元格数 *M*,维数 *k*.

输出:包含结点 *D* 的密度连通分量中的结点的 *ClusterNo* 被标记.

(1) static int *ClusterNo*=1; //未用的聚类编号
(2) *t*=*CurrentTime*(); //取得当前时间
(3) if (*D*→*clusterNo*==0)
(4) *D*→*clusterNo*=*ClusterNo*++;
(5) *C*[][]=the coordinates of all the neighbours of *D*; //*D* 的所有邻接点的坐标
(6) for (int *i*=0;*i*<2×*k*;*i*++){ //最多 2×*k* 个邻接点
(7) *N*=*SearchDenTree*(*C*[*i*]); //查找邻结点
(9) if (*N*→*clusterNo*==0){ //标记簇编号
(10) *N*→*clusterNo*=*D*→*clusterNo*;}}

算法 4 从输入结点 *D* 出发查找 *D* 的密度相邻结点(语句(7)),并做相同的簇标记(语句(10)).

算法 3 和算法 4 是 TDCA 算法的核心算法,它们的时间复杂度决定着 TDCA 的运行效率.命题 5 和命题 6 给出了理论上的分析.

命题 5. 设维数为常数 *k*,单元格数最多为 *m*,常量 $c=2k^2$,则算法 4 的时间复杂度为 $O(c\times\lg(m))$.

证明:算法 4 的基本操作是查找密度树(语句(7)),则最坏情况下共要执行 2*k* 次,而密度树中的结点(包括维结点和稠密结点)不超过 m^k 个,所以执行语句(7)的时间复杂度为 $O(k\times\lg(m))$,因此算法 4 的时间复杂度为 $O(2k\times k\lg(m))=O(c\times\lg(m))$,常量 $c=2k^2$. □

命题 6. 设常量 $c=2k^2$,单元格数最多为 *m*,则算法 3 的时间复杂度为 $O(c\times m\times\lg(m))$.

证明:算法 3 的基本操作为调用算法 4(语句(4)),若密度树中的稠密结点最多 *m* 个,则由命题 5 可知,算法 3 的时间复杂度为 $O(c\times m\times\lg(m))$,常量 $c=2k^2$. □

如果单元格没有新的数据到达,则其时态密度将逐步衰减,对聚类的贡献也越来越小.当稠密结点衰减成为非稠密结点时,应当删除它,以尽可能地节约存储空间(算法 2 语句(9)).这里的关键是必须保证删除是安全的,即当以后该单元格有新的数据到达时,被删除时的单元格密度对以后的聚类也是没有影响的.下面给出的命题 7 和定理 1 保证了这一点.首先给出累积时态密度的定义.

定义 8(累积时态密度). 如果单元格 *c* 每次被删除时都保存了当时的时态密度值和数据点,则这些时态密度值在时刻 *t* 的累积之和称为单元格 *c* 在时刻 *t* 的累积时态密度,记为 $D_\alpha(c,t)$.

命题 7. 设 $D_\alpha(c,t)$是单元格 *c* 的累积时态密度,并设历史上 *c* 被删除了 *m* 次,每次被删除的时刻为 t_i,$1\le i\le m$,则有 $D_\alpha(c,t)=\sum_{i=1}^{m}D(c,t_i)2^{-\lambda(t-t_i)}+D(c,t)$.

证明:设 *c* 被删除的历史为$(0,t_1),(t_1+1,t_2),\ldots,(t_{m-1}+1,t_m)$.由于每次删除后,新建立的单元格的密度从 0 开始计算,所以 $D(c,t_1),D(c,t_2),\ldots,D(c,t_m)$与 $D(c,t)$等彼此之间互不影响.因此,根据定义 1 和定理 1,在当前时刻 *t*,各部分密度已经衰减为 $2^{-\lambda(t-t_i)}D(c,t_i)$,所以 $D_\alpha(c,t)=\sum_{i=1}^{m}D(c,t_i)2^{-\lambda(t-t_i)}+D(c,t)$. □

定理 1. 设 $D_\alpha(c,t)$是单元格 *c* 的累积时态密度,并设 *c* 最近一次被删除的时间是 t_d,最近有数据到达的时间是 t_u,当前时间是 t,$t_d+1<t_u<t$,$\sigma(c,t,t_u)$是当前的阈值.如果 $D(c,t)<\sigma(c,t,t_u)$,则 $D_\alpha(c,t)<\sigma(c,t,0)$.

证明:设单元格 *c* 被删除的历史为$(0,t_1),(t_1+1,t_2),\ldots,(t_{d-1}+1,t_d)$,由命题 7 可知,

$$D_a(c,t)=\sum_{i=1}^{d}D(c,t_i)2^{-\lambda(t-t_i)}+D(c,t) \tag{2}$$

又由算法 2 语句(9)可知,$D(c,t_i)<\sigma(c,t_i,t_{i-1}+1)$.所以由式(2)和已知可得:

$$D_a(c,t)<\sum_{i=1}^{m}\sigma(c,t_i,t_{i-1}+1)2^{-\lambda(t-t_i)}+\sigma(c,t,t_u).$$

又已知 $t_d+1<t_u$,由命题 3 可知,$\sigma(c,t,t_u)<\sigma(c,t,t_d+1)$,所以有

$$\begin{aligned}D_a(c,t) &< \sum_{i=1}^{m}\sigma(c,t_i,t_{i-1}+1)2^{-\lambda(t-t_i)}+\sigma(c,t,t_d+1)\\ &=\sum_{i=1}^{m-1}\sigma(c,t_i,t_{i-1}+1)2^{-\lambda(t-t_i)}+\sigma(c,t,t_{d-1}+1)\\ &=\sum_{i=1}^{m-2}\sigma(c,t_i,t_{i-1}+1)2^{-\lambda(t-t_i)}+\sigma(c,t,t_{d-2}+1)\\ &\cdots\\ &=\sigma(c,t,0).\end{aligned}$$

结论成立. □

定理 1 表明,即使一个单元格 c 在其时态密度小于阈值时未被删除,它也不可能再次成为稠密结点.因此,算法 2 对非稠密结点的删除是安全的.

5 实验和分析

实验平台:Intel Core 2 双核 CPU,主频 2.0GHz,2 级缓存 4MByte,内存 2GByte,Linux 操作系统,GCC 编译器.实验程序用 C 语言编制.

实验在仿真数据集上进行.在仿真环境下随机产生 1000KB 具有稀疏、倾斜特性的数据.当不考虑数据随时间的衰减时,其数据分布如图 3 所示.实验包括 4 个部分,分别考察了衰减速度不同时 TDCA 算法的聚类结果,验证了 TDCA 算法能否有效捕捉聚类的时态特征,比较了 TDCA 算法和其他典型的数据流聚类算法的功能,最后测试了 TDCA 算法在数据集规模变化时的性能.

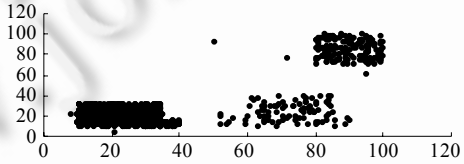
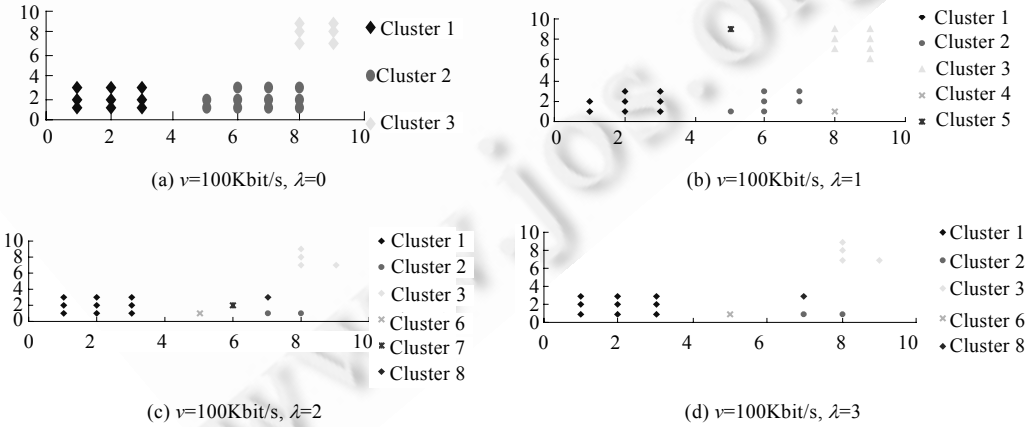


Fig.3 Distribution of test data without decay

图 3 不考虑衰减时的测试数据的分布

5.1 实验1:不同衰减速度时TDCA算法的有效性

图 4 是算法在速率 v=100Kbit/s,衰减参数λ=3,2,1,0 时的聚类结果,图中的一个点代表图 3 中按 10 等分的一个单元格.



(a) v=100Kbit/s, λ=0 (b) v=100Kbit/s, λ=1

(c) v=100Kbit/s, λ=2 (d) v=100Kbit/s, λ=3

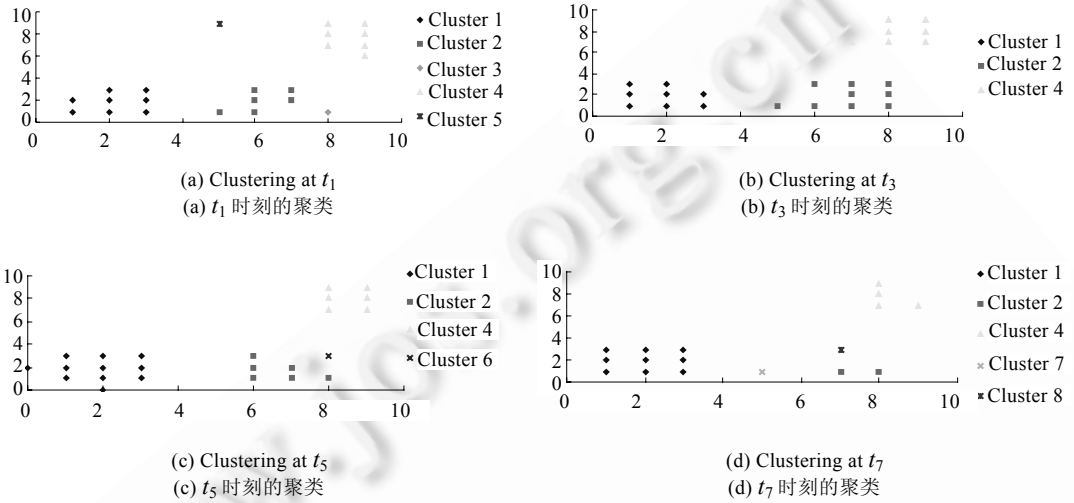
Fig.4 Result clusters with varying speed of decay

图 4 不同衰减速度时的聚类结果

图 4 显示了不同的衰减速度对聚类结果的影响.当衰减速度λ为 0 时,数据不衰减,所有的数据都参与最终的聚类.当衰减速度逐步增大时,较稀疏的聚类中时态密度衰减的效果更加明显,聚类 2 和聚类 3 中参与聚类的单元格数逐渐减少.当λ等于 2 时,聚类发生了分裂,出现了聚类 4 和聚类 5.当λ达到 3 时,在λ等于 2 时出现过的聚类 7 因为没有新的数据到达而衰减后消失.所有衰减速度下,聚类 1 中的单元格数量保持不变,这是因为新数据引起的密度增加,超过了密度衰减的速度.

5.2 实验2:聚类的时态特征

图 5 显示了当数据速率为 100Kbit/s,衰减速度为 3 时,t_1,t_3,t_5,t_7时刻的聚类结果.



(a) Clustering at t_1
(a) t_1 时刻的聚类

(b) Clustering at t_3
(b) t_3 时刻的聚类

(c) Clustering at t_5
(c) t_5 时刻的聚类

(d) Clustering at t_7
(d) t_7 时刻的聚类

Fig.5 Clustering results of TDCA on different time

图 5 TDCA 算法在不同时刻的聚类结果

图 5 表明,不同时刻的聚类结构发生了变化.这些变化包括:

(1) 聚类的消失和出现.例如,聚类 3 在 t_3 时刻消失;聚类 6 在 t_5 时刻出现.

(2) 聚类的合并与分裂.例如,在 t_3 时刻聚类 3 和聚类 2 合并;在 t_7 时刻聚类 2 分裂出聚类 7 和聚类 8.

(3) 聚类内部单元格的增减.例如,在 t_1 时刻单元格 c(9,6)出现在聚类 3 中,在 t_3 时刻消失;单元格 c(7,7)在 t_3 时刻出现在聚类 3 中.

5.3 实验3:功能比较

实验 3 在 TDCA 算法和 ACluStream 算法[14]之间进行功能比较.图 6 显示了 ACluStream 算法的最终聚类结果.

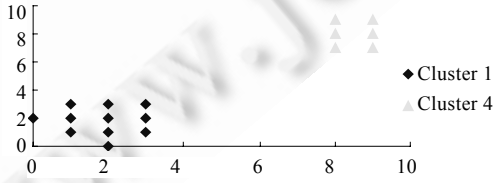


Fig.6 Clustering results of algorithm AcluStream

图 6 ACluStream 算法的聚类结果

将图 6 和图 5(d)比较之后可以发现,AcluStream 算法无法识别聚类 2、聚类 7 和聚类 8.这是因为 ACluStream

算法采用了全局的、固定的密度阈值,既无法区分同一时刻不同局部密度不同的聚类,又无法捕捉因为不同时刻同一局部的密度变化而造成的聚类变化.当密度阈值较大时,AcluStream 算法无法识别局部密度较为稀疏的聚类 2;同时由于密度阈值是固定的,不随时间而改变,因此无法识别在历史上曾经出现过、而后又因为衰减而逐渐消失的聚类 7 和聚类 8.与之相反,TDCA 算法因为所使用的密度阈值是一个不同区域历史密度和时间的函数,具有自适应的特性,因此能够识别在不同时间出现的较为稀疏的聚类(聚类 2、聚类 7、聚类 8).

5.4 实验4:性能测试

图 7 显示了 TDCA 算法处理不同大小的数据集的运行时间.由图 7 可知,TDCA 的运行时间和数据集规模呈现近似的线性关系,当数据规模增大时,TDCA 表现出了良好的规模可伸缩性.其主要原因是 TDCA 算法采用增量的方式计算单元格的密度.根据命题 1 的结论,TDCA 算法在更新单元格时态密度时,可以按照增量的方式进行计算,避免每次访问单元格中所有点,显著地提高了计算效率.根据命题 5 和命题 6 的结论,TDCA 的时间复杂度为 $O(c\times m\times \lg(m))$,因此,当数据规模增大时,TDCA 的性能保持了较好的适应性.

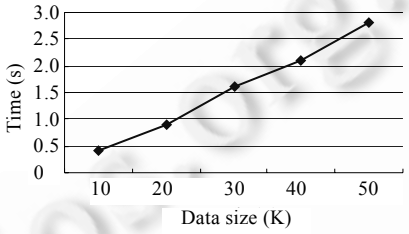


Fig.7 Performance of TDCA

图 7 TDCA 的性能

6 结论和展望

针对数据流聚类的难点问题,本文提出了一种新的数据流聚类算法——TDCA 算法.与传统的数据流聚类算法相比,TDCA 算法具有以下特点:(1) 具有坚实的数学基础;(2) 不需要关于簇个数的先验知识;(3) 能够同时处理数据流的时态特征和倾斜分布特征;(4) 具有较高的时间和空间效率.

实验结果表明,由于考虑了数据空间不同局部的时态密度差异,TDCA 算法不仅能够捕捉数据流时态特征,还能有效地处理数据倾斜分布的情况,同时识别稀疏程度不同的聚类.此外,理论和实验证明,TDCA 算法具有较高的性能,时间复杂度不超过 $O(c\times m\times \lg(m))$.

目前的算法只能处理欧氏空间的单数据流.在实际应用中,分布式环境下多数据流相互影响,相互作用,情况更为复杂;另一方面,越来越多的数据流存在于非欧氏空间.如何有效地将 TDCA 算法应用于非欧空间数据流和分布式环境下的多数据流中的聚类分析,则是我们进一步的研究方向.

References:

[1] Golab L, TamerOzsu M. Issues in data stream management. ACM SIGMOD Record, 2003,32(2):5−14.

[2] Muthukrishnan S. Data streams: Algorithms and Applications. Boston: Now Publisher Inc., 2005.

[3] Henzinger MR, Raghavan P, Rajagopalan S. Computing on data streams. Technical Report, SRC Technical Note l998-011, Palo Alto: Digital Systems Research Center, 1998.

[4] Papadimitriou S, Sun J, Faloutsos C. Streaming pattern discovery in multiple time-series. In: Klemens B, Christian SJ, Laura MH, Martin LK, Per-Ake L, Beng CO, eds. Proc. of the 3lst Int'l Conf. on Very Large Data Bases. New York: VLDB Endowment, 2005. 697−708.

[5] Babcock B, Babu S, Datar M, Motwani R, Widom J. Models and issues in data stream systems. In: Proc. of the 21st ACM Symp. on Principles of Database Systems (PODS 2002). New York: ACM, 2002. 1−16.

[6] Jin CQ, Qian WN, Zhou AY. Analysis and management of streaming data: A survey. Journal of Software, 2004,15(8):1172−1181 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1172.htm

[7] O'Callaghan L, Mishra N, Meyerson A, Guha S, Motwani R. Streaming-Data algorithms for high-quality clustering. In: Proc. of the 18th Int'l Conf. on Data Engineering. Washington: IEEE Computer Society, 2002. 685−694.

[8] Guha S, Mishra N, Motwani R, O'Callaghan L. Clustering data streams: Theory and Practice. IEEE Trans. on Knowledge and Data Engineering, 2003,15(3):515−528.

[9] Aggarwal CC, Han J, Wang J, Yu PS. A framework for clustering evolving data streams. In: Johann CF, Peter CL, Serge A, Michael JC, Patricia GS, Andreas H, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. New York: VLDB Endowment, 2003. 81−92.

[10] Ester M, Kriegel HP, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the ACM KDD'96. New York: ACM, 1996. 226−231.

[11] Ankerst M, Breunig M, Kriegel HP, Sander J. Optics: Ordering points to identify the clustering structure. In: Proc. of the ACM SIGMOD'99. New York: ACM, 1999. 49−60.

[12] Hinneburg A, Keim DA. An efficient approach to clustering in large multimedia databases with noise. In: Proc. of the ACM KDD'98. New York: ACM, 1999. 58−65.

[13] Duan L, Xu LD, Guo F, Lee J, Yan BP. A local-density based spatial clustering algorithm with noise. Information Systems, 2007, 32(7):978−986.

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

[15] Sun HL, Yu G, Bao YB, Zhao FX, Wang DL. CDS-Tree: An effective index for clustering arbitrary shapes in data streams. In: Proc. of the 15th Int'l Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications (RIDE-SDMA 2005). Washington: IEEE Computer Society, 2005. 81−88.

附中文参考文献:

[6] 金澈清,钱卫宁,周傲英.流数据分析与管理综述.软件学报,2004,15(8):1172−1181. http://www.jos.org.cn/1000-9825/15/1172.htm

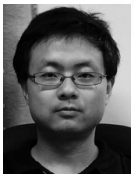
[14] 朱蔚恒,印鉴,谢益煌.基于数据流的任意形状聚类算法.软件学报,2006,17(3):379−387. http://www.jos.org.cn/1000-9825/17/379.htm

杨宁(1974－),男,四川成都人,博士生,主要研究领域为机器学习,数据挖掘.

陈瑜(1974－),男,博士,讲师,主要研究领域为数据库系统,数据挖掘.

唐常杰(1946－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据库系统,数据挖掘.

郑皎凌(1981－),女,博士生,讲师,CCF 学生会员,主要研究领域为数据库系统,数据挖掘.



王悦(1981－),男,博士生,主要研究领域为数据库系统,数据挖掘.