软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2019,30(3):667−683 [doi: 10.13328/j.cnki.jos.005693]

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

基于网格耦合的数据流聚类*

张东月[1], 周丽华[1], 吴湘云[2], 赵丽红[1]

[1](云南大学 信息学院,云南 昆明 650000)

[2](丽江师范高等专科学校,云南 丽江 674199)

通讯作者: 周丽华, E-mail: lhzhou@ynu.edu.cn

摘 要: 随着越来越多的应用程序产生数据流,数据流聚类分析的研究受到了广泛关注.基于网格的聚类通过将数据流映射到网格结构中形成数据概要,进而对概要进行聚类.这种方法通常具有较高的效率,但是每个网格独立处理,没有考虑网格之间的相互影响,因此聚类质量有待提高.在聚类过程中不再独立处理网格,而是考虑了网格之间的耦合关系,提出了一种基于网格耦合的数据流聚类算法.网格的耦合更加准确地表达了数据之间的相关性,从而提高了聚类的质量.在合成和真实数据流上的实验结果表明,所提算法具有较高的聚类质量和效率.

关键词: 数据流;聚类分析;网格耦合;网格结构;聚类质量

中图法分类号: TP311

中文引用格式: 张东月,周丽华,吴湘云,赵丽红.基于网格耦合的数据流聚类.软件学报,2019,30(3):667−683. http://www.jos.org.cn/1000-9825/5693.htm

英文引用格式: Zhang DY, Zhou LH, Wu XY, Zhao LH. Data stream clustering based on grid coupling. Ruan Jian Xue Bao/ Journal of Software, 2019,30(3):667−683 (in Chinese). http://www.jos.org.cn/1000-9825/5693.htm

Data Stream Clustering Based on Grid Coupling

ZHANG Dong-Yue[1], ZHOU Li-Hua[1], WU Xiang-Yun[2], ZHAO Li-Hong[1]

[1](School of Information Science and Engineering, Yunnan University, Kunming 650000, China)

[2](Lijiang Teachers College, Lijiang 674199, China)

Abstract: As more and more applications generate data streams, the research on data stream clustering analysis has received extensive attention. Grid-based clustering maps data streams into grid structures to form data summaries, and then clusters data summaries. This method usually has high efficiency, but each grid is processed independently, and the interaction between the grids is not considered, so the clustering quality needs to be improved. In this study, the coupling relationship between grids is considered rather than processed independently in the clustering process, and an algorithm for clustering data stream based on grid coupling is proposed. The proposed approach improves the quality of clusters as the coupling of the grid more accurately captures the correlation amongst the data. Experimental evaluations on synthetic and real data streams illustrate the superiority of the proposed approach compared with the state-of-the-arts approaches.

Key words: data stream; clustering analysis; grid coupling; grid structure; the quality of cluster

数据流是一种随着时间增加而顺序、快速、大量、连续到达的数据序列.近年来,随着软硬件的发展,大量

* 基金项目: 国家自然科学基金(61762090, 61262069, 61472346, 61662086); 云南省自然科学基金(2016FA026, 2015FB114); 云南省创新研究团队项目(2018HC019); 云南省高等学校科技创新团队项目(IRTSTYN)

Foundation item: National Natural Science Foundation of China (61762090, 61262069, 61472346, 61662086); Natural Science Foundation of Yunnan Province (2016FA026, 2015FB114); Project of Innovative Research Team of Yunnan Province; Program for Innovation Research Team (in Science and Technology) in University of Yunnan Province (IRTSTYN)

本文由智能数据管理与分析技术专刊特约编辑樊文飞教授、王国仁教授、王朝坤副教授推荐.

收稿时间: 2018-07-20; 修改时间: 2018-09-20; 采用时间: 2018-11-01

的数据流不断产生,如金融数据、视频监控数据、传感数据和网络流量数据等.这些数据流大部分是无标签的,所以实时聚类数据流并从中提取有价值的信息,成为了数据挖掘领域的重要问题之一[1−4].然而,由于数据流的连续性、无限性、演变性等特点,要求数据流聚类算法只能在资源约束的条件下单次扫描数据流,并能够随时间的变化追踪簇的形状和位置的变化[5,6].除此之外,提高数据流聚类算法的精度和效率,也是一直存在的重要挑战.

现有的数据流聚类算法大多采用经典的在线/离线框架[7]来处理数据流.在线阶段将到达的数据对象映射到一组支持快速查找的网格结构中,以此汇总数据流并提取数据流的摘要信息.每个网格都类似于一组数据对象的集合,是在单次扫描数据流的环境下创建的.离线阶段,根据用户或应用程序的需要,使用传统的(或改进的)聚类算法将网格结构合并,生成最终聚类[8−10].采用在线/离线框架的数据流聚类算法通过网格提取数据流的概要信息,能够较快地处理数据流并支持实时聚类,但是这些聚类算法在将数据对象映射到网格并增量更新网格时,通常假设网格之间彼此独立,忽略了网格之间的相互影响,使得提取的数据流概要信息不够精确,从而影响了聚类精度.

为了提高聚类精度,MR-Stream[11]映射数据时使用了尺寸更加精细的网格;DBSTREAM[12]引入共享密度来检测微簇内数据的分布状态,避免了两个微簇相交区域密度较低却仍然将他们聚为一类的现象,提高了聚类质量.但是网格的精细化既增大了内存占用又降低了算法效率;而共享密度不仅需要计算微簇之间关系,还要计算数据对象与微簇的关系,这样才能捕捉到两个微簇相交区域的数据量,同样降低了算法效率.本文提出了一种基于网格耦合的数据流聚类算法,称为 GCStream.首先,基于网格内的数据对象定义网格权重,并在聚类过程中不再独立处理网格,而是基于网格内数据对象的分布状态考虑网格之间权重的相互影响,即,一个网格权重的变化会使相邻网格的权重增加或减小,网格的耦合更加准确地表达了数据之间的相关性,从而提高聚类精度.其次,基于网格内数据的分布,通过搜索密度相连的网格完成聚类,并根据高密度网格的变化捕捉簇的演化.

本文的主要贡献包括:

(1) 聚类过程中不再独立处理网格,而是考虑了网格之间的耦合关系,从而更加准确地表达了数据之间的相关性;

(2) 提出了一种基于网格耦合的数据流聚类算法,该算法不需要指定簇的数目,只需通过搜索密度相连的网格完成聚类,并根据高密度网格的变化捕捉簇的演化;

(3) 在人工和真实数据流上进行了实验验证,通过实验对比了所提算法的性能.

本文第 1 节介绍数据流聚类的典型算法.第 2 节介绍网格耦合的相关概念和定义.第 3 节给出 GCStream 的算法流程并分析算法的时间复杂度.第 4 节给出在合成和真实数据集上的对比实验.最后,总结论文工作.

1 相关工作

现有的数据流聚类算法可以分为基于划分的、基于层次的以及基于密度的方法[13].前两种方法主要基于对象之间的距离进行聚类,比如 STREAM[14]算法采用类似批处理的方式将数据流分块,将每一批数据分为 k 个簇,通过保留 k 个簇的中心点汇总每一批数据.STREAM 算法无法在任意时刻给出当前数据流的聚类结果,并且也没有考虑数据流的演变性.CluStream[7]算法提出了在线/离线两阶段处理框架:在线阶段通过微簇结构以增量方式维护数据流的概要信息,离线阶段基于概要信息和用户输入产生聚类结果,克服了 STREAM 算法不能实时产生聚类结果的问题.但是 CluStream 算法没有体现近期数据与历史数据对聚类结果的不同影响,并且在高维数据流的聚类上表现不佳.HPStream 算法[15]通过投影技术和衰减簇结构对 CluStream 算法进行了改进,能够集成当前数据和历史数据,更好地聚类高维数据流.但是,HPStream 算法仍然不能聚类任意形状的数据流,并对噪声敏感.

基于密度的方法通过查找被低密度区域包围的高密度区域来进行聚类,能够发现任意形状的簇并且可以去除噪声.DenStream 算法[16]、D-Stream 算法[8]、MuDi-Stream 算法[9]、MR-Stream[11]和 DBSTREAM 算法[12]都继承了 CluStream 的在线/离线框架.DenStream 算法的在线阶段通过将数据点分配给离它最近的微簇来进行

微聚类,并且提出了核心微簇、潜在核心微簇和离群微簇的概念来区分正常簇的数据、可能发展为正常簇的数据和噪声数据;离线阶段通过 DBSCAN 算法进行宏聚类,所以它能发现任意形状的簇.D-Stream 算法的在线阶段将数据流映射到相应的网格,并通过网格密度对网格进行分类;离线阶段通过合并相邻网格产生聚类结果.MuDi-Stream 算法的在线阶段计算数据点与网格中心的距离,并将数据点分配给距离其最短的网格;离线阶段通过改进的 DBSCAN 算法生成聚类结果,在多密度数据流上表现较好.MR-Stream 算法使用网格树结构来存储网格,树中每个节点代表一个网格,并且存有其父和子节点的概要信息.MR-Stream 算法的在线阶段将当前数据映射到网格树的相应叶子节点,离线阶段在不同网格树高度上通过合并相邻网格进行聚类.该方法本质上是通过将大的网格细分为多个小网格来提高聚类质量,但是网格细分导致内存占用成倍地增加,降低了聚类效率.DBSTREAM 算法通过共享密度对数据流进行聚类,其在线阶段通过微簇来维护数据流的概要信息,并且通过计算微簇与微簇、数据对象与微簇之间的关系来捕捉两个微簇之间共同拥有的数据,即共享密度;离线阶段在进行宏聚类时不仅考虑微簇之间的距离关系,同时还考虑不同微簇共同拥有的数据量.这种通过引入共享密度来检测微簇内数据的分布状态的方法,避免了两个微簇相交区域密度较低却仍然将它们聚为一类的现象,提高了聚类质量.但是 DBSTREAM 算法需要的计算量较大,降低了聚类效率,这一点在高维数据流上尤为突出.

除了聚类效率,上述算法在新数据到达而更新数据摘要时均独立处理摘要结果,忽略了概要之间的相互影响,从而影响了算法的聚类精度.

2 网格耦合的相关概念和定义

本节首先介绍网格耦合的相关概念,然后给出网格耦合的定义.

2.1 基本概念

假设 $S=s_1\times s_2\times\ldots\times s_d$ 是一个 d 维的空间,将空间 $s_i(i=1,2,\ldots,d)$ 均匀分为 p_i 份,即 $s_i=s_{i,1}\cup s_{i,2}\cup\ldots\cup s_{i,p_i}$,则空间 S 被划分为 $N=\prod_{i=1}^{d}p_i$ 个网格.每一个网格 g 由 $s_{1,j_1}\times s_{2,j_2}\times\ldots\times s_{d,j_d}$ 组成,其中 $j_i=1,2,\ldots,p_i$.

g 可以表示为 $g=(j_1,j_2,\ldots,j_d)$.如果网格 $g_1=(j_1^1,j_2^1,\ldots,j_d^1)$ 和 $g_2=(j_1^2,j_2^2,\ldots,j_d^2)$ 在某一维度 $m(1\leqslant m\leqslant d)$ 上满足 $|j_m^1-j_m^2|=1$ 并且 $j_i^1=j_i^2\ (1\leqslant m\leqslant d\ \&\&\ i\neq m)$,则称网格 g_1 和 g_2 在第 m 维相邻.两个网格只要在某一维度相邻,则称这两个网格为相邻网格.

设输入数据流中的每个数据对象 $X=(x_1,x_2,\ldots,x_d)$ 是 d 维空间中的一个点,如果 $x_i\in s_{i,j_i}$,则可以将数据对象 X 映射到空间 S 的网格 g 中,记为 $g(x)=(j_1,j_2,\ldots,j_d),\ x_i\in s_{i,j_i}$.映射到同一网格中的数据对象距离相近,因此可以对各个网格内的数据进行汇总形成概要,从而聚类过程只针对概要进行处理,降低存储空间和计算成本.

数据流中的数据对象有不同的到达时间,对聚类的贡献也不同.为了区分历史数据和新数据,许多数据流聚类算法[1,7,8]为数据流中每个数据对象分配一个带有衰减因子的权重,使其重要性(新鲜度)随时间推移而下降.

定义 2.1(数据权重). 如果数据对象 X 在 t_p 时刻到达,那么 X 在 t 时刻的权重 $W(X,t)$ 定义为

$$W(X,t)=\lambda^{-a(t-t_p)} \tag{1}$$

其中,$\lambda^{-a}(0<\lambda^{-a}<1)$ 为衰减因子.参数 λ 和 a 控制衰减因子的衰减速度,a 的绝对值越大,衰减速度越快.

由于数据对象被映射到网络中,因此可以基于数据对象的权重定义网格权重.

定义 2.2(网格权重). 设 $E(g,t)$ 表示到时刻 t 为止,映射到网格 g 中的数据对象集合,则网格 g 在时刻 t 的权重定义为网格 g 内所有数据对象的权重之和,即:

$$W(g,t)=\sum_{X\in E(g,t)}W(X,t) \tag{2}$$

网格权重的大小反映了网格内数据对象的数目和数据对象的新鲜程度.网格内数据的变化会引起网格权重的变化,即使没有新的数据对象映射到网格 g,g 的权重 $W(g,t)$ 也会减小,因为 g 中数据对象的权重是随时间逐渐衰减的.Chen 和 Tu 在文献[8]中指出,从 0 时刻开始到任意时刻读取到的数据流总权重不超过 $1/(1-\lambda^{-a})$.假设网格数量为 N,那么每个网格权重为 $1/[N(1-\lambda^{-a})]$.

不同的网格包含的数据量不同,因此,不同的网格具有不同的权重.基于网格权重的大小,Chen 和 Tu[8]还将网格分为稀疏网格和稠密网格:如果 t 时刻网格 g 的权重 $W(g,t)$满足 $W(g,t)\leqslant C_s/[N(1-\lambda^{-a})]$,则称网格 g 为稀疏网格;如果 $W(g,t)\geqslant C_d/[N(1-\lambda^{-a})]$,则称网格 g 为稠密网格,其中,$C_d(C_d>1)$,$(0<C_s<1)$是阈值参数.

2.2 网格耦合

许多基于网格的聚类算法在聚类过程中独立处理网格,忽略了网格之间的相互影响,从而影响了聚类质量.如图 1 所示,图中显示了截止到 t_p 时刻数据流映射到网格 1~网格 6 的数据分布.网格 6 由于数据量太少,可能会被当成噪声网格,使得网格 6 中的数据不被聚类;网格 3 和网格 4 是相邻的非稀疏网格,因此网格 3 和网格 4 中的数据会聚在一个簇中.实际上,网格 6 中的数据应该与网格 4 和网格 5 中的点聚成一簇,网格 3 与网格 4 中的点应该分属不同的簇.针对这个问题,本文提出了一种基于网格耦合的数据流聚类方法.该方法在聚类过程中不再独立处理网格,而是基于网格内数据对象的分布状态考虑网格之间权重的相互影响,即,一个网格权重的变化会使相邻网格的权重增加或减小,比如图 1 中网格 4 权重的增大使得网格 6 权重增加、网格 3 权重减少,从而避免独立处理网格带来的问题.

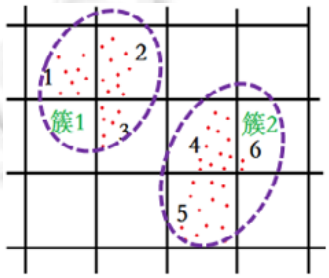


Fig.1 Data distribution within the grid as the time t_p

图 1 截止到 t_p 时刻网格内的数据分布

为了表示网格内数据对象的分布状态,本文将网格内带权数据对象的中心定义为网格质心.由于数据流是动态的,因此网格质心也会随时间而变化.如果数据对象在网格内均匀分布,则网格质心位于网格中心;如果数据对象在网格内分布不均匀,则网格质心不在网格中心.如图 1 中,网格 3 和网格 4 的网格质心就不在网格中心.

定义 2.3(网格质心). 设 $E(g,t)$为截止到 t 时刻映射在网格 g 中的数据对象集合,$W(X,t)$代表数据对象 X 在 t 时刻的权重,则网格 g 在 t 时刻的质心 $C(g,t)$定义为网格 g 内带权数据对象的加权平均,即:

$$C(g,t)=\frac{\sum_{X\in E(g,t)}W(X,t)\cdot X}{\sum_{X\in E(g,t)}W(X,t)} \tag{3}$$

为了快速计算网格质心,定理 2.1 给出了基于 t_p 时刻的网格质心 $C(g,t_p)$来计算 t 时刻网格质心 $C(g,t)$的更新方式.

定理 2.1. 假设网格 g 在 t_p 时刻的质心是 $C(g,t_p)$,t 时刻有新数据对象 X'映射进来,则 t 时刻网格 g 的质心 $C(g,t)$的计算公式为

$$C(g,t)=\frac{C(g,t_p)\cdot kW(g,t_p)\cdot\lambda^{-a(t-t_p)}+X'}{k\lambda^{-a(t-t_p)}\cdot W(g,t_p)+1} \tag{4}$$

其中,k 是一个质心调节参数:如果 $0<k<1$,则表示降低历史数据权重对网格质心的影响,提高网格质心的实时性;如果 $k>1$,则表示增加历史数据权重对网格质心的影响,降低网格质心的实时性.

证明:假设每一时刻数据流中只有一个数据对象到达,网格 g 在 t_p 时刻的质心是 $C(g,t_p)$.根据网格质心定义公式(3)、网格权重公式(2)可以得出 t 时刻质心公式:

$$C(g,t)=\frac{\sum_{X\in E(g,t)}W(X,t)\cdot X}{\sum_{X\in E(g,t)}W(X,t)}=\frac{\sum_{X\in E(g,t)}W(X,t)\cdot X}{W(g,t)}=\frac{\sum_{X\in E(g,t)}\lambda^{-a(t-t_p)}\cdot X}{\lambda^{-a(t-t_p)}W(g,t_p)+1} \tag{5}$$

因为 t 时刻网格权重可以根据 t_p 时刻网格权重迭代得出,所以通过公式(5)可推得公式(6):

$$C(g,t)=\frac{\lambda^{-a(t-t_p)}W(X,t_p)\cdot X+X'\cdot 1}{\lambda^{-a(t-t_p)}W(g,t_p)+1} \tag{6}$$

另一方面,根据将 t_p 时刻的质心公式可得到公式(7):

$$C(g,t_p)\cdot W(X,t_p)=\sum_{X\in E(g,t_p)}W(X,t_p)\cdot X \tag{7}$$

将公式(7)带入公式(6),可得到网格质心迭代公式(8):

$$C(g,t)=\frac{C(g,t_p)\cdot W(g,t_p)\cdot\lambda^{-a(t-t_p)}+X'}{\lambda^{-a(t-t_p)}\cdot W(g,t_p)+1} \tag{8}$$

网格质心表示网格内数据的分布状态,为了度量两个网格内数据分布的距离,本文定义了网络间的质心距离. □

定义 2.4(网格质心距离). 设 $C(g_i,t)=\{c_{i1},c_{i2},\ldots,c_{id}\}$ 和 $C(g_j,t)=\{c_{j1},c_{j2},\ldots,c_{jd}\}$ 分别是两个相邻网格 g_i 和 g_j 的质心,则在 t 时刻,这两个相邻网格质心间的距离 $dis_C(g_i,g_j)$ 定义为

$$dis_C(g_i,g_j)=\sqrt{\sum_{n=1}^{d}(c_{in}-c_{jn})^2} \tag{9}$$

为了减少计算量,本文只考虑相邻网格间的耦合,因此,两个不相邻网格间的质心距离定义为无穷大.

网格之间的相互影响与网格质心之间的距离有关:距离越近,影响越大;反之越小.实际上,质心距离越近的网格,网格内的数据点属于同一个簇的可能性越大;而距离较远的网格内的数据点属于不同簇的可能性大.属于同簇的网格,其权重的变化趋势应该相同;属于异簇的网格,其权重的变化趋势应该相反.为了区分网格之间的不同影响,本文定义了正影响和负影响的概念.设 $Dislen$ 为影响区域阈值,如果 $dis_C(g_i,g_j)\leqslant Dislen$,则网格 g_i 对网格 g_j 产生正影响;反之产生负影响.正影响表明 g_i 权重增加,g_j 权重随之增大;负影响则表示 g_i 权重增加,g_j 权重随之减小.影响系数定量度量了网格间的影响强度.

定义 2.5(影响系数(*Ideg*)). 网格 g_i 和 g_j 之间的影响系数定义如下:

$$Ideg(g_i,g_j)=\begin{cases}1-\dfrac{dis_C(g_i,g_j)}{Dislen}, & (dis_C(g_i,g_j))\leqslant Dislen\\ \dfrac{Dislen-dis_C(g_i,g_j)}{MaxCdis-Dislen}, & (dis_C(g_i,g_j))>Dislen\end{cases} \tag{10}$$

其中,$MaxCdis$ 为相邻网格质心距离的最大值(体对角上的两点间距离).假设网格空间为 d 维,网格边长为 len,则 $MaxCdis$ 定义为

$$MaxCdis\approx\sqrt{(2len)^2+(Dim-1)\times len^2}=\sqrt{(Dim+3)}len \tag{11}$$

如果网格 g_i 对网格 g_j 产生正影响,则 $Ideg(g_i,g_j)>0$;反之,$Ideg(g_i,g_j)<0$.如图 2 中,两个星形符号分别表示网格 3 和网格 6 的质心.假设时刻 t 有一个数据映射到网格 4 中,则以网格 4 的质心为圆心、$Dislen$ 为半径,生成一个实线圆,圆内区域是网格 4 的正影响区域,圆外是网格 4 的负影响区域.由图可见:网格 4 权重的变化对网格 6 产生正影响,对网格 3 产生负影响,使网格 6 的权重有所增加,网格 3 的权重有所减少.网格间的耦合增大了网格 4 和网格 6 中数据聚到同一个簇的可能,减小了网格 4 和网格 3 中数据聚到同一个簇的可能,克服了独立处理网格带来的问题.

在基于网格的聚类算法中,每个簇都是由一组相连的网格组成,每个簇被稀疏网格包围.通常,处于簇中心的网格,其权重与相邻网格的权重之和较大;而位于簇边缘的网格,其权重与相邻网格的权重之和较小.为了区分这两种不同的状态,本文定义了核心网格的概念,并将簇定义为密度相连的网格内的数据集合.

定义 2.6(核心网格). 设 $L(g,t)$ 是网格 g 在 $Dislen$ 影响范围内的网格集合,如果 $L(g,t)$ 内所有网格的权重之和

大于阈值,即 $LW(g,t)=\sum_{g'\in L(g,t)}W(g',t)>\varepsilon$,则称网格 g 为核心网格.所有核心网格的集合表示为 LD.由于核心网格很可能为簇中心网格,其权重与相邻网格权重之和大于簇边缘上的稀疏网格及其相邻网格权重之和,所以阈值 $\varepsilon\geqslant C_d/[N(1-\lambda^{-a})]$.

定义 2.7(密度相连). 设网格 g_i 是一个核心网格,如果 g_j 是 LD 中离网格 g_i 质心距离最近的网格,则称网格 g_j 与 g_i 密度相连,网格 g_j 中的数据点被分配到 g_i 中数据点所属的簇.核心网格 g_i 及其密度相连的网格内数据对象构成的集合称为簇,记为 C_{g_i}.

设网格 g_i 和 g_j 密度相连,g_k 和 g_l 密度相连,如果 $dis_C(g_p,g_q)<Dislen(p=i,j;q=k,l)$,则 g_i,g_j,g_k 和 g_l 密度相连,即 g_i 和 g_j 构成的簇与 g_k 和 g_l 构成的簇合并.

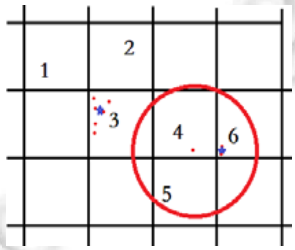


Fig.2 Effect of mapping data objects in grid 4 on grid 3 and grid 6

图 2 网格 4 中映射数据对象对网格 3 和网格 6 的影响

3 GCStream 算法

本文所提的 GCStream 算法也是基于在线/离线框架,如图 3 所示.

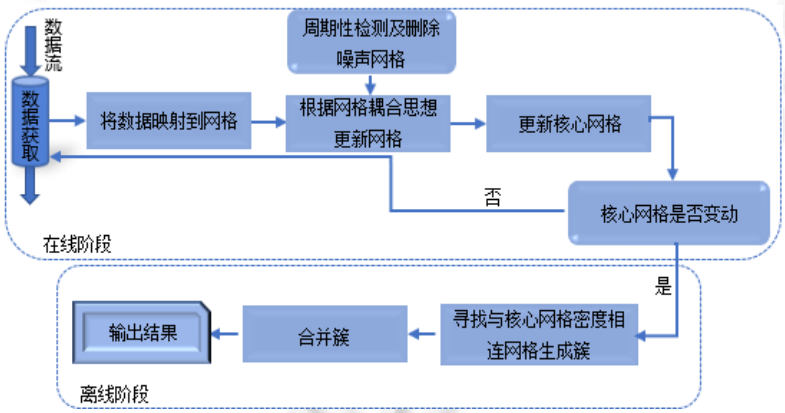


Fig.3 GCStream algorithm flow chart

图 3 GCStream 算法流程图

在线阶段创建网格并将数据流中到达的数据对象映射到相应网格中,然后根据新到达的数据对象更新核心网格及网格的权重、质心等,并周期性检测及删除噪声网格.离线阶段主要基于更新的核心网格和网格的质心寻找密度相连的网格,从而完成聚类,追踪簇的变化.每个步骤详细介绍如下.

- 在线阶段

(1) 将数据映射到网格.

首先初始化一个红黑树用以存储网格列表,而每个网格由一个多元组($key,W,cvec,status,clusterid,t_g,N_g keys$)组成.其中,$key$ 是由网格的位置信息生成的哈希码,W 为网格的权重,$cvec$ 为网格的质心向量,$status$ 代表网格的

稠密状态,*clusterid* 为该网格所属的簇号,t_g 记录的为该网格上次的更新的时刻,N_g*keys* 为该网格的邻居列表.当数据对象到来时,根据该数据对象的属性向量为其寻找对应的网格进行映射.如果该网格在网格列表中不存在,则创建一个新的网格单元.

(2) 根据网格耦合思想更新网格.

当数据对象映射到网格之后,需要更新该网格的元组.值得注意的是,网格之间是相互影响的,所以本文在更新当前网格时,还通过网格质心距离捕捉该网格与周围网格的关系,以此来确定周围网格的更新.

(3) 更新核心网格.

当网格权重发生变化时,需要判断该网格是否满足核心网格条件.如果满足,则将该网格替换进核心网格列表中.

(4) 周期性检测及删除噪声网格.

噪声网格为一些由噪声生成的网格或一些由簇衰退形成的零星网格.在数据流不断到达的过程中,这些网格不断累积,会造成内存空间的浪费,所以需要定期地检测删除这些噪声网格.

本文根据噪声网格比较稀疏并不可能变为稠密网格的特性,将经过 t_u 时间段还没由稀疏转为稠密的网格定义为噪声网格.定理 3.1 给出了一个稀疏网格转换为稠密网格所需时间的计算公式.

定理 3.1. 设网格 g 是一个稀疏网格,t_u 是 g 转换为稠密网格所需的时间,则:

$$t_u=\frac{1}{a}\log_\lambda\frac{N}{N-C_d+C_s} \tag{12}$$

证明:设网格 g 在 t_1 时刻为稀疏网格,则:

$$W(g,t_1)\leqslant\frac{C_s}{N(1-\lambda^{-a})} \tag{13}$$

设网格 g 在 $t_2(t_2>t_1)$时刻转为稠密网格,则:

$$W(g,t_1)\geqslant\frac{C_d}{N(1-\lambda^{-a})} \tag{14}$$

如果要求网格 g 能在最短的时间内由稀疏网格转为稠密网格,则需要时间段 $t_u(t_u=t_2-t_1)$内到达的数据对象均映射在网格 g 内,因此,

$$W(g,t_2)\leqslant\sum_{X\in E(g,t_1)}W(X,t_2)+\sum_{X\in E(g,t_2)-E(g,t_1)}W(X,t_2) \tag{15}$$

其中,$X\in(E(g,t_2)-E(g,t_1))$表示时间段 t_u 映射到网格 g 的数据对象集合.

由数据权重公式可得时间段 t_u 映射到网格 g 的数据权重为 $\{\lambda^{-a(t_2-t_1-1)},\lambda^{-a(t_2-t_1-2)},...,1\}$.可以看到,该组数据权重满足等比公式,所以不等式(15)可变形为

$$W(g,t_2)\leqslant\lambda^{-at_u}W(g,t_1)+\frac{1-\lambda^{-at_u}}{1-\lambda^{-a}}\leqslant\frac{C_s}{N(1-\lambda^{-a})}\cdot\lambda^{-at_u}+\frac{1-\lambda^{-at_u}}{1-\lambda^{-a}} \tag{16}$$

联立不等式(14)和不等式(16)可以得到:

$$\lambda^{-at_u}\leqslant1-\frac{C_d-C_s}{N} \tag{17}$$

然后,根据公式(17)可以得到:$t_u\geqslant\frac{1}{a}\log_\lambda\frac{N}{N-C_d+C_s}$. □

(5) 检测核心网格是否变动.

随着时间的推移,网格中历史数据的权重逐渐衰减,新映射进来的数据具有较大权重从而导致网格权重发生变化,进而引起网格类型发生变化:稠密网格与稀疏网格互换、核心网格与非核心网格互换.这些变化使得数据流中的簇也是不断变化的,有的簇会随着时间的流逝慢慢消失,有的簇会随着新数据点的映射而慢慢扩大.所以,本文根据核心网格是否变动来调用离线组件.

在线阶段的 5 个步骤主要用于映射数据流以及收集数据流的概要信息.值得注意的是:在高维空间中数据

是比较稀疏的,这有可能划分出许多空网格或数据对象数较少的网格.针对这个问题,本文在线阶段在生成网格时,根据当前到达数据样本的属性向量动态生成网格.即当数据流映射进来时,查找网格列表中是否有与其对应的网格:如果有,则将该数据对象映射到该网格并更新该网格的元组;否则,为该数据对象创建一个新网格单元.除此之外,在线阶段的周期性检测及删除噪声网格等步骤则能定期删除一些数据对象数较小的网格.通过以上两个策略,使得 GCStream 算法能够不生成空网格以及减少稀疏网格的数量,既降低了内存占用,也提高了算法的效率.

- 离线阶段

(1) 寻找与核心网格密度相连的网格生成簇.该阶段通过为每个核心网格寻找与其密度相连的网格来将网格进行划分,形成以核心网格为中心的簇;

(2) 合并簇.上述生成的以核心网格为中心的簇可能存在两个网格数据分布接近,使得两个簇相连,所以进一步判断是否存在能够合并的簇是有必要的.

基于核心网格的离线聚类时间复杂度分析.假设某一时刻网格列表中的总网格数为 n,核心网格集合 LD 的大小为 N_{cg}.首先执行算法 2 的第 3 步~第 5 步,生成簇.该过程将剩余网格分配给核心网格,所需要的时间复杂度为 $O(0.5\times N_{cg}(n-N_{cg}))$;然后执行算法 2 的第 6 步~第 10 步,合并簇.在合并簇的时候,需要遍历每个网格的邻居.假设每个网格有 N_{ng} 个邻居网格,则该阶段时间复杂度为 $O(N_{cg}\times n)$.所以基于核心网格的离线聚类算法的时间复杂度为 $O(N_{cg}(n-N_{cg})+N_{ng}\times n)$.

算法 1. GCStream 的在线阶段.

输入: $\varepsilon=\dfrac{C_d}{N(1-\lambda^{-a})}, Disnlen, k, N_{cg}$ (核心网格集合 LD 的大小);

输出:网格列表.

步骤:

1. t_c=0
2. 初始化网格列表;
3. **while** (数据流没有结束)
4. 从数据流中读取数据对象 $X=(x_1,x_2,\ldots,x_d)$;
5. 确定数据对象 X 所属的网格 g;
6. **if** (网格 g 不存在于网格列表)
7. 创建网格 g 并将其插入到网格列表;
8. **else**
9. 将数据对象 X 映射到网格 g;
10. *UpdateGrid*($W(g,t_c)$,$C(g,t_c)$,t_p);
11. *UpdateAffectedGrid*($W(g',t_c)$);
12. $LW(g,t_c)$=*CalculateLocalWeight*(g);
13. *UpdateCoreGride*($LW(g,t_c)$);
14. **if** t_c mod t_p==0
15. 检测及删除噪声网格;
16. $t_c=t_c+1$;
17. **end while**

算法 2. GCStream 的离线阶段.

输入:网格列表信息;

输出:聚类结果.

步骤:

1. 核心网格集合 *LD* 发生改变;
2. 初始化簇集合;
3. **for** (核心网格集合中的所有核心网格 g')
4. 在网格列表中寻找与 g 密度相连的网格,生成簇;
5. **end for**;
6. **for** (所有簇);
7. **if** 存在两个簇密度相连;
8. 合并这两个簇;
9. **end if**
10. **end for**

4 实验评估

本节对 GCStream 算法适应和捕捉数据流演变的能力、去除噪声数据的能力、聚类质量以及聚类效率进行了实验评估.

4.1 实验准备

本文实验的操作系统为 64 位 Windows 7 旗舰版,硬件环境为 Intel(R) Core(TM) i3-3240(3.40GHz),RAM 为 4GB.

- 测试数据集

本文实验总共用到 5 个数据集:两个人工数据集(MTD 和 MOAD)和 3 个 UCI 真实数据集(KDDCUP99[17], CoverType[18]和 PAMAP2[19]).其中,人工数据集 MTD 使用 MATLAB 生成,包含两个凸型簇和两个非凸型簇并带有 10%均匀分布的噪声.该数据集用以测试 GCStream 算法适应和捕捉簇演变的能力以及去除噪声数据的能力.MOAD 使用 MOA(massive online analysis)工具生成[20,21],该工具是一个处理演变数据流的框架,广泛用于数据流挖掘工作.MOAD 数据集由 12 072 个数据对象组成,分属 10 个簇,每个数据对象包含 1 000 个属性,用以测试各算法在不同维度上的效率.KDDCUP99 数据集是麻省理工学院林肯实验室收集的网络入侵检测数据集,包含 494 020 条 TCP 连接记录,分属于 23 种不同的网络连接类型.CoverType 数据集是美国森林服务信息系统提供的数据集,包含 581 012 条记录,每条记录由一块 30×30 平方英尺上的 54 个地理数据组成.PAMAP2 数据集是 Reiss 和 Strickere 收集的身体活动监测数据,由 9 名佩戴 3 个惯性测量单位和心率监测仪的受试者进行 18 项不同的身体活动(如步行、骑自行车、踢足球等)产生的数据组成.本文通过将上述数据集中数据的输入顺序作为数据流的传输顺序,把所有数据集转为流.每个数据集的大小、维度、簇数以及簇之间的最小距离见表 1.

Table 1 Dataset feature summary

表 1 数据集特征汇总

Data set	Instances	Dim	Clusters	MinDIS
MTD	115 240	2	4	70
MOAD	12 072	1 000	10	12.4
KDDCUP99	494 021	34	23	100
CoverType	581 012	54	7	250
PAMAP2	447 000	51	13	5

- 对比算法

本文使用 D-Stream[8],DenStream[16],DBSTREAM[12],GCStream-UC 作为本文的对比算法.其中,GCStream-UC 为在线阶段不考虑网格耦合的 GCStream 算法,即 GCStream-UC 算法只是更新映射了数据对象的网格的权重,相邻网格的权重不受新映射了数据对象的网格权重变化的影响.

- 聚类质量评估方法

本文中采用的聚类质量评价方法为 Purity 和 CMM[22].Purity 定义如下:

$$purity=\frac{\sum_{i=1}^{K}\frac{|C_i^d|}{|C_i|}}{K}\times 100\% \tag{18}$$

其中,K 代表簇的数量,C_i 代表簇 i 中数据对象总数,C_i^d 代表簇 i 中被正确划分的数据对象数目.*Purity* 度量了各个簇中正确聚类的对象比例.

CMM(clustering mapping measure)是一种考虑了数据流中数据对象的权重(新鲜度)并可以反映簇生成、移动、分裂过程固有错误(比如簇的移动导致部分数据对象丢失;簇的合并和分裂产生重叠的簇,导致一些数据对象被错误划分)的评价指标.此外,CMM 还能对数据流中的噪声情况进行度量.CMM 定义如下:

$$CMM(C,CL)=1-\frac{\sum_{o\in F}W(o)\cdot pen(o,C)}{\sum_{o\in F}W(o)\cdot con(o,Cl(o))} \tag{19}$$

其中,$Cl=\{Cl_1,\ldots,Cl_l\}$ 是真实簇集合,$C=\{C_1,\ldots,C_k,C_\phi\}$ 是聚类结果,$W(o)$是数据对象 o 的权重,$pen(o,C)$是聚类过程中对遗漏数据对象、错分及噪声的惩罚,$con(o,Cl(o))$度量了数据对象 o 与其所属的簇 $Cl(o)$之间的点连通度,F 是错误划分的数据对象集合.$CMM\in[0,1]$,*CMM* 值越大,代表聚类质量越好.

4.2 算法参数选择

在进行对比实验之前,需要统一环境变量以及对算法参数进行调整.本文默认设置各数据流数据点到达速率为 1000*pt*/s,各算法中数据点的衰减速度一致.在 GCStream 中,设置λ=1.002,a=1;在 D-Stream 算法中,设置λ=0.998,a=−1;在 DenStream 和 DBSTREAM 中,设置λ=2,a=0.0028,使得权重衰减函数 $f=\lambda^{-a}$=0.998.对比算法的其他参数设置需参考其原始文献.由于 GCStream 算法主要受质心调节参数 k,*Dislen* 以及 N_{cg} 的影响,所以本节将通过实验探索这 3 个参数的选择.

4.2.1 质心调节参数 k

质心调节参数 k 决定着历史数据权重对网格质心的影响程度,是网格质心实时性的调节因子.0<k<1 表示提高网格质心的实时性,k>1 表示降低网格质心的实时性,k=1 代表不考虑网格质心实时性,所以本文分别选择小于 1、等于 1 以及大于 1 的 k 值进行对比实验.实验结果如图 4 所示:在 KDDCUP99 数据流上,k=1 时效果最好,于是本文在此基础上进一步缩小参数 k 的调整幅度,最终确定了 k=0.96;在 CoverType 数据流上,k=0.8 和 k=1 时结果较好,本文在此范围内继续微调参数 k,最终确定了 k=0.97;在 PAMAP2 数据流上,聚类结果相差不大,于是本文最终选择了 k=0.87.

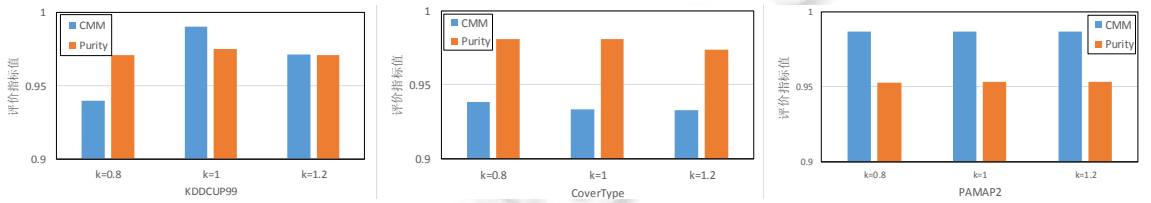


Fig.4 CMM and Purity comparison of GCStream algorithm under different k

图 4 GCStream 算法在不同 k 值下的 CMM 和 Purity 对比

4.2.2 距离阈值 Dislen

距离阈值 Dislen 为影响区域阈值,直接影响着网格之间的耦合以及簇的合并.本节实验通过对比 GCStream 算法在不同 Dislen 值下的 CMM 和 Purity 来探索 Dislen 的取值,实验结果如图 5 所示.

在 KDDCUP99 数据流上,当 *Dislen*=70 时,聚类结果的 CMM 值明显好于另外两个,并且 Purity 值也是比较高的,所以本文选择在 *Dislen*=70 的基础进一步微调参数.在 CoverType 数据流上,*Dislen*=80 和 *Dislen*=140 时聚类结果的 CMM 值高于 *Dislen*=300 的 CMM 值.进一步观察发现,*Dislen*=140 时聚类结果的 Purity 值高于 *Dislen*=80 时的 Purity 值,所以本文选择在 *Dislen*=140 附近继续寻找最优值.在 PAMAP2 数据流上,3 个参数值的聚类结

果相近,所以本文便选择了 *Dislen*=5 为本文实验的参数值.

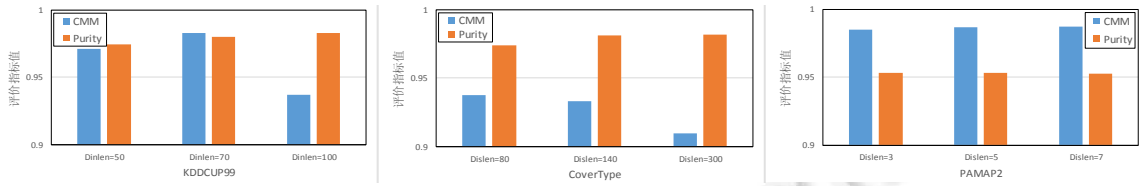


Fig.5 CMM and Purity comparison of GCStream algorithm under different Dislen

图 5 GCStream 算法在不同 Dislen 值下的 CMM 和 Purity 对比

4.2.3 核心网格集合 LD 大小 N_{cg}

核心网格集合由大于阈值ε的网格组成.本节实验选用不同的 N_{cg} 值进行聚类结果的 CMM 和 Purity 对比,实验结果如图 6 所示.在 KDDCUP99 数据流上,N_{cg}=6,8 时聚类结果的 CMM 值较高.进一步对比这两个不同 N_{cg} 值的聚类结果评价指标值发现,他们的 CMM 值相差不大,但是 N_{cg}=6 时,聚类结果的 Purity 值要高些,所以本文在 KDDCUP99 数据集上设置 N_{cg}=6;在 CoverType 数据流上,当 N_{cg}>12 时,算法聚类结果的 CMM 指标值较小,当 $N_{cg}\leqslant$12 时,聚类结果的两个评价指标相差不大,所以本文在 CoverType 数据流上设置 N_{cg}=12;在 PAMAP2 数据流上,不同 N_{cg} 值得到的聚类结果 CMM 值相差不大,但是当 N_{cg}=12 时,*Purity* 值要高些,所以本文在 PAMAP2 数据流上设置 N_{cg}=12.

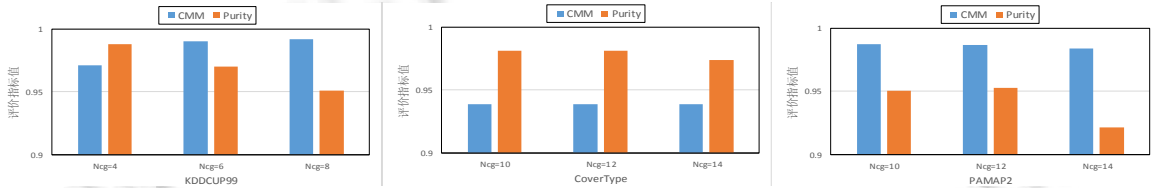


Fig.6 CMM and Purity comparison of GCStream algorithm under different N_{cg}

图 6 GCStream 算法在不同 N_{cg} 值下的 CMM 和 Purity 对比

综上,本文后续实验在 3 个 UCI 真实数据集上,质心调节参数 k 分别设置为 0.96,0.97,0.87;网格质心距离阈值 Dislen 分别设置为 64,144,5;核心网格集合 LD 大小 N_{cg} 分别设置为 6,12,12.

4.2.4 数据集处理

在实验进行之前,有时需要对数据集进行标准化处理.因为实验数据的不同维度代表不同含义,有时数据跨度差别非常大.这就需要对数据进行标准化处理以消除不同维度间的量纲差异,使数据具有可比性.但如果对数据跨度不大的数据集也进行处理,则可能会丢失数据集的真实性和原始性.为此,本文在 3 个 UCI 真实数据集(标准化和非标准化)上测试了 GCStream 算法和 3 种对比算法的聚类 Purity 值来决定是否对数据集进行标准化处理.实验结果见表 2,可以看到各算法在 3 个数据集标准化和非标准化下的聚类 *Purity* 值差异很小,所以本文选择不对数据集进行标准化处理.

Table 2 Purity comparison under standardized and non-standardized data sets

表 2 标准化和非标准化数据集下的 Purity 对比

测试数据集	标准化			非标准化		
	KDDCUP99	CoverType	PAMAP2	KDDCUP99	CoverType	PAMAP2
GCStream	0.975	0.899	0.957	0.970	0.980	0.952
D-Stream	0.910	–	–	0.916	–	–
DenStream	–	0.720	–	–	0.749	–
DBSTREAM	–	–	0.689	–	–	0.692 9

4.3 聚类质量评价

4.3.1 UCI 数据集的聚类结果

为了验证 GCStream 算法的聚类质量,本文在 3 个 UCI 数据集上运行了 GCStream 算法、GCStream-UC 算法、D-Stream 算法、DenStream 算法和 DBSTREAM 算法,比较了这些算法的 Purity 和 CMM 指标.实验中,各个算法每隔 25s 对流数据进行一次聚类,图 7 比较了各种算法的平均 Purity.

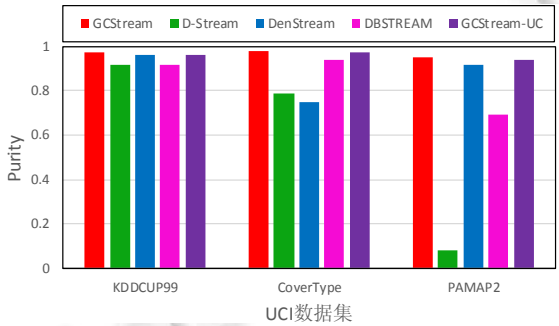


Fig.7 Purity comparison on UCI datasets

图 7 各算法在 UCI 数据集上的 Purity 对比

由图 7 可见,GCStream 算法在 3 个数据流上的平均 Purity 值均大于其他算法的平均 Purity 值.D-Stream 算法在 PAMAP2 数据流上的 Purity 值较小,这是因为 PAMAP2 数据流中同一时刻内生成的簇较多并且维度较高,D-Stream 算法对这类数据流比较敏感.图 8 展示了各种算法在每次聚类时得到的 CMM 值,图 9 比较了各种算法的平均 CMM.由图 8 可见:大多数时刻,GCStream 算法的 CMM 值都是优于对比算法的,并且比较稳定.值得注意的是:GCStream 算法的 CMM 值在 3 个数据流上的多个时刻均高于 GCStream-UC,说明基于网格耦合思想更新网格结构能够提高算法聚类质量.图 9 表明,在 3 个数据集上,GCStream 算法的 CMM 均值均大于其他算法的平均 CMM.

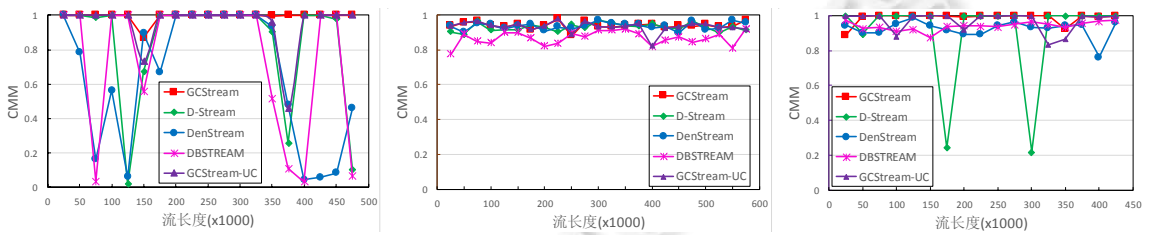


Fig.8 CMM comparison of algorithms on UCI datasets

图 8 各算法在 UCI 数据集上的 CMM 对比

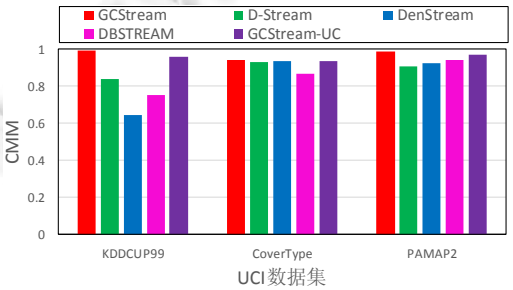


Fig.9 CMM mean comparison on UCI datasets

图 9 UCI 数据集上的 CMM 均值对比

4.3.2 GCStream 算法在不同数据流速度下的聚类质量

能够快速聚类数据流,是数据流聚类算法的一个重要特性.因此,本文在 KDDCUP99 数据流上以不同的数据流速度(1*k*/s,2*k*/s,7*k*/s)验证本文算法聚类质量.聚类结果如图 10 所示.首先,本文算法能够在这 3 种速度下处理完数据流,说明 GCStream 算法有能力处理速度较快的数据流.然后,分析聚类质量评价指标结果可以得出,随着数据流速度的上升,CMM 指标值有所下降,但是下降幅度并不大;Purity 指标值下降幅度比 CMM 值略大,但仍保持在较高的水平.说明 GCStream 算法在聚类高速数据流时依然可以保存较高的聚类质量.

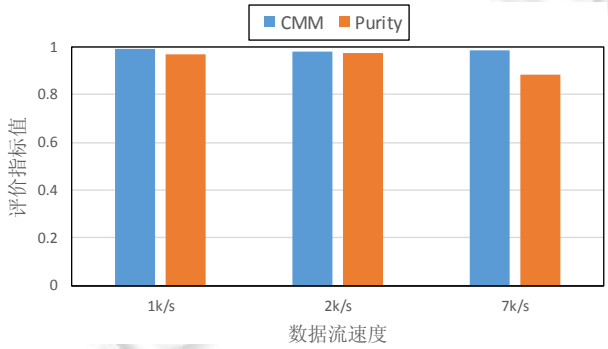


Fig.10 Cluster quality comparison under different stream rate

图 10 不同数据流速度下的聚类质量对比

4.3.3 GCStream 算法在不同网格边长下的聚类质量

本节实验主要测试不同网格边长对聚类质量的影响.以 KDDCUP99 为测试数据流,我们分别设置网格边长 *len*=40,100,120,160,其中,*len*=100 为本文整理数据集时发现的 KDDCUP99 数据集中簇之间的最小距离.聚类结果如图 11 所示.从图 11 可以看出,当网格边长大于 100 时,聚类结果的 CMM 值和 Purity 值随着网格边长的增加均有明显的下降.当网格边长小于 100 时,聚类质量总体相对稳定.实验结果说明:本文实验设置的网格边长 *len*=100 是比较准确的,并且 GCStream 算法聚类质量随着网格边长的增加而有所下降.

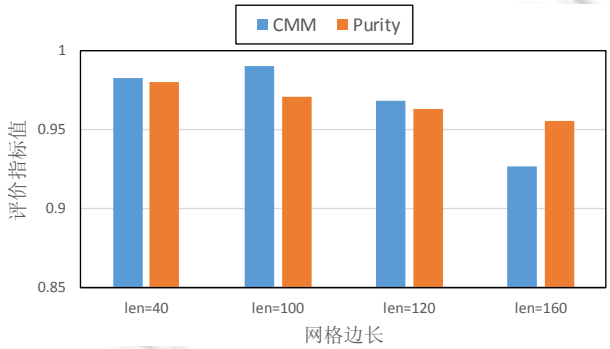


Fig.11 Cluster quality comparison under different grid sides

图 11 不同网格边长下的聚类质量对比

4.3.4 GCStream 算法捕捉簇的演变能力

数据聚类算法的一个重要特性是能够适应和捕捉簇的演变.为了验证 GCStream 算法的这两个特性,本文在人工数据集 MTD 上对 GCStream 算法进行了评估.在这个测试中,本文设置数据流到达速度为 1000*pt*/s,整个 MTD 数据流在 116s 内处理完.该数据集的分布如图 12 所示.图 12(a)~图 12(c)分别显示了 MTD 数据集中簇的生成顺序.其中,簇 1 和簇 2 中的数据是交叉分布的,在同一时刻,既有簇 1 中的数据到达也有簇 2 中的数据到达,

所以簇 1 和簇 2 能够同时生成.图 13 中显示了 GCStream 算法处理下的 MTD 数据分布.图 13(a)~图 13(d)分别显示了在 t=5,t=54,t=84,t=116 时刻生成的聚类结果.图中深颜色的区域代表当前时刻的生成的簇,浅蓝色的区域代表即将消失掉的簇.可以看出,GCStream 能够发现 4 个不同形状的簇并且不受噪声影响.图 14 显示了 MTD 数据流中簇的演变时刻.不同颜色的线条表示不同的簇,线条的长度表示簇存在的时间段.可以看到,簇 1 和簇 2 在初始时刻产生,在 55 时刻消失;簇 3 在 54 时刻产生,在 85 时刻消失;簇 4 在 84 时刻产生.除此之外,本文测得 GCStream 算法在人工数据集 MTD 的上的 Purity 均值为 0.983,CMM 均值为 1.说明 GCStream 算法具有较高的聚类质量.

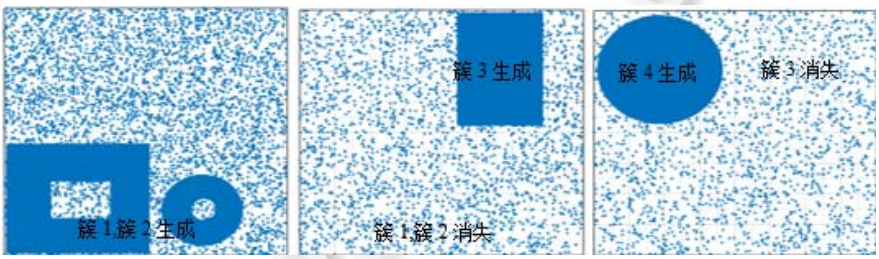


Fig.12 MTD data distribution

图 12 MTD 数据分布

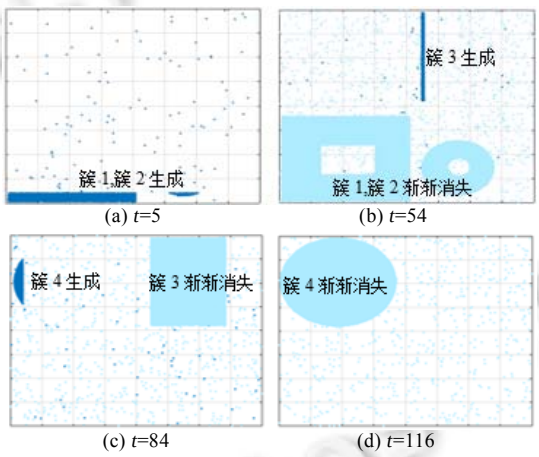


Fig.13 Data distribution of MTD data set changes with time

图 13 MTD 数据集的数据分布随时间的变化

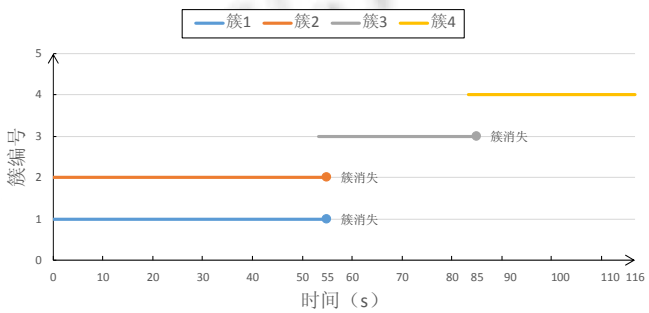


Fig.14 Evolution of clusters in MTD datasets

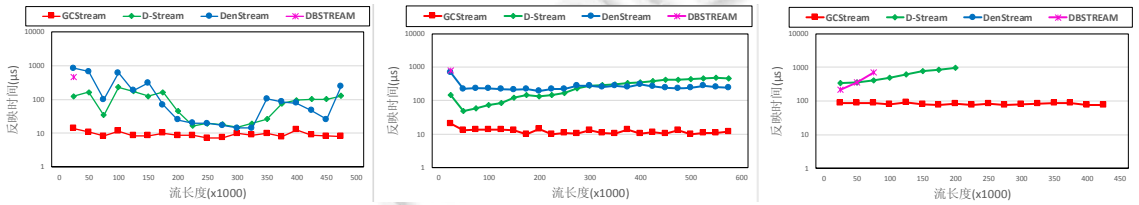
图 14 MTD 数据集中簇的演变

4.4 聚类效率评价

实时更新聚类结果对于数据流聚类算法至关重要.本文分别在多个数据集和不同维度上对各算法的效率进行了对比.

4.4.1 GCStream 算法在不同数据集上的效率

本节在 3 个 UCI 数据流上测试了 GCStream 与对比算法的聚类效率.设置数据流到达速率为 1000*pt*/s,并且每隔 25s 显示一次聚类结果.如果各算法能够在 25s 内处理完这段时间内到达的数据,则证明该算法能够正常运行;否则,说明该算法的效率不足以处理 1000*pt*/s 的数据流.图 15 显示了 25s 间隔内不同算法的响应时间对比.其中,DBSTREAM 算法在 3 个数据流上只在开始时正常运行,随后便运行失败;DenStream 和 D-Stream 算法在 PAMAP2 数据流上运行失败;而本文的 GCStream 算法能以 1000*pt*/s 的速度正常处理 3 个数据流并且所需时间最少,这说明 GCStream 算法效率比对比算法高.



(a) KDDCUP99 (b) CoverType (c) PAMAP2

Fig.15 Response time comparison on multiple datasets

图 15 多数据集上反映时间对比

4.4.2 网格边长与数据维度对 GCStream 算法效率的影响

本文在 MOAD 数据流上测试 GCStream 与对比算法在不同维度和不同网格边长上的聚类效率.图 16(a)、图 16(b)分别显示了网格边长 *len*=6 和 *len*=12.4 时,各算法在不同维度上平均效率.在不同大小的网格边长上比较可看出:随着网格边长的增加,GCStream,D-Stream 以及 DenStream 算法效率都有所提升.在不同数据维度上的算法效率比较可以看到:在数据维度小于 100 维时,GCStream 算法的效率是最高的;当数据维度大于 100 维时,GCStream 算法的效率也是比较高的,基本处于各算法效率的第 2 位.

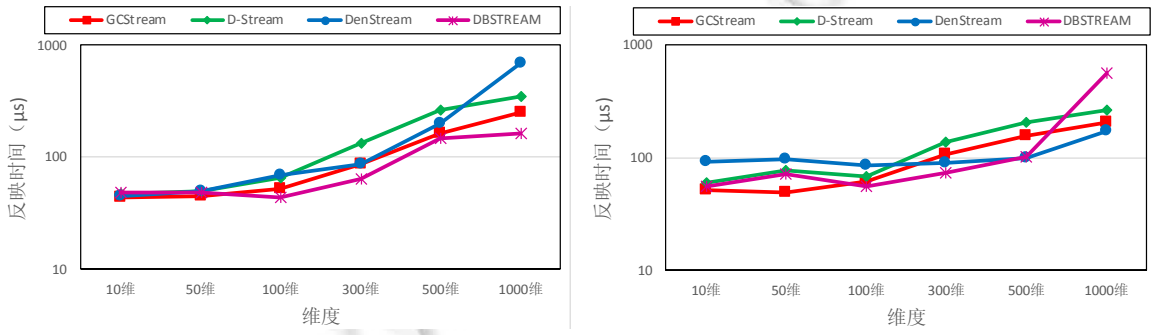


Fig.16 Response time comparisons in multiple dimensions and different grid lengths

图 16 不同网格边长和多维度上反映时间对比

5 结束语

本文针对现有数据流聚类算法在实时处理高速、大量的数据流时聚类效率和精度不高的问题,提出了一种基于网格耦合和核心网格的数据流聚类算法 GCStream.首先,通过网格耦合实现了对数据流更精确的汇总,提

高算法聚类质量;其次,本文根据数据流中局部权重较高的网格相比于局部权重较低的网格更可能为簇中心的特点引入了核心网格,然后以核心网格为簇中心生成簇,并且根据核心网格集合的变化来捕捉簇的演变;最后,通过真实数据集上进行实验,对比了本文所提方法与其他方法的聚类效果和聚类效率.实验结果表明,本文所提算法的聚类效果和聚类效率都优于对比方法.

由于本文算法的实验都是在网格边长相等的基础上进行的,没有考虑不同维度上的数据分布差异.所以本文的未来研究工作将着重研究根据不同维度上的数据分布采用不同的网格边长来使网格划分更精确,以进一步提高聚类质量.

References:

[1] Isaksson C, Dunham MH, Hahsler M. SOStream: Self organizing density-based clustering over data stream. In: Proc. of the Machine Learning and Data Mining in Pattern Recognition. Berlin, Heidelberg: Springer-Verlag, 2012. 264−278. [doi: 10.1007/978-3-642-31537-4_21]

[2] Silva JA, Faria ER, Barros RC, *et al*. Data stream clustering: A survey. ACM Computing Surveys, 2013,46(1):1−31.

[3] Zhang X, Furtlehner C, Germain-Renaud C, Sebag M. Data stream clustering with affinity propagation. IEEE Trans. on Knowledge and Data Engineering, 2014,26(7):1644−1656. [doi: 10.1109/TKDE.2013.146]

[4] Gong SF, Zhang YF, Yu G. Clustering stream data by exploring the evolution of density mountain. Proc. of the VLDB Endowment, 2017,11(4):393−405. [doi: 10.1145/3164135.3164136]

[5] Gama J, Žliobaitė I, Bifet A, Pechenizkiy M, Bouchachia A. A survey on concept drift adaptation. ACM Computing Surveys, 2014, 46(4):1−37. [doi: 10.1145/2523813]

[6] Masud M, Gao J, Khan L, Han J, Thuraisingham BM. Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans. on Knowledge and Data Engineering, 2011,23(6):859−874. [doi: 10.1109/TKDE.2010.61]

[7] Aggarwal CC, Han J, Wang J, 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. [doi: 10.1016/B978-012722442-8/50016-1]

[8] Chen Y, 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]

[9] Amini A, Saboohi H, Herawan T, Wah TY. MuDi-stream: A multi density clustering algorithm for evolving data stream. Journal of Network and Computer Applications, 2016,59(1):370−385. [doi: 10.1016/j.jnca.2014.11.007]

[10] Tu L, Chen Y. Stream data clustering based on grid density and attraction. ACM Trans. on Knowledge Discovery from Data, 2009, 3(3):1−27. [doi: 10.1145/1552303.1552305]

[11] Wan L, Ng WK, Dang XH, Yu PS, Zhang K. Density-based clustering of data streams at multiple resolutions. ACM Trans. on Knowledge Discovery from Data, 2009,3(3):1−28. [doi: 10.1145/1552303.1552307]

[12] Hahsler M, Bolaños M. Clustering data streams based on shared density between micro-clusters. IEEE Trans. on Knowledge and Data Engineering, 2016,28(6):1449−1461. [doi: 10.1109/TKDE.2016.2522412]

[13] Nguyen HL, Woon YK, Ng WK. A survey on data stream clustering and classification. Knowledge & Information Systems, 2015, 45(3):535−569. [doi: 10.1007/s10115-014-0808-1]

[14] O'callaghan L, Mishra N, Meyerson A, Guha S, Motwani R. Streaming-data algorithms for high-quality clustering. In: Proc. of the ICDE. 2002. 685−694. [doi: 10.1109/ICDE.2002.994785]

[15] Aggarwal CC, Han J, Wang J, Yu PS. A framework for projected clustering of high dimensional data streams. Proc. of the VLDB Endowment, 2004. 852−863. [doi: 10.1016/B978-012088469-8.50075-9]

[16] Cao F, Estert M, Qian W, Zhou A. Density-based clustering over an evolving data stream with noise. In: Proc. of the Siam Int'l Conf. on Data Mining. Bethesda, 2006. 328−339. [doi: 10.1137/1.9781611972764.29]

[17] Stolfo J, Fan W, Lee W, Prodromidis A, Chan PK. Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. In: Proc. of the Results from the JAM Project by Salvatore. 2000. 1−15.

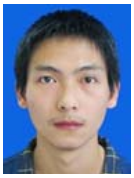
[18] Reiss A, Stricker D. Introducing a new benchmarked dataset for activity monitoring. In: Proc. of the Int'l Symp. on Wearable Computers. IEEE Computer Society, 2012. 108−109. [doi: 10.1109/ISWC.2012.13]

[19] Reiss A, Stricker D. Creating and benchmarking a new dataset for physical activity monitoring. In: Proc. of the Workshop on Affect & Behaviour Related Assistance. 2012. 1−8. [doi: 10.1145/2413097.2413148]

[20] Bifet A, Holmes G, Kirkby R, Pfahringer B. MOA: Massive online analysis. Journal of Machine Learning Research, 2010,11(2): 1601−1604.

[21] Kranen P, Kremer H, Jansen T, Seidl T, Bifet A, Holmes G, Pfahringer B. Clustering performance on evolving data streams: Assessing algorithms and evaluation measures within MOA. In: Proc. of the Int'l Conf. on Data Mining Workshops. 2010. 1400−1403. [doi: 10.1109/ICDMW.2010.17]

[22] Kremer H, Kranen P, Jansen T, Seidl T, Bifet A, Holmes G, Pfahringer B. An effective evaluation measure for clustering on evolving data streams. In: Proc. of the SIGKDD. San Diego, 2011. 868−876. [doi: 10.1145/2020408.2020555]



张东月(1993－),男,河北衡水人,硕士,主要研究领域为数据挖掘.

吴湘云(1964－),男,讲师,主要研究领域为微分方程,概率论与数理统计,数据分析.

周丽华(1968－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为数据挖掘,社交网络分析.

赵丽红(1974－),女,讲师,主要研究领域为数据挖掘.