一种适用数据流概念漂移检测与适应的增量密度聚类算法

陆昊阳,范玉雷,高 楠,杨良怀*
(浙江工业大学计算机科学与技术学院,浙江杭州 310023)

摘 要: 为处理随时间不断演化、非平稳数据流中的概念漂移问题,本文提出一种适用数据流概念漂移检测和适应的增量密度聚类算法(InCremental Density-based Clustering algorithm,ICDC). ICDC改进了1次遍历聚类框架,采用惰性方式处理离群点,由新达数据触发离群点评估,以区分潜在微簇和噪声;聚类过程中要求数据点和微簇满足特征依赖及时序依赖的条件,有效去除离群点集中的异常值,克服了现有离群点处理方式中因异常点的加入导致类簇结构以不可逆转方式持续恶化的情形;设计了一种离群点生命周期调节机制,有效控制缓存大小的增长;以类簇结构变化作为概念漂移指示器,设计了相应检测算法,提升了增量密度聚类算法对数据流演变过程中局部模式和全局模式变化的敏感性. 在多个真实和合成数据集上对数据流聚类质量及聚类性能、概念漂移检测和适应、算法的内存开销和计算开销等方面开展实验,结果表明,该算法在大多数数据集上的聚类结果都优于现有算法,同时能够有效检测概念漂移.

关键词: 概念漂移;增量聚类;密度聚类;数据流
基金项目: 国家重点研发计划项目(No.2022YFB3304100);浙江省重点研发计划"领雁"项目(No.2022C01088)
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112(2025)06-2050-13
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250060

An Incremental Density-Based Clustering Algorithm for Concept Drift Detection and Adaption over Data Stream

LU Hao-yang, FAN Yu-lei, GAO Nan, YANG Liang-huai*
(*College of Computer Science and Technology*, *Zhejiang University of Technology*, *Hangzhou*, *Zhejiang* 310023, *China*)

Abstract: To address concept drift in non-stationary data streams that evolve over time, this paper proposes incremental density-based clustering algorithm (ICDC), an incremental density-based clustering algorithm designed for concept drift detection and adaptation over data stream. ICDC enhances the one-pass clustering framework by introducing a lazy outlier handling mechanism, where outlier evaluation is triggered by newly arrived data to distinguish between potential micro-clusters and noise. During clustering, data points and micro-clusters must satisfy feature dependency and temporal dependency conditions, effectively filtering outliers from the potential outlier set. This approach prevents irreversible deterioration of cluster structures caused by incorporating outliers—a limitation of existing outlier processing methods. Additionally, ICDC incorporates an outlier life cycle adjustment mechanism to control buffer size growth efficiently. By leveraging cluster structure changes as concept drift indicators, we propose a detection algorithm that enhances ICDC's sensitivity to local and global pattern shifts during data stream evolution. We evaluate ICDC on multiple real and synthetic dataset, assessing clustering quality, performance, concept drift detection and adaptation, memory overheade, and computational overhead. Experimental results demonstrate that ICDC outperforms existing algorithms on most datasets, achieving superior clustering accuracy and effectively detecting concept drift.

Key words: concept drift; incremental clustering; density clustering; data stream
Foundation Item(s): National Key Research and Development Program of China (No.2022YFB3304100); Zhejiang Provincial Key Research and Development Program ("Ling Yan" Project) (No.2022C01088)

收稿日期:2025-01-16;修回日期:2025-05-08;责任编辑:孙瑶
*通讯作者:杨良怀

1 引言

技术进步催生了大量数据,如在机器状态检测、大气科学数据监测、社交媒体监测等领域,这种数据持续涌现被称为流数据,是重要类型的数据[1]. 流数据具有高速、无限性和时效性等特点[2]. 与传统的静态数据不同,流数据在不断变化过程中带来一个问题,即概念漂移,概念漂移的检测与适应成为数据处理的重要挑战之一.

概念漂移是指流数据所代表的概念或特征随时间推移而发生变化的现象[3]. 在针对真实世界具体任务建模过程中,如因外界环境变化、元器件老化、机器组件修改或升级等原因,状态检测和监测过程具有非平稳性,数据分布会随时间发生变化,导致基于原有数据的模型无法适用于新样本[4,5]. 对于需要实时或快速响应的应用,及时识别和适应漂移至关重要.

增量聚类是数据流分析中最常见和最有用的工具之一,它是一个从动态和不断增长数据中提取模式的实时过程[6,7]. 由于数据流的无限性,且随时间变化,有必要提供一种根据内存约束和数据到达速度可以连续处理大量数据的算法,因此提出1次遍历聚类框架[8,9]. 该框架仅遍历1次数据进行聚类分析,实现对流数据的实时处理和分析;框架中采用微簇和宏簇两阶段聚类方法,微簇是用来表示数据流中聚类信息的一种数据结构,用于维护数据点的统计摘要. 微簇通常是一组紧密相邻的数据点,它们代表了数据流中的一个子集或一种模式. 在处理数据流时,微簇会动态地根据新的数据点进行更新,以适应数据流的变化. 宏簇则可以看作是对微簇的进一步聚合,它代表了数据流中更广泛的模式或更高层次的聚类结构.

尽管1次遍历聚类框架具有高效性,同时也存在一些问题. 在处理实际应用数据流时,1次遍历聚类框架面临的挑战之一是存在噪声点. 由于缺乏对微簇形成密度的严格要求,这些噪声点会被错误地分配并保留到宏簇中,从而影响最终聚类结果的准确性. 图1对该情况进行说明. 微簇 m_2 在第 k 个数据点 x_k 处创建,数据点 x_k 为一个异常噪声点;后续到达的与 m_1 中属于相同分布的数据点 x_{k+1}、x_{k+2} 与 x_{k+3},会依据就近原则误分到微簇 m_2. 如此持续下去会导致更坏的情况,越来越多的异常噪声点被分配到微簇 m_2,同时由于微簇 m_2 的存在,代表不同数据分布的微簇 m_1 及 m_3,连同微簇 m_2 构成宏簇,导致最终聚类结果以一种不可逆的状态持续恶化.

解决这一问题的方法之一是引入对微簇密度的限制或者进一步的微簇过滤和清洗. 通过设置密度阈值,只有当微簇中包含的数据点数量超过一定阈值时才将其视为有效的聚类结果,可以减少对噪声点的过度响应,提高聚类的准确性. 另一种方法是采用更为复杂的聚类算法,或将1次遍历聚类框架与离线聚类算法相结合,周期性地对数据进行批处理以纠正和优化1次遍历聚类框架生成的宏簇结果,从而提高聚类的质量和稳定性.

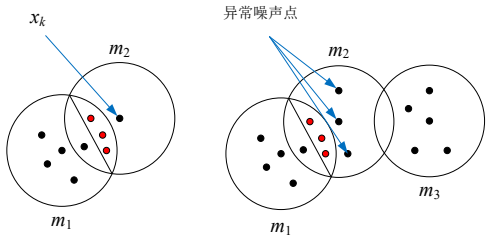


图1　特殊微簇情况说明

本文基于现有的1次遍历聚类框架,提出ICDC(In-Cremental Density-based Clustering algorithm). 在算法的第1阶段,引入缓存存储离群点,在离群点集合中寻找达到密度阈值要求的微簇. 为了控制缓存大小,依据缓存中离群点之间的特征依赖性及时序依赖性对其生命周期进行调节,新加入缓存中的离群点会延长与其在数据空间中距离相近的离群点的生命周期,而噪声点往往处于数据空间中较为稀疏的区域,最终从缓存中除去. 该机制在有效控制缓存大小的同时,也促进构成微簇的数据点在数据空间及时序上都保持密集状态. 在算法的第2阶段,对微簇进一步聚合构成宏簇,并通过图结构的形式,记录聚类构成的变化.

在本文所提算法中,微簇代表着数据流中的局部模式,宏簇则代表更广泛的模式及更高层次的类簇结构. 在增量更新过程中,新宏簇会出现,旧宏簇会过时,宏簇会发生彼此合并或分裂,这由数据流中数据分布发生变化所导致. 宏簇数量的变化、宏簇簇心的移动表征类簇结构的变化,潜在指示概念漂移的发生. 因此,可依据类簇结构的变化对数据流中的概念漂移进行检测. 主要贡献如下.

(1)改进现有1次遍历聚类框架,降低对噪声点的过度响应,提高聚类质量和稳定性.

(2)设计一种离群点生命周期调节机制,有效控制缓存大小. 通过对缓存中离群点之间的特征及时序依赖进行限制,保证构成微簇的数据点在时间与空间上的关联性.

(3)将类簇结构的变化作为数据流概念漂移指示器,设计相应检测算法.

(4)使用多个真实数据集及合成数据集对本文所提算法进行全面评估,通过与多种算法的比较,验证本文所提算法在数据流聚类方面的优越性,以及类簇结构变化在概念漂移检测中的有效性.

2 相关工作

2.1 基于密度的数据流聚类算法

聚类技术在连续数据流分析中有重要应用背景.通常数据流呈现非平稳性、时变性,其上形成的类簇可表现为数据空间中的任意形状.离线聚类方法,如Chameleon[10]、DBScan[11]和SPARCL[12]等,可构建任意形状聚簇,通常需要访问整个数据集,不适用于数据流场景.为解决数据流和任意形状类簇所面临的挑战,已提出多种聚类方法[2,8,13,14].CluStream是一种值得注意的方法[13],它采用基于滑动窗口内局部密度估计的两阶段聚类过程.但CluStream仅限于检测超椭圆聚簇,在类簇呈现更复杂形状的情形会受到限制[15,16].随后提出基于CluStream的改进版聚类算法DenStream[8]、DEC[14]和CEDAS[2],其能够处理具有任意形状聚簇和不断演化的数据流.DenStream[8]引入微簇和宏簇概念,利用自适应权重调整机制来适应各种聚类形状;DEC[14]维护核心和非核心类簇列表,并根据数据点权重动态调整类簇状态,以有效检测和处理概念漂移;CEDAS[2]是采用通用1次遍历聚类框架的一种全在线聚类算法,其利用超球微簇并维护聚簇图结构联系,实现在数据维度增加时维度稳定性、高速和最小计算开销,能自适应地以全在线方式加入和分离宏簇.

在处理不断演化的数据流时,以上方法都旨在解决与非平稳数据流和任意形状类簇相关的特定挑战,对离群点的处理不充分.主要不足如下.

(1)对离群点的识别不够准确.一些算法对离群点的定义和识别不够明确,导致一些重要的异常数据被错误归类为普通数据点,或一些正常数据被误认为是离群点.

(2)对离群点的处理能力有限.一些算法无法有效处理大量的离群点或者离群点分布不均匀的情况.在处理大规模数据流时,离群点的数量很多,因此算法需要能够有效地过滤和处理离群点.

(3)对离群点的响应不够弹性.一些算法对离群点的存在非常敏感,一旦出现离群点就会导致聚类结果的不稳定或者错误.理想情况下,算法应该能够在一定程度上抵抗离群点的影响,保持聚类结果的稳定性.

为解决上述不足,本文所提算法针对动态数据流中离群点识别问题进行设计,能够鲁棒地处理离群点.

2.2 概念漂移检测算法

在处理动态数据流时,概念漂移会导致模型失效,因此需要及时检测并适应概念漂移.目前,针对概念漂移检测,已有许多有效算法.ADWIN(ADaptive WINdowing)算法[17]是一种常用的检测算法,它通过调整窗口大小适应数据流变化,具有较低的内存消耗和算法复杂度.然而,ADWIN算法对于增量漂移等发生在较长时间段内的概念漂移类型不够敏感,会错过漂移检测.一些增强的ADWIN算法,如Multi-level ADWIN[18],通过引入多级ADWIN,用以处理数据流中不同级别的概念漂移,提高漂移检测的准确性.

HDDM(Hoeffding's inequality-based Drift Detection Method)[19]是一种常用的概念漂移检测算法.HDDM基于霍夫定不等式,通过比较2个样本均值的差异来检测概念漂移,它对于小幅度的概念漂移具有较高的灵敏度,但受到样本大小的限制,在处理大规模数据流时会出现误报或漏报.一些改进方法如FASE[20]、SDDE[21]等尝试利用模型集成,汇总多种检测结果,提高整体检测的稳定性和准确性;EI-kMeans[22]、FW-DA[23]等根据数据流的波动情况自动调整阈值,使检测器在概念漂移发生时更敏感,并减少误报概率.

KSWIN(Kolmogorov-Smirnov WINdowing)算法[24]引入核方法处理非线性概念漂移,并在滑动窗口内应用核函数捕获概念漂移的非线性特征,使其具有更高的灵活性和适用性.该算法能够捕捉更复杂的概念漂移模式,但需要调整核函数的参数以适应不同类型的数据流,并且在处理高维数据时,会受到维度灾难的影响.

目前的增量式数据流聚类算法虽已考虑了概念漂移的挑战[25],并且设计了相应机制来改进聚类质量,如CDAP-DSS[26]通过对样本集动态维护,实现更准确的预测.但与之前提及的概念漂移检测算法不同,大部分聚类算法仅通过最终的聚类质量来评估对数据流的适应性,而没有对算法在处理概念漂移时的检测延迟、检测准确率和漂移幅度等的具体表现进行充分评估.本文所提算法将通过类簇结构前后变化作为检测数据流中概念漂移的指示器,在检测延迟、检测准确率等指标上评估算法在概念漂移检测方面的性能,而不仅仅以聚类质量来评估对数据流的适应性.

3 基于增量密度聚类的概念漂移处理方法

针对现有1次遍历聚类方法易受离群点干扰的问题,提出ICDC,在提高聚类质量的同时实现对数据流中概念漂移的检测.ICDC的主体框架采用CEDAS的微簇与宏簇结构.在给定预设半径大小的球形区域所包含数据点的密度达到给定阈值时,该空间中的点集构成1个微簇;1个孤立的微簇或多个相互交叠的微簇构成宏簇.微簇表示空间中数据分布的局部模式(球形),宏簇则表示更为广泛的模式及更高层次的聚簇结构(可以是任意形状的空间分布).随着数据分布发生变化,在增量更新的过程中,新微簇会出现,与其他微簇合并或加入已有宏簇;旧宏簇会过时而消亡,宏簇之间会彼此合并,甚至1个宏簇也会分裂为2个子宏簇.以上微簇、

宏簇状态的变迁均随数据流数据分布的变化而改变,与概念漂移存在紧密联系. 本文将概念漂移的指示器归结为聚簇结构发生的2种变化:宏簇数量的变化、宏簇簇心的移动. 依据这2种指示器对数据流中的概念漂移进行检测.

ICDC处理新到来的数据分为2个阶段. 第1阶段是数据点放置,确定新到数据点是归入已有微簇或作为离群点. 若尚未有微簇形成,则将新数据点加入离群点集合中;若已有微簇,则计算新数据点与各簇心之间的余弦距离,若距离均大于预设距离阈值,则将新数据点归入离群点集,否则将该数据点放置到最近的微簇并更新该微簇的属性.

ICDC采用"惰性"方式处理"离群点",理由是离群点是新出现的"新概念"种子. 离群点的出现是因数据噪声、异常行为或者数据集不完全性所导致. ICDC将按照预设规则无法归属任何微簇的数据点临时保留在缓存中,这些点起初处于数据空间中的稀疏区域,聚集形成离群小团簇,其密度尚未达到形成微簇的阈值. 离群点小团簇密度或升或降,长成微簇或过时夭折. 缓存终有限,其大小通过离群点之间的特征及时序依赖进行限制. 新入缓存的离群点由于其特性未知,延长它及其距离相近离群点的生命周期以观其效,处于数据空间中孤立、稀疏区域的噪声点最终会从缓存中除去,从而有效控制缓存大小;同时,保障构成微簇的数据点在数据空间及时序上都保持合适的密度.

ICDC的第2阶段是聚合微簇构成宏簇,维护相应聚簇图结构,记录聚簇构成的变化. 1个孤立的微簇或多个相交的微簇构成宏簇. 第1阶段长成的微簇如果与宏簇相交,则会加入宏簇中;随着时间推移,某个(些)微簇会因衰老而终结,从宏簇中删除. 删除微簇会导致相应宏簇的分裂,由此引起数据分布模式的变更,即会发生概念漂移,可以启动检测.

3.1　ICDC

ICDC根据数据点的密度进行聚类,涉及离群小团簇、微簇、宏簇的动态生长、演化、消亡过程,需要经验确定几个超参数:微簇的半径(r)、每个微簇所需的最少数据点数量(MinPts)、衰减周期(Decay). 微簇采用超球结构,其半径决定在密度聚类过程可落入超球的数据点的距离范围,只有在半径范围内的数据点才被认为是彼此的近邻并形成微簇. 在密度聚类中,仅当给定区域的密度达到一定程度时才能够形成微簇,最少点数量规定了形成一个有效聚类所需的最少数据点数量.

在数据流中,模式的出现、消亡存在时效性,引入微簇存活时间,即衰减周期来反映变化. 如以空气污染模式随时间、空间变化的情况,来区分短期和长期的变化,发现暂时的局部异常和总体分布的极端情况. 衰减周期定义了1个时间间隔Decay,微簇或离群簇最后更新时间与新到数据时间间隔若超过衰减周期,则删除相应过时的微簇或离群簇. 衰减周期的设置与具体应用存在关联.

算法中采用数据结构表示微簇及离群簇. 微簇数据结构mc:{c:簇心,n:微簇中数据点个数,ct:微簇创建时间,lut:微簇最后更新时间},离群簇数据结构olc:{c:簇心,n:包含数据点个数,ct:离群簇创建时间,lut:离群簇最后更新时间},其中mc.ct表示离群簇演化为微簇的时间,lut则代表分配给该离群簇或微簇的最后1个数据点的时间戳. 全体微簇集记为mcSet、全体离群簇集记为olcSet. 其中,各个微簇的簇心是由所包含数据点的综合特征向量均值形成的虚拟点,能够同时表征离散属性的分布模式和连续属性的统计特性,适用于混合数据类型. 为统一特征空间,对离散属性进行独热(One-Hot)编码,将编码结果与连续属性合并为综合特征向量,并对合并后的特征向量归一化.

当ICDC接收新数据点x,其中数据点包含时间戳信息x.ts,基于设定的分配规则将新数据点分配到离群点集或现存的微簇. 需要检查微簇集mcSet是否为空,如果为空,则将x作为离群点进行更新,由更新离群点集过程updtOutlier实施(算法1代码行2~3);否则,它会计算x与各个微簇簇心之间的距离,并将x归类到最近的微簇中(代码行6~9). 如果x与最近微簇的距离小于给定的半径r,并且数据点的时间戳与该微簇最后更新时间在时序上接近,则将其分配到微簇中,由更新微簇过程updtmc实施更新(代码行10~11);否则,将其标记为离群点,由更新离群点集过程updtOutlier实施. ICDC依据前述方法更新微簇和离群簇,在更新过程中执行清理与合并操作,清理过时的离群簇与微簇(由过程deleteOlc&mc实施),或在满足合并条件时合并微簇形成宏簇,同时,维护聚簇图结构记录宏簇与微簇之间的关系(由过程updtMC&Graph实施). ICDC伪代码如算法1所示.

3.1.1　更新离群簇

当新数据点归类为离群点后,需要检查离群簇集是否为空. 若为空,算法会创建1个新离群簇,并将当前数据点作为其唯一成员加入新离群簇,同时记录该数据点的时间戳为离群簇的创建时间与最后更新时间(过程1代码行1~4);若不为空,则算法会遍历现有的离群簇,将数据点与每个离群簇的中心进行距离计算(代码行8),判断是否满足加入条件. 若存在离群簇被更新(代码行9~11),判断该离群簇的密度是否达到阈值要求,若达到要求,则代表1个新微簇形成(代码行12~16). 如图2所示,A、B、C分别表示1个新数据点加入离群簇集的3种情况,圆代表现存的离群簇,算法针

算法 1 ICDC

输入:数据点 x、半径 r、衰减周期 Decay、微簇集 mcSet、最少点数量 MinPts

输出:微簇集 mcSet、离群点集 olcSet、宏簇集 MacroSet、簇图结构 graph

1. 新数据点到达
2. If mcSet=∅ then {
3. updtOutlier(x, olcSet);}
4. else{
5. minDist=∞; mc_{clst}=NULL;//找最近微簇
6. for mc ∈ mcSetdo{
7. d=disFun(x, mc.c);//计算 x 与 mc 簇心 c 的距离
8. If d<minDist then{//找到距离最近的微簇 mc_{clst}
9. minDist=d; mc_{clst}=mc;}}
10. If minDist<$r$and x.ts−mc.lut<Decay then{
11. updtmc(x, mc_{clst});}//将数据点加入距离最近的微簇;
12. else{//将数据点加入 olcSet
13. updtOutlier(x, olcSet);}}

对 3 种不同情况分别进行处理. A 与 B 表示数据点满足分配给现存离群簇的条件,在情况 B 中,数据点只落在 1 个离群簇的半径范围内,因此只需要更新该离群簇的属性,A 则代表一种更复杂的情况,它处于 2 个离群簇的交集区域,显然它的存在对 2 个离群簇都产生影响,需要对它们的属性做更新. 因此,在聚类结构的增量更新过程中,会出现 1 次更新出现 2 个新微簇的现象,有利于获得更全面的数据分布信息. C 表示数据点未满足分配条件,与现存的离群簇都不建立联系,需要创建 1 个新离群簇来代表该数据点(代码行 17~21).

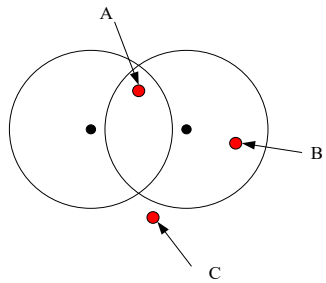


图 2 离群点的 3 种情况

3.1.2 更新微簇

若新数据点被分配到某个现存的微簇,该部分算法对微簇的属性进行更新,为了使簇心的更新更加平滑,采用“距离加权”方式,距离簇心越近的数据点被赋予的权重越大,减少微簇中边缘点对簇心的影响. 同时,对微簇的最后更新时间进行更新,以便后续算法分析不同微簇之间的关系. 伪代码如过程 2 所示.

3.1.3 更新宏簇和图结构

微簇为图结构中的节点,若 1 个新微簇被创建或 1 个旧微簇被更新,需要判断它与其他微簇在数据空间中的相交情况,用边的形式将代表相交微簇的节点进行连接. 2 个微簇相交需要满足 2 个条件:空间相交、时间相交.

定义 1 (空间相交) 若 2 个微簇所代表的空间有部分重叠,2 个微簇之间的距离<2 倍的微簇预设半径,则称这 2 个微簇在空间上相交.

定义 2 (微簇的生命周期) 微簇的创建时间 C 及最后更新时间 U 构成微簇的生命周期,记为 $[C, U]$.

定义 3 (时间相交) 指 2 个微簇各自生命周期时间有重叠. 形式上,微簇 i 的生命周期为 $[C_i, U_i]$ 与微簇 j 的生命周期为 $[C_j, U_j]$,若两者的交集不为空,则称这 2 个微簇在时间上相交.

过程 1 更新离群簇集合(updtOutlier)

输入:x、r、Decay、x.ts、MinPts、微簇集 mcSet 和离群簇集 olcSet

输出:离群簇集 olcSet

1. If olcSet=∅ then {
2. create a new olc;//创建 1 个新离群簇
3. olc.c=x; olc.n=1;
4. olc.ct=olc.lut=x.ts; olcSet←olcSet∪olc;
5. else{
6. matched = False;
7. for olc ∈ olcSet do{
8. d=disFun(x, olc.c);//计算 x 和 olc 簇心的距离 d
9. If d<$r$and x.ts−olc.lut<Decaythen{
10. olc.lut=x.ts; $olc.c=\frac{olc.c\cdot olc.n+x}{n-1}$;
11. olc.n=olc.n+1; matched=True;}
12. If olc.n>MinPtsthen{
13. create a new mc;//创建 1 个新微簇
14. mc.n=olc.n; mc.c=olc.c;
15. mc.ct=mc.lut=x.ts; mcSet←mcSet∪mc;
16. updtMC&Graph(mc);}}
17. If matched = False then{
18. create a new olc;
19. olc.c=x; olc.n=1;
20. olc.ct=olc.lut=x.ts;
21. olcSet←olcSet∪olc
22. deleteOlc&mc(mcSet, olcSet, Decay);}

宏簇可以通过图结构的形式记录 1 个宏簇是由哪些微簇相交形成,其中图中的节点表示微簇,若微簇 mc_i 与微簇 mc_j 相交,则在图中存在 1 条边 $<mc_i, mc_j>$ 连接图中代表这 2 个微簇的节点,由此形成的图结构称为簇图结构.

该部分算法依序检查微簇集中的每对微簇,计算

过程2　更新微簇(updtmc)

输入:微簇 mc、微簇集 mcSet、离群簇集 olcSet 和 Decay

输出:更新后的微簇 mc

1. $w \Leftarrow \exp(-\frac{\text{disFun}(x, \text{mc}.c)}{r})$;
2. $\text{mc}.c \leftarrow (1-w)\cdot \text{mc}.c + w \cdot x$;
3. $\text{mc}.n \leftarrow \text{mc}.n + 1$;
4. $\text{mc.lut} \leftarrow x.\text{ts}$;
5. deleteOlc&mc(mcSet, olcSet, Decay).

簇心的距离与最后更新时间的差值,若满足定义的2个相交条件,则在簇图结构中建立1条边,连接这对微簇,如过程3所示.

过程3　更新宏簇与图结构(updtMC&Graph)

输入:微簇集 mcSet、Decay 和 mc

输出:宏簇集 MacroSet、簇图结构 graph

1. foreach mc1 ∈ mcSet do
2. $d = \text{disFun}(\text{mc}.c, \text{mc1}.c)$;
3. If $d < r$ and $|\text{mc.lut} - \text{mc1.lut}| < \text{Decaythen}$
4. 在图结构 graph 中添加边.

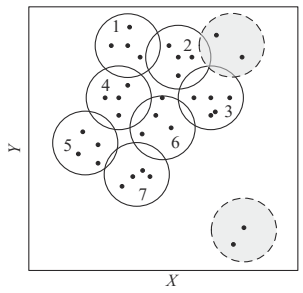
数据点、微簇、离群簇及宏簇的关系如图3(*a*)所示,用点代表数据空间中的数据样本,实心圆代表微簇,虚线圆代表离群簇,微簇之间彼此相交形成宏簇.从图3(*a*)可以观察到,离群簇虽然在数据空间中与某些微簇相交,但在算法中并不选择建立微簇和离群簇之间的联系,目的是避免离群簇中的异常值被分配到微簇中,影响聚类质量.在图3(*b*)中,每个节点代表1个微簇,它们之间通过边的形式进行连接,形成最终的宏簇,宏簇以簇图结构的形式进行记录.

3.1.4　删除过时离群簇与微簇

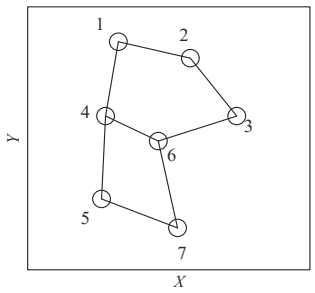
当1个微簇或离群簇在一段时间内没有接收到新数据点时,就被认为是过时的,需要从对应的集合中删除.删除微簇时,会导致对应宏簇分裂,宏簇的分裂代表对应数据分布的模式发生改变,即发生概念漂移.因此,在删除过时微簇时,需要判断对应宏簇的连通状态.

如1个宏簇是连通的,则从代表该宏簇图结构中的任意1个节点出发,都可以通过边的路径到达图中的其他任意节点.对于连通的宏簇,任意宏簇中的2个微簇 mc_{i_0}、mc_{j_0},存在1条包含 m 个微簇 $\text{mc}_1, \text{mc}_2, \cdots, \text{mc}_m$ 的通路,其中 mc_{i_0} 与 mc_1 相交,mc 与 mc_2 相交,$\cdots$,mc_m 与 mc_{j_0} 相交.如前面所述,2个微簇相交是空间、时间的同时相交.

算法首先遍历离群簇集,若存在离群簇超过1个衰减周期未进行更新,则将其从集合中删除;其次,遍历



(*a*) 数据点、微簇、离群簇及宏簇



(*b*) 微簇、宏簇及簇图结构

图3　微簇、离群簇、宏簇及簇图结构关系

微簇集,同样地,若找到超过1个衰减周期未进行更新的微簇,则将其从微簇集和对应的宏簇中删除,判断删除该微簇后对应宏簇的连通状态,如不再连通,原宏簇分裂为2个(或多个)子宏簇,则只保留子宏簇中最后更新时间最大的微簇.主要动机是删除未过期但较久未更新的微簇,有助于避免历史数据的干扰,使用最新微簇来代表当前数据分布状态,同时能够简化模型结构,减少在数据流处理过程中的计算和存储开销.算法伪代码如过程4所示.

过程4　删除过时离群簇与微簇(deleteOlc&mc)

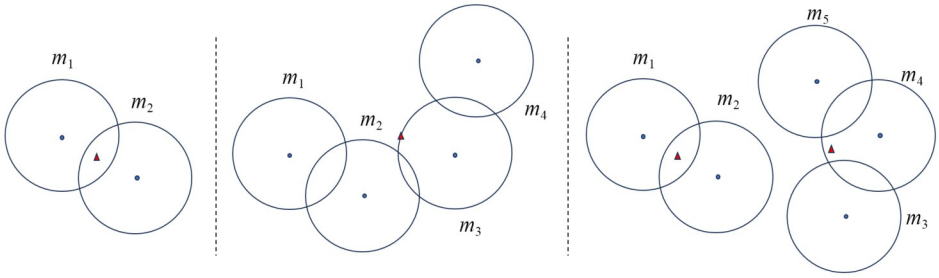
输入:微簇集 mcSet、outlier、Decay、graph 和 x.ts

输出:微簇集 mcSet、离群簇集 olcSet

1. foreach olc ∈ olcSetdo
2. If $x.\text{ts} - \text{olc.lut} < \text{Decaythen}$
3. olcSet.remove(olc);
4. foreach mc ∈ mcSetdo
5. If $x.\text{ts} - \text{mc.lut} < \text{Decaythen}$
6. mcSet.remove(mc);
7. 找到 mc 归属的宏簇 Macro_cluster;
8. 检查 Macro_cluster 是否保持连通状态;
9. IfMacro_cluster 未保持连通状态 then
10. 删除子宏簇中除最后更新时间最大的微簇以外微簇,删除 graph 中相应的节点与边.

3.1.5 概念漂移检测

对概念漂移的检测，主要包含在对宏簇与图结构的更新过程中，如图4所示. 图4(a)表示漂移发生前的类簇情况，用三角形表示宏簇簇心所在位置，可以将概念漂移的形式归结为如图4(b)及图4(c)所示的2种情况.



(a) 漂移发生前的类族情况 (b) 宏簇簇心随增量更新移动 (c) 新宏簇结构

图4 概念漂移情况归结

KL散度[27]是用来衡量2个概率分布之间差异的一种方法. 以一维随机变量为例，对于2个高斯分布 $p(x)\sim N(\mu_1,\sigma^2)$ 与 $q(x)\sim N(\mu_2,\sigma^2)$，其中 μ_1 和 μ_2 分别为2个高斯分布的均值，假定它们的方差同为 σ^2，可得：

$$
\begin{aligned}
&D_{\mathrm{KL}}\left(p\|q\right)\\
&=\int\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)\log\left(\frac{\dfrac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\dfrac{(x-\mu_1)^2}{2\sigma^2}\right)}{\dfrac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\dfrac{(x-\mu_2)^2}{2\sigma^2}\right)}\right)\mathrm{d}x\\
&=\int\frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu_1)^2}{2\sigma^2}\right)\left(\frac{(x-\mu_2)^2-(x-\mu_1)^2}{2\sigma^2}\right)\mathrm{d}x\\
&=\frac{1}{2}\left(1+\frac{(\mu_2-\mu_1)^2}{\sigma^2}\right)-\frac{1}{2}=\frac{(\mu_2-\mu_1)^2}{2\sigma^2}
\end{aligned}
\tag{1}
$$

由式(1)可知，$D_{\mathrm{KL}}(P\|Q)$ 的值与 $(\mu_2-\mu_1)^2$ 成正比，当均值差越大，KL散度越大，2个数据分布之间的差异也越大. 由正态分布特性知，均值落在 $(\mu_i-3\sigma,\ \mu_i+3\sigma)$ 中的概率为0.997 3. 因此，对于稳定分布 $|\mu_2-\mu_1|<6\sigma$ 是一个大概率事件，若超出这个阈值，认为2个分布发生了变化，即发生了概念漂移.

宏簇簇心代表对应数据分布的均值，对于图4(b)中宏簇簇心随着增量更新不断移动的情况，通过求取更新的标准差，使用 3σ 规则，若2个簇心之间的差值大于 6σ，则认为发生了概念漂移. 而图4(c)相较于图4(a)出现了1个新宏簇结构，代表出现了1个新数据分布，将这种情况也视为出现概念漂移的指示.

3.2 算法复杂性分析

假定微簇个数上限为 k_{mc} 个，离群簇个数上限为 k_{olc} 个. 在分析算法复杂度中，主要考虑开销大的操作次数. 本文所提ICDC主体有 k_{mc} 次距离计算，再加上1次updtmc更新微簇开销，或是1次updtOutlier更新离群点开销；其中updtmc开销主要在于1次deleteOlc&mc；updtOutlier开销主要包括 k_{olc} 次距离计算、k_{olc} 次updtMC&Graph及1次deleteOlc&mc；deleteOlc&mc主要涉及图连通性判定的开销 $\mathrm{O}(k_{\mathrm{mc}}+k_{\mathrm{edge}})$，其中 k_{edge} 是图结构最大边数；updtMC&Graph最多需要 k_{mc} 次距离计算. 因此算法总体计算开销 $\mathrm{O}(k_{\mathrm{mc}}+k_{\mathrm{olc}}+k_{\mathrm{olc}}\times k_{\mathrm{mc}})$ 次距离计算以及 $\mathrm{O}(k_{\mathrm{mc}}+k_{\mathrm{edge}})$ 次图操作. 鉴于距离计算开销远大于边的遍历，因此算法的开销主要在于 $\mathrm{O}(k_{\mathrm{mc}}+k_{\mathrm{olc}}+k_{\mathrm{olc}}\times k_{\mathrm{mc}})$ 次距离计算.

4 实验

为验证ICDC在概念漂移适应和概念漂移检测两方面的有效性，实验在5个数据集上对ICDC与目前增量聚类性能方面最优的4种算法CEDAS[2]、DenStream[8]、FIDC[9]和TWStream[28]进行比较. 另外，为评价概念漂移检测方面的性能，选择4种主流的漂移检测算法ADWIN、HDDM_A[17]、HDDM_W[17]及KSWIN[24]在多个合成数据集上对检测性能进行比较.

4.1 评价指标

为对各种算法的聚类性能进行评价，采用了3项评价指标：归一化互信息(Normalized Mutual Information，NMI)[29]、兰德指数(Rand Index，RI)[30]、调整兰德指数(Adjusted Rand Index，ARI)[31]. 令 S 是具有 n 个元素的待聚类样本集，U 和 V 分别表示2种聚类结果，设 $I(U;V)$ 是 U 和 V 之间的互信息，$H(U)$ 和 $H(V)$ 分别是它们各自的簇信息熵，则归一化互信息 $\mathrm{NMI}(U,V)=\dfrac{2\cdot I(U;V)}{H(U)+H(V)}$. NMI是用于衡量2个聚类结果之间相似性的指标，其取值范围为0~1，值越高表示聚类结果越相似.

令 e_i、$e_j\in S$，元素对 (e_i,e_j) 是在 U 和 V 中被划分为同一簇的元素对数，记为 a、b，是在 U 和 V 中隶属于不

同簇的元素对数，C_2^n表示数据集S中可以组成的总元素对数，则$\mathrm{RI}(U,V)=\frac{a+b}{C_2^n}$. RI值在0~1之间，值越高表示聚类结果越相似.

ARI是RI的概率修正版本，概率校正通过使用随机聚类模型所得的聚类结果之间所有成对元素比较的期望相似性来建立基线. 令RI是兰德尔系数，$E(\mathrm{RI})$是兰德尔系数的期望，$\max(\mathrm{RI})$是在给定聚簇数下的最大可能兰德尔系数，则$\mathrm{ARI}(U,V)=\frac{\mathrm{RI}-E(\mathrm{RI})}{\max(\mathrm{RI})-E(\mathrm{RI})}$. ARI取值范围为−1~1.

同时，为了评估算法对数据流中概念漂移的检测能力，从检测延迟(Delay)[31]、精确度(Precision)、漏报率(FNR)和F1度量(F1-score)4个方面评价. 实验中每个数据流的漂移节点已知，令t_d是概念漂移实际发生的时间节点，漂移前后数据流均保持1个稳定的数据分布，漂移检测延迟是指自漂移事件发生t_d后至第1次成功检测到漂移的时间间隔，反映了算法对概念漂移的灵敏性. 精确度、漏报率及F1度量则从算法的准确性、对正例的漏报程度等方面进行综合评价. 令R是检测范围，当算法正确在时间t检测到概念漂移，并且t落在$[t_\mathrm{d},t_\mathrm{d}+R]$时，将它视为一个真正例(TP)；当算法在时间$t$发出警报，但$t$不在$[t_\mathrm{d},t_\mathrm{d}+R]$时，表明发生了误报，将它视为一个假正例(FP)；当算法未能在$[t_\mathrm{d},t_\mathrm{d}+R]$检测到概念漂移，将它视为一次漏报(FN). 精确度定义为$\mathrm{Precision}=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$；漏报率定义为$\mathrm{FNR}=\frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}}$；F1度量定义为$\mathrm{F1-score}=\frac{2\cdot\mathrm{TP}}{2\cdot\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$.

在实验中，通过随机抽样的方法，使用95%的置信度确定获得可靠分布估计所需要的样本量，将检测范围R的大小设置为累计所需样本量需要的时间. 该范围提供一个理想的窗口，算法应在其中检测到概念漂移.

4.2　评价指标实验数据与参数设置

实验使用5个真实数据集对ICDC的聚类有效性进行评价，表1列出了数据集的相关信息，数据集KddCup99、Covtype、Waveform和Pen-based digit均来自UCI[32]. NSL_KDD[33]数据集是KddCup99的改进子集，删除冗余记录，并对数据进行重新采样，从而使数据集具有更多的数据分布. 此外，算法运行前需对原始数据进行预处理. 以KddCup99数据集为例，包含离散属性(如“服务类型”“标志”)与连续属性(如“流量”“时延”). 为统一特征空间，算法对离散属性进行One-Hot编码，将编码结果与连续属性合并为综合特征向量，并对合并后的特征向量经Min-Max归一化.

表1　聚类性能实验数据集

名称	样本量	样本维数	类别数	属性特点
KddCup99	494 021	38	23	混合
NSL_KDD	125 973	38	2	混合
Covtype	581 012	54	7	连续
Waveform	5 000	21	3	连续
Pen-based digit	7 494	16	10	连续

同时，基于River[34]生成了4个包含9 000个实例的数据流用来评估不同算法的检测性能，River是一个机器学习库，专注于在线学习和流式数据处理，提供一系列用于构建存在概念漂移的数据流方法. SEA[35]中包含9 000个实例，有3个属性，其中2个相关、2个类的决策边界由$f_1+f_2=b$给出，其中f_1和f_2是2个相关特征，b是预定义阈值，每隔3 000个样本改变一次b值，数据集中包含10%的噪声，表2列出了3个合成数据流的具体信息.

表2　漂移检测合成数据集

名称	样本量	1~3 000数据样本分布	3 001~6 000数据样本分布	6 001~9 000数据样本分布	漂移类型与特点
SD1	9 000	Mean=2, Std=1	Mean=8, Std=1	Mean=14, Std=1	突然漂移
SD2	9 000	Mean=1, Std=1	Mean=−1, Std=1	Mean=0, Std=1	增量漂移
SD3	9 000	Mean每500条数据 交替为5或10, Std=1	Mean每500条数据 交替为5或10, Std=1	Mean每500条数据 交替为5或10, Std=1	高频突移
SD4	9 000	Mean从2线性增至8, Std从1增至2	混合分布$N(0,1)$和$N(15,1)$, 权重比7:3	Mean=0, Std=1, 特征f_1与f_2的协方差从0渐变为0.8	混合漂移
SEA	9 000	Class 0:f_1+f_2 >= 8, Class 1:f_1+f_2 <= 3	Class 0:f_1+f_2 >= 10, Class 1:f_1+f_2 <= 4	Class 0:f_1+f_2 >= 15, Class 1:f_1+f_2 <= 6	突然漂移

合成数据集(Synthetic Dataset，SD)1与SD2分别用来模拟在相邻时间节点附近漂移幅度较大的快速漂移，以及变化幅度较小的缓慢漂移，突然漂移和增量漂移是这2类情况的典型代表. SD3与SD4分别模拟高频漂移和混合漂移场景，验证算法在复杂漂移场景下的有效性.

实验中，通过网格搜索参数的方法得到了最佳结果，并确定ICDC的参数设置. 具体而言，从数据流初始窗口抽取子样本(如前10 000条)，生成参数候选集(如$r\in[0.1,0.7]$，$\mathrm{Decay}\in[20,200]$)，对每组参数运行算法，

选择性能最优的参数组合. 在不同数据集上的参数设置如表3所示.

表3 参数设置

Datasets	r	MinPts	Decay
KddCup99	0.4	4	50
NSL_KDD	0.6	4	40
Covtype	0.2	4	250
Waveform	0.4	3	100
Pen-based digit	0.3	5	50
SD1	1.5	4	100
SD2	1.2	4	100
SD3	1.5	4	100
SD4	1.3	4	100
SEA	0.7	4	50

4.3 实验结果及分析

为了防止实验误差,本文通过多次实验取平均值作为实验结果. 实验中取10次结果的平均值作为实验结果.

4.3.1 聚类性能的比较

使用NMI、RI和ARI对算法的聚类性能进行评估,实验中针对不同数据集特征采用差异化的评估策略:对于KDDCup99、NSL-KDD和Covtype等大规模数据集,每处理10 000条数据记录进行1次指标计算;而对样本规模较小的Waveform和Pen-based digit数据集,则分别设置1 000和500条数据的评估间隔. 对得到的性能指标取平均值,结果如表4~表6所示,同时绘制了动态曲线图(图5)以直观呈现各算法评估指标在时序维度上的演变规律.

如表4~表6和图5所示,本文所提ICDC在NMI、RI和ARI这3个指标上均优于对比算法. DenStream、CEDAS及FIDC这3个算法基于通用的1次遍历聚类框架进行设计,虽然三者采用不同的策略提高聚类性能,但是在增量更新中,受框架制约,噪声数据易被分配到最终的聚类结构中. 相比之下,ICDC采用“惰性”处理“离群点”的方式,区分“离群点”中噪声与“新概念”的种子,使聚类结构的变化对“离群点”的响应更加弹性. 同时,ICDC从时间和空间2个维度获取数据模式,可以更好地适应数据流随时间的变化. 实验结果表明,ICDC有效提高了聚类质量及聚类性能.

表4 不同数据集的NMI比较

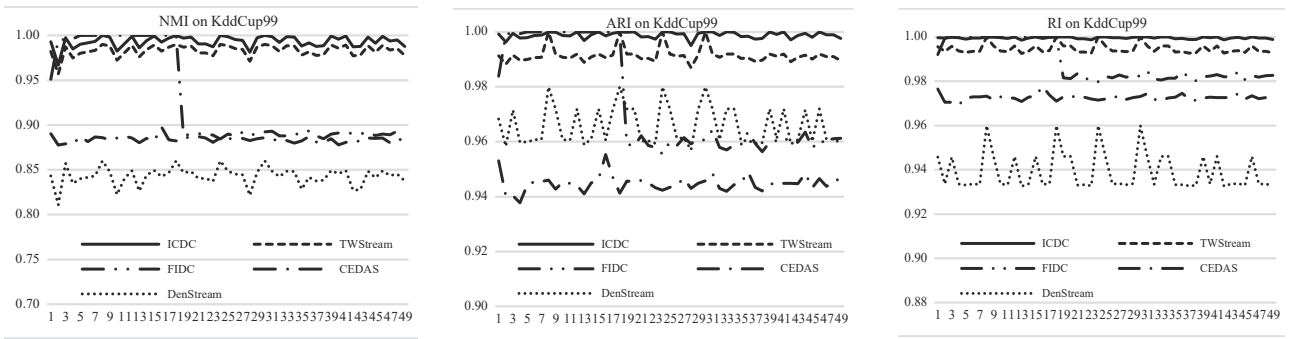
NMI	ICDC	DenStream	CEDAS	FIDC	TWStream
KddCup99	0.993 151	0.859 100	0.883 928	0.929 045	0.982
NSL_KDD	0.828 315	0.514 358	0.445 900	0.536 751	0.705
Covtype	0.319 344	0.277 594	0.162 100	0.249 509	0.315
Waveform	0.793 237	0.317 866	0.332 957	0.555 197	0.705
Pen-based digit	0.966 109	0.781 805	0.765 045	0.935 994	0.922

表5 不同数据集的RI比较

RI	ICDC	DenStream	CEDAS	FIDC	TWStream
KddCup99	0.999 562	0.975 800	0.972 661	0.988 422	0.995
NSL_KDD	0.974 898	0.693 661	0.683 200	0.770 931	0.908
Covtype	0.553 209	0.513 213	0.538 600	0.545 780	0.534
Waveform	0.857 788	0.687 343	0.667 934	0.757 996	0.802
Pen-based digit	0.971 045	0.929 849	0.928 300	0.967 606	0.961

表6 不同数据集的ARI比较

ARI	ICDC	DenStream	CEDAS	FIDC	TWStream
KddCup99	0.998 882	0.949 600	0.944 811	0.974 196	0.991
NSL_KDD	0.931 221	0.387 986	0.368 300	0.543 444	0.791
Covtype	0.139 327	0.081 966	0.049 400	0.134 952	0.122
Waveform	0.698 983	0.119 483	0.002 522	0.506 126	0.652
Pen-based digit	0.934 331	0.600 913	0.528 358	0.900 179	0.886



(a) NMI (b) ARI (c) RI

图5 KddCup99数据集中不同算法性能指标动态曲线图

4.3.2 概念漂移检测性能的比较

在概念漂移的检测延迟、精确度、漏报率、F1度量方面对算法的概念漂移检测性能进行评估,所得实验结果如表7~表10所示. 对于SD1模拟的概念漂移幅度较大、变化速度较快的漂移情况,ICDC在3个评价指标上均要优于对比算法. 通过观察新宏簇结构的出现,捕

捉新数据模式的出现,最少需要1个微簇自然聚集的时间就能够检测到漂移的发生,保证了较低的检测延迟与较高的准确度. 同时,在应对SD3和SD4模拟的高频漂移和混合漂移场景,ICDC均表现出了优异的检测性能.

表7　不同算法的检测延迟比较

Ave Delay	ICDC	ADWIN	HDDM_A	HDDM_W	KSWIN
SD1	3.5	11.0	12.33	53.0	14.0
SD2	37.0	91.0	72.67	39.0	15.0
SD3	4.2	22.5	18.70	65.0	19.3
SD4	48.7	105.0	88.30	75.0	35.5
SEA	144.5	379.0	203.00	198.0	68.0

表8　不同算法的精确度比较

Precision	ICDC	ADWIN	HDDM_A	HDDM_W	KSWIN
SD1	0.667 00	0.428 57	0.220 34	0.075 76	0.250 00
SD2	0.357 10	0.500 00	0.082 19	0.043 47	0.280 00
SD3	0.782 00	0.521 00	0.305 00	0.102 00	0.413 00
SD4	0.612 00	0.325 00	0.174 00	0.088 00	0.487 00
SEA	0.151 52	0.050 00	0.112 93	0.125 26	0.103 45

表9　不同算法的漏报率比较

FNR	ICDC	ADWIN	HDDM_A	HDDM_W	KSWIN
SD1	**0**	0.000	0.000 00	0.000	0.000
SD2	**0**	0.000	0.142 86	0.667	0.000
SD3	**0**	0.167	0.083 00	0.333	0.042
SD4	**0**	0.250	0.125 00	0.500	0.083
SEA	**0**	0.500	0.000 00	0.000	0.000

表10　不同算法的F1度量比较

F1-score	ICDC	ADWIN	HDDM_A	HDDM_W	KSWIN
SD1	0.800 00	0.444 6	0.361 11	0.140 84	0.440 00
SD2	0.526 31	0.667 0	0.150 00	0.076 92	0.476 19
SD3	0.845 00	0.602 0	0.427 00	0.153 00	0.625 00
SD4	0.734 00	0.398 0	0.241 00	0.132 00	0.598 00
SEA	0.263 16	0.091 0	0.184 23	0.200 34	0.187 50

对于SEA及SD2模拟的更为复杂的漂移情况,漂移前后数据分布差异小,通过判断宏簇簇心的变化来检测漂移,需要积累并观察更多的数据,因此ICDC的检测延迟要略慢于KSWIN. 此外,在某些情况下,虽然新宏簇结构的出现能很好地指示概念漂移的发生,但是在复杂数据流中,它的出现是因为当前数据流中体现的数据分布不完全,导致发出误报警,在SD2上的准确度及F1度量要略低于对比算法. 综上,在复杂情况下,ICDC在4个评价指标上保持了均衡,维持了较好的检测性能,能够有效检测概念漂移的发生.

使用非参数检验方法Friedman检验[36]对ICDC与其他对比方法在上述数据集上的Ave Delay、FNR、Precision及F1-score进行统计检验,对于给定的$K=5$种方法和$N=5$个数据集,Friedman检验在$\alpha=0.05$,即95%置信情况下认为样本集的分布不同,即各个方法之间的性能存在显著的差异. 另外,还使用Bonferroni-Dunn(BD)测试[37]计算了所有方法之间的显著性差异,首先计算各种方法的平均分类性能排名并进行排序,然后计算其秩和平均差值,即平均性能排名差值是否大于临界差值(Critical Difference,CD):

$$\mathrm{CD}=q_{\alpha=0.05}\sqrt{\frac{K(K+1)}{6M}} \tag{2}$$

其中,K为对比的方法种类;M为每个方法包含的对比数据集个数;$q_{\alpha=0.05}$为当置信度为95%时的q值,可以通过查表得到,约为2.724. 当2种方法的排名差值大于临界差值CD时,就可以认为这2种方法的性能存在显著差异. 图6~图9为5个方法在Ave Delay、FNR、Precision及F1-score上的CD图,其中ICDC表现出最低的检测延迟与漏报率,同时在Precision与F1-score这2个指标上的排名最高,且与较低的4种方法有显著差异.

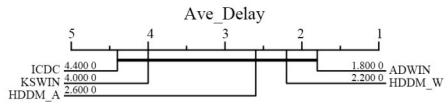


图6　各方法在Ave Delay上的CD图

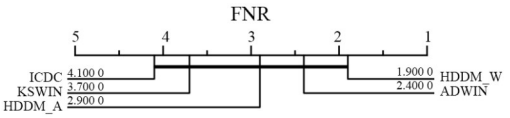


图7　各方法在FNR上的CD图

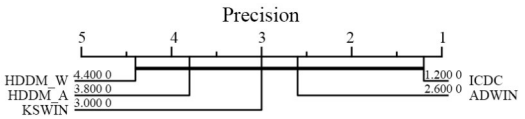


图8　各方法在Precision上的CD图

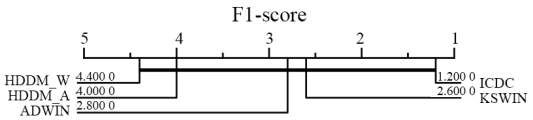


图9　各方法在F1-score上的CD图

4.3.3　算法开销比较

为评估ICDC的运行效率,在KddCup99数据集上与其他算法进行对比实验,每处理10 000条数据,评估1次算法的计算与内存开销. 距离函数是各个基于密度的增量聚类算法中调用频率最高的模块,实验中也采用CEDAS[2]以距离函数的调用次数作为算法开销的评价指标. 算法运行过程中需要对簇结构进行存储,离群

簇与微簇的数据结构大致相同,因此以存储的节点数量作为内存开销的评价指标. 图10和图11分别给出了各个算法在KddCup99数据集上的计算开销与内存开销情况.

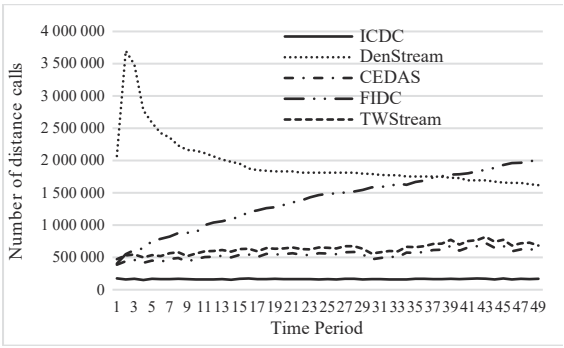


图10 不同算法计算开销比较

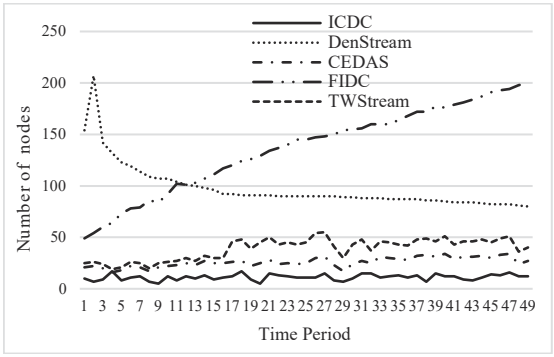


图11 不同算法内存开销比较

从实验结果可知,使用1次遍历聚类框架的算法其内存开销与计算开销呈正相关,微簇数量的增加意味着需要更多次调用距离函数来分配数据点. ICDC通过从时间和空间2个维度对微簇的形成进行限制,提高了算法效率. 相较于对比算法中表现最优的CEDAS,ICDC的平均计算开销减少了63%,内存开销减少了56%.

4.3.4 参数敏感性分析

在实际应用中,ICDC通常需要处理不断变化的数据流,聚类结果通常对一些关键参数非常敏感,如距离阈值r、生命周期Lifecycle及更新策略等. 为了更好地理解算法的稳定性和鲁棒性,本文在KddCup99、NSL_KDD和Covtype这3个真实数据集对算法的r、Decay进行敏感性分析,评估其对聚类质量的影响,实验结果如表11~表13所示.

从实验结果中可以观察到,不同的参数设置对不同数据集的聚类质量存在不同程度的影响,但在一定范围内均存在1个值可以得到最优的聚类质量. 具体而言,参数r的变化主要体现在,若r过大,会导致单一簇的聚类过于宽泛;若r过小,则会导致聚类过细,都会导致聚类质量的下降. 参数Decay的变化主要体现在,若Decay过小,会导致部分数据点被识别为异常噪声点,无法被归入某一个簇中;若Decay过大,则会导致部分已过时概念在聚类结构中无法被及时删除,导致代表新概念的数据点被错误聚类. 两者在不同的场景(数据集)运行过程中,需要寻找一个合适的参数设置,来得到较优的聚类质量.

表11 在KddCup99上不同r和Decay的聚类质量

关键参数	NMI	RI	ARI
r=0.2, Decay=50	0.930 876	0.999 478	0.930 875
r=0.3, Decay=50	0.973 599	0.999 497	0.997 964
r=0.4, Decay=50	0.993 151	0.999 562	0.998 882
r=0.5, Decay=50	0.986 149	0.999 431	0.998 581
r=0.4, Decay=30	0.976 428	0.999 748	0.996 983
r=0.4, Decay=40	0.985 917	0.999 621	0.998 609
r=0.4, Decay=60	0.984 551	0.999 095	0.998 069

表12 在NSL_KDD上不同r和Decay的聚类质量

关键参数	NMI	RI	ARI
r=0.4, Decay=40	0.736 163	0.908 245	0.808 043
r=0.5, Decay=40	0.779 509	0.937 288	0.864 279
r=0.6, Decay=40	0.828 315	0.974 898	0.931 221
r=0.7, Decay=40	0.796 291	0.959 382	0.905 124
r=0.6, Decay=20	0.678 875	0.893 162	0.758 386
r=0.6, Decay=30	0.776 515	0.963 623	0.878 889
r=0.6, Decay=50	0.806 231	0.951 844	0.895 972

表13 在Covtype上不同r和Decay的聚类质量

关键参数	NMI	RI	ARI
r=0.1, Decay=250	0.140 001	0.384 304	0.011 528
r=0.2, Decay=250	0.319 344	0.553 209	0.139 327
r=0.3, Decay=250	0.277 353	0.571 229	0.054 118
r=0.4, Decay=250	0.279 325	0.579 907	0.102 699
r=0.2, Decay=230	0.266 624	0.546 561	0.038 026
r=0.2, *Decay*=240	0.267 503	0.547 588	0.035 316
r=0.2, Decay=260	0.266 947	0.551 414	0.040 751

5 结论

针对非平稳数据集中的概念漂移问题,提出一种适用于概念漂移检测和适应的增量密度聚类算法. 算法采用的离群点惰性处理方式,由新数据触发对离群点评估,可以有效区分潜在的微簇和噪声,克服了CEDAS离群点处理方式中因异常点的加入导致类簇结构以一种不可逆的方式持续恶化的情形. 与该相关领域现有的最优算法进行实验比较. 从数据流聚类质量及聚类性能的实验看,ICDC优于所有对比算法;从概念漂移检测和概念漂移适应两方面实验来看,本文所提算法总体优于对比的主流算法;本文算法的内存开销和计算开销显著优于CEDAS算法.

本文所提算法还不能区分数据流中“噪声簇”和“新概念”种子,而在实际工业生产过程中,如传感器损坏等,异常数据连续出现,导致聚集成簇,但这些簇的出现并不意味着“新概念”的出现或概念漂移的发生,如何区分“测量偏差”引起的概念漂移仍是一个挑战.此外,如何在运行过程中依据数据分布的变化动态调整超参,需要进一步研究以提升模型的适应能力.

参考文献

[1] 陈志强, 韩萌, 李慕航, 等. 数据流概念漂移处理方法研究综述[J]. 计算机科学, 2022, 49(9): 14-32.
CHEN Z Q, HAN M, LI M H, et al. Survey of concept drift handling methods in data streams[J]. Computer Science, 2022, 49(9): 14-32. (in Chinese)

[2] HYDE R, ANGELOV P, MACKENZIE A R. Fully online clustering of evolving data streams into arbitrarily shaped clusters[J]. Information Sciences, 2017, 382: 96-114.

[3] WARES S, ISAACS J, ELYAN E. Data stream mining: Methods and challenges for handling concept drift[J]. SN Applied Sciences, 2019, 1(11): 1412.

[4] 乔俊飞, 孙子健, 汤健. 面向工业过程软测量建模的概念漂移检测综述[J]. 控制理论与应用, 2021, 38(8): 1159-1174.
QIAO J F, SUN Z J, TANG J. Overview of concept drift detection for industrial process soft sensor modeling[J]. Control Theory & Applications, 2021, 38(8): 1159-1174. (in Chinese)

[5] AGRAHARI S, SINGH A K. Concept drift detection in data stream mining: A literature review[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(10): 9523-9540.

[6] GAMA J, ŽLIOBAITĖ I, BIFET A, et al. A survey on concept drift adaptation[J]. ACM Computing Surveys, 2014, 46(4): 1-37.

[7] DING S F, WU F L, QIAN J, et al. Research on data stream clustering algorithms[J]. Artificial Intelligence Review, 2015, 43(4): 593-600.

[8] CAO F, ESTERT M, QIAN W N, et al. Density-based clustering over an evolving data stream with noise[C]//Proceedings of the 2006 SIAM International Conference on Data Mining. Philadelphia: Society for Industrial and Applied Mathematics, 2006: 328-339.

[9] LAOHAKIAT S, SA-ING V. An incremental density-based clustering framework using fuzzy local clustering[J]. Information Sciences, 2021, 547: 404-426.

[10] CAO H L, CHU Y H, ZHAO C Y, et al. Software multi-fault localization via Chameleon clustering in parallel[J]. Journal of King Saud University-Computer and Information Sciences, 2023, 35(8): 101676.

[11] ESTER M, KRIEGEL H, SANDER J, et al. A density-based algorithm for discovering clusters in large spatial databases with noise[C]//Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 1996: 226-231.

[12] BACH F. Optimization with sparsity-inducing penalties[J]. Foundations and Trends in Machine Learning, 2011, 4(1): 1-106.

[13] AGGARWAL C C, YU P S, HAN J W, et al. A framework for clustering evolving data streams[J]. Proceedings 2003 VLDB Conference, 2003, 1: 81-92.

[14] GUO X, ZHU E, LIU X, et al. Deep embedded clustering with data augmentation[C]//Asian Conference on Machine Learning. New York: ACM, 2018: 550-565.

[15] SAYED D, RADY S, AREF M. Enhancing CluStream algorithm for clustering big data streaming over sliding window[C]//2020 12th International Conference on Electrical Engineering (ICEENG). Piscataway: IEEE, 2020: 108-114.

[16] AHSANI S, SANATI M Y, MANSOORIZADEH M. Improvement of CluStream algorithm using sliding window for the clustering of data streams[C]//2021 11th International Conference on Computer Engineering and Knowledge (ICCKE). Piscataway: IEEE, 2021: 434-440.

[17] BIFET A, GAVALDÀ R. Learning from time-changing data with adaptive windowing[C]//Proceedings of the 2007 SIAM International Conference on Data Mining. Los Alamitos: Society for Industrial and Applied Mathematics, 2007: 443-448.

[18] SUREGAONKAR S, ASHOK S D, KARAR V, et al. Change point monitoring for vehicle incident detection at intersection[C]//2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM). Piscataway: IEEE, 2017: 100-105.

[19] FRÍAS-BLANCO I, DEL CAMPO-ÁVILA J, RAMOS-JIMÉNEZ G, et al. Online and non-parametric drift detection methods based on hoeffding's bounds[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(3): 810-823.

[20] FRÍAS-BLANCO I, VERDECIA-CABRERA A, ORTIZ-DÍAZ A, et al. Fast adaptive stacking of ensembles[C]//Proceedings of the 31st Annual ACM Symposium on Applied Computing. New York: ACM, 2016: 929-934.

[21] KOMORNICZAK J, ZYBLEWSKI P, KSIENIEWICZ P. Statistical drift detection ensemble for batch processing of data streams[J]. Knowledge-Based Systems, 2022,

252: 109380.
[22] LIU A J, LU J, ZHANG G Q. Concept drift detection via equal intensity k-means space partitioning[J]. IEEE Transactions on Cybernetics, 2021, 51(6): 3198-3211.
[23] LIU A J, ZHANG G Q, LU J. Fuzzy time windowing for gradual concept drift adaptation[C]//2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). Piscataway: IEEE, 2017: 1-6.
[24] TOGBE M U, CHABCHOUB Y, BOLY A, et al. Anomalies detection using isolation in concept-drifting data streams[J]. Computers, 2021, 10(1): 13.
[25] 朱颖雯, 陈松灿. 基于随机投影的高维数据流聚类[J]. 计算机研究与发展, 2020, 57(8): 1683-1696.
ZHU Y W, CHEN S C. High dimensional data stream clustering algorithm based on random projection[J]. Journal of Computer Research and Development, 2020, 57(8): 1683-1696. (in Chinese)
[26] 韩光洁, 赵腾飞, 刘立, 等. 基于多元区域集划分的工业数据流概念漂移检测[J]. 电子学报, 2023, 51(7): 1906-1916.
HAN G J, ZHAO T F, LIU L, et al. Concept drift detection of industrial data flow based on multivariate region set partition[J]. Acta Electronica Sinica, 2023, 51(7): 1906-1916. (in Chinese)
[27] STREHL A. Cluster ensembles-A knowledge reuse framework for combining multiple partitions[J]. Journal of Machine Learning Research, 2002, 3(3): 583-617.
[28] SUN J R, DU M J, LEW Z, et al. TWStream: Three-way stream clustering[J]. IEEE Transactions on Fuzzy Systems, 2024, 32(9): 4927-4939.
[29] RAND W M. Objective criteria for the evaluation of clustering methods[J]. Journal of the American Statistical Association, 1971, 66(336): 846-850.
[30] HUBERT L, ARABIE P. Comparing partitions[J]. Journal of Classification, 1985, 2(1): 193-218.
[31] KHAMASSI I, SAYED-MOUCHAWEH M, HAMMAMI M, et al. Self-adaptive windowing approach for handling complex concept drift[J]. Cognitive Computation, 2015, 7(6): 772-790.
[32] WANG Y Z, QIAN J X, HASSAN M, et al. Density peak clustering algorithms: A review on the decade 2014-2023[J]. Expert Systems with Applications, 2024, 238: 121860.
[33] TAVALLAEE M, BAGHERI E, LU W, et al. A detailed analysis of the KDD CUP 99 data set[C]//2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. Piscataway: IEEE, 2009: 1-6.
[34] AGUIAR G J, CANO A. A comprehensive analysis of concept drift locality in data streams[J]. Knowledge-Based Systems, 2024, 289: 111535.
[35] STREET W N, KIM Y. A streaming ensemble algorithm (SEA) for large-scale classification[C]//Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2001: 377-382.
[36] PEREIRA D G, AFONSO A, MEDEIROS F M. Overview of Friedman's test and post-hoc analysis[J]. Communications in Statistics-Simulation and Computation, 2015, 44(10): 2636-2653.
[37] DEMSAR J. Statistical comparisons of classifiers over multiple data sets[J]. Journal of Machine Learning Research, 2006, 7: 1-30.

作者简介

陆昊阳 男,1999年生. 现为浙江工业大学硕士研究生. 主要研究方向为数据挖掘.
E-mail: luhaoyang.zjut@gmail.com

高 楠 女,1983年生. 现为浙江工业大学计算机学院副教授. 主要研究方向为跨模态生成与检索、自然语言处理、医学图像处理.

范玉雷 男,1984年生. 博士,浙江工业大学讲师. 主要研究方向为数据库系统、数据流与数据挖掘.
E-mail: fyl815@zjut.edu.cn

杨良怀 男,1967年生. 现为浙江工业大学计算机学院教授. 主要研究方向为数据科学与工程.
E-mail: yanglh@zjut.edu.cn