论　文

一种基于优化模型的演化数据流聚类方法

杜航原[①], 王文剑[①②*], 白亮[②]

① 山西大学计算机与信息技术学院, 太原 030006
② 计算智能与中文信息处理教育部重点实验室, 太原 030006
* 通信作者. E-mail: wjwang@sxu.edu.cn

收稿日期: xxxxxxxx; 接受日期: xxxxxxxx
国家自然科学基金重点项目(批准号: 61432011, U1435212), 国家自然科学基金(批准号: 61673249)

摘要　本文针对数据流演化聚类问题, 建立了基于模糊最大熵的优化模型, 利用模糊隶属度表达类别划分的模糊性, 通过信息熵描述类别划分的有效性. 在此基础上定义了优化目标函数, 在滑动窗口下将数据子集的聚类过程理解为一个优化问题, 使聚类结果能有效描述数据内在结构特征, 同时维持相邻窗口间聚类模型的连续性. 将优化问题的解作为概念漂移检测的依据, 保证了检测结果的有效性, 有利于捕获聚类结构的变化趋势. 在仿真实验中, 利用人造数据集和真实数据集对新算法的有效性进行了验证, 并通过实验与多种演化聚类方法在聚类精度、概念漂移检测精度以及计算效率等多个方面进行了比较. 仿真结果表明了该算法的有效性, 在相同条件下其聚类精度和概念漂移检测精度相比其他聚类算法具有显著优势,能够同时降低计算耗费时间和存储空间.

关键词　数据流　演化聚类　优化模型　模糊隶属度　信息熵

1　引言

随着计算机网络和无线传感网络的大量应用, 在各个领域中大量数据以数据流的形态不断产生, 例如, 金融交易信息、电话呼叫记录、交通监测数据、网络访问日志、疾病监控数据等[1]. 相比传统的静态数据库形态, 数据流已经成为一种新兴且日趋重要的数据存在形式. 其典型特征包括[2~4]: a. 新数据不断产生, 数据总量潜在无限. b. 新数据产生的速度和时间间隔可能难以确定. c. 数据高速流动, 对其扫描次数仅限于单遍. d. 受多方面原因影响, 数据底层分布模型可能随时间发生变化. 对于这种蕴含大量信息的数据流, 人们迫切需要从中获取感兴趣的知识和规律, 同时新数据的不断产生和动态变化使得传统的数据挖掘方法难以取得良好效果, 因此对数据流进行挖掘和分析是一项具有重要实际意义且极富挑战的工作. 聚类分析作为一种能够探测数据类别结构的无监督学习方法, 在数据流的模式发现和知识挖掘中存在巨大潜力[5]. 对数据流进行聚类分析, 就是在对当前到达数据进行分簇的同时, 随着新数据的不断获得, 对聚类结果进行动态更新和调整, 以获取反映数据流内在结构的可靠聚类形态. 同时, 数据流增量更新和在线调节的特征也对聚类分析方法提出了新的要求和挑战[2].

研究人员针对数据流的聚类问题开展深入研究, 对静态数据集上的经典聚类方法进行了改进和拓展, 提出了一些全新的思路和解决方法. 在早期的数据流聚类研究中, 人们将数据流视为大数据集的特殊情况, 假定数据流整体服从某一未知分布, 将数据流聚类限定为单遍扫描获得数据集的聚类结果,

引用格式: 杜航原, 王文剑, 白亮. 一种基于优化模型的演化数据流聚类方法. 中国科学: 信息科学, 2014, 44: 1–?, doi: xxxxxxxx

将传统聚类方法应用于数据流, 相继提出了BIRCH 算法、STREAM聚类算法、Incremental K-Means算法、数据流网格聚类、数据流层次聚类等方法[1, 6~9]. 而事实上, 数据流中蕴含的模型和结构可能处于不断变化中, 对这些变化进行分析和了解能帮助我们对事物发展的客观规律有更好的认识. 针对这一需要, 有学者提出了演化数据流聚类, 认为数据流中隐藏的模型是动态变化的, 利用时间窗口模型对数据流进行分段处理. 例如, CluStream[10] 将演化聚类分为在线聚类和离线分析两个步骤, 在线聚类对不断到达的数据维护微簇结构, 并保存在不同时间粒度的金字塔时间结构中, 离线分析阶段生成特定时间段的聚类结果. SWClustering 算法[11]将指数柱状图(Exponential Histogram, EH)与时间聚类特征(temporal cluster features, TCF)结合, 提出了一种新的数据结构—Exponential Histogram of Cluster Features (EHCF), 利用EH处理簇内结构演化, 利用TCF表达类分布的变化. Cao等人[12]为了避免算法性能受到数据分布影响, 提出了一种基于衰减窗口模型的演化数据流密度聚类方法, 即DenStream算法, 能够支持包含多种复杂簇形状和噪声的演化聚类. ClusTree算法[13]利用分层树(hierarchical tree) 结构维护加权CF向量, 能随时中断向树结构中添加新目标并输出聚类结果, 无须对聚类模型的规模进行适当假设, 通过合并和分割操作对模型规模进行自动控制, 该方法还具有处理快速或慢速数据流的自我调节能力. Zhang 等人[14, 15] 将仿射传播（affinity propagation, AP）聚类应用于数据流, 提出了StrAP算法. 该方法引入分层AP(hierarchical AP, Hi-AP)概念将数据集随机划分为若干子集, 在每个子集上利用AP聚类算法产生一系列聚类中心(exemplars), 再对这些从子集中提取出的聚类中心构成的数据集进行加权AP(weighted AP, WAP)聚类, 进而形成新的聚类中心组. 该方法利用Hi-AP建立初始化聚类模型, 对于新到来的数据项, 通过计算与其最近的聚类中心的距离, 将距离小于阈值的数据项关联到该聚类中心并进行模型更新, 将距离大于阈值的数据项放入暂存池. 当暂存池规模达到一定程度后, 利用Page-Hinkley test 方法[16]实现模型重建.

相比静态数据, 数据流的生成模型或数据分布存在动态变化, 这将导致人们关心的聚类结构发生变化, 即概念漂移[17, 18]. 因而如何及时检测概念漂移的发生, 进而探究类结构演化趋势和事物发展内在规律是演化聚类的一个关键问题. 为此, 聚类方法必须在对不断流入的数据流生成簇结构的同时, 提供概念漂移检测能力, 以便及时更新聚类模型, 使聚类结果更具可用性和可解释性. 目前已有一些研究工作围绕概念漂移检测问题开展, 例如, 文献 [19]中作者使用信息熵分层树(Hierarchical Entropy Tree)结构维护聚类结构的信息熵特性, 并通过最优簇数量的变化检测概念漂移的发生. 文献 [20]基于粗糙集理论提出了一种概念间的距离度量, 并用来发现数据流的概念漂移. N Lu 等人[21]设计了一个基于事例的推理系统, 不再直接度量实例的实际分布, 而是引入了一种新的能力模型(competence model)通过检测能力变化中的差异来发现概念漂移. 这种基于能力的概念漂移检测方法无需示例分布的先验信息, 能获得可信度较高的检测结果, 并能对概念漂移做出具有可解释性的描述. 文献 [22]设计了一个动态系统模型(dynamical system modeling), 利用浸入定理(immersion theorem)将连续时间窗口中的数据观测展开到拓扑空间以表达数据观测间的时间关系(temporal relationship)和随时间变化情况, 进而实现概念漂移检测. D Liu 等人在文献 [23]中提出了一种遗忘参数极限学习机(forgetting parameters extreme learning machine, FP-ELM), 在对数据流进行增量在线处理时, 依据当前学习结果设置前一个数据块的遗忘参数, 以适应新的时间窗口下数据模型可能发生的变化.

现有的演化聚类方法中主要存在三个问题. 首先, 大多数方法是硬聚类, 即将目标严格划分到某一类别中. 而事实上, 现实世界中的很多对象由于本身具有模糊性和不确定性, 在类别划分时并不适合进行硬聚类, 而是表现出一定的中间性(intermediateness). 在处理这样的聚类问题时, 模糊集理论[24, 25]为我们提供了很好的工具. 第二, 在进行聚类时未考虑类模型与概念漂移间的相关性, 将聚类和概念漂移检测作为两个独立的过程处理, 忽略了聚类有效性对概念漂移检测的影响. 当新的数据流入时, 如

果现有聚类模型自身的有效性较差, 再依据这一模型判断类结构是否发生变化, 则概念漂移检测结果的有效性难以保证. 第三, 在聚类时, 仅通过某种相似性度量来为新流入的数据分配类别标签, 无法对聚类模型做出优化或调整, 从而难以获得最优的聚类结果. 针对这些问题, 本文提出了一种基于优化策略的最大熵数据流演化聚类方法, 利用模糊隶属度表达类别划分的模糊性, 通过信息熵描述类别划分的有效性, 在此基础上将聚类求解转化为一个优化问题, 依据优化问题的解进行概念漂移检测. 该方法能够保证获得有效的聚类结构, 进而避免忽略聚类有效性对概念漂移检测结果的影响, 使聚类过程和概念漂移检测同时达到最优.

本文的主要工作如下:

1) 我们提出了一种基于优化策略的模糊最大熵数据流演化聚类方法. 该方法利用模糊隶属度表达类别划分的模糊性, 利用信息熵描述类别划分的有效性, 并在此基础上定义了新的优化目标函数, 将聚类问题转化为一个优化问题. 当新的数据块到来时, 通过在约束条件下求解优化问题从而获得新的聚类模型, 这一模型能有效描述数据内在结构特征, 同时维持与前一个聚类模型的连续性. 基于这一策略, 推导了最大熵聚类优化问题的迭代求解方法;

2) 将数据流聚类和概念漂移检测进行整合, 依据优化问题的解进行概念漂移检测. 这样, 一方面能够保证获得有效的聚类结构, 进而避免忽略聚类有效性对概念漂移检测结果的影响, 使聚类过程和概念漂移检测同时达到最优; 另一方面, 由于聚类结果保持了模型变化的连续性, 概念漂移检测过程能更好的捕获聚类结构的变化趋势;

3) 在不同数据集上的实验结果表明, 我们提出的演化聚类方法在聚类精度和概念漂移检测精度等方面具有显著优势, 同时将算法的时间复杂度和空间复杂度维持在较低水平.

本文结构如下: 第2部分介绍了模糊最大熵聚类方法的原理. 第3部分描述了演化数据流的优化聚类模型和概念漂移检测机制. 在第4部分, 我们利用多种数据展示了提出模型的性能. 最后, 在第5部分给出结论, 并对未来工作进行展望.

2 最大熵聚类

最大熵聚类是一种重要的聚类算法, 能利用模糊隶属度获得比硬聚类更复杂的关于数据对象和所属簇间的关系, 且隶属度取值的连续性也更适合用来描述模糊簇间的边界. 该方法在FCM聚类[25]的基础上引入香农的信息量概念-即熵, 通过最大熵推理机制(Maximum-Entropy Inference, MEI)在给定任意信息的情况下获得隶属度的无偏估计[26, 27]. 对于静态数据集$\boldsymbol{X}=\{x_1,x_2,\ldots,x_N\}$, $N>1$为数据集包含的样本数量, 最大熵聚类的目标函数定义为:

$$\min_{W,C} J(\omega,c)=\sum_{l=1}^{k}\sum_{i=1}^{N}\omega_{li}\|x_i-c_l\|^2+\alpha^{-1}\sum_{l=1}^{k}\sum_{i=1}^{N}\omega_{li}\ln\omega_{li} \quad . \tag{1}$$

其中, $W=[\omega_{li}]_{k\times N}$为模糊隶属度矩阵, ω_{li}为对象x_i关于第l个簇的隶属度, c_l 表示第l 个簇的聚类中心, $C=[c_{lk}]$为聚类中心构成的聚类模型向量, $k>1$为划分簇的数量, α为大于0 的常数. 引入拉格朗日乘子, 利用EM 方法求得式(1)的局部最优解为:

给定$\widehat{\omega}_{li}$, 则

$$c_l = \frac{\sum\limits_{i=1}^{N} \widehat{\omega}_{li} x_i}{\sum\limits_{i=1}^{N} \widehat{\omega}_{li}} \quad . \tag{2}$$

给定$\widehat{c}_l$, 则

$$\omega_{li} = \frac{e^{-\alpha \|x_i - \widehat{c}_l\|^2}}{\sum\limits_{l=1}^{k} e^{-\alpha \|x_i - \widehat{c}_l\|^2}} \quad . \tag{3}$$

对于静态数据集, 通过有限次迭代, 算法能获得局部最优解, 这点已经在文献 [26]中得到证明.

3 基于最大熵的数据流演化聚类优化模型

3.1 数据流时间窗口处理框架

在对数据流进行聚类分析时, 为了避免过期的历史数据对聚类结果产生不利影响, 人们通常不直接对数据流中的所有数据进行处理, 而是只关心最近一段时间内数据流的分布状况. 为此, 滑动窗口技术被用来截获当前一段时间内的数据流记录, 成为数据流分析的主要工具之一. 定义$\boldsymbol{S} = \{\boldsymbol{S}^1, \boldsymbol{S}^2, \ldots, \boldsymbol{S}^T\}$是利用滑动窗口按照数据对象的到达时间对数据流$\boldsymbol{X}$ 的一个划分, 对于任意$1 \leqslant p \neq q \leqslant T$, 满足$\bigcup\limits_{p=1}^{T} \boldsymbol{S}^p = \boldsymbol{X}$和$\boldsymbol{S}^p \bigcap \boldsymbol{S}^q = \varnothing$. 通过滑动窗口的划分, 我们对当前窗口中的数据对象进行聚类分析和概念漂移检测, 将演化聚类任务分解为三个问题的解决:

- 对于第一个滑动窗口内的数据子集如何进行初始聚类, 以及对于已经判定为发生概念漂移的子集如何重构聚类模型?
- 对于最新流入的数据子集, 如何利用已形成的聚类模型进行聚类更新?
- 如何检测数据流中是否发生了概念漂移?

其中, 第一个问题可通过最大熵聚类方法进行求解. 在此基础上, 我们将对该方法进行扩展, 构建一个优化模型用于解决其余两个问题. 当新的数据子集流入时, 这一优化模型能提供有效的簇划分准则, 获得当前窗口数据蕴含的聚类结构, 并维持与上一个窗口中聚类模型的连续性. 同时, 将概念漂移检测问题集成到聚类优化模型中, 利用优化问题的解作为判断是否发生概念漂移的依据. 对于一个新到来的数据子集, 可能存在多种聚类模型, 这些不同的聚类模型将导致不同的概念漂移检测结果, 模型的有效性越优, 则以此做出的概念漂移检测结果可靠性也越高. 为此, 我们利用优化模型找到新数据子集的最优聚类结构, 若这一最优结构仍无法对数据子集进行有效的类结构划分, 则认为数据流中确实发生了概念漂移.

3.2 聚类优化模型

当新数据子集流入时, 考虑到数据流的连续性, 新的聚类模型应当在上一个模型的基础上更新而来, 为此在设计优化问题的目标函数时应当相比静态数据集的聚类考虑更多因素. 本文提出的目标函数综合考虑了新数据流入时聚类准则的物理意义和概念漂移检测的需要, 其中聚类准则将在本节进行描述, 概念漂移检测将在3.3节中进行讨论.

对于数据流第p个滑动窗口中的数据子集$\boldsymbol{S}^p$, 令x_i^p表示其中的第i个数据对象$(1 \leqslant i \leqslant |\boldsymbol{S}^p|)$, $W^p = [\omega_{li}^p]$是$\boldsymbol{S}^p$的模糊隶属度矩阵, $C^p = [c_l^p]$表示数据子集$\boldsymbol{S}^p$流入后形成的聚类模型, k^p表示C^p中簇的数量, C^{p-1}表示第$p-1$个（前一个）数据子集的聚类模型, 对$\boldsymbol{S}^p$而言这一模型是已知的. 当新的数据子集流入时, 如果未发生概念漂移, 则簇的数量不发生变化, 当前子集的聚类结果(C^p, W^p)可由上一个窗口内的聚类结果(C^{p-1}, W^{p-1})更新获得. 聚类优化目标函数定义如下:

$$M(W^p, C^p) = E(W^p, C^p) + D(C^p, C^{p-1}) \quad . \tag{4}$$

其中,

$$E(W^p, C^p) = \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l=1}^{k^p} \omega_{li}^p \|x_i^p - c_l^p\|^2 + \alpha^{-1} \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l=1}^{k^p} \omega_{li}^p \ln \omega_{li}^p \quad . \tag{5}$$

$$D(C^p, C^{p-1}) = \beta \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l}^{k^p} \omega_{li}^p \left\| c_l^p - c_l^{p-1} \right\|^2 \quad . \tag{6}$$

这一目标函数由两项构成, 在聚类优化过程中分别起到不同的作用:

- 第一项$E(W^p, C^p)$用来度量新窗口中数据子集到来时新的类结构模型概念描述的有效性, 其中α为大于0 的常数. 新的聚类模型由前一窗口中数据子集的聚类模型演化而来, 在未发生概念漂移的情况下, 保持其簇数量不变. 若仅使用这一项作为目标函数, 则退化为静态数据集下的最大熵聚类. 为此, 我们加入第二项作为对前后邻接的两个窗口内数据子集内在结构连续性的表达.
- 第二项$D(C^p, C^{p-1})$用来度量新聚类模型和前一个窗口中聚类模型间的差异性, 其中参数$\beta > 0$用于调节目标函数最小化过程中第二项所占权重. 在未出现概念漂移的情况下, 前一个窗口中的聚类模型应该能较好反映新数据子集的内在结构, 即类模型在演化过程中存在连续性. $D(C^p, C^{p-1})$的值越小, 表明两个模型间的差异越小, 前一个窗口中的聚类模型对新数据子集的类结构描述越准确; 而$D(C^p, C^{p-1})$的值越大, 说明前后两个数据子集的类结构之间差异较大, 即旧模型不再适合于描述新数据子集的类结构, 可能出现了概念漂移.

将上述两项进行整合构建新的优化目标函数, 能够同时考虑聚类模型描述的有效性和模型演化的连续性, 我们将新窗口中数据子集的聚类问题转化为如下的优化问题:

$$\min_{W^p, C^p} M(W^p, C^p), \text{ 满足} \begin{cases} \omega_l^p \in [0,1], \sum_{l=1}^{k^p} \omega_{li}^p = 1 \\ 0 < \sum_{i=1}^{N} \omega_{li} < |S^p| \end{cases} \quad . \tag{7}$$

对于这一优化问题, 可以使用EM方法迭代求解.

E阶段: 给定$C^p = \widehat{C}^p$, 求解$\min\limits_{W^p} M\left(W^p, \widehat{C}^p\right)$. 引入拉格朗日乘子$\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_{|\boldsymbol{S}^p|}]^T$, 得到

$$\begin{aligned} G(W^p, \Lambda) &= \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l=1}^{k^p} \omega_{li}^p \left\| x_i^p - \widehat{c}_l^p \right\|^2 + \alpha^{-1} \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l=1}^{k^p} \omega_{li}^p \ln \omega_{li}^p + \beta \sum_{i=1}^{|\boldsymbol{S}^p|} \sum_{l}^{k^p} \omega_{li}^p \left\| \widehat{c}_l^p - c_l^{p-1} \right\|^2 + \sum_{i=1}^{|\boldsymbol{S}^p|} \lambda_i \left(\sum_{l=1}^{k^p} \omega_{li}^p - 1 \right) \\ &= \sum_{i=1}^{|\boldsymbol{S}^p|} G_i(W^p, \lambda_i) \end{aligned} \quad . \tag{8}$$

其中

$$G_i(W^p, \lambda_i) = \sum_{l=1}^{k^p} \omega_{li}^p \left\| x_i^p - \widehat{c}_l^p \right\|^2 + \alpha^{-1} \sum_{l=1}^{k^p} \omega_{li}^p \ln \omega_{li}^p + \beta \sum_{l}^{k^p} \omega_{li}^p \left\| \widehat{c}_l^p - c_l^{p-1} \right\|^2 + \lambda_i \left(\sum_{l=1}^{k^p} \omega_{li}^p - 1 \right) \quad . \tag{9}$$

即式(8)的优化问题可分解为$|\boldsymbol{S}^p|$个独立的式(9)表示的子优化问题. 令$\frac{\partial}{\partial \omega_{li}^p} G_i\left(W^p, \lambda_i\right)=0$, 求得

$$\omega_{li}^p=\exp\left[-\alpha\left(\left\|x_i^p-\widehat{c}_l^p\right\|^2+\beta\left\|\widehat{c}_l^p-c_l^{p-1}\right\|^2\right)\right]\exp\left(-1-\alpha\lambda\right) \quad . \tag{10}$$

由约束条件$\sum_{l=1}^{k^p}\omega_{li}^p=1$得

$$\sum_{l=1}^{k^p}\exp\left[-\alpha\left(\left\|x_i^p-\widehat{c}_l^p\right\|^2+\beta\left\|\widehat{c}_l^p-c_l^{p-1}\right\|^2\right)\right]\exp\left(-1-\alpha\lambda\right)=1 \quad . \tag{11}$$

即

$$\exp\left(-1-\alpha\lambda\right) \;=\frac{1}{\sum_{l=1}^{k^p}\exp\left[-\alpha\left(\left\|x_i^p-\widehat{c}_l^p\right\|^2+\beta\left\|\widehat{c}_l^p-c_l^{p-1}\right\|^2\right)\right]} \quad . \tag{12}$$

将式(12)带回式(10)得

$$\omega_{li}^p=\frac{\exp\left[-\alpha\left(\left\|x_i^p-\widehat{c}_l^p\right\|^2+\beta\left\|\widehat{c}_l^p-c_l^{p-1}\right\|^2\right)\right]}{\sum_{l=1}^{k^p}\exp\left[-\alpha\left(\left\|x_i^p-\widehat{c}_l^p\right\|^2+\beta\left\|\widehat{c}_l^p-c_l^{p-1}\right\|^2\right)\right]} \quad . \tag{13}$$

M阶段: 给定$W^p=\widehat{W}^p$, 求解$\min\limits_{C^p} M\left(\widehat{W}^p, C^p\right)$. 令$\frac{\partial}{\partial c_l^p} M\left(\widehat{W}^p, C^p\right)=0$, 求得

$$c_l^p=\frac{\sum_{i=1}^{|S^p|}\widehat{\omega}_{li}^p\left(x_i^p+c_l^{p-1}\right)}{(1+\beta)\sum_{i=1}^{|S^p|}\widehat{\omega}_{li}^p} \quad . \tag{14}$$

当新数据子集流入时, 可以通过迭代计算式(13)和式(14)获得新子集的聚类结果. 算法的时间复杂度为$O\left(|\boldsymbol{S}^p|\, k^p t^p\right)$, 其中$k^p$和$t^p$分别为数据子集$\boldsymbol{S}^p$的簇划分数量和获得最优划分需要的迭代次数. 在对数据子集$\boldsymbol{S}^p$的聚类优化过程中需要$O\left(|\boldsymbol{S}^p|+|\boldsymbol{S}^p|\, k^p+2k^p\right)$的存储空间用于分别存放数据子集$\boldsymbol{S}^p$中的对象, 模糊隶属矩阵$W^p$, 以及相邻两个窗口内数据的聚类模型$C^{p-1}$和$C^p$.

3.3 概念漂移检测机制

当新的数据子集到来时, 为了判断是否发生概念漂移, 我们需要对新旧聚类模型间的演化情况进行分析. 在式(7)的优化目标下求得的最优解同时满足两个条件: 第一, 聚类模型能对当前数据子集进行有效描述; 第二, 当前聚类模型能保证与前一个窗口内的数据子集聚类结构的最佳连续性. 如果这个最优的聚类模型仍然无法用于描述当前数据子集中蕴含的概念, 则说明数据流由子集$\boldsymbol{S}^{p-1}$向子集$\boldsymbol{S}^p$演化的过程中, 其类结构发生了较大的变化, 出现了概念漂移. 因此, 可以将式(4)取得最优解时的取值$M_{opt}\left(W^p, C^p\right)$用于概念漂移检测. 对于数据子集$\boldsymbol{S}^p$而言, 值$M_{opt}\left(W^p, C^p\right)$是使模型有效性和连续性同时最优的聚类结果对应的目标函数取值, 即极小值, 这一取值反映了最优聚类结果对当前数据子集类结构的描述能力. 考虑到数据子集规模的影响, 我们设定概念漂移阈值θ, 在获得每个时间窗口内的数据子集的聚类模型(W^p, C^p)后, 通过判断式(15)是否成立来检测是否出现概念漂移.

$$M_{opt}\left(W^p, C^p\right)/\left|\boldsymbol{S}^p\right|>\theta \quad . \tag{15}$$

当检测到概念漂移发生后, 我们需要对新到来的数据子集进行重新聚类, 此时该子集的聚类优化问题就转换成静态数据集的聚类问题, 利用传统的最大熵方法进行重聚类即可.

3.4 算法总体流程

将前文提出的聚类优化问题求解和概念漂移检测机制进行整合, 我们得到了数据流的演化聚类算法, 其总体实施流程如下:

算法 1 数据流演化聚类算法

Require: $\boldsymbol{X}, \alpha > 0, \beta > 0, \theta > 0$;
Ensure: 聚类模型$\{(W^1, C^1), (W^2, C^2), \cdots, (W^T, C^T)\}$, 漂移检测结果$\{M^1, M^2, \cdots, M^T\}$;
初始化: 估计数据子集$\boldsymbol{S}^1$的初始簇数量k^1, 利用式(2)和式(3)计算聚类模型 (W^1, C^1) ;
for $p = 2 : T$ **do**
  $k^p = k^{p-1}$;
  通过式(13)和式(14)计算 (W^p, C^p);
  $M_{opt} = M(W^p, C^p)$;
  if $M_{opt}/|\boldsymbol{S}^p| > \theta$ **then**
    重新计算数据子集$\boldsymbol{S}^p$的簇数量k^p;
    通过式(2)和式(3)重新计算聚类模型(W^p, C^p);
  end if
end for

算法中参数α, β, θ的取值依赖于具体数据集的领域知识, 难以设定通用的最优取值. 在未获得足够领域知识的情况下, 一般取$1/\alpha = \beta$, 表示信息熵和模型偏差所占权重相同. M取值可以通过一系列时间窗口内的数据进行测试, 如果未发生概念漂移, 连续时间窗口内数据子集聚类获得的M值应当比较接近, 因此, 若某个窗口内的M值相比其他窗口大得多, 则可利用该值确定θ 的取值.

算法的时间复杂度为$O\left(\sum_{p=1}^{T} |S^p|\, k^p t^p\right)$, 与数据流中的对象数量和簇数量线性相关. 当新的数据子集流入时, 我们可以删除过期数据, 只需要保存每个数据子集的聚类模型和当前滑动窗口中的数据对象. 这样, 算法的平均空间复杂度可降低至$O\left(N/T + \sum_{p=1}^{T} k^p\right)$, 其中$N$为数据流中对象的总数. 由此可见, 本文算法在对数据流进行演化聚类分析时, 其时间复杂度和空间复杂度可以控制在较低水平.

4 仿真及结果分析

在仿真实验中, 我们利用3个数据集对本文聚类算法的有效性进行了验证, 并与一些现有算法进行了比较, 包括:CluStream[3], StreamKM[1], Denstream[12], ClusTree[13], StrAP[14]以及Entropy based FCM[28]. 其中, CluStream算法的参数设定为: 初始化点数$InitNumber = |S^p|$, 微簇数量$q = 10 \times k$, 最大边界参数$t = 2$, 最小相关性标记$\delta = 300$; StreamKM算法中数据块规模$M = |S^p|$; DenStream 算法中参数设定为: $InitNumber = |S^p|$, 衰减因子$\lambda = 0.25$, 最大半径阈值$\varepsilon = 16$, 加权阈值$\mu = 10$, 离群阈值$\beta = 0.2$; ClusTree算法中参数设定为: 非叶节点包含的条目数为$M = 3$, 叶节点包含的条目数为$L = 1000$, 遗忘因子$\beta = 2$; StrAP算法中参数设定为: 初始化点数$InitNumber = |S^p|$, 离群阈值ε 设为初始化阶段数据点的平均距离, 暂存池最大规模为$M = 100$, Page Hinkley测试阈值为$\lambda = 0.01$; Entropy based FCM算法中加权指数$m = 2$, 概念漂移阈值$e_{th} = 0.25$; 本文算法中参数设置情况为: $\alpha = 2$,

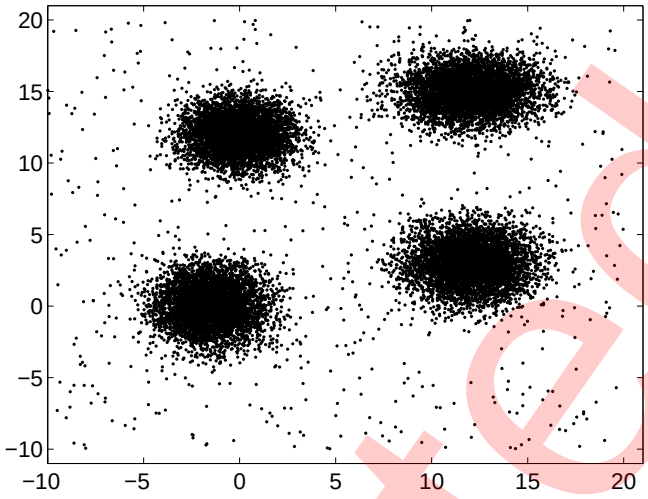


图 1　人工数据集分布情况

Figure 1　The distribution of the synthetic data stream

$\beta = 1/2$, θ=5. 实验使用的PC 配置为Intel Core i7-4790 3.60G Hz处理器和8GB内存, Windows 7 操作系统, 仿真程序由MATLAB R2012b编写和运行.

4.1　仿真及结果分析

为了对算法的概念漂移检测能力和聚类有效性进行评价, 在仿真实验中我们使用了1个人工数据集和2个真实数据集. 人工数据集由若干服从高斯分布的点集以及一些随机分布的噪声点共同组成, 共计20000个数据记录, 如图1所示. 真实数据集分别使用了来自UCI KDD Archive的KDD-CUP'99数据集和Forest-CoverType数据集. KDD-CUP'99数据集最早来源于MIT林肯实验室的一项入侵检测评估项目, 记录了9周时间内TCP网络连接和系统审计数据, 仿真各种不同的用户类型、网络流量和攻击手段. 这些原始数据包含两个部分: 约5,000,000条连接记录的训练集和2,000,000条记录的测试集. 每个连接记录包含42个属性, 其中34个数值型属性, 1个属性为类型标记. 这些连接记录被标记为正常（normal）或异常（attack）, 其中异常情况又包含4种攻击类型：DOS, R2L, U2R以及PROBING. 类似于文献 [10], 我们在仿真实验中使用了每条记录的34个数值型属性. Forest-CoverType 数据集来源于US Geological Survey (USGS) 和US Forest Service(USFS)对位于罗斯福国家森林的四片荒野区域的观测. 数据集中包含58,1012条记录, 这些记录最终被分为7种类型. 每条观测记录包含54个地质学和地理学属性, 其中10个数值型属性被用于仿真实验.

4.2　数据流聚类精度评价

为了对各种聚类算法的精度进行评价, 我们引入了3项评价指标[29]: a) Accuracy(AC), b) Precision(PE), c)Recall(RE). 对于数据集$\boldsymbol{X}$, 令N表示数据集中的对象数量, $C = \{C_1, C_2, \cdots C_k\}$表示某一聚类算法得到的聚类结果, $P = \{P_1, P_2, \cdots P_{k'}\}$表示$\boldsymbol{X}$中实际的类别划分, $N_{ij} = |C_i \cap P_j|$表示划分C_i和P_j中公共对象的数量, N_i为C_i中的对象数量, N_j为P_j中的对象数量. 聚类精度评价指标定义如

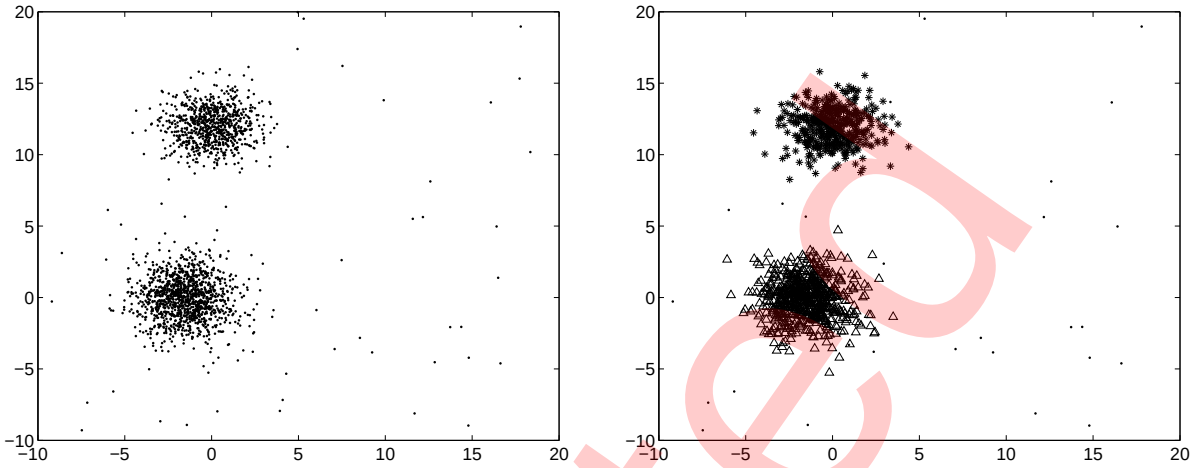


图 2　第4个滑动窗口中本文算法的聚类结果.

Figure 2　Proposed algorithm on synthetic dataset: 4th sliding window. (a) $\boldsymbol{S}^1 \cup \boldsymbol{S}^2 \cup \cdots \boldsymbol{S}^4$; (b) Clustering in $\boldsymbol{S}^4$

下:

$$AC = \frac{1}{N}\sum_{i=1}^{k}\max_{j=1}^{k'} N_{ij} \quad . \tag{16}$$

$$PE = \frac{1}{k}\sum_{i=1}^{k}\frac{\max_{j=1}^{k'} N_{ij}}{N_i} \quad . \tag{17}$$

$$RE = \frac{1}{k}\sum_{i=1}^{k}\frac{\max_{j=1}^{k} N_{ij}}{N_j} \quad . \tag{18}$$

4.2.1　可视化聚类结果

我们首先将人造数据集分割为40个数据子集, 每个子集由500个样本点构成, 这些样本点抽取自不同的类别. 为这些子集分配先后顺序, 模拟数据流进入时间窗口的过程, 每个子集对应一个时间窗口. 利用本文方法对数据流进行聚类分析, 图2˜ 图5给出了第4个、第8个、第25个以及第40个子集到达时数据流样本分布情况及各滑动窗口内的演化聚类结果. 图2 显示了第4个子集到达后数据流中的样本分布情况(图2a), 以及对第4个子集进行聚类的结果(图2b). 可以看出, 本文算法能够较好的识别出类结构. 在图3中, 数据流中出现了新的类(3a), 导致数据子集内的类结构发生了变化, 本文算法能有效检测这种概念漂移, 识别新的类结构(3b). 在第25个滑动窗口中, 子集$\boldsymbol{S}^{25}$中出现了新的数据分布类型(4a), 原有类结构模型不再适用, 本文方法能够有效发现这一变化趋势, 通过对窗口内样本进行重聚类建立新的类结构模型(4b). 图5给出了最后一个滑动窗口内的聚类结构(图5a), 以及整个数据流的聚类结果(图5b). 特别是通过对比图5b 和图1, 可以看出分布在数据空间中的噪声被很好的识别出来.

图6绘制了上述仿真中本文算法在每个滑动窗口中获得的聚类精度. 由图可知, 在整个数据流的40个数据子集中, 本文算法获得的聚类精度整体处于较高水平, 只有在处理第8个和第25个数据子集时聚类

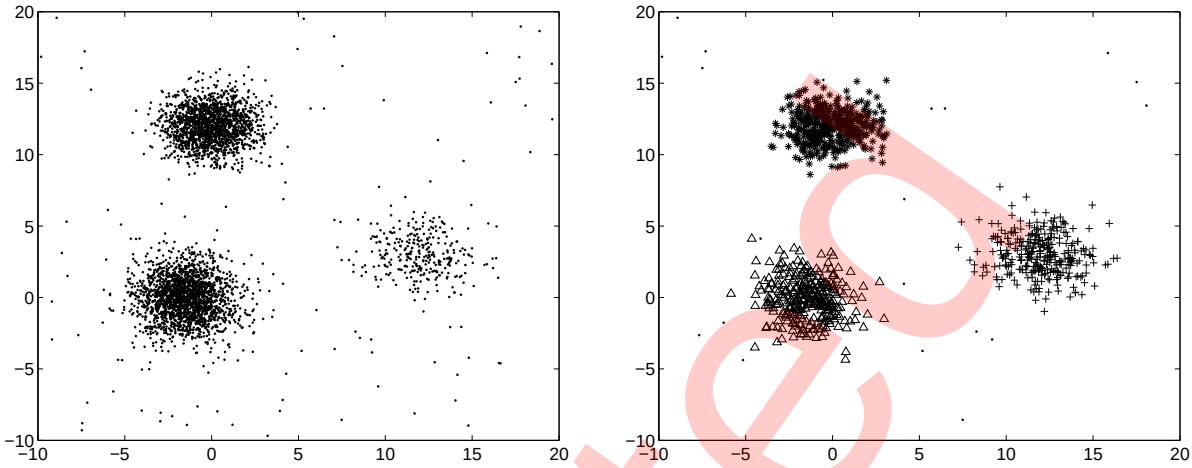


图 3　第8个滑动窗口中本文算法的聚类结果.

Figure 3　Proposed algorithm on synthetic dataset: 8th sliding window. (a) $\boldsymbol{S}^1 \cup \boldsymbol{S}^2 \cup \cdots \boldsymbol{S}^8$; (b) Clustering in $\boldsymbol{S}^8$

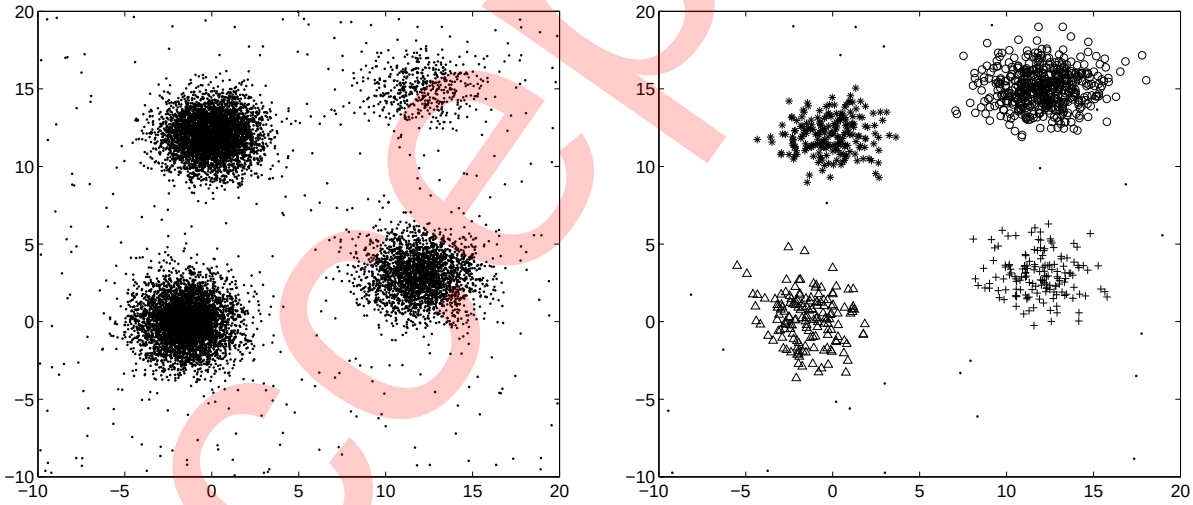


图 4　第25个滑动窗口中本文算法的聚类结果.

Figure 4　Proposed algorithm on synthetic dataset: 4th sliding window. (a) $\boldsymbol{S}^1 \cup \boldsymbol{S}^2 \cup \cdots \boldsymbol{S}^{25}$; (b) Clustering in $\boldsymbol{S}^{25}$

精度有一定程度的降低, 这是因为在这两个子集中出现了新的聚类结构, 一些分布稀疏的对象在类别划分时容易受到噪声影响.

4.2.2　聚类精度比较

在此, 我们分别利用人造数据集和真实数据集对各种算法的聚类精度进行了综合比较. 表1给出了这些算法在人工数据集上对滑动窗口内数据聚类结果的平均聚类精度, 由这些结果可以看出, 在噪声的干扰下, CluStream和StreamKM算法的聚类精度受到一定影响, ClusTree, StrAP和DenStream算

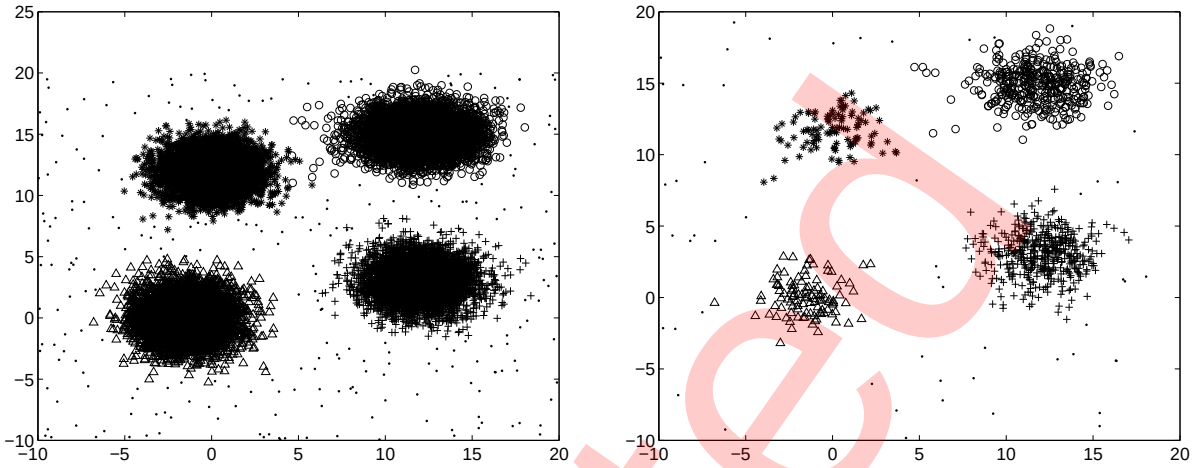


图 5　第40个滑动窗口中本文算法的聚类结果.

Figure 5　Proposed algorithm on synthetic dataset: 40th sliding window. (a) Clustering in $\boldsymbol{S}^{40}$; (b) The overall clustering results

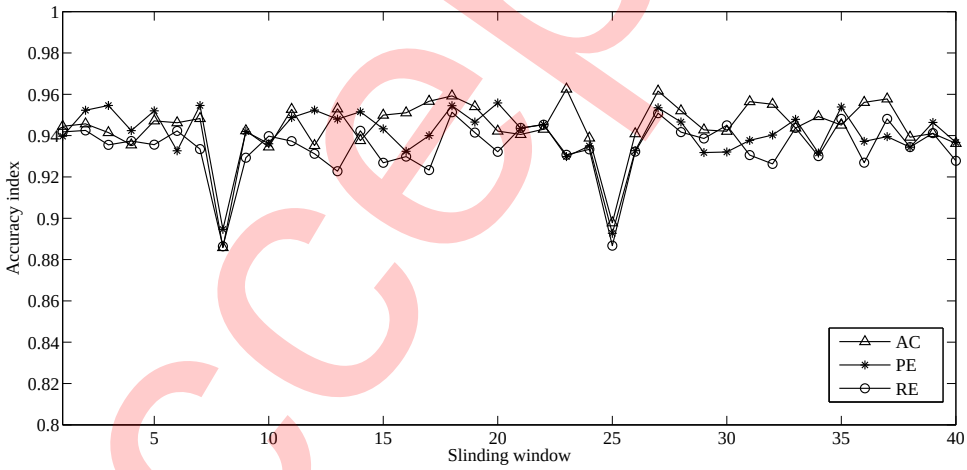


图 6　本文算法在人造数据集上每个滑动窗口中获得的聚类精度

Figure 6　Clustering accuracy on each sliding window for the synthetic data set

法通过各自的离群检测策略去除噪声点, 有利于维护数据流中的潜在微簇, 但这些方法对噪声点的去除是非在线的, 一些过期的历史微簇仍被维护, 只有在模型重建时才对微簇进行聚类合并, 导致错误合并的发生, 因此聚类精度仍然无法达到满意效果. Entropy based FCM 和本文算法利用模糊隶属度通过迭代优化获得最佳模糊划分, 在一定程度上削弱了噪声影响. 同时, 本文算法获得了最优的聚类精度, 这主要是由于一方面算法引入信息熵约束, 使形成的簇结构具有更高的可靠性, 另一方面将类别划分和概念漂移检测融合在同一优化目标中, 有利于识别新的簇结构.

为了比较各种算法在真实数据集上的聚类精度, 我们从KDD-CUP'99和Forest-CoverType两个数据集中分别抽取8个数据子集, 每个子集由具有不同类别标签的1000个对象构成, 数据子集的具体构成

表 1　不同算法在人造数据集上的平均聚类精度

Table 1　The results of different clustering methods on the synthetic data set

Index	CluStream	StreamKM	DenStream	ClusTree	StrAP	Entropy based FCM	Our algorithm
AC	0.8874	0.9085	0.9281	0.9182	0.9122	0.9285	0.9441
PE	0.8946	0.8924	0.9332	0.9116	0.9053	0.9306	0.9406
RE	0.9045	0.8938	0.9147	0.9269	0.9164	0.9298	0.9342

表 2　数据子集的具体构成情况

Table 2　The construction of each subset sampled from real-world data

Category label	Subset1	Subset2	Subset3	Subset4	Subset5	Subset6	Subset7	Subset8	Subset9	Subset10
Class1	200	100	100	400	300	300	500	200	300	300
Class2	200	100	200	200	300	100	100	100	400	200
Class3	200	200	300	100	200	300	200	400	100	200
Class4	200	300	100	100	100	200	100	200	100	100
Class5	200	300	300	200	100	100	100	100	100	200

情况如表2所示, 表中的5个类别标签对于KDD-CUP'99数据集是指正常连接状态和异常连接状态的4种攻击类型; 对于Forest-CoverType数据集则是从7种森林覆盖类中型选取的5个类别标签. 在对这些子集进行聚类时, 我们首先选取其中一个子集作为前一窗口(last window), 并假设其类分布情况已知, 从其余子集中轮流选取一个作为当前窗口(new window), 假设其类分布情况未知. 分别利用各种算法依据前一窗口中的类分布情况对新窗口内数据进行聚类, 聚类结果的平均精度如表3和表4所示. 通过比较可知, 本文方法在前一窗口的不同类分布情况下获得的新窗口内的聚类结果在精度和稳定性方面都优于其他算法.

进一步的, 我们由KDD-CUP'99和Forest-CoverType数据集分别随机抽取数据生成10组数据流, 每组数据流被20个滑动窗口划分为一系列数据子集, 每个数据子集具有随机类结构分布. 当数据子集的规模分别为1000, 3000和5000时, 我们比较了几种聚类算法在这些数据流上的平均聚类精度, 比较结果如表5所示. 这些结果表明, 相比其他数据流聚类方法, 新算法的聚类精度和稳定性在不同尺度的滑动窗口上都有所提高.

4.3　概念漂移检测精度评价

在对数据流进行演化聚类分析时, 能否及时准确的检测数据流中出现的概念漂移现象是评价聚类方法的重要性能指标之一, 在此我们利用KDD-CUP'99数据集对各种数据流聚类算法的概念漂移检测精度进行比较. 利用滑动窗口将数据集划分成一系列具有随机类结构分布的数据子集, 分别令子集的规模为1000, 3000和5000以组成不同的数据流. 在KDD-CUP'99数据集中, 所有的网络连接被分为"正常"和" 受攻击"两种类型, 当网络连接由一种类型转变为另一种类型时, 就意味着出现了概念漂移. 给定一个滑动窗口后, 可以利用类别标签定义概念漂移, 我们认为如果一个窗口内相比上一个窗口有超过10%的对象类别属性发生了变化, 表明这两个窗口间出现了概念漂移. 定义向量$Dx = [dx_1, dx_2, \cdots, dx_T]$记录数据流中实际的概念漂移状态, 其中$dx_p = 1$表示第$p$个滑动窗口发生了概念漂移, 否则$dx_p = 0$，$1 < p < T$. 令$Dx' = [dx'_1, dx'_2, \cdots, dx'_T]$表示通过概念漂移检测机制获得的各窗口

表 3　不同算法在KDD-CUP'99数据集上获得的聚类结果

Table 3　The clustering results of different algorithms on the KDD-CUP'99 data set

The last window	Index	CluStream	StreamKM	DenStream	ClusTree	StrAP	Entropy based FCM	Our algorithm
Subset1	AC	0.8525	0.8461	0.8733	0.8645	0.8856	0.8671	0.9133
	PE	0.8328	0.8547	0.8664	0.8629	0.8748	0.8563	0.9241
	RE	0.8263	0.8484	0.8653	0.8454	0.8734	0.8514	0.9177
Subset2	AC	0.8247	0.8236	0.8714	0.8475	0.8503	0.8570	0.9033
	PE	0.8016	0.8277	0.8553	0.8338	0.8480	0.8322	0.8947
	RE	0.8172	0.8339	0.8628	0.8504	0.8331	0.8438	0.9035
Subset3	AC	0.7277	0.7174	0.8326	0.8220	0.8464	0.8346	0.9066
	PE	0.7164	0.7452	0.8406	0.8347	0.8206	0.8421	0.9127
	RE	0.7253	0.7283	0.8343	0.8392	0.8311	0.8384	0.8936
Subset4	AC	0.8065	0.8159	0.8182	0.8166	0.8330	0.7092	0.9076
	PE	0.8272	0.8146	0.8253	0.8243	0.8265	0.6957	0.8918
	RE	0.7943	0.8060	0.8207	0.8307	0.8223	0.7144	0.8960
Subset5	AC	0.8166	0.8023	0.8346	0.8217	0.8356	0.8296	0.9012
	PE	0.8227	0.8275	0.8105	0.8255	0.8344	0.8376	0.9044
	RE	0.8181	0.8282	0.8331	0.8274	0.8295	0.8339	0.9057
Subset6	AC	0.6514	0.6918	0.8013	0.8244	0.8692	0.8135	0.9125
	PE	0.7022	0.6742	0.8252	0.8657	0.8506	0.8324	0.9073
	RE	0.6758	0.7065	0.8546	0.8114	0.8342	0.8316	0.9088
Subset7	AC	0.8123	0.7973	0.8103	0.8562	0.8801	0.7059	0.8942
	PE	0.8246	0.8055	0.8220	0.8666	0.8533	0.6993	0.8975
	RE	0.8060	0.8076	0.8048	0.8445	0.8595	0.6985	0.9011
Subset8	AC	0.7024	0.7262	0.8365	0.8571	0.8340	0.8146	0.9023
	PE	0.7156	0.7004	0.8254	0.8055	0.8172	0.8243	0.9102
	RE	0.7007	0.6995	0.8332	0.8229	0.8108	0.8212	0.9088
Subset9	AC	0.8317	0.8248	0.8766	0.8298	0.8007	0.7135	0.8964
	PE	0.8484	0.8127	0.8578	0.8554	0.8284	0.7267	0.9034
	RE	0.8206	0.8557	0.8446	0.8204	0.8216	0.7091	0.8992
Subset10	AC	0.8247	0.8236	0.8786	0.8714	0.9035	0.8442	0.9130
	PE	0.8016	0.8277	0.8895	0.8662	0.8976	0.8476	0.9226
	RE	0.8172	0.8339	0.8842	0.8685	0.8866	0.8336	0.9114

内的漂移情况. 在此引入3个指标对数据流聚类方法的概念漂移检测精度进行评价: 准确度(precision, PD), 召回率(recall, RD)以及偏差距离(Euclidean distance, ED). 这些指标定义如下:

$$PD = \frac{|\{dx_p == 1 \wedge dx'_p == 1, 1 \leqslant p \leqslant T\}|}{|\{dx'_p == 1, 1 \leqslant p \leqslant T\}|} \quad . \tag{19}$$

$$RD = \frac{|\{dx_p == 1 \wedge dx'_p == 1, 1 \leqslant p \leqslant T\}|}{|\{dx_p == 1, 1 \leqslant p \leqslant T\}|} \quad . \tag{20}$$

表 4　不同算法在**Forest-CoverType**数据集上获得的聚类结果

Table 4　The clustering results of different algorithms on the Forest-CoverType data set

The last window	Index	CluStream	StreamKM	DenStream	ClusTree	StrAP	Entropy based FCM	Our algorithm
Subset1	AC	0.8642	0.8556	0.8703	0.8664	0.8684	0.8572	0.9025
	PE	0.8836	0.8541	0.8552	0.8358	0.8729	0.8660	0.9148
	RE	0.8855	0.8632	0.8686	0.8451	0.8653	0.8534	0.9112
Subset2	AC	0.8306	0.8381	0.8463	0.8774	0.8805	0.8348	0.8974
	PE	0.8577	0.8106	0.8331	0.8426	0.8430	0.8556	0.9025
	RE	0.8742	0.8227	0.8276	0.8514	0.8442	0.8283	0.9011
Subset3	AC	0.8246	0.8252	0.8264	0.8106	0.8546	0.8152	0.8940
	PE	0.7859	08411	0.8361	0.8368	0.8353	0.8364	0.8962
	RE	0.8114	0.8243	0.8221	0.8005	0.8471	0.8377	0.9077
Subset4	AC	0.7556	0.7431	0.8033	0.8332	0.8341	0.7621	0.9034
	PE	0.7187	0.7457	0.8216	0.8140	0.8218	0.7559	0.8973
	RE	0.7228	0.7264	0.8113	0.8225	0.8115	0.7415	0.8864
Subset5	AC	0.8341	0.8351	0.8678	0.8557	0.8641	0.8253	0.9112
	PE	0.8205	0.8146	0.8842	0.8622	0.8850	0.8411	0.8978
	RE	0.8332	0.8334	0.8635	0.8787	0.8746	0.8127	0.8741
Subset6	AC	0.8210	0.8314	0.8712	0.8746	0.8841	0.8304	0.9072
	PE	0.8452	0.8082	0.8736	0.8663	0.8734	0.8220	0.9115
	RE	0.7966	0.8140	0.8559	0.8702	0.8833	0.7857	0.8996
Subset7	AC	0.7159	0.6742	0.7936	0.8354	0.7884	0.7026	0.8831
	PE	0.6687	0.7018	0.8014	0.7944	0.8021	0.6775	0.9004
	RE	0.6934	0.7229	0.7993	0.8326	0.7935	0.6812	0.8918
Subset8	AC	0.8475	0.8417	0.8438	0.8559	0.8414	0.8361	0.9155
	PE	0.8352	0.8345	0.8330	0.8673	0.8378	0.8544	0.9037
	RE	0.8068	0.8441	0.8541	0.8509	0.8558	0.8272	0.8977
Subset9	AC	0.7463	0.7229	0.8245	0.7936	0.8267	0.7145	0.9011
	PE	0.7170	0.7552	0.8309	0.8177	0.8335	0.7433	0.9008
	RE	0.6524	0.7018	0.8262	0.8006	0.8064	0.7706	0.8944
Subset10	AC	0.8046	0.8213	0.8664	0.8265	0.8768	0.8391	0.9140
	PE	0.7559	0.8176	0.8413	0.8445	0.8712	0.8464	0.9039
	RE	0.8211	0.8224	0.8476	0.8406	0.8553	0.8017	0.8988

$$ED = \sqrt{\|Dx - Dx'\|^2} \quad . \tag{21}$$

其中指标PD和RD来自文献 [30], 它们的值越大表示概念漂移监测机制的性能越优. ED用于计算数据流的实际概念漂移情况和聚类算法检测到的概念漂移情况间的差异性, ED的值越小表明检测到的概念漂移情况越接近数据流真实情况. 对数据流划分的簇数量取不同值时, 各算法对数据流的概念漂移检测结果如表6所示. 首先设置簇数量为$k = 2$, 当数据窗口内的子集规模为1000 时, 数据流中

表 5　不同算法在真实数据形成的数据流上获得的聚类结果

Table 5　The clustering results of different algorithms on the data stream sampled from real-world data

Data set	Subset size	Index	CluStream	StreamKM	DenStream	ClusTree	StrAP	Entropy based FCM	Our algorithm
KDD-CUP'99	1000	AC	0.8133	0.7956	0.8542	0.8401	0.8548	0.8174	0.8846
		PE	0.8250	0.8116	0.8557	0.8336	0.8631	0.8028	0.8933
		RE	0.7143	0.7652	0.8361	0.8475	0.8442	0.7734	0.8641
	3000	AC	0.8362	0.8268	0.8630	0.8362	0.8652	0.8257	0.9013
		PE	0.8445	0.8413	0.8441	0.8573	0.8546	0.8556	0.9120
		RE	0.8233	0.8030	0.8572	0.8442	0.8471	0.8393	0.9181
	5000	AC	0.7819	0.7647	0.8064	0.8774	0.8637	0.8186	0.8974
		PE	0.8176	0.8207	0.8144	0.8746	0.8822	0.8335	0.9063
		RE	0.7484	0.7903	0.8132	0.8821	0.8506	0.7840	0.8848
Forest-Cover Type	1000	AC	0.8362	0.8485	0.8342	0.8475	0.8646	0.8369	0.9042
		PE	0.8501	0.8339	0.8557	0.8395	0.8638	0.8552	0.9117
		RE	0.7214	0.7864	0.8476	0.8477	0.8584	0.8004	0.8985
	3000	AC	0.8551	0.8530	0.8872	0.8662	0.8763	0.8464	0.9133
		PE	0.8468	0.8671	0.8805	0.8743	0.8671	0.8516	0.9253
		RE	0.7704	0.8269	0.8623	0.8777	0.8796	0.8284	0.9070
	5000	AC	0.8436	0.8224	0.8401	0.8456	0.8862	0.8260	0.9176
		PE	0.8145	0.8345	0.8352	0.8379	0.8817	0.8411	0.9144
		RE	0.7551	0.7988	0.8445	0.8443	0.8635	0.8164	0.9092

有38个窗口发生了概念漂移, 本文方法能够正确识别出其中的31个, 同时也误判了13个实际未发生概念漂移的窗口, 其ED 值处于较低水平. 其余数据流聚类算法能够检测出部分发生概念漂移的窗口, 同时也存在较多的误判窗口. 通过比较可知, 本文方法获得了最优的概念漂移检测精度. 当数据窗口内的数据子集规模为3000和5000时, 本文方法能够检测出所有真实发生的概念漂移, 而其他算法仍然只能检测到部分发生概念漂移的窗口, 随着数据子集规模的增大这些算法的ED值逐渐减小. 通过比较PD, ED和RD值, 可以看出在这些窗口规模下, 本文算法的概念漂移检测精度优于其他算法. 当簇数量分别取$k = 5, 10, 20$时, CluStream, StreamKM以及ClusTree算法仍然存在较多的误判窗口, 这主要是由于这些算法依靠簇规模的变化作为检测概念漂移的标准, 而事实上, 在很多情况下簇规模发生变化并不代表出现了概念漂移. DenStream与StrAP算法通过离群点检测策略实现概念漂移检测, 其概念漂移检测精度相比上述方法有所提高. Entropy based FCM通过度量信息熵的大小判断数据流是否发生概念漂移, 克服了依靠簇规模进行检测的局限性, 使检测精度有所提高, 但是这一方法割裂了数据流演化的连续性, 对窗口内的聚类结果较为敏感, 仍然存在相当数量的误判窗口. 综合来看，在不同的簇数量和子集规模设定下, 本文算法都获得了最优的概念漂移检测结果.

4.4　算法计算量分析

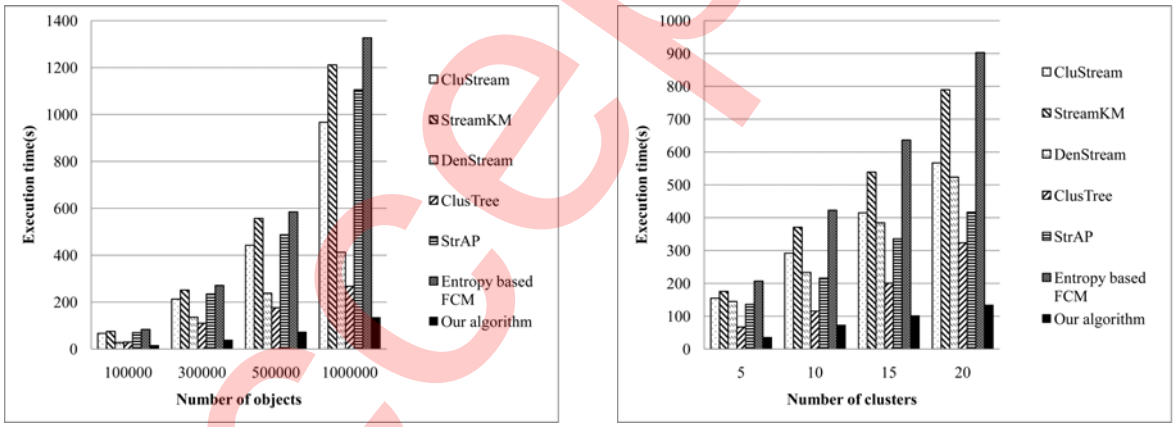
本节对各种算法的运行时间进行了比较, 这些聚类算法的运行时间主要包括2个部分: 构造和更新簇结构花费的时间, 以及为各对象分配类别标签花费的时间. 例如, 在CluStream算法中, 运行时间主要用于在线微聚类(micro-clustering)和离线宏聚类(macro-clustering) 两个部分; 在StreamKM算法中, 运

表 6　不同算法在KDD-CUP'99 数据流上的概念漂移检测结果

Table 6　The results of different methods for drifting detection on the KDD-CUP'99 data stream

Number of Clusters	Window size	Index	CluStream	StreamKM	DenStream	ClusTree	StrAP	Entropy based FCM	Our algorithm
		PD	20/62	19/53	25/45	24/50	29/48	21/47	31/44
	1000	RD	20/38	19/38	25/38	24/38	29/38	21/38	31/38
		ED	7.7460	7.2801	5.7446	6.3246	5.2915	6.5574	4.4721
		PD	17/50	15/36	20/41	19/40	20/38	18/33	25/31
k=2	3000	RD	17/25	15/25	20/25	19/25	20/25	18/25	25/25
		ED	6.4031	5.5678	5.0990	5.1962	4.7958	4.6904	2.4495
		PD	12/35	13/33	14/28	14/29	14/32	13/28	18/26
	5000	RD	12/18	13/18	14/18	13/18	14/18	13/18	18/18
		ED	5.3852	5.0000	4.2426	4.4721	4.6904	4.4721	2.6458
		PD	18/55	16/50	26/51	20/48	19/45	18/44	30/39
	1000	RD	18/38	16/38	26/38	20/38	19/38	18/38	30/38
		ED	7.5498	7.4833	6.0828	6.7823	6.7082	6.7823	4.1231
		PD	16/45	15/38	17/32	16/40	20/39	16/30	23/29
k=5	3000	RD	16/25	15/25	17/25	16/25	20/25	16/25	23/25
		ED	6.1644	5.7446	4.7958	4.7958	4.8990	4.7958	3.0000
		PD	11/33	12/32	11/24	13/24	12/27	13/26	18/20
	5000	RD	11/18	12/18	11/18	13/18	12/18	13/18	18/18
		ED	5.3852	5.0990	4.4721	4.0000	4.5826	4.2426	1.4142
		PD	18/52	15/48	22/46	20/50	18/47	20/43	30/37
	1000	RD	18/38	15/38	22/38	20/38	18/38	20/38	30/38
		ED	7.3485	7.4833	6.2450	6.9282	7.0000	6.4031	2.8284
		PD	16/42	13/35	18/37	16/35	20/32	15/30	21/29
k=10	3000	RD	16/25	13/25	18/25	16/25	20/25	15/25	21/25
		ED	5.9161	4.9000	5.0990	5.2915	4.1231	5.0000	3.4641
		PD	11/27	11/26	12/25	12/26	14/27	11/25	17/20
	5000	RD	11/18	11/18	12/18	12/18	14/18	11/18	17/18
		ED	4.7958	4.6904	4.3589	4.4721	4.1231	4.5826.	2.0000
		PD	16/53	15/43	18/47	16/44	20/46	18/36	29/35
	1000	RD	16/38	15/38	18/38	16/38	20/38	18/38	29/38
		ED	7.6811	7.1414	7.0000	7.0711	6.6332	6.1644	3.8730
		PD	12/48	14/44	15/36	17/35	18/38	16/33	23/27
k=20	3000	RD	12/25	14/25	15/25	17/25	18/25	16/25	23/25
		ED	7.0000	6.4031	5.5678	5.0990	5.1962	5.0990	2.4495
		PD	13/45	13/38	14/26	13/30	14/24	14/27	16/20
	5000	RD	13/18	13/18	14/18	13/18	14/18	14/18	16/18
		ED	6.0828	5.4772	4.0000	4.6904	3.7417	4.1231	2.4495

行时间主要用于两个部分: 第一次遍历数据流时产生特定数量的簇中心, 以及第二次遍历数据流时将各对象归入相应的簇; 在StrAP和DenStream算法中, 运算时间主要用于建立初始簇以及将新的数据项分配到相应簇中或判定为离群点; 在Entropy based FCM和本文算法中, 运行时间主要用于迭代计算或更新簇中心, 以及获取各对象对簇的隶属度. 我们利用KDD-CUP'99数据集生成数据流, 假设数据流的每个滑动窗口内包含1000个对象, 并利用这些聚类方法进行处理. 每个算法重复运行10次, 取其平均运行时间作为最终结果, 比较结果如图7所示. 在算法相关参数给定的情况下, 算法的运行时间与数据流规模以及划分的簇数量相关. 图7(a)给出了k=2时各算法在不同规模数据流上的运行时间. 图7(b) 给出了数据流规模为N=100,000时各算法对应不同簇划分数量的运行时间. 由图可知, 随着数据流中对象总数或簇数量的增加, 这些算法的运行时间呈现出上升趋势, 且在相同条件下本文算法的运行时间明显低于其他算法. CluStream, StreamKM和Entropy based FCM算法在滑动窗口环境下需要在每个新对象到来时存储一次快照, 将其作为一个滑动窗口进行处理, 这带来了大量额外计算代价, 而且Entropy based FCM算法需要在每个滑动窗口中迭代计算簇中心和隶属度产生较大计算量, 花费的运行时间最长: StrAP和DenStream算法的离群点检测机制能在簇重构时清除部分过期信息, 一定程度上降低了计算量, 但由于非在线清除在处理过程中仍然保存了相当数量的过期历史数据项, 导致部分计算资源的浪费; ClusTree算法利用层次树结构维护数据流摘要信息, 通过简洁高效的自适应索引机制将计算量保持在较低水平; 本文算法利用滑动窗口将数据流划分为一系列子集, 针对每个新窗口中的数据子集进行迭代处理, 所需运行时间最少.



(a) (b)

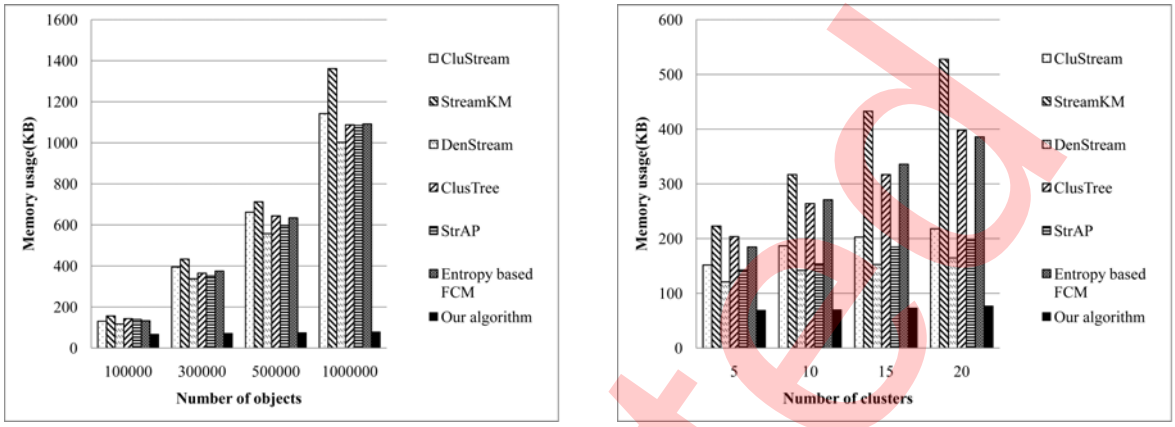
图 7 不同聚类方法的运行时间.

Execution times of different clustering algorithms. (a) Execution times of clustering algorithms with different data stream sizes; (b) Execution times of clustering algorithms with different cluster numbers

4.5 存储空间占用分析

我们还利用KDD-CUP'99数据集对几种算法运行时占用的存储空间进行了比较, 比较结果如图8所示. 图8(a)给出了簇划分数量k=2 时各算法对不同规模的数据流进行聚类时所占存储空间, 图8(b)给出了数据流规模为N=100,000时不同簇数量设定下各算法所占存储空间. 可以看出, 在不同条件下本文算法运行时使用的存储空间最少, 并且对数据流规模和簇数量的变化不敏感, 这得益于算法只需要

保存每个滑动窗口中的聚类模型和当前滑动窗口中的数据对象, 即算法所需存储空间主要与滑动窗口中数据子集的规模相关.



(a) (b)

图 8 不同聚类方法占用的存储空间.

Memory usages of different clustering algorithms. (a) Memory usages of clustering algorithms with different data stream sizes; (b) Memory usages of clustering algorithms with different cluster numbers.

5 结论

本文提出了一种数据流演化聚类方法, 该方法基于最大熵思想建立聚类问题的优化模型, 同时考察新的滑动窗口中聚类的有效性以及窗口间聚类模型的差异, 并在优化模型获得最优解的情况下进行概念漂移检测. 这一方法能有效描述数据内在结构特征, 维持滑动窗口间聚类模型的连续性, 以及避免有效性较低的聚类结果对概念漂移检测产生不良影响. 在实验分析中, 我们利用人造数据集和真实数据集对新算法的性能进行了全面的测试和分析, 实验结果表明该算法在聚类精度、概念漂移检测精度以及计算存储效率等方面具有显著优势. 同时应当注意的是, 该算法仍然存在一些问题需要解决, 例如, 簇中心和隶属度的迭代更新过程产生一定计算量, 可以尝试通过数据压缩或抽样方法进一步降低其计算量. 此外, 本文方法基于最大熵聚类, 作为一种类K-means方法, 其聚类性能容易受到簇分布形状的影响. 为解决这一问题, 下一步我们将尝试将聚类优化模型思想用于其他聚类方法, 例如支持向量聚类和图聚类等非线性聚类方法.

参考文献

1 Guha S, Meyerson A, Mishra N, Motwani R, O'Callaghan L. Clustering data streams: theory and practice. IEEE Transactions on Knowledge and Data Engineering, 2003, 15(3): 515–528

2 Khalilian M, Mustapha N. Data Stream Clustering: Challenges and Issues. In: Proceedings of the 18th International Conference of Engineers and Computer Scientists, Hong Kong, 2010. 546–549

3 Silva J A, Faria E R, Barros R C, Hruschka E R, Carvalho A C P L F, Gama J. Data Stream Clustering: A Survey. ACM Computing Surveys, 2014, 46(1): 125–134

4 Guha S, Mishra N. Data stream management. Springer Berlin Heidelberg, 2016, 15(3): 359–366

5 Xu R. Survey of clustering algorithms. IEEE Transactions on Neural Networks, 2005, 16(3): 645–678

6 Zhang T. BIRCH: an efficient data clustering method for very large databases. ACM Sigmod Record, 1999, 25(2): 103–114
7 Park N H, Lee W S. Statistical grid-based clustering over data streams. ACM Sigmod Record, 2004, 33(1): 32–37
8 Ordonez C. Clustering binary data streams with K-means. In: Proceedings of the 8th ACM Sigmod Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego: ACM, 2003. 12–19
9 Rodrigues P, Gama J, Pedroso J P. Hierarchical time-series clustering for data streams. In: Proceedings of the 1st International Workshop on Knowledge Discovery in Data Streams, 2004. 22–31
10 Aggarwal C G, Han J, Wang J, Yu P S. A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases, Berlin: Morgan Kaufmann, 2003, 81–92
11 Zhou A, Cao F, Qian W, Jin C. Tracking clusters in evolving data streams over sliding windows. Knowledge and Information Systems, 2008, 15(2):181–214
12 Cao F，Ester M，Qian W，Zhou A. Density-based clustering over an evolving data stream with noise. In: Proceedings of the 6th SIAM International Conference on Data Mining, Maryland: Bethesda, 2006. 328–339
13 Kranen P, Assent I, Baldauf C, Seidl T. The ClusTree: indexing micro-clusters for anytime stream mining. Knowledge & Information Systems, 2011, 29(2): 249–272
14 Zhang X, Furtlehner C, Perez J, Germain-Renaud C, Sebag M. Toward autonomic grids: analyzing the job flow with affinity streaming. In: Proceedings of the ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, Paris, 2009. 987–996
15 Zhang X, Furtlehner C, Germain-Renaud C, Sebag M. Data stream clustering with affinity propagation. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(7): 1644–1656
16 Hinkley D V. Inference about the change-point from cumulative sum tests. Biometrika, 1971, 58(3): 509–523.
17 Goncalves P M, Santos S G T D C, Barros R S M, Vieira D C L. A comparative study on concept drift detectors. Expert Systems with Applications, 2014, 41(18): 8144–8156
18 Faria E R, Goncalves I J, Carvalho A, Gama J. Novelty detection in data streams. Artificial Intelligence Review, 2016, 45(2): 235–269
19 Chen K, Liu L. HE-Tree: a framework for detecting changes in clustering structure for categorical data streams. Vldb Journal, 2009, 18(6):1241–1260
20 Cao F，Liang J，Bai L，Zhao X，Dang C. A framework for clustering categorical time-evolving data. IEEE Transactions on Fuzzy Systems, 2010, 18(5):872–882
21 Lu N, Zhang G, Lu J. Concept drift detection via competence models. Artificial Intelligence, 2014, 209(1):11–28
22 Costa F G D, Rios R A, Mello R F D. Using dynamical systems tools to detect concept drift in data streams. Expert Systems with Applications, 2016, 60:39–50
23 Liu D, Wu Y X, Jiang H. FP-ELM: An online sequential learning algorithm for dealing with concept drift. Neurocomputing, 2016, 207: 322–334.
24 Frigui H. Advances in Fuzzy Clustering and its Applications. New York: John Wiley & Sons, 2007. 112–130
25 Pal N R, Bezdek J C. On Cluster Validity for the Fuzzy c-Means Model. IEEE Transactions on Fuzzy Systems, 1995, 3(3): 370–379
26 Li R P, Mukaidono M. A maximum entropy approach to fuzzy clustering. In: Proceedings of the 4th IEEE International Conference on Fuzzy Systems, Yokohama, 1995, 2227 – -2232
27 Ren S J, Wang Y D. A proof of the convergence theorem of maximum-entropy clustering algorithm. Science China, 2010, 53(6): 1151–1158
28 Zhang B, Qin S, Wang W, Wang D, Xue L. Data stream clustering based on Fuzzy C-Mean algorithm and entropy theory. Signal Processing, 2015, 126: 111–116
29 Yang Y. An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1999, 1: 69–90
30 Chen H, Chen M, Lin S. Catching the trend: a framework for clustering concept-drifting categorical data. IEEE Transactions on Knowledge and Data Engineering, 2008, 21(5): 652–665

mization model

Hangyuan DU[1], Wenjian WANG[1,2*] & Liang Bai[2]

1 *School of Computer and Information Technology, Shanxi University, Taiyuan* 030006, *China*;
2 *Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan* 030006, *China*
*E-mail: wjwang@sxu.edu.cn

Abstract An optimization model based on the fuzzy maximum entropy method is proposed in this paper for the data stream evolving clustering problem. In the model, the fuzziness and effectiveness of cluster partition are described by fuzzy membership and information entropy respectively, based on what, the optimization object function is defined. In the sliding window, the clustering processing of data subset is construed as a optimization problem. In this way, the inner structural feature can be depicted effectively, and the continuity between contiguous windows is preserved simultaneously. The solution of the optimization problem is used as the basis of concept drift detection, as a result, the validity of detection result is guaranteed, and the variation trend of cluster structure can be easily captured. In the simulation, some artificial and real datasets are used to verify the performance of the proposed method, and some existing evolving clustering algorithms are introduced to compare with our algorithm for the testing purposes. The simulation results demonstrate the validity of the developed algorithm. Under the same conditions, the new method is superior to other clustering algorithms with respect to the accuracy of clustering and concept drift detection, as well as it saves the computational efforts and memory usages effectively.

Keywords data stream evolving clustering optimization model fuzzy membership information entropy

Hangyuan Du was born in 1985. He received the Ph.D. degree in navigation and control from the Harbin Engineering University, in 2012. Currently, he is a lecturer at Shanxi university. His main research interests include clustering analysis and complex network theory.

Liang Bai was born in 1982. He received the Ph.D. degree in computer science and technology from the Shanxi University, in 2012. Currently, he is a associate professor at Shanxi university. His main research interests include clustering analysis and complex network theory.

Wenjian Wang was born in 1968. She received the Ph.D. degree from the Xi'an Jiaotong University in 2004. Currently, she is a professor at Shanxi university. Her main research interests include computational intelligence, machine learning and machine vision. Professor Wenjian Wang is a Senior member of China Computer Federation.