一种混合属性数据流聚类算法

杨春宇 周 杰
（清华大学自动化系 北京 100084）

摘 要 数据流聚类是数据流挖掘中的重要问题.现实世界中的数据流往往同时具有连续属性和标称属性,但现有算法局限于仅处理其中一种属性,而对另一种采取简单舍弃的办法.目前还没有能在算法层次上进行混合属性数据流聚类的算法.文中提出了一种针对混合属性数据流的聚类算法;建立了数据流到达的泊松过程模型;用频度直方图对离散属性进行了描述;给出了混合属性条件下微聚类生成、更新、合并和删除算法.在公共数据集上的实验表明,文中提出的算法具有鲁棒的性能.

关键词 数据挖掘;数据流;聚类分析;混合属性;泊松过程
中图法分类号 TP311

A Heterogeneous Data Stream Clustering Algorithm

YANG Chun-Yu ZHOU Jie
(*Department of Automation*, *Tsinghua University*, *Beijing* 100084)

Abstract Data stream clustering is an important issue in data stream mining. Many real-world data streams have both continuous attributes and categorical attributes, which are usually called heterogeneous attributes. However, most of the existing stream mining algorithms can manipulate only continuous attributes or categorical attributes. To our best knowledge, there is no algorithm designed to manipulate heterogeneous attributes. Simply omitting categorical or continuous attributes may lose important information about the data stream and decrease the mining quality. This paper proposes a novel approach for clustering data stream with heterogeneous features and the Poisson Arrival model of the data stream, and gives the updating algorithm of the parameter of the process. Secondly it defines the histogram description of the discrete attributes in Micro Cluster and corresponding distance metric. Finally it proposes the framework describing the generation, evolution, merging and deletion of the Micro Clusters, and designs the detailed algorithms for each procedure. Experimental results on public data sets show that the proposed algorithm is robust.

Keywords data mining; data stream; clustering; heterogeneous attributes; Poisson process

1 引 言

近年来,随着计算机技术、通信技术以及网络技术的飞速发展,许多领域中出现了连续到达、持续增长、动态演化的数据——数据流.典型例子包括电信呼叫数据、股票交易数据、网站访问日志、互联网通信数据、搜索引擎数据、大型零售企业销售数据等等[1-2].文献[2]给出了一个描述性的数据流定义:数据流是指实时、连续、有序的数据序列,其中元素的出现顺序、速率与时刻均不可控制.

数据流管理与分析是数据挖掘研究领域的热点

收稿日期:2007-03-05;修改稿收到日期:2007-06-04.**杨春宇**,1982 年生,博士研究生,研究兴趣包括数据挖掘、机器学习等.E-mail: yangchunyu@mails.thu.edu.cn.**周 杰**,1968 年生,教授,博士生导师,研究领域包括模式识别、信息融合、图像处理和计算机视觉.

问题之一，其中研究如何从数据流中获取知识的数据流挖掘更是获得了广泛的关注[3-4]. 在众多数据流挖掘任务中，数据流聚类作为知识发现的重要手段得到了深入研究. 与常规聚类一样，数据流聚类也是将指定样本集划分为若干不相交部分的过程，但是巨大的样本数量与在线的处理需求使得常规的聚类算法难以在数据流上直接应用.

目前已有的数据流聚类算法大部分局限于处理只具有连续属性的数据流[5-7]，另外有少量的算法局限于处理只具有标称属性的数据流[8-9]. 其中所谓的连续属性是指属性的取值为连续数值，如长度、重量；所谓的标称属性是指属性的取值为有限的状态，如颜色、职业. 现实世界中的许多数据流同时具有连续属性和标称属性，如网络数据包等. 上述两种处理一类属性的算法在混合属性条件下必然损失数据信息，影响数据挖掘的质量. 本文提出了一种适用于处理混合属性数据流的聚类算法——HCluStream (Heterogeneous CluStream). 该算法沿用了 CluStream 算法[6]的在线-离线两阶段聚类框架，为混合属性构建新的信息汇总方式及距离度量. 本文的主要贡献在于：构建了数据流到达的泊松过程模型；定义了离散属性的频度直方图描述；给出了混合属性条件下微聚类生成、更新、合并和删除算法. 在公共数据集上的实验表明，本文提出的算法具有鲁棒的性能.

本文第 2 节介绍数据流聚类的研究背景及典型算法；第 3 节给出在线-离线两阶段聚类的算法框架，并详细介绍在线阶段的初始化、微聚类生成、演化、合并与消亡以及离线阶段的宏聚类生成的算法流程；第 4 节进行实验比较分析；第 5 节总结全文，提出展望.

2　研究背景

数据流聚类的一个典型算法是 Guha 等提出的 STREAM 算法[5]. 这种算法根据分治原理，利用有限的空间对数据流进行分层次的聚类. Aggarwal 指出上述算法无法处理演化的数据流，并在总结了上述方法本质缺陷的基础上提出了一个数据流聚类框架——CluStream[6]. 其核心思想是将聚类过程分为在线和离线两个阶段. 在线部分的任务是存储数据流的汇总结果，生成一种称为微聚类的信息存储结构，并按金字塔式时间结构将中间结果进行保存. 离线部分即是根据用户指定的观察时段及聚类数量，快速生成聚类结果的过程. 由于离线部分的聚类是在在线部分生成的微聚类基础上进行聚类，它也被称为宏聚类.

这一框架得到了数据流挖掘领域研究者的普遍赞同. 几年来许多研究者围绕这一框架开展了大量工作. Aggarwal 等后续提出了专门针对高维连续属性数据流的 HPStream 算法[7]. 该算法引入了子空间聚类，并提出了具有遗忘特性的聚类结构. 他们报导了在高维数据流条件下，HPStream 的性能优于 CluStream. 但是这一算法抛弃了 CluStream 框架中关键的快拍存储，这使 HPStream 无法根据用户需求进行历史查询. Cao 等人[10]结合具有遗忘特性的微聚类与 DBSCAN 算法，提出了基于密度的两阶段聚类方法. Ong 等人[9]根据 CluStream 提出了用标称属性频度作为微聚类描述处理标称属性的算法，称为 SCLOPE. 最近，Aggarwal 等也独立提出了处理文本及标称属性数据流聚类的算法[8].

主流的数据流聚类算法都是多层次多阶段的：首先对原始数据流进行聚类，然后对这些结果进行再次聚类. 这样能够保证较好的时间和空间性能以及易用性和精度. 这些工作大部分局限于处理仅包含连续属性的数据流[5-7]，少量的工作可以处理仅包含标称属性的数据流[8-9]. 同时具有连续属性与标称属性的混合属性数据流在实际中普遍存在，但是目前尚未有文献进行混合属性的数据流聚类方法的研究. 在此方面，我们曾做过一些初步的工作[11]，本文将深入探讨相关问题.

3　HCluStream 算法

3.1　算法框架

因借鉴了 CluStream 的思想，我们将该算法命名为 HCluStream，其中的“H”表示混合属性(Heterogeneous). 与 CluStream 算法类似，HCluStream 将数据流聚类问题分为两个阶段——在线阶段与离线阶段，如图 1 所示. 在线阶段的任务是随着数据流中元素的不断到达，实时提取其汇总信息并以适当的结构进行有效的存储. 每条汇总信息是由一定数量的数据流元素形成的聚类特征向量，称为微聚类. 这些微聚类按金字塔式时间间隔进行存储，以平衡存储消耗和精确性. 离线阶段的任务是根据用户的聚类需求，如时段及粒度，利用微聚类生成指定时段内的聚类结果. 与微聚类相对应，离线阶段的聚类结果称为宏聚类.

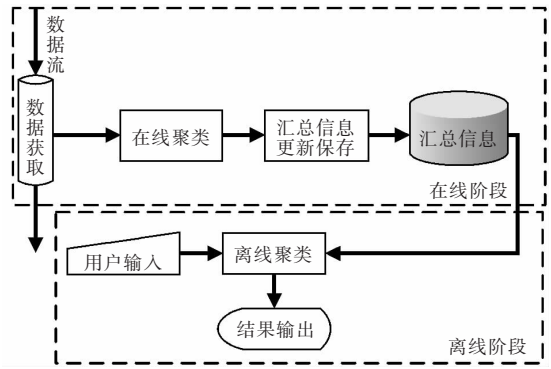

图 1 HCluStream 算法流程

首先定义本文中实用的若干符号. 待处理数据流是一个数据序列 $\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_i,\cdots$,其中样本的到达时刻分别为 $T_1,T_2,\cdots,T_i,\cdots$. 每一个样本具有 c 维连续属性与 b 维标称属性,表示为 $\boldsymbol{X}_i=\boldsymbol{C}_i:\boldsymbol{B}_i=(x_i^1,x_i^2,\cdots,x_i^c,y_i^1,y_i^2,\cdots,y_i^b)$,其中 $\boldsymbol{C}_i$ 是由 c 维连续属性 $x_i^1,\cdots x_i^c$ 构成的向量,$\boldsymbol{B}_i$ 是由 b 维标称属性 $y_i^1,y_i^2,\cdots,y_i^b$ 构成的向量. 标称属性 $y^p(1\leq p\leq b)$ 的全部可能取值数记为 F^p,y^p 的第 $k(1\leq k\leq F^p)$ 种可能值记为 v_k^p. 我们将文献[6]中的微聚类结构进行扩展,并命名为 m-HCluster.

定义 1. 定义在样本集 $\{\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n\}$ 上的微聚类可以表示为一个包含 $\sum_{p=1}^{b}F^p+2c+3$ 个元素的元组:$\boldsymbol{CFT}=(\boldsymbol{H},\boldsymbol{F},\boldsymbol{D},t_0,t_n,n)$,其中 $\boldsymbol{H}$ 是标称属性的频度直方图,包含 $\sum_{p=1}^{b}F^p$ 个元素,其第 p 行的第 k 个元素对应于第 p 个标称属性的第 k 个取值的频度,可以用公式记作 $H_{(p,k)}=\sum_{m=1}^{n}h_{p,k}^m$,其中 $h_{p,k}^m$ 表示第 m 个样本的第 p 个标称属性是否取值为 v_k^p,如式(1)所示

$$h_{p,k}^m=\begin{cases}0, & y_m^p\neq v_k^p\\ 1, & y_m^p=v_k^p\end{cases}\tag{1}$$

$\boldsymbol{F}$ 与 $\boldsymbol{D}$ 分别是连续属性的一阶矩和二阶矩,均包含 c 个元素,且 $\boldsymbol{D}^k=\sum_{m=1}^{n}(x_m^k)^2$,$\boldsymbol{F}^k=\sum_{m=1}^{n}x_m^k(1\leq k\leq c)$;$t_0$ 是微聚类创建时间;t_n 是微聚类最后更新时间;n 表示集合中的样本数.

一个数据流可分为许多个不相交的样本子集,每一个样本子集 C_j 上的微聚类记作 $M_j=\boldsymbol{CFT}(C_j)$. 微聚类的属性用"·"表示,如 $M_j\cdot n$ 表示微聚类 M_j 包含的样本数量.

3.2 微聚类存储调度

微聚类采用金字塔式时间间隔的存储方式. 在每一个存储时刻,当前全部微聚类被存储在一个快拍之中. 存储时刻分成若干层级. 相同层级上的连续两个快拍间隔相同,高层级的间隔是相邻的低层级间隔的 α 倍,其中 α 为大于 1 的整数. 每一层级上的快拍维护方法如下.

(1)第 i 层级上的快拍时间间隔为 α^i. 即数据流开始之后 α^i 整数倍时间点是该层级保存快拍的时刻;

(2)在任意时刻,仅保留第 i 层级上最新的 $\alpha+1$ 个快拍.

按层级保存快拍的目的是平衡存储量与精确性,因此每一时刻对应的快拍仅需一个. 但严格按上述方法执行,则将产生层间冗余,因此需要在不同层级之间共享快拍. 如表 1 所示,当前时刻为 $T=25$,$\alpha=2$. 第 3 层级上本应有 8、16、24 三个时刻的快拍,但是由于第 4 层级上已有 16 时刻的快拍,因此在第三层级上不再保留 16 时刻快拍. 其余层级情况类似.

表 1 $T=25,\alpha=2$ 快拍示例

层级	快拍时刻		
0	21	23	25
1	14	18	22
2	4	12	20
3	8	24	
4	16		

文献[6]指出,对于任意指定的时间窗长 h,在从当前时刻向前追溯 $2h$ 个时间单位,至少存在一个已存储的快拍. 按照这种方式存储的快拍数量相当少:一个运行 10 年的数据流系统,假设时钟周期为 1s,则 $\alpha=2$ 时快拍数量为 $(2+1)\log_2(10\times365\times24\times60\times60)\approx85$. 为了获得更精确的近似,可以修改快拍存储方式,增加每一层级存储的快拍数量. 例如,每一层级上存储的快拍数量由 $\alpha+1$ 增加到 $\alpha^\chi+1$,则任意时间窗长的近似因子将减小为 $1/\alpha^\chi+1$,其中 $\chi>1$. 当 $\chi=10$ 时,前述系统的快拍需求量为 $(2^{10}+1)\log_2(10\times365\times24\times60\times60)\approx28939$.

表 2 T=25,α=2 紧致快拍示例

层级	快拍时刻				
0	17	19	21	23	25
1	10	14	18	22	
2	4	12	20		
3	8	24			
4	16				

另外,在具体实现时,由于系统存储容量是确定的,如果严格按上述金字塔形结构,并保证每层快拍

数量上限相同，在系统启动阶段将浪费许多存储空间. 表 1 中第 3、4 层共有三个快拍节点闲置，可以将其空间分配给低层使用. 例如给第 1 层增加 10，给第 0 层增加 19、17，形成如表 2 中的形式，这样可以提高初期查询的精确程度. 随着数据增长，闲置节点将被高层占用，系统进入稳定运行状态.

3.3　在线聚类过程

为描述微聚类过程，我们首先基于定义 1 给出混合属性下的样本与样本之间、样本与微聚类之间以及微聚类与微聚类之间的距离定义. 其中每一维连续属性均采用了文献[7]中提出的方式归一化，使其方差为 1. 样本与样本的距离定义为

$$Dss(\boldsymbol{X}_i,\boldsymbol{X}_j)=Dss(\boldsymbol{C}_i,\boldsymbol{C}_j)+\beta Dss(\boldsymbol{B}_i,\boldsymbol{B}_j) \quad (2)$$

其中 $Dss(\boldsymbol{C}_i,\boldsymbol{C}_j)=\sqrt{\sum_{k=1}^{c}(x_i^k-x_j^k)^2}$ 是连续属性部分的距离；$Dss(\boldsymbol{B}_i,\boldsymbol{B}_j)=\sum_{k=1}^{b}\delta(y_i^k,y_j^k)$ 是标称属性部分的距离，其中

$$\delta(y_i^k,y_j^k)=\begin{cases}1, & y_i^k\neq y_j^k\\ 0, & y_i^k=y_j^k\end{cases} \quad (3)$$

β 是标称属性部分的权重. 样本与微聚类之间的距离定义为

$$Dsm(\boldsymbol{X}_i,M_j)=Dss\left(\boldsymbol{C}_i,\frac{M_j\cdot\boldsymbol{F}}{M_j\cdot n}\right)+\beta Dsm(\boldsymbol{B}_i,M_j\cdot\boldsymbol{H}) \quad (4)$$

其中 $M_j\cdot\boldsymbol{F}/(M_j\cdot n)$ 是 M_j 连续属性的中心，$Dsm(\boldsymbol{B}_i,M_j\cdot\boldsymbol{H})$ 是标称属性部分距离，简记为 $Dsm(\boldsymbol{B}_i,\boldsymbol{H})$

$$Dsm(\boldsymbol{B}_i,\boldsymbol{H})=\sum_{p=1}^{b}\sum_{k=1}^{F^p}\left(\frac{\delta(y_i^p,v_k^p)\boldsymbol{H}_{(p,k)}}{\sum_{m=1}^{F^p}\boldsymbol{H}_{(p,m)}}\right) \quad (5)$$

微聚类与微聚类之间的距离定义为

$$Dmm(M_i,M_j)=Dss\left(\frac{M_i\cdot\boldsymbol{F}}{M_i\cdot n},\frac{M_j\cdot\boldsymbol{F}}{M_j\cdot n}\right)+\beta Dmm(M_i\cdot\boldsymbol{H},M_j\cdot\boldsymbol{H}) \quad (6)$$

其中 $Dss(M_i\cdot\boldsymbol{F}/(M_i\cdot n),M_j\cdot\boldsymbol{F}/(M_j\cdot n))$ 是连续属性部分的距离，$Dmm(M_i\cdot\boldsymbol{H},M_j\cdot\boldsymbol{H})$ 是标称属性部分的距离，用 $Dmm(\boldsymbol{H}^i,\boldsymbol{H}^j)$ 替代

$$Dmm(\boldsymbol{H}^i,\boldsymbol{H}^j)=b-\sum_{p=1}^{b}\frac{\boldsymbol{H}_p^i\boldsymbol{H}_p^{j\,\mathrm{T}}}{\|\boldsymbol{H}_p^i\|_2\|\boldsymbol{H}_p^j\|_2} \quad (7)$$

其中，$\boldsymbol{H}_p$ 表示 $\boldsymbol{H}$ 的第 p 行，即标称属性 p 的取值频度向量.

在线聚类的过程可分为初始化、更新、添加、删除及合并等 5 个顺序执行的步骤. 首先利用静态批量聚类算法从初始样本集上生成一定数量的微聚类. 然后对于每一个新到样本，判断其是否应当加入某现有微聚类中抑或单独形成新的微聚类，当判断为并入已有微聚类时，对该微聚类实行更新操作；而当判断为形成新的微聚类时，实行微聚类添加操作. 当微聚类数量达到上限时，需要判断是否可以删除其中的某些微聚类，如果可以则删除满足条件的微聚类. 如微聚类数量已达上限并且无满足删除条件的微聚类时，则应用微聚类合并算法. 每个步骤的详细描述如下.

① 初始化. 初始化步骤采用静态数据聚类算法. 即从数据流中累积一定数量的样本，利用 K-protoypes 算法[12]将其进行聚类，并为每一微聚类分配一个 ID. 一般选取系统能够承受条件下尽可能多的样本，以保证获得较好的效果. K-protoypes 算法需要指定最终形成的微聚类总数量 ψ. ψ 值越大，在线阶段保存的原始信息就越多，有利于提高宏聚类精度，但会增加存储量与计算量；ψ 值越小，在线阶段保存的原始信息就越少，会降低宏聚类精度，但可以减少存储量与计算量. 在运行 K-protoypes 算法时，式(2)定义的距离被用于作为样本与样本之间以及样本与代表向量之间的距离度量. 算法结束后，可以根据定义 1，计算出微聚类 $M_1,M_2,\cdots,M_\psi$ 的相关信息.

② 更新. 对 T 时刻的样本 $\boldsymbol{X}$，根据式(4)计算 $D_j=Dsm(\boldsymbol{X},M_j)$，其中 $1\leq j\leq\psi$. 令 $J=\arg\min_{1\leq j\leq\psi}D_j$，则 M_J 为距离样本 $\boldsymbol{X}$ 最近的微聚类. 由于式(4)已经进行了归一化，因此可以给定常数阈值来决定 $\boldsymbol{X}$ 是否应合并入 M_J：当 $D_J<\rho(c+\beta b)$ 时，$\boldsymbol{X}$ 并入 M_J. 其中 ρ 为预设值，本文设为 1. 利用可加性更新 M_J 的信息

$$\begin{cases}M_J\cdot\boldsymbol{H}_{(p,q)}=M_J\cdot\boldsymbol{H}_{(p,q)}+h_{p,q}\\ M_J\cdot\boldsymbol{F}^k=M_J\cdot\boldsymbol{F}^k+x^k\\ M_J\cdot\boldsymbol{D}^k=M_J\cdot\boldsymbol{D}^k+(x^k)^2\\ M_J\cdot t_n=T\\ M_J\cdot n=M_J\cdot n+1\end{cases}.$$

③ 添加. 当 $D_J\geq\rho(c+\beta b)$ 时，将 $\boldsymbol{X}$ 的属性赋给一个新的微聚类，并为其分配新 ID：

$$\begin{cases}M_{\psi+1}\cdot\boldsymbol{H}_{(p,q)}=h_{p,q}\\ M_{\psi+1}\cdot\boldsymbol{F}^x=x^k\\ M_{\psi+1}\cdot\boldsymbol{D}^x=(x^k)^2\\ M_{\psi+1}\cdot t_0=T\\ M_{\psi+1}\cdot t_n=T\\ M_{\psi+1}\cdot n=1\end{cases}.$$

④ 删除. 理想的删除算法应考虑两种情况：一种是过小的微聚类，表明该聚类很可能是由噪声或野值引入的，不是主要模式；另一种是陈旧的微聚

类，表明该聚类代表的模式已经过时，不能描述当前数据. 由于包含少量样本的微聚类也可能是刚产生的新模式，所以删除这种微聚类有风险. 为了避免误删除新生聚类，文献[6]采用了一种倾向于删除陈旧微聚类的策略，这一策略的基础是时间戳正态分布的假设. 但是这一假设对于大多数实际的数据流是不正确的. 大量研究表明，在实际的流式服务系统中，如电信呼叫流、收银台顾客流、交通流等等，样本的到达近似为泊松过程，即到达时间间隔服从指数分布. 因此研究泊松过程下的聚类删除问题更具一般性. 我们利用时齐泊松过程模型

$$P[N(s+t)-N(t)]=\frac{(\lambda t)^k}{k!}\mathrm{e}^{-\lambda t} \tag{8}$$

对同一聚类中的样本的到达时刻进行建模. 相邻两样本的到达时之差 $y_n=t_n-t_{n-1}$ 应服从参数为 λ 的指数分布. 利用 y 的均值作为 $1/\lambda$ 的无偏估计，可得 $1/\hat{\lambda}=\bar{y}=\sum_{i=1}^{n}y_n/n=(t_n-t_0)/n$. 算法需要维护聚类的初始化时刻、最后更新时刻及样本总量，这些都在聚类更新和添加的流程中已经包含，不再赘述.

当微聚类总量达到上限时，对每个微聚类进行删除判断：如果 $T-M_j\cdot t_n>\xi(M_j\cdot t_n-M_j\cdot t_0)/n$，删除 M_j. ξ 是截断阈值，经验设定为 $\xi=-\ln(0.001)\approx 7$.

⑤合并. 由于聚类一旦被合并将无法拆分，所以仅当无合适聚类可删除时才考虑合并操作. 根据式(6)定义的距离，寻找到相距最近的两个微聚类 M_μ 与 M_ν. 这两个聚类合并后形成新的聚类 M_ω，同时继承 M_μ 与 M_ν 的 ID，其结构可表示为

$$\begin{cases} M_\omega\cdot \boldsymbol{H}_{(p,q)}=M_\mu\cdot \boldsymbol{H}_{(p,q)}+M_\nu\cdot \boldsymbol{H}_{(p,q)} \\ M_\omega\cdot \boldsymbol{F}^k=M_\mu\cdot \boldsymbol{F}^k+M_\nu\cdot \boldsymbol{F}^k \\ M_\omega\cdot \boldsymbol{D}^k=M_\mu\cdot \boldsymbol{D}^k+M_\nu\cdot \boldsymbol{D}^k \\ M_\omega\cdot t_0=\min(M_\mu\cdot t_0,M_\nu\cdot t_0) \\ M_\omega\cdot t_n=T \\ M_\omega\cdot n=M_\mu\cdot n+M_\nu\cdot n \end{cases}.$$

在执行完上述操作后，算法将检查当前时间是否为层级快拍存储时间点，是则将当前所有微聚类存储为一个快拍.

3.4 离线聚类过程

在这一阶段，用户指定要观察的时间窗口长度 h，并指定期望获得的聚类数量 K. 因为微聚类反映的是其创建以来的全部信息，所以需要利用微聚类的可减特性才能获得指定时段内的信息. 令 C_1 与 C_2 为两个样本集，$C_1\supseteq C_2$，则 C_1-C_2 的聚类特征向量 $\boldsymbol{CFT}(C_1-C_2)$ 为

$$\begin{cases} M_{12}\cdot \boldsymbol{H}_{(p,q)}=M_1\cdot \boldsymbol{H}_{(p,q)}-M_2\cdot \boldsymbol{H}_{(p,q)} \\ M_{12}\cdot \boldsymbol{F}^k=M_1\cdot \boldsymbol{F}^k-M_2\cdot \boldsymbol{F}^k \\ M_{12}\cdot \boldsymbol{D}^k=M_1\cdot \boldsymbol{D}^k-M_2\cdot \boldsymbol{D}^k \\ M_{12}\cdot t_0=\min(M_1\cdot t_0,M_2\cdot t_0) \\ M_{12}\cdot t_n=M_1\cdot t_n \\ M_{12}\cdot n=M_1\cdot n-M_2\cdot n \end{cases}.$$

由快拍性质可知，对任意窗长 h，存在快拍点 τ，使 $T-\tau<(1/\alpha^{\chi-1}+1)h$. 令 T 时刻的微聚类集合为 Ω_T，τ 时刻的微聚类集合为 Ω_τ，则时间窗 h 之内的微聚类可近似表示为 $\Omega_h\approx\Omega_T-\Omega_\tau$. 微聚类集合的减运算定义如下：首先令 $\Omega_h=\Omega_T$，如果对 $M_j\in\Omega_\tau$，存在 $M_s\in\Omega_h$，使 M_j 的 ID 包含在 M_s 的 ID 列表中，则 $M_s=M_s-M_j$. 如果 $M_s\cdot n=0$，将其从 Ω_h 中删除. 这样就得到了时间窗 h 内的微聚类集合 Ω_h.

利用 K-prototypes 算法对时间窗 h 内的微聚类集合 Ω_h 进行宏聚类，即可得到最终结果. K-prototypes 算法中的样本即为 Ω_h 中的微聚类 M_s，其权重定义为 $M_s\cdot n/\left(\sum_{M_j\in\Omega_h}M_j\cdot n\right)$，距离度量采用式(6).

4 实验结果

本文全部实验均在 PC 上完成. 所用计算机配置如下：CPU 为 Intel Pentium Ⅳ 3.0GHz，内存为 1GB DDR RAM，操作系统为 Windows XP Professional Edition，编程语言为 C++.

本文的实验数据集与文献[7]中所用标准数据集一致，为 KDD-CUP′99 网络入侵检测数据集与 ForestCoverType 数据集. 这两个数据集均为 UCI 数据挖掘数据集①. KDD-CUP′99 数据集来自于麻省理工学院林肯实验室连续两个星期的网络流检测. 数据集共包含 494020 条 TCP 连接记录，分属于 23 种不同的网络连接类型. 494020 条记录中有一条为野值记录，弃之不用. 每条记录包含 41 维属性，连续属性 34 维，标称属性 7 维. ForestCoverType 是美国森林服务信息系统提供的数据，共包含 581012 条记录. 每条记录是一块面积为 30×30 平方英尺土地上的地理数据，包含 54 维属性，其中连续属性 10 维，标称属性 44 维，对应于 7 种森林覆盖类型.

由于样本含有标称属性，因此文献[6]中基于平方距离和(SSQ)的质量评价方式不适合评价 HCluStream 聚类算法的性能. 我们采用了文献[7]中使用的聚类纯度作为比较 HCluStream 与 CluS-

① http://kdd.ics.uci.edu

tream 性能的标准. 聚类纯度定义为每个聚类结果中真实主导类别所占比例的平均值.

我们采用了与文献[7]相同的实验方式. 定义速度 s 为同一个处理时刻到达的样本个数. 而窗口长度 h 是创建宏聚类回溯的时间窗长. HCluStream 算法中标称属性权值 $\beta=1$，其他参数采用前面章节中的默认值. CluStream 算法的设置与文献[7]相同.

两个算法的平均聚类纯度如表 3 所示，平均聚类纯度是指在所有窗口内聚类纯度的平均值. 由于 KDD-CUP′99 数据集的绝大多数窗口内仅包含一种类型的连接，CluStream 算法与 HCluStream 算法均能正确聚类，这导致两种算法的平均精度都较高. Forest CoverType 数据则不同，只有少量窗口仅包有一种类型，HCluStream 算法精度比 CluStream 算法要高. 我们可以在一些代表性的时间戳上进行进一步分析.

表 3　平均聚类纯度

算法		平均聚类纯度	
		CluStream	HCluStream
KDD-CUP′99	$h=1, s=200$	99.75	99.81
	$h=10, s=100$	99.44	99.79
Forest CoverType	$h=1, s=200$	76.60	79.67
	$h=10, s=100$	82.58	86.22

CluStream 算法与 HCluStream 算法在代表性时间戳上的实验结果如图 2～图 5 所示，这些代表性时间戳均为文献[7]中采用的数据.

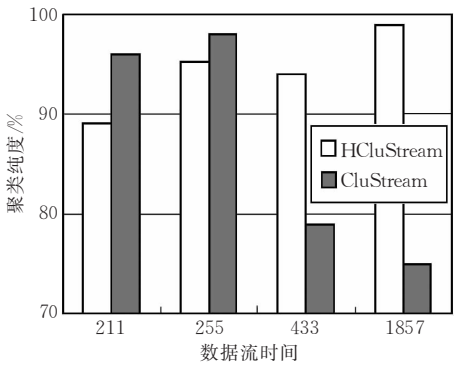

图 2　KDD-CUP′99 网络入侵检测数据集上的实验结果，$h=1, s=200$

在图 2 中，HClustream 在时间戳 211 与 255 比 CluStream 性能略差，但是在时间戳为 343 与 1857 的时候性能显著比 CluStream 要好，此时用 Clustream 得到的聚类纯度不足 80%. 在图 3 中，HClustream 在时间戳为 1500 与 4500 性能显著比 CluStream 要好，在时间戳为 2500 和 3500 时两者的性能近似.

在图 4 中，HClustream 在时间戳为 320、640、1280、2560 的性能均好于 CluStream，在时间戳为 320 时 HClustream 性能略差. 在图 5 中，HClustream 在时间戳为 200、400、800、1600 的性能均好于 CluStream，在时间戳为 3200 时 HClustream 性能略差.

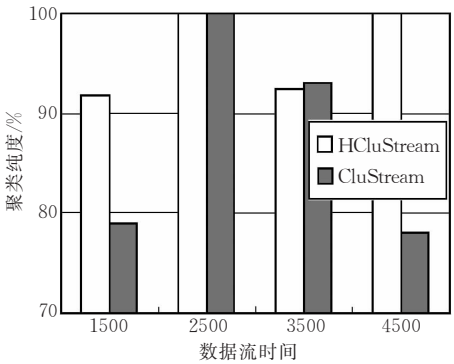

图 3　KDD-CUP′99 网络入侵检测数据集上的实验结果，$h=10, s=100$

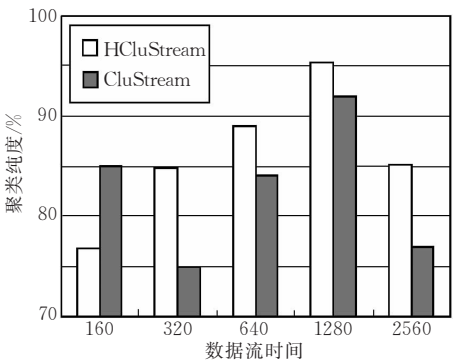

图 4　Forest CoverType 数据集上的实验结果，$h=1, s=200$

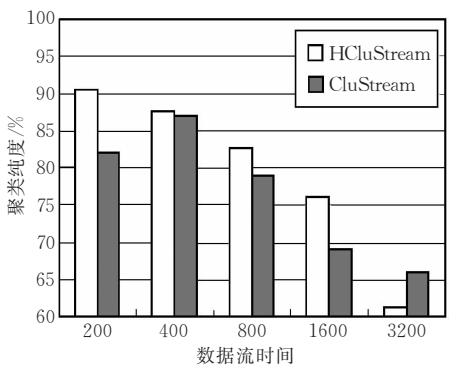

图 5　Forest CoverType 数据集上的实验结果，$h=10, s=100$

由于 HClustream 比 Clustream 增加了标称属性部分的处理，所以 HClustream 的运行速度比 Clustream 要慢一些. 在未经专门优化的情况下，在 KDD-CUP99 数据集上 HCluStream 平均每秒能处理 5000 左右个样本，而 CluStream 能处理 6000 个左右；在 ForestCoverType 数据集上 HCluStream 平均每秒能处理 3000 左右个样本，而 CluStream 能处理 3000 个左右. 实际上，这两种算法的运行速度

都已经足够的快,处理几十万条数据仅需数分钟.

CluStream 简单地将混合属性数据流中的标称型数据忽略,仅利用其中的连续型属性,损失了信息.这些标称型属性很可能是区分数据流不同聚类的关键属性.统计意义上讲,引入更多的特征,尤其是引入相互独立或者互补的特征将有助于获得稳定的聚类性能.表 3 中的平均聚类纯度结果表明,HCluStream 比 CluStream 鲁棒,代表性时间戳上的纯度对比更直观地显示了其优越性.因此,对于存在混合属性的数据流聚类问题,应采用 HCluStream 处理混合属性,而非忽略其标称部分以采用 CluStream 算法.

5 结论与展望

数据流聚类作为数据流挖掘的重要方法,得到了广泛的研究,但现有的数据流聚类算法大部分局限于处理仅包含连续属性的数据流或者仅包含标称属性的数据流.而事实上同时具有连续属性与标称属性的混合属性数据流在实际中普遍存在.为了处理这种属性混合数据流聚类问题,我们在借鉴了 CluStream 微聚类-宏聚类两阶段框架的基础上提出了混合属性数据流聚类算法 HCluStream.对于混合属性的标称型部分,提出了微聚类的直方图表示方式.针对现实世界中数据流的特性,提出了利用泊松过程对样本到达时间进行建模的方法.在标准数据集上,本文的算法表现出了良好的性能.

样本到达时刻的建模方面,本文仅利用了最简单的时齐的泊松过程.事实上,对于许多数据流,尤其是具有批量特性服务系统数据流,其样本的到达时刻具有复合泊松特性,即每一批次的到达时刻是泊松过程,且每批到达的顾客数是一个随机变量.另外,数据流系统的演化可能导致在不同时刻样本的到达速率是不同的,即对应于非时齐的泊松过程.研究如何在复杂模型条件下进行模型参数估计是下一步需要开展的工作.

参 考 文 献

[1] Muthukrishnan S. Data Streams: Algorithms and Applications. Hanover, MA, USA: Now Publishers Inc., 2005

[2] Golab L, Özsu M T. Issues in data stream management. SIGMOD Record, 2003, 32(2): 5-14

[3] Garofalakis M N, Gehrke J. Querying and mining data streams: You only get one look//Proceedings of the 28th International Conference on Very Large Data Bases. Hong Kong, China, 2002: 635-635

[4] Gaber M M, Zaslavsky A B, Krishnaswamy S. Mining data streams: A review. SIGMOD Record, 2005, 34(2): 18-26

[5] 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

[6] Aggarwal C C, Han Jia-Wei, Wang Jian-Yong, Yu P S. A framework for clustering evolving data streams//Proceedings of the 29th International Conference on Very Large Data Bases. Berlin, Germany, 2003: 81-92

[7] Aggarwal C C, Han Jiawei, Wang Jianyong, Yu P S. A framework for projected clustering of high dimensional data streams//Proceedings of the 30th International Conference on Very Large Data Bases. Toronto, Canada, 2004: 852-863

[8] Aggarwal C C, Yu P S. A framework for clustering massive text and categorical data streams//Proceedings of the 6th SIAM International Conference on Data Mining. Bethesda, MD, USA, 2006: 477-481

[9] Ong K L, Li Wen-Yuan, Ng W K, Lim E P. SCLOPE: An algorithm for clustering data streams of categorical attributes//Proceedings of the 6th International Conference on Data Warehousing and Knowledge Discovery. Zaragoza, Spain, 2004: 209-218

[10] Cao Feng, Ester M, Qian Wei-Ning, Zhou Ao-Ying. Density-based clustering over an evolving data stream with noise//Proceedings of the 6th SIAM International Conference on Data Mining. Bethesda, MD, USA, 2006: 326-337

[11] Yang Chun-Yu, Zhou Jie. HCluStream: A novel approach for clustering evolving heterogeneous data stream//Workshops Proceedings of the 6th IEEE International Conference on Data Mining. Hong Kong, China, 2006: 682-688

[12] Huang Zhe-Xue. Extensions to the *k*-means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, 1998, 2(3): 283-304

YANG Chun-Yu, born in 1982, Ph. D. candidate. His research interests include data mining and machine learning.

ZHOU Jie, born in 1968. Ph. D., Ph. D. supervisor. His research interests include pattern recognition, information fusion, image processing and computer vision.

Background

Recently, the ability to capture measurements of data increased continuously. As a result, large volume and continuous growing data sequences become available to people. These data sequences are often called data streams. Examples include sensor networks, Web click streams and internet traffic flow. The most important characteristics of data stream are one pass, continuous arriving and evolving. Managing and mining data stream has gained much attention. As a fundamental machine learning and data mining technique, clustering has received extensive research in both machine learning and data mining community. While in data stream circumstance, the one pass constraint and the limitation of storage resource challenge traditional clustering algorithms designed for static database mining. The one pass constraint means that the elements in data stream can be accessed only once except explicitly stored. The limitation of storage resource means that not all the elements in the data streams can be stored even if some of them can be cached. Clustering algorithms designed for static database mining task have to be modified to accommodate these constraints under data stream environment.

Many approaches have been proposed for data stream clustering. Among these algorithms, CluStream using pyramidal time frame with online and offline components proves to be efficient in many applications. Just as other algorithms, including its modified version HPStream, CluStream is designed to manipulate continuous data streams only with continuous or so called numeric features. But in real application, many data streams contain both continuous and categorical attributes. The data stream with both continuous and categorical attributes are called heterogeneous data stream.

Inspired by the CluStream framework and driven by the urgent need to solve heterogeneous stream clustering problems, the authors propose an approach to manipulate the heterogeneous data stream clustering while adopt the main frame of the CluStream algorithm. The authors refer their approach as HCluStream framework, which is short for Heterogeneous CluStream.