ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.5, May 2009, pp.1313−1328
doi: 10.3724/SP.J.1001.2009.03303

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

基于概率数据流的有效聚类算法*

戴东波+, 赵 杠, 孙圣力

(复旦大学 计算机与信息技术系,上海 200433)

Effective Clustering Algorithm for Probabilistic Data Stream

DAI Dong-Bo+, ZHAO Gang, SUN Sheng-Li

(Department of Computing and Information Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: E-mail: daidongbo@fudan.edu.cn, http://www.fudan.edu.cn

Dai DB, Zhao G, Sun SL. Effective clustering algorithm for probabilistic data stream. *Journal of Software*, 2009,20(5):1313−1328. http://www.jos.org.cn/1000-9825/3303.htm

Abstract: An effective clustering algorithm called "P-Stream" for probabilistic data stream is developed in this paper for the first time. For the uncertain tuples in the data stream, the concepts of strong cluster, transitional clusters and weak cluster are proposed in the P-Stream. With these concepts, an effective strategy of choosing candidate cluster is designed, which can find the sound cluster for every continuously arriving data point. Then, in order to further cluster on the high level and analyze the evolving behaviors of data streams, snapshots of micro-clusters are stored at every checkpoint. At last, an "aggressive" two-tier clustering model is introduced to judge whether the most recently arrived data point is fitting in with the first level clustering model or not. Probabilistic data streams in the experiments include KDD-CUP'98 and KDD-CUP'99 real data sets and synthetic data sets with changing Gaussian distributions. Comprehensive experimental results demonstrate that P-Stream is of high quality, fast processing rate and is efficiently fitting in with the evolving situations of data streams.
Key words: probabilistic data stream; clustering; evolving analysis

摘 要: 提出一种在概率数据流上进行聚类的有效方法 P-Stream.P-Stream 针对数据流上的概率元组提出强簇、过渡簇和弱簇的概念,设计一种有效的在线候选簇选择策略,为每个不断到达的数据元组合理地找到可能归属的簇,并在每个检查点存储微簇快照,以便离线进一步高层聚类和演化分析.最后设计一个"积极"的二层聚类模型来判断现有的第 1 层聚类模型是否还适应数据流中最近到达的概率元组.实验采用 KDD-CUP'98 和 KDD-CUP'99 真实数据集以及变换高斯分布的人工数据集构造概率数据流.实验结果表明,P-Stream 具有良好的聚类质量、较快的处理速度,能够有效地适应数据演化情况.
关键词: 概率数据流;聚类;演化分析
中图法分类号: TP181 **文献标识码**: A

由于数据产生的随机性、测量错误和数据收集不完全,概率数据(又称为不确定数据)在很多应用领域,如数据清洗、信息集成、多传感器计算中普遍存在.因此,在数据的一个重要存在场合——数据流上也会出现不确

* Supported by the National Basic Research Program of China under Grant No.2005CB321905 (国家重点基础研究发展计划(973))
Received 2007-11-13; Accepted 2008-03-06

定性,这种数据流就是概率数据流[1−3].

由于数据的不确定性,在概率数据流上和确定数据流上聚类存在一些差别,主要体现在以下两个方面:

(1) 确定数据流聚类形成的簇是确定(存在概率为 1)存在的,评价其聚类质量的标准是体现簇紧凑程度的指标,如平方距离和 SSQ(sum of squared distance),它通过计算所有点到各自的聚类中心的距离的总和来衡量算法所给出的聚类质量,SSQ 值越小,说明算法聚类质量越好.但概率数据流不同,聚类形成的簇是以一定概率(存在概率一般小于 1)形式存在的,其评判聚类质量除了指标 SSQ 之外,另外一个指标就是形成的簇存在概率的大小,簇的存在概率越大,对应用越有意义.比如在电子商务网站中,由于对访问用户数据收集的不完全性,致使用户访问的日志文件是概率数据形式的,对这些概率数据进行聚类,如果得到某个簇的 SSQ 指标很低,则表明用户的访问模式很相似;但如果簇中概率数据的平均存在可能性(即簇的存在概率)很小,则这个簇对分析者很可能是不感兴趣的.所以,概率数据流形成的簇有双重评价尺度,即簇的紧凑性和存在可能性大小.

(2) 不论是确定数据流还是概率数据流,随着数据的不断高速到达,数据流上的聚类在不断演化.聚类形成的每个簇所包含的数据个数、中心点位置和半径大小在每个时刻都是不断变化的,这种变化反映了数据在某段时间内的分布情况.但概率数据流还有一种重要的演化:每个簇的存在概率的大小变化情况.随着数据流的数据不断到达,每个簇的存在概率在不断变化,从分析者偏好存在概率较大的簇的角度来看,簇存在概率不断增大或保持簇在一个较高的存在概率水平上较为“有利”.

概率数据流上聚类的另一个演化问题是,当现有的聚类模型不能适应数据流中最近到达的概率元组时会产生过多的异常点,由于这时形成的簇要兼顾簇的紧凑性和存在概率尽可能大的要求,不能采用 CluStream[4]方法中删除一个“老”簇或合并两个最近的簇的方法来为异常点建立一个新簇,而是需要建立一个新的聚类模型.本文引用一种基于统计的方法来判断现有聚类模型是否适应一定时间间隔内到达的数据.

本文提出一种在概率数据流上的有效聚类方法 P-Stream,其主要贡献是:1) 设计了一种有效的在线候选簇选择策略,为每个不断到达的概率元组合理地找到可能归属的簇,并且理论上分析了与单纯地考虑最近距离因素的传统方法相比,这种策略会找到更多存在概率较大的簇,但其所形成的簇的 SSQ 值不超过传统方法的常数倍;2) 合理地确定了检查点时间间隔值,保证了及时地捕捉到簇的存在概率变化,并在每个检查点存储快照,以便离线进一步高层聚类(即把获得的微簇中心点当作伪点,进一步聚类用户指定的 k 个簇)和演化分析;3) 引进一个基于统计的自适应采样方法检查聚类模型是否过期,并在此基础上设计一个“积极”的二层聚类模型来提前聚类异常点,避免了较大的存储空间存储异常点并减少了较大的在线延迟.通过应用真实数据集和人工合成数据集的实验表明,P-Stream 具有良好的聚类质量、较快的处理速度,能够有效地适应数据演化情况.

本文第 1 节介绍相关工作.第 2 节给出有关概念及其算法 P-Stream 的基本框架.第 3 节给出算法 P-Stream 的具体实现策略,包括在线部分的候选簇选择策略、模型过期处理方法和检查点时间间隔值的确定方法以及离线部分的高层聚类和演化分析.第 4 节给出实验结果及其分析.第 5 节对全文作总结并给出后续研究方向.

1 相关工作

数据的不确定性可以分为两种:元组的存在不确定性和值不确定性.元组的存在不确定性是指概率关系数据库中的一个元组关联一个概率值表明这个元组存在的可能性大小[5−7].值不确定性是指数据元组中的每个属性值是不精确的,精确程度是以一定概率(或概率密度函数)表示的[8,9].在文献[9]中,用 pdf(概率密度函数)来表征数据点位置的可能范围,提出在静态数据集上如何剪枝以避免计算点之间距离的昂贵代价,从而有效地在不确定数据上进行聚类.本文的数据模型是基于元组的存在不确定性的,文献[9]中的数据模型是基于值不确定性的,并且计算距离是根据其概率分布函数的,计算代价较大,不适于应用在数据流上,否则会产生较大的在线延迟.上述对概率数据的研究工作主要是集中在静态的不确定数据库中,如何保持模型语义的完整和查询的高效上.已有一些工作涉及到在概率数据流上的数据分析[1−3].文献[1−3]都是解决在数据流的环境下如何一遍扫描有效计算概率流的一些聚集查询结果,如计算概率数据流中不同元素个数(F_0)、元素个数(F_1)、概率流中不同元素个数的平方和(F_2)、平均值(mean)、中位点(median)、最小值(min)、最大值(max)等.这些聚集查询结果定义

为概率数据流中数据所形成的“可能世界”分布上的相应期望值.

目前已经提出了很多数据流上的聚类方法.这些算法可分为两类:一遍扫描算法[10−15]和演化算法[4,16−18].

一遍扫描算法把数据流上的所有数据当作单个数据库一遍处理.传统的静态数据集上聚类方法如 *K*-means 或 *K*-median 被扩展到数据流环境下,并假设数据流是以成块的形式到达[10−13].这些方法采用分而治之的策略,可在较小空间内获得常数因子的近似结果.例如:在文献[10,11]中把到达的每块数据先用传统的 *K*-means 方法在底层局部聚类成 k 个簇,然后把每个簇中心点当作加权的伪点,在上层把这些加权的伪点再进一步聚类;文献[12]提出一种 STREAM 算法来聚类数据流,STREAM 首先决定采样的大小,如果每次到达块数据大小超过了样本大小,就调用子程序 LOCALSEARCH 来获得每块数据所形成的簇.在文献[10]中,最后聚类结果的常数近似因子是随着聚类层次的增加而不断变大的,于是,文献[13]中提出了另外一种基于 *K*-median 的随机算法,只用 $O(k\times\text{polylog } n)$的空间产生固定常数因子的近似聚类结果.此外,周等人提出一种在数据流上基于核密度估计的单遍扫描算法来产生聚类结果[14].Nam 等人提出一种基于统计网格的方法,用于单遍扫描的聚类[15].

由于一遍扫描算法是聚类全部历史数据,没有考虑到历史数据的衰退和模型的过期,不适合数据流上的聚类演化分析,所以提出了很多演化聚类算法.文献[4]中的 CluStream 算法采用在线维护微簇和离线的高层聚类和演化分析方法,用金字塔式的时间框架来存储微簇快照,在线的微簇可动态删除和合并,并维护一个 ID 列表记录合并的微簇便于离线的演化分析.同样地,文献[16]中提出了高维数据流中的子空间聚类方法 HP-Stream,采用了一个衰退数据结构来反映数据的演化.文献[17]扩展了文献[19]中提出的指数直方图 EH 来维护滑动窗口内的 *K*-medians.文献[18]中算法 D-Stream 采用基于密度的数据流聚类方法,对于每个空间数据格,使其数据密度指数衰退,这样体现了每个格中微簇随时间的演化情况.

与本文最相近的工作是 Aggarwal 等人提出的 CluStream[4]算法.本文提出的 P-Stream 算法也采用在线微簇维护和离线高层聚类和演化分析.但 CluStream 算法不适合用在概率数据流上,主要因为 CluStream 是工作在确定数据流上,在为每个到达数据点查找可能归属的候选簇时,只考虑最近距离这个因素,并且在演化分析中也不会去分析簇存在概率的变化.

2 P-Stream算法的基本思想

2.1 相关概念

我们定义概率数据流 S(简称概率流)为不确定元组的序列,即 $S=\{\langle v_1,p_1\rangle\ldots\langle v_k,p_k\rangle\ldots\}$,其中,$\langle v_i,p_i\rangle(i\geq 1)$是一个不确定数据元组,$v_i$ 是元组的值(设维度为 d),p_i 是元组的存在概率.不失一般性,假设存在概率值 p_i 取自阈值 $[\theta,1](0\leq\theta<1)$,且假设各元组之间是相互独立的.进一步假设每个离散的时刻 t,概率流中到达且仅到达一个不确定元组.下面给出反映簇存在概率大小和簇存在概率变化趋势的定义.

定义 1. 假设一个有 n 个数据元组的簇 $C=\{\langle v_1,p_1\rangle\ldots\langle v_n,p_n\rangle\}$,其存在概率 EP^c 定义为各元组存在概率的几何平均,即 $EP^c=\left(\prod_{i=1}^{n}p_i\right)^{\frac{1}{n}}$. 对用户给定的参数$\alpha(1\geq\alpha>0)$,若簇 C 满足 $EP^c\geq\alpha$,则簇 C 称为强簇,α称为强簇下界.

从定义 1 容易看出,任何一个簇 C 的存在概率满足 $EP^c\geq\theta$,且当在一个存在概率为 EP^c 的簇 $C=\{\langle v_1,p_1\rangle\ldots\langle v_n,p_n\rangle\}$中加入一个新的元组$\langle v',p'\rangle$后,变成了有 n+1 个数据的新簇 C',其存在概率为

$$EP^{c'}=((EP^C)^n\times p')^{\frac{1}{n+1}}=EP^C\times\left(\frac{p'}{EP^C}\right)^{\frac{1}{n+1}}.$$

由此可以得到下面的性质:

性质 1. 在一个存在概率为 EP^c 的簇 $C=\{\langle v_1,p_1\rangle\ldots\langle v_n,p_n\rangle\}$中加入一个新的元组$\langle v',p'\rangle$后,当 $p'>EP^c$ 时,新簇存在概率会增大,即 $EP^{c'}>EP^c$;当 $p'\leq EP^c$ 时,新簇存在概率不变或减少,即 $EP^{c'}\leq EP^c$.

为了反映簇的存在概率的变化,我们定义弱簇和过渡簇.

定义 2. 弱簇 C 是存在概率小于某一阈值β的簇,即 $EP^c<\beta$,其中,β满足 $\beta<\min\left(\alpha^{\frac{1}{2}}\theta^{\frac{1}{2}},\alpha^2\right)$.过渡簇 C 就是存在概率位于强簇和弱簇的存在概率之间的簇,即$\beta\le EP^c<\alpha$.

过渡簇存在概率范围$[\beta,\alpha]$不能太小,否则,一个簇在强簇和弱簇之间有可能频繁变化,为了捕捉簇存在概率的演化情况,就会频繁地存储簇的快照,导致在线处理速度降低.当 $\beta<\min\left(\alpha^{\frac{1}{2}}\theta^{\frac{1}{2}},\alpha^2\right)$时,任意一个强簇(或弱簇)不会在加入任何一个数据元组后变为弱簇(或强簇),因此有以下引理:

引理 1. 当 $\beta<\min\left(\alpha^{\frac{1}{2}}\theta^{\frac{1}{2}},\alpha^2\right)$时,概率流中任意一个元组$\langle v,p\rangle$加入强簇(或弱簇)$C$ 后,新形成的簇 C'不会立即变为弱簇(或强簇).

证明:略. □

定义 3. 若簇 C 在时段$[t,t+T]$内有 s 个概率流中的数据元组加入,则称下列情况之一是簇 C 在时段$[t,t+T]$内的正向变化:① 簇 C 在时刻 t 是弱簇,在 $t+T$ 时刻是过渡簇或强簇;② 簇 C 在时刻 t 是过渡簇,在 $t+T$ 时刻是强簇.同理,称下列情况之一是簇 C 在时段$[t,t+T]$内的负向变化:① 簇 C 在时刻 t 是强簇,在 $t+T$ 时刻是过渡簇或弱簇;② 簇 C 在时刻 t 是过渡簇,在 $t+T$ 时刻是弱簇.

定义 4. 对于簇的 3 种状态:强簇、过渡簇和弱簇,在ΔT 时间内,若强簇或弱簇发生下面的变化之一,则称其为簇在ΔT 内的一次完整变化:① 对于初始的强簇,每加入数据流中的一个数据元组,其状态要么不变,要么发生负向变化,且最后时刻簇的状态处于弱簇;② 对于初始的弱簇,每加入数据流中的一个数据元组,其状态要么不变,要么发生正向变化,且最后时刻簇的状态处于强簇.

算法 P-Stream 扩展 BIRCH[20]算法中的 CF(clustering feature)数据结构,采用微簇来概要地存储簇中的信息.

定义 5. 对于 n 个元组的簇 $C=\{\langle v_1,p_1\rangle\ldots\langle v_n,p_n\rangle\}$,其微簇 PCF(probabilistic clustering feature)定义为 $\langle\overrightarrow{CF^2},\overrightarrow{CF^1},EP^c,n,MID\rangle$, 其中, $\overrightarrow{CF^2}$ 和 $\overrightarrow{CF^1}$ 是 d 维向量,每维向量值是簇 C 中 n 个元组各 V 值相应维的二阶矩和一阶矩,EP^c 是簇 C 的存在概率,n 是簇 C 中数据元组个数.MID 是簇 C 处于未过期时的聚类模型标号,所有处于同一未过期的聚类模型中的簇都用同一个标号.

对于定义 5 中的微簇 C,当每个新加入的元组$\langle v,p\rangle$变为 C'后,PCF 中 $\overrightarrow{CF^2},\overrightarrow{CF^1},EP^c,n,MID$ 都可增量维护,其中 $EP^{C'}=((EP^C)^n\times p)^{\frac{1}{n+1}}$.特别地, $\dfrac{\overrightarrow{CF^1}}{n}$ 是簇 C 的中心点, $\sqrt{\dfrac{\sum_{i=1}^{n}(v_i-v_c)^2}{n}}$ 是簇 C 的半径(其中,v_c 是簇 C 的中心点),其值可以通过 $\overrightarrow{CF^2}$, $\overrightarrow{CF^1}$ 进行计算.任意元组$\langle v,p\rangle$与簇 C 的距离定义为 v 值与 C 中心点的距离.

2.2 P-Stream算法的基本框架

算法 P-Stream 的基本框架如下所示.其中,时标用离散的整数 0,1,2,…,n,...表示.P-Stream 采用两层的聚类方法,每层 PCF 数目是 q.一般情况下,q 的值远大于数据中的自然簇个数,但远小于数据流中很长一段时间内到达的数据个数[4].

P-Stream:

1. { *total*=0;*Mnormal*=0;*total'*=0;*Mnormal'*=0;*TimeStamp*=0;*Cnormal*=0;*MID*=0;
   //*total*,*Mnormal* 分别计数在同一模型内第 1 层处理的元组总数和被第 1 层簇成功接受的元组数;*total'*,*Mnormal'*分别计数第 2 层处理的元组总数和被第 2 层簇成功接受的元组数;*TimeStamp* 为时标;*Cnormal* 用来计数从检查点开始被第 1 层成功接受的元组数;*MID* 是所有第 1 层簇处于同一未过期模型时的 ID 号.//
2. 把概率流中前 q 个元组作为 q 个微簇中心点,初始化其 PCF 集 *PS*;

```
3.  while (数据流没有到尽头)
4.    {读取概率流中一个元组⟨v,p⟩;TimeStamp=TimeStamp+1;
5.     C_cd=Find_Candidate_Cluster(v,p,PS);
6.     if (Acc_or_Rej(v,p,C_cd)=true)    //⟨v,p⟩落在候选簇 C_cd 的边界范围内
7.       {将⟨v,p⟩放入第 1 层簇 C_cd 中,更新其 PCF,Mnormal=Mnormal+1;Cnormal=Cnormal+1;}
8.     else   //⟨v,p⟩是第 1 层的异常点
9.     {   total'=total'+1;
10.        if (第 2 层聚类中心点少于 q 个)
11.          {将⟨v,p⟩作为微簇中心点,初始化其 PCF,加入 PCF 集 PS';Mnormal'=Mnormal'+1;}
12.        else
13.          { C'_cd =Find_Candidate_Cluster(v,p,PS');
14.            if (Acc_or_Rej(v,p, C'_cd )=true)
15.              {将⟨v,p⟩放入第 2 层簇 C'_cd 中,更新其 PCF; Mnormal'=Mnormal'+1;}
16.                else 丢弃元组⟨v,p⟩;   //⟨v,p⟩是全局异常点
17.              }}
18.      total=total+1;
19.     Model_Outdated_Process();   //模型过期处理
20.     CheckPoint_Process ();    //存储第 1 层微簇快照}}
```

P-Stream 的在线部分先把概率流中的前 q 个元组作为簇中心(第 2 行),当 q 远大于 k 时(k 是离线聚类形成的簇个数),对聚类质量影响比较小.然后对概率流中的每个时间点到达的元组⟨v,p⟩,综合考虑它与第 1 层每个簇的距离和簇的存在概率两个因素,通过调用候选簇选择策略算法 *Find_Candidate_Cluster* 在第 1 层为其找到合适的候选簇 C_{cd}(第 5 行).若⟨v,p⟩落在候选簇一定的边界范围内(第 6 行、第 7 行),则加入 C_{cd};否则,被当作是第 1 层的异常点放入第 2 层,进行异常点聚类处理(第 8~15 行).若第 2 层也未接受⟨v,p⟩,则⟨v,p⟩作为全局异常点而丢弃(第 16 行).然后,调用模型过期处理算法 *Model_Outdated_Process* 检查模型是否过期(第 19 行),若异常点过多,则表明现有聚类模型不适合当前到达的元组数据,要把第 2 层的簇放入第 1 层作为当前的聚类模型. *CheckPoint_Process* 算法是每隔一段时间来存储第 1 层的微簇快照以便离线高层聚类和演化分析(第 20 行),其中,时间间隔值的确定很重要,这将影响存储微簇快照的频率和演化分析的合理性.算法 *Find_Candidate_Cluster*,*Model_Outdated_Process* 以及 *CheckPoint_Process* 将在第 3 节详细加以说明.

确定新到达数据点⟨v,p⟩是否落在候选簇一定边界范围内的是布尔函数 *Acc_or_Rej*,它判断⟨v,p⟩与 C_{cd} 中心点的距离是不是小于候选簇 C_{cd} 半径的 τ 倍.τ 值怎么确定呢?我们知道:如果⟨v,p⟩到簇 C_{cd} 中心点的距离符合正态分布,则当 τ=2 时,簇 C_{cd} 中 95%以上的数据点将落在 τR 范围内(其中 R 是簇 C_{cd} 的半径).在实验中,τ 被设置为 2.若簇 C_{cd} 中只有一个元组,则其半径初始值可设为 C_{cd} 与离 C_{cd} 最近的簇之间距离的 1/2.

3 P-Stream算法的实现策略

3.1 候选簇选择算法Find_Candidate_Cluster

设现有簇集 $C=\{C_1,\ldots,C_n\}$,$C_1,\ldots,C_n$ 的 PCF 集为 PS,当概率流中到达一个元组⟨v,p⟩时,要在 C 中为其找到一个 $C_i(1\le i\le n)$作为候选簇.设新到达的元组⟨v,p⟩与 C 中簇 C_i 中心点的距离为 D_i,$D_{\min}$ 为⟨v,p⟩与 $C_1,\ldots,C_n$ 各中心点的最近距离,$C_{\min}$ 为取值是 $D_{\min}$ 的中心点所在的簇.为⟨v,p⟩选定一个候选簇要兼顾距离 D_i 和存在概率 EP^{C_i} 两个因素.设文献[4]中 CluStream 算法采用的只单纯考虑距离因素的候选簇选择方法为选择策略 I,算法 P-Stream 中 *Find_Candidate_Cluster* 子程序中的候选簇选择方法为选择策略 II,即:

选择策略 I. 挑选 $C_{\min}$ 为候选簇.

选择策略 II. 把 C 中的簇按 $D_i((1\le i\le n)$非递减的顺序排列,形成一个新的簇序列 $C'=\{C_{j_1},...,C_{j_n}\}$, 其中 C_{j_1} 就是 $C_{\min}$.设 C'中 C_{j_i} 满足$1\le\frac{D_{j_i}^2}{D_{\min}^2}\le M$($M$是常数,称为松弛参数,且$M\ge 1$) 的有 W 个簇,它们也一定是 C'中前 W 个簇: $C_{j_1},...,C_{j_w}$. 设 C 中任何簇 C_i 加入$\langle v,p\rangle$后变为簇 C_i'.

按照下面的启发式规则依次对 $C_{j_1},...,C_{j_w}$ 进行一遍扫描确定候选簇:

Rule 1. 在对 $C_{j_1},...,C_{j_w}$ 依次顺序扫描的过程中,若第 1 次遇到的簇 C_{j_i} ($1\le i\le W$)满足 C_{j_i} 是过渡簇,加入$\langle v,p\rangle$后的 C_{j_i}' 是强簇,则把 C_{j_i} 作为候选簇.

Rule 2. 若 Rule 1 不满足,则在对 $C_{j_1},...,C_{j_w}$ 依次扫描的过程中,若第 1 次遇到的簇 C_{j_i} ($1\le i\le W$)满足 C_{j_i} 是强簇,加入$\langle v,p\rangle$后的 C_{j_i}' 仍是强簇,则把 C_{j_i} 作为候选簇.

Rule 3. 若 Rule 1 和 Rule 2 都不满足,则在对 $C_{j_1},...,C_{j_w}$ 扫描过程中,选取簇 C_{j_i} ($1\le i\le W$),使得 $\frac{p/EP_{j_i}^C}{D_{j_i}^2/D_{\min}^2}$(称为簇 C_{j_i} 的增益值)值最大,将 C_{j_i} 作为候选簇.进一步地,若得到的候选簇 C_{j_i} 是强簇,加入$\langle v,p\rangle$后的 C_{j_i}' 是过渡簇,则不选 C_{j_i} 而把 C_{j_1} (即 $C_{\min}$)作为候选簇.

启发式规则Rule 1是为了以最小的距离代价来增加强簇个数,Rule 2是为了保持强簇个数,Rule 3选择增益值 $\frac{p/EP_{j_i}^C}{D_{j_i}^2/D_{\min}^2}$ 最大的簇作为候选簇,在数据元组$\langle v,p\rangle$到达后,p 和 $D_{\min}^2$ 可看作常量,所以,Rule 3 实际上是选择 $\frac{1}{EP^{C_{j_i}}\times D_{j_i}^2}$ 最大,即 $EP^{C_{j_i}}\times D_{j_i}^2$ 最小的簇作为候选簇.直观上看,在 Rule 1 和 Rule 2 得不到满足的情况下,Rule 3 倾向于选择概率较小和距离较近的簇作为候选簇.为了解释 Rule 3 在聚类过程中的作用,我们先证明以下定理:

定理 1. 设 C_{j_k} 是为$\langle v,p\rangle$应用 Rule 3 选择的候选簇,对于排序后的簇序列 C'中的簇 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_k}$, 若存在一些簇,当$\langle v,p\rangle$加入其中后能提高其存在概率,那么 C_{j_k} 是在 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_k}$ 中满足此条件且存在概率最小的一个簇.

证明:因为排序后的簇序列满足 $D_{j_1}^2(D_{\min}^2)\le D_{j_2}^2\le...\le D_{j_k}^2$, 而应用 Rule 3 选中的候选簇 C_{j_k}, 其 $EP^{C_{j_k}}\times D_{j_k}^2$ 值是最小的,从而有 $EP^{C_{j_k}}\le EP^{C_{j_1}},...,EP^{C_{j_k}}\le EP^{C_{j_{k-1}}}$.若$\langle v,p\rangle$加入 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_k}$ 中的某些簇能使其存在概率提高,不妨设其中一个这样的簇为 C_{j_i}, 则由性质 1 有 $p>EP^{C_{j_i}}$, 而 $EP^{C_{j_i}}\ge EP^{C_{j_k}}$, 因此 $p>EP^{C_{j_k}}$, 仍由性质 1,$\langle v,p\rangle$加入 C_{j_k} 也能使其存在概率提高,且它是 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_k}$ 中存在概率最小的一个簇. □

定理 1 表明,在$\langle v,p\rangle$加入候选簇能够提高其存在概率的前提下,Rule 3 优先选择点$\langle v,p\rangle$附近(簇序列 C'中 C_{j_k} 及其前 k–1 个簇)存在概率最小的簇.如果多次选中 Rule 3,就会使点$\langle v,p\rangle$附近的簇的整体存在概率提高.当每个不断到达数据点附近的簇的整体存在概率都提高时,可以认为全部微簇中的存在概率较低的微簇个数相对较少,这是概率流聚类所期望的.另一方面,Rule 1 和 Rule 2 满足的前提条件是候选簇的存在概率相对较大(Rule 1 要求候选簇在加入一个数据点后就能变为强簇,Rule 2 要求候选簇是强簇),当应用 Rule 3 将数据点$\langle v,p\rangle$加入候选簇 C_{j_k} 后,若使得其存在概率降低,则由定理 1 可推出,数据点$\langle v,p\rangle$加入 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_{k-1}}$ 中的任何一个簇也能使其存在概率降低.这时选中 C_{j_k} 就相当于“保护”了$\langle v,p\rangle$附近存在概率较大的簇 $C_{j_1}(C_{\min}),C_{j_2},...,C_{j_{k-1}}$ 不使其存在概率降低,也就不会降低这些较近的簇下次命中 Rule 1 和 Rule 2 的可能性.

总之,Rule 3 的作用就是在不能命中 Rule 1 和 Rule 2 的情况下,减少全部微簇中的存在概率较低的簇,并且使得下次新到达数据点来到时,保持它附近存在概率较大的簇命中 Rule 1 和 Rule 2 的可能性.

下面证明策略 II 在 SSQ 有界的前提下,能够比策略 I 找到更多的强簇.

定理 2. 设ΔSSQ(I)和ΔSSQ(II)分别为根据选择策略 I 和选择策略 II 将$\langle v,p\rangle$加入各自选中的候选簇后引起的 SSQ 值的增量,则有ΔSSQ(II)<2$M\times\Delta SSQ$(I).

证明:为方便起见,我们假设概率元组中 V 值是一维的,对于多维的情况,证明类似.由选择策略 I 找到的簇为 $C_{\min}$,不妨设 $C_{\min}$ 有 i 个元组,即 $C_{\min}=\{\langle v_1,p_1\rangle\ldots\langle v_i,p_i\rangle\}$,$C_{\min}$ 的中心点 $\overline{V_{\min}}$,加入$\langle v,p\rangle$后中心点为 $\overline{V'_{\min}}$.设由选择策略 II 找到的簇为 C_r,不妨设 C_r 有 j 个元组,即 $C_r=\{\langle v'_1,p'_1\rangle,\ldots,\langle v'_j,p'_j\rangle\}$,$C_r$ 的中心点为 $\overline{V_r}$,加入$\langle v,p\rangle$后中心点为 $\overline{V'_r}$,则

$$\begin{aligned}\Delta SSQ(\mathrm{I}) &= \sum_{t=1}^{i}(v_t-\overline{V'_{\min}})^2+(v-\overline{V'_{\min}})^2-\sum_{t=1}^{i}(v_t-\overline{V_{\min}})^2\\ &= \sum_{t=1}^{i}v_t^2+v^2-(i+1)\times\overline{V'_{\min}}^2-\sum_{t=1}^{i}v_t^2+i\times\overline{V_{\min}}^2\\ &= v^2+i\times\overline{V_{\min}}^2-\frac{(i\times\overline{V_{\min}}+v)^2}{i+1}\\ &= \frac{i}{i+1}(v-\overline{V_{\min}})^2=\left(1-\frac{1}{i+1}\right)D_{\min}^2,\end{aligned}$$

$$\sum_{t=1}^{i}v_t^2+v^2-(i+1)\times\overline{V'_{\min}}^2-\sum_{t=1}^{i}v_t^2+i\times\overline{V_{\min}}^2=v^2+i\times\overline{V_{\min}}^2-\frac{(i\times\overline{V_{\min}}+v)^2}{i+1}=\frac{i}{i+1}(v-\overline{V_{\min}})^2=\left(1-\frac{1}{i+1}\right)D_{\min}^2.$$

同理,$\Delta SSQ(\mathrm{II})=\left(1-\frac{1}{j+1}\right)D_r^2$.

所以,$\Delta SSQ(\mathrm{II})=\left(1-\frac{1}{j+1}\right)D_r^2=\frac{1-\frac{1}{j+1}}{1-\frac{1}{i+1}}\times\frac{D_r^2}{D_{\min}^2}\times\left(1-\frac{1}{i+1}\right)\times D_{\min}^2\le\frac{1-\frac{1}{j+1}}{1-\frac{1}{i+1}}\times M\times\Delta SSQ(\mathrm{I})<2M\Delta SSQ(\mathrm{I})$. □

概率流中的前 q 个数据元组作为 q 个微簇中心点后,若分别按选择策略 I 和选择策略 II 为不断新到达的数据元组选择候选簇并加入,则有初始值$\Delta SSQ(\mathrm{I})=\Delta SSQ(\mathrm{II})=0$,由定理 2 和增量的可加性可知,在任何时刻 t,都有$\Delta SSQ(\mathrm{II})<2M\times\Delta SSQ(\mathrm{I})$,这就保证了在概率流中的任何时刻 t,若分别按选择策略 I 和选择策略 II 为不断新到达的元组选择候选簇并加入,则选择策略 II 形成的簇 SSQ 值总小于选择策略 I 所形成的簇 SSQ 值的常数倍.

定理 3. 设当任意元组$\langle v,p\rangle$到达时,应用选择策略 II 引起的强簇数增量为$\Delta Cnum(\mathrm{II})$,应用选择策略 I 引起的强簇数增量为$\Delta Cnum(\mathrm{I})$,则$\Delta Cnum(\mathrm{II})\ge\Delta Cnum(\mathrm{I})$.

证明:分 3 种情况证明.

1) 应用选择策略 II 满足 Rule 1.这时有$\Delta Cnum(\mathrm{II})=1$,若应用策略 I,满足 $C_{\min}$ 是过渡簇,$C'_{\min}$ 是强簇,策略 II 也选择 $C_{\min}$,则$\Delta Cnum(\mathrm{II})=\Delta Cnum(\mathrm{I})=1$;否则,$\Delta Cnum(\mathrm{I})<1$,有$\Delta Cnum(\mathrm{II})\ge\Delta Cnum(\mathrm{I})$.

2) 应用策略 II 不满足 Rule 1 而满足 Rule 2.这时,$\Delta Cnum(\mathrm{II})=0$,应用策略 I 肯定不会满足 $C_{\min}$ 是过渡簇,$C'_{\min}$ 是强簇,否则,策略 II 会满足 Rule 1.所以,应用策略 I 不会使强簇数增多,即$\Delta Cnum(\mathrm{I})\le 0$,有$\Delta Cnum(\mathrm{II})\ge\Delta Cnum(\mathrm{I})$.

3) 应用策略 II 不满足 Rule 1 和 Rule 2 而满足 Rule 3.这时应用策略 II,选取增益值最大的簇 C_{j_i} 作为候选簇,若 C_{j_i} 是强簇,C'_{j_i} 是过渡簇,根据 Rule 3,应该选 $C_{\min}$ 为候选簇,这时有$\Delta Cnum(\mathrm{II})=\Delta Cnum(\mathrm{I})$;否则,$\Delta Cnum(\mathrm{II})=0$,而应用策略 I 不会使得 $C_{\min}$ 是过渡簇,$C'_{\min}$ 是强簇,否则策略 II 满足 Rule 1,所以$\Delta Cnum(\mathrm{I})\le 0$,有$\Delta Cnum(\mathrm{II})\ge\Delta Cnum(\mathrm{I})$.

由情况 1)~情况 3)的证明,无论哪种情况,均有$\Delta Cnum(\mathrm{II})\ge\Delta Cnum(\mathrm{I})$. □

由定理 2 和定理 3 可知,当任意数据元组$\langle v,p\rangle$到达并加入候选簇时,对所形成的簇的 SSQ 值,策略 II 不超过策略 I 的常数倍,并且策略 II 会比策略 I 找到相等或更多的强簇.

在选择策略 II 中,松弛参数 M 是一个敏感参数.当 $M=1$ 时,选择策略 II 完全不考虑簇的存在概率,这时退化为确定数据流算法 CluStream 中的候选策略 I.当 M 取无穷大时,排序后的簇序列 C'中全部簇都要进行扫描,这时簇的存在概率成为主要的考虑因素.当 M 在 1 到无穷大之间取值时,M 取值越大,扫描的簇数越多,优先满足 Rule 1 和 Rule 2 的可能性变大,从而形成的强簇数可能变大,但 SSQ 也可能随之增大.所以,参数 M 对平衡强簇

数和 SSQ 指标之间的矛盾起着重要作用.鉴于概率流数据本身概率分布和空间分布很难预先把握,我们考虑一种理想情况来为选择 M 提供参考.

设概率流中每个数据点的存在概率服从$[\theta,1]$上的均匀分布,当前所有微簇包含的元组数相等,设为 k.所有微簇包含的数据元组总数设为 A.进一步假设各个簇中只包含存在概率在 3 个区间$[\theta,\beta]$,$[\beta,\alpha]$,$[\alpha,1]$中某一个区间的元素,由簇的存在概率定义,这些簇分别是弱簇、过渡簇和强簇.且假设当前数据元组与排序后的簇序列 C' 中各簇的距离成比例增长,设比例因子为 r(r>1).我们应设法让当前到达数据优先选中 Rule 1 和 Rule 2,因为这两条规则都是把第 1 次满足各自条件的簇作为候选簇,既考虑了 SSQ,又使强簇数目保持不变或增大.Rule 1 和 Rule 2 满足的一个前提条件是候选簇必须是强簇或过渡簇.由以上假设,强簇或过渡簇出现的概率可近似为

$$\frac{\dfrac{(1-\alpha)A}{(1-\theta)k}}{\dfrac{A}{k}}+\frac{\dfrac{(\alpha-\beta)A}{(1-\theta)k}}{\dfrac{A}{k}}=\frac{1-\beta}{1-\theta}.$$

所以,平均在$\left\lceil\dfrac{1-\theta}{1-\beta}\right\rceil$个簇中可遇到 1 个强簇或过渡簇,有$1\le\dfrac{\left(r^{\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}\times D_{\min}\right)^2}{D_{\min}^2}\le M$,即$M\ge r^{2\times\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}$.

此结果是在理想假设下成立的,在后面的实验中会看到,设$r=\sqrt{2}$并且取M的下界值$M=r^{2\times\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}=2^{\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}$作为选择 M 的平均参考值对我们实验所采用的数据集是比较合适的.

3.2 模型过期处理方法Model_Outdated_Process

对于确定数据流上的聚类,CluStream 处理异常点一般采取为异常点新建一个簇的方法.在 CluStream 中,当一个到达元组不能加入候选簇(基于最近距离)时,要么删除一个基于时间衰退的“老”簇,要么合并两个最近的簇腾出一个微簇空间,为此数据元组建立一个新簇.然而,这种方法对概率流上的聚类有两个不足之处:

1) 删除“老”簇有可能删除一个强簇,从用户对最近数据更感兴趣的角度来看,删除“老”簇是合理的,但从对强簇感兴趣的角度来看,删除一个强簇是不可取的.

2) 若合并两个最近的簇,一方面,合并后簇的半径变大,从而最大边界变大,对合并前的两个簇都不能接受的某些元组,这时则有可能接受,使得应该是异常点的数据都被作为正常数据来处理,聚类质量下降;另一方面,合并两个簇只考虑了最近距离,其存在概率没有考虑,有可能把强簇和弱簇合并在一起,使强簇变成过渡簇甚至弱簇,这对于找到更多的强簇是不利的.

基于上述不足,本文采取基于自适应采样[21]的模型过期处理方法.在一段时间内到达的数据元组,只有当有“足够多”的数据被现有的簇接受时,才能说此时的聚类模型是适应的;否则,当有太多的异常点时,现有的聚类模型就不适应当前到达的数据了.为了以高概率确保当前聚类模型是适应的,我们可以利用文献[21]中的自适应采样方法来找到应被现有簇接受的最少数据元组个数.

设 X_t 是一个随机变量,若在 t 时刻到达的元组$\langle v,p\rangle$被 P-Stream 中的第 1 层的簇接受,则 X_t=1;否则,X_t=0.在任意时刻 t_0 开始的一段时间$[t_0,t_0+Sample_Num]$(*Sample_Num* 可看成是采样个数)内,设

$$X=\sum_{t=t_0}^{t=t_0+Sample\_Num}X_t,\ p=\Pr(X_t=1),$$

即 p 为一个数据元组被第 1 层簇接受的概率.由于在概率流的环境下,我们并不知道 p 的值,但有 p 的估算值 $\hat{p}=\dfrac{X}{Sample\_Num}$.对任意给定的参数$\varepsilon$和$\delta$,若满足以下不等式(1),则 $\hat{p}$ 能够以 1−δ的概率落在(1±ε)p 范围内.

$$\Pr(|\hat{p}-p|\le\varepsilon p)>1-\delta \tag{1}$$

为了满足不等式(1),我们可以应用以下定理来界定被第 1 层的簇成功接受的数据元组数目 S 以及采样数目 *Sample_Num* 满足的条件.

定理 4. 文献[21]在自适应采样中,为了满足不等式(1),被第 1 层的簇成功接受的数据个数 S 应满足下面的不等式(2).当 S 满足不等式(2)时,采样数目 *Sample_Num* 以大于 $1-\frac{1}{2}\delta$ 的概率满足不等式(3).

$$S > \frac{3(1+\varepsilon)}{(1-\varepsilon)\varepsilon^2 p} In\left(\frac{2}{\delta}\right) \tag{2}$$

$$Sample\_Num \le \frac{3(1+\varepsilon)}{(1-\varepsilon)\varepsilon^2 p} In\left(\frac{2}{\delta}\right) \tag{3}$$

由定理 4 中的不等式(3),若处理了 *Sample_Num*(取不等式(3)的上界值)个数据元组,被第 1 层成功接受的数据元组 S 若不满足不等式(2),则第 1 层的异常点太"多",此时,第 1 层的簇不适合当前以及以后将要到达的数据,应该把第 1 层的簇全部丢弃,作为数据演化的一个"拐点".我们采用"积极"的二层聚类模型来提前聚类孤立点,即在检测每个模型是否过期的开始,就"积极地"将第 1 层的异常点放在第 2 层,进行同步的异常点聚类.这些异常点聚类形成的簇,当第 1 层簇过期时,就代表了当前新的聚类模型而成为新的第 1 层,如图 1 所示.若两层都不接受元组$\langle v,p\rangle$,则元组$\langle v,p\rangle$在第 1 层模型和第 2 层模型都不适应而作为全局异常点丢弃.文献[22]中的模型过期处理办法不适合用在数据流中,因为文献[22]中的 Tracking 算法开始时就存储异常点而不进行异常点聚类,当判断当前的簇过期而把簇全部丢弃时,再对这些异常点重新聚类.这种算法在数据流中存储每个异常点而不是簇的概要数据结构 PCF,会产生较大存储空间,并且重新聚类会产生较大的在线延迟.

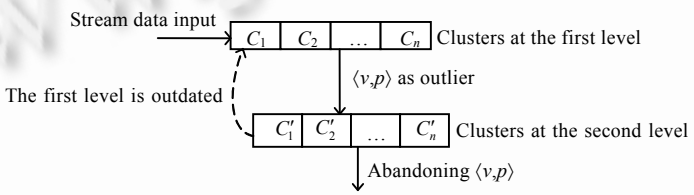


Fig.1 Two-Tiers clustering model

图 1 两层聚类模型

由不等式(2)、不等式(3)可知,S 值不依赖于 p,而 *Sample_Num* 值的确定依赖于 p,由于我们不知道 p,只能用估计值来代替它.类似于文献[22]中的 p 值估计方法,在第 1 层中开始的 p 值可以由前 Q 个点(比如前 100 个点)中成功聚类的元组个数 S 来确定($P=\frac{S}{Q}$).当有 *Sample_Num* 个元组到达后,检查第 1 层模型是否过期.若不过期,则利用算法 P-Stream 中的 *Mnormal* 和 *total*,把 $\frac{Mnormal}{total}$ 赋给 p 计算 *Sample_Num*,进行下一轮的模型过期检查,并且把 *total*,*Mnormal* 以及 *total'*,*Mnormal'* 都置为 0,重新开始计数.若第 1 层模型过期,则利用 P-Stream 中第 2 层统计的 *Mnormal'* 和 *total'*,把 $\frac{Mnormal'}{total'}$ 赋给 p 计算 *Sample_Num*,并把 *total'* 和 *Mnormal'* 的值相应地赋给 *total* 和 *Mnormal*,把 *total'* 和 *Mnormal'* 都置为 0,重新开始计数,开始对新的第 1 层进行模型过期检查.

3.3 检查点时间间隔值的确定方法CheckPoint_Process

设 P-Stream 中在 t 时刻计算的存储微簇快照的间隔时间为 $MinCheckTime_t$.由定义 4,概率流中簇的完整变化反映了簇的存在概率在整体上增大或减少的趋势,完整变化中的强簇或弱簇变为过渡簇是概率流中簇的一种重要演化形式.为了捕捉簇的这两种变化,检查点的时间间隔既不能太长,也不能太短,太长就有可能使得簇的连续两次微簇快照时段内中有 1 次或 1 次以上的完整变化,不能捕捉到完整变化中的强簇或弱簇变为过渡簇的变化趋势;太短就会造成频繁地存储微簇快照,使得在线的处理速度不能匹配概率流的输入速度.为此,我们考虑一个强簇变为弱簇的最短时间和一个弱簇变为强簇的最短时间.

定理 5. 若任意时刻 t,第 1 层聚类模型中各强簇中的数据元组个数最小值为 $SMIN_t>0$,则任何强簇变为弱

簇的最短时间 $T_S > SMIN_t \times \frac{\log(\alpha/\beta)}{\log(\beta/\theta)}$.

证明:设任意一个强簇 C_i,其元素个数为 n_i,$n_i \geq SMIN_t$,且 $EP^{C_i} \geq \alpha$. 簇 C_i 变弱的最快时间是每次到达的元组是最小存在概率(存在概率为 θ)的元组,且都加入簇 C_i 中,由性质 1,这样才能每次最大限度地减少簇 C_i 的存在概率.假设连续加入 T_s 个存在概率为 θ 的元组后,簇 C_i 变为弱簇,则

$$EP^{C_i'} = ((EP^{C_i})^{n_i} \times \theta^{T_S})^{\frac{1}{n_i+T_S}} < \beta.$$

两边取对数, $\frac{n_i}{n_i+T_S}\log EP^{C_i} + \frac{T_S}{n_i+T_S}\log\theta < \log\beta,$

解之,得到: $T_S > n_i \times \frac{\log(EP^{C_i}/\beta)}{\log(\beta/\theta)} \geq SMIN_t \times \frac{\log(\alpha/\beta)}{\log(\beta/\theta)}$. □

定理 6. 若任意时刻 t,第 1 层聚类模型中各弱簇中的数据元组个数最小值 $WMIN_t>0$,则任何弱簇变为强簇的最短时间 $T_w > WMIN_t \times \frac{\log(\alpha/\beta)}{\log(1/\alpha)}$.

证明:弱簇 C_i 变强的最快时间是每次到达的元组是最大存在概率(存在概率为 1)的元组,且都加入簇 C_i 中,其他证明与定理 5 类似. □

我们把 t 时刻检查点时间间隔值定义如下:

如果 $WMIN_t$ 和 $SMIN_t$ 都不为 0,则 $MinCheckTime_t = \min\left(SMIN_t \times \frac{\log(\alpha/\beta)}{\log(\beta/\theta)}, WMIN_t \times \frac{\log(\alpha/\beta)}{\log(1/\alpha)}\right)$;

如果 $SMIN_t=0$,$WMIN_t\neq 0$,则 $MinCheckTime_t = WMIN_t \times \frac{\log(\alpha/\beta)}{\log(1/\alpha)}$;

如果 $WMIN_t=0$,$SMIN_t\neq 0$,则 $MinCheckTime_t = SMIN_t \times \frac{\log(\alpha/\beta)}{\log(\beta/\theta)}$;

如果 $WMIN_t=SMIN_t=0$,则在 t 时刻第 1 层的聚类模型中只有过渡簇,我们可以一直等到 t' 时刻第 1 次出现了强簇或弱簇再重新计算 $MinCheckTime_{t'}$,在 t 到 t' 期间不存储微簇快照.

定理 7. 对任意的起始时刻 t,任何强簇或弱簇 C 都不会在时段 $[t,t+MinCheckTime_t]$ 内有 1 次或 1 次以上的完整变化.

证明:由定理 5 和定理 6 易得证. □

定理 7 保证了若概率流中某个强簇(或弱簇)C 有完整变化,则 C 处在过渡簇的状态会记录在 C 的某次微簇快照中,而不会只存在某连续两次的微簇快照之间而不被任何快照记录.这样就可以利用微簇快照来比较准确地分析簇存在概率的演化趋势.

在检查点时存储微簇快照的算法 *CheckPoint_Process*()如下:

```
CheckPoint_Process()
1.   {  if (在当前时刻 MID 变化了)     //第 1 层聚类模型在当前时刻已过期
2.        {根据当前新的第 1 层聚类模型中的微簇 PCF 集计算新的 MinCheckTime_t';Cnormal=0;}
3.      else if (Cnormal=MincheckTime_t)     //第 1 层聚类模型未过期,但检查点时刻已到
4.           {存储第 1 层的 PCF 集快照至磁盘,并用存储时间作索引;
5.           根据当前时刻第 1 层聚类模型中微簇 PCF 集计算新的 MinCheckTime_t';
6.           Cnormal=0;
7.         }}
```

3.4 离线高粒度聚类和演化分析

在线存储的微簇快照的主要作用就是把概率流中的数据转化为中间统计信息摘要,离线高粒度聚类就是

要利用这些信息摘要来进行.当用户的聚类请求(t,k)(即要求概率流中在 t 时刻所处的未过期模型中的 k 个簇)到达时,由存储的时间索引从磁盘快照中提取一个离时刻 t 最近的一次微簇快照 PCF 集.由性质 1,强簇和强簇合并仍是强簇,所以这时聚类不必考虑簇之间的存在概率关系.从 PCF 集中找出强簇,可用任意一种聚类方法(如 K-means 方法)聚类成用户指定的 k 个簇(当强簇个数大于 k 时),若强簇个数小于或等于 k,则将这些强簇直接返回给用户.聚类时,将每个微簇的中心点看成是伪点进行加权聚类[10],其权值是簇所包含的数据元组个数.

概率流上的聚类演化分析是在同一个聚类模型(即同一 MID)中的微簇快照集上进行的,因为不同 MID 内的微簇对应第 1 层不同的聚类模型,放在一起分析是不合适的.设某个新模型建立时刻为 t,过期时刻为 $t+Sample\_Num$,这段时间内 q 个微簇 $PCF_1,PCF_2,\ldots,PCF_q$ 存储了 N 次快照,每次存储的快照依次为

$$\{PCF_1^1,PCF_2^1,...,PCF_q^1\},\ldots,\{PCF_1^N,PCF_2^N,...,PCF_q^N\}.$$

演化分析分为两个方面:

1) 分析在同一模型 MID 内,每个簇所接受数据的分布演化情况.即对任意一个微簇 PCF_i,设前、后两个连续快照 PCF_i^j,PCF_i^{j+1} $(1\le i\le q,1\le j\le N-1)$的时间跨度为$\Delta T_j$.在$\Delta T_j$ 内,PCF_i 快照所包含数据元组个数的增量 Δn_i^j、中心点的移动增量 $\overrightarrow{\Delta C_i^j}$、半径大小的变化 ΔR_i^j.这些增量可利用 PCF_i^j,PCF_i^{j+1} 中 $\overrightarrow{CF^2},\overrightarrow{CF^1},n$ 得到.

(1) 比较不同微簇的 Δn_i^j $(1\le i\le q)$,反映了在第 j 次和第 $j+1$ 次快照之间,到达的元组主要被哪些簇所接受.Δn_i^j 越大,表明微簇 C_i 在ΔT_j 内接受的数据元组越多.

(2) 比较不同微簇 $\overrightarrow{\Delta C_i^j}$ $(1\le i\le q)$,反映了在第 j 次和第 $j+1$ 次快照之间,新到达的数据主要分布在哪几个方向.注意,$\overrightarrow{\Delta C_i^j}$ 是一个向量,q 个 $\overrightarrow{\Delta C_i^j}$ $(1\le i\le q)$越分散,表明新到达数据在空间内越分散;否则,表明新到达数据在ΔT_j 内分布的方向比较集中.

(3) 比较不同的 ΔR_i^j,反映了在第 j 次和第 $j+1$ 次快照之间,新到达的数据中,被每个簇接受的数据离其中心点远近的情况.当 ΔR_i^j 为正时,表明簇 C_i 接受的数据"拉大"了其半径,且 ΔR_i^j 越大,表明簇 C_i 接受的数据离其中心点平均距离越远.当 ΔR_i^j 为负时,表明簇 C_i 接受的数据"缩小"了其半径,且 ΔR_i^j 越小,表明簇 C_i 接受的数据离其中心点平均距离越近.

2) 分析在同一模型 MID 内,每个簇的存在概率的演化情况:

由定义 3,对每个簇 C_i,在 N 次快照中,定义 $\frac{PN}{\Delta n_i}$ 为簇 C_i 的正向变化率 PR_i,其中,PN 是 N 次快照中正向变化次数,Δn_i 是簇 C_i 在模型 MID 内到达的数据元组数.同理,定义 $\frac{NN}{\Delta n_i}$ 为簇 C_i 的负向变化率 NR_i,其中,NN 是 N 次快照中负向变化次数.

(1) 由定理 7,任何强簇或弱簇 C_i 的任何一次完整变化中,C_i 变为过渡簇都会记录在微簇快照中,在 MID 内若有 C_i 的快照从强簇变为过渡簇,则反映了这段时间内 C_i 的存在概率整体上是减少趋势;若有 C_i 的快照从弱簇变为过渡簇,则反映了这段时间内 C_i 的存在概率整体上是增大趋势.

(2) PR_i+NR_i 越小,说明簇 C_i 相对其他簇,在模型 MID 内,其存在概率保持得越稳定.

上面的情况 1)、情况 2)的演化分析是在同一模型 MID 内进行的.当用户想分析时间窗$[t,t+h]$内的聚类演化情况时,可分两种情况:

1. $[t,t+h]$内只包含同一个 MID 的微簇快照集.这时从磁盘中找出包含在$[t,t+h]$内的所有快照集,用上面情况 1)、情况 2)的方法进行演化分析.

2. $[t,t+h]$内包含不同 MID 的微簇快照集,则把不同 MID 的微簇快照分成若干含相同 MID 的快照子集,用上面情况 1)、情况 2)的方法对每个子集进行演化分析.

4 实验结果及其分析

所有实验在 2.0 GHz 的 PC 上进行,操作系统为 Windows 2000XP,算法用 Java 语言实现.尽管 CluStream 与 P-Stream 算法都工作在数据流环境下,都采用在线维护微簇和离线高层聚类和演化分析,并都采用类似于 BIRCH 算法中的微簇数据结构,但我们不把整个 CluStream 和 P-Stream 作比较.因为这两种算法在选择候选簇策略、存储微簇快照集方法以及处理异常点数据方法上都不相同,并且 CluStream 是针对确定数据流的,而 P-Stream 是针对概率流的.为了便于合理地比较实验结果,我们把 P-Stream 算法中采用 CluStream 中选择候选簇策略 I 的算法称为 CP-Stream,把 P-Stream 中采用 CluStream 中金字塔式时间框架存储微簇快照的算法称为 SP-Stream,把 P-Stream 中采用 CluStream 中处理异常点数据的算法称为 OP-Stream.我们把算法 P-Stream 分别与 CP-Stream,SP-Stream,OP-Stream 的实验结果进行比较.

实验数据中的概率数据包括两部分,其值部分分别采用真实数据集和人工合成数据集,存在概率部分均采用在$[\theta,1]$上均匀分布的合成数据.

值部分的真实数据集采用 KDD-CUP'99 和 KDD-CUP'98 两个数据集.这两个数据集也是文献[4]中的实验所采用的.KDD-CUP'99 是一个数据演化比较显著的数据集,由一个局域网中 TCP 连接的原始记录构成.各记录含有连接的时间、从源到目的传输的字节数等.数据集中的每条记录对应一个正常连接或者是 4 种网络攻击类型之一,如拒绝服务攻击、非授权访问远程机器攻击等.和文献[4,11]一样,我们从数据集中的 42 个属性中抽取 34 个连续属性作为概率数据的值部分.KDD-CUP'98 是一个相对稳定的数据集,该数据集记录的是慈善捐赠的信息,共有 95 412 条记录,对此数据集进行聚类,可反映捐赠行为的相似性.和文献[4]一样,我们从 481 个维度中选取 56 维的值作为概率数据的值部分,把记录的输入顺序模拟为数据流的到达顺序.

人工合成数据采用与文献[4]类似的方法,生成一系列满足高斯分布的数据.通过每 10K 个数据改变一次当前高斯分布的中心点和方差来模拟数据流的演化,并用如下常用的标记描述来人工数据集:B 表示数据集中包含的点数,C 表示簇的个数,D 表示数据的维数.例如,$B400C25D20$ 表示数据集中包含 400K 个数据点,分别属于 25 个簇,这些数据的维度是 20.

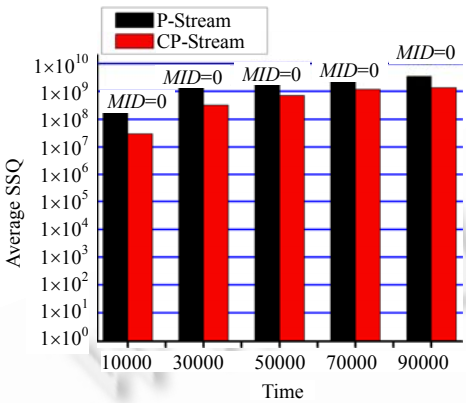
在所有实验中,不同算法都采用同样的 PCF 个数,即各算法使用的内存大小都是一样的.除非特别说明,各参数分别设为 θ=0.05,ε=0.1,δ=0.1.

4.1 聚类质量

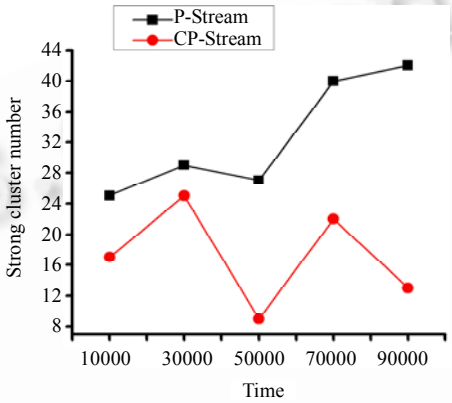
聚类质量评价标准采用 SSQ 和强簇数目.由于最能影响聚类质量的是选择候选簇的策略和处理异常点的方法,我们分别比较算法 P-Stream 与 CP-Stream,OP-Stream 的实验结果.算法 P-Stream 与 CP-Stream 的比较采用相对稳定的数据集 KDD-CUP'98.因为在演化比较显著的数据集上更能体现算法处理异常数据的优劣,算法 P-Stream 与 OP-Stream 的比较采用 KDD-CUP'99 数据集.为了保证结果更准确,算法采用不同的α,β和 M 值各执行 5 次,并计算平均 SSQ 和平均强簇数目作为结果值.

P-Stream 与 CP-Stream 的聚类质量比价如图 2 所示(慈善捐赠数据集).由于β的平均值为 0.2,θ=0.05,M 的平均参考值应为 $2^{\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}$=4,$M$ 的 5 次取值其平均值为 5,接近 M 的平均参考值.从图 2(a)中可以看出,如定理 2 所言,P-Stream 的 SSQ 不会超过 CP-Stream 的 10 倍.但在每个时标处,如图 2(b)所示,P-Stream 找到的强簇都比 CP-Stream 的要多,如在时标 90 000 处,强簇个数的差别最大.并且 P-Stream 找到强簇个数主要是呈增长趋势,而 CP-Stream 找到的强簇个数保持不稳定.我们又从存储的 2 271 次快照中发现,P-Stream 在一些连续的快照中记录的强簇个数比较稳定并且整体上是增加的,而 CP-Stream 在同样多的连续快照中时而增加,时而减少.这主要得益于 P-Stream 算法中的候选规则,在 M 不是很小的情况下,P-Stream 以应用 Rule 1 和 Rule 2 为主来找到候选簇,而这两条规则用以保持强簇个数不变或增大.而 CP-Stream 只考虑了最近距离而没有考虑簇的存在概率,致使强簇个数不稳定.从图 2(a)中我们还可以看出,在 5 个时标点,快照集中簇的 MID 都为 0,这表明在整个数据集的处理过程中,聚类模型都没有过期,从而说明 KDD-CUP'98 这个数据集比较稳定.

P-Stream 与 OP-Stream 的聚类质量比较如图 3 所示(网络入侵检测数据集).OP-Stream 严格按照 CluStream 中处理异常点的方法.从图 3(a)中我们可以看出,整个数据集的处理过程中模型变化了 7 次,这说明 KDD-CUP'99 这个数据集演化比较显著,并且整体上 P-Stream 的 SSQ 比 OP-Stream 要小.其原因是,当新到达的数据点为异常点时,采取合并簇为异常点建立新簇的方法会使合并簇的半径变大,致使以后到达的异常数据点被当作正常数据点而接受,导致半径有可能进一步增大,从而增大了 SSQ,降低了聚类质量.图 3(b)说明了 OP-Stream 在处理异常点时,采取合并簇或删除过时簇时可能会把强簇合并或删除强簇而致使强簇减少,所以,P-Stream 找到的强簇要比 OP-Stream 找到的多.图 3 表明,相比 CluStream 中处理异常点的方法,P-Stream 能够更有效地适应数据流演化的情况.

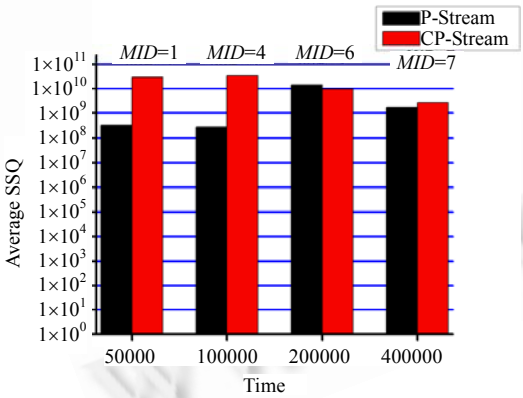


(a) SSQ comparison between P-Stream and CP-Stream
(a) P-Stream 与 CP-Stream 的 SSQ 比较

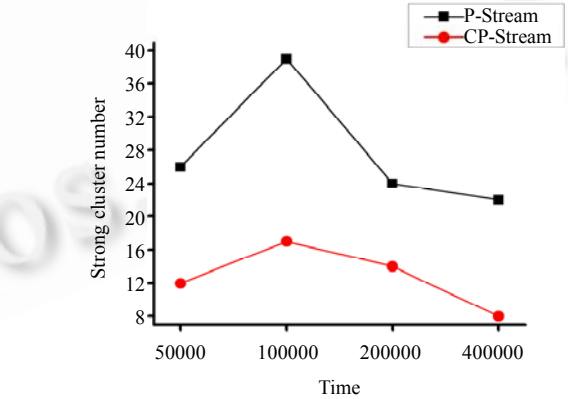


(b) Strong cluster number comparison between P-Stream and CP-Stream
(b) P-Stream 与 CP-Stream 的强簇数比较

Fig.2 Quality comparison between P-Stream and CP-Stream
图 2 P-Stream 与 CP-Stream 的聚类质量比较



(a) SSQ comparison between P-Stream and OP-Stream
(a) P-Stream 与 OP-Stream 的 SSQ 比较



(b) Strong cluster number comparison between P-Stream and OP-Stream
(b) P-Stream 与 OP-Stream 的强簇数比较

Fig.3 Quality comparison between P-Stream and CP-Stream
图 3 P-Stream 与 OP-Stream 的聚类质量比较

4.2 聚类处理时间

影响 P-Stream 时间开销的最主要部分是存储快照的频率.我们比较算法 P-Stream 与 SP-Stream 的处理时间,采用的数据集是在 KDD-CUP'99 上和合成数据上.

从图 4(网络入侵检测数据集)中我们可以看出,P-Stream 处理的时间效率要优于 SP-Stream,并且算法执行

时间几乎都是随数据流长度的增加而呈线性增长,且各条直线具有不同的斜率.注意到低斜率对应高处理速度,因而 P-Stream 处理速度比 SP-Stream 更加高效.由于 SP-Stream 采取金字塔式时间框架的方法存储快照,几乎在每个时标点要存储快照一次.我们从 P-Stream 中的快照集发现,存储快照集的时间间隔最短是 3,最长的是 842.所以,P-Stream 平均存储快照的次数比 SP-Stream 要少.

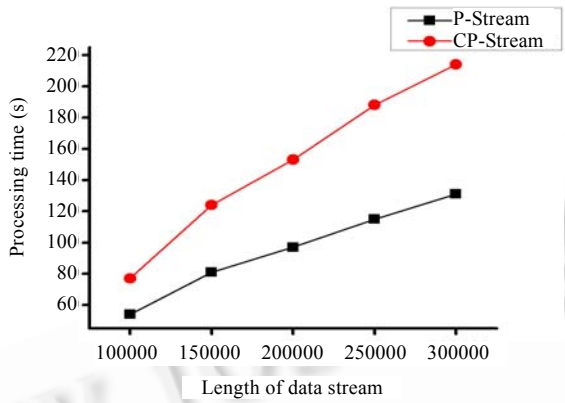
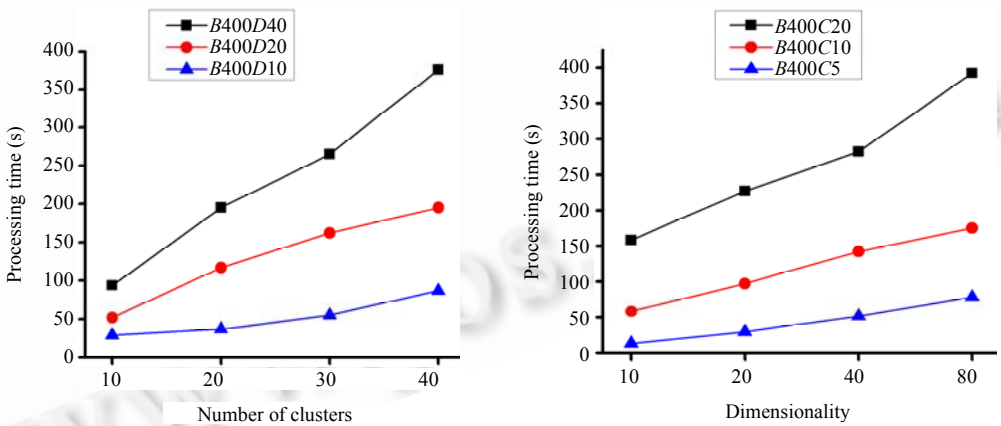


Fig.4 Processing time vs. length of data stream

图 4 处理时间随数据流长度变化

然后,我们又采用不同维度和不同簇个数的人工合成数据流对 P-Stream 的处理时间进行评价.由于人工数据集可以得到任意的维度和簇个数的组合数据流,我们生成了一系列的数据集.如图 5 所示,我们固定数据流的长度和各维度,簇的个数从 10 变化到 40,P-Stream 的处理时间随簇个数的增加几乎呈线性增长.同时,我们也固定数据流的长度和各簇的个数,维度从 10 维变化到 80,P-Stream 的处理时间随维度的增加也几乎呈线性增长.



(a) Processing time vs. number of clusters

(a) 处理时间随簇个数的变化

(b) Processing time vs. dimensionality

(b) 处理时间随维数的变化

Fig.5

图 5

4.3 参数影响

影响 P-Stream 的强簇个数和 SSQ 的两个主要参数是强簇下界α和松弛参数 M,比较不同的 M 值,可以验证 M 的平均参考值是否合理.我们在数据集 KDD-CUP'98 中设置了不同的α和 M 来考察它们与强簇数目和 SSQ 的关系.

我们先固定α,分别设置 M 为 2,5,10 和 15.见表 1,SSQ 和强簇数目 *StrNum* 随 M 的增大而增大,但 SSQ 增长最多的是在 M 从 10 变化到 15 这一段.这说明,当松弛参数 M 变大时,P-Stream 在寻找候选簇时有更大的选择范围,致使最先用 Rule 1 找到候选簇的可能性变大,从而使得强簇数量不断增大.但强簇数目增大是以增大 SSQ 为代价的,所以不同应用中可根据需要来设置 M 取得 SSQ 和强簇数的折衷.就 KDD-CUP'98 这个数据集而言,M 取 5 相对于 M 取 10 和 15 要好,因为 M 取 5 增加的强簇数 *StrNum* 比 M 取 10 和 15 增加的强簇数都要多,而增加的 SSQ 比 M 取 10 和 15 增加的 SSQ 都要少.在 KDD-CUP'98 这个数据集中,我们设置的 M 平均参考值是 $2^{\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}$=4,此时,$M$ 取 5 比取 10 和 15 更接近 M 的参考值,所以说,$2^{\left\lceil\frac{1-\theta}{1-\beta}\right\rceil}$=4 作为 M 的平均参考值是合理的.

Table 1 Impact of parameter M on clustering quality

表 1 参数 M 对聚类质量的影响

M=2		M=5		M=10		M=15	
SSQ	*StrNum*	SSQ	*StrNum*	SSQ	*StrNum*	SSQ	*StrNum*
5.52E+09	17	7.92E+09	38	1.04E+10	44	1.43E+10	49

然后我们固定 M 的值,分别设置α为 0.7,0.8,0.9 和 0.97.从表 2 中可以看出,α值对 SSQ 值的影响不是很明显.但当α不断变大时,强簇数目在不断减少.例如,当α设为 0.97 时对形成强簇的条件非常苛刻,导致强簇数目为 0.

Table 2 Impact of parameter α on clustering quality

表 2 参数α对聚类质量的影响

α=0.7		α=0.8		α=0.9		α=0.97	
SSQ	*StrNum*	SSQ	*StrNum*	SSQ	*StrNum*	SSQ	*StrNum*
5.57E+09	32	6.02E+09	18	5.82E+09	6	6.47E+09	0

5 结论和展望

本文提出一种概率数据流上的有效聚类方法 P-Stream.P-Stream 比 CluStream 中基于最近距离的候选簇选择策略能找到更多的强簇,但其 SSQ 不会超过后者方法的常数倍.P-Stream 找到了合理存储微簇快照的时机,既不会像 CluStream 中金字塔式时间框架方法那样频繁存储快照,又准确地捕捉到了数据流的演化情况.最后,P-Stream 用两层的过期模型方法来处理异常点,整体上,其 SSQ 和强簇数都要优于 CluStream 中采取合并或删除簇来处理异常点的方法.实验结果表明,在概率数据流中,本文提出的 P-Stream 算法相对于 CluStream 中的传统方法有较好的聚类质量、较快的处理时间,能够有效地适应数据流的演化情况.今后,我们将对概率数据流在滑动窗口模型下的聚类问题进行研究.

References:

[1] Cormode G, Garofalakis M. Sketching probabilistic data streams. In: Chan CY, Ooi BC, Zhou A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Beijing: ACM Press, 2007. 281−292.

[2] Jayram TS, McGregor A, Muthukrishan, Vee E. Estimating statistical aggregates on probabilistic data streams. In: Libkin L, ed. Proc. of the 26th ACM SIGMOD-SIGACT-SIGART Symp. Principles of Database Systems. Beijing: ACM Press, 2007. 243−252.

[3] Jayram TS, Kale S, Vee E. Efficient aggregation algorithms for probabilistic data. In: Bansal N, Pruhs K, Stein C, eds. Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA). New Orleans: SIAM, 2007. 346−355.

[4] Aggarwal CC, Han J, Yu PS. A framework for clustering evolving data streams. In: Freytag JC, Lockmann PC, Abiteboul S, Carey MJ, Seling PG, Heuer A, eds. Proc. of the Int'l Conf. on Very Large Data Bases. Berlin: Morgan Kaufmann Publishers, 2003. 81−92.

[5] Dalvi N, Suciu D. Efficient query evaluation on probabilistic databases. In: Nascimento MA, Ozsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the VLDB. Toronto: Morgan Kaufmann Publishers, 2004. 864−875.

[6] Burdick D, Deshpande PM, Jayram TS, Ramakrishnan R, Vaithyanathan S. OLAP over uncertain and imprecise data. In: Bohm K, Jensen CS, Haas LM, Kersten ML, Larson P, Ooi BC, eds. Proc. of the Int'l Conf. on Very Large Data Bases. Trondheim: ACM

Press, 2005. 970−981.

[7] Sarma AD, Benjelloum O, Halevy A, Widom J. Working models for uncertain data. In: Liu L, Reuter A, Whang KY, Zhang J, eds. Proc. of the 22nd Int'l Conf. on Data Engineering. Atlanta: IEEE Computer Society, 2006.

[8] Cheng R, Kalashnikov D, Prabhakar S. Querying imprecise data in moving object environments. IEEE Trans. on Knowledge and Data Engineering, 2004,16(9):1112−1127.

[9] Ngai WK, Kao B, Chui CK, Cheng R, Chau M, Yip KY. Efficient clustering of uncertain data. In: Cliton CW, Zhong M, Liu JM, Wah BW, Wu XD, eds. Proc. of the 6th IEEE Int'l Conf. on Data Mining. Hong Kong: IEEE Computer Society, 2006. 436−445.

[10] Guha S, Mishra N, Motwani R, Callaghan LO. Clustering data streams. In: Yong DC, ed. Proc. of the 41st Annual Symp. on Foundations of Computer Science. Redondo Beach: IEEE Computer Society, 2000. 359−366.

[11] Guha S, Meyerson A, Mishra N, Motwani R, Callaghan LO. Clustering data streams: Theory and practice. IEEE Trans. on Knowledge and Data Engineering, 2003,3(15):515−528.

[12] Callaghan LO, Mishra N, Meyerson A, Guha S, Motwani R. Streaming-Data algorithms for high-quality clustering. In: Agrawal R, Dittrich K, Ngu AHH, eds. Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 685−694.

[13] Charikar M, Callaghan LO, Panigragh R. Better streaming algorithms for clustering probelms. In: Proc. of the 35th Annual ACM Symp. on Theory of Computing. San Diego, 2003. 30−39.

[14] Zhou A, Cai Z, Wei L, Qian W. M-Kernel merging: Towards density estimation over data streams. In: Titsworth F, ed. Proc. of the Int'l Conf. on Database System for Advanced Applications (DASFAA). Kyoto: IEEE Computer Society, 2003. 12−19.

[15] Nam H, Won S. Statistical grid-based clustering over data streams. SIGMOD Record, 2004,33(1):32−37.

[16] Aggarwal CC, Han J, Yu PS. A framework for projected clustering of high dimensional data streams. In: Nascimento MA, Ozsu MT, Kossmann D, Miller RJ, Blakeley JA, Schiefer KB, eds. Proc. of the VLDB. Toronto: Morgan Kaufmann Publishers, 2004. 852−863.

[17] Babcock B, Datar M, Motwani R, Callaghan LO. Maintaining variance and *k*-medians over data stream windows. In: Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems. San Diego: ACM Press, 2003. 234−243.

[18] Chen YX, Tu L. Density-Based clustering for real-time stream data. In: Berkhin P, Caruana R, Wu X, eds. Proc. of the 13th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. San Jose: ACM Press, 2007. 133−142.

[19] Datar M, Gionis A, Indyk P, Motwani R. Maintaining stream statistics over sliding windows. In: Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA). San Francisco: SIAM, 2002. 635−644.

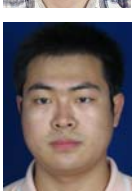
[20] Zhang T, Ramakrishnan R, Livny M. Birch: An efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS, eds. Proc. of the SIGMOD. Montreal: ACM Press, 1996. 103−114.

[21] Watanabe O. Simple sampling techniques for discovery science. IEICE Trans. on Information and Systems, 2000,E83-D(1):19−26.

[22] Barbara D, Chen P. Tracking clusters in evolving data sets. In: Russell I, Kolen JF, eds. Proc. of the 14th Int'l Florida Artificial Intelligence Research Society Conf. Key West: AAAI Press, 2001. 239−243.

戴东波(1977－),男,湖南娄底人,博士生,主要研究领域为数据挖掘,生物信息.

孙圣力(1979－),男,博士,CCF 学生会员,主要研究领域为基于流数据的查询处理,数据挖掘,分布式数据库.



赵杠(1980－),男,博士生,主要研究领域为数据挖掘,隐私保护,高维计算几何.