文章编号:2095-476X(2014)01-0098-05

基于动态滑动窗口的改进数据流聚类算法

许颖梅

(商丘师范学院 计算机与信息技术学院, 河南 商丘 476000)

摘要:提出一种采用滑动窗口处理数据的优化算法DCluStream. 该方法基于CluStream算法双层框架思想,在聚类特征中引入数据流入和流出滑动窗口的实际时间,动态调整窗口大小以适应有限内存;对历史数据通过时间衰减机制来降低它对新数据对象的影响,使聚类效果更好. 实验结果表明,与CluStream相比,本算法处理数据的效率更高且相对节约内存.

关键词:滑动窗口;数据流聚类算法;时间衰减机制

中图分类号:TP311 **文献标志码**:A DOI:10.3969/j.issn.2095-476X.2014.01.021

Improved data stream clustering algorithm over sliding window

XU Ying-mei

(*College of Computer and Information Technology*, *Shangqiu Normal University*, *Shangqiu* 476000, *China*)

Abstract: An optimization algorithm DCluStream was proposed which processed data over sliding window. The method adopted online-offline clustering framework of CluStream. The real time of the data object coming and out of sliding window was introduced into the characteristics of the cluster, adjusting the window size reasonably in the limited memory resources environment. Using the time decay mechanism on historical data could reduce the impact of new data object, which could get better clustering results. The experimental results showed that compared with the algorthm CluStream, data processing efficiency of the algorithm was relatively higher with saving memory.

Key words: sliding window; data stream clustering algorithm; time decay mechanism

0 引言

数据流就是连续到达的一个序列,具有无限大且不可预知性. 对数据流的查询结果往往不是一次性而是持续的,即随着底层数据的到达而不断返回最新的结果. 数据流聚类算法作为数据流挖掘的工具,具有很好的研究和应用前景,也是目前应用研究的热点. 聚类就是按一定特征将一个对象的集合分成若干个类,每个类内的对象是相似的,但与其他类的对象是不相似的[1].

数据流聚类已经有很多算法. S. Guha等[2]提出了Localsearch算法,在有限的空间内对数据流进行聚类,使用一个不断迭代的过程进行k-means聚类. L. O'Callaghan等[3]在Localsearch的基础上又提出了Stream算法,但这种算法是基于静态数据流的,不能反映数据流的变化情况. C. C. Aggarwal等[4]提出了一个解决数据流聚类问题的框架CluStream,将数据流的聚类分成在线微聚类和离线宏聚类2个阶段. 周晓云等[5]提出基于Hoeffding界的高维数据流的子空间聚类发现及维护算法SHStream,在数据分

收稿日期:2013-09-17

基金项目:河南省基础与前沿技术研究计划项目(132300410395;122300410395)

作者简介:许颖梅(1979—),女,河南省商丘市人,商丘师范学院讲师,硕士,主要研究方向为网络安全.

段上进行子空间聚类,通过迭代逐步得到满足聚类精度要求的聚类结果. 杨春宇等[6]基于数据流的连续属性和标称属性提出一种适用于处理混合属性数据流的聚类算法 HCluStream,可为混合属性构建新的信息汇总方式及距离度量. 吴枫等[7]在数据流聚类形状问题上提出了一种滑动窗口内进化数据流任意形状聚类算法 SWASCStream. 周傲英等[8]提出了基于滑动窗口的数据流聚类算法 CluWin,通过拒伪和纳真解决了滑动窗口中的误差问题. 在以后的研究中又出现了新的研究方向,比如基于网格和密度的不确定数据研究,还有支持泛在应用的数据流聚类[9],在滑动窗口中实现对数据流的裁剪和增量更新,提高了数据挖掘的效率.

但是,上述算法也都有一定的局限性,如内存占用率高、效率低下等,鉴于此,本文拟提出一种基于动态滑动窗口的改进数据流聚类算法.

1　相关概念

1.1　问题定义

定义 1　数据流是由数据项 $< i_1, t_1^1, t_1^2 >$, $< i_2, t_2^1, t_2^2 >$, … 组成的无限集合,其中,i 表示数据流中的元组,t_i^1 表示此元组流入滑动窗口的时刻,t_i^2 表示此元组流出滑动窗口的时刻.

定义 2　对数据流中的数据取样本集 $D = \{x_1, x_2, \cdots, x_i, \cdots, x_n\}$,从中挖掘出具有相似程度的 k 个数据簇 ($\{C_1, C_2, \cdots, C_k\}$), 其中 $D = \bigcup_{i=1}^{k} C_i, C_i \cap C_j = \varphi, i \neq j$. 同一簇中的对象之间是相似的,不同簇中的对象是相异的.

定义 3　给定最小支持度阈值 δ 和误差因子 ε,假设 $|W|$ 表示滑动窗口 W 的宽度,即 W 中包含的事务数,$f_w(A)$ 表示模式 A 在滑动窗口中的支持度计数. 对于模式 A,如果有 $f_w(A) \geqslant \delta |W|$,则称 A 为滑动窗口 W 中的微簇;如果有 $f_w(A) \geqslant \varepsilon |W|$,则称 A 为滑动窗口 W 中的临界微簇;如果有 $f_w(A) < \varepsilon |W|$,则称 A 为滑动窗口 W 中的过期微簇.

定义 4　数据流聚类特征是定义在线聚类阶段的数据集. 对于数据项 $< i_1, t_1^1, t_1^2 >$, $< i_2, t_2^1, t_2^2 >$, … 组成的无限集合,该数据项上的微聚类特征表示为 $CF = \{W', n, F, Q, t_1, t_2\}$. 其中,$W'$ 为此时窗口的实际大小,n 为簇中数据的个数,$F^j = \sum_{k=1}^{n} X_k^j$,表示元组中的数据在第 j 维的一阶距,$Q^j = \sum_{k=1}^{n} (X_k^j)^2$,表示元组中的数据在第 j 维的二阶距,t_1 表示数据流进滑动窗口的时刻,t_2 表示数据流出滑动窗口的时刻.

1.2　时间衰减机制

随着数据源源不断地流入,在数据流聚类过程中,也应该有些过期的数据被淘汰,这就要采用一定的衰减机制对过期元组进行衰减. 本文采用时间衰减模型,在这种模型中,数据流中每个项集都有一个权重. 权重随时间改变,新到来的项集对该项集的频度影响大于原来的项集. 在时刻 t,每个元组的衰减因子的大小满足 $2^{-\lambda t} < \varepsilon (\lambda > 0)$,其中 ε 表示时间界定阈值,衰减系数 λ 值越大,过去数据的重要性就越低.

数据流总的权重

$$W(t) = v \sum_{t=0}^{t_c} 2^{-\lambda t} = \frac{v}{1 - 2^{-\lambda}}$$

其中,t_c 表示当前时间,v 表示数据流的流速.

2　改进的数据流聚类算法

2.1　算法思想

本算法基于 CluStream 的 2 层聚类框架思想,在动态调整滑动窗口的基础上将挖掘过程分为在线和离线 2 个过程. 在线过程不断接收数据流摘要信息,利用 k-means 算法从初始样本集中挖掘出一定数量的微簇更新到内存结构中,其产生的结果作为挖掘的中间结果维护起来,一定时间后将这些中间结果保存到外存中作为离线过程的初始数据. 离线过程由用户调用,针对用户的查询,以在线聚类阶段形成的微聚类为基础进行离线聚类,利用衰减因子对微聚类进行动态维护,及时更新和衰减,得到相应时间段内的宏聚类. 通过在线和离线 2 个过程的不同算法,实现动态数据的快速处理.

在线聚类过程可以分为微簇初始化、更新及删减 3 个步骤. 假设数据流的流速是均匀的,初始化时滑动窗口大小设定为 W,数据流入窗口的时间点为 T_1,流出窗口的时间点为 T_2,那么数据匀速流入时在滑动窗口内的驻留时间 $\Delta T = T_2 - T_1$. 但现实中,数据流的流速是不断变化的,假设 t_1 为数据流入窗口的时刻,t_2 为数据流出窗口的时刻,数据项在窗口中的实际停留时间为 $\Delta t = t_2 - t_1$.

假设时间界定阈值为 ε,$\Delta t - \Delta T > \varepsilon$ 时,意味着数据传输较慢,此时滑动窗口较大,浪费了内存开销;$-\varepsilon \leqslant \Delta t - \Delta T \leqslant \varepsilon$,说明数据流速度接近匀速,此

时滑动窗口的大小是适中的;$\Delta t - \Delta T < -\varepsilon$,此时数据流的传输速度很快,而滑动窗口的大小相对较小,需适当增大.

因此,适当调整滑动窗口的大小,可以降低算法的复杂度. 设 ΔW 为窗口调整变量的阈值,实际窗口大小 W'. 在第 1 种情况下,$W' = W - \Delta W$;第 2 种情况下,$W' = W$,不需调整;第 3 种情况下 $W' = W + \Delta W$. 经过以上调整,数据流在滑动窗口内基本保持匀速,这样既可以使得算法适应数据流的流速,也使内存得到充分利用.

2.2 算法公式

在对数据点进行聚类的过程中用到以下几个距离公式.

元组之间的距离为

$$D(X_a, X_b) = \sqrt{\sum_{j=1}^{d} (X_a^j - X_b^j)^2}$$

式中,X_a^j,X_b^j 分别为元组 X_a 和 X_b 的第 j 维.

元组到聚类中心点的距离为

$$D(X_a, CF_b) = \sqrt{\sum_{j=1}^{d} \left(X_a^j - \frac{F_b^j}{n}\right)^2} \qquad ①$$

式中,F_b^j/n为聚类中心 F_b/n的第 j 维.

聚类中心之间的距离为

$$D(CF_a, CF_b) = \sqrt{\sum_{j=1}^{d} \left(\frac{F_a^j}{n} - \frac{F_b^j}{n}\right)^2}$$

2.3 在线层算法

该算法在第 1 个元组进入滑动窗口后,形成 1 个微聚类特征,随着数据的流入,当判断新到达的元组可以加入已有微聚类时,对该微聚类特征进行更新;若新到达元组是一个新的微聚类时,看此时微聚类是否已饱和,若是,则通过计算合并最近的 2 个微聚类,否则产生新的微聚类,同时对新建的微簇的概要信息进行更新. 而在数据流入时需检测是否要调整滑动窗口的大小,计算后决定对数据在窗口内停留的时间做怎样的调整. 图 1 是在线层算法的执行流程.

整个过程包括微聚类初始化、计算元组之间的距离、聚类合并或生成新的微聚类、调整窗口大小、输出微聚类,算法描述如下:

Input:数据流 DS,窗口大小 W,窗口可调整的阈值 ΔW,数据项在窗口内停留的时间界定阈值 ε,微簇半径阈值 R,预定义所容纳的微聚类个数 M.

Output:微聚类数 n.

DClu-Online($DS, W, \Delta W, \varepsilon, R, M$)

Begin

$n = 0$;/ * 对聚类数目初始化 * /

对数据流 DS 中的每个到达的元组 X_i;

通过上一节公式①计算数据元组 X_i 与每一聚类特征 CF 之间的距离 $D(X_i, CF)$,从中找出相距最近的那个微聚类;

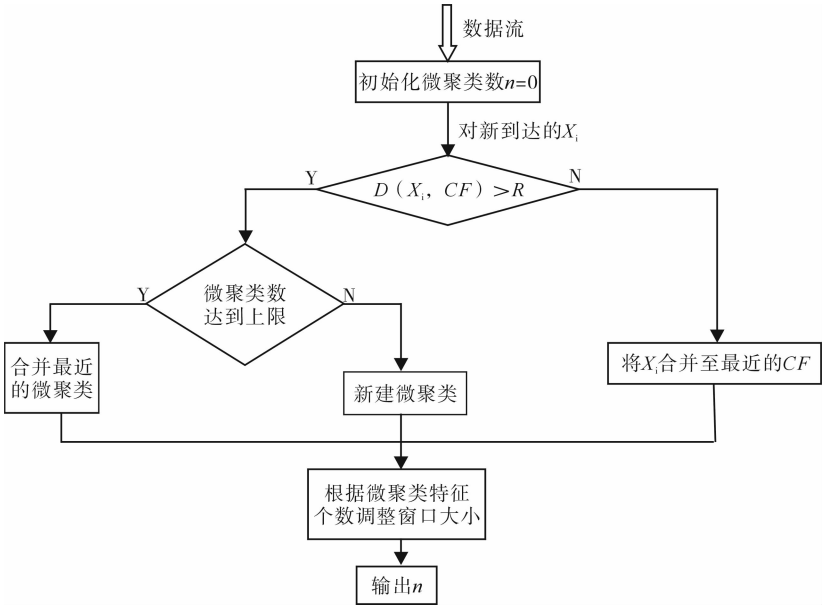

图 1 在线层算法的执行流程图

If($D(X_i, CF) > R$)

{If($n = M$)

{合并相距最近的 2 个微聚类特征;

n = 此时微聚类数目;}

新建微聚类,并对微聚类特征中的每一项进行更新;

n = 此时微聚类数目;}

Else

元组 X_i 合并至与它相距最近的那个微聚类 CF 中;

If($\Delta t - \Delta T > \varepsilon$)

$W' = W - \Delta W$;

Else if($-\varepsilon \leqslant \Delta t - \Delta T \leqslant \varepsilon$)

$W' = W$;

Else

$W' = W + \Delta W$;

输出 n;

End

2.4　离线层算法

离线层通常分析某时间段的聚类结果,针对用户的查询以在线聚类阶段形成的微聚类为基础进行离线聚类,利用衰减因子对微聚类进行动态维护,及时更新和衰减,得到相应时间段内的宏聚类. 算法中 t_1,t_2 为 2 个较近的时间点,时间表阈值为 ε.

算法实现如下:

Begin

判断 t_1,t_2 为 2 个合法的时间点;

将 t_1 时刻的概要信息作为该时刻的中心微簇;

for 在内存中存储的每一个微聚类特征 CF

$W(t) = v\sum_{t=0}^{t_c} 2^{-\lambda t} = \frac{v}{1-2^{-\lambda}} < \varepsilon$,每一微聚类特征按权重进行衰减;

endfor

采用 k-means 算法对内存中的微聚类特征进行聚类,生成 k 个聚类;

End

3　实验分析

本实验是在配置为 Intel PentiumIV 3.0 GHz,内存 1 GB 的 PC 机上实现的,操作系统是 Windows XP. 所有程序采用 Visual C + + 开发环境实现,并与基于界标窗口模型的 CluStream 算法进行性能比较.

实验中所使用的数据是将网络入侵检测数据集 KDDCUP99 与 IBM 合成数据发生器产生的数据集 T1516D1000K 融在一起. KDDCUP99 数据集共包含 283 490 条数据记录,每条数据记录有 41 维固定特征属性,对其中 22 个连续型、9 个离散型共 31 个与本实验相关属性进行分析. 数据集 T1516D1000K 共包含 305 732 条数据记录,每条记录包含 50 维属性,其中,数值属性 44 维,分类属性 6 维.

首先比较了在相同最小支持度阈值下 2 个算法对 1 000 K 事务的平均处理时间,取最小支持度阈值 $\delta = 0.5\%$,图 2 给出了 DCluStream 算法与 CluStream 算法随事务到达的平均处理时间对比. 实验结果表明,DCluStream 算法时间效率明显高于 CluStream 算法.

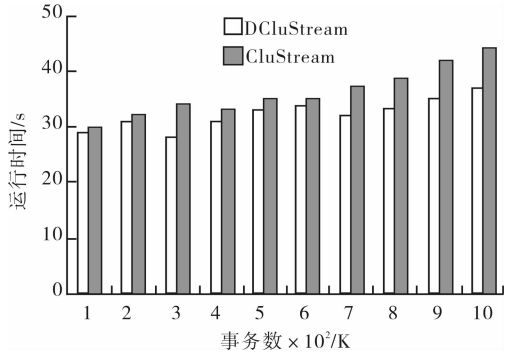

图 2　2 种算法相同事务数的平均处理时间比较

接下来对内存使用情况进行比较. 依然选取 2 个数据集产生的 1 000 K 个事务,图 3 是处理 KDDCUP99 和 T1516D1000K 数据集的试验比对结果. 图 3 显示,随着数据流量的增多,DCluStream 的内存节省率高于 CluStream,说明有效的衰减机制能够明显地节约内存开销.

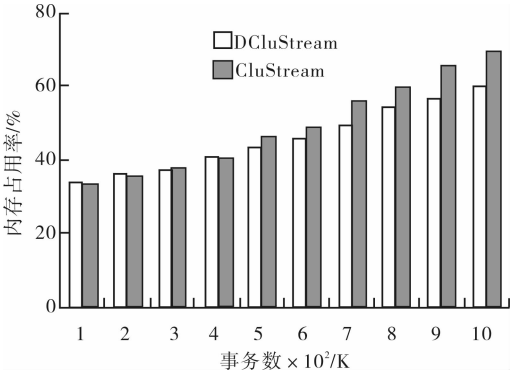

图 3　2 种算法内存使用情况比较

4 结论

本文提出了一种基于动态滑动窗口的数据流聚类算法,它是在 CluStream 算法双层框架(在线和离线)基础上,在线阶段在聚类特征中引入数据流入和流出滑动窗口的实际时间,并可以动态调整窗口大小,解决了有限内存存储无限数据的可能. 离线阶段借助在线层保存的数据流概要信息,根据用户需要,对概要数据通过 k-means 算法进行宏聚类,并采用时间衰减机制对历史数据进行衰减,使聚类结果更合理. 实验结果表明,改进的动态滑动窗口的数据流处理算法在准确度和运行效率上都有所提高,且更节约内存开销.

参考文献:

[1] 金澈清,钱卫宁,周傲英. 流数据分析与管理综述[J]. 软件学报,2004,15(8):1172.

[2] Guha S,Mishra N,Motwani R,et al. Clustering data streams [C]//Proceedings of 41st Annual Symposium on Foundations of Computer Science,Los Alamitos,CA:IEEE Computer Society Press,2000:359.

[3] O'Callaghan L,Mishra N,Meyerson A,et al. Streaming data algorithms for high-quality clustering[C]// Proceeding of 18th Internationl Conference on Data Engineering. Los Alamitos, CA: IEEE Computer Society Press, 2002:685.

[4] Aggarwal C C,Han J,Wang J,et al. A framework for clustering evolving data streams[C]//Proceeding of 29th Internationl Conference on Very Large Data Bases,Berlin: Morgan Kaufmann,2003:81.

[5] 周晓云,孙志挥,张柏礼,等. 高维数据流子空间聚类发现及维护算法[J]. 计算机研究与发展,2006,43(5):834.

[6] 杨春宇,周杰. 一种混合属性数据流聚类算法[J]. 计算机学报,2007,30(8):1364.

[7] 吴枫,仲妍,金鑫,等. 滑动窗口内进化数据流任意形状聚类算法[J]. 小型微型计算机系统,2009,30(5):887.

[8] 常建龙,曹锋,周傲英. 基于滑动窗口的进化数据流聚类[J]. 软件学报,2007,18(4):905.

[9] 宋宝燕,张衡,于洋,等. 基于滑动窗口的支持泛在应用的流聚类挖掘算法[J]. 小型微型计算机系统,2008,29(12):2262.