基于密度网格的数据流聚类算法

米 源 杨 燕 李天瑞
(西南交通大学信息科学与技术学院 成都 610031)

摘 要 针对基于密度网格的数据流聚类算法中存在的缺陷进行改进,提出一种基于D-Stream算法的改进算法NDD-Stream。算法通过统计网格单元的密度与簇的数目,动态确定网格单元的密度阈值;对位于簇边界的网格单元采用不均匀划分,以提高簇边界的聚类精度。合成与真实数据集上的实验结果表明,算法能够在数据流对象上取得良好的聚类质量。

关键词 数据挖掘,数据流,聚类分析,密度网格,不均匀划分

Data Stream Clustering Algorithm Based on Density Grid

MI Yuan YANG Yan LI Tian-rui
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, China)

Abstract On the basis of improvements on defects in data stream clustering algorithm based on density grid, a data stream clustering algorithm was proposed which improved D-Stream algorithm. The algorithm set density threshold of grid cell dynamically by statistics on density of grid cell and number of clusters. To increase the precision of cluster boundary, a non-uniform division was employed on the grid boundary cell. The result of experiments on synthetic and real data set shows that the algorithm has fast processing speed and the ability to detect dynamic changes of data for data stream clustering, and improves clustering quality.

Keywords Data mining, Data stream, Cluster analysis, Density grid, Non-uniform division

1 引言

聚类分析作为数据挖掘的一种重要方法,在各领域中有着广泛的应用。随着网络通信与硬件技术的高速发展,在气象与环境监控、股票零售、Web点击流、科学和工程实验等动态环境当中产生了大量的实时数据。这些数据与传统数据库当中的数据相比,具有快速连续到达、顺序访问、动态变化、数据量无限等特点。我们称这种数据为数据流。

数据流聚类分析是聚类分析在数据流环境下的延伸,具有重要的研究价值。由于数据流种种特性的限制,传统聚类算法无法适用于数据流对象,因此数据流聚类算法采用传统聚类算法的基本思想加以改进,其本身也带有传统聚类算法的特性。

2 问题分析

定义1(数据流,Data Stream) Henzinger等人在文献[1]中首次将数据流作为一种数据处理模型提出来:数据流为一个不断增长的d维元组的集合$\{\vec{X}_1,\vec{X}_2,\cdots,\vec{X}_i,\cdots\}(i\geqslant 1)$,每个元组$\vec{X}_i$均有一相应的时间戳$T_i$(Time Stamp),对于任意的$l<m$,均有$T_l<T_m$。

传统的聚类算法大都使用基于距离的度量方法,聚类结果偏好于球形,在非凸形分布的数据上聚类质量较差。基于密度的聚类方法能够发现任意形状的簇,而单纯基于密度的聚类方法其计算量与数据对象的数目成正比,不适用于大数据量的场合。基于密度网格的聚类方法能够发现任意形状的簇,且处理速度与具体对象的数目无关,以牺牲少量精度为代价换取时间效率的提高,适用于数据流聚类分析。

D-Stream采用带衰变因子的密度计算法,避免了历史数据对当前数据分布的影响过大,并在该密度定义上严格证明了网格的最大密度和网格渐变时间,使算法能够以有效的方式及时探查到数据的动态变化[2]。针对CluStream算法[3]不能发现任意形状簇的缺陷,朱蔚恒等人提出了ACluStream算法,其使用的空间分割方法与网格方法并无本质区别,算法能够发现任意形状的簇,但在空间划分粒度较小、簇的数目比较多的情况下误差较大[4]。刘青宝等人提出的GCluStream算法在数据流处理过程中对密度相近、位置相邻、形状大小相同的聚类块进行合并,对数据分布极不平衡的大聚类快进行拆分,使算法效率提高[5]。文献[6]中采用移动网格技术对簇边界进行处理,提高了聚类准确性。DENCLUE算法[7]认为数据会对周围空间产生影响,何勇等为解决算法丢失数据空间影响信息的问题,对单世明的方法[8]进行改进,拓展了网格区域[9]。

基于密度网格的数据流聚类算法在算法速度和效率上都有了很大提高,但由于其本质是对传统密度网格方法的改进,

到稿日期:2011-01-27 返修日期:2011-04-28 本文受中央高校基本科研业务费专项资金(SWJTU11ZT08)资助。

米 源 硕士生,主要研究方向为数据挖掘、聚类分析,E-mail:yuan.m12@gmail.com;杨 燕 教授,主要研究方向为数据挖掘、模式识别;李天瑞 教授,主要研究方向为智能信息处理、粗糙集、粒度计算。

因此存在原有方法的固有缺陷，包括以下两个方面：

(1)网格单元的密度阈值参数难以设定。缺乏数据先验知识和专业领域知识的用户很难对此参数进行适当设定。

(2)簇边界难以精确。使用网格对数据压缩存储，使我们失去了数据在网格单元内的分布信息，造成了聚类结果的偏差。

本文针对以上问题进行改进，在 D-Stream 算法的基础上提出了基于密度网格的数据流聚类算法 NDD-Stream(Non-uniform Division D-Stream)。算法通过统计网格单元的密度与簇的数目，动态确定网格单元的密度阈值；对位于簇边界的网格单元采用不均匀划分，以提高簇边界的聚类精度。

3 基本概念

3.1 密度网格

设数据 $x=(x_1,x_2,\cdots,x_d)$ 共有 d 维属性，数据空间 $S=S_1\times S_2\times\cdots\times S_d$。

定义 2(网格单元) 将每维空间 $S_i(1\leqslant i\leqslant d)$ 均匀划分为 p 份，则一个网格单元 g 由各维数据空间中一段划分 S_{d,j_i} $(j_i=1,\cdots,p)$ 的交集组成：

$$g=S_{1,j_1}\cap S_{2,j_2}\cap\cdots\cap S_{d,j_d}$$

简化表示为：

$$g=(j_1,j_1,\cdots,j_d)$$

共有 $N=\prod_{i=1}^{d}p$ 个网格单元，任意一条数据记录 x 都可以采取如下方式映射到网格单元 g：

$$g(x)=(j_1,j_2,\cdots,j_d)\ \text{where}\ x_i\in S_{i,j_i}$$

定义 3(网格单元密度) 为体现数据流随时间变化的特性，引入衰减系数 $\lambda\in(0,1)$，使数据密度值随时间不断衰减。设数据 x 在 t 时刻的密度值为：$D(x,t)$，则其在 $t+1$ 时刻的密度为：

$$D(x,t+1)=\lambda D(x,t) \tag{1}$$

网格单元密度为单元内所有数据点当前时刻密度之和。单元接收新数据时更新密度值：设网格单元 g 在 t_n 时刻吸收了一个新数据，接收上一个数据的时刻为 $t_l(t_n>t_l)$，则 g 的密度更新为：

$$D(g,t_n)=\lambda^{t_n-t_l}D(g,t_l)+1 \tag{2}$$

定义 4(网格单元特征向量) 对于每个网格单元，我们存储一个特征向量作为概要数据结构，用来提取数据概要信息。一个网格单元 g 的特征向量如下表示：

$$(t_g,t_m,D,label,status,Datainf)$$

式中，t_g——g 最近一次接收数据的时刻；

t_m——g 被作为空网格被移除的时刻(若曾经进行过此操作)；

D——g 上一次更新特征向量时的密度；

$label$——g 所属簇的标号；

$Status$——网格单元的状态位，共有两种状态：空状态或正常状态；

$Datainf$——落入网格单元数据的分布位置、到达时刻。

3.2 网格密度阈值的确定

网格单元的密度阈值参数很大程度上影响算法的聚类质量。本文借鉴平均密度的思想。

积累一定数据量 M 的数据，对非空网格单元的密度进行统计。记最大的网格单元密度为 $Den_{\max}$，最小的网格密度为 $Den_{\min}$，则非空网格单元的平均密度 Den_{ave} 为：

$$Den_{ave}=\frac{\sum_{i=1}^{K}Den_i}{K} \tag{3}$$

式中，Den_i 为非空网格单元密度；K 为非空网格单元数目。

密集网格阈值：

$$D_m=\frac{Den_{\max}+Den_{ave}}{2} \tag{4}$$

稀疏网格阈值：

$$D_l=\frac{Den_{\min}+Den_{ave}}{2} \tag{5}$$

定义 5(密集网格单元、过渡网格单元与稀疏网格单元) 在 t 时刻，对于网格单元 g：

密集网格单元：$D(g,t)\geqslant D_m$；

过渡网格单元：$D_l\leqslant D(g,t)<D_m$；

稀疏网格单元：$D(g,t)<D_l$。

在数据流处理过程中，为避免固定的密度阈值不能体现数据的动态变化，通过监控数据空间中的簇数目，对网格单元的密度阈值做出动态调整。

(1)给定数据空间内簇的最大数目 $Clu_{\max}$ 与最小数目 $Clu_{\min}$。

(2)若簇数目 Clu 大于 $Clu_{\max}$，说明密度阈值设置过低，导致多数密集网格单元成为独立的簇。需要提高 D_m 进行调整。若 Clu 小于 $Clu_{\min}$，说明密度阈值设置过高，导致部分簇被作为过渡或稀疏密度网格单元进行处理。需要降低 D_m 进行调整。

(3)密集网格阈值与稀疏网格阈值的比值，即 $r=D_m/D_l$ 不应过小。r 值过小时，密集与稀疏网格单元在数据空间中占多数，过渡网格单元较少，簇很可能包含噪声数据，使簇内密度降低，聚类精度下降。在对 D_m 的调整过程中，将 r 的取值范围限定为 $[r_l,r_m]$，若超出取值范围，则对 D_l 做出调整以保证取值范围。

3.3 格簇

定义 6(相邻网格单元) 两个密集网格单元 g_1、g_2，若它们在某一维的取值范围中相邻，其余任意维的取值范围相等，则称这两个网格单元为相邻网格单元，表示为 $g_1\sim g_2$。

定义 7(网格群) 网格群是一组密集网格单元的集合 $G=(g_1,\cdots,g_n)$，对任意网格单元 $g_i、g_j\in G$，存在路径 $g_{k1},\cdots,g_{kn},g_{k1}=g_i,g_{kn}=g_j$，路径当中任意位置相邻的网格单元为相邻网格单元，即：$g_{k1}\sim g_{k2},g_{k2}\sim g_{k3},\cdots,g_{kn-1}\sim g_{kn}$。

定义 8(格簇) 网格群 G 中的密集网格单元相互连接，若其内部的网格单元都为密集网格单元，边缘的网格单元都为稀疏网格单元或过渡网格单元，则 G 为一个格簇。

4 基于密度网格的数据流聚类算法

4.1 NDD-Stream 算法基本思想

NDD-Stream 算法分为在线处理算法与离线调整算法的双层处理框架。在线算法快速处理到达数据点，探查状态发生变化的网格单元，调整格簇。离线部分对位于簇边缘的网格单元进行再次划分，调整簇边界以提高簇边界的划分精确度。

4.2 NDD-Stream 在线处理算法

4.2.1 网格探测时间间隔

数据流是动态变化的，为发现这种动态变化，需要每隔一

个时间段就对网格单元的状况进行探测。这个时间段 gap 选取为网格单元由密集单元蜕变为稀疏单元或由稀疏单元成长为密集单元所需的最短时间：

$$gap=\left\lfloor \log_{\lambda}\left(\max\left\{\frac{D_l}{D_m},\frac{1-D_m(1-\lambda)}{1-D_l(1-\lambda)}\right\}\right)\right\rfloor \tag{6}$$

4.2.2 空网格的探测与移除

对数据空间进行网格划分会产生大量的网格单元，随着时间增长，大部分单元仅含有少量数据，且这些数据大部分是过期的历史数据。这种长期不接收新数据或含有极少量数据的网格单元我们称其为空网格单元，在数据的在线维护过程中需要不断地将这些空网格单元进行删除以避免存储空间的过度占用与运算速度的下降。

定义 9（空网格密度阈值） 设网格单元 g 上一次接收数据的时刻为 t_g，则在时刻 t（$t>t_g$），空网格密度阈值为：

$$\pi(t_g,t)=D_l(1-\lambda^{t-t_g+1}) \tag{7}$$

将满足条件 C1、C2 的网格单元 g 的状态位标记为空网格状态：

(C1) $D(g,t)<\pi(t_g,t)$；

(C2) $t\geqslant(1+\beta)t_m$（若网格单元 g 曾经被作为空网格单元移除过，$\beta>0$ 为常参数）。

将网格单元状态位标记为空网格后，在下一次探测期间继续关注这些网格单元的状态。若此期间网格单元未接受任何新数据，则移除该单元；否则若单元不满足条件 C1、C2，我们改变网格的状态位，重新将其设置为正常状态。

4.2.3 NDD-Stream 在线算法描述

NDD-Stream 在线处理算法步骤如下：

输入：数据流 DataStream，网格划分参数 p，密度衰减系数 λ，格簇最大最小数目 $Clu_{\max}$、$Clu_{\min}$，密度阈值比取值范围 $[r_{\max},r_{\min}]$

输出：格簇集合

1. 对数据空间 S 进行划分，建立密度网格结构；
2. while 数据流未结束 do
3. 读取数据记录 x；
4. 将数据映射至相应密度网格单元 g；
5. 更新 g 的特征向量；
6. if 读取数据量为 M then
7. 统计网格单元密度信息，确定网格单元密度阈值 D_m、D_l，网格单元探测时间段 gap；
8. 形成初始格簇；
9. end if
10. if 数据流经过时间段 gap then
11. 探测并移除空网格单元；
12. 响应网格的变化对格簇进行调整；
13. 检测当前格簇数目 Clu 的取值，对网格单元密度阈值 D_m 与 gap 进行调整；
14. end if
15. end while

步骤 8 形成初始格簇：从任意密集网格单元开始，对相邻的密集单元分配相同的类标签以形成网格群，对所有密集单元重复此操作直至密集单元的类标签不再发生变化，网格群吸收其周围的过渡网格单元作为簇边缘。

步骤12为响应网格单元的变化对格簇进行调整：若格簇中的密集单元变为稀疏单元，则将该单元从格簇中剔除；若稀疏单元变为过渡或密集单元，则将其分配给与其最近的最大格簇。

4.3 NDD-Stream 离线调整算法

4.3.1 非均匀划分网格

若使用均匀划分的网格结构对数据进行压缩存储，舍弃数据的原有信息，会使我们无法探查到数据在网格单元内部的分布信息，造成聚类结果的偏差[10]。这种误差在位于簇边界的网格单元中尤为常见：

(1)与簇内区域相比，簇的边界区域相对密度较低。如图 1 中网格单元 $g1$，当中包含有少量噪声数据。

(2)由于网格的均匀划分，可能导致处于簇边界网格单元内的数据点过少，而无法达到过渡网格密度阈值，造成簇边界的丢失。如图 1 中网格单元 $g2$。

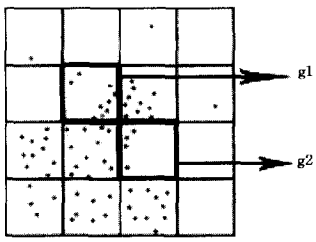


图 1 簇边界网格单元误差

为解决上述问题，本文采用非均匀的网格划分方法，对位于格簇边缘的过渡网格进行再划分。

4.3.2 NDD-Stream 离线算法描述

NND-Stream 算法适量保存位于过渡网格单元中的数据信息：在数据点映射至网格单元时，若该单元为非密集单元，则保存数据分布与到达时刻。在数据流处理过程中，当网格单元褪变为空网格单元被移除时，删除单元特征向量中的 Datainf 项，在单元再次开始接收数据时重新开始记录数据信息；同样，在单元成长为密集单元时，我们无需再保存有关该单元的数据信息。

离线调整算法对簇边界的过渡网格单元进行再划分，使它们成为细粒度网格单元，在此基础上进行格簇的调整，以提高簇边界的精度。

NND-Stream 离线调整算法步骤如下：

输入：过渡网格单元划分参数 p'，格簇集合

输出：最终聚类结果

1. 将格簇中过渡网格单元各维再划分，划分后的网格单元密度阈值 $D_m=D_m/d^k$，$D_l=D_l/d^k$；
2. 使用保存的数据位置与到达时刻将数据映射至细粒度的网格单元；
3. 对格簇进行调整，输出最终聚类结果。

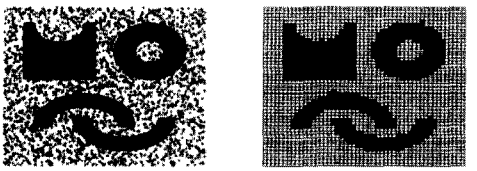
步骤 2、3 中格簇的形成和调整与在线算法中采用的方式相似，不同之处在于所操作网格单元的粒度更为细小。

5 实验结果和分析

实验环境：Intel Core2 T7250 处理器 2.00GHz，1G 内存，Window XP 操作系统，算法采用 C++语言编写。实验所用数据分为仿真数据与真实数据两部分。仿真数据为用 Matlab 生成的 50k 二维数据，其中含有 4 个类，5%的噪声，数据分布如图 2(a)所示；真实数据为网络入侵检测数据集 KDD-CUP99，共有 41 维属性，其中连续属性 34 维，离散属性 7 维，取其 10%的样本数据集进行测试，共有数据记录 494021 条。所有实验数据都经过标准化处理至[0,1]区间。

算法参数设置为：$p=50$，$\lambda=0.95$，$Clu_{max}=50$，$Clu_{min}=5$，密度阈值比 $D_m/D_l\in[4,6]$，数据流速 $v=500$。

首先为检测算法对任意形状数据的处理能力，对仿真数据进行处理。实验中对每条数据按其到达时间赋予相应的时间戳，设置 $\lambda=0.998$，结果如图 2(b)所示。实验结果显示算法能够完好识别出数据中 4 个非凸形状的类。



(a) 原始数据　　(b) 聚类结果

图 2　任意形状聚类

对真实数据 KDDCUP99 不同时刻的聚类结果如图 3 所示，聚类结果与 D-Stream 算法比较，采用平均聚类纯度评价聚类质量，聚类纯度即类中主要类标号数据在类中所占的百分比，聚类纯度越高，表示数据属于同一类的比例越大，聚类质量越好。从结果中看出，在不同时刻的聚类结果中 NDD-Stream 取得了较高的类纯度，这是由于 NDD-Stream 在输出聚类结果时采用不均匀的网格划分提高聚类精度，因此具有较好的聚类质量。

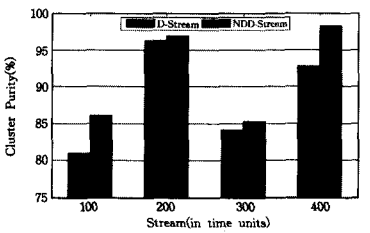


图 3　聚类纯度比较

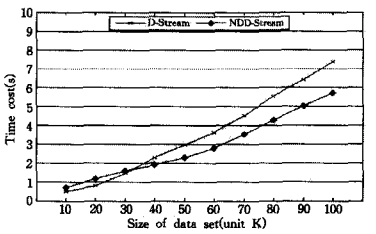


图 4　聚类时间比较

图 4 为 NDD-Stream 在线处理算法与 D-Stream 算法的处理速度比较。结果显示随着数据量的增长，NDD-Stream 所用时间的增长速率低于 D-Stream，因为 NDD-Stream 在运行过程中动态调整 gap 值，使其适应当前数据空间的密度分布情况，避免了过于频繁的格簇调整，提高了算法效率。

结束语　NDD-Stream 算法实现了密度阈值参数的自适应设置，并通过对簇边缘采用不均匀的网格划分提高了簇的精度。虽然在线处理过程中需要消耗部分时间来保存数据，但由于算法动态调整了密度阈值，在一定程度上减少了不必要的簇调整工作，节约了运算时间。实验证明，算法能够取得较高的聚类质量以及较快的运行速度。

参 考 文 献

[1] Guha S, Mishra N, motwani R, et al. Clustering Data Streams [C]// Proc. of the 41st Annual Symposium on Foundations of Computer Science. 2000:359-366

[2] Chen Y, Tu L. Density-Based Clustering for Real-Time Stream Data[C]// Proc, of the International Conference on Knowledge Discovery and DataMining. August 2007:12-15

[3] Aggarwal C, Han J, Wang J, et al. A Framework for Clustering Evolving Data Streams[C]// Proc. of the 29th VLDB Conference. 2003:81-92

[4] 朱蔚恒，印鉴，谢益煌. 基于数据流的任意形状聚类算法[J]. 软件学报，2006，17(3)：379-386

[5] 刘青宝，戴超凡，邓苏，等. 基于网格的数据流聚类算法[J]. 计算机科学，2007，34(3)：159-161

[6] 郑盈盈，倪志伟，吴姗，等. 基于移动网格和密度的数据流聚类算法[J]. 计算机工程与应用，2009，45(8)：129-131

[7] Hinneburg A, Keim D A. An Efficient Approach to Clustering in Large Multimedia Databases with Noise[C]// Proc of the International Conference on Knowledge Discovery and DataMining. 1998:58-65

[8] 单世明. 基于网格和密度的数据流聚类方法研究[D]. 大连：大连理工大学，2006

[9] 何勇，刘青宝. 基于动态网格的数据流聚类分析[J]. 计算机应用研究，2008，25(11)：3281-3284

[10] Sun Y, Lu Y. A Scalable Grid-based Clustering Algorithm for Very Large Spatial Databases[C]// Proc. of the International Conference on Computational Intelligence and Security. 2006: 763-768

(上接第 146 页)

[6] Jia Chun-xin, Lu Chao-jun. Extending BPEL to Model Business Processes Involving Human Activities[C]// 2009 International Conference on Frontier of Computer Science and Technology. 2009:524-529

[7] OASIS. Web Services Business Process Execution Language. Version 2.0[S]. April 2007

[8] Paci E, Ferrini R, Bertino E. Identity Attribute-based Role Provisioning for Human WS-BPEL processes[C]// 2009 IEEE International Conference on Web Services(ICWS2009). 2009:535-542

[9] Juric M. Buisiness Process Execution Language for Web Services [M]. Second Edition. Birmingham: Packt Publishing Ltd, 2006: 145-150

[10] Wang Xin, Zhang Yan-chun, Shi Hao. Access Control for Human Tasks in Service Oriented Architecture[C]// 2008 IEEE International Conference on e-Business Engineering (ICEBE'08). 2008:455-460

[11] 倪晚成，刘连臣，吴澄. Web 服务组合方法综述[J]. 计算机工程，2008，34(4)：79-81