

文章编号:1001-9081(2010)04-1093-03

滑动窗口内基于密度网格的数据流聚类算法

李子文,邢长征

(辽宁工程技术大学 电子与信息工程学院,辽宁 葫芦岛 125105)

(liziwen2009@126.com)

摘要:提出了一种基于密度网格的数据流聚类算法。通过引入“隶度”,对传统的基于网格密度的数据流聚类算法,以网格内数据点的个数作为网格密度的思想加以改进,解决了一个网格内属于两个类的数据点以及边界点的处理问题。从而既利用了基于网格算法的高效率,还较大幅度地提高了聚类精度。

关键词:聚类;数据流;网格;滑动窗口;隶度

中图分类号: TP311 **文献标志码:** A

Density grid-based data stream clustering algorithm over sliding window

LI Zi-wen, XING Chang-zheng

(School of Electronics and Information Engineering, Liaoning Technical University, Huludao Liaoning 125105, China)

Abstract: This paper introduced a density grid-based data stream clustering algorithm. Through the introduction of the "subject degree", the traditional density grid-based clustering algorithm for data stream was improved by taking the data points within the grid as the grid density, thereby resolving the problem of data points belonging to two classes in one grid as well as the treatment of boundary points. Therefore, not only the high efficiency of the grid-based algorithm was utilized, but also the clustering accuracy was raised significantly.

Key words: clustering; data stream; grid; sliding window; subject degree

0 引言

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,常见的应用有网络监控日志、电话通信记录、银行交易信息等。从数据流中获取知识的数据挖掘研究得到了广泛的关注,数据流聚类作为知识发现的重要手段也得到了深入的研究^[1]。

文献[2]提出了单层数据流聚类算法 STREAM^[2],并利用 SSQ 证明了在多种情况下,STREAM 算法的聚类效果比 BRICH 算法要好。但此类算法的缺点是只能提供对当前数据流的一种描述,而不能反映数据流的变化情况。文献[3]中提出了解决数据流聚类的一个双层算法框架及其聚类算法 CluStream,这种双层框架可以较好地解决数据流聚类问题中实时要求与聚类质量间的矛盾。CluStream 算法离线层使用的是改进的 k-means 算法,对于非球状的簇不能很好地描述。文献[4]中提出了一种基于密度和网格的数据流聚类算法 D-Stream,解决了对任意形状的数据流聚类问题,但是 D-Stream 算法是基于密度网格的,由其局限性难免会产生以下问题(如图1所示)。

边界点问题 对于 a1、c1 以及 b3 中的点,如果处理不当,很可能被当作噪声点而影响聚类精度。

类属问题 对于 b2 中属于类1的点,因为其所在网格包含有大量类2的点,很可能被划归为类2从而影响聚类精度。

为解决上述问题,本文提出了 SD-Stream 算法,引入“隶

度”概念,通过计算数据点的“隶度”确定其应归属的网格。有效地解决了基于绝对密度聚类算法所存在的边界点和类属问题。

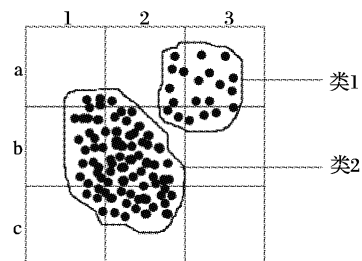


图1 网格密度

1 SD-Stream 算法基本概念

1.1 隶度

“隶度”就是数据点对周围空间网格的隶属程度。本文通过引入“隶度”的概念,改进了传统基于网格密度算法以网格内数据点的数目作为网格密度的缺点,减少了数据点对周围空间网格信息的丢失,提高了聚类精度。通过图2可以直观地说明“隶度”的具体优越性。

为方便起见,以网格单元大小作为计算数据点隶属范围的尺度,即以数据点为中心,网格单元大小尺度的扩展网格。图2可见,由点 p 生成的扩展网格在 a2 中覆盖的数据点个数 0,在 a3 中覆盖的数据点个数为 7,在 b2 中覆盖的数据点个数 1,在 b3 中覆盖的点的个数为 3(包括其自身),规定数据点隶属于扩展网格与其他网格单元所覆盖数据点最多的网

收稿日期:2009-11-06;修回日期:2009-12-14。

作者简介:李子文(1984-),男,山西朔州人,硕士研究生,主要研究方向:数据挖掘、数据流聚类; 邢长征(1967-),男,辽宁阜新,教授,博士,主要研究方向:数据挖掘、聚类。

格单元,即 a3,由图2可见,点 p 也应该划归在 a3 所在的类1中。

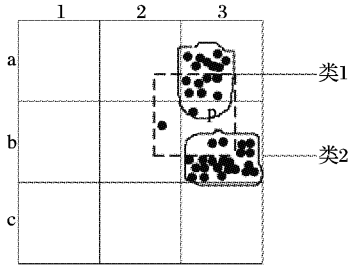


图2 “隶度”计算

1.2 基本概念

数据流是一系列数据记录 X_1, \dots, X_k, \dots , 它们到达的时间分别为 T_1, \dots, T_k, \dots , 数据流中的每一个记录 X_i 是 d 维, $X_i = (x_i^1, \dots, x_i^d)$. d 维数据空间 $S = S_1 \times S_2 \times \dots \times S_d$, 其中 S_i 为第 i 维空间。将 d 维数据空间 S 划分为(密度)网格, 其中每一维 S_i 被分成 p_i 个部分, $S_i = S_{i,1} \cup S_{i,2} \cup \dots \cup S_{i,p_i}$, 数据空间 S 被划分为 $N = \prod_{i=1}^d p_i$ 个(密度)网格。对每一个网格 $g = S_{1,j_1} \times S_{2,j_2} \times \dots \times S_{d,j_d}$, 其中 $j_i = 1, \dots, p_i$, 每一个数据 $x = (x_1, x_2, \dots, x_d)$ 映射到网格 $g(x)$, $g(x) = (j_1, j_2, \dots, j_d)$, 其中 $x_i \in S_{i,j_i}$ 。对每一个数据 x , 赋予它一个随着时间递减的密度系数, 即当 x 在时间 t_c 到达, 则定义它的时间戳为 $T(x) = t_c$, 密度系数 $D(x, t) = \lambda^{t-T(x)} = \lambda^{t-t_c}$, 这里 $\lambda \in (0, 1)$, 是一个常数, 称为蜕化系数。

定义1 网格密度。对一个网格 g 在确定的时间 t 所包含的数据元素有 x_1, x_2, \dots, x_n , 它们到达网格 g 的时间分别为 t_1, t_2, \dots, t_n , 则网格 g 在时刻 t 的密度 $D(g, t) = \sum_{i=1}^n D(x_i, t_i) = \sum_{i=1}^n \lambda^{t-t_i}$, 网格密度是随时间变化的, 但是没有必要随时间更新其密度, 只在当有新的数据到达该网格的时候更新其密度。

引理1 假设网格 g 在时间 t_n 接收了新数据, 且其最近接收数据的时间为 t_l ($t_n > t_l$), 这样网格密度可以按如下公式进行更新:

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

证明 设网格 g 在时间 t_l 的数据集合为 $X = (x_1, x_2, \dots, x_m)$, 则有:

$$D(g, t_l) = \sum_{i=1}^m D(x_i, t_l)$$

从而:

$$D(x_i, t_n) = \lambda^{t_n-T(x_i)} = \lambda^{t_n-t_l} \lambda^{t_l-T(x_i)} = \lambda^{t_n-t_l} D(x_i, t_l);$$

$$i = 1, \dots, m$$

所以:

$$D(g, t_n) = \sum_{i=1}^m D(x_i, t_n) + 1 = \sum_{i=1}^m \lambda^{t_n-t_l} D(x_i, t_l) + 1 = \lambda^{t_n-t_l} \sum_{i=1}^m D(x_i, t_l) + 1 = \lambda^{t_n-t_l} D(g, t_l) + 1$$

证毕。

引理2 设 $X(t)$ 为从开始时刻到时刻 t 的数据流数据集, 则有:

$$1) \sum_{x \in X(t)} D(x, t) \leq \frac{1}{1-\lambda};$$

$$2) \lim_{t \rightarrow \infty} \sum_{x \in X(t)} D(x, t) = \frac{1}{1-\lambda}.$$

证明 设 $X(t)$ 中从开始时刻到时刻 t 的数据流数据元素为: x_1, x_2, \dots, x_i , 其中 x_i 在 i 时刻到达, 则数据元素的密度分别为: $\lambda_1, \dots, \lambda_i, 1$ 。它们的和:

$$\sum_{x \in X(t)} D(x, t) = \sum_{i=0}^t \lambda^{t-i} = \frac{1-\lambda^{t+1}}{1-\lambda} \leq \frac{1}{1-\lambda}$$

所以:

$$\lim_{t \rightarrow \infty} \sum_{x \in X(t)} D(x, t) = \lim_{t \rightarrow \infty} \frac{1-\lambda^{t+1}}{1-\lambda} = \frac{1}{1-\lambda}$$

证毕。

定义2 网格特征向量。一个网格 g 的特征向量是一个5元组 $T = (t_g, t_m, D, label, status)$ 。其中: t_g 是 g 最近更新的时间; t_m 是网格 g 作为孤立网格被从 g_list 中清除的最近时间; D 是网格 g 最近更新的密度, $label$ 是网格的类标签, $status = (孤立, 正常)$ 作为清除孤立网格的标签。

定义3 稠密网格、稀疏网格、过渡网格。设 $X(t)$ 为从时间 $0 \sim t$ 的所有数据集, 由引理2可知系统中所有数据元素的密度之和不会超过 $\frac{1}{1-\lambda}$, 设系统中有 $N = \prod_{i=1}^d p_i$ 个网格, 则网格的平均密度不会超过 $\frac{1}{N(1-\lambda)}$, 则有如下定义。

$$\text{稠密网格 } D(g, t) \geq \frac{C_m}{N(1-\lambda)} = D_m; C_m > 1;$$

$$\text{稀疏网格 } D(g, t) \leq \frac{C_l}{N(1-\lambda)} = D_l; 0 < C_l < 1;$$

$$\text{过渡网格 } \frac{C_l}{N(1-\lambda)} \leq D_l \leq \frac{C_m}{N(1-\lambda)}.$$

定义4 相邻网格。对 $g_1 = (j_1^1, j_1^2, \dots, j_1^d)$ 和 $g_2 = (j_2^1, j_2^2, \dots, j_2^d)$, 如果存在常数 $k, 1 \leq k \leq d$, 使得:

$$1) j_i^1 = j_i^2, i = 1, \dots, k-1, k+1, \dots, d;$$

$$2) |j_k^1 - j_k^2| = 1;$$

则称 g_1 和 g_2 在第 k 维上是相邻的。标记为 $g_1 \sim g_2$ 。

定义5 网格簇。设有网格集合 $G = (g_1, \dots, g_m)$, 如果对任意两个网格 $g_i, g_j \in G$, 存在一个网格序列 g_{k_1}, \dots, g_{k_l} , 有 $g_{k_1} = g_i, g_{k_l} = g_j$, 并且 $g_{k_1} \sim g_{k_2}, g_{k_2} \sim g_{k_3}, \dots, g_{k_{l-1}} \sim g_{k_l}$, 则称 G 为一个网格簇。

定义6 内部网格与边界网格。在网格簇 G 中, 如果一个网格在每一维都有相邻的网格, 则称其为内部网格, 否则称其为边界网格。

定义7 网格聚类。设 $G = (g_1, g_2, \dots, g_m)$ 是一个网格簇, 如果一个网格簇的每个内部网格都是稠密网格并且每个边界网格不是稠密网格就是过渡网格, 则定义 G 为一个网格聚类。

2 SD-Stream 算法设计

SD-Stream 算法借鉴经典算法 CluStream 分为在线层和离线层两个框架。在线层快速接收输入的数据流, 其产生的结果作为中间结果被维护起来, 并且随着新数据点的流入该中间结果实时动态更新。依据一定的时间框架周期性地选取某些特定时刻的中间结果保存到外存作为离线层算法的输入。

离线层由用户调用,针对用户的挖掘请求,利用金字塔时间框架给出其感兴趣的在不同时间粒度上的结果^[5]。

2.1 在线层算法

开始;

$t_c = 0$;

初始化一个空的哈希表 g_list ;

如果数据流不为空,则:

读入新数据 $x = (x_1, x_2, \dots, x_d)$;

新数据处理算法;

检测包含数据元 x 的网格单元 g ;

如果网格单元 g 不在 g_list 中,则将其插入 g_list ;

如果 t_c 等于滑动窗口大小,则:

执行微聚类,统计概要数据结构;

如果 t_c 等于滑动窗口的倍数,则:

从 g_list 检测并剔除空的网格单元;

调整微聚类;

$t_c = t_c + 1$;

结束。

2.2 新数据元处理算法

1) 读取数据流中的数据元;

2) 以数据元为中心建立扩展网格,分别查找扩展网格与其他网格单元重叠形成的四个子网格所覆盖的数据元的数目;

3) 如果四个子网格所覆盖的数据元数目相同,则将数据元归入其所在的网格单元中;

4) 如果四个子网格所覆盖的数据元数目不相同,则将数据元归入覆盖数据元数目最大的网格单元中;

5) 更新该网格单元的特征向量;

6) 读取下一个数据元。

2.3 离线层算法

通常查询某个时刻的聚类情况意义不大,而往往都是查询分析某时间段的聚类情况。所以有:

输入:时间 t_1, t_2 , 阈值 ε 。

输出: t_1 到 t_2 的聚类结果。

1) 检查 t_1 和 t_2 的合法性(是否超出时间范围);

2) 根据 t_1 时刻的概要信息将该时刻的微簇作为核心簇,分别赋予类标号;

3) 对 t_1 到 t_2 范围内的其余微簇进行检查;

4) 如果在阈值范围内有相似类,将其并入该类,更新该类;

5) 如果没有相似类,则将其作为核心簇,赋予新的类标号;

6) 如果在阈值范围之内,有两个或多个相似的类,则合并相似类及微簇,更新类标号;

7) 直到没有新的微簇可以处理;

8) 输出聚类结果。

3 实验验证

实验环境: Intel Pentium 4 处理器, 1 GB 内存。操作系统: Windows XP。算法是用 Microsoft Visual C++ 实现的。实验数据采用仿真数据集和真实数据集。

3.1 算法效率验证

使用仿真数据集对 CluStream 算法和 SD-Stream 算法效率进行比较,实验结果如图3所示,可以发现在数据集大小相同时 SD-Stream 算法使用的时间远少于 CluStream 算法,即

SD-Stream 算法的效率要高于 CluStream 算法。

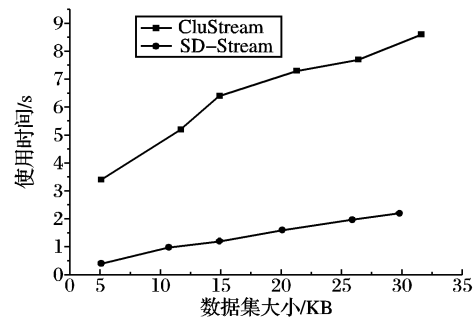


图3 CluStream 算法和 SD-Stream 算法效率对比

3.2 算法精度验证

使用真实数据集 KDD-CUP-99 对 D-Stream 算法和 SD-Stream 算法的聚类精度进行了比较。实验结果见图4和图5。由图4可以看出,SD-Stream 的平均聚类精度达到了 98% 以上。图5可以发现 SD-Stream 比 D-Stream 的聚类精度要高,这是由于 SD-Stream 算法通过计算数据点的“隶度”,将其更合理地分配到了隶属程度更高的网格单元中,降低了因为分配不当而引起的误差。

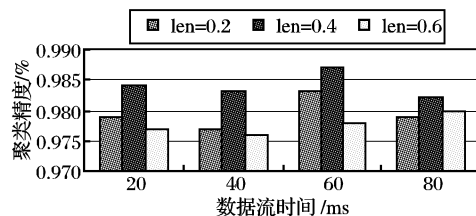


图4 SD-Stream 算法聚类精度

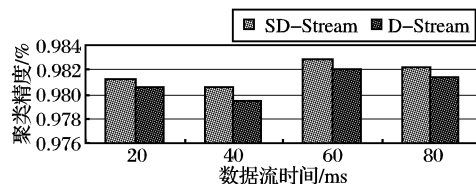


图5 SD-Stream 算法和 D-Stream 算法聚类精度对比

4 结语

本文研究了基于网格密度的数据流聚类问题,提出了 SD-Stream 算法。算法通过引入“隶度”的概念,克服了传统基于网格密度算法精度不高的问题。实验证明该算法能够快速准确地识别出聚类,具有一定的可行性。

参考文献:

- [1] 李敏. 基于网格和密度的数据流聚类算法研究 [D]. 武汉: 武汉理工大学, 2009.
- [2] O' CALLAGHAN L, MISHRA N, MEYERSON A, et al. Streaming-data algorithms for high quality clustering [C] // Proceedings of IEEE International Conference on Data Engineering. Washington, DC: IEEE Computer Society, 2002: 685.
- [3] AGARWAL C, HAN J, WANG J, et al. A framework for clustering evolving data streams [C]. VLDB 2003: Proceedings of the 29th International Conference on Very Large Data Bases. Berlin: VLDB Endowment, 2003, 29: 81-92.
- [4] CHEN Y, TU L. Density-based clustering for real-time stream data [C] // KDD' 07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2007: 133-142.
- [5] 单世民. 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.