ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.8, August 2006, pp.1796−1803
DOI: 10.1360/jos171796

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

基于动态网格的数据流离群点快速检测算法*

杨宜东[1+], 孙志挥[1], 朱玉全[2], 杨 明[3], 张柏礼[1]

[1](东南大学 计算机科学与工程系,江苏 南京 210096)

[2](江苏大学 计算机科学与通信工程学院,江苏 镇江 212013)

[3](南京师范大学 计算机科学系,江苏 南京 210000)

A Fast Outlier Detection Algorithm for Data Streams Based on Dynamic Grids

YANG Yi-Dong[1+], SUN Zhi-Hui[1], ZHU Yu-Quan[2], YANG Ming[3], ZHANG Bo-Li[1]

[1](Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

[2](School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013, China)

[3](Department of Computer Science, Nanjing Normal University, Nanjing 210000, China)

+ Corresponding author: Phn: +86-25-83795451, E-mail: ydyang@seu.edu.cn, http://www.seu.edu.cn

Yang YD, Sun ZH, Zhu YQ, Yang M, Zhang BL. A fast outlier detection algorithm for data streams based on dynamic grids. *Journal of Software*, 2006,17(8):1796−1803. http://www.jos.org.cn/1000-9825/17/1796.htm

Abstract: As an important task of data mining, outlier detection has been applied to many fields. Recently, research on mining in data stream is receiving more and more attention. For solving outlier detection in data stream, a new fast outlier detection algorithm is presented. Based on dynamically grid partitioning data space, the method separates dense areas from sparse areas. Data in dense areas are filtered simply, which reduces greatly the size of objects the algorithm should consider. Outliernesses of candidates in sparse areas are approximated efficiently. Data with high outlierness are outputted as outliers. Results of experiments on synthetic and real data sets show promising availabilities of the approaches.

Key words: data stream; outlier detection; time-sensitive dynamic grids partitioning

摘 要: 离群点检测问题作为数据挖掘的一个重要任务,在众多领域中得到了应用.近年来,基于数据流数据的挖掘算法研究受到越来越多的重视.为了解决数据流数据中的离群点检测问题,提出了一种基于数据空间动态网格划分的快速数据流离群点检测算法.算法利用动态网格对空间中的稠密和稀疏区域进行划分,过滤处于稠密区域的大量主体数据,有效地减少了算法所需考察的数据对象的规模.而对于稀疏区域中的候选离群点,采用近似方法计算其离群度,具有高离群度的数据作为离群点输出.在保证一定精确度的条件下,算法的运行效率可以得到大幅度提高.对模拟数据集和真实数据集的实验检测均验证了该算法具有良好的适用性和有效性.

关键词: 数据流;离群点检测;时间相关动态网格划分

中图法分类号: TP393 文献标识码: A

近年来,基于数据流模型的管理系统、数据挖掘算法等已成为重要的研究课题[1−3].数据流数据具有数据量

* Supported by the National Natural Science Foundation of China under Grant No.60572112 (国家自然科学基金)
Received 2004-09-30; Accepted 2005-10-11

大、潜在无限、被处理顺序不能控制、到达速率不确定等特点.由于不能将全部数据加载到内存中运算,而且存储及访问代价大,因此,迫切需要提出高效、可行的基于数据流模型的算法,使得在给定的有限运行空间上,能够通过对数据流进行一次或较少次数的线性扫描,对其进行管理以及进一步的知识发现研究.

离群点检测由于其在知识发现中所发挥的重要作用而得到重视和深入的研究.假设检验是最早用来发现异常样本的基于统计学原理的方法,由于它必须假设数据集符合特定的分布模型,因此具有很大的局限性.在很多聚类算法研究中,离群点检测也经常作为一种副产品出现[4],离群点被定义为不属于任何聚类的数据点.由于这些算法着重于从聚类的角度进行考虑,因此通常不能满足实际应用的效率要求.

近年来,基于数据挖掘概念的离群点检测研究取得了重要的成果,提出了一些有效的检测算法并得到应用.例如:基于距离的算法 FindAllOutsD[5]、面向高维数据离群点检测算法 FindOut[6]、带离群度的离群点检测算法 LOF(local outlier factor)[7]和 LOCI(local correlation integral)算法[8]等.由于 FindAllOutsD,FindOut,LOF 等算法对于数据集中属于常规聚类模式的数据缺乏必要的预过滤处理,导致较大的空间复杂度;其次,上述算法一般具有 $O(N\log N)$以上的时间复杂度,在处理大数据集时无法获得令人满意的响应速度,特别是不能够处理只能进行一遍扫描的数据流数据.最新提出的 LOCI 算法,在计算每个数据的离群度时采用近似计算,但仍然是将整个数据集作为考察对象,并需要进行两遍扫描.

目前,基于数据流的算法研究已成为重要的研究课题[1−3].文献[3]指出数据流主要包括:Time Series,Cash Register,Turnstile 这 3 种模型,其一般性逐渐增大.在数据流算法研究中,离群点检测也引起了研究者的关注.文献[9]提出了第一种针对大规模数据流的异常检测算法,该算法引用文献[10]中提出的异常(deviant)作为离群点的概念,仅仅解决 Time Series 这一特殊数据流模型上的离群点检测问题,并且没有考虑概念转移情况.

本文的工作主要包括以下几个方面:针对 Cash Register 模型数据流,提出数据流中离群点的快速检测算法 FODDS(fast outlier detection in data streams)及其快速、简单的版本 FODDS-S;算法还可以扩展到 Turnstile 模型.FODDS 采用一种快速、直接、时间相关的网格动态划分方法,将空间中密度稀疏和稠密区域分开,得到候选离群点集合,而对其他数据进行过滤.FODDS-S 作为对 FODDS 的简化,在可能的精确度损失下进一步提高了算法速度.实验表明,FODDS 和 FODDS-S 通过对计算耗费与精确度之间的折衷,可显著提高算法运行效率.

1 问题描述及相关定义

一般地,设 $A=(A_1,A_2,\ldots,A_k)$为欧式空间下的属性集合,相应的 k 维数据空间 $S=(A_1\times A_2\times\ldots\times A_k)$.

定义 **1**(数据流). 数据流 DS 可以看成一个关系 R,其中的元组 $\vec{r}_i\in S,i=1,2,...$ 连续到达. $t(\vec{r}_i)$ 为 $\vec{r}_i$ 到达时间.

本文对所研究的数据流中离群点检测问题描述为:给定数据流 DS,在一定的内存和时间的限制下,动态维护相对于当前窗口(宽度为 w)数据流分布的候选离群点集合 $CandOutlier$.当用户提出查询 Top$Q$($0<Q<100\%$)的离群点时,算法对 $CandOutlier$ 进行筛选,得到离群点集合 $Outlier$ 并提供给用户.候选离群点集合的大小为预先设置的参数,设定$|CandOutlier|=P\times w$($0<P<100\%,P\geq Q$).

Hawkins 给出了一个基本的、直观上的离群点定义[11].

定义 **2**(**Hawkins**). 如果一个数据样本与其他样本之间存在足以引起怀疑的差异,则称其为离群点.

文献[8]中提出多粒度偏差因子(multi-granularity deviation factor,简称 MDEF)的概念,并在其基础上给出一种新的离群点定义.设 $d(p_i,p_j)$为数据 p_i,p_j 之间的欧式距离,$N(p_i,r)=\{p\in S|d(p,p_i)\leq r\}$为 p_i 的 r-近邻,$n(p_i,r)=|N(p_i,r)|$为 p_i 的 r-近邻的基数,即与 p_i 距离不大于 r 的所有数据的个数,$\hat{n}(p_i,r,\alpha)=\dfrac{\sum_{p\in N(p_i,r)}n(p,\alpha r)}{n(p_i,r)}$为 p_i 的 r-近邻中所有数据 p 的 $n(p,ar)$的平均值,$\sigma_{\hat{n}}(p_i,r,\alpha)=\sqrt{\dfrac{\sum_{p\in N(p_i,r)}(n(p,\alpha r)-\hat{n}(p_i,r,\alpha))^2}{n(p_i,r)}}$,为 p_i 的 r-近邻中所有 p 的 $n(p,ar)$的标准偏差,并设 $\sigma_{MDEF}(p_i,r,\alpha)=\dfrac{\sigma_{\hat{n}}(p_i,r,\alpha)}{\hat{n}(p_i,r,\alpha)}$.

定义 **3**. 对于任意 p_i,设其在 r-近邻内的多粒度偏差因子为

$$MDEF(p_i,r,\alpha)=\frac{\hat{n}(p_i,r,\alpha)-n(p_i,\alpha r)}{\hat{n}(p_i,r,\alpha)}.$$

由 Chebyshev 不等式,对于任意点与点之间距离的分布,有 $P\{MDEF(p_i,r,\alpha)>k_\sigma\sigma_{MDEF}(p_i,r,\alpha)\}\le\frac{1}{k_\sigma^2}$.

定义 **4**. 对于 p_i,如果存在某个 r 和α,使得 $MDEF(p_i,r,\alpha)>k_\sigma\sigma_{MDEF}(p_i,r,\alpha)$,则 p_i 为离群点.

文献[8]利用 $MDEF(p_i,r,\alpha)$和$\sigma_{MDEF}(p_i,r,\alpha)$对每个点周围的数据分布密度进行估计,以此判断其离群度,具有简单、直观、对参数依赖小等特点.但是,LOCI 算法及其近似算法 aLOCI 需要至少两次扫描数据:第 1 次获得每个点的 $n(p,ar)$;第 2 次计算每个点在不同 r 下的$\sigma_{MDEF}(p_i,r,\alpha)$,并判断其是否为离群点.并且,算法仍然是将整个数据集作为检测对象.本文仅采用文献[8]的定义作为局部数据点的离群度分析依据.

2 FODDS 算法及其构造

2.1 算法思路

在实际应用中,离群点定义具有一定的模糊性,并且现实数据存在主体聚类性的特点[11].主体聚类性是指数据集的绝大部分数据来自于正常数据源.相对于离群点而言,这些数据具有明显的聚类特征,即大量数据密集分布于较小的空间范围内.因此,在进行离群点检测时,对于分布于稠密区域的数据的过滤处理并不影响检测效果.因此,FODDS 算法采用两种近似计算,通过对计算耗费与精确度之间的折衷来提高检测效率.

首先,利用动态网格划分方法.FODDS 算法维护数据流数据分布的统计信息,根据这些信息可以筛选出候选离群点数据集 *CandOutlier*.相对于整个数据流,*CandOutlier* 的规模显著减小.

其次,在更一般的情况下,为了进一步提高算法的精度,FODDS 算法根据可用资源调整 *CandOutlier* 的大小,并采用近似算法对 *CandOutlier* 进行筛选,选择其中具有较高离群度的数据作为 *Outlier* 输出.

2.2 FODDS算法

2.2.1 对数据流数据的动态网格划分

针对传统的静态数据集,已提出一些基于网格划分的方法,应用于例如聚类发现等方面.但是,这些方法均采用预先设定的参数对数据空间进行统一的网格划分,并且由于所研究的数据集是静态的,网格划分在算法的运行过程中也是静态的.FODDS 算法针对所研究的数据流数据的特点,采用动态划分的方法,根据不断到来的数据进行网格动态分裂、合并.算法保存网格中与分布相关的统计信息,同时为得到候选离群点集合,对网格中可能成为离群点的数据进行存储.

为解决数据流数据的概念转移问题,可以对历史数据赋予较小权重,使其影响逐步减小,这样网格能够更好地反映当前数据的分布情况.定义数据权重的衰减系数$\theta(0\le\theta\le1)$,设数据在时刻 t 的权重为 $\theta^{t-t_{occur}}(t\ge t_{occur})$, t_{occur} 为数据出现的时间,初始时数据权重为 1.通过衰减系数的设定,数据权重随着时间的推移逐渐减少.为设定衰减系数θ,设数据的生命周期为 t_{life},权重阈值为 $w_{\min}$,并满足 $\theta^{t_{life}}\le w_{\min}$, 即数据在经过 t_{life} 之后,其权重将不大于 $w_{\min}$.通过不同的 t_{life} 和 $w_{\min}$ 可以设置相应的θ.

定义 **5**(网格). 一个网格由一个六元组 $Cell(C,\vec{S}^1,\vec{S}^2,n_C,O_C,t_{latest})$ 表示.其中,C 为 k 维数据空间中的一个超方体,n_C 为落入 C 中的数据点数,t_{latest} 为该网格最近一次更新的时间, $\vec{S}^1=\left\langle s_1^1,...,s_k^1\right\rangle$, $s_i^1=\sum_{\vec{r}_i\in C}\theta^{t_{latest}-t_{occur_i}}r_i(1\le i\le k)$, $\vec{S}^2=\left\langle s_1^2,...,s_k^2\right\rangle$, $s_i^2=\sum_{\vec{r}_i\in C}\theta^{t_{latest}-t_{occuri}}r_i^2(1\le i\le k)$,O_C 为该网格中候选离群点集合.

由网格定义容易得到以下性质:

性质 **1**. 在时刻 $t(t\ge t_{latest})$时,网格统计信息 $\vec{S}^1$ 和 $\vec{S}^2$ 满足 $\vec{S}_t^1=\theta^{t-t_{latest}}\times\vec{S}_{t_{latest}}^1$, $\vec{S}_t^2=\theta^{t-t_{latest}}\times\vec{S}_{t_{latest}}^2$.

设当前时间为 t_{cur},由性质 1,算法可以对数据 $\vec{r}$ 所在网格 C 的统计信息进行增量更新,

$$n_C=\theta^{t_{cur}-t_{latest}}\times n_C+1,\ s_i^1=\theta^{t_{cur}-t_{latest}}\times s_i^1+r_i,\ s_i^2=\theta^{t_{cur}-t_{latest}}\times s_i^2+r_i^2,\ t_{latest}=t_{cur}.$$

FODDS 算法在初始化时,一次性读入若干数据,并对其进行分割,生成初始网格[12].根据 $\vec{S}^1$ 和 $\vec{S}^2$,容易计算每个网格中数据的平均以及标准偏差: $\mu_i=\frac{S_i^1}{n_C}$, $\sigma_i=\sqrt{\frac{S_i^2-2\mu_i S_i^1}{n_C}+\mu_i^2}$.

当网格密度达到一定阈值时,对其进行分割操作.在本文中,网格分裂合并操作的基本原则是能够使不同的数据聚集被划分到不同的网格中.因此,算法保存网格在每个维度上的均值和方差,并且选择方差最大的维度,在其均值处进行划分,这个操作可以看作是将两部分数据分别归到新生成的两个子网格中.随着数据不断到来,对网格的划分也在不断细化,实验验证在若干次划分后具有较好的效果.

对于网格统计信息的维护,借鉴文献[12]的计算方法.但是由于采用了与时间相关网格划分方法,在计算时需要进行进一步修改.同时,算法需要选择具有较高离群度的网格,并且保存网格中的离群点信息,因此对选择方法和存储结构需作进一步考虑.

设分割的网格为 $Cell(C,\vec{S}^1,\vec{S}^2,n_C,O_C,t_{latest})$, $\sigma_j=\max(\sigma_1,\sigma_2,\ldots,\sigma_k)$,算法在该维度上对网格进行分割.对于其他维度,对应的 s_i^1 和 s_i^2 保持不变.对于维度 j,在正态分布的假设下,将 $Cell$ 从 μ_j 处分割为 $Cell_1$ 和 $Cell_2$,并且

$$n_{C_1}=n_{C_2}=n_C/2,\quad t_{latest_1}=t_{latest_2}=t_{latest},\quad \mu_j^{C_1}=\int_{\min_j}^{\mu_j}x\frac{1}{\sqrt{2\pi}\sigma_j}\mathrm{e}^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}\mathrm{d}x,\quad \mu_j^{C_2}=\int_{\mu_j}^{\max_j}x\frac{1}{\sqrt{2\pi}\sigma_j}\mathrm{e}^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}\mathrm{d}x,$$

$$\mu_j^{C_1}=\sqrt{\int_{\min_j}^{\mu_j}x^2\frac{1}{\sqrt{2\pi}\sigma_j}\mathrm{e}^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}\mathrm{d}x-(\mu_j^{C_1})^2},\quad \sigma_j^{C_2}=\sqrt{\int_{\mu_j}^{\max_j}x^2\frac{1}{\sqrt{2\pi}\sigma_j}\mathrm{e}^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}\mathrm{d}x-(\mu_j^{C_2})^2}.$$

其中,$\min_j$ 和 $\max_j$ 分别是网格中第 j 维度上的最大、最小值.

当对网格进行合并操作时,为便于讨论,设要合并的两个网格分别为 $Cell_1(C_1,\vec{S}_{C_1}^1,\vec{S}_{C_1}^2,n_{C_1},O_{C_1},t_{latest_1})$ 和 $Cell_2(C_2,\vec{S}_{C_2}^1,\vec{S}_{C_2}^2,n_{C_2},O_{C_2},t_{latest_2})$,设合并后的网格为 $Cell(C,\vec{S}^1,\vec{S}^2,n_C,O_C,t_{latest})$,合并操作时的时间为 t_{cur},则

$$n_C=\theta^{t_{cur}-t_{latest_1}}\times n_{C_1}+\theta^{t_{cur}-t_{latest_2}}\times n_{C_2},\ \vec{S}^1=\theta^{t_{cur}-t_{latest_1}}\times\vec{S}_{C_1}^1+\theta^{t_{cur}-t_{latest_2}}\times\vec{S}_{C_2}^1,$$

$$\vec{S}^2=\theta^{t_{cur}-t_{latest_1}}\times\vec{S}_{C_1}^2+\theta^{t_{cur}-t_{latest_2}}\times\vec{S}_{C_2}^2,\ t_{latest}=t_{cur}.$$

性质 **2**. 以上网格合并操作得到的新网格仍然满足定义 5 中的权重衰减定义.在正态分布的假设下,以上网格分割操作得到的新网格也满足定义 5.

利用类似于 R 树的结构构造 Grid-Tree,有效实现网格存储、分割、合并、更新等操作.网格以超方体 C 的形式存放在 Grid-Tree 叶子结点中,同时还包括网格的 $\vec{S}^1,\vec{S}^2,n_C,t_{latest}$ 以及指向 O_C 的指针.在网格分割时,对其所在的叶子结点进行划分,在此结点上生成包含子网格的孩子节点;在网格合并时,将各个网格所在的孩子节点合并到其父节点上.在分割合并时,对于网格的统计信息以及网格中候选离群点集合也要进行相应的分割合并操作.为控制网格数量,算法根据内存限制设定树的最大深度 $h_{\max}$,当树的深度达到 $h_{\max}$ 时,不再对网格进行分割.

2.2.2 *CandOutlier* 的生成

为描述 Grid-Tree 的叶子节点的离群度,引入文献[13,14]中介绍的平滑因子概念.

给定一个项目集 I,设其幂集为 $P(I)$,并设存在差异函数 $D:P(I)\to R_0^+$,基数函数 $C:P(I)\to R_0^+$,使得对于任意的 $I_1,I_2\subseteq I$,有 $I_1\subset I_2\Rightarrow C(I_1)<C(I_2)$成立,定义 I 的任意子集 I_j 的平滑因子为 $SF(I_j)=C(I-I_j)\times(D(I)-D(I-I_j))$.

如果对于某个集合 $I_x\subset I,I_x\neq I$,及 $I_j\subset I$,有 $SF(I_x)\geq SF(I_j)$成立,则说 I_x 为离群点集合.

每个网格对应于数据流 DS 中的一个子集.取基数函数为网格中的数据点数 n_C,差异函数为网格中数据分布密度的倒数,即 V_C/n_C,其中,V_C 为超方体 C 的体积.可以看到,SF 值较大的网格具有较小的数据点数和与整个数据分布密度较大的差异,能够很好地反映整个网格中数据的离群度.本文所采用的离群点定义是以每个数据点周围的数据分布密度为依据,SF 值较大的网格中数据密度较小,将比 SF 值较小的网格中的数据更有可能成为离群点.对 SF 值较小的网格中的数据进行过滤,可以有效地减少候选数据点的规模.

通过计算各网格的 SF 值,可以得到 SF 值最大的前 m 个网格集合 Max$Cell=\{cell_1,cell_2,\ldots,cell_m\}$,并满足:

$\sum_{Cell\in \mathrm{Max}Cell} Cell.n_C \geq P\times w$ 且 $\sum_{Cell\in \mathrm{Max}Cell-\{Cell_m\}} Cell.n_C < P\times w$,于是得到: $CandOutlier = \bigcup_{Cell\in \mathrm{Max}Cell} Cell.O_C$.

为方便 Max*Cell* 的维护,建立 Max*Cell* 的索引结构.在索引中为每个 *cell* 建立一个节点,并记录 *cell* 的 SF 值和数据计数,链表以 *cell* 的 SF 值降序排序.索引可以降低由于 SF 值变化造成的 Max*Cell* 调整耗费.

生成的网格包括两类:属于 Max*Cell* 的和不属于 Max*Cell* 的.在读入新数据时,如果其属于 Max*Cell* 中的网格,则将该数据加入其所在网格的 O_C,并更新网格的统计信息;否则,只对网格信息进行更新,然后将数据释放.由于算法维护数据流中当前窗口的 *CandOutlier*,因此,对于 $t(\vec{r})$ 距离当前时间大于窗口宽度 w 的数据,需要对其进行删除操作.在具体实现时,可以采用链表结构存储 O_C,以方便删除操作.

随着 Max*Cell* 中各 O_C 的变化,*CandOutlier* 的大小也会随之发生变化.因此,在|*CandOutlier*|与 $P\times w$ 误差较大时,对 Max*Cell* 进行调整,加入或者删除其中的某些网格.当从 Max*Cell* 中删除某个网格时,对于其 O_C,在内存允许的情况下,不对其进行清空操作,而将清空操作推迟到可用内存不足的时候进行,这样有助于防止有用信息被过早地删除.类似于 Max*Cell* 索引,可以为 *CandOutlier* 建立相应的索引,收集 Max*Cell* 中各网格的候选离群点.

在数据具有较好的主体聚类性的情况下,通过 *CandOutlier* 与实际的离群点对比发现,*CandOutlier* 能够很好地覆盖离群点.因此,通过设置|*CandOutlier*|≈$Q\times w$,将仅仅维护 *CandOutlier* 的算法作为 FODDS 的一个简单、快速的版本 FODDS-S,并在实验中对其进行测试.

2.2.3 离群点检测

文献[8]通过计算每个数据在不同半径 r 下的 MDEF,得到其离群程度.而本文所维护的动态网格反映了实际的数据分布情况,并且已经获得候选离群点集合,可以方便、快速地对 MDEF 和 σ_{MDEF} 进行近似计算.

设 $p_i\in CandOutlier$,p_i 所在的网格为 $Cell_{p_i}(C,\vec{S}^1,\vec{S}^2,n_C,O_C,t_{latest})$,记超方体 C 的直径(即定义超方体的两个顶点之间的距离)为 $\Phi(C)$,设 $r=\Phi(Cell_p\cdot C)$,α=1/2,网格中数据可作为 $N(p_i,r)$的替代.于是,用网格 $Cell_{p_i}$ 中的数据数 n_C 估计 $n(p_i,\alpha r)$,即 $n(p_i,\alpha r)=n_C$.

为估计 $\hat{n}(p_i,r,\alpha)$,设以 p_i 为中心、$2r$ 为直径的超方体为 $Cell_r$,与 $Cell_r$ 相交的网格集合为 *Rcell*.为方便描述,定义符号: $\beta(C_1,C_2)=\frac{V_{C_1}\bigcap V_{C_2}}{V_{C_1}}$, $\delta(r,C_1)=\left(\frac{r}{\Phi(C_1)}\right)^3$,其中:$C_1,C_2$ 均为超方体;V_C 为超方体 C 的体积.

设 $n_{Cell}(p_i,\alpha r)=\beta(Cell.C,Cell_r.C)\times Cell.n_C$,在网格 *Cell* 中数据近似均匀分布的情况下,它可以作为 *Cell* 中与 p_i 距离在 r 范围内的数据点数的估计.对 $\hat{n}(p_i,r,\alpha)$ 进行近似计算为

$$\hat{n}(p_i,r,\alpha)=\frac{\sum_{cell\in RCell} n_{Cell}(p_i,r)\times\delta(r,Cell.C)\times Cell.n_C}{\sum_{cell\in RCell} n_{Cell}(p_i,r)}.$$

由 $n(p_i,\alpha r)$和 $\hat{n}(p_i,r,\alpha)$ 可计算 $MDEF(p_i,r,\alpha)$ $MDEF(p_i,r,\alpha)$ 和 $\sigma_{MDEF}(p_i,r,\alpha)$,从而进一步得到每个候选点的离群程度.

2.2.4 讨 论

算法可以进一步扩展到一般的数据流模型,即 Turnstile 模型.在这种模型中,数据流中的数据存在着到来和离去两种情况.网格中数据计数和统计信息不再仅仅是增加,而可能随着数据离去而减少.由于我们的算法对于更新操作并没有进行限制,因此可以很容易进行扩展.在这种情况下,当一个数据点离去时,我们在更新其所在网格时,需要减去其已经衰减后的计数和统计量,并且对于候选离群点也要进行更新,将其从中删除.

在前面的讨论中,我们将整个网格空间的体积作为近似计算的一个参数.对于各种数据集,数据分布的扭曲度可能不一样,这样,在某些网格中,虽然经过划分,可能数据仍然仅存在于网格中的某一个子空间(在较稀疏的网格中更容易出现).因此,我们可以采用网格中数据的最小闭包(包含网格中所有数据的最小超立方体)大小作为网格的实际体积进行计算.应该看到,计算中稠密网格对于结果影响较大,而根据我们的划分方法,这类网格可以看作数据拟均匀分布,采用网格空间体积仍然可以得到较好的结果,这也是本文中采用的方法.

3 算法性能分析与实验结果

3.1 复杂度分析

本文构造的数据流中离群点检测算法 FODDS 以及 FODDS-S 具有良好的空间与时间效率.

在算法运行过程中,需要维护有关网格信息的树状结构以及候选离群点集合 *CandOutlier*.其中:树状结构的大小通过分割合并网格以及 $h_{\max}$ 的限制进行控制;*CandOutlier* 的大小通过参数 P 进行调整.算法所需要保留的信息总可以控制在有限的内存范围内.下一节的实验结果进一步说明本文中的算法仅需要较少的内存.

性质 **3**. FODDS 算法具有相对于数据流数据量 N 线性规模的时间复杂度.

证明:对数据所在的网格信息更新、在分割合并网格的时间复杂度为 $O(N(h_{\max}+k))$.在最坏情况下,对网格离群度索引的维护需要 $O(2^{h_{\max}})$.当需要作离群点查询时,算法对于当前的 *CandOutlier* 进行筛选操作,最坏情况所需要的时间为 $O(|CandOutlier|\times 2^{h_{\max}})$.因此,总共需要 $O(N(h_{\max}+k)+(|CandOutlier|+1)\times 2^{h_{\max}})$. □

3.2 实验结果

采用仿真和实际数据对算法进行测试.选取 LOCI[8],GridOF[11]作为对比算法.这两种算法采用了一定的近似计算,具有可比性.但它们都是针对传统数据集的算法,没有时间维度的概念,因此对它们在整个数据集上进行测试.LOCI 算法的参数设置为 k_σ=3,α=1/2.实验在 CPU 为 PIV1.7G,内存 256M 的 PC 机上进行,算法由 C++ 实现.

为了测试内存使用情况,生成 2 维、3 维和 5 维仿真数据集,大小均为 100K,并设算法窗口长度为 w=10K,参数 P 分别取 1%,5%,10%,15%.数据集分为两种:DataSet1 包括两个主要聚类,一个服从正态分布,另一个服从平均分布,并加入 3%和两个聚类都具有较大偏差的离群点;DataSet2 包括一个服从正态分布的聚类和 3%的离群点.但与 DataSet1 不同的是,其中大约一半左右的离群点偏差较小,使得聚类性小于 DataSet1.算法运行稳定时,使用的内存如图 1 和图 2 所示.可以看到,算法所需要的内存保持在一个较小的范围内.

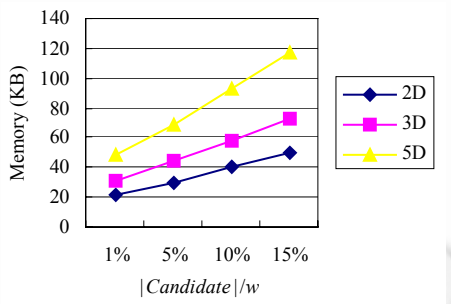

Fig.1 Memory used by FODDS-1

图 1 FODDS 内存使用-1

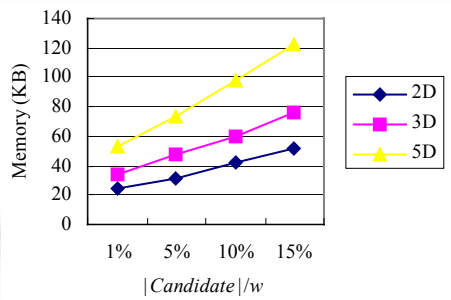

Fig.2 Memory used by FODDS-2

图 2 FODDS 内存使用-2

为了测试算法速度,对比 FODDS,FODDS-S,GridOF 和 LOCI 算法.采用 UCI KDD Archive 中森林覆盖类型分布数据集(forest CoverType)作为测试集.该数据集包含 581 012 条数据,每条数据包含 55 个属性字段,实验选择海拔、方向、坡度和树种属性.FODDS-S 算法取 P=1%,w=100K;FODDS 算法取 P=10%,w=100K.实验结果如图 3 所示.可以看到:FODDS 算法和 FODDS-S 算法由于对大量主体数据的过滤,在运行速度上表现出了其优越性.

为了测试算法的精确度,采用以下量度对算法进行评价:

$$Precision=\frac{Number\ of\ correct\ outliers}{|Outlier|}.$$

生成两个仿真 2 维数据集 DataSet1 和 DataSet2,大小均为 10K.数据采用与测试内存使用量时相同的生成方式.为了模拟数据流,将离群点均匀分布在整个数据集中,保证在每个窗口 w=1K 中都存在大致相同比例的离

群点.在两个数据集上运行 FODDS-S,FODDS 和 LOCI 算法,结果如图 4 所示.其中,FODDS-S(3),FODDS-S(10),FODDS(5)分别表示取 P=3%,10%,5%.

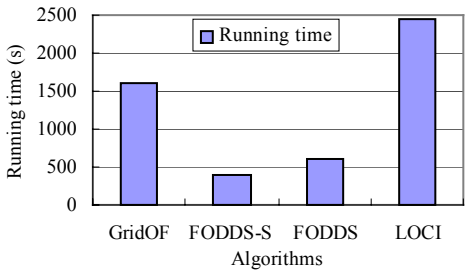

Fig.3 Running times of FODDS, FODDS-S, GridOF, and LOCI

图 3 FODDS,FODDS-S,GridOF 和 LOCI 的运行时间对比

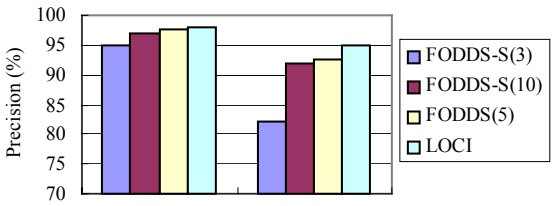

Fig.4 Precisions of FODDS, FODDS-S, GridOF, and LOCI

图 4 FODDS,FODDS-S,GridOF 和 LOCI 的精确度对比

可以看到:在数据集具有较好聚类性时,3 种算法都表现出了较好的精确度;而在 DataSet2 中,算法的精确度都有所下降,尤其是 FODDS-S 算法.但 FODDS-S(10)通过增加 *Outlier*(也就是 *CandOutlier*)的规模,得到了较好的精确度,代价是其中包含了更多的非离群点.考虑到 FODDS-S 算法在运行速度上表现出来的优势(如图 5 所示),它仍然具有较大的应用价值.

为了测试 FODDS 算法对概念转移的处理能力,生成服从不同正态分布的两组 2 维数据,大小均为 5K,类似地插入 3%偏离较大的离群点,并将两组数据合并,模拟发生概念转移的数据流.取窗口宽度为 w=2K,t_{life}=2K,$w_{\min}$=0.1,选择 FODDS(5)算法.对比了第 2K,4K,6K 和 8K 数据开始的 4 个窗口(分别标记为第 1、第 2、第 3 和第 4 个窗口)中离群点的精度,如图 6 所示.由于在第 2 个窗口两组数据发生混合现象,造成检测精确度降低.但随着时间的推移,第 1 组数据的影响逐渐减少,算法精确度也随之提高.

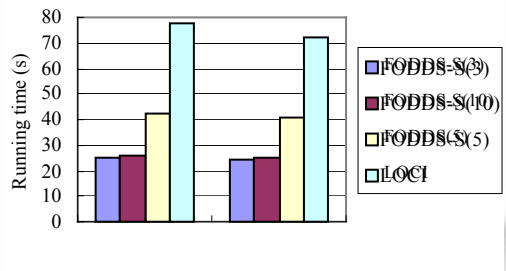

Fig.5 Running times of FODDS, FODDS-S, and LOCI

图 5 FODDS,FODDS-S,GridOF 和 LOCI 的运行时间对比

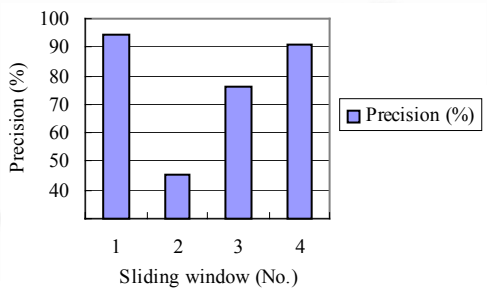

Fig.6 Precisions for different windows

图 6 不同窗口中的算法精确度

4 结 论

本文研究了数据流中离群点的检测问题,提出了 FODDS 算法及其快速版本 FODDS-S.算法利用具有时间特性的动态网格划分方法,对数据流的分布情况进行近似估计.通过对各网格的离群度分析,得到候选离群点集合 *CandOutlier*.对 *CandOutlier* 的筛选可以得到离群点集合 *Outlier* 作为结果提交给用户查询.实验结果表明,本文提出的算法是可行且有效的.进一步提高算法的实现效率,将其扩展到更一般的数据流模型,以及与数据流中其他相关的数据挖掘算法(例如聚类)进行结合,是下一步的研究内容.

References:

[1] Henzinger M, Raghavan P, Rajagopalan S. Computing on data streams. Technical Report, Report No.1998-011, Palo Alto: HP Digital Systems Research Center, 1998.

[2] Babcock B, Babu S, Datar M, Motawani R, Widom J. Models and issues in data stream systems. In: Popa L, ed. Proc. of the 21st ACM Symp. on Principles of Database Systems. Wisconsin: ACM Press, 2002. 1−16.

[3] Muthukrishnan S. Data streams algorithms and applications. In: Littleton R, ed. Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms. Philadelphia: Society for Industrial and Applied Mathematics, 2003. 413.

[4] Zhang T, Ramakrishnan R, Livny M. BIRCH: An efficient data clustering method for very large databases. SIGMOD Record, 1996, 25(2):103−114.

[5] Knorr EM, Ng RT. Algorithms for mining distance-based outliers in large datasets. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Databases. New York: ACM Press, 1998. 392−403.

[6] Yu D, Sheikholeslami G, Zhang A. Findout: Finding outliers in very large datasets. Knowledge and Information Systems, 2002, 4(4):387−412.

[7] Breunig MM, Kriegel H, Ng RT, Sander J. LOF: Identifying density-based local outliers. In: Dunham M, Naughton J, Chen W, eds. Proc. of the 2000 ACM SIGMOD Int'l Conf. on Management of Data. Dallas: ACM Press, 2000. 93−104.

[8] Papadimitirou S, Kitagawa H, Gibbons PB, Faloutsos C. LOCI: Fast outlier detection using the local correlation integral. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering. Bangalore, 2003. 315−326.

[9] Muthukrishnan S, Shah R, Vitter J. Mining deviants in time series data streams. In: Proc. of the 16th Int'l Conf. on Scientific and Statistical Database Management. Santorini Island: IEEE Computer Society, 2004. 41−50.

[10] Jagadish HV, Koudas N, Muthukrishnan S. Mining deviants in a time series database. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the 25th Int'l Conf. on Very Large Data Bases. Edinburgh: Morgan Kaufmann Publishers, 1999. 102−113.

[11] Li CH, Sun ZH. GridOF: An efficient outlier detection algorithm for very large datasets. Journal of Computer Research and Development, 2003,40(11):1586−1592 (in Chinese with English abstract).

[12] Park NH, Lee WS. Statistical grid-based clustering over data streams. SIGMOD Record, 2004,33(2):32−37.

[13] Arning A, Agrawal R, Raghavan P. A linear method for deviation detection in large databases. In: Simoudis E, Han J, Fayyad UM, eds. Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining. Portland: AAAI Press, 1996. 164−169.

[14] Chaudhary A, Szalay AS, Moore AW. Very fast outlier detection in large multidimensional data sets. In: Ganti V, Garofalakis M, eds. Proc. of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Madison: ACM Press, 2002.

附中文参考文献:

[11] 李存华,孙志挥.GridOF:面向大规模数据集的高效离群点检测算法.计算机研究与发展,2003,40(11):1586−1592.

杨宜东(1979－),男,安徽安庆人,博士生,主要研究领域为数据挖掘,知识发现.

杨明(1964－),男,博士,教授,主要研究领域为数据挖掘,机器学习,粗糙集理论及其应用.

孙志挥(1941－),男,教授,博士生导师,CCF高级会员,主要研究领域为数据库系统应用,知识发现.

张柏礼(1970－),男,博士,工程师,主要研究领域为数据仓库,数据挖掘,WEB.

朱玉全(1966－),男,博士,副教授,主要研究领域为复杂信息系统集成,知识发现,入侵检测.