ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.9, September 2006, pp.1969–1979
DOI: 10.1360/jos171969

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

基于分形技术的数据流突变检测算法*

秦首科+, 钱卫宁, 周傲英

(复旦大学 计算机科学与工程系,上海 200433)

Fractal-Based Algorithms for Burst Detection over Data Streams

QIN Shou-Ke+, QIAN Wei-Ning, ZHOU Ao-Ying

(Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-65643024, Fax: +86-21-65643503, E-mail: skqin@fudan.edu.cn, http://www.fudan.edu.cn

Qin SK, Qian WN, Zhou AY. Fractal-Based algorithms for burst detection over data streams. *Journal of Software*, 2006,17(9):1969−1979. http://www.jos.org.cn/1000-9825/17/1969.htm

Abstract: Burst detection over data streams has been attracting more and more attention from academic and industry communities due to its broad potential applications in venture analysis, network monitoring, trend analysis and so on. This paper aims at detecting bursts of both monotonic and non-monotonic aggregates over multiple windows in data streams. A burst detection algorithm through building monotonic search space based on fractal technique is proposed. First, the piecewise fractal model on data stream is introduced, and then based on this model the algorithm for detecting bursts is presented. The proposed algorithm can decrease the time complexity from $O(m)$ to $O(\log m)$, where m is the number of sliding windows being detected. Two novel piecewise fractal models can model the self-similarity and compress data streams with high accuracy. Theoretical analysis and experimental results show that this algorithm can achieve a higher precision with less space and time complexity as compared with the existing methods, and it could be concluded that the proposed algorithm is suitable for burst detection over data streams.

Key words: data stream; burst detection; fractal; piecewise fractal model; search space

摘 要: 数据流上的突变检测技术由于其在风险分析、网络监测、趋势分析等领域广阔的应用前景而受到学术界和工业界越来越多的关注.为了在数据流上检测多个滑动窗口上的单调聚集函数值和非单调聚集函数值的突变,提出了基于分形技术的构建单调搜索空间的突变检测算法.首先给出了数据流上的分段分形模型,进而基于该模型设计了突变检测算法.该算法能够将突变检测处理时间复杂度从 $O(m)$降为 $O(\log m)$(m 为需要被检测的滑动窗口数目).提出的两种新颖的分段分形模型能够准确地对数据流的自相似性进行建模并压缩数据流.理论分析和实验结果表明,与已有研究成果相比,算法具有较高的精度和较低的时间/空间复杂度,更加适用于进行数据流的突变检测.

关键词: 数据流;突变检测;分形;分段分形模型;搜索空间
中图法分类号: TP311 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60496325, 60496327, 60503034 (国家自然科学基金); the Shanghai Rising-Star Program of China under Grant No.04QMX1404 (上海市青年科技启明星计划)

Received 2005-09-19; Accepted 2005-12-13

数据流上的突变检测在金融风险分析、通信网监测、网络流量管理、趋势分析、Web 日志分析、网络入侵检测、传感器网络管理等领域具有广泛的应用.以金融风险分析为例,监管者和投资者希望通过突变检测来实时了解正在发生的异常过高的股票交易价格或交易量.在 Web 日志分析应用中,为避免网站因受到攻击或大量访问而瘫痪,网络系统管理人员希望能够借助突变检测技术来发现那些在一定时间内发生的过大的访问量.针对这类实时监控应用,人们提出了聚集突变检测技术来检测数据流上的聚集算子计算结果的突变行为.

对聚集计算结果的检测被认为是数据流上的查询模型的重要内容[1].许多现有的数据流管理系统(DSMS)都是由这类需求的驱动所设计的[2].在已有的研究工作中,文献[3]提出了预测聚集结果异常的方法.本文着重研究数据流上聚集计算结果突变的实时检测.Zhu 和 Shasha 在文献[4]中提出了一种检测 Gamma 射线突变的方法,该方法使用平移的小波树(shifted wavelet tree)在多个滑动窗口上检测聚集计算结果的突变.2005 年,Bulut 提出了一种更加高效的突变检测算法[5],该算法通过维护分层的最小限制矩形(MBR)来索引多个滑动窗口上的聚集计算结果.然而,以上两个突变检测算法[4,5]只能检测随滑动窗口长度单调变化的聚集函数值的突变(如 sum, count),而不能检测非单调的聚集函数(如 average)值的突变.文献[6,7]中提出了用于网络流量突变行为的检测方法.虽然它们使用的方法不同,但它们都有一个共同点,即只能同时检测单一窗口的变化.事实上,实际应用需要检测的滑动窗口长度往往难以确定.因此,为了能够检测到所有要求的突变,简单的办法就是分别运行多种算法.显然,这会导致大量的空间和时间开销,使得系统性能降低.本文与文献[8]的区别在于:提出了数据流上的分段分形模型和基于该模型的突变检测算法,完全基于分形技术对数据流建模并检测.

分形(fractal)模型适用于描述自然界中复杂的形状[9].在此基础上演变而成的分段分形模型更适合于描述实际应用中具有近似分形特性的数据,已被成功地运用于给定数据集的建模以及压缩[10,11].但是,为给定数据集求得具有全局最优解的分段分形模型是一个 NP-Hard 问题[12].现有的工作只能在给定数据集上求得局部最优解[11],且算法具有多项式的时间复杂度,并需要多次扫描数据集.鉴于数据流的实时性、无限性和连续性的特点[13],现有方法无法为数据流建立有效的分段分形模型.

本文的主要贡献可以归纳为以下几点:

1) 利用现实数据中存在的幂律伸缩性关系,定义了搜索空间的单调性,并给出了构造单调搜索空间的算法.将处理时间从 $O(m)$降为 $O(\log m)$,m 是需要被检测的滑动窗口的数目.

2) 提出了数据流上的分段分形模型,该模型利用不同长度滑动窗口上的聚集函数值的自相似性,对数据流上的聚集函数值进行了压缩,并提供了误差限制.

3) 基于单调的搜索空间以及数据流上的分段分形模型提出了数据流上的突变检测算法,可以在任意长度的滑动窗口上检测多种聚集函数值的突变,并通过实验进行了验证.

1 问题定义

基于对实际应用的分析,数据流上的突变检测可以形式化地定义如下:

定义 **1**. 设数据流 X 为$\{x_i:i=1,n\}$.已知聚集函数 F 和 m 个不同长度的滑动窗口$\{w_{li}:i=1,m\}$.$\{T(w_{li}):i=1,m\}$是与 m 个滑动窗口对应的突变阈值.滑动窗口 w_{li} 是数据流中长为 l_i 的序列.检测数据流 X 上的突变就是找到数据流中所有的 w_{li},在这些 w_{li} 上的聚集函数 F 的值 $F(w_{li})$大于或等于 $T(w_{li})$.

聚集函数 F 在本文中是指单调聚集函数 sum,count 以及非单调聚集函数 average,variance.现有的研究工作只能在数据流上检测单调的聚集函数值的突变.max 和 min 虽不具有分形特性,但它们的突变检测较为简单.

数据流上的突变检测主要存在以下几个难点:1) 在处理非单调的聚集函数时,无法依赖函数值的单调性减少要检测的不同长度的滑动窗口数目,因此突变检测的时间复杂度较大;2) 鉴于流数据的实时性、无限性和连续性的特点,在数据流上维护任意长度滑动窗口的聚集值是困难的.

2 分形的数学基础

一般把在自然界存在的自相似的现象称为分形(fractal).已有大量研究工作证明:自相似性是自然界中普遍

存在的现象.例如:网络上的通信行为、股票交易数据、人类的生理现象、气象数据等都是自相似的.式(1)所示的幂律伸缩性关系(power law scaling relationship)[9]作为分形模型的重要性质,是对自相似性进行数学描述的有力工具.s 为特征线度,如时间、长度;q 为 s 上的度量,如聚集计算;p 为比例常数;d 为伸缩性指数,对式(1)两端同时取对数可得式(2).

$$q=p\cdot s^d \tag{1}$$

$$\log q=\log p+d\cdot\log s \tag{2}$$

准确分形的数据是指在任意长度 s 上的度量 q 的值都遵守式(1)的数据,幂律伸缩性关系是判断准确分形的充要条件.因此,对于准确分形的数据,所有的点($\log s$,$\log q$)都在式(2)代表的直线上,d 为直线的斜率.而对于近似分形的数据,式(2)两端的值只具有相同的分布,d 则是所有点($\log s$,$\log q$)的线性回归线的斜率.

对于离散的时间序列数据以及流数据,式(1)可以变换为式(3)[10].t为基本的时间单位,s为度量q所需的基本时间单位的数目,s在数据流中即为滑动窗口的大小.因此,d可以由式(4)求得.

$$q(s\cdot t)=s^d q(t) \tag{3}$$

$$d=\frac{\log(q(s\cdot t)/q(t))}{\log s} \tag{4}$$

分段分形模型(piecewise fractal model)由有限多的收缩映射$\{M_i:i=1,k\}$组成.每一个收缩映射 M_i 对应一组分段 P_i 和 P_i',形如式(5).它的数学基础是循环迭代函数系统(recurrent iterated function system)[14].

$$M_i\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}a_{11}^i & a_{12}^i\\ a_{21}^i & a_{22}^i\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}b_1^i\\b_2^i\end{bmatrix},\quad i=1,2,\ldots,m \tag{5}$$

在 M_i 的系数中,a_{22}^i 称作收缩因子,且 $\left|a_{22}^i\right|<1$.为使该映射为单值映射,置 $a_{12}^i=0$.现有的方法只能在给定数据集上建立局部最优化的分段分形模型,目前还没有在数据流上构建分段分形模型或利用分形技术进行突变检测的研究工作.

3 建立单调的搜索空间

现有的突变检测算法虽然通过多种索引结构优化了对多个窗口进行检测的时间开销,但在最坏情况下仍需逐一检查所有长度的窗口,因此其时间复杂度仍为 $O(m)$,m 为不同长度的滑动窗口的数目.本文提出将所有要检测的滑动窗口排序,进而把无序的搜索空间变为单调的搜索空间.搜索空间的单调性是指,如果在某个窗口上检测到聚集函数值的突变,那么排在其前面的所有窗口上的聚集函数值也必然发生了突变.因此,检测突变时只需对单调的搜索空间进行折半查找(binary search),其时间复杂度仅为 $O(\log m)$.

3.1 基于分形的单调搜索空间

我们首先假设数据流$\{x_i:i=1,\ldots,n\}$是准确的分形数据.如图 1 所示[8],设单位时间长度滑动窗口上的聚集函数值为 $F(1)$,$F(1)>0$.数据流$\{x_i:i=1,\ldots,n\}$可以确定一条形如式(2)的斜线,其斜率为 d.d 是聚集函数 F 的幂律指数,可由式(4)求得.不同的F在不同的数据集上具有不同的d.假设给定聚集函数 F 的幂律指数 d 已知,$T(w_{li})$和 $T(w_{lj})$分别是窗口 w_{li} 和 w_{lj} 上给定的突变阈值.由图 1[8]可以看到,点$(\log(l_i),\log(T(w_{li})))$在斜线上方,点$(\log(l_j),\log(T(w_{lj})))$在斜线下方,因此有以下两个不等式成立:

$$\frac{\log(T(w_{li}))-\log(F(1))}{\log(l_i)}>d=\frac{\log(F(w_{li}))-\log(F(1))}{\log(l_i)},$$

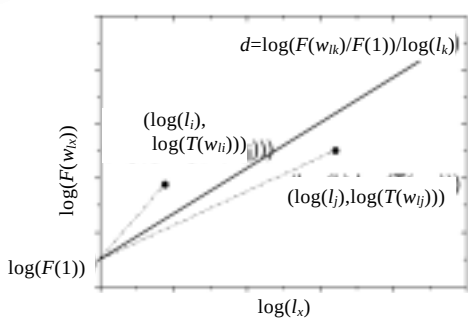


Fig.1 Monotonic search space of exact fractals

图 1 准确分形数据的单调搜索空间

$$\frac{\log(T(w_{lj}))-\log(F(1))}{\log(l_j)}<d=\frac{\log(F(w_{lj}))-\log(F(1))}{\log(l_j)}.$$

当处理数据流 X 时,总能够在 w_{lj} 上检测到突变,而 w_{li} 上则永远不可能发生突变,这是因为总有 $\log(T(w_{lj})/F(1))/\log(l_j)<d<\log(T(w_{li})/F(1))/\log(l_i)$ 成立,即总有 $T(w_{lj})<F(w_{lj})$ 和 $T(w_{li})>F(w_{li})$ 成立.因此,可以通过比较 $\log(T(w_{li})/F(1))/\log(l_i)$ 与 d 的大小达到检测突变的目的.在此基础上,可以得到如下性质:

性质 **1**. 设$\{T(w_{li}):i=1,\ldots,m\}$是与数据流 X 上的 m 个滑动窗口对应的突变阈值,并且所有窗口按照比值 $\log(T(w_{li})/F(1))/\log(l_i)$的升序排列.如果窗口 w_{li} 上发生了突变,那么排在其前面的所有窗口均会产生突变[8].

对于实际应用中的近似分形数据,可以利用性质 1 对实际应用中的数据进行分形变换.这样会导致两种误差,我们将它们分别定义为分形近似的误差 err_{FA}(error of fractal approximation)和错排序的误差 err_{MS}(error of mis-sorting).分形近似的误差 err_{FA} 是由于对数据进行分形变换导致的.设 $F(w_{li})$的数学期望和标准差分别为 $E(F(w_{li}))$和 $D(F(w_{li}))$,假设$(F(w_{li})-E(F(w_{li})))/D(F(w_{li}))\sim Norm(0,1)$.分形变换就是用幂律伸缩性关系来近似实际应用中的数据,即用统计值来近似 $F(w_{li})$.如果有 $F(w_{li})<T(w_{li})$,那么这种近似不会带来任何误差,因为它不会导致错误地报告突变;而如果有 $F(w_{li})\geq T(w_{li})$,那么,窗口 w_{li} 上的突变则有可能被误报.设 err_{FA} 是产生这种误差的概率,当 $T(w_{li})$给定时,err_{FA} 的上限可以被确定

$$err_{FA}\leq P(F(w_{li})\geq T(w_{li}))=P\left(\frac{F(w_{li})-E(F(w_{li}))}{D(F(w_{li}))}\geq\frac{T(w_{li})-E(F(w_{li}))}{D(F(w_{li}))}\right).$$

错排序的误差 err_{MS} 是由使用固定不变的 $F(1)$导致的.如前所述,对准确分形数据来说,$F(1)$和 d 都是定值.而对于近似分形数据,$F(1)$和 d 都是随时变化的.例如,对聚集函数 sum 来说, $F(1)=\frac{1}{n}\sum_{i=1}^{n}x_i$,当有新的 x_n 到来时,$F(1)$就可能产生变化.通过人工构建单调的搜索空间可以完全避免错误排序的误差.

3.2 单调搜索空间的构建算法

本节将给出通用的单调搜索空间构建算法,并通过例子对算法的流程予以描述,见算法 1.算法 1 是一个增量算法,它描述了当有新的窗口 w_{lk} 需要被检测时,如何更新现有的单调搜索空间.当需要检测的窗口数为 2 时,已知两个突变阈值 $T(w_{li})$和 $T(w_{lj})$,$w_{li}<w_{lj}$.通过两点$(\log(l_i),\log(T(w_{li})))$和$(\log(l_j),\log(T(w_{lj})))$的直线交 y 轴于点 A.如果点 $(0,\log(F(1)_{d_p}))$ 在点 A 的上方,那么,与 d_p 对应的单调搜索空间为$\langle w_{li},w_{lj}\rangle$;如果点 $(0,\log(F(1)_{d_p}))$ 在点A的下方,则与 d_p对应的单调搜索空间为$\langle w_{lj},w_{li}\rangle$.这是为两个窗口建立单调搜索空间的方法.当有更多窗口时,只需运行该算法,增量地更新现有的搜索空间即可.下面以 3 个窗口为例加以说明.

算法 **1**. 单调搜索空间的构建算法.

```
buildMonotonicSearchSpace (w_i,T(w_i)).
create a new point T_i(log(l_i),log(T(w_li)));
if m≥1 then
    for j=1 to m do
line T_iT_j intersect vertical axis at point X_i;
        if w_li<w_lj then
            add ⟨w_lj,w_li⟩ to the lower intervals of X_i;
            add ⟨w_li,w_lj⟩ to the upper intervals of X_i;
        else
            add ⟨w_li,w_lj⟩ to the lower intervals of X_i;
            add ⟨w_lj,w_li⟩ to the upper intervals of X_i;
        end if
    end for
end if
increase m by 1
```

当 w_{lk} 加入时,更新各个区间上的单调搜索空间的过程如下:假设 w_{li} 和 w_{lj} 的单调搜索空间已经建立,即区间 $(-\infty,A]$对应$\langle w_{lj},w_{li}\rangle$,区间$(A,+\infty)$对应$\langle w_{li},w_{lj}\rangle$.算法 1 在第 1 行生成点 T_k,此时要检测的窗口数目 m=2.第 4 行用直线连接两点 T_i 和 T_k,交 y 轴于点 B,将$(A,+\infty)$分为两个区间$(A,B]$和$(B,+\infty)$.因为 $w_{li}<w_{lk}$,所以将$\langle w_{lk},w_{li}\rangle$添加到点 B

下方所有区间各自对应的搜索空间中,即将$\langle w_{lk},w_{li}\rangle$添加到$(-\infty,A]$和$(A,B]$各自对应的搜索空间中.再将$\langle w_{li},w_{lk}\rangle$添加到点 B 上方所有区间各自对应的搜索空间中,即将$\langle w_{li},w_{lk}\rangle$添加到$(B,+\infty)$对应的搜索空间中.接下来回到算法的第 4 行继续执行,用直线连接两点 T_j 和 T_k,交 y 轴于点 C,将$(B,+\infty)$分为两个区间$(B,C]$和$(C,+\infty)$,然后更新每个区间的单调搜索空间.最后将要检测的窗口数目增加到 3.

因为突变检测的阈值一般在检测突变前设置完成,所以该算法可以离线完成单调搜索空间的构建任务,因此其性能对数据流的检测不会有任何影响.与文献[8]中的单调搜索空间构建算法相比,算法 1 将时间复杂度从$O(m^3)$降为 $O(m^2)$,m 是目前需要被检测的窗口数目.

4 数据流上的分段分形模型

上一节介绍了构建单调搜索空间的方法,在获得了单调的搜索空间以后,根据第 3.1 节中的分析,只需求得当前的 d_p,将其与单调搜索空间中的窗口所对应的值 $\log(T(w_{li})/F(1))/\log(l_i)$进行比较,从而高效地完成突变检测的任务.因此,为了在数据流上维护任意长度的滑动窗口的聚集值以计算 d_p,本节提出了数据流上的分段分形模型,该模型利用不同长度滑动窗口上的聚集函数值的自相似性,既可以压缩单调的聚集函数值,又可以压缩非单调的聚集函数值.基于单调的搜索空间以及数据流上的分段分形模型,本节又提出了数据流上的突变检测算法.

当前数据流上聚集函数值的幂律关系 d_p 就是点集$\{\log(l_i),\log(F(w_{li})):l_i=1,\ldots,n\}$的线性回归线的斜率.$F(w_{li})$是需要被检测的滑动窗口 w_{li} 上的聚集函数值.假设 F 是 sum,那么,$F(w_{li})$就是当前数据流$\{x_i:i=1,\ldots,n\}$的后缀和,即 $F(w_{li})=\sum_{j=n-li+1}^{n}x_j$.由此可见,在数据流上维护 d_p 的开销非常大,并且无法通过增量算法实现.然而,如果点集$\{\log(l_i),\log(F(w_{li})):l_i=1,\ldots,n\}$中的数据点是分段线性的,那么只需保留这些线性分段的端点,因为对这些端点求得的线性回归线与对整个点集求得的线性回归线完全相同.在本文下面的叙述中,均假设 F 是 sum 且不失一般性.

因此,我们接下来的工作就是要为当前的数据流建立分段的分形模型,通过将数据流划分为具有分形特性的分段,我们便可求得点集$\{\log(l_i),\log(F(w_{li})):l_i=1,\ldots,n\}$中数据点的线性分段.

4.1 建立数据流上的分段分形模型

设数据流中的点可以表示为(时间戳:i,值:y_i),一段数据流 P 的起始端点为(s,y_s),结束端点为(e,y_e),$e>s$.由于数据流中的点是离散的,我们将 M 定义为形如 $M(i,y_i)=(\mathrm{int}(i\cdot a_{11}+b_1),a_{21}\cdot i+a_{22}\cdot y_i+b_2)$的收缩映射.$M$ 将数据流中较大的区间 $P\{(s,y_s),(e,y_e)\}$内的数据点映射到同一个数据流中较小的区间 $P'\{(s',y_{s'}),(e',y_{e'})\}$ 内,P'是 P 的子集,即 $P'\subset P$ 且 $e-s>e'-s'$.每两个相邻的 P'只在端点处重合,其余没有相互重叠的部分,所有分段被合并后能够覆盖原有数据流,即 $\bigcup_{i=1}^{k}P_i=\bigcup_{i=1}^{k}P_i'=\{x_j:j=1,\ldots,n\}$.已知 P 和 P',如何设置收缩因子 a_{22},使由 M 映射到分段 P'内的数据序列 $M(P)$与 P'内原有的数据序列之间的 L_2 误差$(E=(M(P)-P')^2)$最小,这是本节要解决的问题.

M 对应的误差 $E=\sum_{i=s}^{e}(A_ia_{22}-B_i)^2$, $A_i=y_i-\left(\frac{e-i}{e-s}y_s+\frac{i-s}{e-s}y_e\right)$, $B_i=y_j-\left(\frac{e-i}{e-s}y_{s'}+\frac{i-s}{e-s}y_{e'}\right)$.根据文献[11],具有最小误差的收缩映射 M_{opt} 的系数 a_{22} 可由 E 相对于 a_{22} 的偏导数等于 0 求得,即 $\frac{\partial E}{\partial a_{22}}=2\sum_{i=s}^{e}(A_ia_{22}-B_i)A_i=0$,$a_{22}=\sum_{i=s}^{e}A_iB_i\Big/\sum_{i=s}^{e}A_i^2$.由此可见,只要在数据流上维护 $\sum_{i=s}^{e}A_iB_i$ 和 $\sum_{i=s}^{e}A_i^2$,便可求得具有最优解的收缩映射 M_{opt}.由于需要单遍扫描数据以及次线性的空间开销,在数据流上无法准确地维护 $\sum_{i=s}^{e}A_iB_i$.因此,我们的算法通过近似地计算 $\sum_{i=s}^{e}A_iB_i$ 获得近似的最优解 M_{opt},近似算法用 a'_{22} 代替 a_{22}, $(a'_{22})^2=\sum_{i=s}^{e}B_i^2\Big/\sum_{i=s}^{e}A_i^2$.本文提出两种数据流上的分段分形模型,模型 II 在模型 I 的基础上进行了改进.

如图 2 所示,分段分形模型 I 由最近的 B 个桶(bucket)构成,每个桶 b_i 对应一个收缩映射 M_i 和一组分段 P_i 及 P_i' .b_i 只需保存从分段 P_i' 的起始端点 $(si', y_{si'})$ 开始的后缀和 $F(w_{n-si'})$ 及其所包含的数据点的数目.在保证 $(a'_{22})^2<1$ 的前提下,不断向当前桶 b_i 对应的分段 P_i 及 P_i' 内加入新到达的流数据 x_n,并以 x_n 作为 P_i 及 P_i' 的结束端点.当 $(a'_{22})^2\geq 1$ 时,创建一个新的桶 b_{i+1},将 P_i' 作为新的 P_{i+1} 并生成新的 P'_{i+1}, x_{n-1} 作为 P'_{i+1} 的起始端点,新到达的 x_n 作为 P_{i+1} 及 P'_{i+1} 的结束端点.因此, $P_{i+1}=P_i'\cup P'_{i+1}$.

由于分形描述了整段数据与其自身某一部分数据的相似性,为了对整个数据集进行更好的近似,分段分形模型 II 扩展了模型 I 中的 P_i,使其包括截止当前数据点 x_n 之前的所有流数据.如图 3 所示,分段分形模型 II 由 B 个桶构成,每个桶 b_i 对应一个收缩映射 M_i 和一组分段 P_i 及 P_i' .b_i 只需保存从分段 P_i' 的起始端点 $(si', y_{si'})$ 开始的后缀和 $F(w_{n-si'})$ 及其所包含的数据点的数目.由于数据流中不断有新值到来,所以需要将每一个新到来的值都添加到现有的桶中.在保证 $\sum_{i=1}^{n}B_i^2\Big/\sum_{i=1}^{n}A_i^2<1$ 的前提下, $\sum_{i=1}^{n}B_i^2=\frac{n-1}{n-si'}\sum_{i=si'}^{n}\left(y_i-\left(\frac{n-i}{n-si'}y_{si'}+\frac{i-si'}{n-si'}y_n\right)\right)^2$,不断向当前桶 b_i 对应的分段 P_i 及 P_i' 内加入新到达的流数据 x_n,并以 x_n 作为 P_i 及 P_i' 的结束端点.当 $\sum_{i=1}^{n}B_i^2\Big/\sum_{i=1}^{n}A_i^2\geq 1$ 时,创建一个新的桶 b_{i+1},将 P_i 作为新的 P_{i+1} 并生成新的 P'_{i+1}, x_{n-1} 作为 P'_{i+1} 的起始端点,新到达的 x_n 作为 P_{i+1} 及 P'_{i+1} 的结束端点.因此,总有 $P_{i+1}=P_i\cup P'_{i+1}$.当桶的数目大于 B 时,删除最早建立的桶,只保存最近的 B 个桶.如上所述,在数据流上维护分段分形模型 II 的算法见算法 2.

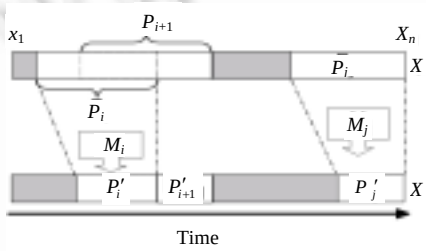


Fig.2 Piecewise fractal model I

图 2 分段分形模型 I

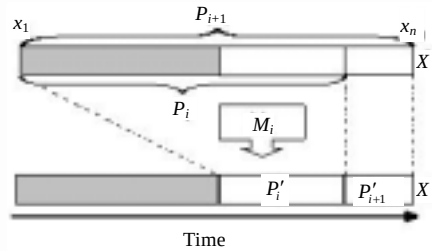


Fig.3 Piecewise fractal model II

图 3 分段分形模型 II

算法 **2**. 分段分形模型构建算法.

updatePiecewiseFractalModel2 (x_n).

compute $\sum_{i=1}^{n}B_i^2$ approximately;

compute $\sum_{i=1}^{n}A_i^2$;

if $\sum_{i=1}^{n}B_i^2\Big/\sum_{i=1}^{n}A_i^2<1$

add x_n to current bucket;

else

create a new bucket for x_{n-1} and x_n;

bucketNum++;

if *bucketNum*>*B*

delete the oldest bucket;

end if

end if

定理 **1**. 算法 2 给出近似的局部最优解 M_{app} 将分段 P 映射到 P' 的 L_2 误差为 $E(M_{app})$,局部最优解 M_{opt} 将分段 P 映射到 P' 的 L_2 误差为 $E(M_{opt})$,那么总有 $E(M_{app})\leq 2E(M_{opt})$.

证明:略.

算法 2 的时间开销为 $O(n)$,空间开销为 $O(B)$.为了优化算法,将误差限制在更小的范围内,我们提供参数α用以控制当前分段的误差 $E(M_{app}) \le \alpha \cdot E(P'_{LSF})$, $E(P'_{LLF})$ 是分段 P'与其上的最小线性近似(least linear fitting)的 L_2 误差.因此,分段分形模型对当前分段 P'的近似误差 $E(M_{app}) \le \min\{\alpha \cdot E(P'_{LSF}), 2E(M_{opt})\}$.实现参数$\alpha$的控制非常简单,只需在将 x_n 加入桶 b_i 时判断是否存在 $E(M_{app}) \le \alpha \cdot E(P'_{LSF})$,否则,新建一个桶 b_{i+1},其余与算法 2 相同.

4.2 基于分段分形模型的突变检测算法

当数据流中新到来一个点 x_n 时,先将 x_n 加入到分段分形模型中,并对分段分形模型中由现有的桶所保存的 B 个分段的端点构成的点集 $\{(\log(n-si'), \log(F(w_{n-si'})) : i=1,...,B\}$ 求得其上的线性回归线,d_p 为线性回归线的斜率.计算线性回归线的方法较为简单,这里不再赘述.然后,使用算法 3 检测此时是否有突变发生.算法 3 不含输入参数,它的输出是突变报警信息,本文在这里返回的是发生突变的窗口大小.该算法在第 1 行定位当前的 $\log(F(1)_{d_p})$ 对应的区间 I_k,第 2 行获取区间 I_k 的单调搜索空间,并放入 *Array* 中.从第 4 行~第 13 行对单调的搜索空间 *Array* 进行折半查找.算法 3 的时间开销仅为 $O(\log m)$,m 是需要被检测的滑动窗口数目.如果用户只想在 x_n 到达时检测是否有突变发生,而并不关心突变的数目以及发生突变的窗口大小,那么时间开销仅为 $O(1)$.设分段分形模型所用桶的数目为 B,数据流长度为 n,可以得到如下定理:

定理 **2**. 算法 3 在数据流上同时检测 m 个不同长度的滑动窗口上的突变信息时,空间复杂度为 $O(B)$,时间复杂度为 $O(n\log m)$.

算法 **3**. 突变检测算法.

```
detectBursts().
locate  log(F(1)_{d_p})  at interval I_k;
Array←retrieve monotonic search space of I_k;
low←0, high←Array. size()−1;
    while low≤high do
mid←(low+high)/2;
        if log(T(Array[mid])/F(1)) / log(Array[mid]) ≤d_p then
            low←mid+1;
        else
            if log(T(Array[mid])/F(1)) / log(Array[mid]) >d_p then
                high←mid−1;
            end if
        end if
    end while
return high
```

5 实验分析

5.1 实验设置

本文实验中用到的所有代码,包括单调搜索空间的构建算法、分段分形模型的构建算法以及突变检测算法,均采用 Microsoft Visual C++ Version6.0 实现.运行实验的机器是一台 2.4GHz CPU 的 Dell PC,具有 512MB 主存,使用 Windows 2000 操作系统.我们在多组数据集上进行了初步的实验,由于篇幅有限,这里只给出部分实验结果.

- 数据集 1(真实数据):这个数据集是 2002 年在香港联交所主板挂牌交易的多家上市公司的股票数据.我们随机选取了一家公司全年的交易数据.其中每一条记录包括交易时间和交易价格等.该数据集共有 n=485983 条记录.

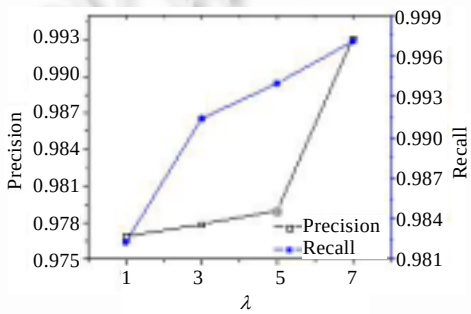
• 数据集 2(真实数据):该数据集是从 Internet Traffic Archive (http://ita.ee.lbl.gov/)下载的.它包含了从 1998 年 4 月 26 日~1998 年 7 月 26 日期间对“世界杯”网站的所有访问请求.在这 92 天里,“世界杯”网站共收到 1 352 804 107 个访问请求.数据集大小为 n=92×24×3600s=7948800 条记录.

这里,我们使用查全率(recall)和查准率(precision)来描述突变检测算法的精度.查全率是指检测到真实突变的数目与所有真实突变数目的比值.查准率是指检测到真实突变的数目与所有检测到的突变数目的比值.

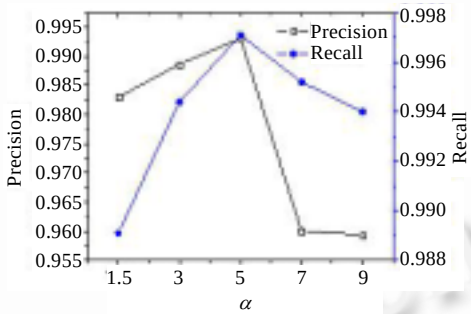
对 $T(w_{li})$的设置,首先从数据集 1 和数据集 2 中分别取出前 1 000 和前 10 000 条记录作为训练数据,然后在每个训练数据集上计算不同长度滑动窗口 w_{li} 的聚集值,聚集函数 F 为 sum,这样对应每一个长度为 l_i 的滑动窗口将生成一个新的时间序列 y_{li}.$T(w_{li})=avg(y_{li})+\lambda std(y_{li})$,$avg(y_{li})$和 $std(y_{li})$分别是 y_{li} 的均值和方差.因为 avg 和 std 已被证明具有分形特性,所以根据式(1), $avg(y_{li}) = p_{avg} \cdot s_{avg}^{li}$, $std(y_{li}) = p_{std} \cdot s_{std}^{li}$.可以用λ调节突变检测的阈值.需要被检测的滑动窗口长度依次为 4,4×2,4×3,…,4×NW.NW (number of windows)为需要被检测的窗口数目.

5.2 性能分析

如图 4 所示,第 1 组实验分析了基于分段分形模型Ⅰ的突变检测算法的准确性.阈值的大小由λ调节.同时,检测的窗口数目 NW=70,所用桶的数目 B=8,α=7,采用数据集 1.突变检测的准确性与突变阈值之间的关系如图 4(a)所示.可以看到,在任意λ设置下,算法都有较高的准确率.分段分形模型的误差与突变检测结果的准确性之间的关系如图 4(b)所示.当λ=7 时,随着系数α的增大,分段分形模型对数据流的近似误差也在增大,此时查全率和查准率在一定范围内上升;当α超过某个值后又开始下降.



(a) Varying threshold
(a) 调节阈值



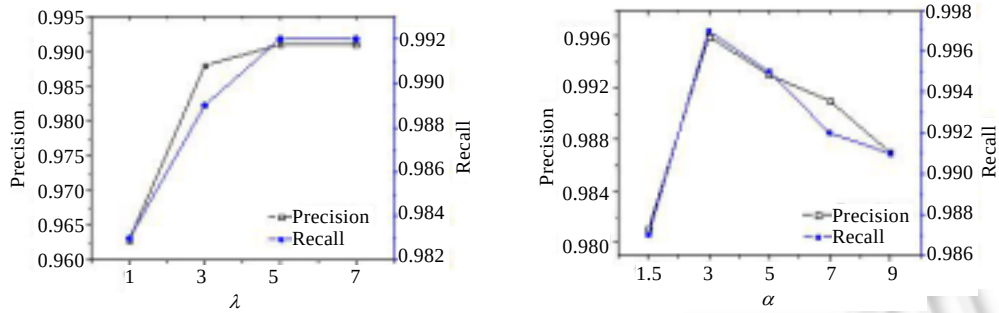
(b) Varying α
(b) 调节α

Fig.4 Accuracy of PFM Ⅰ

图 4 分段分形模型Ⅰ的准确性

第 2 组实验分析了基于分段分形模型Ⅱ的突变检测算法的准确性.实验设置与第 1 组实验相同,只是该组实验均在较大的数据集 2 上进行.经验证,基于分段分形模型Ⅱ的突变检测算法比基于模型Ⅰ的算法更为准确.但由于篇幅有限,这里只给出基于分段分形模型Ⅱ的实验结果.突变检测的准确性与突变阈值之间的关系如图 5(a)所示.可以看到,在任意λ设置下,算法都有较高的准确率,并且随着突变阈值的增大,即随着发生突变的可能性的减小,查全率和查准率都在上升.这正如第 3.2 节中的分析:当突变阈值增大时,分形近似误差 err_{FA} 随之减小.在实际应用中,发生突变的数据点只占所有数据点中很小的一部分,因此,本文的方法能以很高的准确率检测到数据流中的突变.

分段分形模型的误差与突变检测结果的准确性之间的关系如图 5(b)所示, λ=5,与第 1 组实验相同的是,查全率和查准率在一定范围内上升,当α超过某个值后又开始下降.这说明,α的设置并不是越小越好.这是因为每一个分段的误差越小,那么该分段所包含的数据点也就越少.对于一个给定桶的数目为 B 的分段分形模型来说,其所覆盖的数据流的长度就越小,对大窗口的聚集函数值的估计就会产生较大的误差,因此,突变检测的误差也就随之增大.反之则很显然,将α设置得越大,每一个桶对其所保存的数据流分段的近似程度就越差,因此,突变检测的误差也会随之增大.

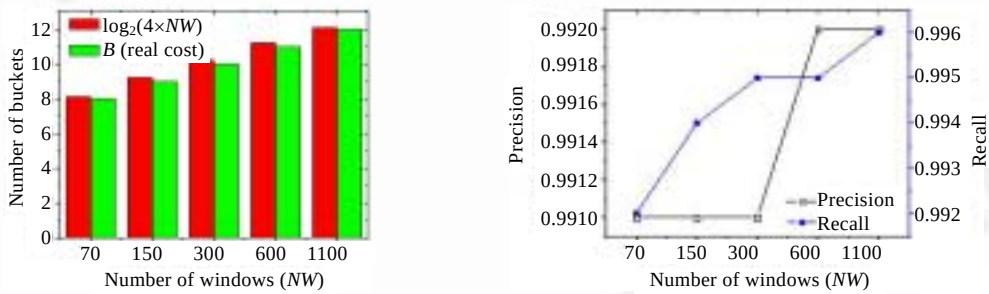


(a) Varying threshold
(a) 调节阈值
(b) Varying α
(b) 调节α

Fig.5 Accuracy of PFM Ⅱ

图 5 分段分形模型 Ⅱ 的准确性

第 3 组实验分析了同时检测的窗口数目增多对突变检测结果的影响.实验设置如下:误差系数α=7,突变阈值λ=5.随着检测窗口数目 NW 的增大,所用桶的数目 B 也随之增大,但如图 6(a)所示,算法保证 $B<\log_2(4\times NW)$.因此,我们的空间开销小于所检测的数据流长的次线性需求.如图 6(b)所示,在空间 B 很小的情况下检测精度依旧很高.随着被检测窗口数目的增多,查全率和查准率都在上升,这是因为在被检测窗口中,大窗口数目所占的比例在上升,由于数据量越大越能体现出自相似性,所以分段分形模型对大窗口上聚集函数值的估计则更加准确,突变检测的误差也就更小.

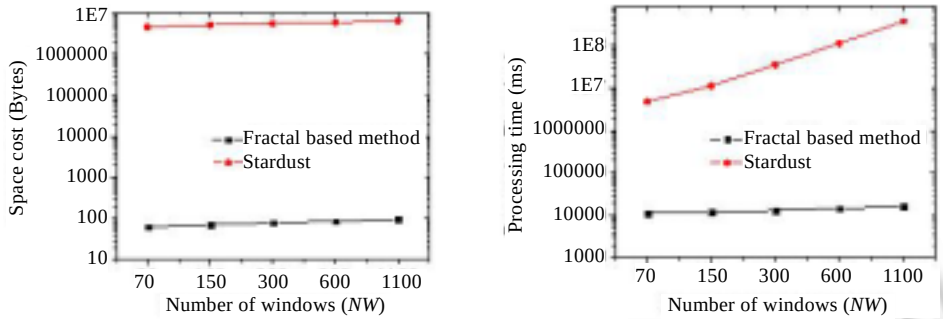


(a) Space cost
(a) 空间开销
(b) Accuracy
(b) 准确性

Fig.6 Burst detection on multiple windows

图 6 多窗口上的突变检测

第 4 组实验将本文提出的突变检测算法与现有算法中最新的研究成果进行了对比.我们使用由文献[5]的作者 Bulut 博士提供的 Stardust 的源代码进行对比测试.实验设置如下:误差系数α=9,突变阈值λ=5,基于数据集 2. Stardust 是一种准确的突变检测算法,即不会误报突变,但是其时间和空间开销对数据流上多窗口的突变检测任务来说非常巨大.如图 7 所示为空间和时间开销的对比实验结果.首先,通过对比可以看到,虽然本文的算法是一种近似算法,但在允许一定误差的前提下,本文的突变检测方法所节约的时间和空间开销非常可观,因此更加适用于解决数据流上的突变检测问题;其次,随着被检测窗口数目的增多,检测任务也在不断加大,而此时,本文的方法在时间和空间开销方面并没有明显增大,这说明本文的突变检测方法在处理大规模问题时具有更好的可伸缩性(scalability).



(a) Compare space cost
(a) 空间开销对比

(b) Compare time cost
(b) 时间开销对比

Fig.7 Piecewise fractal model on data stream

图 7 数据流上的分段分形模型

6 结论及展望

本文首次将分形技术应用于解决数据流的突变检测问题.基于分形技术,我们提出了构建单调搜索空间的算法以及数据流上的分段分形模型,并在此基础上提出了数据流的突变检测算法.本文设计的突变检测方法将突变检测的处理时间从 $O(m)$降为 $O(\log m)$,m 是需要被检测的滑动窗口数目.理论分析和实验结果表明,本文提出的算法具有较高的检测精度以及较低的时间和空间复杂度.我们将继续扩展目前的研究工作,致力于动态地决定最合适的检测窗口的大小.

致谢 在此,我们向为本文提供了 Stardust 源代码的 Ahmet Bulut 博士表示感谢.

References:

[1] Brian B, Shivanath B, Mayur D, Rajeev M, Jennifer W. Models and issues in data stream systems. In: Michael JF, ed. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2002. 1−16.

[2] The STREAM group. STREAM: The Stanford stream data manager. IEEE Data Engineering Bulletin, 2003,26(1):19−26.

[3] Li JZ, Guo LJ, Zhang DD, Wang WP. Processing algorithms for predictive aggregate queries over data streams. Journal of Software, 2005,16(7):1252−1261 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1252.htm

[4] Zhu YY, Shasha D. Efficient elastic burst detection in data streams. In: Getoor L, Senator TE, Domingos P, Faloutsos C, eds. Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. New York: ACM Press, 2003. 336−345.

[5] Bulut A, Singh AK. A unified framework for monitoring data streams in real time. In: Kawada S, ed. Proc. of the 21st Int'l Conf. on Data Engineering (ICDE 2005). Tokyo: IEEE Computer Society, 2005. 44−55.

[6] Cormode G, Muthukrishnan S. What's new: Finding significant differences in network data streams. IEEE/ACM Trans. on Networking, 2005,13(6):1219−1232.

[7] Krishnamurthy B, Sen S, Zhang Y, Chen Y. Sketch-Based change detection: Methods, evaluation, and applications. In: Crovella M, ed. Proc. of the 3rd ACM SIGCOMM Conf. on Internet Measurement. New York: ACM Press, 2003. 234−247.

[8] Qin SK, Qian WN, Zhou AY. Approximately processing multi-granularity aggregate queries over a data stream. In: Liu L, ed. Proc. of the 22nd Int'l Conf. on Data Engineering (ICDE 2006). Atlanta: IEEE Computer Society, 2006. 67−76.

[9] Mandlebrot BB. The Fractal Geometry of Nature. New York: Freeman, 1982. 37−47.

[10] Borgnat P, Flandrin P, Amblard PO. Stochastic discrete scale invariance. IEEE Signal Processing Letters, 2002,9(6):181−184.

[11] Mazel DS, Hayes MH. Using iterated function systems to model discrete sequences. IEEE Trans. on Signal Processing, 1992,40(7):1724−1734.

[12] Fiedler B. Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems. New York: Springer-Verlag, 2001. 617−647.

[13] Jin CQ, Qian WN, Zhou AY. Analysis and management of streaming data: A survey. Journal of Software, 2004,15(8):1172−1181 (in Chinese with English abstract). http://www. jos.org.cn/1000-9825/15/1172.htm

[14] Barnsley MF, Elton JH, Hardin DP. Recurrent iterated function systems. Constructive Approximation, 1989,5(1):3−31.

附中文参考文献:

[3] 李建中,郭龙江,张冬冬,王伟平.数据流上的预测聚集查询处理算法.软件学报,2005,16(7):1252−1261. http://www.jos.org.cn/1000-9825/16/1252.htm

[13] 金澈清,钱卫宁,周傲英.流数据分析与管理综述.软件学报,2004,15(8):1172−1181. http://www.jos.org.cn/1000-9825/15/1172.htm

秦首科(1978－),男,陕西宝鸡人,博士生,主要研究领域为数据流管理,数据挖掘,对等计算.

周傲英(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据流管理与分析,对等计算,Web 数据管理.

钱卫宁(1976－),男,博士,讲师,主要研究领域为数据流处理,对等计算.

第 2 届中国分类技术及应用学术会议(CCTA 2007)
征 文 通 知

由中国计算机学会人工智能与模式识别专业委员会主办，郑州大学承办，河南大学，河南省计算机学会协办的第 2 届中国分类技术及应用学术会议(CCTA 2007)将于 2007 年 5 月 27 日~29 日在郑州召开。

征稿范围（不局限于下述范围）

分类和聚类的基础理论与算法：监督学习、半监督学习和非监督学习，分类和聚类算法与复杂性，分类与聚类的数据预处理技术，分类与聚类模型的评估与选择，集成/组合分类技术，基于约束的聚类，基于分类和聚类的异常检测

复杂数据类型和特定领域的分类和聚类技术：动态数据流的分类和聚类，社会网络分析，空间数据的分类和聚类，文本分类和聚类，数据密集场景中的分类，Web 页面分类和聚类，时间序列数据的分类和聚类，图像与视频检索，计算机视觉中的分类，生物特征识别中的分类

分类和聚类技术应用：分类和聚类在各行业、各领域的应用，包括但不限于：银行、金融、保险、交通、通信、市场营销、经济分析、商务智能、知识工程、目标识别、生物信息学、生物统计学、医药和健康科学、信息安全等

投稿要求

详见会议网站：http://202.196.73.187/ccta2007

重要日期

截稿日期： 2007-02-04
录用通知日期： 2007-03-12
论文清样提交截止日期： 2007-03-22

联系方式

通信地址：郑州市大学北路 75 号，郑州大学信息工程学院，王黎明
电话：0371-67763882-2125（王黎明），0371-67783097（郭淑艳），0371-67761542（范明）
传真：0371-67761542

E-mail: ccta2007@zzu.edu.cn