CSDTW：一种时间序列流上的受限动态弯曲距离*

陈树广[1]，李俊奎[2]，陈胜利[1]
（1. 西安财经学院 管理学院，西安 710100；2. 支付宝（中国）网络技术有限公司，杭州 310000）

摘　要：针对 SPRING 提出的使用精确 DTW 距离造成弯曲矩阵中有许多无用的计算数据格的不足，提出一种受限的动态时间弯曲距离 CSDTW。通过限制某时刻弯曲路径的弯曲程度，同时结合 DTW 上的提前终止算法，以减少无用数据格的出现。实验对比表明，CSDTW 能够避免大量冗余数据格的计算，加快流环境下精确 DTW 处理的效率。

关键词：时间序列流；动态弯曲；受限；提前终止
中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1001-3695(2012)08-2939-04
doi:10.3969/j.issn.1001-3695.2012.08.036

CSDTW:constrained dynamic time warping distance in stream context

CHEN Shu-guang[1], LI Jun-kui[2], CHEN Sheng-li[1]
(1. *School of Management, Xi' an University of Finance & Economics, Xi' an* 710100, *China*; 2. *Alipay Network Technology Co., Ltd, Hangzhou* 310000, *China*)

Abstract: Aimed at the disadvantage of existing exact dynamic time warping (DTW), namely many unnecessary computations on data cells in the warping matrix. This paper proposed a constrained dynamic time warping distance (CSDTW) in stream context. CSDTW combined the global constraints with the early abandon technique to reduce the computations, which confined the warping path in a limited scope. The comparative experiments show, CSDTW avoids a large number of computations and thus improves the efficiency of exact DTW processing under stream context.

Key words: time series stream; dynamic time warping; constrained; early abandon

时间序列流(time series stream)是实际科研生产中经常出现的一种数据类型，由于其在金融分析、传感器网络以及医疗服务等领域具有广泛应用[1]，对时间序列流的处理和挖掘激发了越来越多的研究热情，目前正成为一个研究热点[2]。

动态时间弯曲距离(dynamic time warping，DTW)允许时间序列中部分数据沿时间轴延伸，在时间序列相似性搜索中取得了巨大成功。人们在静态时间序列的动态弯曲上已经做了大量的工作，其中有文献[3]提出了 LB_Kim，文献[4]提出了 LB_Yi 的 DTW 距离下界函数；随后 LB_Keogh[5~7] 被提出用于取代这两种下界函数，它被认为是当前比较好的一种精确 DTW 的近似；文献[8]提出了 LB_PAA 方法，采用分段平均近似 PAA[9] 的方式来计算下界距离。文献[10]将 LB_Keogh 与 APCA[11] 结合起来进行处理；文献[12]中注意到 LB_Keogh 不是一种对称的距离函数，提出了对称的 DTW 下界函数 LB_HUST。文献[13]提出了 FTW 方法，文献[14]提出了 FastDTW 来快速计算 DTW，但是 FastDTW 可能发生漏查问题。文献[15]分析了 DTW 计算中最小弯曲矩阵的性质，提出了 EA_DTW 技术，提前终止在已确定不是最终候选查询序列上的 DTW 计算。

由于 DTW 普遍被认为是高计算复杂度 $O(mn)$（其中 m、n 分别是两序列的长度），目前在时间序列流上还较少应用 DTW。文献[16]提出了 Stream-DTW 的方法，用于流上的时间序列弯曲计算，但是这仍然是一种近似的方法，采用了 DTW-like 的方式来定义 DTW。我国一些学者也对时间序列流进行了研究，得到了一些不错的结果，如文献[17~21]。文献[22]首次将精确的 DTW 计算引入到时间序列流的监控中，提出了 SPRING 方法，该方法修改数据格的结构，加入一个起始标志位来指示该数据格所在子段的起始位置，以递进的方式对时间序列流进行监控。但是 SPRING 方法在每个时间点都计算新列中各数据格，而其中许多数据格并不会在最终的结果中，造成计算资源的浪费。

本文提出一种受限的 DTW 计算方法 CSDTW(constrained streaming DTW)，该方法引入在静态时间序列 DTW 计算中常用的全局限制，缩小弯曲路径的弯曲程度，同时结合提前终止技术，减少冗余数据格的计算。

1　SPRING 方法及其问题

1.1　相关定义

定义 1　时间序列流。它是一串按照时间顺序无限到达的元组 $T=t_1,t_2,\cdots,t_i$，其中 $t_i(i\geqslant 1)$ 是流中标志（采用显式时间戳或隐式到达时间点标志）为 i 的数据。

为简便处理，本文统一采用隐式到达时间点作为流中数据标志，但显式时间戳不影响讨论。

定义 2　时间序列流序列查询。给定时间序列流 T 和查询序列 $q=q_1,q_2,\cdots,q_m(m>1)$、距离度量 D，以及距离差异阈值 $\varepsilon(\varepsilon>0)$，找出在 T 中的子序列 $T_q=t_b,\cdots,t_e(1\leqslant b<e)$，满

收稿日期：2011-12-31；**修回日期**：2012-02-21　　**基金项目**：国家自然科学基金资助项目(10802061)

作者简介：陈树广(1972-)，男，硕士，主要研究方向为数据可视化、决策分析(xacsg@126.com)；李俊奎(1983-)，男，博士，主要研究方向为数据挖掘；陈胜利(1974-)，男，博士，主要研究方向为数据挖掘.

足 $D(T_q,q)\leqslant\varepsilon$。

本文讨论的是精确 DTW 在时间序列流中的计算，所以距离度量 D 为 DTW。

定义 3 DTW 距离。给定时间序列 $U=u_1,u_2,\cdots,u_m(m>1)$ 和 $V=v_1,v_2,\cdots,v_n(n>1)$，$U$、$V$ 之间的 DTW 距离可以计算为

$$DTW(U,V)=d(m,n) \tag{1}$$

$$d(i,j)=D_{\text{base}}(u_i,v_j)+\min\{d(i-1,j),d(i-1,j-1),d(i,j-1)\}$$

$$d(0,0)=0,d(0,\infty)=\infty,d(\infty,0)=\infty$$

$$(i=1,2,\cdots,m;\ j=1,2,\cdots,n) \tag{2}$$

其中：$D_{\text{base}}:(\text{int},\text{int})\rightarrow\text{double}$ 是 DTW 计算的基距离，虽然基距离可以有多种定义方式，与文献[5]一致，本文采用 $D_{\text{base}}(u_i,v_j)=(u_i-v_j)^2$；$d(i,j)$ 是弯曲路径上从数据格(1,1)到数据格(i,j)处的距离累计和，如图 1 所示。

	0	3	6	0	6	1
2	4	1	16	4	16	1
5	25	4	1	25	1	16
2	4	1	16	4	16	1
5	25	4	1	25	1	16
3	9	0	9	9	9	4

(a)U,V之间的弯曲矩阵

	0	3	6	0	6	1
2	4	5	21	25	41	42
5	29	8	6	31	26	42
2	33	9	22	10	26	27
5	58	13	10	35	11	27
3	67	13	19	19	20	15

(b)U, V之间的累计弯曲矩阵，数据格中即为$\gamma(i,j)$

图1 $U=\{2,5,2,5,3\}$,$V=\{0,3,6,0,6,1\}$

图 1 中灰色数据格标明了 U、V 之间的最小弯曲路径，U、V 之间的 DTW 距离为 15。

限于篇幅，具体的关于 DTW 计算详细原理和弯曲路径的性质请参见文献[5,15]。

1.2 SPRING 方法

下面简要介绍文献[22]中提出的 SPRING 方法，为后面分析其问题作铺垫。

1）基本思想 SPRING 首次将精确的 DTW 计算引入到时间序列流环境中，其基本数据结构是采用两个与查询序列相同大小的向量，用于记录当前时间点和前一时间点的数据格的状态，记为 A 和 A'。同时记时间序列流为 $U=u_1,u_2,\cdots,u_{t-1},u_t,\cdots$，当前流经的数据为 u_t，查询序列为 $V=v_1,v_2,\cdots,v_n$。

向量中数据格的结构采用的是二元组$\langle d(t,i),s(t,i)\rangle$来表示数据格中的数据。其中：$d(t,i)$为 t 时刻的数据格 i 中数据，$s(t,i)$为 t 时刻的数据格 i 所处的子序列在流中的开始位置（流中某个历史时间点到达的数据，一般认为处理系统有足够的内存空间以缓存流中过去的部分数据）。由于记录了子序列的开始位置，一旦找到匹配的序列末端，即可从流中还原出实际的序列结果。

SPRING 采用的是递进式的计算处理方式，t 时刻根据流经数据和向量 A'_{t-1}来计算向量 A_t，以确定是否有合乎条件的子序列，有则报告，处理完后将 A_t 写入 A'_t，以供后续计算。

2）核心算法描述 下面是 SPRING 核心算法的描述。

Input：ε，U，V。

Output：U 中子序列 u，$D(u,V)\leqslant\varepsilon$。

初始阶段(doInit(i))

$$d(t,i)\leftarrow 0;d'(t,i)\leftarrow 0;$$

递进式格局变化(t 时刻)

$$A_t\equiv\langle d(t,i),s(t,i)\rangle^m \tag{3}$$

$$d_{\text{best}}\leftarrow\min\{d(t,i-1),d(t-1,i),d(t-1,i-1)\} \tag{4}$$

$$d(t,i)\leftarrow D_{\text{base}}(u_t,v_i)+d_{\text{best}} \tag{5}$$

$$s(t,i)\leftarrow\begin{cases}s(t,i-1)\ (d_{\text{best}}=d(t,i-1))\\ s(t-1,i)\ (d_{\text{best}}=d(t-1,i))\\ s(t-1,i-1)\ (d_{\text{best}}=d(t-1,i-1))\end{cases} \tag{6}$$

if $d(t,n)\leqslant\varepsilon$ then

report($d(t,n)$,$s(t,n)$,t);

doInit(t);

$$\text{else } A'\leftarrow A; \tag{7}$$

end if

3）SPRING 的问题 从上述对 SPRING 核心算法的描述中可以看出，SPRING 每次都会计算式(4)～(6)以得到新时刻的向量 A(式(3))进而计算 A'(式(7))，并根据式(1)的原理得到最终的 DTW 距离 $d_{(t,n)}$。如果小于距离差异阈值 ε，则说明找到了子序列 $u_{s(t,n)},\cdots,u_t$。

但是它存在的问题是：

a）每次为确定最终的 DTW 距离 $d_{(t,n)}\leqslant\varepsilon$，全部计算 A 和 A'中的所有数据格，但是在多数情况下 $d_{(t,n)}>\varepsilon$，在进行部分计算后即可确定最终的 $d_{(t,n)}$将超过阈值，这样没有必要每次全部计算 A 和 A'，从而减少计算量。

b）对弯曲路径的弯曲程度未限制，但是在时间序列处理时弯曲程度一般在一定的范围内。

2 CSDTW 方法

本文对 DTW 距离进行限制，提出 CSDTW 方法。CSDTW 方法基于 DTW 的全局限制和提前终止技术。

1）DTW 的全局限制 大量时间序列数据处理研究者在进行 DTW 处理时都引入了全局限制(global constraint)，其中最为常用的限制是 Sakoe-Chiba Band(图 2)和 Itakura Parallelogram(图 3)。Sakoe-Chiba Band 中窗口大小 w 为一个常数，而 Itakura Parallelogram 中则为一个随下标变化的函数。普遍认为引入全局限制可有效减少弯曲空间，提高处理效率。

2）提前终止技术 提前终止技术是在受限查询时采用的一项技术，本文在文献[15]中将其引入到精确的 DTW 计算中，提出 EA_DTW。如果某个数据格中数据超过阈值 ε，则称为溢出。EA_DTW 的基本结论有：

定理 1 溢出传递定理。任意 t 时刻，若 $d(t,i-1)$，$d(t-1,i-1)$，$d(t-1,i)$溢出，则 $d(t,i)$溢出。

定理 2 溢出省略定理。若已知 $d(t,i)$溢出，则可以省去计算 $D_{\text{base}}(u_t,v_i)$。

定理 3 溢出替换定理。若 $d(t,i)$溢出，则将 $d(t,i)$赋值为 $\forall\varepsilon^*(\varepsilon^*>\varepsilon)$，不影响最终查询结果。

定理 4 DTW 提前终止定理。若最小弯曲路径上存在一数据格溢出，则可中止计算 DTW。

定理 5 若弯曲矩阵的一行或一列的数据格均溢出，则可中止计算 DTW，即 $d_{(t,n)}>\varepsilon$。

EA_DTW 的详细说明和定理证明可参见文献[15]。

3）基本思想 CSDTW 的基本思想是在计算向量 A(从而 A')时不是每次都全部计算出 A 中数据格，而是根据 Sakoe-Chiba Band 的限制，确定需要计算的数据格的范围，在计算这些数据格时又同时结合 EA_DTW 的方法，尽快结束不符合要求的子序列匹配过程，跳跃到下一个时间点重新开始计算。

与全序列相似性搜索不同,子序列搜索并不能确定 DTW 计算的起点和终点,因此,在限制弯曲路径时不能只考虑当前起点的子序列计算,还需要考虑后续起点的子序列计算。因此在对向量 A 进行 Sakoe-Chiba Band 限制时,只能限制向量 A 中计算范围的尾端。

4)算法描述

Input:ε, U, V, w^*(窗口大小)

Output:U 中子序列 $u, D(u,V) \leqslant \varepsilon$

初始阶段(doInit(i))

$d(t,i) \leftarrow 0; d'(t,i) \leftarrow 0;$

递进式格局变化(t 时刻)

```
1  earlyabandon←true;// 初始化标志
2  w←w* +(t-i); // 确定范围
3  if (w>n) then w←n; endif
4        A_t ≡ ⟨d(t,i), s(t,i)⟩^⌊w⌋ (i=1,2,…,⌊w⌋)                (8)
5  d_best←min{d(t,i-1), d(t-1,i-1), d(t-1,i)};
6  if (d_best>ε) then
7  d(t,i)←D_MAX;
8  else d_base←D_base(u_t, v_i);
9    if (d_base + d_best>ε) then
10     d(t,i)←D_MAX;
11     else d(t,i)←d_base + d_best;
12 earlyabandon←false;
13    endif
14 endif
15 if (! earlyabandon) then
16 s(t,i)←{ s(t,i-1)   (d_best = d(t,i-1))
          { s(t-1,i)   (d_best = d(t-1,i))
          { s(t-1,i-1) (d_best = d(t-1,i-1))
17 if ⌊w⌋ = n and d(t,n) ≤ε then
18    report(d(t,n), s(t,n), t);
19    doInit(t);
20 else
21    A'←A;
22    endif
23 else // 进行提前终止
24    doInit(t);
25 endif
```

5)方法分析　CSDTW 方法结合了全局限制和提前终止技术。处理时,随着时间的推移,窗口大小亦改变(行 2、3)。某 t 时刻,按式(8)计算向量 A 中落入窗口内的数据格范围。

在计算数据格时,进行提前终止判断和处理。根据定理 1 和 2,首先判断数据格是否溢出,溢出则用系统中最大浮点数(D_MAX)填充,省略计算基距离(行 6、7);否则计算基距离,并判断此时数据格是否可能溢出,若溢出则同样用最大浮点数填充。已经确定数据格不会溢出时,按照普通 DTW 数据格的计算方法进行计算。

计算完毕向量 A 中落入窗口的所有数据格后,如果发现均溢出,则根据定理 4 和 5,有理由相信 DTW 可以提前终止,故进行提前终止操作(行 23 ~ 25)。

CSDTW 计算步骤示意如图 4 所示(图中结合了 Sakoe-Chiba Band 限制和 EA、DTW 技术)。

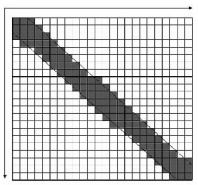

图2　Sakoe-Chiba Band限制

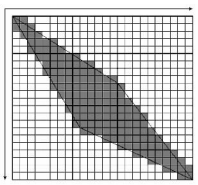

图3　Itakura Parallelogram限制

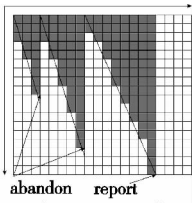

图4　CSDTW的计算示意图

由以上对 CSDTW 算法的分析,有如下关于 CSDTW 复杂度的结论:

定理 6　CSDTW 在任意时刻都需要 $O(n)$ 的空间和 $O(w)$ $(w \leqslant n)$ 的时间。

虽然 CSDTW 在复杂度上较 SPRING 没有明显优势(SPRING 需要 $O(n)$ 空间和 $O(n)$ 时间),但是在第 3 章可以看出,采用 CSDTW 处理可以有效地减少冗余数据格的计算,提高处理效率。

3　实验研究

3.1　实验准备

本节对 CSDTW 进行实验研究。CSDTW 将全局限制和提前结束结合起来,用于减少在计算向量 A 中的冗余数据格。

使用 C ++ 分别实现了 SPRING 和 CSDTW 算法。实验环境如表 1 所示。

表 1　实验环境配置

配置项目	项目值
CPU	Intel Pentium 3-866
操作系统	Ubuntu Linux 4.1.1.13(内核版本:2.6.2)
RAM	256 MB
磁盘	40 GB
编译环境	GNU g + + 4.1.2.20060928

为了客观地评价 CSDTW 方法的效果,首先必须给出实验的评价标准。主要考察 CSDTW 的减少冗余数据格计算的情况,称为跳跃率,定义为

$$\text{跳跃率} = \frac{\text{总数据格数} - \text{计算的数据格数}}{\text{总数据格数}} \times 100\% \qquad (9)$$

跳跃率越大,则跳过的冗余数据格数越多,算法越有效。SPRING 计算的数据格数即总数据格数,故 SPRING 的跳跃率为 0。同时记录计算数据格的总时间开销。

实验数据流来自使用程序产生的随机数发生器 GenStream,它能够在固定时间间隔内产生满足正态、普通和指数分布的在[0,1]间的随机数。由于数据流的不可复制性,故在相同参数下进行多次实验,最终实验结果取多次实验的平均值。实验中 w^* 取 1,查询序列长度为 50。

3.2　实验结果及分析

实验 1　跳跃率实验。实验测试在相同的查询序列、不同的数据流在不同的查询阈值 ε 时的跳跃率。其结果分别如图 5 ~ 7 所示。实验结果表明,CSDTW 方法在各阈值时都能省略冗余的数据格,在阈值较小时更加明显,其中在阈值为 5 时的指数随机数数据流下,取得了多达 90% 的跳跃率,因为阈值小时提前终止的条件更易达到。

实验 2　处理时间实验。为统一结果显示,记录选取在实验 1 中跳跃率最接近 50% 时的查询阈值下(分别是图 5 中的 25,图 6 中的 12,图 7 中的 22)运行 80 s 过程中的处理时间,结

果如图 8 ~ 10 所示。实验结果表明,与 SPRING 方法相比,同等条件下,CSDTW 方法处理的时间更少,这是因为由于减少了向量中数据格的计算,节省了在冗余数据格上的计算时间。

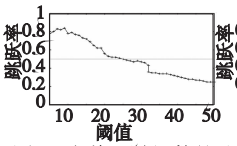

图5 在普通随机数的数据流上的跳跃率

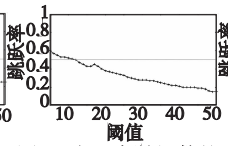

图6 在正态随机数的数据流上的跳跃率

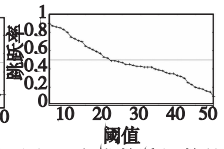

图7 在指数随机数的数据流上的跳跃率

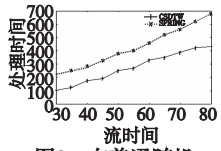

图8 在普通随机数数据流上的处理时间结果

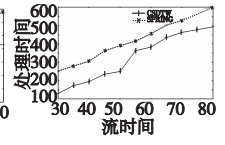

图9 在正态随机数数据流上的处理时间结果

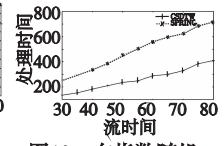

图10 在指数随机数数据流上的处理时间结果

4 结束语

本文提出了 CSDTW 方法用于时间序列流环境下的精确 DTW 计算,该方法在序列上施加时间序列全局限制,从而不需要每次全部计算更新向量。同时在计算更新向量时结合提前终止技术,尽量避免在冗余的数据格/不匹配的子序列上进行计算。实验结果表明,CSDTW 可以缩小匹配范围,与需要全部更新向量的 SPRING 方法相比,节省了计算资源,提高了处理效率。

但在进一步研究中,下列几个问题值得关注:a)在时间序列处理过程中的相关参数设置处理。本文的时间序列处理算法都依赖于用户指定算法运行的部分参数,其效果并不能从根本上改变参数设置的重要性。如何帮助用户指定相关参数将是未来的一项重要工作,因为这往往可以帮助用户根据实际的环境来发挥算法的效果。b)时间序列的前期处理过程。本文是在特定数据集上进行的研究工作,这些数据集一般都已经过了仔细的前期处理过程,一旦实际应用则需要根据实际的数据特点研究如何进行前期处理的过程。c)相似比较的距离度量。在关注比较距离度量的同时,还需要关注距离度量在其他时间序列挖掘任务中(如聚类和分类)的应用,将相似比较与其他挖掘任务隔离开来,将大大限制距离度量的应用范围。d)相似性问题的实际应用和扩展。

参考文献:

[1] MUTHUKRISHNAN S. Data streams algorithms and applications [C]//Proc of the 14th Annual ACM-SLAM Symposium on Discrete Algorithms. 2003:413-413.

[2] BABCOCK B, BAHU S, DATER M, *et al*. Models and issues in data stream systems[C]// Proc of the 21st ACM Symposium on Principles of Database Systems. 2002:1-16.

[3] KIM S, PARK S, CHU W W. An index-based approach for similarity search supporting time warping in large sequence databases[C]// Proc of the 17th International Conference on Data Engineering. 2001: 607-614.

[4] YI B K, JAGADISH H V, FALORTSOS C. Efficient retrieval of similar time sequences under time warping[C]//Proc of the 14th International Conference on Data Engineering. 1998:451-457.

[5] KEOGH E. Exact indexing of dynamic time warping[C]//Proc of the 28th International Conference on Very Large Data Bases. 2002:406-417.

[6] KEOGH E, LI Wei, XI Xiao-peng, *et al*. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures[C]// Proc of the 32nd Very Large Databases Conference. 2006:223-227.

[7] LI Wei, KEOGH E, Van HERLE H, *et al*. Atomic wedgie: efficient query filtering for streaming time series[C]// Proc of the 5th IEEE International Conference on Data Mining. 2005: 490-497.

[8] ZHU Yun-yue. High performance data mining in time series: techniques and case studies[D]. New York: New York University, 2004.

[9] KEOGH E, PAZZANI M. A simple dimensionality reduction technique for fast similarity search in large time series databases[C]// Proc of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2000:122-133.

[10] SHOU Yu-tao, MAMOULIS N, CHEUNG D W. Fast and exact warping of time series using adaptive segmental approximations[J]. Machine Learning, 2005, 7(3):231-267.

[11] KEOGH E J, CHAKRABARTI K, MEHROTRA S, *et al*. Locally adaptive dimensionality reduction for indexing large time series databa ses [C]//Proc of ACM SIGMOD Conference. 2001: 151-162.

[12] LI Jun-kui, WANG Yuan-zhen, LI Xin-ping. LB_HUST: a symmetrical boundary distance for clustering time series[C]//Proc of the 9th International Conference on Information Technology. 2006:245-249.

[13] SAKURAI Y, YOSHIKAWA M, FALOUTSOS C. FTW: fast similarity search under the time warping[C]//Proc of ACM SIGMOD Sympo sium on Principles of Database Systems. 2005:326-337.

[14] SALVADOR S, CHAN P. FastDTW: toward accurate dynamic time warping in linear time and space[C]//Proc of the 3rd Workshop on Mining Temporal and Sequential Data. 2004:456-460.

[15] LI Jun-kui, WANG Yuan-zhen. EA_DTW: early abandon to accele rate exactly warping matching of time series[C]//Proc of International Conference on Intelligent Systems and Knowledge Engineering. 2007:231-235.

[16] CAPITANI P, CIACCIA P. Warping the time on data streams [J]. Data Knowledge and Engineering, 2006, 8(3):35-39.

[17] 李晓光,宋宝燕,张昕. 基于滑动多窗口的时间序列流趋势变化检测[J]. 电子学报,2010,38(2):34-37.

[18] 李晓光,宋宝燕,于戈,等. 基于小波的时间序列流伪周期检测方法[J]. 软件学报,2010,21(9):23-28.

[19] 陈华辉,施伯乐. 时间序列流的分层段模型[J]. 小型微型计算机系统,2009,30(4):56-61.

[20] 张洪祥,毛志忠. 基于时间序列的模糊聚类与规则提取信用评价模型[J]. 东北大学学报:自然科学版,2010,31(4):234-238.

[21] 徐雪松. 时间序列不确定数据流中异常数据检测方法[J]. 电子设计工程,2011,19(4):19-21.

[22] SAKURAI Y, FALOUTSOS C, YAMAURO M. Stream monitoring under the time warping distance[C]//Proc of International Conference on Data Engineering. 2007:1046-1055.

[23] KO M H, WEST G, VENKATESH S, *et al*. Using dynamic time warping for online temporal fusion in multisensor systems [J]. Information Fusion, 2006, 9(3):370-388.