

基于统计特征的时序数据符号化算法

钟清流 蔡自兴

(中南大学信息科学与工程学院智能系统与智能软件研究所 长沙 410083)

摘 要 为克服 SAX(符号聚合近似)算法对时序信息描述不完整的缺陷,提出基于统计特征的时序数据符号化算法,与 SAX 不同的是,该算法将时序符号看作矢量,而各时序子段的均值和方差则分别作为描述其平均值及发散程度的分量.由于该算法能够比 SAX 提供更多的描述信息,因而在时序数据挖掘应用中能够获得比 SAX 更精确的结果.大量的实验也证实了它的出色表现.

关键词 时序数据挖掘;符号化表示;符号聚合近似

中图法分类号 TP18

The Symbolic Algorithm for Time Series Data Based on Statistic Feature

ZHONG Qing-Liu CAI Zi-Xing

(Center of Intelligent System and Software, School of Information Science and Engineering, Central South University, Changsha 410083)

Abstract A new symbolic algorithm for Time Series Data Based on Statistic Feature is put forward in order to surmount the bugs with which SAX (Symbolic Aggregate Approximation) Algorithm can not describe time series information fully. This algorithm, differing from the SAX, considered the symbolic as vector, and Mean and variance from each subsequence were regarded as components by which its mean value and radiation degree are described respectively. Since it could provide more information described time series than SAX do, more accuracy result could be get when it is applied to time series data-mining. Its excellent behave. have been proved by a lot of experiments.

Keywords time series analysis; symbolic representation; symbolic aggregate approximation(SAX)

1 引 言

符号时序分析的关键是如何有效提取嵌入到时序数据中的相关信息,这涉及寻找不同原始时序数据的最佳分割方法^[1-2].在目前有文献报道的时序表示方法中,较著名的包括离散付氏变换(DFT)^[3]、离散小波变换(DWT)^[4]、分段聚合近似(PAA)^[5]、自适应分段常数近似(APCA)^[6]、基于符号假近邻的方法(SFNN)^[7]和符号聚合近似(SAX)方法^[8].

其中,大多数符号表示方法都存在以下致命缺陷:(1)时序数据是典型高维数据,时序数据降维是其高度期望的属性.但大多数符号时序无法降维;(2)定义的符号时序无法进行距离计算,也就无法进行相似比较,因而缺乏实用价值.而 Lin 和 Keogh 在 2003 年提出的新型的符号化表示方法——符号聚合近似,以下简称为 SAX 方法^[8],首次有效地克服了上述致命缺陷并在许多应用中性能都优于或至少不低于其它的方法^[9],因而受到了广泛关注,成为倍受推宠的时序符号化算法之一^[9].它的主要优点

是,比其它符号算法更简便、高效;在符号化过程中实现了减维降噪,保证在符号空间计算出的两个符号序列的距离满足实际的两个时间序列的距离的下界要求,即不会出现漏报^[8].然而,它主要适合于遵循高斯分布且在有限方差范围内有较高分布密度的时序数据.由于 SAX 在作时序数据的符号化转换时,只是利用了各相应时间子段数据的均值信息,因而不可避免地存在一些弱点.由于在符号时序分析中 SAX 模型所提供的描述信息不够充分,难以表现时序数据更细微的特征等.在符号时序数据模式识别(异常检测或相似查询)中的精度不够理想,有时甚至可能导致错误结果.

例如将图 1(a)中时序子段转化成一个符号时,尽管时序段 1 及时序段 2 数据方差完全不相同,但由于它们的平均值相同, SAX 算法会把它们转化成相同的符号.而在做时序相似分析时,这两个在图中看来完全不同的时序段将会被判定为相同.显然,这

是一个明显的缺陷.这种算法的直接后果是符号化过程中丢失了重要的数据方差(均方差)信息,包括一些重要的极值信息.因而不适合于那些需要更精确分析的场合.图 1(b)是用 SAX 将某一时序转换为符号时数据与符号的对应映射图.此图表明,尽管位于上下边界区的数据都分别划分为相应的最大、最小字符表示,但由于这种划分过分粗糙,实际上造成隐性的信息丢失.这将导致一些时序分析任务中的较大失误.图 1(c)显示的是将这一 SAX 符号化结果用于相似查询分析的情况,如果查询串恰好位于边界区,则无论其幅度多大都会被转化为最大或最小字符,尽管图 1(c)中第 3 个波峰幅度比第 1 个(查询串)高了几倍,但由于同属于边界区字符,它们都会被转化为同样的字符(在计算时被当作相同的波幅对待),因此,在相似查询中出现错误匹配也就在所难免了.

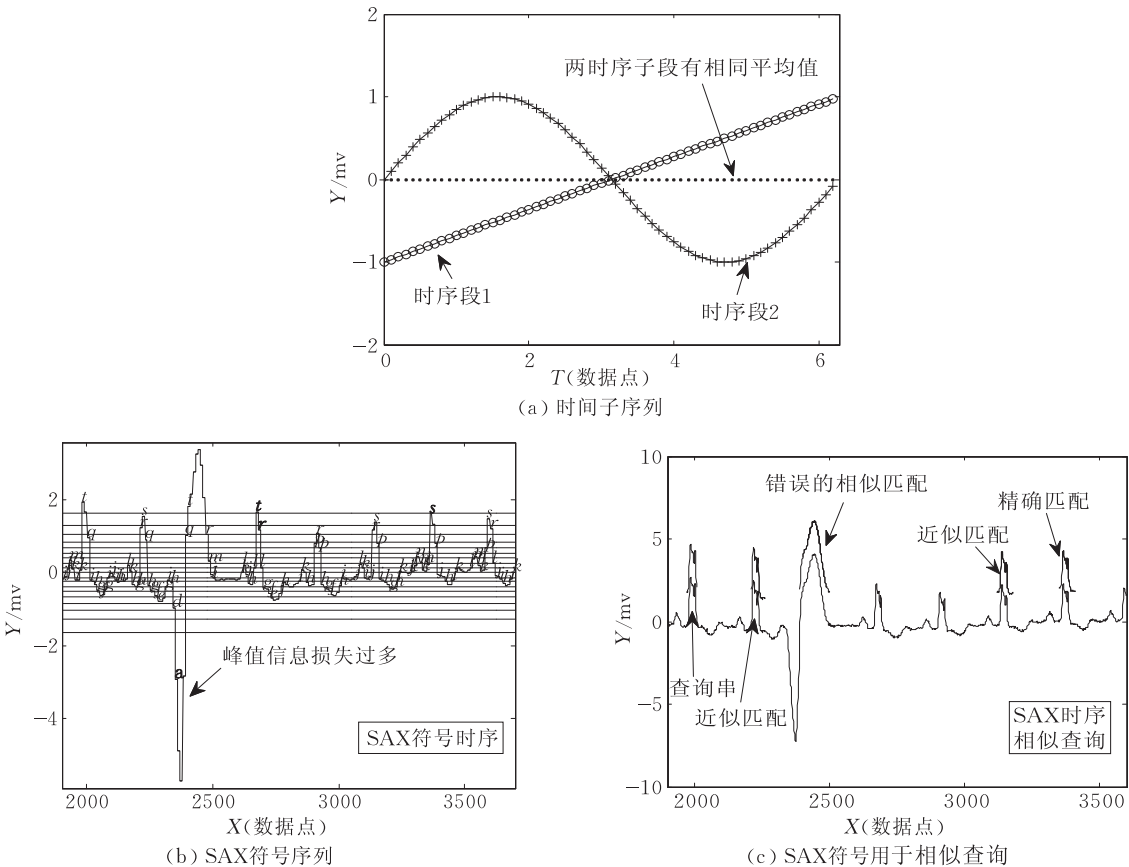


图 1 SAX 符号时序的缺陷

SAX 算法只能近似描述时序数据的大致特征的缺陷限制了它的应用范围.在时序数据分析中,数据的方差信息对分析结果至关重要,在很多情况下甚至会对时序分析的结果产生决定性影响.例如,对

前面所说的图 1(a),只需进一步比较二者的方差,立即就会推翻前面的错误结论.基于这个认识,本文提出的 SFVS 基于(统计特征矢量符号化)算法作为时序数据符号化算法探新的一种尝试,其基本思路

是:利用描述时序数据统计特征的均值与方差分别作为描述其“平均值”特征及“发散程度”特征的分量,而时序符号则看作是由这些分量构成的特征矢量.这样,在实施时序数据符号化过程中,每个时序子段将转化成一个有二个分量的符号矢量.其中,各时序子段的均值作为刻画其“平均值”特征的分量,而相应时序子段的方差则作为刻画发散程度特征的另一个分量.一个时序符号的总体特征则是该二分量的矢量和.由于二者从不同视角(维度)来描述该时序子段的特征,因而能够在不增加符号数的前提下比 SAX 方法提供更为全面的描述信息.这有利于在时序模式识别的应用中实现更精确的分析.仿真实验表明,本算法在时序相似搜索、异常检测应用中能够得到比 SAX 更好结果.本文第 2 节讨论时序数据的符号化的预处理;第 3 节给出 SFVS(统计特征矢量符号化)算法;第 4 节给出仿真实验结果;第 5 节为结论.

2 时序数据符号化的预处理

2.1 概述

本文要解决的问题是,将一段长度为 n 的任意时序数据 $X = x_1, x_2, \dots, x_n$ 转化成长度为 N 的用字符集 $A_n = \{a, b, c, \dots\}$ 表示的符号数据 $\hat{X} = \hat{x}_1, \hat{x}_2, \dots, \hat{x}_N$. 这里, $N \ll n$. 在转化过程中应当尽可能地减少信息丢失.为此,本文采用如下方案:首先计算待转化时序数据的最大压缩比 w (又叫最大分段长度,即每个符号所代表的时序数据点数);据此再用分段聚合近似(PAA^[10])方法计算各时段数据的近似表示,并计算各时段数据的方差,进而确定最佳字符集的分量数 A ,最终将其转化成符号集.与 SAX 方法不同的是,本文所用的符号化方法中每个符号包含了 2 个分量.但最终得到的符号数将与 SAX 方法相同.

2.2 符号化前期的数据预处理

时序数据符号化目的是要在误差允许范围内通过时序数据的减维来降低计算代价.更具体地说是通过减维操作将维数较大的连续长时序数据压缩为维数较小的离散短符号序列,同时又保证经压缩后的符号时序保留原始时序的重要特征信息,以便在各种时序分析任务中能够以较小的计算代价达到与原始时序分析尽可能相同的效果.时序数据符号化的关键内容之一是寻找最佳压缩比.

压缩比相当于一个符号所能代表的时序数据点

数,显然,压缩比越高,一个符号所能代表的时序数据点数越多,当然对所代表的那段时序描述也就越粗糙(信息丢失越多),计算代价越小.然而,压缩比越低,一个符号所能代表的时序数据点数越少,对所代表的那段时序描述也就越精细(信息丢失越少),同时也带来了计算代价变大的问题.因而如何确定最大压缩比,是影响时序数据符号化的质量和成败的核心问题之一.不幸的是,在目前所报道的有关时序数据符号化的文献中,尚未发现相关的研究和探讨,大多数文献确定压缩比的方法是凭经验或实验地确定^[11].

本文在研究中发现压缩比与时序数据的变化频率呈相反变化关系,同时也与允许误差相关.但确切的关系目前难以分析地描述,下面是本文根据实验分析后提出并采用的经验公式:

$$W = \sqrt{2k\theta/F}, F = \sum_{i=1}^{n-1} |x_{i+1,j} - x_{i,j}| / (n-1) \quad (1)$$

这里, w 为最大压缩比, F 为时序数据变化频率. θ 为允许误差, k 为标准常数, n 为数据点总数.本文建议 θ 取值为 0.05, k 取值为 100. 根据这个思路可按数据特点经验地确定最大压缩比.

为了使符号化时序数据能够适用于一般情况,在降维前还须将待转化连续时序数据转化为零均值,均方差为 1 的规范化数据.然后用 PAA^[10] 方法将标准化时序数据 $X = x_1, x_2, \dots, x_n$ 按适当的压缩比 w 降维,生成一个 $N = n/w$ ($N \ll n$) 维空间向量 $\bar{X} = \bar{x}_1, \bar{x}_2, \dots, \bar{x}_N$. 其中, \bar{X} 的第 i 个元素可由下式计算:

$$\bar{X}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} X_j \quad (2)$$

另一个须要同时预处理的量是相应(第 i 个)时序子段的方差,考虑到 $w = n/N$ 有

$$S_i = \frac{1}{w} \sum_{k=1}^w (x_{ik} - \bar{X}_i)^2 \quad (3)$$

3 SFVS(统计特征矢量符号化)算法

3.1 算法描述

针对 SAX 符号时序算法容易丢失极值信息的缺陷, Lkhagva 等人于 2006 年曾经提出一种“改进方案”^[9] 称为 Extended_SAX(下面缩写为 EXT_SAX). 其基本思路是:分别引入各字符对应时序子段的极大极小值作为新的字符表示,从而将原来的每时序子段一个 SAX 字符改为 3 个字符,本文研究并验证

过这种方案,发现这种方案效果有限.原因很简单,因为 SAX 在符号化过程中,若压缩比为 w ,则通常是将 w 个时序数据点用一个代表其平均值的符号来表示.其中的极大值、极小值只是这 w 个点中的单个特殊点,只占其中的 $1/w$.若按“改进方案”,将极大值、极小值单独作为字符,则原来的每时序子段将出现 3 个字符,即每个字符将代表 $w/3$ 个时序数据点,这意味着将极大值、极小值的代表性人为地扩大为 $w/3$ 倍.这不但会造成人为的信息失真,而且由于符号数变为原来的 3 倍使计算代价也相应地大大增加(有关 EXT_SAX 与 SAX 符号运算代价的比较实验请见表 1).因此本文通过具体地实验验证和理论分析后,认为文献[9]采用增加符号来描述极值信息的方案有待商榷.但它利用相关信息来改善 SAX 信息丢失问题的思路却给我们带来了灵感.

表 1 EXP_SAX 与 SAX 运行时间比较

数据名	数据集长	算法运行时间/s		数据源
		EXP_SAX	SAX	
1 chfdb_chf01_275	3751	0.0305	0.009	ECG
2 chfdb_chf13_45590	3750	0.0337	0.0094	ECG
3 ECG106_test	1216	0.0307	0.01	ECG
4 JENKINS8	100	0.0025	0.0008	TSDL
5 stdb_308_0	5400	0.1527	0.0482	ECG
6 xmitdb_x108	5400	0.1761	0.0476	ECG
7 beefconvert	9098	0.2554	0.0811	UCR
8 Beeftrain	30	0.00028	9.00E-05	UCR
9 synthetic_control	600	0.0165	0.0049	UCI
10 synthetic_data1	10000	0.2503	0.0894	UCI

考虑到时序数据大多可以看作是一种随机系列,时序分析实质上是对所观测到的时序系列进行统计分析,而时序数据的最基本统计特征可以由其均值与方差来确定.因此当人们将时序数据转化为符号表示时,自然应当充分利用这些统计特征信息.本文提出的 SFVS 算法的基本要点是:利用描述时序数据统计特征的均值与方差(或均方差—注:本文采用方差)分别作为描述其平均值及发散程度的分量,而时序符号则是由这些分量构成的矢量.这样,在实施时序数据符号化过程中,每个时序子段将转化成一个有二个分量的符号矢量.其中,各时序子段的均值作为刻划其平均值特征的分量,而相应时序子段的方差则作为刻划数据对平均值 \bar{X} 的偏离程度(显然也包括了极值的影响)特征的分量.而一个时序符号的总体特征则是该二分量的矢量和.由于二者从不同视角(维度)来描述该时序子段的特征,因而能够在不增加符号数的前提下比 SAX 方法提供更为全面的描述信息.相应的数学描述为

$$\hat{x}_{ik}=A_{i1}, \text{ iff } C_{j-1} \leq \bar{X}_i = \frac{1}{w} \sum_{k=1}^w x_{ik} < C_j, \quad k=1 \tag{4}$$

$$\hat{x}_{ik}=A_{i2}, \text{ iff } C_{j-1} \leq S_i = \frac{1}{w} \sum_{k=1}^w (x_{ik} - \bar{X}_i)^2 < C_j, \quad k=2 \tag{5}$$

其中, C_{i-1}, C_i 分别代表第 i 个字符划分区间的上限与下限.

于是可按式(4)、式(5)将落入不同的划分区间的 \bar{X}_i, S_i 值转化为相应的符号分量.这里, \bar{X}_i 为第 i 个时序子段的均值, S_i 为相应时序子段方差 s :

$$\hat{X}_i = \hat{x}_{i1} \cdot i + \hat{x}_{i2} \cdot j = A_{i1} \cdot i + A_{i2} \cdot j \tag{6}$$

式中, \hat{X}_i 就是代表第 i 个时序子段的符号矢量.

3.2 时序数据的符号化

完成时序数据的预处理后,就可根据字符集和数据分布得到描述各个字符所代表数据区间的划分点.进而将已经降维的数据离散成符号化数据,通常,若已经确定最佳字符集规模为 k (k 值的大小是根据描述精度要求自己确定的,精度越高,则 k 值取得越大,一般 k 取 3~20 范围内的值),则一个有 k 个字符组成的字符集需要有 $k-1$ 个划分点.因此划分点集可表为

$$C_l = \{C_1, C_2, \dots, C_{k-1}\} \tag{7}$$

一旦确定了划分点集,就可以实现从预处理数据到符号化时序的转换.而划分点集 C_i 是通过将整个正态分布区间划分成 k 个等概率区间的方式确定,并由划分点集生成相应的划分点查找表^[8].若用 A_i 来表示字符集 A_n 中的第 i 个字符,则从数据近似到符号近似的映射,可根据预处理数据所落入的对应符号划分区间分别由式(4)、(5)确定.

其划分规则为:将所有小于 C_1 的预处理时序数据映射为符号 $A_1 = A_{\min}$,同理,也可将所有大于 C_1 而小于 C_2 区间的相应数据映射为符号 A_2 ,而将所有大于 C_{k-1} 区间的数据映射为符号 $A_n = A_{\max}$.若采用英语字符来表示,则可将每个预处理时序序列转化为相应的符号集,进而按式(6)生成类似于 $\hat{X} = \hat{X}_1, \hat{X}_2, \dots, \hat{X}_k = (a_1 i + a_2 j), (b_1 i + b_2 j), \dots$ 这样的符号矢量集,从而完成时序数据符号化的全部过程.

为了对符号化过程有一个更直观具体的认识,下面不妨以一个实例加以说明:如图 2 所示,若欲将一个长为 $n=128$ 的一维原始时序转化为一个用字符集 $k=3$ 表示的符号集,则可按高斯分布生成的两

个划分点 C_1, C_2 将全部数据划分到 3 个等概率区间,假设确定的压缩比为 $w=16$,则可将所有小于划分点 C_1 的预处理数据表示为符号 a ,而所有大于划分点 C_1 同时又小于划分点 C_2 的数据表示为符号 b ,最后,将所有大于划分点 C_2 的数据表示为符号 c . 于是,按此方法,图中的原始时序数据 ($n=128$) 最终被转化为只有 8 个符号的 ($N=n/w=8$) 符号集 $accabcb$. 对于多维时序,可用类似方法将多维时序矩阵各维同步转化成相应的符号矩阵.

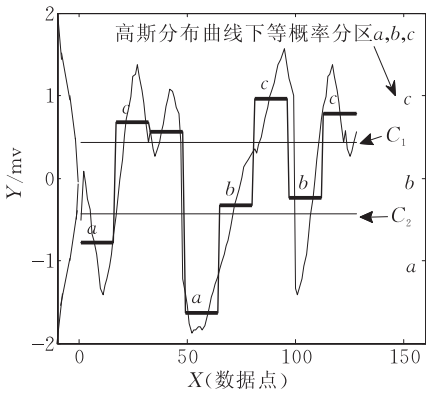


图 2 时序数据符号化示意图

3.3 SFVS 算法实现要点

- 1. 用 PAA 方式按式(1)对时序数据降维;
- 2. 计算各时序子段的数值平均及方差两个数值分量;
- 3. 按式(4)、(5)分别将两数值分量转化为符号分量;
- 4. 按式(6),二者的合成可构成符号矢量;
- 5. 重复以上步骤直到完成全部数据的符号化.

3.4 算法相关的符号距离计算

由于从数据空间向符号空间的映射操作实际上存在着明显的对应关系,因而可以根据这些对应关系实现符号空间的距离计算及数据空间的欧式距离计算. 回顾我们的符号化操作过程,第一步是采用 PAA 减维,进而求方差,将原始数据转化成了预处理数据,两个原始时序 A, B 的欧氏距离可表为

$$D(A, B) = \sqrt{\sum_d \sum_{i=1}^n (A_{id} - B_{id})^2} \tag{8}$$

式中的 d 代表数据的维数,相应的预处理时序 A', B' 的距离则为

$$DR(A', B') = \sqrt{\frac{n}{N}} \sqrt{\sum_d \sum_{i=1}^N (A'_{id} - B'_{id})^2} \tag{9}$$

可以证明,两个预处理时序 A', B' 的距离与原始时序的欧式距离之间存在下面关系^[13]:

$$D(A, B) \geq DR(A', B') \tag{10}$$

与文献[8]类似,当我们进一步将预处理数据转

化成符号数据后:两个符号时序 \hat{A}, \hat{B} 间的距离可由下式计算:

$$dist(\hat{A}_i, \hat{B}_j) = \sqrt{\sum_d (\hat{A}_{id} - \hat{B}_{jd})^2} \tag{11}$$

考虑到本文其中的符号包含两个分量,可得出下面的式子:

$$\begin{aligned} dist(\hat{X}_i, \hat{X}_j) &= \sqrt{\sum_{d=1}^2 (\hat{X}_{id} - \hat{X}_{jd})^2} \\ &= \sqrt{(A_{i1} - A_{j1})^2 + (A_{i2} - A_{j2})^2} \end{aligned} \tag{12}$$

由此可进一步计算它所代表相应的原始时序的最小距离为

$$MinDist(\hat{X}_i, \hat{X}_j) = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^N (dist(\hat{X}_i, \hat{X}_j))^2} \tag{13}$$

同样可以证明预处理时序与符号时序距离间存在下面关系^[14]:

$$DR(A', B') \geq MinDist(\hat{X}_i, \hat{X}_j) \tag{14}$$

综合考虑式(10)及式(14)可得

$$D(A, B) \geq DR(A', B') \geq MinDist(\hat{X}_i, \hat{X}_j) \tag{15}$$

即按 SFVS 符号算得的符号距离是其时序数据相应欧式距离的下限.

3.5 SFVS 算法特点及与 SAX 的比较

在计算时间代价方面,与 SAX 方法类似,SFVS 方法的计算代价主要包括符号化过程的代价和符号距离计算的代价. 符号化过程的代价与符号化过程的所有环节相关,由于符号化过程包括数据预处理(式(2)、(3))、数据划分(式(4)、(5))、数据符号转换等众多环节,很难得出一个精确的分析结论,但可以通过对下面表 2 中典型时序数据的符号化过程 20 次实验平均结果对比来得出大致结果.

表 2 符号化过程时间复杂度比较实验

数据集名 序号	数据集长 详见表 1	总运行时间/s		预处理部分时间/s	
		SFVS	SAX	SFVS	SAX
1	3751	0.0034	0.0032	1.90E-04	3.90E-05
3	1216	0.00131	0.0013	1.20E-04	3.10E-05
5	5400	0.0036	0.0034	2.10E-04	4.10E-05
10	10000	0.0049	0.0046	2.20E-04	4.50E-05

表 2 的第 5,6 两列记录的是 SFVS 及 SAX 两种方法在符号化过程中的数据预处理环节(对应于式(2)、(3))的时间代价比较,显然,此环节中的 SFVS 方法的计算代价较 SAX 方法大约几倍,然

而,由于预处理环节的代价在整个符号化过程中所占比例微不足道,因此,两种方法在符号化过程中的时间复杂度相差不大,具体差异请参见表 2 中的第 3,4 列.

下面来比较两种方法应用最多的符号距离计算代价. SFVS 与 SAX 的符号距离计算可分别用式(12)和式(12)根式中除去第 2 项构成的式子表示,两种符号距离计算式之比可表为

$$\begin{aligned} MinDist(\hat{X}_i, \hat{X}_j) / MinDist(A_{i1}, A_{j1}) = \\ \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^N ((A_{i1} - A_{j1})^2 + (A_{i2} - A_{j2})^2)} / \\ \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^N (A_{i1} - A_{j1})^2} \end{aligned} \tag{16}$$

显然,除了作为分子的根式内部比分母根式多出一项外,其余的计算形式是相同的,这意味着 SFVS 与 SAX 符号距离计算的时间复杂度是同一数量级的,只有渐进常数的差异. 考虑到分子根式中第二项与第一项的计算代价相同,因此,两种符号距离计算的时间复杂度之比大致为: $1 < T_{SFVS}(n) / T_{SAX}(n) \leq \sqrt{2} = 1.414$. 即与 SAX 相比, SFVS 符号距离计算增加的计算代价不大于 50%. 后面的实验对此做了具体的比较(详见表 3).

4 仿真实验

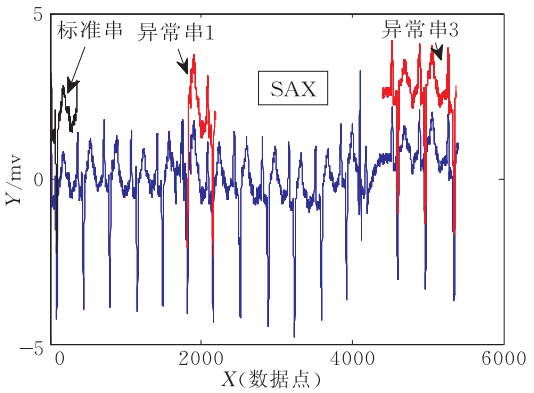
为了检验本文算法的有效性. 刻意采用表 1 中所示各种不同来源及不同长度的时序数据集. 表 1 中也附带记录了文中提到的 EXP_SAX 与 SAX 运行时间代价比较实验. 表中的数据是两种算法用于检测运算 20 次实验平均结果. 实验表明, EXP_SAX 的计算代价约为 SAX 的 3 倍左右.

下面的内容是分别用本文提出的 SFVS 算法与 SAX 方法将各时序数据集转化成相应的符号时序数据集做的对比实验. 其大致方法是: 分别将被查询序列子串 T_i 与待查询时序数据集 T 转化成相应的符号时序,再用改进的 Brute_Force^[15] 算法进行相应的时序分析实验,对比实验的结果如下.

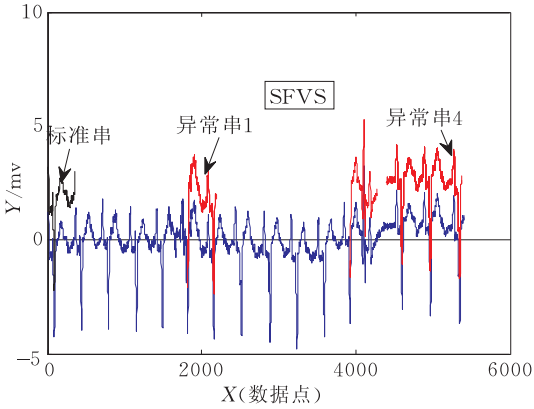
4.1 异常检测

字符集都取相同规模时,随机选择一段时序子串 T_i 作为标准串,并设定相应的阈值,用经改进的 Brute_Force^[15] 算法在整个时序数据集 T 中搜索比较,通过寻找超出阈值的时序子串来检测异常,图 3

(a)、(b)分别记录了 SAX 及 SFVS 符号时序用于同一数据集作异常检测的真实情况片断,图中较粗的黑色曲线片断是用作检测的标准串,而较细的红色曲线片断是检测出来的异常串,从图中可以直观地看到,用 SAX 符号检测出的异常串为 3 个,而用 SFVS 符号时,检测出的异常串为 4 个,显然 SFVS 的漏检率要低于 SAX. 为了全面比较两种符号时序的优劣,我们还用表 1 中的数据集做了对比实验,其具体结果详见表 3.



(a) SAX符号用于异常检测



(b) SFVS符号用于异常检测

图 3 SFVS 与 SAX 符号用于异常检测的比较

4.2 相似查询

通过对比相似查询实验,可以从另一个角度来检验 SFVS 符号时序与 SAX 各自的特点. 用 Brute_Force 算法,在时序数据集 T 中(在一定误差范围一阈值内)查找能够与给定长度标准子串 T_i 相匹配的子序列 T_j . 图 4 所示的就是一个典型实例.

图 4(a)记录的是用 SAX 时序作相似查询的结果片断,其中,最左边曲线是查询串,右边则记录了其在相似查询中出错的真实情况. 图 4(b)记录的是用 SFVS 对同一数据集做相似查询时的情况,图中显示它排除了类似 SAX 那样的错误.

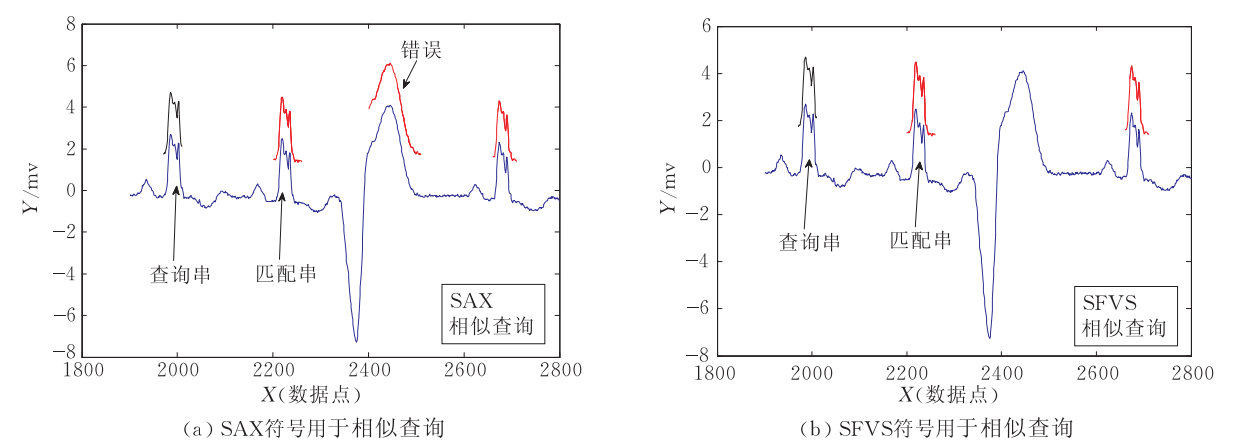


图 4 SFVS 与 SAX 符号用于相似查询的比较

表 3 记录的是表 1 所列数据集在作异常检测及相似查询 20 次实验中的平均结果. 为简便和便于比较,表 3 中第 3,4,5,6 列记录了在实验中用选定查询串测出的异常检测漏检率与相似查询误检率而第 7,8 列记录了两种不同算法的运行时间.

表 3 SFVS 与 SAX 应用比较

数据名 详见表 1	数据集 长度	异常检测漏检率/%		相似查询误检率/%		算法运行时间/s	
		SFVS	SAX	SFVS	SAX	SFVS	SAX
1	3751	18.7	36.7	5.9	23.5	0.0103	0.009
2	3750	16.7	38.1	20	44.2	0.0106	0.0094
3	1216	22.2	41.7	5	35	0.011	0.01
4	100	28.6	47.4	25	35.1	0.0008	0.00075
5	5400	10	37.5	5.6	16.7	0.054	0.0482
6	5400	14	31	23.3	43.7	0.0573	0.0476
7	9098	7.6	42	20	41.6	0.0931	0.0811
8	30	16	48.3	33	33	0.00013	0.00009
9	600	14.2	27.1	16.7	33.3	0.0055	0.00493
10	10000	7.14	57.12	21.1	58.9	0.0952	0.0894

表 3 中的每一行代表在相同条件下用两种算法对同一数据集分别进行异常检测和相似查询的比较结果,而不同行代表不同数据集的实验比较结果. 实验表明,无论是用于异常检测还是相似查询, SFVS 算法都要好于或至少不差于 SAX 算法. 从表 3 中第 3,4 列的异常检测对比结果可见:对所列的 10 个数据集而言,SFVS 检测出异常的漏检率要低于 SAX. 其原因是,当用 SAX 符号来检测异常时,实际上是检测两个符号所代表的时序子段平均值之间的距离,当此距离小于阈值时,被检测时序子段将被判定为正常;然而,采用 SFVS 符号检测时,即使两个符号的均值分量间距离小于阈值,两个符号的矢量距离仍可能大于阈值(例如当两个符号的方差分量相差很大时)而被检测为异常. 表中第 5,6 列的相似查询对比结果则说明, SFVS 的误检率要低于或至少不高于 SAX,其原因是 SFVS 的相似匹配需要同时满足两个分量分别匹配的条件,而 SAX 只需满足一个均值匹配条件即可,因此, SAX 更容易产生较多

的错误或不精确匹配. 另外,表 3 中第 7,8 列记录的两种不同算法在实验中平均运行时间的数据说明: SFSV 增加的计算代价远不及 SAX 的 50%,因此,综合算法的精度和代价这两个因素考虑,应当可以说,对比实验确实证实了 SFVS 是一种比 SAX 更为出色的符号化方法.

5 结 论

针对 SAX 符号时序方法在边界区信息丢失较多所带来的缺陷,本文提出了 SFVS 解决方案, SFVS 方法可以看作是 SAX 的改进方案,它通过引入两个统计特征分量将原来的 SAX 标量符号转化为矢量符号. 由于 SFVS 方案能够比 SAX 提供更多的对时序数据的描述信息,因而它在时序分析应用中能够获得比 SAX 更出色的表现. 实验证明它确实是一种行之有效的符号化新方案.

参 考 文 献

- [1] Daw C S, Finney C E A, Tracy E R. A review of symbolic analysis of experimental data. *Review of Scientific Instruments*, 2003, 74(2): 915-930
- [2] Kantz H, Schreiber T. *Nonlinear Time Series Analysis*. 2nd Edition. Cambridge, UK: Cambridge University Press, 2004
- [3] Faloutsos C, Ranganathan M, Manolopoulos Y. Fast subsequence matching in time-series databases//*Proceedings of the ACM SIGMOD International Conference on Management of Data*. Minneapolis, MN, 1994: 419-429
- [4] Chan K, Fu A W. Efficient time series matching by wavelets//*Proceedings of the 15th IEEE International Conference on Data Engineering*. Sydney, Australia, 1999: 126-133
- [5] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Locally adaptive dimensionality reduction for indexing large time series databases//*Proceedings of the ACM SIGMOD Conference on Management of Data*. Santa Barbara, CA, 2001: 151-162
- [6] Geurts P. Pattern extraction for time series classification//*Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*. Freiburg, Germany, 2001: 115-127
- [7] Kennel M B, Buhl M. Estimating good discrete partitions from observed data: Symbolic false nearest neighbors. *Physical Review E*, 2003, 68(8): 084102
- [8] Lin J, Keogh E J, Lonardi S, Chiu B Y. A symbolic representation of time series, with implications for streaming algorithms//Zaki M J, Aggarwal C C eds. *Proceedings of the 8th SIGMOD Workshop on DMKD 2003*. San Diego: ACM Press, 2003: 2-11
- [9] Lkhagva Battuguldur, Suzuki Yu, Kawagoe Kyoji. Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation//*DEWS2006 4A-i8*. 2006
- [10] Keogh E, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 2000, 3(3): 263-286
- [11] Lin J, Keogh E. Group SAX: Extending the notion of contrast sets to time series and multimedia data//*Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*. Berlin, Germany, 2006: 284-296
- [12] <http://www.robjhyndman.com/TSDL/>. Time Series Data Library\Industry\JENKINS8. DAT Power station data; In-phase current deviations. Source: Jenkins & Watts, 1980
- [13] Keogh E J, Chakrabarti K, Pazzani M, Mehrotra S. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 2000, 3(3): 263-286
- [14] Lin Jessica, Keogh Eamonn, Li Wei, Lonardi Stefano. Experiencing SAX: A novel symbolic representation of time series. *DMKD Journal*, 2007, 15(2): 107-144
- [15] Keogh E, Lin J, Fu A. HOT SAX: Efficiently finding the most unusual time series subsequence//*Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005)*. Houston, Texas, 2005: 226-233



ZHONG Qing-Liu, born in 1954, Ph. D. candidate, associate professor. His research interests include artificial intelligence pattern recognize etc.

CAI Zi-Xing, born in 1938, professor. Ph. D. supervisor. His research interests include artificial intelligence, robots, intelligent control etc.

Background

This research is supported by National Project for basal Research of P. R. China (A1420060159). Research on the Theory, Model and Method of mobile cooperate technology. The research group has been working on many aspects of the

Project since 2006, have published many papers. In this paper, authors discuss a new model of time series calculation for detecting mobile action. This model is helpful to improve the performance in detection.