# 基于滑动窗口及局部特征的时间序列符号化方法*

谭宏强，牛　强
（中国矿业大学 计算机科学与技术学院，江苏 徐州 221116）

**摘　要**：针对时间序列的数据挖掘将时间序列数据转换为离散的符号序列，提出了一种基于滑动窗口及局部特征的时间序列符号化方法。该方法采用了滑动窗口的方法将时间序列分割，每个分段采用多个斜率表示，最后采用 K-均值聚类算法对斜率表示的分段进行聚类，实现时间序列的符号化。实验证明了该方法的有效性与准确性。

**关键词**：时间序列；滑动窗口；局部特征；符号化
**中图分类号**：TP311　　**文献标志码**：A　　**文章编号**：1001-3695(2013)03-0796-03
doi:10.3969/j.issn.1001-3695.2013.03.039

# Symbolic representation algorithm for time series based on sliding window and local features

TAN Hong-qiang, NIU Qiang
(*School of Computer Science & Technology*, *China University of Mining & Technology*, *Xuzhou Jiangsu* 221116, *China*)

**Abstract**: This paper put forward a symbolic method for time series which based on the sliding window and the local features. Firstly, this method divided time series according to the sliding window method, and used multiple slopes to show the each subsection of time series. Then it used K-means clustering algorithm to clustering the slope representation of subsection, and realized the symbolization of time series. Experiments show that the method is effective and accurate.

**Key words**: time series; sliding window; local features; symbolization

在数据挖掘领域，时间序列数据是一类重要的数据对象，在科学研究、商业应用、交通控制、工业生产中普遍存在。通过对时间序列进行分析研究，可以揭示事物运动、变化和发展的内在规律，对人们正确地认识事物并据此作出科学的决策具有重要的现实意义。由于时间序列数据具有高维度、大量噪声干扰以及非稳态等特性，因此直接在原始时间序列上进行相似性查询、分类、聚类、模式挖掘等工作，不但会造成存储和计算效率低下，而且会影响算法的准确性和可靠性，从而难以获得满意的结果。因此，在进行时间序列数据挖掘之前，需要对时间序列数据进行处理。许多研究者提出了时间序列的模式表示，其中时间序列符号化表示是一种重要的方法。

时间序列符号化的基本思想是通过一些离散化方法将时间序列的连续实数值或者一段时间内的时间序列波形映射到有限的符号表上，将时间序列转换为有限符号的有序集合，即把时间序列用符号集 $S=\{s_1,s_2,\cdots,s_{|s|}\}$ 表示出来，其中 $|S|$ 是符号集的大小（符号个数）。许多相关的文献对时间序列符号化方法展开了研究，文献[1]提出的新型符号化表示方法——符号聚合近似算法（SAX）正是吸取了这种思想，然而该方法只适用于能够等分并且服从高斯分布的时序数据，在符号化过程中只是利用了子段数据的均值信息作为局部特征，造成符号化描述信息不够充分；文献[2]在此基础上加入方差统计信息，仍未解决时间序列必须能够等分的缺陷；文献[3]研究了时间序列的规则发现，采用定长滑动窗口获取子序列得到元模式，但窗口宽度的选取没有一个明确的标准，难以确定一个合理的窗口宽度使得所有时间序列片段都具有相对独立的变化模式，并且不经过任何特征提取与转换直接对时间序列数据进行聚类，计算量大且聚类效果不理想；文献[4]采用线性分段拟合得到元模式，但合理的分段数难以确定，使得序列中混杂不同尺度的元模式，影响模式发现的准确性。其他的还有如文献[5]对时间序列数据采用等间隔和最大熵方法实现符号化；文献[6]提出了基于云模型的符号化方法；Xia[7]利用区间离散化方法实现了时间序列的符号表示。

针对时间序列的特性，本文结合滑动窗口以及斜率的思想，提出了一种新的时间序列符号化的算法——基于局部特征的符号化算法（SFSA）。

## 1　基于局部特征聚类的时间序列符号化算法

### 1.1　时间序列的分段及局部特征表示

在对时间序列进行符号化前，需要对时间序列按照某种方法进行分割，针对当前存在的多种分割方法，本文采用了固定滑动窗口的分割方法，该方法能够快速地实现时间序列的分割；但是采用该方法进行分割后的时间序列在每个分段中还存在着各种形态模式，如何表示每个分段的形态模式特征，本文提出了采用斜率来表示每个分段的方法。

对于每个分段，本文采用多个斜率来描述分段，假如每个

**收稿日期**：2012-07-10；**修回日期**：2012-08-06　**基金项目**：国家教育部博士点基金资助项目(20100095110003)；中国矿业大学青年科技基金资助项目(2011QNB23)

**作者简介**：谭宏强(1988-)，男，四川射洪人，硕士研究生，主要研究方向为数据挖掘(hongqiangtan@163.com)；牛强(1974-)，男，副教授，博士，主要研究方向为智能优化算法、数据挖掘.

分段采用 SlopeNum 个斜率来表示,因此每个分段演变为 SlopeNum + 1 个数据点,每个分段的第一个点和最后一个点保持不变,其余的 SlopeNum-1 个点通过求数据的平均值的方法计算得来。根据定义 3 可以知道,子序列 $i$ 表示为 SlopeNum 个斜率分别为 $sp_{i,0}, sp_{i,1}, \cdots, sp_{i,\mathrm{SlopeNum}}$,子序列 $i$ 的第 $j$ 个斜率为

$$sp_{i,j} = (t_{i,(j+1)'} - t_{i,(j)'})/((j+1)' - (j)')$$

其中:$t_{i,j}$ 表示子序列 $i$ 的第 $j$ 个点。

将每个分段用斜率表示后,较好地保留了每个分段的形态信息,为后面的聚类及符号化提供了更准确的信息。下面通过某一时间序列为例对该方法进行说明。一子序列为 4 8 1 2 6,本文采用 2 个斜率来表示该分段,因此该分段只需要 3 个数据来获得斜率,第 1 个点 4 与第 4 个点 6 不变。由于 $(2+3)/2 = 2.5$,$(8+12)/2 = 10$,因此该分段的第一个斜率为 $(10-4)/(2.5-1) = 4$,第 2 个斜率为 $(6-10)/(4-2.5) = -2.66$,最后该分段表示为 $(4, -2.66)$。

### 1.2　时间序列符号化

采用上述的时间序列分段斜率表示后,要进一步实现时间序列的符号化,需要采用聚类算法,对每个分段进行聚类,对每个类别赋予相应的符号标志,从而实现时间序列的符号化[8]。

聚类就是根据样本之间的相似度将样本集合分为多个类或簇,要求同一个类中的样本之间具有较高的相似度,不同的类中样本差别较大。选取合适的相似度度量标准对聚类的结果十分重要。由于通过上述的分段表示方法后获得时间序列分段等长,因此本文采用经典的欧几里德相似性度量作为时间序列子序列的相似性度量。两条长度为 $n$ 的时间序列分别为 $Q$ 和 $C$,两条时间序列之间的欧几里德相似性度量定义为

$$D(Q,C) \equiv \sqrt{\sum_{i=1}^{n}(q_i - c_i)^2}$$

欧几里德以距离来衡量两个时间序列之间的相似度,$D(Q,C)$ 越小,则说明它们之间的相似度越大;反之,则反。它只适用于等长的时间序列,不适用于非等长的情况。本文由于分段子序列是等长的序列,故选择此种相似性度量。

本文根据实际的情况,选择了 K-均值聚类算法,对于聚类的结果分别赋予相应的符号,实现时间序列的符号化。

### 1.3　SFSA 算法的具体实现步骤

通过上面的阐述,本文提出的时间序列符号化方法主要实现步骤如下:

a)设定大小为 $W$ 的滑动窗口,将时间序列 $X = \langle x_0 = (x_0, t_0), x_1 = (x_1, t_1), \cdots, x_n = (x_n, t_n) \rangle$ 分段表示为 $\lfloor \frac{n}{W} \rfloor + 1$ 个子段 $s_0, s_1, \cdots, s_{\lfloor \frac{n}{W} \rfloor + 1}$;

b)将每个子段 $s_0, s_1, \cdots, s_{\lfloor \frac{n}{W} \rfloor + 1}$ 分别用 SlopeNum 个斜率表示,即 $s_0 = (sp_{0,0}, sp_{0,1}, \cdots, sp_0, \mathrm{SlopeNum})$, $\cdots$, $S_{\lfloor \frac{n}{W} \rfloor + 1} = (sp_{\lfloor \frac{n}{W} \rfloor + 1,0}, sp_{\lfloor \frac{n}{W} \rfloor + 1,1}, \cdots, sp_{\lfloor \frac{n}{W} \rfloor + 1}, \mathrm{SlopeNum})$;

c)对原始时间序列的分段子序列 $s_0, s_1, \cdots, s_{\lfloor \frac{n}{W} \rfloor + 1}$ 采用 K-均值算法进行聚类,符号集大小为 $|s|$,分段序列被分为 $|s|$ 类,将每一类赋予相应的符号表示,即可完成时间序列的符号化表示。

## 2　实验与分析

实验的硬件环境为:CPU 为 Intel Core T5800 2.00 GHz;内存为 2 GB,硬盘为 250 GB。软件环境为:Microsoft Windows XP,在 Visual Studio 2008 编译,采用 C#编写实现算法。

本文通过两个方面的实验来论证所提方法,一方面论证该不同参数对于该时间序列符号化方法的影响;另一方面通过与经典的时间序列符号化方法 SAX 进行对比,分析本文方法在聚类应用中的优劣。

### 2.1　实验数据

实验选用了 UCI KDD archive 的 synthetic control chart time series(SCCTS)数据集作为研究对象。这些数据来源于工业生产控制图,SCCTS 包含 6 类时间序列,每类包含 100 条时间序列,共 600 条时间序列,每条时间序列长度为 60。

### 2.2　SFSA 方法

任意抽取一条时间序列,利用本文的符号化方法,将该时间序列表示为符号序列,取滑动窗口大小 $W$ 为 4,迭代次数 $I$ 为 20 次,每个分段采用 2 个斜率表示,字符集大小为 4,得到结果如图 1 所示。

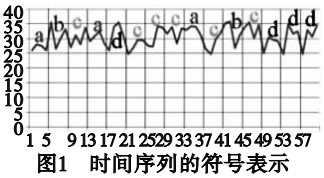
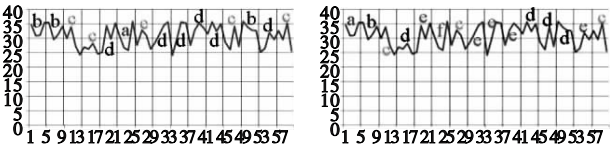


图1　时间序列的符号表示

利用 SFSA 方法,原始时间序列被表示为"abcadcccacbcddd"这样一个字符串序列,因此该方法能够实现时间序列的降维。通过观察图 1 可以大致看出,每个符号代表了一种局部特征,因此本文提出的 SFSA 方法不仅能够有效地降低原始时间序列的维度,而且能够保留原始时间序列的形态模式信息。

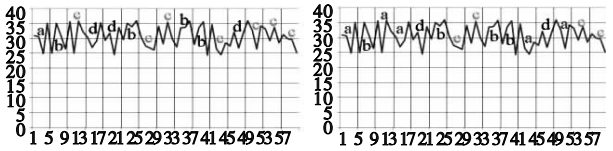
通过前面的阐述,SFSA 方法的参数主要有滑动窗口的大小 $W$、迭代次数 $I$、斜率个数 SlopeNum 以及符号集大小 $|S|$。本文从实验数据集中任取时间序列,分别在不同的参数情况下,结果如图 2 ~ 5 所示。



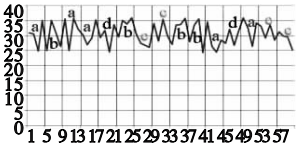
(a) $W$=4,$I$=40,SlopeNum=2,$|S|$=4　　(b) $W$=4,$I$=40,SlopeNum=2,$|S|$=6

图2　不同符号集大小的结果

通过比较图 2 可以发现,在其他参数相同的情况下,符号集的大小影响了符号序列所表示的原始时间序列的细致程序,符号集越大,符号序列所能表征的原始时间序列局部特征越细致,同一个符号表示的分段更准确;反之亦然。但是在这里存在一个问题,符号集过大影响了降维效果,所以符号集的大小应该根据实际情况来设定。



(a) $W$=4,$I$=1,SlopeNum=2,$|S|$=4　　(b) $W$=4,$I$=5,SlopeNum=2,$|S|$=4

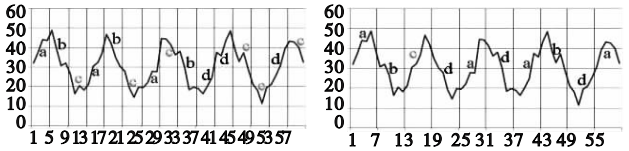


(c) $W$=4,$I$=40,SlopeNum=2,$|S|$=4

图3　不同迭代次数的结果

由图 3 可知,随着迭代次数的增加,符号序列表示的准确率逐渐提高,每一个符号表示分段近似度逐渐变高,但是,当迭代次数的增加到一定程度后,符号序列将不会改变,这是由于

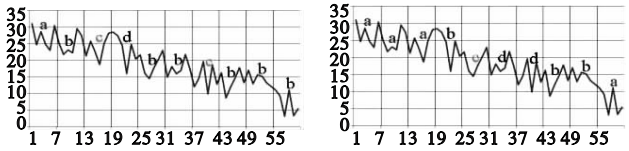
在对分段进行聚类时采用了 K-均值聚类算法,而聚类算法存在结果向局部最优靠近的特点,所以会出现这样的现象。另外,迭代次数的增多同时影响着算法的运行效率,可以从上面知道迭代次数达到一定的值后,结果将不再变化,所以迭代次数并不是越大越好。



(a) $W$=4,$I$=40,SlopeNum=2,|$S$|=4 (b) $W$=6,$I$=40,SlopeNum=2,|$S$|=4

**图4 不同滑动窗口大小的结果**

通过对比图 4 可以发现,在其他参数不变的情况下,滑动窗口的大小影响了算法的准确性及压缩率。滑动窗口增大,压缩率变高,但是相应的准确率下降;反之,滑动窗口减小,压缩率降低,准确率变高。



(a) $W$=6,$I$=40,SlopeNum=2,|$S$|=4 (b) $W$=6,$I$=40,SlopeNum=3,|$S$|=4

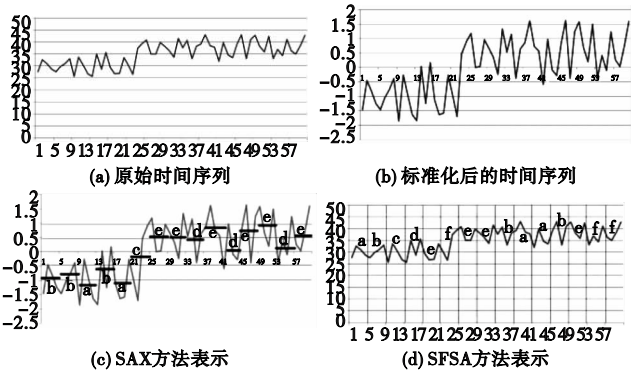
**图5 不同斜率个数的结果**

由图 5 分析可知,每个分段用较多的斜率表示,该算法的准确率提高了,即具有相同或相似的原始分段能够用更准确的符号表示。但是该算法过多的斜率表示分段将导致算法复杂性提高,运行效率降低,因此在准确率与效率之间,应该根据实际情况设置该参数。

综合分析上述内容,由于 SFSA 算法采用了 K-均值聚类算法对原始时间序列的分段进行聚类,因此迭代次数 $I$ 以及符号集大小|$S$|是影响算法性能的主要指标。滑动窗口大小 $W$ 以及斜率个数 SlopeNum 是影响算法准确性及压缩率的主要指标。

### 2.3 SFSA 方法与 SAX 算法的对比

为了说明本文提出的方法的优点,本文选取经典的时间序列符号化方法 SAX 方法与本文提出的方法进行对比实验,比较任选的一条时间序列。SFSA 方法设置 $W=4$, $I=40$, SlopeNum=2, $|S|=6$;SAX 方法设置 $W=4$, $|S=6|$,结果如图 6 所示。



(a) 原始时间序列 (b) 标准化后的时间序列

(c) SAX方法表示 (d) SFSA方法表示

**图6 SFSA与SAX方法的对比**

通过分析图 6(c)(d)可知,SAX 方法采用 PAA 的近似表示方法,采用平均值来表示每个分段,忽略了原始时间序列原有的局部特征信息,只能近似地描述时间序列大致特征;本文提出的方法能够有效降维,同时考虑到了原始时间序列的局部特征,利用斜率来表示分段,能够尽可能地保留原始时间序列的信息,该方法使得每个符号代表了原始时间序列的一个形态模式,使得结果更加准确。因此,本文提出的 SFSA 方法能够包含更多的原始时间序列的信息,比 SAX 方法提供了更为全面的描述信息,为后续的进一步的时间序列数据挖掘提供了可靠的保证。

此外,本文还比较了两种时间序列符号化方法在聚类方面的效果,本文采用编辑距离进行相似性度量,这是因为该度量方式计算简单及快速。对于聚类准确率,利用聚类结果与原始的数据所属的类别进行比较,从而得出本文提出的方法的聚类准确率。

从数据集中每类随机抽取 15 条时间序列数据,随机分成三组进行实验,实验结果如表 1 所示。

表 1 两种算法的实验结果

| 序号 | 聚类准确率/% | | 平均消耗时间/ms | |
|---|---|---|---|---|
| | SAX | SFSA | SAX | SFSA |
| 1 | 84.20 | 90.10 | 220 | 340 |
| 2 | 89.30 | 92 | 249 | 310 |
| 3 | 85.67 | 91.04 | 216 | 320 |

通过表 1 可以知道,SFSA 方法在准确性方面优于 SAX 方法,这是由于 SFSA 方法较多地保留了原始时间序列的形态特征。然而,由于 SFSA 方法采用了 K-均值聚类算法对子时间序列进行聚类,运行时间方面多于 SAX 方法,但是在运行时间方面仍然是能够接受的范围。

## 3 结束语

本文提出了一种基于滑动窗口及局部特征的时间序列符号化算法(SFSA),它利用滑动窗口算法实现时间序列的快速分段,对于每个分段采用多个斜率表示后进行聚类,将每类标志相应的符号以实现时间序列的符号化。该方法不仅能够有效地进行时间序列降维处理,而且能够保留更多的原始时间序列的形态特征,为时间序列的进一步研究提供了更为可靠的符号化表示方法。实验证明了该方法的有效性和优越性,是一种行之有效的符号化方法。

**参考文献:**


[1] LIN J, KEOGH E, LONARDI S, *et al*. A symbolic representation of time series, with implications for streaming algorithms[C]//Proc of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Disco-very. San Diego: ACM Press, 2003: 2-11.

[2] 钟清流,蔡自兴. 基于统计特征的时序数据符号化算法[J]. 计算机学报, 2008, **31**(10): 1857-1864.

[3] GAUTAM D, DAVID L, HEIKKI M, *et al*. Rule discovery from time series[C]//Proc of the 4th Annual Conference on Knowledge Discovery and Data Mining. 1998: 16-22.

[4] LI Bin, TAN Li-xiang, ZHANG Jin-song, *et al*. Using fuzzy neural network clustering algorithm in the symbolization of time series[C]// Proc of IEEE Asia-Pacific Conference on Circuits and System. 2000: 379-382.

[5] SANGHYUN P, WESLEY W C, JEEHEE Y, *et al*. Efficient searches for similar subsequences of different lengths in sequence database [C]//Proc of the 16th International Conference on Data Engineering. Washington DC: IEEE Computer Society. 2009: 23-32.

[6] 蒋嵘毅. 基于形态表示的时间序列相似性搜索[J]. 计算机研究与发展, 2000, **37**(5): 601-608.

[7] XIA B B. Similarity search in time series datasets[D]. [S. l.]: Simon Fraser University, 1997.

[8] 陈湘涛,李明亮,陈玉娟. 基于分割模式的时间序列矢量符号化算法[J]. 计算机工程, 2011, **37**(4): 55-57.