基于 SFVS 的时序关联规则动态发现方法*

张新玉，夏士雄，牛　强
（中国矿业大学 计算机学院，江苏 徐州 221116）

摘　要：针对时间序列关联规则挖掘存在时间复杂度高、效率低等问题，将基于 SFVS（统计特征矢量符号化）的时间序列表示方法引入到时序关联规则发现中，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及发散程度的分量，实现时间序列表示的矢量化，然后再进行动态关联规则挖掘。实验结果表明，基于该方法所获取的关联规则具有更高的精确度和可信度。

关键词：时间序列；统计特征矢量；符号化表示；关联规则
中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1001-3695(2012)07-2571-04
doi:10.3969/j.issn.1001-3695.2012.07.046

Dynamically temporal association rules mining based on SFVS

ZHANG Xin-yu, XIA Shi-xiong, NIU Qiang
(*School of Computer Science & Technology, China University of Mining & Technology, Xuzhou Jiangsu* 221116, *China*)

Abstract: There are many disadvantages of discovering the association rules directly on the temporal series, such as high time complexity, low efficiency. So This paper considered that introduced the temporal series method based the SFVS(statistic feature vector symbolic) algorithm into the association rules discovering. Used the mean and variance describing the characteristics of temporal series data as the component describing the average and the degree of divergences, that was to make the temporal series be vectors. Then discovered the association rules dynamically. The experimental results show that the association rules discovered based SFVS algorithm have a higher accuracy and reliability.

Key words: time series; SFVS; symbolic representation; association rules

时间序列是指按照时间先后顺序排列的观测记录的有序集合，已广泛应用于商业、经济、科学工程和社会科学等领域。关联规则是数据挖掘领域中的一个非常重要的研究课题，广泛应用于各个领域，既可以检验行业内长期形成的知识模式，也能够发现隐藏的新规律。有效地发现、理解、运用关联规则是完成数据挖掘任务的重要手段，因此对时间序列关联规则的研究具有重要的理论价值和现实意义。

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。钟清流等人[1]提出了基于统计特征的时序数据符号化（SFVS）算法。该算法将时序符号看做矢量，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的程度。

1　相关定义

定义 1　时间序列。它是由记录值和记录时间组成的元素的有序集合。记为 $X=\{x_1=(v_1,t_1),x_2=(v_2,t_2),\cdots,x_n=(v_n,t_n)\}$，元素 $x_i=(v_i,t_i)$ 表示时间序列在 t_i 时刻的记录值为 v_i，记录时间 t_i 间是严格增加的[2]。一般情况下，时间序列的采样间隔时间 Δt 相等，可以看做 $t_1=0,\Delta t=1$。此时将时间序列 $X=\{x_1=(v_1,t_1),x_2=(v_2,t_2),\cdots,x_n=(v_n,t_n)\}$ 简记为 $X=\{x_1,x_2,\cdots,x_n\}$。

定义 2　时间序列的标准化[3]。设有时间序列 $X=\{x_1=(v_1,t_1),x_2=(v_2,t_2),\cdots,x_n=(v_n,t_n)\}$，集合 $X=\{x_1,x_2,\cdots,x_n\}$ 的均值和方差分别为 $\mu(x)$ 和 $\rho(x)$，则时间序列 X 标准化为

$$N(X)=\{n_{x1},n_{x2},\cdots,n_{xn}\} \tag{1}$$

其中，$n_{xi}=(x_i-\mu(x))/\rho(x)$。

定义 3　序列的符号化[4,5]。时间序列的符号化是指将以连续数值表示的时间序列用离散的相对抽象的符号表示成符号序列。时间序列 $X=\{x_1,x_2,\cdots,x_n\}$ 经过符号化后表示为 $S=\{a_1,a_2,\cdots,a_n\}$。其中，$a_k=\{a,b,c,d,\cdots,\}$，$k\in(0,1,2,\cdots,N)$。

定义 4　关联规则[6]。设 $I=\{i_1,i_2,\cdots,i_m\}$ 是 m 个不同的项目组成的集合，给定一个事务数据库 D，其中的每一个事务 T 是 I 中一组项目的集合，即 $T\subseteq I$，T 有一个唯一的标志符 TID。若项集 $X\subseteq I$ 且 $X\subseteq T$，事务集 T 包含项集 X。一条关联规则就是形如 $X\Rightarrow Y$ 的蕴涵式，其中 $X\subseteq I,Y\subseteq I,X\cap Y=\varnothing$。关联规则 $X\Rightarrow Y$ 成立的条件：a）它具有支持度 s，即事务数据库 D 中至少有 $s\%$ 的事务包含 $X\cup Y$；b）它具有置信度 c，即在事务数据库 D 中包含 X 的事务至少有 $c\%$ 同时也包含 Y[7]。

2　基于 SFVS 的时间序列动态关联规则发现方法

2.1　SFVS 算法[1]

传统符号化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。基于这个认识，SFVS 算法作为时序数据符

收稿日期：2011-11-16；**修回日期**：2011-12-29　　**基金项目**：中国矿业大学青年科技基金资助项目（2011QNB23）

作者简介：张新玉（1988-），男，硕士研究生，主要研究方向为时间序列处理与分析（zhangxinyu247@163.com）；夏士雄（1961-），男，博士，教授，主要研究方向为智能信息处理；牛强（1974-），男，副教授，博士，主要研究方向为数据挖掘.

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,而时序符号则看做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,每个时序子段将转换成一个有两个分量的符号矢量。其中各时序子段的均值作为衡量其平均值特征的分量,而相应时序子段的方差则作为衡量其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。

2.1.1 时间序列预处理

1)时间序列标准化

由于对具有不同偏移量和振幅的时间序列进行比较是没有意义的,因此在对时间序列符号化之前必须先进行标准化处理,标准化后的时间序列的平均值为 0,方差为 1,服从高斯分布,标准化后的时间序列记为 $X=\{x_1,x_2,\cdots,x_n\}$。

2)确定最佳压缩比[1]

压缩比相当于一个符号所能代表的时序数点数,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。大多数文献确定压缩比的方法是凭经验或实验来确定。

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描述,下面是本文采用的经验式[8]:

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

其中: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。

3)计算出各时序子段的均值和方差

用 PAA 方法将标准化时序数据 $X=\{x_1,x_2,\cdots,x_n\}$ 按适当的压缩比 w 降维,生成 $N=n/w(N<<n)$ 维空间向量 $\overline{X}=\bar{x}_1,\bar{x}_2,\cdots,\bar{x}_n$。其中,$\overline{X}$ 的第 i 个元素可由下式计算:

$$\bar{X}_i=\frac{N}{n}\sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i}X_j \tag{3}$$

另一个需要同时预处理的量是相应(第 i 个)时序子段的方差,当 $w=n/N$ 时,

$$S_i=\frac{1}{w}\sum_{k=1}^{w}(x_{ik}-\overline{X_i})^2 \tag{4}$$

2.1.2 SFVS 算法的实现

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[1]。这样,在实施时序数据符号化过程中,每个时序子段将转换成一个有两个分量的符号矢量。其中,各时序子段的均值作为描述其平均值特征的分量,而相应时序子段的方差则作为描述数据对平均值的偏离程度特征的分量。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其他符号化方法提供更为全面的描述信息。

数学描述为

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

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

其中:C_{j-1}、C_j 分别代表第 j 个字符划分区的上限与下限。则 $\hat{X}_i=\hat{x}_{i1}\cdot i+\hat{x}_{i2}\cdot j=A_{i1}\cdot i+A_{i2}\cdot j$。

经过预处理后的时间序列服从高斯分布,所以划分点集 C_j 可以通过将整个正态分布区间划分成 K 个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。则一个有 K 个字符组成的字符集需要有 $K-1$ 个划分点,因此划分点集可以表示为 $C=\{C_1,C_2,\cdots,C_{k-1}\}$。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,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。若采用英语字符来表示,可将每个预处理时序序列转换为相应的符号集,而按式生成类似于以下符号矢量集:

$$\hat{X}_i=\hat{X}_1,\hat{X}_2,\cdots,\hat{X}_n=$$
$$(a_1\cdot i+b_1\cdot j),(a_2\cdot i+b_2\cdot j),\cdots,(a_n\cdot i+b_n\cdot j) \tag{6}$$

其中:$a_i,b_i\in(A_1,A_2,\cdots,A_n)$。

2.2 基于动态的关联规则的频繁子序列挖掘

通过一个具体的例子说明该算法如何挖掘频繁子序列(给定最小支持度为 2)。某时间序列 X 经过 SFVS(符号集 $K=3$)符号化后表示为

$$S=(a\cdot i+b\cdot j),(a\cdot i+c\cdot j),(b\cdot i+a\cdot j),(c\cdot i+a\cdot j),$$
$$(c\cdot i+c\cdot j),(a\cdot i+b\cdot j),(b\cdot i+b\cdot j),(a\cdot i+a\cdot j),$$
$$(c\cdot i+a\cdot j),(b\cdot i+c\cdot j),(a\cdot i+b\cdot j),$$
$$(a\cdot i+c\cdot j),(b\cdot i+a\cdot j) \tag{7}$$

简记为

$$S=(a,b),(a,c),(b,a),(c,a),(c,c),(a,b),(b,b),$$
$$(a,a),(c,a),(b,c),(a,b),(a,c),(b,a) \tag{8}$$

符号化表示后的挖掘步骤如下:

a)扫描符号化后的时间序列,如表 1 所示。

表 1 符号出现在符号化后的时间序列的位置

(a,a)	(a,b)	(a,c)	(b,a)	(b,b)	(b,c)	(c,a)	(c,b)	(c,c)
8	1	2	3	7	10	4	无	5
无	6	12	13	无	无	9	无	无
无	11	无	无	无	无	无	无	无

表 1 中第一行表示一个符号,第二~四行表示该符号出现的位置。

b)挖掘长度为 1 的频繁子序列 $L(1)$。

扫描表 1,出现次数大于等于 2 的符号组成 $L(1)$。根据表 1 容易得到

$L(1)=\{(a,b):(1,6,11),(a,c):(2,12),(b,a):(3,13),(c,a):(4,9)\}$(数字表示第一个字符出现的位置)

c)挖掘长度为 2 的频繁子序列。

根据 $L(1)$ 挖掘出 $L(2)$,首先用 (a,b) 和其他项比较,因为 (a,b) 出现的位置为 1、6、11,在 1、6、11 上面都加 1 得到 2、7、12。如果其他项中出现这三个数字中的任意两个即为所得 $L(2)$。显然只有 (a,c) 符合条件,则 $L(2)=\{[(a,b),(a,c)]:(1,11)\}$。然后用 (a,c) 和其他项比较,因为 (a,c) 出现的位置为 2、12,在 2、12 上面都加 1 得到 3、13。显然也只有 (b,a) 符合条件,则 $L(2)=\{[(a,c),(b,a)]:(2,12)\}$。同理用 (b,a) 和 (c,a) 两项和其他项比较,无符合条件的 $L(2)$ 出现。则

$L(2)=\{[(a,b),(a,c)]:(1,11),[(a,c),(b,a)]:(2,12)\}$

d)挖掘长度为 3 的频繁子序列。

用[(a,b),(a,c)]项和其他项比较,因为[(a,b),(a,c)]出现位置为 1 和 11,且末项为(a,c)。则在其他项中寻找以(a,c)打头的项,且出现位置为 1 + 2 - 1 = 2,11 + 2 - 1 = 12。显然[(a,c),(b,a)]:(2,12)符合条件。则

$$L(3) = \{[(a,b),(a,c),(b,a)]:(1,11)\}$$

3　实验及结果分析

实验结果比较基于 SFVS 和 SAX 两种符号化方法来进行关联规则发现时,所需的时间复杂度、效率及精准度。实验运行环境为 Intel Core™2 Duo CPU E7500 双核,2 GB 内存和 320 GB 硬盘,操作系统是 Windows 7(32 bit),开发工具为 Microsoft Visual Studio 2008 ,SQL Server 2005,使用的数据集如表 2 所示。

表 2　数据集

序号	序列名称	序列长度	数据源
1	Burst	9 385	UCR
2	Powerplant	2 400	UCI
3	chfdb_chf　01_275	3 751	ECG

实验考虑了最大压缩比、最小支持数阈值和符号集的个数分别取不同值时的挖掘结果,实验结果如下。

表 3 给出了三组数据集在基于 SFVS 和 SAX 两种符号化方法在不同压缩比下的频繁子序列的数量。该组实验中符号集为 4,最小支持度为 5。

表 3　不同压缩比下的实验结果

数据集	压缩比 w	符号化方法	频繁子序列的数量 K					
			1	2	3	4	5	6
1	10	SFVS	12	34	51	60	66	68
		SAX	4	14	23	41	49	53
	20	SFVS	12	30	48	54	59	61
		SAX	4	12	20	37	45	47
2	10	SFVS	12	28	43	51	57	63
		SAX	4	8	15	28	32	35
	20	SFVS	11	26	39	47	53	57
		SAX	4	8	14	26	30	31
3	10	SFVS	12	31	46	56	62	65
		SAX	4	13	19	35	39	41
	20	SFVS	12	27	42	49	54	56
		SAX	4	12	16	33	35	36

表 4 给出了三组数据集在基于 SFVS 和 SAX 两种符号化方法在不同支持度下的频繁子序列的数量。该组实验中符号集为 4,压缩比为 10。

表 4　不同最小支持度下的实验结果

数据集	支持度	符号化方法	频繁子序列的数量 K					
			1	2	3	4	5	6
1	5	SFVS	12	34	51	60	66	68
		SAX	4	14	23	41	49	53
	10	SFVS	10	26	47	55	62	61
		SAX	4	11	19	35	39	42
2	5	SFVS	12	28	43	51	57	63
		SAX	4	8	15	28	32	35
	10	SFVS	6	26	36	27	23	22
		SAX	4	8	13	25	29	30
3	5	SFVS	12	31	46	56	62	65
		SAX	4	13	19	35	39	41
	10	SFVS	11	24	29	36	39	31
		SAX	4	10	14	27	31	33

SFVS 和 SAX 两种算法运行时间对比如图 1 所示。

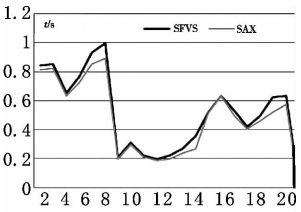

图1　两种算法运行时间对比

从表 3 数据可以看出,虽然随着压缩比的增大,两种符号化方法对原序列的表示都变得不精确,使本身在低压缩率下相似的子序列在高压缩率下变得不相似,从而挖掘出来的频繁子序列数量减少,但是基于 SFVS 方法挖掘出来的频繁子序列比基于 SAX 方法挖掘出来的频繁子序列多,特别是当 K 值比较小时,更加明显,证明了基于 SFVS 方法优于基于 SAX 的方法。

从表 4 数据可以看出,随着最小支持度的增大,两种方法挖掘出来的频繁子序列都逐渐较少,特别是在数据集 2 和 3 中,支持度为 10,K 值等于 5、6 时,基于 SAX 方法挖掘出来的频繁子序列反而比基于 SFVS 方法挖掘出来的频繁子序列多。从表面上看,似乎 SAX 方法优于 SFVS 方法,其实并不是如此。之所以出现这样的情况,是因为数据集 2 和 3 的序列长度比较短,当 K 等于 5 和 6 时,子序列总数量急剧减少,且 SFVS 方法的候选子序列远高于 SAX 方法的候选子序列,这样就使得当最小支持度较大时的频繁子序列数目大大减少,出现这样的结果能更好地证明基于 SFVS 的方法优于基于 SAX 的方法。

表 5 给出了三组数据集在基于 SFVS 和 SAX 两种符号化方法在不同符号集下的频繁子序列的数量。该组实验中压缩比为 10,最小支持度为 5。

从表 5 数据可以看出,随着符号集的增大,候选子序列也增加,对原始时间序列的表示越精确,使得 K 等于 1、2 的频繁子序列极度增大,同时随着符号集的增加,使得本身在低符号集下相似的两条子序列在高符号集下不相似,致使 K 等于 4、5、6 的频繁子序列的数量相对减少,在基于 SFVS 方法中,变得更加明显,证明基于 SFVS 方法比 SAX 方法具有更高的可信度。

表 5　不同数量符号集下的实验结果

数据集	符号集	符号化方法	频繁子序列的数量 K					
			1	2	3	4	5	6
1	4	SFVS	12	34	51	60	66	68
		SAX	4	14	23	41	49	53
	5	SFVS	14	42	67	65	62	61
		SAX	5	21	39	43	36	34
	6	SFVS	14	59	85	72	64	59
		SAX	6	32	46	52	51	45
2	4	SFVS	12	26	42	54	56	53
		SAX	4	8	15	28	32	35
	5	SFVS	12	31	47	50	43	41
		SAX	5	13	27	31	36	33
	6	SFVS	12	38	52	49	42	36
		SAX	6	21	35	35	34	30
3	4	SFVS	12	31	42	52	61	58
		SAX	4	13	19	35	39	41
	5	SFVS	12	35	58	54	51	50
		SAX	5	19	30	42	39	38
	6	SFVS	13	44	63	54	50	48
		SAX	6	27	38	40	37	35

表 6 给出了三组数据集在基于 SFVS 和 SAX 两种符号化方法在不同符号集、压缩比和支持度下所需的时间。该组实验中压缩比为 10,最小支持度为 5。

表6 两种算法的运行时间

数据集	符号化方法	压缩比	运行时间	支持度	运行时间	符号集	运行时间
1	SFVS	10	0.843 4	5	0.842 8	4	0.853 7
	SAX		0.792 9		0.813 2		0.821 1
	SFVS	20	0.653 3	10	0.763 1	5	0.934 2
	SAX		0.632 2		0.721 7		0.854 8
	SFVS					6	0.998 5
	SAX						0.891 3
2	SFVS	10	0.206 1	5	0.312 5	4	0.223 1
	SAX		0.196 6		0.298 4		0.213 6
	SFVS	20	0.198 8	10	0.223 3	5	0.264 9
	SAX		0.187 6		0.196 6		0.237 2
	SFVS					6	0.356 4
	SAX						0.264 5
3	SFVS	10	0.520 7	5	0.634 2	4	0.531 7
	SAX		0.512 5		0.633 7		0.496 3
	SFVS	20	0.418 3	10	0.493 1	5	0.624 9
	SAX		0.405 0		0.461 7		0.521 5
	SAX					6	0.631 4
	SFVS						0.574 8

从表6和图1可以看出,基于SFVS方法的时间复杂度与基于SAX方法的时间复杂度是同一个级别的。

综上所述,SFVS算法比其他符号化的时间序列具有更高的精确度,基于SFVS挖掘出来的关联规则具有更高的可信度,并且有着不高于其他方法的时间复杂度。

4 结束语

由于时间序列的海量和复杂的数据特点,直接在时间序列上进行数据挖掘不但在储存和计算上要花费高昂代价,而且可能会影响算法的准确性和可靠性。因此,时间序列的符号化显得特别重要。然而,传统的符号时序方法在边界区信息丢失较多所带来的缺陷,从而在其进行数据挖掘、关联规则的发现、相似性查询及故障诊断时就会出现偏差甚至完全不一样的结果。本文提出了基于SFVS的时间序列的动态关联规则发现方法的解决方案,SFVS方法可以看做是SAX的改进方案,它通过引入两个统计特征分量将原来的SAX标量符号转换为矢量符号。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,同样在其基础上进行的关联规则的发现也更符合实际。

参考文献:

[1] 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.

[2] PRAT K B, FINK E. Search for patterns in compressed time series [J]. International Journal of Image and Graphics, 2002, 2(1): 89-106.

[3] 胡晓琳,陈晓云. 基于符号化表示的时间序列频繁子序列挖掘[J]. 计算机工程,2008,34(10):60-63.

[4] ZAKI M J. Scalable algorithm for association mining [J]. IEEE Trans on Knowledge and Data Mining, 2000, 12(3): 372-390.

[5] RAJAGO P V, RAY A. Wavelet based space partitioning for symbolic time series analysis[C]//Proc of IEEE Conference on Decision and Control and European Control Conference. 2005: 5245-5250.

[6] 詹艳艳, 陈晓云, 徐荣聪. 基于时间序列的模式表示挖掘频繁子模式[J]. 计算机工程与应用, 2006, 42(21): 139-142.

[7] 向旭, 蒋静坪. 时间序列的符号化方法研究[J]. 模式识别与人工智能, 2007, 20(2): 154-161.

[8] 曲文龙, 张克君, 杨炳儒. 基于奇异事件特征聚类的时间序列符号化方法[J]. 系统工程与电子技术, 2006, 28(8): 1131-1134.

[9] KENNEL M B, BUHL M. Estimating good discrete partitions form observed data: symbolic false nearestneighbors [J]. Physical Review Letters, 2003, 91(8): 84-102.

[10] 钟清流, 蔡自兴, 陈名权. 时序数据的动态有界符号化方法[J]. 控制与决策, 2008, 23(10): 1110-1116.