ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.4, April 2010, pp.702−717
doi: 10.3724/SP.J.1001.2010.03609

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

基于整体和局部相似性的序列聚类算法*

戴东波+, 汤春蕾 熊 赟

(复旦大学 计算机科学技术学院,上海 200433)

Sequence Clustering Algorithms Based on Global and Local Similarity

DAI Dong-Bo+, TANG Chun-Lei, XIONG Yun

(School of Computer Science and Technology, Fudan University, Shanghai 200433, China)

+ Corresponding author: E-mail: daidongbo@fudan.edu.cn

Dai DB, Tang CL, Xiong Y. Sequence clustering algorithms based on global and local similarity. *Journal of Software*, 2010,21(4):702−717. http://www.jos.org.cn/1000-9825/3609.htm

Abstract: Many current sequence clustering algorithms are based on the hypothesis that sequence can be characterized by its local features, without differentiating between global similarity and local similarity of sequences in different applications, which is applicable to biological sequences such as DNA and protein with conserved sub-patterns. However, in some domains such as the comparison of customers' purchase behaviors in retail transaction database and the pattern match in time series data, due to difficulties in forming frequent sub-pattern, it is more reasonable to cluster these sequence data based on global similarity. Besides, among sequence clustering algorithms based on local similarity, the ability that sub-patterns characterize sequence should be improved. So, this paper proposes two clustering algorithms, GSClu (global similarity clustering) and LSClu (local similarity clustering), for different application fields, based on global and local similarity respectively. GSClu uses bisecting *k*-means technique and CSClu adopts sub-patterns with gap constraint to cluster the sequence data of corresponding application field. Sequence data in the experiments include retail transaction data and protein data. The experimental results show that GSClu and LSClu are of fast processing rate and high clustering quality.

Key words: sequence data; similarity; clustering

摘 要: 现有的很多序列聚类算法是基于"局部特征可以表征整个序列"的假设来进行的,即不区分实际应用中序列的整体相似性和局部相似性.这对存在保守子模式的序列,如 DNA 和蛋白质序列是适用的,但对一些注重整体序列相似性的应用领域,如:在交易数据库中用户购买行为的比较,时间序列数据中全局模式的匹配等,由于难以产生频繁子模式,用基于全局相似性的度量方法进行聚类显得更为合理.此外,在基于局部相似性的序列聚类算法中,选取的局部子模式表征序列的能力也有待进一步提高.由此,针对不同应用领域,分别提出基于整体相似性的序列聚类算法 GSClu 和基于局部相似性的序列聚类算法 LSClu.GSClu 和 LSClu 分别利用带剪枝策略的二分 *k* 均值算法和基于有 gap 约束的强区分度子模式方法对各自领域的序列数据进行聚类.实验采用交易序列数据和蛋白质序列数据,实验结果表明,GSClu 和 LSClu 对各自领域的序列数据具有较快的处理速度和良好的聚类质量.

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2006AA02Z329 (国家高技术研究发展计划(863))

Received 2008-07-09; Revised 2008-12-18; Accepted 2009-02-24

关键词: 序列数据;相似性;聚类
中图法分类号: TP311 **文献标识码**: A

序列数据是一种很重要的数据类型,广泛存在于各个领域中,如:商业领域的用户交易数据和时间序列数据、媒体领域的语音序列和文本序列、Web 领域的用户访问序列、生物信息中的 DNA 序列和蛋白质序列等[1−3].对序列数据进行聚类分析可以发现数据潜在的结构和知识,如:对交易序列数据聚类可以对客户进行划分,从而制定不同的有针对性的市场广告,对蛋白质序列进行聚类可以帮助发现各个簇中序列共享的子结构,从而推测共有的生物学功能.由于序列数据具有非数值类型、高维且长度不一以及特有的序关系等特征,使得对其进行聚类成为一个研究难点.

聚类的一个核心问题就是如何合理地定义相似性度量函数,在现有的各种应用需求中,序列数据的相似性可以分为两种:

(1) 基于整体的序列相似性.在如交易数据库和语音序列等长序列数据中,序列的特征由整个序列来表征,这时考察序列的相似性必须由整体相似性出发.即使两条序列有非常相似的局部子序列,如整体相似性水平较低,则仍认为两条序列是不相似的.例如有 5 个客户 $C_1,C_2,\ldots,C_5$ 在不同时间点(设每个时间点只购买一项商品)的购买商品序列如图 1 所示.若按基于局部频繁子模式挖掘算法,客户 C_3 的购买行为和客户 C_4,C_5 是相似的,因为他们的购买商品序列有共同的频繁子模式 IJ(频繁度阈值为 3).但因客户 C_3 与 C_2,C_3 与 C_1,C_1 与 C_2 在对应时间点的相同购买商品数分别为 4,3,3,所以 C_1,C_2 和 C_3 的购买行为在整体上更相似.然而客户 C_1,C_2 和 C_3 的购买序列之间没有任何长度的频繁子模式,基于局部频繁子模式的方法挖掘不到这种整体的购买相似性行为.这说明,在不同的两两序列之间,整体相似往往由不同的相似子序列来体现,由于这些相似的子序列产生的“信号”太弱,难以形成频繁的子模式(在序列长度较长和频繁度阈值较高的情况下尤其如此),使得基于局部相似性的挖掘算法不能捕捉到序列的整体相似性.

Customer C_1's purchase sequence :U	O	V	D	W	F	X	H	Y	Z
Customer C_2's purchase sequence: A	O	C	P	E	Q	G	Y	Y	Z
Customer C_3's purchase sequence: A	B	C	D	E	F	G	H	I	J
Customer C_4's purchase sequence: R	R	R	R	K	K	K	K	I	J
Customer C_5's purchase sequence: T	T	T	T	S	S	S	S	I	J

Fig.1 Customers' purchase sequences

图 1 客户的购买序列

>SPR3…… CTGGTCGTAATACAAATAGAAGAGGTAAACCAATCAATGGCCCGTTAGTTTGCCATT……

>COX6……AGGCTATACTGATGGCCGTATCGCTCCATACGAGCCAATCAGGGCCCCGCGCGTTA……

>QCR8……TGACTAGTCCAAGGATTTTTTTTAAGCCAATTAAAATGAAGAAATGCGTGATCGG……

>CYC1……GGGCTTGATCCACCAACCAACGCTCGCCAAATGAACTGGCGCTTTGGTCTTCT……

Fig.2 Common sub-pattern in the promoter regions of genes

图 2 基因启动区共同的子模式

(2) 基于局部的序列相似性.在生物信息学领域,分子序列,如 DNA 或蛋白质存在局部的保守功能片段,这些功能片段是最能体现整个序列性质的“核心”部位.这时局部片段作为序列的关键特征,可以用来度量序列相似性水平.图 2 表示共调控的 4 个基因(4 个基因分别为 SPR3,COX6,QCR8 和 CYC1)上游启动区的一些序列片段,尽管这些序列在整体上相似度很低,但它们有共同的子模式 CCAA- - A(生物信息领域称结合位点,“-”表示任意字符).这个共同的子模式使得这些序列都有共同的基因表达调控机制,所以认为它们都是相似的.亦即对基于局部相似性的序列聚类算法,关键就是要识别表征不同序列的“核心”子模式来作为序列的特征,从而在子模式的基础上定义序列的相似性度量函数.

本文针对不同应用领域,分别提出了相应的序列聚类算法,主要贡献是:1) 不同于以往只注重局部相似性的序列聚类算法,本文区分了不同应用对序列数据的相似性要求,分别提出了针对整体相似性和局部相似性的序列聚类算法,使得不同应用可选用各自需求的聚类算法;2) 在对基于整体相似性序列聚类算法中,由于采用能够较好度量整体相似性的编辑距离[4],为了克服其计算复杂度较高的缺点,本文提出了一种带剪枝策略的二分 k 均值的聚类算法 GSClu,可避免不必要的编辑距离计算或只需计算部分的编辑距离,并且可以通过启发式方法求出簇的质心,从而有效地提高算法性能;3) 设计了局部相似性序列聚类算法 LSClu,为了更好地体现应用需求,对子模式挖掘加入了 gap 约束(即子模式各元素间距离约束),并设计了一种改进的度量子模式区分不同序列能力的函数,可以进一步提高序列聚类质量.

本文第 1 节介绍相关工作.第 2 节和第 3 节分别给出基于整体相似和局部相似性的序列聚类算法 GSClu 和 LSClu 的实现过程.第 4 节给出实验结果及其分析.第 5 节对全文进行总结并给出后续研究方向.

1 相关工作

分类属性数据的聚类[5−7]已被广泛地加以研究,序列数据中的各元素一般也是分类属性,与本文工作相近的有两大类:一类是文本聚类[8−10],另一类是简单交易数据聚类[7].虽然文本在本质上是单词集合上的序列数据,但在文献[8−10]中,都是把文本看成是一个“词包”,不考虑文本中单词之间的顺序依赖关系,从而聚类形成的文本簇在语义上的相关性不是很强.文献[7]定义了一种共享最近邻的相似度度量方法,使得相似性函数综合考虑了局部和整体的数据信息.但其聚类算法针对的是简单的交易数据,即只考虑客户在一个时间点上的购买商品集,这时用户购买的商品项之间的序关系并不重要,所以本质上仍然不是序列数据的聚类方法.

当前一些序列聚类算法大多是针对局部相似性来进行的.Chaudhuri 等人认为,DNA 中“关键”的 DNA 词频可以给整条 DNA 序列做一个统计概貌,然后将 DNA 序列间的相似性转化为这些关键“词频”统计分布的相似性[11,12].这对 DNA 分析领域是合适的,但单纯地基于频数统计还是比较粗糙的,因为 DNA 词的频数指标不能反映 DNA 词区分不同序列的能力.文献[13]使用频繁子序列来描述序列特征,用凝聚的层次聚类方法来实现序列的聚类,其中,簇之间的相似性是基于各簇共享的子序列来度量的.Wang 等人认为序列的序关系和各元素之间的依赖关系在序列数据中是十分重要的,他们提出的 CLUSEQ 方法[14]使用了在一个序列片段已知的情况下,下一个字符出现条件概率来作为度量序列相似性的基础,并使用概率后缀树来组织一个簇中的序列条件概率分布.该方法能够自动调节序列中簇的数目,并且可以有效地处理异常点.Guralink 等人[15]把整个序列投影到一个向量空间,其中向量属性集就是找到的频繁子序列模式,然后应用 k-means 的方法对这些代表序列数据的向量进行聚类.由于此方法找到的频繁子模式一般都很多,所以表征每条序列的向量是高维且稀疏的,这会影响聚类质量.为了克服这些缺点,文献[16]也使用频繁子序列的基本方法,但只选用一条最能区分不同序列能力的频繁子序列来作为这条序列的特征,使得聚类效率和质量有所提高.与本文提出的基于局部相似性的聚类方法 LSClu 相近的是文献[16]中的 CONTOUR 算法,但本文是基于 gap 约束的频繁子模式,更符合实际应用模型,且对频繁子序列的区分不同序列能力的度量函数有所改进,能够进一步提高聚类质量.

在文献[17,18]中的序列聚类算法中,由于编辑距离的计算复杂性太高,所以文献[17,18]中都避免使用编辑距离来作为相似性度量函数而采用其他启发式方法,这样便不能很好地反映序列的全局相似性.文献[17]把每条 DNA 序列用一个反映全局序列信息的 12 维向量表示,其中包括每个字符的出现次数、每个字符到序列首字符的距离之和以及每个字符到序列首字符距离的方差和.然后应用标准向量聚类算法进行聚类.但向量间的相似性不能准确刻画序列间的相似性,因为不相似的序列可能会映射到向量空间很近的两个数据点.在文献[18]中,作者用赋予不同权重给每级共享前缀的方法来度量两个 URL 序列的相似性,这与 URL 具有地址分层的特性是相符的,但不考虑插入和删除操作,这对度量 URL 序列全局的相似性是不够的.编辑距离是刻画序列整体相似水平的一个有效度量函数,本文提出的基于整体相似性的序列算法 GSClu 是基于编辑距离的,通过找到编辑距离的上下界以及利用等长前缀子序列编辑距离的非递减性质来进行剪枝操作,可以高效地对序列进行聚类.

2 基于整体相似性的序列聚类算法 GSClu

2.1 相关概念和性质

定义 1. 一条序列 S 是字母表$\Sigma=(a_1,a_2,\ldots,a_T)$上的有序字符串,记做 $S=s_1s_2\ldots s_n,s_k\in\Sigma(1\le k\le n)$称为 S 的元素,其中 $T=|\Sigma|$称为字母表的基数,$n=|S|$称做 S 的长度.

定义 2. 设序列 $S=s_1s_2\ldots s_n,n_i$ 是字母表中字母 a_i 在 S 中的出现次数$(1\le i\le T)$,则称 $SN(S)=(n_1,n_2,\ldots,n_T)$是序列 S 的签名.若有序列 S_1 和 S_2,其各自的签名为 $SN(S_1)=(n_1^1,n_2^1,...,n_T^1)$, $SN(S_2)=(n_1^2,n_2^2,...,n_T^2)$, 则称函数 $SD(S_1,S_2)=\max\left(\sum_{i=1}^{T}I_i^P(n_i^1-n_i^2),\sum_{j=1}^{T}I_j^N(n_j^2-n_j^1)\right)$为序列 S_1 和 S_2 的签名距离,其中 I_i^P 和 I_j^N 分别为示性函数,当 $n_i^1>n_i^2$ 时, $I_i^P=1$, 否则 $I_i^P=0$;当 $n_j^2>n_j^1$ 时, $I_j^N=1$,否则 $I_j^N=0$.

两条序列 S_1 和 S_2 的签名距离只反映了 S_1 和 S_2 在字母组成上的差异,没有考虑序列中字母的序关系.S_1 和 S_2 的编辑距离 $ED(S_1,S_2)$定义为将 S_1 映射到 S_2 所需要的最少操作步骤,其中的操作包括插入、删除和替代[19]. 计算 S_1 和 S_2 的编辑距离的基本方法是应用动态规划方法,将 S_1 和 S_2 的每个局部片段的编辑距离存储在一张最优比对表中,$ED(S_1,S_2)$就是表中的最后一个元素值.下面的定理给出了序列的签名距离和编辑距离的关系,这给后面的序列聚类算法中的剪枝策略提供了理论依据.

定理 1. 设有两条序列 S_1 和 S_2,则值 $SD(S_1,S_2)$和$|S_1|+|S_2|$分别是 $ED(S_1,S_2)$的下界和上界,即 $SD(S_1,S_2)\le ED(S_1,S_2)\le|S_1|+|S_2|$.

证明:设序列 S_1 和 S_2 的签名分别为 $SN(S_1)=(n_1^1,n_2^1,...,n_T^1)$ 和 $SN(S_2)=(n_1^2,n_2^2,...,n_T^2)$.不失一般性,可设 $SN(S_1)$ 向量中只有前 k 个项比 $SN(S_2)$中的对应前 k 个项要小, $n_1^1<n_1^2,...,n_k^1<n_k^2,n_{k+1}^1\ge n_{k+1}^2,...,n_T^1\ge n_T^2$,且设 $\left(\sum_{i=1}^{k}n_i^2-n_i^1\right)\ge\left(\sum_{j=k+1}^{T}n_j^1-n_j^2\right)$,则 $SD(S_1,S_2)=\left(\sum_{i=1}^{k}n_i^2-n_i^1\right)$.$S_1$ 经过 $ED(S_1,S_2)$步插入、删除或替代操作后变为 S_2,则在 $ED(S_1,S_2)$步操作中,肯定存在一些操作要包括把 S_1 中的字母 $a_1,a_2,\ldots,a_k$ 的出现次数 $n_1^1,n_2^1,...,n_k^1$ 增加为 $n_1^2,n_2^2,...,n_k^2$. 因为增加字母出现次数的操作是插入操作和替代操作,且每个插入操作使得字母出现次数增 1,而替代操作使得一个字母的出现次数增 1,另一个字母的出现次数减 1.所以,在 $ED(S_1,S_2)$步操作中,S_1 至少要经过 $SD(S_1,S_2)=\left(\sum_{i=1}^{k}n_i^2-n_i^1\right)$步插入或替代操作才能转化为 S_2,即 $SD(S_1,S_2)\le ED(S_1,S_2)$.

在 S_1 和 S_2 的最优比对表中,当进行回溯求解将 S_1 变为 S_2 的操作序列时,从表的(m,n)位置开始(其中 $m=|S_1|$, $n=|S_1|$),回溯到位置(0,0)终止,每次回溯要么是 m 减 1,要么是 n 减 1,要么是 m 和 n 同时减 1,所以最多经过 $m+n-2$(即$|S_1|+|S_1|$)步可以回溯到位置(0, 0).即 $ED(S_1,S_2)\le|S_1|+|S_2|$.

综上所述,$SD(S_1,S_2)\le ED(S_1,S_2)\le|S_1|+|S_2|$. □

由定理 1,易得出下面的推论:

推论 1. 设有 3 条序列 S_1,S_2 和 S_3,若 $SD(S_1,S_2)\ge|S_1|+|S_3|$,则 $ED(S_1,S_2)\ge ED(S_1,S_3)$.

由编辑距离的定义易知,两条序列的编辑距离是一个度量,所以满足三角不等式,从而有以下定理:

定理 2. 设有 3 条序列 S_1,S_2 和 S_3,已知 S_1 和 S_2 的编辑距离为 $ED(S_1,S_2)$,若 $ED(S_1,S_2)\ge2\times(|S_1|+|S_3|)$,则 $ED(S_1,S_3)\le ED(S_2,S_3)$.

证明:由于$|S_1|+|S_3|$是 $ED(S_1,S_3)$的上界,所以有 $ED(S_1,S_2)\ge2\times(|S_1|+|S_3|)\ge2\times ED(S_1,S_3)$,即 $ED(S_1,S_2)-ED(S_1,S_3)\ge ED(S_1,S_3)$.因为编辑距离满足三角不等式,所以 $ED(S_2,S_3)\ge ED(S_1,S_2)-ED(S_1,S_3)$,从而有 $ED(S_1,S_3)\le ED(S_2,S_3)$. □

因为对于长度分别为 m 和 n 的序列,它们之间编辑距离计算的时间开销为 $O(m\times n)$,而计算两条序列的签名距离和长度的时间开销都是 $O(m+n)$,若推论 1 和定理 2 中的前提条件满足,则可不必计算编辑距离,从而可以高效地进行编辑距离大小的比较.若推论 1 和定理 2 中的前提条件不满足,我们有可能只要进行编辑距离的部分

计算来进行大小比较.当对序列 $S=s_1s_2\ldots s_m$ 和 $P=p_1p_2\ldots p_n$ 进行比对求其编辑距离时,设其最优比对表为 $T(n+1$ 行和 $m+1$ 列),则有下面性质:

性质 1. $\left|T(i,j)-T(i-\Delta_i,j-\Delta_j)\right|\le 1$.其中,$T(i,j)$表示表 T 中第 i 行和第 j 列的值,$0\le i\le n$,$0\le j\le m$,$\Delta_i=0$ 或 1,$\Delta_j=0$ 或 1,且 $i-\Delta_i\ge 0, j-\Delta_j\ge 0$.

证明:最优比对表的生成过程是:$T(0,j)=j$,$T(i,0)=i(0\le i\le n,0\le j\le m)$和递归式 $T(i,j)=\min\begin{cases}T(i-1,j-1)+P(s_i,p_j)\\T(i,j-1)+1\\T(i-1,j)+1\end{cases}$ $(i\ge 1$,且$j\ge 1)$, 其中,如果 $s_i=p_j$,则 $P(s_i,p_j)=0$,否则,$P(s_i,p_j)=1$.由归纳法易得证. □

性质 1 反映了在生成的最优比对表中,任何一个值与其左边、左上角和上边近邻的差值的绝对值不超过 1,表 1 所示的序列 writers 和 vintner 的最优比对表反映了性质 1.这个性质可以用来证明下面的定理 3.

Table 1 Optimal alignment table of sequence writers and vintner

表 1 序列 writers 和 vintner 的最优比对表

$T(i,j)$	–	*w*	*r*	*i*	*t*	*e*	*r*	*s*
–	0	1	2	3	4	5	6	7
v	1	1	2	3	4	5	6	7
i	2	2	2	2	3	4	5	6
n	3	3	3	3	3	4	5	6
t	4	4	4	4	3	4	5	6
n	5	5	5	5	4	4	5	6
e	6	6	6	6	5	4	5	6
r	7	7	6	7	6	5	4	5

定理 3. 设序列 $S=s_1s_2\ldots s_m$ 和 $P=p_1p_2\ldots p_n(m\ge n)$的最优比对表为 T,则当 $i\le j\le n$ 时,有 $T(i,i)\le T(j,j)$.

证明:我们证明 $T(i,i)\le T(i+1,i+1)$,即可证明原定理.由表 T 的生成过程:

(1) 若 $T(i+1,i+1)=T(i,i)+P(s_i,t_i)$,由 $P(s_i,t_i)=0$ 或 1,有 $T(i,i)\le T(i+1,i+1)$.

(2) 若 $T(i+1,i+1)=T(i+1,i)+1$,由性质 1,$|T(i+1,i)-T(i,i)|\le 1$,分 3 种情况:① $T(i+1,i)-T(i,i)=1$,$T(i+1,i+1)=T(i,i)+2$,$T(i,i)<T(i+1,i+1)$.② $T(i+1,i)-T(i,i)=0$,$T(i+1,i+1)=T(i,i)+1$,$T(i,i)<T(i+1,i+1)$.③ $T(i+1,i)-T(i,i)=-1$,$T(i+1,i+1)=T(i,i)$.

(3) 若 $T(i+1,i+1)=T(i,i+1)+1$,证明与(2)类似,也有 $T(i,i)\le T(i+1,i+1)$.

综合上述(1),(2)和(3),有 $T(i,i)\le T(i+1,i+1)$. □

定理 3 表明, 在两序列的全局比对中,其局部等长前缀子序列的编辑距离具有非递减性质.如表 1 中两个不同阴影部分所示,有 $ED(wr,vi)\le ED(wri,vin)$.对于不同长度的序列,下面的定理 4 给出了在进行其编辑距离的比较时,其中一个编辑距离只需部分求出即可进行比较.

定理 4. 设有序列 $S=s_1s_2\ldots s_m$,$P=p_1p_2\ldots p_n(m\ge n)$和 Q,且已知 $ED(S,Q)$,若存在某个 $k\le n$,使得 $ED(s_1s_2\ldots s_k,p_1p_2\ldots p_k)-(m-n)\ge ED(S,Q)$成立,则有 $ED(S,P)\ge ED(S,Q)$.

证明:设 S 和 P 的最优比对表为 T,由性质 1,$T(n,m)+1\ge T(n,m-1)$,$T(n,m-1)+1\ge T(n,m-2)$,…,$T(n,n+1)+1\ge T(n,n)$.即 $T(n,m)\ge T(n,n)-(m-n)$.而由定理 3 可知,当 $k\le n$ 时,$T(n,n)\ge T(k,k)$.所以当存在某个 $k\le n$,使得 $ED(s_1s_2\ldots s_k,p_1p_2\ldots p_k)-(m-n)\ge ED(S,Q)$成立时,有 $ED(S,P)=T(n,m)\ge T(n,n)-(m-n)\ge T(k,k)-(m-n)\ge ED(S,Q)$. □

定理 4 表明,只要部分地计算 $ED(S,P)$,即可与 $ED(S,Q)$进行比较.如表 1 所示,若已知序列 vintner 和某序列 P 的 $ED(\text{vintner},P)$小于 3,只需计算 $ED(wri,vin)$(其值为 3),表中虚线部分的值无需计算即可推知 $ED(vintner,writers)\ge ED(vintner,P)$.

2.2 序列聚类算法GSClu的实现过程

给定 t 条平均长度为 L 的序列 $S_1,S_2,\ldots,S_t$,在编辑距离作为序列相似性度量的前提下,若采用基于两两比较的方法来进行聚类(如基于凝聚的层次聚类算法),算法时间复杂度为 $O(t^2L^2)$,在一般应用中,t 和 L 都很大,这对算法的执行效率影响很大,由此我们可以从以下几个方面来考虑加速算法的运行:

① 在算法的选择上,我们选用二分 k 均值算法.二分 k 均值算法最先是在文本聚类中提出来的[20],它避免了两两数据点之间的距离计算,时间复杂度是与数据集大小呈线性增长的.二分 k 均值基本算法如下所示:

二分 k 均值算法:

把所有数据点初始化为一个簇加入簇表.(1) 从簇表中选出一个总体相似度水平最低的簇 C;(2) 利用 k-menas 方法二分簇 C 为 C_1 和 C_2;(3) 将 C_1 和 C_2 加入簇表中.

重复(1)~(3)步,直到簇表中有 k 个簇为止.

当应用二分 k 均值方法来聚类现有的序列数据时,由于其中用到了基本的 k-means 算法,所以簇间数据点的移动和重新计算簇内质心称为核心问题.簇间数据点的移动涉及到每条序列数据与两个质心之间编辑距离的计算,如何利用剪枝策略高效地移动簇间数据点在下面的②和③中会加以讨论,这里给出如何计算簇内质心的启发式方法.对于序列数据组成的簇 C,使得函数 $\sum_{S_i\in C} ED(S_i,S)$ 最小的序列 S 可定义为簇 C 内的中心点[19].由于没有有效的方法来找到 S,可在 C 中选一条序列 S_m 来近似 S,其中 S_m 使得 $\sum_{S_i\in C} ED(S_i,S_m)$ 最小.文献[19]已证明,$\sum_{S_i\in C} ED(S_i,S_m)\le\left(2-\frac{2}{|C|}\right)\times\sum_{S_i\in C} ED(S_i,S)$, 所以 S_m 是 S 的一个有界近似.由于找到 S_m 要进行 C 中所有序列两两编辑距离的计算,所以会使得二分 k 均值算法关于序列数 t 仍是二次项时间复杂度的.下面给出一种启发式的质心求解方法,关于序列数 t 是多项式时间复杂度的.

假设簇 C 中的 $t'(t'<t$=条序列为 $S_1,S_2,\ldots S_{t'}$,我们先创建一张$(T+1)\times Q$ 的表 *Tab*,其中$(T+1)$行表示字母表中的 T 个字母和空格符"–"(在比对中表示插入一个字母),Q 列对应当前比对结果.*Tab*[*i*,*j*]表示比对结果中第 j 列中的第 i 个字母在此列中的出现次数. 我们用著名的 Needleman-Wunsch 方法[21]先比对 S_1 和 S_2,将结果填入表 *Tab* 中,然后比对表 *Tab* 和 S_3,接着更新表 *Tab*,继续比对 *Tab* 和 S_4,再更新表 *Tab*,…,直到最后比对完序列 $S_{t'}$,得到更新的表 *Tab*.如表 2 所示,假设字母表是$\Sigma=\{a,b,c\}$,当前比对的结果放在 *Tab* 表中.假设 *Tab* 表和下一条序列 aabbc 进行比对,比对方法仍可用 Needleman-Wunsch 方法,只是把 *Tab* 表的每一列当成一个字母,比对距离是序列 aabbc 中每个字母与 *Tab* 中每列的每个字母比对的加权之和.如图 3 所示,在 *Tab* 表和序列 aabbc 比对结果中 a 和 *Tab* 表中第 1 列比对,则这列比对距离是 0×0.75+1×0.25=0.25,因为在表 2 中的当前比对结果表 *Tab* 的第 1 列中,a 的出现概率是 0.75,c 的出现概率是 0.25.更新表结果只是在原表每列中增加所比对字母的计数,新增加的列也要加入表中.结果如表 3 中更新后的 *Tab* 表所示,其中第 2 列 C_2 是新添加的列,即字母 a 和空格符号"–"的比对结果,而第 1 列 C_1 的更新是在表 2 中 C_1 列的基础上,通过对 a 的出现次数增 1 的操作完成的.表 *Tab* 可以看作是当前序列比对的一个摘要(profile),它刻画了当前所有比对序列的"一致性"水平.

当 C 中所有序列比对完毕后,从最后更新的 *Tab* 表中提取一致序列作为 C 的质心,一致序列就是从 *Tab* 表每列中提取出现次数最多的字母组成的序列.从后面的实验可以看到,这样求得的近似质心能够比较准确地逼近有界质心 S_m.设簇 C 中逐步比对 *Tab* 表和序列的距离总和为 *T_Score*,*T_Score* 可用来估算簇的总体相似性水平.由于 *Tab* 表和序列比对距离是与序列长度相关的,所以可用规范化后的距离值 $C\_Sim=\frac{T\_Score}{t'L'}$ 来表示簇 C 中序列的总体相似性水平,其中 L'是簇 C 中各序列的平均长度,*C_Sim* 越大,C 中各序列的相似性水平越低.所以,在二分 k 均值算法的第 1 步,我们选择 *C_Sim* 最大的簇作为即将进行二分的候选簇.

Table 2 Current alignment result in table *Tab*
表 2 当前在 *Tab* 表中的比对结果

	C_1	C_2	C_3	C_4	C_5	C_6
a	4	1	0	1	0	2
b	0	0	4	0	4	0
c	1	0	1	2	0	2
–	0	1	0	2	1	1

Table 3 Updated table *Tab*
表 3 更新后的 *Tab* 表

	C_1	C_2	C_3	C_4	C_5
a	3	0	1	0	2
b	0	3	0	3	0
c	1	1	2	0	1
–	0	0	1	1	1

a	a	b	–	b	c
C_1	–	C_2	C_3	C_4	C_5

Fig.3 Alignment result between Table *Tab* and sequence aabbc

图 3 *Tab* 表与序列 aabbc 的比对结果

② 在处理簇间数据点的移动时,我们采取剪枝策略避免不必要的编辑距离计算.假设在应用二分 k 均值算法时,当前的两个质心是 C_1 和 C_2,p 点表示序列集中的任何一条序列,$ED(C_1,C_2)$已知.当找到离 p 点最近的簇时,需计算 $ED(p,C_1)$和 $ED(p,C_2)$,并进行大小比较.由第 2.1 节中的推论 1 和定理 2 可知:Ⅰ.当 $\max(SD(p,C_1),ED(C_1,C_2)/2)\geq|C_2|+|p|$时,有 $ED(p,C_2)\leq ED(p,C_1)$,p 点放入 C_2 所代表的簇.Ⅱ.当 $\max(SD(p,C_2),ED(C_1,C_2)/2)\geq|C_1|+|p|$时,有 $ED(p,C_1)\leq ED(p,C_2)$,p 点放入 C_1 所代表的簇.

若上述Ⅰ,Ⅱ条件不满足,需计算 $ED(p,C_1)$和 $ED(p,C_2)$并进行比较,但在进行编辑距离的直接计算时,我们也有可能没必要把整个编辑距离完全计算出来,这时有下面③的剪枝策略.

③ 对于序列 S,我们设计 S 的一个概貌向量 $V_S=\{n_1,n_2,\ldots,n_T,d_1,d_2,...,d_T\}$,其中 $n_i(1\leq i\leq T)$是 S 签名向量中的值,$d_i(1\leq i\leq T)$是 S 中字母 a_i 到序列首字母的距离之和.S 的概貌向量 V_S 反映了 S 中各字母组成和位置的基本情况.在 $ED(p,C_1)$和 $ED(p,C_2)$未计算之前,我们是不能准确知道 $ED(p,C_1)$和 $ED(p,C_2)$之间的大小的,但通过一遍扫描序列得到序列 p,C_1 和 C_2 各自的概貌向量以及 $L_1(V_p,V_{C_1})$和 $L_1(V_p,V_{C_2})$,其中 L_1 表示曼哈顿距离函数,V_p,V_{C_1} 和 V_{C_2} 分别表示序列 P,C_1 和 C_2 的概貌向量.若 $L_1(V_p,V_{C_1})\leq L_1(V_p,V_{C_2})$,由第 2.1 节中定理 4,则先计算 $ED(p,C_1)$,后计算 $ED(p,C_2)$时期望只需计算 p 和 C_2 的某个等长前缀子序列的编辑距离.尽管序列之间概貌向量的 L_1 距离不能等同于编辑距离,但当 $ED(p,C_1)$和 $ED(p,C_2)$相差较大时,它可有效指导编辑距离的计算次序,且 $ED(p,C_1)$和 $ED(p,C_2)$中较大者的计算一般会在较短的等长前缀子序列编辑距离计算完毕后停止.

综合上述①~③点,基于整体相似性的序列聚类算法 GSClu 如下所示:

GSClu 算法.

输入:t 条平均长度为 L 的序列集 $SS=\{S_1,S_2,\ldots,S_t\}$,参数 k;

输出:k 个序列簇.

```
{CL←SS;    CN=1;   //初始化簇表 CL 为 SS,开始只有一个全序列组成的簇 SS,CN 计数簇的个数
  for(i=1;i≤t;i++)
     一遍扫描序列集 SS,计算 Si 的签名 SN(Si)和概貌向量 VSi ;
  end for;
  while (CN<k)
     从簇表 CL 中选一个距离值 C_Sim 最大的簇 C;
     从 C 中选取最长序列和最短序列作为质心 CO1 和 CO2;
     计算出 ED(CO1,CO2), VCO1 和 VCO2 ;
     while (C 中所有序列不再在 CO1 和 CO2 所代表的簇间移动)
           for (C 中每条序列 S′)
              if ( max (SD(S′,CO1),ED(CO1,CO2)/2)≥|CO2|+|S′|)
                 序列 S′放入 CO2 所代表的簇 C2;
              else if (max(SD(S′,CO2),ED(CO1,CO2)/2)≥|CO1|+|S′|)
                       序列 S′放入 CO1 所代表的簇 C1;
                    else if (L1(VS′, VCO1 )≥L1(VS′, VCO2 ))
                             先计算 ED(S′,CO2);
                              若存在S′和 CO1 的等长前缀子序列满足定理4,则停止计算并把S′放入 CO2
                              所代表的簇 C2,否则继续计算完 ED(S′,CO1),然后决定 S′点的放入.
```

else

先计算 $ED(S',CO_1)$;

若存在 S'和 CO_2 的等长前缀子序列满足定理 4,则停止计算并把 S'放入 CO_1 所代表的簇 C_1,否则继续计算完 $ED(S',CO_2)$,然后决定 S'点的放入.

end for;

对簇 C_1 和 C_2 中的序列分别进行逐步比对求出 *Tab* 表,并从 *Tab* 表中提取一致序列更新各自的质心 CO_1 和 CO_2;

end while;

$CN=CN+1$;计算簇 C_1 和 C_2 各自的 *C_Sim* 值,并把 C_1 和 C_2 加入到簇表 *CL* 中;

end while;

}

因为在文献[22]中,*k*-means 算法的初始质心采用启发式方法:首先在数据集中随机选取一个质心,然后对于每个后继初始质心,选择离选取过的初始质心最远的数据点.所以在算法 GSClu 中,当用 *k*-means 方法二分选定的簇 *C* 时,我们不是随机初始化质心,而是分别选最长序列和最短序列来作为初始化质心.因为在一般情况下,这样操作比随机选取质心的方法更能加快算法的收敛.GSClu 的最坏时间复杂度是 $O(ITER\times k\times t\times L^2)$,*ITER* 是内循环 *k*-means 算法收敛时的平均循环次数.由于 GSClu 不采用基于序列两两比对的方法,且通过启发式方法获得质心,使得整个算法关于序列数 *t* 是多项式时间复杂度的.当序列数据集 *SS* 的自然分类情况比较明显时,在移动簇间数据点时采取的剪枝策略可以有效地加速算法的运行.

3 基于局部相似性的序列聚类算法 LSClu

现有很多基于局部相似性的序列聚类算法都采用频繁子模式的方法进行[13,15,16].下面先给出频繁子模式的定义:

定义 3. 设有长度分别为 *n* 和 *k* 的序列 $S=s_1s_2\ldots s_n$ 和 $S'=t_1t_2\ldots t_k$,如果 $n\geq k$,且存在有序整数集 $1\leq i_1<i_2\ldots<i_k\leq n$,使得 $t_1=s_{i_1},t_2=s_{i_2},\ldots,t_k=s_{i_k}$,则称 S'是 S 的子序列,也称 S 支持 S'(或 S'被 S 支持),记做 $S'\prec S$.若进一步地,对于任意的 $j(1\leq j\leq k-1)$,都有 $S_{i_{j+1}}-S_{i_j}\leq d+1$ 成立,则称 S'在 S 中满足 *d*-gap 约束.

定义 4. 设有序列集 $SS=S_1,S_2,\ldots,S_p$,给定一条序列 S 和支持度阈值 α,若 $\frac{\left|\{S'\mid S'\in SS,\text{且}S\prec S'\}\right|}{p}\geq\alpha$,则称序列 S 是序列集 *SS* 中的子模式.

子模式挖掘算法最先是在文献[23]中提出来的,其主要思想是先产生候选项,然后利用 Apriori 反单调性质进行剪枝.文献[24]给出了子模式的更高效挖掘算法 PrefixSpan.算法 PrefixSpan 为了避免大量候选项的产生和多次扫描整个数据库,先迭代地求出子模式的前缀,然后在前缀投影数据库中求出频繁项来逐步增长子模式.

文献[16]给出一种有效挖掘强区分度子序列(discriminating subsequence)的方法 CONTOUR,对给定序列数据库中的每条序列 *S*,只用 *S* 所支持的一条强区分度子模式序列来表征 *S*,然后在强区分度子序列的基础上定义序列的相似度函数,用层次凝聚的方法对序列数据库进行聚类.尽管算法 CONTOUR 改进了文献[15]中用多个子模式表征序列 *S* 的方法,在聚类质量和效率上有所提高,但仍存在以下两个方面的问题:

(1) CONTOUR 算法从序列数据库中找到的子模式不具有 gap 约束,即子模式中各元素在被支持序列中的距离可任意大,这不符合很多应用场合的要求,且降低了算法的执行效率.文献[25]中指出,很多实际中的序列数据都只有短程记忆性,即在一个字母 c 之前的子序列已知的前提下,c 在序列中出现的概率可近似认为只与这个子序列至多 *L* 长的后缀序列片段有关.此外,在生物序列中,许多功能片段在整个序列上也满足 gap 约束,即功能片段中各元素是处于某个局部区域的,而不会随机、均匀地散布在整个序列的各个位置中.如图 2 中所示的子序列 CCAA- - A 就满足 2-gap 约束.

(2) 用文献[24]中 PrefixSpan 方法为每条序列 $S=s_1s_2\ldots s_n$ 找到所支持的子模式有很多,CONTOUR 算法吸取

文本聚类中应用 TF-IDF 方法的思想,用函数值 $W_{S'}=\sum_{i=1}^{m}W_{t_i}=\sum_{i=1}^{m}\left(\frac{1}{\sup(t_i)}\right)^{\tau}$ (其中τ为加权区分因子,默认设置为 1)来表示 S 所支持的子模式 $S'=t_1t_2\ldots t_m$ 区分不同序列的能力,即 S'的区分度,其中 $sup(t_i)$表示支持字母 t_i 的序列数.CONTOUR 算法为 S 选取所支持的具有最大区分度的子模式 S'作为 S 的强区分度子序列.在一般的应用中,相对字母表的基数 T 来说,序列数据库中序列平均长度远远大于 T,这样每个字母几乎出现在每条序列中.如 DNA 序列定义在字母表$\Sigma=\{A,G,C,T\}$上,字母表基数是 4,对于很长的 DNA 序列,每个字符基本上都出现过.这样,对于字母表上的所有字母,$sup(t_i)$几乎都是相等的,所以在这种情况下,CONTOUR 算法选取的强区分度子序列实际上就是选择长度最长的子模式.

然而,在许多基于序列局部相似性的应用中,表征序列的子模式一般具有“相对位置偏好”的性质,即子模式中各元素在被支持的序列中相对间隔距离比较固定.下面给出一种定量度量“相对位置偏好”程度的定义.

定义 5. 设有序列集 SS 和 SS 集中的子模式 $S'=t_1t_2\ldots t_m$,且 SS 中支持 S'的序列集为 $S_1,S_2,\ldots,S_p$,S'中连续元素 t_i 和 t_{i-1} 在 S_j 中的相隔距离为 $d_{i,j}(2\le i\le m,1\le j\le p)$,则称 $S'_R=\frac{\sum_{i=2}^{m}\sum_{j=1}^{p}(d_{i,j}-\mu_i)^2/p}{m-1}=\frac{\sum_{i=2}^{m}\sum_{j=1}^{p}(d_{i,j}-\mu_i)^2}{p(m-1)}$ 为子模式 S'的规整度,其中 $\mu_i=\frac{\sum_{j=1}^{p}d_{i,j}}{p}$.子模式 S'规整度 S'_R 反映了 S'中各元素在被支持的序列集中相对间隔距离的波动程度,波动程度越大,S'_R 值越大,表明 S'中各元素相对位置偏好程度越弱,从而 S'表征被支持序列的能力越差,即 S'区分度越差.所以,S'的规整度也是衡量子模式区分度的一个因素.倾向选择最长子模式的 CONTOUR 算法没有考虑子模式规整度这个因素.如图 4 左边所示,设有 5 条序列 S_1,S_2,S_3,S_4,S_5 支持子模式 ABBA,其中“–”表示任意字符.假设 CONTOUR 算法为序列 S_1 选择了最长的 ABBA 作为强区分度的子模式,子模式 ABBA 的规整度是 3.07,图 4 右边表示序列 S_1 的支持子模式还有 ACB,支持 ACB 的其他序列是 S_7,S_3,S_4,S_9,子模式 ACB 的规整度是 0.2.从规整度角度来看,为序列 S_1 选择子模式 ACB 更合理,因为子模式 ACB 的相对位置偏好程度比 ABBA 要强,ACB 更有可能是一个重要的“功能片段”,而 ABBA 很有可能是一个随机模式,其表征序列的能力较差.

对于上述问题 1,当用户根据应用需要指定参数 d 时,最直接的方法就是先用 PrefixSpan 方法找出所有子模式,然后挑出满足 d-gap 约束的子模式.但这样会找到很多不满足 d-gap 约束的子模式,增加了算法的时间开销.我们扩展了文献[24]中的方法来找出所有满足 d-gap 约束的子模式及其规整度值,其算法见 GapPattern 算法.

S_1.	$A---B-B-----A$	S_1	$A--C----B$
S_2.	$A-B---B-A$	S_7	$A--C---B$
S_3.	$A--B--B---A$	S_3	$A---C---B$
S_4.	$A-B--B----------A$	S_4	$A--C----B$
S_5.	$AB-B--A$	S_9	$A--C---B$

Fig.4 Regularity of different sub-paterns

图 4 不同子模式的规整度

GapPattern 算法.

输入:gap 约束参数 d,支持度阈值α,序列集 $SS=S_1,S_2,\ldots,S_n$;

输出:所有满足 d-gap 和支持度α约束的子模式及其规整度值.

{返回所有在 *GetPattern*(〈〉,0,*SS*)中得到的子模式和规整度值;

//初始调用前缀投影为空,前缀长度为 0,投影数据库即为输入序列集 *SS*

}

GetPatern(*PF*,*L*,*SS*′)

{*FItemSet*←从 *SS′*集中找出频繁项;
If (*FItemSet* 不为空)
{**for** (*FItemSet* 中每个频繁项 *F*)
ST←*SS′*中支持 *F* 的序列集;
if (*PF* 不为空且 *F* 到 *ST* 中所有序列的首元素距离≤*d*−1)
PF′←把 *F* 连接到前缀 *PF* 之后;
D_{PF}←D_{PF}中 *ST* 集的距离∪*F* 到 *ST* 中所有序列的首元素距离;
PF'_R←根据 $D_{PF'}$中距离信息计算 *PF′*的规整度;
SP←*SP*∪*PF′*;
SP←*SP*∪*GetPatern*(*PF′*,*L*+1,*ProjectedS*(*PF′*,*SS′*));
else //GetPattern 函数初次被调用或 *F* 不满足距离要求而终止扩展,开始寻找下一个子模式//
SP←*SP*∪*F*;
初始化 *F* 的距离信息表 D_F 为 0;
SP←*SP*∪*GetPatern*(*F*,1,*ProjectedS*(*F*,*SS′*));
end for;
}
返回 *SP* 和 *SP* 中各子模式的规整度值;
}

算法利用递归调用 GetPattern 来得到所有满足要求的子模式及其规整度值.函数 *ProjectedS*(*PF*,*SS*)返回前缀 *PF* 在 *SS* 中的投影序列集.在投影序列集中,当满足距离要求时,前缀 *PF* 用频繁项 *F* 扩展形成 *PF′*.计算 *PF′* 的规整度需要利用 *PF′*在被支持的序列集中的距离信息表 $D_{PF'}$,因为支持 *PF′*的序列也一定支持 *PF*,所以可以通过前缀 *PF* 的距离信息表 D_{PF} 中支持 *PF′*的距离信息与频繁项 *F* 的距离信息来合成距离信息表 $D_{PF'}$.具体实现时,不必为 *PF′*建立单独的距离信息表,可以通过指针共享 D_{PF} 中支持 *PF′*的距离信息来增量生成距离信息表 $D_{PF'}$,这样可以节省存储开销.

对于上述问题 2,我们综合 $W_{S'}$和 S'_R 值来考虑子模式的 $S'=t_1t_2\ldots t_m$ 的区分度,其中 $W_{S'}=\sum_{i=1}^{m}W_{t_i}=\sum_{i=1}^{m}\left(\frac{1}{sup(t_i)}\right)^{\tau}$ (其中 τ 为加权区分因子,默认设置为 1),$W_{S'}$相当于文本聚类中的文档频率倒数 IDF 指标.我们用函数 $S'_d=\frac{W_{S'}}{S'_R+C}$ (其中 *C* 为常数,且 *C*>0)来度量子模式 *S′*的区分度水平.其中常量 *C* 是 S'_R 为 0 时的一个缓冲量.当序列 *S* 支持很多子模式时,序列 *S* 不是选择具有最大 $W_{S'}$值的子模式,而是选择具有最大 S'_d 值的子模式 *S′*来作为 *S* 的强区分度子序列.若在整个序列集 *SS* 中,含有子模式 *S′*中各元素的序列数越少(相当于 $W_{S'}$越大),且子模式 *S′*中各元素在 *S′*的被支持序列集中位置偏好程度越强,则此子模式 *S′*的区分度水平越高.

算法 LSClu 先用 GapPattern 方法得到输入序列集 *SS* 的子模式及其规整度值,然后对 *SS* 集中每条序列 *S* 所支持的子模式,选择区分度水平值 S'_d 最大的 *S′*来作为 *S* 的强区分度子模式.若序列 *S* 不支持任何子模式,则把 *S* 作为异常点来处理.*SS* 集中所有具有相同强区分度子模式的序列子集形成一个“微簇”.一般情况下,“微簇”个数要比用户指定的参数 *k* 要大,这时可以把“微簇”看成单个数据点,采用基于层次的聚类算法来得到用户想要的 *k* 个簇.初始数据点之间的相似度和中间要合并的簇之间的相似度的定义与文献[16]中类似,但在计算相似度值用到的子模式区分度是考虑了子模式规整度这个因素.基于局部相似性的序列聚类算法 LSClu 如下所示:

LSClu 算法.

输入:序列集 *SS*=$S_1,S_2,\ldots,S_n$,gap 约束参数 *d*,支持度阈值 α 和簇个数 *k*;
输出:序列集 *SS* 的 *k* 个簇.
{满足 *d*-gap 约束的子模式集 *SP*←*GapPattern*();
for (*SS* 集中的每条序列 *S*)
if (*S* 不支持 *SP* 中的任何子模式)

把 S 标记为异常点;
else
在 SP 中为 S 找到具有最大区分度值 S'_d 的子模式 S';
把 S 放入具有相同子模式 S'的“微簇”中;
end for;
现有簇个数 N←所得“微簇”个数;
while($N>k$)
选择现有的 N 个簇中具有最大相似度值 $SIM(C_1,C_2)$的两个簇 C_1 和 C_2 进行合并;
N←N–1;
end while;
}

GSClu 和 LSClu 都是序列聚类算法,若有先验的领域知识,可选用具有领域偏向性的聚类算法.如:在生物信息领域,若具有共同功能模式的生物序列所形成的簇比整体相似性水平较高的簇更有意义,这时选用 LSClu 算法较好.而在交易数据领域,较低规整度的子模式较少,整体相似性可能更有意义,这时适宜采用 GSClu 算法.当没有关于给定序列数据集的先验领域知识或领域知识不足时,评价算法的聚类质量可采用“内部质量”指标[15].设某个簇 C_j 中有 n_j 条序列,我们用 $CS_j=\frac{\sum_{S\in C_j,S'\in C_j} ED(S,S')}{n_j(n_j-1)}$ 值作为体现簇 C_j 紧凑程度的指标,CS_j 越小,簇的整体相似性水平越高,簇越紧凑.设算法对给定数据集生成 k 个簇,则加权距离之和为

$$WCS=\sum_{j=1}^{k} n_j \times CS_j \tag{1}$$

可用来衡量此算法的聚类质量,WCS 值越大,算法的聚类质量越差.这时,可把给定序列数据集在 GSClu 和 LSClu 算法上各运行一次,选择聚类指标 WCS 较低的作为聚类结果.

4 实验结果及其分析

所有实验在 2.0 GHz 的 PC 上进行,操作系统为 Windows 2000 XP,算法用 Java 语言实现.

4.1 算法GSClu的实验结果

类似文献[23]中实验所采用的方法,GSClu 算法实验数据集采用人工合成的交易序列数据.由于人工合成数据集可以生成任意长度和簇个数的组合序列数据,我们生成了一系列的数据集.生成人工数据集的方法如下:给定字母表(字母表基数都设为 10),先随机产生 k 条不同长度的种子序列代表 k 个序列簇,然后在每条种子序列的基础上,通过变化其长度和元素来生成簇中其他各序列.人工数据集用如下标记进行描述:K 表示生成序列的簇个数,C 表示数据集的大小,L 表示序列的基准长度,Δ表示不同簇中种子序列的长度变化,VL 表示相对种子序列长度同一簇中各序列长度变化百分率上限,VP 表示在种子序列基础上,同一簇中各序列变化的元素百分率上限.例如:$K5C5000L100\Delta50$ $VL5$ $VP10$ 表示数据集中有 5 000 条序列,分属 5 个不同的簇,由于基准长度是 100,且不同种子序列长度变化是 50,所以 5 条种子序列的长度是 100,150,200,250,300.每个簇都是在种子序列 S 的基础上,通过变化 S 长度的(包括在 S 中随机位置上增加或删除任意元素)至多 5%以及变化其元素(随机变化成字母表中其他字母)的至多 10%来产生簇中其他序列的.

据我们所知,现有关于序列聚类的算法没有专门针对整体相似性的,即无论是具有整体相似性还是局部相似性的序列数据,都是用统一的算法进行聚类.此外,CONTOUR 算法是最近性能和质量较好的算法之一,所以我们用 CONTOUR 和 GSClu 算法进行对比来验证 GSClu 算法对基于整体相似性序列的聚类有更好的聚类质量.我们采用第 3 节中式(1)的加权距离和指标 WCS 来衡量算法的聚类质量.在注重整体相似性的合成交易序列数据集上,GSClu 和 CONTOUR 算法的聚类质量比较如图 5 所示.

两种算法采用同样的 3 组合成交易序列集,如表 4 中的 3 组数据集 dataset1,dataset2 和 dataset3 所示.从图 5

可以看出,在 3 组数据集中,GSClu 算法聚类效果都优于 CONTOUR 算法.这主要是因为从 3 个数据集的随机生成过程可以看出,序列的相似性比较是基于整体相似性进行的,不存在能够表征整条序列特征的明显局部子模式,所以用基于局部相似性的 CONTOUR 算法找到的子模式集大多是随机模式,表征序列特征的能力较弱,从而导致进行聚类的质量较差.

Table 4 Three different synthetic datasets of transaction sequences

表 4 3 组不同的合成交易序列数据集

Dataset1	*K*3*C*2000*L*50Δ15*VL*10*VP*10
Dataset2	*K*5*C*5000*L*100Δ25*VL*12*VP*15
Dataset3	*K*10*C*8000*L*150Δ30*VL*15*VP*18

Table 5 Inner iterations of two different methods of selecting initial centroids

表 5 两种不同选择初始质心方法的内循环次数

	Random selection (RS)	Heuristic selection (HS)
CN=1	4	2
CN=2	5	3

我们也在真实的交易数据集 Retail 上比较了 GSClu 和 CONTOUR 算法.Retail 数据集来自比利时一个匿名超市的零售交易数据(http://fimi.cs.helsinki.fi/data/).该数据集含有 16 470 个商品项,共有 88 162 条交易记录.我们随机取其中 10 000 条交易记录作为算法数据集.GSClu 和 CONTOUR 算法的聚类质量如图 6 所示,从中可以看出,在簇的数量 *k* 取不同值的情况下,GSClu 算法都优于 CONTOUR 算法,这是因为 Retail 数据集的字母表较大,在一般支持度阈值的情况下,CONTOUR 算法得到的子模式规整度值较高且数量较少,即 Retail 数据集不存在明显的局部子模式,从而用基于子模式方法的 CONTOUR 算法的聚类效果较差.

对于 GSClu 算法中初始质心的取法,我们在表 4 中的 Dataset1 上比较了随机方法 RS 和 GSClu 启发式初始质心设置方法 HS,表 5 给出了在两种不同方法下 GSClu 算法每次增加一个新簇的内循环 while 次数(开始时是所有数据形成一个簇).从表中可以看出,GSClu 算法中启发式选取初始质心方法比随机选取方法更能加速算法的收敛.如当 *CN*=1,即开始所有数据为一个簇时,随机选取质心方法经过 4 次内循环把初始簇分为两个簇,而启发式选择初始质心方法只用了两次内循环.

我们通过对比不带剪枝策略的聚类算法来考察算法 GSClu 的聚类质量和执行效率.不带剪枝策略的聚类算法称为 Naïve_GSClu 和 Naïve_GSClu1.Naïve_GSClu 的具体实现方法如下:基本方法也是采用二分 *k* 均值,但数据点在簇间移动是基于完全编辑距离的计算来比较进行的,且求簇 *C* 中的质心是找到 C 中的有界质心,即在 *C* 中找一条序列 *S*,满足条件 $\min(\sum_{S_i \in C} ED(S_i,S))$.Naïve_GSClu1 与 Naïve_GSClu 不同的是,在求簇的质心时,是按 GSClu 中方法进行的.测试数据集见表 6.对所用的 6 组合成交易序列数据,我们固定住其他参数,让参数 *C* 从 500 变化到 10 000.其中 dataset6 的“自然”分类情况相对于前 5 组数据较差,其生成过程如下:先产生一条长度为 100 的随机序列 *S*,然后在 *S* 的基础上,通过变化其中元素的 50%来随机生成其他 4 条种子序列,然后按给定参数生成各簇中的其他序列.由于 dataset6 中子序列不是完全独立地随机生成的,所以相比前 5 组数据集,这里生成的 5 个簇的自然分离状况不是很明显.

Table 6 Six different synthetic datasets of transaction sequences

表 6 6 组不同的合成交易序列数据集

Dataset1	*K*5*C*500*L*100Δ15*VL*10*VP*10
Dataset2	*K*5*C*1000*L*100Δ15*VL*10*VP*10
Dataset3	*K*5*C*4000*L*100Δ15*VL*10*VP*10
Dataset4	*K*5*C*7000*L*100Δ15*VL*10*VP*10
Dataset5	*K*5*C*10000*L*100Δ15*VL*10*VP*10
Dataset6	*K*5*C*10000*L*100Δ15*VL*10*VP*10 (each seed sequence is produced dependently)

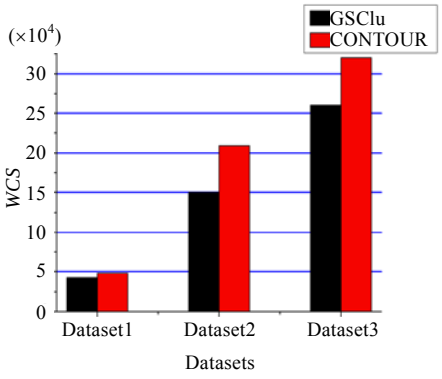

Fig.5 *WCS* comparison between GSClu and CONTOUR on synthetic dataset

图 5 GSClu 和 CONTOUR 的 *WCS* 比较(合成数据)

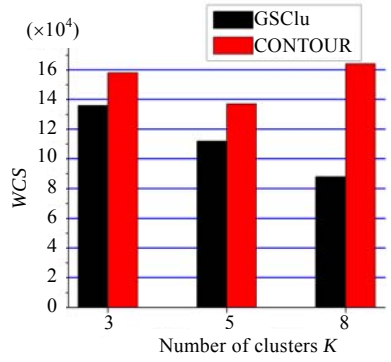

Fig.6 *WCS* comparison between GSClu and CONTOUR on real dataset

图 6 GSClu 和 CONTOUR 的 *WCS* 比较(真实数据)

由于 Naïve_GSClu 算法在数据集 dataset3,dataset4,dataset5 和 dataset6 中运行时间过长,我们只在 dataset1 和 dataset2 中运行了算法 Naïve_GSClu(运行时间分别为 4.85×10^4 s 和 1.92×10^5 s).GSClu 与 Naïve_GSClu 算法的聚类质量比较如图 7 所示,尽管 GSClu 是采用启发式方法来求得簇的近似质心,但在两组数据集中,其聚类质量指标 *WCS* 与 Naïve_GSClu 算法相差都不超过 8%,这表明,通过逐步比对方法,从最后的比对表中提取一致序列能够很好地逼近 Naïve_GSClu 算法中的有界质心.但因为 GSClu 在 dataset1 和 dataset2 中的运行时间分别为 87s 和 184s,所以算法 GSClu 相对于 Naïve_GSClu 算法的运行效率分别提高 557 倍和 1 042 倍.

GSClu 与 Naïve_GSClu1 算法的执行时间比较如图 8 所示.

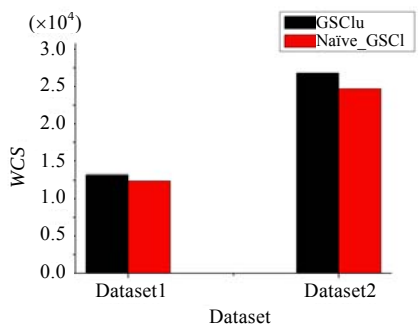

Fig.7 *WCS* comparison between GSClu and Naïve_GSClu

图 7 GSClu 与 Naïve_GSClu 的 *WCS* 比较

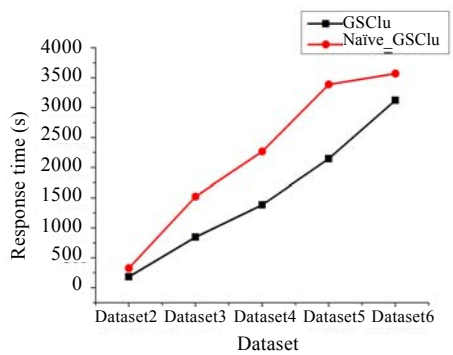

Fig.8 Time vs. number of sequences

图 8 处理时间随序列数的变化

前 4 组数据集是序列数等距增长的 dataset2,dataset3,dataset4 和 dataset5,从图 8 中可以看出,GSClu 算法处理时间关于序列数 *C* 几乎是呈线性增长的,这主要得益于 GSClu 采用了关于序列数线性复杂度的启发式算法.在 4 组数据集中,由于 GSClu 算法采用了剪枝策略,其执行时间都比 Naïve_GSClu1 要少.但在 dataset6 中,尽管序列数和 dataset5 一样,但由于自然分类情况较 dataset5 要差,致使剪枝策略前提条件满足的可能性较小,剪枝效果相对较差,从而运行时间比 dataset5 多了 977s.这表明,避免编辑距离计算和部分编辑距离计算的剪枝策略在数据的自然分类情况比较明显的情况下,剪枝效果较好.

4.2 算法LSClu的实验结果

测试算法 LSClu 执行效率和聚类质量的数据集是真实的蛋白质序列.我们从蛋白质家族数据库 Pfam(http://pfam.sanger.ac.uk//family)中选取球蛋白(Globins)、运输蛋白(ABC2_membrane)等 10 个蛋白质家族

中的序列,各家族序列条数和长度见表 7.

我们对比文献[16]中提出的 CONTOUR 算法.由于 CONTOUR 算法选取的特征子模式也一定是闭合子序列,所以文献[16]利用这一性质进行了剪枝优化.算法 LSClu 选取的强区分度子模式因为考虑了规整度这个因素,从而不一定是闭合子模式,但由于加进了 gap 约束,可以减少子模式数目,从而也可加速算法的运行.图 9 给出了算法 LSClu 和 CONTOUR 的执行效率比较情况.从图中可以看出,当 gap 约束 d=15 时,LSClu 的运行时间大于 CONTOUR,但当 d 分别为 10,5 和 2 时,LSClu 的执行效率比 CONTOUR 要高.这是因为 CONTOUR 虽然有剪枝优化,但无 gap 约束,即 d 可以任意大.从而当 d 较小时,LSClu 挖掘的子模式大为减少,节约的运行时间大于 CONTOUR 因为只挖掘闭合子模式而节约的运行时间,使得当 d 较小时,在各支持度阈值情况下,LSClu 的执行效率都优于 CONTOUR,并且在 gap 约束值 d 较小的情况下,也符合基于局部相似性的序列聚类要求.所以,尽管当 d 较大时,LSClu 的执行效率低于 CONTOUR,但对 LSClu 算法在实际应用中的影响较小.图 10 给出了在支持度阈值为 5%时,算法 GSClu 在不同 gap 约束值 d 下的子模式数量与 CONTOUR 算法子模式数量的对比情况,这也验证了图 9 中两种算法的时间性能情况.在其他阈值情况下,我们也得到了类似结果.

对于表 7 所示的数据集,因为已知类标号,我们可用熵来衡量一个聚类算法的聚类质量.对聚类算法生成的每个簇 C_j,先计算 C_j 中序列分布的熵值.计算公式如下:$E_j=-\sum_i P_{ij}\log P_{ij}$,其中 P_{ij} 等于原本属于簇 C_i 中的序列在簇 C_j 中所占的比例.算法聚类质量可用加权的熵值总和 $ECS=\sum_{j=1}^{k}\frac{n_j\times E_j}{n}$ 来衡量,其中 k 表示聚类所形成的 k 个簇,n_j 是簇 C_j 中的序列数,n 是所聚类的序列总数.ECS 越大,表明聚类算法的聚类质量越差,当加权熵 ECS 为 0 时,表示此聚类算法所产生的结果是一个完美聚类.

LSClu 和 CONTOUR 算法在表 7 所示实验数据集中的聚类质量比较见表 8,其中 gap 约束参数 d 设为 10,支持阈值都设为 0.7%.从表中可以看出,在所形成的 10 个簇中,LSClu 算法中 8 个簇的熵指标都优于 CONTOUR,且 LSClu 的整体聚类指标 ECS 也比 CONTOUR 要小,即整体聚类质量也优于 CONTOUR 算法.这主要是因为 LSClu 算法加进了规整度这个因素,增强了子模式表征序列的能力,从而在整体上提高了序列聚类的质量.

Table 7 Experimental data set (from Pfam)

表 7 实验数据集(来自 Pfam 蛋白质家族数据库)

Family description	Family Pfam No.	Number of sequences	Average length
Globin	PF00042	75	99.2
ABC2_membrane	PF01061	303	195
ADAM_CR	PF08516	70	110.6
ketoacyl-synt	PF00109	167	212.6
RWD	PF05773	129	116.4
PAD_porph	PF04371	22	330.9
Myco_haema	PF05692	32	352
Helicase_C	PF00271	492	80
Exo_endo_phos	PF03372	162	258
iPGM_N	PF06415	35	349.3

Table 8 Results for LSClu and CONTOUR

表 8 LSClu 和 CONTOUR 的运行结果

Protein family	Number of sequences	LSClu	CONTOUR
		Cluster entropy E_j	Cluster entropy E_j
Globin	75	0.071	0.089
ABC2_membrane	303	0.027	0.034
ADAM_CR	70	0.055	0.127
ketoacyl-synt	167	0.016	0.039
RWD	129	0	0.069
PAD_porph	22	0	0.224
Myco_haema	32	0.115	0.097
Helicase_C	492	0.018	0.006
Exo_endo_phos	162	0.005	0.029
iPGM_N	35	0.145	0.249
ECS		0.026	0.046

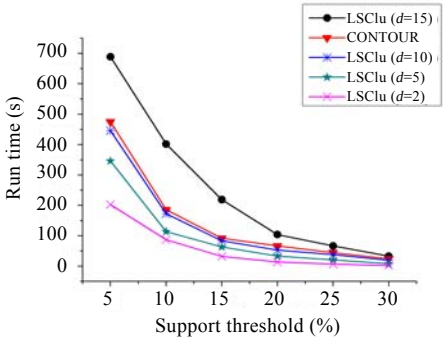

Fig.9 Time vs. support threshold
图 9 运行时间随支持度阈值的变化

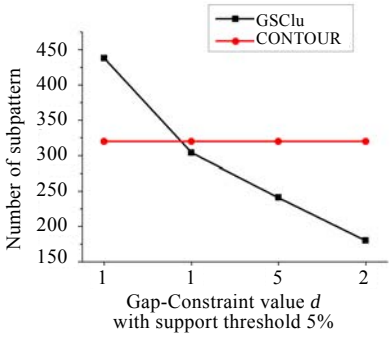

Fig.10 Number of sub-patterns vs. gap-constraints value *d*
图 10 子模式数量随着 gap 约束值 *d* 的变化

5 结论和展望

本文针对不同应用需求,分别基于序列的整体相似性和局部相似性提出序列聚类算法 GSClu 和 LSClu.算法 GSClu 通过应用二分 *k* 均值方法和启发式求解簇的质心,并应用剪枝策略高效地对序列进行聚类,实验结果表明,GSClu 的聚类质量接近算法 Naïve_GSClu,且在执行效率上明显优于 Naïve_GSClu 和 Naïve_GSClu1.算法 LSClu 考虑了 gap 约束和子模式规整度两个因素,符合实际应用需求,在实际应用中提高了算法效率和聚类质量.由于在 GSClu 算法中,最坏时间复杂度关于序列平均长度 *L* 仍是二次方的,我们今后将对应用随机近似方法进一步降低其复杂度问题进行研究.

References:

[1] Dong GZ, Pei J. Sequence Data Mining. Heidelberg: Springer-Verlag, 2007.

[2] Sarawagi S. Advanced Methods for Knowledge Discovery from Complex Data. Heidelberg: Springer-Verlag, 2005. 153−187

[3] Zhu YY, Xiong Y. DNA sequence data mining technique. Journal of Software, 2007,18(11):2766−2781 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

[4] Wanger RA, Fischer MJ. The string-to-string correction problem. Journal of the ACM, 1974,21(1):168−173. [doi: 10.1145/321796.321811]

[5] Ganti V, Gehrke J, Ramakrishnan R. CACTUS-Clustering categorical data using summaries. In: Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. San Diego: ACM Press, 1999. 73−83. http://doi.acm.org/10.1145/312129.312201

[6] Gibson D, Kleinberg JM, Raghavan P. Clustering categorical data: An approach based on dynamical systems. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the Int'l Conf. on Very Large Data Bases. New York: Morgan Kaufmann Publishers, 1998. 311−322.

[7] Guha S, Rastogi R, Shim K. ROCK: A robust clustering algorithm for categorical attributes. In: Proc. of the 15th Int'l Conf. on Data Engineering. Sydney: IEEE Computer Society, 1999. 512−521. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=754967

[8] Hatzivassiloglou V, Gravano L, Maganti A. An investigation of linguistic features and clustering algorithms for topic document clustering. In: Belkin NJ, Ingwersen P, Leong MK, eds. Proc. of the ACM SIGIR. Athens: ACM Press, 2000. 224−231.

[9] Slonum N, Tishby N. Document clustering using word clusters via the information bottleneck method. In: Belkin NJ, Ingwersen P, Leong MK, eds. Proc. of the ACM SIGIR. Athens: ACM Press, 2000. 208−215.

[10] Zamir O, Etzioni O. Web document clustering: A feasibility demonstration. In: Proc. of the ACM SIGIR. 1998. 46−54. http://portal.acm.org/citation.cfm?id=290956

[11] Chaudhuri P, Das S. SWORDS: A statistical tool for analyzing large DNA sequences. Journal of Biosciences, 2002,27(1):1−6. [doi: 10.1007/BF02703678]

[12] Chaudhuri P, Das S. Statistical analysis of large DNA sequences using distribution of DNA words. Current Science, 2001,80(9):1161−1166.

[13] Morzy T, Wojciechowski M. Scalable hirrarchical clustering method for sequences of categorical values. In: Cheung DW, Williams GJ, Li Q, eds. Proc. of the 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD). LNCS 2035, Hong Kong: Springer-Verlag, 2001. 282−293.

[14] Yang J, Wang W. CLUSEQ: Efficient and effective sequence clustering. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engneering. Bangalore: IEEE Computer Society, 2003. 101−112.

[15] Guralnik V, Karypis G. A scalable algorithm for clustering sequential data. In: Cercone N, Lin TY, Wu XD, eds. Proc. of the IEEE Int'l Conf. on Data Mining. Washington: IEEE Computer Society, 2001. 179−186.

[16] Wang JY, Zhang YZ, Zhou LZ, Karypis G, Aggarwal CC. Discriminating subsequence discovery for sequence clustering. In: Proc. of the 7th Int'l Conf. on Data Mining. Minneapolis: SIAM, 2007. 605−610. htt://www.siam.org/proceedings/datamining/2007/dm07_069wang.pdf

[17] Liu LB, Ho YK, Yau S. Clustering DNA sequences by feature vectors. Molecular Phylogenetics and Evolution, 2006,41(1):64−69. [doi: 10.1016/j.ympev.2006.05.019]

[18] Wang W, Zaiane OR. Clustering Web sessions by sequence alignment. In: Proc. of the 13th Int'l Workshop on Database and Expert Systems Applications(DEXA). Aix-en-Provence: IEEE Computer Society, 2002. 394−398. http://www.cs.ualberta.ca/~zaiane/postscript/dexa2

[19] Gusfield D. Algorithms on Strings, Trees, and Sequences. New York: Cambridge Press, 1997.

[20] Steinbach M, Karypis G, Kumar V. A comparison of document clustering techniques. In: Proc. of the KDD Workshop on Text Mining. 2000. http://rakaposhi.eas.asu.edu/cse494/notes/clustering-doccluster.pdf

[21] Needleman SB, Wunsch C. A general method applicable to search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology. 1970,48(3):443−453. [doi: 10.1016/0022-2836(70)90057-4]

[22] Tan PN, Steinbach M, Kumar V. Introduction to Data Mining. Addison-Wesley, 2005.

[23] Agrawal R, Srikant R. Mining sequential patterns. In: Yu PS, Chen ALP, eds. Proc. of the 11th Int'l Conf. on Data Engineering. Taipei: IEEE Computer Society, 1995. 3−14.

[24] Pei J, Han JW, Mortazavi-Asl B, Pinto H. PrefixSpan: Mining sequential patterns efficiently by prefix-projected growth. In: Proc. of the 17th Int'l Conf. on Data Engineering. IEEE Computer Society, 2001. 215−224. http://www.cs.uiuc.edu/homes/hanj/pdf/span01.pdf

[25] Ron D, Singer Y, Tishby N. The power of Amnesia: Learning probabilistic automata with variable memory length. Machine Learning, 1996,25(2-3):117−149. [doi: 10.1007/BF00114008]

附中文参考文献:

[3] 朱扬勇,熊赟.DNA 序列数据挖掘技术.软件学报,2007,18(11):2766−2781. http://www.jos.org.cn/1000-9825/18/2766.htm [doi: 10.1360/jos182766]

戴东波(1977−),男,湖南娄底人,博士,主要研究领域为数据挖掘,数据库,生物信息.

汤春蕾(1976−),女,博士生,主要研究领域为数据挖掘.

熊赟(1980−),女,博士,讲师,CCF 会员主要研究领域为数据挖掘,数据库,生物信息.