第48卷第4期 中国科学技术大学学报 Vol.48,No.4
2018年4月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Apr. 2018

文章编号:0253-2778(2018)04-0261-14

动态时间弯曲距离推导

邓 波,丁 鲲,蒋国权,张 宾

(南京电讯技术研究所,江苏南京 210007)

摘要:已有研究成果表明,在大多数时间序列处理应用领域中,动态时间弯曲是最为有效的相似度计算方法,但该方法计算时间复杂度较高,并且不满足距离三角不等式,无法进行快速推导.目前,动态时间弯曲优化方法集中在设计低计算复杂度的下界距离,以加快时间序列的比较,然而,这些下界距离同样不能推导,因此在相似度计算时都必须对时间序列数据进行逐一比较,导致I/O代价高,为此提出一种新颖的可推导动态时间弯曲近似距离以及相应的索引构建方法和相似时间序列查询算法.这是首次针对动态时间弯曲距离的推导问题的研究.大量实验结果表明,与现有方法相比,我们提出的方法在时间复杂度和I/O代价两方面都是高效的.

关键词:时间序列;动态时间弯曲;距离推导

中图分类号:TP311 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2018.04.001

引用格式:邓波,丁鲲,蒋国权,等.动态时间弯曲距离推导[J].中国科学技术大学学报,2018,48(4):261-274.
DENG Bo, DING Kun, JIANG Guoquan, et al. Deducing for dynamic time warping distance[J]. Journal of University of Science and Technology of China, 2018,48(4):261-274.

Deducing for dynamic time warping distance

DENG Bo, DING Kun, JIANG Guoquan, ZHANG Bin

(*Nanjing Telecommunication Technology Institute, Nanjing* 210007, *China*)

Abstract: The current research achievements show that the dynamic time warping (DTW) is the best measure in most area of time series similarity measurements. However, the high time complexity for calculating DTW distance directly, and the fact that DTW does not satisfy the triangle inequality, render it impossible to deduce TWD quickly. Nowadays DTW optimizing methods are mainly devoted to designing low time complexity DTW low bound distances with low time complexity to accelerate time series comparison. Unfortunately, these DTW low bound distances cannot be deduced, either. Therefore, it must be compared one by one to compute time series similarity, which has high I/O cost. A novel educible DTW low bound distance is thus proposed, along with a corresponding index building method and a similar time series query algorithm. It is the first research on the DTW deducing problem. Extended experiment results show that compared to current technologies, the proposed method is efficient in both time complexity and I/O cost.

Key words: time series; dynamic time warping; distance deducing

收稿日期:2017-05-22;**修回日期**:2017-06-24

基金项目:国家自然科学基金(61473001,71071045,71131002)资助.

作者简介:邓波,男,1975年生,博士/副研究员,研究方向:大数据和人工智能. E-mail: dengbomail@163.com

通讯作者:丁鲲,博士/研究员.E-mail:njdingkun@163.com

0 引言

时间序列分析和挖掘是近年来的研究热点，在语音处理[1]、模式识别[2]、环境监测[3]、医学[4]等领域应用广泛.距离测量是时间序列相似度计算最重要的任务之一，是解决时间序列查询、分类、聚类、异常检测等问题的基础[5-11].

研究表明，在大多数时间序列处理领域中，动态时间弯曲（dynamic time warping，DTW）是最为有效的距离测量方法[12-13].DTW 的优点[14]包括：①克服了欧氏距离点对必须对应的问题，允许不同步的点对应计算；②允许两时间序列具有不同长度；③对时间序列的同步问题不敏感.缺点[14]包括：①DTW 的计算复杂度较高，对于长度分别为 n 和 m 的时间序列，准确计算 DTW 距离需要 $O(nm)$ 的时间复杂度；②DTW 并不满足距离三角不等式[15]，无法进行快速推导，这极大影响了大规模时间序列数据集的距离计算效率.

如图 1 所示，在时间序列 C_1，C_2 和 C_3 中找出与时间序列 Q 距离最近的时间序列.若时间序列距离值不满足三角不等式，无法推导，则只能分别计算 Q 与 C_1，Q 与 C_2，Q 与 C_3 的距离，最后判断出 C_1 与 Q 距离最近.

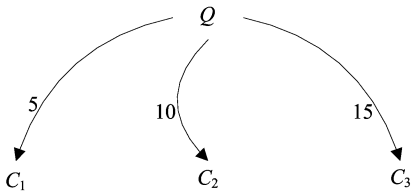

图 1 不可推导情况下的距离计算示意图

Fig.1 Distance computing under non educible case

图 2 给出了可推导情况下的距离计算示意图.这种情况下，可以预先计算出 C_1 与 C_2，C_1 与 C_3 的距离.若要找出与时间序列 Q 距离最近的时间序列，只要计算 Q 与 C_1 的距离，就可以根据事先计算出的 C_1 与 C_2，C_1 与 C_3 的距离，推导出 Q 与 C_2，Q 与 C_3 的距离范围（虚线所示）.由图 2 可以看到，Q 与 C_3 的距离范围最小值为 7，大于 Q 与 C_1 的距离 5，因此可以直接排除掉 Q 与 C_3 距离最近的可能，然后只需要进一步计算 Q 与 C_2 的距离就可以得到距离最近的时间序列.在时间序列数据集规模较大时，可以根据推导出来的范围值，剔除不符合要求的时间序列，大大减少时间序列之间的距离计算次数，提高计算效率.

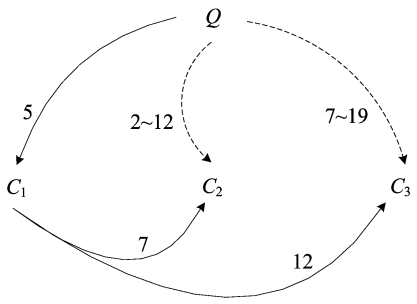

图 2 可推导情况下的距离计算示意图

Fig.2 Distance computing under educible case

目前，动态时间弯曲改进方法主要目的是用低计算复杂度的近似算法完成初步比较和剔除，从而提高计算效率.主要方法包括采用逐段聚集近似（piecewise aggregate approximation，PAA）[16]、自适应逐段常量近似（adaptive piecewise constant approximation，APCA）[17]、多元索引[18]等方法进行分段计算；采用 LB_Kim[19]、LB_Yi[15]、LB_Keogh[20]、LB_HUST[14]等可快速计算的下界距离来近似替代 DTW 距离.与 DTW 距离类似，这些近似距离同样不能进行推导，因此在距离测量时，只能对时间序列进行逐对比较.对有 n 条时间序列的数据集，需要进行 $(n-1)+(n-2)+\cdots+1=n(n-1)/2$ 次比较，I/O 代价高.

针对动态时间弯曲中的快速推导难题，本文首次提出了一种新颖的可推导动态时间弯曲近似距离以及相应的距离推导算法，实现了时间序列的 1 对多比较.

从可推导距离计算过程可知，要发挥出可推导距离的优点，必须事先计算出 C_1 与 C_2，C_1 与 C_3 的距离，供后续距离测量时使用。在实际应用中，一般是把事先计算出的距离值作为一种索引信息，在相似时间序列查询时基于索引信息进行推导。

1 相关工作

1.1 时间序列及 DTW 距离

表 1 列出了本文常用的符号及其含义.

定义 1.1 令时间序列 $T=t_1,t_2,\cdots,t_u$ 为一个 u 元序列.其中每个值 t_i 对应一个时间点上的记录值，记录时间是严格递增的，即若 $i<j$，则 t_i 记录时间在 t_j 之前.

时间序列 T 中记录值 t_i 可以是多种数据类型，包括数值、离散符号、结构数据、多媒体数据等，本文只考虑 t_i 为实数值的数据类型.

表 1　常用符号表

Tab.1　Common symbols

符号	含义
T	时间序列
t_i	时间序列在时间点 i 的记录值
Q	查询时间序列
Ω	查询时间序列集合
C	候选时间序列
Ψ	候选时间序列集合
DTW(Q,C)	时间序列 Q,C 之间的 DTW 距离
r	DTW 距离计算的全局约束常数
q^U	时间序列 Q 在全局约束 r 下滑动窗口内最大值组成的序列串
q^L	时间序列 Q 在全局约束 r 下滑动窗口内最小值组成的序列串
LB_Keogh(Q,C)	时间序列 Q,C 之间的 LB_Keogh 距离
LB_D(C,C')	时间序列 C,C'之间的 LB_D 距离
M^{Ψ}	Ψ 中时间序列之间 $\lvert\Psi\rvert\times\lvert\Psi\rvert$ 的 LB_D 距离矩阵
Θ	从 Ψ 中生成的样本集合
M^{Θ}	Θ 和 Ψ 时间序列之间 $\lvert\Theta\rvert\times\lvert\Psi\rvert$ 的 LB_D 距离矩阵
link_{S_i}	Θ 中时间序列 S_i 根据与 Ψ 中时间序列 LB_D 距离构成的降序链表
I	由 Θ 中时间序列的降序链表构成的索引

时间序列 T 是指采集到的原始数据,其序列长度很长且不统一,子序列是从时间序列中提取出的待分析的定长序列,子序列定义如下.

定义 1.2　令子序列 $T_{i,v}$ 是 T 中从位置 i 开始,长度为 v 的时间序列,即 $T_{i,v}=t_i,t_{i+1},\cdots,t_{i+v-1}$,$1\leqslant i\leqslant u-v+1$.下文中,如无特殊说明,所述时间序列都是指子序列.

查询时间序列和候选时间序列是时间序列距离测量中两个常用的概念,查询时间序列一般是指需要进行距离测量的时间序列,候选时间序列一般是指备查的时间序列.例如,在最近邻时间序列查询中,需要测量查询时间序列与多个候选时间序列的距离,并将候选时间序列集合中距离最近的时间序列输出为查询结果.实际应用中,查询时间序列和候选时间序列的数量一般是多条.下面给出查询时间序列集合和候选时间序列集合的定义.

定义 1.3　令查询时间序列集合为 Ω,$\Omega=\{Q_1,Q_2,\cdots,Q_m\}$,其中 $Q_i(1\leqslant i\leqslant m)$ 是长度为 v 的时间序列.

定义 1.4　令候选时间序列集合为 Ψ,$\Psi=\{C_1,C_2,\cdots,C_n\}$,其中 $C_i(1\leqslant i\leqslant n)$ 是长度为 v 的时间序列.

欧氏距离是最常用的时间序列距离测量指标,计算简单,但是对不等长、偏移、扭曲等时间序列的计算效果差.如图 3 所示,时间序列 $C=<-1,-1,1,1,-1,-1,-1>$ 和 $Q=<-1,-1,-1,1,1,-1,-1>$ 非常相似,仅相位不同,但是欧氏距离为 $\sqrt{8}$,难以体现这两个序列间的相似性.

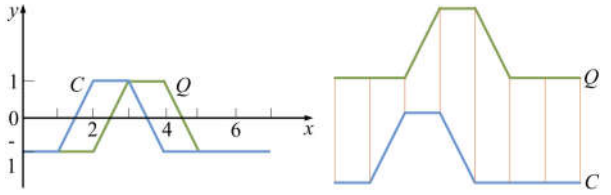

图 3　欧式距离示意图

Fig.3　Euclid distance diagram

在欧式距离基础上,研究人员提出了多种改进的距离计算方法,其中 DTW 是目前众多领域中应用最有效的距离计算方法.下面介绍 DTW 方法的基本原理.

定义 1.5　弯曲路径.对长度为 m 和 n 的两个时间序列 $Q=\{q_1,q_2,\cdots,q_m\}$,$C=\{c_1,c_2,\cdots,c_n\}$,构造大小为 $m\times n$ 的距离矩阵 $\boldsymbol{M}$,其中元素值 $M_{i,j}$ 为 q_i 和 c_j 的距离值 $d(q_i,c_j)=(q_i-c_j)^2$.弯曲路径 $W=w_1,w_2,\cdots w_e,\cdots,w_K$ ($\max(m,n)\leqslant K<m+n-1$) 是由 $\boldsymbol{M}$ 中满足下列约束条件的元素集合:

(Ⅰ)端点条件:$w_1=M_{1,1}$ 且 $w_K=M_{m,n}$.该条件规定了弯曲路径起始位置和终止位置在距离矩阵 $\boldsymbol{M}$ 中所处位置.

(Ⅱ)连续性:若 $w_e=M_{a,b}$,$w_{e-1}=M_{a',b'}$,则有 $a-a'\leqslant 1$ 且 $b-b'\leqslant 1$.该条件保证了弯曲路径的每一步在距离矩阵 $\boldsymbol{M}$ 中都是相邻的.

(Ⅲ)单调性:若 $w_e=M_{a,b}$,$w_{e-1}=M_{a',b'}$,则有 $a-a'\geqslant 0$ 且 $b-b'\geqslant 0$.该条件保证了弯曲路径上的元素在时间轴上的单调性.

定义 1.6 对于弯曲路径 $W=w_1,w_2,\cdots w_e,\cdots,w_K$ ($\max(m,n)\leqslant K<m+n-1$),令 W 的长度为 $\sqrt{\sum_{e=1}^{K}w_e}$.

根据弯曲路径的定义可知,对时间序列 Q,C ,可以构造出多条弯曲路径.

定义 1.7 DTW 距离.对长度为 m 和 n 的两个时间序列 $Q=\{q_1,q_2,\cdots,q_m\},C=\{c_1,c_2,\cdots,c_n\}$, Q,C 的 DTW 距离 DTW(Q,C) 为距离矩阵 $\boldsymbol{M}$ 中所有弯曲路径中的最短弯曲路径的长度.

DTW(Q,C) 值可以采用动态规划方法进行累积计算:

$$\gamma(i,j)=d(q_i,c_j)+\min\{\gamma(i-1,j-1),\gamma(i-1,j),\gamma(i,j-1)\}.$$

DTW(Q,C) 的计算时间和空间复杂度均为$O(mn)$.

根据 DTW 距离定义,如图 4 所示,时间序列 $C=<-1,-1,1,1,-1,-1,-1>$ 和 $Q=<-1,-1,-1,1,1,-1,-1>$为 0,比欧式距离更能反映两个时间序列的相似性.

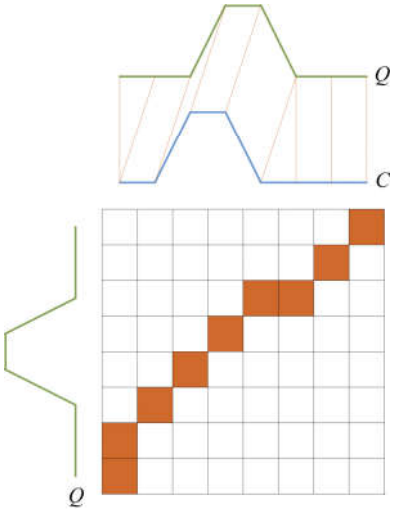

图 4 DTW 距离示意图

Fig.4 DTW distance diagram

1.2 DTW 距离计算的全局约束

由于 DTW 距离允许不同步的点对应计算,在实际应用中,为了防止大量的点对应同一个点之类的过度扭曲变形的对应关系,一般都是通过全局约束条件,把动态弯曲路径限制在距离矩阵对角线附近.最常用的全局约束条件有 Sakoe-Chiba 带[21] 和 Itakura 平行四边形[22].

对于 Sakoe-Chiba 带和 Itakura 平行四边形来说,若弯曲路径中的步骤 w_e 对应距离矩阵 $\boldsymbol{M}$ 中位置为 (i,j) ,则有 $j-r\leqslant i\leqslant j+r$.其中,对于 Sakoe-Chiba 带, r 是一个预先设定的常数,如图 5 所示.对于 Itakura 平行四边形, r 是一个与 i 相关的变量.

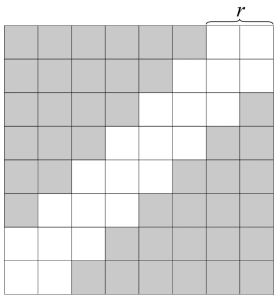

图 5 Sakoe-Chiba 带示意图

Fig.5 Sakoe-Chiba band diagram

1.3 DTW 距离下界函数

由于 DTW 具有较高的计算复杂度,实际的相似性分析往往不是直接计算原始 DTW 距离,而是首先计算 DTW 的距离下界,符合条件后再精确计算序列间的 DTW 距离[20].

定义 1.8 距离下界函数.设 D 为对象空间 O 中的距离度量,若某定义在 O 对象空间上的函数 LB,有 $\forall o_i,o_j\in O,0\leqslant \mathrm{LB}(o_i,o_j)\leqslant D(o_i,o_j)$,则称 LB 为距离度量 D 的下界函数.

一般来说,距离下界函数计算复杂度低,可以快速计算.根据距离下界函数的定义,若要判断 o_i,o_j 是否相似(距离相近),可以先计算 LB 下界函数值,若下界小于预先设定的阈值,则判断 o_i,o_j 可能是相似的,否则必然不相似.对于可能相似的对象再进行进一步计算距离度量 D 的值.

目前,常用的 DTW 距离下界函数包括 LB_Kim、LB_Yi、LB_Keogh 等,其中 LB_Keogh 的逼近效果最好,应用范围最广.

LB_Keogh 是考虑全局约束后,通过计算一个序列超出另外一个序列的 $[q^L,q^U]$ 组成的边界部分,作为 DTW 距离的下界,其中 q^L,q^U 分别是全局约束下滑动窗口内数据的最小值和最大值组成的序列串.LB_Keogh 下界函数如图 6 中的阴影部分所示.

$$\mathrm{LB\_Keogh}(Q,C)=\sqrt{\sum_{i=1}^{v}\mathrm{diff}_i^2}.$$

其中,

$$\mathrm{diff}_i=\begin{cases}c_i-q_i^L, & c_i<q_i^L\\ c_i-q_i^U, & c_i>q_i^U\\ 0, & \text{其他}\end{cases},$$

$$q_i^U=\max(q_{i-r},\cdots,q_{i+r}),$$
$$q_i^L=\min(q_{i-r},\cdots,q_{i+r}).$$

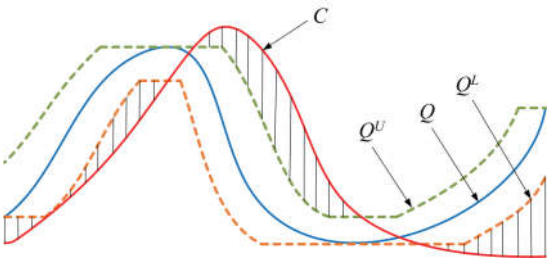

图 6　LB_Keogh 示意图

Fig.6　LB_Keogh diagram

上述 DTW 下界都不满足距离三角不等式，不适用于 1 对多比较时的快速推导.实际应用中，除了 DTW 下界函数外，在特定应用中有时还采用 DTW 上界函数(例如欧式距离)提高计算效率.

2　可推导 DTW 下界函数

下面给出我们提出的可推导 DTW 下界函数的定义.

定义 2.1　LB_D 下界函数.设时间序列 C,C' 长度为 v，全局约束条件值为 r.LB_D 下界函数为

$$\mathrm{LB\_D}(C,C')=\sqrt{\sum_{i=1}^{v}\mathrm{diff}_i^2}.$$

其中，

$$\mathrm{diff}_i=\begin{cases}c'^U_i-c_i^L, & c'^U_i<c_i^L\\ c'^L_i-c_i^U, & \\ 0, & \text{其他}\end{cases},$$

$c_i^U=\max(c_{i-r},\cdots,c_{i+r})$，$c_i^L=\min(c_{i-r},\cdots,c_{i+r})$，$c'^U_i=\max(c'_{i-r},\cdots,c'_{i+r})$，$c'^L_i=\min(c'_{i-r},\cdots,c'_{i+r})$.

显然，$0\leqslant \mathrm{LB\_D}(C,C')\leqslant \mathrm{LB\_Keogh}(C,C')\leqslant \mathrm{DTW}(C,C')$，LB_D 为一个 DTW 下界函数，其计算复杂度为时间序列长度的线性时间.

LB_D 下界函数如图 7 中的阴影部分所示.

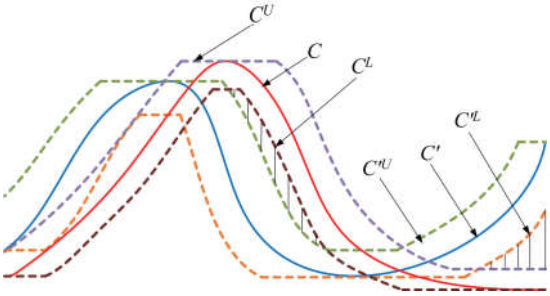

图 7　LB_D 示意图

Fig.7　LB_D diagram

定理 2.1　设时间序列 C,C' 长度为 v，全局约束条件值为 r，有 $\mathrm{LB\_D}(C,C')=\mathrm{LB\_D}(C',C)$.

证明　见附录.

定理 2.1 表明 LB_D 距离函数是对称的.

定理 2.2　设全局约束条件值为 r，对于任意的长度为 v 的时间序列 Q,C,C'，有 $\mathrm{LB\_D}(C,C')\leqslant \mathrm{LB\_Keogh}(C,Q)+\mathrm{LB\_Keogh}(C',Q)$.

证明　见附录.

根据定理 2.1 以及 LB_D 和 LB_Keogh 函数值可快速推导多个时间序列之间 DTW 距离下界.

推论 2.1　设全局约束条件值为 r，对于任意的长度为 v 的时间序列 Q,C,C'，有 $\mathrm{DTW}(Q,C')\geqslant \mathrm{LB\_D}(C,C')-\mathrm{LB\_Keogh}(C,Q)$.

证明　由定理 2.2 可知，

$$\mathrm{LB\_Keogh}(C',Q)\geqslant \mathrm{LB\_D}(C,C')-\mathrm{LB\_Keogh}(C,Q).$$

由于 LB_Keogh 函数为 DTW 距离下界函数，即 $\mathrm{DTW}(Q,C')\geqslant \mathrm{LB\_Keogh}(C',Q)$，因此 $\mathrm{DTW}(Q,C')\geqslant \mathrm{LB\_D}(C,C')-\mathrm{LB\_Keogh}(C,Q)$.

3　基于 LB_D 的索引

可推导 DTW 下界函数 LB_D 的特点是可以根据推论 2.1 中三角不等式推导出两个时间序列的 DTW 下界距离，这对提升相似度计算效率极为有利.根据推论 2.1，对查询时间序列 Q 以及候选时间序列集合 $\Psi=\{C_1,C_2,\cdots,C_n\}$，$\Psi$ 中任意时间序列 $C_i\in\Psi(1\leqslant i\leqslant n)$，如果预先计算出 C_i 与 Ψ 中其他时间序列的 LB_D 函数值，那么当计算出 $\mathrm{LB\_Keogh}(C_i,Q)$ 之后，就可以根据 LB_D 函数值快速推导出 Ψ 中其他时间序列与 Q 的 DTW 距离下界，而不需要对 Q 与 Ψ 中时间序列原始数据逐一比较.

根据这一结论，下文给出基于可推导 DTW 下界函数 LB_D 来构建索引的方法.

3.1　索引构建

基于可推导 DTW 下界函数 LB_D 来构建索引的一个简单方法是，针对候选时间序列集合 $\Psi=\{C_1,C_2,\cdots,C_n\}$，建立一个 $n\times n$ 的距离矩阵 $\boldsymbol{M}^{\Psi}$，存储各候选时间序列之间的 LB_D 函数值，即 $\boldsymbol{M}^{\Psi}(i,j)=\mathrm{LB\_D}(C_i,C_j)(1\leqslant i\leqslant n,1\leqslant j\leqslant n)$.查询时，计算出查询时间序列 Q 和 $\boldsymbol{M}^{\Psi}$ 中任意时间序列的 LB_Keogh 函数值之和，就可以根据 $\boldsymbol{M}^{\Psi}$ 来推导出其他时间序列和 Q 的 DTW 下界.实际应用中，候选时间序列集合数据量往往很大，并且要进行

更新,因此维护 $\boldsymbol{M}^{\Psi}$ 的代价非常高.

为了提升索引构建和维护效率,我们采用基于样本的比较方法,即选取少量的候选时间序列作为样本集合 $\Theta=\{S_1,S_2,\cdots,S_k\}(k\ll n)$,然后构建一个 $k\times n$ 的距离矩阵 $\boldsymbol{M}^{\Theta}$,存储 Ψ 和 Θ 中时间序列之间的 LB_D 函数值,即 $\boldsymbol{M}^{\Theta}(i,j)=\mathrm{LB\_D}(S_i,C_j)(S_i\in\Theta,C_j\in\Psi,1\leqslant i\leqslant k,1\leqslant j\leqslant n)$,然后对 $\boldsymbol{M}^{\Theta}$ 各行按降序排序,即 $\mathrm{LB\_Deng}(S_i,C_{S_i}^{j})\geqslant\mathrm{LB\_Deng}(S_i,C_{S_i}^{j+1})$.这样,我们只需要计算样本集合 Θ 中时间序列和 Q 的 LB_Keogh 函数值,就可以迅速根据推论 2.1 和 $\boldsymbol{M}^{\Theta}$ 来剔除不相似的时间序列.

在上述索引构建方法中,样本集合 Θ 的确定非常关键.确定样本集合 Θ 的策略可以采用以下 2 种方法:

(Ⅰ)自动筛选:根据已有的分类或者聚类方法,从 Ψ 中计算出样本集合 Θ.在该方法中,由于实际应用中 Ψ 数据量往往很大,并且要进行更新,因此,一般采用定期更新 Θ 的方式.若 Ψ 数据量非常大,每次更新时,可以先采用抽样方法抽取 Ψ 中部分时间序列,在此基础上计算样本集合 Θ.

(Ⅱ)指定:根据已有知识直接指定 Ψ 中部分时间序列为样本集合 Θ.例如,在故障分析中,我们可以把已知的典型故障时间序列指定为样本集合 Θ.

在上述两种策略中,自动筛选策略通用性强,在 Ψ 经常更新时,维护样本集合 Θ 的代价较高;指定策略可直接确定样本集合 Θ,维护简单,但需要有相应的知识库来支撑.

3.2 索引存储和维护

当候选时间序列集合 Ψ 更新时,索引信息也需要进行更新.索引维护包括两个方面:一是样本集合 Θ 的维护,二是距离矩阵 $\boldsymbol{M}^{\Theta}$ 的维护.

考虑到 $\boldsymbol{M}^{\Theta}$ 中每一行的值都是按降序排列,并且行与行之间没有关联性,因此可以把每一行都用一个双向链表按值大小的降序存储,以便于检索和更新.Θ 中任意时间序列 $S_i(1\leqslant i\leqslant k)$ 对应的双向链表记为 $\mathrm{link}_{S_i}=\{<C_{S_i}^{1},\mathrm{LB\_D}(S_i,C_{S_i}^{1})>,<C_{S_i}^{2},\mathrm{LB\_D}(S_i,C_{S_i}^{2})>,\cdots,<C_{S_i}^{n},\mathrm{LB\_D}(S_i,C_{S_i}^{n})>\}$,其中 $\forall j(1\leqslant j\leqslant n-1)$,都有 $C_{S_i}^{j}\in\Psi$ 且 $\mathrm{LB\_D}(S_i,C_{S_i}^{j})\geqslant\mathrm{LB\_D}(S_i,C_{S_i}^{j+1})$,整个索引结构记为 $I=\{\mathrm{link}_{S_1},\mathrm{link}_{S_2},\cdots,\mathrm{link}_{S_k}\}$.

举例来说,假如有 $\Psi=\{C_1,C_2,C_3,C_4\},\Theta=\{S_1,S_2,S_3\}$,且

$$\mathrm{LB\_D}(S_1,C_1)=2,\mathrm{LB\_D}(S_1,C_2)=3,$$
$$\mathrm{LB\_D}(S_1,C_3)=5,\mathrm{LB\_D}(S_1,C_4)=1;$$
$$\mathrm{LB\_D}(S_2,C_1)=5,\mathrm{LB\_D}(S_2,C_2)=2,$$
$$\mathrm{LB\_D}(S_2,C_3)=6,\mathrm{LB\_D}(S_2,C_4)=4;$$
$$\mathrm{LB\_D}(S_3,C_1)=2,\mathrm{LB\_D}(S_3,C_2)=7,$$
$$\mathrm{LB\_D}(S_3,C_3)=1,\mathrm{LB\_D}(S_3,C_4)=5;$$

则索引 I 存储结构如图 8 所示.

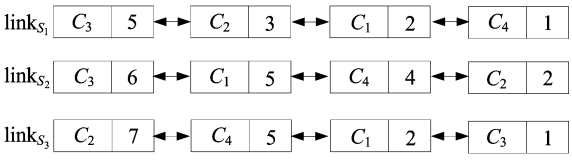

图 8 索引存储结构示意图

Fig.8 Index storage structure diagram

更新时,对于样本集合 Θ 中时间序列的新增或减少,直接增加或删除对应的链表;对于时间序列集合 Ψ 中时间序列的新增,需要逐个计算新的时间序列与样本集合 Θ 中各时间序列的 LB_D 函数值,然后分别插入到相应的链表中;对于时间序列集合 Ψ 中时间序列的减少,则从各链表中删除相对应的节点.

4 基于 LB_D 的相似时间序列查询算法

4.1 基本思想

基于推论 2.1 和索引结构,在相似时间序列查询中,只需要计算出查询时间序列 Q 和样本集合 Θ 中时间序列之间的 LB_Keogh 距离,就可得到 Q 和 Ψ 中各时间序列 DTW 距离下界,这样就能快速的剔除不相似的时间序列.

剔除不相似的时间序列时,可以选择以下策略:

策略 4.1 根据 Θ 中时间序列,对 I 各链表逐一遍历,尽可能剔除 Ψ 中不相似的时间序列.这种策略需要记录 Ψ 中保留和剔除的时间序列,因此剔除率较高,但是存在大量的重复剔除,时间和空间复杂度较高.

策略 4.2 根据 Θ 中时间序列,对 I 各链表逐一遍历,找出被剔除最多的行,以该行对应的 Θ 中时间序列和索引链表为基础,对 Ψ 中剩余的时间序列进行后续相似度计算.这种策略不需要记录被剔除的时间序列,但是需要对 I 各链表逐一遍历.

策略 4.3 根据查询时间序列 Q 和样本集合 Θ 中时间序列之间的 LB_Keogh 距离,选取 Θ 中与 Q 距离最小的时间序列作为参照时间序列 S_{ref},然后根据 I 中参照时间序列对应的索引链表 $\mathrm{link}_{S_{\mathrm{ref}}}$,进行剔除.这种策略基本思想是从 Θ 中找出和 Q 最相似的时间序列,然后根据 $\mathrm{link}_{S_{\mathrm{ref}}}$,推导 Ψ 中时间序列与 Q 的 DTW 下界,剔除不相似的时间序列.与前

两种策略相比，该策略的剔除率较低，但时间和空间复杂度最低.

与策略 4.1 和 4.2 相比，策略 4.3 的剔除率较低，但是在时间和空间复杂度方面优势明显，因此在我们在相似时间序列查询算法中采用策略 4.3.

这样，相似时间序列查询分为三个阶段：

阶段 1　计算出查询时间序列 Q 和样本集合 Θ 中时间序列之间的 LB_Keogh 距离，选取距离最近的作为参照时间序列 S_{ref}；

阶段 2　根据 S_{ref} 在 I 中的索引链表 $link_{S_{ref}}$，剔除 Ψ 中不相似的时间序列；

阶段 3　根据 $link_{S_{ref}}$，在 Ψ 剩余的时间序列中查询出相似时间序列.

4.2　最近邻时间序列查询算法

下面引入可推导 DTW 近似距离之后的最近邻时间序列查询算法（deducing for the nearest neighbor searching，D-NNS）描述.

算法 4.1　D-NNS

输入：查询时间序列 Q，候选时间序列集合 $\Psi=\{C_1, C_2, \cdots, C_n\}$，全局约束条件值 r，样本集合 $\Theta=\{S_1, S_2, \cdots, S_k\}$，索引 $I=\{link_{S1}, link_{S2}, \cdots, link_{Sk}\}$

输出：最佳匹配 C_{best}，$C_{best}\in\Psi$ 且 $\forall C_i\in\Psi(1\leqslant i\leqslant n)$ 有 $DTW(Q,C_{best})\leqslant DTW(Q,C_i)$

1　$<S_{ref}, LB\_Keogh(S_{ref}, Q), DTW(S_{ref}, Q)>\leftarrow FindRefSeries(Q, r, \Theta)$

2　$RefinePoint\leftarrow EarlyAbandoning(link_{S_{ref}}, S_{ref}, LB\_Keogh(S_{ref}, Q), DTW(S_{ref}, Q))$

3　$C_{best}\leftarrow Refine(link_{S_{ref}}, RefinePoint, LB\_Keogh(S_{ref}, Q), DTW(S_{ref}, Q), \Psi)$

算法第 1 行通过 FindRefSeries 过程确定参照时间序列 S_{ref} 及其与 Q 的 LB_Keogh 距离，第 2 行通过 EarlyAbandoning 过程剔除不相似的候选时间序列，确定剩余时间序列在 S_{ref} 对应的索引链表的起始节点，第 3 行通过 Refine 过程从剩余时间序列中查询最佳匹配时间序列 C_{best}.

FindRefSeries 过程遍历样本集合 Θ，找出与 Q 的 LB_Keogh 距离最小的时间序列作为参照时间序列，算法描述如下：

过程算法 4.1.1　FindRefSeries

输入：查询时间序列 Q，全局约束条件值 r，样本集合 $\Theta=\{S_1, S_2, \cdots, S_k\}$

输出：$<S_{ref}, LB\_Keogh(S_{ref}, Q)>$

```
1 S_ref ← S_1
2 LK_bsf ← LB_Keogh(S_1, Q)
3 for i=2 to k
4    if LB_Keogh(S_i, Q)<LK_bsf
5       S_ref ← S_i
6          LK_bsf ← LB_Keogh(S_i, Q)
7 DTW_bsf←DTW(S_ref, Q)
8 Output <S_ref, LK_bsf, DTW_bsf>
```

FindRefSeries 过程第 2 行时间复杂度为 $O(v)$，第 3～6 行时间复杂度为 $O(kv)$，因此总的时间复杂度为 $O(kv)$.

EarlyAbandoning 过程根据 S_{ref} 对应的索引链表 $link_{S_{ref}}$，剔除不相似的候选时间序列，确定剩余时间序列在 S_{ref} 对应的索引链表的起始节点，算法描述如下：

过程算法 4.1.2　EarlyAbandoning

输入：索引链表 $link_{S_{ref}}$，参照时间序列 S_{ref}，$LB\_Keogh(S_{ref}, Q)$，$DTW(S_{ref}, Q)$，

输出：RefinePoint

```
1 for i=1 to n
2    if LB_D(S_ref, C^i_Sref) − LB_Keogh(S_ref, Q)<DTW(S_ref, Q)
3          break
4 Output i
```

EarlyAbandoning 过程第 2 行 $LB\_D(S_{ref}, C^i_{S_{ref}})$ 值可在索引链表 $link_{S_{ref}}$ 直接获得，因此 EarlyAbandoning 过程复杂度为 $O(n)$.

Refine 过程从剩余时间序列中查询最佳匹配时间序列 C_{best}，这一过程通过对 Q 与 Ψ 中剩余时间序列的原始数据进行一一相似度计算和比较完成. 这一过程基本的计算过程如下：

过程算法 4.1.3　Refine

输入：索引链表 $link_{S_{ref}}$，RefinePoint，$LB\_Keogh(S_{ref}, Q)$，$DTW(S_{ref}, Q)$，Ψ

输出：最佳匹配 C_{best}

```
1 bsf←DTW(S_ref, Q)
2 LK_bsf ← LB_Keogh(S_ref, Q)
3 index_bsf ← 0
4 for i=RefinePoint to n
5   if LK_dist←LB_Keogh(Q, C^i_Sref)<bsf
6       if DTW_dist←DTW(Q, C^i_Sref)<bsf
7             bsf←DTW_dist
8             LK_bsf←LK_dist
9             index_bsf ← i
10 if index_bsf =0
11         Output S_ref
12 else
13         Output C^index_bsf_Sref
```

Refine 过程第 5 行的计算复杂度为 $O(v)$，第 6 行的计算复杂度为 $O(v^2)$，因此总的计算复杂度为 $O(v^2n)$.

与现有算法相比，D-NNS 算法增加了第 1 阶段和第 2 阶段来剔除不相似的时间序列，时间复杂度分别为 $O(kv)$ 和 $O(n)$，D-NNS 算法第 3 阶段与现有算法是类似的，时间复杂度为 $O(v^2n)$，但是候选时间序列集合方面现有算法是 Ψ，而 D-NNS 算法是进行第 2 阶段剔除之后的时间序列集合.由于 $k \ll n$，因此第 1 阶段和第 2 阶段的时间复杂度为远低于第 3 阶段.实验表明，当第 2 阶段的剔除率较高时，D-NNS 算法能够显著提升最近邻时间序列查询效率.

与现有最近邻时间序列查询算法类似，D-NNS 算法只需要对第 3 阶段稍作修改，可以很容易扩展成为 k 近邻时间序列查询算法以及基于阈值的相似时间序列查询算法.

5 实验

5.1 实验设置

实验中，我们针对时间序列最近邻查询场景，通过 D-NNS 算法和目前基于 DTW 的时间序列距离测量最代表性算法 UCR-DTW[13] 进行比较，验证本文提出的 DTW 下界推导方法的有效性.

UCR-DTW 算法综合采用了当前大部分已知的时间序列距离测量优化方法，包括去掉开根号运算、在线归一化、基于 LB_Kim、LB_Keogh 下界值的早期剔除等.该算法源代码可在文献[23]中获得.

实验所采用的数据均来源于文献[24].该数据源包含了来自于不同领域的 85 种时序数据集，是目前具有代表性公开时序数据集.该数据源中每种数据集都包含测试数据和训练数据两部分，测试数据和训练数据中时间序列长度相同.本文实验中，把测试数据作为候选时间序列集合，把训练数据作为查询时间序列集合.

如无特殊说明，实验中采用的全局约束条件 $r = v \times 5\%$，并且在构建各实验数据集的基于 LB_D 索引时，采用 3.1 节所述自动筛选策略，按照 k 中心点算法 PAM[25] 选出 20 个时间序列作为样本集合，以此建立索引.选取其他实验参数时结果是类似的.

实验结果的比较采用剔除率和算法执行时间两项指标，其中剔除率定义如下：

$$\text{剔除率} = \frac{\text{候选时间序列集合中根据下届函数判断为不相似的时间序列数量}}{|\text{候选时间序列集合}|} \times 100\%.$$

剔除率越高，意味着需要后续进行相似度比较的时间序列越少.一般来说，后续处理往往需要直接计算时间序列间的 DTW 距离，从而找出最相似的时间序列，计算机复杂度高，因此剔除率越高，表示下届距离函数应用效果越好.所有实验在一台 CPU 为 Intel Core I7 4×3.4 GHz 和内存为 4 GB 机器上运行，操作系统为 NeoKylin 3.2.2(内核为 Linux 2.6.32).

5.2 实验结果分析

实验结果如表 2 所示.表 2 列举了各种数据集基于 UCR-DTW 算法得出的 LB_Keogh 剔除率以及基于 D-NNS 算法得到的 LB_D 剔除率.表 2 最后两列是在进行数据平滑处理后得到的结果.

表 2 剔除率和算法执行时间比较

Tab.2 Comparison of abandon ratio and algorithm execution time

序号	数据集名称	数据集大小	查询集大小	时序长度	LB_Keogh 剔除率(%)	UCR-DTW 算法执行时间(s)	LB_D 剔除率(%)	D-NNS 算法执行时间(s)	D-NNS 算法阶段 1 和阶段 2 执行时间(s)	平滑窗口长度 5%时 LB_D 剔除率(%)	平滑窗口长度 10%时 LB_D 剔除率(%)
1	Yoga	3 000	300	426	97.482	3.635	57.425 0	3.416	0.093	65.674	76.620
2	Synthetic Control	300	300	60	46.275 56	0.499	3.125 6	0.343	<0.001	16.400	50.340
3	Gun-Point	150	50	150	93.08	0.016	76.720 0	0.016	<0.001	79.907	79.320
4	CBF	900	30	128	78.3	0.171	5.807 4	0.156	<0.001	70.833	80.256
5	Face (all)	1 690	560	131	50.405 33	9.033	0.018 5	6.942	0.032	0.831	6.838
6	OSU Leaf	242	200	427	84.725 21	1.341	5.012 4	0.983	0.093	13.955	31.909

续表 2

序号	数据集名称	数据集大小	查询集大小	时序长度	LB_Keogh 剔除率(%)	UCR-DTW 算法执行时间(s)	LB_D 剔除率(%)	D-NNS 算法执行时间(s)	D-NNS 算法阶段 1 和阶段 2 执行时间(s)	平滑窗口长度 5%时 LB_D 剔除率(%)	平滑窗口长度 10%时 LB_D 剔除率(%)
7	Swedish Leaf	625	500	128	79.667 84	1.279	8.564 2	1.264	0.031	16.111	32.790
8	50Words	455	450	270	91.575 58	1.357	14.904 0	0.999	0.063	22.311	33.264
9	Trace	100	100	275	88.51	0.14	70.550 0	0.125	0.031	82.500	85.270
10	Face (four)	88	24	350	41.524 62	0.187	5.350 4	0.172	0.015	15.767	45.502
11	Lightning-2	61	60	637	66.912 57	0.718	4.316 9	0.609	0.016	53.169	70.246
12	Lightning-7	73	70	319	70.861 06	0.265	10.958 9	0.234	<0.001	63.797	76.693
13	ECG	100	100	96	73.9	0.047	34.320 0	0.047	<0.001	54.200	67.410
14	Adiac	391	390	176	57.598 53	1.56	5.2076	1.498	0.063	14.284	32.377
15	Fish	175	175	463	64.924 08	1.825	0.098 0	1.794	0.048	3.226	35.971
16	Plane	105	105	144	79.818 59	0.078	16.598 6	0.078	<0.001	30.340	60.880
17	Car	60	60	577	66.277 78	0.468	4.333 3	0.39	0.047	19.028	37.861
18	Beef	30	30	470	72.666 67	0.094	50.333 3	0.062	0.016	61.889	73.556
19	Coffee	28	28	286	19.387 76	0.094	1.785 7	0.078	<0.001	12.372	15.051
20	OliveOil	30	30	570	1	0.297	0.000 0	0.234	0.016	0.778	19.444
21	ChlorineConcentration	3 840	467	166	86.452 48	2.714	0.143 2	2.73	0.03	26.007	66.554
22	DiatomSizeReduction	306	16	345	70.506 54	0.125	1.736 1	0.125	0.015	2.737	32.149
23	ECGFiveDays	861	23	136	89.733 88	0.047	13.896 9	0.047	0.015	36.394	58.633
24	FacesUCR	2 050	200	131	70.568 05	2.777	0.025 9	2.73	0.03	17.148	39.198
25	Haptics	308	155	1092	78.956 85	11.076	8.961 0	10.561	0.339	17.627	31.649
26	InlineSkate	550	100	1882	96.881 82	6.77	50.981 8	5.772	0.423	59.707	66.444
27	ItalyPowerDemand	1 029	67	24	86.545 41	0.046	46.554 4	0.032	<0.001	52.828	67.713
28	MALLAT	2 345	55	1 024	85.752 28	12.917	0.000 0	12.933	0.031	0.444	19.764
29	MedicalImages	760	381	99	91.676 34	0.265	34.087 2	0.265	0.015	40.325	48.152
30	MoteStrain	1 252	20	84	86.206 07	0.046	14.273 2	0.047	0.015	30.032	51.282
21	SonyAIBORobot SurfaceII	953	27	65	69.488 17	0.062	0.0155	0.046	<0.001	8.286	30.391
22	SonyAIBORobot Surface	601	20	70	64.284 53	0.032	0.324 5	0.031	<0.001	2.912	30.599
23	Symbols	995	25	398	95.545 73	0.156	73.829 1	0.14	0.016	76.430	80.756
24	TwoLeadECG	1139	23	82	83.482 84	0.047	15.936 9	0.046	<0.001	35.512	60.644
25	WordsSynonyms	638	267	270	93.379 36	0.952	9.469 0	0.795	0.031	17.689	39.226
26	Cricket_X	390	390	300	86.078 9	1.935	18.221 6	1.56	0.077	41.526	60.811
27	Cricket_Y	390	390	300	85.383 96	2.075	18.082 2	1.747	0.111	49.092	64.849
28	Cricket_Z	390	390	300	86.6449 7	1.981	18.790 9	1.623	0.125	42.210	57.389
29	InsectWingbeatSound	1 980	220	256	95.684 34	1.17	2.577 1	1.138	0.045	9.683	38.706
30	ArrowHead	175	36	251	73.682 54	0.125	20.746 0	0.094	<0.001	32.111	35.381
31	BeetleFly	20	20	512	35	0.093	0.000 0	0.078	<0.001	4.250	23.250
32	BirdChicken	20	20	512	63.5	0.062	40.250 0	0.032	0.016	52.500	63.750
33	Ham	105	109	431	51.585 85	1.857	0.000 0	1.358	0.031	4.972	35.308

续表 2

序号	数据集名称	数据集大小	查询集大小	时序长度	LB_Keogh 剔除率(%)	UCR-DTW 算法执行时间(s)	LB_D 剔除率(%)	D-NNS 算法执行时间(s)	D-NNS 算法阶段 1 和阶段 2 执行时间(s)	平滑窗口长度 5%时 LB_D 剔除率(%)	平滑窗口长度 10%时 LB_D 剔除率(%)
34	Herring	64	64	512	55.664 06	0.406	0.000 0	0.358	0.032	0.293	14.673
35	PhalangesOutlinesCorrect	858	1 800	80	56.002 53	5.648	0.122 4	6.411	0.092	0.228	2.478
36	ProximalPhalanx OutlineAgeGroup	205	400	80	27.680 49	0.452	0.308 5	0.437	<0.001	13.413	1.940
37	ProximalPhalanx OutlineCorrect	291	600	80	26.804 12	0.951	0.000 0	0.983	0.031	11.427	0.908
38	ProximalPhalanxTW	400	205	80	26.564 63	0.405	0.356 1	0.375	<0.001	12.101	4.560
39	ToeSegmentation1	228	40	277	79.846 49	0.171	10.614 0	0.125	<0.001	27.215	58.333
40	ToeSegmentation2	130	36	343	79.871 8	0.14	12.970 1	0.094	0.015	34.145	50.641
41	DistalPhalanx OutlineAgeGroup	400	139	80	21.350 72	0.328	0.000 0	0.327	0.015	0.157	1.871
42	DistalPhalanx OutlineCorrect	600	276	80	32.837 56	0.842	0.430 0	0.936	<0.001	0.937	1.715
43	DistalPhalanxTW	400	139	80	24.118 71	0.359	0.528 8	0.312	<0.001	5.034	8.919
44	Earthquakes	322	139	512	8.000 804	8.549	0.000 0	8.611	0.14	1.448	11.978
45	MiddlePhalanx OutlineAgeGroup	400	154	80	11.806 82	0.344	0.165 6	0.375	<0.001	0.331	0.445
46	MiddlePhalanx OutlineCorrect	600	291	80	16.805 27	0.936	0.049 8	1.03	<0.001	0.217	1.194
47	MiddlePhalanxTW	399	154	80	25.227 03	0.344	0.091 1	0.343	<0.001	1.361	4.062
48	ShapeletSim	180	20	500	0	1.84	0.000 0	1.856	<0.001	0.000	4.694
49	Wine	54	57	234	13.937 62	0.141	0.000 0	0.109	0.015	0.000	6.530
50	Computers	250	250	720	74.996 8	7.784	6.323 2	7.083	0.202	26.707	57.928
51	LargeKitchenAppliances	375	375	720	84.578 13	10.514	0.342 8	8.846	0.314	21.198	59.465
52	Meat	60	60	448	16.916 67	0.873	0.000 0	0.733	<0.001	0.583	19.250
53	RefrigerationDevices	375	375	720	20.494 93	34.898	0.0320	35.536	0.357	0.371	3.765
54	ScreenType	375	375	720	72.362 67	19.173	11.385 6	13.79	0.31	34.015	51.773
55	ShapesAll	600	600	512	89.878 33	9.033	16.477 2	4.664	0.22	30.380	42.478
56	SmallKitchenAppliances	375	375	720	74.047 29	15.725	0.000 0	14.18	0.187	0.653	14.094
57	Strawberry	613	370	235	81.480 98	1.653	7.366 5	1.279	0.046	18.685	43.291
58	Worms	181	77	900	57.738 39	4.181	10.167 2	3.619	0.063	24.991	43.022
59	WormsTwoClass	181	77	900	57.738 39	4.149	10.167 2	3.588	0.092	24.991	43.022
60	Two Patterns	4 000	1 000	128	97.085 08	5.023	4.468 4	5.928	0.124	11.334	26.997
61	Wafer	6 174	1 000	152	98.178 02	2.933	80.159 3	4.93	0.107	83.359	81.567
62	CinC_ECG_torso	1 380	40	1 639	97.778 99	2.98	19.367 8	3.713	0.142	30.292	49.246

续表 2

序号	数据集名称	数据集大小	查询集大小	时序长度	LB_Keogh 剔除率(%)	UCR-DTW 算法执行时间(s)	LB_D 剔除率(%)	D-NNS 算法执行时间(s)	D-NNS 算法阶段 1 和阶段 2 执行时间(s)	平滑窗口长度 5%时 LB_D 剔除率(%)	平滑窗口长度 10%时 LB_D 剔除率(%)
63	StarLightCurves	8 236	1 000	1 024	96.841 14	166.124	61.501 1	171.429	1.663	67.257	75.915
64	uWaveGestureLibrary_X	3 582	896	315	97.198 76	9.563	55.692 7	8.627	0.247	60.012	68.340
65	uWaveGestureLibrary_Y	3 582	896	315	98.226 44	6.88	62.858 6	6.396	0.17	65.710	72.588
66	uWaveGestureLibrary_Z	3 582	896	315	97.889 15	7.753	53.038 3	7.894	0.235	60.224	69.194
67	Non-Invasive Fetal ECG Thorax1	1 965	1 800	750	81.866 81	172.38	2.374 4	168.886	2.062	10.753	41.357
68	Non-Invasive Fetal ECG Thorax2	1 965	1 800	750	81.670 79	117.157	3.032 3	114.614	1.64	9.120	43.020
69	ECG5000	4 500	500	140	83.842 62	8.377	24.475 4	5.507	0.079	56.287	68.053
70	ElectricDevices	7 711	8 926	96	80.082 47	211.77	1.593 6	251.722	0.376	7.402	30.957
71	FordA	3 601	1 320	500	23.191 45	761.234	0.000 0	757.553	0.531	0.004	11.733
72	FordB	3 636	810	500	24.627 39	439.39	0.000 0	414.801	0.449	0.000	11.690
73	HandOutlines	1 000	370	2709	85.251 62	143.177	1.902 7	129.215	3.924	2.802	5.606
74	Phoneme	1 896	214	1 024	37.332 65	203.331	0.000 0	185.063	0.451	0.339	7.929
75	UWaveGestureLibraryAll	3 582	896	945	93.994 84	100.979	6.478 2	96.362	1.203	16.649	43.094

由表 2 可知，基于 LB_D 的剔除率约为 15%，基于 LB_Keogh 的约为 65%.由于基于 LB_D 进行剔除是基于索引完成的，因此只需要前期一次性建立索引，比较时不需要读取原始数据.这意味着采用基于 LB_D 方法，平均可以减少约 15%的 I/O 代价.在运行速度方面，UCR-DTW 算法和 D-NNS 算法执行时间相当，但是 D-NNS 算法阶段 1 和阶段 2 执行时间(即基于 LB_D 进行剔除的部分)只占总的执行时间 1%左右，也就是说，采用基于 LB_D 进行剔除的时间非常短，大部分时间消耗在对阶段 3 剩余时间序列的精确比较.这是因为基于 LB_D 的剔除功能是在索引基础上进行的，所以运算速度快.

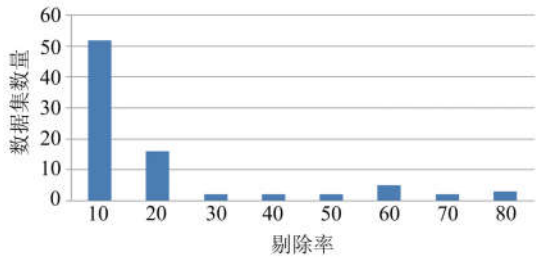

图 9　基于 LB_D 剔除率分布图

Fig.9　Abandon ratio distribution chart based on the LB_D

如图 9 所示，从分布来看，大部分数据集基于 LB_D 的剔除率在 10%以下，但是有部分数据集具有较高的剔除率.

5.3　数据平滑性影响分析

从 5.2 节分析可知，D-NNS 在阶段 1 和阶段 2 时是基于 LB_D 进行过滤的，由 LB_D 的定义可知，LB_D 是在全局约束下滑动窗口内两条时间序列之间的一种特殊的欧式距离，这种欧氏距离由两条时间序列滑动窗口内最小值和最大值之间的最小差值决定的，因此 LB_D 对时序数据的局部变化比较敏感.时序数据局部变化越剧烈，LB_D 越小，剔除率越低.

一种提高基于 LB_D 剔除率的方法是对数据进行平滑，以减少数据的局部变化.这种方法在很多应用场景是合理可行的，特别是对于长期性的、趋势性或者需要进行除噪和平滑处理的时间序列数据的分析和处理.

为了验证上述方法的可行性，实验采用首先基于滑动窗口，分别计算表 2 中数据集所有时间序列的滑动窗口内的平均值，用该平均值替代原有数据，滑动窗口长度分别设置为时间序列长度的 0%，5%和 10%，用于验证不同滑动窗口长度下的实验效果；然后在平滑后的数据集上运行 D-NNS 算法，对

数据进行平滑之后的实验结果如表 2 中最后两列所示，平均剔除率如图 10 所示.从实验结果看，随着滑动窗口的增大，即数据平滑程度的增大，剔除率明显增大.当滑动窗口长度为整个序列长度的 5%时，平均剔除率增加到 25%左右，滑动窗口长度为整个序列长度的 10%时，平均剔除率增加到 40%左右.

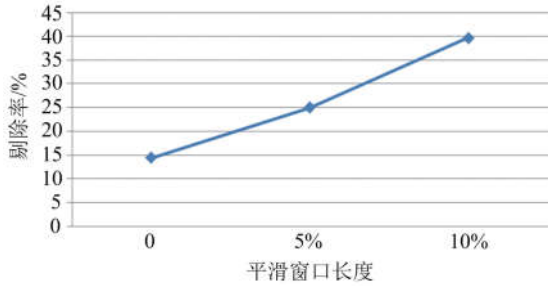

图 10 平滑处理后的基于 LB_D 剔除率

Fig.10 Abandon ratio based on the LB_D after smooth-processing

5.4 优化策略讨论

在提高基于 LB_D 的剔除率方面，除了对时间序列数据进行平滑处理之外，还可以在阶段 2，在现有索引结构上增加 B+树之类的索引，加快搜索速度.如果样本集合 Θ 较大，也可以在样本集合上建立索引，并且在阶段 1 过程中采用早期丢弃策略找出最佳样本.另外，D-NNS 算法阶段 3 可以用现有算法替代，如 UCR_DTW，这样可以利用现有研究成果来提高算法效率，包括计算 LB_Keogh 函数值或 DTW 距离时可以采用累积计算方法，并在计算过程中进行早期过滤；还可以结合去掉开根号运算、基于 LB_KimFL 早期过滤等方法提高计算效率和准确度.

6 结论

本文首次针对动态时间弯曲距离的快速推导难题，提出了可推导的动态时间弯曲近似距离以及相应的索引构建方法和相似时间序列查询算法.大量实验表明，本文提出的近似距离和距离推导算法能够有效实现 1 对多时间序列比较时的距离快速推导，在时间复杂度和 I/O 代价两方面都是高效的.此外，本文还讨论了提高时间序列过滤效果的优化策略.下一步，将在此基础上针对大规模时间序列数据的相似度计算开展深入研究.

参考文献(References)

[1] ADAMS N, MARQUEZ D, WAKEFIELD G. Iterative deepening for melody alignment and retrieval [C]//Proceedings of the 6th International Society Music Information Retrieval Conference. London: IEEE Press, 2005: 199-206.

[2] ALON J, ATHITSOS V, YUAN Q, et al. A unified framework for gesture recognition and spatiotemporal gesture segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 31(9): 1685-1699.

[3] BEGUM N, KEOGH E. Rare time series motif discovery from unbounded streams[J]. Proceedings of VLDB Endowment, 2014, 8(2): 149-160.

[4] CHADWICK N, MCMEEKIN D, TAN T L. Classifying eye and head movement artifacts in EEG signals[C]// 4th International Conference on Digital Ecosystems & Technologies. Shenzhen, China: IEEE Press, 2011: 285-291.

[5] BEGUM N, ULANOVA L, WANG J, et al. Accelerating dynamic time warping clustering with a novel admissible pruning strategy[C]// Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Sydney: ACM Press, 2015: 49-58.

[6] FERREIRA L N, ZHAO L. Time series clustering via community detection in networks [J]. Information Sciences, 2016, 326(C): 227-242.

[7] LI L, PRAKASH B A. Time series clustering: Complex is simpler! [C]// The 28th International Conference on Machine Learning. Washington: ACM Press, 2011: 185-192.

[8] ZAKARIA J, MUEEN A, KEOGH E. Clustering time series using unsupervised-shapelets[C]// Proceedings of the 12th International Conference on Data Mining. Brussels: IEEE Press, 2012: 785-794.

[9] ZHOU J, ZHU S F, HUANG X D, et al. Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning[J]. Journal of Computer Science and Technology, 2015, 30(4), 859-873.

[10] PAPARRIZOS J, GRAVANO L. k-Shape: Efficient and accurate clustering of time series[C]// Proceedings of the ACM SIGMOD International Conference on Management of Data. Melbourne, Australia: ACM Press, 2015: 1855-1870.

[11] ZOU P C, WANG J D, YANG G Q, et al. Distance metric learning based on side information autogeneration for time series[J]. Journal of Software, 2013,24(11): 2642-2655.

[12] DING H, TRAJCEVSKI G, SCHEUERMANN P, et al. Querying and mining of time series data:

Experimental comparison of representations and distance measures [J]. Proceedings of the VLDB Endowment, 2008, 1(2):1542-1552.

[13] RAKTHANMANON T, CAMPANA B, MUEEN A, et al. Searching and mining trillions of time series subsequences under dynamic time warping [C]// Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Beijing: ACM Press, 2012: 262-270.

[14] LI J Q. Research on time series similarity[D]. Wuhan, Huazhong University of Science and Technology, 2008.

[15] YI B K, JAGADISH H V, FALOUTSOS C. Efficient retrieval of similar time sequences under time warping [C]// Proceeding of the 14th International Conference on Data Engineering. Orlando, USA: 1998: 201-208.

[16] KEOGH E, CHAKRABARTI K, PAZZANI M, et al. Dimensionality reduction for fast similarity search in large time series databases [J]. Knowledge and Information Systems, 2001, 3(3): 263-286.

[17] KEOGH E, CHAKRABARTI K, PAZZANI M, et al. Locally adaptive dimensionality reduction for indexing large time series databases[J]. ACM Transations on Database Systems, 2002, 27(2): 188-288.

[18] LI Z X, ZHANG F M, LI K W, et al. Index structure for multivariate time series under DTW distance metric [J]. Journal of Software, 2014, 25(3): 560-575.

[19] KIM S, PARK S, CHU W. An index-based approach for similarity search supporting time warping in large sequence databases [C]// Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany: IEEE Press, 2001: 607-661.

[20] KEOGH E, RATANAMAHATANA C A. EXact indexing of dynamic time warping[J]. Knowledge and Information Systems, 2005, 7(3): 358-386.

[21] SAKOE H, CHIBA S. Dynamic programming algorithm optimization for spoken word recognition[J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1978, 26(1). 43-49.

[22] RABINER L, JUANG B. Fundamentals of Speech Recognition [M]. Englewood Cliffs, N. J, Prentice Hall, 1993.

[23]The UCR suite: Funded by NSF IIS - 1161997 II [EB/OL].[2016-10-12] http://www.cs.ucr.edu/~eamonn/UCRsuite.html.

[24] The UCR time series classification archive[EB/OL]. [2016-10-12] www.cs.ucr.edu/~eamonn/time_series_data/.

[25] KAUFMAN L, ROUSSEEUW P. Finding Group in Data: An Introducing to Cluster Analysis [M]. New York, Willey, 1990.

附录

定理 2.1　设时间序列 C,C' 长度为 v，全局约束条件值为 r，有 LB_D$(C,C')=$LB_D(C',C).

证明　由 LB_D 下界函数定义可知：

$$\text{LB\_D}(C,C')=\sqrt{\sum_{i=1}^{v}\text{diff}_i^2},\ \text{LB\_D}(C',C)=\sqrt{\sum_{i=1}^{v}\text{diff}'^2_i}$$

其中，

$$\text{diff}_i=\begin{cases}c'^U_i-c^L_i, & c'^U_i<c^L_i\\ c'^L_i-c^U_i, & c'^L_i>c^U_i,\\ 0, & \text{其他}\end{cases}$$

$$\text{diff}'_i=\begin{cases}c^U_i-c'^L_i, & c^U_i<c'^L_i\\ c^L_i-c'^U_i, & c^L_i>c'^U_i,\\ 0, & \text{其他}\end{cases}$$

$$c^U_i=\max(c_{i-r},\cdots,c_{i+r}),c^L_i=\min(c_{i-r},\cdots,c_{i+r}),$$

$$c'^U_i=\max(c'_{i-r},\cdots,c'_{i+r}),c'^L_i=\min(c'_{i-r},\cdots,c'_{i+r}).$$

对于 $\forall i(1\leqslant i\leqslant v)$，有 $\text{diff}_i^2=\text{diff}'^2_i$，因此 LB_D$(C,C')=LB_D(C',C)$.

定理 2.2　设全局约束条件值为 r，对于任意的长度为 v 的时间序列 Q,C,C'，有 LB_D$(C,C')\leqslant$ LB_Keogh$(C,Q)+$LB_Keogh(C',Q).

证明　根据 LB_D 函数定义，可以构造时间序列 $\tilde{C}=\{\tilde{c}_1,\tilde{c}_2,\cdots,\tilde{c}_v\}$ 和 $\tilde{C}'=\{\tilde{c}'_1,\tilde{c}'_2,\cdots,\tilde{c}'_v\}$，并且

$$\tilde{c}_i=\begin{cases}c^L_i, & c'^U_i<c^L_i\\ c^U_i, & c'^L_i>c^U_i\\ 0, & \text{其他}\end{cases},\tilde{c}'_i=\begin{cases}c'^U_i, & c'^U_i<c^L_i\\ c'^L_i, & c'^L_i>c^U_i.\\ 0, & \text{其他}\end{cases}$$

其中，

$$c_i^U=\max(c_{i-r},\cdots,c_{i+r}),c_i^L=\min(c_{i-r},\cdots,c_{i+r}),$$
$$c'^U_i=\max(c'_{i-r},\cdots,c'_{i+r}),c'^L_i=\min(c'_{i-r},\cdots,c'_{i+r}).$$

这样，LB_D(C,C') 就转换为 $\tilde{C}$ 和 $\tilde{C}'$ 的欧氏距离，即

$$\text{LB\_D}(C,C')=\text{ED}(\tilde{C},\tilde{C}')=\sqrt{\sum_{i=1}^{v}(\tilde{c}'_i-\tilde{c}_i)^2}.$$

构造时间序列 $\tilde{Q}=\{\tilde{q}_1,\tilde{q}_2,\cdots,\tilde{q}_v\}$，其中，

$$\tilde{q}_i=\begin{cases}c_i^L, & c'^U_i<c_i^L<q_i\\ q_i, & c'^U_i<q_i<c_i^L\\ c'^U_i, & q_i<c'^U_i<c_i^L\\ c_i^U, & c'^L_i>c_i^U>q_i\\ q_i, & c'^L_i>q_i>c_i^U\\ c'^L_i, & q_i>c'^L_i>c_i^U\\ 0, & \text{其他}\end{cases}.$$

时间序列 $\tilde{Q}$ 是在 Q 的基础上，把记录值在 $c'^U_i<c_i^L$ 时限定在 $[c'^U_i,c_i^L]$ 中，在 $c'^L_i>c_i^U$ 时限定在 $[c'^L_i,c_i^U]$ 中.

下面讨论 $\tilde{Q}$ 和 $\tilde{C}$，$\tilde{C}'$ 之间的欧式距离，以及 LB_Keogh 函数值之间的关系.

(a) $c'^U_i<c_i^L$ 时，根据 LB_Keogh 函数定义，LB_Keogh$(C,Q)=\sqrt{\sum_{i=1}^{v}\text{diff}_i^2}$，其中，

$$\text{diff}_i=\begin{cases}q_i-c_i^L, & q_i<c_i^L\\ q_i-c_i^U, & q_i>c_i^U\\ 0, & \text{其他}\end{cases}.$$

显然，$\text{diff}_i{}^2\geqslant(\tilde{q}_i-c_i^L)^2$，因此 LB_Keogh$(C,Q)\geqslant\text{ED}(\tilde{C},\tilde{Q})$.

类似的，可以推导出 LB_Keogh$(C',Q)\geqslant\text{ED}(\tilde{C}',\tilde{Q})$.

(b) $c'^L_i>c_i^U$ 时，与情况(a)类似，根据 LB_Keogh 函数定义，可以推导出 LB_Keogh$(C,Q)\geqslant\text{ED}(\tilde{C},\tilde{Q})$ 以及 LB_Keogh$(C',Q)\geqslant\text{ED}(\tilde{C}',\tilde{Q})$.

(c)对于 $c'^U_i<c_i^L$ 和 $c'^L_i>c_i^U$ 之外的情况，根据定义可知 $\text{ED}(\tilde{C},\tilde{Q})=\text{ED}(\tilde{C},\tilde{Q}')=0$，LB_Keogh$(C,Q)\geqslant 0$，LB_Keogh$(C,Q')\geqslant 0$.因此 LB_Keogh$(C,Q)\geqslant\text{ED}(\tilde{C},\tilde{Q})$，且 LB_Keogh$(C',Q)\geqslant\text{ED}(\tilde{C}',\tilde{Q})$.

综合(a)、(b)、(c)情况，有

LB_Keogh$(C,Q)\geqslant\text{ED}(\tilde{C},\tilde{Q})$ 且 LB_Keogh$(C',Q)\geqslant\text{ED}(\tilde{C}',\tilde{Q})$.

根据欧氏距离三角不等式有

$$\text{ED}(\tilde{C},\tilde{C}')\leqslant\text{ED}(\tilde{C},\tilde{Q})+\text{ED}(\tilde{C}',\tilde{Q}),$$

因此有

$$\text{LB\_D}(C,C')=\text{ED}(\tilde{C},\tilde{C}')\leqslant\text{ED}(\tilde{C},\tilde{Q})+\text{ED}(\tilde{C}',\tilde{Q})\leqslant\text{LB\_Keogh}(C,Q)+\text{LB\_Keogh}(C',Q).$$