基于关键点引导的 u-shapelet 时间序列聚类算法

陈 梅†, 王 钰

(兰州交通大学 电子与信息工程学院，兰州 730070)

摘 要: 针对大多数基于 u-shapelet 的时间序列聚类方法未能同时兼顾 u-shapelet 提取效率和质量的问题, 提出一种基于关键点引导的 u-shapelet 时间序列聚类算法 UKey. 首先, 从时间序列数据集中随机采样一个时间序列子集, 采用所提出两步法识别采样时间序列中的关键点; 然后, 利用这些关键点提取子序列得到 u-shapelet 候选集, 这一策略不仅确保所提取的候选子序列包含关键波动区域, 还能有效减少候选子序列的数量; 接着, 引入 Davies-Bouldin (DB) 指数作为一种新的子序列质量评估方法, 旨在通过综合考虑类间分离度与类内紧凑性, 以确保所获取的 u-shapelet 集合具有较高的质量; 最后, 采用k-Means 算法对基于 u-shapelet 集合构建的距离矩阵进行聚类. 在 10 个不同数据集上的实验结果表明, UKey 算法的性能优于 14 种对比算法, 具有较高的准确性和可解释性.

关键词: 时间序列；u-shapelet；聚类；关键点；子序列提取

中图分类号: TP301.6 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2025.0021

引用格式: 陈梅, 王钰. 基于关键点引导的 u-shapelet 时间序列聚类算法 [J]. 控制与决策, 2025, 40(10): 3117-3126.

A u-shapelet time series clustering algorithm based on key points guidance

CHEN Mei†, *WANG Yu*

(School of Electronic and Information Engineering，Lanzhou Jiaotong University，Lanzhou 730070，China)

Abstract: Focusing on the problem that most existing time series clustering algorithms based on u-shapelet (unsupervised-shapelet) fail to simultaneously balance the efficiency and quality of u-shapelet extraction, a u-shapelet time series clustering algorithm based on key points guidance named UKey is proposed. Firstly, a subset of time series is selected by random sampling from the time series dataset. A two-step method is proposed to identify the key points in the sampling time series. Then, these key points are utilized to extract subsequences to obtain the u-shapelet candidate set. This strategy not only ensures that the extracted candidate subsequences capture the key fluctuation regions but also effectively reduces the number of candidate subsequences. Subsequently, the Davies-Bouldin (DB) index is introduced as a new quality evaluation method, aiming to ensure that the obtained u-shapelet set exhibits high quality by comprehensively considering inter-class separability and intra-class compactness. Finally, the k-Means is used to cluster the distance matrix constructed based on the u-shapelet set. Experimental results on 10 different datasets demonstrate that the UKey outperforms 14 comparison algorithms, achieving higher accuracy and better interpretability.

Keywords: time series；u-shapelet；clustering；key points；subsequences extraction

0 引 言

时间序列是按照时间顺序以固定间隔记录的一系列数据点, 广泛存在于金融[1]、工程[2]、生物医学[3]等领域. 作为数据挖掘的重要分支之一, 时间序列聚类的主要目标是在缺乏先验知识的情况下, 将具有相似模式或特征的时间序列归类为同一簇, 从而揭示时间序列中的潜在模式和趋势[4]. 随着时间序列数据规模的持续增长, 如何高效地进行时间序列聚类分析已经成为数据挖掘领域的重要研究课题[5].

传统的时间序列聚类方法通常关注于处理整个

收稿日期: 2025-01-07；**录用日期:** 2025-03-19.

基金项目: 国家自然科学基金项目 (62266029)；甘肃省高等教育产业支撑计划项目 (2022CYZC-36)；甘肃省重点研发计划项目 (24YFGA036).

责任编委: 李新宇.

†通信作者. E-mail: mei.chen.lzjtu@hotmail.com.

时间序列数据[6]. 然而, 由于时间序列数据通常包含关键的局部模式, 并且具有高信噪比特征, 仅依赖全局特征进行聚类往往难以准确捕捉这些重要的模式变化, 最终的结果也只能反映数据而非类别间的实际差异, 可解释性较差. 因此, 研究基于局部特征的时间序列聚类方法具有重要的现实意义.

2012 年, Zakaria 等[7] 在 BruteForce 方法中首次提出了 u-shapelet 的概念. 该方法通过分析时间序列中具有判别能力的局部特征, 以捕捉不同类别之间的差异, 实现对不同类别时间序列的高效区分. 与传统的时间序列聚类方法相比[8-10], u-shapelet 不仅可以准确捕捉时间序列的局部模式, 还具有较强的可解释性和较高的聚类准确性, 因此受到了学术界的广泛关注. 然而, BruteForce 方法存在时间复杂度过高的问题. 具体而言, 如果一个数据集有n条时间序列, 每条时间序列的长度为m, 则 BruteForce 的时间复杂度为$O(n^2m^4)$.

为解决时间复杂度过高的问题, 许多学者致力于加速 u-shapelet 选择过程. Ulanova 等[11] 通过符号聚合近似 (symbolic aggregate approximation, SAX) 对时间序列数据进行转换, 然后从大量子序列中筛选出 1% 的子序列加入 u-shapelet 候选集. Zakaria 等[12] 通过采用剪枝策略和优化候选 u-shapelet 的选择顺序来加速 u-shapelet 的发现. Luo 等[13] 引入局部敏感哈希算法检索查询子序列的相似序列, 然后根据所代表的数量判断其是否有可能成为 u-shapelet, 通过这种方法排除相似的子序列以减少候选 u-shapelet 集合的量级. Yu 等[14] 通过多元 top-k 查询技术去除冗余子序列, 以减少候选 u-shapelet 的数量. 李晨等[15] 构建一个两阶段的筛选框架, 在一定程度上缩小了候选子序列集合的规模. 王子豪等[16] 提出了一种基于 u-shapelets 的拓扑识别方法, 在计算子序列距离时通过缓存部分中间结果将算法的均摊时间复杂度由线性降到常数. 尽管这些方法一定程度上提高了 u-shapelet 的发现效率, 但它们并未直接从时间序列的局部特征及趋势着手, 且实现过程相对复杂. 此外, 现有的 u-shapelet方法大多使用 gap 函数 (gap function) 评估子序列的质量. 然而, gap 函数仅通过衡量被子序列划分的两个子集之间的分离度来确认子序列的质量, 并未同时考虑被划分的子集之间的分离度以及子集内部的紧凑性. 因此, 选择合适的子序列质量评估方式, 使提取到的 u-shapelet 更具判别力, 也是基于 u-shapelet的时间序列聚类方法需要解决的问题之一.

为了解决上述问题, 本文从候选子序列选择和子序列质量评估两个方面进行改进, 提出一种基于关键点引导的 u-shapelet 时间序列聚类算法 UKey. 首先, 算法聚焦于时间序列中的关键特征, 通过寻找时间序列中的关键点以缩小候选子序列的选取范围. 这一策略不仅有效减少了候选集的规模, 而且确保提取的子序列具有更高的代表性. 其次, 选用 DB 指数 (davies-bouldin index) 作为评估子序列质量的新方法, 通过综合考虑类间分离度与类内紧凑性来改善 u-shapelet 集合的辨识度, 进而提升算法聚类的准确性.

1 相关理论基础

时间序列数据集D是n个时间序列$\{T_1, T_2, \ldots, T_n\}$的集合. 具体而言, 每个实例$T_i(1 \leqslant i \leqslant n)$都包含一组有序的实数值, 表示为$T_i = \{t_1, t_2, \ldots, t_m\}$, 其中$m$为$T_i$的长度.

定义 1(子序列) 一个子序列S是一条时间序列的连续序列. 从时间序列T_i的位置j开始的长度为l的子序列S可以表示为$S = T_{j,l} = t_j, t_{j+1}, \ldots, t_{j+l-1}$.

定义 2(子序列距离) 子序列S与时间序列T_i之间的子序列距离$\mathrm{sdist}(S, T_i)$定义为子序列S与长度等于S的T_i的所有可能子序列之间的最小距离.

定义 3(u-shapelet 候选集) u-shapelet 候选集包含通过筛选得到的所有候选子序列.

定义 4(u-shapelet) 最具辨识度的时间序列片段. 一个 u-shapelet$(\hat{S})$可以根据划分标准$\mathrm{sdist}(\hat{S}, D_A) \ll \mathrm{sdist}(\hat{S}, D_B)$将数据集$D$分成$D_A$和$D_B$两个子集.

定义 5(u-shapelet 距离矩阵) 如果 u-shapelet 集合为$\{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_p\}$, 则基于该 u-shapelet 集合所构建的距离矩阵可以表示为

$$\mathrm{DIST} = \begin{bmatrix} \mathrm{sdist}(\hat{S}_1, T_1) & \cdots & \mathrm{sdist}(\hat{S}_p, T_1) \\ \vdots & \ddots & \vdots \\ \mathrm{sdist}(\hat{S}_1, T_n) & \cdots & \mathrm{sdist}(\hat{S}_p, T_n) \end{bmatrix}. \tag{1}$$

定义 6(gap 函数) 用于评估候选 u-shapelet(S')的质量, 有

$$\mathrm{gap} = \mu_B - \sigma_B - (\mu_A + \sigma_A). \tag{2}$$

其中: μ_A、μ_B分别为$\mathrm{sdist}(S', D_A)$和$\mathrm{sdist}(S', D_B)$的均值, σ_A、σ_B分别为$\mathrm{sdist}(S', D_A)$和$\mathrm{sdist}(S', D_B)$的标准差, D_A和D_B为用S'对数据集D进行划分得到的两个子集.

2 基于关键点引导的 u-shapelet 聚类算法

本文提出一种基于关键点引导的 u-shapelet

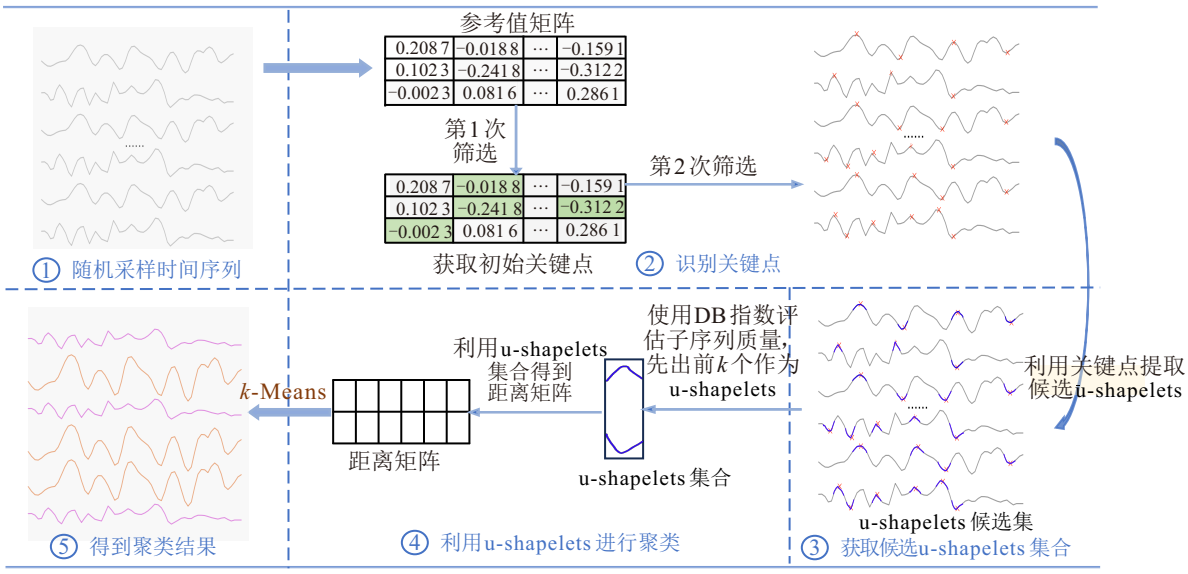


图1　UKey 算法流程

时间序列聚类算法 UKey, 整体框架如图 1 所示. 该算法主要包括 5 个步骤: 首先, 通过随机采样从数据集中选取部分时间序列 (①); 然后, 采用提出的两步法对所选时间序列进行关键点识别 (②); 其次, 根据关键点提取子序列以获得 u-shapelet 候选集 (③); 最后, 利用 Davies-Bouldin(DB) 指数评估候选子序列质量, 从中筛选出前k个子序列加入u-shapelet 集合, 并通过k-Means 算法对基于该u-shapelet 集合构建的距离矩阵进行聚类 (④); 最终获得聚类结果 (⑤).

2.1　随机采样

在基于 u-shapelet 的时间序列聚类中, 每一个提取到的 u-shapelet 都可以被视为某个类的核心公共特征. 与该 u-shapelet 同属于一个类别的时间序列, 都会有一个与 u-shapelet 相似的子序列, 这些子序列都可以得到与 u-shapelet 相同的聚类结果. 这一特点使得在时间序列聚类过程中, 无需对数据集中的所有时间序列的子序列进行全面计算和比较来发现 u-shapelet.

为了减少计算量, 本文采用随机采样的策略, 从数据集中选取部分时间序列用于 u-shapelet 的发现. 其中随机采样遵循均匀抽样原则, 即每条时间序列被选中的概率是相等的.

2.2　两步法: 识别关键点

图 2 展示了 Car 数据集的 4 条不同类别的时间序列, 其中红色片段为这 4 个类别对应的 u-shapelet, 反映了每个类别的核心公共特征.

可以看出, 好的 u-shapelet 通常包含时间序列中的关键波动区域, 这些波动区域常发生在某个关键点附近[17]. 因此, 如果能识别出时间序列中的关键点, 并以此为依据提取 u-shapelet, 则可以显著减少候选

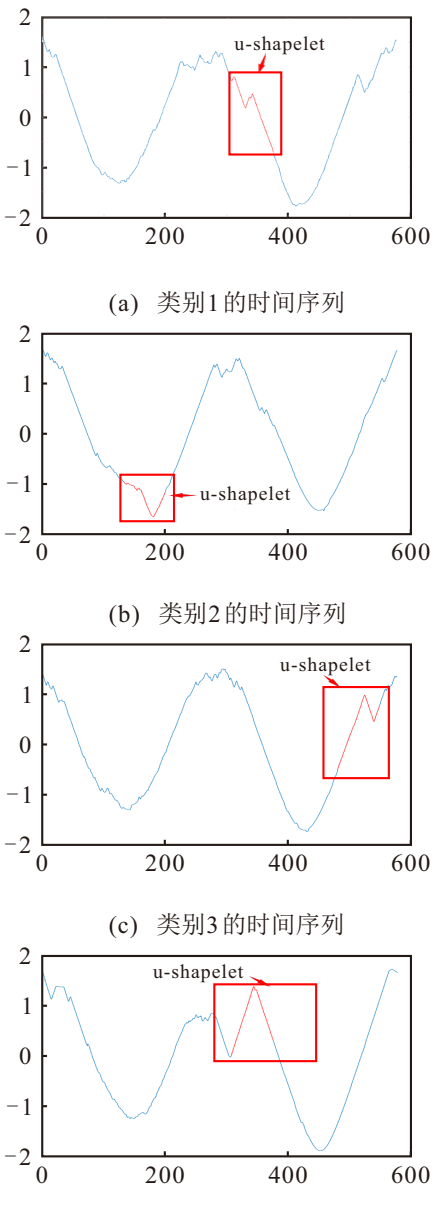


图2　Car 数据集的 4 个 u-shapelets 可视化展示

u-shapelet 的数量, 从而提高算法的效率. 请注意, 时间序列的关键点对应的是图 2 中的某些 x 坐标. 为了根据关键点提取候选 u-shapelet, 需要将这些坐标映射回原始时间序列, 从而定位出对应的时间序列片段.

两步法具体步骤如下所示.

step 1: 计算参考值.

转折点是指时间序列中当前时间点与其前后两个时间点的观测值之间出现趋势方向变化的点, 即序列在该点处由上升变为下降或由下降变为上升的转折位置. 转折点的识别对于捕捉时间序列的局部特征至关重要[18]. 为此, 引入参考值公式来识别时间序列中的转折点. 对于每条采样的时间序列, 计算除去头尾数据点$t_j(2 \leqslant j \leqslant m-1)$的参考值, 有

$$\text{ref} = (t_j - t_{j-1}) \times (t_{j+1} - t_j). \tag{3}$$

其中: t_j为时间序列中的第j个数据点, $t_j - 1$和$t_j + 1$分别为其前后相邻的数据点. 参考值用于衡量数据点两侧的趋势变化, 负值反映两侧斜率的方向差异, 绝对值大小反映变化幅度. 当参考值为负且绝对值较大时, 表明该数据点左右两侧的形态发生了变化, 即该点为可能的波动点.

假设采样得到p条长度为m的时间序列, 则会得到一个$p(m-2)$的参考值矩阵. 通常, 具有高辨识度的子序列位于时间序列形态波动较为显著的位置. 因此, 本文将参考值为负的数据点选定为初始关键点.

step 2: 识别出关键点.

在初始关键点的基础上, 通过结合关键点的参考值和一阶差分来共同衡量相邻关键点之间的变化幅度 (即梯度变化), 以进行关键点的二次筛选. 具体而言, 对于任意一条时间序列的关键点, 假设两个相邻关键点p和q的参考值分别为ref_p和ref_q, 则其一阶差分的计算公式为

$$\Delta p, q = |\text{ref}_q - \text{ref}_p|. \tag{4}$$

若一阶差分较大, 表明相邻关键点之间的变化幅度较大; 反之, 则变化幅度较小. 因此, 本文从初始关键点中筛选出一阶差分大于设定阈值α的关键点, 用于后续候选 u-shapelet 的提取. 这种做法一方面避免了距离过近且变化幅度过小的关键点被同时选择; 另一方面确保了保留下来的关键点均为局部波动显著的点.

2.3 获取候选 u-shapelet 集合

时间序列中的每个子序列都可以视为一个候选 u-shapelet. Zakaria 等[7] 使用穷举法从时间序列数据集中提取并评估所有子序列, 以寻找 u-shapelet, 但该方法时间复杂度过高. 为此, 在本步骤中, 仅提取包含关键点的子序列作为候选 u-shapelet. 具体而言, 将关键点视作候选 u-shapelet 的中间位置, 并从其左右两侧提取固定长度的序列片段. 假设关键点位置为t_j, 子序列长度为slen, 则提取的候选 u-shapelet 的范围为$[t_{j-\text{sLen}/2}, \ldots, t_j, \ldots, t_{j+\text{sLen}/2}]$. 这种提取方法具有以下两个显著优势:

1) 确保包含关键点. 每个提取的候选 u-shapelet 均包含至少一个关键点, 从而保证了子序列的代表性和区分能力.

2) 保留波动区域. 提取的每个候选 u-shapelet 都具有波动, 有助于识别具有辨别能力的序列片段.

通过这种方式, 不仅能显著减少候选 u-shapelet 的数量, 而且可以保证所提取的子序列质量.

2.4 利用 u-shapelet 进行聚类

在获取 u-shapelet 候选集后, 现有的基于 u-shapelet 的时间序列聚类方法大多采用 gap 值评估子序列的质量. 然而, gap 值仅通过衡量被子序列划分后两个子集之间的分离度来评估子序列的质量, 并没有考虑子集中数据的紧凑性. 如果两个子集之间的分离度较大, 但各自内部数据分布较为分散, 则 gap 值仍然可能较高, 然而这并不代表这个子序列的质量高. 因此, 一个优秀的子序列质量评估方法应该同时考虑被划分的子集之间的分离度以及子集内部的紧密程度.

DB 指数是一种衡量聚类结果质量的指标, 用于评估聚类的两个重要性质: 簇内紧密度和簇间分离度. 其中, 簇内紧密度衡量同一簇中数据点彼此之间的相似程度, 而簇间分离度则衡量不同簇之间的距离. DB 指数综合考虑了这两个因素, 其具体定义为

$$\text{DB} = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d_{i,j}}\right). \tag{5}$$

其中: k为簇的数量; σ_i为簇i内所有点到簇中心的平均距离, 反映簇内的紧密程度; $d_{i,j}$为簇i与簇j簇中心之间的距离, 体现簇间的分离度. DB 指数的值越小, 表明簇内点分布越紧密, 簇间分离越明显.

基于此, 本文采用 DB 指数作为子序列质量评估方法. 从候选 u-shapelet 集合中选择 DB 指数最小的前k个子序列, 构成最终的 u-shapelet 集合. 然后, 利用该集合构建一个与时间序列数据集相关的距离矩阵, 并通过k-means 对距离矩阵进行聚类, 从而获得最终的聚类结果.

2.5 算法聚类过程

UKey 可分为以下步骤: 1) 随机抽样选取部分时

间序列; 2) 采用两步法识别关键点; 3) 获取 u-shapelet 候选集; 4) 基于 u-shapelet 距离矩阵进行聚类. UKey 的详细过程如算法 1 所示.

算法 1　UKey 聚类过程.

input: 数据集D, u-shapelet 长度sLen, 簇数k, 阈值α;
output: 聚类结果C.
step 1: 从数据集D中随机采样p条时间序列
step 2: 使用两步法识别出p条时间序列的关键点
1 keypoint ← [ ]; //关键点集合
2 for $i \leftarrow 1$ to p do
3　for $j \leftarrow 1$ to m do
4　　使用式 (3) 计算数据点的参与值ref;
5　　if ref < 0 then
6　　将该数据点加入keypoint;
7 计算keypoint中数据点的一阶差分;
8 从keypoint中移除一阶差分小于α的数据点;
step 3: 获取 u-shapelets 候选集.
1 allUsh ← [ ];
2 db ← 0;
3 foreach point ∈ keypoint do
4 allUsh ←提取以该数据点为中心且长度为 sLen 的子序列;
step 4: 利用 u-shapelets 进行聚类.
1 finalUsh ← [ ];
2 Dis ← [ ];
3 DB ← [ ];
4　foreach ush ∈ allUsh do
5　db ← 使用式 (4) 计算ush的 DB 值;
6　DB ← db;
7　finalUsh ← DB 值最小的前k个子序列;
8 for cnt ← 1to k do
9　$\hat{S} \leftarrow$ finalUsh[cnt];
10　dis ← sdist ($\hat{S}$, D);
11　Dist ← Dist ∪ dis;
12 $C \leftarrow k$-means(Dist, k);
13 return C

2.6　时间复杂度

UKey 算法的时间复杂度取决于每一步骤的时间复杂度. 在 step 1, 假设从数据集D中采样p条时间序列, 时间复杂度为$O(p)$. 在 step 2, 寻找p条时间序列的初始关键点的时间复杂度为$O(p \cdot m)$. 假设有q个初始关键点, 根据一阶差分进行二次筛选的时间复杂度为$O(q)$, 则这一步的时间复杂度为$O(p \cdot m + q)$. 在 step 3, 根据关键点集合提取子序列的时间复杂度为$O(p \cdot m)$, 根据 DB 指数选取前k个子序列的时间复杂度为$O(n \cdot m)$, 则这一步的时间复杂度为$O(p \cdot n \cdot m^2)$. 在 step 4, 构建一个k个 u-shapelet 和n条时间序列的距离矩阵的时间复杂度为$O(k \cdot n \cdot m)$; 使用k-means 聚类, 迭代次数为t, 时间复杂度为$O(t \cdot k \cdot n)$, 则这一步的时间复杂度为$O(k \cdot n \cdot m + t \cdot k \cdot n)$.

综上, 总的时间复杂度为$O(p + p \cdot m + q + p \cdot n \cdot m^2 + k \cdot n \cdot m + t \cdot k \cdot n)$, 其中 $p, q, t \ll n$, $k \ll n$, 因此 UKey 的时间复杂度最终可简化为$O(n \cdot m^2)$.

3　实验与分析

为了验证 UKey 算法的可行性和有效性, 在配备 20 GB 内存和 2.30 GHz CPU 的 Windows 10 电脑上进行实验. 实验中, 阈值α均设置为 0.001.

3.1　实验设置

3.1.1　数据集介绍

本节在 10 个广泛使用的 UCR 时间序列数据集上对 UKey 算法进行实验评估. 详细信息汇总于表 1. 在以下实验中, UKey 的参数sLen取值范围为 [5, m/2], 其中m为时间序列的长度.

表1　数据集信息

Beef (BEE)	5	60	470	光谱
Meat (MEA)	3	120	488	光谱
Wine (WIN)	2	111	234	光谱
Car (CAR)	4	120	577	传感器
FaceAll (FAL)	14	2250	131	图像
OSULeaf (OSU)	6	442	427	图像
SwedishLeaf (SWL)	15	1125	128	图像
WordSynonyms (WDS)	25	905	270	图像
ToeSegmentation2 (TS2)	2	166	343	运动
GunPoint (GUN)	2	200	150	运动

3.1.2　对比算法介绍

将 UKey 与属于 3 种不同类型的 14 种时间序列聚类算法进行比较: 1) 基于 u-shapelet 的算法, 包括 BruteForce[7]、SUSH[11]、USSL[19]、FOTS[20] 和 SE-Shapelets[21]; 2) 基于特征选择的算法, 包括 UDFS[22]、NDFS[23]、RUFS[24]、RSFS[25] 和 RUSLP[26]; 3) 传统算法, 包括 k-means[10]、k-shape[6]、R-Clustering[8] 和 Time-C[27].

在实验中, 基于特征选择的方法除 RUSLP 外, 其余 4 类算法因缺少源码无法复现, 实验结果引自文献 [19]. 鉴于此, 在后续的实验结果分析中, 本文未将这 4 类算法纳入 ARI 指标的对比分析. 传统算法中, k-means 和 k-shape 通过 Python 的“scikit-

learn”库实现, 其余算法的代码由其原作者提供. 为使每种比较算法达到最佳性能, 本文按照相应文献中规定的参数设置, 对每种算法进行 30 次测试, 以获得其最佳结果.

3.1.3 聚类评价指标

实验中使用兰德系数 (Rand index, RI)、归一化互信息 (normalized mutual information, NMI) 和调整兰德指数 (adjusted Rand index, ARI)3 种经典评价指标评估聚类效果.

1) 兰德系数: 考虑真阳性 (true positive,TP)、真阴性 (true negative,TN)、假阳性 (false positive, FP) 和假阴性 (false negative, FN) 4 个因素, 计算公式如下:

$$\mathrm{RI}=\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FN}+\mathrm{FP}}. \tag{6}$$

其中: TP 表示在真实标签和聚类结果中被正确分配到同一簇的样本对数量; TN 表示在真实标签和聚类结果中被正确分配到不同簇的样本对数量; FP 表示在真实标签中被分配到不同簇, 但在聚类结果中被分配到同一簇的样本对数量; FN 表示在真实标签中被分配到同一簇, 但在聚类结果中被分配到不同簇的样本对数量.

2) 归一化互信息: 衡量真实标签与聚类结果之间的信息一致性, 计算公式如下:

$$\mathrm{NMI}(\Omega,C)=\frac{2\times I(\Omega,C)}{H(\Omega)+H(C)}. \tag{7}$$

其中: $I(\Omega,C)$表示聚类结果和数据真实分布的互信息, $H(\Omega)$和$H(C)$分别表示聚类结果和数据真实分布的熵.

3) 调整兰德指数: 用于衡量两个数据分布的吻合程度, 为 RI 的调整形式, 计算公式如下:

$$\mathrm{ARI}=\frac{\mathrm{RI}-\mathrm{E}[\mathrm{RI}]}{\max(\mathrm{RI})-\mathrm{E}[\mathrm{RI}]}, \tag{8}$$

其中 E[RI] 为 RI 的期望值.

3 个指标的取值越大表明聚类质量越高.

3.2 精度实验

表 2 ～ 表 4 记录了 UKey 与对比算法在 10 个真实数据集上的聚类结果, 其中最佳结果以粗体标出. 观察表 2 ～ 表 4 可以看出, UKey 在 RI、NMI 和 ARI 上均表现出色. 尤其在 RI 上, UKey 的平均值显著高于其他对比算法, 展现了其在时间序列聚类方面的显著优势.

表2 UKey 与 14 个对比算法在 10 个数据集上聚类结果的量化评价 RI

数据集	u-shapelet 方法						特征选择方法					传统方法			
	UKey	BruteForce	SUSH	USSL	FOTS	SE-Shapelets	UDFS	NDFS	RUFS	RSFS	RUSLP	*k*-Means	*k*-shape	R-Clustering	Time-C
BEE	**0.7531**	0.6870	0.7446	0.6401	0.7514	0.6476	0.6759	0.7034	0.7149	0.6975	0.6729	0.6713	0.5402	0.6774	0.3853
MEA	**0.9574**	0.6742	0.5359	0.7440	0.6500	0.6902	0.6483	0.6635	0.6578	0.6657	0.7702	0.6595	0.6575	0.8271	0.7294
WIN	**0.5348**	0.4975	0.4975	0.4958	0.5200	0.4961	0.4987	0.5123	0.5021	0.5033	0.4965	0.4961	0.5011	0.5073	0.4975
CAR	0.7197	0.4136	0.4136	**0.7345**	0.7171	0.6287	0.6757	0.6260	0.6667	0.6708	0.5976	0.6345	0.7028	0.7318	0.6803
FAL	0.9005	**0.9011**	0.8861	0.7206	0.8401	0.6197	0.5671	0.8652	0.8247	0.4982	0.8642	0.8667	0.7825	0.8968	0.8577
OSU	0.7341	0.5525	0.5239	0.6551	0.7548	0.6982	0.5372	0.5622	0.5497	0.5665	0.7153	0.5615	0.5538	**0.7782**	0.7558
TS2	**0.9187**	0.5020	0.7981	0.6778	0.4800	0.6780	0.5257	0.5968	0.5968	0.5826	0.4976	0.5257	0.5257	0.6780	0.6795
GUN	**0.7298**	0.6278	0.5773	0.5363	0.6382	0.4975	0.5029	0.5102	0.6498	0.4994	0.4975	0.4975	0.6278	0.6218	0.4975
WDS	0.8775	0.8230	0.8479	0.8540	0.8658	0.7918	0.8697	0.8760	0.8861	0.8817	0.8900	0.8775	0.7844	**0.8984**	0.8536
SWL	0.9005	0.6154	0.3451	0.8547	0.8727	0.8986	0.4923	0.5500	0.5192	0.5038	0.8858	0.8761	0.5333	**0.9355**	0.8573
Average	**0.8026**	0.6294	0.6170	0.6913	0.7090	0.6646	0.5994	0.6466	0.6568	0.6070	0.6888	0.6666	0.6209	0.7552	0.6794

从数据集的角度看, UKey 在 10 个数据集上的 RI、NMI 和 ARI 的平均值为 0.802 6、0.478 1 和 0.397 0, 分别比第 2 名高 0.047 4、0.073 5 和 0.068 8. 这表明 UKey 在时间序列聚类中具有较高的准确性和鲁棒性. 具体而言, UKey 在 MEA、TS2 和 GUN 等数据集体量较小且数据特征显著的数据集上表现尤为突出. 在这些数据集上, UKey 的 RI 值分别比第 2 名高出 0.130 3、0.120 6 和 0.080 0, 充分展示了其有效捕捉具有显著局部特征的能力. 这一优势在 NMI 和 ARI 指标上同样得到了体现. 在这类数据集上, UKey 的 NMI 和 ARI 值均位列第 1, 显著优于其他对比算法. 在 FAL 上, 虽然 UKey 的 RI、NMI 和 ARI 值比 BruteForce 低 0.000 6、0.030 5 和 0.091 3, 但差距较小. 此现象可能归因于 BruteForce 算法通过穷举法找到了最具辨识度的u-shapelets 进行聚类, 从而在 FAL 上取得略微的优势. 而在 SWL 上, UKey 的表现略逊于 R-Clustering算法. 这可能是因为 SWL 属于图像类数据集, 数据集体量较大, 而 R-Clustering 算法擅长处理这类数据. 然而, UKey 在 CAR、OSU 和 WDS 数据集上表现相对较弱. 这可能是由于 UKey 算法通过随机抽样获得的时间序列包含较少的高辨识度的局部特征区域, 因此在这些数

表3　UKey 与 14 个对比算法在 10 个数据集上聚类结果的量化评价 NMI

数据集	u-shapelet 方法						特征选择方法					传统方法			
	UKey	BruteForce	SUSH	USSL	FOTS	SE-Shapelets	UDFS	NDFS	RUFS	RSFS	RUSLP	*k*-Means	*k*-shape	R-Clustering	Time-C
BEE	**0.3802**	0.2185	0.2612	0.2001	0.1597	0.3211	0.2718	0.3647	0.3799	0.3597	0.2406	0.2499	0.0985	0.2954	0.2651
MEA	**0.8997**	0.6224	0.5486	0.7277	0.6805	0.6807	0.2832	0.2416	0.1943	0.3016	0.5456	0.6375	0.3290	0.6337	0.5067
WIN	**0.0586**	0.0498	0.0529	0.0014	0.0465	0.0008	0.0045	0.0259	0.0065	0.0096	0.0010	0.0035	0.0038	0.0183	0.0025
CAR	0.3430	0.2044	0.3228	0.3491	0.2509	0.2174	0.2319	0.2361	0.2511	0.2920	0.1783	0.1732	0.1953	**0.4975**	0.2622
FAL	0.5133	**0.5438**	0.5114	0.4303	0.4719	0.2196	–	–	–	–	0.2383	0.1111	0.1452	0.5063	0.4750
OSU	0.4238	0.2657	0.2143	0.0983	0.4314	0.1035	0.0200	0.0352	0.0246	0.0463	0.1188	0.1701	0.3574	**0.4359**	0.2534
TS2	**0.7043**	0.4559	0.5560	0.6241	0.4173	0.2039	0.0727	0.1713	0.1713	0.1625	0.0003	0.0197	0.2848	0.1379	0.2298
GUN	**0.4623**	0.3038	0.1816	0.2595	0.0406	0.3585	0.0220	0.0334	0.2405	0.0152	0.0000	0.0020	0.0010	0.3437	0.0001
WDS	0.4715	0.1693	0.1368	0.1188	0.1663	0.2644	0.4745	0.5396	**0.5623**	0.5462	0.3737	0.1377	0.0786	0.4101	0.3260
SWL	0.5240	0.2917	0.2108	0.3248	0.3961	0.4520	0.0082	0.0934	0.0457	0.0269	0.3737	0.3358	0.3358	**0.6584**	0.4791
Average	**0.4781**	0.3219	0.3039	0.3260	0.3224	0.2882	0.1543	0.1935	0.2085	0.2667	0.2033	0.1767	0.1923	0.4046	0.2816

表4　UKey 与 10 个对比算法在 10 个数据集上聚类结果的量化评价 ARI

数据集	u-shapelet方法						传统方法				特征选择方法
	UKey	BruteForce	SUSH	USSL	FOTS	SE-Shapelets	*k*-Means	*k*-shape	R-Clustering	Time-C	RUSLP
BEE	**0.3419**	0.2198	0.2258	0.2005	0.3333	0.2353	0.0831	0.0314	0.0833	0.0773	0.0667
MEA	**0.9035**	0.6030	0.5827	0.6499	0.2945	0.6239	0.5967	0.276	0.6191	0.4197	0.4928
WIN	**0.0696**	0.0529	0.0335	0.0511	0.0498	– 0.0059	– 0.0082	– 0.003	0.0147	– 0.0036	0.0052
CAR	0.2969	0.1426	0.2549	**0.3929**	0.3531	0.2628	0.1095	0.1287	0.3565	0.1755	0.0968
FAL	0.3524	**0.4437**	0.4078	0.3041	0.2839	0.3392	0.0555	0.0597	0.3188	0.2934	0.1034
OSU	**0.3815**	0.3138	0.2917	0.2698	0.2778	0.2396	0.0979	0.3023	0.3447	0.1605	0.0744
TS2	**0.8259**	0.6632	0.7623	0.7047	0.7613	0.7276	– 0.0035	0.3433	0.2551	0.342	– 0.0061
GUN	**0.4599**	0.4013	0.3045	0.4525	0.3900	– 0.0051	– 0.0047	0.004	0.2472	– 0.0049	– 0.0051
WDS	0.1629	0.1092	0.1210	0.0092	0.0101	0.0986	0.0916	0.0356	**0.1719**	0.1582	0.1391
SWL	0.3152	0.3327	0.1433	0.1938	0.2734	0.2496	0.0843	0.0843	**0.4324**	0.2434	0.1483
Average	**0.3970**	0.3282	0.3128	0.3229	0.3008	0.2765	0.1102	0.1262	0.2844	0.1862	0.1116

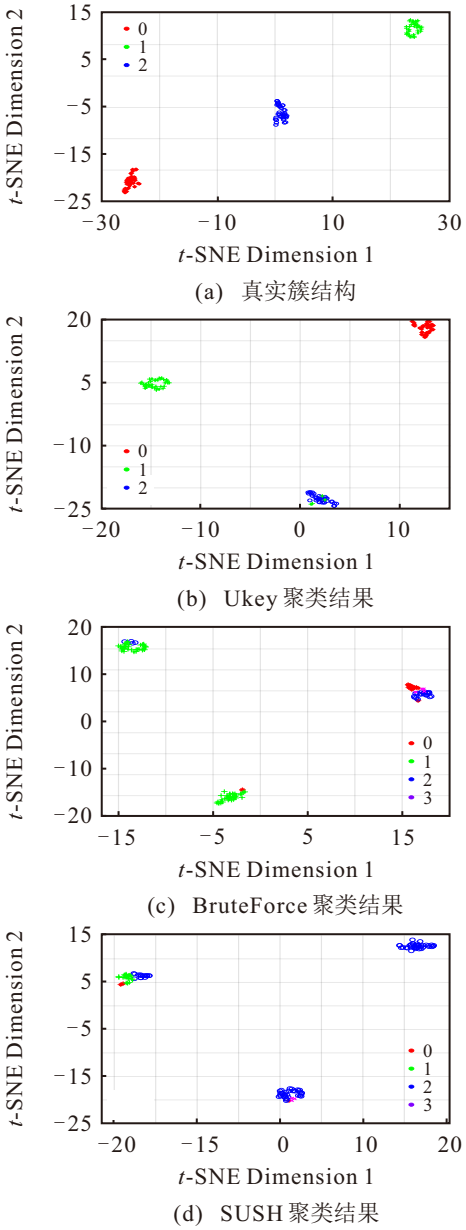
据集上的性能有限.

从对比算法角度看, UKey 显著优于 u-shapelet 方法中的 4 种对比算法. 以 MEA 数据集为例, UKey 的 RI 值为 0.957 4, 与 BruteForce、SUSH、USS 和 FOTS 相比分别提高了 0.283 2、0.421 5、0.213 4 和 0.3074. 这表明 UKey 能有效地从时间序列中挖掘出关键的局部特征. 与 5 种基于特征选择方法相比, UKey 在大多数数据集上的表现都更好. 特别在 SWL 数据集上, UKey 的 RI 值达到了 0.900 5, 与 UDFS、NDFS、RUFS、RSFS 和 RUSLP 相比, 分别提高了 0.408 2、0.350 5、0.381 3、0.396 7 和 0.0147. 然而, 在 WDS 数据集上, UKey 的 RI 值低于 RUFS、RSFS 和 RUSLP. 这可能是因为 WDS 数据集的特征相对统一, UKey 更注重局部特征, 而特征选择方法更善于捕捉全局模式. 尽管如此, UKey 在其余 9 个数据集中始终表现出较好的性能. 在传统方法中, R-Clustering 的 RI 平均值为 0.755 2, 表现最佳, 但仍低于 UKey 的 RI 平均值 0.8026. 其他传统方法的 RI 平均值也明显低于 UKey 的 RI 平均值. 结果表明, 简单的聚类方法难以有效应对时间序列的复杂性, 而 UKey 通过捕捉局部特征进行聚类, 在处理复杂的时间序列数据时具有明显的优势. 此外, 从 NMI 和 ARI 来看, UKey 同样表现出色. 在大多数数据集上, UKey 的 NMI 和 ARI 值均显著高于其他对比算法, 这进一步验证了其在时间序列聚类中的准确性和鲁棒性. 例如, 在 MEA 和 TS2 数据集上, UKey 的 NMI 和 ARI 均位列第 1, 显著优于其他算法. 这表明 UKey 不仅在 RI 上表现优异, 在衡量聚类一致性和准确性的 NMI 和 ARI 指标上也具有显著的优势.

3.3　t-SNE 可视化

由于实验中使用的数据集的特征维度都大于二维, 无法直观地以图形显示. 因此, 本节引入 *t*-SNE (*t*-distributed stochastic neighbor embedding) 降维技术, 在尽可能保留特征信息的情况下, 将高维数据映射到二维空间, 从而实现数据集的可视化展示. 在图 3 中, 将 UKey、BruteForce 和 SUSH 算法在 MEA 数据集上的聚类结果可视化, 并与数据集的真实簇结构进行对比, 以直接观察算法之间的聚类差异, 每个

簇分别用不同的颜色表示.



(a) 真实簇结构

(b) Ukey 聚类结果

(c) BruteForce 聚类结果

(d) SUSH 聚类结果

图3 3 种算法在 MEA 数据集上的聚类结果可视化展示

从图 3(a) 中的 *t*-SNE 可视化结果可以清楚地看出, 数据集中的 3 个簇彼此分离, 簇与簇之间有明显的边界, 没有重叠. 这表明数据集本身的簇结构比较清晰, 各类数据点的分布也比较集中. 在图 3(b) 中, UKey 算法的聚类结果与数据集的真实簇结构基本一致, 成功分离出 3 个簇, 只有极少数数据点被错误归类到蓝色簇中. 这表明, 在处理该数据集时, UKey 算法能够有效捕捉数据的内在结构, 表现出较强的聚类能力. 相比之下, BruteForce 算法和 SUSH 算法的聚类结果与真实的簇结构存在较大差异. 具体而言, SUSH 算法的聚类结果显示簇之间有很大程度的重叠, 不能准确捕捉数据的结构特征. 此外, 由于这两种算法采用迭代法对数据集进行划分, 并没有事先确定聚类的数量, 最终聚类结果中的簇个数与真实的簇个数并不相符.

综上, UKey 算法对 MEA 的聚类效果最好, 能很好地还原数据的真实簇结构, 显示出优越的聚类性能.

3.4 阈值α取值探究

在两步法中, UKey 算法利用阈值α进行关键点的二次筛选. 理想的α应该能够避免距离过近且变化幅度过小的关键点被同时选择, 还需要确保保留下来的关键点均为局部波动显著的点. 因此, 赋予α恰当的数值对后续候选 u-shapelet 的提取至关重要. 本节选取部分数据集进行阈值α的取值探究, α的取值范围被设定为 [0, 0.05], 步长为 0.000 5. 其中, α的最大值被设定为 0.05 是因为当筛选的点过多时, UKey 无法获得多样性的候选 u-shapelet, 从而影响算法聚类效果.

图 4 展示了 UKey 算法的 NMI 指标随阈值α变化的曲线. 从图 4 可以看出, WIN 数据集的 NMI 指标对α值的变化表现出较低的敏感性, 算法在不同α值下的性能相对稳定, 这表明 UKey 算法在该数据集上具有较好的鲁棒性. 相比之下, TS2 和 BEE 数据集的 NMI 指标在α值变化时呈现出一定合理范围内的波动性, 这种波动性反映了 UKey 算法在不同数据分布下的灵活性和适应性. 通过对比不同数据集上的实验结果可以发现, 当α值在 0.001 附近时, UKey 算法在这些数据集上均表现出较好的聚类效果. 因此, 本文将α值固定为 0.001. 这一选择具有双重优势: 既减少冗余点的引入以提升关键点质量, 又确保候选 u-shapelet 集合的多样性和代表性, 避免遗漏重要的局部波动信息.

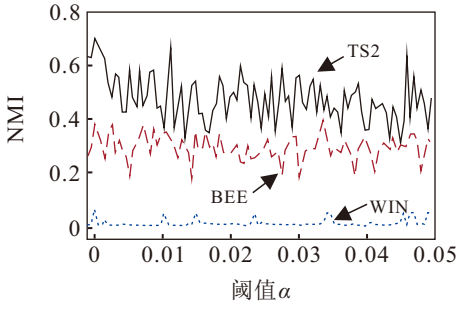


图4 不同阈值α关于 NMI 指标的变化分布

3.5 质量评估方法比较

为了验证 DB 指数评估子序列质量的有效性, 本文采用 UKey 算法进行实验验证. 由于 UKey 需要用户自定义 u-shapelet 的长度, 为保证实验结果的公平性, 两种评估方法 (gap 函数和 DB 指数) 均采用相同的长度值进行验证. 两种评估方法在 10 个数据集上的具体表现如表 5 所示, 最佳结果以粗体标出.

表5　DB 和 gap 比较

数据集	sLen	rand index	
		gap	DB
BEE	24	0.5825	**0.7531**
MEA	70	0.7772	**0.9574**
WIN	5	0.5135	**0.5348**
CAR	74	0.6094	**0.7197**
FAL	32	0.8843	**0.9005**
OSU	10	0.7178	**0.7341**
TS2	58	0.6258	**0.9187**
GUN	22	0.5138	**0.7289**
WDS	10	0.8743	**0.8775**
SWL	12	0.8979	**0.9005**

如表 5 所示, 在 10 个数据集上, DB 指数均优于 gap 函数, 特别是在 TS2、GUN 和 MEA 数据集上, DB 指数的 RI 值分别比 gap 函数的 RI 值高 0.292 9、0.215 1 和 0.180 2. 这表明 DB 指数在评估子序列质量方面十分有效. DB 指数在子序列质量评估中的有效性在于: gap 函数只度量子集间的分离度, 忽略了子集内数据的分布, 而 DB 指数既考虑了子集内的紧凑性, 又考虑了子集之间的分离度.

综上, 实验结果验证了 DB 指数作为子序列质量评估方法的有效性. 通过同时考虑子集内的紧凑性和子集间的分离度, DB 指数能更好地指导 u-shapelet 的选择. 这不仅为时间序列聚类提供了更具辨识度的局部特征, 也为后续聚类结果提供了更可靠的支持.

3.6　运行时间分析

在实验中, 本文比较了 UKey 与 BruteForce 在 10 个数据集上的运行时间, 结果如表 6 所示.

表6　运行时间和加速比

数据集	BruteForce/min	UKey/min	加速比
BEE	11.57	4.39	2.64
MEA	22.68	7.01	3.24
WIN	12.57	5.65	2.22
CAR	59.01	6.45	9.15
FAL	3 534.14	1 041.68	3.39
OSU	471.7	159.37	2.96
TS2	10.80	5.04	2.14
GUN	12.20	1.05	11.62
WDS	4 463.02	180.28	24.76
SWL	635.57	530.65	1.20

由表 6 可知, 在所有数据集上, UKey 都能显著缩短计算时间. 平均而言, UKey 加快了 6.33%. 最大和最小速度分别为 24.76% 和 1.20%. 效率的显著提高主要归功于两种算法在提取候选子序列时采用的不同策略. BruteForce 算法需要从数据集中的所有时间序列中提取所有可能的子序列, 这导致了巨大的计算开销. 相比之下, UKey 算法通过识别部分时间序列中的关键点, 只提取有波动的子序列, 从而大大减少了候选子序列的数量, 显著降低了计算成本. 综上, 在计算效率方面, Morph 明显优于 BruteForce.

4　结　论

本文提出了一种基于关键点引导的 u-shapelet 时间序列聚类算法 UKey. 该算法通过提出的两步法识别时间序列中的关键点, 在有效减少候选子序列数量的同时, 确保了候选子序列具有较高的代表性. 此外, 通过综合考虑类间分离度与类内紧凑性, 引入 DB 指数作为新的子序列质量评估方法, 从而保证了所获取的 u-shapelet集合的高质量. 在 10 个数据集上的实验结果表明, UKey 在聚类准确性和可解释性方面优于 14 种对比算法. 在未来的工作中, 将探索更有效的方法, 进一步优化 u-shapelet 提取的质量和效率.

参考文献 (References)

[1] Setoudehtazangi F, Manouchehri T, Nematollahi A R, et al. Time series clustering based on latent volatility mixture modeling with applications in finance[J]. Mathematics and Computers in Simulation, 2024, 223: 543-564.

[2] 林子谦, 张坤, 樊重俊, 等. 面向分布式系统标签噪声的时间序列分类方法[J]. 控制与决策, 2024, 39(12): 4118-4126.
(Lin Z Q, Zhang K, Fan C J, et al. Time series classification method for tag noise of distributed system[J]. China Industrial Economics, 2024, 39(12): 4118-4126.)

[3] Munir T, Khan M, Cheema S A, et al. Time series analysis and short-term forecasting of monkeypox outbreak trends in the 10 major affected countries[J]. BMC Infectious Diseases, 2024, 24(1): 16.

[4] 陈梅, 柳博雅, 王钰, 等. 基于时间序列形态的模糊聚类算法[J]. 控制与决策, DOI: 10.13195/j.kzyjc.2024.0755.
(Chen M, Liu B Y, Wang Y, et al. Fuzzy clustering algorithm based on time series morphology [J]. Control and Decision, DOI: 10.13195/j.kzyjc.2024.0755.)

[5] Li J B, Izakian H, Pedrycz W, et al. Clustering-based anomaly detection in multivariate time series data[J]. Applied Soft Computing, 2021, 100: 106919.

[6] Paparrizos J, Gravano L. *K*-shape: Efficient and accurate clustering of time series[C]. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. Melbourne Victoria Australia, 2015: 1855-1870.

[7] Zakaria J, Mueen A, Keogh E. Clustering time series using unsupervised-shapelets[C]. 2012 IEEE 12th

International Conference on Data Mining. Brussels, 2012: 785-794.

[8] Marco-Blanco J, Cuevas R. Time series clustering with random convolutional kernels[J/OL]. 2023, arXiv: 2305.10457.

[9] Maciej Ł. Hierarchical clustering of time series data with parametric derivative dynamic time warping[J]. Expert Systems with Applications, 2016, 62: 116-130.

[10] Hotelling H. A generalized T test and measure of multivariate dispersion[C]. Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability. Berkeley: University of California Press, 1951: 23-42.

[11] Ulanova L, Begum N, Keogh E. Scalable clustering of time series with U-shapelets[C]. Proceedings of the 2015 SIAM International Conference on Data Mining. New York, 2015: 900-908.

[12] Zakaria J, Mueen A, Keogh E, et al. Accelerating the discovery of unsupervised-shapelets[J]. Data Mining and Knowledge Discovery, 2016, 30(1): 243-281.

[13] Luo L H, Lv S. An accelerated U-shapelet time series clustering method with LSH[J]. Journal of Physics: Conference Series, 2020, 1631(1): 012077.

[14] Yu S Q, Yan Q Y, Yan X M. Improving U-shapelets clustering performance: An shapelets quality optimizing method[J]. International Journal of Hybrid Information Technology, 2017, 10(4): 27-40.

[15] 李晨, 万源. 基于优化和两阶段筛选的时间序列Shapelets提取研究[J]. 计算机科学, 2023, 50(2): 146-157.
(Li C, Wan Y. Study on time series shapelets extraction based on optimization and two-phase filtering[J]. Computer Science, 2023, 50(2): 146-157.)

[16] 王子豪, 余涛, 黄毅, 等. 基于 *U*-shaplets 特征子序列的新型低压配电网拓扑识别方法[J]. 电测与仪表, https://link.cnki.net/urlid/23.1202.th.20240510.1939.007.
(Wang Z H, Yu T, Huang Y, et al. A novel topology identification method for low-voltage distribution networks based on *U*-shapelets feature subsequence[J/OL]. Electrical Measurement \& Instrumentation, https://link.cnki.net/urlid/23.1202.th.20240510.1939.007.)

[17] Li G L, Yan W H, Wu Z D. Discovering shapelets with key points in time series classification[J]. Expert Systems with Applications, 2019, 132: 76-86.

[18] 张豹, 应励志, 余宇峰. 基于趋势特征的时间序列符号聚集近似表示方法[J]. 计算机应用, 2022, 42(S1): 123-129.
(Zhang B, Ying L Z, Yu Y F. Approximate representation of time series symbol aggregation based on trend features[J]. Journal of Computer Applications, 2022, 42(S1): 123-129.)

[19] Zhang Q, Wu J, Zhang P, et al. Salient subsequence learning for time series clustering[J]. IEEE Trans Pattern Anal Mach Intell, 2019, 41(9): 2193-2207.

[20] Siyou Fotso V S, Mephu Nguifo E, Vaslin P. Frobenius correlation based *u*-shapelets discovery for time series clustering[J]. Pattern Recognition, 2020, 103: 107301.

[21] Cai B R, Huang G Y, Yang S Q, et al. SE-shapelets: Semi-supervised clustering of time series using representative shapelets[J]. Expert Systems with Applications, 2024, 240: 122584.

[22] Yang Y, Shen H T, Ma Z, et al. *l*2, 1-norm regularized discriminative feature selection for unsupervised learning[C]. Proceedings of IJCAI International Joint Conference on Artificial Intelligence. Piscataway: IEEE, 2011: 1589-1594.

[23] Li Z C, Yang Y, Liu J, et al. Unsupervised feature selection using nonnegative spectral analysis[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2012, 26(1): 1026-1032.

[24] Qian M, Zhai C. Robust unsupervised feature selection[C]. Proceedings of the 23rd International Joint Conference on Artificial Intelligence. Piscataway: IEEE, 2013: 1621-1627.

[25] Shi L, Du L, Shen Y D. Robust spectral learning for unsupervised feature selection[C]. 2014 IEEE International Conference on Data Mining. Shenzhen, 2014: 977-982.

[26] Luo C, Zheng J, Li T R, et al. Orthogonally constrained matrix factorization for robust unsupervised feature selection with local preserving[J]. Information Sciences, 2022, 586: 662-675.

[27] Wang X, Song R B, Xiao J M, et al. Accelerating k-shape time series clustering algorithm using GPU[J]. IEEE Transactions on Parallel and Distributed Systems, 2023, 34(10): 2718-2734.

作者简介

陈梅 (1973−), 女, 教授, 博士, 主要研究方向为数据挖掘、人工智能、复杂网络, E-mail: mei.chen.lzjtu@hotmail.com;

王钰 (2000−), 女, 硕士生, 主要研究方向为复杂数据挖掘、时间序列聚类, E-mail: 2420635819@qq.com.