Data Mining and Knowledge Discovery (2021) 35:2369–2388
https://doi.org/10.1007/s10618-021-00798-w

Time series clustering in linear time complexity

Xiaosheng Li[1] · Jessica Lin[1] · Liang Zhao[1]

Received: 6 May 2021 / Accepted: 4 September 2021 / Published online: 18 September 2021

Abstract
With the increasing power of data storage and advances in data generation and collection technologies, large volumes of time series data become available and the content is changing rapidly. This requires data mining methods to have low time complexity to handle the huge and fast-changing data. This article presents a novel time series clustering algorithm that has linear time complexity. The proposed algorithm partitions the data by checking some randomly selected symbolic patterns in the time series. We provide theoretical analysis to show that group structures in the data can be revealed from this process. We evaluate the proposed algorithm extensively on all 128 datasets from the well-known UCR time series archive, and compare with the state-of-the-art approaches with statistical analysis. The results show that the proposed method achieves better accuracy compared with other rival methods. We also conduct experiments to explore how the parameters and configuration of the algorithm can affect the final clustering results.

Keywords Time series · Clustering · Linear time · Symbolic representation

1 Introduction

Time series data widely exist in various scientific disciplines and industrial processes, thus the mining of time series data has attracted substantial interest. Time series clustering is one of the most important tasks in time series data mining. As an unsupervised

Responsible editor: Eamonn Keogh

✉ Xiaosheng Li
xli22@gmu.edu

Jessica Lin
jessica@gmu.edu

Liang Zhao
lzhao9@gmu.edu

1 George Mason University, 4400 University Dr., Fairfax, USA

technique, it does not require the data to be annotated or have class labels. Time series clustering has been applied in a variety of domains including astronomy (Rebbapragada et al. 2009), biology (Subhani et al. 2010), climate (Steinbach et al. 2003), finance (Kumar et al. 2002), and so on (Aghabozorgi et al. 2015).

Time series clustering problem can be formulated as follows. Given a set of unlabeled time series instances, the objective is to place them into separate, homogeneous groups. In this article, we consider the partitional clustering problem of whole time series, i.e. we regard a time series instance as an object and cluster the time series objects into pairwise-disjoint groups.

With the advancement of technologies, for example, lighter, smaller and cheaper sensors widely embedded in various devices and machines, the amount of time series data becomes huge and the content is changing rapidly. This requires the data mining algorithms to have low time complexity. Although there have been a wealth of work on time series clustering, little work is on providing a linear time solution with reasonable performance. Existing super-linear time complexity methods may not be applicable when the dataset is large, or when real-time analytics are required.

In this article we propose a novel time series clustering algorithm, called Symbolic Pattern Forest (SPF), which has linear time complexity. The approach checks if some randomly selected symbolic patterns exist in the time series to partition the data instances. This partition process is executed multiple times, and the partitions are combined by an ensemble process to generate the final partition. Figure 1 shows the framework structure of the proposed method, and the details in the figure will be described later.

We demonstrate that group structures in the data can emerge from the random partition process. Further analysis shows that the ensemble size needed to achieve

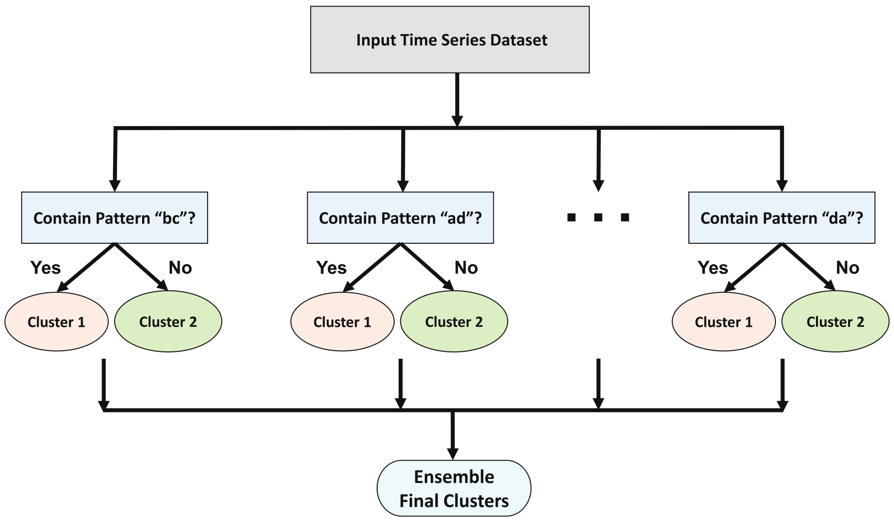

Fig. 1 Framework of SPF, each branch in the figure is a tree and all the branches constitute a forest

good results does not directly depend on the input data size, and thus we can set the ensemble size to a proper fixed value for a specific data pattern.

The application of symbolic patterns in SPF has several benefits. Real-world time series often contain noise, amplitude change, phase-shift and irrelevant portion of data. The normalization step in symbolization can provide scale-invariance against the amplitude change. In the averaging and symbolization step, some noise can be smoothed out. The symbolic patterns do not preserve the pattern location information in the time series, thus they are not affected by phase-shift. If a time series contains a symbolic pattern in some portion, changing the values in other portion does not affect the appearance of the pattern, making SPF robust against irrelevant data.

Further, the utilization of symbolic patterns makes the pattern space finite, and we can use the symbolic patterns to partition the data without using a distance measure. Checking the boolean indicating array to assign clusters in SPF is efficient as boolean operations are very fast. Boolean values are space-efficient which can take more advantage of the CPU cache to speed up the program.

We evaluate the SPF algorithm on all 128 datasets in the well-known UCR time series archive (Dau et al. 2018), and compare with other state-of-the-art approaches with statistical analysis. The results show that SPF is better in accuracy compared with other rival methods. We also conduct experiments to study how the parameters and configuration of the algorithm can affect the final clustering results.

The rest of the paper is organized as follows. Section 2 provides the background and related work. Section 3 provides the details of the proposed method and some analysis. The experimental evaluation is presented in Sect. 4, and Sect. 5 concludes the paper.

2 Background and related work

2.1 Definitions and notations

This subsection provides the definitions and notations to precisely describe the problem under investigation and to present the proposed method.

Definition 1 A time series T is a ordered sequence of real-value data points $[t_1, t_2, \ldots, t_m]$, where m is the length of the time series.

Definition 2 A subsequence S of time series T is a sequence of contiguous values taken from T: $S = [t_i, t_{i+1}, \ldots, t_{i+l-1}]$, where l is the length of the subsequence, $1 \leq i \leq m - l + 1$ and $1 \leq l \leq m$. All subsequences of a certain length from a time series can be extracted using a sliding window of the same length from the first data point to the $(m - l + 1)$-th point.

Definition 3 Given a set of time series $\{T_i\}_{i=1}^{n}$, where n is the number of time series instances, time series partitional clustering assigns a group relationship c_i for each T_i, with $c_i = r_j$, $j \in \{1, 2, \ldots, k\}$. r_j is a group value and k is the number of clusters. Usually we have $k \ll m$ and $k \ll n$. For presentation simplicity, we assume all the time series in the dataset have the same length m. The proposed algorithm in the paper can also work on datasets with varying-length time series.

2.2 Related work

Some research on time series clustering is based on the k-means algorithm (MacQueen 1967). In the k-means algorithm, the clustering group relationship is generated by an iterative refinement procedure. In initialization, k centroids are randomly selected. In each iteration, the distances from the instances to the centroids are computed and the instances are assigned to their nearest centroids. Centroids are then updated according to the new assignment.

In the standard k-means algorithm, Euclidean Distance (ED) (Faloutsos et al. 1994) is used as the distance metric and arithmetic mean is adopted to calculate the centroids. However, it is common for real-world time series data to contain phase-shift, warping, distortion and amplitude change. The simple ED may not be able to cope with these situations. Therefore, many time series distance measures are proposed (Wang et al. 2013), and one of the most popular ones is the Dynamic Time Warping (DTW) (Berndt and Clifford 1994) which can align the data points from the two time series under comparison to find the optimal matching.

Methods of generating centroids under new time series distance measures have been proposed. Examples of these methods are NonLinear Alignment and Averaging Filters (NLAAF) (Gupta et al. 1996), Prioritized Shape Averaging (PSA) (Niennattrakul and Ratanamahatana 2009) and Dynamic Time Warping Barycenter Averaging (DBA) (Petitjean et al. 2011).

K-Spectral Centroid (KSC) (Yang and Leskovec 2011) proposes a distance measure that finds the optimal alignment and scaling for matching two time series. The centroids are generated to minimize the distances between the centroids and the instances under this distance measure.

K-shape (Paparrizos and Gravano 2015) is one of the state-of-the-art time series clustering algorithms based on k-means. It proposes a distance measure called Shape Based Distance (SBD), which is based on the cross-correlation of the time series. The centroids are generated by optimizing the within-cluster squared normalized cross-correlation between the centroids and the time series instances.

Another category of algorithms on time series clustering transform the time series into flat features and then apply classic clustering algorithms on the features to generate the cluster assignment. In (Kumar et al. 2005), the authors transform a time series into a bitmap, which is composed of the counts of all the symbolic patterns in time series. The bitmap representation provides a new distance measure for the classic clustering algorithms to run on time series data.

In the work by Zakaria et al. (Zakaria et al. 2012), the authors propose to enumerate all the subsequences in the time series dataset to select a subset of subsequences called U-shapelets that can best separate the data. The distances between the time series and these subsequences are computed and regarded as new feature values. Finally k-means is applied on the new feature values to get the clustering result. Although the shapelet-based method and our proposed technique both use local shapes in time series for clustering, they are quite different. The shapelet-based method adopts an iterative procedure to refine the clusters, while the proposed algorithm directly partitions the data and combines the partitions to get the clusters.

In (Zhang et al. 2016), instead of enumerating all the subsequences in the time series dataset, the shapelets are learned by optimizing an objective function.

In a recent work (Lei et al. 2019), the Similarity PreservIng RepresentAtion Learning (SPIRAL) method samples pairs of time series to calculate their DTW distances and builds a partially-observed similarity matrix. The matrix is an approximation for the pair-wise DTW distances matrix in the dataset. The new features are generated by solving a symmetric matrix factorization problem, such that the inner product of the new feature matrix can approximate the partially-observed similarity matrix.

With the popularity of deep learning in recent years, there are also methods that adopt the autoencoder architecture for time series clustering. One example is the Deep Temporal Clustering (DTC) (Madiraju et al. 2018). In DTC, Mean Square Error (MSE) is used to measure the reconstruction loss, and the clustering loss is measured by a KL divergence.

2.3 Symbolic aggregate approximation

Since our method uses Symbolic Aggregate approXimation (SAX) (Lin et al. 2007) to transform a time series or subsequence to a symbolic pattern, we briefly describe this technique. Figure 2 shows an example of transforming a subsequence to a symbolic pattern (SAX word). The subsequence is z-normalized and divided into ω segments (ω is 2 in this example). The mean value for each segment is computed (the green line and yellow line in the figure for the two segments respectively). These mean values are mapped to symbols according to a set of break points (the gray lines in the figure). These break points divide the value space in equal-probable regions. In this example the alphabet size of SAX is 4 (with an alphabet of ‘a’, ‘b’, ‘c’ and ‘d’). The subsequence in the figure is transformed to the symbolic pattern “da”. The alphabet size γ, number of segments (word length) ω, and subsequence length l are supplied by the users.

3 Symbolic pattern forest

For clarity, we present a small concrete example to illustrate the idea of the proposed method. Then we provide the formal description and analysis of the algorithm.

3.1 A concrete illustrative example

Figure 3 (Left) shows a small dataset of 4 time series instances belonging to 2 different classes (in blue and red respectively). These time series are taken from the FaceFour dataset from the UCR time series archive (Dau et al. 2018) and the time series in the dataset reflect the face outlines of different individuals under different conditions (Ratanamahatana and Keogh 2004).

Given SAX parameters γ and ω, we can enumerate γ^{ω} all possible symbolic patterns. The proposed method randomly picks a symbolic pattern from all available patterns and, for each time series instance, checks if it contains the pattern. More

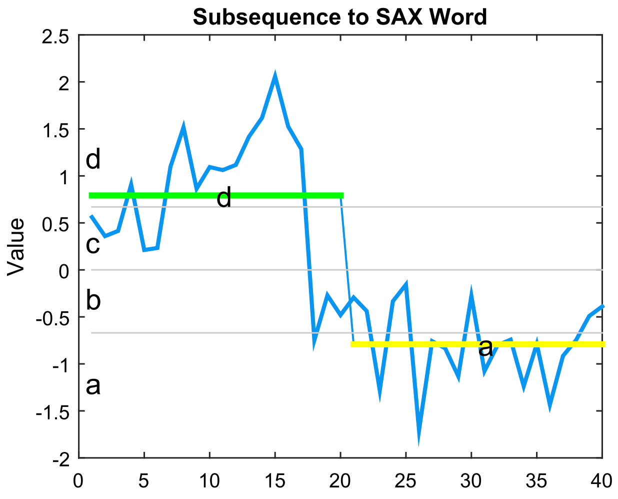

Fig. 2 Transforming a subsequence to a symbolic pattern with Symbolic Aggregate approXimation (SAX)

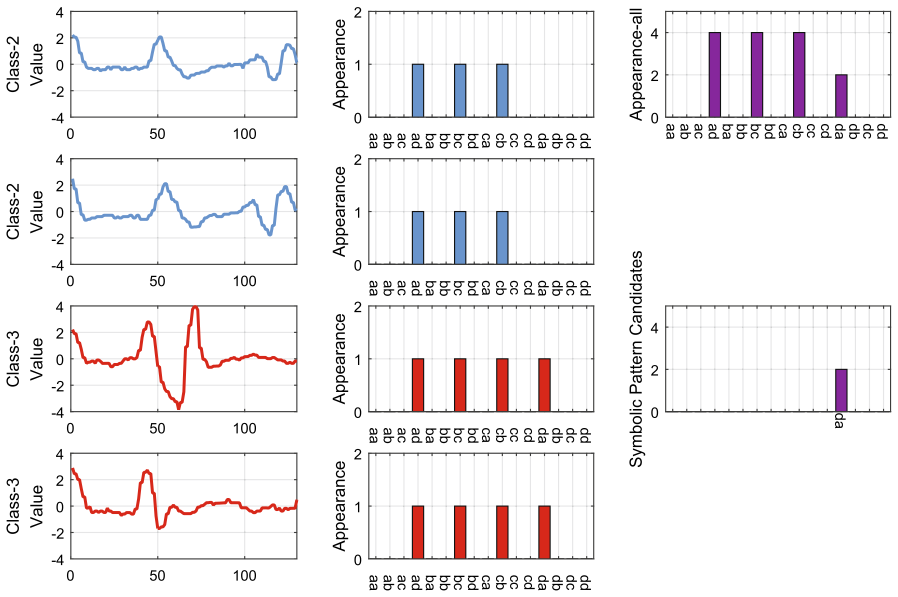

Fig. 3 An illustrative example of the SPF clustering process. (Left) Time series from 2 classes. (Center) Boolean indicating arrays of the time series on the left. (Right) Total appearance count of each pattern and final symbolic pattern candidates

specifically, we extract all subsequences of a pre-defined length l from a time series instance via a sliding window. Each subsequence is transformed to a symbolic pattern and compared with the randomly chosen pattern.

This process is repeated multiple times so here we can optimize the process by scanning the time series in the beginning and storing the appearance of each pattern in a boolean indicating array. Boolean false (0) means the pattern does not appear in the time series, and boolean true (1) indicates the pattern appears. When checking a specific random pattern, we can directly look at the boolean array without needing to scan the time series again. Figure 3 (Center) shows the boolean indicating arrays for the four time series on the left respectively.

According to whether the time series contain a certain randomly selected pattern, the time series are partitioned into two groups. Those containing the pattern go to one group and the others are assigned to the other group. These two groups form two clusters. If the number of clusters k is more than 2, we perform the division with a new random symbolic pattern on the larger group. Previously chosen symbolic patterns are excluded from the pool of available pattern candidates. This process continues until we obtain k clusters, and the tree constructed from the procedure is called Symbolic Pattern Tree (SPT).

The above procedure are repeated multiple times with randomly selected patterns, and multiple trees are constructed as a result, hence the name Symbolic Pattern Forest (SPF). Each tree is a partition of the data, and we combine the partitions in the forest by ensemble to get the final output partition.

The main idea of the symbolic pattern tree is that it uses a symbolic pattern to separate different time series groups. If a pattern appears in all the instances, it cannot separate the instances and thus we can exclude it from the symbolic pattern candidate pool. The same applies to the patterns that do not appear in any instances.

In SPF, the number of time series instances that contain the pattern in the dataset is counted, and the patterns with a count greater than an upper bound or less than a lower bound are excluded from the symbolic pattern candidate pool. The settings of these two bounds will be introduced later. Figure 3 (Right) shows the total occurrence count of each pattern in the dataset, as well as the final symbolic pattern candidate “da”. SPF will select “da”, which can separate the two classes into two clusters correctly.

3.2 SPF algorithm

3.2.1 Cluster ensemble

In SPF, each SPT generates a cluster assignment for all the time series instances, and these clusters are combined by ensemble to generate the final cluster assignment. Hybrid Bipartite Graph Formulation (HBGF) (Fern and Brodley 2004), which has a linear time complexity, is adopted in SPF to perform the cluster ensemble. The idea of HBGF is to build a graph model where the instances and clusters of the ensemble are the vertices. Partitioning the graph generates the ensemble consensus clusters. In our implementation, we use Metis (Karypis and Kumar 1998) to partition the graph.

Figure 4 gives an concrete example of combining two clustering assignments to show the idea of HBGF. The upper part of the figure shows the cluster-instance relationship of the clustering A and the lower part gives the cluster-instance relationship of clustering B. If an instance (gray circle in the figure) belongs to a certain cluster (oval in the figure), there is a black line connecting the respective instance and cluster. The cluster-instance relationship in the figure forms a graph where the circles and ovals

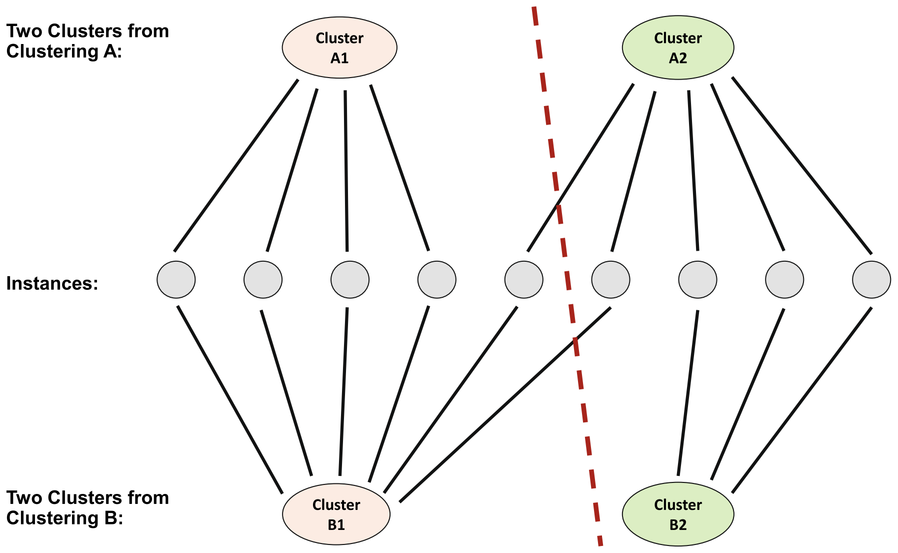
Two Clusters from Clustering A:
Cluster A1
Cluster A2
Instances:
Two Clusters from Clustering B:
Cluster B1
Cluster B2

Fig. 4 An illustrative example of HBGF

are the vertices, and the black lines are the edges. HBGF then partitions this graph (red dashed line) to obtain the ensemble partition of the two clustering partitions A and B.

3.2.2 Efficient SAX computation

In SPF, we need to compute the SAX symbolic pattern of each subsequence in the time series. The cumulative sum technique proposed for BOPF (Li and Lin 2017) is applied to calculate the symbolic pattern of an arbitrary subsequence in $O(1)$ time (given the cumulative sums). The cumulative sum series U and cumulative squared sum series V of a time series T can be computed as: $u_j = \sum_{i=1}^{j} t_i$, $v_j = \sum_{i=1}^{j} t_i^2$, where $j = 1, \ldots, m$ and u_0, v_0 are set to 0.

For a subsequence $x = [t_i, \ldots, t_{i+l-1}]$, the mean value μ_x and standard deviation σ_x can be calculated as: $\mu_x = (u_{i+l} - u_{i-1})/l$, $\sigma_x = \sqrt{(v_{i+l} - v_{i-1})/l - \mu_x^2}$. x is divided in ω segments and each of the segment is mapped to a symbol. The normalized mean value of a segment $y = [t_i, \ldots, t_j]$ is $\mu_y = ((u_j - u_{i-1})/(j - i + 1) - \mu_x)/\sigma_x$. Then μ_y is mapped to a fixed break points region and transformed to a respective symbol (Lin et al. 2007).

3.2.3 Parameters of SAX

In SAX, the number of segments ω, alphabet size γ, and subsequence length l are user-set parameters. In SPF, γ is set to 4 as previous research (Lin et al. 2007) suggests this value is suitable for most datasets. A grid search on the combinations of ω and l is performed. Each combination will generate a cluster assignment and all these assignments

Algorithm 1 Symbolic Pattern Forest (SPF)

```
Input:
    $D$: time series dataset
    $q$: ensemble size
    $k$: number of clusters
Output:
    $C$: cluster assignment of $D$
1: $T = loadData(D)$
2: $[U, V] = CumulativeSum(T)$
3: for each $\omega \in wd$ do
4:     for each $l \in wl$ do
5:         $SP = SymbolicPattern(T, \omega, l, U, V)$
6:         $PC = PatternCount(SP)$
7:         $SPC = FindCandidates(PC)$
8:         for $i = 1$ to $q$ do
9:             $CA = SPT(SPC, SP, k)$
10:            $ES.add(CA)$
11:        end for
12:        $CA2 = Ensemble(ES)$
13:        $ES2.add(CA2)$
14:    end for
15: end for
16: $C = Ensemble(ES2)$
```

are combined by ensemble to give the final algorithm output. ω takes the values from $wd = \{3, 4, 5, 6, 7\}$ and l takes the values from $wl = \{0.025, 0.05, 0.075, \ldots, 1\}m$, i.e. an arithmetic sequence from $0.025m$ to m with a common difference of $0.025m$, where m is the length of the time series. Duplicate values and values less than 10 are removed. In total at most 200 l and ω combinations are tested.

Algorithm 1 gives the pseudo-code of SPF. Given a time series dataset D, an ensemble size q and the number of clusters k, the algorithm returns a cluster assignment C as output.

In Line 1, the dataset D is loaded and stored in T. Line 2 computes the cumulative sum U and cumulative squared sum V from T, which will be used to calculate the symbolic patterns. Lines 3-4 perform the grid search on ω and l as discussed before. Line 5 calculates the symbolic pattern appearance indicating boolean array SP. Line 6 counts the number of occurrence of each symbolic pattern and stores the result in PC. Line 7 takes the pattern count PC to select the symbolic pattern candidate SPC that will be used in the SPT random selection process. The patterns that have a count greater than an upper bound or less than a lower bound are removed from the candidate pool. In our method, we set the lower bound to $0.25 \times n/k$ where n is the number of instances and k is the number of clusters. The upper bound is set to $n - 0.25 \times n/k$. These bounds are set so as to remove non-distinguishing patterns according to our experiments.

In Lines 8-11, SPT uses the symbolic pattern candidates to generate a cluster assignment CA, and CA is added to an ensemble set ES. This process is repeated q times. In Line 12, we combine the cluster assignments in ES by ensemble and get the consensus cluster assignment $CA2$. $CA2$ is added to the ensemble set $ES2$ in Line 13. Finally in

Algorithm 2 Symbolic Pattern Tree (SPT)

```
Input:
    $SPC$: Symbolic Pattern Candidate
    $SP$: Symbolic Pattern appearance indicating arrays
    $k$: number of clusters
Output:
    $CA$: Cluster Assignment
1: $CA = initialize(SP, k)$
2: $curentNode = SP$
3: for $i = 1$ to $k - 1$ do
4:     $RSP, SPC = randomSelect(SPC)$
5:     $Left, Right = partition(curentNode, RSP)$
6:     $CA = setCluster(CA, Left, i)$
7:     $curentNode = Right$
8: end for
```

Line 16, the cluster assignments in $ES2$ is combined by ensemble to obtain the final cluster assignment C as the output of the algorithm.

One purpose of using ES is to reduce the space cost of the algorithm. If the algorithm does not ensemble the partitions in ES and instead saves them in $ES2$, it then needs to store the q partitions for each time series instances for all the SAX parameter combinations. The other consideration of using ES is to prevent certain SAX parameter combinations from dominating the final ensemble. For example, one certain parameter combination may have the same q partitions, in which case, this partition may dominate the final ensemble.

Algorithm 2 gives the pseudo-code of the function SPT used in Algorithm 1, and the idea of SPT is introduced in the previous Sect. 3.1. In Line 1 the array CA is set to represent the cluster assignment. Its length equals the number of time series instances and the array initial values are set to equal k. In Line 2, the current node of SPT is set to contain all the symbolic pattern indicating array instances SP.

In Lines 3-8, in each loop a Random Symbolic Pattern (RSP) is randomly selected from SPC, and this RSP is then removed from SPC. The RSP is used to partition the instances in the current node (Line 5). Those instances containing the RSP form one group and the others constitute another group. The smaller group is assigned to the left node ($Left$) and the larger one becomes the right node ($Right$). In Line 6, the cluster labels of instances in the left node in CA are set to equal the current loop iteration number i. The right node ($Right$) becomes the current node in Line 7. After the loops end, CA is the output of SPT.

3.3 Analysis of SPF

3.3.1 Time complexity

Recall n denotes the number of time series instances, and m denotes the length of time series. The number of clusters, k, is considered as a constant much smaller than m and n. In Algorithm 1, the computation of cumulative sums takes $O(nm)$ time. The number of grid search combinations is at most 200. It takes $O(nm)$ time to obtain the symbolic pattern boolean indicating arrays and the symbolic pattern candidates. SPT takes $O(n)$ time and the ensemble process also takes $O(n)$ time. So the total time complexity of SPF is $O(nm)$, which is linear to the input data size.

3.3.2 Effectiveness of SPF

In this subsection, we show that if there exists a group structure in the dataset related to one pattern or some patterns, then SPF will output such group relationship. The intuition is that the unrelated patterns distribute uniformly among the instances so their effects cancel each other out in the ensemble, and only the structure on the related patterns is maintained and revealed. More formally, We have the following theorem:

Theorem 1 *Consider two instances T_1 and T_2 in the same class, if they agree with each other on some related patterns, then SPF will put them in the same cluster.*

Proof Assume β is the percentage of related patterns in the symbolic candidate patterns. In the random selection process, if a related pattern is selected, then $P(C(T_1) = C(T_2)) = 1$, where $P(\cdot)$ is the probability function and $C(\cdot)$ is the cluster assignment function. If an unrelated pattern is selected, $P(C(T_1) = C(T_2)) = P(C(T_1) \neq C(T_2)) = 1/2$. So overall $P(C(T_1) = C(T_2)) = \beta \times 1 + (1-\beta) \times 1/2$ and $P(C(T_1) \neq C(T_2)) = (1-\beta) \times 1/2$. We have:

$$P(C(T_1) = C(T_2)) > P(C(T_1) \neq C(T_2)) \tag{1}$$

Each tree is independent, according to the law of the large numbers, when we have sufficiently large ensemble size:

$$Count(C(T_1) = C(T_2)) > Count(C(T_1) \neq C(T_2)) \tag{2}$$

where $Count(C(T_1) = C(T_2))$ is the count of cases that T_1 and T_2 are in the same cluster. So in the ensemble result, T_1 and T_2 are assigned in the same cluster. □

In the above analysis, we assume the ensemble size is sufficiently large. The following theorem quantifies how large the size should be.

Theorem 2 *Assume the ensemble size is q, the lower bound of q to obtain a good clustering result is $-2\ln\alpha/\beta^2$, where 1-α is the confidence level, β is the related pattern percentage in the symbolic pattern candidate pool.*

Proof Let X denote the random variable where there are X cases with $C(T_1) = C(T_2)$. Then X follows the binomial distribution:

$$P(X = z) = \binom{q}{z} p^z (1-p)^{q-z} \tag{3}$$

where $p = P(C(T_1) = C(T_2))$. Equation (2) should hold with high probability, so our goal can be formulated as:

$$P(X \leq z) = \sum_{i=0}^{z} \binom{q}{i} p^i (1-p)^{q-i} \leq \alpha \tag{4}$$

where $z = q/2$, $1-\alpha$ is the confidence level, e.g. 95%. Here considering Hoeffding's inequality (Hoeffding 1994):

$$P(E[\bar{X}] - \bar{X} \geq t) \leq e^{-2qt^2} \tag{5}$$

where $t \geq 0$. Considering $E[\bar{X}] = p$, we have:

$$P(E[\bar{X}] - \bar{X} \geq t) = P(qE[\bar{X}] - q\bar{X} \geq qt) \tag{6}$$

$$= P(X \leq qp - qt) \leq e^{-2qt^2} \tag{7}$$

Let $z = qp - qt$, we have $t = (qp - z)/q$:

$$P(X \leq z) \leq e^{-2(qp-z)^2/q} \leq \alpha \tag{8}$$

Recall $z = q/2$, $p = \beta \times 1 + (1-\beta) \times 1/2$, we can solve the above inequality and get:

$$q \geq \frac{-2\ln\alpha}{\beta^2} \tag{9}$$

□

Here is a concrete example of this boundary value: let the confidence level be 99% and assume 50% of the patterns in the pattern candidate pool are related patterns. So $\alpha = 0.01$ and $\beta = 0.5$, we get $q \geq 36.84$.

One observation from equation (9) is that the ensemble size lower bound does not directly depend on the number of instances n and time series length m. In the experimental section, we will show that the accuracy results indeed do not change with fixed ensemble size and varying instances numbers and time series lengths, given the same data characteristics.

4 Experimental evaluation

4.1 Experimental setup

To evaluate the proposed algorithm, we run it on all 128 datasets from the UCR time series archive (Dau et al. 2018). This public archive contains different types of labeled time series from various fields. Each dataset in the archive contains a training set and a testing set. We fuse both sets and use all the data in the experiments. Some datasets contain varying-length time series. We append zeros at the end of the series so that all time series in a dataset have the same length. While our proposed algorithm also works for time series of different lengths, we adjusted the length for the convenience of experiments as some of the compared methods require such input condition. The "NaN" values in some datasets are replaced by the respective interpolation values.

The results of SPF are compared with other rival methods. The widely used k-means algorithm (MacQueen 1967) is selected as the baseline. Standard k-means adopts ED as the distance metric and uses arithmetic mean to calculate centroids. kDBA (Petitjean et al. 2011), KSC (Yang and Leskovec 2011), k-shape (Paparrizos and Gravano 2015), SPIRAL (Lei et al. 2019), DTC (Madiraju et al. 2018) are also run on the same datasets. These methods are described in Sect. 2 and are used as representatives of the state-of-the-arts.

The source code of k-means, kDBA, KSC, k-shape are obtained from the authors of (Paparrizos and Gravano 2015). The source code of SPIRAL[1] and DTC[2] is available online. The number of clusters k is set to equal the number of classes of the dataset in use, and we follow the default parameter settings in the source code. The ensemble size of SPF is set to 100 in all the experiments.

Following (Paparrizos and Gravano 2015), we use the Rand Index to measure the accuracy of the clustering results. Rand Index is defined as:

$$Rand\ Index = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

where TP is the number of instances belonging to the same class and assigned in the same cluster, TN is the number of instances belonging to different classes and assigned in different clusters, FP is the number of instances belonging to different classes but assigned in the same cluster, and FN is the number of instances belonging to the same class but assigned in different clusters.

To verify the time complexity of SPF, we run it on the widely used CBF dataset (Saito and Coifman 1994) of different sizes and record the average running time for each size. This dataset is a synthetic dataset, so with its underlying data generation rules in (Saito and Coifman 1994), we can conveniently generate the datasets with different number of instances and time series lengths.

The C++ source code of SPF is available in the supplementary material[3]. The experiments are conducted in a batch-processing cluster. A single core of AMD Opteron Processor 6276 (2299 MHz) and 16 GB memory are used.

4.2 Experimental results

The methods under comparison are run on the 128 datasets and the Rand Index for each dataset are recorded. The results of SPF are the average results of 10 runs. Due to space limitation, the results for each dataset are not listed here and all the results are available in the supplementary material[3]. Here we present the summarization of the

[1] https://github.com/cecilialeiqi/SPIRAL

[2] https://github.com/FlorentF9/DeepTemporalClustering

[3] https://github.com/xiaoshengli/SPF-DMKD

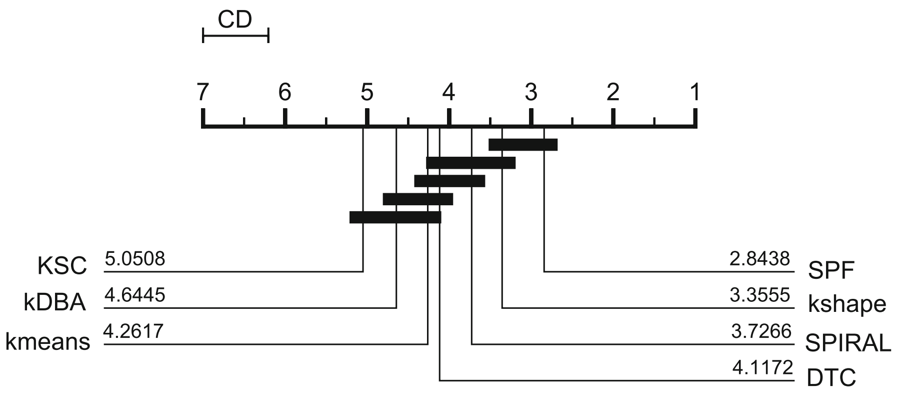

Fig. 5 Critical difference diagram of the comparison on Rand Index with other methods

comparison. Figure 5 shows the critical difference diagram (Demšar 2006) (at 95% confidence level) for the Rand Index comparison. The value beside each method in the figure is the rank mean (lower is better) for the respective method. The methods that are connected by a bold bar have no significant difference at the 95% confidence level.

From the figure one can see the accuracy of SPF is better than that of k-shape, but the difference is not significant. SPF is significantly better than all the other methods: kDBA, KSC, SPIRAL, and DTC. So the overall accuracy of SPF is superior compared to the state of the art techniques.

The time complexities of k-means, k-shape and SPF are $O(nm)$, $O(\max(nm^2, m^3))$, and $O(nm)$ respectively. Compared with k-means, SPF has the same time complexity but is significantly more accurate. Compared with k-shape, SPF has lower time complexity and slightly better accuracy. The total actual running time of k-means, k-shape and SPF on the 128 datasets are 1579, 44765, 2033 seconds (or 26.3, 746.1 and 33.9 minutes) respectively. Note that these methods are implemented in different languages so these time values just show how fast we can cluster the data using the available source code.

Figure 6 shows the average running time of SPF on the CBF datasets. The number of instances is changed from 1000 to 10000 and the length of time series is fixed at 1000. In the figure, the x-axis value is the number of instances and the y-axis value is the average running time of 30 runs. Linear regression curve fitting is performed on the data and the black line in the figure is the fit line. From the figure one can see the R^2 value, which is the coefficient of determination of the fitting, is 0.99356. This value is very close to 1, indicating the average running time of SPF and the number of instances have a strong linear relationship.

Figure 7 gives the average running time of SPF on the CBF datasets of different lengths. The number of instances is fixed at 1000 and the length is changing from 1000 to 10000. Each time value in the figure is the average time of 30 runs. The coefficient of determination R^2 value is 0.99923. This value is very close to 1, indicating the average running time of SPF and the length of time series have a strong linear relationship.

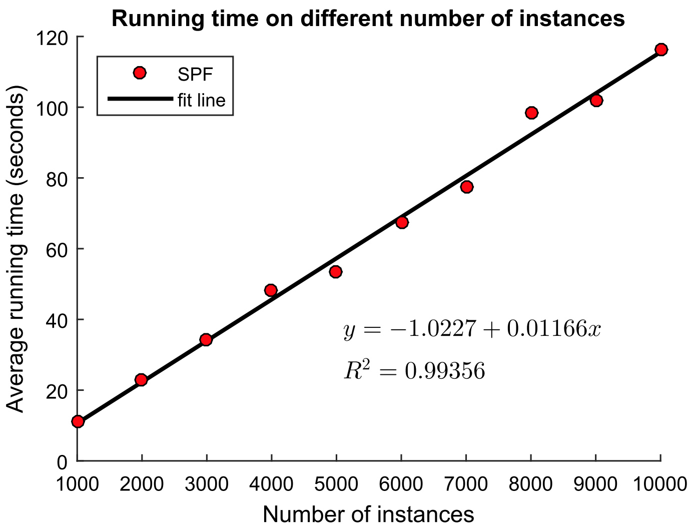

Fig. 6 Running time of SPF on different number of instances

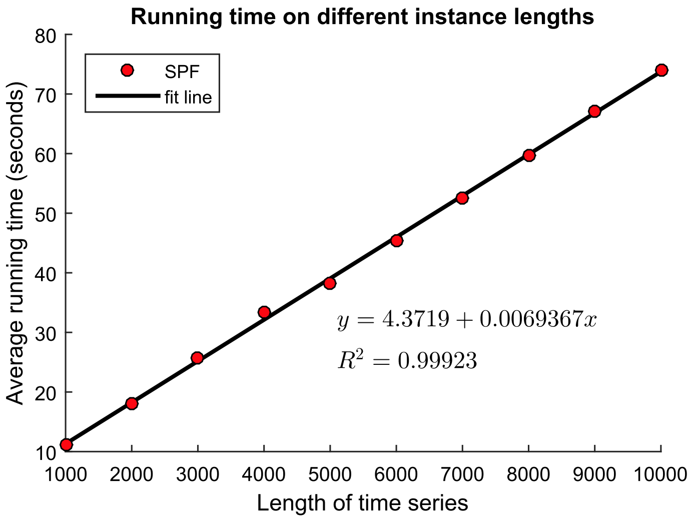

Fig. 7 Running time of SPF on different time series lengths

The above results coincide with the time complexity analysis in Sect. 3.3. Also, in the experiments we record the average Rand Index of SPF on different number of instances and time series lengths combinations. The Rand Index values remain the same at 1 under all the cases. The ensemble size is fixed in the experiment, and this result is in accord with the analysis in Sect. 3.3, that given the data has the same underlying pattern distribution, the ensemble size to obtain good accuracy results does not directly depend on the number of instances or time series lengths.

Figure 8 shows a visual example of how the time series from different classes are partitioned correctly in the SPF process. The left column contains 4 time series instances as well as the total appearance count of each symbolic pattern. The 4 time series belong to 4 different classes from the dataset FaceFour which is one of the 128 tested datasets. The right column of the figure gives the boolean indicating arrays for the respective time series on the left, which shows the appearance of each symbolic pattern in the time series. In this example the number of segments ω is set to 3 and the subsequence length l takes the value of 26. This parameter combination is one of the SAX parameter combinations used in the actual SPF algorithm.

The symbolic patterns in the figure from left to right are from "aaa", "aab", ..., to "ddc" and "ddd" in an ascending alphabetical order. The number of segments, or SAX word length, is 3 and the alphabet size γ is 4, so in total there are $4^3 = 64$ symbolic patterns. The symbolic pattern names are not showed in the figure to avoid overcrowding.

The 4 red vertical dashed lines in the figure align the locations of 4 symbolic pattern candidates in SPC and in each boolean indicating arrays, i.e. each red dashed line connects the same symbolic pattern vertically. From the figure, one can see that each of the 4 aligned symbolic pattern candidates appears only in one of the four classes respectively. More concretely, the left most candidate symbolic pattern appears only in Class 4; the second left most candidate symbolic pattern appears only in Class 2 and so on. So in the SPF process, using these symbolic patterns can partition the 4 classes from each other correctly.

4.3 Parameter study

This subsection presents the experimental results for exploring the effects of the parameters and configuration in SPF. The results in this subsection are all average results of 10 runs with different random seeds. In the above experiments, we set the ensemble size of SPF to 100. Here we halve and double the ensemble size to 50 and 200 respectively. The two SPF versions are denoted as SPF-e50 and SPF-e200 respectively. We run SPF-e50 and SPF-e200 on the 128 datasets and Fig. 9 shows the comparison between SPF and these two versions.

From the figure, one can see that the three versions do not have significant difference. This shows that changing the ensemble size in a certain range does not affect the results significantly on the tested datasets.

SPF adopts SAX to transform time series into symbolic patterns and there are two parameters in the SAX configuration: the number of segments (word length) ω, and subsequence length l. In SPF, ω takes the values from $wd = \{3, 4, 5, 6, 7\}$ and l takes the values from $wl = \{0.025, 0.05, 0.075, \ldots, 1\}m$, i.e. an arithmetic sequence from $0.025m$ to m with a common difference (i.e. the step size) of $0.025m$, where m is the length of the time series. We change the common difference of $0.025m$ for l to $0.01m$ and $0.05m$. The respective two versions are SPF-l01 and SPF-l05. We also vary ω to $\{3, 4, 5, 6\}$ and $\{3, 4, 5, 6, 7, 8\}$ and the two variants are denoted as SPF-d4 and SPF-d6 respectively.

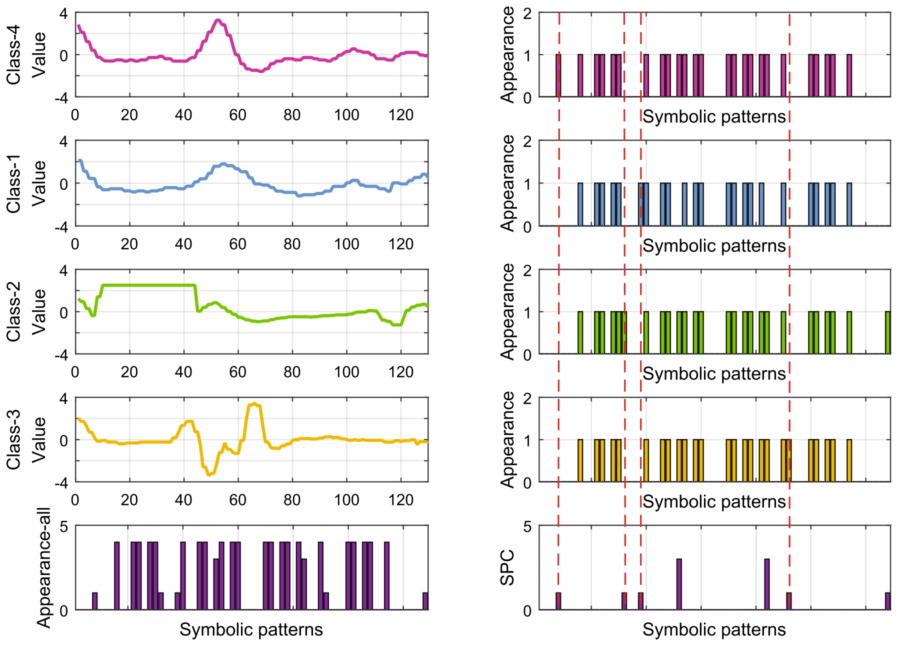

Fig. 8 A visual example of time series from different classes are partitioned correctly in the SPF process. (Left) 4 time series from 4 classes in the tested dataset FaceFour as well as the total appearance count for each symbolic pattern . (Right) Boolean indicating arrays of the respective time series on the left and the Symbolic Pattern Candidates SPC

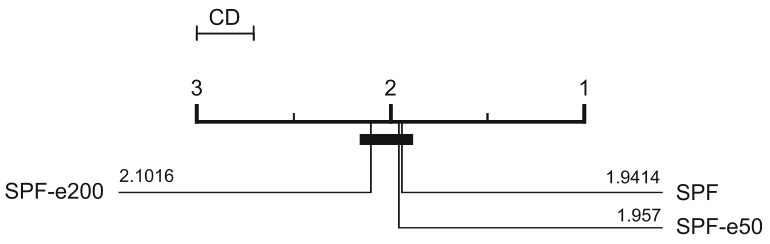

Fig. 9 Critical difference diagram of the comparison between different ensemble sizes (SPF-e50 and SPF-e200)

Figure 10 gives the comparison between SPF, SPF-l01 and SPF-l05 on Rand Index. Figure 11 shows the comparison between SPF, SPF-d4 and SPF-d6. From Fig. 10 one observes that SPF-l01 is slightly better than SPF and SPF-l05 is slightly worse than SPF. This shows that decreasing the common difference, or enlarging the subsequence length set wl can slightly improve the accuracy of SPF. From Fig. 11 one can see that SPF is significantly better than SPF-d4 and significantly worse than SPF-d6. This shows that enlarging the word length set can enhance the accuracy of SPF.

Enlarging the subsequence length set or the word length set can make the SAX symbolic patterns capture more information from the raw time series, thus improve the accuracy of SPF. However, the improvement is at the expense of higher running

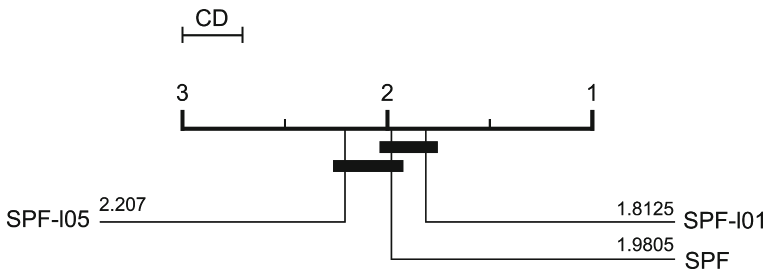

Fig. 10 Critical difference diagram of the comparison between different SAX subsequence length settings (SPF-l01 and SPF-l05)

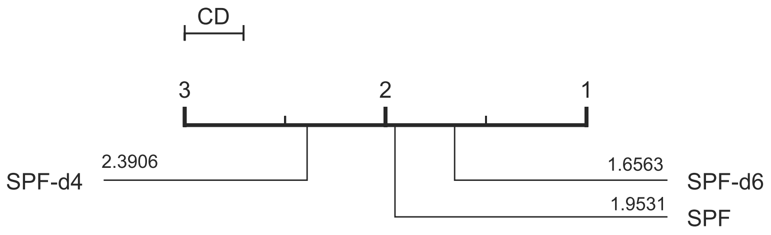

Fig. 11 Critical difference diagram of the comparison between different SAX word length settings (SPF-d4 and SPF-d6)

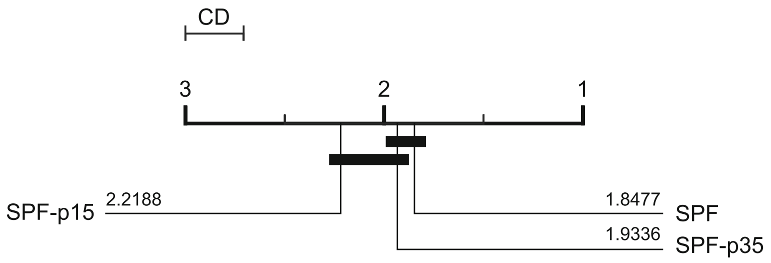

Fig. 12 Critical difference diagram of the comparison between different pattern candidate lower bound settings (SPF-p15 and SPF-p35)

time. The average running time on all 128 datasets for SPF-l05, SPF, and SPF-l01 are 1014, 2033, and 4391 seconds (or 16.9, 33.9 and 73.2 minutes) respectively. The average running time on all 128 datasets for SPF-d4, SPF, SPF-d6 are 1262, 2033, and 2491 seconds (or 21.0, 33.9 and 41.5 minutes) respectively.

In SPF, the pattern candidate lower bound is set to $0.25 \times n/k$ where n is the number of instances and k is the number of clusters. The symbolic patterns that have counts in the dataset less than the lower bound are removed from the symbolic pattern candidate pool. To study the effect of this lower bound value, we change it to $0.15 \times n/k$ and $0.35 \times n/k$, and the respective SPF versions are SPF-p15 and SPF-p35. Figure 12 provides the comparison between SPF, SPF-p15, and SPF-p35 on the 128 datasets.

We can see from the figure that either increasing or decreasing the lower bound value deteriorates the accuracy performance of SPF. If the lower bound value is too small, some noisy symbolic patterns that only appear in a few time series instances will go to the candidate pool. Then they may be selected in the random process and used to partition the time series dataset. This will not bring meaningful results, thus decrease the accuracy of SPF. If the lower bound value is too large, some symbolic patterns that can separate the time series into correct groups may be excluded from the candidate pool, which will harm the clustering of SPF.

5 Conclusion

This article presents a Symbolic Pattern Forest (SPF) algorithm for time series clustering. The method partitions the time series instances by checking some randomly selected symbolic patterns and the partitions of multiple runs are combined to give a final cluster assignment. Analysis is conducted on the time complexity and effectiveness of the algorithm. We evaluate the algorithm extensively on all 128 datasets from the UCR time series archive and the results show that SPF is very competitive compared with other rival methods.

In this work we consider the univariate time series clustering problem but the algorithm can be extended to handle multivariate time series data naturally. One straight-forward way is to use the symbolic patterns in all the dimensions to form the symbolic candidate pool and other parts of the algorithm remain unchanged. In this way the algorithm can cluster multivariate data and the linear-time complexity property can be preserved. Other methods to deal with multivariate data could also work and we plan to explore them in the future.

References

Aghabozorgi S, Shirkhorshidi AS, Wah TY (2015) Time-series clustering-a decade review. Inf Syst 53:16–38

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. KDD workshop, Seattle, WA 10:359–370

Dau HA, Keogh E, Kamgar K, Yeh CCM, Zhu Y, Gharghabi S, Ratanamahatana CA, Yanping, Hu B, Begum N, Bagnall A, Mueen A, Batista G, Hexagon-ML (2018) The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7(Jan):1–30

Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases, vol 23. ACM

Fern XZ, Brodley CE (2004) Solving cluster ensemble problems by bipartite graph partitioning. In: Proceedings of the twenty-first international conference on Machine learning, ACM, p 36

Gupta L, Molfese DL, Tammana R, Simos PG (1996) Nonlinear alignment and averaging for estimating the evoked potential. IEEE Trans Biomed Eng 43(4):348–356

Hoeffding W (1994) Probability inequalities for sums of bounded random variables In the collected works of Wassily Hoeffding. Springer, Berlin, pp 409–426

Karypis G, Kumar V (1998) A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J Sci Comput 20(1):359–392

Kumar M, Patel NR, Woo J (2002) Clustering seasonality patterns in the presence of errors. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 557–563

Kumar N, Lolla VN, Keogh E, Lonardi S, Ratanamahatana CA, Wei L (2005) Time-series bitmaps: a practical visualization tool for working with large time series databases. In: Proceedings of the 2005 SIAM international conference on data mining, SIAM, pp 531–535

Lei Q, Yi J, Vaculin R, Wu L, Dhillon IS (2019) Similarity preserving representation learning for time series clustering. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, pp 2845–2851

Li X, Lin J (2017) Linear time complexity time series classification with bag-of-pattern-features. In: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, pp 277–286

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing sax: a novel symbolic representation of time series. Data Min Knowl Discov 15(2):107–144

MacQueen J (1967) Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA 1:281–297

Madiraju NS, Sadat SM, Fisher D, Karimabadi H (2018) Deep temporal clustering: Fully unsupervised learning of time-domain features

Niennattrakul V, Ratanamahatana CA (2009) Shape averaging under time warping. In: 2009 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, IEEE, vol 2, pp 626–629

Paparrizos J, Gravano L (2015) k-shape: Efficient and accurate clustering of time series. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, pp 1855–1870

Petitjean F, Ketterlin A, Gançarski P (2011) A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognit 44(3):678–693

Ratanamahatana CA, Keogh E (2004) Everything you know about dynamic time warping is wrong. Citeseer, USA

Rebbapragada U, Protopapas P, Brodley CE, Alcock C (2009) Finding anomalous periodic time series. Mach Learn 74(3):281–313

Saito N, Coifman RR (1994) Local feature extraction and its applications using a library of bases. PhD thesis, Yale University

Steinbach M, Tan PN, Kumar V, Klooster S, Potter C (2003) Discovery of climate indices using clustering. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 446–455

Subhani N, Rueda L, Ngom A, Burden CJ (2010) Multiple gene expression profile alignment for microarray time-series data clustering. Bioinformatics 26(18):2281–2288

Wang X, Mueen A, Ding H, Trajcevski G, Scheuermann P, Keogh E (2013) Experimental comparison of representation methods and distance measures for time series data. Data Min Knowl Discov 26(2):275–309

Yang J, Leskovec J (2011) Patterns of temporal variation in online media. In: Proceedings of the fourth ACM international conference on Web search and data mining, pp 177–186

Zakaria J, Mueen A, Keogh E (2012) Clustering time series using unsupervised-shapelets. In: 2012 IEEE 12th International Conference on Data Mining, IEEE, pp 785–794

Zhang Q, Wu J, Yang H, Tian Y, Zhang C (2016) Unsupervised feature learning from time series. In: IJCAI, pp 2322–2328