Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Three-dimensional piecewise cloud representation for time series data mining

Gangquan Si*, Kai Zheng, Zhou Zhou, Chengjie Pan, Xiang Xu, Kai Qu, Yanbin Zhang

State Key Laboratory of Electrical Insulation and Power Equipment, Shaanxi Key Laboratory of Smart Grid, School of Electrical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi Province 710049, China

A R T I C L E I N F O

Article history:
Received 4 May 2018
Revised 21 June 2018
Accepted 3 July 2018
Available online 4 August 2018

Communicated by Dr. Wojciech Froelich

Keywords:
Representation
Three-dimensional cloud
Time series data mining
Dimensionality reduction
Overlap partitioning

A B S T R A C T

Many researchers have taken interests in time series data mining to discover potential knowledge and information as the amount of data from various domains rapidly increases. Representation, as a necessary implementation component of data mining, is critical to reduce the high dimensionality of time series data and generate a corresponding distance measure to process time series data effectively and efficiently. Many high-level representation approaches for mining time series data have been proposed in the past decades, e.g., PAA, SAX, PWCA and 2D-NCR. In this paper, a novel representation method for time series data, which is named Three-Dimensional Piecewise Cloud Representation (TDPCR), is proposed. The new representation contains a flexible partitioning strategy which protects the connection information between consecutive points by overlapping two adjacent segments. Using the improved cloud model theory, the proposed representation achieves the reduction of the data dimensionality and captures distribution and variation features of segments. Furthermore, a new distance measure, which has adaptive weight factors to adjust the proportion of data information, is defined to describe the relationship between two three-dimensional clouds. Accompanied with the comparisons of state-of-the-art representation methods, a sufficient performance evaluation for the proposed representation is carried out in the classification and query by content tasks. The experimental results show that TDPCR is effective and competitive on most of datasets from several domains.

1. Introduction

Time series, as a sequences of discrete-time data, are being continually generated at an alarming speed from various domains, e.g., finance, economics, meteorology, hydrology, engineering technology, biology, medicine, etc. Many researchers are quite dedicated to mining such data to discover potential knowledge and information, and carry out a large amount of mining tasks covering a wide range of problems from real life. Most of the time series mining tasks can be categorized into seven main areas [1]: query by content [2–4], classification [5–7], clustering [8,9], anomaly detection [10], motif discovery [11], prediction [12–15], segmentation [16,17].

However, it is not easy to carry out the mining work. Time series data mining usually suffers high computational overhead, which results from the high dimensionality of time series data. As a result, it is inexpedient and impolitic to apply mining techniques to raw time series directly. What is worse, with the rapid growth of digital source, the analysis of massive datasets will become more universal and unavoidable.

One of the most promising solutions is to develop representation techniques to achieve significant reduction of dimensionality straightly. In this way, massive datasets can be transformed to a manageable size easily. But a larger compression ratio always results in a more loss of information. Thus, any representation is also required to possess the properties of emphasis the essential characteristics of time series on both local and global scales, insensitivity to random noise and low computational cost to guarantee the efficient and effective conduct of time series mining tasks [18]. So far, many high level representation methods have been proposed [19], including Discrete Cosine Transform (DCT) [21], Discrete Fourier Transform (DFT) [23], Discrete Wavelet Transform (DWT) [25], Piecewise Aggregate Approximation (PAA) [27], Indexable Piecewise Linear Approximation (IPLA) [30], PieceWise Cloud Approximation (PWCA) [31], Two-Dimensional Normal Cloud Representation (2D-NCR) [32], Adaptive Piecewise Constant Approximation (APCA) [33], Piecewise Vector Quantized Approximation (PVQA) [34], Multiresolution Vector Quantized approximation (MVQ) [35], Symbolic Aggregate approXimation (SAX) [37],

* Corresponding author.
E-mail address: sigangquan@mail.xjtu.edu.cn (G. Si).

https://doi.org/10.1016/j.neucom.2018.07.053
0925-2312/© 2018 Elsevier B.V. All rights reserved.

Indexable Symbolic Aggregate approXimation (iSAX) [38], Singular Value Decomposition (SVD) [41], Piecewise Linear Approximation (PLA) [43], Derivative time series Segment Approximation (DSA) [44], Symbolic Essential Attributes Approximation (SEAA) [45], etc. All of these methods have the ability to approximate the time series data in good quality and a majority has been applied to facilitate the data mining tasks of time series [20].

Although most of representations achieve the reduction of the high dimensionality of time series data, there is still a common weakness exposed. The performance is unstable on different occasions. On one hand, these representations often get excellent performance on a part of datasets but work badly on the others, even on the same type of datasets. On the other hand, their performance usually degrades rapidly in much lower reduced space. There seem to be three reasons for this phenomenon: (1) Most of existing representation methods don't acquire sufficient and essential information of raw time series for solving various mining tasks, especially in lower dimensionality. For example, most approaches only consider the distribution of time series, but ignore the variation information. (2) The common partitioning strategy which divides a time series into some equal-length "frame" may cut off some effective information chains resulting in the loss of essential characteristics. (3) Many representations can't handle the implicit noise, uncertainty and vagueness of time series data produced during acquisition process.

Therefore, our primary motivation is to propose a new representation method which can overcome the shortcomings mentioned above and manage time series data from various datasets effectively and efficiently. Firstly, and most importantly, the new method should preserve enough fundamental characteristics of time series for mining tasks. In other words, the new technique should excavate both distribution and variation information to preserve the fundamental characteristics fully. Secondly, the new method should adjust the proportion of information for different datasets in different mining tasks adaptively. For instance, the distribution information from a dataset may be highly effective for a specific task and the sensible way is to increase the proportion of distribution information to obtain better results. Thirdly, a new partitioning strategy should be proposed to protect the important connection of points from damage. Finally, the transformation needs to be insensitive to noise and capable of addressing uncertainty and vagueness of various datasets.

In our work, the proposed representation referred to as Three-Dimensional Piecewise Cloud Representation (TDPCR) is based on the cloud model [21]. Firstly, we divide a time series into a series of segments by a flexible partitioning strategy which allows two adjacent segments to overlap each other. Secondly, we calculate the first-order and second-order differences of raw time series and combine each segment with its first-order and second-order differences as an independent segment group. Thirdly, each segment group is transformed into a three-dimensional cloud. Then, a three-dimensional cloud sequence for the time series data is generated. Finally, an adaptive weight distance measure is proposed to calculate the similarity between two three-dimensional clouds and further applied to discover the relation between two corresponding time series.

The new method proposed has following advantages:

- The new partitioning strategy in TDPCR considers the connection between consecutive points on time series and avoids the loss of effective information at all possible by overlapping two adjacent segments. Meanwhile, it also solves the problem existing in other methods such as Piecewise Aggregate Approximation (PAA), PieceWise Cloud Approximation (PWCA), Two-Dimensional Normal Cloud Representation (2D-NCR) that the length of the time series of a particular dataset is indivisible by many given number of segments.
- TDPCR not only inherits the good properties of the cloud model that describes the linkage between randomness and fuzziness via numerical characteristics, but also captures more fundamental characteristics from the first-order and second-order differences. Thus, TDPCR collects sufficient distribution and variation information of time series and performs better both in much lower and high reduced space.
- The distance measure in TDPCR can adjust the weights on the distribution and variation information of time series adaptively according to different mining tasks in different dataset and improve performance on various occasions.

In order to demonstrate the effectivity of our representation, we conducted an extensive experimental evaluation on 84 time series datasets (including 5 device datasets, 7 ecg datasets, 29 image datasets, 14 motion datasets, 16 sensor datasets, 6 simulated datasets and 7 spectro datasets) within the classic data mining tasks of classification and query by content. This evaluation involved prominent state-of-the-art methods for time series representation. The experimental results show that TDPCR is a more effective and competitive representation method for time series data mining, especially in lower reduced space.

The rest of this paper is organized as follows. Section 2 briefly reviews the state-of-the-art methods for time series representation. Section 3 introduces the TDPCR algorithm and a corresponding similarity measure in detail. In Section 4, extensive experiments are conducted to validate the performance of TDPCR within two classic time series data mining tasks. Section 5 summarizes the conclusions and offers some suggestions for future work.

2. Related work

There are lots of time-series representation methods proposed in the past decades for solving various data mining tasks on high-level abstraction of data. According to the taxonomy of [22], representation methods can be divided into two basic categories, namely non data-adaptive and data-adaptive.

2.1. Non data-adaptive representation

The non data-adaptive methods construct an approximate representation based on a specific pattern for every time series regardless of its nature.

One of the well-known non data-adaptive representations is Discrete Fourier Transform (DFT), which is first used in the seminal work [23]. The basic idea of DFT is to represent a time series by the super position of a finite number of sine/cosine functions. In other words, DFT transforms a raw time series into a sequence of complex numbers (known as Fourier coefficients) based on sine/cosine functions. In this way, DFT is implemented to convert time series from real domain to the frequency domain [24]. However, only a small part of Fourier coefficients have higher amplitude and contain most effective information. Thus, the representation is Fourier coefficients with high amplitude which can reflect the global features of a time series. Another related representation method is Discrete Wavelet Transform (DWT) [25], which similarly transforms a raw time series and represents it by a choice of wavelet coefficients. However, unlike DFT, DWT employs the sum and difference of a prototype function (called a mother function) instead of fixed functions. In addition, the choice of wavelet coefficients reflects a series of local features subsection of a time series.

Discretization is another popular way to represent time series. The simplest method is sampling [26], which collects a sequence of points from the raw time series according to a certain sampling rate. Obviously, the sampling method can't capture any effective information like the trends, shapes and patterns contained within the raw time series. Thus, an enhanced method referred to as Piecewise Aggregate Approximation (PAA) is proposed by Keogh et al. [27]. As one of most typical discretization methods, PAA divides a time series into multiple segments with equal length and represents it through the mean values of data points within corresponding segments. (Because many extensions to the PAA representation belong to the data-adaptive category, we will discuss them in next subsection.) Whatis more, Lee et al. [28] use a feature vector including the sum of variation of each segment (SSV) to represent a time series, Ratanamahatana et al. [29] propose a clipped representation method which transforms each time series data point into a single bit and has the property of allowing lower bounding, and Chen et al. [30] presents an Indexable Piecewise Linear Approximation (IPLA) which respectively approximates each segment with a best fitted line segment after dividing a time series into many disjoint segments of equal length. Another promising discretization method is PieceWise Cloud Approximation (PWCA) proposed by Li et al. [31] where the cloud theory is first considered in the time series data mining community. PWCA partitions a time series into equal-length frames and transform each frame into cloud model with three numerical characteristics. And the sequence of cloud characteristics is used to represent the raw time series. Meanwhile, Li et al. propose a similarity measure to describe the relationship between two cloud sequences based on expectation curves of cloud models. In this way, PWCA not only retains the character of cloud model but also contains more information. An extension to the PWCA representation is piecewise Two-Dimensional Normal Cloud Representation (2D-NCR) proposed by Deng et al. [32]. The most obvious difference is that 2D-NCR both captures the distribution and variation of time series and employs the two-dimensional normal cloud model to describe each frame with six numerical characteristics. Compared with PWCA, 2D-NCR performs better in much lower dimensionality in time series mining tasks.

2.2. Data-adaptive representation

In the data-adaptive approaches, assignment of the parameters within a transformation is depending on the data available. In other words, a tuning process will be defined to modify the parameters of a transformation. In particular, a non data-adaptive method can be changed into a data-adaptive method by adding a data-sensitive tuning process.

As mentioned above, a part of data-adaptive approaches are the extensions to Piecewise Aggregate Approximation (PAA). A straight extended version is adaptive piecewise constant approximation (APCA), introduced by Keogh et al. [33]. Instead of using equal-length segments, this APCA allows segments to have different lengths and adds an extra vector to record their length. In [34], Piecewise Vector Quantized Approximation (PVQA) is proposed for efficient similarity analysis. The partitioning step of PVQA is to obtain equal-length segments, same as that of PAA. Next, it applies vector quantization (VQ) to create a codebook of key-sequence and represents each segment by the closest codeword. For some datasets, PVQA actually performs better than PAA in time series similarity analysis but the improvement is not obvious. Thus, Multiresolution Vector Quantized (MVQ) approximation proposed by Megalooikonomou et al. [35] uses multiple codebooks with different resolution levels to keep more local information and gets a better result. A similar approach is also proposed by Stiefmeier et al. [36] which maps motion vectors into a codebook to realize the classification of gestures. Another well-known extension is Symbolic Aggregate approXimation (SAX) [37] which reduces a time series into a string. The main process after discretization via Piecewise Aggregate Approximation (PAA) contains two phases. Firstly, it defines breakpoints which can divide a Gaussian distribution to equal-sizes areas and produces a small alphabet of symbols with a cardinality which is specified by the user. Next, according to the mean values, it transforms segments into symbols and represents the raw time series as a discrete string. In this way, it preserves more the shape information within the time series. Moreover, it is proved that SAX has two obvious advantages, dimensionality reduction and Lower Bounding, and plays a greater competitiveness on clustering, classification and indexing. In [38], Shieh et al. modify SAX to be a multi-resolution representation, namely Indexable Symbolic Aggregate approXimation (iSAX) which is able to transform a time series to a string using different cardinalities and achieves the comparison of strings with different cardinalities. And the multi-resolution property makes fast indexing possible with zero overlap at leaf nodes. However, both SAX and iSAX only consider the average values of segments, which results the loss of other effective information. The Extended SAX (ESAX) [39] adds the maximum and minimum of each segment and uses the three values to fully represent a time series. This method can handle some critical points of a time series, especially in financial data. Besides, Sun et al. [40] takes into account the variation information between segments and puts forward a trend distance for SAX using the starting and the ending points and the average of each segment, namely SAX-TD.

Other inherently data-adaptive representations have also been proposed. Singular Value Decomposition (SVD) proposed by Korn et al. [41] has been used for query by content at first. Later, Kanth et al. [42] have applied it to indexing images and other multimedia objects successfully. The principle of SVD is to use a linear combination of basis shapes to represent the shape of a time series, which minimizes the reconstruction error. However, there are two serious drawbacks in SVD. One is the higher space and time complexity. The other is the inflexibility of the transformation: SVD needs to consider all time series at a time when the content of a dataset changes. Piecewise Linear Approximation (PLA) [43] represents a time series with straight lines. In other words, each segment is approximated by linear function. Its properties determine a wide applications in the time series segmentation task. There are three major derivatives of PLA: Sliding Windows, Top-Down and Bottom-UP. In [44], Gullo et al. propose a concise yet feature-rich time series representation, namely Derivative time series Segment Approximation (DSA) representation model. It features derivative estimation, segmentation and segment approximation to transform a time series into a shorter sequence of values, which results in a sensitivity in trend information and data compression. By using Dynamic Time Warping as the similarity measure, DSA performs well in time series similarity detection. In [45], Krawczak et al. form a nominal representation, namely Symbolic Essential Attributes Approximation (SEAA). Firstly, it generates a series of upper and lower envelopes to achieve the dimensionality reduction. Secondly, a vector of real-valued attributes are generated by a multilayer neural network. Finally, the process of discretization transforms all real-valued attributes into nominal-valued attributes. By representing a time series by a vector of nominal-valued attributes, SEAA preserves the main characteristics of time series. To support fast and accurate similarity detection, Zhang et al. [46] present a new time series representation model and a corresponding similarity measure, namely fragment alignment distance (FAD). As a concise yet feature-rich representation, FAD accelerates the similarity detection process through aligning the segments of time series in linear time.

All non data-adaptive and data-adaptive approaches mentioned above can represent a raw time series in a concise way and achieve the reduction of the high dimensionality of time series data. However, in different reduced space, there methods can't maintain a

Algorithm 1 Overlap partitioning strategy (OPS).

```
Input: A time-series $T = (t_1, t_2, \ldots, t_N)$
       The number of segments $W$ and the number of overlap points $M$
Output: Segments $\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_W$; Overlapping rate $P$
Step 1: Calculate the expected length of all segments and overlapping rate $P$, i.e.,
       $L = \sum_{i=1}^{W} length(\boldsymbol{S}_i) = N + (W-1) \times M$
       $P = (W-1) \times M / L \times 100\%$
Step 2: Calculate the quotient $Q$ and remainder $R$ when $L$ is divided by $W$, i.e.,
       $Q = [L/W] \quad R = L - Q \times W$
Step 3: Generate segments, i.e.,
       For $i \leftarrow 1$ to $R$ do
           $S_i = (t_{i\times(Q+1-M)+1}, t_{i\times(Q+1-M)+2}, \ldots, t_{i\times(Q+1-M)+1+Q})$
       For $j \leftarrow 1$ to $W$-$R$ do
           $S_{W-j+1} = (t_{N-(j-1)\times(Q-M)-Q+1}, t_{N-(j-1)\times(Q-M)-Q+2}, \ldots, t_{N-(j-1)\times(Q-M)})$
```

desirable performance. Because high compress ratio which always results in the loss of information, often causes a negative impact on subsequent mining work. Therefore, we propose a new representation, namely Three-Dimensional Piecewise Cloud Representation (TDPCR), which can preserve sufficient characteristics of various datasets in lower reduced space to complete the major time series data mining tasks effectively and efficiently.

3. Three-dimensional piecewise cloud representation (TDPCR)

In this section, a flexible method of segmentation, namely overlap partitioning strategy (OPS), is first introduced. Then, the theory of a three-dimensional cloud model is produced. Finally, the three-dimensional piecewise cloud representation (TDPCR) including its similarity measure is presented.

3.1. Overlap partitioning strategy (OPS)

Let us denote a time-series as $T = (t_1, t_2, \ldots, t_N)$ and the set of time series acquired from the same object constitutes a dataset. Generally, there are various categories and each sequence is N units long. At first, it is required to provide a convenient approach to divide any time-series into$W$segments keeping the critical information as far as possible. The obvious and simple way is to cut out multiple segments with the same length N/W when the length of the time-series is divisible by the number of segments. For example, a time series with ten data points can be divided into five equal-length segments in Fig. 1(a). However, there are two major problems: (1) Information loss: if point 2, 3 are closely linked and contain key information, this approach will cut off their connection causing the loss of information. Same problem holds for any breakpoint. (2) Indivisible problem: the choice of W is limited. We fail to divide the same time series into three segments in this way because 10 cannot be divided by 3 with no reminder. Especially, when the length of the time-series is a prime number, the partitioning approach no longer works. However, the partitioning approach introduced in SAX has tried to solve the undividable problems. It puts a part of a whole point into two adjacent segments with different weights. For example, in Fig. 1(c), point 4, 7 contribute their 2/3 to segment 2 and the rest 1/3 parts are given to segment 1, 3, respectively. In this way, segment 1, 2, 3 all possess $3\frac{1}{3}$ points. This approach can guarantee the divisibility of a time series and each segment contains the same number of data points. Yet, it never considers how to preserve critical information as the first problem mentioned above. What's worse, the disintegration of

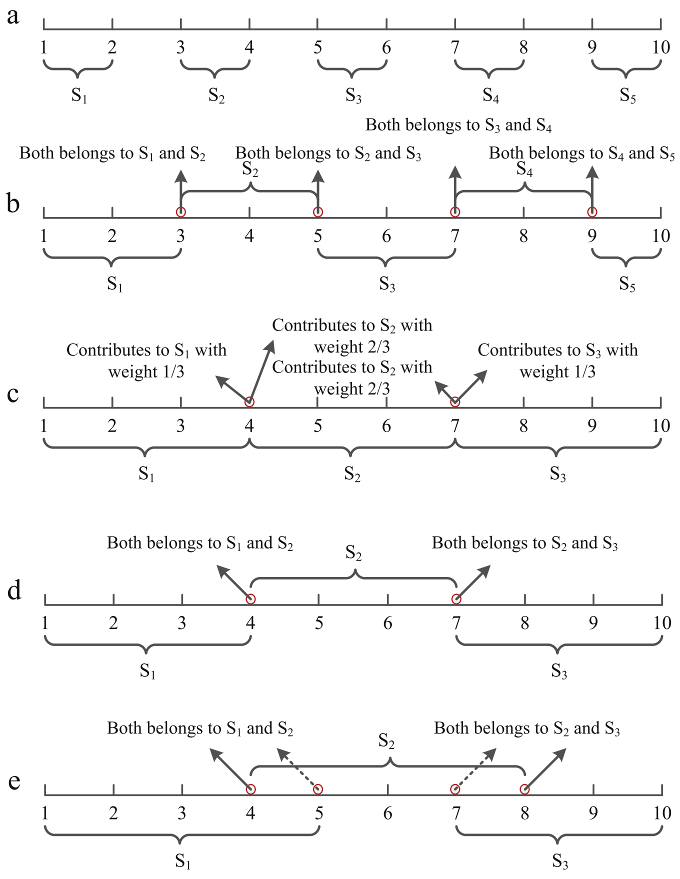

a
b
c
d
e
1 2 3 4 5 6 7 8 9 10
S1 S2 S3 S4 S5
Both belongs to S3 and S4
Both belongs to S1 and S2
Both belongs to S2 and S3
Both belongs to S4 and S5
Contributes to S1 with weight 1/3
Contributes to S2 with weight 2/3
Contributes to S2 with weight 2/3
Contributes to S3 with weight 1/3

Fig. 1. (a) Ten data points are divided into five equal-length segments, namely S1, S2, S3, S4 and S5. (b) Ten data points are divided into five segments by OPS with one overlap point. Every common point (point 3, 5, 7, 9) both belongs to two adjacent segments. (c) Ten data points are divided into three segments by SAX. Point 4, 7 contribute their 2/3 to S2 and the rest 1/3 parts are given to S1 and S3 respectively. (d) Ten data points are divided into three equal-length segments by OPS with one overlap point. Point 4 belongs to S1 and S2, and point 7 belongs to S2 and S3. (e) Ten data points are divided into three segments by OPS with two overlap points. Point 4, 5 belong to S1 and S2, and point 7, 8 belong to S2 and S3.

several points may cause serious information distortion and limit its application area.

Based on the discussion above, splitting a complete point into two adjacent segments with different weights may be not an advisable way to finish the segmentation work. Although it solves the indivisible problem successfully, the loss of information still remains. Thus, we intend to ensure the integrity of each point and deal with the two problems in another way. It is found that the overlap of two segments can keep the connection between consecutive points and protect more effective information from damage. On this basis, we propose a flexible overlap partitioning strategy (OPS) described as Algorithm 1. We allow two adjacent segments to overlap each other and have M common points. The value of M is required as an input variable. In step 1 and 2, the expected length L of all segments and overlapping rate P are calculated. In Step3, the employ segments generated from a time series may have two kinds of lengths, i.e., the length of $\boldsymbol{S}_1 \sim \boldsymbol{S}_R$ is Q+1, and that of $\boldsymbol{S}_{R+1} \sim \boldsymbol{S}_W$ is Q, which purpose is to solve the indivisible problem. For example, in Fig. 1(b), we divide a length 10 time-series into five segments by OPS and adjacent segments have 1 common point. The length of $\boldsymbol{S}_1 \sim \boldsymbol{S}_4$ is 3, and $\boldsymbol{S}_5$ contains point 9, 10. Every common point (point 3, 5, 7, 9) both belongs to two different segments. Compared with Fig. 1(a), OPS doesn't cut off the connection between point 2 and 3, point 4 and 5, point 6 and 7, point 8 and 9, which preserves complete information between neighbor points. When increasing the value of W, the common points and the length of segments change. In Fig. 1(d), the same time-series is divided into three segments and M is still equal to 1. Every segments contain four points and point 4, 7 become the common point. Although 10 points can't be divided evenly into 3 segments, it's possible to partition them into three same length segments in an overlapping manner using OPS. With this method, we can recapture a part of lost information by increasing the value of M. If point 3, 4, 5 make up a inseparable information chain, we set M equal to 2 in Fig. 1(e) where $\boldsymbol{S}_1$ contains that vital information chain.

Selection of the values of parameters W and M makes obvious influence on time series representation. If the value of W is too small, few segments may lose lots of useful information involved in time series. On the contrary, too large value of W can't reflect the significance of dimensionality reduction [47]. Similarly, selecting a proper value of M can not only help in recapturing vital information but also avoid that redundant information gets into segments. Therefore, we select the values of W and M in an experimental way and every dataset has its unique parameters. The results of experiments will demonstrate the usefulness of the overlap partitioning strategy and influence of the parameters W and M, which are discussed in details in Section 4.

3.2. Three-dimensional cloud model

The cloud model is first proposed by Li to explore the essential uncertainty. It realizes the conversion between qualitative conception and quantitative description based on the probability theory and fuzzy mathematics theory, and the mapping between qualitative conception and quantitative data is realized by forward and backward generators. In addition, it is implemented to describe the linkage between randomness and fuzziness. So far, the cloud model proves to be an effective tool for the application in lots of fields, such as data mining [48], image segmentation [49] and so on. However, although the cloud model has so many advantages, it can't represent an object in multiple directions which may result in the loss of information. For example, the cloud model can only capture a part of information of a time series like distribution, but lose other potential and important characteristics like variation. Therefore, it is reasonable to extend it to the three-dimensional cloud model to make a more comprehensive description of an object while retaining those advantages. Based on the discussion above, we propose the three-dimensional cloud model as follows:

Definition 1. Let $U\{X,\ Y,\ Z\}$ be a three-dimensional quantitative universal and $C(Ex,\ Ey,\ Ez,\ Enx,\ Eny,\ Enz,\ Hex,\ Hey,\ Hez)$ be a qualitative conception related to U. If $(x,\ y,\ z)$, $(x\in X,\ y\in Y,\ z\in Z)$ is a random realization of the conception C and $(x,\ y,\ z)$ obeys three-dimensional normal distribution

$$(x,y,z)\sim N((Ex,Ey,Ez),\,(Enx'^2,Eny'^2,Enz'^2)) \tag{1}$$

where $(Enx',\ Eny',\ Enz')$ is a random realization of three-dimensional normal distribution

$$(Enx',Eny',Enz')\sim N((Enx,Eny,Enz),\,(Hex^2,Hey^2,Hez^2)) \tag{2}$$

and the certainty degree of $(x,\ y,\ z)$ on U is

$$\mu(x,y,z)=e^{-\frac{(x-Ex)^2}{2Enx'^2}-\frac{(y-Ey)^2}{2Eny'^2}-\frac{(z-Ez)^2}{2Enz'^2}} \tag{3}$$

then the distribution of $(x,\ y,\ z)$ on U is a three-dimensional normal cloud, and each $(x,\ y,\ z)$ is defined as a three-dimensional cloud drop.

From the definition, it can be found that the three-dimensional cloud model inherits almost all features of the cloud model such as randomness, fuzziness. It can transform the qualitative conception into quantitative data by "three-dimensional forward normal cloud generator (TFCG)". In Algorithm 2, Step 2 generates a three-dimensional normal-distributed random vector and Step 3 produces a three-dimensional cloud drop. Similarly, it's available to acquire the qualitative conception from three-dimensional cloud drops by "three-dimensional backward normal cloud generator (TBCG)". In Algorithm 3, Step 1, 2, 3 respectively calculate Expectation $(Ex\sim Ez)$, Entropy $(Enx\sim Enz)$ and Hyper-entropy $(Hex\sim Hez)$ which are the elements of a three-dimensional cloud model. Moreover, the three-dimensional cloud model is implemented to analyze an ensemble of three different variables, even if the three variables are independent. For example, the three-dimensional cloud model is able to represent spatial coordinate changers of objects.

Algorithm 2 Three-dimensional forward normal cloud generator (TFCG).

```
Input: Numerical characteristics $(Ex, Ey, Ez, Enx, Eny, Enz, Hex, Hey, Hez)$
       The number of three-dimensional cloud drops $n$
Output: Three-dimensional cloud drops $(x_i, y_i, z_i)$ with
        certainty degree $\mu(x_i, y_i, z_i)(i = 1, 2, , n)$
Step 1: Set the value of $i$ equal to 1.
Step 2: Generate a three-dimensional normal-distributed random
        vector $(Enx'_i, Eny'_i, Enz'_i)$ with expectation $MU_1$, and
        variance $SIGMA_1$ where
        $MU_1 = (Enx, Eny, Enz); SIGMA_1 = (Hex^2, Hex^2, Hez^2)$
        $(Enx'_i, Eny'_i, Enz'_i) = mvnrnd(MU_1, SIGMA_1, 1)$
Step 3: Generate a three-dimensional cloud drop $(x_i, y_i, z_i)$ with
        expectation $MU_2$, and variance $SIGMA_2$ where
        $MU_2 = (Ex, Ey, Ez); SIGMA_2 = (Enx'^2_i, Eny'^2_i, Enz'^2_i)$
        $(x_i, y_i, z_i) = mvnrnd(MU_2, SIGMA_1, 1)$
Step 4: Generate the certainty degree
        $\mu(x_i, y_i, z_i) = e^{-\frac{(x-Ex)^2}{2Enx'^2}-\frac{(y-Ey)^2}{2Eny'^2}-\frac{(z-Ez)^2}{2Enz'^2}}$
Step 5: Judge whether $i$ is equal to $n$, and if $i \neq n, i = i + 1$ and go
        back to Step 2. When $i = n$ is reached, return $n$ three-
        dimensional cloud drops with their certainty degree.
```

Algorithm 3 Three-dimensional backward normal cloud generator (TBCG).

```
Input: Vectors $X, Y, Z$
Output: A three-dimensional cloud model with numerical
        characteristics $(Ex, Ey, Ez, Enx, Eny, Enz, Hex, Hey, Hez)$
Step 1: Calculate the Expectation of each vector, i.e.,
        $Ex = mean(X); Ey = mean(Y); Ez = mean(Z)$
Step 2: Calculate Entropy $Enx, Eny$ and $Enz$
        $Enx = \sqrt{\frac{\pi}{2}}mean(|X - Ex|); Eny = \sqrt{\frac{\pi}{2}}mean(|Y - Ey|);$
        $Enz = \sqrt{\frac{\pi}{2}}mean(|Z - Ez|)$
Step 3: Calculate the variance of each vector and Hyper-entropy
        $Hex, Hey$ and $Hez$ $Hex = \sqrt{var(X) - Enx^2};$
        $Hey = \sqrt{var(Y) - Eny^2}; Hez = \sqrt{var(Z) - Enz^2}$
```

3.3. Three-dimensional piecewise cloud representation

Having introduced the overlap partitioning strategy and three-dimensional cloud model, we present the three-dimensional piecewise cloud representation to represent time series with reduced dimensionality.

Given a time series $T = (t_1,\ t_2,\ldots,t_N)$, the reduced dimension $W(1\le W\le N)$ and the number of overlap points M, the three-dimensional piecewise cloud representation can be described in the following three steps, as shown in Fig. 2.

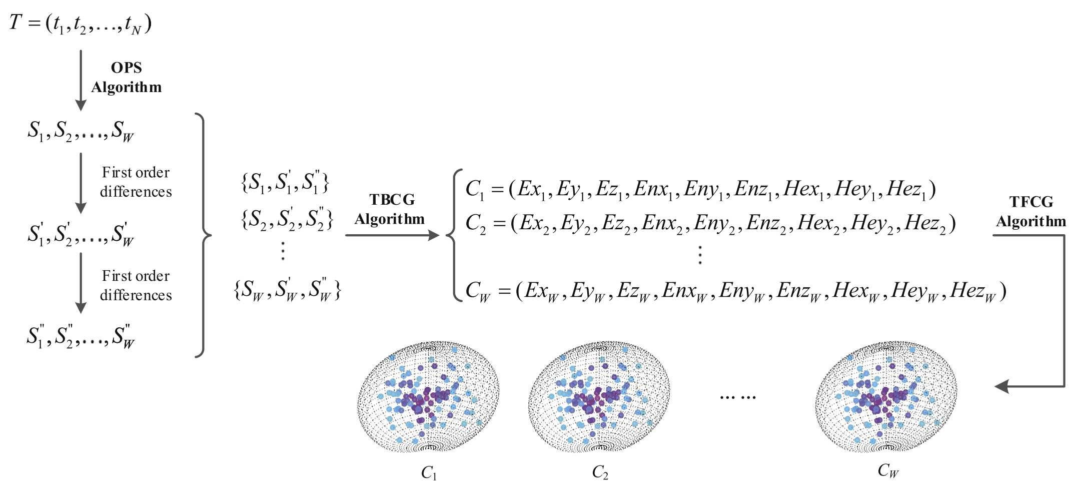


Fig. 2. The framework of TDPCR which contains the overlap partitioning strategy, the process of differential, three-dimensional cloud transformation and visualization.

Step 1: Divide the time series into a series of segments $S_1, S_2, \ldots, S_W$ by OPS Algorithm. The length of each segment may not be exactly the same. For convenience, we will denote the i^{th} segment as $(S_i = t_{i1}, t_{i2}, \ldots, t_{iL_S})(i = 1, 2, \ldots, W)$ and its length as

$$L_S = \begin{cases} lcQ, & 1\le i\le R \\ Q+1, & R<i\le W \end{cases} \tag{4}$$

Step 2: Calculate the first-order and second-order differences of S_i, then denote as S_i' and S_i'' respectively where

$$\begin{aligned} &S_i' = \triangle S_i = (t_{i1}', t_{i2}', \ldots, t_{i(L_S-1)}') \\ &t_{ij}' = t_{i(j+1)} - t_{ij}, \ \ (j = 1, 2, \ldots, L_S - 1) \\ &S_i'' = \triangle S_i' = \triangle(\triangle S_i) = (t_{i1}'', t_{i2}'', \ldots, t_{i(L_S-2)}'') \\ &t_{ik}'' = t_{i(k+1)}' - t_{ik}' = t_{i(k+2)} - 2t_{i(k+1)} + t_{ik}, \ \ (k = 1, 2, \ldots, L_S - 2) \end{aligned} \tag{5}$$

Step 3: Combine each segment with its first-order and second-order differences as an independent segment group $\{S_i, S_i', S_i''\}$. Then perform the TBCG Algorithm to transform each segment group into a three-dimensional cloud C_i. All characteristics of clouds $C_1 \sim C_W$ represent the original time series. Beside, an optional step is to perform the TFCG Algorithm to show three-dimensional clouds in a graphical view.

Each point in a time series reflects the state at its particular moment, and the distribution and variation of points estimates the whole characteristic over a period of time. Many representation techniques only consider the distribution of time series as features to reduce dimensionality, for example: Piecewise Aggregate Approximation (PAA), Symbolic Aggregate approXimation (SAX), PieceWise Cloud Approximation (PWCA), etc. These representations have common advantages like fast, flexible and easy to implement. However, the distribution of time series provides only limited information which may cause the lose effectiveness of these representations for some kinds of datasets. For instance, in Fig. 3, two different segments Q_1 and Q_2, of which the elements are generated by two functions respectively, i.e., $y_1 = \sin(t) - 2/\pi$ and $y_2 = 2/\pi - \sin(t)$, where $t \in [0, \pi]$, have the same mean and variance. PAA (or SAX) fails to distinguish between Q_1 and Q_2, shown in Fig. 3. Two one-dimensional normal clouds produced by PWCA according to Q_1 and Q_2 are $C_{Q_1} = (0, 0.3359, 0)$ and $C_{Q_2} = (0, 0.3359, 0)$ respectively. Obviously, PWCA is also unable to recognize each of them. In order to overcome this problem, Two-Dimensional Normal Cloud Representation (2D-NCR) has tried to employ both distribution and variation of points, and actually performs well in some datasets. Yet, the first-order differences of time

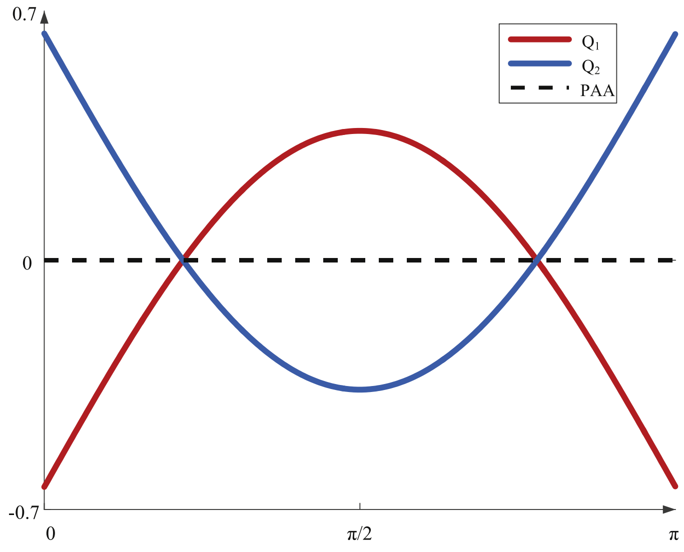


Fig. 3. Two different segments Q_1 and Q_2 have the same mean and variance.

series used in 2D-NCR provides only limited variation information and 2D-NCR does not consider the trade-offs between distribution and variation information according to different datasets. For the above issues, TDPCR proposed in the paper captures more variation information by first-order and second-order differential processing and remains the distribution information of time series simultaneously. Overlap partitioning strategy, the segmentation method of TDPCR, can also provide more points information which has been discussed in Section 3.1. Moreover, TDPCR introduces weight factors in distance measure to adjust the proportion of different kinds of information to meet the need of various tasks, which will be described in following subsection. On the other hand, TDPCR is insensitive to white noise. Although the first-order and second-order differences of segments may have smaller values of means, the key parameters $Ex \sim Ez$ are not affected by noise. In addition, the effect of noise on the Entropy ($Enx \sim Enz$) is limited and we can reduce this influence during distance measure which will be discussed in the next subsection. Therefore, TDPCR will perform better than other dimension reduction methods mentioned above under the same dimension, and have good reconstruction quality.

3.4. Distance measure

After dimensionality reduction by TDPCR, a distance measure should be provided to describe the relationship between two time

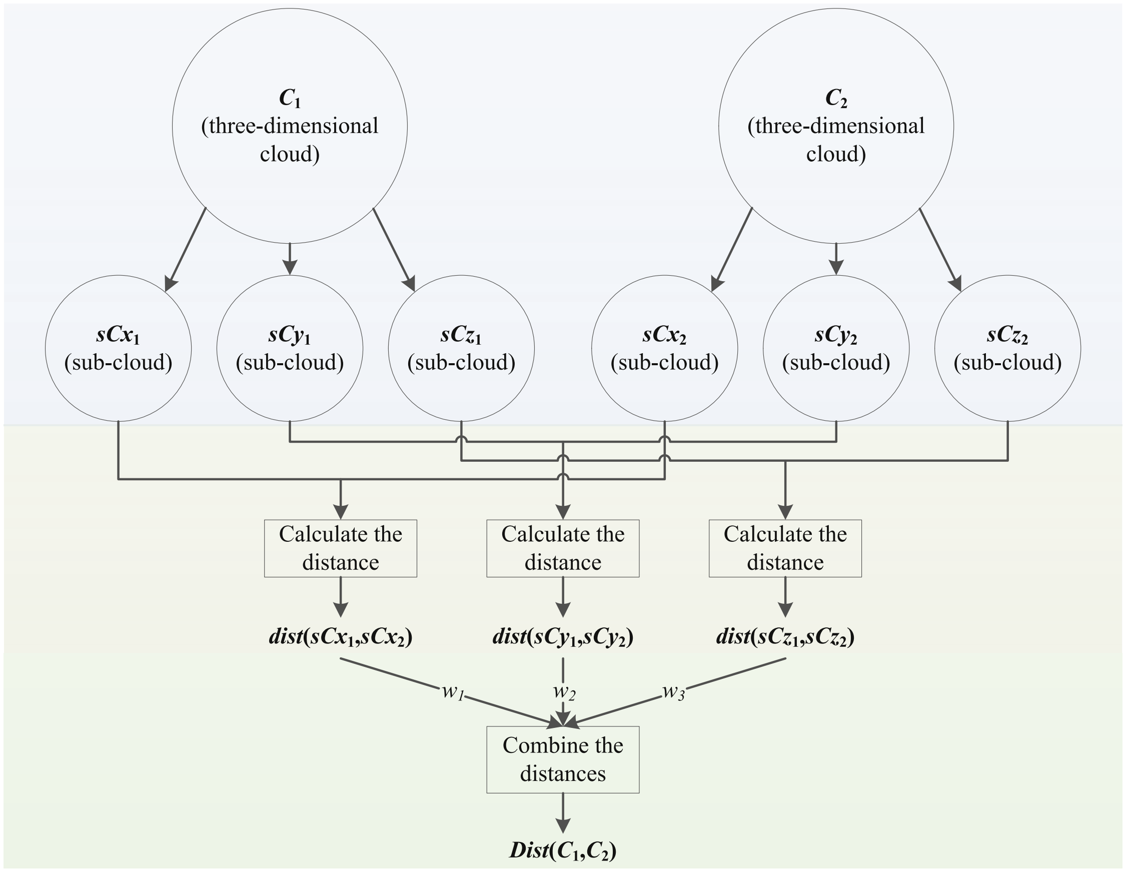


Fig. 4. The framework of the distance measure of two three-dimensional clouds.

series. So far, many classical distance measures such as Euclidean distance [50], Dynamic Time Warping [51] and Optimal Subsequence Bijection [52] are not directly applicable to TDPCR because of the particularity of their numerical characteristics. Thus, a new distance measure is required for it. For an illustration, given two time series T_1, T_2 and the reduced dimension W, TDPCR transforms these time series into the sets C_{T_1} and C_{T_2} respectively where

$$C_{T_1} = \{C_{11}, C_{12}, \ldots, C_{1W}\} \quad C_{T_2} = \{C_{21}, C_{22}, \ldots, C_{2W}\} \tag{6}$$

$$C_{ij} = (Ex_{ij}, Ey_{ij}, Ez_{ij}, Enx_{ij}, Eny_{ij}, Enz_{ij}, Hex_{ij}, Hey_{ij}, Hez_{ij}) \quad (i = 1, 2;\ j = 1, 2, \ldots, W) \tag{7}$$

Each transformed time series contains W different three-dimensional clouds, and each cloud represents a different group of the time series. In other words, clouds $C_{11} \sim C_{1W}$ and $C_{21} \sim C_{2W}$ are subsets of sets C_{T_1} and C_{T_2} respectively. The distance of two time series is equal to the sum of distances of the corresponding three-dimensional clouds. Thus, we denote the distance of the time series T_1, T_2 as $DIST(T_1, T_2)$ and the distance of two clouds as $Dist(C_{1i}, C_{2i})$ where

$$DIST(T_1, T_2) = \frac{1}{W} \sum_{i=1}^{W} Dist(C_{1i}, C_{2i}) \tag{8}$$

Next, we concentrate on calculating the distance of two three-dimensional clouds. Every three-dimensional cloud can be divided into three different sub-clouds as follows:

$$\begin{cases} sCx_{ij} = (Ex_{ij}, Enx_{ij}, Hex_{ij}) \\ sCy_{ij} = (Ey_{ij}, Eny_{ij}, Hey_{ij}) \\ sCz_{ij} = (Ez_{ij}, Enz_{ij}, Hez_{ij}) \end{cases} \tag{9}$$

Three sub-clouds, which respectively represent the raw segment, the segment first-order and second-order differences, make different contribution on measurement of two three-dimensional clouds. Therefore, it's reasonable to calculate all distances of the corresponding sub-clouds and combine them with different weights as the distance of two three-dimensional clouds. The framework of the distance of two three-dimensional clouds is clearly shown in Fig. 4 and the detailed step descriptions are provided as follows:

Step1: Calculate the distance of each pair of the corresponding sub-clouds. Because each pair of corresponding sub-clouds has similar calculation procedure, we just discuss the distance of sCx_{1j} and sCx_{2j}, denoted as $dist(sCx_{1j}, sCx_{2j})$.

$$dist(sCx_{1j}, sCx_{2j}) = \frac{(Umin_{xj} - Lmax_{xj}) \cdot max(\mu(x_{1j}), \mu(x_{2j}))}{Umax_{xj} - Lmin_{xj}} \tag{10}$$

where $Umin$, $Umax$ are respectively the minimum and maximum values of the upper bounds of sub-clouds satisfying the "3En rule". $Lmin$, $Lmax$ represent the minimum and maximum values of the lower bounds of sub-clouds respectively. x_{1j}, x_{2j} are the intersect points of two expectation curves of sCx_{1j} and sCx_{2j}, while $\mu(x_{ij})$ is the certainty degree. The mathematical formulas of above variables are listed as follows:

$$Umin_{xj} = min(Ex_{1j} + 3Enx_{1j}, Ex_{2j} + 3Enx_{2j})$$
$$Umax_{xj} = max(Ex_{1j} + 3Enx_{1j}, Ex_{2j} + 3Enx_{2j})$$

$$Lmin_{xj} = min(Ex_{1j} - 3Enx_{1j}, Ex_{2j} - 3Enx_{2j})$$
$$Lmax_{xj} = max(Ex_{1j} - 3Enx_{1j}, Ex_{2j} - 3Enx_{2j}) \quad (11)$$

$$x_{1j} = \frac{Ex_{1j} \cdot Enx_{2j} - Ex_{2j} \cdot Enx_{1j}}{Enx_{2j} - Enx_{1j}}$$
$$x_{2j} = \frac{Ex_{1j} \cdot Enx_{2j} + Ex_{2j} \cdot Enx_{1j}}{Enx_{2j} + Enx_{1j}} \quad (12)$$

$$\mu(x_{ij}) = \begin{cases} exp\{-\frac{(x_{ij} - Ex_{ij})^2}{2Enx_{ij}^2}\}, x_{ij} \in [Ex_{ij} - 3Enx_{ij}, Ex_{ij} + 3Enx_{ij}] \\ 0 \ , x_{ij} < (Ex_{ij} - 3Enx_{ij}) \ or \ x_{ij} > (Ex_{ij} + 3Enx_{ij}) \end{cases} \quad (13)$$

Step2: Combine the distances of corresponding sub-clouds with specific weights.

$$Dist\,(C_{1i}, C_{2i}) = \frac{1 - w_1 \cdot dist\,(sCx_{1j}, sCx_{2j}) - w_2 \cdot dist\,(sCy_{1j}, sCy_{2j}) - w_3 \cdot dist\,(sCz_{1j}, sCz_{2j})}{w_1 \cdot dist\,(sCx_{1j}, sCx_{2j}) + w_2 \cdot dist\,(sCy_{1j}, sCy_{2j}) + w_3 \cdot dist\,(sCz_{1j}, sCz_{2j})}$$

$$w_1 + w_2 + w_3 = 1, \ 0 \le w_1, w_2, w_3 \le 1 \quad (14)$$

$w_1 \sim w_3$ respectively represent the contribution of $sCx \sim sCz$ and the larger value has more effects on the measured results.

The weights make great influence on the performance of TDPCR. For different time series mining tasks, it's important to provide suitable weights to solve one by one. For example, TDPCR selects the weights which have best classification accuracy for the training set to deal with the classification task on this dataset. Meanwhile, the choice of weights can also help to reduce the effects of noise. Noise may have different extents of impacts on three different sub-clouds. Reducing the weight which corresponding sub-cloud is more sensitive to noise can restrain influence of noise and improve performance. However, it's not difficult to search suitable values. Since the range of the parameters $w1 \sim w3$ is limited by their constraint conditions, the search space is a small limitary plane. Many common parameter searching methods like grid search method [53], particle swarm optimization (PSO) [54], genetic algorithm [55] can be used to acquire desirable weights.

3.5. Computational complexity analysis

Having introduced the theory part of TDPCR, the computational complexity which consists of time complexity and space complexity is analyzed in this subsection.

The calculation of time complexity can be divided into two parts. Firstly, we analyze the time complexity of the three-dimensional piecewise cloud representation. In Algorithm 1, the Step1 and Step2 which are the preparation for Step3 consume $2 \cdot O(1) + 2 \cdot O(1) = 4 \cdot O(1)$, and generating segments consumes $O(W)$. Thus, the time complexity of the overlap partitioning strategy is $4 \cdot O(1) + O(W)$. Then, obtaining the first-order and second-order differences of all segments consumes about $O(N) + O((W-1) \times M)$. According to Algorithm 3, the time complexity of transforming each segment group into a three-dimensional cloud is $9 \times O([L/W])$. Therefore, the overall time complexity of the three-dimensional piecewise cloud representation is

$$4 \cdot O(1) + O(W) + O(N) + O((W-1) \times M) + 9W \times O\left(\left[\frac{L}{M}\right]\right) \approx O(N) + O(WM) \quad (15)$$

Secondly, we focus on the time complexity of the new distance measure. For Step1 in distance measure, calculating the distance of two sub-clouds from two three-dimensional clouds consumes $9 \times O(1)$. Consequently, the time complexity of calculating the distance of two three-dimensional clouds is $27 \times O(1) + O(1)$. The overall time complexity of the new distance measure is

$$W \times 27 \times O(1) + O(W) \approx O(W) \quad (16)$$

Table 1
Comparison of state-of-the-art representations and classical distance measures with the time complexity.

Method	TDPCR	SAX	PAA	2D-NCR	ED	DTW
Time complexity	$O(N)+O(WM)$	$O(N)+O(W)$	$O(N)+O(W)$	$O(N)+O(W)$	$O(N)$	$O(N^2)$

Table 1 summarizes the time complexity of some state-of-the-art representations and classical distance measures. From Table 1, TDPCR may take more time than other state-of-the-art representations and Euclidean distance to finish various tasks which depends on the values of W and M. However, in practice, the values of W and M are often far less than the values of N. Thus, the overtime required by TDPCR is short. Compared with Dynamic Time Warping, TDPCR has lower time complexity.

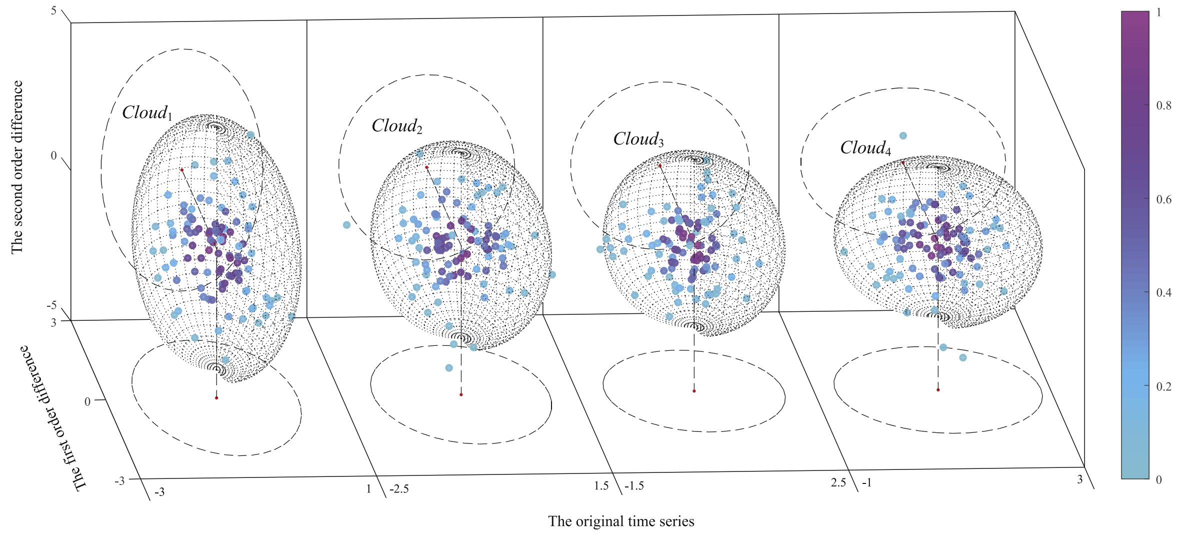
For the space complexity, a three-dimensional cloud which contains 9 numerical characteristics requires 9 storage units. Consequently, representing a raw time series by TDPCR needs $9W$ storage units. In addition, in the process of calculating the distance of two sub-clouds, we use 5 storage units for calculating and 1 unit for recording the distance. Since the 5 calculating units can be reused, the space complexity of calculating the distance of two three-dimensional clouds is 9 units. Therefore, the space used to calculate the distance between two time series (after dimensionality reduction by TDPCR) is $8 + W$ units. Overall, the space complexity of the new distance measure is $O(W)$. Compared with the storage of all the original data, TDPCR saves a lot of storage space.

4. Experimental validation

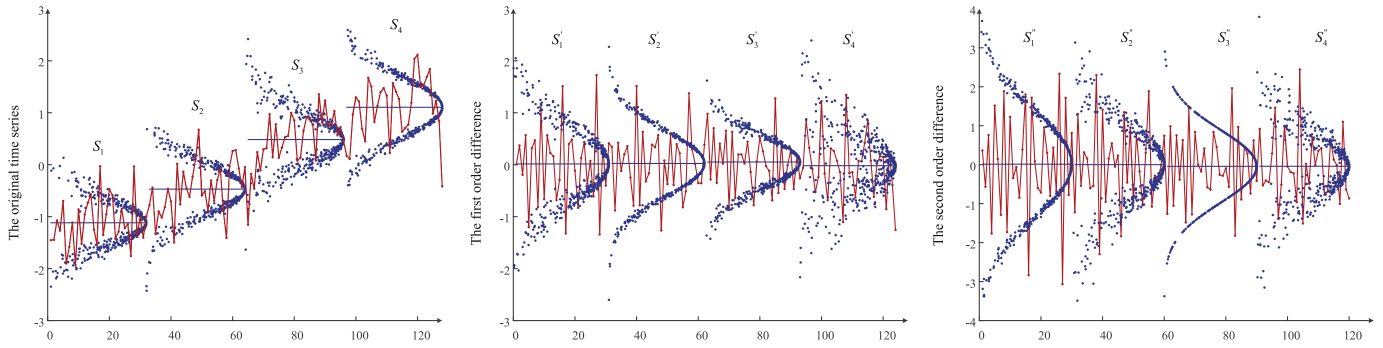
In this section, we use various state-of-the-art representations and TDPCR to perform two common time series mining tasks (classification and Query by content), and compare their results to demonstrate the validation and prominent advantages of TDPCR. The 84 time-series datasets which collected from the UCR Time Series Database, are analyzed in this section and summarized in Appendix A. All datasets are available online at http://www.cs.ucr.edu/~eamonn/. The training/testing split setting is the same as in Chen et al. [56].

4.1. An instance

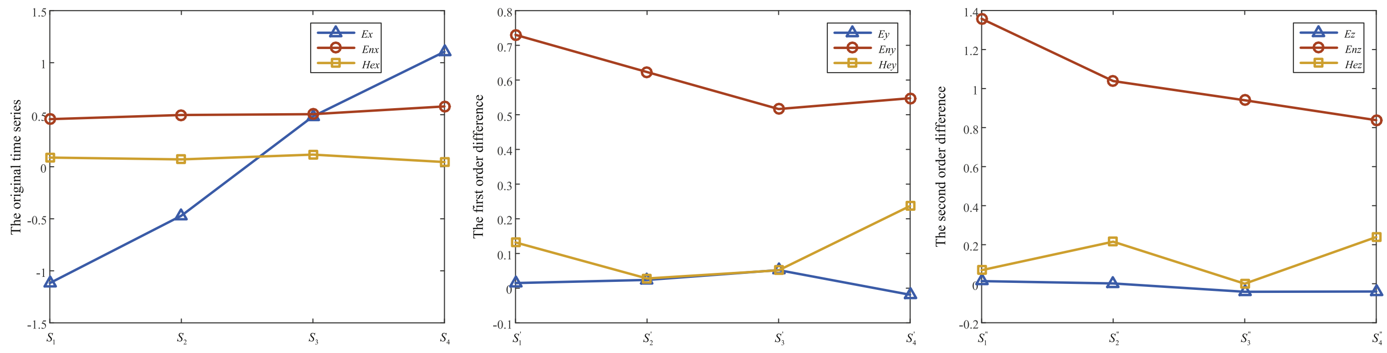
A time series from a simulated dataset (Cylinder-Bell-Funnel dataset) is represented by TDPCR as shown in Fig. 5. The length of the time series is 128, the reduced dimension is 4 and the compress ratio is 32. Fig. 5(a) shows the three-dimensional cloud drops of the segment group $\{S_i, S_i', S_i''\}(i = 1, ..., 4)$. The color of cloud drops represents the certainty degree. The envelope surfaces of $Cloud_1 \sim Cloud_4$ are different ellipsoids with specific shapes. (Ex_i, Ey_i, Ez_i) is the ellipsoid center. The radii of the ellipsoid are $3Enx_i$, $3Eny_i$ and $3Enz_i$ respectively based on the "3En rule" [57]. Fig. 5(b) shows the normal cloud representation of segments, the first-order and second-order differences, and Fig. 5(c) presents the according parameters. The expectations $Ex_1 \sim Ex_4$ are significantly different. $Ey_1 \sim Ey_4$ and $Ez_1 \sim Ez_4$ are nearly equal to zero. The entropy En and hyper-entropy He of the first-order differences are different from those of the second-order differences. Therefore, TDPCR, which captures the distribution and variation information of time series, reduces the dimensionality of time series as well as preserves the fundamental characteristics.



(a)



(b)

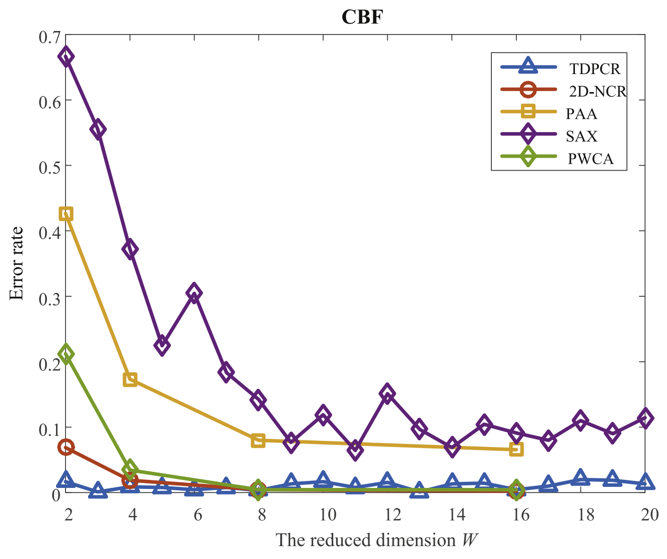


(c)

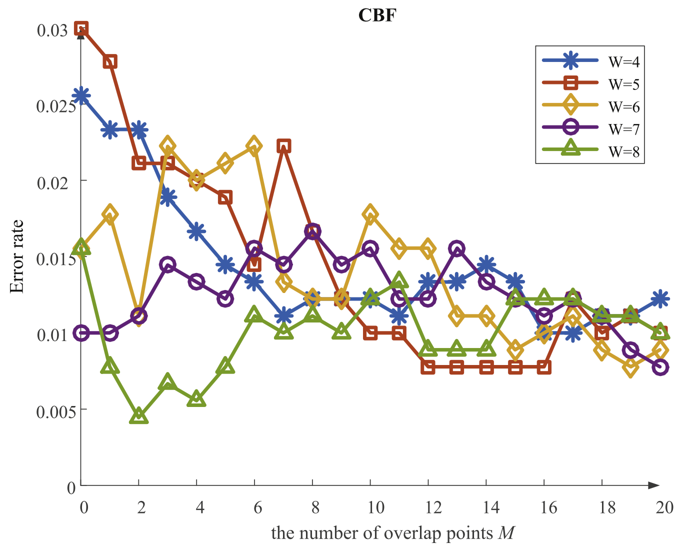
Fig. 5. (a) TDPCR of a time series with 128 data points divided into four segments. (b) The normal cloud representation of the four segments, and their first-order, second-order differences. (c) The parameters of three-dimensional cloud models.

4.2. Classification

Before presenting experimental evaluation on the classification of existing time-series representations, there are two obvious questions that needs to be considered first: (1) What kind of classifier should be used for comparison? There are various classifiers such as one nearest neighbor (1NN) [58], decision trees [59], neural networks [60], Bayesian networks [61], supporting vector machines [62] and so on. In this work, we adopt one nearest neighbor classifier to take into account the objectivity, replicability and effectiveness of the comparison. (2) How to set the parameters M and $w1 \sim w3$ for a dataset in the reduced space? On the one hand, we use the training set for parameter search by performing a leave-one-out classification with 1NN classifier and set the parameters to values that yields the maximum accuracy rate from the leave-one-out tuning process. For $w1 \sim w3$, as mentioned in Section 3.4, the selection of parameter searching methods has effect on the result and we learn them by using particle swarm optimization (PSO). In the PSO, the inertia weight is fixed to 0.8, the maximum number of iterations is set to 20, and the acceleration constants are set to 0.2. On the other hand, the default values of the parameters M and $w1 \sim w3$ are 3, 1/3, 1/3 and 1/3. The selection of default values can skip the optimization process and extend application domains like dealing with a new dataset or a dataset with small number of

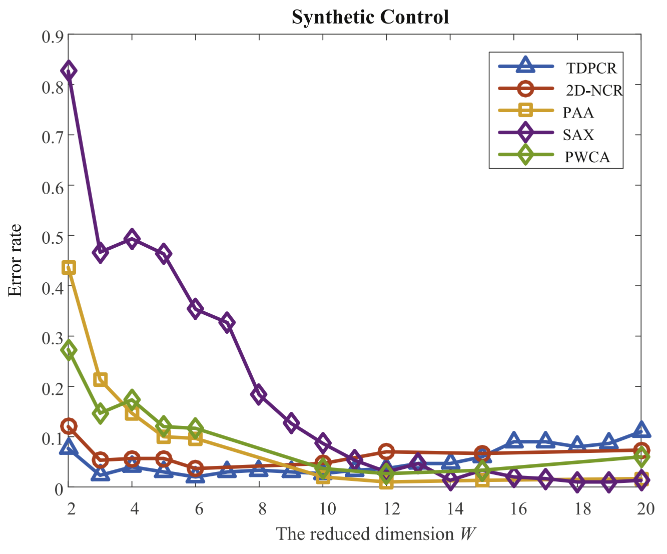


(a)

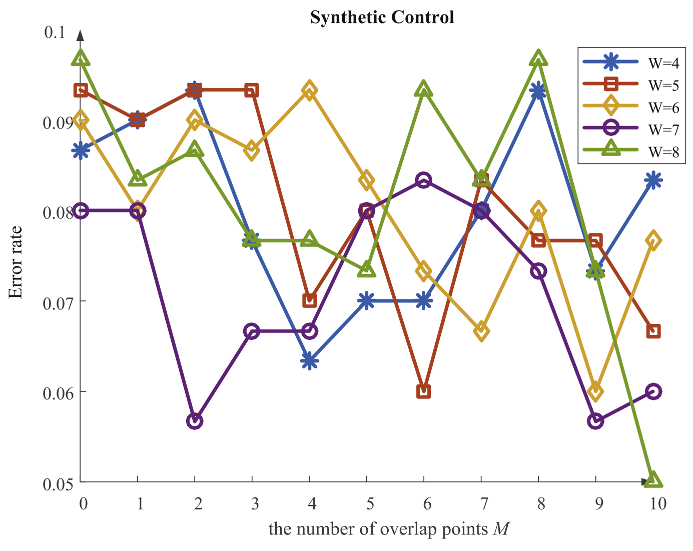


(b)

Fig. 6. (a) Classification results of different representations in the CBF dataset. (b) The classification performance of TDPCR with different overlapping rates in the CBF dataset, according to the reduced dimension $W = 4, 5, 6, 7, 8$.



(a)



(b)

Fig. 7. (a) Classification results of different representations in the Synthetic Contorl dataset. (b) The classification performance of TDPCR with different overlapping rates in the Synthetic Contorl dataset, according to the reduced dimension $W = 4, 5, 6, 7, 8$.

samples, although it may degrade the performance of TDPCR. In fact, the degradation in performance is limited and acceptable, and the experiments in this subsection will demonstrate the flexibility of TDPCR.

Since the major purpose of representation techniques is to reduce the dimensionality of time series and preserve the fundamental characteristics described in Section1, we primarily focus on the classification performance of representations in much lower space. The range of reduced dimension W is limited from 2 to 20. Then, we perform experiments on the CBF and Synthetic Control datasets to compare the reconstruction quality of TDPCR and other state-of-the-art representations, i.e., PWCA, 2D-NCR, SAX and PAA. In fairness, we use the training set to search the best value for SAX parameter, size of the alphabet, between 3 and 10 by performing a leave-one-out classification with 1NN classifier.

For the CBF dataset, we conduct four experiments according to the reduced dimensions $W = (2, 4, 8, 16)$ when using PWCA, 2D-NCR or PAA, because of the limitation of their partitioning strategy which has been discussed in Section 3.1. However, the choice of reduced dimensions of TDPCR and SAX is not under extra constraint. Besides, the range of the number of overlap points M is limited from 2 to 20 according to the length of time series. The experimental results are shown in Fig. 6(a). It can be seen that the performance of TDPCR is much better than other competitors, especial in the lower reduced space ($W < 8$). In addition, when the reduced dimension decreases, the error rates of PAA, SAX, PWCA and 2D-NCR increase significantly, whereas TDPCR is still performing well and at a good standard. Next, we evaluate the influence of the choice of M in the performance with different reduced dimensions according to specific weights $w_1, w_2, w_3 = 1/3$. Fig. 6(b)

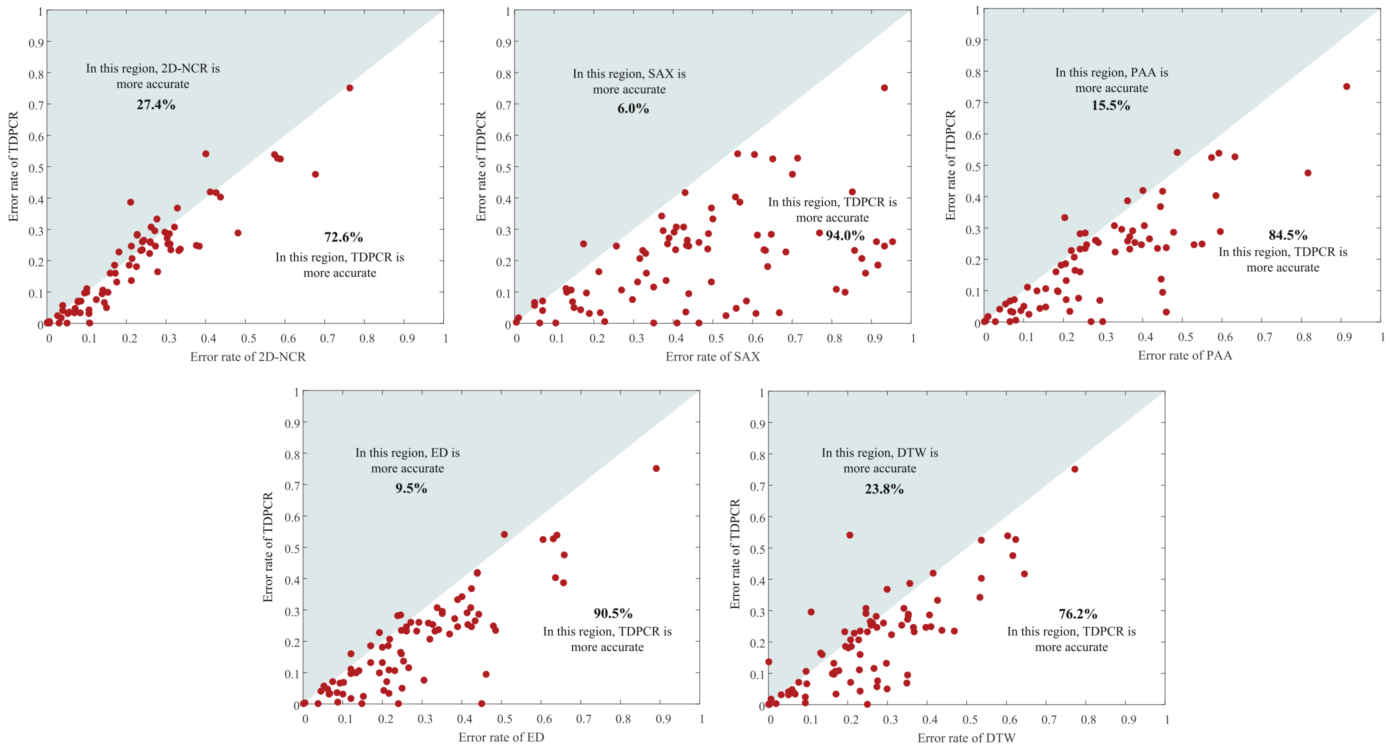


Fig. 8. Scatter plots of error rates of TDPCR against five other approaches. Each point represent a dataset, and points inside the lower triangle region mean that TDPCR has better accuracy than that of the competitive approach. In addition, the proportion of points distributed in different regions is calculated in each plot.

illustrates that when M increases, the performances of TDPCR is changing, and the best choices of M in different reduced dimensions are different. The best values of M are respectively 16, 12, 19, 20, 2 according to $W = 4, 5, 6, 7, 8$. It demonstrates that OPS is an effective way to protect more connection information between consecutive points resulting in better performance. And, selecting a suitable parameter M can actually further improve the performance of TDPCR.

For the Synthetic Control dataset, we make nine experiments according to the reduced dimensions $W = (2, 3, 4, 5, 6, 10, 12, 15, 20)$ when using PWCA, 2D-NCR or PAA, and nineteen experiments when using TDPCR or SAX. The range of the number of overlap points M is limited from 2 to 10 according to the length of time series. The experimental results are shown in Fig. 7(a). It can be seen from the error rate curves that when the reduced dimension decreases, the performances of SAX, PAA, PWCA degrade rapidly, whereas TDPCR and 2D-NCR still keep a low error rate. Fig. 7(a) also shows that TDPCR outperforms other representations in the lower reduced space ($W < 10$). The results validate the good property of TDPCR in much lower reduced space. Then, we analyze the influence of the choice of M when $w_1, w_2, w_3 = 1/3$. In Fig. 7(b), when M increases, error rates in different reduced dimensions all show a wide fluctuation. Fig. 7(b) also shows that the best the choices of M in different reduced dimensions are respectively 4, 6, 9, 2, 10 according to $W = 4, 5, 6, 7, 8$. Because the increasing of W brings both effective information and redundant information, selecting a proper value of W can offer a reasonable trade-off which results in a better performance.

Further, extensive experiments on the other 82 time-series datasets are executed to validate the effectiveness and the efficiency of TDPCR, especially in much lower reduced space. In consideration of various lengths of time series from different datasets, we set a threshold 20% for overlapping rate P to limit the range of M. The reconstruction quality of TDPCR, 2D-NCR, SAX and PAA is compared in different reduced dimensions $W = 2, 3, \ldots, 20$ and their best performances in each dataset are recorded. Besides, we provide the performance of TDPCR(default) which selects the same value of W with TDPCR and the default values of the parameters M and $w1 \sim w3$ to demonstrate the flexibility of TDPCR. The 1NN performances of Euclidean distance (ED) and Dynamic Time Warping with no warping window (DTW) are also considered in our comprehensive evaluation and their results are collated from the literature.

The classification error rates are shown in Appendix. For each dataset, the best performance is indicated in bold. The win-lose-tie results of each competitor compared to TDPCR are also calculated.

Table 2
Average ranks of six approaches across the 84 datasets split by dataset type. When multiple approaches have the same error rate for a dataset, their ranks are equal to the average rank. For example, the ascending order of error rates is "TDPCR" = "DTW" < "2D-NCR" < "ED" < "PAA" < "SAX" for the Trace dataset, and the ranks are 1.5, 1.5, 3, 4, 5, 6.

Type (the number of samples)	TDPCR	2D-NCR	SAX	PAA	ED	DTW
DEVICE (5)	2.80	2.20	5.60	3.40	5.00	2.00
ECG (7)	1.93	3.14	4.43	4.21	3.00	4.29
IMAGE (29)	1.69	2.67	5.31	4.24	4.00	3.09
MOTION (14)	1.86	2.32	4.57	4.32	4.79	3.14
SENSOR (16)	1.47	2.69	4.88	4.34	4.09	3.53
SIMULATED (6)	1.83	3.67	3.92	4.08	4.67	2.83
SPECTRO (7)	1.57	3.64	5.43	3.43	3.00	3.93
Overall (84)	1.76	2.78	4.96	4.14	4.09	3.27

Table 3
The parameter settings of K-Nearest Neighbors query for four different datasets. (n - the number of query series, K - the number of the most similar time series, W - the range of reduced dimensions.)

Dataset	n	K	W
Computers	40	80	$2\sim20$
Synthetic Control	30	60	$2\sim12$
CricketX	30	60	$2\sim20$
Plane	20	20	$2\sim20$

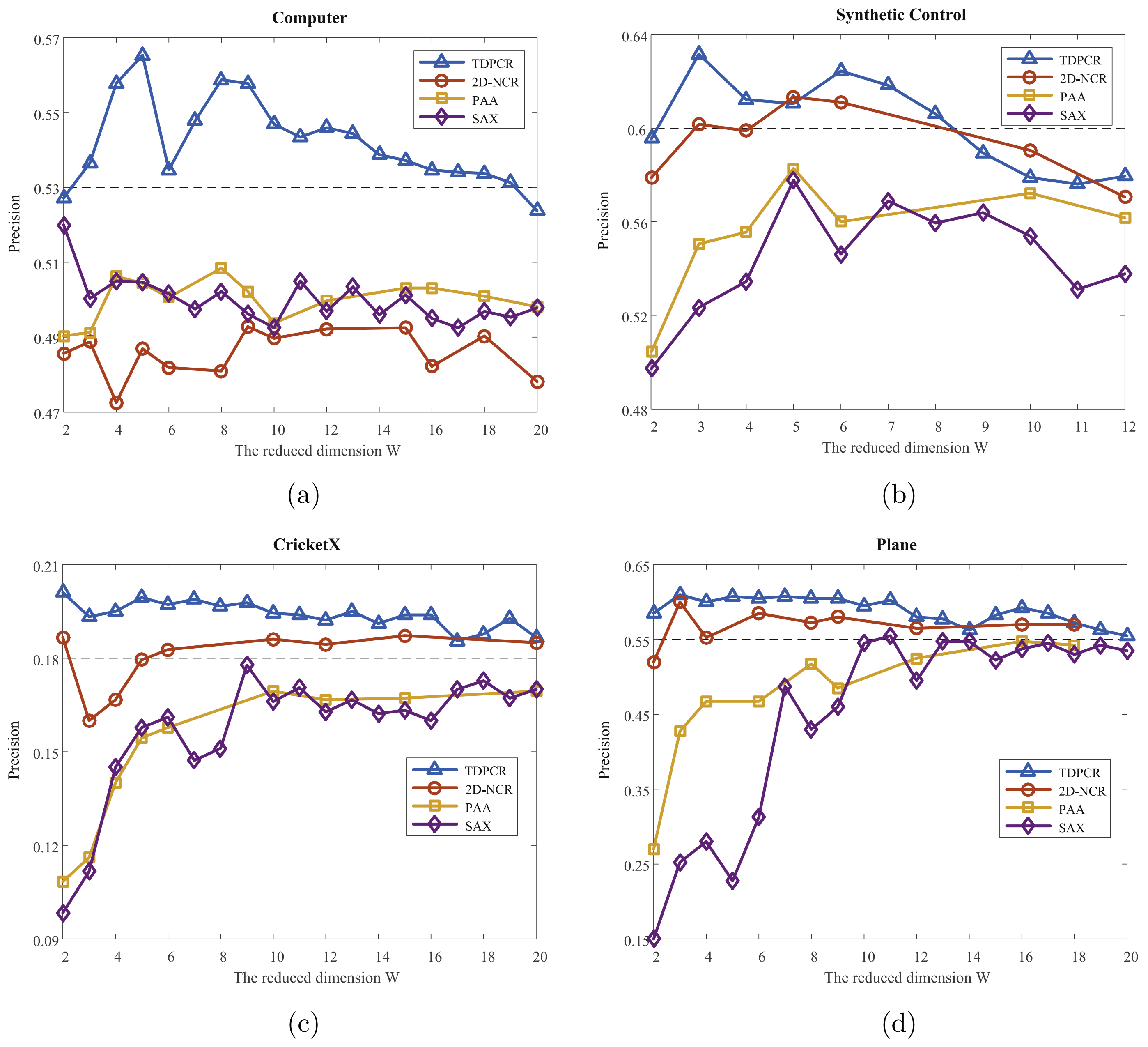


Fig. 9. The results of K-Nearest Neighbors query by TDPCR, 2D-NCR, PAA and SAX according to different reduced dimensions. (a) Computers. (b) Synthetic Control. (c) Cricket X. (d) Plane.

From the Table in Appendix B, we can see that TDPCR ranks first 52 times, gets the lowest average error rate and has the best performance on most of the datasets. TDPCR(default) which selects the default values of parameters also performs well on most of the datasets. The average error rate of TDPCR(default) is lower than other competitors except TDPCR. Table 2 shows the average ranks of these approaches on the 7 different dataset types. The average rank of TDPCR for all datasets is 1.76, less than that of other competitors. On details, TDPCR is best on 6 dataset types, especially for ECG, SENSOR and SPECTRO, but relatively poor on DEVICE. What is more, we provide a more detailed view of the results though scatter plots of error rate of TDPCR versus each competitor. On a scatter plot, there are 84 different points which represent the 84 datasets listed in Appendix A. In Fig. 8, the x-axis is related to the error rates of the competitive approach and the y-axis is the error rates of TDPCR. Points inside the lower triangle region mean that TDPCR has better accuracy than that of the competitive approach for those datasets. From five plots in Fig. 8, we can note that the vast majority of points fall into the lower triangle region and the rest is distributed in the vicinity of the border. It follows that TDPCR performs much better than other four state-of-the-art representation methods and two classical distance measures. Generally, TDPCR is superior to state-of-the-art representation methods for classification, especially in much lower reduced space, and TDPCR(default) which selects the default values of parameters also has satisfactory performance.

4.3. Query by content

Query by content, the most active role in time series data mining, is based on retrieving a set of time series which are most similar to a query series. There are two main types of query performed over the datasets in [63]: ε-range query and K-Nearest Neighbors (KNN) query. The former finds series within distance ε from the query, and the latter just returns the K most similar series. By comparison, K-Nearest Neighbors query is more convenient and practical. Therefore, K-Nearest Neighbors query is used in this study for performance analysis and comparison of TDPCR and the state-of-the-art representations, e.g., 2D-NCR, SAX and PAA. To demonstrate the outstanding quality of TDPCR, we conduct experiments on four datasets Computers, Synthetic Control, CricketX and Plane, and these datasets are from different categories. For each dataset, we randomly choose n time series from the testing set as query series and the rest of the testing set are assembled as queried time series dataset, set a reasonable value of K to return a set of most similar time series, and take experiments in different reduced spaces. The same operation will also be carried out in training set to search the best value for TDPCR parameters M and $w_1 \sim w_3$, and the parameter setting for search is the same as that for classification above. Table 3 gives the values of n, K and the range of W for four datasets. For every query operation, the K answer series are classified according to their class labels, the number of time series of which the class labels are the same as that of a query series is recorded as rq, and the rate of right query is $r = rq/K$. We evaluate the performance of KNN query by the precision Pr, i.e.,

$$Pr = \frac{1}{n}\sum_{i=1}^{n} r_i \tag{17}$$

where r_i represents the rate of right query of the i-th query series in an experiment.

We summarize the results of query by content by plotting the precision for the four datasets from four different application domains, according to different reduced dimensions and show them in Fig. 9. In the experiment of Computer dataset, TDPCR is obvious better than other three approaches. In the experiment of Synthetic Control, the precision of query by content using TDPCR shows a decreasing tendency when W increases and the performance of TDPCR is worse than that of 2D-NCR in the much larger reduced dimensions such as $W = 9, 10, 11,$ but it has a better performance in the lower reduced space ($W < 8$). In the experiments of CricketX and Plane datasets, TDPCR keeps a higher precision and outperforms other competitors in different reduce dimensions. Overall, comparing with other methods, our study shows that TDPCR has superior performance in different type of datasets.

5. Conclusions and future work

In this paper, we propose a novel time series representation based on cloud model, namely Three-Dimensional Piecewise Cloud Representation (TDPCR). It contains a flexible partitioning strategy which not only solves the problem existing in some methods such as Piecewise Aggregate Approximation (PAA), PieceWise Cloud Approximation (PWCA), Two-Dimensional Normal Cloud Representation (2D-NCR) that a time series can't be divided into arbitrary segments, but also protects the connection between consecutive points from damage, thus keeping more effective information. In addition, the three-dimensional cloud model can capture sufficient distribution and variation information of each segment for various tasks, even in much lower dimensionality. Moreover, the new distance measure has the capacity to adjust the weights on the distribution and variation information of time series adaptively in different mining tasks for better performance. We experimentally evaluate TDPCR in the classification and query by content frameworks and compare it with other state-of-the-art representation methods. The experimental results demonstrate that TDPCR is competitive among the state-of-the-art representations, especially in much lower reduced space.

For the future work, we intend to apply the representation to other data mining tasks such as clustering, anomaly detection and prediction. A fast and efficient multi-parameter optimization approach can be developed to further reduce the cost of time and improve TDPCR. In addition, if there is a similarity measure between two three-dimensional clouds using nine characteristics rather than six characteristics used in this work, the performance of TDPCR for tasks may be better.

Acknowledgments

This work is supported by the China Scholarship Council Fund (Grant No. 201806285022) and the National Natural Science Foundation of China (Grant No. 61304118).

Appendix A

Table 4.

Table 4
Summary of the 84 time series datasets: the size of training and testing sets, the length of time series, the number of classes and the types of datasets.

Dataset	TrainSize/TestSize	Length	Classes	Type	Dataset	TrainSize/TestSize	Length	Classes	Type
Adiac	390/391	176	37	IMAGE	MedicalImages	381/760	99	10	IMAGE
ArrowHead	36/175	251	3	IMAGE	MiddlePhalanxAgeGroup	400/154	80	3	IMAGE
Beef	30/30	470	5	SPECTRO	MiddlePhalanxCorrect	600/291	80	2	IMAGE
BeetleFly	20/20	512	2	IMAGE	MiddlePhalanxTW	399/154	80	6	IMAGE
BirdChicken	20/20	512	2	IMAGE	MoteStrain	20/1252	84	2	SENSOR
Car	60/60	577	4	SENSOR	NonInvasiveFatalECGThorax1	1800/1965	750	42	ECG
CBF	30/900	128	3	SIMULATED	NonInvasiveFatalECGThorax2	1800/1965	750	42	ECG
ChlorineConcentration	467/3840	166	3	SIMULATED	OliveOil	30/30	570	4	SPECTRO
CinCECGtorso	40/1380	1639	4	ECG	OSULeaf	200/242	427	6	IMAGE
Coffee	28/28	286	2	SPECTRO	PhalangesOutlinesCorrect	1800/858	80	2	IMAGE
Computers	250/250	720	2	DEVICE	Phoneme	214/1896	1024	39	SENSOR
CricketX	390/390	300	12	MOTION	Plane	105/105	144	7	SENSOR
CricketY	390/390	300	12	MOTION	ProximalPhalanxAgeGroup	400/205	80	3	IMAGE
CricketZ	390/390	300	12	MOTION	ProximalPhalanxCorrect	600/291	80	2	IMAGE
DiatomSizeReduction	16/306	345	4	IMAGE	ProximalPhalanxTW	400/205	80	6	IMAGE
DistalPhalanxAgeGroup	400/139	80	3	IMAGE	RefrigerationDevices	375/375	720	3	DEVICE
DistalPhalanxCorrect	600/276	80	2	IMAGE	ScreenType	375/375	720	3	DEVICE
DistalPhalanxTW	400/139	80	6	IMAGE	ShapeletSim	20/180	500	2	SIMULATED
Earthquakes	322/139	512	2	SENSOR	ShapesAll	600/600	512	60	IMAGE
ECG200	100/100	96	2	ECG	SmallKitchenAppliances	375/375	720	3	DEVICE
ECG5000	500/4500	140	5	ECG	SonyAIBORobotSurface1	20/601	70	2	SENSOR
ECGFiveDays	23/861	136	2	ECG	SonyAIBORobotSurface2	27/953	65	2	SENSOR
FiftyWords	450/455	270	50	IMAGE	StarlightCurves	1000/8236	1024	3	SENSOR
FaceAll	560/1690	131	14	IMAGE	Strawberry	613/370	235	2	SPECTRO

(continued on next page)

Table 4 (*continued*)

Dataset	TrainSize/TestSize	Length	Classes	Type	Dataset	TrainSize/TestSize	Length	Classes	Type
FaceFour	24/88	350	4	IMAGE	SwedishLeaf	500/625	128	15	IMAGE
FacesUCR	200/2050	131	14	IMAGE	Symbols	25/995	398	6	IMAGE
Fish	175/175	463	7	IMAGE	SyntheticControl	300/300	60	6	SIMULATED
FordA	3601/1320	500	2	SENSOR	ToeSegmentation1	40/228	277	2	MOTION
FordB	3636/810	500	2	SENSOR	ToeSegmentation2	36/130	343	2	MOTION
GunPoint	50/150	150	2	MOTION	Trace	100/100	275	4	SENSOR
Ham	109/105	431	2	SPECTRO	TwoLeadECG	23/1139	82	2	ECG
HandOutlines	1000/370	2709	2	IMAGE	TwoPatterns	1000/4000	128	4	SIMULATED
Haptics	155/308	1092	5	MOTION	UWaveGestureLibraryAll	896/3582	945	8	MOTION
Herring	64/64	512	2	IMAGE	UWaveGestureLibraryX	896/3582	315	8	MOTION
InlineSkate	100/550	1882	7	MOTION	UWaveGestureLibraryY	896/3582	315	8	MOTION
InsectWingbeatSound	220/1980	256	11	SENSOR	UWaveGestureLibraryZ	896/3582	315	8	MOTION
ItalyPowerDemand	67/1029	24	2	SENSOR	Wafer	1000/6164	152	2	SENSOR
LargeKitchenAppliances	375/375	720	3	DEVICE	Wine	57/54	234	2	SPECTRO
Lightning2	60/61	637	2	SENSOR	WordSynonyms	267/638	270	25	IMAGE
Lightning7	70/73	319	7	SENSOR	Worms	181/77	900	5	MOTION
Mallat	55/2345	1024	8	SIMULATED	WormsTwoClass	181/77	900	2	MOTION
Meat	60/60	448	3	SPECTRO	Yoga	300/3000	426	2	IMAGE

Appendix B

Table 5.

Table 5
The error rates of seven different approaches on the 84 datasets. The empty cell references that the corresponding approach cant represent time series from a particular dataset within the range of reduced dimension $W = 1 \sim 20$. The best performance of each dataset is indicated in bold. As seen, TDPCR gets better performance on most of the datasets. The win-lose-tie results of each competitor compared to TDPCR and the average error rates are also calculated. (* The last two columns in the table are the value of the parameters when TDPCR gets best performance.)

Dataset	TDPCR	TDPCR (default)	2D-NCR	SAX	PAA	ED	DTW	*W/M	*w1/w2/w3
Adiac	**0.2455**	0.2915	0.2737	0.9335	0.5294	0.3887	0.3960	16/1	0.4926/0.4432/0.0642
ArrowHead	**0.1314**	0.1714	–	0.4971	–	0.2000	0.2970	8/3	0.0260/0.6339/0.3401
Beef	**0.2333**	0.3667	0.3333	0.6333	0.3667	0.3333	0.3670	19/3	0.0000/0.8087/0.1913
BeetleFly	**0.0500**	0.1000	0.1500	0.1500	0.1000	0.2500	0.3000	9/1	0.1594/0.5000/0.3406
BirdChicken	**0.0000**	0.0000	0.0500	0.3500	0.3000	0.4500	0.2500	2/23	0.0000/0.5000/0.5000
Car	**0.1167**	0.2000	–	0.3500	–	0.2667	0.2670	16/7	0.4945/0.3053/0.2002
CBF	**0.0011**	0.0078	0.0056	0.0644	0.0656	0.1478	0.0030	3/4	0.6281/0.0966/0.2753
ChlorineConcentration	**0.2891**	0.3344	0.4807	0.7693	0.5956	0.3500	0.3520	11/4	0.0000/0.0000/1.0000
CinCECGtorso	**0.0688**	0.0920	0.0775	0.1471	0.2913	0.1029	0.3490	14/3	0.3560/0.6290/0.0150
Coffee	**0.0000**	0.0714	0.1071	0.4643	**0.0000**	**0.0000**	**0.0000**	4/0	0.1563/0.7843/0.0594
Computers	0.3680	0.4080	0.3280	0.4960	0.4440	0.4240	**0.3000**	12/6	0.1835/0.5000/0.3165
CricketX	0.3077	0.3256	0.3205	0.4077	0.4051	0.4231	**0.2460**	19/0	0.3881/0.1389/0.4730
CricketY	0.2641	0.2974	**0.2436**	0.4359	0.4179	0.4333	0.2560	9/9	0.3371/0.1895/0.4733
CricketZ	0.2897	0.3051	0.2974	0.4026	0.3744	0.4128	**0.2460**	19/2	0.2764/0.4169/0.3067
DiatomSizeReduction	**0.0327**	0.0915	0.0523	0.6078	0.0719	0.0654	0.0330	3/2	0.2266/0.6839/0.0895
DistalPhalanxOutlineAgeGroup	**0.2075**	0.2200	0.2150	0.8750	0.2275	0.2175	0.2080	2/6	0.1833/0.1461/0.6706
DistalPhalanxOutlineCorrect	0.2350	0.2550	0.2400	0.6283	0.2550	0.2483	**0.2320**	2/4	0.0686/0.5667/0.3647
DistalPhalanxTW	**0.2600**	0.2900	**0.2600**	0.9125	0.2825	0.2725	0.2900	2/15	0.6088/0.2353/0.1559
Earthquakes	0.2547	0.3074	0.3106	**0.1739**	0.2888	0.3261	0.2580	12/0	0.0000/0.5000/0.5000
ECG200	0.1100	0.1800	**0.1000**	0.1300	0.1100	0.1200	0.2300	7/0	0.8656/0.1250/0.0094
ECG5000	0.0716	0.0775	0.0778	**0.0713**	0.0764	0.0751	0.0760	19/0	0.1768/0.5000/0.3232
ECGFiveDays	**0.0430**	0.1336	0.1057	0.1672	0.1417	0.2033	0.2320	12/1	0.0429/0.8965/0.0606
FiftyWords	**0.2242**	0.2352	0.2593	0.3297	0.3297	0.3692	0.3100	13/3	0.4849/0.2744/0.2406
FaceAll	0.2331	0.2503	–	0.3231	–	0.2864	**0.1920**	16/1	0.6049/0.3680/0.0271
FaceFour	**0.0341**	0.0795	0.0682	0.2159	0.2159	0.2159	0.1700	15/0	0.8167/0.0670/0.1163
FacesUCR	0.1059	0.1307	–	0.2707	–	0.2307	**0.0950**	16/2	0.8026/0.0401/0.1574
Fish	**0.1086**	0.1257	–	0.8114	–	0.2171	0.1770	10/0	0.3631/0.3820/0.2549
FordA	**0.2363**	0.2405	0.3355	0.4874	0.4593	0.3410	0.4380	2/7	0.1090/0.4491/0.4419
FordB	**0.2852**	0.3000	0.3086	0.4884	0.4774	0.4422	0.4060	14/3	0.4977/0.1020/0.4003
GunPoint	**0.0067**	0.0133	**0.0067**	0.2267	0.0800	0.0867	0.0930	4/2	0.0865/0.9003/0.0132
Ham	**0.3429**	0.5429	–	0.3714	–	0.4000	0.5330	2/40	0.4935/0.5000/0.0065
HandOutlines	**0.1800**	0.1920	0.2260	0.6380	0.1950	0.1990	0.2020	19/35	0.4486/0.1936/0.3578
Haptics	**0.5260**	0.5682	0.5812	0.7143	0.6331	0.6299	0.6230	12/23	0.1533/0.5219/0.3248
Herring	**0.2344**	0.3594	0.3125	0.4063	0.4375	0.4844	0.4690	4/1	0.4337/0.5000/0.0663
InlineSkate	**0.4764**	0.5254	0.6764	0.7000	0.8164	0.6582	0.6160	8/12	0.1464/0.3028/0.5508
InsectWingbeatSound	**0.4182**	0.4237	0.4258	0.4303	0.4505	0.4384	0.6450	20/0	0.7934/0.0893/0.1173
ItalyPowerDemand	0.0408	0.0583	**0.0398**	0.0700	0.0389	0.0447	0.0500	6/0	0.4819/0.5000/0.0181
LargeKitchenAppliances	0.5413	0.5707	0.4000	0.5627	0.4880	0.5067	**0.2050**	15/9	0.3586/0.1945/0.4469

(*continued on next page*)

Table 5 (*continued*)

Dataset	TDPCR	TDPCR (default)	2D-NCR	SAX	PAA	ED	DTW	*W/M	*w1/w2/w3
Lightning2	0.1639	0.2459	0.2787	0.2131	0.2295	0.2459	**0.1310**	3/39	0.4104/0.0075/0.5821
Lightning7	**0.2466**	0.4109	0.3836	0.4384	0.3973	0.4247	0.2740	3/11	0.5196/0.4640/0.0164
Mallat	**0.0358**	0.0375	0.0559	0.4316	0.0934	0.0857	0.0660	8/36	0.5336/0.2380/0.2283
Meat	**0.0333**	0.1000	0.0833	0.6667	0.0667	0.0667	0.0670	4/34	0.4054/0.5848/0.0099
MedicalImages	**0.2579**	0.2671	0.2605	0.4658	0.3618	0.3158	0.2630	7/4	0.2901/0.6594/0.0505
MiddlePhalanxOutlineAgeGroup	**0.2325**	0.2725	0.2375	0.8575	0.2425	0.2600	0.2500	2/5	0.2889/0.6249/0.0862
MiddlePhalanxOutlineCorrect	0.2833	0.4950	0.2283	0.6467	0.2550	**0.2467**	0.3520	3/8	0.8512/0.1250/0.0238
MiddlePhalanxTW	0.4185	0.4311	0.4110	0.8521	**0.4010**	0.4386	0.4160	2/10	0.2977/0.5000/0.2023
MoteStrain	0.0958	0.1294	**0.0950**	0.1805	0.1861	0.1214	0.1650	17/1	0.2580/0.7207/0.0213
NonInvasiveFatalECGThorax1	0.1847	0.1990	0.2066	0.9170	0.2046	**0.1710**	0.2090	17/3	0.5872/0.2369/0.1760
NonInvasiveFatalECGThorax2	0.1593	0.1679	0.1715	0.8840	0.1817	**0.1201**	0.1350	17/0	0.5264/0.3491/0.1245
OliveOil	**0.1000**	0.1667	**0.1000**	0.8333	0.1333	0.1333	0.1670	4/0	0.8654/0.0706/0.0640
OSULeaf	**0.2479**	0.2479	0.3760	0.4339	0.5496	0.4793	0.4090	7/2	0.2353/0.4608/0.3039
PhalangesOutlinesCorrect	0.2821	0.2925	**0.2273**	0.6131	0.2413	0.2389	0.2720	10/2	0.3892/0.5000/0.1108
Phoneme	**0.7505**	0.7664	0.7637	0.9320	0.9135	0.8908	0.7720	18/10	0.2400/0.5000/0.2600
Plane	**0.0000**	0.1333	**0.0000**	0.1048	0.0286	0.0381	**0.0000**	2/26	0.7529/0.1206/0.1264
ProximalPhalanxOutlineAgeGroup	0.1854	0.2292	**0.1707**	0.9171	0.2049	0.2146	0.1950	2/2	0.4790/0.5000/0.0210
ProximalPhalanxOutlineCorrect	0.2268	0.3367	**0.1821**	0.6838	0.2199	0.1924	0.2160	3/7	0.0968/0.7636/0.1396
ProximalPhalanxTW	0.2600	0.5925	**0.2400**	0.9525	0.2800	0.2925	0.2630	3/8	0.1120/0.8582/0.0298
RefrigerationDevices	**0.5253**	0.5787	0.5893	0.6507	0.5733	0.6053	0.5360	3/24	0.7874/0.1148/0.0979
ScreenType	**0.5387**	0.5707	0.5733	0.6053	0.5920	0.6400	0.6030	2/3	0.5861/0.0535/0.3604
ShapeletSim	**0.0944**	0.1222	0.1389	0.4389	0.4500	0.4611	0.3500	2/3	0.2967/0.5661/0.1372
ShapesAll	**0.1600**	0.1733	**0.1600**	0.3333	0.2417	0.2483	0.2320	15/8	0.2646/0.5000/0.2354
SmallKitchenAppliances	0.3867	0.4293	**0.2107**	0.5680	0.3627	0.6560	0.3570	12/3	0.0000/0.5000/0.5000
SonyAIBORobotSurface1	**0.0765**	0.1032	0.1248	0.2978	0.2379	0.3045	0.2750	3/6	0.2432/0.2749/0.4819
SonyAIBORobotSurface2	**0.1060**	0.1721	0.1406	0.1427	0.1553	0.1406	0.1690	3/8	0.0796/0.1659/0.7545
StarlightCurves	**0.0248**	0.0254	0.0259	0.5339	0.1139	0.1512	0.0930	8/8	0.3029/0.2143/0.4828
Strawberry	**0.0473**	0.0669	0.0718	0.5595	0.1550	0.0620	0.0600	10/2	0.7902/0.2020/0.0078
SwedishLeaf	**0.0720**	0.0976	0.0864	0.5856	0.2080	0.2112	0.2080	19/1	0.3597/0.5546/0.0856
Symbols	**0.0312**	0.0372	0.1055	0.1889	0.4603	0.1005	0.0500	8/8	0.4324/0.2545/0.3131
SyntheticControl	0.0167	0.0866	0.0367	0.0100	0.0100	0.1200	**0.0070**	6/0	0.5916/0.4008/0.0076
ToeSegmentation1	**0.2061**	0.2938	–	0.3158	–	0.3202	0.2280	11/6	0.4954/0.3403/0.1643
ToeSegmentation2	**0.1000**	0.1461	0.1538	0.1308	0.1846	0.1923	0.1620	13/5	0.7974/0.1969/0.0057
Trace	**0.0000**	**0.0000**	0.0300	0.4100	0.2700	0.2400	**0.0000**	2/1	0.2513/0.7188/0.0298
TwoLeadECG	0.1378	0.2546	0.2125	0.3819	0.4469	0.2529	**0.0000**	6/0	0.8418/0.0467/0.1114
TwoPatterns	0.0678	0.2240	0.1443	**0.0513**	0.0650	0.0933	0.0960	9/3	0.9593/0.0403/0.0004
UWaveGestureLibraryAll	0.0567	0.0854	**0.0394**	0.0508	0.0550	0.0519	0.2730	8/10	0.4813/0.5000/0.0187
UWaveGestureLibraryX	0.2468	0.2602	**0.2127**	0.2566	0.2596	0.2607	0.3660	4/4	0.3913/0.4550/0.1537
UWaveGestureLibraryY	0.3082	0.3188	**0.2621**	0.4269	0.3280	0.3384	0.3420	3/2	0.2552/0.5000/0.2448
UWaveGestureLibraryZ	0.2956	0.3018	0.2711	0.3749	0.3470	0.3504	**0.1080**	5/10	0.3842/0.3011/0.3147
Wafer	0.0039	0.0060	**0.0008**	0.0050	0.0047	0.0045	0.0200	7/5	0.4421/0.3139/0.2441
Wine	0.3333	0.5740	0.2778	0.5000	**0.2037**	0.3889	0.4260	3/4	0.8586/0.0757/0.0657
WordSynonyms	**0.2680**	**0.2680**	0.3025	0.3887	0.3668	0.3824	0.3510	17/0	0.3333/0.3333/0.3334
Worms	**0.4033**	**0.4033**	0.4365	0.5580	0.5856	0.6354	0.5360	15/16	0.2274/0.4430/0.3297
WormsTwoClass	**0.2541**	0.3149	0.3039	0.3867	0.3812	0.4144	0.3370	14/6	0.0414/0.5000/0.4586
Yoga	**0.1327**	0.1513	0.1750	0.3100	0.2073	0.1697	0.1640	14/6	0.5698/0.0000/0.4302
Win/lose/tie	/	0/81/3	19/61/4	5/79/0	11/71/2	7/76/1	17/64/3	/	/
Average error rate	0.1981	0.2468	0.2884	0.4548	0.3503	0.2861	0.2546	/	/

References

[1] C.A. Ralanamahatana, J. Lin, D. Gunopulos, E. Keogh, M. Vlachos, G. Das, Mining time series data, in: Data Mining and Knowledge Discovery Handbook, Springer, Boston, MA, 2005, pp. 1069–1103.

[2] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proc. VLDB Endow. 1 (2) (2008) 1542–1552.

[3] S. Miao, U. Vespier, R. Cachucho, M. Meeng, A. Knobbe, Predefined pattern detection in large time series, Inf. Sci. 329 (2016) 950–964.

[4] M. Jalili, M. Perc, Information cascades in complex networks, J. Complex Netw. 5 (5) (2017) 665–693.

[5] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.

[6] J. Ares, J.A. Lara, D. Lizcano, S. Surez, A soft computing framework for classifying time series based on fuzzy sets of events, Inf. Sci. 330 (2016) 125–144.

[7] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Min. Knowl. Discov. 29 (3) (2015) 565–592.

[8] L.N. Ferreira, L. Zhao, Time series clustering via community detection in networks, Inf. Sci. 326 (2016) 227–242.

[9] H. Yahyaoui, H.S. Own, Unsupervised clustering of service performance behaviors, Inf. Sci. 422 (2018) 558–571.

[10] H. Wang, M. Tang, Y. Park, C.E. Priebe, Locality statistics for anomaly detection in time series of graphs, IEEE Trans. Signal Processing 62 (3) (2014) 703–717.

[11] N. Begum, E. Keogh, Rare time series motif discovery from unbounded streams, Proc. VLDB Endow. 8 (2) (2014) 149–160.

[12] M. Perc, Self-organization of progress across the century of physics, Scientific Reports 3 (2013) 1720.

[13] L.S. Johnson, S.R. Eddy, E. Portugaly, Hidden markov model speed heuristic and iterative HMM search procedure, BMC Bioinform. 11 (1) (2010) 431.

[14] E. Erdem, J. Shi, ARMA based approaches for forecasting the tuple of wind speed and direction, Appl. Energy 88 (4) (2011) 1405–1414.

[15] T. Kuhn, M. Perc, D. Helbing, Inheritance patterns in citation networks reveal scientific memes, Phys. Rev. X 4 (4) (2014) 041036.

[16] E. Fuchs, T. Gruber, J. Nitschke, B. Sick, Online segmentation of time series based on polynomial least-squares approximations, IEEE Trans. Pattern Anal. Mach. Intell. 32 (12) (2010) 2232–2245.

[17] H. Zhao, Z. Dong, T. Li, X. Wang, C. Pang, Segmenting time series with connected lines under maximum error bound, Inf. Sci. 345 (2016) 1–8.

[18] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Min. Knowl. Discov. 30 (2) (2016) 476–509.

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

[20] T.C. Fu, A review on time series data mining, Eng. Appl. Artif. Intell. 24 (1) (2011) 164–181.

[21] D. Li, Membership clouds and membership cloud generators, Comput. Res. Dev. 32 (6) (1995) 15–20.

[22] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, Proc. VLDB Endow. 1 (2) (2008) 1542–1552.

[23] R. Agrawal, C. Faloutsos, A. Swami, Efficient similarity search in sequence databases, in: Proceedings of International Conference on Foundations of Data Organization and Algorithms, Springer, Berlin, Heidelberg, 1993, pp. 69–84. October.

[24] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, Fast subsequence matching in time-series databases, 23, ACM, 1994, pp. 419–429.

[25] K.P. Chan, A.W.C. Fu, Efficient time series matching by wavelets. in data engineering, 1999, in: Proceedings of the 15th International Conference on (pp. 126–133), IEEE, 1999. March.
[26] K.J. Astrom, On the choice of sampling rates in parametric identification of time series, Inf. Sci. 1 (3) (1969) 273–278.
[27] N.Q.V. Hung, D.T. Anh, An improvement of PAA for dimensionality reduction in large time series databases, in: Proceedings of the Pacific Rim International Conference on Artificial Intelligence (pp. 698–707), Springer, Berlin, Heidelberg, 2008. December.
[28] S. Lee, D. Kwon, S. Lee, Dimensionality reduction for indexing time series based on the minimum distance, J. Inf. Sci. Eng. 19 (4) (2003) 697–711.
[29] C. Ratanamahatana, E. Keogh, A.J. Bagnall, S. Lonardi, A novel bit level time series representation with implication of similarity search and clustering, Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 771–777), Springer, Berlin, Heidelberg, 2005. May.
[30] Q. Chen, L. Chen, X. Lian, Y. Liu, J.X. Yu, Indexable PLA for efficient similarity search, in: Proceedings of the 33rd International Conference on Very Large Data Bases (pp. 435–446, VLDB Endowment, 2007. September.
[31] H. Li, C. Guo, Piecewise cloud approximation for time series mining, Knowl.-Based Syst. 24 (4) (2011) 492–500.
[32] W. Deng, G. Wang, J. Xu, Piecewise two-dimensional normal cloud representation for time-series data mining, Inf. Sci. 374 (2016) 32–50.
[33] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, ACM Sigmod Record 30 (2) (2001) 151–162.
[34] V. Megalooikonomou, G. Li, Q. Wang, A dimensionality reduction technique for efficient similarity analysis of time series databases, in: Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management (pp. 160–161), ACM, 2004. November.
[35] V. Megalooikonomou, Q. Wang, G. Li, C. Faloutsos, A multiresolution symbolic representation of time series, in: Proceedings of the 21st International Conference on Data Engineering, 2005. ICDE 2005 (pp. 668–679), IEEE, 2005. April.
[36] T. Stiefmeier, D. Roggen, G. Troster, Fusion of string-matched templates for continuous activity recognition, in: Proceedings of the 11th IEEE International Symposium on Wearable Computers, 2007 (pp. 41–44), IEEE, 2007. October.
[37] J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series, Data Min. Knowl. Discov. 15 (2) (2007) 107–144.
[38] J. Shieh, E. Keogh, iSAX: disk-aware mining and indexing of massive time series datasets, Data Min. Knowl. Discov. 19 (1) (2009) 24–57.
[39] B. Lkhagva, Y. Suzuki, K. Kawagoe, Extended SAX: extension of symbolic aggregate approximation for financial time series data representation, DEWS2006 4A-i8 7 (2006).
[40] Y. Sun, J. Li, J. Liu, B. Sun, C. Chow, An improvement of symbolic aggregate approximation distance measure for time series, Neurocomputing 138 (2014) 189–198.
[41] F. Korn, H.V. Jagadish, C. Faloutsos, Efficiently supporting ad hoc queries in large datasets of time sequences, in: Proceedings of the ACM Sigmod Record (Vol. 26, No. 2, pp. 289–300), ACM, 1997. June.
[42] K.V. Ravi Kanth, D. Agrawal, A. Singh, Dimensionality reduction for similarity searching in dynamic databases, in: Proceedings of the ACM SIGMOD Record (Vol. 27, No. 2, pp. 166–176), ACM, 1998. June.
[43] E. Keogh, S. Chu, D. Hart, M. Pazzani, An online algorithm for segmenting time series, in: Proceedings of the IEEE International Conference on Data Mining, 2001. ICDM 2001 (pp. 289–296), IEEE, 2001.
[44] F. Gullo, G. Ponti, A. Tagarelli, S. Greco, A time series representation model for accurate and fast similarity detection, Pattern Recognit. 42 (11) (2009) 2998–3014.
[45] M. Krawczak, G. Szkatula, An approach to dimensionality reduction in time series, Inf. Sci. 260 (2014) 15–36.
[46] M. Zhang, D. Pi, A new time series representation model and corresponding similarity measure for fast and accurate similarity detection, IEEE Access 5 (2017) 24503–24519.
[47] S.J. Wilson, Data representation for time series data mining: time domain approaches, Wiley Interdiscip. Rev.: Comput. Stat. 9 (1) (2017).
[48] Y.Q. Feng, H.L. Wang, M.K. Cao, Intelligent decision support system based on cloud model, in: Proceeding of the 2006 Chinese Control and Decision Conference, Tianjin, China (pp. 1081–1084), 2006. July.
[49] K. Qin, K. Xu, F. Liu, D. Li, Image segmentation based on histogram analysis utilizing the cloud model, Comput. Math. Appl. 62 (7) (2011) 2824–2833.
[50] P.E. Danielsson, Euclidean distance mapping, Comput. Graph. Image Process. 14 (3) (1980) 227–248.
[51] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, Addressing big data time series: mining trillions of time series subsequences under dynamic time warping, ACM Trans. Knowl. Discov. Data (TKDD) 7 (3) (2013) 10.
[52] L.J. Latecki, Q. Wang, S. Koknar-Tezel, V. Megalooikonomou, Optimal subsequence bijection, in: Proceeding of the Seventh IEEE International Conference on Data Mining, ICDM 2007. (pp. 565–570), IEEE, 2007. October.
[53] V.V. Markellos, W. Black, P.E. Moran, A grid search for families of periodic orbits in the restricted problem of three bodies, Celest. Mech. 9 (4) (1974) 507–512.
[54] G.C. Chen, J.S. Yu, Particle swarm optimization algorithm, Inf. Control-SHENYANG 34 (3) (2005) 318.
[55] D.J. Hand, Genetic algorithms in search, optimization and machine learning, Stat. Comput. 4 (2) (1994) 158.
[56] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, The UCR time series classification archive, 2015, http://www.cs.ucr.edu/~eamonn/time_series_data/.
[57] D. Li, Y. Du, Artificial Intelligence with Uncertainty, CRC Press, 2007.
[58] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.
[59] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[60] F.V. Jensen, An Introduction to Bayesian Networks (Vol. 210, pp. 1–178), UCL Press, London, 1996.
[61] D.F. Specht, Probabilistic neural networks, Neural Netw. 3 (1) (1990) 109–118.
[62] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support Vector Machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.
[63] P. Esling, C. Agon, Time-series data mining, ACM Comput. Surv. (CSUR) 45 (1) (2012) 12.

Gangquan Si received the M.Sc. and Ph.D. degrees in engineering from Xian Jiaotong University (XJTU), Xian, China, in 2005 and 2009, respectively. He is currently an Associate Professor with the School of Electrical Engineering, XJTU. His current research interests include time series data mining, anomaly detection and knowledge discovery in databases.

Kai Zheng received the B.Sc. degree from the School of Electrical Engineering, Xian Jiaotong University (XJTU), China, in 2016, and he is currently pursuing the M.Sc. degree there. His current research interests include time series data mining, feature extraction and pattern recognition.

Zhou Zhou received the B.Sc. degree from the School of Electrical Engineering, Fuzhou University, China, in 2015. Now he is a Ph.D. candidate in the Department of Electrical Engineering, Xian Jiaotong University. His current research interests include the application of data mining, deep learning and time series analysis.

Chengjie Pan received the B.Sc. degree from the School of Electrical Engineering, Xian Jiaotong University (XJTU), China, in 2018, and he is currently pursuing the M.Sc. degree there. His research activities are within the areas of knowledge and data management, and artificial intelligence.

Xiang Xu received the B.Sc. degree from the School of Electrical Engineering, Xian Jiaotong University (XJTU), China, in 2017, and he is currently pursuing the M.Sc. degree there. His research activities are within the areas of artificial intelligence and data integration.

Qu kai received the B.Sc. degree from the School of Electrical Engineering, Xidian University, China, in 2017. He is currently pursuing the M.Sc. degree in electrical engineering at Xian Jiaotong University. His current research interests include time series data mining, and pattern recognition.

Yanbin Zhang received the B.Sc. degree from the School of Electrical Engineering, Xian Jiaotong University (XJTU), China, in 1977, and he is currently a Professor there. His research interests are data mining, big data analysis and predictive modeling.