IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 24, 2017, accepted October 16, 2017, date of publication October 20, 2017,
date of current version November 28, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2764633

A New Time Series Representation Model and Corresponding Similarity Measure for Fast and Accurate Similarity Detection

MIAOMIAO ZHANG AND DECHANG PI
College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
Corresponding author: Dechang Pi (nuaacs@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant U1433116 and in part by the Fundamental Research Funds for the Central Universities under Grant NP2017208.

ABSTRACT Data representation and similarity measurement are two basic aspects of similarity detection in time series data mining. In this paper, we present two novel approaches to perform similarity detection efficiently and effectively. One is composed of a new time series representation model and a corresponding similarity measure, which is called fragment alignment distance (FAD); the other applies dynamic time warping to the representation model of FAD and is called FAD_DTW. The new data representation model is based on the trend information of time series, which can provide a concise yet feature-rich representation of time series. FAD is able to align the segments of time series in linear time, which greatly accelerates the similarity detection process. We extensively compare FAD and FAD_DTW with state-of-the-art time series representation models and similarity measures in classification and clustering frameworks. Experimental results from efficiency and effectiveness validations on various data sets demonstrate that FAD and FAD_DTW can achieve fast and accurate similarity detection. In particular, FAD is much faster than the other methods.

INDEX TERMS Time series data mining, data representation models, similarity measure.

I. INTRODUCTION

A time series is a sequence of ordered numeric values between which an interval of points is defined. Time series are generally used to indicate the change of an object with time; hence, large amounts of such data are available from many domains, including speech recognition [1], financial and market data analysis [2], biomedical measurement [3], sensor networking [4], and moving-object trajectory tracing [5].

The management and analysis of time series data mainly focus on the similarity search and detection of time series, which give rise to many research tasks such as query by content [6], clustering [7], classification [8], segmentation [9], prediction [10], anomaly detection [11], and motif discovery [12], [13]. Considering the representativeness of classification and clustering tasks in the field of time series data mining, in this work, we take them as evaluation frameworks for similarity search and detection.

Time series data mining is a complex process due to numerous factors. The most salient problems lie in the high dimensionality of time series data and the difficulty of devising an appropriate similarity measure for various data. To address the high-dimensionality problem, researchers have developed various dimensionality reduction methods. Many of these methods have great effects on accelerating the process and saving storage space. However, they inevitably affect the accuracy of the similarity search. To normalize the similarity detection problem and guide the research work, many scholars have noted various benchmarks for similarity measurement algorithms [12]. Most of them can be classified as one of two types:

- Data representation models. Representing data in a form that can be effectively processed is the first step of data mining. The ideal representation of time series not only can maintain the original features of the data but also has a simple format. In addition, it can accelerate the detection process with a simultaneous emphasis on accuracy improvement. Hence, the representation model should be realized in a low-dimensional space and consider the basic distribution of the data.
- Similarity measures. Similarity measurement is the central technique of similarity search and detection. Distinguishing between two time series or formalizing

the difference between two time series in accordance with human common sense is the crucial problem of similarity measures. Thus, a reasonable similarity measure should have the following characteristics: consistency with human cognition, consideration of the most prominent features on both the local and global scales, and the capability to unconditionally identify arbitrary objects.

In this respect, we insist that some special requirements should be satisfied by all similarity measures and representation models to support fast and accurate similarity detection of time series, which are listed as follows:

- Time-warping awareness. It is quite common that the time series are shifted along the time axis [15]. Although anyone could confirm that shifted time series are very similar, not all similarity measures are able to address this type of similarity. In such cases, two time series might be unexpectedly identified as dissimilar to each other. Since dynamic time warping can automatically align the numerical values on different time axes, it is widely used to handle the time series shifting problem.
- Capability to handle time series with unequal lengths. An algorithm that can only cope with time series of the same length has severe limitations in applications because most time series in practical scenarios are not equal in length. To apply such algorithms to time series of unequal length, additional preprocessing is required to transform time series into series of equal length, which usually leads to a decrease in performance.
- Low computational complexity. To address the high dimensionality of time series, similarity detection should be performed with reasonably low complexity.
- Ability to capture essential features. It is without doubt that time series approximation should preserve as much information of the original series as possible. To obtain good accuracy in similarity detection, time series representation models should capture the essential features of the original data rather than discarding them in the pursuit of dimension reduction. For this purpose, a reasonable time series representation model should be able to capture the important features of data.

According to the above requirements, the motivation of this paper is to design a new time series representation model and a new corresponding similarity measure that can achieve fast and accurate similarity detection. First, the new representation model should be able to not only approximate a time series in lower dimensions but also preserve the important features. Second, it should be sensitive to the distribution and variation of the time series. In other words, the new method ought to produce different results for different distributions of data. Moreover, the new algorithm should be applicable to different kinds of data sets. With the rapid development of the information industry, a great variety of data are produced every day, and an algorithm that can only address a single type of data has severe limitations in application. Finally, it ought to cope with time series faster than existing algorithms. Due to the challenge of large amounts of data, many existing similarity measures cannot accomplish tasks in reasonable amounts of time. To address this, the new approach must handle time series as quickly as possible.

In this paper, we propose a novel time series representation model and a corresponding similarity measure named FAD. FAD first estimates the derivative of the time series, which can capture trend information about the series. Then, by setting a threshold, FAD transforms the derivative sequences into symbolic ones. A fragment (segment) of the time series is obtained by merging adjacent points with the same symbols. In other words, FAD is based on the alignment between fragments, while the majority of existing methods are based on the alignment between points. The similarity measure of FAD aims at diagonally mapping similar change trends between time series. Moreover, the computational complexity of most state-of-the-art similarity measures is $O(n^2)$, whereas FAD can be carried out in linear time.

The contributions of our work are as follows:

- Like other data representation models, Dynamic Time Warping can be applied to the representation model of FAD directly;
- FAD is devised based on the concept that similar time series have similar change trends. It is consistent with human cognition and available for various data types;
- FAD is time-warping-aware and can deal with data of unequal length in linear time;
- FAD transforms the comparison between points into the comparison between change trends, which can address the high dimensionality as well as capture the essential features of time series.

To evaluate the performance of FAD, we conducted an extensive experiment by using clustering and classification frameworks. This evaluation inevitably involved prominent state-of-the-art methods for both the time series representation models and similarity measures. Experimental evidence that FAD is an effective and efficient method in similarity detection is presented.

The rest of the paper is organized as follows. Section 2 introduces the state-of-the-art methods for time series representation models and similarity measures. Section 3 presents the FAD algorithm in detail. Section 4 demonstrates the experimental frameworks. Section 5 discusses the data sets and the experiment results. The paper is concluded in Section 6.

II. RELATED WORKS

A. DATA REPRESENTATION MODEL AND DIMENSIONALITY REDUCTION

As we mentioned in Introduction, time series are high-dimensional, and it is time consuming to address them directly. The main purpose of time series representation is to express time series in a concise as well as feature-rich way. In the past two decades, numerous time series representation models have been developed [16], [17]. Most of them can be divided into three categories, namely, approximating time

series based on piecewise-discontinuous functions, approximating time series based on continuous polynomials and extracting features of time series based on hidden models.

The first category includes many widely used representation models. Piecewise Aggregate Approximation (PAA) [18] can transform a time series of n points into a new sequence with p segments ($p \ll n$), where each segment is of size equal n/p and is represented by the mean value of the data points falling within the sliding window. PAA is easy to implement and quite effective in many applications. Adaptive Piecewise Constant Approximation (APCA) [19] is similar to PAA; it also approximates a time series by a sequence of segments. Each segment is the mean value of its data points. However, unlike PAA, APCA is able to handle segments with different lengths, which makes it more suitable for dealing with burst signals. Due to the difference in the segments' lengths, APCA requires a two-dimensional array to represent data, while PAA only needs a one-dimensional array. Piecewise Linear Approximation (PLA) [20] represents time series by a piecewise linear function, such as a set of line segments. There are several kinds of implementation of PLA, which can be grouped into three classes, namely, window-sliding, bottom-up and top-down [21]. The bottom-up approach is the most commonly used one. Symbolic Aggregate approXimation (SAX) [22] transforms a time series into some discrete strings, which are composed of different symbols. This algorithm includes two phases. The first phase uses the PAA algorithm to reduce the dimensionality of the time series. The second transforms the mean values of the segments into discrete strings. Derivative time series Segment Approximation (DSA) [14] approximates time series in the derivative version of the original series. DSA is able to segment time series automatically by setting the standard deviation of each segment as a natural threshold.

The time complexity of representing a time series of n points according to PAA, SAX and DSA is $O(n)$, whereas that of APCA is $O(nlog(n))$. The complexity of PLA depends on the implementation method. The fastest version of PLA can be performed in linear time; when using the bottom-up approach, the complexity of PLA is $O(nlog(n))$. The approaches based on piecewise discontinuous functions are easily combined with similarity measure algorithms to perform better and faster in time series data mining.

Another approach to dimensionality reduction is based on approximating time series with continuous polynomials. The most commonly used methods including Discrete Wavelet Transform (DWT) [23], [24], Discrete Fourier Transforms (DFT) [25], Singular Value Decomposition (SVD) [26], [27] and Chebyshev polynomials [28], [29]. DFT projects time series on a basis of sine and cosine functions in the real domain. The representation model is a set of sinusoidal coefficients, which can be regarded as features of the original data. The Chebyshev polynomial approach is quite similar to DFT; the only difference is that the Chebyshev approach takes Chebyshev polynomials as the basis functions. Instead of using a fixed set of orthonormal basis functions, DWT employs scaled and shifted versions of a mother wavelet function. Therefore, DWT can give a multiresolution decomposition of the time series and take into account low frequencies over larger intervals, thereby yielding better accuracy [30]. A great number of mother wavelet functions are available and the Haar function is frequently used [31]. To find the best mapping in a lower-dimensional space, SVD uses space rotation and truncation of the data matrix. Hence, SVD is computationally expensive compared with the other methods described above.

In the representation models based on continuous polynomials, DFT is of complexity $O(nlog(n))$. The computational complexity of DWT and the Chebyshev approach is $O(n)$, while that of SVD is $O(n^3)$.

The last approach to dimensionality reduction is based on the assumption that the measured time series are produced by underlying models. The aim of this kind of approach is to find the parameters of such a model as a representation. Two time series are therefore regarded as similar to each other if they are produced by the same set of parameters of the underlying model. There are several parametric temporal models, included ARMA models [32], Markov Chains (MCs) [33], and HMM [34]. MCs are simpler than HMM and more suitable for handling shorter time series because of the limitation in expressive power.

B. SIMILARITY MEASUREMENT

Similarity measurement is of great importance in the field of time series data mining. There are two basic approaches to similarity measurement: one is represented by the Dynamic Time Warping (DTW) algorithm and the other is based on Edit Distance (ED).

DTW [35] is widely applied to execute similarity search and detection in time series. DTW performs a non-linear mapping of one series to the other by minimizing the total distance between them. Although DTW is time warping-aware, ''singularity'' problems tend to emerge, which reduce the accuracy and lead to misalignment between time series. For example, a single point of one series may be aligned with a large partition of another. This phenomenon is unexpected in most cases. To address this problem, Keogh and Pazzani [36] presented a variant of DTW called Derivative Dynamic Time Warping (DDTW). The novelty of DDTW is that derivatives of time series data points are estimated to obtain the trend information about the original data and search for a new warping path that is more robust to singularities. For instance, two data points that have the same values and different trends are correctly not aligned with each other when using DDTW, whereas with DTW, the two points will be mapped as similar. Indeed, DDTW can be viewed as DTW equipped with a preprocessing step. Another way to address the singularity phenomenon is to constrain the warping path of DTW [37]. This technique uses DTW's ''warping envelope'' constraint to obtain the lower bound of the DTW distance, thereby pruning off some of the costly distance

computations. The learned constraints have two additional benefits: the improvement of accuracy due to relief of the singularity problem and faster similarity search and detection by pruning off computations. Several approaches for constraining the warping path of DTW have been developed, including Sakoe-Chiba Band, Itakura Parallelogram and Ratanamahatana-Keogh Band [38]. All of these methods are of computational complexity $O(n^2)$.

ED is used to calculate the similarity between two strings; the distance between them is the minimum number of operations needed to transform one string into the other. The Longest Common SubSequence (LCSS) [39] is a variant of ED that defines the distance between two series as the length of their longest common subsequence. By setting a threshold, LCSS can address noisy time series by executing approximate matching rather than exact matching of time series. Like LCSS, Edit Distance with Real sequences (EDR) [40] sets a threshold to remove noisy effects; however, EDR computes the difference between two series rather than the similarity. Edit distance with Real Penalty (ERP) [41] is a metric and it does not require any thresholds to address noise; however, ERP needs to choose a value for gap elements. It should be noted that the methods based on ED can be performed in $O(n^2)$.

III. FRAGMENT ALIGNMENT DISTANCE

A. DEFINITIONS

The purpose of this subsection is to give clear definitions of the terms used throughout this article.

Definition 1) A time series T consisting of n ordered numerical points can be defined as:

$$T = (x_1, x_2, \ldots, x_n), \quad x_i \in \mathbb{R}.$$

A time series is usually an observation result of an underlying process. It can thus be defined as a set of successive time instants.

Time series can record the full set of observations of a process and may be of sizable length. Especially for streaming processes, they are semi-infinite as time instants are continually added to the series. Thus, it is necessary to consider the subsequences of a series.

Definition 2) Given a time series T of length n, a subsequence S of T is a part of the series T that consists of contiguous time instants of length m ($m \leq n$):

$$S = (x_k, x_{k+1}, \ldots, x_{k+m-1}), \quad 1 \leq k \leq n-m+1.$$

Definition 3) Given a time series T of length n, a representation of T is a model $\bar{T}$ of reduced dimensionality d ($d \ll n$) such that $\bar{T}$ approximates or extracts the main features of T.

Definition 4) The similarity measure $D(T, U)$ of time series T and U is a function that measures the distance between them. $D(T, U)$ takes two time series as inputs and returns the distance between them. This distance cannot be a negative value, that is, $D(T, U) \geq 0$.

B. DERIVATIVE ESTIMATION

The derivative estimate of a given time series T is a sequence $\hat{T} = (\hat{x}_1, \ldots, \hat{x}_h, \ldots, \hat{x}_n)$, in which each element is the first-derivative estimate of the point in T.

1) DDTW ESTIMATION MODEL

Keogh and Pazzani [36] proposed an easy yet effective derivative estimation model, which we hereinafter call the DDTW estimation model. This model computes each point's first-derivative estimate by using the mean value of the slopes of the lines from the left adjacent point to the current point and the right adjacent point. Formally,

$$\hat{x}_h = \begin{cases} \hat{x}_{h+1} & h = 1 \\ \frac{1}{2}\left[(x_h - x_{h-1}) + \frac{1}{2}(x_{h+1} - x_{h-1})\right] & h \in [2, \ldots, n-1] \\ \hat{x}_{h-1} & h = n \end{cases}. \tag{1}$$

Although the DDTW estimation model is simple, the derivative estimates of the first and last points in the series are not accurate enough.

2) DSA ESTIMATION MODEL

Gullo *et al.* [14] presented a more accurate way to approximate the first derivative of a time series; we refer to it as the DSA estimation model, which modifies the DDTW estimation model by considering the slope of the line from the current point to the right adjacent one. In other words, the DSA estimation model only considers the slope of the line from the left adjacent point to the right adjacent point. The derivatives of the first and last points in the series are calculated by their adjacent points as well. Formally,

$$\hat{x}_h = \begin{cases} x_{h+1} - x_h & h = 1 \\ \frac{1}{2}(x_{h+1} - x_{h-1}) & h \in [2, \ldots, n-1] \\ x_h - x_{h-1} & h = n \end{cases} \tag{2}$$

Compared to the DDTW estimation model, the derivative forms of the first and last points of the DSA estimation model are more reasonable. According to Gullo *et al.* [14], the DSA derivative estimation model results in a better derivative-based feature space than DDTW.

C. SEGMENTATION AND DATA REPRESENTATION

A derivative estimation sequence is a feature-rich series that contains trend information about the original data. Due to the better performance of DSA derivative estimation in approximating the derivative values of time series, FAD performs segmentation based on it.

In our approach, the segmentation process is the same process as the formation of fragments. The design concept of FAD is based on the notion that similar time series have similar change trends. Thus, the key idea of the segmentation step is to divide a time series according to the derivative estimation values of the points. Specifically, we set a threshold ε to judge

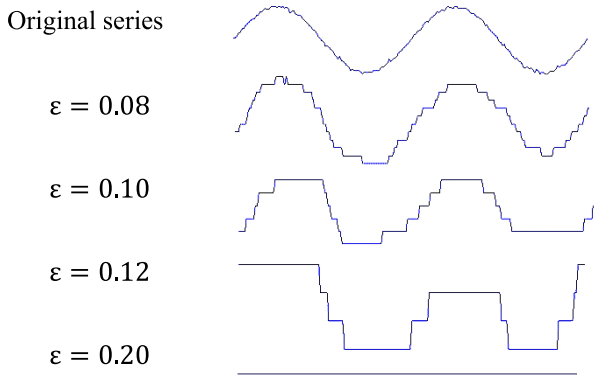

FIGURE 1. The original series is a sinusoidal signal with random noise. The series with different scales can be obtained by adjusting the value of ε. The series become flatter as the threshold value increases.

the change magnitude of the data. If the derivative estimation value of a point is less than ε, that point has little change compared to the previous one. Both points are represented by the same symbol. Otherwise, if there is an obvious difference between them, this will lead to different representation symbols of adjacent points. In this way, FAD can transform the derivative sequence into a symbolic representation sequence. Formally,

$$R_h = \begin{cases} \lambda & \hat{x}_h > \lambda \cdot \varepsilon \\ \cdots & \cdots \\ 1 & \varepsilon < \hat{x}_h \leq 2 \cdot \varepsilon \\ 0 & \left|\hat{x}_h\right| \leq \varepsilon \\ -1 & -2 \cdot \varepsilon < \hat{x}_h \leq -\varepsilon \\ \cdots & \cdots \\ -\lambda & \hat{x}_h < -\lambda \cdot \varepsilon \end{cases} \tag{3}$$

where R_h is the symbolic representation of $\hat{x}_h$ and ε is a threshold value for the change trend; the value of ε is not less than zero. The parameter λ indicates the number of symbols used to represent the time series. For instance, we can transform the original data into series composed of -1, 0, 1 or -2, -1, 0, 1, 2, etc.; this depends on the characteristics of the data. It is worth noting that λ is an integer and is not less than one. Fig. 1 illustrates the change trends of series with different threshold values.

The second step of FAD is to transform the symbolic representation sequence R into a feature series $\bar{T} = (S_1, \ldots, S_p)$, where $S_j = (R_{j1}, \ldots, R_{jk_j})$. Noting that S_j is the j-th subsequence of sequence R, R_j. is the representation symbol of S_j and k_j indicates the number of symbols in S_j. In our work, S_j can be represented a s(R_j, k_j) as well, that is, $S_j = (R_{j1}, \ldots, R_{jk_j}) = (R_j, k_j)$. Fig. 2 shows an example of a symbolic series R and the transformation process from series R to series $\bar{T}$.

It is easy to observe that most of the adjacent points in the series have the same change trends, that is, they have the same symbolic representations. To express the time series in

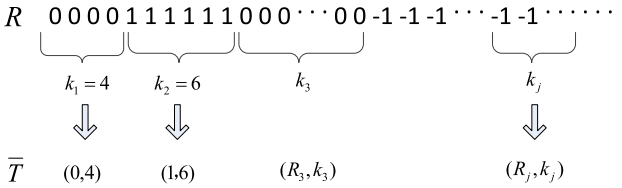

FIGURE 2. Symbolic representation of time series.

a concise way, we merge the adjacent points with the same symbols. Thus, a fragment of the time series is obtained by merging adjacent points with the same symbols. To avoid the loss of time axis information, every fragment also records the number of adjacent points that have the same symbols during the process of merging them. For example, the first subsequence of R in Fig. 2 is composed of four zero symbols and can be represented by (0, 4) in $\bar{T}$.

At this point, we can determine that a fragment is represented by two elements: one is the representation symbol and the other is the number of symbols. The representation symbol can be viewed as an approximation of the trend of the time series, and the number of symbols can reflect the trend's duration. In this way, time series can be segmented and the segment delimiters are the break points in the series. Therefore, the feature series $\bar{T} = (S_1, \ldots, S_p)$ of length p can be expressed as $((R_1, k_1), \ldots, (R_p, k_p))$ where S_j can be represented as $S_j = (R_j, k_j)$.

The symbolic sequences obtained by FAD can be combined with DTW, which is called the FAD_DTW algorithm, to measure the distances between time series. Thus, it is necessary to briefly introduce DTW in this part.

Given two time series $T_1 = (x1_1, \ldots, x1_n)$ and $T_2 = (x2_1, \ldots, x2_m)$, there is an n-by $-$m matrix with elements $d(x1_i, x2_j)$. To align the two time series, DTW needs to find the warping path between the two time series that has the minimum total cumulative distance between points. This path can be obtained by using dynamic programming to evaluate the minimum cumulative distance $g(i, j)$ of the current step:

$$g\,(i,j) = d\left(x1_i, x2_j\right) + \min \begin{cases} g(i, j-1), \\ g(i-1, j-1), \\ g(i-1, j). \end{cases} \tag{4}$$

where $d(x1_i, x2_j)$ denotes the distance between points $x1_i$ and $x2_j$; usually, the Euclidean distance is used. Since each point in the two sequences is considered, the computational complexity of DTW is $O(mn)$.

The pseudocode implementation of FAD_DTW is given in Table 1. The first step is to obtain the derivative estimates of T_1 and T_2 by Eq. (2) and transform them into symbolic sequences $R1$ and $R2$ by Eq. (3) (lines 2-9). The second step is to obtain the feature sequences $\bar{T}_1$ and $\bar{T}_2$ by merging the same symbols in $R1$ and $R2$ (lines 10-11). The above two steps are the main steps of the representation process of FAD. Finally, use the DTW algorithm to compute the distance between the two merged symbolic sequences (line 13).

TABLE 1. Implementation of FAD_DTW.

Algorithm 1 Fragment Alignment Distance with Dynamic Time Warping (FAD_DTW)

FAD_DTW(T_1, T_2, ε, λ);
Convert time series $T_1 = (x1_1, \ldots, x1_n)$ to derivative estimation series $\hat{T}_1 = (\widehat{x1}_1, .., \widehat{x1}_n)$ by Eq.(2);
Convert time series $T_2 = (x2_1, \ldots, x2_m)$ to derivative estimation series $\hat{T}_2 = (\widehat{x2}_1, \ldots, \widehat{x2}_m)$ by Eq.(2);
for $i = 1{:}\, n$
    obtain the symbolic representation $R1_i$ of $\widehat{x1}_i$ by Eq. (3); // obtain symbolic sequence $R1$
end for
for $j = 1{:}\, m$
    obtain the symbolic representation $R2_j$ of $\widehat{x2}_j$ by Eq. (3); // obtain symbolic sequence $R2$
end for
Merge the same symbols in $R1$ and obtain feature sequence $\bar{T}_1 = ((R1_1, k1_1), \ldots, (R1_{p1}, k1_{p1}))$
Merge the same symbols in $R2$ and obtain feature sequence $\bar{T}_2 = ((R2_1, k2_1), \ldots, (R2_{p2}, k2_{p2}))$
// p1 and p2 are the numbers of fragments in $\bar{T}_1$ and $\bar{T}_2$, respectively
$D(T_1, T_2) = \boldsymbol{DTW}(\bar{T}_1[1], \bar{T}_2[1])$
// $\bar{T}_1[1]$ and $\bar{T}_2[1]$ represent symbolic sequences $(R1_1, \ldots, R1_{p1})$ and $(R2_1, \ldots, R2_{p2})$, respectively
Return $D(T_1, T_2)$;

D. SIMILARITY MEASURE

After obtaining feature series $\bar{T}$, we now present a new similarity measure to further speed up similarity detection. We transform the comparison between the points of two time series into comparison between fragments of them. As noted in [38], researchers using DTW constrain the warping path in a global sense by limiting how far it can stray from the diagonal. This allows us to align two fragment series diagonally. Moreover, the design concept of FAD is that similar time series have similar change trends. Thus, FAD needs to find a warping path that can diagonally map the similar change trends in two series. "Diagonally" does not mean the warping path completely lines up with the diagonal; it tends to map two similar fragments around the diagonal. For instance, Fig. 3A shows two series with similar change trends. To map the two series, the ideal warping path should be close to the diagonal of the matrix. Fig. 3B shows two series with different lengths. Since only half of the two series are similar and they can be mapped with each other, the ideal warping path ought to be close to the diagonal of half of the matrix.

In the following part, we propose a new computational method that can find such a warping path in linear time. To align the two series, three cases need to be considered satisfied in the course of similarity measurement.

1) The mapped fragments in $\bar{T}_1$ and $\bar{T}_2$ have different symbols, which means the trends of the two subsequences (fragments) are different. In this case, we define the distance between them as Eq. (5).

$$D\left(S1_i, S2_j\right) = 1, \quad if\ R1_i \neq R2_j \tag{5}$$

where $S1_i$ and $S2_j$ are the subsequences of $\bar{T}_1$ and $\bar{T}_2$, respectively, and $R1_i$ and $R2_j$ are the corresponding symbols of $S1_i$ and $S2_j$.

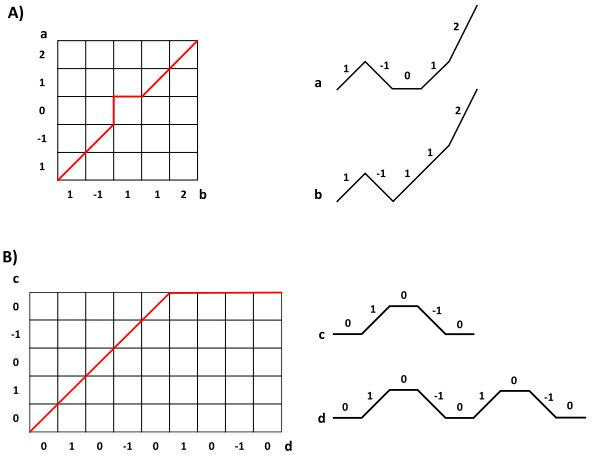

FIGURE 3. The numbers on the line denote the symbols of different fragments. The red line denotes the ideal warping path of FAD. A) Two fragments (a and b) with the same length. B) Two fragments (c and d) with different lengths.

2) The mapped fragments in $\bar{T}_1$ and $\bar{T}_2$ have same symbols, which indicates they have similar change trends. Hence, the distance between them mainly depends on the difference in their lengths. We compute the distance between them by Eq. (6).

$$D\left(S1_i, S2_j\right) = \gamma \times \left(\frac{\max\left\{k1_i, k2_j\right\}}{\min\left\{k1_i, k2_j\right\}} - 1\right), \quad if\ R1_i = R2_j \tag{6}$$

where $k1_i$ and $k2_j$ are the numbers of points for $S1_i$ and $S2_j$, respectively, and γ is an adjustable parameter to change the distance ratio of same symbols to different symbols. Intuitively, the distance between the same symbol fragments must be less than the distance between different ones. Thus, in case 2), we have $0 \leq D(S1_i, S2_j) < 1$ and $\gamma \in [0, 1)$.

TABLE 2. Fragment alignment distance.

Algorithm 2 Fragment Alignment Distance (FAD)

1. **FAD**(T_1, T_2, γ, ε, λ);
2. Convert time series T_1 to derivative estimation series $\hat{T}_1$ by Eq. (2);
3. Convert time series T_2 to derivative estimation series $\hat{T}_2$ by Eq. (2);
4. Convert $\hat{T}_1$ to feature series $\bar{T}_1 = ((R1_1, k1_1), \ldots, (R1_{p1}, k1_{p1}))$ by Eq. (3);
5. Convert $\hat{T}_2$ to feature series $\bar{T}_2 = ((R2_1, k2_1), \ldots, (R2_{p2}, k2_{p2}))$ by Eq. (3);
6. $i = 1$; $j = 1$; $D(\bar{T}_1, \bar{T}_2) = 0$;
7. **While** $i \le p1$ and $j \le p2$
8. **If** $R1_i == R2_j$
9. $D(\bar{T}_1, \bar{T}_2) = D(\bar{T}_1, \bar{T}_2) + \gamma \times \left(max\{k1_i, k2_j\}/min\{k1_i, k2_j\} - 1\right)$;
10. $i = i + 1$; $j = j + 1$;
11. **else**
12. $D(\bar{T}_1, \bar{T}_2) = D(\bar{T}_1, \bar{T}_2) + 1$;
13. **If** $p1 - i > p2 - j$
14. $i = i + 1$;
15. **else**
16. $j = j + 1$;
17. **end if**;
18. **end if**;
19. **end while**;
20. $D(\bar{T}_1, \bar{T}_2) = D(\bar{T}_1, \bar{T}_2) + p1 - i + 1 + p2 - j + 1$;
21. **return** $D(\bar{T}_1, \bar{T}_2)$;

3) Due to the time series' unequal lengths and the time warping awareness of FAD, some fragments usually remain in one of the series with no fragments in the other series for mapping. Such fragments can be viewed as not being similar to any fragments, which corresponds to case 1). We define the distance as Eq. (7).

$$D(-, S_i) = 1 \tag{7}$$

In general, we can define the distance between two fragments as Eq. (8).

$$D(S1_i, S2_j) = \begin{cases} \gamma \times \left(\dfrac{max\{k1_i, k2_j\}}{min\{k1_i, k2_j\}} - 1\right) & R1_i = R2_j \text{ and } S2_j \text{ exists} \\ 1 & otherwise \end{cases} \tag{8}$$

Given two feature series $\bar{T}_1 = (S1_1, \ldots, S1_{p1})$ and $\bar{T}_2 = (S2_1, \ldots, S2_{p2})$ with lengths of $p1$ and $p2$, respectively, the distance between them can be defined as Eq. (9).

$$D(\bar{T}_1, \bar{T}_2) = \sum D(S1_i, S2_j) \tag{9}$$

Table 2 shows the pseudocode implementation of FAD in greater detail. The algorithm starts off by computing the derivative estimates of time series T_1 and T_2 (lines 2 and 3). Then, the derivative estimation series are transformed into fragment series (lines 4 and 5). Assume i and j are the subscripts of the current fragments in $\bar{T}_1$ and $\bar{T}_2$, respectively, and initialize the distance between the two series as zero (line 6). For mapping fragments with the same symbols, the distance between them can be computed by Eq. (6), and the two subscript values should be both increased because the two fragments are correctly mapped (lines 8-10). Otherwise, if the mapping fragments have different symbols, it may be incorrectly mapped. Thus, one of the fragments should be aligned to the next fragment in the other series. In this situation, only one series' subscript needs to be increased, and we update it according to the left lengths of the two series. To balance the left lengths of the two series, we increase the longer one's subscript. As a punishment, the distance between them is calculated by Eq. (7) (lines 12-17). Then, in case 3), the distance of the left series can be computed by $p1 - i + 1 + p2 - j + 1$. Finally, the total distance is calculated in line 20.

According to the above analysis, FAD can map the fragments with similar change trends and restrict the warping path to close to the diagonal by handling the unmapped fragments based on the left lengths of the two series (lines 13-17). That is, FAD calculates the distance between two series by finding a warping path that can diagonally map their similar change trends.

It is easy to show that FAD satisfies the triangle inequality. We prove in the Appendix that FAD is a metric. Therefore, the existing indexing structures proposed for metrics are available for FAD.

E. COMPUTATIONAL COMPLEXITY ANALYSIS

In this subsection, the computational complexity of FAD is analyzed. First, we consider the complexity of the representation stage. According to lines 2-11 in Table 1, the time complexity of representation is

$$2\cdot O(n)+2\cdot O(m)+O(\lambda n)+O(\lambda m)\approx O(\lambda n)+O(\lambda m) \quad (10)$$

where n and m are the lengths of time series T_1 and T_2, respectively. Second, we analyze the time complexity of the similarity measurement stage. According to lines 7-19 in Table 2, the time complexity of calculating the similarity between two symbolic sequences is $O(min\{p1, p2\})$. Therefore, the overall computational complexity of FAD is

$$O(\lambda n)+O(\lambda m)+O(min\{p1, p2\})\approx O(max\{\lambda n, \lambda m\}). \quad (11)$$

Usually, p1 and p2 are far less than n and m. To conclude, the computational complexity of FAD is $O(max\{\lambda n, \lambda m\})$, which is much less than those of other similarity measures. The implementation of FAD is quite easy.

IV. EXPERIMENTAL METHODOLOGIES

We designed a validation experiment to evaluate the ability of FAD in supporting efficient and effective similarity search and detection within classification and clustering frameworks. We compared FAD with state-of-the-art approaches for time series representation models and similarity measures, including PAA, SAX, and DSA as time series models and LCSS, EDR, ERP, and DDTW as similarity measures. Since representation models cannot be directly used to measure the distance between time series, we chose to employ DTW over the segments computed by each particular representation model.

A. ALGORITHMS

Since the objective of our work is to assess the ability of FAD in time series data mining tasks, we employed standard classification and clustering frameworks for assessment, which include K-means clustering, agglomerative hierarchical clustering [42] and nearest-neighbor classification [43].

1) K-MEANS CLUSTERING

In our work, we employed the popular K-means algorithm, which is characterized by simplicity and low computational requirements. For the K-means algorithm, one needs to specify the number of output clusters; for simplicity, we adopted data sets for which relevant classifications are available. Thus, we set the number of output clusters equal to the actual number of classes in each clustering evaluation. Moreover, since the initial cluster centroids greatly affect the performance of the K-means algorithm, we not only randomly selected the initial cluster centroids but also executed multiple runs of the K-means algorithm to obtain relatively reliable results.

2) HIERARCHICAL CLUSTERING

The hierarchical clustering enables us to evaluate the competing approaches in a clustering framework that does not depend on the cluster initialization. Hierarchical clustering is an approach to cluster analysis that seeks to build a hierarchy of clusters. It generally has two types of strategies: one is agglomerative and the other is divisive. The agglomerative strategy is a ''bottom-up'' approach in which each observation starts off in its own cluster and then pairs of clusters are merged to construct the hierarchy. The divisive strategy is a ''top-down'' approach, in which all points start as one cluster and the cluster is recursively split as one moves down the hierarchy. Since agglomerative clustering has lower complexity compared to divisive clustering, we employed the Unweighted Pair Group Method using arithmetic Averages (UPGMA) algorithm, which is an agglomerative clustering algorithm that uses group-average linkage to calculate the distance between two clusters [42].

3) ONE NEAREST NEIGHBOR (1-NN) CLASSIFICATION

Nearest-neighbor classification is widely known to be a straightforward and effective method to assess the performances of various algorithms [43]. The one-nearest-neighbor classification classifies each data instance according to the most similar instance to it. As the most basic instance-based classification approach, the 1-NN classification algorithm is essential in our evaluation work. Compared with K-NN, 1-NN classification has the advantages of having no parameters and allowing comparisons between methods.

B. ASSESSMENT CRITERIA

To assess the effectiveness of the competing methods, we directly compared the results of classification and clustering with the inherent distribution of the data, which is available for our selected data sets.

F-measure or F-score (F) [44] is the most widely used external criterion. It is defined as the harmonic mean of the information retrieval concepts of precision (P) and recall (R). Formally,

$$F = \frac{2 \times P \times R}{P + R}. \quad (12)$$

Given a set $\emptyset$ of time series, assume $\Gamma = \{\Gamma_1, \ldots, \Gamma_H\}$ is the inherent distribution of the series in $\emptyset$, and the output distribution of a classification or clustering algorithm is $C = \{C_1, \ldots, C_K\}$. The precision of C_j with respect to Γ_i is the proportion of the series in C_j that has been correctly classified; formally $P_{ij} = \left|C_j \bigcap \Gamma_i\right| / \left|C_j\right|$. The recall of C_j with respect to Γ_i is the proportion of the series in Γ_i that has been correctly classified; formally, $R_{ij} = \left|C_j \bigcap \Gamma_i\right| / |\Gamma_i|$.

TABLE 3. Data sets.

No.	Data set	Classes	Size of training/testing set	Time series length	Type
1	OSULeaf	6	200/242	427	Shape
2	FaceFour	4	24/88	350	Shape
3	Lighting7	7	70/73	319	Real
4	ECG200	2	100/100	96	Real
5	Beef	5	30/30	470	Real
6	OliveOil	4	30/30	570	Real
7	Wine	2	57/54	234	Real
8	ShapeletSim	2	20/180	500	Synthetic
9	BirdChicken	2	20/20	512	Shape
10	Herring	2	64/64	512	Shape
11	Earthquakes	2	139/322	512	Real
12	BeetleFly	2	20/20	512	Shape
13	Plane	7	105/105	144	Shape

In the classification frameworks, since $H = K$, the overall precision and recall are

$$P = \frac{1}{H}\sum_{i=1}^{H} P_{ii}, \quad R = \frac{1}{H}\sum_{i=1}^{H} R_{ii}. \tag{13}$$

whereas in the case of clustering, the overall precision and recall are defined as

$$P = \frac{1}{H}\sum_{i=1}^{H} P_{i}, \quad R = \frac{1}{H}\sum_{i=1}^{H} R_{i}, \tag{14}$$

where $P_i = P_{ij^*}$, $R_i = R_{ij^*}$, and $j^* \in argmax_{j=1\ldots K}\{P_{ij}, R_{ij}\}$.

It is obvious that the higher the F-measure, the better the performance of the method.

C. SETUP OF THE PARAMETERS FOR THE COMPETING METHODS

Most state-of-the-art approaches require one or more parameters to be set. For some approaches, including LCSS, EDR, and ERP, typical settings have been suggested in their respective works. For LCSS and EDR, it is suggested that the matching thresholds be equal to $(max\sigma\ (T_i))/4$ and $min\sigma\ (T_i)$, respectively, where $\sigma(T_i)$ denotes the standard deviation over the points in the i-th time series of all the series in a data set. The constant gap for ERP is set to zero. Such parameter settings proved to be good enough to enable the corresponding methods to achieve their best performances in terms of accuracy. For the other methods, to make a comparative assessment possible in terms of accuracy and efficiency, we prepared a set of parameter values to evaluate them. Since the alphabet size of SAX is usually no more than ten, the parameter value of alphabet size can be set as (2, 3, 4, 5, 6, 7, 8, 9, 10), and we varied the number of symbols from 2 to one-fifth of the series' length using an increment of two symbols in each step. Analogously, we varied the window length in PAA from 2 points to one-fifth of the series length in the same way. We measured the corresponding classification and clustering F-measure scores and ultimately selected the settings that corresponded to the best performance.

V. RESULTS

A. DATA DESCRIPTION

The 13 time series data sets employed in this experiment are collected from the UCR Time Series Classification /Clustering Homepage [45]. The data resource is provided by Keogh et al., which is derived from a variety of applications. Detailed information about the data is given in Table 3. The type labels require a brief explanation. Some data sets are real, which simply means they were recorded as natural time series from some physical process, for instance, the earthquake signal from sensor reading. Some data sets are shapes: these are one-dimensional time series that were extracted by processing some two-dimensional shapes, such as leaf profiles or the silhouettes of planes. Finally, one of the data sets is synthetic; it was created by researchers to test some property of a time series algorithm. We note that none of the current authors have created any of the synthetic data sets used here.

B. EFFECTIVENESS EVALUATION

In this part, we mainly focus on assessing the ability of FAD and other state-of-the-art methods in supporting time series clustering and classification. The performances of the parametric methods were measured using their best settings.

We directly employed the data sets obtained from the UCR Time Series Classification/Clustering Homepage [45] without any preprocessing steps because the data sets have already been normalized. In general, it is necessary to normalize the data before performing experiments.

1) TUNING PARAMETERS OF FAD

According to Section 3, the ratio parameter γ of FAD is non-negative and less than one. We found that when the value

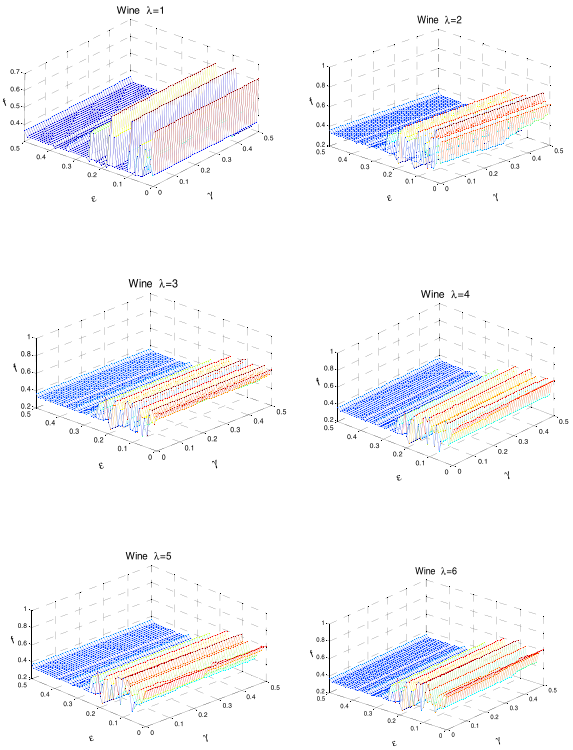

FIGURE 4. The above pictures show the performance surfaces of different λ values in data set Wine. Note that the parameter *f* in the pictures represents the F-score of the classification results. For convenience of presentation, the changes of F-score values are drawn in different colors.

of parameter γ becomes larger than 0.5, the performance of our method does not greatly improve. Thus, for the sake of rigorousness, we varied the value of γ from 0 to 0.5 by 0.01 each step. If the threshold value is too large, the feature series $\bar{T}$ will lose details about the original data. This will degrade the performance of FAD. Thus, we suggest that the maximum value of the threshold not exceed half of the standard deviation. Since the data sets have been normalized and the standard deviation is equal to one, varied the threshold parameter ε from 0 to 0.5 by 0.01 each step as well. We tentatively set the value of parameter λ to (1, 2, 3, 4, 5, 6), successively. To study the influence of different parameter settings on the performance of FAD, we selected four data sets and classified them using the 1-NN algorithm. The results for data set Wine are illustrated in Fig. 4 and the best performances on all data sets corresponding to different λ values are listed in Table 4.

It can be inferred from Fig. 4 that the performance of FAD is more sensitive to the value of ε, while parameter γ has little effect on FAD. Additionally, the performance surfaces for different λ values are similar to each other in Fig. 4; this phenomenon is quite common among different data sets. Table 4 gives us more information about the relationship between the performance of FAD and the value of parameter λ. Usually, a small value of λ is sufficient to achieve high performance.

TABLE 4. Classification results (F-score) for different λ values.

	$\lambda=1$	$\lambda=2$	$\lambda=3$	$\lambda=4$	$\lambda=5$	$\lambda=6$
ECG200	0.817	0.817	0.798	0.805	0.805	0.805
OliveOil	0.760	0.766	0.684	0.782	0.733	0.693
Wine	0.629	0.709	0.645	0.678	0.653	0.653
Herring	0.646	0.646	0.646	0.646	0.646	0.646

According to the above analysis, we adjusted the set of values of the threshold ε, ranging from 0 to 0.2, by 0.01 each step, and increasing the value of parameter γ from 0 to 0.5 by 0.05 each step in the following experiments. We set the value of parameter λ to (1, 2, 3) in the following studies.

2) ACCURACY IN TIME SERIES CLASSIFICATION

We evaluated the performances of FAD and FAD_DTW, along with those of other methods, namely PAA, SAX, DSA, DDTW, ERP, LCSS and EDR, using the 1-NN classification framework. Table 5 shows the best results achieved by various approaches.

To provide a more intuitive illustration of the performances of different approaches, we sorted the classification results in Table 6. If multiple approaches have the same F-scores, we set their rankings as the average of their corresponding rankings. For example, EPR and EDR have same classification results on the ECG200 data sets and their rankings are supposed to be 2 and 3, respectively; thus, we set their rankings as $(2+3)/2 = 2.5$.

According to Table 6, FAD_DTW attained the highest average ranking in the classification tasks, and the rankings of FAD_DTW are quite stable compared with those of the other methods, except for DSA_DTW, which had worse performance than FAD_DTW. DSA segments time series by setting the standard deviation as the threshold, which introduces some uncertainty about the segments. For instance, if the previous segment is long, the standard deviation might be large. This will lead to the omission of the next burst signal. In this case, the segments obtained by DSA cannot accurately express the original series. FAD and LCSS have the same average rankings, while FAD is more stable than LCSS. SAX_DTW outperforms DSA_DTW, DDTW and PAA_DTW. The threshold of SAX is fixed and it has better ability to overcome the noise than DSA, DDTW and PAA. It should be noted that the methods without derivative estimation (LCSS, EDR, ERP, SAX, and PAA) have difficulty handling data points that have same value but different trends, which degrades their performances. Moreover, compared with FAD, DDTW is more sensitive to noisy data. Since the segmentation process of FAD and FAD_DTW is based on the derivative estimation of the original series and the threshold is fixed, it overcomes the shortcomings of other methods. The experimental results show the good performance of our methods.

TABLE 5. Summary of results (F-score) for 1-NN classification.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	0.669	0.666	0.799	0.738	**0.877**	0.631	0.813	0.583	0.687
FaceFour	0.605	0.827	0.451	0.767	0.462	0.848	**0.915**	0.761	0.574
Lighting7	0.438	**0.845**	0.583	0.458	0.480	0.810	0.751	0.557	0.565
ECG200	0.857	0.777	0.729	0.795	0.823	0.879	**0.913**	0.879	0.752
Beef	0.476	0.699	0.554	**0.752**	0.697	0.607	0.719	0.256	0.681
OliveOil	0.667	0.643	0.467	**0.769**	0.745	0.750	0.071	0.071	0.766
Wine	0.653	0.629	0.574	0.672	0.537	0.648	0.333	0.333	**0.709**
ShapeletSim	0.646	0.684	0.507	0.853	0.492	0.858	**0.947**	0.771	0.832
BirdChicken	0.828	**0.908**	0.677	0.852	0.752	0.550	**0.908**	0.651	0.9
Herring	0.584	**0.720**	0.579	0.646	0.366	0.590	0.532	0.373	0.646
Earthquakes	0.587	0.629	0.538	0.613	0.521	0.533	0.536	0.586	**0.653**
BeetleFly	0.952	0.852	0.789	0.908	**1**	0.719	0.867	0.789	0.852
Plane	**1**	0.983	0.962	0.990	**1**	**1**	0.991	0.820	**1**

TABLE 6. Rankings of different methods for 1-NN classification results.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	6	7	3	4	1	8	2	9	5
FaceFour	6	3	9	4	8	2	1	5	7
Lighting7	9	1	4	8	7	2	3	6	5
ECG200	4	7	9	6	5	2.5	1	2.5	8
Beef	8	3	7	1	4	6	2	9	5
OliveOil	5	6	7	1	4	3	8.5	8.5	2
Wine	3	5	6	2	7	4	8.5	8.5	1
ShapeletSim	7	6	8	3	9	2	1	5	4
BirdChicken	5	1.5	7	4	6	9	1.5	8	3
Herring	5	1	6	2.5	9	4	7	8	2.5
Earthquakes	4	2	6	3	9	8	7	5	1
BeetleFly	2	5.5	7.5	3	1	9	4	7.5	5.5
Plane	2.5	7	8	6	2.5	2.5	5	9	2.5
Average values	5.1 ± 2.0	4.2 ± 2.3	6.7 ± 1.7	$\mathbf{3.7 \pm 2.0}$	5.6 ± 2.8	4.8 ± 2.7	4.0 ± 2.8	7.0 ± 2.0	4.0 ± 2.1

3) ACCURACY IN TIME SERIES CLUSTERING

In this part, we assessed FAD and other approaches in two clustering frameworks, namely, UPGMA and K-means. Instead of splitting each data set for training and testing as in the classification frameworks, we used the whole data sets to evaluate the performance of each method.

Table 7 presents the results (F-score) of the competing methods for UPGMA clustering. The rankings of all approaches are reported in Table 8.

According to Table 7 and Table 8, FAD is the top-ranked method on average over all the datasets. FAD_DTW outperforms all methods except FAD. Another symbolic method, namely, SAX_DTW, also has good performance. As a time series representation model, FAD is more accurate than SAX, PAA and DSA, which means FAD can approximate the original series very well. As a similarity measure, FAD has time-warping awareness and is more accurate than DDTW, LCSS, EDR and ERP. Unlike in the classification task, LCSS has greatly decreased performance in UPGMA clustering.

Table 9 reports the results attained in the K-means clustering framework. Table 10 shows the rankings of the results. Since the uncertainty of initialization greatly affects the results of the K-means algorithm, we performed multiple runs and set the average value of the runs as the final result.

According to Tables 9 and 10, FAD_DTW has the best performance among all the competing methods. FAD has

TABLE 7. Summary of results (F-score) for UPGMA clustering.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	0.281	0.300	0.293	**0.322**	0.291	0.266	0.263	0.267	0.300
FaceFour	0.430	0.404	0.390	**0.480**	0.389	0.367	0.377	0.380	0.406
Lighting7	0.297	0.273	0.287	**0.376**	0.303	0.257	0.259	0.251	0.302
ECG200	0.698	0.698	0.685	0.698	0.688	0.690	0.698	0.698	**0.699**
Beef	0.362	0.329	0.295	0.393	0.3	0.303	0.307	0.329	**0.403**
OliveOil	0.491	0.436	0.409	0.484	0.432	0.468	0.460	0.436	**0.520**
Wine	0.660	**0.665**	0.660	**0.665**	0.660	0.660	0.660	**0.665**	**0.665**
ShapeletSim	0.666	0.666	0.666	**0.955**	0.630	0.647	0.666	0.652	0.810
BirdChicken	0.619	**0.661**	0.655	**0.661**	0.655	0.596	0.655	**0.661**	**0.661**
Herring	**0.678**	**0.678**	0.677	**0.678**	**0.678**	0.627	0.677	0.670	**0.678**
Earthquakes	0.775	0.777	0.743	0.754	**0.782**	0.775	0.646	0.771	0.777
BeetleFly	0.673	0.661	0.655	**0.848**	0.655	0.574	0.661	0.661	0.665
Plane	0.243	0.249	0.235	**0.306**	0.225	0.229	0.229	0.233	0.270

TABLE 8. Rankings of different methods for the UPGMA clustering task.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	6	2.5	4	**1**	5	8	9	7	2.5
FaceFour	2	4	5	**1**	6	9	8	7	3
Lighting7	4	6	5	**1**	2	8	7	9	3
ECG200	4	4	9	4	8	7	4	4	**1**
Beef	3	4.5	9	2	8	7	6	4.5	**1**
OliveOil	2	6.5	9	3	8	4	5	6.5	**1**
Wine	7	**2.5**	7	**2.5**	7	7	7	**2.5**	**2.5**
ShapeletSim	4.5	4.5	4.5	**1**	9	8	4.5	7	2
BirdChicken	8	**2.5**	6	**2.5**	6	9	6	**2.5**	**2.5**
Herring	**3**	**3**	6.5	**3**	**3**	9	6.5	8	**3**
Earthquakes	4.5	2.5	8	7	1	4.5	9	6	2.5
BeetleFly	2	5	7.5	**1**	7.5	9	5	5	3
Plane	4	3	5	**1**	9	7.5	7.5	6	2
Average values	4.2 ± 1.8	3.9 ± 1.3	6.6 ± 1.7	2.3 ± 1.7	6.1 ± 2.5	7.5 ± 1.6	6.5 ± 1.6	5.8 ± 1.9	$\mathbf{2.2 \pm 0.8}$

a relatively high ranking compared with other methods. PAA_DTW attains better accuracy in K-means clustering than in the UPGMA framework. Although SAX_DTW performs worse in K-means clustering than in UPGMA clustering, it is still more accurate than LCSS, DDTW, EDR and ERP.

Moreover, compared with classification frameworks, both FAD and FAD_DTW show greatly improved accuracy in clustering tasks.

C. EFFICIENCY EVALUATION

We evaluated the time performances of FAD and the other approaches in achieving the tasks of modeling and clustering time series. All experiments were conducted on a platform with an Intel E5-2620 CPU with 64.0 GB memory and running Microsoft Windows XP.

1) PERFORMANCE IN TIME SERIES MODELING

Table 11 summarizes the time performance results (in milliseconds) of the competing methods in modeling time series using their best settings. The corresponding compression ratios of different representation models are also shown in the table; for example, C_PAA represents the compression ratio of the PAA algorithm.

According to Table 11, FAD is the fastest representation model among all the competing methods.

TABLE 9. Summary of results (F-score) for K-means clustering.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	0.271	0.299	0.259	**0.367**	0.262	0.254	0.233	0.251	0.306
FaceFour	0.403	0.406	0.366	**0.498**	0.465	0.426	0.356	0.373	0.447
Lighting7	0.290	0.275	0.372	**0.384**	0.308	0.260	0.252	0.232	0.309
ECG200	0.665	0.699	0.519	**0.717**	0.578	0.599	0.619	0.616	0.699
Beef	0.409	0.333	0.363	0.396	0.366	0.377	0.339	0.339	**0.468**
OliveOil	0.483	0.449	0.426	**0.556**	0.468	0.391	0.446	0.449	0.538
Wine	0.660	0.667	0.550	**0.785**	0.596	0.641	0.665	0.667	0.684
ShapeletSim	0.632	0.667	0.552	**0.826**	0.582	0.631	0.526	0.526	0.805
BirdChicken	0.625	0.667	0.549	**0.774**	0.549	0.565	0.596	0.565	0.700
Herring	0.624	**0.679**	0.582	**0.679**	0.548	0.546	0.598	0.604	**0.679**
Earthquakes	0.775	0.776	0.568	0.732	0.655	0.635	0.613	0.632	0.775
BeetleFly	**0.875**	0.667	0.615	0.723	0.583	0.625	0.591	0.573	0.699
Plane	0.261	0.25	0.242	**0.301**	0.261	0.243	0.228	0.231	0.288

TABLE 10. Rankings of different methods for K-means clustering results.

	PAA_DTW	SAX_DTW	DSA_DTW	FAD_DTW	DDTW	ERP	LCSS	EDR	FAD
OSULeaf	4	3	6	**1**	5	7	9	8	2
FaceFour	6	5	8	**1**	2	4	9	7	3
Lighting7	5	6	2	**1**	4	7	8	9	3
ECG200	4	2.5	9	**1**	8	7	5	6	2.5
Beef	2	9	5	3	4	6	7.5	7.5	**1**
OliveOil	3	5.5	8	**1**	4	9	7	5.5	2
Wine	6	3.5	9	**1**	8	7	5	3.5	2
ShapeletSim	4	3	7	**1**	6	5	8.5	8.5	2
BirdChicken	4	3	8.5	**1**	8.5	6.5	5	6.5	2
Herring	4	**2**	7	**2**	8	9	6	5	**2**
Earthquakes	2.5	1	9	4	5	6	8	7	2.5
BeetleFly	**1**	4	6	2	8	5	7	9	3
Plane	3	5	7	**1**	4	6	9	8	2
Average values	3.7 ± 1.4	4.0 ± 2.0	7.0 ± 1.9	$\mathbf{1.5 \pm 0.9}$	5.7 ± 2.1	6.5 ± 1.4	7.2 ± 1.5	7.0 ± 1.6	$\mathbf{2.2 \pm 0.5}$

SAX and PAA are also very fast. All four representation models can be implemented in linear time. In general, simpler models result in higher efficiency. It is necessary to examine the impacts on the time series dimensionality of various methods. Table 11 demonstrates that FAD can achieve 74.5% compression of time series length on average, with a maximum compression ratio of 99.8% in the BirdChicken data set. SAX has the highest average compression ratio of 87.6%. This suggests that FAD is able to compactly represent the original data.

2) PERFORMANCE IN TIME SERIES CLUSTERING

After comparing the time performances of various representation methods, we also assessed the time performances of different similarity measures for the K-means clustering task. For the sake of simplicity, we conducted experiments on data set Plane, which contains 210 pieces of time series. Fig. 5 reports the time performances (seconds) of the competing similarity measures with different data sizes. Denoting the run time of the methods is t, the time axis in Fig. 5 is $log_{10}(t)$.

According to Fig. 5, FAD is the fastest method among the competing methods. Moreover, FAD_DTW, SAX_DTW, PAA_DTW and LCSS all have relatively high performances in run time, while ERP, DSA_DTW, and EDR are obviously slower than the other methods, except for DDTW, which is revealed to be the slowest one. Due to the high complexity of DTW, the methods based on it show inferior performances compared with FAD. However, this indicates that the FAD, SAX, PAA and DSA representation methods can greatly

TABLE 11. Summary of time performances (milliseconds) and compression ratios in time series modeling.

	PAA (ms)	SAX (ms)	DSA (ms)	FAD (ms)	C_PAA (%)	C_SAX (%)	C_DSA (%)	C_FAD (%)
OSULeaf	2.9	2.5	24.2	2.0	74.9	82.2	62.5	59.0
FaceFour	1.1	1.7	19.5	1.3	92.9	82.3	63.1	55.4
Lighting7	3.6	1.5	19.7	1.0	49.8	85.6	71.5	77.6
ECG200	0.8	1.2	5.9	0.8	94.8	91.7	59.4	64.1
Beef	5.4	1.7	24.4	1.4	50.0	86.8	59.8	70.0
OliveOil	6.7	3.8	28.4	1.6	50.0	82.1	55.6	84.6
Wine	0.8	1.4	14.1	0.9	96.6	83.8	53.8	88.2
ShapeletSim	5.2	1.7	28.2	1.6	50.0	98.8	61.8	51.3
BirdChicken	1.4	7.2	28.4	1.1	93.2	84.0	61.9	99.8
Herring	0.9	1.4	27.3	1.2	98.4	98.4	59.6	99.6
Earthquakes	6.0	2.8	34.7	2.5	50.0	81.6	90.0	63.8
BeetleFly	0.8	1.3	27.7	0.8	98.6	98.0	61.7	81.6
Plane	2.0	1.3	9.4	0.9	50.0	83.3	61.8	73.3
Average values	2.9 ± 2.1	2.3 ± 1.6	22.5 ± 8.0	$\mathbf{1.3 \pm 0.5}$	73.0 ± 22.1	87.6 ± 6.5	63.3 ± 8.7	74.5 ± 15.2

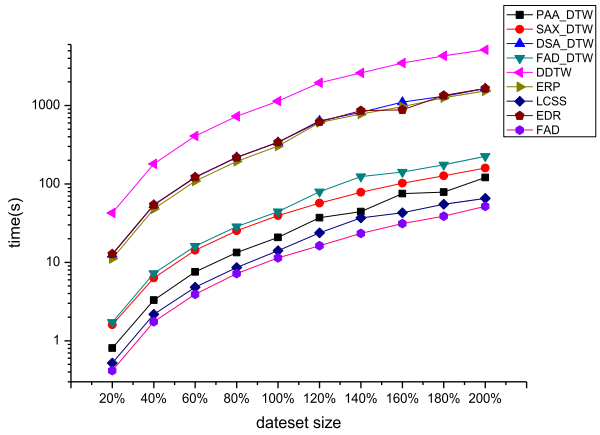

FIGURE 5. Time performance in the K-means clustering task.

improve the time performance of basic DTW. According to the analysis in Section 3, FAD can be carried out in linear time. Thus, FAD will demonstrate great superiority in terms of run time as the size of the data increases.

D. SUMMARY OF RESULTS AND DISCUSSION

To assess the capabilities of FAD and FAD_DTW, we compared them with state-of-the-art methods for supporting similarity detection with classification and clustering frameworks. To maintain the objectivity of the experimental results, we abandoned the data preprocessing step and all of our experiments were conducted using standard classification and clustering algorithms. Since the focus of our work is to devise a fast and accurate similarity measure of time series, the experiment section mainly can be divided into two parts: effectiveness evaluation and efficiency evaluation. We can summarize the main conclusions of our works as follows:

- The time series representation model FAD can not only quickly extract trend features of time series but also provide an accurate approximation of the original data. No matter the classification or clustering task, FAD is the fastest and most accurate method among the representation models.
- FAD is the fastest similarity measure compared to the competing methods. FAD has time-warp awareness and its computational complexity is linear with time series length, which is a great advantage over other similarity measures.
- FAD_DTW and FAD are more accurate than other methods in both classification and clustering tasks. Specially, FAD_DTW is more accurate than FAD in the 1-NN classification and K-means clustering tasks. However, FAD performs better in the UPGMA clustering task.

VI. CONCLUSIONS

In this paper, we proposed the FAD and FAD_DTW algorithms. FAD is composed of a novel representation model and a corresponding similarity measure to support fast and accurate similarity detection in time series data mining tasks. It is able to transform time series into compact yet feature-rich symbolic sequences by extracting trend information of data and diagonally mapping the similar change trends between series. FAD is devised based on the notion that similar time series have similar trends. Hence, it is a method based on the

trend of the data. FAD_DTW applies Dynamic Time Warping on the representation model of FAD; thus, it can extract the same features of time series as FAD. FAD_DTW has high accuracy in classification and clustering tasks. Moreover, as a symbolic representation method, FAD is drastically faster than SAX. The computational complexity of FAD is linear with the length of the time series, which is much faster than other similarity measures. Additionally, FAD and FAD_DTW both have better performance in clustering tasks than in classification.

APPENDIX

In the paper, we claimed that FAD is a metric. We will prove it in this section. In general, a metric must satisfy four conditions: identity of indiscernibles, non-negativity, symmetry and triangle inequality. The formal definitions are as follows:

Conditions: 1. $D(\bar{T}_1, \bar{T}_1) = 0$.
2. $D(\bar{T}_1, \bar{T}_2) \geq 0$.
3. $D(\bar{T}_1, \bar{T}_2) = D(\bar{T}_2, \bar{T}_1)$.
4. $D(\bar{T}_1, \bar{T}_3) \leq D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3)$.

Proof:

Condition 1: Assume $\bar{T}_1 = (S1_1, \ldots, S1_{p1})$. Then, $D(\bar{T}_1, \bar{T}_1) = \sum_i^{p1} \gamma \times (k1_i/k1_i - 1) = 0$. Thus, FAD satisfies condition 1.

Condition 2: If $\bar{T}_1 = \bar{T}_2$, then according to condition 1, $D(\bar{T}_1, \bar{T}_1) = 0$; otherwise, according to the definition of FAD, it is easy to find that $D(\bar{T}_1, \bar{T}_2) > 0$. Thus, FAD satisfies condition 2.

Condition 3: According to the procedure of FAD, it is obvious that FAD is not sensitive to the relative order of two series. Readers can prove this condition themselves. Indeed, FAD satisfies condition 3.

Condition 4: We prove the triangle inequality of FAD with mathematical induction.

Suppose we are given three time series: $\bar{T}_1 = ((a, k_1), (b, k_2), (c, k_3))$, $\bar{T}_2 = ((a, k_1), (d, k_2), (c, k_3))$, and $\bar{T}_3 = ((a, k_1), (e, k_2), (c, k_3))$.

① *Basis:* According to the definition of FAD,

$$D(\bar{T}_1, \bar{T}_2) = \gamma \times \left(\frac{max\{k_1, k_1\}}{min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_3, k_3\}}{min\{k_3, k_3\}} - 1\right) = 2,$$

$$D(\bar{T}_1, \bar{T}_2) = \gamma \times \left(\frac{max\{k_1, k_1\}}{min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_3, k_3\}}{min\{k_3, k_3\}} - 1\right) = 2,$$

$$D(\bar{T}_1, \bar{T}_2) = \gamma \times \left(\frac{max\{k_1, k_1\}}{min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_3, k_3\}}{min\{k_3, k_3\}} - 1\right) = 2.$$

Hence, $D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3) - D(\bar{T}_1, \bar{T}_3) = 2 + 2 - 2 = 2 \geq 0$,
which indicates $D(\bar{T}_1, \bar{T}_3) \leq D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3)$.

② *Inductive Step:* We now change one of the three series. Assume $\bar{T}_2 = ((a, k_1^*), (d, k_2), (c, k_3))$. Then,

$$\begin{aligned} D(\bar{T}_1, \bar{T}_2) &= \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_3, k_3\}}{max\{k_3, k_3\}} - 1\right) \\ &= \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2, \\ D(\bar{T}_2, \bar{T}_3) &= \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_3, k_3\}}{max\{k_3, k_3\}} - 1\right) \\ &= \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2, \\ D(\bar{T}_1, \bar{T}_3) &= \gamma \times \left(\frac{\max\{k_1, k_1\}}{\min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{\max\{k_3, k_3\}}{\max\{k_3, k_3\}} - 1\right) = 2, \\ D(\bar{T}_1, \bar{T}_2) &+ D(\bar{T}_2, \bar{T}_3) - D(\bar{T}_1, \bar{T}_3) \\ &= \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2 + \gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2 - 2 \\ &= 2\gamma \times \left(\frac{max\{k_1, k_1^*\}}{min\{k_1, k_1^*\}} - 1\right) + 2 \geq 0, \end{aligned}$$

which implies $D(\bar{T}_1, \bar{T}_3) \leq D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3)$.

Alternatively, assume $\bar{T}_3 = ((a, k_1), (e, k_2), (c, k_3), (f, k_4))$. Then,

$$\begin{aligned} D(\bar{T}_1, \bar{T}_2) &= \gamma \times \left(\frac{\max\{k_1, k_1\}}{\min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{\max\{k_3, k_3\}}{\min\{k_3, k_3\}} - 1\right) = 2, \\ D(\bar{T}_2, \bar{T}_3) &= \gamma \times \left(\frac{\max\{k_1, k_1\}}{\min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{\max\{k_3, k_3\}}{\min\{k_3, k_3\}} - 1\right) + 1 = 3, \\ D(\bar{T}_1, \bar{T}_3) &= \gamma \times \left(\frac{\max\{k_1, k_1\}}{\min\{k_1, k_1\}} - 1\right) + 2 + \gamma \times \left(\frac{\max\{k_3, k_3\}}{\min\{k_3, k_3\}} - 1\right) + 1 = 3. \end{aligned}$$

Thus, $D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3) - D(\bar{T}_1, \bar{T}_3) = 2 + 3 - 3 = 2 \geq 0$, which implies $D(\bar{T}_1, \bar{T}_3) \leq D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3)$.

The above two derivations show that regardless of changes to the symbols of the series or the corresponding number of symbols, $D(\bar{T}_1, \bar{T}_3) \leq D(\bar{T}_1, \bar{T}_2) + D(\bar{T}_2, \bar{T}_3)$. It should be noted that changing any one of the three series will lead to the same conclusion. Interested readers can prove this.

In fact, the above proof procedure can be applied to the general situation. Though finite adjustment of symbols and

the number of symbols, $\bar{T}_1$, $\bar{T}_2$ and $\bar{T}_3$ can be transformed into any series, so condition 4 can always be satisfied. This completes the proof.

REFERENCES

[1] Y. Fujii, K. Yamamoto, and S. Nakagawa, "Automatic speech recognition using hidden conditional neural fields," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2011, pp. 5036–5039.

[2] J. L. Rodríguez-Sotelo, D. Peluffo-Ordoñez, D. Cuesta-Frau, and G. Castellanos-Domínguez, "Unsupervised feature relevance analysis applied to improve ECG heartbeat clustering," *Comput. Methods Programs Biomed.*, vol. 108, no. 1, pp. 250–261, 2012.

[3] Z.-K. Gao, Q. Cai, Y.-X. Yang, W.-D. Dang, and S.-S. Zhang, "Multiscale limited penetrable horizontal visibility graph for analyzing nonlinear time series," *Sci. Rep.*, vol. 6, Oct. 2016, Art. no. 35622.

[4] X. Wang, S. Wang, J. Ma, and X. Sun, "Energy-aware scheduling of surveillance in wireless multimedia sensor networks," *Sensors*, vol. 10, pp. 3100–3125, Mar. 2010.

[5] X. Luo and K. Mori, "A discriminative structural similarity measure and its application to video-volume registration for endoscope three-dimensional motion tracking," *IEEE Trans. Med. Imag.*, vol. 33, no. 6, pp. 1248–1261, Jun. 2014.

[6] M. Krawczak and G. Szkatuła, "An approach to dimensionality reduction in time series," *Inf. Sci.*, vol. 260, pp. 15–36, Mar. 2014.

[7] L. N. Ferreira and L. Zhao, "Time series clustering via community detection in networks," *Inf. Sci.*, vol. 326, pp. 227–242, Jan. 2016.

[8] J. Ares, J. A. Lara, D. Lizcano, and S. Suárez, "A soft computing framework for classifying time series based on fuzzy sets of events," *Inf. Sci.*, vol. 330, pp. 125–144, Feb. 2016.

[9] E. Fuchs, T. Gruber, J. Nitschke, and B. Sick, "Online segmentation of time series based on polynomial least-squares approximations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2232–2245, Dec. 2010.

[10] I. Zliobaite, J. Bakker, and M. Pechenizkiy, "Beating the baseline prediction in food sales: How intelligent an intelligent predictor is?" *Expert Syst. Appl.*, vol. 39, pp. 806–815, Jan. 2012.

[11] H. Wang, M. Tang, Y. Park, and C. E. Priebe, "Locality statistics for anomaly detection in time series of graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 3, pp. 703–717, Feb. 2014.

[12] P. Esling and C. Agon, "Time-series data mining," *ACM Comput. Surv.*, vol. 45, no. 1, p. 12, 2013.

[13] N. Begum and E. Keogh, "Rare time series motif discovery from unbounded streams," *Proc. VLDB Endowment*, vol. 8, no. 2, pp. 149–160, 2014.

[14] F. Gullo, G. Ponti, A. Tagarelli, and S. Greco, "A time series representation model for accurate and fast similarity detection," *Pattern Recognit.*, vol. 42, no. 11, pp. 2998–3014, 2009.

[15] U. Mori, A. Mendiburu, and J. A. Lozano, "Similarity measure selection for clustering time series databases," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 181–195, Jan. 2016.

[16] Z.-K. Gao, Y.-X. Yang, P.-C. Fang, Y. Zou, C.-Y. Xia, and M. Du, "Multiscale complex network for analyzing experimental multivariate time series," *EPL(Europhys. Lett.)*, vol. 109, no. 3, p. 30005, 2015.

[17] Z.-K. Gao, W.-D. Dang, Y.-X. Yang, and Q. Cai, "Multiplex multivariate recurrence network from multi-channel signals for revealing oil-water spatial flow behavior," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 27, p. 035809, Mar. 2017.

[18] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowl. Inf. Syst.*, vol. 3, no. 3, pp. 263–286, 2001.

[19] Q. Li, B. Moon, and I. F. V. Lopez, "Skyline index for time series data," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 6, pp. 669–684, Jun. 2004.

[20] Z. Bankó and J. Abonyi, "Mixed dissimilarity measure for piecewise linear approximation based time series applications," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7664–7675, 2015.

[21] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining Knowl. Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[22] W. Deng, G. Wang, and J. Xu, "Piecewise two-dimensional normal cloud representation for time-series data mining," *Inf. Sci.*, vol. 374, pp. 32–50, Dec. 2016.

[23] Y.-L. Wu, D. Agrawal, and A. El Abbadi, "A comparison of DFT and DWT based similarity search in time-series databases," in *Proc. 9th Int. Conf. Inf. Knowl. Manage.*, 2000, pp. 488–495.

[24] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Proc. Int. Conf. Data Eng.*, Mar. 1999, pp. 126–133.

[25] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *ACM SIGMOD Rec.*, vol. 23, no. 2, pp. 419–429, 1994.

[26] K. V. R. Kanth, D. Agrawal, A. El Abbadi, and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *Comput. Vis. Image Understand.*, vol. 75, pp. 59–72, Jul. 1999.

[27] H. Hassanpour, A. Zehtabian, and S. J. Sadati, "Time domain signal enhancement based on an optimized singular vector denoising algorithm," *Digit. Signal Process.*, vol. 22, no. 5, pp. 786–794, Sep. 2012.

[28] Y. Cai, "Indexing spatio-temporal trajectories with Chebyshev polynomials," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Paris, France, Jun. 2004, pp. 599–610.

[29] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*. New York, NY, USA: CRC Press, 2002.

[30] I. Popivanov and R. J. Miller, "Similarity search over time-series data using wavelets," in *Proc. Int. Conf. Data Eng.*, Feb./Mar. 2002, pp. 212–221.

[31] K.-P. Chan, W.-C. Fu, and C. Yu, "Haar wavelets for efficient similarity search of time-series: With and without time warping," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 686–705, May 2003.

[32] K. Kalpakis, D. Gada, and V. Puttagunta, "Distance measures for effective clustering of ARIMA time-series," in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 273–280.

[33] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis, "Discovering dynamics using Bayesian clustering," in *Advances in Intelligent Data Analysis*. Amsterdam, The Netherlands: Springer-Verlag, Aug. 1999, pp. 199–210.

[34] A. Panuccio, M. Bicego, and V. Murino, "A hidden Markov model-based approach to sequential data clustering," in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. Windsor, ON, Canada: Springer-Verlag, Aug. 2002, pp. 734–742.

[35] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. Knowl. Discovery Databases Workshop*, 1994, pp. 359–370.

[36] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. SIAM Int. Conf. Data Mining*, 2001, pp. 1–11.

[37] S. H. Dumpala, K. N. R. K. R. Alluri, S. V. Gangashetty, and A. K. Vuppala, "Analysis of constraints on segmental DTW for the task of query-by-example spoken term detection," in *Proc. IEEE India Conf.*, Dec. 2015, pp. 1–6.

[38] C. Ratanamahatana and E. J. Keogh, "Making time-series classification more accurate using learned constraints," in *Proc. SIAM Int. Conf. Data Mining*, Lake Buena Vista, FL, USA, Apr. 2004, pp. 11–22.

[39] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, Feb./Mar. 2002, pp. 673–684.

[40] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 491–502.

[41] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proc. 13th Int. Conf. Very Large Data Bases*, 2004, pp. 792–803.

[42] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2010.

[43] *Machine Learning*, Mitchell, San Diego, CA, USA, 2003.

[44] G. Salton and D. Harman, *Information Retrieval*. Hoboken, NJ, USA: Wiley, 2003, p. 777.

[45] Y. Chen *et al.* (2015). *The UCR Time Series Classification Archive*. [Online]. Available: https://www.cs.ucr.edu/~eamonn/time_series_data/

MIAOMIAO ZHANG received the B.Sc. degree from the Department of Science, Anhui University of Science and Technology, Anhui, China, in 2015. She is currently pursuing the M.Sc. degree with the Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her current research interests include time series data mining and anomaly detection of ECG.

DECHANG PI received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China. He is currently a Professor and a Ph.D. Supervisor with NUAA. He has authored over 100 journals and conference papers. His research interests are data mining and privacy and security issues about moving objects. He presided over 30 research projects of the National Natural Science Foundation of China, National 863 Program, the National Technical Foundation, Civil Aerospace Foundation, and the Aviation Science Foundation.