k-Shape: Efficient and Accurate Clustering of Time Series

John Paparrizos
Columbia University
jopa@cs.columbia.edu

Luis Gravano
Columbia University
gravano@cs.columbia.edu

ABSTRACT

The proliferation and ubiquity of temporal data across many disciplines has generated substantial interest in the analysis and mining of time series. Clustering is one of the most popular data mining methods, not only due to its exploratory power, but also as a preprocessing step or subroutine for other techniques. In this paper, we describe k-Shape, a novel algorithm for time-series clustering. k-Shape relies on a scalable iterative refinement procedure, which creates homogeneous and well-separated clusters. As its distance measure, k-Shape uses a normalized version of the cross-correlation measure in order to consider the shapes of time series while comparing them. Based on the properties of that distance measure, we develop a method to compute cluster centroids, which are used in every iteration to update the assignment of time series to clusters. An extensive experimental evaluation against partitional, hierarchical, and spectral clustering methods, with the most competitive distance measures, showed the robustness of k-Shape. Overall, k-Shape emerges as a domain-independent, highly accurate, and efficient clustering approach for time series with broad applications.

1. INTRODUCTION

Temporal, or sequential, data mining deals with problems where data are naturally organized in sequences [28]. We refer to such data sequences as time-series sequences if they contain explicit information about timing (e.g., stock, audio, speech, and video) or if an ordering on values can be inferred (e.g., streams and handwriting). Large volumes of time-series sequences appear in almost every discipline, including astronomy, biology, meteorology, medicine, finance, robotics, engineering, and others [1, 5, 21, 23, 29, 43, 59, 62]. The ubiquity of time series has generated a substantial interest in querying [2, 38, 39, 41, 52, 61, 65], indexing [8, 11, 34, 35, 37, 63], classification [30, 47, 58, 70], clustering [36, 45, 54, 69, 71], and modeling [3, 31, 68] of such data.

Among all techniques applied to time-series data, clustering is the most widely used as it does not rely on costly human supervision or time-consuming annotation of data. With clustering, we can identify and summarize interesting patterns and correlations in the underlying data [27]. In the last few decades, clustering of time-series sequences has received significant attention [4, 14, 21, 40, 51, 54, 56, 69, 71], not only as a powerful stand-alone exploratory method, but also as a preprocessing step or subroutine for other tasks.

The original version of this paper was published in ACM SIGMOD 2015 [53].

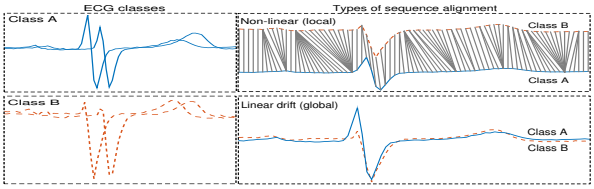


Figure 1: ECG sequence examples and types of alignments for the two classes of the ECGFiveDays dataset [1].

Most time-series analysis methods, including clustering, critically depend on the choice of distance measure. A key issue when comparing two sequences is how to handle the variety of distortions, as we will discuss, that are characteristic of the sequences. To illustrate this point, consider the ECGFiveDays dataset [1], with ECG sequences recorded for the same patient on two different days. While the sequences seem similar overall, they exhibit patterns that belong in one of the two distinct classes (see Figure 1): Class A is characterized by a sharp rise, a drop, and another gradual increase while Class B is characterized by a gradual increase, a drop, and another gradual increase. Ideally, a *shape-based* clustering method should generate a partition similar to the classes shown in Figure 1, where sequences exhibiting similar patterns are placed into the same cluster based on their *shape* similarity, regardless of differences in amplitude and phase. As the notion of shape cannot be precisely defined, dozens of distance measures have been proposed [9, 10, 12, 16, 18, 46, 64] to offer invariances to multiple inherent distortions in the data. However, it has been shown that distance measures offering invariances to amplitude and phase perform exceptionally well [15, 66] and, hence, such measures are used for shape-based clustering [44, 50, 54, 69].

Due to these difficulties and the different needs for invariances from one domain to another, more attention has been given to the creation of new distance measures rather than to the creation of new clustering algorithms. It is generally believed that the choice of distance measure is more important than the clustering algorithm itself [6]. As a consequence, time-series clustering relies mostly on classic clustering methods, either by replacing the default distance measure with one that is more appropriate for time series, or by transforming time series into "flat" data so that existing clustering algorithms can be directly used [67]. However, the choice of clustering method can affect: (i) accuracy, as every method expresses homogeneity and separation of clus-

ters differently; and (ii) efficiency, as the computational cost differs from one method to another. For example, spectral clustering [17] or certain variants of hierarchical clustering [33] are more appropriate to identify density-based clusters (i.e., areas of higher density than the remainder of the data) than partitional methods such as k-means [42] or k-medoids [33]. On the other hand, k-means is more efficient than hierarchical, spectral, or k-medoids methods.

Unfortunately, state-of-the-art approaches for shape-based clustering, which use partitional methods with distance measures that are scale- and shift-invariant, suffer from two main drawbacks: (i) these approaches cannot scale to large-volumes of data as they depend on computationally expensive methods or distance measures [44, 50, 54, 69]; and (ii) these approaches have been developed for particular domains [69] or their effectiveness has only been shown for a limited number of datasets [44, 50]. Moreover, the most successful shape-based clustering methods handle phase invariance through a local, non-linear alignment of the sequence coordinates, even though a global alignment is often adequate. For example, for the ECG dataset in Figure 1, an efficient linear drift can reveal the underlying differences in patterns of sequences of two classes, whereas an expensive non-linear alignment might match every corresponding increase or drop of each sequence, making it difficult to distinguish the two classes (see Figure 1). Importantly, to the best of our knowledge, these approaches have never been extensively evaluated against each other, against other partitional methods, or against different approaches such as hierarchical or spectral methods. We summarize such an experimental evaluation below. Our original paper [53] has further details.

In this article, we discuss k-Shape, a novel algorithm for shape-based time-series clustering that is efficient and domain independent. k-Shape is based on a scalable iterative refinement procedure similar to the one used by the k-means algorithm, but with significant differences. Specifically, k-Shape uses both a different distance measure and a different method for centroid computation from those of k-means. As argued above, k-Shape attempts to preserve the shapes of time-series sequences while comparing them. To do so, k-Shape requires a distance measure that is invariant to scaling and shifting. Unlike other clustering approaches [44, 54, 69], for k-Shape we adapt the cross-correlation statistical measure and we show: (i) how we can derive in a principled manner a time-series distance measure that is scale- and shift-invariant; and (ii) how this distance measure can be computed efficiently. Based on the properties of the normalized version of cross-correlation, we develop a novel method to compute cluster centroids, which are used in every iteration to update the assignment of time series to clusters.

To demonstrate the effectiveness of the distance measure and k-Shape, we have conducted an extensive experimental evaluation on 48 datasets and compared the state-of-the-art distance measures and clustering approaches for time series using rigorous statistical analysis. We took steps to ensure the reproducibility of our results, including making available our source code as well as using public datasets. Our experimental evaluation suggests that: (1) cross-correlation measures, which are not widely adopted as time-series distance measures, outperform Euclidean distance (ED) [16] and are as competitive as state-of-the-art measures, such as constrained Dynamic Time Warping (cDTW) [60], but significantly faster; (2) the k-means algorithm with ED, in contrast to what has been reported in the literature, is a robust approach for time-series clustering, but inadequate modifications of its distance measure and centroid computation can reduce its performance; (3) the choice of clustering method, which was believed to be less important than that of distance measure, is as important as the choice of distance measure; and (4) k-Shape outperforms all scalable approaches in terms of accuracy. Furthermore, k-Shape also outperforms all non-scalable (and hence impractical) approaches, with one exception that achieves similar accuracy results. However, unlike k-Shape, this approach requires tuning of its distance measure and is two orders of magnitude slower than k-Shape. Overall, k-Shape is a highly accurate and scalable choice for time-series clustering that performs exceptionally well across domains [53].

We start by reviewing the state of the art for clustering time series, as well as with our problem definition (Section 2). We then describe our approach, as follows:

- We show how a scale-, translate-, and shift-invariant distance measure can be derived in a principled manner from the cross-correlation measure and how this measure can be efficiently computed (Section 3.1).
- We present a novel method to compute a cluster centroid when that distance measure is used (Section 3.2).
- We describe k-Shape, a centroid-based algorithm for time-series clustering (Section 3.3).
- We summarize our extensive experimental evaluation (Sections 4 and 5).

We conclude with the implications of our work (Section 6). Please refer to [53] for further details on our approach and the experimental evaluation.

2. PRELIMINARIES

In this section, we review distortions that are common in time series (Section 2.1) and the most popular distance measures for such data (Section 2.2). Then, we summarize existing approaches for clustering time-series data (Section 2.3) and for centroid computation (Section 2.4). Finally, we formally present our problem of focus (Section 2.5).

2.1 Time-Series Invariances

Based on the domain, sequences are often distorted in some way, and distance measures need to satisfy a number of invariances in order to compare sequences meaningfully. In this section, we review common time-series distortions and their invariances. For a more detailed review, see [6].

Scaling and translation invariances: In many cases, it is useful to recognize the similarity of sequences despite differences in amplitude (scaling) and offset (translation). In other words, transforming a sequence $\vec{x}$ as $\vec{x}' = a\vec{x} + b$, where a and b are constants, should not change $\vec{x}$'s similarity to other sequences. For example, these invariances might be useful to analyze seasonal variations in currency values on foreign exchange markets without being biased by inflation.

Shift invariance: When two sequences are similar but differ in phase (global alignment) or when there are regions of the sequences that are aligned and others are not (local alignment), we might still need to consider them similar. For example, heartbeats can be out of phase depending on when we start taking the measurements (global alignment) and handwritings of a phrase from different people will need alignment depending on the size of the letters and on the spaces between words (local alignment).

Uniform scaling invariance: Sequences that differ in length require either stretching of the shorter sequence or shrinking of the longer sequence so that we can compare them effectively. For example, this invariance is required for heartbeats with measurement periods of different duration.
Occlusion invariance: When subsequences are missing, we can still compare the sequences by ignoring the subsequences that do not match well. This invariance is useful in handwritings if there is a typo or a letter is missing.
Complexity invariance: When sequences have similar shape but different complexities, we might want to make them have low or high similarity based on the application. For example, audio signals that were recorded indoors and outdoors might be considered similar, despite the fact that outdoor signals will be more noisy than indoor signals.

For many tasks, some or all of the above invariances are required when we compare time-series sequences. To satisfy the appropriate invariances, we could preprocess the data to eliminate the corresponding distortions before clustering. For example, by z-normalizing [24] the data we can achieve the scaling and translation invariances. However, for invariances that cannot be trivially achieved with a preprocessing step, we can define sophisticated distance measures that offer distortion invariances. In the next section, we review the most common such distance measures.

2.2 Time-Series Distance Measures

The two state-of-the-art approaches for time-series comparison first z-normalize the sequences and then use a distance measure to determine their similarity, and possibly capture more invariances. The most widely used distance metric is the simple ED [16]. ED compares two time series $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$ of length m as follows:

$$ED(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \quad (1)$$

Another popular distance measure is DTW [60]. DTW can be seen as an extension of ED that offers a local (non-linear) alignment. To achieve that, an m-by-m matrix M is constructed, with the ED between any two points of $\vec{x}$ and $\vec{y}$. A *warping path* $W = \{w_1, w_2, \ldots, w_k\}$, with $k \geq m$, is a contiguous set of matrix elements that defines a mapping between $\vec{x}$ and $\vec{y}$ under several constraints [37]:

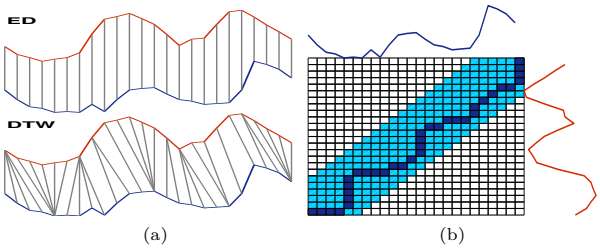
$$DTW(\vec{x}, \vec{y}) = \min \sqrt{\sum_{i=1}^{k} w_i} \quad (2)$$

This path can be computed on matrix M with dynamic programming for the evaluation of the following recurrence:

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

It is common practice to constrain the warping path to visit only a subset of cells on matrix M. The shape of the subset matrix is called *band* and the width of the band is called *warping window*. The most frequently used band for constrained Dynamic Time Warping (cDTW) is the Sakoe-Chiba band [60]. Figure 2a shows the difference in alignments of two sequences offered by ED and DTW distance measures, whereas Figure 2b presents the computation of the warping path (dark cells) for cDTW constrained by the Sakoe-Chiba band with width 5 cells (light cells).

Recently, Wang et al. [66] extensively evaluated 9 distance measures and several variants thereof. They found that ED is the most efficient measure with a reasonably high accuracy, and that DTW and cDTW perform exceptionally well in comparison to other measures. cDTW is slightly better



(a) (b)

Figure 2: Similarity computation: (a) alignment under ED (top) and DTW (bottom), (b) Sakoe-Chiba band with a warping window of 5 cells (light cells in band) and the warping path computed under cDTW (dark cells in band).

than DTW and significantly reduces the computation time. Several optimizations have been proposed to further speed up cDTW [55]. In the next section, we review clustering algorithms that can utilize these distance measures.

2.3 Time-Series Clustering Algorithms

Several methods have been proposed to cluster time series. All approaches generally modify existing algorithms, either by replacing the default distance measures with a version that is more suitable for comparing time series (raw-based methods), or by transforming the sequences into "flat" data so that they can be directly used in classic algorithms (feature- and model-based methods) [67]. Raw-based approaches can easily leverage the vast literature on distance measures (see Section 2.2), which has shown that invariances offered by certain measures, such as DTW, are general and, hence, suitable for almost every domain [15]. In contrast, feature- and model-based approaches are usually domain-dependent and applications on different domains require that we modify the features or models. Because of these drawbacks of feature- and model-based methods, in this paper we follow a raw-based approach.

The three most popular raw-based methods are agglomerative hierarchical, spectral, and partitional clustering [6]. For hierarchical clustering, the most widely used "linkage" criteria are the single, average, and complete linkage variants [33]. Spectral clustering [49] has recently started receiving attention [6] due to its success over other types of data [17]. Among partitional methods, k-means [42] and k-medoids [33] are the most representative examples. When partitional methods use distance measures that offer invariances to scaling, translation, and shifting, we consider them as shape-based approaches. From these methods, k-medoids is usually preferred [67]: unlike k-means, k-medoids computes the dissimilarity matrix of all data sequences and uses actual sequences as cluster centroids; in contrast, k-means requires the computation of artificial sequences as centroids, which hinders the easy adaptation of distance measures other than ED. However, from all these methods, only the k-means class of algorithms can scale linearly with the size of the datasets. Recently, k-means was modified to work with (i) DTW [54] and (ii) a distance measure that offers pairwise scaling and shifting of time-series sequences [69]. Both of these modifications rely on new methods to compute cluster centroids that we will review next.

2.4 Time-Series Averaging Techniques

The computation of an average sequence or, in the context of clustering, a centroid, is a difficult task that critically

depends on the distance measure used to compare time series. We now review the state-of-the-art methods for the computation of an average sequence.

With Euclidean distance, the arithmetic mean is used to compute an average sequence (e.g., as is the case in the centroid computation of the k-means algorithm). However, as DTW is more appropriate for many time-series tasks [37, 55], several methods have been proposed to average time-series sequences under DTW. Nonlinear alignment and averaging filters (NLAAF) [26] uses a simple pairwise method where each coordinate of the average sequence is calculated as the center of the mapping produced by DTW. This method is applied sequentially to pairs of sequences until only one pair is left. Prioritized shape averaging (PSA) [50] uses a hierarchical method to average sequences. The coordinates of an average sequence are computed as the weighted center of the coordinates of two time-series sequences that were coupled by DTW. Initially, all sequences have weight one, and each average sequence produced in the nodes of the tree has a weight that corresponds to the number of sequences it averages. To avoid the high computation cost of previous approaches, Ranking Shape-based Template Matching Framework (RSTMF) [44] approximates an ordering of the time-series sequences by looking at the distances of sequences to all other cluster centroids, instead of computing the distances of all pairs of sequences.

Several drawbacks of these methods have led to the creation of a more robust technique called Dynamic Time Warping Barycenter Averaging (DBA) [54], which iteratively refines the coordinates of a sequence initially picked from the data. Each coordinate of the average sequence is updated with the use of barycenter of one or more coordinates of the other sequences that were associated with the use of DTW. Among all these methods, DBA seems to be the most efficient and accurate averaging approach when DTW is used [54]. Another averaging technique that is based on matrix decomposition was proposed as part of K-Spectral Centroid Clustering (KSC) [69], to compute the centroid of a cluster when a distance measure for pairwise scaling and shifting is used. In our approach, which we will present in Section 3, we also rely on matrix decomposition to compute centroids.

2.5 Problem Definition

We address the problem of domain-independent, accurate, and scalable clustering of time series into k clusters, for a given value of the target number of clusters k.[1] Even though different domains might require different invariances to data distortions (see Section 2.1), we focus on distance measures that offer invariances to scaling and shifting, which are generally sufficient (see Section 2.2) [15]. Furthermore, to easily adopt such distance measures, we focus our analysis on raw-based clustering approaches, as we argued in Section 2.3. Next, we describe our k-Shape clustering algorithm.

3. K-SHAPE CLUSTERING ALGORITHM

Our objective is to develop a domain-independent, accurate, and scalable algorithm for time-series clustering that is invariant to scaling and shifting. We propose k-Shape, a clustering algorithm built on (i) a distance measure and (ii) a centroid computation method that can preserve the shapes of time series. We first discuss our distance measure, which is based on the cross-correlation measure (Section 3.1). Based on this distance measure, we propose a method to compute centroids of time-series clusters (Section 3.2). Finally, we describe k-Shape, our centroid-based clustering algorithm, which relies on an iterative refinement procedure that scales linearly in the number of sequences and generates homogeneous and well-separated clusters (Section 3.3).

3.1 Time-Series Shape Similarity

As discussed earlier, capturing shape-based similarity requires distance measures that can handle distortions in amplitude and phase. Unfortunately, the best performing distance measures offering invariances to these distortions, such as DTW, are computationally expensive (see Section 2.2). To circumvent this efficiency limitation, we adopt a normalized version of the cross-correlation measure.

Cross-correlation is a measure of similarity for time-lagged signals that is widely used for signal and image processing. However, cross-correlation, a measure that compares one-to-one points between signals, has largely been ignored in experimental evaluations for the problem of time-series comparison. Instead, starting with the application of DTW decades ago [7], research on that problem has focused on elastic distance measures that compare one-to-many or one-to-none points [9, 10, 37, 46, 64]. In particular, recent comprehensive and independent experimental evaluations of state-of-the-art distance measures for time-series comparison — 9 measures and their variants in [15, 66] and 48 measures in [22] — did not consider cross-correlation. Different needs from one domain or application to another hinder the process of finding appropriate normalizations for the data and the cross-correlation measure. Moreover, inefficient implementations of cross-correlation can make it appear as slow as DTW. As a consequence of these drawbacks, cross-correlation has not been widely adopted as a time-series distance measure. In the rest of this section, we show how to address these drawbacks. Specifically, we will show how to choose normalizations that are domain-independent and efficient, and lead to a shape-based distance measure for comparing time series efficiently and effectively.

Cross-correlation measure: Cross-correlation is a statistical measure with which we can determine the similarity of two sequences $\vec{x} = (x_1, \ldots, x_m)$ and $\vec{y} = (y_1, \ldots, y_m)$, even if they are not properly aligned.[2] To achieve shift-invariance, cross-correlation keeps $\vec{y}$ static and slides $\vec{x}$ over $\vec{y}$ to compute their inner product for each *shift* s of $\vec{x}$. We denote a shift of a sequence as follows:

$$\vec{x}_{(s)} = \begin{cases} (\overbrace{0, \ldots, 0}^{|s|}, x_1, x_2, \ldots, x_{m-s}), & s \geq 0 \\ (x_{1-s}, \ldots, x_{m-1}, x_m, \underbrace{0, \ldots, 0}_{|s|}), & s < 0 \end{cases} \quad (3)$$

When all possible shifts $\vec{x}_{(s)}$ are considered, with $s \in [-m, m]$, we produce $CC_w(\vec{x}, \vec{y}) = (c_1, \ldots, c_w)$, the cross-correlation sequence with length $2m-1$, defined as follows:

$$CC_w(\vec{x}, \vec{y}) = R_{w-m}(\vec{x}, \vec{y}), \quad w \in \{1, 2, \ldots, 2m-1\} \quad (4)$$

where $R_{w-m}(\vec{x}, \vec{y})$ is computed, in turn, as:

[1] Although the exact estimation of k is difficult without a gold standard, we can do so by varying k and evaluating clustering quality with criteria that capture information intrinsic to the data alone [33].

[2] For simplicity, we consider sequences of equal length even though cross-correlation can be computed on sequences of different length.

$$R_k(\vec{x},\vec{y}) = \begin{cases} \sum_{l=1}^{m-k} x_{l+k} \cdot y_l, & k \geq 0 \\ R_{-k}(\vec{y},\vec{x}), & k < 0 \end{cases} \quad (5)$$

Our goal is to compute the position w at which $CC_w(\vec{x},\vec{y})$ is maximized. Based on this value of w, the optimal shift of $\vec{x}$ with respect to $\vec{y}$ is then $\vec{x}_{(s)}$, where $s = w - m$.

Depending on the domain or the application, different normalizations for $CC_w(\vec{x},\vec{y})$ might be required. The most common normalizations are the biased estimator, NCC_b, the unbiased estimator, NCC_u, and the coefficient normalization, NCC_c, which are defined as follows:

$$NCC_q(\vec{x},\vec{y}) = \begin{cases} \frac{CC_w(\vec{x},\vec{y})}{m}, & q = \text{“}b\text{”}\ (NCC_b) \\ \frac{CC_w(\vec{x},\vec{y})}{m-|w-m|}, & q = \text{“}u\text{”}\ (NCC_u) \\ \frac{CC_w(\vec{x},\vec{y})}{\sqrt{R_0(\vec{x},\vec{x})\cdot R_0(\vec{y},\vec{y})}}, & q = \text{“}c\text{”}\ (NCC_c) \end{cases} \quad (6)$$

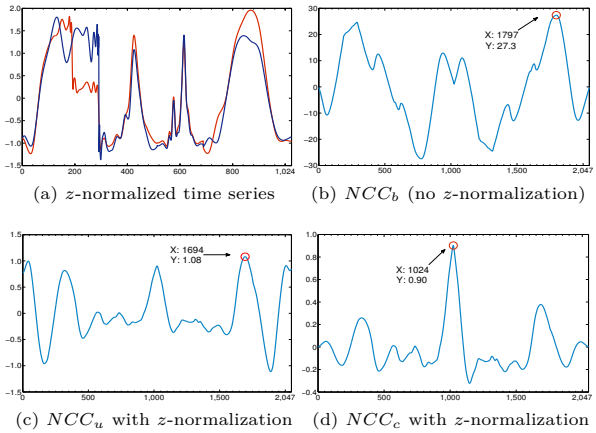
Beyond the cross-correlation normalizations, time series might also require normalization to remove inherent distortions. Figure 3 illustrates how the cross-correlation normalizations for two sequences $\vec{x}$ and $\vec{y}$ of length $m = 1024$ are affected by time-series normalizations. Independently of the normalization applied to $CC_w(\vec{x},\vec{y})$, the produced sequence will have length 2047. Initially, in Figure 3a, we remove differences in amplitude by z-normalizing $\vec{x}$ and $\vec{y}$ in order to show that they are aligned and, hence, no shifting is required. If $CC_w(\vec{x},\vec{y})$ is maximized for $w \in [1025, 2047]$ (or $w \in [1, 1023]$), one of $\vec{x}$ or $\vec{y}$ should be shifted by $i - 1024$ to the right (or $1024 - i$ to the left). Otherwise, if $w = 1024$, $\vec{x}$ and $\vec{y}$ are properly aligned, which is what we expect in our example. Figure 3b shows that if we do not z-normalize $\vec{x}$ and $\vec{y}$, and we use the biased estimator, then NCC_b is maximized at $w = 1797$, which indicates a shifting of a sequence to the left $1797 - 1024 = 773$ times. If we z-normalize $\vec{x}$ and $\vec{y}$, and use the unbiased estimator, then NCC_u is maximized at $w = 1694$, which indicates a shifting of a sequence to the right $1694 - 1024 = 670$ times (Figure 3c). Finally, if we z-normalize $\vec{x}$ and $\vec{y}$, and use the coefficient normalization, then NCC_c is maximized at $w = 1024$, which indicates that no shifting is required (Figure 3d).

As illustrated by the example, normalizations of the data and the cross-correlation measure can have a significant impact on the cross-correlation sequence produced, which makes the creation of a distance measure a non-trivial task. Furthermore, as in Figure 3, cross-correlation sequences produced by pairwise comparisons of multiple time series will differ in amplitude based on the normalizations. Thus, a normalization that produces values within a specified range should be used to meaningfully compare such sequences.

Shape-based distance (SBD): To devise a shape-based distance measure, and based on the previous discussion, we use the coefficient normalization that gives values between -1 and 1, regardless of the data normalization. Coefficient normalization divides the cross-correlation sequence by the geometric mean of autocorrelations of the individual sequences. After normalization of the sequence, we detect the position w where $NCC_c(\vec{x},\vec{y})$ is maximized and we derive the following distance measure:

$$SBD(\vec{x},\vec{y}) = 1 - \max_w \left(\frac{CC_w(\vec{x},\vec{y})}{\sqrt{R_0(\vec{x},\vec{x}) \cdot R_0(\vec{y},\vec{y})}} \right) \quad (7)$$

which takes values between 0 to 2, with 0 indicating perfect similarity for time-series sequences.



(a) z-normalized time series (b) NCC_b (no z-normalization)

(c) NCC_u with z-normalization (d) NCC_c with z-normalization

Figure 3: Time-series and cross-correlation normalizations.

Up to now we have addressed shift invariance. For scaling invariance, we transform each sequence $\vec{x}$ into $\vec{x}' = \frac{\vec{x}-\mu}{\sigma}$, so that its mean μ is zero and its standard deviation σ is one.

Efficient computation of SBD: From Equation 4, the computation of $CC_w(\vec{x},\vec{y})$ for all values of w requires $\mathcal{O}(m^2)$ time, where m is the time-series length. The convolution theorem [32] states that the convolution of two time series can be computed as the Inverse Discrete Fourier Transform (IDFT) of the product of their individual Discrete Fourier Transforms (DFT). Cross-correlation is then computed as the convolution of two time series if one sequence is first reversed in time, $\vec{x}^{(t)} = \vec{x}^{(-t)}$ [32], which equals taking the complex conjugate in the frequency domain. However, DFT and IDFT still require $\mathcal{O}(m^2)$ time. By using a Fast Fourier Transform (FFT) algorithm [13], the time reduces to $\mathcal{O}(m \log(m))$. Data and cross-correlation normalizations can also be efficiently computed; thus the overall time complexity of SBD remains $\mathcal{O}(m \log(m))$. Moreover, recursive algorithms compute an FFT by dividing it into pieces of power-of-two size [20]. Therefore, to further improve the performance of the FFT computation, when $CC(\vec{x},\vec{y})$ is not an exact power of two we pad $\vec{x}$ and $\vec{y}$ with zeros to reach the next power-of-two length after $2m - 1$.

This section described effective cross-correlation and data normalizations to derive a shape-based distance measure. Importantly, we also discussed how the cross-correlation distance measure can be efficiently computed. Our experiments show that SBD is highly competitive, achieving similar results to cDTW and DTW while being orders of magnitude faster. We now turn to the critical problem of extracting a centroid for a cluster, to represent the cluster data consistently with the above shape-based distance measure.

3.2 Time-Series Shape Extraction

Many time-series tasks rely on methods that summarize a set of time series by only one sequence, often referred to as an *average sequence* or, in the context of clustering, as a *centroid*. The extraction of meaningful centroids is a challenging task that critically depends on the choice of distance measure. We now show how to determine such centroids for time-series clustering for the SBD distance measure, to capture shared characteristics of the underlying data.

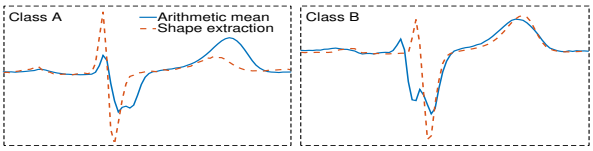


Figure 4: Examples of centroids for each class of the ECGFiveDays dataset, based on the arithmetic mean property (solid lines) and our shape extraction method (dashed lines).

The easiest way to extract an average sequence from a set of sequences is to compute each coordinate of the average sequence as the arithmetic mean of the corresponding coordinates of all sequences. This approach is used by k-means, the most popular clustering method. In Figure 4, the solid lines show such centroids for each class in the ECGFiveDays dataset of Figure 1: these centroids do not capture effectively the class characteristics (see Figures 1 and 4).

To avoid such problems, we cast the centroid computation as an optimization problem where the objective is to find the minimizer of the sum of squared distances to all other time series sequences. However, as cross-correlation intuitively captures the similarity — rather than the dissimilarity — of time series, we can express the computed sequence as the maximizer of the squared similarities to all other time-series sequences. Such similarity (Equation 6) requires the computation of an optimal shift for every sequence. As this approach is used in the context of iterative clustering, we use the previously computed centroid as reference and align all sequences towards this reference sequence. This is a reasonable choice because the previous centroid will be very close to the new centroid. For this alignment, we use SBD, which identifies an optimal shift for every sequence. Subsequently, as sequences are already aligned towards a reference sequence, we can reduce this maximization to a well-known problem called maximization of the Rayleigh Quotient [25]. (See details of this reduction in [53].)

A desirable property of the above formulation is that we can extract the most representative shape from the underlying data in a few lines of code [53]. In Figure 4, the dashed lines show the centroids of each class in the ECGFiveDays dataset, extracted with our shape extraction method and using randomly selected sequences as reference sequences. This method for shape extraction can more effectively capture the characteristics of each class (Figure 1) than by using the arithmetic mean property (solid lines in Figure 4). We now show how our shape extraction method is used in a time-series clustering algorithm.

3.3 Shape-based Time-Series Clustering

We now describe k-Shape, our novel algorithm for time-series clustering. k-Shape relies on the SBD distance measure of Section 3.1 and the shape extraction method of Section 3.2 to efficiently produce clusters of time series.

k-Shape Clustering Algorithm: k-Shape is a partitional clustering method that is based on an iterative refinement procedure similar to the one used in k-means. Through this iterative procedure, k-Shape minimizes the sum of squared distances and manages to: (i) produce homogeneous and well-separated clusters, and (ii) scale linearly with the number of time series. Our algorithm compares sequences efficiently and computes centroids effectively under the scaling, translation, and shift invariances. k-Shape is a non-trivial instantiation of k-means and, in contrast to similar attempts in the literature [54, 69], its distance measure and centroid computation method make k-Shape the only scalable method that significantly outperforms k-means.

In every iteration, k-Shape performs two steps: (i) in the assignment step, the algorithm updates the cluster memberships by comparing each time series with all computed centroids and by assigning each time series to the cluster of the closest centroid; (ii) in the refinement step, the cluster centroids are updated to reflect the changes in cluster memberships in the previous step. The algorithm repeats these two steps until either no change in cluster membership occurs or the maximum number of iterations allowed is reached. In the assignment step, k-Shape relies on the distance measure of Section 3.1, whereas in the refinement step it relies on the centroid computation method of Section 3.2.

k-Shape expects as input the time series set and the number of clusters that we want to produce. (Please refer to [53] for the full algorithm.) Initially, we randomly assign the time series in to clusters. Then, we compute each cluster centroid with the shape extraction method (see Section 3.2). Once the centroids are computed, we refine the memberships of the clusters by using the SBD distance measure. We repeat this procedure until the algorithm converges or reaches the maximum number of iterations (usually a small number, such as 100). The output of the algorithm is the assignment of sequences to clusters and the centroids for each cluster.

We now turn to the experimental evaluation of k-Shape against the state-of-the-art time-series clustering approaches.

4. EXPERIMENTAL SETTINGS

In this section, we describe the experimental settings for the evaluation of both SBD and our k-Shape algorithm.

Datasets: We use 48 class-labeled time-series datasets, both synthetic and real, which span several different domains [1].

Platform: We ran our experiments on a cluster of 10 servers with identical configuration: Dual Intel Xeon X5550 processor with clock speed at 2.67 GHz and 24 GB RAM. Each server runs Ubuntu 12.04 and Matlab R2012b.

Implementation: We implemented our approach and all state-of-the-art approaches that we compare against under the same framework, in Matlab, for a consistent evaluation in terms of both accuracy and efficiency. For repeatability purposes, we make all datasets and source code available.[3]

Baselines: We compare SBD against the strongest state-of-the-art distance measures for time series (see Section 2.2 for a detailed discussion), namely, ED, DTW, and cDTW. Only cDTW requires setting a parameter, to constrain its warping window. We consider two cases from the literature: (i) cDTW^{opt}: we compute the optimal window by performing a leave-one-out classification step over the training set of each dataset; (ii) cDTW^{w}: we use as window 5%, for cDTW^{5}, of the length of the time series of each dataset. We compare k-Shape against the three strongest types of scalable and non-scalable clustering methods, namely, partitional, hierarchical, and spectral methods (see Section 2.3 for a detailed discussion), combined with the most competitive distance measures discussed previously (we denote them as Dist). As scalable methods, we consider the classic k-means algorithm with ED (k-AVG+ED) [42], and the following vari-

[3] http://www.cs.columbia.edu/~jopa/kshape.html

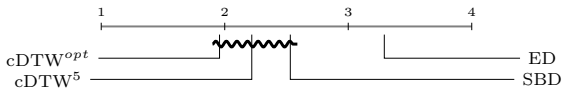


Figure 5: Ranking of distance measures based on the average of their ranks across datasets. The wiggly line connects all measures that do not perform statistically differently according to the Nemenyi test.

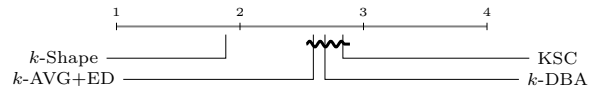


Figure 6: Ranking of k-means variants based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

ants: (i) k-means with DTW as distance measure and the DBA method for centroid computation (k-DBA) [54] and (ii) k-means with a distance measure offering pairwise scaling and shifting of time series and computation of the spectral norm of a matrix for centroid computation (KSC) [69]. As non-scalable methods, among partitional methods we consider the Partitioning Around Medoids (PAM+Dist) implementation of the k-medoids algorithm [33]. Among hierarchical methods, we use agglomerative hierarchical clustering with single (H-S+Dist), average (H-A+Dist), and complete (H-C+Dist) linkage criteria [33]. Finally, among spectral methods, we consider the popular normalized spectral clustering method (S+Dist) [49]. Overall, we compared k-Shape against 20 clustering approaches.

Metrics: We compute CPU time utilization and report time ratios for our comparisons. We use the one nearest neighbor classification accuracy to evaluate the distance measures and the Rand Index [57] to evaluate clustering accuracy.

Statistical analysis: We use the Friedman test [19] followed by the post-hoc Nemenyi test [48] for comparison of multiple algorithms over multiple datasets and we report statistical significant results with a 95% confidence level.

5. EXPERIMENTAL RESULTS

We now provide highlights of the detailed experimental evaluation in [53]. First, we evaluate SBD against the state-of-the-art distance measures. Then, we compare k-Shape against scalable and non-scalable clustering approaches.

Evaluation of SBD: All distance measures, including SBD, outperform ED with statistical significance. The difference in accuracy between SBD and DTW is in most cases negligible: SBD performs at least as well as DTW in 30 datasets. Considering the constrained versions of DTW, we observe that SBD performs similarly to or better than cDTW^{opt} and cDTW^{5}in 22 and 18 datasets, respectively. To better understand the performance of SBD in comparison with cDTW^{opt} and cDTW^5, we evaluate the significance of their differences in accuracy when considered all together. Figure 5 shows the average rank across datasets of each distance measure. cDTW^{opt} is the top measure, with an average rank of 1.96, meaning that cDTW^{opt} performed best in the majority of the datasets. The Friedman test rejects the null hypothesis that all measures behave similarly, and, hence, we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. The wiggly line in the figure connects all measures that do not perform statistically differently according to the Nemenyi test. We observe that the ranks of cDTW^{opt}, cDTW^5, and SBD do not present a significant difference, and ED, which is ranked last, is significantly worse than the others. In terms of efficiency, SBD is only 4.4x slower than ED and remains one order of magnitude faster than cDTW^{opt} and cDTW^5. In conclusion, SBD is a very efficient, parameter-free distance measure that significantly outperforms ED and achieves similar results to both constraint and unconstraint versions of DTW.

Evaluation of k-Shape Against Other Scalable Methods: Figure 6 shows the average rank across datasets of each k-means variant. k-Shape is the top technique, with an average rank of 1.89, meaning that k-Shape was best in the majority of the datasets. The Friedman test rejects that all algorithms behave similarly, so we proceed with a post-hoc Nemenyi test, to evaluate the significance of the differences in the ranks. We observe that the ranks of KSC, k-DBA, and k-AVG+ED do not present a statistically significant difference, whereas k-Shape, which is ranked first, is significantly better than the others. Modifying k-means with inappropriate distance measures or centroid computation methods might lead to unexpected results. In terms of efficiency, k-Shape is one order of magnitude faster than KSC, two orders of magnitude faster than k-DBA, and one order of magnitude slower than k-AVG+ED.

Evaluation of k-Shape Against Non-Scalable Methods: To show the robustness of k-Shape in terms of accuracy beyond scalable approaches, we now ignore scalability and compare k-Shape against hierarchical, spectral, and k-medoids methods. Among all existing state-of-the-art methods that use ED or cDTW^5 as distance measures, only partitional methods perform similarly to or better than k-AVG+ED. In particular, PAM+cDTW^5 is the only method that outperforms k-AVG+ED. Figure 7 shows that k-Shape, PAM+SBD, PAM+cDTW^5, and S+SBD (i.e., all methods outperforming k-AVG+ED) do not present a significant difference in accuracy, whereas k-AVG+ED, which is ranked last, is significantly worse than the others.

In short, our experimental evaluation suggests that SBD is as competitive as state-of-the-art measures, such as cDTW and DTW, but faster, and k-Shape is the only method that is both accurate and efficient. In [53], we provide further details on these findings and on the performance of hierarchical and spectral methods as well.

6. CONCLUSIONS

We presented k-Shape, a partitional clustering algorithm that preserves the shapes of time series. k-Shape compares time series efficiently and computes centroids effectively under the scaling and shift invariances. We have identified many interesting directions for future work. For example, k-Shape currently operates over a single time-series representation and cannot handle multiple representations. Considering that several transformations (e.g., smoothing) can reduce noise and eliminate outliers in time series, an extension of k-Shape to leverage characteristics from multiple representations can significantly improve its accuracy. Another future direction is to explore the usefulness of k-Shape as a "subroutine" of other methods. For example, nearest centroid classifiers rely on effective clustering of time series and subsequent extraction of centroids for the clusters.

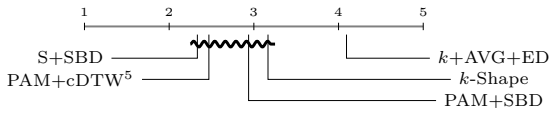


Figure 7: Ranking of methods that outperform k-AVG+ED based on the average of their ranks across datasets. The wiggly line connects all techniques that do not perform statistically differently according to the Nemenyi test.

Acknowledgments: We thank Or Biran, Christos Faloutsos, Eamonn Keogh, Kathy McKeown, Taesun Moon, François Petitjean, and Kapil Thadani for invaluable discussions and feedback. We also thank Ken Ross for sharing computing resources for our experiments. This research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D11PC20153. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government. This material is also based upon work supported by a generous gift from Microsoft Research. John Paparrizos is an Alexander S. Onassis Foundation Scholar.

7. REFERENCES

[1] The UCR Time Series Classification/Clustering Homepage. http://www.cs.ucr.edu/~eamonn/time_series_data. Accessed: May 2014.

[2] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, pages 69–84, 1993.

[3] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *CVPR*, pages 375–381, 2003.

[4] A. J. Bagnall and G. J. Janacek. Clustering time series from ARMA models with clipped data. In *KDD*, pages 49–58, 2004.

[5] Z. Bar-Joseph, G. Gerber, D. K. Gifford, T. S. Jaakkola, and I. Simon. A new approach to analyzing gene expression time series data. In *RECOMB*, pages 39–48, 2002.

[6] G. E. Batista, E. J. Keogh, O. M. Tataw, and V. M. de Souza. CID: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, pages 1–36, 2013.

[7] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI Workshop on KDD*, pages 359–370, 1994.

[8] Y. Cai and R. Ng. Indexing spatio-temporal trajectories with Chebyshev polynomials. In *SIGMOD*, pages 599–610, 2004.

[9] L. Chen and R. Ng. On the marriage of Lp-norms and edit distance. In *VLDB*, pages 792–803, 2004.

[10] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, pages 491–502, 2005.

[11] Q. Chen, L. Chen, X. Lian, Y. Liu, and J. X. Yu. Indexable PLA for efficient similarity search. In *VLDB*, pages 435–446, 2007.

[12] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. Tung. Spade: On shape-based pattern detection in streaming time series. In *ICDE*, pages 786–795, 2007.

[13] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[14] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *KDD*, pages 16–22, 1998.

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

[16] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, pages 419–429, 1994.

[17] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[18] E. Frentzos, K. Gratsias, and Y. Theodoridis. Index-based most similar trajectory search. In *ICDE*, pages 816–825, 2007.

[19] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, 1937.

[20] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[21] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market: Which measure is best? In *KDD*, pages 487–496, 2000.

[22] R. Giusti and G. E. Batista. An empirical comparison of dissimilarity measures for time series classification. In *BRACIS*, pages 82–88, 2013.

[23] S. Goddard, S. K. Harms, S. E. Reichenbach, T. Tadesse, and W. J. Waltman. Geospatial decision support for drought risk management. *Communications of the ACM*, 46(1):35–37, 2003.

[24] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In *CP*, pages 137–153, 1995.

[25] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[26] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos. Nonlinear alignment and averaging for estimating the evoked potential. *IEEE Transactions on Biomedical Engineering*, 43(4):348–356, 1996.

[27] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, 2001.

[28] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 3rd edition, 2011.

[29] R. Honda, S. Wang, T. Kikuchi, and O. Konishi. Mining of moving objects from time-series images and its application to satellite weather imagery. *Journal of Intelligent Information Systems*, 19(1):79–93, 2002.

[30] B. Hu, Y. Chen, and E. Keogh. Time series classification under more realistic assumptions. In *SDM*, pages 578–586, 2013.

[31] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *ICDM*, pages 273–280, 2001.

[32] Y. Katznelson. *An introduction to harmonic analysis*. Cambridge University Press, 2004.

[33] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[34] E. Keogh. A decade of progress in indexing and mining large time series databases. In *VLDB*, pages 1268–1268, 2006.

[35] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*, pages 151–162, 2001.

[36] E. Keogh and J. Lin. Clustering of time-series subsequences is meaningless: Implications for previous and future research. *Knowledge and Information Systems*, 8(2):154–177, 2005.

[37] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and Information Systems*, 7(3):358–386, 2005.

[38] C. Kin-pong and F. Ada. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[39] F. Korn, H. V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *SIGMOD*, pages 289–300, 1997.

[40] C.-S. Li, P. S. Yu, and V. Castelli. MALM: A framework for mining sequence database at multiple abstraction levels. In *CIKM*, pages 267–272, 1998.

[41] X. Lian, L. Chen, J. X. Yu, G. Wang, and G. Yu. Similarity match over high speed time-series streams. In *ICDE*, pages 1086–1095, 2007.

[42] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *BSMSP*, pages 281–297, 1967.

[43] R. N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

[44] W. Meesrikamolkul, V. Niennattrakul, and C. A. Ratanamahatana. Shape-based clustering for time series data. In *PAKDD*, pages 530–541. 2012.

[45] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *ICDE*, pages 668–679, 2005.

[46] M. D. Morse and J. M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD*, pages 569–580, 2007.

[47] A. Mueen, E. Keogh, and N. Young. Logical-shapelets: An expressive primitive for time series classification. In *KDD*, pages 1154–1162, 2011.

[48] P. Nemenyi. *Distribution-free Multiple Comparisons*. PhD thesis, Princeton University, 1963.

[49] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2002.

[50] V. Niennattrakul and C. A. Ratanamahatana. Shape averaging under time warping. In *ECTI-CON*, pages 626–629, 2009.

[51] T. Oates. Identifying distinctive subsequences in multivariate time series by clustering. In *KDD*, pages 322–326, 1999.

[52] P. Papapetrou, V. Athitsos, M. Potamias, G. Kollios, and D. Gunopulos. Embedding-based subsequence matching in time-series databases. *TODS*, 36(3):17, 2011.

[53] J. Paparrizos and L. Gravano. k-Shape: Efficient and accurate clustering of time series. In *SIGMOD*, pages 1855–1870, 2015.

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

[55] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, pages 262–270, 2012.

[56] T. Rakthanmanon, E. J. Keogh, S. Lonardi, and S. Evans. Time series epenthesis: Clustering time series streams requires ignoring some data. In *ICDM*, pages 547–556, 2011.

[57] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[58] C. A. Ratanamahatana and E. Keogh. Making time-series classification more accurate using learned constraints. In *SDM*, pages 11–22, 2004.

[59] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. In *WSDM*, pages 513–522, 2012.

[60] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

[61] Y. Shou, N. Mamoulis, and D. Cheung. Fast and exact warping of time series using adaptive segmental approximations. *Machine Learning*, 58(2-3):231–267, 2005.

[62] K. Uehara and M. Shimada. Extraction of primitive motion and discovery of association rules from human motion data. In *Progress in Discovery Science*, pages 338–348. 2002.

[63] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh. Indexing multidimensional time-series. *The VLDB Journal*, 15(1):1–20, 2006.

[64] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *ICDE*, pages 673–684, 2002.

[65] H. Wang, Y. Cai, Y. Yang, S. Zhang, and N. Mamoulis. Durable queries over historical time series. *TKDE*, 26(3):595–607, 2014.

[66] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.

[67] T. Warren Liao. Clustering of time series data - a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[68] Y. Xiong and D.-Y. Yeung. Mixtures of ARMA models for model-based time series clustering. In *ICDM*, pages 717–720, 2002.

[69] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, pages 177–186, 2011.

[70] L. Ye and E. Keogh. Time series shapelets: A new primitive for data mining. In *KDD*, pages 947–956, 2009.

[71] J. Zakaria, A. Mueen, and E. Keogh. Clustering time series using unsupervised-shapelets. In *ICDM*, pages 785–794, 2012.