Efficient Classification of Long Time Series by 3-D Dynamic Time Warping

Anooshiravan Sharabiani, Houshang Darabi, *Senior Member, IEEE*, Ashkan Rezaei, Samuel Harford, Hereford Johnson, and Fazle Karim

***Abstract*—Throughout recent years, dynamic time warping (DTW) has remained as a robust similarity measure in time series classification (TSC). 1-nearest neighbor (1-NN) algorithm with DTW is the most widely used classification method on time series serving as a benchmark. With the increasing demand for TSC on low-resource devices and the widespread of wearable devices, the need for a efficient and accurate time series classifier has never been higher. Although 1-NN DTW attains accurate results, it highly falls back on efficiency due to its quadratic complexity in the length of time series. In this paper, we propose a new approximation method for reducing the length of the time series as the input of DTW. We call it control chart approximation (CCA), after a similar concept used in statistical quality control processing. CCA representation approximates raw time series by transforming them into a set of segments with aggregated values and durations forming a reduced 3-D vector. We also propose an adaptation of DTW in 3-D space as a distance measure for 1-NN classifier, and denote the method as 1-NN 3-D DTW. Our experiments on 85 datasets from UCR archive—including 28 long-length (>500 points) time series datasets—show up to two orders of magnitude performance gain in running time compared to the state-of-the-art 1-NN DTW implementation. Moreover, it shows similar or better accuracy on the long time series in the experiment.**

***Index Terms*—Approximation methods, dynamic time warping (DTW), time series classification (TSC).**

I. Introduction

OVER the past few years, substantial amounts of data has been collected and analyzed to assist in optimal decision making. This collection and analysis of large datasets has become a primary contributor toward the innovation and growth of the current and emerging markets. A significant portion of this collected data is in the form of time series. The exponentially increasing volume and complexity of time series data is a result of emerging new sensing technologies (robot sensors, wearable sensors, smart meters, satellites, smart mobile phones, etc.), along with an influx of inexpensive storage. The key objective of time series analysis is to extract hidden insights from raw data. Time series classification (TSC) is one of the most important tasks in time series analysis. There are a variety of TSC techniques in the literature. Distance-based classification techniques such as 1-nearest neighbor (1-NN) require a similarity measure to calculate the distance (similarity) between two time series.

Euclidean distance (ED) [1] and dynamic time warping (DTW) [2]–[5] are the most popular distance-measures for time series. 1-NN ED is often used for fast classification of the time series of equal length. However, its accuracy is highly sensitive to noise and it cannot be used when the lengths of time series are different.

1-NN DTW classifier is one of the best distance-based classifiers in terms of accuracy [6], [7]. It has been successfully applied in a variety of domains and problems such as analyzing robots sensory signals [8], [9], medicine [5], [10], astronomy, biometric data [11], geology, historical manuscripts, speech/music, gesture, signature [12], and fingerprint recognition. 1-NN DTW can handle classification of time series of different lengths, and is typically used as a benchmark among time series classifiers [6]–[8], [13]. The drawback of 1-NN DTW is its quadratic time complexity, which has reportedly undermined its usefulness in many use cases. Multiple research efforts would have used 1-NN DTW if it had not been so computationally expensive [13].

The state-of-the-art implementation of 1-NN DTW [13] reduces the computational complexity by omitting square root calculations, using lower bounding, and applying early abandoning. However, it is still inefficient when classifying long time series and large datasets. In this paper, we propose a new representation method, called control chart approximation (CCA), which transforms raw time series data, from the 2-D space of time-values, to a sequence of segments each forming a tuple of (segment begin time, segment average of values, segment duration); with the durations being z-normalized across the training time series. The data reduction results in reduced noise, space, and length of the time series before classification. An example of CCA transformation is presented in Fig. 1.

To perform nearest neighbor classification in this new data space, we propose the 3-D DTW (1-NN 3-D DTW), which is based on 1-NN DTW and works in 3-D space. We show that 1-NN 3-D DTW significantly reduces the computation time of TSC while still remaining competitive in terms of accuracy.

Manuscript received September 29, 2016; revised February 11, 2017; accepted March 23, 2017. Date of publication May 17, 2017; date of current version September 15, 2017. This paper was recommended by Associate Editor G.-B. Huang. *(Corresponding author: Houshang Darabi.)*

A. Sharabiani, H. Darabi, S. Harford, H. Johnson, and F. Karim are with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: ashara3@uic.edu; hdarabi@uic.edu; sharfo2@uic.edu; hjohns23@uic.edu; karim1@uic.edu).

A. Rezaei is with the Department of Computer Science, University of Illinois at Chicago, Chicago IL 60607 USA (e-mail: arezae4@uic.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2017.2699333

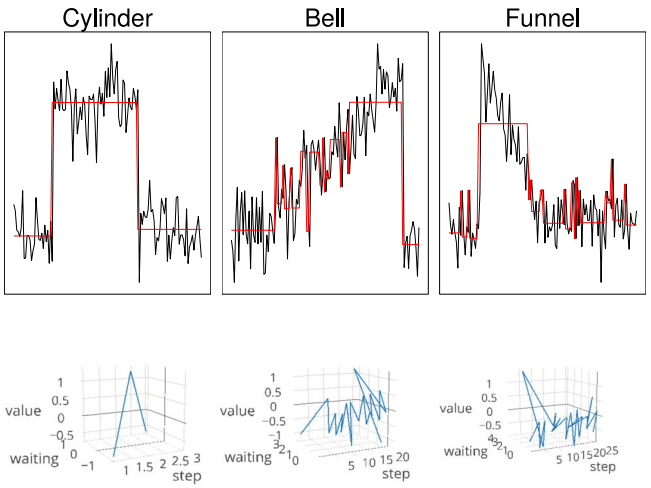

Fig. 1. Three classes of CBF time series dataset (cylinder, bell, and funnel): the raw data are shown in black, the segments are shown in red and CCA transformed time series in a 3-D space are shown in blue.

The performance gain is much more significant on long time series, where the running time is orders of magnitude faster than the state-of-the-art implementation of 1-NN DTW while scoring close or better in accuracy.

This paper is organized as follows. In Section II, background and related works are reviewed. Section III explains the CCA time series representation, 1-NN 3-D DTW, and its efficiency improvement techniques. Section IV showcases two case studies, and compares the accuracy and running time of 1-NN 3-D DTW against other state-of-the-art classifiers. Finally, Section V concludes this paper.

II. Background and Related Work

A. Definitions

Definition 1: A time series $T(t_1, t_2, \ldots t_n)$, is a sequence set of n real values recorded over time.

Defining a similarity/distance measure between two time series is at the core of most TSC techniques.

Definition 2: For two time series $X(x_1, x_2, \ldots x_n)$ and $Y(y_1, y_2, \ldots y_n)$, the ED is defined as

$$\mathrm{ED}(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}.$$

ED is the most basic and classic distance measure which is still competitive in some problems and domains [1], [14]. Fig. 2 illustrates an example of the ED visualization.

Generally, raw time series contain various sorts of distortion and noise, which makes their comparison difficult. The common distortions and invariances are amplitude and offset invariance, local scaling (warping) invariance, uniform scaling invariance, phase invariance, occlusion invariance and complexity-invariant distortions [15], [16].

Some invariances can be addressed and fixed at a preprocessing step. For example, normalizing raw data for the time

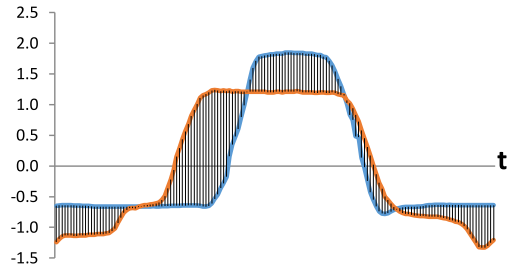

Fig. 2. Visualization of ED (linear alignment of the paired points) between two time series in orange and blue.

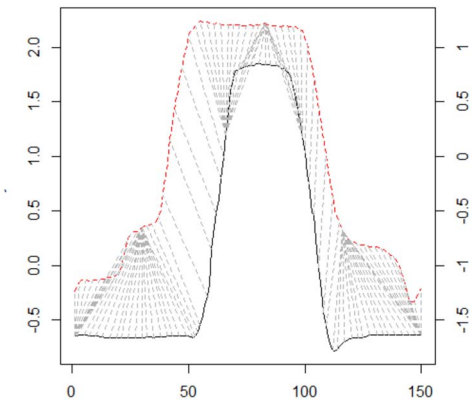

Fig. 3. Nonlinear realigning of the pairs of points in DTW between two time series in red and black.

series X can be used to prevent amplitude and offset invariance for this time series

$$\text{Z-norm}(X) = \left(x'_1, x'_2, \ldots, x'_n\right)$$

where $x'_i = [(x_i - \mu)/\sigma]$, μ is the mean of X and σ is the standard deviation of X.

In addition, a variety of distance function and data representation techniques have been developed to prevent different invariances. DTW is the most prominent example which addresses a local scaling invariance [2]–[5].

B. Brief Review of DTW

DTW allows nonlinear pairing of points in two sequence (Fig. 3). As a result, DTW allows the two compared time series to be of different lengths—which is a feature our proposed model relies on as we explain in the next section.

DTW computes the minimum distance of two time series by matching the points nonlinearly. To calculate the distance between the two time series of $X(x_1, x_2, \ldots, x_n)$ and $Y(y_1, y_2, \ldots, y_m)$ with corresponding length of n and m, DTW finds the minimum Minkowski distance with l_p norm over the allowed matching between points of the two time series. An n-by-m matrix can be constructed to cover all possible matching and alignments between points of x_i and y_j. Each element (i, j) of this matrix represents the alignment between x_i and y_j to which it can be considered as the distance between x_i and y_j, as well as defined by $d(x_i, y_j) = \|(x_i - y_j)\|_p$, where $\|\cdot\|_p$ represents the l_p norm. For example, for $p = 2$, $d(x_i, y_j) = |x_i - y_j|^2$.

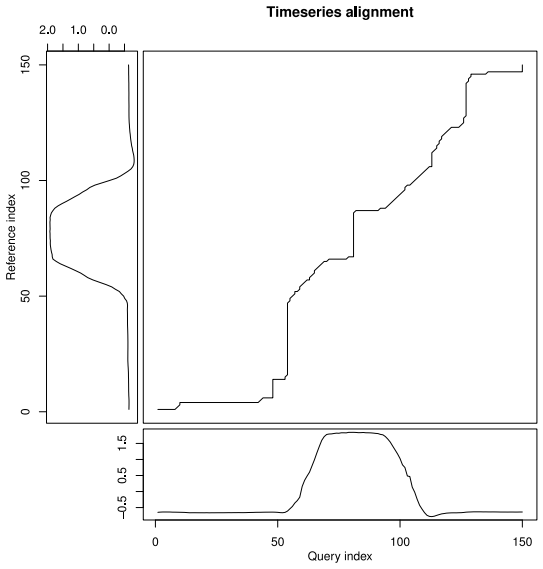

Fig. 4. Warping matrix and the optimal warping path for two time series (reference and query).

W is defined as a continuous path of elements in the matrix (alignments) from the beginning (x_1, y_1) to the ending (x_n, y_m) of time series. Thus, W is the mapping path between X and Y with the length S. The lth element of W is defined as $w_l = d(i, j)_l$; that is

$$W = w_1, w_2, \ldots, w_l, \ldots, w_S$$
$$\max(n, m) \leq S \leq n + m - 1.$$

DTW finds permissible matching sequences, paths, and then finds the optimal path with the minimum distance. Fig. 4 shows a graphical example of warping matrix and the optimal warping path.

Permissible n-by-m matching points, or alignment paths, are those that satisfy to the following conditions [17].

1) *Boundary Conditions:* The path should not skip a part at the beginning or ending of the sequence. The first and the last matches should be (x_1, y_1) and (x_n, y_m) meaning $w_1 = (1, 1)$ and $w_S = (n, m)$.
2) *Continuity Conditions:* There should be no jumps in the path. The previous step for each point (i, j) in the path must be $(i - 1,\ j)$, $(i,\ j - 1)$, or $(i - 1,\ j - 1)$.
3) *Monotonicity Conditions:* The warping path cannot go back in time. This means in each step of the warping path, i and j indexes stay the same or increase. Considering $w_k = (a, b)$, we have $w_{k-1} = (a', b')$ where $a - a' \gg 0$ and $b - b' \gg 0$.

DTW must find the path which minimizes the warping distance

$$\mathrm{DTW}_p(X, Y) = \min\left\{\sqrt[p]{\sum_{l=1}^{S} w_l}\right\}.$$

The optimal path can be found by using dynamic programming and the recursive equation given as

$$\mathrm{DTW}_p(X, Y) = \sqrt[p]{\gamma(i, j)}$$

	5	**3**	**2**	**2**	**4**
5	0	4	9	9	1
6	1	9	16	16	4
5	0	4	9	9	1
4	1	1	4	4	0
3	4	0	1	1	1

(a)

	5	**3**	**2**	**2**	**4**
5	0	4	13	22	23
6	1	9	20	29	26
5	1	5	14	23	24
4	2	2	6	10	10
3	6	2	3	4	5

(b)

Fig. 5. $X = \{5, 6, 5, 4, 3\}$ and $Y = \{5, 3, 2, 2, 4\}$. (a) Distance of aligned points of X and Y. (b) Cumulative distance matrix.

where $\gamma(i, j)$ is calculated by

$$\gamma(i, j) = |x_i - y_j|^p + \min\left\{\begin{matrix}\gamma(i-1, j-1) \\ \gamma(i, j-1) \\ \gamma(i-1, j)\end{matrix}\right\}$$
$$\gamma(0, 0) = 0,\ \ \gamma(0, \infty) = \infty,\ \ \gamma(\infty, 0) = \infty$$
$$(i = 1, 2, \ldots, n; j = 1, 2, \ldots, m).$$

The value of cell $\sqrt[p]{\gamma(n, m)}$ is the DTW distance of X and Y. Fig. 5 demonstrates an example of direct and cumulative distances in the DTW cost matrix. The optimal path is shown in pink cells and (setting $p = 2$), the DTW distance is the square root of 5.

Several extensions of DTW have been proposed in the literature. Using the distance between derivatives instead of point to point distance was proposed in [18]–[20]. This method is called derivative DTW (DDTW). DDTW transforms the points of the time series to a higher level feature.

The relative importance of the different phases of each alignment is considered in [21]. This method is called weighted DTW (WDTW).

In [21], the WDTW concept is combined with DDTW and it is called weighted DDTW (WDDTW). Different variations of DTW often outperform each other in different application domains.

It has been shown that DTW distance using 1-NN is one of the most robust classification techniques which is hard to beat [6], [22].

C. Improving the Efficiency of DTW

1) Global Warping Constraints: The purpose of warping constraints is to limit the wandering distance of the search paths by defining a band (warping scope) in the DTW warping matrix. This band does not necessarily retrieve the optimal path, but it is expected to provide a decent path. It is expected that a decent path is close to the matrix diagonal, and warping constraints are defined around the diagonal. The Sakoe–Chiba band [2] and the Itakura parallelogram [23] are the most common constraint methods. Global constraints reduce the computation cost however, the classification time and accuracy depend on a parameter (r), where r is the allowed range of warping from the diagonal of the matrix. Fig. 6 shows a graphical example of constraints for the Sakoe–Chiba band and the Itakura parallelogram. Cross validation on a training dataset is used to learn the best constraint (r). The classifier

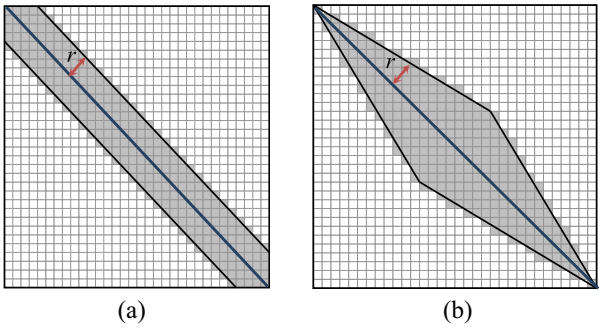

Fig. 6. Example of global constraints. (a) Sakoe–Chiba band. (b) Itakura parallelogram.

which uses 1-NN DTW with a learned warping constraint on cross validation increases accuracy in some problems [24], but training process is time expensive.

2) Lower Bounding: The idea of lower bounding is to do the expensive full calculation of optimal path in DTW matrix, only when it is absolutely unavoidable. Applying lower bounding makes the nearest neighbor search faster by pruning off unhopeful candidates [6], [25]. LB_Kim, LB_Yi, and LB_Keogh are the most common lower bound for time TSC which are graphically presented on an example for candidate C and query Q time series in Fig. 7.

LB_Kim [26] is the total summation of the squared differences between the two sequence's first (A), last (D), minimum (B), and maximum points (C). These distances are presented in Fig. 7(a).

LB_Yi [27] is calculated as follows. The sum of the squared length of gray lines in Fig. 7(b) represents LB_Yi.

LB_Keogh bound is explained in detail in [6], [26], and [27]. LB_Keogh first defines U and L as the upper and the lower bound of query time series as.

$U_i = \max(q_{i-r}, \ldots, q_{i+r})$ and $L_i = \min(q_{i-r}, \ldots, q_{i+r})$, where r is the allowed warping range. Then LB_Keogh is formulated as

$$\text{LB\_Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - U_i) & \text{if } c_i > U_i \\ (c_i - L_i) & \text{if } c_i < L_i \\ 0 & \end{cases}}.$$

The sum of the squared length of gray lines in Fig. 7(c) represents LB_Keogh.

A review and evaluation of lower bounding methods used for DTW is presented in [28].

3) Early Abandoning: Early abandoning is another method to accelerate 1-NN DTW calculations (as well as the other distance-based classifiers such as 1-NN ED). This is done by calculating partial distance accumulation for a candidate sequence and comparing it to a threshold which is the best-so-far candidate. If partial accumulation goes beyond the threshold (best-so-far) at any time, the calculation is terminated, and the candidate is discarded (Fig. 8). This technique reduces redundant calculations in the similarity search. Early abandoning was proposed and applied in [4] and [25] to accelerate the calculation of ED, as well as in [5] and [29], to make 1-NN DTW calculations faster.

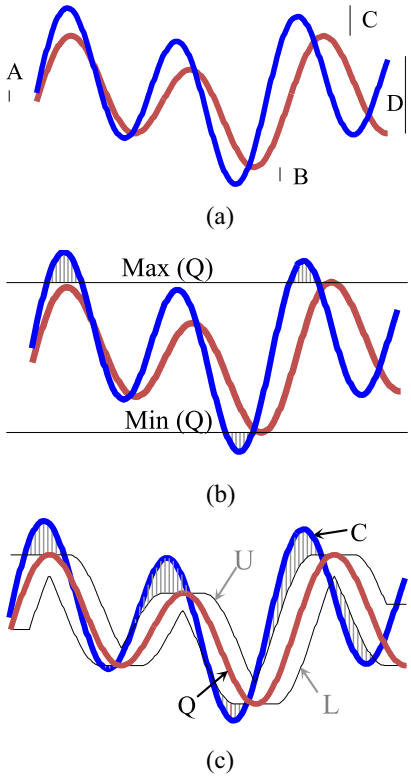

Fig. 7. Example of lower bounding techniques. (a) LB_Kim. (b) LB_Yi. (c) LB_Keogh. Used with permission from E. Keogh [25].

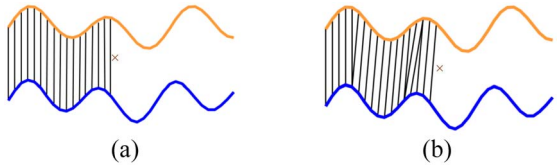

Fig. 8. Early abandoning technique. Early abandoning on (a) ED and (b) DTW.

4) Time Series Representation: Each value in a time series can be considered as a dimension. Time series are typically high-dimensional data (i.e., long length). Time series representations project points into a new space where it can be processed more efficiently. Time series representation techniques (also called dimensionality reduction techniques) aim to reduce the dimensionality (length) of time series by extracting features from the raw data. Time series representation should follow the following goals [25]: dimensionality reduction, storage reduction, noise removal, computational costs reduction, and minimizing information loss [the reconstruction error (RE)]. A review of representation methods, as well as distance-based classification techniques, is presented in [6]. Time series representation in the literature can be mainly divided into two groups of nonsymbolic and symbolic representations. Some of the nonsymbolic representation methods are listed in [6], including discrete Fourier transformation [30], single value decomposition [31], discrete wavelet transformation [32], piecewise aggregate approximation (PAA) [33], [34], adaptive piecewise constant approximation (APCA) [35], and Chebyshev polynomials [36].

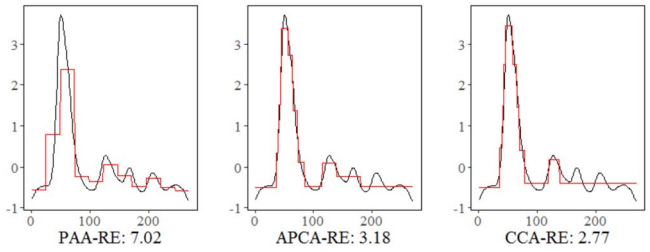

Fig. 9. Comparison of approximation methods and their reconstruction error. The raw data is in black and approximation is in red.

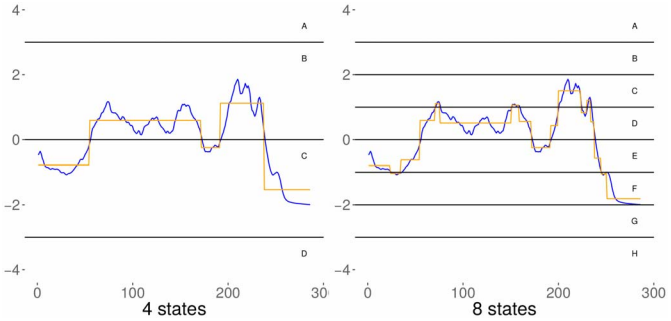

Fig. 10. Time series approximation. The raw time series is presented in blue and its approximation in orange.

Reference [37] lists some of the symbolic representation methods, including shape description alphabet [38], interactive matching of patterns with advanced constraints in time series databases [39], clipping [40], persist [41], piecewise vector quantized approximation [42], and symbolic aggregate approximation [43] and its extensions [44], [45]. More recent representation methods focus on histogram-based representation for time series data (especially for long time series). These methods consider the distance/similarity based on the substructures (higher-level features) which are extracted from a time series. Bag-of-patterns [46], time series based on a bag-of-features representation [47], and bag-of-symbolic Fourier approximation (SFA)-symbols (BOSS) [48] are some examples of this type of representations. BOSS VS [49] also uses bag-of-SFA in its representation and it is much faster than BOSS.

This paper proposes a new numeric representation of time series denoted as CCA. CCA is discussed in detail in Section III. An example is presented in Fig. 9 to compare CCA with PAA and APCA. ED between the raw time series and the approximation is the reconstruction error (RE). In this example, it can be seen that after approximation with the same compression ratio (14 segments), CCA has the lowest RE. The output of CCA looks graphically similar to the output of PAA and APCA. A time series is divided into flat segments. In PAA, the length of the segments must be same, but APCA and CCA relax this constraint. The value of each segment is the mean of all the points which fall within that segment.

III. Three-Dimensional DTW

A. Control Chart Approximation: Time Series Representation

In this section, we present our numeric representation of time series CCA.

Given a time series $X = (x_1, x_2, \ldots x_n)$, we want to produce CCA representation, which is presented as

$$C_X = \{(t_1, v_1, u_1), \ldots (t_k, v_k, u_k)\}$$

where the tuple (t_i, v_i, u_i) defines the ith segment, t_i is the start time, v_i is the value, u_i is the duration of the segment in the time series, and k is the total count of segments. Each segment contains a continuous sequence of data points in the time series whose values fall into the same range (state). The segments embody the continuous state change of the time series values. Similar to control limits in control charts, certain thresholds need to be defined in order to detect such state changes over time. A segment starts when a data point passes a threshold and its time value marks the beginning of a new segment (t_i). We call these special points—at the beginning and end of the segments—the *jump points*. The value v_i of each segment is defined as the mean of the raw data points falling in that segment. In our experiments mean performed better than other summary statistics such as median. The value and duration of each segment are defined as follows:

$$u_i = t_{i+1} - t_i \tag{1}$$

$$v_i = \left(\sum_{j=t_i}^{t_{i+1}} x_j\right)/u_i. \tag{2}$$

Fig. 10(left) shows a standardized control chart with $\pm 3\sigma$ control limits used to partition a time series into four states (A, B, C, and D) and results in a CCA representation of the time series with five segments. In this example, the standard control limits of ± 3 are used, and the range between the two control limits are divided to 2. For more exact approximations, the same concept can be used by dividing the space between two control limits into more partitions. In control charts, K represents the distance (in the terms of standard deviation) between the control limits and the center line (control lines: $\mu \pm K\sigma$). Typically, three-sigma limits are employed (i.e., K is usually set to 3). In order to provide more flexibility for a user, in our algorithm the parameter K is defined and used but its default value is 3.

This number of partitions between control limits in CCA is controlled by the parameter s, which is the distance between the thresholds within the upper and lower control limits. In Fig. 10(right), the distance between controls limits, s is set to 1, which creates eight states. This results in a more accurate approximation; however, it increases the number of produced segments by 8 (17 segments). Algorithm 1, represents how CCA algorithm detects the jump points and produce the segments in detail. The next step in CCA is normalizing the segments durations to have the same the scale as segments values. Thereby, each segment duration (u_i) is z-normalized by relative to mean (μ_u) and standard deviation (σ_u) of all the segments durations. We denote the normalized duration of each segment as its *waiting time* (w_i)

$$w_i = (u_i - \mu_u)/\sigma_u. \tag{3}$$

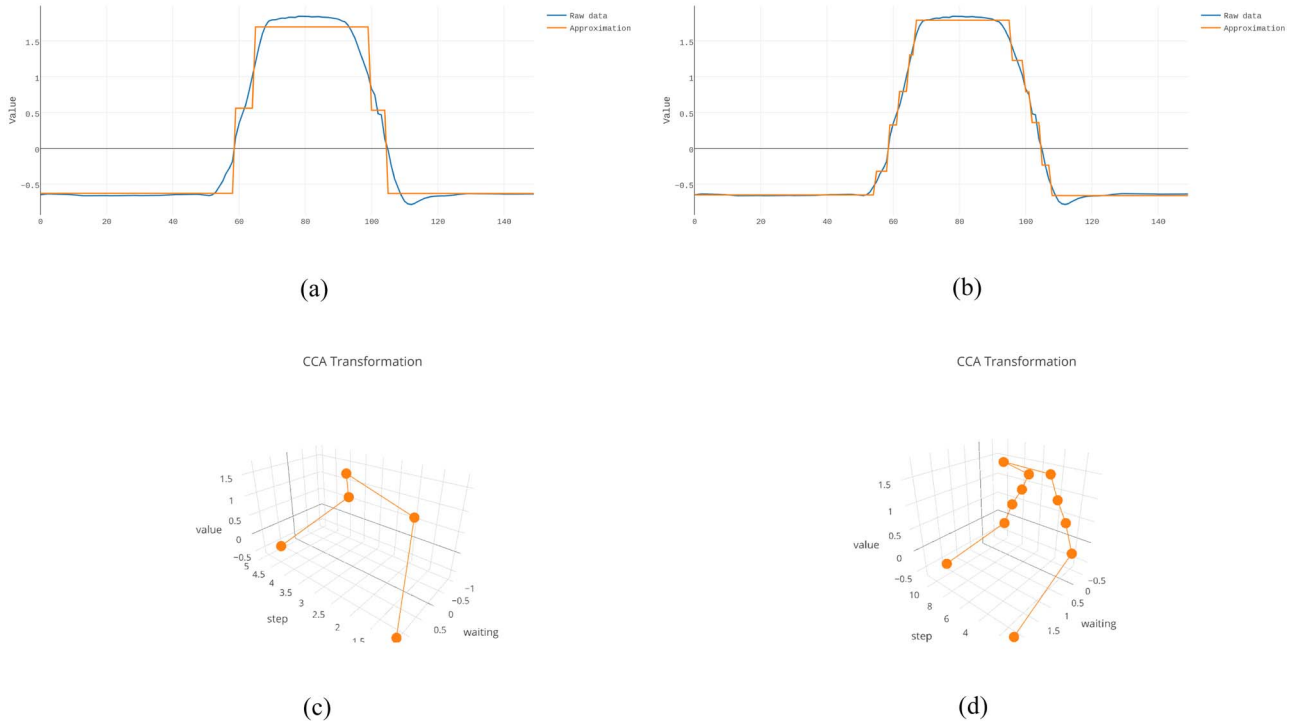

Fig. 11. CCA examples. (a) $s = 1$, approximation. (b) $s = 0.5$, approximation. (c) $s = 1$, CCA transformation. (d) $s = 0.5$, CCA transformation.

Algorithm 1 CCA Algorithm With z-Normalized Time Series X, Control Limit K, and Distance Parameter s

function COMPUTE-CCA($X = \{x_1, \ldots x_n\}, K, s$)
  $j \leftarrow 1,\ t_j \leftarrow 0$
  $C_X \leftarrow \phi$
  if $n = 1$ **then return** $C_X = \{(0, x_1, 1)\}$
  end if
  for $i \leftarrow 1 \ldots n$ **do**
    Find $r_i \in \{(-\infty, -K], (-K, -K + s], \ldots, (\ldots, K], (K, \infty)\}$ such that $x_i \in r_i$.
    if $(i > 1\ \&\ r_i \neq r_{i-1})$ **or** $i = n$ **then**
      $t_{j+1} \leftarrow i$ ▷ start of the $(j+1)$th segment
      $u_j = t_{j+1} - t_j$
      $v_j = (\sum_{i=t_j}^{t_{j+1}} x_i)/u_j$
      $C_X \leftarrow C_X \cup (t_j, v_j, u_j)$
    end if
  end for
  return C_X
end function

The final result of CCA are sequential normalized segment values (v) and waiting times (w)

$$C_X = \{(t_1, v_1, w_1), \ldots (t_k, v_k, w_k)\}.$$

The graphical presentation of this output can be shown in 3-D graphs. In Fig. 11, CCA time series representation is presented with two different s values. In the upper left graph, the threshold distance parameter is set to 1 ($s = 1$); it has created five segments. Then, the segments durations are normalized. The lower left graph shows the 3-D representation of the final CCA segments. In the upper right graph, the CCA representation of the same time series with $s = 0.5$ shows 11 resulting segments. The lower right graph shows the final values after normalization in 3-D space.

TABLE I
SAMPLE OF POSSIBLE VALUES FOR s AND NUMBER OF STATES

Number of States	s	Number of States	s
4	3	11	0.67
5	2	12	0.60
6	1.5	13	0.55
7	1.2	14	0.50
8	1	15	0.46
9	0.86	16	0.43
10	0.75	...	...

In CCA algorithm, the range of the distance between control limits (s) is from 3 to ε. The lower s will create more states and expectedly more segments in time series transformation. In Table I, some sample of s values along with the number of states are shown.

When s is extremely small (ε), the smallest change in the value of each point compared to its predecessor will make that point a jump point. In this special case, with the exception of constant remaining points every point in the time series will become a jump point, and the resulting CCA will be very close to, or the same as, the original time series. In the process of TSC, the best choice of s is dependent on the structure of data. In order to find optimal s for a specific time series data, cross validation can be used to learn the best choice of s for that problem.

B. Three-Dimensional DTW Distance

Suppose we have two time series C and Q, and we want to measure the distance of these time series after CCA representation (transformation). In order to calculate this distance, we propose a new version of DTW called 3-D DTW which is an adaptation of traditional DTW on more axis. Suppose the transformed time series are

$$\text{Transformed } C = \left\{\left(v_1^c, w_1^c\right), \ldots, \left(v_i^c, w_i^c\right), \ldots, \left(v_m^c, w_m^c\right)\right\}$$
$$\text{Transformed } Q = \left\{\left(v_1^q, w_1^q\right), \ldots, \left(v_j^q, w_j^q\right), \ldots, \left(v_n^q, w_n^q\right)\right\}.$$

Note that we eliminated the t_i values in the tuples as they are not considered as a dimension for DTW calculation. Similar to conventional DTW, the segments are ordered along the matrix based on their occurrence in time, and time distance is inherent in DTW path calculation. Hence, we do not consider the starting time of each segment in the distance calculation. Note that even if two time series are from the same problem, and if the lengths of the original time series are equal, after CCA transformation they will not necessarily have equal lengths (the same number of segments/steps). First, we define a distance matrix between every two segment. Each element of the matrix is defined as

$$D(i,j) = \left(v_i^c - v_j^q\right)^2 + \left(w_i^c - w_j^q\right)^2 \tag{4}$$

where $i \in [1, n]$ and $j \in [1, m]$.

Then, a matrix is constructed based on pairwise distance $D(i, j)$ of segments starting from $\gamma(1, 1)$ to $\gamma(n, m)$ similar to conventional DTW.

$\gamma(i, j)$ is calculated by

$$\gamma(i,j) = \left(v_i^c - v_j^q\right)^2 + \left(w_i^c - w_j^q\right)^2 + \min\begin{Bmatrix}\gamma(i-1,j-1)\\ \gamma(i,j-1)\\ \gamma(i-1,j)\end{Bmatrix}$$
$$\gamma(0,0) = 0,\ \ \gamma(0,\infty) = \infty, \gamma(\infty,0) = \infty,$$
$$(i = 1, 2, \ldots, n; j = 1, 2, \ldots, m). \tag{5}$$

After calculating all elements of the matrix, the last element, $\gamma(n, m)$ is the square of distance between transformed C and Q.

3-D DTW follows the same conditions (i.e., boundary conditions, continuity, and monotonicity) and the same algorithm as regular DTW, but the distance matrix is changed according to the structure of a transformed time series. Calculation of 3-D DTW distance is presented in Algorithm 2. We demonstrate an example here in order to illustrate the technique.

Example 1: We randomly selected two time series from the *Gun_Point* train dataset from [50]. The original time series are shown in Fig. 12. We are interested in calculating their 3-D DTW distance after applying the CCA transformation. Setting s equal to 1, CCA results in the following segment list:

$$\text{Transformed } Q = \{(0.56, 1.28), (0.59, -0.67), (1.51, -0.76), (2.03, -0.09), (1.52, -0.67), (0.5, -0.81), (-0.6, 1.81)\}$$
$$\text{Transformed } C = \{(-0.66, 1.38), (0.48, -0.76), (1.66, 0.71), (0.47, -0.67), (-0.63, 1.43)\}.$$

Algorithm 2 3-D DTW Algorithm With Early Abandoning

function 3D-DTW($\mathbf{Q} = \{(v_1^q, w_1^q), \ldots, (v_m^q, w_m^q)\}$, $\mathbf{C} = \{(v_1^c, w_1^c), \ldots, (v_n^c, w_n^c)\}$, ***BSF***: best so far (closest query))
  $E[0, 0] \leftarrow (v_1^q - v_1^c)^2 + (w_1^q - w_1^c)^2$ ▷ Initialize first cell
  for $j \leftarrow 1 \ldots m$ **do** ▷ Initialize first row
    $E[1, j] \leftarrow E[1, j-1] + (v_1^c - v_j^q)^2 + (w_1^c - w_j^q)^2$
  end for
  for $i \leftarrow 1 \ldots n$ **do** ▷ Initialize first column
    $E[i, 1] \leftarrow E[i-1, 1] + (v_i^c - v_1^q)^2 + (w_i^c - w_1^q)^2$
  end for
  for $i = 1 \ldots n$ **do**
    for $j = 1 \ldots m$ **do**
      $k \leftarrow \min\begin{Bmatrix}E(i-1,j-1)\\ E(i,j-1)\\ E(i-1,j)\end{Bmatrix}$
      $E[i, j] \leftarrow k + (v_i^c - v_j^q)^2 + (w_i^c - w_j^q)^2$
    end for
    min_cost $\leftarrow \min(E[i, .])$
    if min_cost $>$ ***BSF*** **then**
      return ∞
    end if
  end for
  return $E[n, m]$
end function

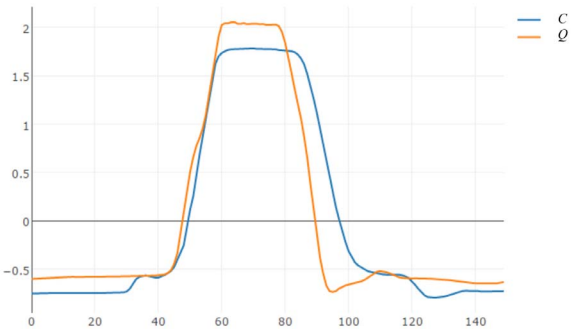

Fig. 12. Original time series of C and Q.

Fig. 13 shows these time series after the transformation in a 3-D space. Fig. 14 shows the matrix for calculating the distance of two time series after CCA transformation, and the alignment between points.

The first element $\gamma(1, 1)$ in this matrix is calculated as

$$\gamma(1,1) = (v_1^c - v_1^q)^2 + (w_1^c - w_1^q)^2$$
$$= (-0.66 - (-0.56))^2 + (1.38 - 1.28)^2$$

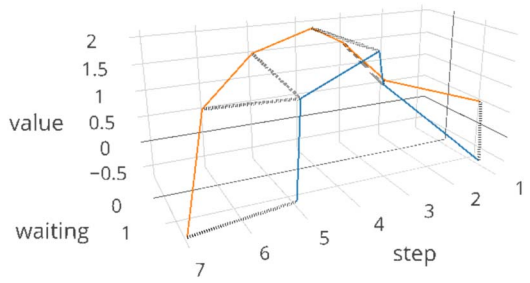

Fig. 13. CCA transformed time series of C (in blue) and Q (in orange) and their 3-D-DTW alignment.

C \ Q	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[1]	0.020	5.750	15.000	24.410	33.320	39.440	39.620
[2]	5.280	0.040	1.090	3.950	5.040	5.040	12.810
[3]	10.530	3.090	2.240	1.880	3.800	7.480	11.370
[4]	15.380	3.110	3.330	4.670	3.000	3.020	10.270
[5]	15.400	8.970	12.450	12.740	12.000	9.260	3.160

Fig. 14. Three-dimensional DTW cost matrix after CCA transformation of time series.

which is 0.02. Then the first row is calculated as

$$\gamma(1,j) = \left(v_1^c - v_j^q\right)^2 + \left(w_1^c - w_j^q\right)^2 + \gamma(1,j-1), \quad j = 1, 2, \ldots, m. \tag{6}$$

For example

$$\begin{aligned}\gamma(1,2) &= \left(v_1^c - v_2^q\right)^2 + \left(w_1^c - w_2^q\right)^2 + \gamma(1,1) \\ &= ((-0.66) - 0.59)^2 + (1.38 - (-0.67))^2 + 0.02\end{aligned}$$

which is 5.750.

Then, the first column is calculated as

$$\gamma(i,1) = \left(v_i^2 - v_1^q\right)^2 + \left(w_1^c - w_2^q\right)^2 + \gamma(i-1,1), \quad i = 1, 2, \ldots, n. \tag{7}$$

For example

$$\begin{aligned}\gamma(1,2) &= \left(v_1^c - v_2^q\right)^2 + \left(w_1^c - w_2^q\right)^2 + \gamma(1,1) \\ &= ((-0.66) - 0.59)^2 + (1.38 - (-0.67))^2 + 0.02\end{aligned}$$

which is 5.280.

The remaining matrix elements, from [2, 2] to [5, 7], are calculated using (5). For example

$$\begin{aligned}\gamma(2,2) &= \left(v_2^c - v_2^q\right)^2 + \left(w_2^c - w_2^q\right)^2 \\ &\quad + \min\{\gamma(1,1), \gamma(2,1), \gamma(1,2)\} \\ &= (0.48 - 0.59)^2 + ((-0.76) - (-0.67))^2 \\ &\quad + \min\{0.02, 5.28, 5.75\}\end{aligned}$$

which is 0.040.

The value of the last element of the matrix, i.e., $\gamma(5,7)$, shows the squared distance between these transformed time series. The pink cells in the matrix show the optimal selected path and the aligned points. The aligned points are connected together with gray dashed lines in Fig. 13.

One of the advantages of 3-D DTW is that all of the efficiency improvement techniques (such as the squared distance, warping constrain, lower bounding, early abandoning, etc.) in regular DTW, as well as DTW extensions techniques (such as DDTW, WDTW, WDDTW, etc.), are applicable on 3-D DTW after minor changes in the algorithm. In the following section, we show how to apply lower bounding and early abandoning in 3-D DTW.

C. Early Abandoning on 3-D DTW

Early abandoning in 3-D DTW works similar to Early abandoning in regular DTW [29]. The objective is to eliminate the search if the accumulative 3-D distances between two CCA transformed time series is greater than a predefined threshold (best-so-far). Algorithm 2, shows how to calculate 3-D DTW distance along with early abandoning.

IV. Experiments

A recent study on all rival TSC techniques proposed in last five years, on 85 datasets (100 resampling experiments on each dataset), concluded that DTW is hard to beat, and where it is beaten in accuracy the benefit is small [22]. DTW is significantly more accurate than benchmark classifiers such as naive Bayes, C45, logistic regression, ED, linear and quadratic SVM, and Bayesian networks. Significantly more accurate classifiers than DTW either exhibits small improvement margin or huge tradeoff in runtime. The study shows that COTE [51], shapelet transform [52], [53], BOSS [48], and elastic ensemble [7] are the best classifiers beating DTW. The most accurate one, COTE is an ensemble of 35 classifiers with a runtime complexity bounded by shapelet transform $O(n^2m^2)$ and as high as $O(n^3)$ [22].

The recently proposed BOSS VS method [49] is much faster than the state-of-the-art implementation of DTW, SVM quadratic kernel, and BOSS with almost same accuracy rate as BOSS. BOSS VS is the state-of-the-art classifier considering both accuracy and runtime, and is one of our baselines.

We compared the efficiency and accuracy of 3-D DTW (3-D DTW—Algorithm 2) with the state-of-the-art implementation of DTW [13] (using efficiency improvement methods mentioned in Section II-C) and also BOSS VS [49] as our baselines. We did not include direct comparison with other classifiers, because BOSS VS is only second to COTE, shapelet transform, BOSS, and Elastic Ensemble on average accuracy [22] and faster than all of them [49]. Thus, it makes a good representative baseline for our method which favors efficiency over accuracy. Also the comparison with exact DTW, should exhibit the effect of our approximation method on accuracy.

Three-dimensional DTW performance is evaluated using the UCR public benchmark datasets archive [50]. In the UCR TSC archive, there are 85 time series from different domains and problems. Each dataset in this archive comes in two parts, a train partition and a test partition. The train partition is used in the model building process, and the test partition is used for measuring the classification accuracy. In all time-series

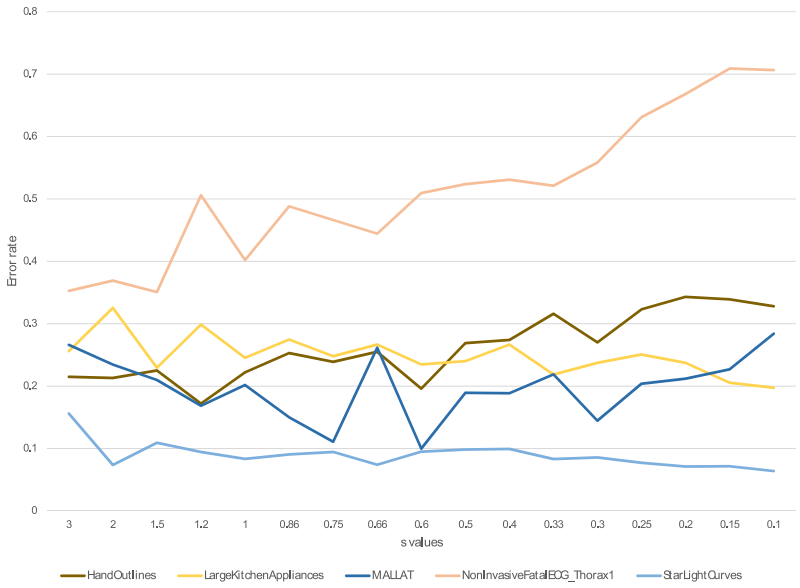

Fig. 15. Effect of s values on some example datasets.

datasets, the data is z-normalized prior to the experiment to have a mean of 0 and variance of 1.

One challenge by using the CCA is choosing the right s for approximation. Our experiments with various range of s values over all the UCR datasets, could not verify a meaningful trend of accuracy relative to s consistent with all the datasets. Therefore, coming up with a right set of candidates that can be used for efficient cross validation of s on a general dataset is not trivial. However, with the right selection of candidates, based on the structure of the time series in question, cross validation can be made efficient and effective when improving accuracy is of higher concern.

Our results showed that, on most time series, choosing an s value, which creates an even number of partitions (with the middle threshold falling on zero line), generally obtain better results in terms of accuracy. On a large scale, some particular s values were observed to perform better on average. In Fig. 15, the effect of s values on accuracy (the error rate) for some example datasets is presented. With an exception of few datasets, they scored competitively well, and with little sacrifice on accuracy, while maintaining the two order of magnitude performance gain compared to other state-of-the-art methods.

In this experiment, we show the classification result with $s = 0.6$ and show that even without prior knowledge about the structure of the data and suitable s candidates, 1-NN 3-D DTW with a default value for s will work well in terms of accuracy and running time, especially on the long time series. For parameter K, we used the default value ($K = 3$) but a user can change it according to the specification of any problem.

Our homepage[1] reports all raw numbers and contains the C++ source code for 1-NN 3-D DTW. All experiments were performed on an Intel Xeon E5-2620 (2.00 GHz) machine with single core setting.

[1]The 3-D-DTW results http://prominent.mie.uic.edu/3d-dtw (2017).

A. Case Study

For the purpose of the case study, two long and large time series datasets are selected from the UCR archive. The purpose is to demonstrate the very efficient running time of 1-NN 3-D DTW classification, as well as its high accuracy on long time series.

1) Phoneme: Phonemes are the smallest units of intelligible sound produced by a human being, and phonetic spelling is the sequence of phonemes that a word comprises. The original phoneme dataset is presented in [54], which has 370 000 phonemes and massive amounts of noise. It is one of the largest single dimension TSC dataset. The data in the original dataset are of unequal length. In UCR archive [50], the data is interpolated to make all time series equal length (1024), and contains a subset of 214 train and 1896 test samples from the original dataset. Having as much as 39 classes, phoneme time series are very challenging to classify. Three sample time series with different classes are randomly selected from the dataset, which are shown in Fig. 16. The black graph shows the raw data, and the red graph shows the jump points and steps after CCA representation and data reduction. From the left graph to the right graph, respectively, 1024 points of the raw data are reduced to 160, 17, and 310 points (segments) by CCA transformation, respectively.

The classification accuracy and process time of 1-NN 3-D DTW are compared with state-of-the-art implementation of 1-NN DTW and BOSS VS in Table II. 3-D DTW has the highest accuracy with 25.2%. Classification of this problem by 1-NN DTW on a single core machine takes around 45 min, while 1-NN 3-D DTW prediction is done in less than 5 min (close to BOSS VS) with better accuracy.

2) Starlight Curves: Starlight curves is the largest dataset available in the UCR time series archive [50]. It contains 1000 train and 8236 test records with a length of 1024 in three types of classes (star objects): 1) eclipsed binaries; 2) Cepheid; and 3) RR Lyrae variables. Fig. 17 shows sample graphs in

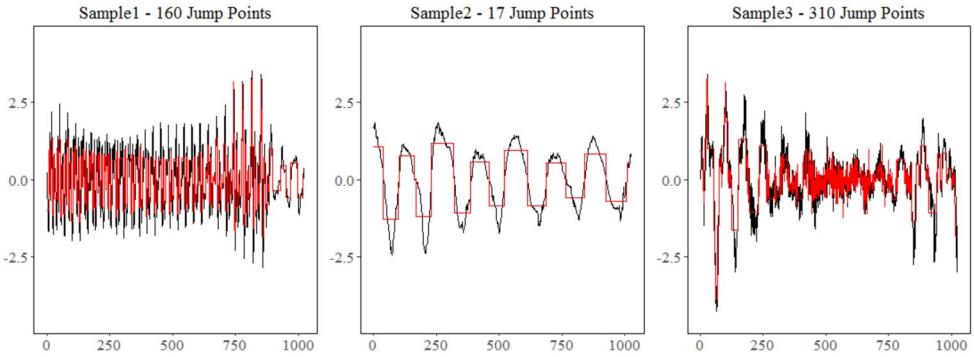

Fig. 16. Graphical presentation of three samples of phoneme datasets with three different classes. The raw data is presented in black line and the approximation in red line.

TABLE II
CLASSIFICATION ACCURACY AND TIME COMPARISON OF PHONEME

Classifier name	Accuracy	Process time in s
1-NN DTW	22.8%	2657.32
BOSS VS	17.5%	183.17
1-NN 3D DTW	25.2%	283.61

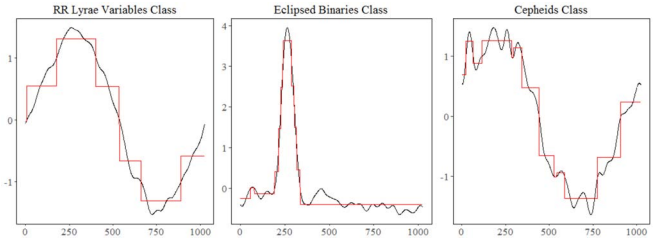

Fig. 17. Graphical presentation of three samples of starlight curves datasets with three different classes. The raw data is presented in black line and the approximation in red line.

TABLE III
CLASSIFICATION ACCURACY AND TIME COMPARISON OF STARLIGHT

Classifier name	Accuracy	Process time in s
1-NN DTW	90.7%	7773.95
BOSS VS	90.7%	165.32
1-NN 3D DTW	90.5%	7.97

each class. The difficult part of classification is detecting RR Lyrae variables from Cepheids due to their similar shapes. Since there is no sudden value fluctuation in this problem, the CCA representation is very effective, and it can significantly reduce the number of points after transformation. This will result faster classification process. Fig. 17 shows, from left to right, reduction of 1024 raw data points to 7, 10, and 12 points (segments) by CCA transformation, respectively.

Table III compares the classification accuracy and time among classifiers. 1-NN 3-D DTW accuracy is nearly the same as 1-NN DTW and BOSS VS while significantly outperforming these classifiers in time. While the state-of-the-art 1-NN DTW takes 2 h, and BOSS VS takes 3 min to process the classification of this problem, 1-NN 3-D DTW does the same job in less than 8 s on the same machine with the same accuracy. Whereas BOSS VS is among the fastest classifiers and is significantly faster than 1-NN DTW CV, 1-NN DTW, SVM, and random forest [49], it is outperformed by 1-NN 3-D DTW by about 20 times.

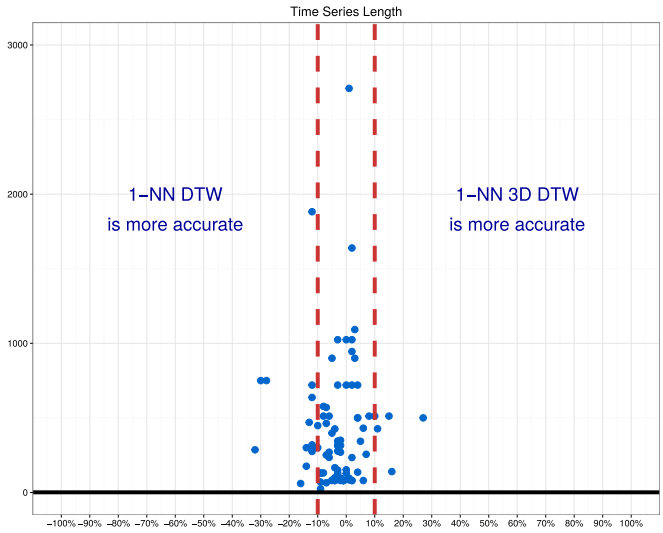

Fig. 18. Difference in accuracy between 1-NN 3-D DTW and 1-NN DTW classification on all 85 time series datasets in UCR archive with different time series lengths.

B. Classification Accuracy

Fig. 18 shows the comparison of 1-NN 3-D DTW to 1-NN DTW classification accuracy for all datasets in UCR archive. 1-NN DTW is frequently used as the benchmark for comparison [6], [13]. In this figure, each point represents one dataset. The x-axis is the difference between 1-NN 3-D DTW and 1-NN DTW classification accuracies, and the y-axis is the length of time series. The points on the right side of the y-axis are the datasets which 1-NN 3-D DTW performs better, and the points on the left side of the y-axis are the datasets which 1-NN DTW does better in terms of classification accuracy.

Fig. 18 shows that 1-NN 3-D DTW performance in term of accuracy is competitive to 1-NN DTW. The majority of datasets fall within a range of -10% to 10%. However, the main advantage of our method is in accelerating the classification process rather than improving the accuracy.

The classification time becomes a more significant factor for long time series, and it is where most current classifiers

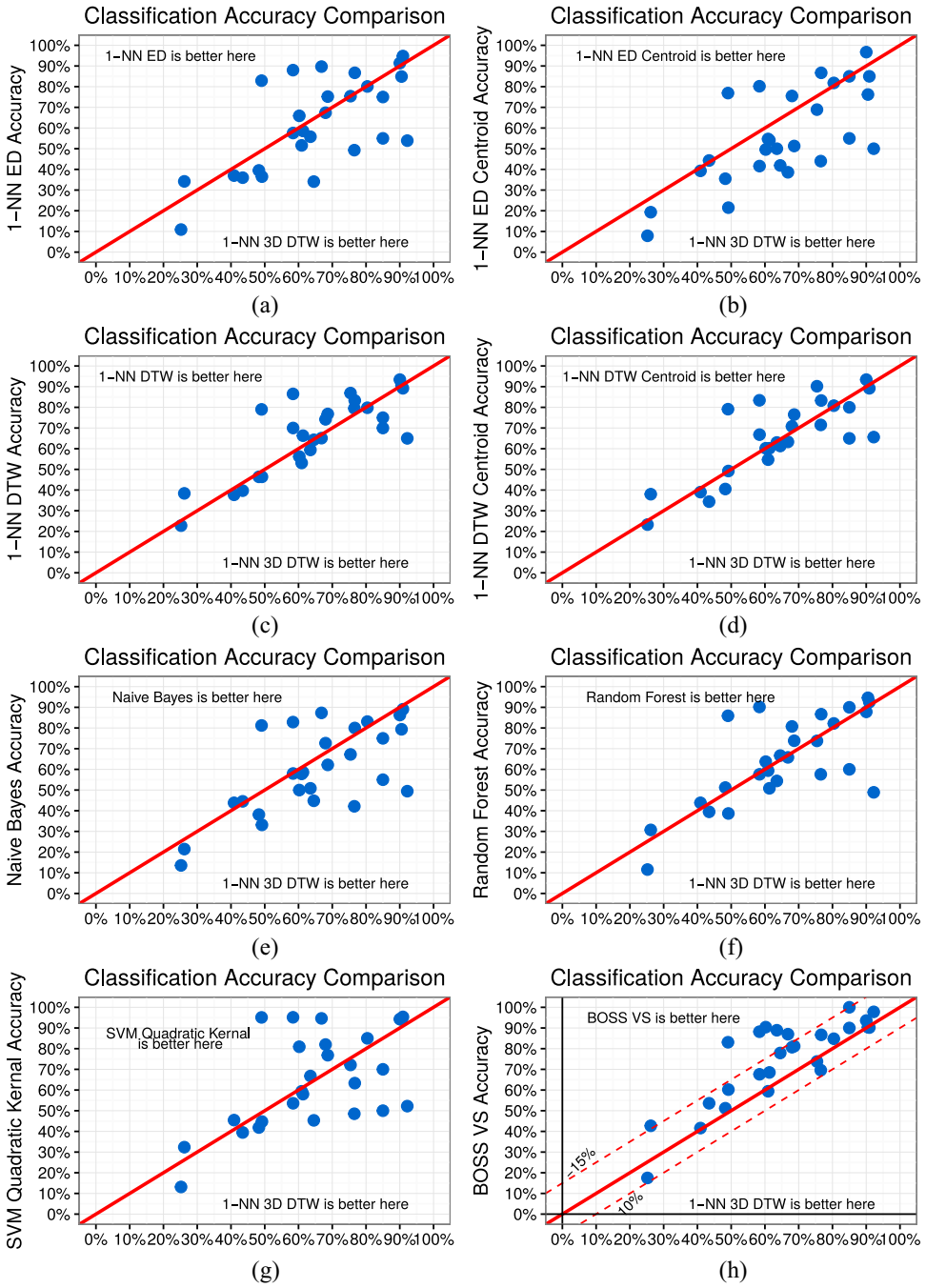

Fig. 19. Pairwise classification accuracy comparison of 1-NN 3-D DTW with (a) 1-NN ED, (b) 1-NN ED centroid, (c) 1-NN DTW, (d) 1-NN DTW centroid (8), (e) naive Bayes, (f) random forest, (g) SVM quadratic kernel, and (h) BOSS VS on all long datasets.

fall short. In many use cases a faster prediction is much more important than a nonsignificant increase in accuracy. For that reason, we are interested to see how our method works on long time series (500 points and above). There are 28 datasets in UCR archive that have more than 500 points in length. Table IV shows the list of these datasets along with their lengths.

First, we compare the accuracy of 1-NN 3-D DTW on long time series with 1-NN ED and 1-NN DTW which are the most common time series classifiers. We also review the accuracy of 1-NN ED Centroid and 1-NN DTW Centroid which are considered to be very fast classifiers.

Fig. 19 shows pairwise classification accuracies comparison between 1-NN 3-D DTW and Fig. 19(a) 1-NN ED, Fig. 19(b) 1-NN ED centroid, Fig. 19(c) 1-NN DTW, Fig. 19(d) 1-NN DTW centroid (8), Fig. 19(e) naive Bayes, Fig. 19(f) random forest, Fig. 19(g) SVM quadratic kernel, and Fig. 19(h) BOSS VS. Each long time series dataset is represented by one blue point. Each point under the straight red line indicates that 1-NN 3-D DTW is more accurate. The 1-NN 3-D DTW offers a significantly higher accuracy than 1-NN ED, 1-NN ED centroid, and 1-NN DTW centroid. Our 1-NN 3-D DTW also outperforms the accuracy of 1-NN DTW for more than half of long time series datasets (15 win/13 lose). Despite the loss

TABLE IV
LIST OF LONG TIME SERIES DATASETS IN URC ARCHIVE

Dataset name	Length	Dataset name	Length
BeetleFly	512	Mallat	1024
BirdChicken	512	NonInvasiveFatalECG1	750
Car	577	NonInvasiveFatalECG2	750
CinC_ECG_torso	1639	OliveOil	570
Computers	720	Phoneme	1024
Earthquakes	512	RefrigerationDevices	720
FordA	500	ScreenType	720
FordB	500	ShapeletSim	500
HandOutlines	2709	ShapesAll	512
Haptics	1092	SmallKitchenAppliances	720
Herring	512	StarlightCurves	1024
InlineSkate	1882	UWaveGestureLibraryAll	945
LargeKitchenAppliances	720	Worms	900
Lighting2	637	WormsTwoClass	900

of information by approximation, 3-D DTW exhibits superiority through the effect of noise and redundancy reduction. Although the information loss caused by approximation could have an adverse impact on accuracy, reducing noise and redundancy on the other hand is an improving factor. This result is consistent with other studies such as effect of noise reduction in BOSS [48], and also removing redundant data by discriminative subsequences in time series shapelets [52], [53]. In the next section, we show that 1-NN 3-D DTW is orders of magnitude faster than 1-NN DTW.

The 1-NN 3-D DTW classifier outperforms naive Bayes, random forest, and SVM. On most long time series datasets, the accuracy difference of 1-NN 3-D DTW to BOSS VS is within -15% to 10%. Note that there are many circumstances in which we would prefer to sacrifice some levels of accuracy for considerable efficiency [35], [45]. The advantage of the 1-NN 3-D DTW classifier is its efficient running time before its classification accuracy.

C. Classification Computational Cost

Fig. 20 demonstrates the pairwise comparison of classification time of 1-NN 3-D DTW with 1-NN DTW on all 85 time series in URC archive. Each point represents a classification time (in seconds) of one time series dataset. The points above the straight blue line indicate less classification runtime in 1-NN 3-D DTW compared to 1-NN DTW.

As seen in Fig. 20 1-NN 3-D DTW has a huge performance leap over 1-NN DTW in terms of time on all UCR time series datasets.

It takes only around 45 min to classify all 85 time series datasets by the 1-NN 3-D DTW using one CPU core, whereas the same job takes more than 18 h by the state-of-the-art 1-NN DTW implementation and also more than 1.2 h by a single-core running BOSS VS classifier.

Using 1-NN 3-D DTW, 73 out of 85 datasets (86%) are classified in less than 10 s, and 80 out of 85 datasets (92%) are classified less than 1 min. Fig. 21 demonstrates the pairwise comparison of classification time of 1-NN 3-D DTW with 1-NN DTW and BOSS VS only on long time series. Note that Fig. 20 shows the same comparison (between1-NN 3-D DTW with 1-NN DTW) but on all 85 datasets.1-NN 3-D DTW is significantly faster than 1-NN DTW with similar or better

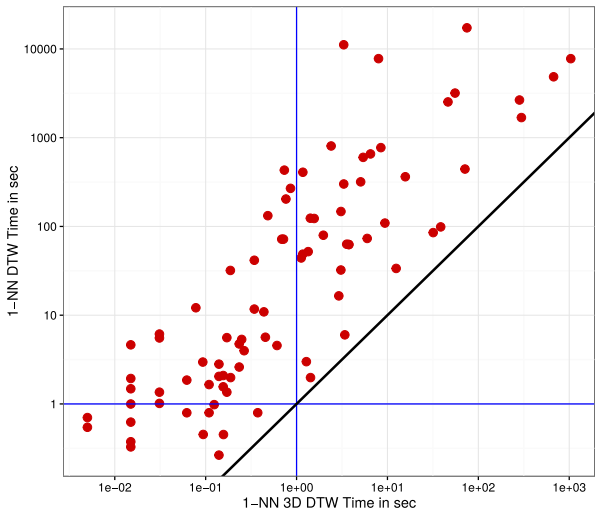

Fig. 20. Pairwise classification time comparison of 1-NN 3-D DTW with 1-NN DTW on all 85 time series datasets.

accuracy. It is orders of magnitude faster than 1-NN DTW in all long time series. 1-NN 3-D DTW is faster than BOSS VS with a slightly declined accuracy. In all the long time series datasets, 1-NN 3-D DTW is faster than BOSS VS except only three time series datasets *(Phoneme, FordA, and FordB)*. The details of classification accuracy and process time comparison between 1-NN 3-D DTW, 1-NN DTW and BOSS VS on all long time series datasets are presented in Table V. For the long time series datasets, using 1-NN 3-D DTW, the classification of all datasets finishes roughly in 37 min. Using 1-NN DTW and BOSS VS the classification finishes approximately after 17 h and 1 h, respectively.

On the tradeoff between classification accuracy against processing time and computational cost, 1-NN 3-D DTW performs reasonably well. On one hand, it is considerably more accurate than extremely fast (but not accurate) classifiers (such as 1-NN ED, 1-NN ED centroid, naive Bayes, etc.), and, on the other hand, it is significantly faster than extremely accurate (but slow) classifiers (such as COTE ensemble, shapelet, BOSS, elastic ensemble, etc.).

D. Texas Sharpshooter Plot

Along with measuring the accuracy and time classification of 1-NN 3-D DTW and comparing it to other methods, it is needed to predict ahead of time on which datasets and domains 1-NN 3-D DTW will reach superior classification accuracy results.

The Texas sharpshooter plot [15] is the tool to predict if one method or the other is performing better in terms of classification accuracy. The purpose is to predict the test classification accuracy for 1-NN 3-D DTW and the 1-NN ED based on the classification accuracy on the train datasets.

In order to create the Texas sharpshooter plot, we used the expected gain equation which is proposed in [10] and [11]

$$\text{Expected gain} = \frac{1 - \text{NN 3-D DTW accuracy}}{1 - \text{NN ED accuracy}}.$$

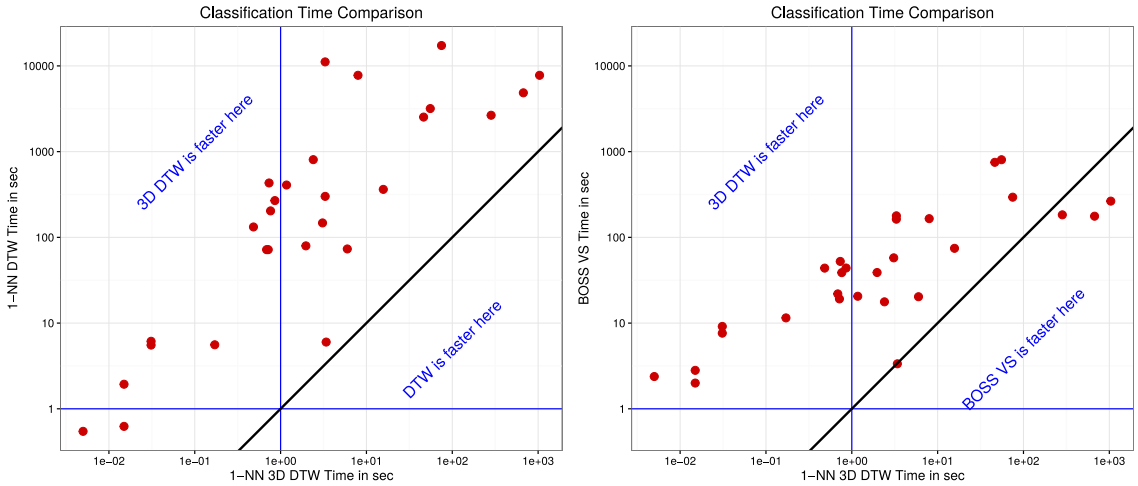

Fig. 21. Pairwise classification time comparison of 1-NN 3-D DTW with 1-NN DTW and BOSS VS on all long datasets.

TABLE V
DETAILS OF ACCURACY AND PROCESS TIME COMPARISON BETWEEN 3-D DTW, BOSS VS, AND DTW FOR LONG TIME SERIES

DatasetName		Accuracy			Time in s		
		3D DTW	BOSS VS	DTW	3D DTW	BOSS VS	DTW
1	BeetleFly	85.0%	70.0%	70.0%	0.02	2.81	0.62
2	BirdChicken	85.0%	75.0%	75.0%	0.00	2.38	0.55
3	Car	65.0%	73.3%	73.3%	0.03	9.14	5.54
4	CinC_ECG_torso	66.8%	65.1%	65.1%	1.17	20.56	407.93
5	Computers	58.4%	70.0%	70.0%	1.97	38.89	79.59
6	Earthquakes	68.0%	74.2%	74.2%	5.99	20.30	73.26
7	FordA	60.2%	56.2%	56.2%	1040.17	264.68	7770.53
8	FordB	63.5%	59.4%	59.4%	673.36	176.54	4854.18
9	HandOutlines	80.4%	79.8%	79.8%	3.30	162.97	11139.90
10	Haptics	40.9%	37.7%	37.7%	0.48	43.78	132.23
11	Herring	60.9%	53.1%	53.1%	0.03	7.63	6.13
12	InlineSkate	26.2%	38.4%	38.4%	0.73	52.42	431.03
13	LargeKitchenAppliances	76.5%	79.5%	79.5%	0.76	38.77	203.63
14	Lighting2	75.4%	86.9%	86.9%	0.17	11.50	5.59
15	MALLAT	90.0%	93.4%	93.4%	2.40	17.70	806.60
16	NonInvasiveFatalECG_Thorax1	49.1%	79.0%	79.0%	55.44	804.73	3181.19
17	NonInvasiveFatalECG_Thorax2	58.4%	86.5%	86.5%	46.35	750.58	2529.62
18	OliveOil	76.7%	83.3%	83.3%	0.02	2.00	1.94
19	Phoneme	25.2%	22.8%	22.8%	283.62	183.17	2657.32
20	RefrigerationDevices	48.3%	46.4%	46.4%	15.73	74.52	362.94
21	ScreenType	43.5%	39.7%	39.7%	3.08	57.69	147.37
22	ShapeletSim	92.2%	65.0%	65.0%	3.40	3.35	6.01
23	ShapesAll	68.7%	76.8%	76.8%	3.31	178.37	300.80
24	SmallKitchenAppliances	64.5%	64.3%	64.3%	0.86	43.90	268.59
25	StarLightCurves	90.5%	90.7%	90.7%	7.98	165.32	7773.95
26	UWaveGestureLibraryAll	90.9%	89.2%	89.2%	74.91	294.29	17296.80
27	Worms	49.2%	46.4%	46.4%	0.69	21.92	71.92
28	WormsTwoClass	61.3%	66.3%	66.3%	0.72	19.14	71.92
Total				Sec	2226.67	3469.04	60587.62
				Min	37.10	57.80	1009.80

We used threefold cross validation to calculate the classification accuracy on the train datasets. If the expected gain is greater than 1, we can assume that 1-NN 3-D DTW will have a better result than 1-NN ED in the testing of that particular dataset too. In Fig. 22, the comparison of the actual gain (based on the testing dataset) against the expected gain (based on the training dataset) of 1-NN 3-D DTW and 1-NN ED is presented.

There are four areas in this plot.

1) *True Positive:* Where expected gain is greater than 1 and actual gain is also greater than 1. It means we expected the better accuracy and we received the better accuracy. Sixteen out of 28 datasets fall into this area.
2) *False Negative:* Where expected gain is less than 1, but the actual gain is greater than 1. It means we expected worst accuracy, but we got the better accuracy. One out of 28 datasets falls into this area (*Haptics*).
3) *True Negative:* Where expected gain and actual gain are both less than 1. It means we correctly expected that the accuracy would decrease. Ten out of 28 datasets fall into this area.

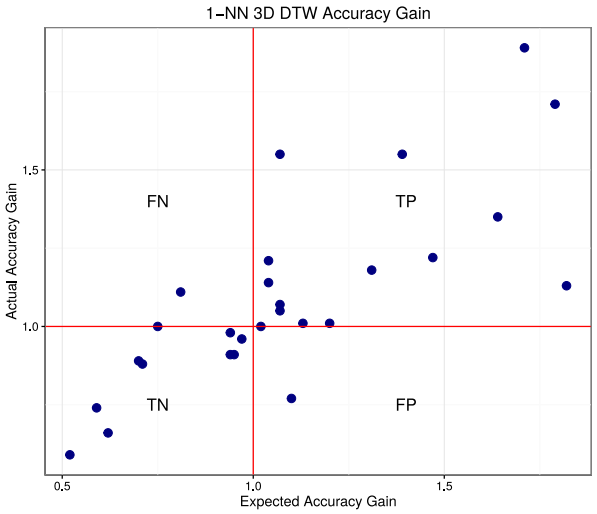

Fig. 22. Expected accuracy gain of 1-NN 3-D DTW/1-NN ED from train data compared to actual accuracy gain on test data for long datasets.

4) *False Positive:* Where expected gain is greater than 1, but the actual gain is less than 1. It means we wrongly expected the accuracy to improve, but it got worse. This is the bad area because we decided to use 1-NN 3-D DTW based on the expected gain on the training but we lost accuracy in testing. There is only 1 out of 28 datasets which falls into this area (*InlineSkate*).

Therefore, in general 1-NN 3-D DTW enjoys a high level of predictability for the classification accuracy of long time series datasets.

V. Conclusion

Our proposed methods, CCA and 1-NN 3-D DTW, have focused on data reduction and improving the efficiency of 1-NN DTW for classification of long time series in large datasets. CCA representation approximates and reduces the raw time series by creating a vector of segments with single values and durations. The output of CCA is the input of 1-NN 3-D DTW classifier.

3-D DTW is the adaptation of common DTW in a 3-D, and is used to measure the distance between two CCA representations of time series.

Using 85 time series, benchmark datasets from UCR archive, including 28 long time series datasets in an exhaustive evaluation, we show that 1-NN 3-D DTW is orders of magnitude faster than the state-of-the-art implementation of 1-NN DTW. It has better or similar level of accuracy for long time series in the experiment.

Regarding tradeoff between accuracy and computational cost, 1-NN 3-D DTW is shown to be competitive among the state-of-the-art classifiers.

Developing tuning methods for parameter s according to the target time series, could further improve the accuracy or efficiency of our technique on specific datasets, and is left for future work.

Almost all of 1-NN DTW extension methods for accuracy or efficiency improvements such as global warping constraints and weighting techniques, are also applicable on 1-NN 3-D DTW.

Acknowledgment

The authors would like to thank A. Kampert and A. Pimentel for assisting with the review of this paper.

References

[1] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *SIGMOD Rec.*, vol. 23, no. 2, pp. 419–429, Jun. 1994. [Online]. Available: http://doi.acm.org/10.1145/191843.191925

[2] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," in *Readings in Speech Recognition*, A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 159–165. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108244

[3] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. KDD Workshop*, Seattle, WA, USA, 1994, pp. 359–370. [Online]. Available: http://dblp.uni-trier.de/db/conf/kdd/kdd94.html#BerndtC94

[4] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Min. Knowl. Discov.*, vol. 7, no. 4, pp. 349–371, Oct. 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1024988512476

[5] N. A. Chadwick, D. A. McMeekin, and T. Tan, "Classifying eye and head movement artifacts in EEG signals," in *Proc. 5th IEEE Int. Conf. Digit. Ecosyst. Technol. (DEST)*, Daejeon, South Korea, 2011, pp. 285–291.

[6] 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," *Proc. VLDB Endowment*, vol. 1, no. 2, pp. 1542–1552, Aug. 2008. [Online]. Available: http://dx.doi.org/10.14778/1454159.1454226

[7] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Disc.*, vol. 29, no. 3, pp. 565–592, 2015.

[8] M. D. Schmill, T. Oates, and P. R. Cohen, "Learned models for continuous planning," in *Proc. AISTATS*, 1999, pp. 278–282.

[9] D. Vail and M. Veloso, "Learning from accelerometer data on a legged robot," presented at the *5th IFAC/EURON Symp. Intell. Auton. Veh.*, 2004.

[10] S. Kar and A. Routray, "Effect of sleep deprivation on functional connectivity of EEG channels," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 3, pp. 666–672, May 2013.

[11] B. Legrand, C. S. Chang, S.-H. Ong, S.-Y. Neo, and N. Palanisamy, "Automated identification of chromosome segments involved in translocations by combining spectral karyotyping and banding analysis," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 38, no. 6, pp. 1374–1384, Nov. 2008.

[12] X. Song, X. Xia, and F. Luan, "Online signature verification based on stable features extracted dynamically," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[13] T. Rakthanmanon *et al.*, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Beijing, China, 2012, pp. 262–270. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339576

[14] W. A. Chaovalitwongse, Y.-J. Fan, and R. C. Sachdeo, "On the time series k-nearest neighbor classification of abnormal brain activity," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 37, no. 6, pp. 1005–1016, Nov. 2007.

[15] G. E. A. P. A. Batista, X. Wang, and E. J. Keogh, "A complexity-invariant distance measure for time series," in *Proc. SDM*, Mesa, AZ, USA, 2011, pp. 699–710. [Online]. Available: http://dblp.uni-trier.de/db/conf/sdm/sdm2011.html#BatistaWK11

[16] G. E. A. P. A. Batista, E. J. Keogh, O. M. Tataw, and V. M. A. de Souza, "CID: An efficient complexity-invariant distance for time series," *Data Min. Knowl. Disc.*, vol. 28, no. 3, pp. 634–669, May 2014. [Online]. Available: http://dx.doi.org/10.1007/s10618-013-0312-3

[17] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, "A multiresolution symbolic representation of time series," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, Tokyo, Japan, 2005, pp. 668–679. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2005.10

[18] C. Ratanamahatana and E. J. Keogh, "Three myths about dynamic time warping data mining," in *Proc. SDM*, Newport Beach, CA, USA, 2005, pp. 506–510.

[19] E. J. Keogh and M. J. Pazzani, "Derivative dynamic time warping," in *Proc. 1st SIAM Int. Conf. Data Min. (SDM)*, Chicago, IL, USA, 2001, pp. 1–11.

[20] M. Kulbacki and A. Bak, "Unsupervised learning motion models using dynamic time warping," in *Intelligent Information Systems 2002*. Berlin, Germany: Springer-Verlag, 2002, pp. 217–226.

[21] Y.-S. Jeong, M. K. Jeong, and O. A. Omitaomu, "Weighted dynamic time warping for time series classification," *Pattern Recognit.*, vol. 44, no. 9, pp. 2231–2240, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2010.09.022

[22] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Disc.*, vol. 31, no. 3, pp. 606–660, 2017.

[23] F. Itakura, "Minimum prediction residual principle applied to speech recognition," in *Readings in Speech Recognition*, A. Waibel and K.-F. Lee, Eds. San Francisco, CA, USA: Morgan Kaufmann, 1990, pp. 154–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=108235.108243

[24] C. A. Ratanamahatana and E. Keogh, "Making time-series classification more accurate using learned constraints," in *Proc. SDM Int. Conf.*, Lake Buena Vista, FL, USA, 2004, pp. 11–22.

[25] E. Keogh *et al.*, "Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures," *Very Large Data Bases J.*, vol. 18, no. 3, pp. 611–630, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1007/s00778-008-0111-4

[26] S.-W. Kim, S. Park, and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," in *Proc. ICDE*, Heidelberg, Germany, 2001, pp. 607–614.

[27] A. W.-C. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C.-W. Wong, "Scaling and time warping in time series querying," *Very Large Data Bases J.*, vol. 17, no. 4, pp. 899–921, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1007/s00778-006-0040-z

[28] H. Nath and U. Baruah, "Article: Evaluation of lower bounding methods of dynamic time warping (DTW)," *Int. J. Comput. Appl.*, vol. 94, no. 20, pp. 12–17, May 2014.

[29] L. Junkui and W. Yuanzhen, "Early abandon to accelerate exact dynamic time warping," *Int. Arab J. Inf. Technol.*, vol. 6, no. 2, pp. 144–152, 2009.

[30] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *Proc. 4th Int. Conf. Found. Data Org. Algorithms (FODO)*, Chicago, IL, USA, 1993, pp. 69–84. [Online]. Available: http://dl.acm.org/citation.cfm?id=645415.652239

[31] F. Korn, H. V. Jagadish, and C. Faloutsos, "Efficiently supporting ad hoc queries in large datasets of time sequences," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, Tucson, AZ, USA, 1997, pp. 289–300. [Online]. Available: http://doi.acm.org/10.1145/253260.253332

[32] K.-P. Chan and A. W. C. Fu, "Efficient time series matching by wavelets," in *Proc. ICDE*, Sydney, NSW, Australia, 1999, pp. 126–133.

[33] 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.

[34] Z. Geng, J. Chen, and Y. Han, "Energy efficiency prediction based on PCA-FRBF model: A case study of ethylene industries," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[35] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, Jun. 2002. [Online]. Available: http://doi.acm.org/10.1145/568518.568520

[36] Y. Cai and R. Ng, "Indexing Spatio-temporal trajectories with Chebyshev polynomials," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, Paris, France, 2004, pp. 599–610. [Online]. Available: http://doi.acm.org/10.1145/1007568.1007636

[37] T.-C. Fu, "A review on time series data mining," *Eng. Appl. Artif. Intell.*, vol. 24, no. 1, pp. 164–181, Feb. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2010.09.007

[38] H. André-Jönsson and D. Z. Badal, "Using signature files for querying time-series data," in *Proc. 1st Eur. Symp. Principles Data Min. Knowl. Disc.*, Trondheim, Norway, 1997, pp. 211–220.

[39] Y.-W. Huang and P. S. Yu, "Adaptive query processing for time-series data," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, San Diego, CA, USA, 1999, pp. 282–286. [Online]. Available: http://doi.acm.org/10.1145/312129.318357

[40] A. J. Bagnall and G. J. Janacek, "Clustering time series from ARMA models with clipped data," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Seattle, WA, USA, 2004, pp. 49–58. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014061

[41] F. Mörchen and A. Ultsch, "Optimizing time series discretization for knowledge discovery," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, Chicago, IL, USA, 2005, pp. 660–665. [Online]. Available: http://doi.acm.org/10.1145/1081870.1081953

[42] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proc. 8th ACM SIGMOD Workshop Res. Issues Data Min. Knowl. Disc. (DMKD)*, San Diego, CA, USA, 2003, pp. 2–11. [Online]. Available: http://doi.acm.org/10.1145/882082.882086

[43] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing SAX: A novel symbolic representation of time series," *Data Min. Knowl. Disc.*, vol. 15, no. 2, pp. 107–144, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1007/s10618-007-0064-z

[44] B. Lkhagva, Y. Suzuki, and K. Kawagoe, "Extended SAX: Extension of symbolic aggregate approximation for financial time series data representation," *DEWS 4A-i8*, vol. 7, Mar. 1, 2006.

[45] K. P. Bennett, U. Fayyad, and D. Geiger, "Density-based indexing for approximate nearest-neighbor queries," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, San Diego, CA, USA, 1999, pp. 233–243.

[46] J. Lin, R. Khade, and Y. Li, "Rotation-invariant similarity in time series using bag-of-patterns representation," *J. Intell. Inf. Syst.*, vol. 39, no. 2, pp. 287–315, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10844-012-0196-5

[47] M. G. Baydogan, G. C. Runger, and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2796–2802, Nov. 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/pami/pami35.html#BaydoganRT13

[48] P. Schäfer, "The boss is concerned with time series classification in the presence of noise," *Data Min. Knowl. Disc.*, vol. 29, no. 6, pp. 1505–1530, 2015. [Online]. Available: http://dblp.uni-trier.de/db/journals/datamine/datamine29.html#Schafer15

[49] P. Schäfer, "Scalable time series classification," *Data Min. Knowl. Disc.*, vol. 30, no. 5, pp. 1273–1298, 2016.

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

[51] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-series classification with COTE: The collective of transformation-based ensembles," in *Proc. 32nd IEEE Int. Conf. Data Eng. (ICDE)*, Helsinki, Finland, 2016, pp. 1548–1549. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2016.7498418

[52] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Min. Knowl. Disc.*, vol. 28, no. 4, pp. 851–881, 2014.

[53] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, Beijing, China, 2012, pp. 289–297.

[54] H. Hamooni and A. Mueen, "Dual-domain hierarchical classification of phonetic time series," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, Shenzhen, China, Dec. 2014, pp. 160–169. [Online]. Available: http://dx.doi.org/10.1109/ICDM.2014.92

Anooshiravan Sharabiani received the B.S. degree in industrial engineering and system analysis from the University of Science and Technology, Meghalaya, Ri-Bhoi, India, in 1999, and the M.S. degree in industrial engineering from the Sharif University of Technology, Tehran, Iran, in 2001. He is currently pursuing the Ph.D. degree in industrial engineering and operation research with the University of Illinois at Chicago (UIC), Chicago, IL, USA.

From 2001 to 2012, he was a Senior Data Analyst in different industrial sections. He was also a Lead Data Scientist with the Process Mining and Intelligent System Analytics (Prominent) Laboratory, UIC. His current research interests include data mining, big data analysis, predictive modeling, time series analysis, data analytics, and BI and dashboards.

Mr. Sharabiani was a recipient of the 2014–2015 Firdawsi Science Fellowship Award by the Deans of LAS, Agriculture, and Engineering at UIUC and UIC.

Houshang Darabi (S'98–A'00–M'10–SM'14) received the Ph.D. degree in industrial and systems engineering from Rutgers University, New Brunswick, NJ, USA, in 2000.

He is currently an Associate Professor with the Department of Mechanical and Industrial Engineering, University of Illinois at Chicago, Chicago, IL, USA, where he is also an Associate Professor with the Department of Computer Science. His research has been supported by several federal and private agencies, such as the National Science Foundation, the National Institute of Standard and Technology, the Department of Energy, and Motorola. He has published in different prestigious journals and conference proceedings, such as the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, and *Information Sciences*. He has been a contributing author of two books in the areas of scalable enterprise systems and reconfigurable discrete event systems. He has extensively published on various automation and project management subjects, including wireless sensory networks for location sensing, planning and management of projects with tasks requiring multimode resources, and workflow modeling and management. His current research interests include the application of data mining, process mining, and optimization in design and analysis of manufacturing, business, project management, and workflow management systems.

Ashkan Rezaei received the B.S. degree in computer science from TU Wien, Vienna, Austria. He is currently pursuing the Ph.D. degree in computer science with the University of Illinois at Chicago, Chicago, IL, USA.

His current research interests include statistical machine learning theory and application, in particular adversarial prediction in supervised learning, deep learning, and convex optimization.

Samuel Harford received the B.S. degree in industrial engineering from the University of Illinois at Chicago (UIC), Chicago, IL, USA, in 2016, where he is currently pursuing the M.S. degree.

He is currently a Research Assistant with the PROMINENT Laboratory, UIC, where he is also an Aspiring Data Scientist. His current research interests include education data mining and time series analysis.

Hereford Johnson received the undergraduate degree in industrial engineering from the University of Illinois at Chicago (UIC), Chicago, IL, USA, in 2016, where he is currently pursuing the M.S. degree.

He is a Research Assistant with the PROMINENT Laboratory, UIC, under the leadership of Dr. H. Darabi, where he performs data mining tasks in the field of higher education and time series classification. He has also earned a Six Sigma Green Belt through IISE, and has industry experience in process improvement, business intelligence, and software engineering. His current research interests include time series classification, NLP techniques for classification, and the application of data mining in finance.

Fazle Karim received the B.Sc. degree in industrial engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2012, and the M.Sc. degree in industrial engineering from the University of Illinois at Chicago (UIC), Chicago, IL, USA, in 2016, where he is currently pursuing the Ph.D. degree with the Mechanical and Industrial Engineering Department.

He is the Lead Data Scientist with the PROMINENT Laboratory, UIC. His current research interests include education data mining, health care data mining, and time series analysis.