Compression Rate Distance Measure for Time Series

Vo Thanh Vinh  
Faculty of Information Technology  
Ton Duc Thang University  
Ho Chi Minh City, Viet Nam  
vthanh@it.tdt.edu.vn

Duong Tuan Anh  
Faculty of Computer Science & Engineering  
Ho Chi Minh City University of Technology  
Ho Chi Minh City, Viet Nam  
dtanh@ese.hcmut.edu.vn

Abstract—In this work, we propose a Compression Rate Distance, a new distance measure for time series data. The main idea behind this distance is based on the Minimum Description Length (MDL) principle. The higher compression rate between two time series is, the closer they should be. Besides, we also propose a relaxed version of the new distance, called the Extended Compression Rate Distance. The Extended Compression Rate Distance can satisfy some crucial characteristics on time series such as Early Abandoning, Lower Bounding, and Relax-Triangular Inequality which help the new distance easily adapt with traditional indexing structures and searching methods. We tested our distances on classification problem with numerous datasets and compared the results with most of the commonly used distances in time series such as Euclidean Distance, Dynamic Time Warping, and a recently proposed Complexity-Invariant Distance. Experimental results reveal that our novel distances outperform several previous important distance measures in a vast majority of the datasets.

Keywords—compression rate distance; MDL principle; time series; similarity; lower bounding; early abandoning; relaxed-triangular inequality

I. INTRODUCTION

Time series data exists everywhere, in a vast number of fields such as finance, economy, medicine, archaeology, biology, geography, hydrometeorology, etc. A time series is a series of real number values of a quantity obtained at successive times, often with equal intervals between them. Time series analysis can help to see how a variable changes over time or how it changes compared to other variables over the same period.

In time series data mining, distance measure (also called similarity measure) is a crucial factor which has a huge impact on several problems such as similarity search, classification, clustering, motif discovery, anomaly detection, prediction, etc. All of these problems need a distance to estimate the similarity between time series. It would not exaggerate to conclude that distance measure has the most important effect on the accuracy of these problems. The better a distance measure is used, the more correct results are obtained. There have been several distance functions proposed for time series. Each distance measure is suitable to some special application domains. In which, Euclidean Distance (ED) and Dynamic Time Warping Distance (DTW) are the most widely used. Recently, we also witness a new distance measure called Complexity-Invariant Distance (CID) proposed by Batista et al. 2014 [1]. The complexity-Invariant distance is simple and increases the time complexity only a small amount.

Euclidean Distance is the most commonly used distance measure in time series because it is fast and easy-to-implement. This distance can be found in most of the studies in time series. Moreover, this distance is considered a baseline distance to compare with when a new distance is invented. Besides, Euclidean Distance satisfies the triangular inequality. Therefore, it can be used with the support of some index structures in the metric space. However, this distance cannot handle shifting, scaling, or time warping problem. Therefore, many techniques such as data normalization before calculating distance, or some flexible similarity measures have been proposed to address these weaknesses. Among these distances are Dynamic Time Warping and Longest Common Subsequence (LCS).

Dynamic Time Warping is a commonly used distance in time series. This distance aims to find the best matching between two time series. During that matching, it calculates the smallest distance between time series. Such a best matching is also called the warping path. However, the Dynamic Time Warping Distance incurs high computational complexity (i.e., O(n^2)). For a more efficient calculation of this distance, some global constraints such as Sakoe-Chiba band [7], and Itakura Parallelogram [8] were proposed. And in the last decade, there have been more techniques to speed up DTW computation such as early abandoning or lower bounding [3]. Moreover, it is really hard to find an appropriate indexing structure that can support this distance in similarity search since DTW does not satisfy the triangular inequality. To the best of our knowledge, the two index structures that can work well with Dynamic Time Warping Distance at this time are R*-tree [3] and TS-Tree [6].

The Longest Common Subsequence was proposed by M. Vlachos et al. in 2004 [5]. The most desirable characteristic in this distance is that it can waive noise and distortions in time series. The main idea behinds this distance is that it is based on the common subsequence. The longer common subsequences the two time series have, the closer they are.

Batista et al. in 2014 [1], proposed an interesting distance called the Complexity-Invariant Distance. This distance multiplies the original distance (Euclidean Distance or Dynamic Time Warping) with a factor called complexity estimate. The term “complex” in this distance means “there are many peaks and valleys in the time series” [1].
In this work, we propose a Compression Rate Distance, a new distance measure for time series data. The main idea behind this distance is based on the Minimum Description Length (MDL) principle. The higher compression rate between two time series is, the closer they should be. Besides, we also propose a relaxed version of the new distance, called the Extended Compression Rate Distance. The Extended Compression Rate Distance can satisfy some crucial characteristics on time series such as Early Abandoning, Lower Bounding, and Relaxed-Triangular Inequality which help the new distance easily adapt with traditional indexing structures and searching methods. Furthermore, we also propose a Secondary Lower Bounding for Extended Compression Rate Distance, which aims to prune more objects when finding nearest neighbors. Finally, our distances can run in a linear time \(O(n)\). We experimented our distances on classification problem with numerous datasets and compared the results with most of the commonly used distances in time series such as Euclidean Distance, Dynamic Time Warping, and a recently proposed Complexity-Invariant Distance. Experimental results reveal that our novel distances outperform several of the previous important distance measures in a vast majority of the datasets.

The rest of this paper is organized as follows. Section II reviews common distance measures as well as the MDL principle. Section III gives details of Compression Rate Distance and Extended Compression Rate Distance, followed by a set of experiments in Section IV. Section V concludes the work and gives suggestions for future work.

II. BACKGROUND AND RELATED WORKS

In this section, we introduce briefly the concept of time series, 1-Nearest Neighbor classifier, and some current widely used distances for time series such as: Euclidean Distance, Dynamic Time Warping Distance, and Complexity-Invariant Distance. We also depict the MDL principle which we applied on our proposed distance measure.

A. Time series

A time series \(T\) is a sequence of real numbers collected at regular intervals over a period of time: \(T = t_1, t_2, \ldots, t_n\).

B. 1-Nearest Neighbor Classification

Among many classification methods such as Artificial Neural Network, Bayesian Network, Decision Tree, etc. 1-Nearest Neighbor (1-NN) has been endorsed to be the best classification method on time series data [12]. In 1-NN, the data object is classified the same class as its nearest object in the training set. Therefore, the accuracy of this method is dependent on the distance measure.

C. Euclidean Distance

In the Euclidean Distance, the similarity between two time series \(Q = q_1, q_2, \ldots, q_n\) and \(C = c_1, c_2, \ldots, c_n\) is defined as follows:

\[
ED(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}
\]

Note that the two time series \(Q\) and \(C\) must have the same length of \(n\). In Fig. 1(a), we visualize the Euclidean Distance between two time series \(Q\) and \(C\). As we can see, Euclidean distance aims to calculate the difference in each dimension of time series \(Q\) and the corresponding dimension in time series \(C\), and this is a linear alignment.

D. Dynamic Time Warping Distance

One problem with time series data is the distortion along the time axis, making Euclidean distance unsuitable. Fortunately, this problem can be effectively addressed by Dynamic Time Warping (DTW), a distance measure that allows non-linear alignment between the two time series to accommodate sequences that are similar but out of phase [2]. Fig. 1 shows the alignment between two time series when we calculate difference values between them. As we can see, the DTW calculates a minimum distance between two time series by finding a warping path which is a non-linear alignment, whereas Euclidean Distance calculates the differences between two time series by subtracting values using a linear alignment, this difference make DTW much more flexible than Euclidean Distance.

Given two time series \(Q\) and \(C\) which have length \(n\) and \(m\) respectively: \(Q = q_1, q_2, \ldots, q_n\) and \(C = c_1, c_2, \ldots, c_m\). DTW is a dynamic programming technique which calculates all possible warping paths between two time series for finding minimum distance. To calculate DTW between the two above time series, firstly we construct a matrix \(D\) with size \(m \times n\). Every element in matrix \(D\) is cumulative distance defined as:

\[
\gamma(i,j) = d(i,j) + \min\{\gamma(i-1,j), \gamma(i,j-1), \gamma(i-1,j-1)\}
\]

where \(\gamma(i,j)\) is \((i,j)\) element of matrix that is a summation between \(d(i,j) = (q_i - c_j)^2\), a square distance of \(q_i\) and \(c_j\) and the minimum cumulative distance of three adjacent elements to \((i,j)\).

Next, we choose the optimal warping path which has minimum cumulative distance defined as:

\[
DTW(Q,C) = \min_{k=1}^{K} w_k
\]

where \(w_k\) is \((i,j)\) at \(k^{th}\) element of the warping path, and \(K\) is the length of the warping path.

In addition, for a more accurate distance measure, some global constraints were suggested to DTW. A well-known constraint is Sakoe-Chiba band [7], shown in Fig. 2. The Sakoe-Chiba band constrains the indices of the warping path \(w_k\).
Distance is \( O(n^2) \) whereas the complexity of Euclidean Distance is \( O(n) \), where \( n \) is the length of time series.

Note that complexity of the algorithm to calculate DTW Distance is \( O(n^2) \) whereas the complexity of Euclidean Distance is \( O(n) \), where \( n \) is the length of time series.

Fig. 2. DTW with Sakoe-Chiba band.

E. Complexity-Invariant Distance

The Complexity-Invariant Distance (CID) was proposed by Batista et al. in 2014 [1]. The core contribution of this distance is to introduce complexity invariance; the term “complex” means having many peaks and valleys. The more peaks and valleys the time series has, the more complex it is. The authors introduce their distance based on Euclidean Distance and the complexity correction factor as follows:

\[
CID(Q, C) = ED(Q, C) \times CF(Q, C)
\]

where \( CF \) is a complexity correction factor defined as:

\[
CF(Q, C) = \frac{\text{max}(CE(Q), CE(C))}{\text{min}(CE(Q), CE(C))}
\]

in which, \( CE(T) \) is a complexity estimate of time series \( T = t_1, t_2, \ldots, t_n \). The \( CE \) is defined as follows:

\[
CE(T) = \left( \sum_{i=1}^{n-1} (t_i - t_{i+1})^2 \right)^{1/2}
\]

Note that the complexity of calculating CID is \( O(n) \).

In this work, our main investigation is to compare our proposed distance with CID on classification problem. Besides, we also compare our distance with Euclidean Distance and DTW.

F. MDL Principle

In this subsection, we review the Minimum Description Length principle. This principle, which comes from information theory, has been applied in many time series data mining tasks, such as motif discovery by Tanaka et al. in 2005 [16], criterion for clustering by Rakthanmanon et al. in 2012 [9], stopping criterion for semi-supervised classification by Begum et al. in 2013 [10], and in our previous work on semi-supervised classification of time series [11]. This principle is also the core spirit in this work.

1) Discrete Normalization Function: A discrete function \( \text{Dis}_\text{Norm} \) is the function to normalize a real-value subsequence \( T \) into \( b \)-bit discrete value of range \([1, 2^b]\). It is defined as follows:

\[
\text{Dis}_\text{Norm}(T) = \text{round}\left((T - \text{min})(\text{max} - \text{min})/(2^b - 1)\right) + 1
\]

where \( \text{min} \) and \( \text{max} \) are the minimum and maximum value in \( T \), respectively.

After casting the original real-valued data to discrete values, we are interested in determining how many bits are needed to store a particular time series \( T \). It is called the Description Length of \( T \).

2) Entropy: The entropy of a time series \( T \) is defined as follows:

\[
E(T) = -\sum_{t} P(T = t) \log_2 P(T = t)
\]

where \( P(T = t) \) is the probability of value \( t \) in time series \( T \).

3) Description Length: A description length \( DL \) of a time series \( T \) is the total number of bits required to represent it.

\[
DL(T) = w \times E(T)
\]

where \( w \) is the length of \( T \).

4) Hypothesis: A hypothesis \( H \) is a time series used to encode one or more time series of the same length.

We are interested in how many bits are required to encode \( T \) given \( H \). It is called the Reduced Description Length of \( T \).

5) Reduced Description Length: A reduced description length of a time series \( T \) given hypothesis \( H \) is the sum of the number of bits required in order to encode \( T \) exploiting the information in \( H \), i.e. \( DL(T|H) \), and the number of bits required for \( H \) itself, i.e. \( DL(H) \).

Thus, the reduced description length is defined as:

\[
DL(T, H) = DL(T) + DL(T|H)
\]

One simple approach of encoding \( T \) using \( H \) is to store a difference vector between \( T \) and \( H \). Therefore: \( DL(T|H) = DL(T - H) \).

Example: Given \( A \) and \( B \), two time series of length 20:

\[
A = [1 2 4 5 6 8 8 9 10 11 13 14 17 16 17 18 19 19]
B = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
\]

Without encoding, the bit requirement to store \( A \) and \( B \) is \( 20 \times E(A) + 20 \times E(B) = 20 \times 3.8219 + 20 \times 4.3219 = 162.876 \) bits. The difference vector \( A' = A - B = [0 1 0 0 0 1 0 0 0 1 0 0 2 0 0 0 0 -1] \). The bit requirement is now just \( 20 \times E(A') + 20 \times E(B) = 20 \times 1.154 + 20 \times 4.3219 = 109.5180 \) bits, which brings out a good data compression.

Now given \( B \) as above and \( C = [20 15 10 19 5 10 2 7 8 6 3 18 16 17 13 6 3 7 8] \).

We compute the bit requirement without encoding: \( 20 \times E(C) + 20 \times E(B) = 20 \times 3.6842 + 20 \times 4.3219 = 160.122 \) bits. With encoding, the bit requirement now is \( 20 \times E(C - B) + \)
20×E(B) = 20×4.0219 + 20×4.3219 = 166.876 bits. This result illustrates that C is not similar to B because there is no compression when using encoding.

The MDL is a method for selecting a good model. A model is considered good if there is some compression obtained through encoding. In this paper, we apply MDL principle to create a new distance measure for time series.

III. CRD/ECRD: THE COMPRESSION RATE DISTANCE

In this section, we explain why DTW and Euclidean Distance cannot work well in some situations. Then, we show how MDL can help to solve the problem and introduce our distance measure as well as its extended version. In addition, we also show some characteristics of our distance such as Early Abandoning, Lower Bounding, and Relaxed-Triangular Inequality. Besides, we also propose a Secondary Lower Bounding to speed up the computation of our distance.

A. The problems of Euclidean and DTW

We begin this subsection by an example. Given three time series \( P, Q, \) and \( R \) of the same length 16 as follows:

\[
Q = [1.2 2.2 2.5 2.7 2.7 2.6 2.5 2.4 2.6 2.7 2.8 3.2 2.6 2.3 2.2 2.2] \\
C = [2.1 2.2 2.5 2.7 2.7 2.6 2.4 2.4 2.5 2.7 2.8 3.2 2.6 2.3 2.2 1.5] \\
R = [2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.5 2.2 2.1 1.4]
\]

As we can easily see from the values of the three time series, \( Q \) and \( C \) are nearly the same; the only difference between them is some values at the beginning and at the end of the time series. In contrast, time series \( Q \) and \( R \) are very different. Fig. 3 shows an illustration of the three time series under consideration. As we can see, time series \( Q \) and \( C \) look similar in comparison to \( C \) and \( R \) which seem to come from different classes.

![Fig. 3. The three time series Q, C, and R where Q and C look very similar; C and R look different.](image)

However, when we use Euclidean Distance to measure the similarity of \( Q, C \) and \( R \), the results seem to be surprisingly different to what we expect. \( ED(Q, C) = 1.0344 \), \( ED(C, R) = 0.8775 \). This is an unexpected result because \( C \) and \( R \) look very different but the Euclidean Distance shows that they are more similar than \( Q \) and \( C \). Even if we normalized our time series using z-score normalization before calculating the distance, the result cannot be improved in this situation. By using z-score normalization before computing Euclidean Distance, \( ED(Q, C) = 2.6185 \), and \( ED(C, R) = 2.5109 \). Now we move to DTW Distance, \( DTW(Q, C) = 1.0344 \), \( DTW(C, R) = 0.8775 \); and by using z-score before computing the distance, \( DTW(Q, C) = 2.6185 \), whereas \( DTW(C, R) = 2.5109 \). The matter seems not better.

The above example shows that, even the two time series look very similar, but if there are some small parts of different points, their distance would become very high.

B. How MDL principle can help?

Originally, the MDL principle helps to select a good model. The more compression the model achieves, the better it is. We can apply this property in two time series when we compute their distance. If we can compress our time series when we use one time series as a hypothesis to encode the other one, their distance is reduced; otherwise, their distance increases.

With regards to the example in subsection III-A, now we can calculate the Reduced Description Length of \( Q \) with \( C \) as hypothesis by applying the spirit of MDL described in subsection II-F. First of all, we cast the three time series \( Q, C \) and \( R \) into 3-bit discrete values by using Dis_Norm function (as given in subsection II-F1):

\[
Q = [1 5 6 7 7 6 6 6 7 6 7 8 6 5 5 4] \\
C = [4 4 6 7 7 6 5 7 7 8 6 5 4 1] \\
R = [8 8 8 8 8 8 8 8 8 6 5 1]
\]

We have:

\[
DL(Q, C) = DL(C) + DL(Q - C) = 37.6355 + 21.1914 = 58.8269.
\]

And the Reduced Description Length of \( R \) with \( C \) as hypothesis is as follows:

\[
DL(R, C) = DL(C) + DL(R - C) = 37.6355 + 32.8806 = 70.5161.
\]

The above result indicates that \( Q \) should be closer to \( C \) than \( R \). Now with \( DL(C) = 37.6355 \), \( DL(Q) = 35.7353 \), \( DL(R) = 15.8943 \). We can calculate a simple compression rate as follows:

\[
SCR(Q, C) = \frac{DL(Q, C)}{DL(Q) + DL(C)} = \frac{58.8269}{35.7353 + 37.6355} = 0.8018
\]

\[
SCR(R, C) = \frac{DL(R, C)}{DL(R) + DL(C)} = \frac{70.5161}{15.8943 + 37.6355} = 1.3173
\]

Now if we multiply the simple compression rate with the Euclidean Distance, the results are as follows:

\[
ED(Q, C) \times SCR(Q, C) = 1.0344 \times 0.8018 = 0.8294
\]

\[
ED(R, C) \times SCR(R, C) = 0.8775 \times 1.3159 = 1.1559
\]

This result is what we expected; \( Q \) and \( C \) now become more similar than \( R \) and \( C \). Note that when calculating the Reduced Description Length, we can select either of the two time series as hypothesis to encode the other one, so the above compression rate is not general. We will redefine the compression rate in the next subsection when we introduce the Compression Rate Distance.

C. The Compression Rate Distance

In subsection III-B, we show that the MDL principle can help to improve the distance measures. Now we would like to introduce a generalized formula for our Compression Rate Distance (CRD).
Given two time series $Q = q_1, q_2, \ldots, q_n$ and $C = c_1, c_2, \ldots, c_m$. The CRD is defined as follows:

$$CRD(Q, C) = CR(Q, C)^\alpha \times ED(Q, C)$$

where $CR$ is the compression rate, $\alpha$ is a bias which is a real number greater than or equal to zero, the greater the value of $\alpha$, the more effect of the compression rate on the distance, and $ED$ is the Euclidean Distance. We would like to redefine the compression rate $CR$ as follows:

$$CR(Q, C) = \frac{DL(Q - C)}{\min[DL(Q), DL(C)] + \varepsilon}$$

where $\varepsilon$ is an extremely small value to prevent division by zero, and $DL$ is the Description Length of time series.

Because the lengths of two time series are equal, the above compression rate is approximated as follows:

$$CR(Q, C) = \frac{E(Q - C)}{\min[E(Q), E(C)] + \varepsilon}$$

Fig. 4 shows the algorithm of computing CRD based on the above formulas. The compression rate factor $CR(Q, C)^\alpha$ is a real value that is greater than or equal to zero, so it is hard to have some important characteristics such as Lower Bounding or Triangular Inequality. To overcome these difficulties, we would like to introduce an extended version of the CRD in the next subsection.

Notice that since CRD is based on Euclidean distance, it has the same requirement as in Euclidean distance: the two time series that we have to find CRD distance between them must be of the same length.

D. The Extended Compression Rate Distance

In this subsection, we define the Extended Compression Rate Distance (ECRD) of two time series $Q = q_1, q_2, \ldots, q_n$ and $C = c_1, c_2, \ldots, c_m$ as follows:

$$ECRD(Q, C) = (CR(Q, C)^\alpha + 1) \times ED(Q, C)$$

where $CR$ and $\alpha$ are described in subsection III-C.

**Table 1.**

<table>
<thead>
<tr>
<th>CRD/ECRD $(x, \text{discrete}_x, y, \text{discrete}_y, \text{alpha}, \text{best so far})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: $x, y$: two time series; $\alpha$: bias value; $\text{discrete}_x, \text{discrete}<em>y$: integer values of time series $x$ and time series $y$ casted by function $\text{Dis}</em>\text{Norm}$; $\text{best so far}$: square of best distance at current time.</td>
</tr>
<tr>
<td>Output: CRD/ECRD between $x$ and $y$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRD/ECRD $(x, \text{discrete}_x, y, \text{discrete}_y, \text{alpha}, \text{best so far})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: $x, y$: two time series; $\alpha$: bias value; $\text{discrete}_x, \text{discrete}<em>y$: integer values of time series $x$ and time series $y$ casted by function $\text{Dis}</em>\text{Norm}$; $\text{best so far}$: square of best distance at current time.</td>
</tr>
<tr>
<td>Output: CRD/ECRD between $x$ and $y$.</td>
</tr>
</tbody>
</table>

1. $dl_x = \text{Entropy}(\text{discrete}_x)$
2. $dl_y = \text{Entropy}(\text{discrete}_y)$
3. $dl_{encoded} = \text{Entropy}(\text{discrete}_x - \text{discrete}_y)$
4. $CR = dl_{encoded} / (\min(dl_x, dl_y) + \varepsilon)$
5. return $\text{pow}(CR, \alpha) \times ED(x, y)$, if CRD
   - $\text{pow}(CR, \alpha) + 1 \times ED(x, y)$, if ECRD

Fig. 5. The CRD/ECRD algorithm with early abandoning.

In Fig. 4, we describe the algorithm for computing Extended Compression Rate Distance. Before using this algorithm, we cast the two time series $x, y$ into integer values by using formula in subsection II-F1. Then, the algorithm computes $DL(x), DL(y)$ and $DL(x - y)$ on these discrete values of time series $x$ and $y$. After that, we compute the compression rate. Finally, we compute the ECRD based on the given formulas.

Now we would like to analyze the computational complexity of CRD and ECRD. Because the complexity of the algorithm of Entropy, and $ED$ are $O(n)$, the overall complexity of CRD/ECRD is also $O(n)$.

E. Early Abandoning

In $k$-nearest neighbor query and range query, one interesting characteristic is that we do not need to continue calculating the distance if its current accumulated value exceeds the best so far value. For example, in $1$-nearest query, assume that we need to find a nearest time series of time series $Q$, and we have known the distance between $Q$ and $R$ is $1.5$, that is the best so far value. Now we need to calculate distance between $Q$ and $C$ to find if $C$ is closer to $Q$ than $R$. While calculating distance from $Q$ to $C$, if the accumulated distance is greater than or equal to $1.5$, we do not need to continue this procedure because $C$ would surely be farther than $R$.

**Table 1.**

<table>
<thead>
<tr>
<th>CRD/ECRD $(x, \text{discrete}_x, y, \text{discrete}_y, \text{alpha}, \text{best so far})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: $x, y$: two time series; $\alpha$: bias value; $\text{discrete}_x, \text{discrete}<em>y$: integer values of time series $x$ and time series $y$ casted by function $\text{Dis}</em>\text{Norm}$; $\text{best so far}$: square of best distance at current time.</td>
</tr>
<tr>
<td>Output: CRD/ECRD between $x$ and $y$ or $+INFINITY$ if there is early abandon.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRD/ECRD $(x, \text{discrete}_x, y, \text{discrete}_y, \text{alpha}, \text{best so far})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: $x, y$: two time series; $\alpha$: bias value; $\text{discrete}_x, \text{discrete}<em>y$: integer values of time series $x$ and time series $y$ casted by function $\text{Dis}</em>\text{Norm}$; $\text{best so far}$: square of best distance at current time.</td>
</tr>
<tr>
<td>Output: CRD/ECRD between $x$ and $y$ or $+INFINITY$ if there is early abandon.</td>
</tr>
</tbody>
</table>

1. $dl_x = \text{Entropy}(\text{discrete}_x)$
2. $dl_y = \text{Entropy}(\text{discrete}_y)$
3. $dl_{encoded} = \text{Entropy}(\text{discrete}_x - \text{discrete}_y)$
4. $CR = dl_{encoded} / (\min(dl_x, dl_y) + \varepsilon)$
5. $\text{CR factor} = \text{pow}(CR, \alpha) + 1$, if CRD
6. $\text{pow}(CR, \alpha) + 1$, if ECRD
7. dist = 0
8. for $i = 1$ to length($x$)
9. dist = dist + square($x(i) - y(i)) \times \text{CR factor}$
10. if dist > best so far
11. return $+INFINITY$
12. end
13. end
14. return dist

Fig. 5. The CRD/ECRD algorithm with early abandoning.

In Fig. 4, we describe the algorithm for computing CRD/ECRD with early abandoning. Since it is easy to apply early abandoning in Euclidean distance and CRD/ECRD is based on Euclidean distance, in this algorithm, we simply modify the phase of calculating the Euclidean distance in order to include early abandoning.

F. Lower bounding for ECRD

Time series data tend to have very high dimensionality. In practice, we usually reduce their dimensionality to keep only basic features to fit them into main memory. There are numerous kinds of these representations. These representations of time series are called reduced or approximate representation. The distance between time series in reduced representation should be a lower bound of their distance in the original space.
i.e. the distance in reduced representation should be less than or equal to the distance in original representation. This means that the following condition must be satisfied:

$$D_{\text{Reduced}}(Q', C') \leq D_{\text{Original}}(Q, C)$$

where $Q'$ and $C'$ are the transformations of $Q$ and $C$, respectively in reduced space. There are several lower bounding representations have been proposed for time series. In this work, we simply inherit these lower bounding representations for our ECRD. For example, in Piecewise Aggregate Approximation (PAA), we have $D_{\text{PAA}}(Q_{\text{PAA}}, C_{\text{PAA}}) = ED(Q, C)$, according to Keogh [13]. Because the compression rate CR is always greater than or equal to zero, we have $ED(Q, C) \leq ECRD(Q, C)$. Therefore, we can conclude that $D_{\text{PAA}}(Q_{\text{PAA}}, C_{\text{PAA}}) \leq ECRD(Q, C)$ and $D_{\text{PAA}}$ is a lower bounding of ECRD.

In this work, we use $LB(Q, C)$ denote $D_{\text{Reduced}}(Q', C')$.

We have known about lower bounding but why do we need the lower bounding. For example, in range query problem of time series, we need to find all the time series in databases, which distances to time series $Q$ is less than or equal to a range $r$, i.e. we must find all time series $C_i$ satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Because we have approximated time series to its reduced dimensionality, we must use the distance function in reduced space to find them. Assume that we found all time series $C_i$ satisfy the requirement, i.e. $LB(Q, C_i) \leq r$. Now we have two cases:

Case 1: $LB(Q, C_i) \leq r \leq D_{\text{Original}}(Q, C_i)$

Case 2: $LB(Q, C_i) \leq D_{\text{Original}}(Q, C_i) \leq r$

On case 1, we would reject $C_i$ because it does not satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Case 2 satisfies the requirement, so we accept time series $C_i$. The lower bounding ensures that our query results do not miss any candidates.

G. Speed up more with Secondary Lower Bounding

In subsection III-F, we have introduced the Lower Bounding (LB) for ECRD. However, this lower bounding seems not close to the ECRD because it is simply inherit the LB of previous distances. In this subsection, we propose a Secondary Lower Bounding (SLB). We use SLB in a second check when finding the nearest neighbor. Note that SLB is not a substitute for the LB, since we will use both of them. The SLB for ECRD is given as follows:

$$SLB(Q, C) = LB(Q, C) \times (CR(Q, C)^a + 1)$$

We can easily prove that SLB is a lower bounding of ECRD: $SLB(Q, C) \leq ECRD(Q, C)$. First, we have $LB(Q, C) \leq D_{\text{Original}}(Q, C)$. We multiply both sides with $CR(Q, C)^a + 1$, then we derive:

$$LB(Q, C) \times (CR(Q, C)^a + 1) \leq D_{\text{Original}}(Q, C) \times (CR(Q, C)^a + 1)$$

Therefore, $LB(Q, C) \times (CR(Q, C)^a + 1) \leq ECRD(Q, C)$. We have known about lower bounding but why do we need the lower bounding. For example, in range query problem of time series, we need to find all the time series in databases, which distances to time series $Q$ is less than or equal to a range $r$, i.e. we must find all time series $C_i$ satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Because we have approximated time series to its reduced dimensionality, we must use the distance function in reduced space to find them. Assume that we found all time series $C_i$ satisfy the requirement, i.e. $LB(Q, C_i) \leq r$. Now we have two cases:

Case 1: $LB(Q, C_i) \leq r \leq D_{\text{Original}}(Q, C_i)$

Case 2: $LB(Q, C_i) \leq D_{\text{Original}}(Q, C_i) \leq r$

On case 1, we would reject $C_i$ because it does not satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Case 2 satisfies the requirement, so we accept time series $C_i$. The lower bounding ensures that our query results do not miss any candidates.

G. Speed up more with Secondary Lower Bounding

In subsection III-F, we have introduced the Lower Bounding (LB) for ECRD. However, this lower bounding seems not close to the ECRD because it is simply inherit the LB of previous distances. In this subsection, we propose a Secondary Lower Bounding (SLB). We use SLB in a second check when finding the nearest neighbor. Note that SLB is not a substitute for the LB, since we will use both of them. The SLB for ECRD is given as follows:

$$SLB(Q, C) = LB(Q, C) \times (CR(Q, C)^a + 1)$$

We can easily prove that SLB is a lower bounding of ECRD: $SLB(Q, C) \leq ECRD(Q, C)$. First, we have $LB(Q, C) \leq D_{\text{Original}}(Q, C)$. We multiply both sides with $CR(Q, C)^a + 1$, then we derive:

$$LB(Q, C) \times (CR(Q, C)^a + 1) \leq D_{\text{Original}}(Q, C) \times (CR(Q, C)^a + 1)$$

Therefore, $LB(Q, C) \times (CR(Q, C)^a + 1) \leq ECRD(Q, C)$. We have known about lower bounding but why do we need the lower bounding. For example, in range query problem of time series, we need to find all the time series in databases, which distances to time series $Q$ is less than or equal to a range $r$, i.e. we must find all time series $C_i$ satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Because we have approximated time series to its reduced dimensionality, we must use the distance function in reduced space to find them. Assume that we found all time series $C_i$ satisfy the requirement, i.e. $LB(Q, C_i) \leq r$. Now we have two cases:

Case 1: $LB(Q, C_i) \leq r \leq D_{\text{Original}}(Q, C_i)$

Case 2: $LB(Q, C_i) \leq D_{\text{Original}}(Q, C_i) \leq r$

On case 1, we would reject $C_i$ because it does not satisfy $D_{\text{Original}}(Q, C_i) \leq r$. Case 2 satisfies the requirement, so we accept time series $C_i$. The lower bounding ensures that our query results do not miss any candidates.

G. Speed up more with Secondary Lower Bounding

In subsection III-F, we have introduced the Lower Bounding (LB) for ECRD. However, this lower bounding seems not close to the ECRD because it is simply inherit the LB of previous distances. In this subsection, we propose a Secondary Lower Bounding (SLB). We use SLB in a second check when finding the nearest neighbor. Note that SLB is not a substitute for the LB, since we will use both of them. The SLB for ECRD is given as follows:

$$SLB(Q, C) = LB(Q, C) \times (CR(Q, C)^a + 1)$$

We can easily prove that SLB is a lower bounding of ECRD: $SLB(Q, C) \leq ECRD(Q, C)$. First, we have $LB(Q, C) \leq D_{\text{Original}}(Q, C)$. We multiply both sides with $CR(Q, C)^a + 1$, then we derive:

$$LB(Q, C) \times (CR(Q, C)^a + 1) \leq D_{\text{Original}}(Q, C) \times (CR(Q, C)^a + 1)$$

Therefore, $LB(Q, C) \times (CR(Q, C)^a + 1) \leq ECRD(Q, C)$.

Recall that in the algorithm to calculate ECRD, we first compute the compression rate factor $CR(Q, C)^a$, and then we compute the ECRD. So, before we compute the ECRD, we should perform a SLB check. Therefore, we compute SLB after calculating LB and before calculating ECRD. The schema here is that: first, we compute LB in reduced representation. If LB is lower than or equal to the best so far value, we continue to calculate SLB. Then we check if SLB is lower than or equal to the best so far. If this condition holds, we calculate the real distance with ECRD. Fig. 6 shows an outline of the algorithm to calculate ECRD with the support of LB and SLB. Note that in line 6, when calculating ECRD, we do not have to calculate compression rate CR, because we have this value calculated in line 3.

![Fig. 6. The CRD/ECRD algorithm with Secondary Lower Bounding.](image)

H. Relaxed-Triangular Inequality for ECRD

There are so many indexing and data mining algorithms for metric spaces that rely on triangular inequality to prune distance calculations. For example, in $k$-means clustering, let $X$ be a point and let $B$ and $C$ be centroids; we need to know that $D(X, C) \geq D(X, B)$ in order to avoid calculating the actual value of $D(X, C)$. Using triangular inequality, we know that $D(B, C) \leq D(B, X) + D(X, C)$. So, we can rewrite: $D(B, C) - D(X, B) \leq D(X, C)$. Note that if we have $D(B, C) \geq 2 \times D(X, B)$, consider the left-hand-side: $D(B, C) - D(X, B) \geq 2 \times D(X, B) = D(X, B)$. So we can conclude that: $D(X, B) \leq D(X, C)$. Therefore, we can safely prune the calculation of the distance $D(X, C)$.

CRD does not obey the triangular inequality; however it does obey a relaxed version of this property.

In this subsection, we would like to show that the ECRD can satisfy the Relaxed-Triangular Inequality in the same way as the Complexity-Invariant Distance [1]. That means we have the following inequality:

$$ECRD(A, B) \leq \delta(ED(A, C) + ECRD(B, C))$$

with $\delta = CR(A, B)^a + 1$.

The above inequality can be proved by using the triangular inequality on Euclidean Distance: $ED(A, B) \leq ED(A, C) + ED(B, C)$. Now we multiply both sides with compression rate of $A$ and $B$:

$$ECRD(A, B) \leq (CR(A, B)^a + 1) \times (ED(A, C) + ED(B, C))$$

We know that the compression rate $CR^a \geq 0$. So we have:

$$ED(A, C) \leq (CR(A, C)^a + 1) \times ED(A, C) = ECRD(A, C)$$
Now we can conclude that:

$$ED(A, B) \leq \delta(ED(A, C) + ED(B, C))$$

The relaxed triangular inequality can help ECRD to be supported by some index structures, such as M-Tree [14] or R-Tree [15].

IV. EXPERIMENTAL EVALUATION

We implemented our proposed CRD/ECRD and some other commonly used distances in time series, such as Euclidean Distance, DTW, and CID [1] with Matlab 2012 and conducted the experiments on the Intel Core i7-740QM 1.73 GHz, 4GB RAM PC. In the experiments, we compare CRD/ECRD with previous distances in terms of the accuracy rate. The accuracy rate is the ratio of the correctly classified on test data to the total number of test instances. In general, the higher the accuracy rate is, the better the distance function is.

A. Datasets

Our experiments were the time series classification problems conducted over the datasets from the UCR Time Series Data Mining archive [4]. There are 45 datasets used in these experiments. Each dataset includes a training set and a test set, and has about 2 to 50 classes. The length of each time series in each dataset is from 60 to 1882. In total, the evaluation includes 45 datasets from different domains, including medicine, engineering, astronomy, signal processing and many others. Due to space limit, we do not show details of the datasets. Interested reader can find more details about these datasets in [4].

B. Comparing CRD with previous distances

This subsection compares the CRD to Complexity-Invariant Distance (CID), Euclidean Distance (ED), and DTW in both cases with and without Sakoe-Chiba band constraint. In this experiment, we set the bias value $\alpha$ of CRD to 2 and 5; $\epsilon$ is set to 0.0001 as in some rarely happened situation, the whole values of time series along the time axis are the same; in this case, the entropy of time series is 0; therefore, $\epsilon$ can help to prevent dividing by 0 for the compression rate. The Sakoe-Chiba band on DTW is set as recommended in the UCR Time Series Data Mining website [4].

<table>
<thead>
<tr>
<th>Table I. SUMMARY OF EXPERIMENTAL RESULTS OF CRD ($\alpha = 2$) COMPARED WITH OTHER DISTANCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRD vs. CID</td>
</tr>
<tr>
<td>Wins</td>
</tr>
<tr>
<td>Draws</td>
</tr>
<tr>
<td>Loses</td>
</tr>
</tbody>
</table>

Table I shows a summary of experimental results of CRD ($\alpha = 2$) compared to the three previous distances. As we can see, CRD outperforms the CID in 38 datasets, draws in 2 datasets, and loses in 5 datasets. In comparison to ED, CRD wins ED in 41 datasets, draws in 1 dataset, and loses in 3 datasets. In comparison to DTW (without Sakoe-Chiba band constraint), CRD wins 28 and loses in 17 datasets. And compared to DTW($r$) (with Sakoe-Chiba band $r$ constraint), CRD outperforms in 27 datasets, draws in 1 dataset, and loses in 17 datasets. These results show that CRD is much better than CID and ED in a vast majority of datasets, and outperforms DTW in nearly two-third of the number of datasets. Notice that the complexity of CRD, CID, and ED is $O(n)$, while the complexity of DTW is $O(n^2)$.

In Fig. 7, we plot four scatter graphs to compare CRD with the other distances. Each circle point in the graph represents one dataset, with its X-axis value being the accuracy of CID, ED or DTW distance, and its Y-axis value being its CRD accuracy. In general, the more circle points deviate on the upper triangle, the better the CRD is.

<table>
<thead>
<tr>
<th>Table II. SUMMARY OF EXPERIMENTAL RESULTS OF CRD ($\alpha = 5$) COMPARED WITH OTHER DISTANCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRD vs. CID</td>
</tr>
<tr>
<td>Wins</td>
</tr>
<tr>
<td>Draws</td>
</tr>
<tr>
<td>Loses</td>
</tr>
</tbody>
</table>

Table II reports the summary of experimental results of CRD $\alpha = 5$ compared to the previous distances. The results do not change much when compared to those of CRD with $\alpha = 2$, the CRD also outperforms the previous distances. Fig. 8 show scatter graphs to compare CRD ($\alpha = 5$) with the other distances.

The results in these experiment shows that our CRD outperforms ED and especially outperforms CID remarkably. Besides, even though we do not want to claim that CRD is
better than DTW, the CRD still outperforms DTW in almost two-third of the datasets.

![Figure 8](image1.png)

Fig. 8. Scatter graphs for comparing CRD ($\alpha = 5$) to the other distances. (a) CRD compared with CID, (b) CRD compared with ED, (c) CRD compared with DTW (with Sakoe-Chiba band), (d) CRD compared with DTW (without Sakoe-Chiba band).

We attribute the high performance of CRD in comparison to CID to the following reason. Both CID and CRD are based on Euclidean Distance, but the way to compute the complexity correction factor in CID is quite different from the way to compute the compression rate factor in CRD. The computation of compression rate factor in CRD is consistently and precisely based on Minimum Description Length principle which can reveal well the similarity of two time series while the computation of complexity correction factor in CID is a rough heuristic which sometimes can not reflect the actual complexity of the time series as well as the similarity of two time series.

The superiority of CRD to Euclidean Distance is quite obvious. The compression rate factor included in CRD distance improves the similarity of the two time series measured by Euclidean distance.

CRD beats DTW for two-third of the datasets. This is a surprising result since it is well-known that several time series domains can benefit from some degree of nonlinear time stretching in DTW, a problem that is not directly tackled by CRD. This is in fact a good news since the time complexity of DTW distance computation is $O(n^2)$ while the time complexity of CRD distance is $O(n)$.

C. Comparing ECRD with previous distances

In this subsection, we also compare the ECRD with Complexity-Invariant Distance (CID), Euclidean Distance (ED), and DTW Distance (DTW) in both cases with and without Sakoe-Chiba band constraint. We set the bias value $\alpha$ of ECRD to 2 and 5. The Sakoe-Chiba band on DTW is set as recommended in the UCR Time Series Data Mining website [4].

![Figure 9](image2.png)

Fig. 9. Scatter graphs for comparing ECRD ($\alpha = 2$) to the other distances. (a) ECRD compared with CID, (b) ECRD compared with ED, (c) ECRD compared with DTW (with Sakoe-Chiba band), (d) ECRD compared with DTW (without Sakoe-Chiba band).

Table III shows a summary of experimental results of ECRD ($\alpha = 2$) compared to previous distances. As we can see, ECRD outperforms the CID in 38 datasets, draws in 1 dataset, and loses in 6 datasets. Compared to ED, ECRD outperforms ED in 43 datasets, and loses in 2 datasets. Compared to DTW (without Sakoe-Chiba band constraint), ECRD wins in 28 datasets and loses in 17 datasets. And in comparison to DTW($r$) (with Sakoe-Chiba band $r$ constraint), ECRD outperforms in 29 and loses in 16 datasets. This results shows that ECRD is much better than CID and ED in a vast majority of datasets, and outperforms DTW in nearly two-third of the number of datasets. Notice that the complexity of ECRD, CID, and ED is $O(n)$, while the complexity of DTW is $O(n^2)$.

Table IV shows a summary of experimental results of ECRD ($\alpha = 5$) compared to previous distances. In Table IV, we report the summary of experimental results of ECRD with $\alpha = 5$ compared to the pervious distances. For $\alpha$
In this work, we have proposed a novel distance measure for times series, called the Compression Rate Distance and its extended version, Extended Compression Rate Distance. Our distances can be computed in a linear time $O(n)$. We also show that our Extended Compression Rate Distance can satisfy the Lower Bounding condition and Relaxed-Triangular Inequality. Besides, we also propose a Secondary Lower Bounding and Early Abandoning for our distance. Experimental results show that our proposed distance can provide a better performance for time series classification in comparison to Euclidean distance, CID and DTW.

As for future work, we plan to apply our distance to other problems such as clustering, semi-supervised classification, finding motif, anomaly detection, etc. We also intend to adapt our distance in time series similarity search with the support of some index structures.

ACKNOWLEDGMENT

We would like to thank Prof. Eamonn Keogh for kindly sharing the datasets and some initial Matlab codes which help us in constructing the experiments in this work.

REFERENCES


TABLE V. The Error Rates (Error_Rate = 1 – Accuracy_Rate) on 1-NN Classification

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ED</th>
<th>CID</th>
<th>DTW</th>
<th>DTW (r)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>r^2</td>
<td>Sakoe-Chiba band (percent of time series length)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50words</td>
<td>0.36923</td>
<td>0.33626</td>
<td>0.30998</td>
<td>0.24176 (6)</td>
</tr>
<tr>
<td>Adiac</td>
<td>0.38875</td>
<td>0.3734</td>
<td>0.39642</td>
<td>0.3913 (3)</td>
</tr>
<tr>
<td>Beef</td>
<td>0.46667</td>
<td>0.5</td>
<td>0.46667(0)</td>
<td>0.6</td>
</tr>
<tr>
<td>CBF</td>
<td>0.14778</td>
<td>0.015556</td>
<td>0.003333</td>
<td>0.0044444 (11)</td>
</tr>
<tr>
<td>ChlorineConcentration</td>
<td>0.35</td>
<td>0.3513</td>
<td>0.35156</td>
<td>0.35 (0)</td>
</tr>
<tr>
<td>CitiClustE C G_toro</td>
<td>0.1029</td>
<td>0.084058</td>
<td>0.34928</td>
<td>0.064493 (1)</td>
</tr>
<tr>
<td>CofiA</td>
<td>0.25</td>
<td>0.17857</td>
<td>0.17857</td>
<td>0.17857 (3)</td>
</tr>
<tr>
<td>Cricket X</td>
<td>0.42564</td>
<td>0.34615</td>
<td>0.22308</td>
<td>0.2339 (7)</td>
</tr>
<tr>
<td>Cricket Y</td>
<td>0.35641</td>
<td>0.31538</td>
<td>0.20769</td>
<td>0.19744 (17)</td>
</tr>
<tr>
<td>Cricket Z</td>
<td>0.37949</td>
<td>0.31538</td>
<td>0.20769</td>
<td>0.19744 (17)</td>
</tr>
<tr>
<td>DiatomSizeReduction</td>
<td>0.065359</td>
<td>0.065359</td>
<td>0.05268</td>
<td>0.065359 (0)</td>
</tr>
<tr>
<td>ECG200</td>
<td>0.12</td>
<td>0.11</td>
<td>0.23</td>
<td>0.12 (10)</td>
</tr>
<tr>
<td>ECGFiveDays</td>
<td>0.20325</td>
<td>0.21835</td>
<td>0.23229</td>
<td>0.20325 (0)</td>
</tr>
<tr>
<td>FaceAll</td>
<td>0.28839</td>
<td>0.26923</td>
<td>0.19231</td>
<td>0.19172 (3)</td>
</tr>
<tr>
<td>FaceFour</td>
<td>0.21591</td>
<td>0.19318</td>
<td>0.17045</td>
<td>0.11642 (2)</td>
</tr>
<tr>
<td>FacesUCR</td>
<td>0.23073</td>
<td>0.23463</td>
<td>0.095122</td>
<td>0.087805 (12)</td>
</tr>
<tr>
<td>FISH</td>
<td>0.21714</td>
<td>0.22286</td>
<td>0.16571</td>
<td>0.16 (4)</td>
</tr>
<tr>
<td>Gun_Point</td>
<td>0.086667</td>
<td>0.073333</td>
<td>0.093133</td>
<td>0.086667 (0)</td>
</tr>
<tr>
<td>Haptics</td>
<td>0.62987</td>
<td>0.58442</td>
<td>0.62338</td>
<td>0.58766 (2)</td>
</tr>
<tr>
<td>InlineRacket</td>
<td>0.65818</td>
<td>0.62909</td>
<td>0.61636</td>
<td>0.61091 (14)</td>
</tr>
<tr>
<td>ItalyPowerDemand</td>
<td>0.044704</td>
<td>0.043732</td>
<td>0.049563</td>
<td>0.044704 (0)</td>
</tr>
<tr>
<td>Lighting2</td>
<td>0.2459</td>
<td>0.2459</td>
<td>0.13115</td>
<td>0.13115 (6)</td>
</tr>
<tr>
<td>Lighting7</td>
<td>0.42466</td>
<td>0.3726</td>
<td>0.27397</td>
<td>0.28767 (5)</td>
</tr>
<tr>
<td>MALLAT</td>
<td>0.085714</td>
<td>0.074627</td>
<td>0.066098</td>
<td>0.085714 (0)</td>
</tr>
<tr>
<td>MedicalImages</td>
<td>0.31579</td>
<td>0.30921</td>
<td>0.26447</td>
<td>0.25263 (20)</td>
</tr>
<tr>
<td>MoteStrain</td>
<td>0.12141</td>
<td>0.21166</td>
<td>0.16454</td>
<td>0.13419 (1)</td>
</tr>
<tr>
<td>OliveOil</td>
<td>0.13333</td>
<td>0.13333</td>
<td>0.13333</td>
<td>0.16667 (1)</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>0.48347</td>
<td>0.43802</td>
<td>0.40909</td>
<td>0.3843 (7)</td>
</tr>
<tr>
<td>SonyAIBORobotSurfaceC ell</td>
<td>0.14061</td>
<td>0.12227</td>
<td>0.16894</td>
<td>0.14061 (0)</td>
</tr>
<tr>
<td>SonyAIBORobotSurface</td>
<td>0.30449</td>
<td>0.18469</td>
<td>0.27454</td>
<td>0.30449 (0)</td>
</tr>
<tr>
<td>StarLightCurves</td>
<td>0.15117</td>
<td>0.057067</td>
<td>0.093371</td>
<td>0.094706 (16)</td>
</tr>
<tr>
<td>SwedishLeaf</td>
<td>0.2112</td>
<td>0.1232</td>
<td>0.208</td>
<td>0.1536 (2)</td>
</tr>
<tr>
<td>Symbols</td>
<td>0.1005</td>
<td>0.08442</td>
<td>0.05025</td>
<td>0.062321 (8)</td>
</tr>
<tr>
<td>synthetic_control</td>
<td>0.12</td>
<td>0.05</td>
<td>0.006667</td>
<td>0.006667 (6)</td>
</tr>
<tr>
<td>Trace</td>
<td>0.24</td>
<td>0.14</td>
<td>0.01</td>
<td>0.01 (3)</td>
</tr>
<tr>
<td>TwoLeadECG</td>
<td>0.25285</td>
<td>0.23178</td>
<td>0.095698</td>
<td>0.11589 (5)</td>
</tr>
<tr>
<td>Two Patterns</td>
<td>0.09325</td>
<td>0.12125</td>
<td>0.00154</td>
<td>0.01</td>
</tr>
<tr>
<td>uWaveGestureLibrary_X</td>
<td>0.26075</td>
<td>0.23841</td>
<td>0.27247</td>
<td>0.22669 (4)</td>
</tr>
<tr>
<td>uWaveGestureLibrary_Y</td>
<td>0.33836</td>
<td>0.29034</td>
<td>0.366</td>
<td>0.30095 (4)</td>
</tr>
<tr>
<td>uWaveGestureLibrary_Z</td>
<td>0.35036</td>
<td>0.29146</td>
<td>0.34171</td>
<td>0.32217 (6)</td>
</tr>
<tr>
<td>water</td>
<td>0.0045425</td>
<td>0.063271</td>
<td>0.020117</td>
<td>0.0045425 (1)</td>
</tr>
<tr>
<td>WordsSynonyms</td>
<td>0.38245</td>
<td>0.35747</td>
<td>0.3511</td>
<td>0.25335 (8)</td>
</tr>
<tr>
<td>yoga</td>
<td>0.16967</td>
<td>0.16433</td>
<td>0.16367</td>
<td>0.15533 (2)</td>
</tr>
<tr>
<td>NonInvasiveFetalECG_Thorax1</td>
<td>0.17099</td>
<td>0.16285</td>
<td>0.20967</td>
<td>0.18931 (1)</td>
</tr>
<tr>
<td>NonInvasiveFetalECG_Thorax2</td>
<td>0.1201</td>
<td>0.12061</td>
<td>0.13537</td>
<td>0.12519 (1)</td>
</tr>
</tbody>
</table>