Elastic Similarity Measures for Multivariate Time Series Classification

Ahmed Shifaz, Charlotte Pelletier, François Petitjean, Geoffrey I Webb

Data Mining And Knowledge Discovery manuscript No.
(will be inserted by the editor)

Elastic Similarity Measures for Multivariate Time Series Classification

Ahmed Shifaz[1] · Charlotte Pelletier[1,2] · François Petitjean[1] · Geoffrey I. Webb[1]

Received: date / Accepted: date

Abstract Elastic similarity measures are a class of similarity measures specifically designed to work with time series data. When scoring the similarity between two time series, they allow points that do not correspond in timestamps to be aligned. This can compensate for misalignments in the time axis of time series data, and for similar processes that proceed at variable and differing paces. Elastic similarity measures are widely used in machine learning tasks such as classification, clustering and outlier detection when using time series data.

There is a multitude of research on various univariate elastic similarity measures. However, except for multivariate versions of the well known Dynamic Time Warping (DTW) there is a lack of work to generalise other similarity measures for multivariate cases. This paper adapts two existing strategies used in multivariate DTW, namely, Independent and Dependent DTW, to several commonly used elastic similarity measures.

Using 23 datasets from the University of East Anglia (UEA) multivariate archive, for nearest neighbour classification, we demonstrate that each measure outperforms all others on at least one dataset and that there are datasets for which either the dependent versions of all measures are more accurate than their independent counterparts or vice versa. This latter finding suggests that these differences arise from a fundamental property of the data. We also show that an ensemble of such nearest neighbour classifiers is highly competitive with other state-of-the-art multivariate time series classifiers.

This research has been supported by Australian Research Council grant DP210100072.

[1]Faculty of Information Technology
25 Exhibition Walk
Monash University, Melbourne
VIC 3800, Australia
[2] IRISA, UMR CNRS 6074
Univ. Bretagne Sud
Campus de Tohannic
BP 573, 56 000 Vannes, France
E-mail: {ahmed.shifaz,francois.petitjean,geoff.webb}@monash.edu,
charlotte.pelletier@univ-ubs.fr

Keywords multivariate time series classification, similarity measures, independent measures, dependent measures, multivariate elastic ensemble

1 Introduction

Elastic similarity measures, of which Dynamic Time Warping (DTW) (Sakoe and Chiba, 1978) is a well known example, are a key tool in many forms of time series analytics. Examples of their application include clustering (Berndt and Clifford, 1994), anomaly detection (Izakian and Pedrycz, 2014), nearest neighbour classification (Lines and Bagnall, 2015; Bagnall et al, 2017) and state-of-the-art time series classifiers such as HIVE-COTE 1.0 (Hierarchical Vote Collective of Transformation-based Ensembles) (Bagnall et al, 2020) and TS-CHIEF (Time Series Combination of Heterogeneous and Integrated Embeddings Forest) (Shifaz et al, 2020).

There are numerous elastic similarity measures, and it has been demonstrated that each outperforms the others on different types of tasks (Lines and Bagnall, 2015). Many of these measures have only been defined for univariate time series. However, many significant tasks involve multivariate time series. This paper extends to the multivariate case seven univariate elastic similarity measures, and demonstrates that each supports strong nearest neighbour classification of different datasets.

One elastic measure that has previously been extended to the multivariate case is DTW (Shokoohi-Yekta et al, 2017). That work identified two key strategies for such extension. The *independent* strategy applies the univariate measure to each dimension and then sums the resulting distances. The *dependent* strategy treats each time step as a multi-dimensional point. DTW is then applied using Euclidean distances between these multidimensional points. It was shown that each of these strategies outperformed the other on some tasks.

We develop methods for applying these two strategies to seven further key univariate similarity measures. One of the significant outcomes is that we demonstrate that there are some tasks for which the independent strategy is superior across all measures and others for which the dependent strategy is better. This establishes a fundamental relationship between the two strategies and different tasks, countering the possibility that differing performance for the two strategies when applied to DTW might have been coincidental.

We develop a multivariate version of the Elastic Ensemble (Lines and Bagnall, 2015), and demonstrate that this ensemble of nearest neighbour classifiers using all multivariate measures provides accuracy competitive with the state of the art.

We organise the rest of the paper as follow: Section 2 presents key definitions and a brief review of existing methods. In Section 3 we present our new multivariate similarity measures. Section 4 presents experiments on the University of East Anglia (UEA) multivariate archive, and includes discussion of the implications of the results. The experiments are set in the domain of multivariate time series classification. Finally, we draw conclusions in Section 5, with suggestions for future work.

2 Related Work

In this section, we first present the main definitions used in this paper. Then we present a summary of the methods used for univariate Time Series Classification (TSC) followed by a summary of the methods used for multivariate Time Series Classification.

2.1 Main Definitions

Definition 1 Time Series

A time series T of length L is an ordered sequence of L time-value pairs $T = \langle (t_1, \boldsymbol{x}_1), \cdots, (t_L, \boldsymbol{x}_L) \rangle$, where t_i is the timestamp at sequence index i, $i \in \{1, \cdots, L\}$, and $\boldsymbol{x}_i$ is a D-dimensional point representing observations of D real-valued variables or features at timestamp t_i. Each time point $\boldsymbol{x}_i \in \mathbb{R}^D$ is defined by $\{x_i^1, \cdots, x_i^d, \cdots, x_i^D\}$. Usually, timestamps t_i are assumed to be equidistant, and thus omitted, which results in a simpler representation where $T = \langle \boldsymbol{x}_1, \cdots, \boldsymbol{x}_L \rangle$.

A univariate (or single-dimensional) time series is a special case where a single variable is observed ($D = 1$). Therefore, $\boldsymbol{x}_i$ is a scalar, and consequently, $T = \langle x_1, \cdots, x_L \rangle$.

Definition 2 Time Series Dataset

A labelled time series dataset $\mathcal{S}$ consists of N labelled time series indexed by n, where $n \in \{1, \cdots, N\}$. Each time series T_n in $\mathcal{S}$ is associated with a label $y_n \in \{1, \cdots, c\}$, where c is the number of classes.

Definition 3 Time Series Classification

In a Time Series Classification (TSC) task, a time series classifier is trained on a labelled time series dataset, and then used to predict labels of unlabelled time series. The classifier is a predictive mapping function that maps from the space of input variables to discrete class labels.

In this paper, to perform TSC tasks, we use 1-nearest neighbour (1-NN) classifiers, which use time series specific *similarity measures* to compute the nearest neighbours between each time series.

Definition 4 Similarity Measure

A similarity measure computes a real value that quantifies the degree of similarity between two sets of values. For two time series Q and C, the similarity measure M is defined as

$$M(Q, C) \to \mathbb{R} \tag{1}$$

All similarity measures used in our work compute a non-negative real value (i.e. $\mathbb{R}_0^+$).

In many cases, a similarity measure is more useful with the following four properties, since they help with indexing and fast look up from large time series databases (Chen and Ng, 2004). Similarity measure M is considered to be a *metric* if it has the following properties:

1. Non-negativity: $M(Q, C) \geq 0$,
2. Identity: $M(Q, C) = 0$, if and only if $Q = C$,

3. Symmetry: $D(Q, C) = D(C, Q)$,
4. Triangle Inequality: $D(Q, C) \leq D(Q, T) + D(T, C)$ for any time series Q, C and T.

2.2 Summary of Univariate TSC

A comprehensive review of the most common univariate TSC methods developed prior to 2017 can be found in (Bagnall et al, 2017). Here we summarise the existing univariate TSC methods using a traditionally used method of categorisation as follows:

- *Similarity-based* methods which compare whole time series using similarity measures, usually in conjunction with 1-NN classifiers. Particularly, 1-NN with DTW (Sakoe and Chiba, 1978; Itakura, 1975) was considered as the de facto standard for univariate TSC. More accurate similarity-based methods combine multiple measures, which include 1-NN-based ensemble Elastic Ensemble (EE) (Lines and Bagnall, 2015), and tree-based ensemble Proximity Forest (PF) (Lucas et al, 2019). In Section 3 we will explore more details of several similarity measures used in TSC.
- *Interval-based* methods use subseries or transformations of subseries in conjunction with its location information as discriminatory features. Examples include Time Series Forest (TSF) (Deng et al, 2013), Random Interval Spectral Ensemble (RISE) (Lines et al, 2018), and Canonical Interval Forest (CIF) (Middlehurst et al, 2020).
- *Shapelet-based* methods extract or learn a set of discriminative subseries for each class which are then used as search keys for the particular classes. The presence, absence or distance of a shapelet to other time series could be used as discriminative information for classification. In shapelet-based methods, information from the shape of the subseries is used without its location information. Examples include Shapelet Transform (ST) (Hills et al, 2014) and Generalized Random Shapelet Forest (gRSF) (Karlsson et al, 2016).
- *Dictionary-based* methods transform time series into a bag-of-word model. The series is either discretized in time domain such as in Bag of Patterns (BoP) (Lin et al, 2012) or it is transformed into frequency domain such as in Bag-of-SFA-Symbols (BOSS) (Schäfer, 2015), and Word eXtrAction for time SEries cLassification (WEASEL) (Schäfer and Leser, 2017a). In addition, Multiple Representation Sequence Learner (MrSEQL) (Le Nguyen et al, 2019) is also another recently introduced dictionary-based classifier, which is more accurate than WEASEL but uses less computational resources.
- *Transformation-based* methods transform the time series using a transformation function and then use a general purpose classifier. A notable example is RandOm Convolutional KErnel Transform (ROCKET) (Dempster et al, 2020) which uses random convolutions to transform the data, and then uses logistic regression for classification.
- *Combinations of Methods* These methods combine previously mentioned methods to form ensembles. Examples include HIVE-COTE (Hierarchical Vote Collective of Transformation-based Ensembles)(Lines et al, 2018) which combines EE, ST, RISE and BOSS in one ensemble, and TS-CHIEF (Time Series Combination of Heterogeneous and Integrated Embeddings Forest) (Shifaz et al,

2020), which is a tree-based ensemble where the tree nodes use similarity, dictionary or interval-based splitters.
– *Deep-learning-based* methods can be divided into two main types of architectures: (1) based on recurrence (Gallicchio and Micheli, 2017) , or (2) based on temporal convolutions such as Residual Neural Network (ResNet) (Wang et al, 2017) and InceptionTime (Fawaz et al, 2020). A recent review of deep learning methods shows that architectures that use temporal convolutions show higher accuracy (Fawaz et al, 2019).

Currently, HIVE-COTE, TS-CHIEF and ROCKET are considered to be the state-of-the-art classifiers for TSC (Bagnall et al, 2020). While all three methods are competitive in accuracy (*i.e.* statistically indistinguishable), TS-CHIEF leads in terms of accuracy and ROCKET leads in terms of speed.

We also note that the latest version of HIVE-COTE, which does not include EE, called HIVE-COTE 1.0, has significantly improved its speed, but is still behind on accuracy with respect to ROCKET and TS-CHIEF (Bagnall et al, 2020).

2.3 Summary of Multivariate TSC

One elastic similarity measure that has previously been extended to the multivariate case is DTW (Shokoohi-Yekta et al, 2017). That work identified two key strategies for such extension. The *independent* strategy applies the univariate measure to each dimension and then sums the resulting distances. The *dependent* strategy treats each time step as a multi-dimensional point. DTW is then applied on the Euclidean distances between these multidimensional points.

Figure 1 shows an illustration of independent DTW (DTW_I) and dependent DTW (DTW_D). We present formal definitions of these two methods in Section 3.

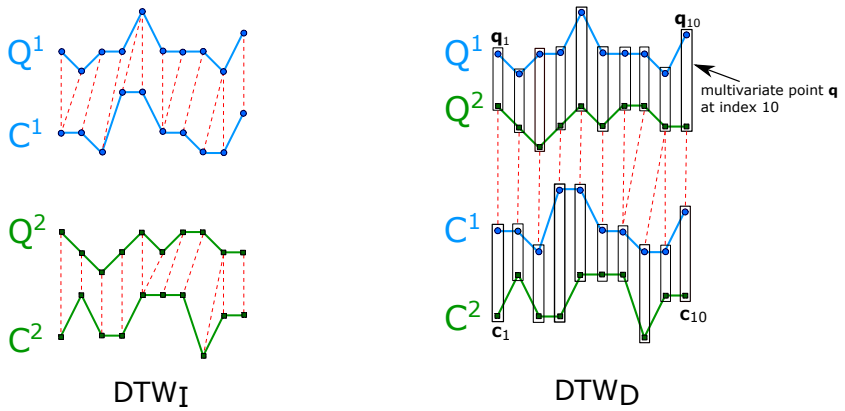

Fig. 1 Independent DTW (DTW_I, on the left) and dependent DTW (DTW_D, on the right). Dimension 1 in series Q and C is shown in blue color, and the dimension 2 is shown in green color.

A variety of other methods also support multivariate TSC. These include WEASEL with a Multivariate Unsupervised Symbols and dErivatives (MUSE)

(a.k.a WEASEL+MUSE, or simply MUSE) (Schäfer and Leser, 2017b) and time contracted Bag-of-SFA-Symbols (CBOSS) (Middlehurst et al, 2019), which are multivariate version of dictionary-based methods that convert time series into a bag-of-word model before using it for classification.

Shapelet-based methods include Generalized Random Shapelet Forest (gRSF) (Karlsson et al, 2016), and time contracted Shapelet Transform (STC) (Bagnall et al, 2020). According to a recent review, STC is the current most accurate multivariate method that uses shapelets (which is ranked below ResNet) (Ruiz et al, 2020).

Interval-based methods that extract location dependent subseries include Random Interval Spectral Ensemble (RISE) (Lines et al, 2018) and Time Series Forest (TSF) (Deng et al, 2013). Currently, the accuracy of these two methods are below DTW_I. However, a recently introduced classifier CIF (Middlehurst et al, 2020) has shown promising results for multivariate classification.

Deep learning methods include Time Series Attentional Prototype Network (TapNet) (Zhang et al, 2020), ResNet and InceptionTime.

A recent paper for multivariare TSC algorithms (Ruiz et al, 2020) presents a comparison of most of these methods on the same set of datasets we use in this paper. The review compared 12 classifiers on 20 UEA multivariate datasets with equal length and which completed in a reasonable time. They found that the most accurate multivariate TSC algorithms are ROCKET (ranked 3.8), InceptionTime (ranked 5.15), MUSE (ranked 5.25), CIF (ranked 5.85) followed by HIVE-COTE (ranked 5.9) in that order (Ruiz et al, 2020, Figure 12).

3 Similarity Measures

In this section we present details of the proposed similarity measures. For this study, we extend the set of similarity measures used in EE and PF (and thus TS-CHIEF and some versions of HIVE-COTE).

The independent strategy proposed by Shokoohi-Yekta et al (2017) extends directly to any univariate measure as follows.

Definition 5 Independent Measures

For any univariate measure $\gamma(Q', C') \rightarrow \mathbb{R}$ and multivariate series Q and C, an independent multivariate derivative of γ is defined by,

$$Ind(\gamma, Q, C) = \left(\sum_{d=1}^{D} \left| \gamma(Q^d, C^d) \right|^p \right)^{1/p} \tag{2}$$

We compute the distance between Q and C separately for each dimension, and then take the p-norm of the results. The parameter p is set to 1 in Shokoohi-Yekta et al (2017). Here, Q^d (or C^d) represents the univariate time series of dimension d such that $Q^d =< q_1^d, \cdots, q_L^d >$ (or $C^d =< c_1^d, \cdots, c_L^d >$).

For ease of reference we indicate an independent variant of a multivariate measure by adding the subscript I. Hence, $DTW_I(Q, C) = Ind(DTW, Q, C)$, $WDTW_I(Q, C) = Ind(WDTW, Q, C)$ and so forth.

However, in most cases it requires more than such a simple formulation to derive a dependent variant, and hence we below introduce each of the univariate measures together with our proposed dependent variant.

3.1 *Lp* Distance (Lp)

3.1.1 Univariate Lp Distance

The simplest way to calculate similarity between two time series is to use *Lp* distance, also known as the Minkowski distance.

Let us denote by Q and C two univariate ($D = 1$) time series of length L where q_i and c_i are scalar values at time point i from the two time series. Equation 3 formulates the *Lp* distance between Q and C.

$$Lp(Q, C) = \left(\sum_{i}^{L} |q_i - c_i|^p \right)^{1/p} \tag{3}$$

The parameter p is the order of the distance. The most commonly used distances are: L_1 distance (Manhattan distance) and L_2 distance (Euclidean distance).

In the context of TSC, *Lp* distances are of limited use because they cannot align two series that are misaligned in the time dimension, since they compute one-to-one differences between corresponding points only.

For example, in an electrocardiogram (ECG) signal, two measurements from a patient at different times may produce slightly different time series which belong the same class (e.g certain heart condition). Ideally, if they belong to the same class, an effective similarity measure should account for such "misalignments" in the time axis, while computing the similarity.

To tackle this issue, *elastic* similarity measures such as DTW were developed. Elastic similarity measures are designed to compensate for temporal misalignments in time series that might be due to stretched or shrunken subsequences. From Section 3.2 to 3.6 we will present various *elastic* similarity measures.

3.1.2 Multivariate Lp Distance

We here show that Independent *Lp* distance(Lp_I) and Dependent *Lp* distance (Lp_D) are identical.

Definition 6 Independent *Lp* Distance (Lp_I)

In this case, we simply compute the *Lp* distance between Q and C separately for each dimension, and then take the p-norm of the results.

$$\begin{aligned} Lp_I(Q, C) &= \left(\sum_{d=1}^{D} \left| Lp(Q^d, C^d) \right|^p \right)^{1/p} \\ &= \left(\sum_{d=1}^{D} \sum_{i=1}^{L} \left| q_i^d - c_i^d \right|^p \right)^{1/p} \end{aligned} \tag{4}$$

Definition 7 Dependent *Lp* Distance (Lp_D)

In this case, we compute the Euclidean distance between each multidimensional point ($\mathbf{q_i} \in \mathbb{R}^D$ and $\mathbf{c_i} \in \mathbb{R}^D$ as defined in Definition 1), and take the p-norm of the results.

$$
\begin{aligned}
Lp_D(Q,C) &= \left(\sum_{i=1}^{L} |Lp(\mathbf{q_i} - \mathbf{c_i})|^p \right)^{1/p} \\
&= \left(\sum_{i=1}^{L} \sum_{d=1}^{D} \left| q_i^d - c_i^d \right|^p \right)^{1/p}
\end{aligned}
\tag{5}
$$

Consequently both the independent and the dependent versions of the *non-elastic Lp* distance will produce the same result. However, as we shall see in the following sections, for *elastic* similarity measures the two strategies are substantially different.

3.2 Dynamic Time Warping (DTW) and Related Measures

3.2.1 Univariate DTW

The most widely used *elastic* similarity measure is DTW (Sakoe and Chiba, 1978). By contrast to measures such as the Euclidean distance, DTW is an elastic similarity measure, which allows one-to-many alignment ("warping") of points between two time series. For many years, 1-NN with DTW was considered as the traditional benchmark algorithm for TSC (Ding et al, 2008).

DTW is efficiently solved using a dynamic programming technique. Let Δ_{DTW} be an L-by-L dynamic programming cost matrix in which the element (i, j) of the matrix is defined as the squared Euclidean distance between two corresponding points q_i and c_j – *i.e.* $\Delta_{DTW}(i,j) = (q_i - c_j)^2$ and the minimum of the cumulative distances of the previous points. Equation 6 defines element (i, j) of the cost matrix as follows:

$$
\Delta_{DTW}(i,j) = (q_i - c_j)^2 + min \begin{cases} \Delta_{DTW}(i-1, j-1) & \textbf{if } i,j > 0 \\ \Delta_{DTW}(i, j-1) \\ \Delta_{DTW}(i-1, j). \end{cases}
\tag{6}
$$

The cost matrix represents the alignment of the two series as according to the DTW algorithm. DTW between two series Q and C is the accumulated cost in the last element of the cost matrix (*i.e.* $i, j = L$) as defined in Equation 7:

$$
DTW(Q,C) = \Delta_{DTW(L,L)}. \tag{7}
$$

DTW has a parameter called "window size" (w) which helps to prevent pathological warpings by constraining the maximum allowed warping distance. For example, when $w = 0$, DTW produces a one-to-one alignment which is equivalent to the Euclidean distance. A larger warping window allows one-to-many alignments where points from one series can match points from the other series over longer time frames. Therefore, w controls the *elasticity* of the similarity measure.

Different methods have been used to select the parameter w. In some methods, such as EE, and HIVE-COTE, w is selected using leave-one-out-cross-validation. Some algorithms select the window size randomly (*e.g.* PF and TS-CHIEF select window sizes from the uniform distribution $U(0, L/4)$).

Parameter w also improves the computational efficiency, since in most cases, the ideal w is much less than the length of the series (Tan et al, 2018). When w is small, DTW runs relatively fast, especially with lower bounding, and early abandoning techniques (Keogh et al, 2009; Lemire, 2009; Tan et al, 2017; Herrmann and Webb, 2020). Time complexity to calculate DTW with a warping window is $O(L \cdot w)$, instead of $O(L^2)$ for the full DTW.

3.2.2 Dependent Multivariate DTW

Definition 8 Dependent DTW (DTW_D)

Dependent DTW (DTW_D) uses all dimensions together when computing the point-wise distance between each point in the two time series. In this method, for each point in the series, DTW is allowed to warp across the dimensions.

In this case, the squared Euclidean distance between two univariate points – $(q_i - c_j)^2$ – in Equation 6 is replaced with two multivariate points $\mathbf{q_i}$ and $\mathbf{c_j}$ as in Equation 8.

$$L_2(\mathbf{q_i}, \mathbf{c_j})^2 = \sum_{d=1}^{D} (q_i^d - c_j^d)^2 \tag{8}$$

3.2.3 Derivative DTW (DDTW)

Derivative DTW (DDTW) is a variation of DTW, which computes DTW on the first derivatives of time series. Keogh and Pazzani (2001) developed this version to mitigate some pathological warpings. Particularly, cases where DTW tries to explain variability in the time series values by warping time-axis, and cases where DTW misaligns features in one series which are higher or lower than its corresponding features in the other series. The derivative transformation of a univariate time point q_i' is defined as:

$$q_i' = \frac{(q_i - q_{i-1} + (q_{i+1} - q_{i-1})/2)}{2} \tag{9}$$

Note that q_i' is not defined for the first and last element of the time series. Once the two series have been transformed, DTW is computed as in Equation 7.

Multivariate versions of DDTW are very straightforward to implement. We calculate the derivatives separately for each dimension, and then use Equations 2 and 8 to compute from the derivatives independent DDTW ($DDTW_I$) and dependent DDTW ($DDTW_D$), respectively.

3.2.4 Weighted DTW (WDTW)

Weighted DTW (WDTW) is another variation of DTW, proposed by Jeong et al (2011), which uses a "soft warping window" in contrast to the fixed warping window sized used in classic DTW. WDTW penalises large warpings by assigning a non-linear multiplicative weight w to the warpings using the modified logistic function in Equation 10,

$$weight_{|i-j|} = \frac{weight_{max}}{1 + e^{-g \cdot ((|i-j| - L)/2)}}, \tag{10}$$

where $weight_{max}$ is the upper bound on the weight (set to 1), L is the series length and g is the parameter that controls the penalty level for large warpings. Larger values of g increases the penalty for warping.

When creating the dynamic programming distance matrix for WDTW Δ_{WDTW}, weight penalty $weight_{|i-j|}$ for a warping distance of $|i-j|$ is applied, so that the (i,j)-th entry in the matrix is $\Delta_{WDTW}(i,j) = weight_{|i-j|} \cdot (q_i - c_i)^2$. Therefore, the new equation for WDTW is defined as,

$$\begin{aligned} \Delta_{WDTW}(i,j) = & \, weight_{|i-j|} \cdot (q_i - c_j)^2 \\ & + min \begin{cases} \Delta_{WDTW}(i-1,j-1) & \textbf{if } i,j > 0 \\ \Delta_{WDTW}(i,j-1) \\ \Delta_{WDTW}(i-1,j). \end{cases} \end{aligned} \tag{11}$$

$$WDTW(Q,C) = \Delta_{WDTW(L,L)}. \tag{12}$$

Parameter g may be selected using leave-one-out cross-validation as in EE and HIVE-COTE, or selected randomly as in PF and TS-CHIEF ($g \sim U(0,1)$).

Since WDTW does not use a constrained warping window (*i.e.* the maximum warping distance $|i-j|$ may be as large as L), its time complexity is $O(L^2)$, which is higher than DTW.

3.2.5 Dependent Multivariate WDTW

Definition 9 Dependent WDTW

The dependent version of WDTW simply inserts the weight into DTW_D. We define Dependent WDTW ($WDTW_D$) as,

$$\begin{aligned} \Delta_{WDTW_D}(i,j) = & \, weight_{|i-j|} \cdot L_2(\mathbf{q_i}, \mathbf{c_j})^2 \\ & + min \begin{cases} \Delta_{WDTW_D}(i-1,j-1) & \textbf{if } i,j > 0 \\ \Delta_{WDTW_D}(i-1,j) \\ \Delta_{WDTW_D}(i,j-1), \end{cases} \end{aligned} \tag{13}$$

$$WDTW_D(Q,C) = \Delta_{WDTW_D}(L,L). \tag{14}$$

3.2.6 Weighted Derivative DTW (WDDTW)

The ideas behind DDTW and WDTW may be combined to implement another measure called Weighted Derivative DTW (WDDTW). This method has also been traditionally used in some ensemble algorithms (Bagnall et al, 2017).

Multivariate versions of WDDTW are also straightforward to implement. We calculate the derivatives separately for each dimension, and then use Equations 2 and 14 with them to compute independent WDDTW ($WDTW_I$) and dependent WDDTW ($WDTW_D$), respectively.

3.3 Longest Common Subsequence (LCSS)

3.3.1 Univariate LCSS

Longest Common Subsequence (LCSS) distance is based on the edit distance algorithm, which is used for string matching (Hirschberg, 1977). In TSC, the LCSS algorithm is modified to work with real-valued data by adding a threshold ϵ for real-value comparisons. Two real-values are considered a match if the difference between them is less than the threshold ϵ. A warping window can also be used in conjunction with the threshold to constrain the degree of local warping.

Formally, the unnormalised LCSS distance ($LCSS_{UN}$) between Q and C is defined as,

$$\Delta_{LCSS}(i,j) = \begin{cases} (q_i - c_j)^2 & \textbf{if } i = 1, j = 1 \\ 1 + \Delta_{LCSS}(i-1, j-1) & \textbf{if } |q_i - c_j| \leq \epsilon \\ max \begin{cases} \Delta_{LCSS}(i-1,j) \\ \Delta_{LCSS}(i,j-1) \end{cases} & \textbf{otherwise}, \end{cases} \tag{15}$$

$$LCSS_{UN}(Q,C) = \Delta_{LCSS}(L,L), \tag{16}$$

In practice, $LCSS_{UN}$ is then normalised based on the series length L.

$$LCSS(Q,C) = 1 - \frac{LCSS_{UN}(Q,C)}{L}, \tag{17}$$

LCSS can be used with a window parameter w similar to DTW. With a window parameter, LCSS has a time complexity of $O(L \cdot w)$. The parameter ϵ is selected from $[\frac{\sigma}{5}, \sigma]$, where σ being the standard deviation of the whole dataset.

3.3.2 Dependent Multivariate LCSS

Definition 10 Dependent LCSS

Dependent LCSS ($LCSS_D$) is similar to Equation 15, except that to compute distance between two multivariate points we use Equation 8 and an adjustment is made to the parameter ϵ (see Section 3.3.3). In this case, parameter ϵ is selected using the standard deviation of the whole dataset. This is similar to the way it was selected in the univariate LCSS, in EE.

$$\Delta_{LCSS_D}(i,j) = \begin{cases} L_2(\mathbf{q_i}, \mathbf{c_j})^2 & \textbf{if } i = 1, j = 1 \\ 1 + \Delta_{LCSS_D}(i-1, j-1) & \textbf{if } L_2(\mathbf{q_i}, \mathbf{c_j})^2 \leq 2 \cdot D \cdot \epsilon \\ max \begin{cases} \Delta_{LCSS_D}(i-1,j) \\ \Delta_{LCSS_D}(i,j-1) \end{cases} & \textbf{otherwise}, \end{cases} \tag{18}$$

$$LCSS_{UND}(Q,C) = \Delta_{LCSS_D}(L,L), \tag{19}$$

Similar to the univariate case, $LCSS_{UND}$ is then normalised based on the series length L.

$$LCSS_D(Q,C) = 1 - \frac{LCSS_{UND}(Q,C)}{L}. \tag{20}$$

3.3.3 Adjusting ϵ parameter in LCSS

In Equation 18 we multiply ϵ by 2 times the number of dimensions D, because the term $L_2(\mathbf{q_i}, \mathbf{c_j})^2$ increases with the number of dimensions and the parameter ϵ (floating point comparison threshold) is independent of the number of dimensions (compare with the univariate definition in Equation 15).

The adjustment factor $2 \cdot D$ can be used with a number of assumptions. Firstly, data should follow a normal distribution. Secondly, the squared Euclidean distance ($L_2(\mathbf{q_i}, \mathbf{c_j})^2$) is required contrary to the Manhattan distance (L_1).

To compensate for the increase in the magnitude of squared Euclidean distance with respect to the increasing number of dimensions, the adjustment factor should be able to normalise this value to make the Equation 18 work similarly to the Equation 15.

Consider two vectors X and Y in D dimension $X_i, Y_i \sim N(0, 1)$ where $\forall i \in [1, D]$. Assume that X and Y are independent and each dimension is independent as well.

Let another random variable $Z = X - Y$. By property of the normal distribution, we have $Z \sim N(0, 2)$ (as X and Y are independent we can add both mean and variance, hence $N(0 + 0, 1 + 1)$).

We are interested in $E[Z^2]$. This follows chi-square distribution, which is the sum of square of independent normally distributed variables.

However, the chi-square distribution is only if the variance is 1 and here we have variance of 2 for Z. Let V be another random variable of variance 1. To do so, we divide Z^2 by squared standard deviation (*i.e.* variance 2), so $V = (Z^2/2) \sim \chi^2(D)$. Then we have

$E[Z^2/2] = E[V]$ and we know $E[V] = D$ by property of the chi square distribution

$E[Z^2]/2 = D$

$E[Z^2] = 2 \cdot D$

Therefore, if data is normally distributed, $(L_2)^2$ distance between points, with any number of dimensions may be scaled by a factor of $2 \cdot D$ to make the values comparable to average magnitude of values in one dimension. For this reason we multiply the right hand side (ϵ) by this factor. Once the left hand side has been adjusted, it may be compared with ϵ similarly as in the univariate LCSS definition.

We found that this adjustment works for the datasets we tested. In practice, if the data distribution differs substantially from the normal distribution, this adjustment factor may need to be revised, hence further investigation of methods to adjust the parameter ϵ is be a potential future work.

3.4 Move-Split-Merge (MSM)

3.4.1 Univariate MSM

Move-Split-Merge (MSM) is introduced by Stefan et al (2013). The goal is to propose a similarity measure that is a metric invariant to translations and robust to temporal misalignments. Measures such as DTW and LCSS are not metrics because they fail to satisfy the triangle inequality (see Definition 4).

MSM is an edit distance-based similarity measure. Similarity between two series is computed based on the number and type of edit operations required to transform one series to the other.

MSM defines three types of edit operations: move, merge and split. The move operation substitutes one value into another value. The split operation inserts a copy of the value immediately after itself, and the merge operation is used to delete a value if it directly follows an identical value.

The cost for a move operation is the pairwise distance between two points, and the cost of split or merge operation depends on the parameter c.

Formally, MSM is defined as,

$$\Delta_{MSM}(i,j) = min \begin{cases} \Delta_{MSM}(i-1,j-1) + |q_i - c_j| \\ \Delta_{MSM}(i-1,j) + cost(q_i, q_{i-1}, c_j) \\ \Delta_{MSM}(i,j-1) + cost(c_j, q_i, c_{i-1}), \end{cases} \tag{21}$$

$$MSM(Q,C) = \Delta_{MSM}(L,L). \tag{22}$$

More precisely, the cost of either a split or a merge operation is defined by Equation 23. In the univariate case, the algorithm either merges two values or splits a value if the the value of a point q_i *is_between* two adjacent values (q_{i-1} and c_j).

$$cost(q_{i-1}, q_i, c_j) = min \begin{cases} c \textbf{ if } q_{i-1} \leq q_i \leq c_j \\ c \textbf{ if } q_{i-1} \geq q_i \geq c_j \\ c + min \begin{cases} |q_i - q_{i-1}| \\ |q_i - c_j| \end{cases} \textbf{ otherwise}. \end{cases} \tag{23}$$

In most algorithms (e.g. EE, PF, HIVE-COTE and TS-CHIEF), the cost parameter c for MSM is selected from an exponential sequence $\{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, \cdots, 1\}$ as proposed in Stefan et al (2013).

3.4.2 Dependent Multivariate MSM

Definition 11 Dependent MSM

Here we combine Equation 21 and Equation 8. The *cost_multiv* function is explained in Section 3.4.3, and presented in Algorithm 1.

$$\Delta_{MSM_D}(i,j) = min \begin{cases} \Delta_{MSM_D}(i-1,j-1) + L_2(\mathbf{q_i}, \mathbf{c_j})^2 \\ \Delta_{MSM_D}(i-1,j) + cost\_multiv\ (\mathbf{q_i}, \mathbf{q_{i-1}}, \mathbf{c_j}) \\ \Delta_{MSM_D}(i,j-1) + cost\_multiv\ (\mathbf{c_j}, \mathbf{q_i}, \mathbf{c_{j-1}}) \end{cases} \tag{24}$$

$$MSM_D(Q,C) = \Delta_{MSM_D}(L,L) \tag{25}$$

3.4.3 Cost function for dependent MSM

A nontrivial issue when deriving a dependent variant of MSM is how to translate the concept of one point being between two others.

A naive approach would test whether a point *is_between* two other points in the multidimensional space by projecting the query point onto the hyperplane defined by the other two points. However, this has serious limitations. For an intuitive example, let us use cities to represent points on a 2-D plane. Assume that we have two query cities Adelaide and Tokyo which is between Melbourne and Perth. If we use vector projections, and project the position of Tokyo on to the line between Melbourne and Perth we will find that it is between the two cities. Similarly, we will also find that Adelaide is between Melbourne and Perth using this method. However, orthogonally Tokyo is extremely far away from both Melbourne and Perth, so it would seem more ideal to define this function in a way that Adelaide is in between Melbourne and Perth, but Tokyo is not. Using this intuition we define cost function in such a way that a point is considered to be in between two points only if the point is "inside the hypersphere" defined by the other two points.

We implement this idea in Algorithm 1. First we find the diameter of the hypersphere in line 1 by computing $||\mathbf{q_{i-1}} - \mathbf{c_j}||$. In line 2 we find the mid point **mid** along the line $\mathbf{q_{i-1}}$ and $\mathbf{c_j}$. Then we calculate distance to the mid point using $||\mathbf{mid} - \mathbf{q_i}||$ (line 3). Once we have the *distance_to_mid*, we check if this distance is larger than half the diameter. If its larger, then the point $\mathbf{q_i}$ is outside the hypersphere, and so we return c (line 5). If *distance_to_mid* smaller than half the diameter, then $\mathbf{q_i}$ is inside the hypersphere, so we check to which point (either $\mathbf{q_{i-1}}$ or $\mathbf{c_j}$ it is closer to). Then we return c plus the distance to the closest point as the cost of the edit operation (line 9 to 12).

Algorithm 1: Cost by checking if mid point is inside the hypersphere defined by the other two points

Input: $cost\_multiv(\mathbf{q_i}, \mathbf{q_{i-1}}, \mathbf{c_j}, c)$: three points, and cost parameter c for MSM
Output: cost of operation

```
1  diameter = ||q_{i-1} − c_j||;
2  mid = (q_{i-1} + c_j)/2;
3  distance_to_mid = ||mid − q_i||;
4  if distance_to_mid ≤ (diameter/2) then
5  |  return c;
6  else
7  |  dist_to_q_prev = ||q_{i-1} − q_i||;
8  |  dist_to_c = ||c_j − q_i||;
9  |  if dist_to_q_prev < dist_to_c then
10 |  |  return c + dist_to_q_prev;
11 |  else
12 |  |  return c + dist_to_c ;
```

3.5 Edit Distance with Real Penalty (ERP)

3.5.1 Univariate ERP

Edit Distance with Real Penalty (ERP) (Chen and Ng, 2004; Chen et al, 2005) is also based on string matching algorithms. In a typical string matching algorithm, two strings, possibly of different lengths, may be aligned by doing the least number of add, delete or change operations on the symbols. When aligning two series of symbols, the authors proposed that the delete operations in one series can be thought of as adding a special symbol to the other series. Chen and Ng (2004) refers to these added symbols as a "gap" element.

ERP uses the Euclidean distance between elements when there is no gap, and a constant penalty when there is a gap. This penalty parameter for a gap is denoted as g (see Equation 26).

For time series, with real values, similar to the parameter ϵ in LCSS, a floating point comparison threshold may be used to determine a match between two values. This idea was used in a measure called Edit Distance on Real sequences (EDR) (Chen and Ng, 2004). However, using a threshold breaks the triangle inequality. Therefore, the same authors proposed a variant, namely ERP, which is a measure that follows the triangle inequality.

ERP can also be used with a window parameter w similar to DTW. With the window parameter, ERP has the same time complexity as DTW. The parameter g is selected from $[\frac{\sigma}{5}, \sigma]$, with σ being the standard deviation of the training data. Formally, ERP is defined as,

$$\Delta_{ERP}(i,j) = min \begin{cases} \Delta_{ERP}(i-1,j-1) + (q_i - c_j)^2 \\ \Delta_{ERP}(i-1,j) + (q_i - g)^2 \\ \Delta_{ERP}(i,j-1) + (c_j - g)^2 \end{cases} \tag{26}$$

$$ERP(Q,C) = \Delta_{ERP}(L,L). \tag{27}$$

3.5.2 Dependent ERP

Definition 12 Dependent ERP

We define Dependent ERP (ERP_D) as,

$$\Delta_{ERP_D}(i,j) = min \begin{cases} \Delta_{ERP_D}(i-1,j-1) + L_2(\mathbf{q_i}, \mathbf{c_j})^2 \\ \Delta_{ERP_D}(i-1,j) + L_2(\mathbf{q_i}, \mathbf{g}) \\ \Delta_{ERP_D}(i,j-1) + L_2(\mathbf{c_j}, \mathbf{g}), \end{cases} \tag{28}$$

$$ERP_D(Q,C) = \Delta_{ERP_D}(L,L). \tag{29}$$

In Equation 28, note that the parameter $\mathbf{g}$ is a vector which represents the standard deviation of each dimension separately. This is in contrast to the univariate case in Equation 26 which uses the standard deviation of the whole training dataset (parameter g).

In this case, all terms increases proportionally with respect to the increase in the number of dimensions. So we do not need to adjust for the parameter $\mathbf{g}$ as we adjusted for ϵ in LCSS in Section 3.3.3.

3.6 Time Warp Edit (TWE)

3.6.1 Univariate TWE

Time Warp Edit (TWE) (Marteau, 2009) is a further edit-distance based algorithm adapted to the time series domain. The goal is to combine an *Lp* distance based technique with an edit-distance based algorithm that supports warping in the time axis, *i.e.* has some sort of *elasticity* like DTW, while also being a distance metric (*i.e.* it respects the triangle inequality). Being a metric helps in time series indexing, since it speeds up time series retrieval process (see Definition 4).

TWE uses three operations named *match*, $delete_A$, and $delete_B$. If there is a *match*, *Lp* distance is used, and if not, a constant penalty λ is added. $delete_A$ (or $delete_B$) is used to remove an element from the first (or second) series to match the second (or first) series. Equations 30, 31 and 32 define TWE and these three operations, respectively.

$$\Delta_{TWE}(i,j) = min \begin{cases} \Delta_{TWE}(i-1,j-1) + \gamma_M & match \\ \Delta_{TWE}(i-1,j) + \gamma_A & delete_A \\ \Delta_{TWE}(i,j-1) + \gamma_B & delete_B \end{cases} \tag{30}$$

$$TWE(Q,C) = \Delta_{TWE}(L,L) \tag{31}$$

$$\begin{array}{ll} \gamma_M = (q_i - c_j)^2 + (q_{i-1} - c_{j-1})^2 + 2 \cdot \nu & match \\ \gamma_A = (q_i - q_{i-1})^2 + \nu + \lambda & delete_A \\ \gamma_B = (c_j - c_{j-1})^2 + \nu + \lambda & delete_B \end{array} \tag{32}$$

The multiplicative penalty ν[1] is called the *stiffness* parameter. When $\nu = 0$, TWE becomes more stiff like the Euclidean distance, and when $\nu = \infty$, TWE becomes less stiff and more elastic like DTW. The second parameter γ is the cost of performing either a $delete_A$ or $delete_B$ operation.

Following Marteau (2009), λ is selected from $\cup_{i=0}^{9} \frac{i}{9}$ and γ from the exponentially growing sequence $\{10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 10^{-3}, 5 \cdot 10^{-3}, \cdots, 1\}$, resulting in 100 possible parameterizations.

3.6.2 Dependent TWE

Definition 13 Dependent TWE

Dependent version of TWE follows a similar pattern. We define Dependent TWE (TWE_D) as,

$$\Delta_{TWE_D}(i,j) = min \begin{cases} \Delta_{TWE_D}(i-1,j-1) + \gamma_M & match \\ \Delta_{TWE_D}(i-1,j) + \gamma_A & delete_A \\ \Delta_{TWE_D}(i,j-1) + \gamma_B & delete_B \end{cases} \tag{33}$$

$$TWE_D(Q,C) = \Delta_{TWE_D}(L,L). \tag{34}$$

[1] In the published definition of TWE (Marteau, 2009), ν is multiplied with the time difference in the timestamps of two consecutive time points. We simplified this equation, for clarity, by assuming that this time difference is always 1 (UEA datasets do not store the actual timestamps).

$$\begin{aligned} \gamma_M &= L_2(\mathbf{q_i}, \mathbf{c_j})^2 + L_2(\mathbf{q_{i-1}}, \mathbf{c_{j-1}})^2 + (2 \cdot \nu) \cdot 2 \cdot D && match \\ \gamma_A &= L_2(\mathbf{q_i}, \mathbf{q_{i-1}})^2 + (\nu + \lambda) \cdot 2 \cdot D && delete_A \\ \gamma_B &= L_2(\mathbf{c_j}, \mathbf{c_{j-1}})^2 + (\nu + \lambda) \cdot 2 \cdot D && delete_B \end{aligned} \tag{35}$$

Similar to LCSS, in Dependent TWE, we need to make an adjustment to the parameters. Once again, we multiply the terms that do not grow with $2 \cdot D$.

4 Experiments

We conduct experiments to investigate three hypotheses.

The first hypothesis is that there will be datasets for which each of the new multivariate distance measures is best suited.

The next hypothesis arises from the observation that there are some datasets for which either the independent or dependent version of DTW are consistently more accurate than the alternative (Shokoohi-Yekta et al, 2017). However, it is not clear whether this is a result of there being an advantage in treating all dimensions in lock step or the reverse, or due to some other property of the algorithms.

It is credible that there should be some time series data for which it is beneficial to treat multiple variables in lock step. Suppose, for example, that the variables each represent the throughput of independent parts of a process and the quantity relevant to classification is aggregate throughput. In this case, the sum of the values at each point is the relevant quantity. In contrast, if classification relates to a failure in any of those parts, it seems clear that independent consideration of each is the better approach.

We seek to assess whether there are multivariate datasets for which each of lock-step and independent analysis are best suited, or whether there are other reasons, such as their mathematical properties, that underlie the systematic advantage on specific datasets of either DTW_I or DTW_D.

The third hypothesis, inspired by EE (Lines and Bagnall, 2015), is that an ensemble of nearest neighbour classifiers, each using a different multivariate distance measure, will be more accurate than any single nearest neighbour classifier using a single distance measure.

We start by describing our experimental setup and the datasets we used. We then conduct an analysis of similarity measures in the context of TSC by comparing accuracy measures of independent and dependent versions. We then conduct a statistical test to determine if there is a difference between independent and dependent versions of the measures. After that we present three methods of ensembling the measures and then compare the relative accuracy of the ensembles.

4.1 Experimental Setup

We implemented a multi-threaded version of the multivariate similarity measures in Java. We also release the full source code in the github repository `https://github.com/dotnet54/multivariate-measures`.

In these experiments, for parametrization of the measures, we use leave-one-out cross-validation of 100 parameters for each similarity measure. We follow the same setting proposed in (Lines and Bagnall, 2015). This parametrization is also used in HIVE-COTE, PF and TS-CHIEF.

In this study, we use multivariate datasets obtained from https://www.timeseriesclassification.com. We also use the standard train/test splits provided in the repository. Out of the available 30 datasets, we use 23 datasets in this study. Since we focus only on fixed-length datasets, the four variable length datasets ("CharacterTrajectories", "InsectWingbeat", "JapaneseVowels", and "SpokenArabicDigits") are excluded from this study. We also we omit "EigenWorms","MotorImagery" and,"FaceDetection", which was taking too long to run the leave-one-out cross-validation for 100 parameters in a practical time frame. Table 2 (on page 26) summarises the characteristics of the 26 fixed length datasets. Further descriptions of each dataset can be found in Bagnall et al (2018).

We ran experiments for all measures for z-normalised and unnormalised datasets. We z-normalised each dataset on a per series, per dimension basis. We found that the accuracy on these datasets are better when used as provided in the archive without normalisation (Note that 4 datasets on the archive are already normalised). This is also in agreement with a recent paper which conducted a similar experiment with normalisation using DTW_I and DTW_D (Ruiz et al, 2020).

However, we note that this does not indicate that not normalising is always the optimal solution for all datasets. Sometimes normalisation can be useful when using similarity measures. For example, consider a scenario with two dimensions temperature (e.g a scale from 0 to 100 degree Celsius) and relative humidity as a percentage (between 0 and 1). In such a case, temperature will be dominating the result of the similarity calculation, and normalisation will help to compute the similarity with similar scales across the dimensions.

We ran the experiments on a cluster of Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50 GHz CPUs, with each experiment run on 32-threads. Total time to train all 23 datasets with leave-one-out-cross-validation was about 1648 hours. The two slowest datasets were "PEMS-SF" (291 hours) and "PhonemeSpectra" (1153 hours) The slowest measure to train is MSM_D, which took a total of took 801 hours across all datasets.

4.2 Accuracy of Independent *Vs* Dependent Measures

First we look at the accuracy of each measure used with a 1-NN classifier. Tables 3 and 4 (on page 27 and 28) present the accuracy for independent measures and dependent measures, respectively. For each dataset, the highest accuracy is typeset in bold. Of the values reported in Tables 3 and 4, accuracy for measures other than than Euclidean distance (labelled "L_2" in the table) and DTW are newly published results in this paper.

Our first observation is that for every similarity measure there is at least one dataset for which that measure obtains the highest accuracy. This supports our first hypothesis, that each measure will have datasets for which it is well suited.

To compare multiple algorithms over the multiple datasets, first a Friedman test is performed to reject the null hypothesis. The null hypothesis is that there is no significant difference in the mean ranks of the multiple algorithms (at a statistical significance level $\alpha = 0.05$.) In cases where the null-hypothesis is rejected, we use the Wilcoxon signed rank test to compare the pair-wise difference in ranks between algorithms, and then use Holm–Bonferroni's method to adjust for family-wise errors (Demšar, 2006; Benavoli et al, 2016).

Figure 2 displays mean ranks (on error) between the 20 similarity measures. Measures on the right side indicate higher rank in accuracy (lower error). We do not include $L2$ distance here because the accuracy for independent and dependent $L2$ is the same. Since we use Holm–Bonferroni's correction, there is not a single "critical difference value" that applies to all pairwise comparisons. Hence, we refer to these visualisations as "average accuracy ranking diagrams".

In Figure 2, DTW_I, which is to the further right, is the most accurate measure on the evaluated datasets. DTW_I obtained a ranking of 7.326 from the Wilcoxon test. By contrast, $DDTWF_D$ (ranked 14.783) is the least accurate measure on these datasets. After Holm–Bonferroni's correction, computed p-values indicate that only the pairs $DDTWF_D$ and DTW_I, and $DDTWF_D$ and $WDTW_D$, are statistically different from each other. This could be because there is a large number of measures, which win or lose on similar datasets.

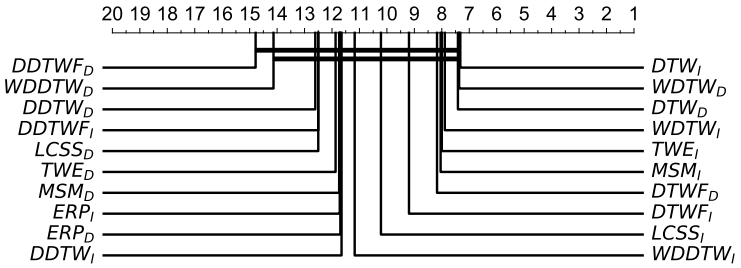

Fig. 2 Average accuracy ranking diagram showing the ranks of measures on the error rates (thus more accurate measures are to the right side).

4.3 Are Independent and Dependent Measures Significantly Different or Similar?

In this section, we test if there are datasets for which independent or dependent version is always more accurate. We also test if there is a statistically significant difference between independent and dependent similarity measures. Answering these questions will help us determine the usefulness of developing these two variations of the multivariate similarity measures. It will also help us to construct ensembles of similarity measures with more diversity, that is expected to perform well in terms of accuracy over a wide variety of datasets.

Figure 3 shows the difference in accuracy between independent and dependent versions of the measures - deeper reddish colours indicate cases where independent is more accurate (positive on the scale), and deeper bluish colours indicate cases where dependent is more accurate (negative on the scale). The datasets are sorted based on average colour values to show contrasting colours on the two ends (dimensions D / length L / number of classes c are given in the bracket after the dataset name).

From Figure 3 we observe that there are datasets for which either independent or dependent is always more accurate. For example, the independent versions of all measures are consistently more accurate for datasets "PEMS-SF" and Basic Motions (indicated by red colour rows in the heatmap). On the other hand, we see that "Handwriting" always wins for the dependent versions (indicated by the blue colour row).

Fig. 3 Heatmap showing the difference in accuracy between independent and dependent versions of the measures - deeper reddish colours indicate cases where independent is more accurate (positive on the scale), and deeper bluish colours indicate cases where dependent is more accurate (negative on the scale).

When we observe the columns, we see that there is no measure where either independent or dependent versions always win across all the datasets.

Next we present the results of a Wilcoxon signed-rank test on each dataset across the 10 pairs of measures (without L_2), to test if the difference between accuracy of independent and dependent versions are statistically significant or not. We conduct this test with the null hypothesis that the mean of the difference between accuracy of independent and dependent versions will be zero with statistical significance value ($\alpha = 0.05$). We reject the null hypothesis, and accept that there is a significant statistical difference in accuracy, if the obtained significance value ($p - value$) obtained after the Wilcoxon signed-rank test is less than 0.05. Table 1 shows the p-value for each dataset and an asterisk marks the p-values obtained datasets for which there is a significant difference (p-value less than 0.05). We also report the p-values after Holm–Bonferroni correction. Before Holm–Bonferroni correction, out of the 23 datasets, we observe that for 7 datasets there is a statistically significant difference in accuracy between independent and dependent measures. After Holm–Bonferroni correction, we conclude there is indeed a statistical difference between independent and dependent measures.

4.4 Ensemble of Measures

We now compare the accuracy of these measures within three ensembles. Within each ensemble, final predicted label is calculated using the maximum class probability, after each class probability have been weighted by the leave-one-out cross validation accuracy of each measure. Any ties are broken using a uniform random choice.

The three ensembles are constructed as follows:

Table 1 p-values for two-sided Signed Rank Wilcoxon test with $\alpha = 0.05$ (significant values are in bold face).

dataset	p-value	p-value-after-Holm-Bonferroni
BasicMotions	**0.0020**	**0.0021**
PEMS-SF	**0.0020**	**0.0022**
Handwriting	**0.0020**	**0.0023**
HandMovementDirection	**0.0059**	0.0024
LSST	**0.0059**	0.0025
Epilepsy	**0.0103**	0.0026
ArticularyWordRecognition	**0.0282**	0.0028
PhonemeSpectra	0.1794	0.0029
PenDigits	0.1934	0.0031
StandWalkJump	0.1988	0.0033
DuckDuckGeese	0.2324	0.0036
AtrialFibrillation	0.2820	0.0038
FingerMovements	0.3071	0.0042
Heartbeat	0.4316	0.0045
Cricket	0.5043	0.0050
SelfRegulationSCP2	0.5566	0.0056
SelfRegulationSCP1	0.5748	0.0063
RacketSports	0.6094	0.0071
ERing	0.6101	0.0083
NATOPS	0.7695	0.0100
UWaveGestureLibrary	0.7695	0.0125
Libras	0.8457	0.0167
EthanolConcentration	1.0000	0.0250

- EE_I : Ensemble of 1-NN formed using 11 independent measures.
- EE_D : Ensemble of 1-NN formed using 11 dependent measures.
- EE_{ID} : Ensemble of 1-NN formed using 11 independent and 11 dependent measures.

Figure 4 shows accuracy ranking of our three ensembles *vs* top seven similarity measures with 1-NN. We can observe that all ensembles are more accurate than the classifiers using a single measure. We found that the EE_{ID} (avg. rank 3.087) performs better than both EE_I (avg. rank 3.848) and EE_D (avg. rank 5.152). Based on the p-values, we found that all pairs of classifiers are statistically indistinguishable except the following pairs: EE_{ID} and $DTWF_D$, EE_{ID} and EE_D, and EE_{ID} and $WDTW_D$.

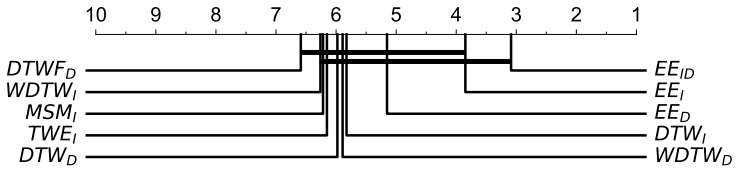

Fig. 4 Average accuracy ranking diagram showing the ranks on the error rate of top 10 measures, and our three ensembles.

Tables 5 and 6 (on page 29 and 30) show the accuracy of these ensembles as well as other multivariate algorithms obtained from the recent paper (Ruiz et al, 2020). These algorithms were briefly introduced in Section 2.3.

From Table 5 we observe that ROCKET has the most number of wins (7) followed by InceptionTime (6 wins). Next is, MUSE and ResNet with 5 wins, and EE_I and CIF with 4 wins. We also observe that EE_D and EE_ID win only on one and two datasets, respectively.

Figure 5 shows accuracy ranking of our ensembles *vs* top 10 other multivariate algorithms. The most accurate algorithm, ROCKET, obtained a rank of 5.087. Although Table 5 reports 2 wins for EE_{ID}, compared to HIVE-COTE 1.0 (4 wins) and CIF (2 wins), Figure 5 shows that average error rate of EE_{ID} (ranked 6.935) is higher than CIF (ranked 7.130) and just below MUSE (ranked 6.450). The computed p-values indicate that EE_{ID} is statistically indistinguishable from any of the other algorithms.

This comparison does not compare algorithms of similar complexity. For example, HIVE-COTE 1.0 is a far more complex algorithm than our ensembles. However from this comparison we get some idea about the standing of our similarity-based ensembles with respect to other dictionary-based, interval-based, and shapelet-based ensembles for multivariate data.

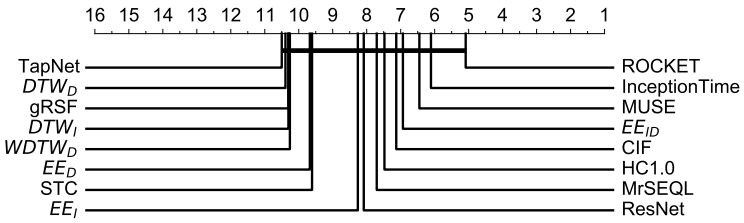

Fig. 5 Average accuracy ranking diagram showing the ranks on the error rate of the top 10 classifiers from (Ruiz et al, 2020), and our three ensembles.

5 Conclusion

In this paper, we present multivariate versions of commonly used elastic similarity measures. Our approach is inspired by independent and dependent DTW measures, which have proven very successful as strategies for extending univariate DTW to the multivariate case.

Our experiments show that each of the univariate similarity measures excel at nearest neighbour classification on different datasets, highlighting the importance of having a range of such measures in our analytic toolkits.

It has been shown that there are some datasets for which the independent version of DTW is more accurate than the dependent version and vice versa. Until now there was no way to determine whether this is a result of a fundamental difference between treating each dimension independently or not, or whether it arises from other properties of the algorithms. Our results showing that there are some datasets for which dependent or independent treatments are consistently superior across all distance measures provides strong support for the conclusion

that it is a fundamental property of the datasets, that either the variables do need to be considered in lock step or do not.

Inspired by the Elastic Ensemble of nearest neighbour classifiers using different univariate distance measures, we then further experiment with ensembles of multivariate similarity measures and show that ensembling results in accuracy competitive with the state of the art.

Our three ensembles establish a baseline in our future plans to create a multivariate TS-CHIEF which would combine similarity-based techniques with dictionary-based, interval-based for multivariate TSC.

References

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery 31(3):606–660

Bagnall A, Dau HA, Lines J, Flynn M, Large J, Bostrom A, Southam P, Keogh E (2018) The UEA multivariate time series classification archive, 2018. arXiv preprint arXiv:181100075

Bagnall A, Flynn M, Large J, Lines J, Middlehurst M (2020) On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (HIVE-COTE v1. 0). In: International Workshop on Advanced Analytics and Learning on Temporal Data, Springer, pp 3–18

Benavoli A, Corani G, Mangili F (2016) Should we really use post-hoc tests based on mean-ranks? The Journal of Machine Learning Research 17(1):152–161

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, Seattle, WA, USA:, vol 10, pp 359–370

Chen L, Ng R (2004) On The Marriage of Lp-norms and Edit Distance. In: Proceedings of the 13th Int. Conf. on Very Large Data Bases (VLDB), pp 792–803

Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. Proceedings of the 2005 ACM SIGMOD international conference on Management of data - SIGMOD '05 pp 491–502

Dempster A, Petitjean F, Webb GI (2020) ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery 34(5):1454–1495

Demšar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7:1–30

Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Information Sciences 239:142–153

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

Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA (2019) Deep learning for time series classification: a review. Data Mining and Knowledge Discovery 33(4):917–963

Fawaz HI, Lucas B, Forestier G, Pelletier C, Schmidt DF, Weber J, Webb GI, Idoumghar L, Muller PA, Petitjean F (2020) InceptionTime: Finding AlexNet for time series classification. Data Mining and Knowledge Discovery 34(6):1936–1962

Gallicchio C, Micheli A (2017) Deep Echo State Network (DeepESN): a brief survey. arXiv preprint arXiv:171204323

Herrmann M, Webb GI (2020) Early abandoning PrunedDTW and its application to similarity search. arXiv preprint arXiv:201005371

Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time series by shapelet transformation. Data Mining and Knowledge Discovery 28(4):851–881

Hirschberg DS (1977) Algorithms for the longest common subsequence problem. Journal of the ACM 24(4):664–675

Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 23(1):67–72

Izakian H, Pedrycz W (2014) Anomaly detection and characterization in spatial time series data: A cluster-centric approach. IEEE Transactions on Fuzzy Systems 22(6):1612–1624

Jeong YS, Jeong MK, Omitaomu OA (2011) Weighted Dynamic Time Warping for time series classification. Pattern Recognition 44(9):2231–2240

Karlsson I, Papapetrou P, Boström H (2016) Generalized Random Shapelet Forests. Data Mining and Knowledge Discovery 30(5):1053–1085

Keogh E, Wei L, Xi X, Vlachos M, Lee SH, Protopapas P (2009) Supporting exact indexing of arbitrarily rotated shapes and periodic time series under Euclidean and warping distance measures. VLDB Journal 18(3):611–630

Keogh EJ, Pazzani MJ (2001) Derivative Dynamic Time Warping. Proceedings of the 2001 SIAM Int Conf on Data Mining pp 1–11

Le Nguyen T, Gsponer S, Ilie I, O'Reilly M, Ifrim G (2019) Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations. Data Mining and Knowledge Discovery 33(4):1183–1222

Lemire D (2009) Faster retrieval with a two-pass dynamic-time-warping lower bound. Pattern Recognition 42(9):2169–2180

Lin J, Khade R, Li Y (2012) Rotation-invariant similarity in time series using bag-of-patterns representation. Journal of Intelligent Information Systems 39(2):287–315

Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Mining and Knowledge Discovery 29(3):565–592

Lines J, Taylor S, Bagnall A (2018) Time series classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation-Based Ensembles. ACM Transactions on Knowledge Discovery from Data (TKDD) 12(5):52

Lucas B, Shifaz A, Pelletier C, O'Neill L, Zaidi N, Goethals B, Petitjean F, Webb GI (2019) Proximity Forest: an effective and scalable distance-based classifier for time series. Data Mining and Knowledge Discovery 33(3):607–635

Marteau PF (2009) Time Warp Edit Distance with Stiffness Adjustment for Time Series Matching. IEEE Trans on Pattern Analysis and Machine Intelligence 31(2):306–318

Middlehurst M, Vickers W, Bagnall A (2019) Scalable dictionary classifiers for time series classification. In: International Conference on Intelligent Data Engineering and Automated Learning, Springer, pp 11–19

Middlehurst M, Large J, Bagnall A (2020) The canonical interval forest (CIF) classifier for time series classification. arXiv preprint arXiv:200809172

Ruiz AP, Flynn M, Large J, Middlehurst M, Bagnall A (2020) The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery pp 1–49

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

Schäfer P (2015) The BOSS is concerned with time series classification in the presence of noise. Data Mining and Knowledge Discovery 29(6):1505–1530

Schäfer P, Leser U (2017a) Fast and accurate time series classification with WEASEL. In: Proceedings of the 2017 ACM on Conf. on Information and Knowledge Management (CIKM), pp 637–646

Schäfer P, Leser U (2017b) Multivariate time series classification with WEASEL+MUSE. arXiv preprint arXiv:171111343

Shifaz A, Pelletier C, Petitjean F, Webb GI (2020) TS-CHIEF: a scalable and accurate forest algorithm for time series classification. Data Mining and Knowledge Discovery 34(3):742–775

Shokoohi-Yekta M, Hu B, Jin H, Wang J, Keogh E (2017) Generalizing DTW to the multi-dimensional case requires an adaptive approach. Data Mining and Knowledge Discovery 31(1):1–31

Stefan A, Athitsos V, Das G (2013) The move-split-merge metric for time series. IEEE Trans on Knowledge and Data Engineering 25(6):1425–1438

Tan CW, Webb GI, Petitjean Fço (2017) Indexing and classifying gigabytes of time series under time warping. In: Proceedings of the 2017 SIAM Int. Conf. on Data Mining, SIAM, pp 282–290

Tan CW, Herrmann M, Forestier G, Webb GI, Petitjean F (2018) Efficient search of the best warping window for dynamic time warping. In: Proceedings of the 2018 SIAM International Conference on Data Mining, SIAM, pp 225–233

Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN), IEEE, pp 1578–1585

Zhang X, Gao Y, Lin J, Lu CT (2020) Tapnet: Multivariate time series classification with attentional prototypical network. In: AAAI, pp 6845–6852

A Summary of the Datasets

Table 2 Summary of the 26 fixed-length multivariate datasets from UAE repository.

#	dataset	code	trainsize	testsize	dims	length	classes
1	ArticularyWordRecognition	AWR	275	300	9	144	25
2	AtrialFibrillation	AF	15	15	2	640	3
3	BasicMotions	BM	40	40	6	100	4
4	Cricket	CR	108	72	6	1197	12
5	DuckDuckGeese	DDG	50	50	1345	270	5
6	EigenWorms	EW	128	131	6	17984	5
7	Epilepsy	EP	137	138	3	206	4
8	EthanolConcentration	EC	261	263	3	1751	4
9	ERing	ER	30	270	4	65	6
10	FaceDetection	FD	5890	3524	144	62	2
11	FingerMovements	FM	316	100	28	50	2
12	HandMovementDirection	HMD	160	74	10	400	4
13	Handwriting	HW	150	850	3	152	26
14	Heartbeat	HB	204	205	61	405	2
15	Libras	LIB	180	180	2	45	15
16	LSST	LSST	2459	2466	6	36	14
17	MotorImagery	MI	278	100	64	3000	2
18	NATOPS	NATO	180	180	24	51	6
19	PenDigits	PD	7494	3498	2	8	10
20	PEMS-SF	PEMS	267	173	963	144	7
21	Phoneme	PS	3315	3353	11	217	39
22	RacketSports	RS	151	152	6	30	4
23	SelfRegulationSCP1	SRS1	268	293	6	896	2
24	SelfRegulationSCP2	SRS2	200	180	7	1152	2
25	StandWalkJump	SWJ	12	15	4	2500	3
26	UWaveGestureLibrary	UW	120	320	3	315	8

B Accuracy of Dependent and Independent Measures

Table 3 Accuracy of independent similarity measures. Note that dtwf and ddtwf refers to measures that use full window.

dataset	L_2	dtwf	dtw	ddtwf	ddtw	wdtw	wddtw	lcss	msm	erp	twe
AWR	0.97	0.98	0.98	0.60	0.69	**0.99**	0.70	**0.99**	**0.99**	**0.99**	0.98
AF	0.27	0.27	0.27	0.07	0.07	0.13	0.27	0.20	0.20	0.27	**0.33**
BM	0.60	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.85	**1.00**	0.85	**1.00**
CR	0.92	0.99	**1.00**	0.96	0.96	0.99	0.96	0.97	0.99	0.96	0.99
DDG	0.34	0.48	0.48	0.54	0.54	0.48	0.54	0.34	**0.60**	0.34	**0.60**
EP	0.67	0.98	0.95	0.96	0.96	0.96	0.96	**0.99**	**0.99**	0.90	0.98
ER	**0.95**	0.92	0.92	0.81	0.91	0.93	0.84	**0.95**	0.89	0.94	0.91
EC	0.32	0.30	0.31	0.29	0.27	0.29	0.27	0.27	**0.33**	**0.33**	0.30
FM	0.56	0.52	0.54	**0.62**	0.54	0.51	0.59	0.52	0.51	0.51	0.52
HMD	0.28	0.30	0.23	0.34	0.34	0.26	0.34	0.26	0.32	0.23	**0.42**
HW	0.34	**0.51**	0.48	0.31	0.34	**0.51**	0.33	0.46	0.49	0.41	0.37
HB	0.66	0.66	0.69	0.69	0.69	0.69	0.69	**0.75**	**0.75**	0.65	0.72
LIB	0.82	0.89	0.89	0.90	0.90	0.89	**0.92**	0.84	0.86	0.82	0.89
LSST	0.45	**0.58**	**0.58**	0.49	0.48	**0.58**	0.49	0.34	0.55	0.46	0.53
NATO	0.82	0.85	**0.87**	0.82	0.83	0.86	0.82	0.81	0.83	0.82	0.83
PEMS	0.77	0.73	0.77	0.62	0.61	0.76	0.64	**0.83**	0.77	0.73	0.79
PD	**0.98**	0.94	**0.98**	0.95	0.97	0.96	0.96	0.96	0.96	0.97	0.94
PS	0.10	0.15	0.15	0.16	0.17	0.16	0.17	0.15	**0.18**	0.10	0.17
RS	0.80	0.84	0.84	0.77	0.76	0.86	0.78	**0.89**	0.82	0.83	0.79
SRS1	**0.78**	0.76	**0.78**	0.63	0.58	**0.78**	0.56	0.75	0.77	**0.78**	**0.78**
SRS2	0.48	0.53	0.54	0.49	0.51	0.53	0.56	0.51	0.51	0.50	**0.57**
SWJ	0.27	0.33	0.33	0.40	0.33	**0.53**	0.20	0.27	0.27	0.33	0.20
UW	0.88	0.87	**0.91**	0.72	0.84	0.88	0.80	**0.91**	0.88	0.90	0.88
Wins	3	3	7	2	1	6	2	7	7	3	6

Table 4 Accuracy of dependent similarity measures. Note that dtwf and ddtwf refers to measures that use full window.

dataset	L_2	dtwf	dtw	ddtwf	ddtw	wdtw	wddtw	lcss	msm	erp	twe
AWR	0.97	**0.99**	0.98	0.35	0.34	**0.99**	0.36	0.98	0.98	0.98	0.97
AF	0.27	0.20	0.27	0.13	**0.33**	0.20	0.27	**0.33**	0.27	0.27	0.13
BM	0.60	**0.97**	**0.97**	0.95	0.95	**0.97**	0.95	0.80	0.68	0.75	0.95
CR	0.94	**1.00**	**1.00**	0.75	0.78	**1.00**	0.75	0.99	**1.00**	0.97	0.97
DDG	0.50	**0.58**	**0.58**	0.32	0.32	**0.58**	0.32	0.42	0.26	0.36	0.26
EP	0.63	**0.96**	**0.96**	0.93	0.93	**0.96**	0.93	0.92	0.94	0.87	0.94
ER	0.94	0.91	0.94	0.79	0.84	0.93	0.83	0.93	0.90	0.94	**0.95**
EC	0.32	0.32	0.32	0.24	0.27	0.30	0.25	0.32	**0.35**	0.31	0.25
FM	0.55	0.53	0.53	0.51	0.51	0.54	0.51	0.50	0.51	**0.56**	0.50
HMD	0.26	0.19	0.24	0.27	0.24	0.23	0.27	0.23	0.16	0.22	**0.35**
HW	0.33	**0.61**	**0.61**	0.42	0.42	**0.61**	0.41	0.54	0.57	0.47	0.45
HB	0.62	0.72	0.70	0.71	0.71	0.68	0.71	0.62	0.68	0.74	**0.75**
LIB	0.83	0.87	0.87	**0.98**	**0.98**	0.88	**0.98**	0.33	0.85	0.83	0.86
LSST	0.45	**0.55**	**0.55**	0.43	0.43	**0.55**	0.43	0.36	0.36	0.45	0.45
NATO	0.84	**0.88**	**0.88**	0.85	0.85	**0.88**	0.79	0.72	0.81	0.84	0.82
PEMS	**0.73**	0.71	**0.73**	0.57	0.57	**0.73**	0.57	0.12	0.71	0.72	0.71
PD	**0.98**	**0.98**	**0.98**	0.97	0.97	**0.98**	0.97	0.95	0.94	**0.98**	**0.98**
PS	0.10	0.15	0.15	0.16	**0.17**	0.16	**0.17**	0.15	0.16	0.10	**0.17**
RS	0.82	0.80	0.82	0.78	0.80	0.85	0.78	**0.89**	**0.89**	0.74	0.78
SRS1	0.78	0.77	0.78	0.58	0.57	0.76	0.57	**0.80**	0.78	0.78	0.42
SRS2	0.48	**0.54**	**0.54**	0.48	0.52	**0.54**	0.49	0.47	0.52	0.49	**0.54**
SWJ	0.20	0.20	0.20	0.20	**0.33**	**0.33**	0.20	**0.33**	**0.33**	0.27	0.27
UW	0.88	**0.90**	**0.90**	0.79	0.80	**0.90**	0.83	0.89	0.88	0.89	0.88
Wins	2	11	11	1	4	13	2	4	4	2	6

C Accuracy of Ensembles *vs* Other multivariate TSC algorithms

Table 5 Accuracy of ensembles of similarity measures compared with existing algorithms. Column names are shortened as follows: RT for ROCKET, IT for InceptionTime, MUSE for WEASEL+MUSE and HC1.0 for HIVE-COTE 1.0. Note that wins are counted across all methods in both Table 5 and 6 (e.g. for the first row 'AWR', ROCKET with 1.0 accuracy in Table 5 wins, therefore Table 6 does not show a win)

dataset	EE_I	EE_D	EE_{ID}	DTW_I	DTW_D	RT	IT	MUSE	CIF	HC1.0
AWR	0.99	0.99	0.99	0.98	0.98	**1.00**	0.99	0.99	0.98	0.98
AF	0.20	0.27	0.27	0.27	0.27	0.25	0.22	**0.74**	0.25	0.29
BM	**1.00**	0.95	0.97	**1.00**	0.97	0.99	**1.00**	**1.00**	**1.00**	**1.00**
CR	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**	0.98	0.99
DDG	0.54	0.58	0.60	0.48	0.58	0.46	**0.63**	N/A	0.56	0.48
EP	0.98	0.96	0.98	0.95	0.96	0.99	0.99	**1.00**	0.98	**1.00**
ER	0.96	0.94	0.97	0.92	0.94	**0.98**	0.92	0.97	0.96	0.94
EC	0.33	0.33	0.35	0.31	0.32	0.45	0.28	0.49	0.73	0.81
FM	**0.59**	0.55	0.57	0.54	0.53	0.55	0.56	0.55	0.54	0.54
HMD	0.32	0.26	0.28	0.23	0.24	0.45	0.42	0.38	**0.52**	0.38
HW	0.51	0.60	0.58	0.48	0.61	0.57	**0.66**	0.52	0.35	0.50
HB	0.72	0.73	0.75	0.69	0.70	0.72	0.73	0.74	**0.77**	0.72
LIB	0.91	0.89	0.91	0.89	0.87	0.91	0.89	0.90	0.92	0.90
LSST	0.60	0.56	0.60	0.58	0.55	0.63	0.34	**0.64**	0.56	0.54
NATO	0.88	0.88	0.88	0.87	0.88	0.89	**0.97**	0.87	0.84	0.83
PEMS	0.77	0.73	0.73	0.77	0.73	0.86	0.83	N/A	**1.00**	0.98
PD	0.97	0.98	0.98	0.98	0.98	**1.00**	**1.00**	0.99	0.99	0.97
PS	0.19	0.19	0.20	0.15	0.15	0.28	**0.37**	N/A	0.27	0.33
RS	0.85	0.84	0.87	0.84	0.82	**0.93**	0.92	0.90	0.89	0.91
SRS1	0.79	0.78	0.77	0.78	0.78	0.87	0.85	0.74	0.86	0.86
SRS2	**0.54**	0.49	**0.54**	**0.54**	**0.54**	0.51	0.52	0.50	0.49	0.52
SWJ	0.33	0.20	0.33	0.33	0.20	**0.46**	0.42	0.35	0.45	0.41
UW	0.91	0.91	0.92	0.91	0.90	**0.94**	0.91	0.90	0.92	0.91
Wins	4	1	2	3	2	7	6	5	4	2

Table 6 Accuracy of ensembles of similarity measures compared with existing algorithms. Column names are shortened as follows: MRS for MrSEQL, RN for ResNet, TN for TapNet, CB for CBOSS. Note that wins are counted across all methods in both Table 5 and 6 (e.g. for the first row 'AWR', ROCKET with 1.0 accuracy in Table 5 wins, therefore Table 6 does not show a win)

dataset	EE_I	EE_D	EE_{ID}	MRS	RN	STC	TN	gRSF	CB	RISE
AWR	0.99	0.99	0.99	0.99	0.98	0.98	0.97	0.98	0.98	0.96
AF	0.20	0.27	0.27	0.37	0.36	0.32	0.30	0.28	0.30	0.24
BM	**1.00**	0.95	0.97	0.95	**1.00**	0.98	0.99	**1.00**	0.99	**1.00**
CR	**1.00**	**1.00**	**1.00**	0.99	0.99	0.99	0.97	0.97	0.98	0.98
DDG	0.54	0.58	0.60	0.39	**0.63**	0.43	0.58	0.44	0.43	0.51
EP	0.98	0.96	0.98	**1.00**	0.99	0.99	0.96	0.96	**1.00**	**1.00**
ER	0.96	0.94	0.97	0.93	0.87	0.84	0.89	0.92	0.84	0.82
EC	0.33	0.33	0.35	0.60	0.29	**0.82**	0.29	0.34	0.40	0.49
FM	**0.59**	0.55	0.57	0.56	0.55	0.53	0.51	0.54	0.51	0.52
HMD	0.32	0.26	0.28	0.35	0.35	0.35	0.32	0.32	0.29	0.28
HW	0.51	0.60	0.58	0.54	0.60	0.29	0.33	0.37	0.49	0.18
HB	0.72	0.73	0.75	0.73	0.64	0.72	0.74	0.75	0.72	0.73
LIB	0.91	0.89	0.91	0.87	**0.94**	0.84	0.84	0.76	0.85	0.82
LSST	0.60	0.56	0.60	0.60	0.43	0.58	0.46	0.58	0.44	0.51
NATO	0.88	0.88	0.88	0.86	**0.97**	0.84	0.90	0.82	0.82	0.81
PEMS	0.77	0.73	0.73	0.97	0.82	0.98	0.79	0.91	0.97	0.99
PD	0.97	0.98	0.98	0.97	**1.00**	0.98	0.94	0.96	0.96	0.87
PS	0.19	0.19	0.20	N/A	0.31	0.31	N/A	0.23	0.19	0.27
RS	0.85	0.84	0.87	0.89	0.91	0.88	0.86	0.88	0.89	0.84
SRS1	0.79	0.78	0.77	0.83	0.76	0.85	**0.96**	0.80	0.81	0.73
SRS2	**0.54**	0.49	**0.54**	0.50	0.50	0.52	0.53	0.49	0.50	0.50
SWJ	0.33	0.20	0.33	0.42	0.31	0.44	0.35	0.38	0.37	0.34
UW	0.91	0.91	0.92	0.91	0.88	0.87	0.88	0.90	0.86	0.71
Wins	4	1	2	1	5	1	1	1	1	2