Noname manuscript No.
(will be inserted by the editor)

FastEE: Fast Ensembles of Elastic Distances for Time Series Classification

Chang Wei Tan · François Petitjean · Geoffrey I. Webb

the date of receipt and acceptance should be inserted later

Abstract In recent years, many new ensemble-based time series classification (TSC) algorithms have been proposed. Each of them is significantly more accurate than their predecessors. The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) is currently the most accurate TSC algorithm when assessed on the UCR repository. It is a meta-ensemble of 5 state-of-the-art ensemble-based classifiers. The time complexity of HIVE-COTE – particularly for training – is prohibitive for most datasets. There is thus a critical need to speed up the classifiers that compose HIVE-COTE. This paper focuses on speeding up one of its components: *Ensembles of Elastic Distances* (EE), which is the classifier that leverages on the decades of research into the development of time-dedicated measures. Training EE can be prohibitive for many datasets. For example, it takes a month on the `ElectricDevices` dataset with 9,000 instances. This is because EE needs to cross-validate the hyper-parameters used for the 11 similarity measures it encompasses. In this work, *Fast Ensembles of Elastic Distances* is proposed to train EE faster. There are two versions to it. The exact version makes it possible to train EE 10 times faster. The approximate version is 40 times faster than EE without significantly impacting the classification accuracy. This translates to being able to train EE on `ElectricDevices` in 13 hours.

Keywords Time series classification, scalable, similarity measures, ensembles

Acknowledgements This research was supported by the Australian Research Council under grant DP190100017. François Petitjean is the recipient of an Australian Research Council Discovery Early Career Award (project number DE170100037) funded by the Australian Government. This material is based upon work supported by the Air Force Office of

Chang Wei Tan · François Petitjean · Geoffrey I. Webb
Faculty of Information Technology
25 Exhibition Walk
Monash University, Melbourne
VIC 3800, Australia
E-mail: chang.tan@monash.edu,francois.petitjean@monash.edu,geoff.webb@monash.edu

Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-18-1-4030. The authors would like to acknowledge the use of the UCR Time Series Classification archive that is made publicly available for time series classification benchmarks. We also would like to acknowledge the use of the source code for Ensemble of Elastic Distances that is freely available at `http://www.timeseriesclassification.com/`.

1 Introduction

Time series data are growing at an unprecedented rate. One of the largest publicly available training datasets currently holds 1,000,000 satellite image time series instances (Tan et al., 2017). These data describe the evolution of an area on Earth and are used to create land-cover maps. Data of this scale are just the tip of the iceberg. Typically, the creation of land-cover maps require at least 100 million time series (Inglada et al., 2015, 2016). The computational demands of learning from these huge amounts of data challenges state-of-the-art techniques (Tan et al., 2017, 2018). For instance, searching through long ECG queries with 0.3 trillion data-points using the Nearest Neighbour classifier with Euclidean distance (NN-ED) without any optimisation took a week (Rakthanmanon et al., 2012). Note that Euclidean distance is the fastest to compute out of the standard similarity measures, as it is linear with the length of the time series. At a smaller scale, processing 45 minutes of speech data using the Dynamic Time Warping (DTW) distance took 3 hours on a single core machine (Srikanthan et al., 2011).

Time series classification (TSC) is an important tool for these applications. Currently the most accurate TSC algorithm is the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) (Lines et al., 2016). HIVE-COTE is an ensemble of five groups of classifiers that uses a hierarchical voting scheme to weight the predictions from each group of classifiers (Lines et al., 2016). The intuition is that combining classifiers from different domains should perform better than using classifiers from a single domain (Bagnall et al., 2015). The superiority in classification accuracy requires learning of the parameters at training time (Bagnall and Lines, 2014).

Unfortunately, training these ensemble-based classifiers is very computationally demanding and is infeasible for large datasets. Two of the core classifiers in HIVE-COTE with the highest training time are the Shapelet Ensembles (SE) (Lines et al., 2016) and the Ensembles of Elastic Distances (EE) (Lines and Bagnall, 2015). SE is an ensemble that combines 8 base learners and uses the transformed time series data from the Shapelet Transform (ST) algorithm (Hills et al., 2014). Given a training set of N time series with length L, the ST algorithm has a training time complexity of $O(N^2 \cdot L^4)$.

EE is an ensemble of NN classifiers with 11 different time series distance measures. State-of-the-art learning of the parameters for all 11 classifiers is done through leave-one-out cross validation (LOO-CV). It has a training complexity of $O(M \cdot N^2 \cdot L^2)$, where $M = 100$ is the number of parameter values to learn for each classifier in EE (Bagnall et al., 2017; Lines et al., 2016). The high training complexity is because the learning algorithm needs to compare

each of the time series in the training set of size N with $N-1$ other instances and each comparison typically requires expensive $O(L^2)$ operations. This process is then repeated for all M parameter values, which is very inefficient and impractical for time series datasets with large N and long L.

The objective of this work is to speed up EE without affecting the classification accuracy. We propose the Fast Ensembles of Elastic Distances (FastEE) to reduce EE training time. FastEE extends recent work on speeding up the training time for the NN-DTW algorithm (Tan et al., 2018) to other distance measures. There are 2 versions of FastEE. The exact version trains FastEE using the full LOO-CV. The approximate version uses approximate LOO-CV. Note that our algorithms are developed using LOO-CV, such that the classification accuracy will be similar to EE, but are equally applicable to other parameter learning algorithms. FastEE minimizes the number of distances that need to be calculated by determining for each value calculated the range of parameter values for which that distance value applies, thus saving the majority of distance calculations undertaken by the standard approach. It also uses lower bounds on the distance measures to speed up each of the NN classifiers. As lower bounds have not been developed for several of the distance measures in EE, new lower bounds are proposed for these distances measures.

We show the significance of our FastEE algorithms in Figure 1, which compares the training time of EE to our FastEE algorithms as a function of training set size. To generate this plot, we sampled the `ElectricDevices` dataset – the largest dataset from the UCR benchmark time series archive (Chen et al., 2015) at different sizes and report the average training time. The red line shows that training EE on this dataset with merely 9,000 instances takes 17 days. Our exact FastEE, in blue, reduces this time to 2 days, while the approximate FastEE, in green, reduces it to 13 hours. The exact version is 10 times faster than EE, while learning the exact same classifier. The approximate version is 40 times faster, but may slightly compromise classification accuracy.

This paper is organised as follows. We provide the necessary background to understand our work and review key related work in TSC in Section 2. As lower bounding a distance measure has shown numerous successes in speeding up NN classifiers (Keogh and Ratanamahatana, 2005; Lemire, 2009; Rakthanmanon et al., 2012), we believe that it can also speed up EE. Section 3 describes the lower bounds required to speed up EE. We also propose new lower bounds for distance measures for which lower bounds have not previously been derived. Then in Section 4, we introduce FastEE to speed up the training time for EE. We evaluate the performance of FastEE in Section 5. Finally, we conclude our paper and provide some future directions in Section 6.

2 Background and related work

This section provides the necessary background to understand our proposed method and describes key related work in speeding up time series classification

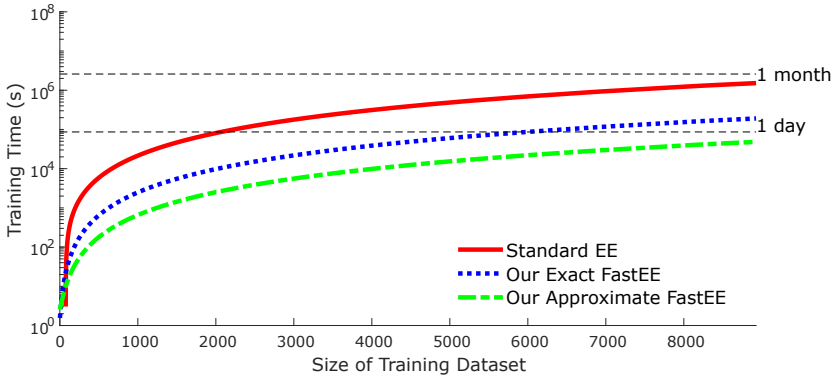


Fig. 1: Training time of EE (17 days), our exact FastEE (2 days) and our approximate FastEE (13 hours) on the `ElectricDevices` dataset at different sizes.

(TSC), specifically the nearest neighbour (NN) type classifiers. For simplicity, we assume that all the time series are of the same length L. However, it is trivial to generalize the algorithm to different lengths.

2.1 Ensembles of Elastic Distances

Similarity in the time domain has been a major focus for a large body of TSC research (Bagnall and Lines, 2014; Bagnall et al., 2017; Ding et al., 2008; Keogh and Ratanamahatana, 2005; Lines and Bagnall, 2015; Petitjean et al., 2012; Silva and Batista, 2016; Tan et al., 2017). The Ensembles of Elastic Distances (EE) is the most accurate TSC classifier in the time domain (Bagnall et al., 2017; Lines and Bagnall, 2015). EE is a meta classifier that consists of 11 NN classifiers, each of which uses a different time series distance measure (Lines and Bagnall, 2015). A NN classifier searches for the nearest neighbour within the training set and labels the query time series with the label of the nearest neighbour. Two time series are compared using a distance measure such as Euclidean distance or Dynamic Time Warping (DTW), that tries to achieve the optimal alignment between the two time series. Many distance measures are proposed for TSC in the last decade (Bagnall et al., 2017; Chen and Ng, 2004; Chen et al., 2005; Marteau, 2009; Sakoe and Chiba, 1971; Stefan et al., 2013). When tested on the UCR benchmark archive, they are not significantly different from each other in terms of classification accuracy (Lines and Bagnall, 2015). An ensemble (EE) generalises this so that the contributions from all the distance measures are considered, which significantly improves the classification accuracy (Lines and Bagnall, 2015).

As is typical for supervised classifiers, EE needs to be trained before performing any classification tasks. One of the key aspects of learning for EE

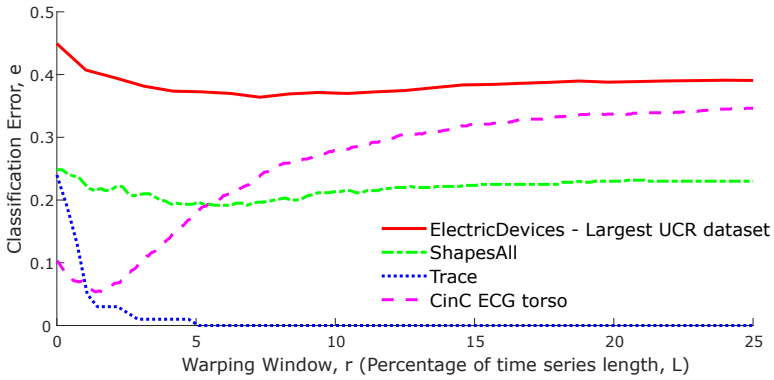


Fig. 2: Classification error for NN-DTW on some datasets from the UCR benchmark archive (Chen et al., 2015) at various warping windows r

.

is to find values for the parameters that affect the behaviour of the distance measures (Lines and Bagnall, 2015). For instance, the choice of warping window (parameter) for one of the distance measures – Dynamic Time Warping (DTW) significantly affects the error rate of NN-DTW, as shown in Figure 2. The best warping window is typically set through leave-one-out cross validation (LOO-CV) on the training set (Bagnall et al., 2017). It tries all of a predefined set of potential values and selects the one that gives the highest accuracy. LOO-CV may not be the best way to learn the best parameter, but it is simple, and effective in practice (Dau et al., 2017). EE learns the values of these parameters in a similar manner (Lines and Bagnall, 2015). Since EE is composed of NN type classifiers, this task then boils down to finding the nearest neighbour of each time series inside the training dataset for each of M parameter values.

Algorithm 1 describes the training process of EE using LOO-CV. The algorithm searches for the parameter value that gives the highest LOO-CV accuracy for each of the NN classifiers. Line 2 initialises the best accuracy for a NN classifier in EE. Lines 3-15 shows the typical procedure of doing LOO-CV for the NN classifier. Line 6 searches for the NN of the query Q from all the instances in the training set $\mathcal{T}$ except Q itself, with respect to the distance d_i. The label (class) of NN is the predicted label for Q. The parameter that gives the highest accuracy is stored.

Then for classification, EE weights the prediction from each of the NN classifiers using their LOO-CV accuracy (Lines and Bagnall, 2015). It has been shown that EE weighted with LOO-CV accuracy is significantly more accurate on the UCR datasets than all the individual NN classifiers Lines and Bagnall (2015). It is also part of the more accurate COTE (Bagnall et al., 2015) and HIVE-COTE (Lines et al., 2016).

Algorithm 1: TrainEE_LOOCV($\mathcal{T}, \mathtt{C}$)

Data: $\mathcal{T}$: training data with size N
Data: $\mathtt{C}$: set of NN classifiers paired with an elastic distance measure
Result: $\mathcal{P}^{\star}$: set of best parameter value for each elastic distance measure
Result: *bestAccuracy*: set of best LOO-CV accuracies for each elastic distance measure

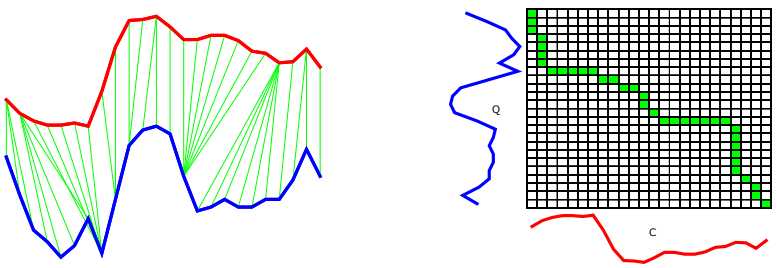
foreach $\mathtt{C}_i \in \mathtt{C}$ **do** // go through all classifiers in the ensemble
  for $\mathcal{P}_p^i \leftarrow \mathcal{P}_1^i$ **to** $\mathcal{P}_M^i$ **do** // go through all M parameters
    $nCorrect \leftarrow 0$
    foreach $Q \in \mathcal{T}$ **do** // do LOO-CV
      NN $= \mathtt{C}_i.\mathtt{nnSearch}(Q, \mathcal{T} \setminus Q, \mathcal{P}_p^i)$
      if NN.$class = Q.class$ **then**
        $nCorrect$++
      end
    end
    if $nCorrect > bestNCorrect_i$ **then** // select the higest accuracy
      $bestNCorrect_i \leftarrow nCorrect$
      $\mathcal{P}_i^{\star} \leftarrow \mathcal{P}_p^i$
    end
  end
  $bestAccuracy_i \leftarrow bestNCorrect_i / |\mathcal{T}|$
  $\mathtt{C}_i.\mathtt{setParam}(\mathcal{P}_i^{\star})$ // set the best parameter for classifier $\mathtt{C}_i$
end

2.2 Elastic distance measures

Since EE consists of NN type classifiers, the ultimate aim of this work is to minimise the training time of NN type classifiers. The NN type classifiers are each paired with a distance measure, that returns a distance, or similarity, score for a pair of time series. Thus it is important to first understand the principles of each distance measure. There are 8 different distance measures that are commonly used in the literature (Lines and Bagnall, 2015). Together with their variants, they form the 11 distance measures used in EE (Lines and Bagnall, 2015). This section briefly describes and generalises the different measures used in EE in order to understand our work in this paper. We refer interested readers to the paper (Lines and Bagnall, 2015) for more technical details.

2.2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a popular time series distance measure. It has been studied and tested extensively on the benchmark time series archive (Chen et al., 2015). DTW was firstly introduced as a spoken word recognition tool (Sakoe and Chiba, 1971, 1978) to handle distortions in the time axis which cannot be handled by the Euclidean distance. It stretches and realigns a time series to better match the other time series (Sakoe and Chiba, 1971). Figure 3a illustrates the matching of two time series Q and C using DTW.



(a) (b)

Fig. 3: (a) DTW alignment for two time series Q (blue) and C (red). (b) Cost matrix D_{DTW} with warping path $\mathcal{W}$ (green)

DTW finds the optimal path along the cost matrix $\mathcal{D}_{\text{DTW}}$. Each cell of the matrix $\mathcal{D}_{\text{DTW}}(i,j)$ represents the cumulative cost of aligning the two time series as described in Equation 1. An example of the optimal path is illustrated as the green path in Figure 3b. Note that most elastic distances also find this form of optimal path but with a different cost function.

$$\mathcal{D}_{\text{DTW}}(i,j) = (q_i - c_j)^2 + \min \begin{cases} \mathcal{D}_{\text{DTW}}(i-1,j-1) \\ \mathcal{D}_{\text{DTW}}(i,j-1) \\ \mathcal{D}_{\text{DTW}}(i-1,j) \end{cases} \tag{1}$$

Finding this optimal path can be very time consuming, especially for long time series. Hence it is common to apply a global constraint to the path where only points within a window range can be aligned (Itakura, 1975; Ratanamahatana and Keogh, 2004; Sakoe and Chiba, 1978). This is known as Constrained DTW. Furthermore, a global constraint also reduces pathological warping that will often reduce the classification accuracy (Keogh and Pazzani, 2001; Ratanamahatana and Keogh, 2004). There are many variants to this constraint such as the Ratanamahatana-Keogh Band (Ratanamahatana and Keogh, 2004), Itakura Parallelogram (Itakura, 1975), and the most widely adopted Sakoe-Chiba Band (Sakoe and Chiba, 1978). In this work, we focus on the Sakoe-Chiba band which is commonly known as warping window, r – a parameter to DTW and we write DTW with r as DTW_r.

2.2.2 Derivative Dynamic Time Warping

Derivative Dynamic Time Warping (DDTW) is a variant to DTW with the motivation to reduce singularities by transforming the time series into a first order derivative (Keogh and Pazzani, 2001). Singularities arise, for example, when a point on a rising trend is mapped to a point on a falling trend. First order derivatives eliminate this problem by considering the higher level feature

of shape rather than just the value of the data point in the time series (Keogh and Pazzani, 2001). DDTW is computed the same way as DTW using the transformed time series (Keogh and Pazzani, 2001). The derivative for a query time series $Q = \{q_1, , ..., q_L\}$ is $Q' = \{q'_2, ..., q'_{L-1}\}$. Equation 2 computes q'_i which is defined as the average of the slopes between q_{i-1}, q_i and q_{i+1} (Keogh and Pazzani, 2001). Note that, q'_i is not defined for the first and last element of the time series (Keogh and Pazzani, 2001). Similarly a warping window r can be applied to DDTW to better reduce pathological warping and speed up DDTW computations.

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

2.2.3 Weighted Dynamic Time Warping

Another variant of DTW – Weighted Dynamic Time Warping (WDTW) was also proposed to reduce pathological warping (Jeong et al., 2011). To prevent the alignment of q_i with c_j that are too far away in the time dimension, WDTW weighs the cost of aligning q_i to c_j using a modified logistic weight function described in Equation 3, where $\mathtt{w}_{max}$ is the upper bound for the weight parameter and is typically set to 1 (Jeong et al., 2011). The parameter g controls the level of penalization for further points (Jeong et al., 2011). The optimal range for g is distributed between 0.01 to 0.6 as suggested by the authors (Jeong et al., 2011). If i is far from j, it will have a larger weight and vice versa. Note that weights can also be applied to variants of DTW such as DDTW – giving Weighted DDTW (Jeong et al., 2011).

$$\mathtt{w}_a = \frac{\mathtt{w}_{\max}}{1 + e^{-g \cdot (a - L/2)}} \tag{3}$$

2.2.4 Longest Common Subsequence

The Longest Common Subsequence – LCSS is commonly used for pattern matching in the context of string sequences (Boreczky and Rowe, 1996; Vlachos et al., 2002). It finds the longest common subsequence that best matches the two string sequences and can be extended to numeric sequences (time series) using a distance threshold ε. Two elements are considered a match if the distance between them is less than ε. Each cell of the matrix $\mathcal{D}_{\text{LCSS}}(i, j)$ indicates the number of matches between the two time series (length of the longest common subsequence) described in Equation 4. A global constraint, Δ can also be applied to the LCSS distance.

$$\mathcal{D}_{\text{LCSS}}(i,j) = \begin{cases} 0 & \text{if } i = 0, j = 0 \\ 1 + \mathcal{D}_{\text{LCSS}}(i-1, j-1) & \text{if } |q_i - c_j| \leq \varepsilon \\ \max \begin{cases} \mathcal{D}_{\text{LCSS}}(i-1, j) \\ \mathcal{D}_{\text{LCSS}}(i, j-1) \end{cases} & \text{otherwise} \end{cases} \tag{4}$$

2.2.5 Edit Distance with Real Penalty

Edit Distance with Real Penalty (ERP) is an edit distance that satisfies the triangular inequality, which is important for indexing (Chen and Ng, 2004; Chen et al., 2005). Edit distances like DTW and LCSS are not metric and do not satisfy the triangular inequality. DTW does not satisfy the triangular inequality because it replicates the previous element when a gap is added. ERP uses the distance between the two points as the penalty if a gap is not added. If a gap is added, the penalty will be the distance between that point and a constant parameter g. This is described in Equation 5. The author suggested a penalty $g = 0$. Similarly the alignment path can also be constrained with the `bandsize` parameter to prevent singularities in alignment.

$$\mathcal{D}_{\text{ERP}}(i,j) = \min \begin{cases} \mathcal{D}_{\text{ERP}}(i-1,j-1) + (q_i - c_j)^2 \\ \mathcal{D}_{\text{ERP}}(i-1,j) + (q_i - g)^2 \\ \mathcal{D}_{\text{ERP}}(i,j-1) + (g - c_j)^2 \end{cases} \tag{5}$$

2.2.6 Move-Split-Merge

The Move-Split-Merge (MSM) distance is a metric that was proposed to overcome the limitations of previous distance measures: the Euclidean distance is not robust to temporal misalignment; edit distances like DTW and LCSS are not metric; the ERP distance is a metric but not translation invariant (Jeong et al., 2011). MSM is a metric, robust to temporal misalignment and translation invariant (Stefan et al., 2013). The cost for the *move* operation is the pairwise distance between two points (Stefan et al., 2013). The cost for the *split* and *merge* operations includes a constant penalty value `c` (Stefan et al., 2013). Equation 6 and 7 show the cost function for MSM and the computation of each elements in the cost matrix $\mathcal{D}_{\text{MSM}}$ (Stefan et al., 2013).

$$\mathcal{C}(q_i, q_{i-1}, c_j) = \begin{cases} \mathtt{c} & \text{if } q_{i-1} \leq q_i \leq c_j \text{ or } q_{i-1} \geq q_i \geq c_j \\ \mathtt{c} + \min \begin{cases} |q_i - q_{i-1}| \\ |q_i - c_j| \end{cases} & \text{otherwise} \end{cases} \tag{6}$$

$$\mathcal{D}_{\text{MSM}}(i,j) = \min \begin{cases} \mathcal{D}_{\text{MSM}}(i-1,j-1) + |q_i - c_j| \\ \mathcal{D}_{\text{MSM}}(i-1,j) + \mathcal{C}(q_i, q_{i-1}, c_j) \\ \mathcal{D}_{\text{MSM}}(i,j-1) + \mathcal{C}(c_j, q_i, c_{j-1}) \end{cases} \tag{7}$$

2.2.7 Time Warp Edit Distance

Previous distance measures do not consider the timestamps of the time series. Timestamps are important when time series are sampled with various sampling rates that might be non-uniform (Marteau, 2009). The Time Warp Edit

Distance (TWED) was proposed with the motivation of comparing time series using their timestamps (Marteau, 2009). There are three main operations ($delete_A$, $delete_B$ and $match$) used in TWED to transform the time series. A constant λ penalty is used in the *delete* operations. The *match* operation computes the distance of the current and previous data points. TWED controls warping in time series by multiplying the difference in timestamps with a constant stiffness parameter v. $v = \infty$ means that points that off-diagonal of the matrix are not considered and is similar to the Euclidean distance while $v = 0$ is similar to DTW (Marteau, 2009). The cost of all the three operations and the computation of each elements in the cost matrix $\mathcal{D}_{\text{TWED}}$ are described in Equation 8 and 9 respectively. Note that t_{q_i} is the timestamps.

$$\begin{aligned} match: \ \gamma_M &= (q_i - c_j)^2 + (q_{i-1} - c_{j-1})^2 + v(|t_{q_i} - t_{c_j}| + |t_{q_{i-1}} - t_{c_{j-1}}|) \\ delete_A: \ \gamma_A &= (q_i - q_{i-1})^2 + v(|t_{q_i} - t_{q_{i-1}}|) + \lambda \\ delete_B: \ \gamma_B &= (c_i - c_{i-1})^2 + v(|t_{c_i} - t_{c_{i-1}}|) + \lambda \end{aligned} \tag{8}$$

$$\mathcal{D}_{\text{TWED}}(i,j) = \min \begin{cases} \mathcal{D}_{\text{TWED}}(i-1, j-1) + \gamma_M & \text{match} \\ \mathcal{D}_{\text{TWED}}(i-1, j) + \gamma_A & \text{delete}_A \\ \mathcal{D}_{\text{TWED}}(i, j-1) + \gamma_B & \text{delete}_B \end{cases} \tag{9}$$

2.2.8 Generalising elastic distance

An elastic distance measure E transforms a time series to better match and align with another time series. All distance measures in EE are *elastic* other than Euclidean distance. Most of the elastic distance measures have $O(L^2)$ complexity to calculate. They can all be solved with dynamic programming using a $L \times L$ cost matrix $\mathcal{D}$. Thus, we say that E has an *alignment* path $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_K\}$ along the cost matrix $\mathcal{D}$ that aligns the two time series Q and C. Each of the element $\mathcal{W}_k = (i, j)$ is a link indicating q_i is aligned to c_j. Equation 10 shows a general equation for elastic distance measures, where $\mathcal{W}_k^1$ indicates the first element in $\mathcal{W}_k$, $\mathcal{W}_k^2$ the second, $\texttt{cost}_{\mathcal{P}}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$, represents the cost of aligning the two points $q_{\mathcal{W}_k^1}$ and $c_{\mathcal{W}_k^2}$, given a parameter $\mathcal{P}$ and is explained in previous section.

$$E(Q, C) = \sum_{k=1}^{K} \texttt{cost}_{\mathcal{P}}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2}) \tag{10}$$

The elastic distance measures are usually parametrised by one or two parameters that need to be learned at training time. Table 1 outlines the respective time complexity and parameters for each of the elastic distance measures. Overall, the parameters for elastic distance measures can be categorised into three types. The first is an alignment constraint parameter that constrains the alignment path of the cost matrix $\mathcal{D}$, such as the warping window in DTW. The second is a penalty cost to the alignment in case of misalignment. The

Elastic Distances Measures	Time Complexity	Path Constraint	Alignment Penalty	Threshold
Euclidean Distance	$O(L)$	-	-	-
Dynamic Time Warping	$O(L^2)$	-	-	-
Constrained DTW	$O(r \cdot L)$	r	-	-
Derivative DTW	$O(L^2)$	-	-	-
Constrained DDTW	$O(r \cdot L)$	r	-	-
Weighted DTW	$O(L^2)$	-	g	-
Weighted DDTW	$O(L^2)$	-	g	-
Longest Common Subsequence	$O(\Delta \cdot L)$	Δ	-	ε
Edit Distance with Real Penalty	$O(\text{bandsize} \cdot L)$	bandsize	g	-
Move-Split-Merge Distance	$O(L^2)$	-	c	-
Time Warp Edit Distance	$O(L^2)$	v	λ	-

Table 1: Elastic distance measures in with their time complexity and parameters

third is a threshold parameter, which determines whether two points are the same.

Algorithm 2 presents a general algorithm to fully compute any given elastic distance measure E (without a global alignment constraint) given their threshold or alignment penalty parameters, denoted by a single variable $\mathcal{P}$, the cost of aligning two points q_i, c_j and the cost of a path from q_1, c_1 to those points. Note that this cost is derived from the cost of the path to some subset of the costs of the paths to q_{i-1}, c_{j-1}; q_i, c_{j-1}; and q_{i-1}, c_j. For the case where the warping path is constrained, only the *start* and *end* variables need to be modified. For instance, given a constraint variable W, the respective *start* and *end* variables are replaced: $end \leftarrow W$ in Line 3, $start \leftarrow \max(2, i-W)$ in Line 12 and $end \leftarrow \min(L, i+W)$ in Line 13. Note that $W = r$ for DTW_r and DDTW_r; $W = \Delta$ for LCSS; $W =$ bandsize for ERP.

2.3 Related work

The accuracy of EE classifier, comes at a high cost of polynomial time complexity, which is prohibitive for long and large datasets. As shown in the previous section, elastic distance measures generally have polynomial $O(L^2)$ time complexity. A single classification of a query using the NN classifier requires $O(N \cdot L^2)$ operations, as the the algorithm needs to compare the query time series with all the N training instances. On the other hand, as indicated in Algorithm 1, with M parameter values to select from, a typical training time for a single NN classifier in EE requires $O(M \cdot N^2 \cdot L^2)$ operations. This is because for each of the N instances in the training set $\mathcal{T}$, the algorithm needs to scan through all $N-1$ examples to find its nearest neighbor and repeat for all M parameter values. This is infeasible when L is long and N is large as illustrated in Figure 1. An exhaustive search of all M parameters is time consuming, thus many applications use a subset of parameter values by setting $M = 100$. However as we will show in Section 5, training with $M = 100$ is still inefficient and slow.

A great amount of research has been done to speed up NN type classifiers, in particular the NN-DTW classifier (Keogh and Ratanamahatana, 2005; Kim et al., 2001; Lemire, 2009; Petitjean et al., 2014; Rakthanmanon et al., 2012;

Algorithm 2: Full Elastic Distance, $E(Q, C, \mathcal{P})$

Input: Q: Query time series
Input: C: Candidate time series
Input: $\mathcal{P}$: Cost or threshold parameter for an elastic distance measure, E
Output: d: Elastic distance measure

$L \leftarrow Q$.length
Let $\mathcal{D}$ be an $L \times L$ matrix initialized to ∞
end $\leftarrow L$
$\mathcal{D}(1,1) \leftarrow \texttt{cost}_{\mathcal{P}}(q_1, c_1)$
for $i \leftarrow 2$ *to end* **do** `// fill up the first column of matrix` $\mathcal{D}$
    $\mathcal{D}(i,1) \leftarrow \texttt{cost}_{\mathcal{P}}(q_i, c_1) + \texttt{pathcost}_{\mathcal{P}}(q_1, c_1, q_i, c_1)$
end
for $j \leftarrow 2$ *to end* **do** `// fill up the first row of matrix` $\mathcal{D}$
    $\mathcal{D}(1,j) \leftarrow \texttt{cost}_{\mathcal{P}}(q_1, c_j) + \texttt{pathcost}_{\mathcal{P}}(q_1, c_1, q_1, c_j)$
end
for $i \leftarrow 2$ *to* L **do** `// fill the rest of matrix` $\mathcal{D}$
    start $\leftarrow 2$
    end $\leftarrow L$
    for $j \leftarrow$ *start to end* **do**
        $\mathcal{D}(i,j) \leftarrow \texttt{cost}_{\mathcal{P}}(q_i, c_j) + \texttt{pathcost}_{\mathcal{P}}(q_1, c_1, q_i, c_j)$
    end
end
if E *is* LCSS **then**
    return $1 - \mathcal{D}(L, L)/L$
else
    return $\mathcal{D}(L, L)$
end

Silva and Batista, 2016; Tan et al., 2017). A common way is to use efficient lower bound functions with $O(1)$ to $O(L)$ complexity to minimise the expensive $O(L^2)$ distance computation and to reduce the search space (Keogh and Ratanamahatana, 2005; Kim et al., 2001; Lemire, 2009; Rakthanmanon et al., 2012; Shen et al., 2018). Different lower bounds can be cascaded to create tighter lower bounds that are more effective at pruning unpromising candidates (Rakthanmanon et al., 2012). It may be possible to determine lower bounds on the final distance during the dynamic programming process. Thus, the computation of an elastic distance measure can also be early abandoned (Rakthanmanon et al., 2012). The idea is to abandon the distance computation as soon as the cumulative distance in the cost matrix $\mathcal{D}$ is greater than the distance to the nearest neighbour (Rakthanmanon et al., 2012). Cells in the cost matrix $\mathcal{D}_{\text{DTW}}$ that are guaranteed to not be part of the DTW warping path can be skipped as well (Silva and Batista, 2016).

Contract algorithms are also popular to speed up a classification algorithm in terms of training and classification time without compromising on the classification accuracy (Flynn et al., 2019; Petitjean et al., 2014; Tan et al., 2017). A recent work proposed c-RISE (Flynn et al., 2019) that gives a contract training time for the Random Interval Spectral Ensemble (RISE) – a component of HIVE-COTE (Lines et al., 2016). Indexing techniques such as building a hierarchical K-means tree when combined with contract algorithms (Tan et al., 2017) are effective in reducing the classification time. Moreover, we can also

classify a time series by comparing it to the average time series in each of the classes in the training set (Petitjean et al., 2014). This significantly reduces the classification time and can improve the classification accuracy (Petitjean et al., 2014).

Since most elastic distance measures are very similar, these techniques developed for NN-DTW can be extended to other elastic distance measures to speed up this process. As will be discussed in Section 4, FastEE leverages off a recent work that utilises some of these techniques.

Recently, we proposed the Proximity Forest algorithm (Lucas et al., 2019), a close relative to EE, as a contribution to scalable time series classification. Proximity Forest is an ensemble of proximity trees where each tree is a combination of elastic distances used in EE (Lucas et al., 2019). Training of a single proximity tree first involves choosing a set of exemplars, a random time series per class. Then the exemplars are compared to every instance in the training set using a random distance measure with a random parameter. Finally, the training set is split based on the proximity of the training set to the exemplars and the process is repeated until the leaf nodes are pure. Proximity Forest has shown to be more scalable and accurate than EE (Lucas et al., 2019).

2.4 Learning the parameters of an elastic distance measure efficiently

As discussed in Section 2.2, most of the elastic distance measures require learning the parameter from a set of M parameter values using LOO-CV to perform well (Bagnall and Lines, 2014; Bagnall et al., 2017). LOO-CV has been used to learn the best parameter for an elastic distance for decades (Bagnall et al., 2017; Dau et al., 2017; Jeong et al., 2011; Ratanamahatana and Keogh, 2005). It is simple and works well in practice but it may not be the best method to learn the best parameter (Dau et al., 2017). There are other alternatives such as random search, k-fold or hold out cross validation to learn the parameters, our work only focus on LOO-CV as this is the learning method for EE. It is important to note that our algorithms developed in this work are equally applicable to the alternatives and will yield the same results. These alternatives may improve the performance of EE but we leave this as our future work.

Typically, the training time for a single NN classifier in EE requires $O(M \cdot N^2 \cdot L^2)$ operations and is infeasible for large N. Although there has been much research into speeding up the NN classifier (Section-2.3), they are still not efficient in dealing with hundreds of parameter values. For instance, our experiments in Section 5 show that there is no significant improvement in training time by using just the lower bounds. Furthermore, as demonstrated in (Tan et al., 2018), learning the exact best warping window (parameter) for DTW requires the enumeration of all possible windows ($M = L$) which is a laborious process (Tan et al., 2018). This process can be seen as searching for the nearest neighbour of all N time series for all windows, creating a $(N \times M)$ NNs table as shown in Table 2 (Tan et al., 2018). Filling this table naively

	Nearest neighbor at different parameters				
	0	1	$\cdots$	$M-1$	M
C_1	$C_{24}(2.57)$	$C_{55}(0.98)$	$\cdots$	$C_{55}(0.98)$	$C_{55}(0.98)$
$\vdots$			$\vdots$		
C_N	$C_{60}(4.04)$	$C_{47}(1.61)$	$\cdots$	$C_{47}(1.61)$	$C_{47}(1.61)$

Table 2: Table of NNs for each m. A cell $(i, m) = C_k(dist)$ means C_i has C_k as its NN for parameter m with distance $dist$.

requires $O(N^2 \cdot L^3)$ operations. Thus, it is common to explore just a subset of these windows, giving an approximate best warping window. Usually at least 100 windows in the range of 0 to L are considered (Bagnall et al., 2017). However this is still not efficient, as the complexity of $N^2 \cdot L^2$ takes over very quickly for large N and long L.

The *Fast Warping Window Search* (FastWWS) is an exact and efficient algorithm to learn the best warping window for DTW that is at least 2 orders of magnitude faster than the state of the art (Tan et al., 2018). The algorithm is based on the observation that many distance computations are redundant in the standard approach to learning the best warping window (Tan et al., 2018). It takes advantage of the relationship between DTW and its warping window (Tan et al., 2018). There are three main properties that form the basis of FastWWS (Tan et al., 2018). First, *a* DTW *warping path can be valid for several windows*. When unconstrained, DTW finds the optimum path that goes through the cost matrix $\mathcal{D}$. This optimum path has a maximum width $r^\star$ from the diagonal. If the window r is larger than $r^\star$, then the path will not change for any smaller window $r^\star \leq r' \leq r$ and thus will return the same distance, as illustrated in Figure 4. Warping windows whose DTW distances are predetermined need not be computed. Second, DTW *is monotone with the warping window*. Lastly, *The Keogh lower bound (lower bound of* DTW*) is monotone with warping window* as well. Lower bounds will be explained in more detail in the next section. These three properties allow the searching for the best warping window to start with the largest window L and proceed through ever smaller windows, skipping the computation of many of the DTW distances. We refer the interested reader to the paper (Tan et al., 2018) for a detailed explanation of the algorithm. With the aim of speeding up EE, this work generalises these properties and extends FastWWS to other elastic distance measures.

3 Proposed lower bounds for elastic distances

Lower bounding has shown a lot of success in speeding up the NN classifier especially the NN-DTW classifier (Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012; Tan et al., 2017). If a sequential search is performed through potential nearest neighbors for series A, if a lower bound on the distance to B exceeds the distance to the nearest neighbor found so far, then B

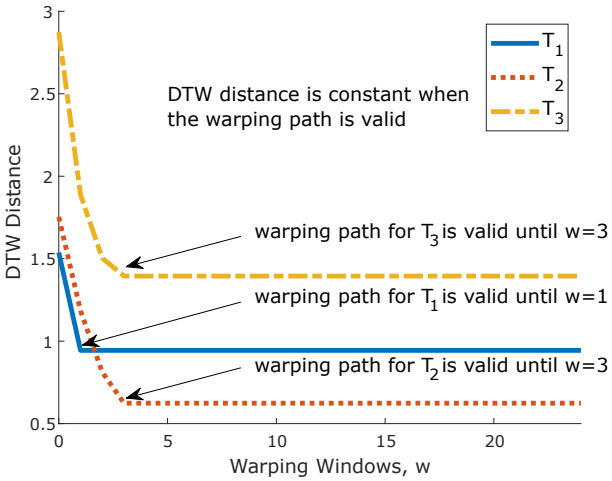


Fig. 4: DTW distances at different r (Tan et al., 2018)

can be discarded as a potential nearest neighbor without the need to calculate its distance. This speeds up NN type classifiers by minimising the number of expensive $O(L^2)$ distance computations. Typically an effective lower bound has cheap $O(1)$ to $O(L)$ complexity. EE is an ensemble of 11 NN classifiers with different elastic distance measures, thus it is possible to speed up EE using lower bounds. Appendix A describes the existing lower bounds that are used in this work.

To the best of our knowledge, no prior lower bounds have been derived for WDTW, MSM or TWED. In this section, we present new lower bounds for these measures. Note that some of these lower bounds may not be tight. Nonetheless, they are sufficient to provide reasonable speed ups. With the addition of these lower bounds, we have useable lower bounds for all the elastic measures in EE. We do not use a lower bound for Euclidean distance because its complexity is linear with respect to series length and hence it is efficient to compute in full.

3.1 WDTW lower bound

We define the lower bound for WDTW (LB_WDTW) in Equation 11 using similar intuition to LB_Keogh. First, we build the envelope time series for C. Since there are no alignment constraints on WDTW, we have the upper envelope $UE = C_{max}$ and the lower envelope $LE = C_{min}$. Then LB_WDTW distance is computed using Equation 11 by multiplying the sum with the minimum weight penalty $\mathtt{w}_0$.

$$\text{LB\_WDTW}(Q,C) = \sqrt{\mathtt{w}_0 \sum_{i=1}^{L} \begin{cases} (q_i - C_{\max})^2 & \text{if } q_i > C_{\max} \\ (q_i - C_{\min})^2 & \text{if } q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases}} \tag{11}$$

Theorem 1 *For any two time series Q and C of length L, and an alignment path $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_P\}$, where $\mathcal{W}_k = (i, j)$ indicates q_i is aligned to c_j, the following inequality holds:* $\text{LB\_WDTW}(Q,C) \leq \text{WDTW}(Q,C)$

Proof We can rewrite the equation of WDTW using $\mathcal{W}$ and the weights from Equation 3 into the following

$$\text{WDTW}(Q,C) = \sqrt{\sum_{k=1}^{P} \mathtt{w}_{|\mathcal{W}_k^1 - \mathcal{W}_k^2|} (q_{\mathcal{W}_k^1} - c_{\mathcal{W}_k^2})^2}$$

where $\mathcal{W}_k^1 = i, \mathcal{W}_k^2 = j$ and we wish to proof the following

$$\mathtt{w}_0 \sum_{i=1}^{L} \begin{cases} (q_i - C_{\max})^2 & \text{if } q_i > C_{\max} \\ (q_i - C_{\min})^2 & \text{if } q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases} \leq \sum_{k=1}^{P} \mathtt{w}_{|\mathcal{W}_k^1 - \mathcal{W}_k^2|} (c_{\mathcal{W}_k^2} - q_{\mathcal{W}_k^1})^2$$

Note that both sides are squared as the terms under the square root are both positive. We know that $L \leq P$, so we can match every term on the left hand side (LHS) with a term on the right hand side (RHS) giving $P - L$ terms unmatched.

$$\mathtt{w}_0 \sum_{i=1}^{L} \begin{cases} (q_i - C_{\max})^2 & \text{if } q_i > C_{\max} \\ (q_i - C_{\min})^2 & \text{if } q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases} \leq \begin{array}{l} \displaystyle\sum_{k \in matched} \mathtt{w}_{|\mathcal{W}_k^1 - \mathcal{W}_k^2|} (q_{\mathcal{W}_k^1} - c_{\mathcal{W}_k^2})^2 + \\ \displaystyle\sum_{k \in unmatched} \mathtt{w}_{|\mathcal{W}_k^1 - \mathcal{W}_k^2|} (q_{\mathcal{W}_k^1 - c_{\mathcal{W}_k^2}})^2 \end{array}$$

Let us consider the relationship between the matched terms on the RHS and the LHS terms. There are three cases to be considered on the LHS of the inequality. We start with the first one $q_i > C_{\max}$. From Equation 3, $\mathtt{w}_0$ is the minimum weight, so $\mathtt{w}_0 \leq \mathtt{w}_{|i-j|}$. Since $C_{\max} \geq c_j$ and if $q_i > C_{\max}$, then $(q_i - C_{\max})^2 \leq (q_i - c_j)^2$. Thus $\mathtt{w}_0 (q_i - C_{\max})^2 \leq \mathtt{w}_{|j-i|} (q_i - c_j)^2$. Similarly if $q_i < C_{\min}$, and $C_{\min} \leq c_j$, then $(q_i - C_{\min})^2 \leq (q_i - c_j)^2$. Since $(q_i - c_j)^2$ is non-negative, the third case yields $0 \leq (q_i - c_j)^2$.

If all the matched terms are larger than the LHS terms, then the unmatched terms will need to be negative for the inequality to be false. Fortunately, it is impossible for the unmatched terms to be negative since $\mathtt{w}_{|i-j|} > 0$ (from Equation 3) and the squared terms can never be negative. Therefore our inequality holds and $\text{LB\_WDTW}(Q,C) \leq \text{WDTW}(Q,C)$. □

3.2 MSM lower bound

We define the lower bound for MSM (LB_MSM) in Equation 12. The first term of LB_MSM is $|q_1 - c_1|$ and is extracted directly from MSM (Stefan et al., 2013). Following Equation 6, if $q_{i-1} \geq q_i > C_{\max}$, the lower bound adds the minimum of $|q_i - C_{\max}|$ and $\mathtt{c}$. Similarly if $q_{i-1} \leq q_i < C_{\min}$, the lower bound adds the minimum of $|q_i - C_{\min}|$ and $\mathtt{c}$.

$$\text{LB\_MSM}(Q, C) = |q_1 - c_1| + \sum_{i=2}^{L} \begin{cases} \min(|q_i - C_{\max}|, \mathtt{c}) & \text{if } q_{i-1} \geq q_i > C_{\max} \\ \min(|q_i - C_{\min}|, \mathtt{c}) & \text{if } q_{i-1} \leq q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Theorem 2 *For any two time series Q and C of length L, and an alignment path $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_P\}$, where $\mathcal{W}_k = (i, j)$ indicates q_i is aligned to c_j, the following inequality holds:* $\text{LB\_MSM}(Q, C) \leq \text{MSM}(Q, C)$

Proof We can rewrite the equation of MSM from Equation 7 using $\mathcal{W}$ into the following:

$$\text{MSM}(Q, C) = |q_1 - c_1| + \sum_{k=2}^{P} \mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

where $\mathcal{W}_k^1 = i, \mathcal{W}_k^2 = j$, $\mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$ represents the cost of aligning $q_{\mathcal{W}_k^1}$ to $c_{\mathcal{W}_k^2}$ under MSM. From Equation 7, MSM is based on three different cost values.

$$\mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2}) = \begin{cases} |q_i - c_j| \\ \mathcal{C}(q_i, q_{i-1}, c_j) \\ \mathcal{C}(c_j, q_i, c_{j-1}) \end{cases}$$

where $\mathcal{C}(x, y, z)$ is defined in Equation 6. The cost is $\mathtt{c} + \min(|x - y|, |x - z|)$ if x is not between y and z otherwise the cost is $\mathtt{c}$. And we wish to proof the following

$$\sum_{i=2}^{L} \begin{cases} \min(|q_i - C_{\max}|, \mathtt{c}) & \text{if } q_{i-1} \geq q_i > C_{\max} \\ \min(|q_i - C_{\min}|, \mathtt{c}) & \text{if } q_{i-1} \leq q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases} \leq \sum_{k=2}^{P} \mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

The proof for the first term is trivial as they are equal on both sides. We know that $L \leq P$, so we can match every term on the LHS with a term on the RHS, giving $P - L$ terms unmatched.

$$\sum_{i=2}^{L} \begin{cases} \min(|q_i - C_{\max}|, \mathtt{c}) & \text{if } q_{i-1} \geq q_i > C_{\max} \\ \min(|q_i - C_{\min}|, \mathtt{c}) & \text{if } q_{i-1} \leq q_i < C_{\min} \\ 0 & \text{otherwise} \end{cases} \leq \sum_{k \in matched} \mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2}) + \sum_{k \in unmatched} \mathtt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

Let us consider the relationship between the matched terms on the RHS and the LHS terms. We wish to proof that for each $i > 1$ term, all the 3 cases inside LB_MSM is less than or equal to the minimum of the cost functions.

For the first case $q_{i-1} \geq q_i > C_{\max}, C_{\max} \geq c_j$, and assuming $|q_i - C_{\max}| \leq \mathtt{c}$,

- $|q_i - C_{\max}| \leq |q_i - c_j|$ because $C_{\max} \geq c_j$, i.e. c_j is further from q_i than $C_{\max}$.
- $|q_i - C_{\max}| \leq \mathcal{C}(q_i, q_{i-1}, c_j)$ because $|q_i - C_{\max}| \leq \mathtt{c}$, $\mathtt{c} \leq \mathcal{C}(q_i, q_{i-1}, c_j)$ and $q_{i-1} \geq q_i > C_{\max} \therefore \mathcal{C}(q_i, q_{i-1}, c_j) = \mathtt{c}$ from Equation 6.
- $|q_i - C_{\max}| \leq \mathcal{C}(c_j, q_i, c_{j-1})$ because $|q_i - C_{\max}| \leq \mathtt{c}$ and $\mathtt{c} \leq \mathcal{C}(c_j, q_i, c_{j-1})$ from Equation 6.

Assuming if $\mathtt{c} \leq |q_i - C_{\max}|$, then $\mathtt{c} \leq |q_i - C_{\max}| \leq |q_i - c_j|$, $\mathtt{c} \leq \mathcal{C}(q_i, q_{i-1}, c_j)$ and $\mathtt{c} \leq \mathcal{C}(c_j, q_i, c_{j-1})$ are still valid and thus the above proof holds.

For the second case $q_{i-1} \leq q_i \leq C_{\min}, C_{\min} \leq c_j$, and $|q_i - C_{\min}| \leq \mathtt{c}$,

- $|q_i - C_{\min}| \leq |q_i - c_j|$ because $C_{\min} \leq c_j$, i.e. c_j is further from q_i than $C_{\min}$.
- $|q_i - C_{\min}| \leq \mathcal{C}(q_i, q_{i-1}, c_j)$ because $|q_i - C_{\min}| \leq \mathtt{c}$, $\mathtt{c} \leq \mathcal{C}(q_i, q_{i-1}, c_j)$ and $q_{i-1} \leq q_i < C_{\min} \therefore \mathcal{C}(q_i, q_{i-1}, c_j) = \mathtt{c}$ from Equation 6.
- $|q_i - C_{\min}| \leq \mathcal{C}(c_j, q_i, c_{j-1})$ because $|q_i - C_{\min}| \leq \mathtt{c}$ and $\mathtt{c} \leq \mathcal{C}(c_j, q_i, c_{j-1})$ from Equation 6.

Similarly, the above proof holds for the case where $\mathtt{c} \leq |q_i - C_{\min}|$. The third case is always true because all of the cost functions are non-negative.

Since all the matched terms are larger than LHS, then the sum of the unmatched terms has to be negative for the inequality to be false. This is not possible because $\mathtt{c} \geq 0$ is non negative and the absolute differences in the cost values and Equation 6 can never be negative. Therefore $\text{LB\_MSM}(Q, C) \leq \text{MSM}(Q, C)$ holds. □

3.3 TWED lower bound

TWED takes into account the differences in timestamps $t_i - t_{i-1}$ in the cost of aligning a time series pair. Since in this work, we only consider time series that are equally spaced and use the indexes of the time series points as the timestamps, we will always have $t_i - t_{i-1} = 1$. A lower bound for TWED was proposed in (Marteau, 2009) for range query search. The lower bound

down-samples the time series and computes the TWED distance of the down-sampled time series. In a worst case scenario, the down-sampled time series could be the full time series and the lower bound will be more expensive to compute than the full distance. Thus, we define a new lower bound for TWED (LB_TWED) in Equation 13. Note that this lower bound is also applicable for $t_i - t_{i-1} > 1$.

$$\text{LB\_TWED}(Q,C) = \min \begin{cases} (q_1 - c_1)^2 \\ q_1^2 + v + \lambda \\ c_1^2 + v + \lambda \end{cases} + \sum_{i=2}^{L} \begin{cases} \min(v, (q_i - \max(C_{\max}, q_{i-1}))^2) & \text{if } q_i > \max(C_{\max}, q_{i-1}) \\ \min(v, (q_i - \min(C_{\min}, q_{i-1}))^2) & \text{if } q_i < \min(C_{\min}, q_{i-1}) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Theorem 3 *For any two time series Q and C of length L, a stiffness parameter $v \geq 0$ and a constant penalty $\lambda \geq 0$, and an alignment path $\mathcal{W} = \{\mathcal{W}_1, ..., \mathcal{W}_P\}$, where $\mathcal{W}_k = (i, j)$ indicates q_i is aligned to c_j, the following inequality holds:* $\text{LB\_TWED}(Q,C) \leq \text{TWED}(Q,C)$

Proof We can rewrite the equation for TWED using $\mathcal{W}$ as the following

$$\text{TWED}(Q,C) = \sum_{k=1}^{P} \texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

where $\mathcal{W}_k^1 = i, \mathcal{W}_k^2 = j$, $\texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$ represents the cost of aligning $q_{\mathcal{W}_k^1}$ to $c_{\mathcal{W}_k^2}$ under TWED. From Equation 8, TWED is based on three different cost values.

$$\texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2}) = \begin{cases} (q_i - c_j)^2 + v|t_i^Q - t_j^C| + (q_{i-1} - c_{j-1})^2 + v|t_{i-1}^Q - t_{j-1}^C| \\ (q_i - q_{i-1})^2 + v|t_i^Q - t_{i-1}^Q| + \lambda \\ (c_j - c_{j-1})^2 + v|t_j^C - t_{j-1}^C| + \lambda \end{cases}$$

where $q_0 = 0$, $c_0 = 0$, $t_0^Q = 0$ and $t_0^C = 0$. and we wish to proof the following

$$\text{LB\_TWED}(Q,C) \leq \sum_{k=1}^{P} \texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

We know that $L \leq P$, so we can match every term on the LHS with a term on the RHS, giving $P - L$ terms unmatched.

$$\text{LB\_TWED}(Q,C) \leq \sum_{k \in matched} \texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2}) + \sum_{k \in unmatched} \texttt{cost}(q_{\mathcal{W}_k^1}, c_{\mathcal{W}_k^2})$$

Let us consider the relationship between matched terms on the RHS and the LHS terms. We wish to prove that for all $i > 0$, all the terms in Equation 13 are less than or equal to all the cost functions on the RHS. Due to boundary conditions, TWED must align the points at $i = 1$, $j = 1$. So for $i = 1$ it is trivial to see that,

$$\min \begin{cases} (q_1 - c_1)^2 & (q_1 - c_1)^2 \\ q_1^2 + v + \lambda & \leq (q_1 - 0)^2 + v t_1^Q + \lambda \\ c_1^2 + v + \lambda & (c_1 - 0)^2 + v t_1^C + \lambda \end{cases}$$

There are three cases for $i \geq 2$, and we shall start with the first one $q_i > \max(C_{\max}, q_{i-1})$, assuming $C_{\max} \geq q_{i-1}$ then $q_i > C_{\max}$. We will first show the proof by assuming $(q_i - C_{\max})^2 \leq v$.

- $(q_i - C_{\max})^2 \leq (q_i - c_j)^2 + v|t_i^Q - t_j^C| + (q_{i-1} - c_{j-1})^2 + v|t_{i-1}^Q - t_{j-1}^C|$ because $C_{\max} \geq c_j$ implies that c_j is further from q_i than $C_{\max}$, so $(q_i - C_{\max})^2 \leq (q_i - c_j)^2$. All the other terms cannot be negative.
- $(q_i - C_{\max})^2 \leq (q_i - q_{i-1})^2 + v|t_i^Q - t_{i-1}^Q| + \lambda$ because $C_{\max} \geq q_{i-1}$ implies that q_{i-1} is further from q_i than $C_{\max}$, so $(q_i - C_{\max})^2 \leq (q_i - q_{i-1})^2$. All the other terms cannot be negative.
- $(q_i - C_{\max})^2 \leq (c_j - c_{j-1})^2 + v|t_j^C - t_{j-1}^C| + \lambda$ because $(q_i - C_{\max})^2 \leq v$, $|t_j^C - t_{j-1}^C| > 0$ and all the other terms cannot be negative.

Assuming $v \leq (q_i - C_{\max})^2$, $v \leq (q_i - c_j)^2$ because $(q_i - C_{\max})^2 \leq (q_i - c_j)^2$. $v \leq |t_i^Q - t_{i-1}^Q|$ and $v \leq |t_j^C - t_{j-1}^C|$ because $|t_i - t_{i-1}| > 0$. Thus the proof still holds.

For the case $q_{i-1} \geq C_{\max}$ then $q_i > q_{i-1}$, and assuming $(q_i - q_{i-1})^2 \leq v$, the opposite $v \leq (q_i - q_{i-1})^2$ will still hold by applying the same reasoning.

- $(q_i - q_{i-1})^2 \leq (q_i - c_j)^2 + v|t_i^Q - t_j^C| + (q_{i-1} - c_{j-1})^2 + v|t_{i-1}^Q - t_{j-1}^C|$ because $q_i > q_{i-1}$, $q_{i-1} \geq C_{\max}$ and $C_{\max} \geq c_j$ implies that c_j is further from q_i than q_{i-1} so $(q_i - q_{i-1})^2 \leq (q_i - c_j)^2$. All the other terms cannot be negative.
- $(q_i - q_{i-1})^2 \leq (q_i - q_{i-1})^2 + v|t_i^Q - t_{i-1}^Q| + \lambda$ is trivial because all the terms cannot be negative.
- $(q_i - q_{i-1})^2 \leq (c_j - c_{j-1})^2 + v|t_j^C - t_{j-1}^C| + \lambda$ because $(q_i - q_{i-1})^2 \leq v$, $|t_j^C - t_{j-1}^C| > 0$ and all the other terms cannot be negative.

A similar proof can be applied to the second case $q_i < \min(C_{\min}, q_{i-1})$. The third case is trivial as all of the costs are non-negative.

Since all the matched terms are larger than the LHS, the unmatched terms have to be negative for the inequality to be false. Fortunately, this is not possible because $v \geq 0$, $\lambda \geq 0$, $|t_i - t_j| \geq 0$ and the squared terms can never be negative. Therefore $\text{LB\_TWED}(Q, C) \leq \text{TWED}(Q, C)$ holds. □

4 FASTEE: FAST Ensembles of Elastic Distances

Given an elastic distance measure E, N training instances and M parameters to learn, learning the best parameter for E can be seen as creating a $N \times M$ table, similar to Table 2, giving the NN for every time series for all parameters. As demonstrated in (Tan et al., 2018), training just a single classifier (NN-DTW) is very computationally expensive. This is even more problematic when there are 11 classifiers to train. In this section, we introduce *Fast Ensembles of Elastic Distances* (FastEE), an extension of FastWWS to speed up the training time for EE.

4.1 Properties for FastEE

We can generalise the main properties of FastWWS to other elastic distances. We let $\mathcal{P}^p$, $\mathcal{P}^c$ and $\mathcal{P}^t$ be the alignment penalty, path constraint and threshold parameter respectively.

Property #1: Any elastic distance measure E is monotone with its alignment penalty constraint parameter $\mathcal{P}^p$

Theorem 4 *Let E be the elastic distance measure of interest, Q, C be two time series, $\mathcal{P}^p$ be an alignment penalty, $\hat{\mathcal{P}}^p$ the smaller penalty and $\mathcal{W}^{\mathcal{P}^p}$ the associated alignment path. Then we have $E_{\mathcal{P}^p}(Q,C) \geqslant E_{\hat{\mathcal{P}}^p}(Q,C)$.*

Proof Let $\texttt{cost}_{\mathcal{P}^p}(q_i, c_j)$ be the cost of aligning the two points q_i and c_j. If $\mathcal{P}^p \geq \hat{\mathcal{P}}^p$, then $\texttt{cost}_{\mathcal{P}^p}(q_i, c_j) \geq \texttt{cost}_{\hat{\mathcal{P}}^p}(q_i, c_j)$. Otherwise, the alignment path will not be optimum. □

Property #2: For any elastic distance measure E, alignment path can be valid for several path constraint parameter $\mathcal{P}^c$

Theorem 5 *Let E be the elastic distance measure of interest, Q, C be two time series, $\mathcal{P}^c_1$ and $\mathcal{P}^c_2$ two path constraint parameters and $\mathcal{W}^{\mathcal{P}^c_1}$ and $\mathcal{W}^{\mathcal{P}^c_2}$ their associated alignment paths. $\mathcal{W}^{\mathcal{P}^c_1} = \mathcal{W}^{\mathcal{P}^c_2} \Rightarrow E_{\mathcal{P}^c_1}(Q,C) = E_{\mathcal{P}^c_2}(Q,C)$. In other words, $E_{\mathcal{P}^c_1}(Q,C)$ can only differ from $E_{\mathcal{P}^c_2}(Q,C)$ if the alignment path differs.*

Proof Let $\mathcal{W}^{\mathcal{P}^c_1} = \langle (i_1^{\mathcal{P}^c_1}, j_1^{\mathcal{P}^c_1}), \cdots, (i_K^{\mathcal{P}^c_1}, j_K^{\mathcal{P}^c_1}) \rangle$, $\mathcal{W}^{\mathcal{P}^c_2} = \langle (i_1^{\mathcal{P}^c_2}, j_1^{\mathcal{P}^c_2}), \cdots, (i_K^{\mathcal{P}^c_2}, j_K^{\mathcal{P}^c_2}) \rangle$. We have:

$$\begin{aligned} E_{\mathcal{P}^c_1}(Q,C) &= \textstyle\sum_{k=1}^{K} \texttt{cost}_{\mathcal{P}^p}(q_{i_k^{\mathcal{P}^c_1}}, c_{j_k^{\mathcal{P}^c_1}}) \qquad \text{Eq 10} \\ &= \textstyle\sum_{k=1}^{K} \texttt{cost}_{\mathcal{P}^p}(q_{i_k^{\mathcal{P}^c_2}}, c_{j_k^{\mathcal{P}^c_2}}) \text{ (By hyp.)} \\ &= E_{\mathcal{P}^c_2}(Q,C) \end{aligned}$$

□

Theorem 6 *Let E be the elastic distance measure of interest, Q, C be two time series, $\mathcal{P}^c$ be a path constraint, $\hat{\mathcal{P}}^c$ the smaller path constraint and $\mathcal{W}^{\mathcal{P}^c}$ the associated alignment path. Then*

$$(|i_k - j_k| < \mathcal{P}^c)_{\forall k} \Rightarrow E_{\mathcal{P}^c}(Q, C) = E_{\hat{\mathcal{P}}^c}(Q, C)$$

In other words, if all the points in an alignment path are within the boundaries of the path constraint $\mathcal{P}^c$, then $E_{\mathcal{P}^c}(Q, C) = E_{\hat{\mathcal{P}}^c}(Q, C)$.

Proof All elastic distance measures find an alignment path $W^{\mathcal{P}^c}$ such that

$$\sum_{k=1}^{K} \texttt{cost}_{\mathcal{P}^p}(q_{i_k}, c_{j_k})$$

is minimized respecting the constraint $|i_k - j_k| \leqslant \mathcal{P}^c$. □

In FastWWS, this property is known as the "window validity" (Tan et al., 2018). Thus, similarly we say that $\mathrm{E}_{\mathcal{P}^c}(Q, C)$ has a "path validity" and the path is valid if all the alignment paths do not change. Note that this property is valid only for a fixed alignment penalty parameter $\mathcal{P}^p$ and threshold parameter $\mathcal{P}^t$.

Property #3: Any elastic distance measure E is monotone with its path constraint parameter $\mathcal{P}^c$

Theorem 7 *Let E be the elastic distance measure of interest except* TWED*, Q, C be two time series, and $\mathcal{P}^c$ a parameter of E, $\hat{\mathcal{P}}^c < \mathcal{P}^c$, we have $E_{\mathcal{P}^c}(Q, C) \leqslant E_{\hat{\mathcal{P}}^c}(Q, C)$ and* $\mathrm{TWED}_{\mathcal{P}^c}(Q, C) \geqslant \mathrm{TWED}_{\hat{\mathcal{P}}^c}(Q, C)$.

Proof Assume $\mathrm{E}_{\mathcal{P}^c}(Q, C) > \mathrm{E}_{\hat{\mathcal{P}}^c}(Q, C)$, then this means that there exists an alignment path $\mathcal{W}_{\hat{\mathcal{P}}^c}$ such that the associated cost is lower than the one for $\mathcal{W}_{\mathcal{P}^c}$. This entails E not having found the optimal solution at $\mathcal{P}^c$, which is a contradiction. □

Note that larger path constraint parameter for TWED, v gives larger distance which is the opposite of the path constraint parameters for all the other elastic distances. Despite that, the proof is still applicable by just inverting the signs.

Property #4: For any elastic distance measure E with a path constraint parameter $\mathcal{P}^c$, its lower bound is monotone with $\mathcal{P}^c$

Theorem 8 *Let E be the elastic distance of interest except* TWED*, Q, C be two time series, $\mathcal{P}^c$ a parameter of E, $\hat{\mathcal{P}}^c < \mathcal{P}^c$, and LB a lower bound of E, we have $LB_{\mathcal{P}^c}(Q, C) \leqslant LB_{\hat{\mathcal{P}}^c}(Q, C)$ and* $\mathrm{LB\_TWED}_{\mathcal{P}^c}(Q, C) \geqslant \mathrm{LB\_TWED}_{\hat{\mathcal{P}}^c}$.

Proof To prove this property, we have to look at each of the lower bounds for distances with path constraint parameter – DTW, ERP, LCSS and TWED. The proof for DTW can be found in (Tan et al., 2018). The lower bound for ERP is an adaptation of LB_Keogh for DTW. Thus, its proof is the same as DTW.

The lower bound for LCSS is bounded by an upper $\mathbb{UE}$ and lower $\mathbb{LE}$ envelopes. Let $\mathcal{P}^c = \Delta$, $\mathbb{UE}_i^{\Delta} = \max(c_{i-\Delta} : c_{i+\Delta}) + \varepsilon$ be the elements in the upper envelope and $\mathbb{LE}_i^{\Delta} = \min(c_{i-\Delta} : c_{i+\Delta}) - \varepsilon$ as the lower envelope. We wish to proof

$$\text{LB\_LCSS}_{\Delta}(Q, C) \leqslant \text{LB\_LCSS}_{\Delta-1}(Q, C)$$

We will assume the opposite and show that it leads to a contradiction:

$$\text{LB\_LCSS}_{\Delta}(Q, C) > \text{LB\_LCSS}_{\Delta-1}(Q, C)$$

$$1 - \frac{1}{L}\sum_{i=1}^{L}\begin{cases}1 & \text{if } \mathbb{LE}_i^{\Delta} \leq q_i \leq \mathbb{UE}_i^{\Delta} \\ 0 & \text{otherwise}\end{cases} > 1 - \frac{1}{L}\sum_{i=1}^{L}\begin{cases}1 & \text{if } \mathbb{LE}_i^{\Delta-1} \leq q_i \leq \mathbb{UE}_i^{\Delta-1} \\ 0 & \text{otherwise}\end{cases}$$

$$\sum_{i=1}^{L}\begin{cases}1 & \text{if } \mathbb{LE}_i^{\Delta} \leq q_i \leq \mathbb{UE}_i^{\Delta} \\ 0 & \text{otherwise}\end{cases} < \sum_{i=1}^{L}\begin{cases}1 & \text{if } \mathbb{LE}_i^{\Delta-1} \leq q_i \leq \mathbb{UE}_i^{\Delta-1} \\ 0 & \text{otherwise}\end{cases}$$

The above implies that larger envelope at $\mathcal{P}^c$ gives a larger lower bound distance than smaller envelopes, which contradicts with Equation 21 that defines LB_LCSS as the percentage of points that are inside the envelope. When $\mathcal{P}^c$ increases, the envelope gets larger, thus more points from Q will fall into the envelope. Hence, the number of q_i points in the larger envelopes will be larger than the smaller envelopes giving longer common subsequence and consequently a smaller distance.

TWED monotonically increases with its path constraint parameter $\mathcal{P}^c = v$ and we let $v' < v$. Since $v > v'$ and the rest of the Equation 13 is constant regardless of v, it is trivial to see that the inequalities for all the terms in Equation 13 holds.

$$\text{LB\_TWED}_{v}(Q, C) \geqslant \text{LB\_TWED}_{v'}(Q, C)$$

$$\min\begin{cases}(q_1 - c_1)^2 \\ q_1^2 + v + \lambda \\ c_1^2 + v + \lambda\end{cases} \geqslant \min\begin{cases}(q_1 - c_1)^2 \\ q_1^2 + v' + \lambda \\ c_1^2 + v' + \lambda\end{cases}$$

$$\sum_{i=2}^{L}\begin{cases}\min(v, (q_i - A)^2) & \text{if } q_i > A \\ \min(v, (q_i - B)^2) & \text{if } q_i < B \\ 0 & \text{otherwise}\end{cases} \geqslant \sum_{i=2}^{L}\begin{cases}\min(v', (q_i - A)^2) & \text{if } q_i > A \\ \min(v', (q_i - B)^2) & \text{if } q_i < B \\ 0 & \text{otherwise}\end{cases}$$

where $A = \max(C_{\max}, q_{i-1})$, $B = \min(C_{\min}, q_{i-1})$. □

Property #5: For any elastic distance measure E with a penalty parameter $\mathcal{P}^p$, its lower bound is monotone with $\mathcal{P}^p$

Theorem 9 *Let E be the elastic distance measure of interest, Q, C be two time series, $\mathcal{P}^p$ a parameter of E, $\hat{\mathcal{P}}^p < \mathcal{P}^p$, and LB a lower bound of E, we have $LB_{\mathcal{P}^p}(Q,C) \geqslant LB_{\hat{\mathcal{P}}^p}(Q,C)$.*

Proof Since the penalty parameter $\mathcal{P}^p \geq 0$ and all lower bounds for elastic distances are additive, it is trivial that $LB_{\mathcal{P}^p}(Q,C) \geqslant LB_{\hat{\mathcal{P}}^p}(Q,C)$ because $\mathcal{P}^p > \hat{\mathcal{P}}^p$ □

Property #6: For any elastic distance measure E with a threshold parameter $\mathcal{P}^t$, its lower bound is monotone with $\mathcal{P}^t$

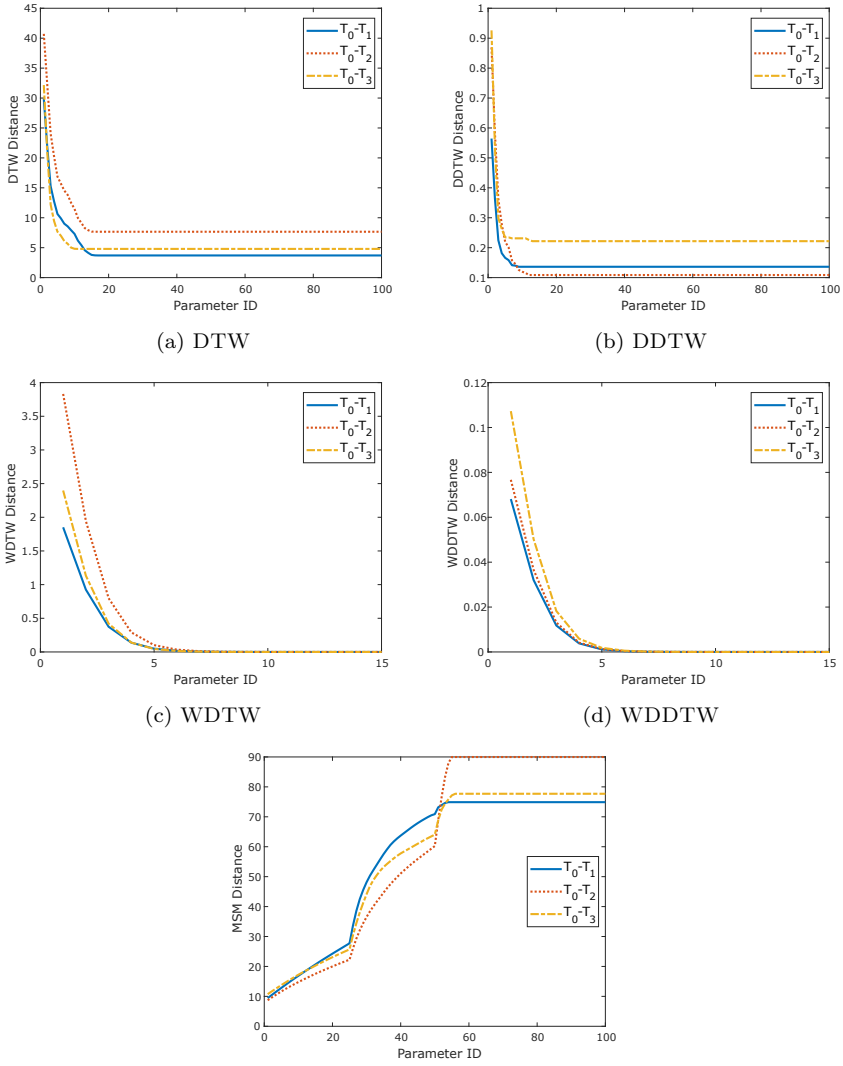
Theorem 10 *Let E be the elastic distance measure of interest, Q, C be two time series, $\mathcal{P}^t$ a parameter of E, $\hat{\mathcal{P}}^t < \mathcal{P}^t$, and LB a lower bound of E, we have $LB_{\mathcal{P}^t}(Q,C) \geqslant LB_{\hat{\mathcal{P}}^t}(Q,C)$.*

Proof Considering the elastic distance measures used in this work, only the LCSS distance has the threshold parameter, thus the proof will be based on LB_LCSS. Let $\mathcal{P}^t = \varepsilon$, LB_LCSS is bounded by an upper $\mathbb{UE}_i = \max(c_{i-\Delta} : c_{i+\Delta}) + \varepsilon$ and lower $\mathbb{LE}_i = \min(c_{i-\Delta} : c_{i+\Delta}) - \varepsilon$ envelopes which is built based on ε. From Equation 21, LB_LCSS is defined as the percentage of points that are inside the envelope. When $\mathcal{P}^t$ increases, the envelope gets larger, thus more points from C will fall into the envelope. Hence, the number of points in the larger envelopes will be larger than the smaller envelopes giving longer common subsequence and consequently a smaller distance. □

Figure 5 and 6 illustrate the combination of all the properties for each of the elastic distances. In this work, to be consistent with EE, only 100 parameters are chosen for each of the distances (Bagnall et al., 2017). Figure 5 shows the properties for elastic distances with a single parameter, while Figure 6 with two parameters. For distances with two parameters, 10 values are chosen uniformly for each of the parameters, creating a combination of 100 parameter values. There are no distances used with three parameters. Note that the nearest neighbour (lowest distance) changes as the parameter changes.

The DTW family, DTW, DDTW, WDTW and WDDTW monotonically decrease with increasing parameter as shown in Figure 5a, 5b, 5c and 5d. Figure 5c and 5d only show 15 parameters as the WDTW distances are very small for larger parameters. The parameter for DTW and DDTW – warping window r is selected from the range $[0, L]$. The parameter g for WDTW and WDDTW is chosen from an uniform distribution of $U(0, 1)$ with 100 values. On the other hand, MSM monotonically increases with its parameter as shown in Figure 5e. Its parameter c is sampled from an exponential sequence in the range $[0.01, 100]$ (Bagnall et al., 2017).

For distances with two parameters, starting with LCSS, LCSS has a monotonically decreasing relationship with its parameters as shown in Figure 6a. The threshold parameter for LCSS, ε is chosen from the range $[\sigma/5, \sigma]$ where



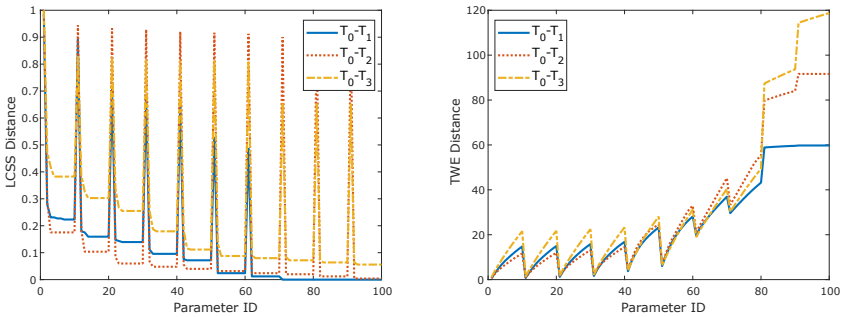
(a) DTW (b) DDTW

(c) WDTW (d) WDDTW

(e) MSM

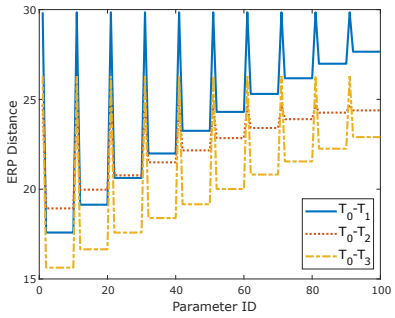
Fig. 5: Relationship between single parameter elastic distances and parameters

σ is the standard deviation of the training set. The path constraint parameter, δ is chosen from the range of $[0, L/4]$ (Bagnall et al., 2017). Figure 6b shows the monotonically increasing relationship of TWED and its parameters. The constraint parameter for TWED, v is chosen from an exponential distribution ranging from $\left[10^{-5}, 1\right]$. The penalty parameter λ is chosen uniformly from the range $[0, 0.1]$. ERP distance increases when its alignment penalty g increases and decreases when its path constraint, bandsize increases as shown in Figure 6c. Both of its parameters are chosen the same way as LCSS.



(a) LCSS

(b) TWED



(c) ERP

Fig. 6: Relationship between double parameter elastic distances and their parameters

4.2 Ordering of parameter values

Ordering the parameter values is important in improving the efficiency of the training algorithm. For instance, FastWWS starts from the largest warping window from to the smallest, as the larger window can be used as a lower bound to prune the computations for the smaller window (Tan et al., 2018). FastEE starts scanning the parameter values for DTW, DDTW, WDTW and WDDTW from the largest to the smallest. Distances with the larger parameter values can be used as a lower bound for the smaller parameter values, which is often tighter and more effective than the lower bound computed with the larger parameter. Both DTW and DDTW distances are constant at larger windows as illustrated in Figure 5a and 5b. This has the effect of pruning the computations for DTW and DDTW at the windows where they are constant. The sigmoid weight function makes WDTW and WDDTW a continuous function as illustrated in Figure 5c and 5d, preventing them from having a constant value. Since it does not satisfy property 2, the computations of WDTW cannot be pruned. For this reason, FastEE will only make use of the distances computed at a larger g as the lower bounds for smaller g. Lower bounds for WDTW are also used to speed up the process.

As MSM has a monotonically increasing relationship with its parameter `c`, FastEE starts from the smallest `c` – using distances at smaller `c` as a lower bound for larger `c`. Note that at larger `c`, MSM is constant and thus the computations can be pruned.

The *spikes* in Figure 6 correspond to the changeover of a new penalty and threshold parameter for elastic distances with two parameters. Currently, FastEE resets the scan at the *spikes*. We note that it is possible to use the distances at larger penalty value as the lower bound to a smaller penalty value but this exploration will be left for future work.

FastEE starts from the largest parameter combination (Δ, ε) for LCSS. For a fixed ε, LCSS distance stays constant for a range of Δ. Thus the computation can be pruned when the distance is constant. Smaller threshold ε means that fewer points will be a match and thus distance will be larger. Hence, LCSS distance at larger ε can be used as the lower bound at smaller ε.

For TWED, FastEE starts scanning from the smallest parameter combination (v, λ). It is interesting to see that in Figure 6b, TWED is continuous at smaller v and does not satisfy property 2. However, it remains constant at larger v. Hence FastEE is still applicable to TWED but will not yield good speed up at these smaller v.

Figure 6c shows that the ERP distance decreases with increasing `bandsize`, FastEE starts searching from the largest parameter combination and resets at every new g. Larger g corresponds to larger penalty and thus larger ERP distance. It is important to note that ERP at `bandsize` $= 0$ has the same value regardless of g which means that we can reuse this value.

4.3 The FastEE algorithms

The previous sections show the theoretical basis of our work. We are now in a better position to explain our algorithms. The FastEE algorithm applies the strategy that underlies FastWWS (Tan et al., 2018) to all the components of EE. Like FastWWS, FastEE is an exact algorithm that is capable of giving the exact best parameter (with respect to LOO evaluation on the training data) and can also be applied to a subset of parameters. To take advantage of all the properties of FastEE, we order the computations across the columns of the NNs table systematically for each distance measure, as described in Section 4.2. This allows FastEE to prune most of the computations across the columns of the table.

4.3.1 Lazy Nearest Neighbour Assessment

Our LazyAssessNN algorithm, presented in Algorithm 3, generalizes FastWWS so that it can be used for all EE's elastic distance measures. It assesses whether a pair of time series can be less than a distance d apart for a given parameter $\mathcal{P}$. LazyAssessNN assesses each of the potential nearest neighbours

Algorithm 3: LazyAssessNN($\mathcal{C}_{(Q,C)}, \mathcal{P}, d, Q, C$)

Input: $\mathcal{C}_{(Q,C)}$: cache storing the previous measure between Q and C
Input: $\mathcal{P}$: parameter
Input: d: the distance to beat
Input: Q, C: the time series to measure
Result: $E_{\mathcal{P}}(Q, C)$ if $\leqslant d$, else `pruned`

```
1  if C_(Q,C) = ∅ then C_(Q,C) ← init(Q, C)
2  switch C_(Q,C).stoppedAt do
       // LB calculated with a previous parameter P̂
3      case E_P̂ do
4          if w ∈ C_(Q,C).valid ∧ C_(Q,C).value < d then
5              return C_(Q,C).value
6          end
7      case LB_P̂ do
8          if C_(Q,C).value ⩾ d then return pruned
9      otherwise do
           // Calculate LB and E at parameter, P
           // Possible to cascade LBs for more than 1 lower bound starting
              from the lowest complexity
10         C_(Q,C) ← LB_P(Q, C)
11         if C_(Q,C).value ⩾ d then return pruned
12         C_(Q,C) ← E_P(Q, C)
13         if C_(Q,C).value ⩾ d then return pruned
14         return C_(Q,C).value
15     end
16 end
```

in a *lazy* fashion, by making the most out of all possible lower bounds. It is lazy in that it postpones calculations for as long as possible. The ordering of the elastic distance measure computations from small to large allows any value previously calculated to become a lower bound to the current parameter. A cache $\mathcal{C}$ is used to store the results from the previous parameter.

First, we have to initialise the cache if it was not initialised previously. For most elastic distance measures, the initialisation is simply creating the cache. But for DTW, DDTW and ERP the cache is initialised with the LB_Kim distance. The reason is that LB_Kim is not well-defined for the other distances. LB_Kim is the loosest and cheapest lower bound to compute. It is sufficient to filter out the obvious unpromising candidates in the training set $\mathcal{T}$. Then we test where the cache last stopped, i.e. was it computing a lower bound for the target parameter $\mathcal{P}$, was it computing a lower bound for previous parameter $\hat{\mathcal{P}}$, or was it computing a distance for previous parameter $E_{\hat{\mathcal{P}}}$. If it stopped at $E_{\hat{\mathcal{P}}}$, then we have to assess if $E_{\hat{\mathcal{P}}}$ is still valid and having a value less than d. On line 7-9, if we cannot prune with $E_{\hat{\mathcal{P}}}$, then we check if we can prune using previously computed bounds. Otherwise, we have to compute the lower bound for the target parameter and test if we can prune them. Note that we can cascade the bounds for distances with more than one bound. The bounds are ordered by their complexity which usually corresponds to their tightness (Rakthanmanon et al., 2012; Tan et al., 2018). In this work, we cascade DTW lower bounds in the order – LB_Kim(Q, C), LB_Keogh(Q, C)

Algorithm 4: FastEE($\mathcal{T}, C$)

Data: $\mathcal{T}$: training data with size N
Data: C: set of NN classifier paired with an elastic distance
Result: $\mathcal{P}^\star$: array of best parameter for each distances
Result: *bestAccuracy*: best LOO-CV accuracy for each distances

$\mathcal{P}^\star \leftarrow \emptyset$
foreach $c_i \in C$ **do**
  $\mathcal{P}^i \leftarrow \{\mathcal{P}^i_1, ..., \mathcal{P}^i_M\}$
  NNs $\leftarrow c_i$.`FastFillNNTable`$(\mathcal{T}, \mathcal{P}^i)$
  $bestNCorrect_i \leftarrow -1$
  for $\mathcal{P}^i_p \leftarrow \mathcal{P}^i_1$ **to** $\mathcal{P}^i_M$ **do**
    $nCorrect \leftarrow 0$
    foreach $Q_q \in \mathcal{T}$ **do**
      if NNs$[q][p].class \neq Q_q.class$ **then**
        $nCorrect$++
      end
    end
    if $nCorrect > bestNCorrect_i$ **then**
      $bestNCorrect_i \leftarrow nCorrect$
      $\mathcal{P}^\star_i \leftarrow \mathcal{P}^i_p$
    end
  end
  $bestAccuracy_i \leftarrow bestNCorrect_i / |\mathcal{T}|$
end

and LB_Keogh(C, Q). Reversing the role of Q and C in LB_Keogh can sometimes provide better bounds (Rakthanmanon et al., 2012). Finally if all bounds failed to prune the candidate, then we have to compute $E_\mathcal{P}$ – the elastic distance measure E at the target parameter $\mathcal{P}$.

4.3.2 FastEE *Algorithm*

Recall that the problem of learning the best parameter for an elastic distance measure, E can be re-framed into creating a $(N \times M)$ NNs table, which gives the NN of every time series in the training set for all M parameters. Such a table is depicted in Table 2 (in Section 2.3). This provides the memory complexity of our FastEE algorithm, $O(N \times M)$. Once this table is built, the best parameter value can be learned in one pass over it as described in Algorithm 4. Algorithm 5 is used to fill the table and returns the parameter value with the highest LOO-CV accuracy on the training set $\mathcal{T}$ of size N.

The core of FastEE actually depends on how efficient we can compute this table. Algorithm 5 describes how we build this table efficiently for a particular elastic distance measure. At the highest level, the algorithm builds this table for a subset $\mathcal{T}' \subseteq \mathcal{T}$ of increasing size until $\mathcal{T}' = \mathcal{T}$. For example we start by building the table for $\mathcal{T}$ comprising of only time series C_1 and C_2 and fill the table as if $\mathcal{T}$ is the entire dataset. It is trivial to see that C_1 is the nearest neighbour for C_2 and vice versa. Then a third time series C_3 is added to the set $\mathcal{T}'$ from $\mathcal{T} \setminus \mathcal{T}'$. Now we have to find the nearest neighbour for C_3 from

Algorithm 5: FastFillNNTable($\mathcal{T}, \bar{\mathcal{P}}$)

Input: $\mathcal{T}$ the set of time series
Input: $\bar{\mathcal{P}}$ ordered parameters to scan
Result: NNs$[N][M]$ the nearest neighbors table

```
1  Define LANN as LazyAssessNN
2  NNs.fillAll (_, +∞)
3  T' ← ∅
4  for q ← 2 to N do
5      Q ← T_q
6      T' ← T' ∪ {T_{q-1}}
7      foreach C ∈ T' do C_{Q,C} ← ∅
8      for p ← M down to 1 do
9          if NNs[q][p] ≠ ∅ then
               // Update table NNs[c][p]_{1≤c≤q-1}
10             for tc ← 1 to q − 1 do
11                 toBeat ← NNs[c][p].distance
12                 res ← LANN(C_{(Q,C_c)}, P̄_p, toBeat, Q, C_c)
13                 if res ≠ pruned then
14                     NNs[c][p] ← (Q, res)
15                 end
16             end
17         else
               // Check Q against previous C ∈ T'
18             foreach C ∈ T' in asc. order using C do
19                 toBeat ← NNs[q][p].distance
20                 res ← LANN(C_{(Q,C)}, P̄_p, toBeat, Q, C)
21                 if res ≠ pruned then
22                     NNs[q][p] ← (C, res)
23                 end
24                 toBeatC ← NNs[c][p].distance
25                 resC ← LANN(C_{(Q,C)}, P̄_p, toBeatC, Q, C)
26                 if resC ≠ pruned then
27                     NNs[c][p] ← (Q, resC)
28                 end
29             end
               // Propagate NN for all valid parameters
30             for p' ∈ NNs[q][p].valid do NNs[q][p'] ← NNs[q][p]
31         end
32     end
33 end
```

$\mathcal{T}' \setminus C_3 = \{C_1, C_2\}$ and check if C_3 is the nearest neighbour for both C_1 and C_2. This process is repeated until $\mathcal{T}' = \mathcal{T}$.

Algorithm 5 starts by initialising the $N \times M$, NNs table to $(\_, +\infty)$. This means that the table is empty and the distances are $+\infty$. The iteration starts from 2 in line 4 as there need to be at least 2 time series. Lines 5 to 7 are some initialisations including creating the cache associated with Q to store the results. After initialisation, we start the NN search with the set of parameters $\bar{\mathcal{P}}_p$ in the order mentioned in the previous section. Then in line 9, we check if Q already has a NN found from previous parameters. If Q already has a NN, then we only need to check if Q is the NN for the other time series in $\mathcal{T}'$. At this point, the LazyAssessNN algorithm assess if Q "beats" the previous NN

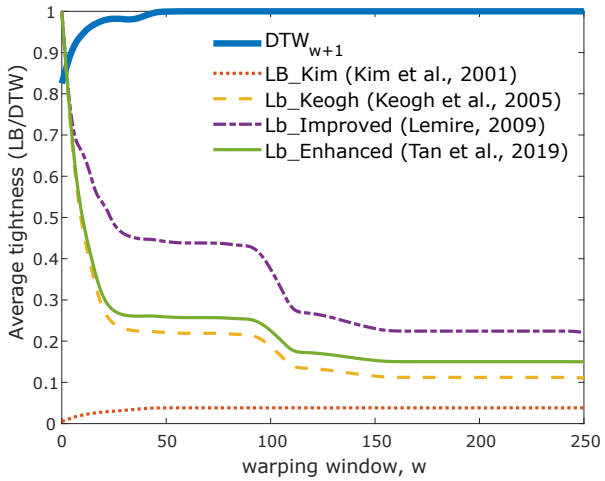


Fig. 7: Average tightness of DTW lower bounds for the `ArrowHead` dataset (better seen in color)

for each of the time series in $\mathcal{T}'$. If LazyAssessNN exits with `pruned` then it means that Q is not the NN, otherwise the NNs table has to be updated with Q as the new NN.

If we do not have the NN for Q from the previous parameter value, then we will need to analyse all $(Q, C)_{C \in \mathcal{T}'}$ and update the NNs table simultaneously for Q and C. At this stage, we already have some information stored in the cache $\mathcal{C}$ about which $C \in \mathcal{T}'$ might be a better NN candidate for Q. Note that the number of computations will be minimised if the first C is actually the NN of Q. Hence, it is important to first assess the candidate with the highest NN potential by ordering the candidates. This method has been previously studied in (Tan et al., 2017) and used in FastWWS (Tan et al., 2018).

As mentioned in (Tan et al., 2018), it is possible that $\mathcal{C}$ contains different type of lower bounds leading to distances with different magnitude. Thus the lower bounds have to be normalised. Most of the lower bounds for the elastic distance measures are of $O(L)$ complexity, so we normalise them by the number of point-wise calculations. Elastic distance measures are then being normalised by a factor of $0.8/L$ which gives higher priority to the time series where an actual elastic distance has been computed, rather than a lower bound distance. This is because the distance at the previous parameter, $E_{\hat{\mathcal{P}}}$ represents a better estimate of the distance at the current parameter, $E_{\mathcal{P}}$ than its lower bound. An example shown in Figure 7, comparing different DTW lower bounds, shows that using DTW computed from the larger window is tighter than all other existing lower bounds (Keogh and Ratanamahatana, 2005; Kim et al., 2001; Lemire, 2009; Tan et al., 2019).

Line 19 gives the distance threshold from NNs$[q][p]$ which each candidate has to beat in order to be the NN of Q. The candidate is assessed using the LazyAssessNN algorithm in Algorithm 3. Initially, this value will be ∞ as

$\text{NNs}[q][p] = \emptyset$ and a distance computation has to be done which will then be stored into $\text{NNs}[q][p]$ in line 22. Then we check if Q is the NN of each candidate $C \in \mathcal{T}'$. Finally after all $C \in \mathcal{T}'$ have been processed, $\text{NNs}[q][p]$ contains the actual NN of Q at the parameter value $\mathcal{P}_p$. This information is then propagated across all $\mathcal{P}$ where the distance is valid.

5 Empirical evaluations

This section describes the experiments that evaluate our FastEE algorithm. Our experiments were performed using all the 85 freely available benchmark UCR time series datasets (2015 version) and the original train/test split (Chen et al., 2015). We performed a search over the 100 parameters specified in Section 4.1. We conducted all of our experiments on a 16 core Xeon-E5-2667-v3 @3.20GHz machine with 16GB RAM. Our source code has been made open-source at `https://github.com/ChangWeiTan/FastEE` and the full results at `http://bit.ly/FastEE`.

5.1 Speed-up against EE

We first perform an experiment comparing FastEE to the following:

- **EE** (Lines and Bagnall, 2015): The standard implementation of the EE classifier. This naïve version is used as the baseline. We use the code from (Lines and Bagnall, 2015).
- **LBEE**: EE with lower bounds. This is the improvised version of naïve EE. It uses lower bounds for the elastic distance measures in all the NN search. NN search with lower bound has a lot of success with speeding up NN-DTW (Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012).

All methods are exact – they all learn the same best parameter, the same LOO-CV accuracy and thus the same classification accuracy. This is shown in Figure 13b which shows the classification accuracy of FastEE compared to EE. Hence our experiments are more focused on the training time.

Figure 8 compares the training time of EE (x-axis) to LBEE and FastEE. Points under the red line indicate that the method is faster than standard EE. The result shows that FastEE is faster than the standard implementation of EE with all the red points consistently under the red line. The critical difference diagram, with $\alpha = 0.05$ in Figure 9 shows that the result is significant. Although Figure 9 shows that LBEE (using lower bounds on the elastic distance measures) is significantly faster than EE, the scatter plot in Figure 8 shows that the speed-up gain is minimal and sometimes worse than EE. This is a surprising result because lower bounds have proven to be successful in speeding up many NN-DTW tasks (Keogh and Ratanamahatana, 2005; Rakthanmanon et al., 2012; Tan et al., 2017) and we expect it to have significant

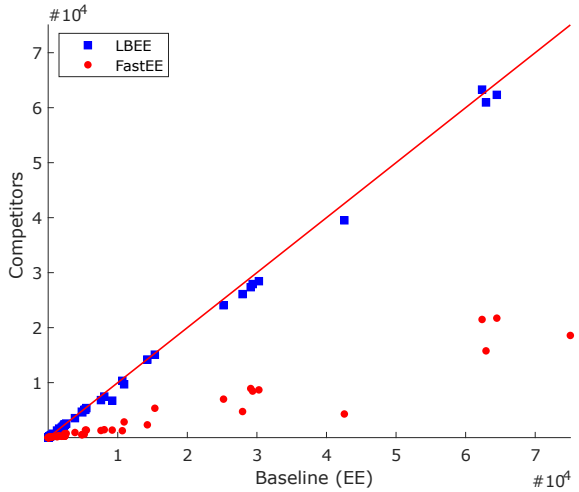


Fig. 8: Total training time on the benchmark datasets (better seen in color)

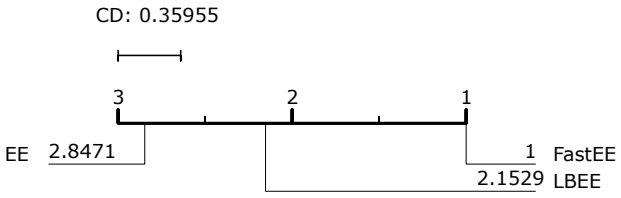


Fig. 9: Critical difference diagram comparing the training time of EE, LBEE and FastEE (Demšar, 2006). The number besides the classifiers represents the average rank of the variant from 85 benchmark time series datasets (Chen et al., 2015). Classifiers are significantly different if the difference between the average rank is larger than the critical difference

improvements. We believe that there are two main reasons for this. For simplicity, we will explain the reasons using the DTW distance. The same reasoning applies to all other elastic distance measures as well.

First is due to the tightness of the lower bounds which is highly dependent on the parameters, especially the path constraint parameter. All lower bounds used in this work are similar to LB_Keogh. They build an envelope to encapsulate the candidate time series (shown in Figure 16b), and sum the distances of all the points in the query time series that fall outside of the envelope. They are designed to ensure that if a point of a query time series that is outside of the envelope is compared to either the upper or lower envelope, the distance between them must at least contribute to the actual distance computation. For

instance, the path constraint parameter affects the envelope in the horizontal direction, i.e. larger path constraint makes the envelope wider and bigger. Similarly larger penalty or threshold parameter increases the vertical direction, making the envelope taller and bigger. If the envelope is bigger, fewer points will be outside of the envelope. As a consequence, the lower bound has a smaller distance and thus decrease in tightness.

This is illustrated in Figure 10a, which compares the training time for NN-DTW and NN-DTW with LB_Keogh at different parameters. Initially, LB_Keogh is effective at small warping windows but loses its effectiveness in pruning NN candidates as the warping window gets larger. In other words, computing lower bounds for DTW at these larger windows becomes redundant. A similar result is observed for the other elastic distance measures.

Second, using lower bounds across the columns of the NNs table is not efficient as they loses their tightness and need to be recomputed multiple times. As shown in Figure 4 and Figure 5a, DTW distance monotonically increases as the warping window decreases and is constant for a wide range of warping windows. This allows FastEE to use DTW from larger windows as the lower bound for a smaller window to skip as many computations as possible and preventing from recomputing DTW at parameters that give the same value. LBEE does not make use of this information and thus has to recompute the distances multiple times. As a consequence of these two reasons, when all the distances are combined, on average the training time of EE does not improve as shown in Figure 10b.

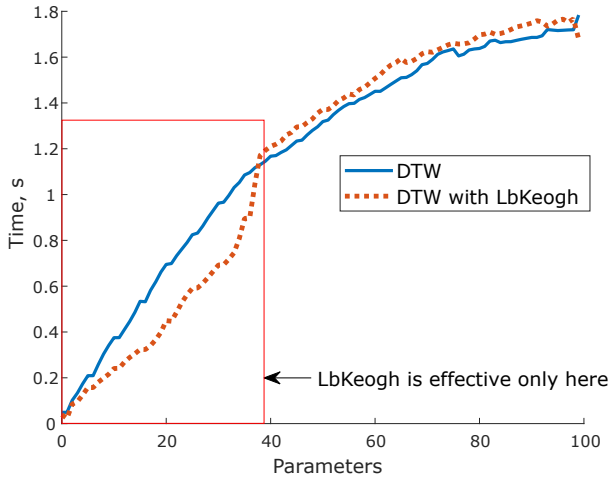
5.2 Can we do better than FastEE?

Training FastEE is at most 10 times faster than the standard EE. This brings up the question of whether we can further reduce the training time of FastEE.

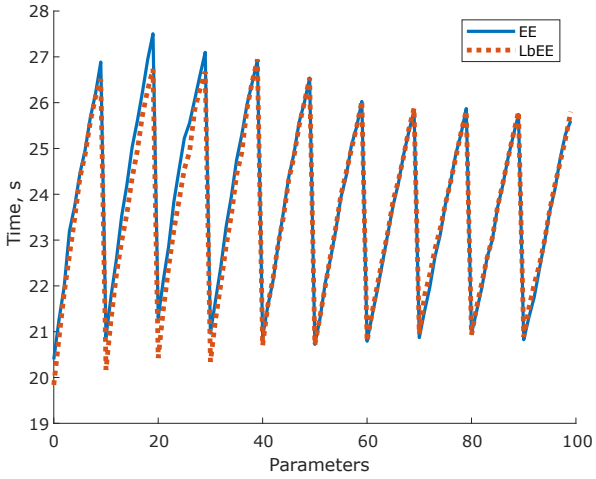
Figure 11 shows the average contribution of each elastic distance measure to EE's total training time across the UCR benchmark archive (Chen et al., 2015). TWED, MSM and WDDTW are the three distances that contribute the most to the training time of EE and LBEE. FastEE significantly reduces the training time of MSM, which leaves WDDTW, WDTW and TWED the top three for FastEE.

Recall that learning the best parameter for each elastic distance measures is typically done with LOO-CV and can be re-framed as filling up a $N \times M$ NNs table. This means that for each $C \in \mathcal{T}$, we search for its nearest neighbour from $\mathcal{T} \setminus C$. However, it is possible that we do no need the full N instances to learn the best parameter. In other words, we want to estimate the best parameter (which could be slightly different) based on a few instances $\mathcal{N}$ from the training set without compromising the classification accuracy. Hence, instead of $N \times M$, we wish to build a $\mathcal{N} \times M$ NNs table where $\mathcal{N} \ll N$. This has the consequence of speeding up the training time since less training instances are examined.

A similar method has been proposed to speed up the training time of a classifier (Flynn et al., 2019). The authors proposed a contract algorithm



(a)



(b)

Fig. 10: (a) Training time of DTW and DTW with LB_Keogh (b) EE and LBEE on the `ProximalPhalanxOutlineAgeGroup` dataset (Chen et al., 2015)

to build the Random Interval Spectral Ensemble (RISE) classifier without compromising the classification accuracy. The authors estimate the interval r in RISE, based on the remaining contract time. They used a least squares linear regression model, which models the relationship of the interval r with training time. Note that it is possible to implement a contract version of FastEE but this will be left for future work.

In this work, we propose a simple technique to approximate the LOO-CV process while not affecting the classification accuracy. Our technique – Ap-

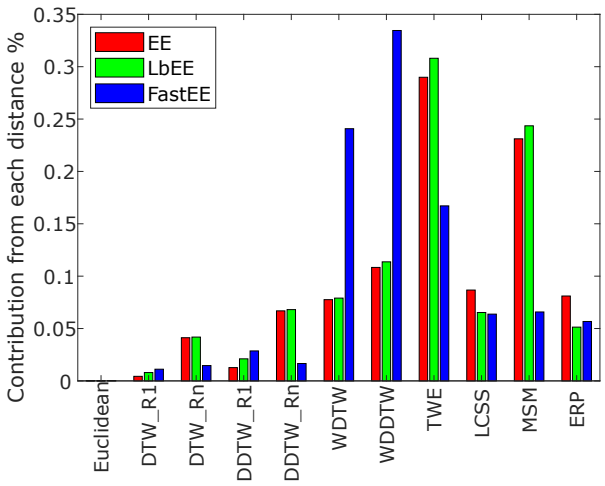
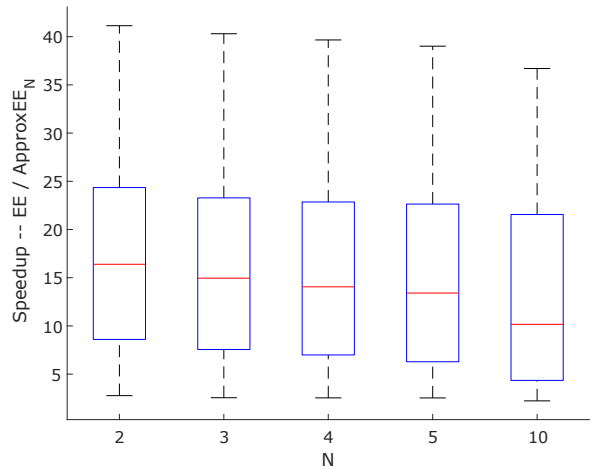


Fig. 11: Contributions from each elastic distance measures to the total training time

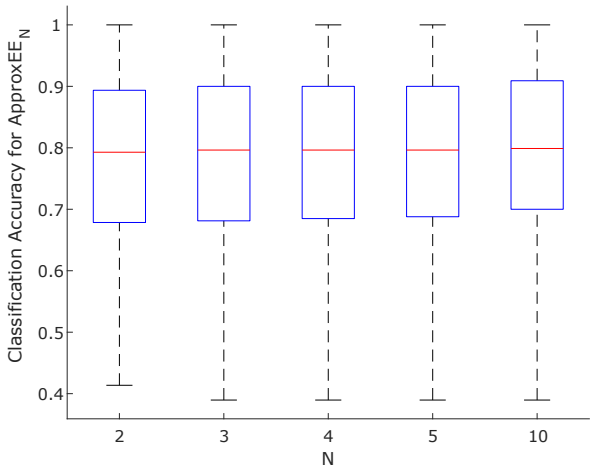
ProxEE builds a $\mathcal{N} \times M$ table but using the full training set, $\mathcal{T}$ of size N. That is, we assess all the M parameters with respect to a random subset of the full training set $\mathcal{T}' \subset \mathcal{T}$. $\mathcal{T}'$ comprises of $\mathcal{N}$ instances sampled uniformly at random from $\mathcal{T}$. For each series in $\mathcal{T}'$, we still find the nearest neighbour within the full training set $\mathcal{T}$. Thus we are keeping the original objective function, but estimating it from a sample of observations. This ensures that the nearest neighbour for each of the $\mathcal{N}$ instances is the same as exact LOO-CV – giving a better estimate of the parameters. Currently only the three distances – TWED, WDTW and WDDTW are approximated as they contribute the most to the training time of FastEE. Approximating other distances is possible but the effect will not be great because they do not contribute much to the total training time.

We report the average training time and classification accuracy of ApproxEE with $\mathcal{N} = \{2, 3, 4, 5, 10\}$ over 5 runs and we write ApproxEE$_{\mathcal{N}}$. Note that $\mathcal{N} = 1$ was not tested because it does not make sense to train with just a single instance. We note that selecting a small $\mathcal{N}$ might potentially cause class imbalance but we will show empirically over multiple runs that there is no significant reduction in the classification accuracy. However if class imbalance is an important problem or if using a metric more sensitive to imbalance (ie not accuracy), one might want to consider another sampling method closer to a stratified one for the rows.

Figure 12a compares the speed up of ApproxEE against EE across all the UCR benchmark datasets (Chen et al., 2015) for all $\mathcal{N}$. The result is expected as more time is required for training as $\mathcal{N}$ tends to N. This is indicated by the decreasing median (red line in the middle of the box plot) over multiple $\mathcal{N}$. Note that the training time across all the $\mathcal{N}$ is not significantly different. Since ApproxEE estimates the exact best parameter and the training time is



(a)



(b)

Fig. 12: (a) Speedup against EE and (b) Classification accuracy for ApproxEE across all the UCR benchmark datasets (Chen et al., 2015) for all the $\mathcal{N}$

similar for all $\mathcal{N}$, we are interested to know the effect on classification accuracy. Figure 12b compares the classification accuracy of ApproxEE across all the 85 UCR benchmark datasets (2015 version) (Chen et al., 2015) for all $\mathcal{N}$. The result shows that there is no significant difference for all $\mathcal{N}$. This suggests that a small $\mathcal{N}$ is sufficient to provide a good speed up to both FastEE and EE without compromising the classification accuracy.

Therefore, we choose $\mathcal{N} = 2$ and compare ApproxEE$_2$ to FastEE. Figure 13a compares the total training time of ApproxEE$_2$ to FastEE. It shows that ApproxEE$_2$ is always much faster than FastEE. This is expected because $\mathcal{N} \ll N$. The largest speed-up gained from ApproxEE$_2$ is 40 times on the `ElectricDevices` dataset where FastEE is only 10 times faster than EE. The critical difference diagram illustrated in Figure 14, shows that ApproxEE$_2$ is significantly faster than FastEE, LBEE and EE.

Finally we show the classification accuracy of the different EE classifiers in Figure 13b. As expected, the classification accuracy of LBEE and FastEE is exactly the same as EE because they are exact, i.e. finding the same best parameter and training accuracy. The classification accuracy of ApproxEE is not significantly different from EE, as most of the points fall very closely on the red diagonal line and shown in the critical difference diagram in Figure 15.

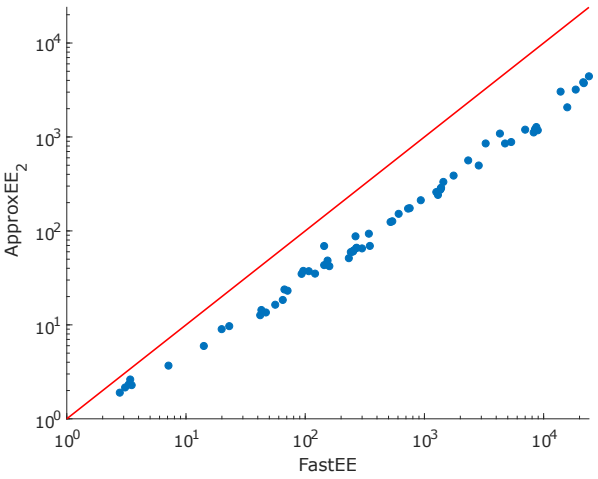
If $\mathcal{N}$ is sufficiently large, the best parameter estimated can be similar to learning with the full N time series but much faster. Then the estimated LOO-CV accuracy will be closer to the exact LOO-CV accuracy and thus does not significantly affect the final classification. However for very small $\mathcal{N}$, it is very likely that the NN classifier will not learn the best parameter. This is because the estimated LOO-CV accuracy can only be 0, 0.5 or 1 if $\mathcal{N} = 2$. This suggests that the classifiers in EE with approximate LOO-CV do not contribute much in classification accuracy.

The results from this experiment suggest that it is possible to ignore an elastic distance in EE without compromising on the classification accuracy. It is possible that EE does not need all 11 elastic distance measures to achieve such high classification accuracy. The exploration of this possibility is left for future work.

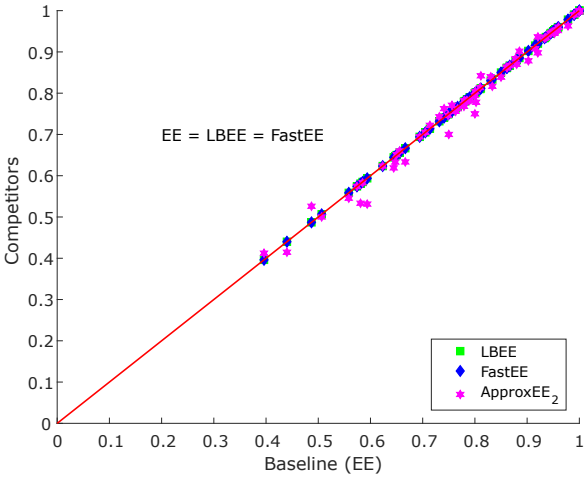
6 Conclusion

We propose the FastEE algorithm – an extension of FastWWS to the other elastic distance measures. New lower bounds have also been proposed for elastic distances without a previous bound, specifically WDTW, MSM and TWED. Lower bounds help in reducing the search space and are critical to the FastEE algorithm. Our results showed that FastEE is significantly faster than the standard implementation of EE and LBEE which uses lower bounds. To our surprise, we did not find any significant speed up in using a lower bound search to speed up the training time of EE. The main reasons are due to the inefficiency of computing the lower bound across all the parameters and that the tightness of the lower bounds degrades as the parameters change.

We also showed that it is possible to do an approximate LOO-CV process to select parameters for computationally expensive measures WDDTW, WDTW and TWED without significantly impacting accuracy. The approximation is done by learning from a subset $\mathcal{N}$ of the full training set $\mathcal{T}$ of size N. We showed that even a small $\mathcal{N}$ is sufficient to provide similar classification accuracy while being 40 times faster. Although the approximate version can



(a)



(b)

Fig. 13: Average (a) training time of FastEE and ApproxEE$_2$ and (b) classification accuracy of the different EE classifiers over 5 runs

be applied to all the other elastic distance measures, as the computation is less intensive the speed-up will be much less and we leave it for future work.

It is interesting to note that our work on Proximity Forest (PF) (Lucas et al., 2019) is more competitive than EE, which raises a question about the potential of the solutions introduced in this paper. We see the contributions in this paper to be orthogonal to the ones in PF: we introduced new lower bounds to 11 similarity measures as well as fast algorithms to cross-validate their parameters. PF showed that the use of sound metrics for time series

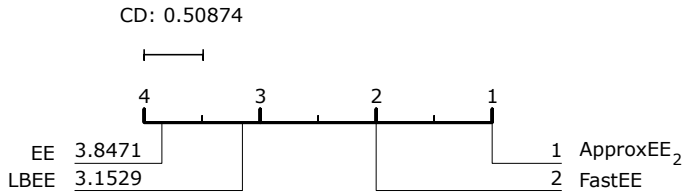


Fig. 14: Critical difference diagram comparing the training time of the different EE variants

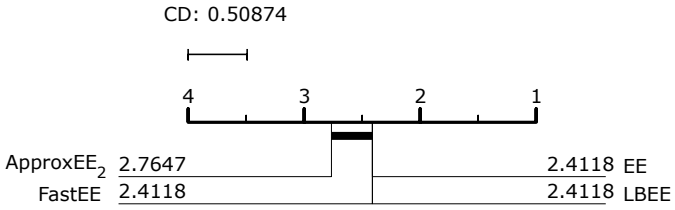


Fig. 15: Critical difference diagram comparing the classification accuracy of the different EE variants

is critical and most of its accuracy rests of the decades of research in that field. Technically, the contributions of this paper could actually be integrated within PF to make it faster and/or more accurate. For example, in PF, at every node, time series have to be compared to a set of reference time series (exemplars). For that task, our work on lower bounding EE's 11 measures would be very useful, because PF uses the same set of measures. PF also uses randomization of the parameters of the measures, as opposed to learning them with EE through cross-validation. The result is that a large number of trees are necessary in the ensemble to hep with potential bad choices of parameters. We believe that the tricks used in FastEE could be used at the node level directly to narrow down the set of parameters from which random values could be chosen.

Our future work also includes the exploration of the possibility of using fewer classifiers in EE, exploring the different alternatives for learning the parameters of EE, indexing the training set and applying a contract time for training. The results presented in this work are important because if these NN classifiers in EE can be trained in a shorter time, then HIVE-COTE (the most accurate TSC algorithm) can also be made faster and more feasible.

References

Bagnall A, Lines J (2014) An experimental evaluation of nearest neighbour time series classification. technical report# cmp-c14-01. Department of Computing Sciences, University of East Anglia, Tech Rep

Bagnall A, Lines J, Hills J, Bostrom A (2015) Time-series classification with COTE: the collective of transformation-based ensembles. IEEE Trans Knowl Data Eng 27(9):2522–2535

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 Min Knowl Discov 31(3):606–660

Boreczky JS, Rowe LA (1996) Comparison of video shot boundary detection techniques. J Electron Imaging 5(2):122–129

Chen L, Ng R (2004) On the marriage of Lp-norms and edit distance. In: Proceedings of the 30th international conference on very large databases (VLDB), pp 792–803

Chen L, Özsu MT, Oria V (2005) Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of data (SIGMOD), pp 491–502

Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. `www.cs.ucr.edu/~eamonn/time_series_data/`

Dau H, Silva D, Petitjean F, Bagnall A, Keogh E (2017) Judicious setting of dynamic time warping's window width allows more accurate classification of time series. In: Proceedings of the 2017 IEEE international conference on big data (Big Data), pp 917–922

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

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. In: Proceedings of the 34th international conference on very large data bases (VLDB), pp 1542–1552

Flynn M, Large J, Bagnall T (2019) The contract random interval spectral ensemble (c-RISE): the effect of contracting a classifier on accuracy. In: Proceedings of 2019 international conference on hybrid artificial intelligence systems (HAIS), pp 381–392

Hills J, Lines J, Baranauskas E, Mapp J, Bagnall A (2014) Classification of time series by shapelet transformation. Data Min Knowl Discov 28(4):851–881

Inglada J, Arias M, Tardy B, Hagolle O, Valero S, Morin D, Dedieu G, Sepulcre G, Bontemps S, Defourny P, Koetz B (2015) Assessment of an operational system for crop type map production using high temporal and spatial resolution satellite optical imagery. Remote Sens 7(9):12356–12379

Inglada J, Vincent A, Arias M, Marais-Sicre C (2016) Improved early crop type identification by joint use of high temporal resolution sar and optical image time series. Remote Sens 8(5):362

Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Trans Acoust, Speech, Signal Process 23(1):67–72

Jeong YS, Jeong MK, Omitaomu OA (2011) Weighted dynamic time warping for time series classification. Pattern Recogn 44(9):2231–2240

Keogh E, Ratanamahatana C (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3):358–386

Keogh EJ, Pazzani MJ (2001) Derivative dynamic time warping. In: Proceedings of the 2001 SIAM international conference on data mining (SDM), pp 1–11

Kim SW, Park S, Chu WW (2001) An index-based approach for similarity search supporting time warping in large sequence databases. In: Proceedings of the 17th international conference on data engineering (ICDE), pp 607–614

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

Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592

Lines J, Taylor S, Bagnall A (2016) HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In: Proceedings of the 16th IEEE international conference on data mining (ICDM), pp 1041–1046

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 Min Knowl Discov 33(3):607–635

Marteau PF (2009) Time warp edit distance with stiffness adjustment for time series matching. IEEE Trans Pattern Anal Mach Intell 31(2):306–318

Petitjean F, Inglada J, Gançarski P (2012) Satellite image time series analysis under time warping. IEEE Trans Geosci Remote Sens 50(8):3081–3095

Petitjean F, Forestier G, Webb GI, Nicholson AE, Chen Y, Keogh E (2014) Dynamic time warping averaging of time series allows faster and more accurate classification. In: Proceedings of the 2014 IEEE international conference on data mining (ICDM), pp 470–479

Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 262–270

Ratanamahatana C, Keogh E (2005) Three myths about DTW data mining. In: Proceedings of the 2005 SIAM international conference on data mining (SDM), pp 506–510

Ratanamahatana CA, Keogh E (2004) Making time-series classification more accurate using learned constraints. In: Proceedings of the 2004 SIAM international conference on data mining, pp 11–22

Sakoe H, Chiba S (1971) A dynamic programming approach to continuous speech recognition. In: Proceedings of the 7th international congress on acoustics, Budapest, Hungary, vol 3, pp 65–69

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust, Speech, Signal Process 26(1):43–49

Shen Y, Chen Y, Keogh E, Jin H (2018) Accelerating time series searching with large uniform scaling. In: Proceedings of the 2018 SIAM international conference on data mining (SDM), pp 234–242

Silva D, Batista G (2016) Speeding up all-pairwise dynamic time warping matrix calculation. In: Proceedings of the 2016 SIAM international conference on data mining (SDM), pp 837–845

Srikanthan S, Kumar A, Gupta R (2011) Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery. In: Proceedings of the 2nd international conference on computer and communication technology (ICCCT), pp 394–398

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

Tan CW, Webb GI, Petitjean F (2017) Indexing and classifying gigabytes of time series under time warping. In: Proceedings of the 2017 SIAM international conference on data mining (SDM), pp 282–290

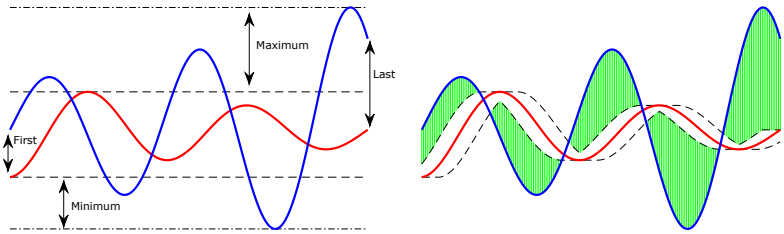
Tan CW, Herrmann M, Forestier G, Webb GI, Petitjean F (2018) Efficient search of the best warping window for dynamic time warping. In: Proceedings of the 2018 SIAM international conference on data mining (SDM), pp 225–233

Tan CW, Petitjean F, Webb GI (2019) Elastic bands across the path: a new framework and methods to lower bound DTW. In: Proceedings of the 2019 SIAM international conference on data mining (SDM), pp 522–530

Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proceedings of the 18th international conference on data engineering (ICDE), pp 673–684

Vlachos M, Hadjieleftheriou M, Gunopulos D, Keogh E (2003) Indexing multidimensional time-series with support for multiple distance measures. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), pp 216–225

Yi BK, Jagadish H, Faloutsos C (1998) Efficient retrieval of similar time sequences under time warping. In: Proceedings of the 14th international conference on data engineering (ICDE), pp 201–208



(a) (b)

Fig. 16: Illustration of (a) KIM and (b) KEOGH lower bound

A Existing lower bounds for elastic distances

A.1 DTW lower bounds

Being the most popular elastic distance, lower bound for DTW has been widely studied (Keogh and Ratanamahatana, 2005; Kim et al., 2001; Lemire, 2009; Shen et al., 2018; Yi et al., 1998). Note that DDTW is a variant of DTW, so the lower bounds for DTW are directly applicable to DDTW.

The simplest and loosest DTW lower bound is the Kim lower bound (LB_KIM) described in Equation 14 (Kim et al., 2001). LB_KIM uses the maximum differences of the maximum, minimum, first and last points of Q and C as the lower bound for DTW. With initialisation, LB_KIM can be computed very quickly with $O(1)$ time. Although a looser lower bound, it is still effective in filtering out the obvious unpromising candidates. Figure 16a illustrates this lower bound.

$$\text{LB\_KIM}(Q,C) = \max \begin{cases} |q_1 - c_1| \\ |q_L - c_L| \\ |\max(Q) - \max(C)| \\ |\min(Q) - \min(C)| \end{cases} \tag{14}$$

The Keogh lower bound (LB_KEOGH) (Keogh and Ratanamahatana, 2005) is arguably one of the most used lower bound for DTW due to its simplicity and medium-high tightness. First, it creates two envelopes encapsulating the candidate time series. The upper envelope (UE) is built by finding the maximum within a warping window r range, and the lower envelope (LE) is by finding the minimum, as shown in Equation 15.

$$\begin{aligned} UE_i &= \max(c_{i-r} : c_{i+r}) \\ LE_i &= \min(c_{i-r} : c_{i+r}) \end{aligned} \tag{15}$$

Then LB_KEOGH distance of Q and C is the Euclidean distance of all points in Q that are outside of the envelope to the envelopes UE and LE, as described in Equation 16. Figure 16b illustrates LB_KEOGH, where the sum of the length of the green lines is the LB_KEOGH distance.

$$\text{LB\_KEOGH}(Q,C) = \sqrt{\sum_{i=1}^{L} \begin{cases} (q_i - UE_i)^2 & \text{if } q_i > UE_i \\ (q_i - LE_i)^2 & \text{if } q_i < LE_i \\ 0 & \text{otherwise} \end{cases}} \tag{16}$$

There are more sophisticated lower bounds that are tighter than LB_KEOGH but has higher computation overheads. The Improved lower bound (LB_IMPROVED) (Lemire, 2009) performs LB_KEOGH in 2 passes. The first pass computes standard LB_KEOGH(Q,C) on the query and the second pass computes LB_KEOGH(Q',C) on the projection of the query Q'

onto the envelopes. The New lower bound (LB_New) (Shen et al., 2018) takes advantages of the boundary and continuity conditions for DTW warping path to create a tighter lower bound. The boundary condition requires that every warping path contains (q_1, c_1) and (q_L, c_L). The continuity condition ensures that every q_i is paired with at least one c'_j, where $j \in \{\max(1, i-r) \dots \min(L, i+r)\}$. The authors (Shen et al., 2018) sorts the points in c'_j and do a binary search if q_i is within the maximum and minimum of c'_j.

A.2 ERP lower bounds

DTW lower bounds can be adapted for the ERP distance by taking into account the ERP's penalty parameter g (Chen and Ng, 2004). Equation 17 describes LB_Kim for ERP by considering that the first and last point may be a gap, where $q'_1 = q_1$ or g, $q'_L = q_L$ or g, $Q'_{\max} = \max(Q_{\max}, g)$, $Q'_{\min} = \min(Q_{\min}, g)$. The same applies the candidate time series C.

$$\text{LB\_Kim}_{\text{ERP}}(Q, C) = \max \begin{cases} |q'_1 - c'_1| \\ |q'_L - c'_L| \\ |Q'_{\max} - C'_{\max}| \\ |Q'_{\min} - C'_{\min}| \end{cases} \tag{17}$$

Similarly to compute LB_Keogh for ERP ($\text{LB\_Keogh}_{\text{ERP}}$), the envelopes need to be adjusted for g where the maximum and minimum values have to include the g parameter. Equation 18 describes these new envelopes. Note that `bandsize` is used instead of r. Then LB_Keogh for ERP is computed exactly the same way as LB_Keogh for DTW using Equation 16 by substituting with the ERP envelopes.

$$\begin{aligned} UE'_i &= \max(g, \max(c_{i-\texttt{bandsize}} : c_{i+\texttt{bandsize}})) \\ LE'_i &= \min(g, \min(c_{i-\texttt{bandsize}} : c_{i+\texttt{bandsize}})) \end{aligned} \tag{18}$$

All the previous lower bounds were developed specifically for DTW. Thus, the authors (Chen and Ng, 2004) develop LB_ERP, a new lower bound specifically for ERP. By setting $g = 0$, LB_ERP is defined in Equation 19 as the absolute difference of the sum of both time series. The authors showed that LB_ERP has better pruning power than $\text{LB\_Keogh}_{\text{ERP}}$. Currently LB_ERP is only defined for $g = 0$ and there are no further proofs for $g \neq 0$. Therefore, we will only be using the LB_Keogh version for ERP in our work.

$$\text{LB\_ERP}(Q, C) = \left| \sum Q - \sum C \right| \tag{19}$$

A.3 LCSS lower bound

The core of LCSS is based on the length of the longest common subsequence between two time series. Then the distance is the percentage of points that are not a match – having distance larger than ε. Recall that LCSS also uses a local constraint Δ, using similar idea as LB_Keogh by constructing an envelope around the candidate time series C, a lower bound function for LCSS distance has been proposed in (Vlachos et al., 2003). The envelope for Q is constructed using ε and Δ as described in Equation 20.

$$\begin{aligned} \mathbb{UE}_i &= \max(c_{i-\Delta} : c_{i+\Delta}) + \varepsilon \\ \mathbb{LE}_i &= \min(c_{i-\Delta} : c_{i+\Delta}) + \varepsilon \end{aligned} \tag{20}$$

The sum of all $q_i \in Q$ within the envelope creates an upper bound (UB) to the longest common subsequence. Then the lower bound distance for LCSS (LB_LCSS) is $1 - \text{UB}$, defined in Equation 21, as the percentage of points that are not within the envelope.

$$\text{LB\_LCSS}(Q, C) = 1 - \frac{1}{L} \sum_{i=1}^{L} \begin{cases} 1 & \text{if } \mathbb{LE}_i \leq q_i \leq \mathbb{UE}_i \\ 0 & \text{otherwise} \end{cases} \tag{21}$$