An Ultra-Fast Time Series Distance Measure to allow Data Mining in more Complex Real-World Deployments

Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall[1], Amirali Darvishzadeh, Eamonn Keogh
University of California, Riverside
[1] *University of East Anglia, Norwich*
{sghar003, siman003}@ucr.edu, anthony.bagnall@uea.ac.uk, darvisha@cs.ucr.edu , eamonn@cs.ucr.edu

Abstract— At their core, many time series data mining algorithms reduce to reasoning about the shapes of time series subsequences. This requires an effective distance measure, and for last two decades most algorithms use Euclidean Distance or DTW as their core subroutine. We argue that these distance measures are not as robust as the community seems to believe. The undue faith in these measures perhaps derives from an overreliance on the benchmark datasets and self-selection bias. The community is simply reluctant to address more difficult domains, for which current distance measures are ill-suited. In this work, we introduce a novel distance measure *MPdist*. We show that our proposed distance measure is much more robust than current distance measures. For example, it can handle data with missing values or spurious regions. Furthermore, it allows us to successfully mine datasets that would defeat any Euclidean or DTW distance-based algorithm. Additionally, we show that our distance measure can be computed so efficiently as to allow analytics on very fast arriving streams.

***Keywords**: Time Series, Distance Measure, Matrix Profile*

1 Introduction

Time series data continues to be one of the most analyzed types of data. A recent KDnuggets poll found that 48% of analysts had analyzed time series data in the last year, second only to table data (relational data), and ahead of text, images, spatial and social network data [26]. While there is a plethora of time series data mining algorithms in the literature, including algorithms for clustering, similarity search, classification, rule-discovery and anomaly detection, at their core, many of these are algorithms that "reason" about the similarity of time series subsequences. Such reasoning requires an effective distance measure, and most algorithms use Euclidean Distance or DTW as

their core subroutine [1][3]. We argue that these distance measures are not as robust as commonly believed. The unwarranted faith in these measures derives from:

- **Optimizing to benchmarks.** The UCR Time Series Archive is doubtless a useful resource for the community [5]. However, as [8] and others have noted, the data in the archive has been contrived in several ways that often make the datasets poor proxies for real-world problems. Failure to be competitive of some of these datasets is an excellent way to screen unpromising ideas. However, being competitive on most datasets in the archive does not necessarily mean the proposed distance measure will be useful in real-world deployments.
- **Self-selection bias.** The community remains reluctant to consider difficult domains, for which current distance measures are unsuited. Consider the snippets of data shown in Fig. 1, which shows eight examples of the same insect behavior. The reader can quickly generalize from these examples as to what constitutes the targeted behavior. We will show, both Euclidean Distance and DTW will fail here.

In this work, we introduce a novel distance measure, MPdist (Matrix Profile distance). We show that MPdist is more robust than current distance measures and allows us to tackle datasets that would defeat any Euclidean or DTW distance-based algorithm.

Note that while we critique the overreliance in the UCR archive benchmarks as an indicator of the progress in time series data mining, this disparagement is not born out of "sour grapes". As we will show in Section 4, the MPdist produces highly competitive results on these benchmarks. However, beyond this, we show that our measure has properties that allow it to tackle much more complex datasets. Among the useful properties of the MPdist are:

- It allows comparisons of time series of different lengths.
- It is robust to spikes, dropouts, wandering baseline and missing values, issues that are common outside of benchmark datasets.
- While it has the invariances to amplitude and offset offered by DTW and Euclidean distance [3], it offers additional invariances, including phase invariance, order invariance, liner trend invariance and stutter invariance.
- It can be computed very efficiently, allowing great scalability.

To preview just one of these desirable features, *stutter* invariance, consider Fig. 1 which shows several examples of the feeding behavior of an Asian citrus psyllid (*Diaphorina citri*) feasting on a flying dragon citrus leaf [30].

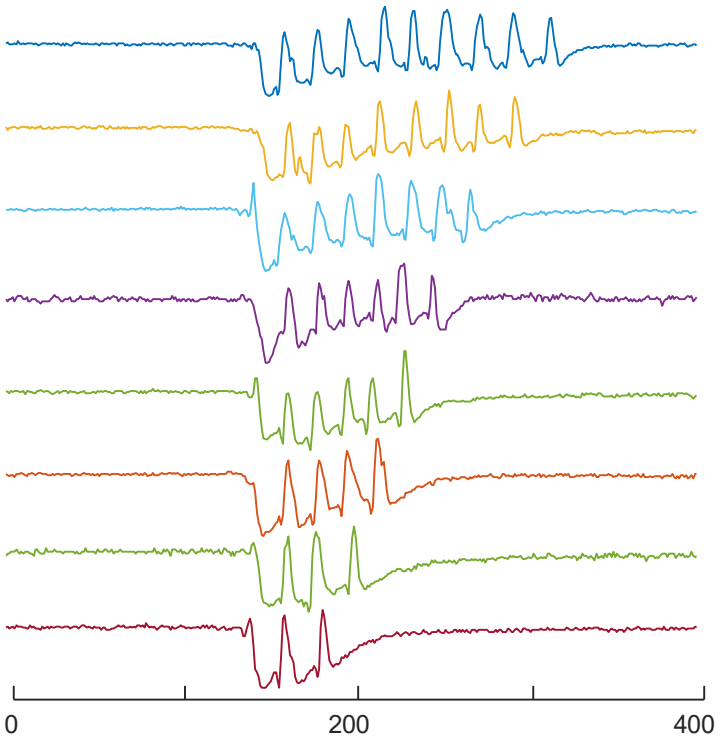


Fig. 1 Eight examples of the Phloem-Ingestion behavior of an Asian citrus psyllid, as measured by an electrical penetration graph (EPG) apparatus [30].

While this pattern is easy for a human to learn, "*from a baseline, a sudden drop, followed by two to nine peaks, as the value returns to the baseline*", this type of behavior is very difficult to model with distance-based algorithms, using current distance measures. To illustrate this, in Fig. 2.*left* we clustered three examples of Phloem-Ingestion behavior with three smoothed random walks.

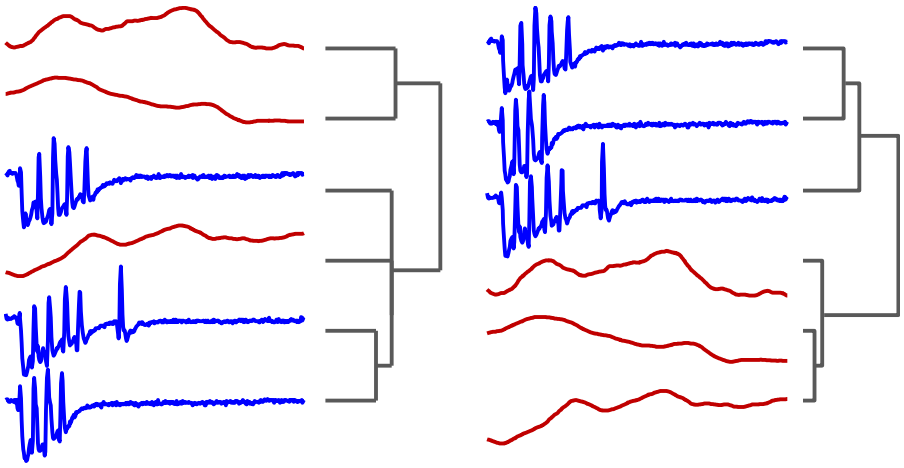


Fig. 2 Three examples of Phloem-Ingestion behavior complete-linkage clustered with three smoothed random walks (all of length 600), using (*left*) Euclidean distance, and (*right*) our proposed distance measure, the MPdist.

The results for Euclidean distance are surprisingly poor, the clustering appears essentially random. Note the problem is not solved by using DTW or other measures. While DTW can "warp" out-of-alignment peaks, it cannot warp say, three peaks to five peaks. Thus, two peaks must be left "unexplained" and occur a high distance cost, swamping any similarity that exists. Similar remarks apply to K-shape [21] and other phase invariant measures. We note in passing that near identical issues has been noted for sign language recognition. For example, [21] notes "*the number of* [sub-shapes] *contained in a sign can vary among signers due to personal signing preference*", and for manufacturing processes etc.

While our proposed measure is completely distinct from DTW, it does share some similarities with it. In particular, like DTW, the proposed MPdist:

- Subsumes Euclidean distance is a special case [22].
- On essentially all datasets, achieves accuracy greater than or equal to Euclidean distance [1]. This is an unsurprising consequence of the previous point.
- Requires just a single, easy-to-learn parameter.
- Is a *measure* but not a *metric*, this non-metricity is an unavoidable consequence of the invariances it supports. As we explain in Section 3.1, this non-metricity is highly desirable.

- While a single comparison is expense (relative to Euclidean distance), the amortized cost of subsequence search is relatively cheap, essentially the same as Euclidean distance.

The remainder of the paper is organized as follows. In Section 2 we introduce the necessary definitions and notations to understand our contributions. In Section 3, we introduce the MPdist, explain its properties and its relationship to other distance measures, and show how we can accelerate MPdist subsequence search. Section 4 offers a detailed empirical evaluation of our ideas. Finally, in Section 5 we offer conclusions and directions for future work.

2 BACKGROUND AND RELATED WORK

2.1 Definitions

Here the necessary definitions and fundamental concepts are introduced, beginning with the definition of a *Time Series*:

Definition 1: A *Time Series* $\mathbf{T} = t_1, t_2, ..., t_n$ is a sequence of n real values.

Our proposed distance measure will quantify the distance between two time series based on local subsections called subsequences:

Definition 2: A *subsequence* $\mathbf{T}_{i,L}$ is a contiguous subset of values with length L starting from position i in time series $\mathbf{T}$; the subsequence $\mathbf{T}_{i,L}$ is in form $\mathbf{T}_{i,L} = t_i, t_{i+1},\ldots, t_{i+L-1}$, where $1 \leq i \leq (n - L + 1)$ and L is a user-defined subsequence length with value in range of $3 \leq L \leq |\mathbf{T}|$.

We choose 3 as the shortest permissible value for L, because it is not meaningful to normalize time series that are shorter, and non-normalized time series are rarely used for measuring distances [22].

For our proposed algorithm, it is required to extract *all* subsequences. This is achieved using a *sliding window*:

Definition 3: *Sliding window:* All possible subsequences of a given time series $\mathbf{T}$ can be extracted by sliding a window of size L across $\mathbf{T}$. There are $(n - L + 1)$ such subsequences, which we denote as *SubseqNum*.

The time series similarity join, also known as all-pairs-similarity-search, is defined in [32]. Due to its importance in our proposed method we briefly review it here.

Intuitively, the task of the similarity join is "*Given a collection of data objects, retrieve the nearest neighbor for each object*" [32]. The similarity join set is defined on a set of all possible subsequences of a time series, referred to as the *All-Subsequences Set*:

Definition 4: An *All-Subsequences Set* $\boldsymbol{A}$ is a set of all possible subsequences of a time series $\mathbf{T}$. The subsequences are obtained from sliding a window of length L across $\mathbf{T}$. Thus,

$$\boldsymbol{A} = \{\mathbf{T}_{1,L}, \mathbf{T}_{2,L}, \ldots, \mathbf{T}_{n-L+1,L}\}.$$

At a high level, our proposed distance measure will compute the distance between two time series $\mathbf{T}_{\boldsymbol{A}}$ and $\mathbf{T}_{\boldsymbol{B}}$, by aggregating the distances between their All-subsequences sets. For this purpose, we need to find the nearest neighbor for each subsequence in $\boldsymbol{A}$ within $\boldsymbol{B}$ (and vice versa). To determine if a member of set $\boldsymbol{B}$ is the nearest neighbor of a member in set $\boldsymbol{A}$ we use *1NN-Join Function*:

Definition 5: *1NN-Join Function* is defined as the first nearest neighbor (1NN) between two subsequences $\boldsymbol{A}[i]$ and $\boldsymbol{B}[j]$. The 1NN-join function $\theta_{\mathrm{1NN}}(\boldsymbol{A}[i], \boldsymbol{B}[j])$ returns "true", if $\boldsymbol{B}[j]$ is the nearest neighbor of $\boldsymbol{A}[i]$.

The 1NN-join function is a similarity join operator, which is applied on two All-subsequences sets; as a result, we can create *AB similarity join set*:

Definition 6: *AB Similarity Join* $\boldsymbol{J_{AB}}$ is a set containing pairs of each subsequence in $\boldsymbol{A}$ with its corresponding nearest neighbor in $\boldsymbol{B}$. In which $\boldsymbol{A}$ and $\boldsymbol{B}$ are two sets of All-subsequences. $\boldsymbol{J_{AB}}$ is defined as:

$$\boldsymbol{J_{AB}} = \{\langle \boldsymbol{A}[i], \boldsymbol{B}[j] \rangle \mid \theta_{\mathrm{1NN}}(\boldsymbol{A}[i], \boldsymbol{B}[j])\}$$

The similarity join set contains tuples, with each subsequence in set $\boldsymbol{A}$ from time series $\mathbf{T}_{\boldsymbol{A}}$, and its nearest neighbor in set $\boldsymbol{B}$ from time series $\mathbf{T}_{\boldsymbol{B}}$. Note that some subsequences in $\mathbf{T}_{\boldsymbol{B}}$ may not be used as neighbors to any elements from $\mathbf{T}_{\boldsymbol{A}}$, and some subsequences in $\mathbf{T}_{\boldsymbol{B}}$ may be used more than once. This is because in general $\boldsymbol{J_{AB}} \neq \boldsymbol{J_{BA}}$.

For our proposed distance measure, we need to obtain the distance between *each* pair in the similarity join set. After obtaining the nearest neighbor of each subsequence in a set, an array which stores the Euclidean *distance* of each pair is called *Matrix Profile*:

Definition 7: *Matrix Profile* $\boldsymbol{P_{AB}}$ is an array in which the Euclidean distance between each pair in $\boldsymbol{J_{AB}}$ is stored. The length of $\boldsymbol{P_{AB}}$ is $(n - L + 1)$ or *SubseqNum*.

Without loss of generality, we assume that the two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ have the same length. Moreover, it rarely makes sense to measure the similarity of time series with significantly different lengths (not to be confused with *subsequence search*, which we show how to perform in Section 3.3). Note the matrix profile is slightly shorter than the time series that was used to create it.

Fig. 3 shows the $\boldsymbol{P}_{AB}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$. As shown, since $\mathbf{T}_A$ and $\mathbf{T}_B$ have mostly common structure, their $\boldsymbol{P}_{AB}$ has low values except for the region where sine-waves change to triangular wave, in which case there is no "explanation" from $\mathbf{T}_B$ in $\mathbf{T}_A$, Hence, there is a bump in $\boldsymbol{P}_{AB}$ indicating a high value.

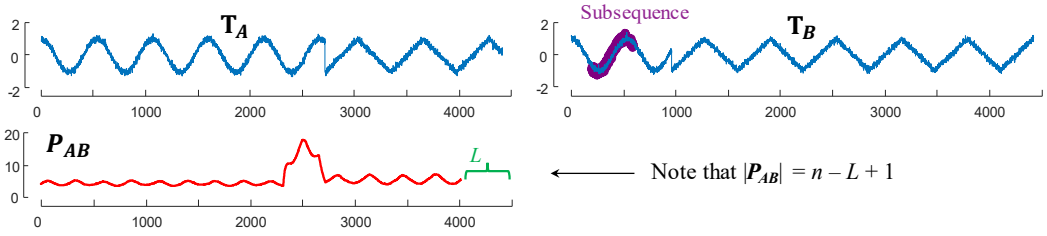


Fig. 3 *top*) Two time series $\mathbf{T}_A$ and $\mathbf{T}_B$, *bottom*) $\boldsymbol{P}_{AB}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with L = 400. Because there is no corresponding section in $\mathbf{T}_A$ from $\mathbf{T}_B$ at the point of signal change, there is a bump in $\boldsymbol{P}_{AB}$.

The time complexity to calculate $\boldsymbol{P}_{AB}$ for two equal-length time series when L is much shorter than n is $O(n^2)$ [36]. If the length of L is a significant fraction of n, then the time complexity grows to $O((n-L+1)\times n)$. In the limit, when $L = n$, this degenerates to the special case of the Euclidean distance between the two time series, which takes $O(n)$. The following notation summarizes this:

$$\text{Time complexity } P_{AB} = \begin{cases} O(n^2), & L \ll n \\ O((n-L+1)\times n), & L < n \\ O(n), & L = n \end{cases}$$

As L approaches n, the time complexity approaches linear time.

To make our distance measure between $\mathbf{T}_A$ and $\mathbf{T}_B$ symmetric, we will need to compute both $\boldsymbol{J}_{AB}$ and $\boldsymbol{J}_{BA}$; this operation we denote as the *ABBA Similarity Join*:

Definition 8: *ABBA Similarity Join* $\boldsymbol{J}_{ABBA}$ is a set containing pairs of each subsequence in $\boldsymbol{A}$ with its nearest neighbor in $\boldsymbol{B}$ and vice versa.

Note that if a subsequence in $\boldsymbol{A}$ (denoted as $\mathbf{T}_{A,i}$) is the nearest neighbor of a subsequence in $\boldsymbol{B}$ (denoted with $\mathbf{T}_{B,j}$) the reverse of that may not be true; that is, $\mathbf{T}_{B,j}$

may not be the nearest neighbor of $\mathbf{T}_{A,i}$. An array which stores all distances in ABBA similarity join set is *Join Matrix Profile*:

Definition 9: *Join Matrix Profile* $\boldsymbol{P_{ABBA}}$ is an array containing the Euclidean distance for each pair in $\boldsymbol{J_{ABBA}}$. The length of the $\boldsymbol{P_{ABBA}}$ is $2 \times (n - L) + 2$ which is twice the length of $\boldsymbol{P_{AB}}$.

The join matrix profile has distances for both similarity joins $\boldsymbol{J_{AB}}$ and $\boldsymbol{J_{BA}}$; thus, it is symmetric in terms of the order of time series. As a result, the distance calculated based on $\boldsymbol{J_{ABBA}}$ between $\mathbf{T}_A$ and $\mathbf{T}_B$ is also equal. Fig. 4 shows an illustration of the $\boldsymbol{P_{ABBA}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with the same length.

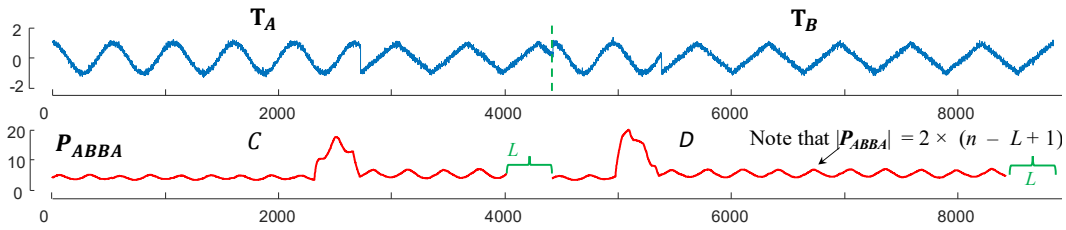


Fig. 4 *top*) The concatenation of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$, *bottom*) $\boldsymbol{P_{ABBA}}$ of two time series $\mathbf{T}_A$ and $\mathbf{T}_B$ with $L = 400$. The distance between each subsequence from $\mathbf{T}_A$ and its nearest neighbor from T_B is calculated in C and the reverse in D. There is a gap between C and D at the middle, because the length of remaining data in $\mathbf{T}_A$ is less than the subsequence length; thus, the distance cannot be calculated.

As we will show in the next section, this data structure $\boldsymbol{P_{ABBA}}$ contains all the information we need to compute the MPdist.

3 The MPdist

Intuitively, our proposed distance measure considers two time series to be similar if they share many similar subsequences, regardless of the *order* of matching subsequences. As the reader will readily appreciate, all such information is available in $\boldsymbol{P_{ABBA}}$ (Definition 9), the question then becomes is how to best exploit it.

If we based the distance on the *largest* value in $\boldsymbol{P_{ABBA}}$, the measure would be brittle to a single noisy spike or dropout that appeared in either time series. At the other extreme, if we based the distance on the *smallest* value in $\boldsymbol{P_{ABBA}}$, there would be little discrimination between most time series. This would be like a distance measure for English sentences that only looked at a *single* word in common. Since most English sentences contain '*the*' or '*a*', almost all sentences would be equidistant.

Instead of these two extremes, we propose to consider the value of the k^{th} smallest number as the reported distance. Concretely, we set the value of k to be equal to 5 percent of $2 \times n$ which is the length of concatenation of $\mathbf{T}_A$ and $\mathbf{T}_B$. While we defer a discussion of this exact value to Section 4.12, the choice of small value helps us to reduce the effect of noise and distorted values in our distance measure algorithm.

In the case when the length of subsequence is close to the length of full time series, then the length of $\boldsymbol{P_{ABBA}}$ is less than *5* percent of length of two time series. In such cases, we used the maximum value of sorted array $\boldsymbol{P_{ABBA}}$ as the distance. The following formula illustrates this:

$$MPdist = \begin{cases} k^{th} \text{ value of sorted } \boldsymbol{P_{ABBA}}, & |\boldsymbol{P_{ABBA}}| > k \\ \max(\boldsymbol{P_{ABBA}}), & |\boldsymbol{P_{ABBA}}| \leq k \end{cases}$$

Note that this implies that when the length of subsequence is equal to the length of full time series, the MPdist degenerates to the classic Euclidean distance. This is because for that setting, the $\boldsymbol{P_{ABBA}}$ has exactly two equal values, each of which is the Euclidean distance between the entire lengths of $\boldsymbol{A}$ and $\boldsymbol{B}$. Thus, the max($\boldsymbol{P_{ABBA}}$) is just the Euclidean distance.

Where appropriate, to denote the particular value of the L parameter used in given experiment, we write MPdist_L, for example in Fig. 2 we used MPdist_{20} (although any value under 60 works well).

In the following sections, we explain the MPdist properties and its relationship with the other distance measures. Then we show how we can significantly accelerate query-by-content (similarity search) under the MPdist.

3.1 On the Lack of Metric Properties for MPdist

Our MPdist distance is a *measure*, not a *metric*. In particular, it does not obey the triangular inequality. The lack of the triangular inequality property is potentially worrisome for two reasons:

- Many speedup techniques for query-by-content, clustering, anomaly detection etc., implicitly or explicitly exploit the triangular inequality to prune the search space; which becomes tenable for large datasets [3].

- Without the triangular inequality property, one can produce distance evaluations that defy human intuitions. For example, claiming that **A** and **B** are similar, and **A** and **C** are similar; but, **B** and **C** are *very* dissimilar.

To some extent, we may be comforted by noting that Dynamic Time Warping (DTW) is also not a metric; yet, it can be speeded up by many other techniques [22]. Furthermore, it has been empirically confirmed as a highly competitive measure for most time series problems in several large-scale comparisons [5].

It might be argued that DTW is *almost* a metric. This is especially true if we have a narrow warping window, which is strongly advocated due to other reasons [22]. However, we believe that there are situations/datasets that *require* a distance measure which can strongly violate the triangular inequality. To see this in practice, let us first consider an analogous problem in string matching. Consider the following common American girl names:

Lisabeth, Beth, Lisa, Maryanne, Anne, Mary

If asked to cluster these names into two groups, we would surely expect [{Lisabeth, Lisa, Beth}, {Maryanne, Mary, Anne}]. However, any distance measure that insists on the triangular inequality would have a hard time placing "Beth" and "Lisa" in the same group; since they do not share a single character with each other. Yet, both share one character with "Anne".

As Fig. 5 shows, we can create perfect analogues of such data in the time series domain.

To create a time series from the words, volunteer writers used Livescribe Echo Smartpens to transcribe the words on special dotted pattern paper. When a writer writes, an infrared camera at the tip of the smartpen detects dot-matrix pattern on the paper [6].

The dot-matrix pattern provides information about which page and where on the page the writing is occurring. This information is stored as a series of data points containing Cartesian XY coordinates and timestamp. The tip of the smartpen is equipped with a switch that is sensitive to pressure. This switch records the timestamps at which the smartpen is writing (i.e pen up and pen down). This data is utilized to split collected data points into separate pen strokes. Finally, these pen strokes are transformed into time series. Since the *X* cartesian location of pen is not discriminative between the words we just used the *Y* cartesian location of time series of names for clustering.

Clearly the Euclidean distance will find it near impossible to find any similarly between Maryanne and Anne, if we try either of the classic ideas of truncation or reinterpolation to make them the same length. Thus, as shown in as Fig. 5.*left* our collection of names cannot be correctly clustered by the Euclidean distance. However, as Fig. 5.*right* also shows, the MPdist *can* correctly cluster the data here. The property that causes the MPdist distance to violate the triangular inequality is an important one. The MPdist measure is able to *ignore* some of the data. In contrast, Euclidean distance and DTW must explain *all* the data in the sequences being compared.

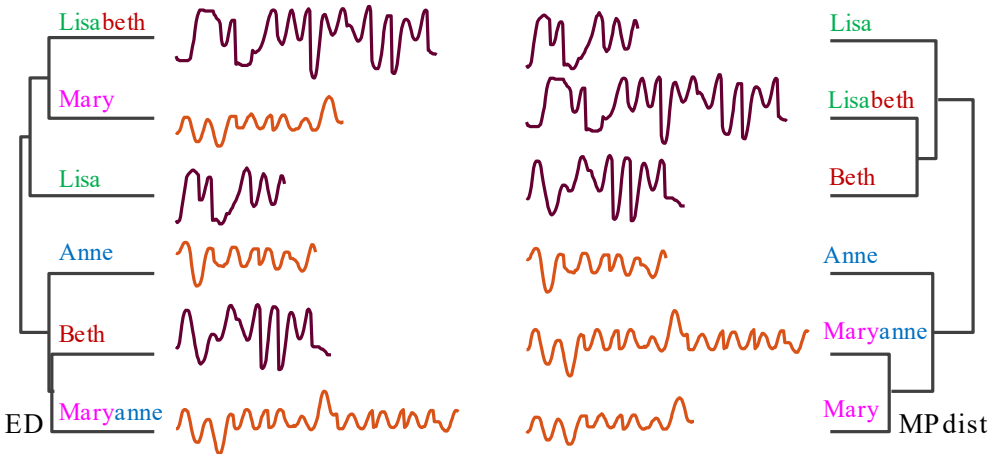


Fig. 5 A visual explanation as to why violating the triangular inequality can be useful. *top*) We created time series versions of the girl's names examples (see main body text), by capturing the Y-axis of cursive handwritten versions of the names. *bottom*) The (equalized length variant) of Euclidean distance fails to cluster such data correctly, but the Mpdist_{30} has no difficulties.

As noted above, the other cited reason is for the desirability for scalability. As we will show in Section 3.3, this is not an issue for us. In fact, we can compute the MPdist at least three orders of magnitude faster than real time data arrival rate in realistic settings. By *realistic settings*, we mean the typical sampling rates of heartbeat monitors, or accelerometers in smartphone, smartwatches etc.

What of the other possible properties of distance measures? The MPdist obeys the non-negativity or separation axiom, as $\text{MPdist}(A,B) \geq 0$. This follows from the fact that the k^{th} largest number in a sorted list of non-negative numbers must be non-negative. It also obeys symmetry, $\text{MPdist}(A,B) = \text{MPdist}(B,A)$, as it takes its values from $\boldsymbol{P}_{ABBA}$, which itself is symmetric [32][36]. However, the MPdist does not obey the identity of indiscernibles. In general $\text{MPdist}(A,B) = 0$, does not imply that $A = B$. To see this,

assume $A = B$, and now concatenate some number to the end of B, as in $B \leftarrow [B\ 3.14]$. Clearly now $A \neq B$, yet it is still the case that MPdist(A,B) = 0.

3.2 The Relationship to other Distance Measures

Having seen the MPdist, it may be useful to place it in the context of the other major distance measures, which are:

- **Euclidean Distance**: Two time series are considered similar if one is a noisy version of the other [1][3][31] [36].
- **Dynamic Time Warping**: Two time series considered are similar if, *after* non-linear adjusting the time axis, they can be made similar under Euclidean Distance [22].
- **LCSS Distance**: Two time series are considered similar if, *after* deleting some small sections from one of them, they can be made similar under the DTW Distance[1][1].
- **K-Shape**: Two time series are considered similar if, *after* some circular shift of the time axis, they can be made similar under Euclidean Distance[2] [21].
- **MPdist**: Two time series are considered similar, if they share many similar subsequences under Euclidean Distance.

This list is by no means exhaustive. Dozens of alterative measures have been introduced in the last decade [1]. However, in several rigorous comparisons the initial enthusiasm for them has cooled [1]. Many of them are perhaps best seen as variants of DTW.

The MPdist may remind the reader of Time Series Shapelets [31]. Both exploit the fact that while attempting to explain *all* the data can sometimes be futile, considering only *local* information may be more fruitful. Recall however that Shapelets are just data fragments augmented with a distance threshold. They are *not* a distance measure.

3.3 Speeding Up MPdist Search

As noted above, the time complexity of MPdist is $O(n^2)$ in the worst case. This lethargy would be a serious problem if we wish to perform MPdist similarity search (i.e. query-

[1] There are several variants of LCSS proposed (under this, and other names). This is the more general explanation of such methods.

[2] This idea is simply the *cross correlation*, *K*-Shape is an algorithm that *uses* cross correlation [21]. However, we abuse terminology a little here to be consistent with the emerging literature.

by-content) in large datasets. By analogy, DTW was introduced to the data mining community in Berndt and Clifford's famous 1994 paper [4]. However, it had almost no impact on practical applications until lower bounding search brought its amortized time complexity down from $O(n^2)$ to just $O(n)$ [22]. In this section we will show that the MPdist is amenable to a similar acceleration for *query-by-content*:

> **Problem Statement**: Given a query **Q** of length *n* and a much longer time series **T** of length *m*, we wish to create a distance vector $\text{MPdist}_{\text{vect}}$, that contains the MPdist between **Q** and $\mathbf{T}_{i:i+n}$, for all *i* in the range 1 to *m*-*n*+1.

The $\text{MPdist}_{\text{vect}}$ is shown in red in Fig. 6. The $\text{MPdist}_{\text{vect}}$ is minimized at the location of the nearest neighbor of **Q**. More generally, this distance vector is all we need to find the *k*-nearest neighbors, or to answer arbitrary range queries etc.

The brute algorithm to compute $\text{MPdist}_{\text{vect}}$ is $O(mn^2)$, which is clearly untenable. However, as we shall now show, we can compute it in just $O(m \times SubseqNum)$ time.

We begin by obtaining the All-subsequences set of **Q** (Definition 4), and then calculating the distance between each individual subsequence in the set to every subsequence in **T**. The MASS algorithm allows us to do this very efficiently [19]. As Fig. 6 shows, this gives us *SubseqNum* Euclidean *distance profiles,* the j^{th} of which we denote as d_j.

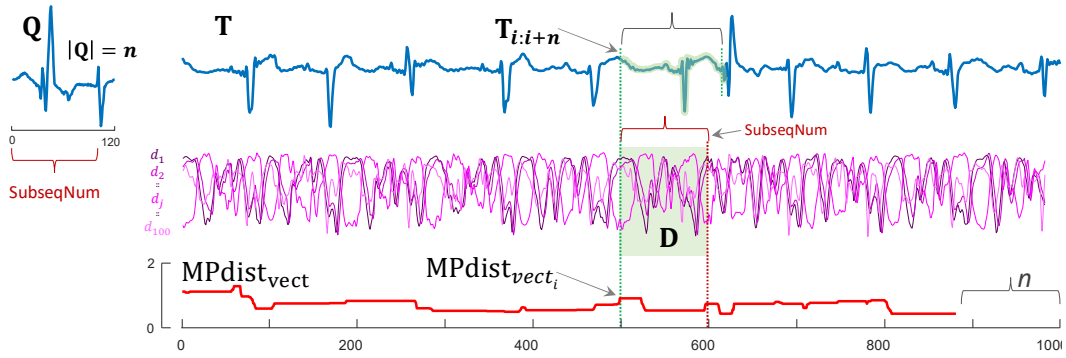


Fig. 6 A query **Q** with a long time series **T** to search. We begin by creating *SubseqNum* Euclidean distance profiles.

Fig. 6 highlights an arbitrary region of **T** beginning at $\mathbf{T}_i$ with a length of *n* and the corresponding region of the distance profiles, each with the length of *SubseqNum*. For notational clarity, we envision this *SubseqNum* × *SubseqNum* region as a matrix that we call **D**.

Perhaps surprisingly, **D** contains all the information needed to compute MPdist(**Q**,$\mathbf{T}_{i:i+n}$). The key observation is that using *just* **D** we can calculate both $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$; therefore, obtaining $\mathbf{P}_{ABBA}$.

The steps to calculate $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ are as follows:

- $\mathbf{P}_{AB}$: The row-wise minimum of **D** corresponds to $\mathbf{P}_{AB}$. Recall that the first value in $\mathbf{P}_{AB}$ is the minimum distance between the first subsequence in $\mathbf{T}_A$ compared to all the subsequences in $\mathbf{T}_B$, which is the minimum of the first row of **D**. In the same manner, the remaining values of $\mathbf{P}_{AB}$ can be obtained as the minimum of all the other rows in **D**.
- $\mathbf{P}_{BA}$: The column-wise minimum of **D** corresponds to $\mathbf{P}_{BA}$. $\mathbf{P}_{BA}$ is simply the minimum distance between the subsequences in $\mathbf{T}_B$ compared to all the subsequences in $\mathbf{T}_A$, which is just the column-wise minimum of **D**.

Thus, by concatenating $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ we can obtain $\mathbf{P}_{ABBA}$ and therefore the MPdist. Fig. 7 illustrates this.

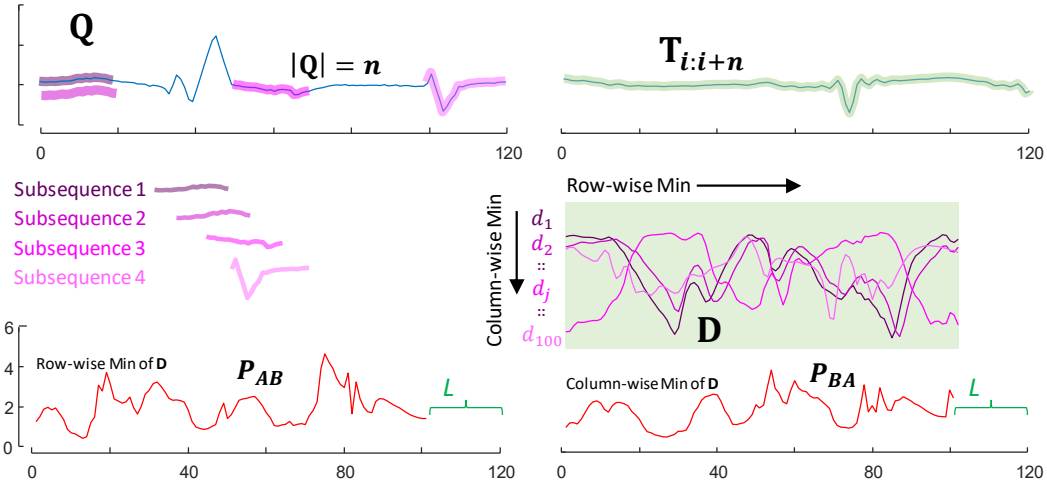


Fig. 7 Exploiting **D** to produce the $\mathbf{P}_{ABBA}$. See also **Fig. 6**.

The time complexity for calculating a single Euclidean distance profile is $O(m\log m)$ [18]; so, the time complexity for calculating the distance profile for all subsequences is $O(SubseqNum \times m\log m)$.

At first blush, this method of computing MPdist seems to have gained us nothing. The time complexity for recreating $\boldsymbol{P_{ABBA}}$ in the region **D** is $O(SubseqNum^2)$; so to compute this for all sliding **D**'s in **T** would be $O(SubseqNum \times m\log m +$

$m \times SubseqNum^2$). However, we can optimize the algorithm to have an amortized time complexity of just $O(SubseqNum \times m)$.

The key observation to allow this dramatic speed up is to realize that as we slide our query from location $\mathbf{T}_i$ to $\mathbf{T}_{i+1}$ to produce a new **D**, we do not need to recalculate everything from scratch, but just update a handful of values. As we slide our query one step, some points will *ingress* into **D** at the right and some points will *egress* from **D** at the left.

Concretely, for each step to the right, we have *SubseqNum* new points in distance profiles added to the **D** and the same number removed from the **D**. Let us consider how these incremental updates change $\mathbf{P}_{ABBA}$, and how we can address them:

- **Ingress**: For $\mathbf{P}_{BA}$ we can find the column-wise minimum of last column of **D** for new arrival point in $O(SubsecNum)$. In addition, for $\mathbf{P}_{AB}$, recall that we must find the minimum of each row in **D**. This problem is equivalent to finding the classic sliding window minimum [25] which can be solved in $O(1)$ for a single row, and in $O(SubsecNum)$ for all rows.
- **Egress**: We can easily remove the first point from $\mathbf{P}_{AB}$ and $\mathbf{P}_{BA}$ in time complexity $O(1)$.

Thus, the amortized time complexity of obtaining matrix profile for new arrival point is $O(SubsecNum)$. As a result, the amortized time complexity for calculating $\text{MPdist}_{\text{vect}}$ between **T** and **Q** is $O(SubsecNum \times m)$.

It may appear that we have a *space* overhead of $O(m \times SubseqNum)$. However, for a long time series of length say one billion, we can simply perform the above in a piecewise fashion, with pieces of length say 10,000. Thus, the space overhead is $O(1)$.

Finally, while we believe that both MPdist and its subsequence search acceleration are simple enough to not warrant the space required for pseudocode. However, we have placed detailed pseudocode (and actual code) at [17].

3.4 Generalizing to Multi-Dimensional Time Series

In multidimensional time series classification, many dimensions may be irrelevant or noisy. These irrelevant or noisy dimensions confuse classification algorithms: as a result, the algorithm gets lower accuracy. In previous attempts to mitigate this issue

relevant dimension selection has been employed to improve classification accuracy in the face of spurious data [1]. Most of these methods tries to find the most relevant dimensions in preprocessing process. However, it may be the case that the set of relevant dimensions changes a query time is ways that are hard to predict in advance. For example, to detect gait, we can normally exploit information from both accelerometers in a smartwatch and accelerometers in smartshoes. However, suppose at query time the user happens to be holding a heavy suitcase. This will reduce the utility of smartwatch data, perhaps to zero.

What is needed, is a system that can dynamically choose the relevant dimensions at query time. Essentially, we want to "tell" the algorithm to use any dimensions that have high similarity to the matching dimensions in one of the training exemplars, otherwise ignore that dimension. This is the idea behind a recent work [14]. However, that proposal requires a lot of training data to build the model that allows the real-time choice of which dimensions to trust. Can we do something similar, with no model building? We believe that the MPdist can offer such a possibility. Our idea is simple. We can concatenate all dimensions into a single dimension, with NaNs in-between to mark the transitions between dimensions. We then simply compute the MPdist between these long meta time series. The intuition is that if a dimension has become irrelevant (for example, because a sensor failed, or its position/orientation changed unexpectedly), then its shape will be so dissimilar that the MPdist will not match any sections of it, instead, it will map more sections of the dimensions that *are* similar.

Our proposed method is robust to present of noisy or irrelevant dimensions and it avoids the need for explicit preprocessing to removing them. In the proposed algorithm we calculate the join matrix profile ($\mathbf{P}_{ABBA}$) of each corresponding dimension between two samples. Then we concatenate $\mathbf{P}_{ABBA}$ of all the dimensions and sort them, by considering only the maximum of the lowest 5% of the values we report distance between two multidimensional samples. This distance removes internally all the noisy and irrelevant data since the noisy and irrelevant data will not show up among the smallest values of $\mathbf{P}_{ABBA}$*'s*.

To show demonstrate our idea we created some synthetic data and perform an experiment. We create a two-dimensional dataset which includes two classes. The sample data for classes is shown in Fig. 8. *left*. We add some random noise and warping

[8][8] to the data to create a train data with the size of forty and test data with the size of ten. To see how robust the proposed method to spurious data, we used random walk as an additional dimensions to the original data. We add zero to eight irrelevant dimensions and compare classification accuracy with DTW and ED. Since dimensions are created by random data, we repeat the process five times and report the average of accuracy. The result in Fig. 8. *right* shows as we increase the number of spurious dimensions, both DTW and ED will decrease in accuracy, but in contrast MPdist result almost constant.

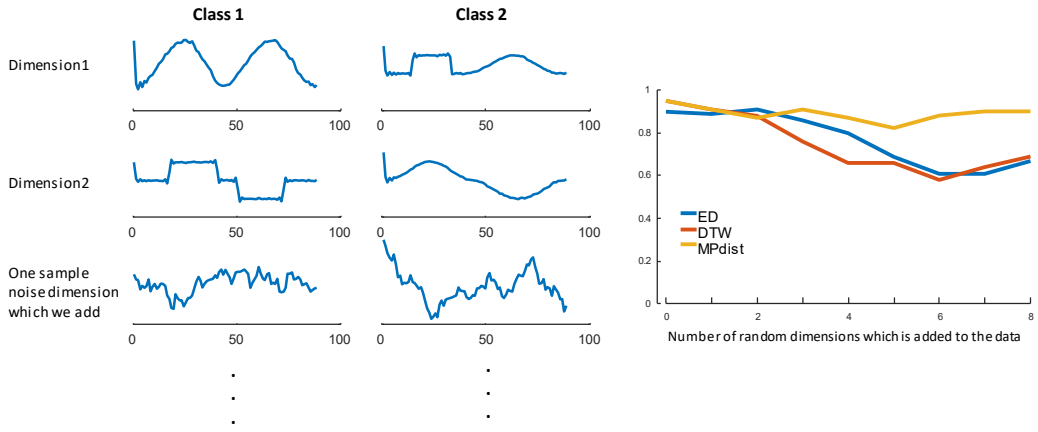


Fig. 8 *left*) Some exemplars from dataset, the original dataset has two dimensions and two classes. We show also one irrelevant dimension which we add it to the dataset. *right*) The accuracy of DTW, ED and MPdist by adding zero to eight noisy dimensions to the dataset.

To be clear, we regard this experiment as merely a proof of concept. It is clear that we could at least contrive datasets for which it would not be so robust. However, we present this result as being suggestive of the flexibility of MPdist in supporting novel research directions.

4 Experimental Evaluation

To ensure that our experiments are reproducible, we have built a website which contains all data/code/raw spreadsheets for the results [17]. This commitment to reproducibility extends to all the examples the *previous* sections. All experiments were run on a Dell XPS 8900, with Intel Corei7 CPU and 32Gb of RAM.

We begin by considering the utility of our approach for classification. When we first developed the MPdist, there were only 85 datasets in the UCR archive [5]. Since then, the archive has expanded to 128 datasets [7], including many datasets that are claimed

to be more reflective of the real world. We will present our results on both versions of the archive independently. In the 85-dataset case, it will allow direct comparisons to hundreds of *previously* published papers. In the 128-dataset case, it will allow direct comparisons to many *future* papers that will inevitably be published using this resource.

4.1 Classification Accuracy: The Eighty-Five Dataset Case

We consider all 85 datasets in the UCR archive 2015 [5], the most used benchmark in time series data mining. As we noted above, the archive has recently been criticized for being unrepresentative of real-world problems [8]. Nevertheless, we can still regard performing well on these datasets as a *necessary*, if not *sufficient* condition for introducing a new distance measure.

We learn MPdist's sole parameter on the training data, and then use the training data with this hardcoded parameter to classify the testing set. We use the original train/test splits suggested in the archive. The complete results are archived at [17]. Below we visually summarize the results with a Texas Sharpshooter plot. This is emerging as a standard way to compare time series distance measures over many datasets. Such plots help guard against the following fallacy, which appears common in the literature. Suppose a proposed algorithm produces slightly better results on say one-third of the datasets tested. The proposer could claim that the new algorithm had merit, at least for those datasets. However, it may be that for a different shuffle of the data into train/test partitions, the proposed algorithm might have been better on a *different* one-third of the datasets! A better interpretation of such a proposed algorithm would be that it is, on average, quite similar to, but a little weaker than the strawman it is being compare to. However, random fluctuations in the different folds mean it will win sometimes, but not in a way that can be predicted in advance, and therefore exploited for real applications.

To create the Sharpshooter plot, we compute two versions of *gain*:

$$gain = \frac{Accuracy\ of\ MPdist}{Accuracy\ of\ Eucldean\ Distance}$$

In particular, we compute both the *expected* gain based on the training data and the *actual* gain based on the testing data. We plot a point on a scatterplot for each dataset using the *expected* gain as the x-axis value and *actual* gain as the y-axis value (note that the "gain" can be negative).

The resulting plot is essentially a real-valued version of a contingency table, and we have labeled the four regions with the four familiar labels. Ideally, we would like to have many points in the TP region. Such points represent cases where we predicted we could do better after seeing only the training data, and we *did* do better. Only points in FP are problematic for us. They represent datasets in which we believed we could do better but actually did worse. Fig. 9 shows the results.

Note that both MPdist and Euclidean Distance are using identical train/test splits, identical classification algorithms (1-NN), and evaluated in the same way (leave-one-out). Thus, all differences can be attributed to *just* the utility of the proposed distance measure.

These results strongly support the claimed utility of MPdist. Out of the 85 datasets, 63 of them fall into the TP region. Many of them are *deep* into the TP region, denoting large improvements.

A handful of points fall into the FP region. In every case, it is because the training set is too small to allow us to robustly learn the single parameter L. However, this is easy to mitigate with a little effort. To see this, consider the worst offender in Fig. 9, the *BeetleFly* dataset. Here, our simple algorithm to learn the single parameter L gave us $L = 185$. It had an expected accuracy gain of 1.81, but an actual "gain" of 0.93. If we learn parameter using the slightly more computationally expensive resampling approach of [8], we instead learn parameter $L = 72$. It has an expected accuracy gain of 1, and an actual "gain" of 1, a significantly better meta-prediction, thus is *BeetleFly* no longer a FP case.

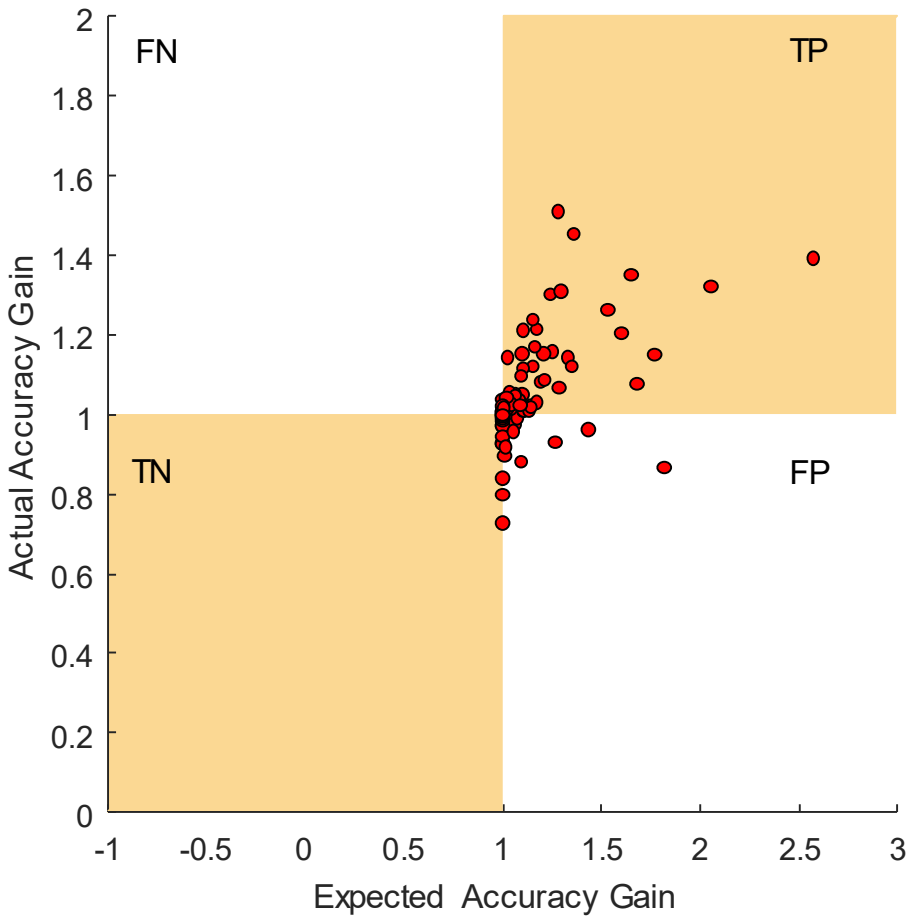


Fig. 9 Expected accuracy gain for MPdist over Euclidean Distance calculated on training data versus actual accuracy gain on testing data, over all 85 datasets in the UCR archive 2015.

We did not repeat this better way to learn MPdist's parameter for the other datasets. The reason being, we wanted to have a *deterministically* reproducible lower bound for our algorithm's performance.

4.2 Classification Accuracy: The One Hundred and Twenty-Eight Dataset Case

We also considered all 128 datasets in the new 2018 UCR archive [7]. As we noted above, the original archive has recently been criticized for being unrepresentative of real-world problems [8], this new version has crowdsourced the community to provide data that more closely reflects real word problem.

In Fig. 10 we show a Texas Sharpshooter plot for all 128 datasets. Because a handful of datasets had such a dramatic *gain* and changed the scale, in Fig. 11 we also show a Texas Sharpshooter plot but with the actual gains axis set to [0 2] and the expected gain axis set to [-1 to 3], making this plot visually comparable to Fig. 9.

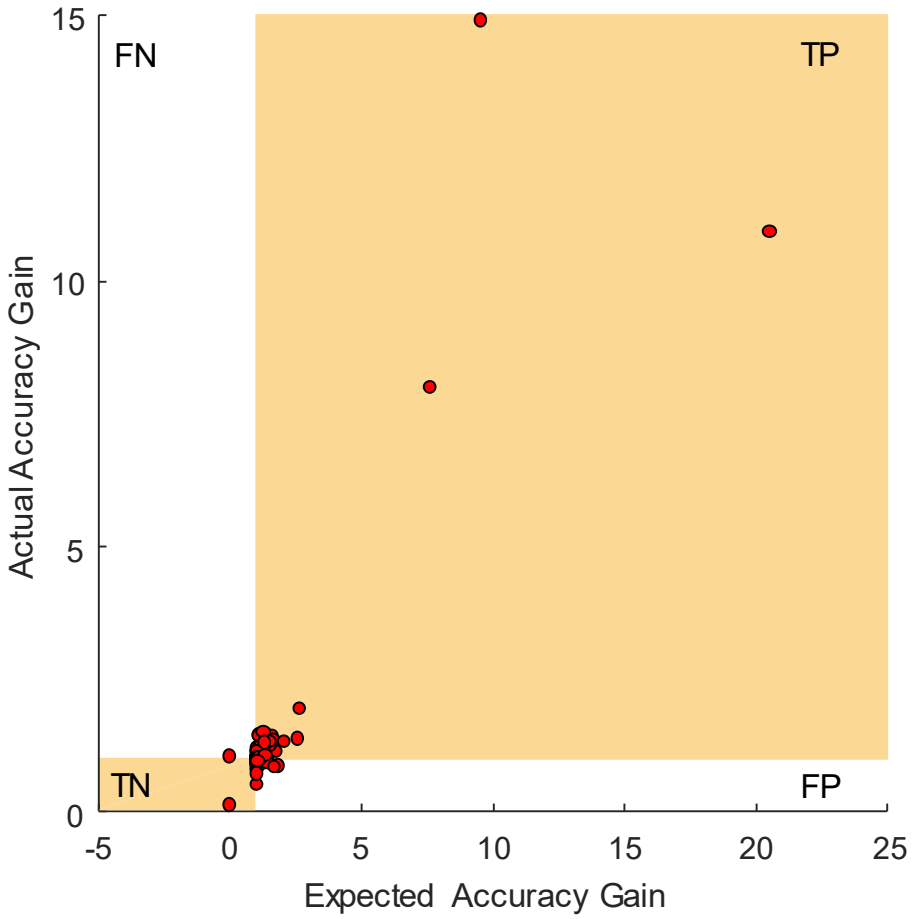


Fig. 10 Expected accuracy gain for MPdist over Euclidean Distance calculated on training data versus actual accuracy gain on testing data, over all 128 datasets in the UCR archive 2018.

The three datapoints in Fig. 10 that are far from the others belong to pigCVP, pigArtPressure and pigAirwayPressure datasets. To better show the scatter of data we removed these datasets and replot the result for remaining dataset in Fig. 11.

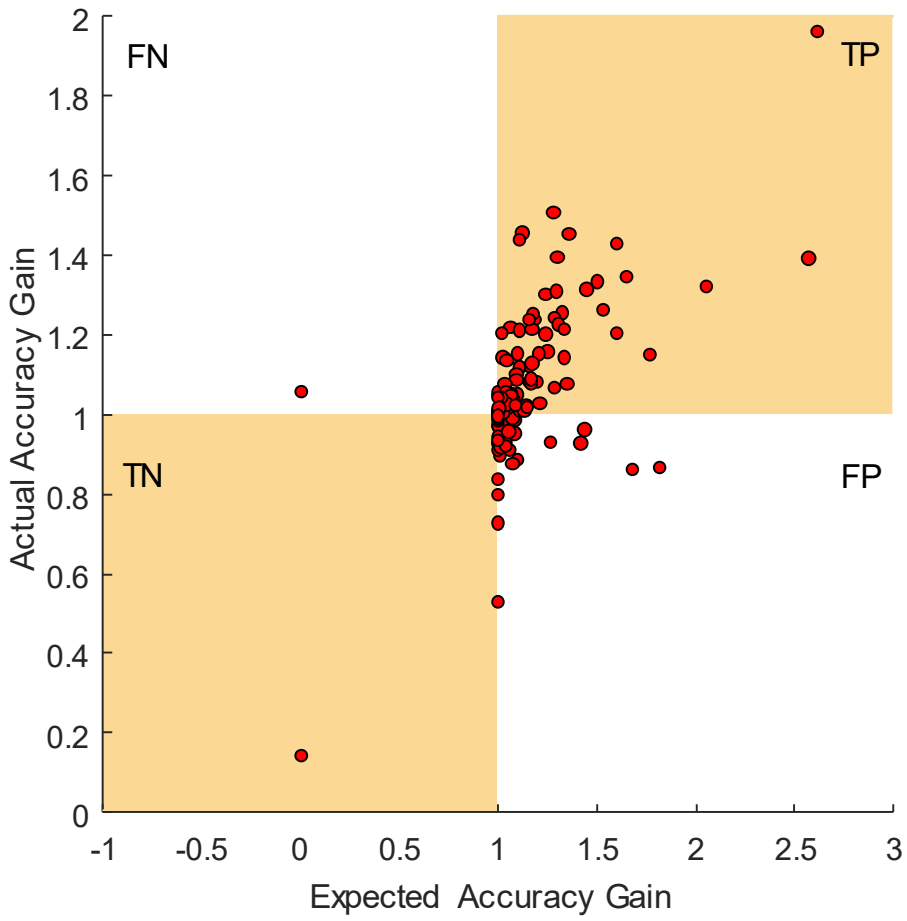


Fig. 11 Expected accuracy gain for MPdist over Euclidean Distance calculated on training data versus actual accuracy gain on testing data, over all 125 datasets in the UCR archive 2018 after removing pigCVP, pigArtPressure and pigAirwayPressure to have clearer figure.

Once again, these results strongly support the claimed utility of MPdist. Of the 128 datasets, 88 of them fall into the TP region. Of the 34 that fall in into FP, most of them are very close to the origin. Many of MPdist's wins are *deep* into the TP region, denoting large improvements. For example, for BirdChicken, ECGFiveDays, CBF, FaceFour, FacesUCR, Fish, FreezerSmallTrain, GesturePebbleZ1, GunPointAgeSpan, InsectEPGRegularTrain, PigArtPressure, PigCVP, Plane, ShakeGestureWiimoteZ, ToeSegmentation1, Trace, TwoLeadECG and PigAirwayPressure the holdout error-rate is at least halved.

Finally, we also compare with cDTW (instead of ED) as the benchmark for the Texas plot. We show that in Fig. 15, out of the 128 datasets, 37 of them fall into the TP region

and 17 of them into FP, again with most of the latter close to the origin. Similar to ED plot, due to large improvement by a handful of datasets, that obscure the results, we have removed the "pig" datasets and plot the remaining results for comparison of MPdist over cDTW on the Fig. 12.

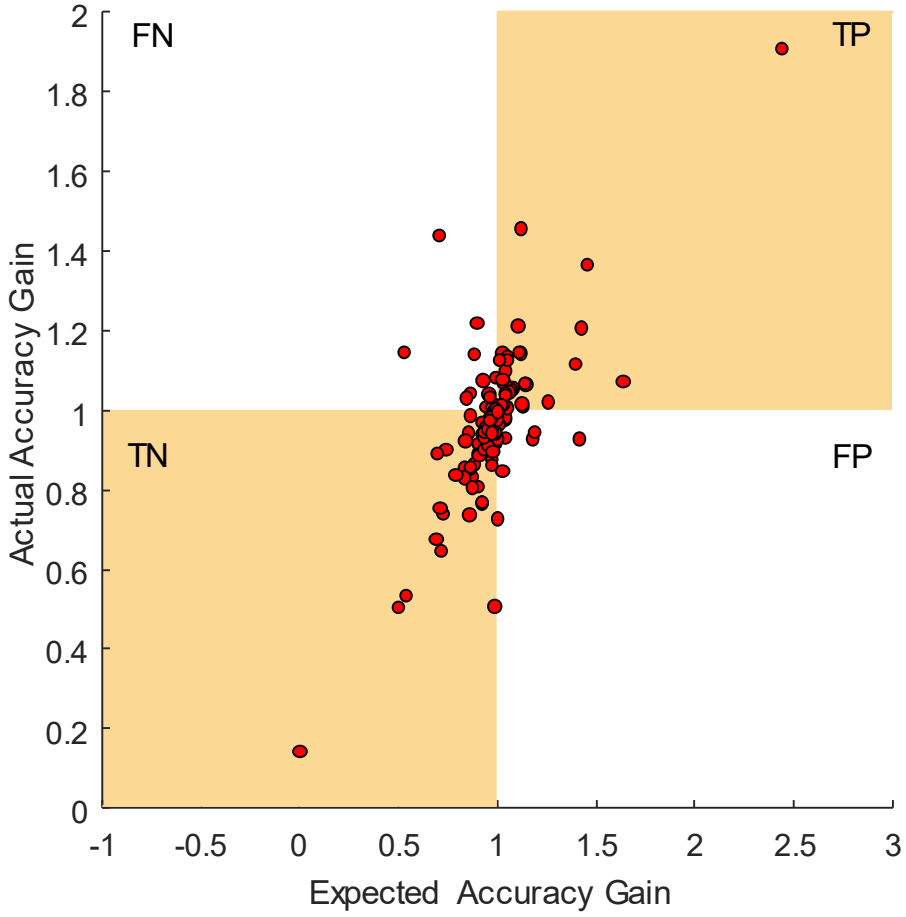


Fig. 12 Expected accuracy gain for MPdist over DTW calculated on training data versus actual accuracy gain on testing data, over all 125 datasets after removing pigCVP, pigArtPressure and pigAirwayPressure to better show the scatter of results in the UCR archive 2018.

4.3 Statistical Comparison to Benchmarks

The previous section showed an initiative and visual comparison of MPdist with the Euclidean distance measure. Here we consider more rival methods and consider statistical measures of difference.

Traditionally, 1-NN classifiers with Euclidean Distance (ED) and full window Dynamic Time Warping (DTW) have been used as benchmarks for Time series classification (TSC). Experimental evidence [1] has shown that the rotation forest classifier [24] and

1-NN dynamic time warping with warping window set through cross validation [1] [8] are much stronger benchmarks. For comparing multiple classifiers on multiple data sets, we follow the recommendation of Demsar [9] and compare classifiers based on average ranks rather than errors. This nonparametric approach removes the possible excessive influence of outliers on overall performance. For each dataset, the algorithm with lowest error gets rank 1, the algorithm with next lowest gets rank 2 etc. In case of tie, the average rank is used (e.g. if two algorithms have equal lowest error, they are assigned a rank of 1.5). To compare whether there is a significant difference in overall rank we use the Friedmann test. If this indicates there is some difference, we perform a post-hoc pairwise Nemenyi test to discover where the differences lie. Fig. 13 shows the average ranks of the four benchmarks and MPdist in a critical difference diagram. The average rank of each classifier is presented on the horizontal axis. Thus, Rotation Fortest (RotF) has the lowest (best) average rank and 1-NN Euclidean distance the highest (worst). The Friedmann test indicates there is an overall significant difference between the five classifiers. The horizontal solid bars represent cliques, i.e. groups of classifiers within which there is no significant difference as determined by the pairwise Nemenyi test.

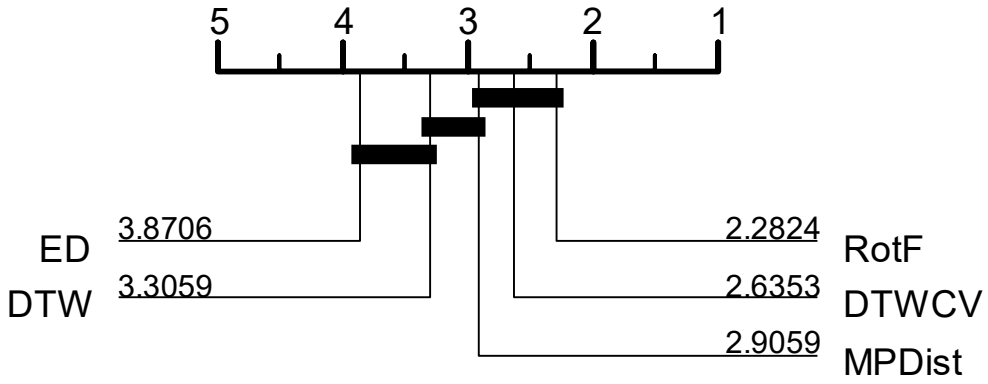


Fig. 13 Critical difference for MPdist against four benchmark classifiers: 1-NN with Euclidean Distance (ED); 1-NN with full window DTW (DTW); 1-NN with DTW window set through cross-validation DTW (CVDTW); and rotation forest with 50 trees (RotF).

There are three distinct cliques: ED and full window DTW; DTW and MPDist; and MPDist, DTWCV and RotF. Hence, we conclude that MPDist is significantly better than ED and not significantly worse than the other three benchmarks.

While these results bode well for our approach, they are pessimistic, and understate the utility of MPdist. As we noted in the previous section, these results reflect only the simple naïve method for setting MPdist parameter. Using the slightly more computationally expensive resampling approach of [8], would have further improved our results on the datasets with a small training split. More importantly, as we note above, and echoing comments in [13], the datasets UCR archive are poor proxies for real-world deployments of time series classification. We demonstrate this with the following two case studies.

4.4 A Case Study in EPG data

"Leafhopper" is the informal name for any species of insect from the family *Cicadellidae* [16]. They are plant feeders that suck sap from plants. This feeding behavior means that they can transmit plant pathogens, such as viruses, phytoplasmas and bacteria. The cost of leafhopper damage to agriculture worldwide is estimated to be in the tens of billions per annum, and this number is projected to increase because of climate change [2]. Given the economic impact of these insects, there is a large community that studies them. This is a difficult task. The 20,000 members of *Cicadellidae* range in size from 20 to just 1mm. In order to effectively study them at "big data" scales, entomologists use an Electrical Penetration Graph (EPG) apparatus to obtain time series data of their behavior. This can be done in parallel, for example Dr. Kerry Mauck of UCR's Entomology Department typically records eight insects at a time, almost every day of the year [18].

For reasons hinted at in Fig. 1 and Fig. 2, we believe that the MPdist can be particularly effective for working with such time series. One basic tool we are building to support research in this area is a query-by-example tool. It allows a researcher to ask, "*does this pattern occur in my data*?". We demonstrate this with the following example. In Fig. 14.*left* we show an example of phloem sap ingestion behavior of tea green leafhopper (*Empoasca vitis*) a leafhopper that specializes in grapevine and kiwi fruit [16].

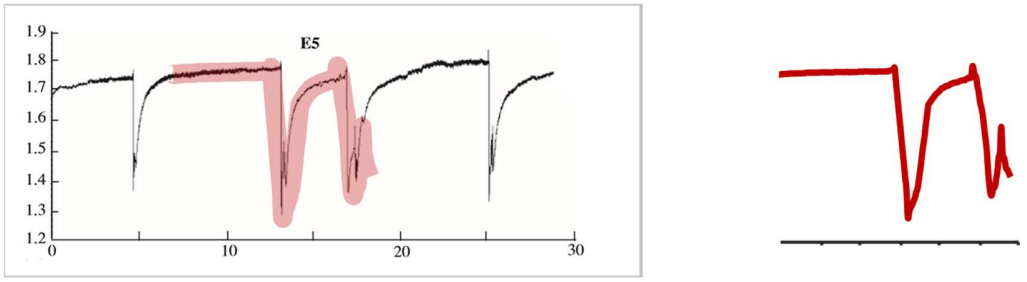


Fig. 14 *left*) A screen grab from [16] (figure 3.B in original) of an example of phloem sap ingestion behavior of tea green leafhopper and (*right*) our extracted version of it. We brushed a light copy (red) of our extracted pattern onto the original.

Note that our tracing of the original screen capture is not perfect. We are relying on the robustness of our distance measure. Does this behavior also occur in Asian Psyllids (*Diaphorina citri*), insects that are vectors of citrus greening disease? There is a large public archive of such data, consisting of about 50 hours of data at 100 Hz [30]. We used the extracted template from Fig. 14.*right* to search this archive under both the Euclidean distance and MPdist. Fig. 15 shows the results.

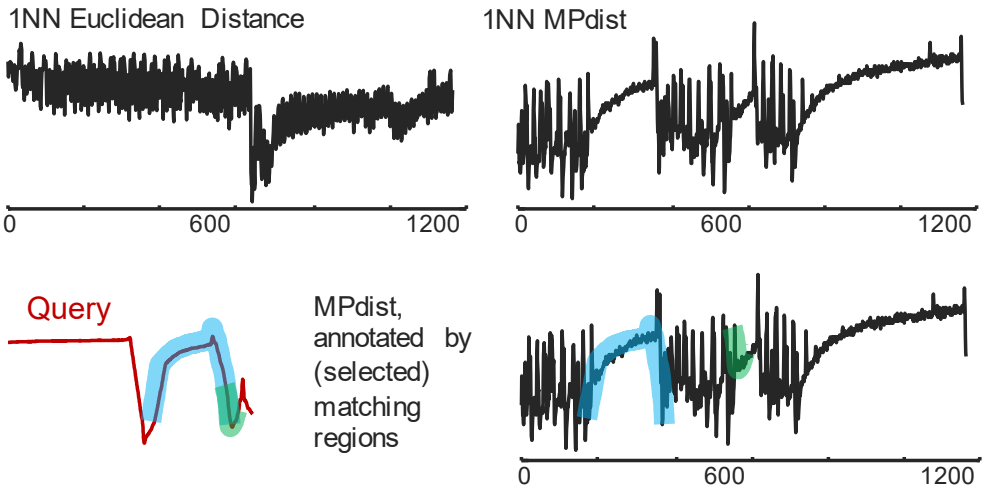


Fig. 15 *top*) The nearest neighbors to the tea green leafhopper template retrieved by Euclidean distance and MPdist. *bottom*) selected subsequences from the query brushed onto the best match suggest *why* this was the MPdist 1NN.

We can confirm that the 1NN under MPdist *is* a region of a phloem sap ingestion behavior of the Psyllids [5]. In contrast, the best match under ED is merely sensor noise from a transition phase, with no obvious semantic meaning. It might be imagined that ED *could* be made to work here. For example, by carefully editing the query to remove the long constant region at its beginning. However, this would require lots of human effort, whereas MPdist simply works, even given a query with spurious regions.

It might be argued that this simple anecdote was the results of luck. To more forcefully demonstrate the utility of MPdist in this domain, in collaboration with the UCR entomologist Dr. Kerry Mauck we created a new dataset for the UCR archive [5]. The dataset, InsectEPG1 is a three-class problem with 311 instances with a 65/246 train/test split. Using the same protocol as in Section 4.1, ED can achieve and error-rate of 28.0%. DTW can improve that significantly to 14.2%, but MPdist achieves just 7.7%.

4.5 A Case Study in Power Demand Data

One of the areas in which time series data mining has been applied the most in recent years is in mining electrical power demand time series. In the absence of a benchmark dataset in this domain, we created one. While examining the REFIT dataset [23], we noticed that House 1 has two freezers that were individually metered. For each freezer, we extract 1,500 40-minute snippets, carefully aligning them (for the benefit of rival methods, MPdist is phase invariant), such that the increase in power demand happened at the third minute. Fig. 16 shows some examples, clustered by both the Euclidean Distance and by the MPdist.

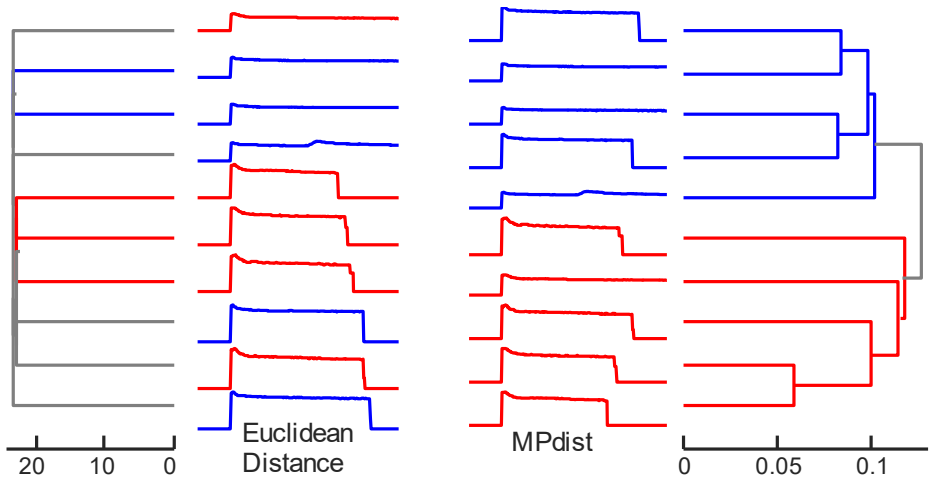
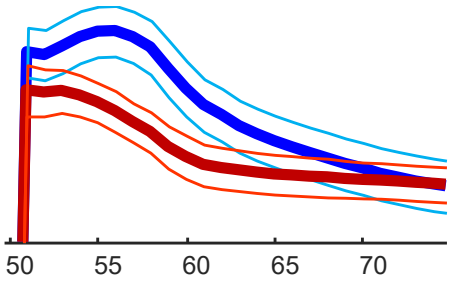


Fig. 16 *left*) Ten examples from the Freezer dataset clustered with Euclidean distance and $MPdist_{40}$.

The clustering results suggest that ED has great difficulties here. This is also true for *classification* of this data. With a 152/2848 train/test split, ED has 35% error-rate. Yet MPdist learns a parameter of 40 on the training data, and then $MPdist_{40}$ achieves a significantly better error-rate of just 5%. What explains such a drastic difference? In Fig. 17, we hint at the answer.



The mean values of sub-region of **class 1** and **class 2** of the Freezer dataset (bold lines), augmented by lines indicating ± one standard deviation (fine lines)

Fig. 17 For a just small sub-region of each class of the Freezer dataset, the shapes of the data is class conserved.

The ability of MPdist to focus on the relatively small amount of class conserved behavior and ignore everything else is critical in this domain, regardless of the analytical task. To see this, let us consider a new task, that of outlier/anomaly/novelty detection. While there are dozens of general algorithms for this problem in general, for time series it has been shown that distance-based anomaly detection is often particularly effective [32]. Of course, this begs the question of *which* distance measure to use. In Fig. 18 we show a random sample of the Freezer dataset imbedded into 2D space using multidimensional scaling (MDS).

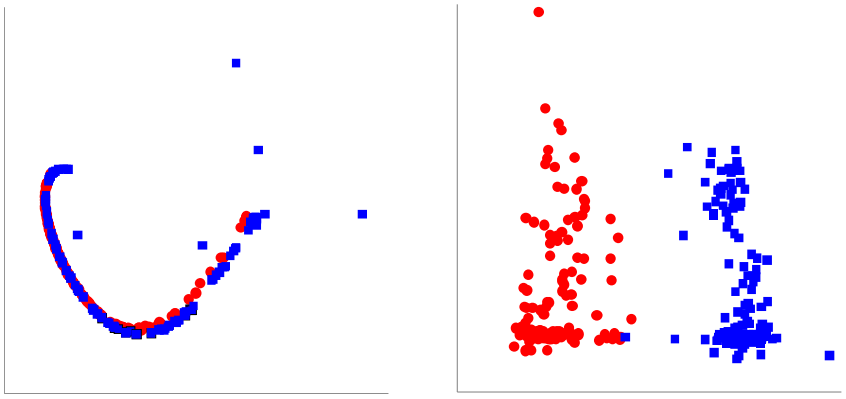


Fig. 18 A 10% subset of the freezer data projected into two-dimensional space using Euclidean distance (*left*) and MPdist_{40} (*right*). Note that the MPdist projection is almost perfectly linearly separable.

In Fig. 19 we repeated the experiment with a minor change. This time we added one addition example for the Freezer dataset, which we first flipped left-right. As the reader will appreciate, this transformation does not change the mean, standard deviation, autocorrelation most other statistical properties of the data object.

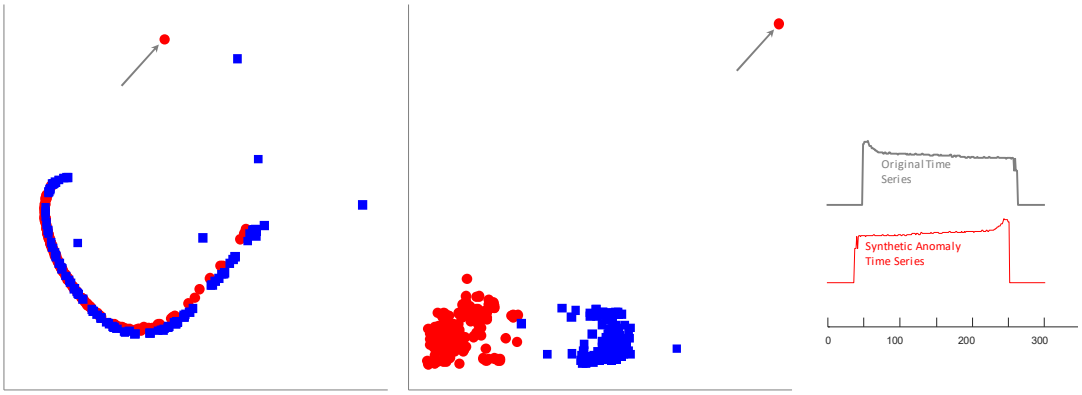


Fig. 19 A 10% subset of the freezer data, augmented by a single synthetic anomaly, projected into two-dimensional space using Euclidean distance (*left*) and MPdist_{40} (*center*). The synthetic anomaly (*right*) is a natural data object from this domain that was flipped left-right.

In the Euclidean projection, the existence of an anomaly is debatable. The point representing the anomaly is not much further from the main grouping than several natural exemplars. In contrast, in the MPdist projection, the anomaly is startlingly obvious.

4.6 Anecdotal Comparison to State-of-the-Art

As we noted above, head-to-head comparisons of classifiers on the UCR benchmarks may say little about the utility of methods on real-world problems. Nevertheless, it is instructive to compare directly with a method that is seen to be among the state-of-the-art.

The BOSS method is claimed to be state-of-the-art [26][29] and independently shown to indeed be very competitive [1]. It is somewhat indirect (working in an abstract quantized Fourier space) and complicated, requiring four parameters to be carefully set. In contrast, MPdist intuitively operates directly on the original data, and requires only one free parameter to be set. In spite of its simplicity, as Fig. 20 shows, MPdist can perform as well on a simple problem designed to showcase BOSS.

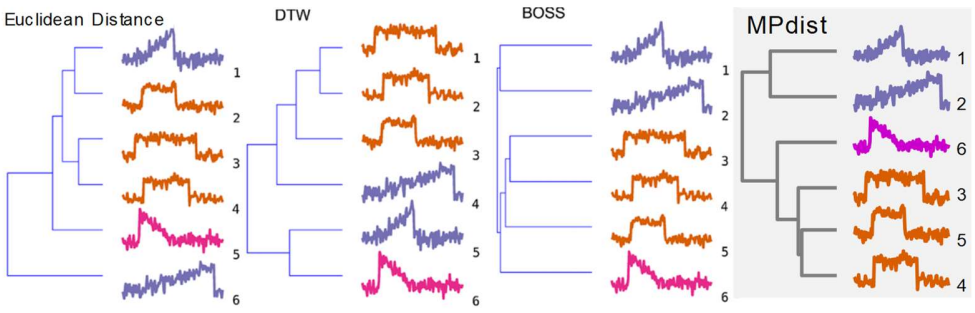


Fig. 20 Hierarchical clustering of CBF data with three distance measures, (adapted from [26]), with the clustering obtained by $MPdist_{50}$. As Schäfer points out, both Euclidean distance and DTW struggle here, and his proposed BOSS algorithm *just* manages to correctly cluster the data. MPdist also correctly clusters the data, with greater separation.

Beyond the number of parameters to tweak, there is the issue of scalability. The BOSS reports 0.36 seconds to compare two time series of length 1024 [29], and does not come with any indexing or speedup measure for subsequence search. Suppose we had just one minute of ECGs sampled at 1,024 (a typical sampling rate for a hospital quality machine), and we want to search it with a one second long query. This requires 60,417 = 61440 – 1024 + 1 distance comparisons, and would take six hours to finish, or about 360 times *slower* than real-time. In contrast, we did this experiment on similar hardware and finished in 0.8 seconds, 73 times *faster* than real-time. Thus, we are ~26,000 times faster than BOSS for query-by-content. For this reason alone, we dismissed BOSS from our case studies above.

4.7 Robustness Tests

While the previous section offers evidence on MPdist's utility for time series classification, as we had previously noted, the UCR archive datasets may be to "clean" and contrived to really highlight MPdist's robustness. To demonstrate this robustness, we consider one of the UCR datasets *FOWL*, and measure the relative performance of Euclidean Distance and MPdist as we add in ever increasing amounts of "distortion". In particular, the distortions are:

- **Linear Trend**: To each instance, we randomly add a linear trend in the range from 0 to ± 0.4.
- **Spurious Data**: To each instance, we randomly prepend a random walk of length 0 to 40 datapoints.

- **Phase Shift**: To each instance, randomly perform a circular shift of 0 to ± 40 datapoints.

Fig. 21 shows examples of these distortions.

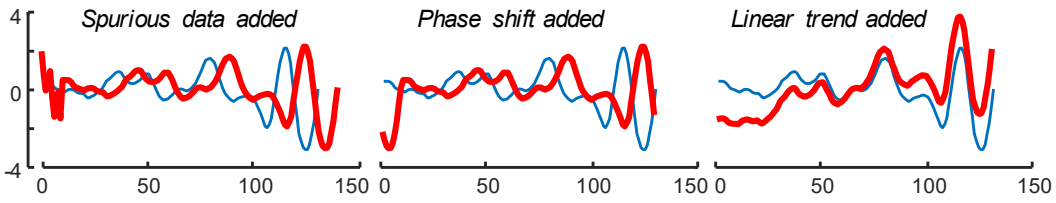


Fig. 21 Some before and after examples of various distortions added to exemplars from the FacesUCR data.

For MPdist, we use the parameter learned in Section 4.1 above. Fig. 22 shows the results.

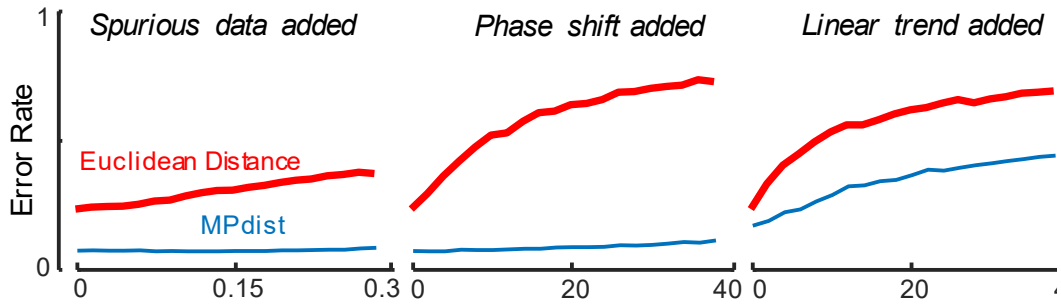


Fig. 22 *left-to-right*) The robustness of MPdist (red) and Euclidean Distance (blue) in the face of increasing corrupted data for three types of distortion.

We note that *some* of these distortions can be addressed in other ways. One way is by preprocessing each subsequence before comparison, another way might be by using a specialized distance measure. For example, K-Shape would have no problem with phase shift. However, to our knowledge, the MPdist is the only measure that can *natively* handle all these issues.

4.8 Alignment Insensitivity Revisited

In the previous section we performed an experiment to show that the MPdist is relatively insensitive to alignment or phase. However, because of the contrived nature of the UCR Archive datasets we had to resort to perturbing an existing dataset with random circular shifts. Here we revisit the experiment with natural data, leveraging an interesting dataset donated to the community by [11].

Undetected internal bleeding after surgical procedures or trauma is a major medical issue. Therefore, there is significant research on early and reliable detection of internal

bleeding. In [11] the authors created the Bleeding Detection Data Set from vital signs of pigs (*Sus domesticus*), measured at 250Hz using a classic bed-side hemodynamic monitoring system designed for human patients. The collected measurements are arterial blood pressure (ABP), central venous pressure (CVP), and airway pressure (AP). The data was collected from a cohort of 52 healthy pigs subjected to induced slow bleeding. Each animal has been sedated, instrumented, left to rest for half an hour, and then bled at the fixed rate of 20mL/min. Fig. 23 shows some sample data, both *before* and *after* the surgical intervention.

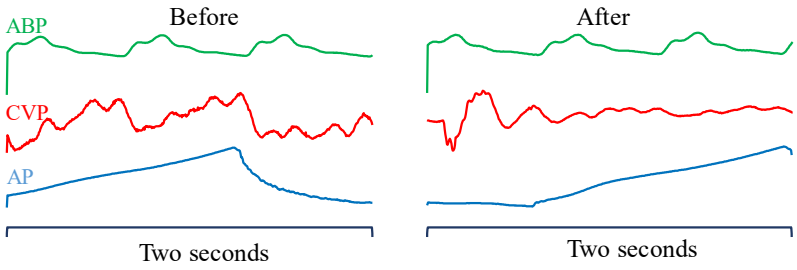


Fig. 23 Two-second long snippets of the "before and after" data from Pig-1 of the internal bleeding dataset. *top to bottom*), Arterial Blood Pressure (ABP), Central Venous Pressure (CVP), and Airway Pressure (AP).

From this collection we created three datasets. In each case we did the following. For each of the 52 pigs, we took the first 2,000 data points (eight seconds) of data from both the *before* and the *after* traces, creating a training set with 52 classes, with just two exemplars per class.

For the test set, we took the second (2,001 to 4,000) and third (4,001 to 6,000) snippets from both the *before* and the *after* traces, creating a test set with four exemplars per class.

This is a difficult classification problem, beyond the fact that the data is not aligned, there are a huge number of classes, and a very small training set. Moreover, while the two examples in each class represent the same individual, they represent a snapshot of the individual under very different medical conditions.

We compared 1-nearest neighbor classification using MPdist, Euclidean Distance, cDTW (DTW, with its warping parameter constrained [11]), BOSS [29], and SBD (K-shape) [21]. In order to make sure we did justice to SBD, we asked the original author to run this experiment, which he graciously agreed to do. The author of BOSS [29] was not available to run our experiments, so ran them ourselves. We found it was difficult

to learn the four parameters on the relatively small training sets, so we allow the algorithm to "cheat", testing many combinations of parameters and choosing the best on the *test* data.

Table 1 Comparing between MPdist, Euclidean, cDTW and SBD (K-Shape), BOSS distance measures over ABP, CVP and AP datasets in terms of error-rate on the three internal bleeding datasets.

	MPdist	Euclidean	cDTW	SBD	BOSS
ABP (%)	0	87.50	86.54	46.64	74.34
CVP (%)	10.09	91.83	90.38	42.79	69.23
AP (%)	18.75	94.23	89.42	85.58	79.76

The reader may wonder if the poor performance of DTW is due to the fact that with such limited training data, it could not learn the best setting for the width of warping window. It has been shown that a careful setting of this parameter can make a significant difference in accuracy [1], however our two-exemplar per class training set may make it challenging to find a good setting [9]. To test what difference (if any) a better setting of the parameter could make, similar to the treatment of BOSS, we can allow DTW to "cheat", and choose the setting that maximizes the accuracy on the *test* data. If we do this, the error-rates of ABP and CVP are slightly improved, to 75.00% and 84.13% respectively.

4.9 Handling Missing Data

The possibility of missing data plagues most time real-world time series data mining applications [33][12][35]. In some cases, the missing data may be explicitly marked with "`NaN`" or "`INF`", or with some other non-numeric special value. In other cases, the missing data may be marked with a special *numeric* value. For example, Aspentech, a major provider of time series analytic tools for the oil and gas industry, uses -9999 to represent missing data. Choosing a numeric value like this to represent missing data can lead to issues, especially if the data is shared beyond a closed system. For example, consider the data shown in Fig. 24. It is clear from common sense (the temperature in a large room cannot change instantaneously) that the many, equal-valued, low-valued readings correspond to missing data. However, the repeated low value here is -4.88909110, which does not seem to be a special value. It may originally have been a special value, say -99, that was cast to an innocuous number during a data normalization step. However, because of the poor provenance of the data, it is difficult to be sure. We call such data *implicitly* missing data, to differentiate it from *explicitly* missing data

case denoted with non-numeric values. In either case, such data offers significant challenges for time series data mining.

In the last decade, dozens of ideas have been introduced to allow similarity measurements in between time series which may have missing values [26][33][34][12]. There are two major approaches. The first is to "repair" the missing data with some data imputation method [12][35]. The second approach is to somehow adapt the distance measure itself [26]. For example, the DUST measure is an attempt to generalize Euclidean distance to missing data [26].

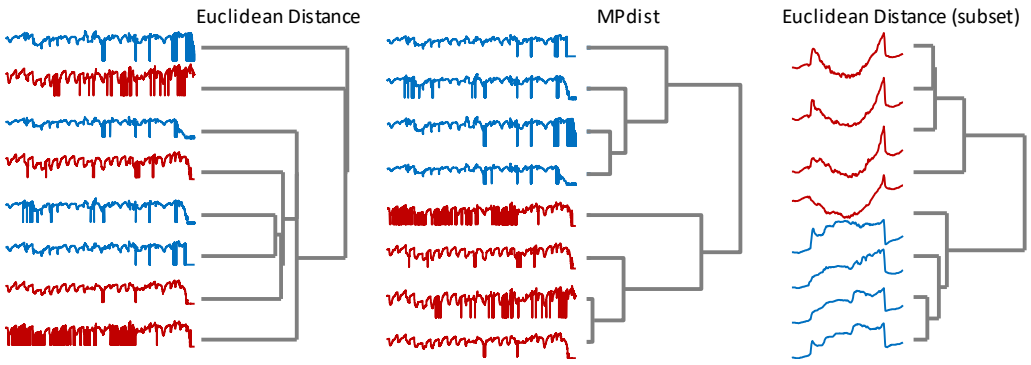


Fig. 24 A subset of the data from the Berkeley Intel Lab Mote Data clustered using complete linkage hierarchical clustering [27]. Each time series is 7,000 data points long, or about 2.5 days. *left*) The clustering obtained by Euclidean distance. *center*) The clustering obtained by $MPdist_{600}$. *right*) The clustering obtained by Euclidean distance using only the region from 5,000 to 5,200.

Note that repairing the data is only a good option for *explicitly* missing data. If an algorithm is given free rein to repair data in the more general case, it may introduce artifacts into true data that simply did not conform to the analyst's assumptions [35].

Fortunately, the MPdist can bypass all such considerations. So long as there exists at least one region in each time series that is longer than *L,* then the MPdist is defined. Note that the location of the error free regions does not have to be in the same place in each time series. Moreover, the MPdist is indifferent to whether the missing data is implicit, explicit or any combination of the two.

To demonstrate the effectiveness of the MPdist for missing data, we clustered data from the Berkeley Intel Lab Mote Dataset, corresponding to about two and a half days [26]. We choose half the data from one side of the building, and the other half from the other side, to provide a natural grouping. Fig. 24.*left* shows that the Euclidean distance has a hard time with this data, returning what looks like a random clustering. In contrast, Fig. 24.*center* shows that the MPdist correctly partitions the data. Here the data is so noisy

that it is not clear with a casual inspection that the clustering is really semantically meaningfully. To allow us to see that it is, we found a contiguous region of 200 data points in which none of the eight time series had missing values, and clustered just those regions as shown in, Fig. 24.*right*. Note that the MPdist clustering used $MPdist_{600}$, however we get the correct portioning of the data if we vary L anywhere in *at least* the order of magnitude range of 60 to 600.

This invariance to implicit/explicit missing data is a very useful feature of our proposed distance measure. Other distance measures can sometimes be adapted to handle missing data, by modifying the data [35] or the algorithm itself [26]. However, the MPdist simply works "out-of-the-box", there is nothing to do.

4.10 Scalability Tests

To demonstrate the scalability of our algorithm, we performed the following experiments. Note that, unlike the state-of-the-art ED/DTW subsequence search algorithm in [22], our algorithm's performance is independent of the *structure* of data. In particular, the time taken depends only on m, the length of the data being searched, n the length of the query, and L, MPdist's sole parameter. Thus, we do not need to average over multiple runs to evaluate performance. We compare brute force MPdist and our subsequence based acceleration of MPdist. We also compare to the MASS algorithm [18]. While it does not really make sense to compare the speeds of algorithms that can return different answers, we include MASS because is the optimally fast subsequence search algorithm under *any* distance measure. Thus, it may be seen as a lower bound on performance[3] of any subsequence search algorithm. We begin by fixing n to 128, and L to $n/2$ (typical values used in Section 4.1) and we measure the time taken for increasing large datasets.

Next, we fix m to 2^{16} and vary n between 32 and 256 (again with L to $n/2$). Finally, we fix m to 2^{16} and n to 128, vary L between 32 and 128. The results of these experiments are shown in Fig. 25.

[3] This is for the case where the query length is *not* known ahead of time. If the query length *is* known, it may be possible to have a faster index-based search.

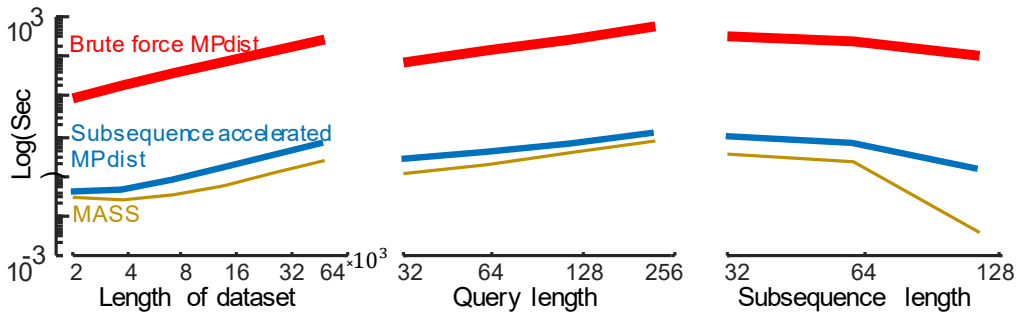


Fig. 25 *left-to-right*) The effect of varying the size of the dataset, the query length and the sliding window length. Note the logarithmic time axis.

These results show us that our subsequence based acceleration of MPdist achieves three orders of magnitude speedup over the brute-force method, and it is not substantially slower than the fastest implementation of ED based subsequence search.

Finally, to ground these numbers, we briefly revisit the insect data considered in the previous section. To search a recording of length 360,000 corresponding to 1 hours of wall-clock time at 100 Hz, with a query length n = 128 and with $L = n/2$, took 4.71 seconds, or about 764 times faster than real-time for that dataset.

4.11 When can MPdist Fail?

All distance measures can be made to perform worse than some rival, by an adversarial creation of data. It can be instructive to consider examples. Fig. 26 shows an example for MPdist.

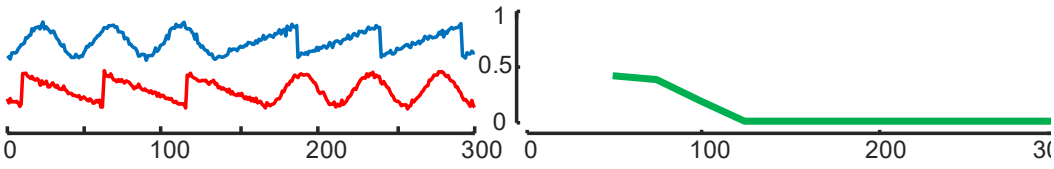


Fig. 26 *left*) Two exemplars from a classification dataset. Class A is 3 bumps and 3 saw-tooth elements, Class B is the mirror image of A. *right*) the error-rate of MPdist on this problem as we increase the sliding window length L.

Here the ultra-liberal invariance of MPdist means that it has a difficult time telling the two classes apart when L is very short. However, as the length of L increases, every subsequence of **A** will include *some* bumps and *some* saw-teeth, in *that* order, allowing the error-rate to fall to zero. Thus at least for this dataset, so long as we have enough data to learn a reasonable value for L, we can do no worse than classic ED. More generally, the fact that MPdist includes the ED as a special case offers us some bounds

on how badly the MPdist could fail, given only the assumption that a reasonable value for L is known or can be learned.

4.12 Revisiting the Hardcoded Choice of k

All distance measures achieve robustness by discarding some information [3]. For example, Z-normalized ED discards all information about the *mean/variance* [1][3], K-shape discards *phase* information [21] and DTW discards some *alignment* information etc. As explained in Section 3, MPdist discards 95% of the Matrix Profile values, considering only the maximum of the lowest 5% of the values to report a distance. We have glossed over this choice of the 5% value. Here we briefly justify this decision.

Clearly k cannot be 0% (or everything would be equidistant). If it was 100% we would be forced to explain *all* the data, including the dropouts, spikes, spurious prefixes/suffixes etc. These are the exact distortions we wish to be invariant to. For example, Fig. 15 shows that MPdist was able to find semantically identical patterns, *only* by ignoring the large amounts of spurious and noisy data. So, our value must be between these two extremes of 0 to 100%, and probably towards the lower end of that range. In Fig. 21, we show the results of an experiment that makes this issue moot, or at least of very low priority. We repeated the experiments of Section 4.1, but this time holding L constant (at its learned value) and changing k from 1 to 10%. At least over this range it makes essentially no difference. Given that we can vary this value over an order of magnitude with no effect, we simply choose the halfway point and defer a more detailed discussion to [17].

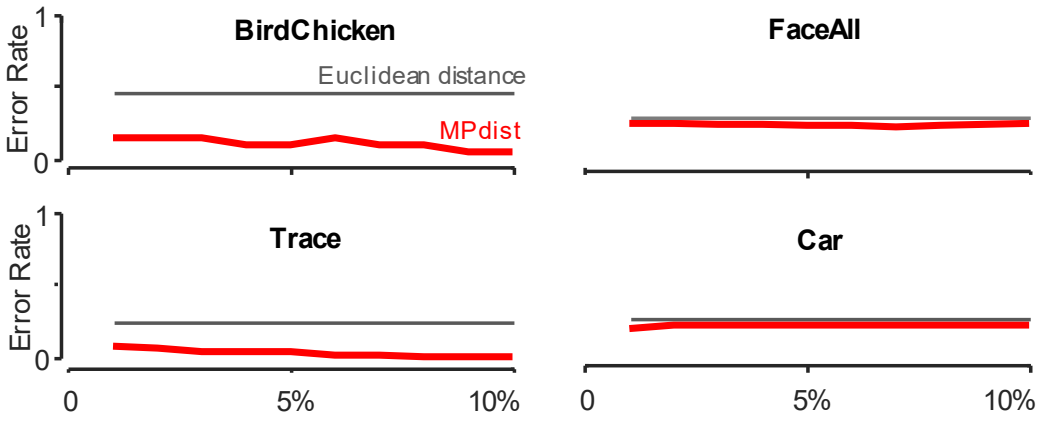


Fig. 27 On 4 random datasets from [5], we tested the effect of changing k between 1 and 10% on holdout classification. Over this range it makes essentially no difference.

5 Conclusions

We have demonstrated that many real-word domains require a distance measure that is more robust than the current state-of-the-art. We have introduced MPdist, a novel distance measure to repair this omission. We have shown that the MPdist is more robust to noise, irrelevant data, misalignment etc., than either Euclidian distance or DTW. Moreover, these desirable features do not come at the cost of lethargy. Under typical assumptions the MPdist can process data three orders of magnitude faster than real-time data streams from accelerometers or medical devices.

As part of this research effort we have created or adapted three new time series classification datasets. We have donated these to the UCR archive (with the exact splits we used) so the community can build on our efforts.

In addition, the first application of MPdist recently appeared in [15]. They introduced the concept of *time series snippets*, a representation for visualizing and summarizing massive time series datasets. The authors argue that their definition of time series snippets is enabled by the unique properties of the MPdist; no other distance measure would work for their task. We believe that this snippet paper will be the first of the many applications of the MPdist.

We discussed the limitations of the MPdist. It is significantly slower than Euclidean Distance and difficult to index for disk resident data. In addition, it requires a parameter to be set. In future work, we hope to show how to index disk-resident data and consider implications of the MPdist for medical data mining and human behavior discovery from wearables.

Finally, we have made all code and data freely available to the community (in perpetuity [17], to allow the community to confirm and extend our findings.

6 Acknowledgments

We thank the reviewers of this version, and the conference version of this paper for their useful comments. We acknowledge funding from NSF IIS-1161997 II.

References

[1] Bagnall, A., et al, 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min. Knowl. Discov., 31(3), pp.606-660.

[2] Baker, M.B., Venugopal, P.D. and Lamp, W.O., 2015. Climate change and phenology: *Empoasca fabae* migration and severity of impact. PloS one, 10(5), p.e 0124915.

[3] Batista, G.E., Wang, X. and Keogh, E.J., 2011, April. A complexity-invariant distance measure for time series. In Proc' of the 2011 SIAM SDM pp. 699-710.

[4] Berndt, D.J. and Clifford, J., 1994, July. Using dynamic time warping to find patterns in time series. In KDD workshop (Vol. 10, No. 16, pp. 359-370).

[5] Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A. and Batista, G., 2015. The UCR time series classification archive. URL *www.cs.ucr. edu/~ eamonn/time_series_data.*

[6] Darvishzadeh, A., Entezari, N. and Stahovich, T., 2018, August. Finding the Answer: Techniques for Locating Students' Answers in Handwritten Problem Solutions. In 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR) (pp. 587-592). IEEE.

[7] Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A. and Keogh, E., 2018. The ucr time series archive. arXiv preprint arXiv:1810.07758.

[8] Dau, H., Begum, N. and Keogh, E., 2016. Semi-Supervision Dramatically Improves Time Series Clustering under Dynamic Time Warping. In Proc' of 25th CIKM pp. 999-1008.

[9] Dau, H.A., Silva, D.F., Petitjean, F., Forestier, G., Bagnall, A. and Keogh, E., 2017, October. Judicious Setting of Dynamic Time Warping's Window Width Allows More Accurate Classification of Time Series. IEEE Conference Publications.

[10] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine learning research, 7(1), pp.1-30.

[11] Guillame-Bert, M. and Dubrawski, A., 2017. Classification of time sequences using graphs of temporal constraints. *The Journal of Machine Learning Research*, *18*(1), pp.4370-4403.

[12] Honaker, J. and King, G. *What to do about missing values in time-series cross-section data*. American Journal of Political Science, 54.2 (2010): 561-581.

[13] Hu, B., Chen, Y. and Keogh, E., 2016. Classification of streaming time series under more realistic assumptions. Data Mining and Knowledge Discovery, 30(2), pp.403-437.

[14] Hu, B., Chen, Y., Zakaria, J., Ulanova, L. and Keogh, E. Classification of Multi-dimensional Streaming Time Series by Weighting Each Classifier's Track Record. ICDM 2013: 281-290

[15] S. Imani, F. Madrid, W. Ding, S. Crouter, E. Keogh, "Matrix Profile XIII: Time Series Snippets: A New Primitive for Time Series Data Mining," in IEEE Int. Conf. on Data Mining (ICBK2018), 2018, (Accepted).

[16] Jin, S., Chen, Z.M., Backus, E.A., Sun, X.L. and Xiao, B., 2012. Characterization of EPG waveforms for the tea green leafhopper on tea plants and their correlation with stylet activities. Journal of insect physiology, 58(9), pp.1235-1244.

[17] Keogh, E. (2017). Supporting website for this paper: https://sites.google.com/site/mpdistinfo/.

[18] Mauck, K. Personal Communication. 1-12-18.

[19] Mueen, A. The MASS algorithm. (accessed 24-5-16) url: *www.cs.unm. edu/~ mueen/FastestSimilaritySearch.html.*

[20] Murray, D., et al., 2015. A data management platform for personalised real-time energy feedback. Proc' of EEDAL'15.

[21] Paparrizos, J. and Gravano, L., 2015, May. k-shape: Efficient and accurate clustering of time series. In Proc' of the 2015 ACM SIGMOD (pp. 1855-1870). ACM.

[22] Rakthanmanon, T., et. al. 2012, August. Searching and mining trillions of time series subsequences under dynamic time warping. In Proc'of the 18th ACM SIGKDD pp. 262-70.

[23] Refitsmarthomes.org. 2018 REFIT Dataset. (Accessed 1-21-18) Available at: *www.refitsmarthomes.org/index.php/data.*

[24] Rodriguez, et al. 2006. Rotation forest: A new classifier ensemble method. IEEE PAMI, 28(10), pp.1619-1630.

[25] Ruutu, J. and Kilkki, M., 2000. System and method employing last occurrence and sliding window technique for determining minimum and maximum values. U.S. Patent 6,023,453.

[26] Piatetsky-Shapiro G. URL retrieved on April-2-2018. https://www.kdnuggets.com/polls/2014/data-types-sources-analyzed.html

[27] Samuel M. (2004). URL http://db.csail.mit.edu/labdata/labdata.html.

[28] Sarangi, S.R. and Murthy, K., 2010, July. DUST: a generalized notion of similarity between uncertain time series. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 383-392). ACM.

[29] Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. Data Min. Knowl. Discov. 29(6): 1505-30 (2015).

[30] Willett, D.S., George, J., Willett, N.S., Stelinski, L.L. and Lapointe, S.L., 2016. Machine learning for characterization of insect vector feeding. PLoS computational biology, 12(11).

[31] Ye, L. and Keogh, E., 2009. Time series shapelets: a new primitive for data mining. Proc' of the 15th SIGKDD pp. 947-56.

[32] Yeh, C.C.M., et al. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View that Includes Motifs, Discords and Shapelets. Proc' of 16th IEEE ICDM, pp. 1317-22.

[33] Yeh, M.Y., Wu, K.L., Yu, P.S. and Chen, M.S., 2009, March. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology* (pp. 684-695). ACM.

[34] Yi, X., Zheng, Y., Zhang, J. and Li, T., 2016. ST-MVL: filling missing values in geo-sensory time series data. IJCAI 2016: 2704-2710.

[35] Zhang, A., Song, S., Wang, J. and Yu, P.S., 2017. Time series data cleaning: from anomaly detection to anomaly repairing. *Proceedings of the VLDB Endowment*, *10*(10), pp.1046-1057.

[36] Zhu, Y., et al. 2016. Matrix Profile II: Exploiting a Novel Algorithm and GPUs to Break the One Hundred Million Barrier for Time Series Motifs and Joins. In Proc' of the 16th IEEE ICDM, 2016 pp. 739-48.