Data Min Knowl Disc
https://doi.org/10.1007/s10618-018-0565-y

Optimizing dynamic time warping's window width for time series data mining applications

Hoang Anh Dau[1] · **Diego Furtado Silva**[2] · **François Petitjean**[3] · **Germain Forestier**[4] · **Anthony Bagnall**[5] · **Abdullah Mueen**[6] · **Eamonn Keogh**[1]

Received: 28 September 2017 / Accepted: 30 March 2018

Abstract Dynamic Time Warping (DTW) is a highly competitive distance measure for most time series data mining problems. Obtaining the best performance from DTW requires setting its only parameter, the maximum amount of warping (w). In the supervised case with ample data, w is typically set by cross-validation in the

Responsible editor: Jian Pei.

✉ Hoang Anh Dau
hdau001@ucr.edu

Diego Furtado Silva
diegofs@ufscar.br

François Petitjean
francois.petitjean@monash.edu

Germain Forestier
germain.forestier@uha.fr

Anthony Bagnall
ajb@uea.ac.uk

Abdullah Mueen
mueen@unm.edu

Eamonn Keogh
eamonn@cs.ucr.edu

[1] University of California, Riverside, Riverside, USA

[2] Universidade Federal de São Carlos, São Carlos, Brazil

[3] Monash University, Melbourne, Australia

[4] University of Haute-Alsace, Mulhouse, France

[5] University of East Anglia, Norwich, UK

[6] University of New Mexico, Albuquerque, USA

Published online: 09 April 2018

training stage. However, this method is likely to yield suboptimal results for small training sets. For the unsupervised case, learning via cross-validation is not possible because we do not have access to labeled data. Many practitioners have thus resorted to assuming that "*the larger the better*", and they use the largest value of *w* permitted by the computational resources. However, as we will show, in most circumstances, this is a naïve approach that produces inferior clusterings. Moreover, the best warping window width is generally non-transferable between the two tasks, i.e., for a single dataset, practitioners cannot simply apply the best *w* learned for classification on clustering or vice versa. In addition, we will demonstrate that the appropriate amount of warping not only depends on the data structure, but also on the dataset *size*. Thus, even if a practitioner knows the best setting for a given dataset, they will likely be at a lost if they apply that setting on a bigger size version of that data. All these issues seem largely unknown or at least unappreciated in the community. In this work, we demonstrate the importance of setting DTW's warping window width correctly, and we also propose novel methods to learn this parameter in both supervised and unsupervised settings. The algorithms we propose to learn *w* can produce significant improvements in classification accuracy and clustering quality. We demonstrate the correctness of our novel observations and the utility of our ideas by testing them with more than one hundred publicly available datasets. Our forceful results allow us to make a perhaps unexpected claim; an underappreciated "low hanging fruit" in optimizing DTW's performance can produce improvements that make it an even stronger baseline, closing most or all the improvement gap of the more sophisticated methods proposed in recent years.

Keywords Time series · Clustering · Classification · Dynamic time warping · Semi-supervised learning

1 Introduction

Clustering and *classification* are perhaps the two most fundamental tasks in time series data mining. They are useful tools in their own right, and they are a subroutine in many higher-level algorithms such as rule-finding, semantic segmentation, anomaly detection, visualization, and data editing (Petitjean et al. 2015). Both clustering and distance-based classification algorithms depend critically on the availability of a good distance measure (Bagnall and Lines 2014; Bagnall et al. 2017; Ding et al. 2008; Górecki and Łuczak 2013, 2014; Lines and Bagnall 2015; Paparrizos and Gravano 2015; Rakthanmanon et al. 2012; Zakaria et al. 2012). Over the last decade, the time series research community seems to have come to the consensus that DTW is a difficult-to-beat baseline for many time series mining tasks. Most recent research efforts in time series data mining have thus treated this distance measure as a default baseline; a competitive rival for justifying a novel distance measure or algorithm (Bagnall et al. 2017; Ding et al. 2008).

However, we believe that the extraordinary competitiveness and utility of DTW is still not fully appreciated in the community. This under-appreciation mostly stems from lacking awareness of the importance of DTW's single parameter, the amount of

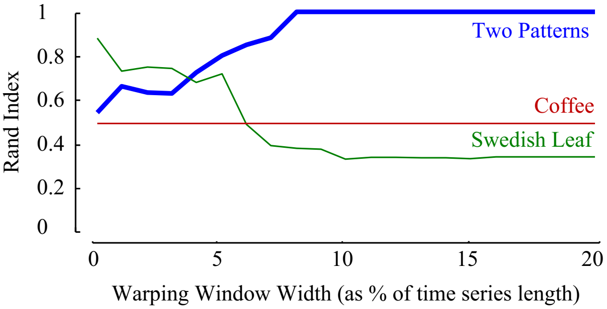


Fig. 1 Rand-Index versus warping window (w) width for three datasets, using a density-based clustering algorithm (Begum et al. 2015). A larger value of w can make things better, worse or have no effect

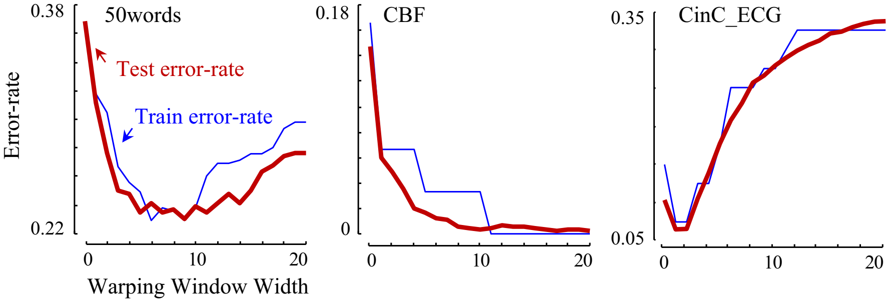


Fig. 2 (Blue/fine) The Leave-One-Out error-rate of three datasets for increasing values of w, using the DTW-based 1-nearest neighbor classifier. (Red/bold) The holdout error-rate. Note that the holdout accuracies closely track the predicted accuracies (Color figure online)

allowable warping (w), by the majority of the community. Moreover, there is a lack of robust methods to set w, even among those who *do* appreciate the critical role this parameter can have in producing good results (Lu et al. 2017).

The w parameter can affect the quality of the returned clusters (in case of clustering) or the class assignments (in case of classification) in unexpectedly different ways. Figure 1 illustrates this sensitivity to w for clustering under DTW. Figures 2 and 3 similarly demonstrate the sensitivity to w for classification. Note that the Euclidean distance is a special case of DTW when the warping constraint w is equal to 0.

Figure 1 shows how changing w affects the quality of clustering on three different datasets. For *Two Patterns*, increasing the amount of warping steadily improves clustering quality until it reaches perfection with $w = 9$. In contrast, for *Swedish Leaf*, a higher w reduces the quality of clustering from a very impressive (for a 15-class problem) Rand-Index of 0.87 at $w = 0$ to a stunningly low score of 0.32 at $w = 10$. This finding is more surprising given that allowing some warping improves the classification accuracy of this dataset slightly (Ding et al. 2008).

These results indicate that blindly using the Euclidean distance for clustering (i.e. $w = 0$) will yield poor results on some datasets. Likewise, another practitioner, perhaps motivated by the observation that DTW generally helps in classification problems (Bagnall et al. 2017; Bagnall and Lines 2014; Ding et al. 2008), and thus simply

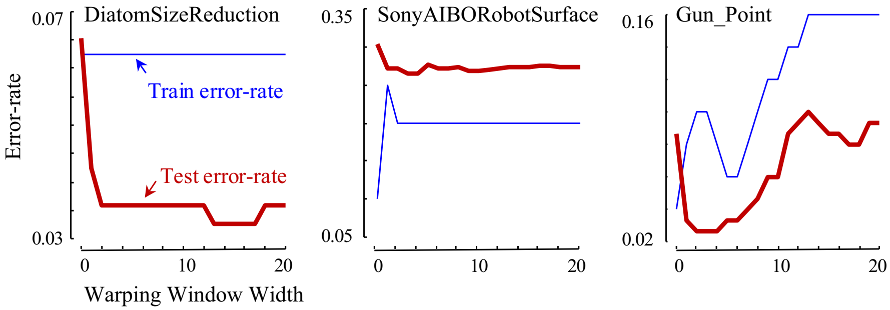


Fig. 3 (Blue/fine) The Leave-One-Out error-rate of three datasets for increasing values of w, using DTW-based 1-nearest neighbor classifier. (Red/bold) The holdout error-rate. In these examples, the holdout accuracies do *not* track the predicted accuracies (Color figure online)

clusters with a hard-coded value of w set at 10, will also do poorly on some datasets (Paparrizos and Gravano 2015).

The *Coffee* dataset is unusual (and empirically *rare*), since it is virtually unaffected by varying w (in Fig. 1 it is 0.48 when $w=0$ and 0.49 everywhere else), but even here it is still possible to make a poor decision. The time taken to compute DTW with a $w=0$ (denoted hereafter as cDTW^0) is about four orders of magnitude less than the time to compute cDTW^{100}. Thus, unnecessary large values of w incur a huge computational burden that produces no improvement.

For classification, we *do* have a way to learn the value of w. We can simply use cross-validation on the labeled training set to examine the error-rate for all values of w, then choose the one that minimizes the predicted error-rate (breaking ties by picking the smaller value). The underlying classifier used is 1-Nearest-Neighbor classifier (1-NN), which is a specific case of the k-NN algorithm when $k=1$. This is the method used by the authors of the UCR Archive (Chen et al. 2015) [and thus, reflected in hundreds of papers, for example Deng et al. (2013) and Górecki and Łuczak (2013)]. Figure 2 demonstrates that on many datasets, this simple method produces favorable results. It predicts the correct optimal value of w for *CinC_ECG*, and it is only off slightly for *CBF* and *50words*.

Nevertheless, the results above contrast with the examples in Fig. 3. In these cases, our estimation of the best value for w is much worse, and this has a detrimental effect on our holdout error. For instance, for *DiatomSizeReduction*, we predicted cDTW^0 to be an appropriate setting, but an oracle would have chosen cDTW^{13} and seen a 3.27% reduction in error-rate. Likewise, we predicted that cDTW^0 is the ideal setting for *Gun_Point*, but cDTW^2 would have reduced the rate of misclassifications by 6%.

The differences that a better estimate of w can make are difficult to overstate, and they have been acknowledged by a handful of other independent research efforts, such as Lu et al. (2017). The authors of this study found that DTW is an effective distance measure to classify HRM (high-resolution melt analysis) curves for identifying fungal species. They exploit the observation that the temporal distortions that DTW can compensate for are analogous to the temperature distortions in HRM data. Therefore, they see a direct application of DTW to their problem at hand. The authors examined the effect of the warping window size on the melt curve clustering by testing all the w values from 1 to 20, corresponding to a temperature range of 0.1–2 °C. They found that $w = 5$ is the most appropriate, and either values in lower or higher range deteriorate the performance.

In this work, we go beyond claiming that tuning the value of w is a good use of a practitioner's time. We argue that the constraint on maximum amount of warping, when set appropriately, can close most of the improvement gap on the "more sophisticated" time series classification/clustering methods proposed in recent years. We do not deny the advances in the state-of-the-art methods thanks to new algorithms and/or new distance measures. However, we strongly believe that a better understanding and methodology in setting w can make the "good old" DTW an *even* stronger baseline, eliminating the need for overly complicated methods. To further support this claim, let us consider some examples from recent literature.

In the context of time series classification, Deng et al. proposes a time series forest ensemble method (Deng et al. 2013). One of their reported successes is in halving the error-rate on *Gun_Point* to 4.7%. However, Fig. 3right shows that when the 1-NN classifier utilizes DTW as its distance measure (1-NN-DTW), a better choice of w could *further* halve their reported error-rate to 2.7%.

Similarly, Górecki and Łuczak (2013) introduce a new distance measure $\mathrm{DD}_{\mathrm{DTW}}$ that combines the DTW distances calculated both on the raw data and its derivatives (i.e. the mixture weights being learned by cross-validation). Among the datasets successfully considered are *Lightning2* and *Lightning7*. The authors note that they can reduce the error-rate of *Lightning2* to 13.11%, but a better choice of w for 1-NN-DTW could significantly beat this with just an 8.2% error-rate. Likewise, with *Lightning7*, their method can reduce the error-rate to 32.88%, which is impressive given a high error-rate of 42.47% with the Euclidean distance. However, if we use $w = 8$ for this dataset, 1-NN-DTW can achieve a much better error-rate of 23.28%.

In a follow-up paper (Górecki and Łuczak 2014), the authors further improve on the previous method by adding the DTW distance between transforms of time series. Now the new method can match the performance provided by DTW with the best w for the two datasets mentioned above. Yet, this is still a somewhat ad-hoc gain after much thought and further processing. The authors themselves admitted, "*due to the high degree of nonlinearity, the method does not easily admit a rigorous theoretical analysis.*"

It is important to clarify that we are not claiming the works above are without merit. A better setting of w might further boost their performance, especially for the works of Jeong et al. (2011) and Kate (2015). Yet, in most cases, the community has been proposing rather complex methods for relatively modest gains. The results in Fig. 3 suggest that similar or greater improvements are possible with *existing* techniques if we have a better method to discover a suitable warping constraint. There are also strong reasons to favor existing techniques, as they are amenable to many optimizations that allow them to scale to trillions of data points or to real-time deployment on resource-constrained devices (Rakthanmanon et al. 2012).

This work unifies two previous research efforts (Dau et al. 2016, 2017) under a coherent theme: learning DTW's warping window width for time series data mining applications. We have (re)structured these texts to tell a single narrative, that carefully setting DTW's warping window width offers more "bang-for-your-buck" than any other simple change you can make. Integrating these two papers in the current work allows us to more forcefully make a point that was only made in passing previously (Dau et al. 2016, 2017). In general, on any given dataset, the best warping window width for clustering is not the best warping window width for classification.

The rest of this the paper is organized as follows. In Sect. 2, we explain the background material of the problem we are solving. In Sect. 3, we offer a semi-supervised method to learn w for time series clustering. In Sect. 4, we discuss a resampling method to learn w for time series classification. Both methods target the scenarios in which access to labeled data is limited. We summarize findings and offer directions for future work in Sect. 5.

2 Related work and background

2.1 Dynamic time warping

DTW is a distance measure that originated in the speech recognition community. Recent work strongly suggests that DTW is the best distance measure for many data mining problems (Ding et al. 2008). In an influential paper (Rakthanmanon et al. 2012), authors state, "*after an exhaustive literature search of more than 800 papers, we are not aware of any distance measure that has been shown to outperform DTW by a statistically significant amount,*" and very recent independent work has empirically confirmed this with exhaustive experiments (Paparrizos and Gravano 2015). Of course, these results must come with several caveats, the most important of which is that almost all papers (including this one) test only on data from the UCR Archive (Chen et al. 2015). While the archive is large and diverse, it reflects only distribution of datasets the curators could make or obtain, not the distribution of real-world problems that are worthy of addressing. Nevertheless, it is telling that in a very competitive research area, there are at least two dozen papers published on time series classification each year, there is still no technique that unambiguously beats DTW on more than half the datasets in the archive.

As illustrated in Fig. 4 left, DTW allows a *one-to-many* mapping between data points, thus enabling a meaningful comparison between two time series that have similar shapes but are locally out of phase. To find the warping path W, we construct the distance matrix between the two time series Q and C. Each element (i, j) in this matrix is the squared Euclidean distance between the ith point of Q and jth point of C. The warping path W is a set of contiguous matrix elements that define the alignment between Q and C. The kth element of W is defined as $w_{k=(i,j)_k}$.

The warping path is subject to several conditions. It must start and finish in diagonally opposite corner cells of the matrix; the subsequent steps must be in the adjacent cells; and all the cells in the warping path must be monotonically spaced in time. Among all the warping paths possible, we are only interested in the path that minimizes the differences between the two time series.

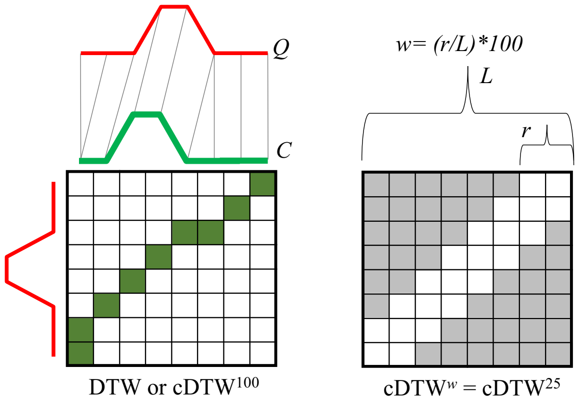


Fig. 4 (Left) The unconstrained warping path for time series Q and C. Such warping paths are allowed to pass through any cell of the matrix. (Right) A constrained DTW. We can choose to constrain the warping path to avoid passing through cells that are too far from the diagonal

$$DTW\,(Q, C) = min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\}$$

DTW computation lends itself to the dynamic programming paradigm. In the dynamic programming implementation of DTW, we construct the alignment cost matrix D. The cell at location (i, j) of this matrix is the minimum cumulative sum of the alignment cost up to Q_i and C_j. The bottom corner cell of the matrix contains the cost of the full alignment between Q and C, which is the DTW distance between the two time series.

$$D(i, j) = (Q_i - C_j)^2 + \min(D(i-1, j), D(i, j-1), D(i-1, j-1))$$

.

A DTW implementation that does not restrict the boundary of the warping paths on the distance matrix is referred to as an unconstrained DTW. A constrained DTW is a variant that imposes a limit on how far the warping path can deviate from the diagonal. This limit is known as the maximum warping window width (w). For example, in Fig. 4 right, the warping path cannot visit the gray cells.

The constrained DTW helps avoid pathological mappings between two time series when one point in the first time series is mapped to too many points in the other time series. For example, DTW should be able to map a short heartbeat to a longer heartbeat, but it would never make sense to map a single heartbeat to ten heartbeats. In addition, the constraints have the additional benefit of reducing the computation cost by narrowing the search for qualified paths. A typical constraint is the *Sakoe–Chiba Band* (Sakoe and Chiba 1978), which expresses w as a percentage of the time series length. We denote DTW with a constraint of w as cDTW^{w}.

The Euclidean distance between the two time series is a special case of DTW when w is set to 0%, enforcing a one-to-one mapping between data points. It is denoted as cDTW^{0}. An unconstrained DTW is denoted as cDTW^{100}. By definition, Euclidean distance is the upper bound, and the unconstrained DTW is the lower bound of the

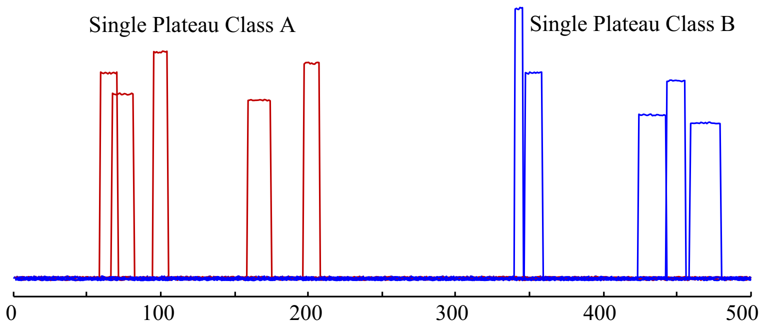


Fig. 5 Five examples of each class of the Single Plateau dataset (Class A and Class B)

constrained DTW (for any amount of constraint). Both bounds have been exploited by various clustering/classification algorithms and similarity search algorithms (Begum et al. 2015).

This review is necessarily brief; we refer the interested readers to other surveys (Ding et al. 2008; Shokoohi-Yekta et al. 2015) and the references therein for more details.

2.2 Factors affecting the best warping window

We note that the detailed discussion below of the factors affecting the best warping window for DTW classification are in the context of one-nearest neighbor classification only. Undoubtedly the other classifiers that use DTW distances [e.g. some variants of Shapelets and DTW embedding methods (Hayashi et al. 2005)] could also benefit from such a discussion. However, 1-NN classification is intuitive and well understood, and it accounts for the vast majority of work in this area (Bagnall et al. 2017; Bagnall and Lines 2014; Ding et al. 2008).

Before proceeding, we must ward off the common misconception that there is a fixed one-time domain dependent value of w. There is *no* single w value that is transferable across different contexts. To help illustrate this, we will create a synthetic dataset, which we call Single Plateau (SP). This dataset (and all others in this paper) is available at the paper supporting webpage (Supporting Page 2018). Each item in the dataset consists of a vector of 500 random numbers taken from a standard Gaussian. We add a “plateau” of height 100 and with a length randomly chosen in the range 5–20 to each exemplar. If the plateau’s location falls in the range of 1–250, it is in class A. If it is between 300 and 500, it is in class B. The plateau never appears in the middle of the time series; Fig. 5 shows examples from each class.

Note that while the SP dataset is synthetic, it closely models several real datasets, including yearly “snow-melt” time series, collect by the National Snow and Ice Data Center (NSIDC) in Boulder, Colorado and used as a critical resource for scientists studying climate change (Hu et al. 2014).

We will use this SP dataset as a running example to demonstrate factors affecting the choice of the maximum warping window width in DTW distance.

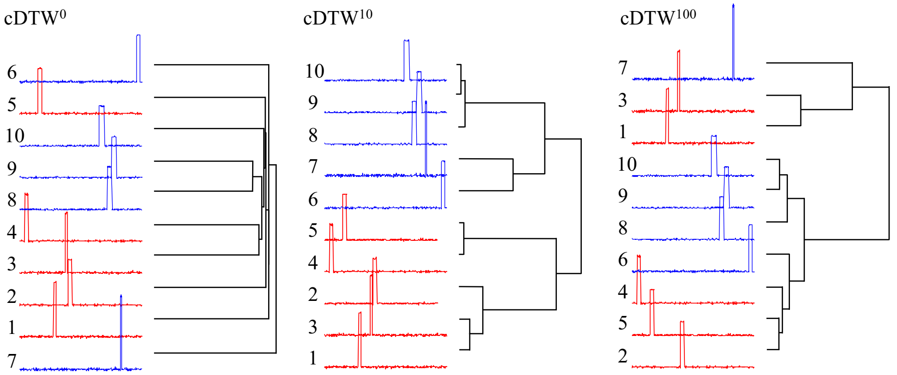


Fig. 6 A hierarchical clustering result for the SP dataset. Exemplars in Class A are numbered 1–5 and are shown in red. Exemplars in Class B are numbered 6–10, and are shown in blue. (Left) Clustering with cDTW^0, (middle) clustering with cDTW^{10}, (right) clustering with cDTW^{100} (Color figure online)

2.2.1 The intrinsic variability of the time axis

If we cluster SP with cDTW^0, we obtain a "random" clustering as shown in Fig. 6 left. This is not surprising, as this is clearly a dataset that needs a warping-invariant distance measure. If we re-cluster using cDTW^{10}, we obtain a clustering that correctly separates the two classes (in Fig. 6 center). Thus far, these observations coincide with most of the community's intuition. However, what happens when we cluster using cDTW^{100}? Again, we obtain a clustering that appears essentially random (Fig. 6 right).

This notion that "*a little warping is a good thing, but too much warping is a bad thing*" is known [although perhaps underappreciated (Ratanamahatana and Keogh 2005)] for time series *classification* (Chen et al. 2013); however, we believe that this is the first explicit demonstration of the effect for *clustering* (Figs. 13, 17 and 18 show examples for real datasets). Note that for classification, the luxury of labeled training data suggests a way to learn the appropriate amount of warping, a possibility we are denied in the unsupervised case of clustering.

This observation prevents us from considering a simple, though computationally expensive solution, which is just performing a clustering/classification under completely unconstrained warping.

2.2.2 The size of the dataset

It might also be imagined that we could discover the best warping window width for a given data *type* and just use that setting for all future datasets from the domain. For example, we might imagine that for the *gesture-recognition-for-tall-males* dataset, cDTW^5 is generally best, but for the *heartbeat-classification-for-the-elderly* dataset, cDTW^{13} is generally best.

However, we can dash such a hope with the following observation: the best value for w also depends on the *size* of the dataset. To see this, we can classify increasing large instances of the SP dataset. For each size, we search over all possible values of w

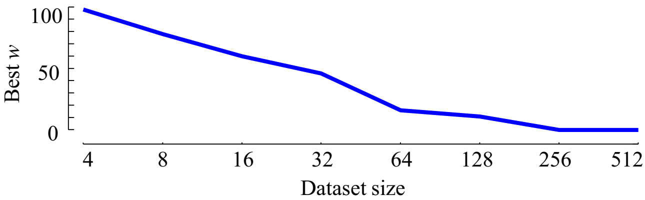


Fig. 7 Classification of increasingly large instances of Single Plateau shows the effect of dataset size on the best *w*

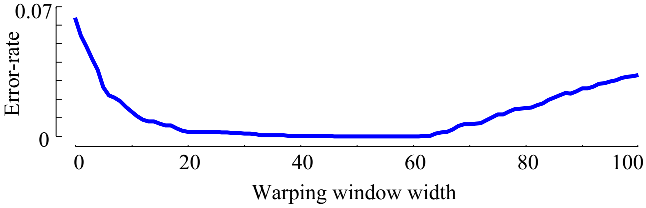


Fig. 8 Classification of 32-objects Single Plateau demonstrates effect of *w* on LOO error-rate. Average result of 100 runs

and record the value that minimizes the error-rate of LOO cross-validation. Figure 7 shows the result, averaged over 100 runs.

Consistent with observations in Ratanamahatana and Keogh (2005), small datasets tend to require much larger settings of *w* compared to larger ones. Note that this size versus the best curve for *w* is different for different datasets. Thus, we cannot generalize the best setting for *w* on one subset of a dataset to a different sized subset of the same dataset.

As shown in Fig. 7, the best value for *w* on this dataset, given that it contains 32 objects, is 46. Let us further consider this particular sized subset of the training set. Figure 8 displays the effect of *w* on the misclassification rate of the 32-objects SP dataset. We can see that allowing too much warping is almost as detrimental as too little warping.

In this case, the *w* versus error-rate curve has a broad flat valley, meaning that even if we choose a *w* value that is too large or small, we could still achieve low misclassification. However, as Fig. 8 suggests, this curve can take on more complex shapes, which makes the choice of *w* more critical.

2.2.3 The effect of the shapes of the time series

A good value for *w* depends not only on the intrinsic variability of the time axis and the size of the dataset, but it is also dependent upon the time series *shapes*. We can illustrate this latter point with a simple experiment. We created two near identical datasets, *Slim Plateau* and *Broad Plateau*, which, as their names suggest, differ only in the width of the plateaus (see Fig. 9). In both datasets, one class has a plateau in the first half, and the other class has a plateau in the second half.

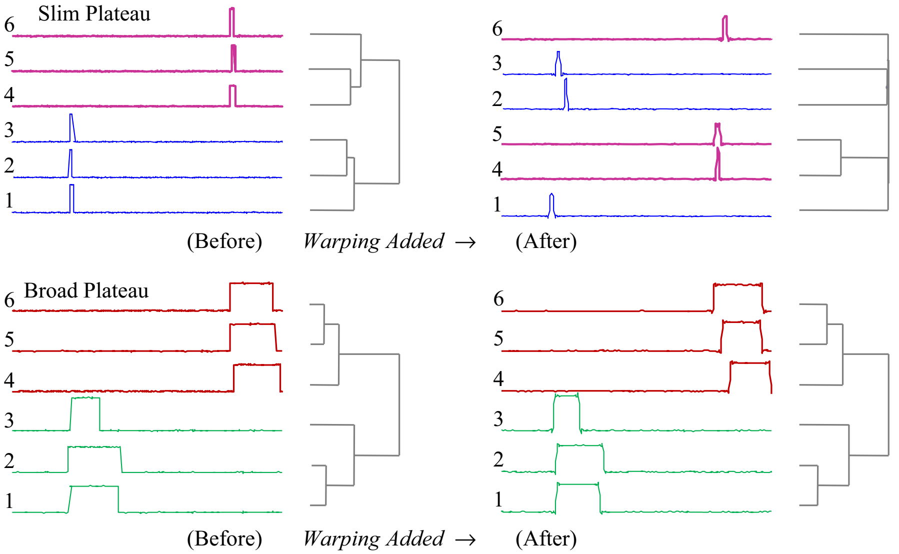


Fig. 9 Warping affects different datasets differently under hierarchical clustering. (Top) The clustering of the Slim Plateau dataset is very brittle when the time axis is warped. (Bottom) In contrast, the Broad Plateau dataset is extremely robust to *identical* levels of warping

As shown in the leftmost column of Fig. 9, we can see that both variants cluster well under cDTW^0 (i.e. Euclidean distance). What would happen if we added an *identical* amount of random warping to both datasets and clustered them again using cDTW^0? (We will explain how we can add synthetic warping to a time series in Sect. 4.1.8). As we can see in the rightmost column of Fig. 9, the clustering of *Slim Plateau* becomes essentially random, whereas *Broad Plateau* is basically unaffected.

The critical message from this experiment is as follows. In this pathological example, we can measure exactly how much warping there is in a dataset because we *placed* it there. But even in this case, we cannot use the amount of warping added to guide the choice of w. Even with a lot of warping in the time axis, the best value of w could still be as low as zero, depending on the time series shapes and the size of the dataset.

In summary, the best value of w depends on both the data size and the structure of the data. This fact bodes ill for any attempt to learn a fixed one-time domain independent value for it. There is not a single prototypical w versus error-rate curve for heartbeats or for gestures. We must learn this curve on a case-by-case basis, which is the argument of this paper.

2.3 Non-transferability of the best setting for *w* between supervised and unsupervised settings

In the introduction, we claimed that the best setting of w for *classification* is generally not an indicator of the best setting of w for *clustering*. Since this assumption has been explicitly made, but never formally tested multiple times in the literature (Paparrizos

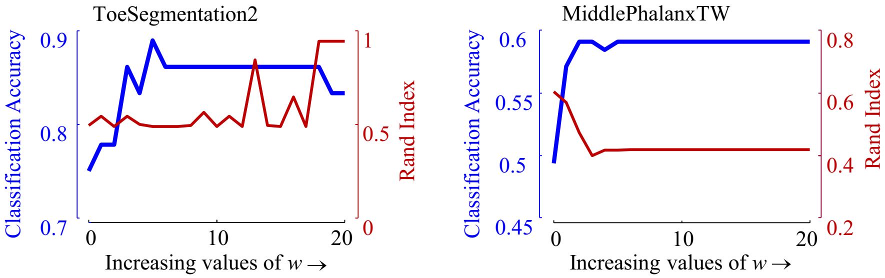


Fig. 10 The Rand-Index (red/fine) and the classification accuracy (blue/bold) versus the warping window width for two representative datasets (Color figure online)

and Gravano 2015), we will demonstrate that it is unwarranted. In Fig. 10, we show both the Rand-Index and the accuracy for two datasets.

In retrospect, it is unsurprising that these values are weakly related. For 1-NN classification (the most commonly used classification technique in the literature (Ding et al. 2008; Ratanamahatana and Keogh 2005), only the distance between the unlabeled exemplar and its *single* nearest neighbor matters. However, for clustering, the mutual distance among small *groups* of objects matters. This observation motivates us to learn the appropriate warping constraint for time series classification and clustering *independently*.

2.4 Classic learning of warping window size

The most popular method for learning the maximum warping window width for DTW-based time series classification is via cross-validation. In the case of the UCR Time Series Archive (Chen et al. 2015), the best value of *w* is determined by performing leave-one-out cross-validation with the 1-NN classifier on the training set over all warping window constraints possible, from 0 to 100% at 1% increments. The window size that maximizes *training* accuracy is selected, as it is expected to also give the best *testing* accuracy. The creators of the UCR Time Series Archive's disclaimer states that this may not be the best way to learn *w*, but it is simple, parameter-free, and works reasonably well in practice. We estimate that at least four hundred papers have used this approach or some variants of it (Bagnall et al. 2017; Ding et al. 2008; Górecki and Łuczak 2014; Jeong et al. 2011; Kate 2015; Lines and Bagnall 2015; Ratanamahatana and Keogh 2005), by either explicitly implementing the method, or directly comparing results to the numbers published in the UCR Archive (Chen et al. 2015).

For time series clustering, we do not have access to labeled data. Common practices involve using as large a value of *w* as the computation resources permit, directly applying the *w* learned from classification of the same problem domain and resorting to a fixed *w* value that is known to work reasonably well for most tasks. For example, in a recent highly cited recent paper, the authors noted, "*...we use as window 5%, for cDTW^5, and 10%, for cDTW^{10}, of the length of the time series of each dataset; this second case is more realistic for an unsupervised setting such as clustering*" (Paparrizos and Gravano 2015).

However, as our motivating examples demonstrate, these practices still require compromising the quality of clustering/classification. For many real-world problems, even a small increase in accuracy matters. To achieve the best possible performance, we need a more systematic approach to tailor *w* for individual tasks/datasets.

2.5 Summary of introductory material

Now that the importance of DTW's warping window width has been established, we are finally in a position to discuss our proposed methods for learning this parameter in the context of time series clustering (Sect. 3) and classification (Sect. 4). To ensure that all our experiments are easily reproducible, we have built a website that contains all data/code/raw spreadsheets for all the results (Supporting Page 2018).

3 Learning warping window width for time series clustering

3.1 Our approach

3.1.1 Introduction

We begin by formalizing the task at hand:

> **Problem Statement**: Given an unlabeled time series dataset D; find the value of *w* that maximizes the clustering quality. Where ties exist, report the smallest *w*.

There are many measures of *clustering quality*; however, measures based on sum-of-squared residual error do not allow for meaningful comparisons among clusterings with different values of *w*. Here, we wish to optimize the objective "correctness" of the clustering. Typically, we will not have access to this ground-truth (by definition); however, for the datasets we consider in this work, we do have class labels that allow us to do a post hoc analysis. Without loss of generality, we will use Rand-Index as the internal scoring function we optimize, and for the external post hoc analysis of the effectiveness of our ideas.

How can we choose the best value for *w* in the absence of class labels? One possibility is to use a semi-supervised clustering (Athitsos et al. 2008; Basu et al. 2002, 2004; Demiriz et al. 1999; Wagstaff and Cardie 2000). Here, we ask the user to annotate a fraction of the data (typically in the form of *must-link*/*cannot-link* constraints), and we attempt to exploit these annotations to guide the clustering algorithm.

One reason why semi-supervised clustering has not been as influential is its *inefficiency*. Suppose we have a mere 1415 items to cluster. This gives us just over one million *pairs* of time series we could ask the user to annotate. However, it may be that the vast majority of such annotations will be irrelevant, since all the clusterings in the search space *agree* (or all *disagree*) with a particular user annotation. Thus, to be sure that we get enough actionable annotations to guide the search in the clustering space, we must ask the user to annotate hundreds or thousands of objects. This is clearly undesirable as the user may be unwilling or unable to provide such an effort.

We introduce a novel semi-supervised clustering method for time series that does all the clustering *up-front* and only then asks for user input. This allows us to ask the user to annotate only informative pairs. Our proposed method offers the following advantages:

- Our approach is independent of the clustering algorithm. We are only learning the best *w* for a particular dataset; therefore, we can produce the final clustering using essentially[1] any partitional, hierarchical, spectral, or density-based clustering.
- The annotations are solicited *after* the clustering has been performed, meaning that we only ask the user to annotate pairs that matter. In contrast, almost all other semi-supervised clustering algorithms require the labels up-front, often asking the user to annotate pairs that will make no difference in all the clusterings considered. Thus, our algorithm is maximally respectful of the cost of human effort.
- Because the annotations are solicited after the clustering has been performed, our approach requires very few annotations; in many cases, as few as sixteen annotations can produce dramatic improvements.
- While we mostly envisage asking a human for annotation, in some situations, these annotations may be gleaned by examining side-information or statistical tests. Our framework can exploit this information.
- Our approach works for both single and multi-dimensional time series.
- Finally, as we shall demonstrate, our approach is highly accurate and robust to mistakes made by the annotator.

3.1.2 Semi-supervised learning

Due to its demonstrated utility in many practical applications, the semi-supervised learning paradigm (SSL) has drawn in significant attention in the data mining and machine learning communities over the last decade (Athitsos et al. 2008; Basu et al. 2002; Demiriz et al. 1999; Wagstaff and Cardie 2000). Existing methods for semi-supervised clustering are generally classified as *constraint-based* or *distance-based*.

Constraint-based methods rely on user-provided constraints to guide the algorithm toward a more accurate data partitioning. This can be accomplished in several (non-exclusive) ways:

- Enforcing constraints during the clustering process itself (Wagstaff and Cardie 2000). This requires modification of the clustering algorithm.
- Modifying the objective function for evaluating candidate clusterings and rewarding solutions that satisfy the most constraints. For example, Demiriz et al. (1999) modify the fitness function of a genetic search algorithm that optimizes clusterings.
- Seeding the clustering using the labeled examples to provide the initial seed clusters (Basu et al. 2002), mitigating the fact that some clustering algorithms are sensitive to the initialization.

In distance-based approaches, an off-the-shelf clustering algorithm is used; however, the underlying distance measure is trained to satisfy the given constraints. For

[1] "Essentially," since some clustering algorithms are not defined (or lose certain guarantees) for non-metric distance measures.

example, a weighted string-edit distance measure could be given the constraint that the words "bare" and "bore" *must-link*, but "bare" and "care" *cannot-link,* allowing the algorithm to suitably weigh the substitution cost in the edit distance lookup table to reflect the fact that while vowels are often confused, consonants are rarely confused (Bilenko and Mooney 2003).

Our proposed algorithm does not fit neatly into any of the categories above. First, our approach is completely agnostic to the clustering algorithm used. Second, we do not specify the constraints *before* the clusterings are performed, we only do so *after* the fact. This provides our approach with a significant advantage. If we ask the user to provide constraints before clustering, either by their choices, or randomly choosing pairs to be labeled, they may label objects of no utility. Specifically, they may label objects as *must-link*, which would have been linked by any clustering in our search space. Conversely, they may label objects as *cannot-link*, which never would have been linked by any clustering that our search algorithm would have considered. By waiting until after all the clustering has been performed, we can ensure that annotations we ask the user for are truly informative.

3.1.3 Clustering algorithm

At the risk of redundancy, again we emphasize that we are *not* proposing a clustering algorithm in this work. We are proposing a post hoc measure that enables us to score candidate clusterings created with different DTW parameters. Nevertheless, we must use *some* clustering algorithms. Without loss of generality, we use the TADPole algorithm of Begum et al. (2015), which is a specialization of the Density Peaks algorithm (Rodriguez and Laio 2014) for DTW. This algorithm is suited to DTW, since it does not require metric properties, and it is particularly amenable to optimizations to its scalability by exploiting both upper and lower bounds of DTW (Begum et al. 2015).

However, it is important to note that TADPole is just the clustering algorithm we use to predict w. Having done so, we could, in principle, use any clustering algorithm (partitional, hierarchical, spectral, or density-based clustering) with the newly-learned w. As it happens, the results using the TADPole algorithm are so good (Begum et al. 2015) that we do not consider this option for simplicity.

3.1.4 Clustering quality measure

We use Rand-Index as the internal scoring function we optimize, and for the external post hoc analysis of the effectiveness of our ideas. The Rand-Index penalizes both false positive and false negative decisions during clustering, and therefore it is not possible to optimize in a trivial way. There are some proposed variants, including the Adjusted Rand-Index (Vinh 2010); however, the classic Rand-Index (Rand 1971) is widely accepted and used. Moreover, at least internally, we are only interested in the *relative* improvements in clustering quality.

With Rand-Index, we assess clustering quality based on a series of decisions, one for each of the unique pair of objects in the dataset. A true positive (TP) decision means that we assign two similar objects to a same cluster. A true negative (NP) means we assign two dissimilar objects to different clusters. Similarly, a false positive

(FP) means that we assign two dissimilar objects to the same cluster. A false negative (FN) decision means that we assign two similar objects to different clusters. The Rand-Index is calculated as follow:

$$Rand\ Index = \frac{TP + TN}{TP + FP + FN + TN}$$

.

As Rand-Index measures the ratio of decisions that are correct, it is in the same spirit as accuracy in the context of classification; however, it is applicable even when class labels are not available. Rand-Index is always a number between 0 and 1; the higher is the better. Note that the Rand-Index penalizes false negatives and false positives equally, meaning grouping dissimilar objects in a same cluster is as bad as separating similar objects.

3.1.5 Choosing constraints

As we have noted above, the fact that we only need to see the constraints *after* the clusterings have been performed gives us a unique opportunity to optimize the user time and attention.

For every possible pair of time series in our dataset, we can build a *constraint vector* based on whether the pair are assigned to the same cluster or not (hereafter referred to as *linked* or *not-linked*). A candidate constraint be a binary vector C, whose length is the number of values of w under consideration. A '0' at the ith position in C indicates that the pair of time series was not linked under DTW^{i}, whereas a '1' indicates that it was linked.

In Fig. 11, we can see four candidate constraints. Constraint (A) is vacillating, and it is likely of little use to us. We can interpret it as being "volatile," since it constantly switches between linked and not-linked for different values of w. These constraints are rare and likely indicate a "hybrid" object on the cusp between two distinct clusters.

Constraints (B) and (C) are always/never linked, respectively. It is pointless to show such constraints to the user, since they "vote" equally for all values of w. In

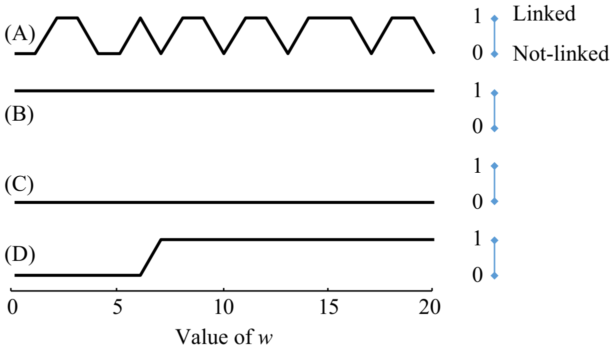


Fig. 11 Four representative constraints. (*A*) a vacillating constraint, (*B*) an *always* linked constraint, (*C*) a *never* linked constraint, (*D*) an ideal constraint

Table 1 Algorithm for finding the constraint set

```
    Input: set of candidate constraints, maximum number of constraints to
    get annotated
    Output: UA, the set of user annotations
 1  constraints ← sort_by(constraints, simplicity)
 2  index  ← 1
 3  while ¬empty(Constraints) AND loopCount < max
 4    $UA_{index}$ ← get_user_annotation(Constraints(index))
 5    answer ← get_user_willingness('Do Another? Y or N')
 6    if answer = 'Y'
 7        index  ←  index + 1
 8    else
 9        index  ←  infinity       // break out of loop
10    end
11  end
```

most datasets we consider, the majority (often the *vast* majority) of constraints are of these two types. With a little introspection, it is obvious that *most* constraints are non-volatile, as it suggests that most of the objects being clustered are in stable clusters. If *all* constraints were highly volatile, it would be difficult to select *any* clustering that is meaningful in any sense.

In contrast to the constraints above, constraint (D) seems to be an ideal constraint. For datasets that need warping invariance, it can be interpreted as: a value for *w* that is between zero to six is not enough, but anything seven or above works.

These observations inform our algorithm design. Constant constraints (types (B) and (C)) should be discarded. Of the remainder of the constraints, "simple" constraints are most likely to be informative. We can measure their simplicity by counting the number of sign changes as we "slide" across the vector. For constraint (A), this yields a value of 12, but for (D), the simplicity score is only 1.

$$\text{Simplicity}(C) = \sum_{k=0}^{(\max w)-1} 0, \quad \text{if } \mathrm{C}_k = \mathrm{C}_{k+1}, \quad \text{else} \;\; 1$$

Our algorithm for finding the set of constraints that we will ask the user to evaluate is presented in Table 1. We begin in line 1 by sorting the constraints with the simplest indicated first, and breaking ties randomly. At this point, we enter a loop, and while we have some constraints left to annotate, we have not reached our preset maximum limit, *and* the user is willing, we will show the two relevant time series to the user and get them to perform the *must-link*/*cannot-link* annotation.

Figure 12 illustrates examples of time series from the *Trace* dataset that are shown to the user. We hope to avail of the user's domain knowledge, intuitions, and pattern recognition ability. For Fig. 12 left, the user may realize that while the two time series are superficially different, most of the difference can be explained by warping the time axis. Therefore, we would expect the user to annotate this as "*must-link.*"

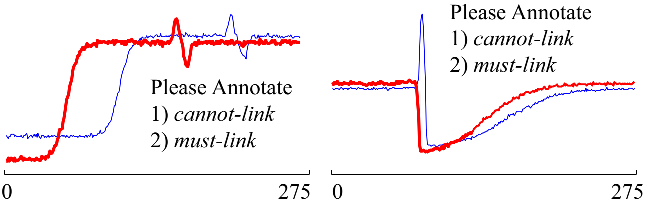


Fig. 12 Examples of pairs of time series from the *Trace* dataset presented to user for annotation. User has to decide whether the two time series should be in a same cluster or not. (Left) The correct label is *must-link* as most of the difference between the two time series is from warping in the time axis. (Right) Ideally, the user should choose *cannot-link* because one time series is missing the short peak that characterizes the other time series

In contrast, for Fig. 12 right, we hope the user would recognize that despite the similarity of the two time series (they have a relatively small Euclidean distance), one time series misses the short peak that seems to characterize the other sequence.

Naturally, we want our algorithm to be insensitive to occasional annotation mistakes. We consider this issue in Sect. 3.2.2. One helpful idea would be to add a third option "*skip this annotation*" to the list of possibilities offered. For simplicity, we ignore this possibility in this work.

Once we obtain the user annotation (UA), we can construct a prediction vector (PV) that tells us which w is most suitable. Note that this vector has little to do with the actual ground-truth Rand-Index vector, but it indicates the expected magnitude difference in Rand-Index (relative clustering quality) at each of the w values. The prediction vector value at index i (PV_i) is equal to the number of UA constraints satisfied (i.e. correctly clustered by our chosen clustering algorithm) over the total number of UA constraints.

$$PV_i = \frac{Number\ of\ UA\ constraints\ correctly\ clustered\ at w_i}{Total\ number\ of\ UA\ constraints}$$

We can see the incremental improvement (anytime algorithm) property of the algorithm by examining the predictions we make for w as we obtain more user annotations. Figure 13 shows an example of this for two datasets.

Note that in both cases, the "shape" of our prediction vector converges to the shape of the ground truth Rand-Index curve after sixteen user annotations. However, it is important to note that this is *not* necessary for our algorithm to be successful. All we require is that the prediction of the best setting for w concurs with the ground truth. Recall that this prediction is the location of the maximum value with ties broken by choosing the smallest w.

3.1.6 Pseudo user annotation

As the results in Fig. 13 suggest, and we will later confirm with an extensive empirical analysis, we can typically learn a good value for w with just a handful of user-interactions. Nevertheless, one might imagine that there are occasions where user annotations may be essentially impossible or especially expensive to obtain. Can we do anything in these situations?

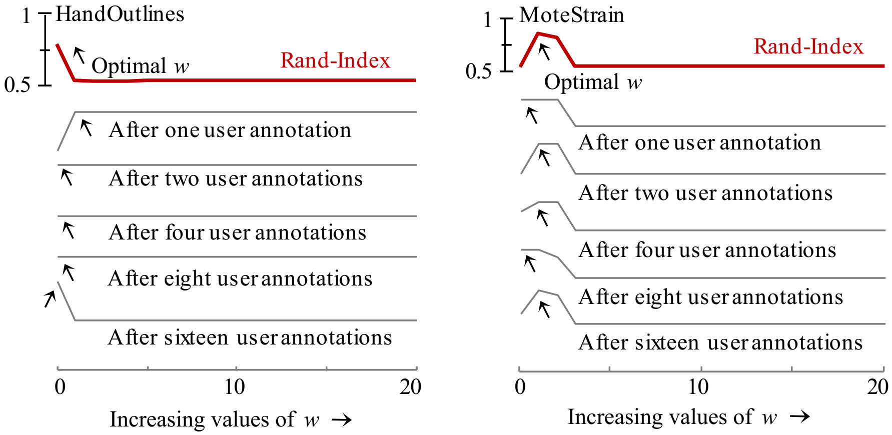


Fig. 13 For the two datasets *HandOutlines* and *MoteStrain*: The ground truth Rand-Index (colored/bold line). The prediction vectors (light/gray lines) learned after 1–16 user annotations allow us to estimate w (arrows). The shapes of the prediction vectors reflect the ratio of constraints satisfied (correctly linked or not linked) at each w (Color figure online)

A similar problem arises in information retrieval, where user feedback is known to improve the effectiveness of search, yet users are reluctant to give explicit feedback. The information retrieval community has addressed this by creating algorithms to give generated *pseudo-relevance feedback* automatically (Lv and Zhai 2010).

The ambition of these approaches is limited. No one claims that *pseudo*-relevance feedback is as useful as *real* human feedback. It suffices because it is better than doing nothing. In this spirit, we present a technique to learn w from pseudo annotations.

The basic idea is simple. Before we perform any clustering, we randomly sample objects from the dataset. For each object O, we create a copy of it that we denote as Ō. We add some warping to Ō, and place it into the dataset with the (pseudo) constraint *must-link*(O, Ō). Since we know that object Ō is just a minor variant of O, we can safely assume that if Ō occurred naturally, it would have been in the same cluster as O, and our *must-link* constraint was warranted. At this point, the list of "user annotations" is like those produced in Table 1.

This idea seems to have a tautological paradox to it. It seems that if we *add* w amount of warping to the dataset, we will *discover* w warping in that dataset. However, this is not the case, as discussed Sect. 2.2.3.

Table 2 outlines algorithm for generating pseudo constraints.

In line 1, we ensure that the data has an arbitrary structure in its ordering. In line 2, we enter a loop that replaces every second data object with a warped version of the data object that precedes it. Since these two objects differ only by the existence of some warping, we annotate them as '*must-link*'. Note that this algorithm produces a new dataset D*new,* which is the same size as D. This is important, as the size of the dataset affects the best setting for w (recall Sect. 2.2.2). The algorithm also outputs PUA, a set of pseudo annotations for D*new*. PUA is essentially identical to UA produced in Table 1 except its annotations are produced without human interventions. Note that with this method of generating annotations, all PUAs are *must-link* constraints.

Table 2 Algorithm for finding the pseudo constraint set

```
Input:  D, the dataset to be clustered
Input:  M, the amount of warping to add
Output: Dnew, a new version of dataset D
Output: PUA, the set of pseudo user annotations for Dnew
Dnew ← random_shuffle(D)
for i = 1 in steps of 2 to numberOfInstances(Dnew)
  Dnew$_{i+1}$ = add_random_warping(D$_i$)     // See Table 3
  PUA$_{(i+1)/2}$ = set_constraint(Dnew$_i$, Dnew$_{i+1}$, 'must-link')
end
```

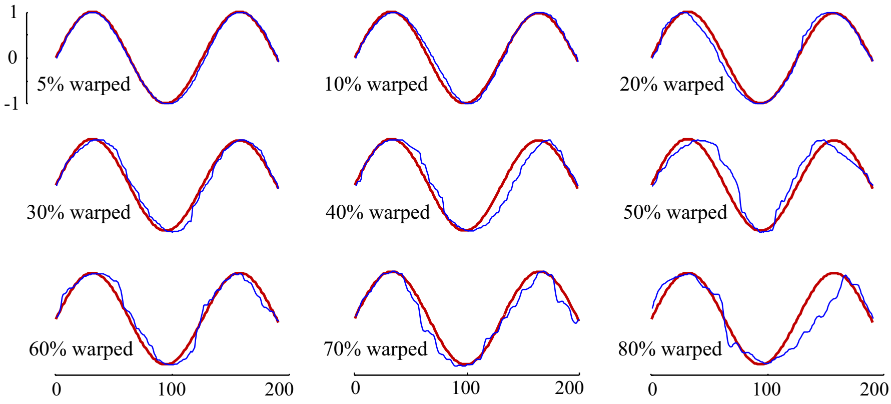


Fig. 14 From left to right, top to bottom: increasingly warped versions of a sine wave. The red/bold curve is the original, and the blue/fine curves are the ones with added warping. The "percentages" have no absolute interpretation; they only allow a relative understanding of the amount of warping added (Color figure online)

Table 3 Code to add warping to a time series

```
function [warped_T] = add_warping(T,p)
 i = randperm(length(T));
 i = sort(i(1:end-floor(length(T) * p)));
 warped_T = smooth(resample(T(i),length(T),length(i)),1);
end
```

Figure 14 shows some examples of synthetic time series with warping added, and for concreteness Table 3 contains the actual MATLAB code used to add warping. We call this variant of our ideas the Pseudo User Annotation (PUA) algorithm.

How well does this idea work compared to using true human annotations? The human annotations are constraints between two real data objects, which is undoubtedly advantageous. However, in most cases, we have only a fraction of D annotated this way. In contrast, every item in D*new* has an annotation, which provides this approach with an advantage if we choose to use them all. Figure 15 shows how this idea works with *Trace* and *Two Patterns*. Here we use 64 out of 1824 pseudo constraints available

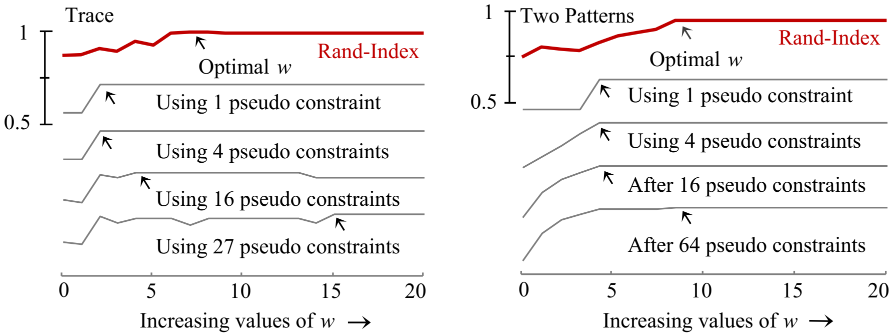


Fig. 15 *Trace* and *Two Patterns*' prediction vectors using pseudo constraints provided by the PUA algorithm

for *Two Patterns* to reach the correct value $w = 8$. Using all 27 constraints available for *Trace*, we arrive at $w = 15$, which gives a Rand-Index of 0.991 (the optimal is 1.0 at $w = 7$).

The reader may wonder how much warping we should use to obtain good pseudo constraints. The good news is that our PUA algorithm is quite robust to this parameter. In this example, we tried all possible warping amounts from 5 to 90% in 5% intervals. We found that for *Two Patterns*, any warping amount in the range 5–65% allows us to estimate the correct w.

3.1.7 Further reducing human effort

There are a handful of techniques we could use to reduce the number of annotations given by the user, and many of these ideas can be borrowed directly from the information retrieval community (Lv and Zhai 2010). For example, suppose the user decides {7,11} *must-link*, and that {11,27} *must-link*, then there is little point in asking their opinion on {7,27}, since they will also label this pair as *must-link* (by transitivity). We do not consider such optimizations here for brevity, because the simplest version of our ideas is already very competitive.

3.1.8 Related work

Zhou et al. (2015) recently introduced a paper entitled "*Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning*". However, the method is perhaps better seen as an *ensemble-based* method for time series clustering. The method has many parameters (at least four: α, β, p, w), and it is not clear how they affect the performance. They only test on twelve of the datasets we consider here, but in every case, they do not perform as well as our proposed approach. For example, for *Trace,* they obtain a best Normalized Mutual Information

(NMI)[2] score of 0.813, whereas, as we will show in Sect. 3.2, we can easily obtain a near-perfect NMI of 0.97.

Beyond this effort, we are not aware of any other work like our approach for semi-supervised learning for time series clustering. The general field of semi-supervised time series clustering is vast; we refer the interested reader to Rani and Sikka (2012) and the references therein. We further briefly review some of the most recent, high-visibility efforts in time series clustering in Sect. 3.2.4 before the direct empirical comparisons to our proposed algorithm.

3.2 Empirical evaluation of using prediction vector for setting *w* for time series clustering

At the risk of redundancy, we restate that we are *not* introducing a new clustering algorithm, merely proposing a technique to learn *w* because this parameter critically affects the quality of clusterings. Nevertheless, in Sect. 3.2.4, we explicitly compare TADPole by using the learned warping window to five recent state-of-the-art clustering algorithms.

3.2.1 Preliminary tests

We denote our algorithm as cDTW^{ss} (DTW Semi-Supervised). We compare to two rivals by clustering with cDTW^0 (Euclidean distance) and clustering with cDTW^{10}. These rival methods account for virtually everything in the literature. For example, Ding et al. (2015) uses cDTW^0, and Paparrizos and Gravano (2015, 2017) use cDTW^{10}. A surprisingly large number of papers neglect to explicitly state what value of *w* they used.

It is important to state that the *only* difference between our approach and the two rival methods is the access to the labeled constraints. Otherwise, the underlying clustering algorithm, TADPole (Begum et al. 2015), is identical for all approaches and completely deterministic (Rodriguez and Laio 2014). Thus, any improvements obtained can be completely attributed solely to our ideas.

We can measure *success* as follows. For each dataset, we compute the maximum Rand-Index obtainable under any setting of *w* from 0 to 20 [as our result shows, and in concurrence with the literature, most datasets in the UCR Archive do not require *w* greater than 10% (Ratanamahatana and Keogh 2005)]. For example, in Fig. 1, the maximum Rand-Index is 1.0 for *Two Patterns* and 0.89 for *Swedish Leaf*. Then, we can compute a score, the ratio of the Rand-Index achieved by an approach over this optimal achievable value. The closer this ratio is to 1.0, the better; we call an approach a *success* if its score is 0.99 or higher.

We begin by considering the utility of our approach if given only sixteen labels; this is about the amount a person can annotate in 1 min. We summarize the result in Table 4. With sixteen labeled constraints, we achieve success of 46 out of 102 datasets, with

[2] NMI is an information-theoretic interpretation of clustering quality. It has values in range 0 and 1, the higher the better.

Table 4 Summary of number of successes on 102 datasets of cDTW^0 (DTW with $w = 0$ a.k.a. Euclidean distance), cDTW^{10} (DTW with $w = 10$) and cDTW^{ss} (DTW semi-supervised, our method)

	CDTW^0	CDTW^{10}	CDTW^{ss}
16 annotations	34 out of 102	31 out of 102	46 out of 102
32 annotations	34 out of 102	31 out of 102	50 out of 102

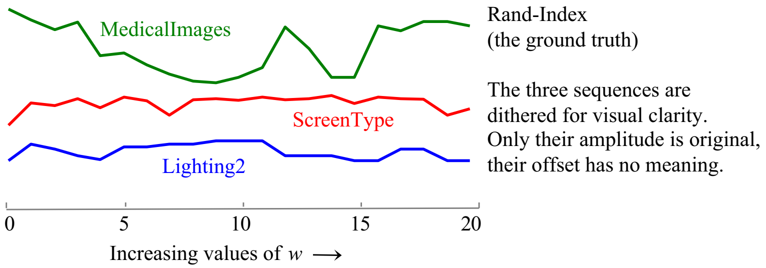


Fig. 16 The Rand-Index versus the warping window width for three small datasets. Contrast the variability of the curves with the relatively smooth curves shown in Fig. 1

cDTW^0 and cDTW^{10} achieving 34 and 31, respectively. If we double the number of constraints to thirty-two, we extend our success to 50 datasets. Recall that thirty-two annotations require only a few minutes of user effort, and they typically represent less than 0.0001% of the labeled pairs.

Despite this significant improvement over the state-of-the-art, it is natural to wonder about the cases we did not score within 0.99 of the optimal. In some cases, we *just* missed out. For example, using thirty-two constraints on the *TwoLeadECG*, *Cricket_Y*, *NonInvasiveFatalECG_2*, and *50words* datasets, we were within at least 0.98 of the optimal.

However, in some cases, we do achieve significantly worse than the optimal. Essentially, all such cases can be attributed to very small datasets (or small, relative to the number of clusters). As shown in Fig. 16, this tends to result in clusterings that are very unstable with small changes in w. The fact that small datasets have poor stability when clustered is well known (Von Luxburg 2010), and the issue is orthogonal to our contributions. We speculate that if the best value of w is poorly defined and unstable, it may be impossible for any algorithm to learn it. Nevertheless, even in such datasets, we do not do worse than the lower scoring of our two rivals.

3.2.2 *Robustness to incorrect constraints*

The experiments in the previous section assume that all the constraints the user provided are correct. However, this assumption may be unwarranted in many circumstances. Our annotator may indicate that two items *cannot-link* when they are in the same class, and really *must-link*, or vice versa. To investigate the robustness of our approach, we revisit some of the experiments above, but this time, we randomly make some of the constraints incorrect.

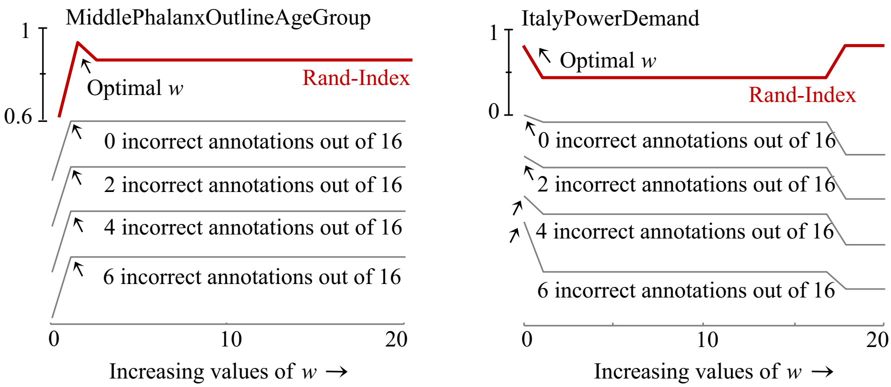


Fig. 17 Robustness to incorrect constraints. In each case, 16 pairs of time series are presented for annotation. The annotator may incorrectly label a pair that should have been *must-link* as *cannot-link* and vice versa. Our algorithm is robust to these mistakes

As shown in Fig. 17, for the *ItalyPowerDemand* and *MiddlePhalanxOutlineAgeGroup* dataset, we can achieve near perfect results even if a fraction of the constraints is incorrect. Among the 16 pairs of time series chosen for annotation, we single out the *must-link* pairs and randomly change the label of some pairs from this list to *cannot-link*. Then, we observe the mean best w predicted averaged over 10 runs. We find that it is consistently 0 for the *ItalyPowerDemand* dataset and 1 for the *MiddlePhalanxOutineAgeGroup*, which concurs with the objective ground truth.

As a practical matter, any system used to garner user feedback should allow three choices to the user, *cannot-link*, *must-link* and *I-don't-know*, which would further enhance robustness by giving the user a chance to simply skip over the difficult or ambiguous case.

3.2.3 Handling the multi-dimensional case

Thus far, we have considered only single dimensional time series; however, the proliferation of sensors from sources such as wearable devices indicates that there is increasing interest in multi-dimensional time series data (Shokoohi-Yekta et al. 2015). Fortunately, there is nothing in our approach that makes any assumption about dimensionality, so we can immediately apply our ideas to the multi-dimensional case. A recent paper notes that there are (at least) two ways that DTW can be generalized to the multi-dimensional case, for simplicity, we use DTW_I (Shokoohi-Yekta et al. 2015), which allows each dimension to warp independently. Let Q and C be two multi-dimensional time series of M dimensions. DTW_I defines their DTW distance as the sum of independent DTW distances between each dimension.

$$DTW_I(Q, C) = \sum_{m=1}^{M} DTW(Q_m, C_m)$$

.

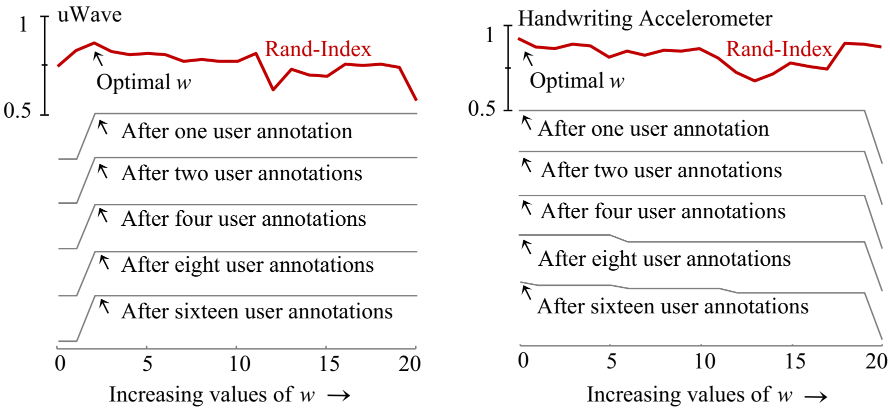


Fig. 18 Three-dimensional *uWave* and *Handwriting Accelerometer* dataset clustered with DTW_I

In Fig. 18, we consider the 4480-objects, three-dimensional *UWave* dataset (Liu et al. 2009), which has become a benchmark for gesture recognition in the last 5 years. We also consider the *Handwriting Accelerometer* dataset using all three of the available accelerometer channel readings. Even though all dimensions are not necessary for this task, we only wish to illustrate that our algorithm can correctly predict a good value for w.

While there are just over one million possible pairwise constraints, our algorithm can find the optimal w with only sixteen annotations. Note that here, the amount of warping is critical. Too much or too little warping yields poor results. This fact might explain the puzzlingly diversity of accuracy claims made for this dataset in the literature. Unfortunately, most papers do not explicitly state the value of w used, but the three most common settings, cDTW^0, cDTW^{10}, and cDTW^{100} are all suboptimal to widely differing degrees.

3.2.4 Comparison to rival methods

In this section, we have two related aims. The first is to compare our methods to other clustering methods in the literature (despite not introducing a new clustering algorithm). Our second aim is higher-level. We wish to demonstrate that finding a good value for w generally produces improvements that dwarf all other choices, including the choice of the clustering algorithm.

Concretely, in this section, we offer some evidence to support the following claim:

> The effect of choosing the correct value of w is critical, and it generally dwarfs any effect of the choice of the clustering algorithm.

This can also be stated as:

> Any discussion of the "best" clustering algorithm for time series is premature, unless the best value of w has been decided.

Because some published research has claimed improvements in creating a clustering algorithm, or in designing an alternative distance measure, which has only provided slight improvements demonstrated in accuracy, this claim is important. We believe that in many cases, a better (but not necessarily *best*) choice of w would have radically changed the outcome in favor of DTW with any "off-the-shelf" clustering algorithm. Our claim somewhat contradicts recent claims such as "*… the choice of algorithm… is as critical as the choice of distance measure*" (Paparrizos and Gravano 2015). We reiterate that we are only offering *some* evidence to support this claim. A more forceful demonstration (that is rigorously fair to all cited works) would require more space than is available here.

In a recent work (Paparrizos and Gravano 2015, 2017), the authors introduce k-Shape, a system that combines a novel time series-clustering algorithm and a novel distance measure named SBD (Shape-Based Distance), which are designed to work together. They perform an extraordinary comprehensive empirical comparison of the proposed method with all the major clustering algorithms and distance measures. For DTW, they *do* recognize that the value of w can make a difference; they compare two possibilities (cDTW^{5} and cDTW^{10}) and conclude that "*SBD is a very competitive distance measure… and achieves similar results to both constraint and unconstraint versions of DTW*."

However, simply choosing a better value of w offers improvements that dwarf the claimed improvements of the SDB algorithm. For example, for the *Trace* dataset, they compare five clustering algorithms using DTW versus the same five clustering algorithms using SBD. The former achieves Rand-Index values of {0.87, 0.75, 0.75, 0.83, 0.77}, and the latter achieves {0.87, 0.87, 0.87, 0.83, 0.87}, suggesting an advantage for SBD. However, using the exact same split of the *Trace* data, we can beat *all* these approaches significantly without any human intervention, as our PUA algorithm can achieve a 0.99 Rand-Index.

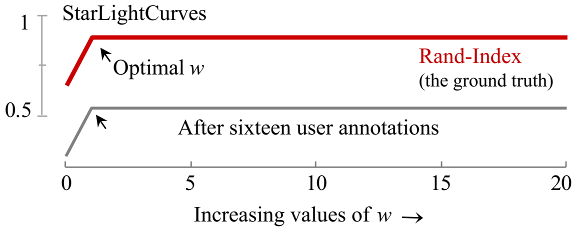
Similarly, we have a large margin of improvements for *Two Patterns*. For example, Paparrizos and Gravano (2015) has the DTW-based algorithms achieving Rand-Index values of {0.87, 0.59, 0.62, 0.97, 0.65}, and SBD variants achieving {0.25, 0.54, 0.64, 0.67, 0.66}, but PUA learns that cDTW^{8} is the best setting and achieves a perfect 1.0.

In a publication of ICML 2011 (Li and Prakash 2011), the authors introduce a clustering method called Complex-valued Linear Dynamical Systems (CLDS) and claim that the "*approach produces significant improvement in clustering quality, 1.5–5 times better than well-known competitors on real motion capture sequences.*" The method involves several layers of complicated sub-procedures, so we refer the interested readers to the original paper. The authors demonstrate the utility of their work on the publicly available *MOCAPANG-Subject-35*, *right-foot-marker* dataset. The evaluation method is based on the conditional entropy,[3] and they score 0.1015, while cDTW^{100} using K-Means scores significantly worse at 0.4229, which is about the same as random guessing.

In revisiting this experiment, we noted that the authors acknowledge that "*the original motion sequences have different lengths; we trim them with equal duration.*"

[3] For conditional entropy, smaller is better.

Fig. 19 The Rand-Index versus the warping window width for *StarLightCurves*. We predict $w = 1$, obtaining a Rand-Index of 0.83, equivalent to a NMI of 0.79



However, it is important to note that this manipulation is only needed for their proposed method; cDTW can handle sequences of unequal lengths. When we re-ran the experiments, we found that cDTW^{20} has a perfect conditional entropy of 0 when using K-Means. TADPole achieves the same superior score for any w from 11 to 20. As before, the correct value of w makes a difference; for example, if forced to use cDTW^{10}, TADPole scores a slightly worse 0.142.

To be clear, we are not claiming the work proposed by Li and Prakash (2011) is without merit. We are simply demonstrating that when using any reasonable choice for w with an off-the-shelf clustering method, cDTW can be a very competitive method for the datasets the original authors used to validate their method.

A recently published work measures the accuracy of eleven carefully optimized clustering algorithms on the *Trace* dataset, of which eight use DTW as the distance measure (Ferreira and Zhao 2016). The Rand-Index of these methods are {0.87, 0.76, 0.86, 0.86, 0.91, 0.86, 0.86, 0.87, 0.87, 0.84, 0.75}. However, as noted above, using the exact same split of *Trace*, we can beat all these approaches without any human intervention, as our PUA algorithm can achieve a Rand-Index of 0.99.

Another recently published time series clustering technique called YADING is shown to "*provide theoretical proof which… guarantees YADING's high performance*" (Ding et al. 2015). However, these guarantees are only with respect to Euclidean distance. The only publicly available real dataset they test on is *StarLightCurves*, for which they obtain a Normalized Mutual Information (NMI) score of 0.60. However, as shown in Fig. 19, with 16 constraints given by the user, we find cDTW^1 to be a good choice and achieve a significantly better NMI of 0.79 (omitted for brevity: in fact, any number of constraints above four also works this well).

Why did the authors of this paper dismiss DTW as a distance measure? They noted that DTW "*is one order of magnitude slower than calculating [Euclidean distance],*" and further noted that it only took them a brief 3.1 s to cluster this dataset. However, this dataset took several years to collect, and many days of careful human effort in preprocessing. Given that, the difference between taking 3.1 s and taking 30 s to do the clustering seems completely inconsequential (but also see Sect. 3.2.5). Of course, the authors are correct in noting that there is sometimes a need for better speed and scalability. However, in many domains, the tradeoff between speed and accuracy will still favor accuracy. For example, in the UCR Archive, many datasets took hours, days, or weeks to collect (*InsectWingbeatSound*, *ElectricDevices*, *Fish*, *Phoneme*, etc.), so the few minutes needed to cluster them is negligible if we can improve accuracy.

Finally, a paper in AAAI tests four algorithms for time series clustering; two are based on DTW (Zhong et al. 2016). These algorithms yield NMI scores of {0.53, 0.45, 0.54, 0.64} for the *Trace* dataset, but our PUA algorithm can achieve an almost perfect NMI score of 0.97 (Rand-Index = 0.99) on this same dataset.

These five examples strongly support our claim. Finding a good value for w (using our method, or *any* method) can produce improvements that make almost all other changes inconsequential.

3.2.5 Scalability

At first, our algorithm appears to require a significant overhead in time complexity, given that the Density Peaks algorithm (Rodriguez and Laio 2014) requires O(n^2) calculations of cDTW, and we need to run this algorithm twenty-one times (for each warping window from 0 to 20). However, this is a pessimistic view. To begin with, note that we use the TADPole version of the algorithm, which is a specialization of the Density Peaks algorithm for DTW that exploits the fact that we can compute tight upper and lower bounds for cDTW^w for any value of w and use these bounds to prune off many computations. The TADPole algorithm is admissible, and it can prune 90%-plus of the cDTW calculations.

In fact, we can improve upon this. Instead of performing twenty-one independent clusterings, we can exploit the fact that for any two time series Q and C, the value of cDTW^w(Q,C) is a very tight lower bound for the value of cDTW^{w+1}(Q,C). Thus, we can perform the clusterings in order, from $w = 0$ to $w = 20$, at each stage by using any cDTW^w calculations as lower bounds in the next level. Thus, the time overhead for our ideas is only slightly more than a single highly optimized clustering. Even to the must-calculated DTW that remains after the lower-bound pruning procedure, we can still apply the work of Silva et al. (2018) to dismiss unpromising alignments.

Finally, we note that there are a wide variety of DTW implementations, and the efficiency differences between them overshadow the small overhead of our approach. For example, a recently published paper that tests a DTW-based clustering on some of the datasets we consider, and it notes "*several experiments were unable to return results within 20 days*" (Zhong et al. 2016). However, we can cluster these same datasets in at most minutes, at least 10,000 times faster.

3.3 Case study: gesture-based identification

We present a case study in the context of gesture-based identification. The goal is to identify/authenticate users based on loosely defined gestures such as "picking-up" or "shaking" a handheld device (Guna et al. 2012). Such a gesture-based identification system can be well suited for personalized applications that only target a small group of users and are not security critical. User login for home sharing Netflix is an example.

The dataset was kindly shared by the authors of the paper (Guna et al. 2012), whose preliminary experiment results show the feasibility of implicit gesture-based user identification. The subset that we use is available for download on our supporting webpage (Supporting Page 2018). The dataset consists of an accelerometer recording of 10 subjects; each performs a "shake" gesture 10 times with a Nintendo Wii Mote remote controller. The users are instructed to shake the control device in no predefined

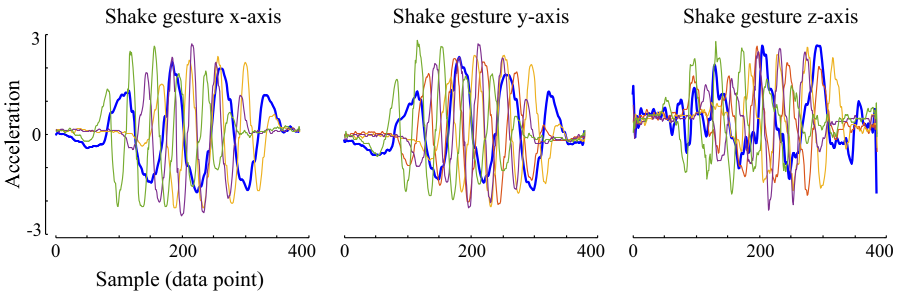


Fig. 20 Five examples of a shake gesture captured with x, y, and z-axis acceleration. Time series of same color correspond to one specific instance. One instance is highlighted for visual clarity (blue/bold time series). The sampling rate is 100 Hz (Color figure online)

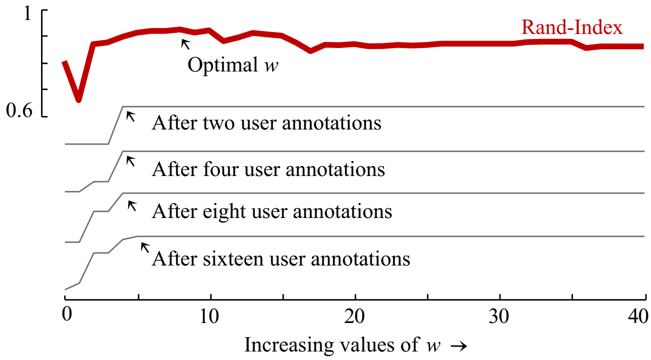


Fig. 21 Clustering result with TADPole (red/bold) and prediction vectors (grey/thin). With 16 user annotations, the algorithm suggests $w = 5$, which gives a Rand-Index of 0.91, being 99% of optimal (Color figure online)

way, just as they would normally do in their everyday life. Figure 20 displays some instances of a shake gesture with acceleration measured in three axes. For simplicity, we only use an x-axis reading for the results presented in Fig. 21.

We resample all the gesture occurrences, so they have a uniform length of 385, which is the length of the longest occurrence recorded. Instead of performing user classification as in the original paper, we are interested in clustering this dataset to see how well each time series cluster characterizes an individual user. Figure 21 top displays the Rand-Index if we have access to the true label. It shows that the highest clustering quality for this dataset is 0.92 at $w = 8$. Using a $w = 0$ yields a much poorer Rand-Index of only 0.82. By applying our method to learn w, we will eventually learn that $w = 5$ is the best, and it gives a Rand-Index of 0.91 (Fig. 21 bottom). We count this a success, because the achieved Rand-Index scores 0.99 of the optimal Rand-Index.

In retrospect, this is clearly a dataset that would benefit from warping invariance. Although the chosen gesture is identical for all users, there exists a subtle systematic variation in how it is performed by each individual, which explains the good clustering result. For instances contributed by a particular subject, there may be shifting in the time axis that a small amount of warping can account for. In this case, a suitable choice of w can make a significant difference in the final cluster assignment.

4 Learning warping window width for time series classification

4.1 Our approach

4.1.1 Introduction

We begin by formalizing the task at hand:

> **Problem Statement**: Given a labeled time series training set D; find the value of *w* that maximizes the classification quality on an unlabeled test set. Where ties exist, report the smallest *w*.

We evaluate the classification quality by the measure of accuracy. Maximizing accuracy means minimizing the classification error-rate. Readers may argue that some other performance measures, such as the F-measure, are more suited. The F-measure penalizes false positive and false negative equally, making it a fairer metric for unbalanced datasets. However, the uneven distribution of classes is not an issue here, since all the datasets we consider are stratified sampling. We are more interested in learning the appropriate value of *w* for the maximal classification accuracy of the test set, given that we only have limited training examples to learn from.

Since there is a growing consensus that the DTW-based *k*-NN (NN-DTW) is a strong baseline for time series classification, we use it as the underlying classification algorithm. This concurrence stems from the fact that time series classification has a universally used collection of benchmark datasets (Chen et al. 2015). There are now many independent comprehensive empirical studies demonstrating a strong performance of NN-DTW (Bagnall and Lines 2014; Ding et al. 2008; Lines and Bagnall 2015). Nevertheless, in recent years, there have been many proposed algorithms that are able to improve upon NN-DTW's accuracy in the general case. Recent papers note that many claims do not hold under rigorous statistical evaluations: "*Based on experiments on 77 problems, we conclude that 1-NN with Euclidean distance is fairly easy to beat but 1-NN with DTW is not*" (Bagnall and Lines 2014) or "*the received wisdom is that DTW is hard to beat*" (Bagnall et al. 2017).

We will show that it is possible to learn *w* more robustly; this is particularly useful when the training data is limited. Our approach is based on resampling the training data. Resampling is normally ill advised in small datasets, where using only a subset of the data compounds all the problems inherent with working with limited data. However, we can address this issue by replacing the non-sampled data with synthetic replacements. Our idea is simple, making it very amendable to existing time series classification tools, but as we will show, the performance improvements it allows are statistically significant.

4.1.2 DTW-based 1-NN classification

The nearest neighbor classifier (NN) works intuitively. It assigns an unseen object to the class of its closest neighbor in the feature space. The general algorithm is referred to as *k*-NN, in which *k* is the number of nearest neighbors under consideration. In the case of 1-NN, the new object is automatically assigned the class label of its nearest neighbor, breaking ties randomly. For k greater than 1, the majority vote is applied. The NN classifier is unique in that there is no explicit model built during training. A new object is simply classified by comparing itself to all the other objects in the training set.

The warping constraint has a direct effect on the k-NN classifier outcome. Kurbalija et al. (2014) study the impact of global constraints on the four most widely used elastic distance measures: DTW, LCS, ERP, and EDR (they note that DTW is the most accurate overall by a wide margin). They test different values of the *Sakoe–Chiba* band and observe how this parameter affects the number of time series changing their nearest neighbors in comparison with the unconstrained case. They found that among the distance measures considered, DTW is the most sensitive to the setting of w. The nearest neighbors of time series objects tend to remain stable for w greater than 15 but change significantly for smaller w values.

Geler et al. (2014) study the effectiveness of the k-NN classifier in relationship to the w values. They found that if the k-NN have equal votes, then the best w value grows as k grows. However, if we use a weighing scheme that favors the first nearest neighbor, then the best w remains approximately similar for different k settings. They argue that such weighing schemes significantly improve the k-NN classifier accuracy. In the absence of a weighing scheme, the k-NN classifier gives the highest accuracy for $k = 1$.

Most practitioners who adopt 1-NN do so for its simplicity, i.e., requiring no parameter tuning. The research focus has thus shifted to improving the distance measure used. 1-NN using DTW has emerged as the new benchmark for many time series classification tasks. This practice of using 1-NN-DTW is supported by a recent survey in time series classification: "*When using a NN classifier with DTW on a new problem, we would advise that it is not particularly important to set k through cross validation, but that setting the warping window size is worthwhile*" (Bagnall and Lines 2014). The importance of setting the right w for DTW is acknowledged here and in a handful of other places in the literature. Nevertheless, we argue that it is under-examined, given that the potential for the improvements that it offers seems to equal the improvements gained at the expense of more complex methods.

4.1.3 Classification quality measure

We evaluate the classification quality by the measure of accuracy. Interchangeably, we sometimes report the classification error-rate as maximizing accuracy means minimizing the classification error-rate (the sum of accuracy and error-rate is 100%). Accuracy measures the proportion of true results among the total number of cases examined, multiplied by 100 to turn it into a percentage. A true positive (TP) or true negative (TN) means that the correct label agrees with the classifier's label.

False positive (FP) refers to the number of negative examples labeled as positive. False negative (FN) refers to the number of positive examples labeled as negative. Accuracy is calculated as follow:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$

Note that accuracy may not be a good classification measure in the presence of imbalanced data. In that case, a classifier that blindly assign all objects with either negative label or positive label will have a high accuracy, even though it is practically useless. However, it is not a problem here because we assume stratified sampling of train data.

In assessing the quality of the classifier during the training phase, it is common to use k-fold cross-validation error-rate. The final error-rate is the average of all error-rates from training on each $(k-1)$ folds and testing on the remaining fold.

4.1.4 Making synthetic data

The idea of making synthetic data to improve classification is not new, but it has been limited to the two-class problem. For example, it has been used to address the problem of class imbalance, in which one class dominates the other (Batista et al. 2004; Chawla et al. 2002; He et al. 2008). Synthetic exemplars of the minority class are added to create a more balanced training set, hence mitigating the tendency for the classifier to be biased towards the majority class. Although oversampling techniques have been used in time series classification with class imbalance (Cao et al. 2013), creating synthetic time series data mining under DTW has only been explored recently (Forestier et al. 2017; Petitjean et al. 2015).

A recent paper also generates synthetic exemplars by adding warping to existing objects (Guennec et al. 2016). However, this is to mitigate convolutional neural network (CNN) weakness "*…that they need a lot of training data to be efficient*" (Guennec et al. 2016). The synthetic examples *do* help improve accuracy over the non-augmented datasets; still, it remains unclear if CNNs are generally competitive for time series problems (Bagnall and Lines 2014), and this issue is orthogonal to the claims of this work.

4.1.5 An intuition to our proposed approach

To understand the effect(s) of dataset size on the most suitable warping window width, we performed the following experiment. We begin with a simple experiment that determines whether what we hope to achieve is possible, and it also offers intuition on how to achieve it. Consider the *Two Patterns* dataset. Because it has 1000 training objects, we will denote it as *Two Patterns*^{1000}. As shown in Fig. 22 left, *Two Patterns*^{1000} is a dataset in which we can correctly learn the best maximum warping window with cross-validation.

Suppose the dataset had significantly fewer training instances; we will call this dataset *Two Patterns*^{20}. We would expect that the holdout error-rate would increase, and we were advised by Ratanamahatana et al. (2005) that we should expect the best value for w to go up slightly. As we can see in Fig. 22 right, these both occur. However, the most visually jarring observation we make is that we have lost the ability to correctly predict the best value for w, as the training error oscillates wildly as we vary this parameter. In fact, Fig. 22 right strongly resembles some of the plots shown in Fig. 3, and for the same reason, we do not have enough training data.

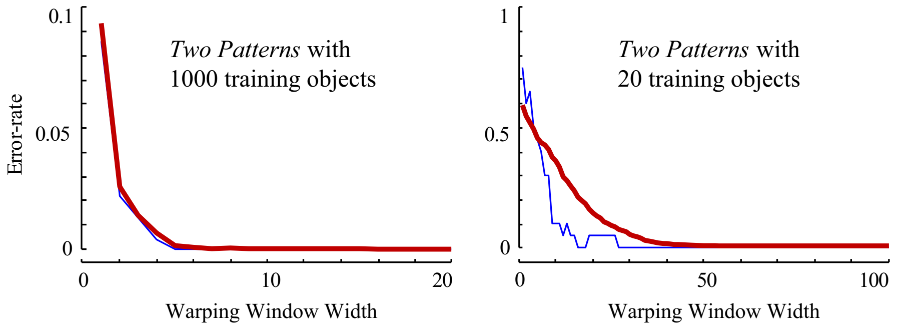


Fig. 22 The LOO error-rate (blue/thin) and the holdout error-rate (red/bold) for increasing values of w. (Left) *Two Patterns*^{1000} dataset, (right) *Two Patterns*^{20} dataset (Color figure online)

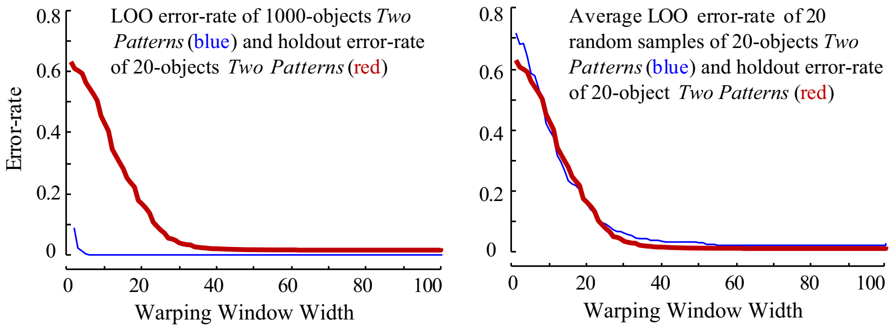


Fig. 23 (Left) The LOO error-rate of the *Two Patterns*^{1000} dataset is a poor predictor of the holdout error on *Two Patterns*^{20}. (Right) In contrast, the average LOO error-rate of 20 random samples of *Two Patterns*^{20} is an excellent predictor of the holdout error on *Two Patterns*^{20}

Let us further suppose that while we are condemned to using *Two Patterns*^{20} to classify new instances, we have one thousand more labeled instances at our disposal. One might ask: if we have more labeled examples, why do we not use them in the training set? Perhaps the time available at classification time is only enough to compare twenty instances.

Clearly, we do not want to use all one thousand labeled instances to learn the best value for w, because, as shown in Fig. 23 left, we will learn the best value of w for *Two Patterns*^{1000}, not for *Two Patterns*^{20}, which is our interest.

The solution suggests itself. Performing cross-validation with *Two Patterns*^{1000} gives us low variance, but it is biased toward the wrong value of w. In contrast, doing cross-validation with *Two Patterns*^{20} is biased toward the correct value for w but has high variance. If we resample many subsets of size twenty from *Two Patterns*^{1000}, do cross-validation on each, and average the resulting w versus error-rate curves, we expect that this average mirrors the curve for the test error-rate and therefore predicts a good value for w. As we can see in Fig. 23 right, this is exactly the case.

The observations above seem to be non-actionable. In general, we do not have 1000 spare objects to resample from. Our key insight is that we can *synthetically generate* plausible training exemplars. We can use these synthetic objects to resample from,

Table 5 Algorithm for making augmented training set

```
    Input:  D, the original training set with n objects
    Input:  M, the amount of warping to add
    Input:  R, the ratio of synthetic objects to create
    Output: Dnew, a new version of dataset D
1   realObjects ← random_sample(D,(1-R)*n objects)
2   fakeObjects ← random_sample(D,(R*n) objects)
3   for i ← 1:1:numberOfInstances(fakeObjects)
4    fakeObjects$_i$ ← add_warping(fakeObjects$_i$,M)
5   end
6   Dnew ← [realObjects;fakeObjects]
```

make as many new instances of the training set as we wish, and learn the best setting for *w*.

Note that this task is easier than it seems. We do not need to produce synthetic exemplars that are perfect in every way or even visually resemble the true objects to the human eye. It is sufficient to create synthetic objects that have the same properties with regards to the best setting for *w*. In the next section, we show our strategy for generating an arbitrary number of such instances.

4.1.6 Our algorithm

We can finally explain our algorithm, which can be tersely summarized as follows:

> Make *N* copies of the original training set. For each copy, replace a fraction of the data with synthetically generated data and perform cross-validation to learn the error-rate versus *w* curve. Use the average of all *N* curves to predict *w*.

This algorithm, outlined in Table 6, contains a subroutine presented in Table 5. Individual elements are motivated and explained in following subsections. The essence of the method we are proposing is in making *N* new training sets by using the algorithm in Table 5. These datasets will be used instead of the original training set to learn *w*. While each dataset may produce a noisy error versus *w* curve (as in Fig. 22 right), the average of all such curves will be smoother, and it will more closely resemble the true noisy error versus *w* curve (as in Fig. 23 right).

As shown in Table 5, we begin in line 1 by randomly sampling portion of the original training objects with replacement. These objects will be included in the new training set and will be unmodified. After that, we randomly sample a portion of the original training set again. These objects are then distorted by adding a warping and are appended to the new training set. Using this algorithm, the new training sets have the same number of objects as the original set. Note that the sampling is performed in a stratified manner; otherwise, when working with small datasets, we run the risk of only adding warping to one class and possibly skewing the results.

The sub-routine of making new training set is invoked over a number of iterations, as shown in line 2 of the main algorithm in Table 6. For each new training set, we run cross-validation to compute the classification error-rate at each setting of the maximum warping width allowed from 0% (Euclidean distance) to 100% (unconstrained DTW),

Table 6 Algorithm for finding the warping window width

```
   Input:  D, the original training set
   Output: w, the predicted best warping window
1  for i ← 1:1:numberOfIterations
2   Dnew ← make_new_train_set(D)          // See Table 4
3   for j ← 1:1:maximumWarpingWindow
4    errorRate$_{i,j}$ ← run_cross_validation(Dnew)
5   end
6  end
7  meanOfAllIterations ← mean(errorRate)
8  [minValue, minIndex] ← min(meanOfAllIterations)
9  w ← minIndex - 1
```

in steps of 1%. Finally, we calculate the mean error-rate of all runs in line 7 and obtain the index of the minimum error-rate. The learned w in line 9 is this index minus one since the item at the first index corresponds to $w = 0$.

4.1.7 Generation of new training set

The new training set has the same size as the original training set, but only a portion of the real objects are retained, and a portion of synthetic objects added. The ratio of real/synthetic objects is 0.2/0.8. This ratio is based on an intuition, which is explained in Sect. 4.1.9 and verified empirically.

4.1.8 Adding warping to make new time series

We add warping to a time series in the same manner as we presented previously in the context of learning w for time series clustering, where we showed how to make pseudo user annotations (Sect. 3.1.6, Table 3). We nonlinearly shrink a time series to a smaller length by randomly removing data points and then linearly stretching the down-sampled time series back to its original size. However, we incorporate a small modification to account for possible "endpoint effects" introduced by the resampling process. Figure 24 illustrates how a time series is transformed into its warped version.

We add extra "paddings" at the beginning and end of the down-sampled time series by repeating its endpoint/start point values ten times. These paddings are removed from the final time series later (Fig. 24 middle). It is important to note, as a recent work indicates, that the endpoints can result in misleading DTW distance (Silva et al. 2017). Recall that DTW's constraints require it to match the pairs of beginning and end points, even though they may be a poor match. The MATLAB code to add warping in Table 7 contains a small modification of Table 3 to reflect these changes. Even though "without padding" still brings about reasonably good results, findings from our experiments presented in Sect. 4.2 confirm that "adding padding" improves the performance.

It may be possible to further improve our overall method if we find better ways to make more "natural" synthetic exemplars. We have experimented with several methods

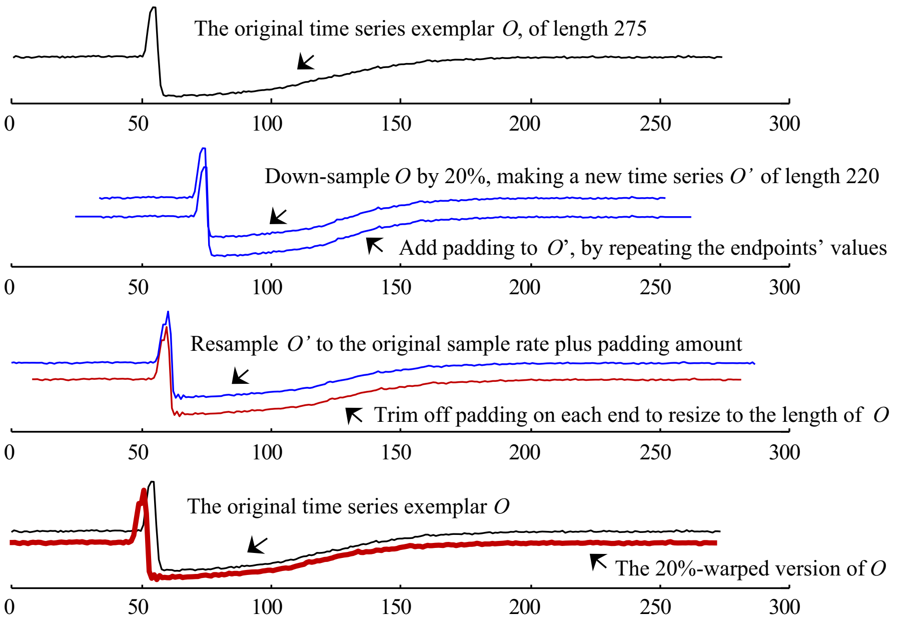


Fig. 24 Adding 20% warping to an exemplar of *Trace*. Note that in the bottom panel, the generated times series (bold/red) is a slightly warped version of the original time series (fine/black) (Color figure online)

Table 7 Code to add warping to a time series

```
function [warped_T] = add_warping(T,p)
 i = randperm(length(T));
 t = T(sort(i(1:end-length(T) * p)));
 t = [repmat(t(1),1,10), t, repmat(t(end),1,10)];
 warped_T = resample(t,length(T) + 20, length(t));
 warped_T = warped_T(11:end - 10);
end
```

to generate synthetic time series (Chawla et al. 2002; Forestier et al. 2017; Petitjean et al. 2015). Interested readers can find more details on the paper's supporting webpage (Supporting Page 2018). In brief, there are *dozens* of methods to produce synthetic examples (averaging, grafting, perturbing etc.), and many of these ideas work well (Esteban et al. 2017). We chose the method shown in Table 7, because it is simple and effective.

4.1.9 On the parameter setting

Readers will have observed that we have three parameters to set. The first is the amount of warping we use to make new data objects, which we will refer to as *synthetic* objects. The second is the ratio between real and synthetic objects in the newly constructed training set, which we will refer to as an *augmented* training set. The third is the number

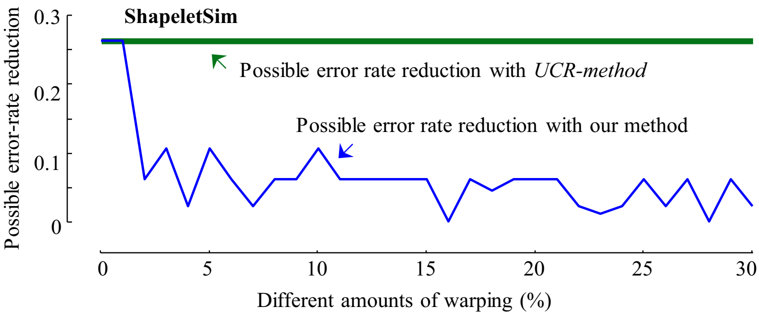


Fig. 25 Effect of the warping amount on the possible error-rate reduction. The vertical axis shows the difference between the error-rate achieved by the *w* learned and the error-rate achieved by the best *w* for this dataset. Possible error-rate reduction is synonymous with room for improvement. Adding warping helps if the blue/fine line is below the green/bold line (Color figure online)

of iterations we would like to repeat the process, i.e., the number of augmented training sets to generate.

At first glance, the idea of adding warping seems to have a tautological air to it. It seems that the amount of warping we will *discover* is the amount of warping we *added*. However, this is not the case. Empirically, we have discovered that if we do not add enough warping, our algorithm will fail, that if we add *exactly* the correct amount it will work well, and if we add too much, it will *still* work well. Given this, we should clearly err on the side of adding more warping.

To demonstrate this, we consider the *ShapeletSim* dataset. We ran our algorithm (Table 5) on this dataset multiple times, changing only M, the amount of warping added to make synthetic exemplars (Table 5 line 4) from 0 to 30%. The lowest error achievable for this size subset of *ShapeletSim* is 26.11%, and the error-rate obtained with the baseline method is 52%. As shown in Fig. 25, adding too little warping hinders our ability to predict the best w, but once we have added at least 15% warping, we learn a setting for w that gives us the lowest error-rate.

Given this observation, for simplicity, we hardcode the amount of warping to 20% for all experiments in this work.

Similarly, different synthetic/real object ratios for the augmented training set can produce different results. However, there is a single value, 0.8 that produces successful results on most datasets. Making the majority of objects in the newly constructed training set synthetic yields more diversity and variance in each training cycle. This value is hardcoded for all experiments presented in this work.

Finally, the parameter N, the number of new (partly synthetic) training sets needs to be determined. This is a simple parameter to set; the more the better, but the gain comes with diminishing returns. N is hardcoded to a conservative 10 for all experiments presented in this work.

Using hardcoded settings for all the datasets in the UCR Archive is an opportunity cost; an adaptive approach *could* be better. However, our strategy guards against overfitting. Moreover, we see this work as proof that a more robust learning of w is possible, and it is not the final word on the matter.

4.1.10 Why tenfold cross-validation

Leave-one-out (LOO) is a common variant of cross-validation (CV) to tune the parameters, and it is the method used by the UCR Time Series Archive to learn the warping window size. LOO has the advantage that no data is wasted. However, as noted in Ng (1997), LOO can be more susceptible to over-fitting. This is because the models trained in each iteration are only slightly different (since the training set differs in only one object each time). Moreover, the entire purpose of creating *N* training sets to learn *w* is to increase the variance of the results. LOO is deterministic, but (with shuffling) *K*-fold CV (when *K* is less than size of training set) is not. On the other hand, if we set $K = 2$, we are learning from a dataset that is only half the size of the dataset we have. As we explained in Sect. 4.1.5, for small training sets, this is likely to result in learning a pessimistic value of *w*, which is much too large. Given these two constraints, we propose to use tenfold CV throughout this work. It provides a good tradeoff between low-variance LOO and the biased-to-large-*w* twofold CV.

Finally, it may appear that performing 10 repetitions of tenfold CV will be computationally expensive. However, recall that the datasets in question are small by definition. Additionally, we can accelerate the entire process by embedding the current state-of-the-art DTW lower bounding and early abandoning techniques. Even without these techniques, our entire learning algorithm only takes 23 min for *Gun_Point*, given that we perform 100 iterations for all *w* from 0 to 100. Note that we can further reduce this time by choosing to perform fewer iterations and a narrower *w* range. Our experiment results demonstrate that even 10 iterations offer statistically significant improvement over the baseline method, and the best *w* for a dataset, regardless of its size, does not exceed 60. This applies for datasets discussed in Sect. 4.2.1, which are framed around small training set problem, not the original UCR splits.

4.1.11 Related work

The more general idea of creating synthetic data to mitigate the problems of imbalanced datasets (Chawla et al. 2002) or to learn a distance measure (Ha and Bunke 1997) is well-known. However, we are not aware of any other research suggesting a window size for improving DTW-based classification. We suspect that the dearth of study on this important problem is likely due to the community's lack of appreciation of the importance of *w* setting.

4.2 Empirical evaluation of the resampling method to learn *w* for time series classification

4.2.1 Datasets

We use the UCR Time Series datasets for our experiments (Chen et al. 2015). As of February 2018, the UCR Time Series Archive has 85 datasets from various domains, has served as the benchmark for the time series community, and is widely referenced

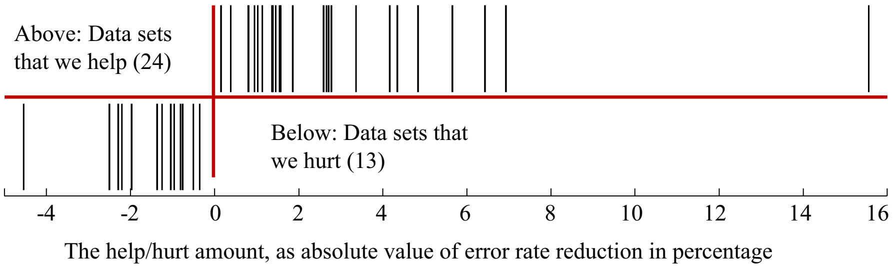


Fig. 26 Help/hurt amount. The number of datasets that we *help* is nearly twice the number of datasets that we *hurt*

in the literature. A more comprehensive version of the UCR Archive together with classification results of different algorithms is hosted by Bagnall et al. (2018).

As we have demonstrated in Fig. 22 left, our ability to learn w depends on the amount of training data. With enough data, the simple baseline method is effective, and we have little to offer. The ideas proposed in this work are most useful for smaller datasets. Some of the train/test splits in the UCR datasets have large enough training sets that our ideas do not offer any advantages. Rather than ignoring these datasets, we will recast them to a smaller uniform size.

We merge the original train and test set together, then randomly sample ten objects per class for training. The remaining objects are used for testing. As three datasets do not have enough ten objects per class, we exclude them from the experiment (the excluded are: *OliveOil*, *50words* and *Phoneme*). Therefore, we are left with 82 datasets. These new splits are published in the paper supporting webpage (Supporting Page 2018) for reproducibility. Note that with these new splits, the training sets all have equal class distribution. However, this distribution may not be true for the test set.

4.2.2 *Performance evaluation*

We compare our method to the standard practice of learning w via cross-validation on the train set. Specifically, we implement the tenfold cross-validation with 1-NN classifier variant. For concreteness, we refer to this as the baseline method.

Using the algorithm in Table 6 to learn the warping window size, we classified the holdout test data on the training set with 1-NN. Figures 26 and 27 show a visual summary of the results. Perceptibly, our method wins more often and by larger margins. We can summarize this in several ways.

We call our proposed method a *success* if it can reduce error-rate in absolute value by at least 0.5% (i.e., we round the error-rate to two decimal places) compared to the baseline method. We call it a *failure* if our method increases the error-rate by more than 0.5%. If the newly learned w results in test error-rate that is less than 1% different from the test error-rate obtained by the traditional method, we consider our method *neutral*. This can happen in two ways. Our method suggests the same value of w as the baseline method, or it recommends a different value of w, which offers similar accuracy.

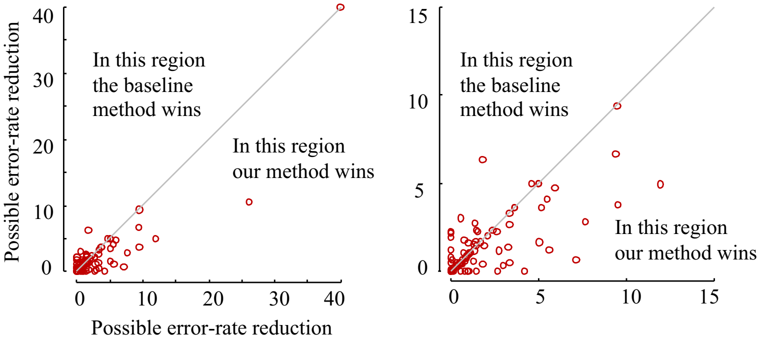


Fig. 27 Possible error-rate reduction (how close a method's error-rate to the optimal error-rate is) of the baseline method and our proposed method

Given this nomenclature, we can say that of the 82 datasets tested, our method improves classification accuracy of twenty-four, with an average improvement of 3.2%, and decreases the accuracy on only thirteen with a smaller average of 1.6%. This statement can in turn be visualized with the linear plot in Fig. 26.

Another way to demonstrate how our proposed method outperforms the traditional method is to look at the possible room for improvement, which is the difference between the error-rate achieved by the learned w and the error-rate of the best w of a dataset (found by exhaustive search). The smaller the difference, the better the method is. This is illustrated in Fig. 27.

While the results are visually compelling, we turn to statistical tests to ensure that the superiority of our method is statistically significant. Both the paired-sample t test and the one-sided Wilcoxon signed rank test confirm that our method is better than the baseline method at the 5% significance level. Details are available on our website (Supporting Page 2018).

4.2.3 *On time complexity*

It is important to clarify that we are optimizing the classification accuracy in trade-off for speed. However, we are only compromising *training* time here. The test time is not affected. Instead of running cross-validation one time as the baseline method, we would need to do that multiple times and average the results of these independent runs. So, if we decide to use ten iterations, the time it takes to learn the right w will be ten times slower than the traditional method (the resampling and adding warping to construct a new training set is linear and inconsequential).

Once the correct setting for w has been learned, we can readily use it for testing. This use of multiple random samples might seem like a computational burden but recall that many datasets in the UCR Archive took days, weeks or even months to collect, so spending a few more seconds or minutes on training the model to improve classification accuracy is well worth the relatively small increase in computational effort. Moreover, recent works such as FastWWSearch (Tan et al. 2018), which exploits various novel lower bounds and pruning strategies, has dramatically reduced the time to search for the best w from training data of NN-DTW. FastWWSearch offers at least one order of magnitude and up to 1000× speed-up than the state-of-the-art (Rakthanmanon et al. 2012). Such algorithms can augment our method.

4.2.4 Beating other algorithms with the UCR splits

As we noted, our contributions are focused on the case in which we have a small training set. The "small training set" problem setting is a common situation. For example, it was used in the "cold-start" learning of gestures for controlling a wearable device (Valsamis et al. 2017). Nevertheless, it is interesting to ask if our algorithm can improve upon the original UCR Archive's train/test splits. The answer is "*yes, at least sometimes*." In most cases, for the larger train splits the baseline method is effective, as in the examples in Fig. 2. However, in several cases, our method does significantly improve on the baseline method, and it even improves on many of the methods that claim to improve upon that strong baseline.

For example, we mentioned that Deng et al. (2013) can reduce the error-rate of *Gun_Point* to 4.7%, but our method suggests $w = 5$, which yields an error-rate of only 3.3%. Similarly, Górecki and Łuczak (2013) can lower error-rate of *Lightning2* and *Lightning7* to 13.1 and 32.9%, but our method can achieve an error-rate of only 8.2 and 29%, respectively. All these improvements are solely from optimizing the maximum warping window width of DTW.

4.3 Case study: fall classification

We conduct a case study in the context of fall classification. We do not claim any expertise in this domain, and we only have a superficial idea of how the data was collected. This is *exactly* the purpose of this case study. We wish to demonstrate that our ideas can be easily applied to any dataset/domain with minimum effort and show the potential for significant gains in accuracy. The accuracy may be improved by several other (mostly orthogonal) methods; for example, by carefully truncating data (Silva et al. 2017), averaging exemplars (Petitjean et al. 2015), and discarding data (Xi et al. 2006). However, we believe our method offers an unusually large "bang-for-the-buck."

Falls are a common source of injury among the elderly. A fall generally has few consequences for the young, but it can lead to fatal consequences to the elderly. According to the US Centers for Disease Control and Prevention, in the USA alone, an older adult is hospitalized due to a fall every 11 s, with one such individual succumbing to their injuries every 19 min. The total cost of fall injuries mounted to $34 billion in 2013 in the US alone (National Council on Aging 2016). The *type* of fall is highly predictive of the extent of the injuries that the victim sustains (Masters 2016). Thus, knowing the cause or manner of a fall may assist timely and relevant medical intervention post-fall, as well as help prevent more fall in the future.

The dataset we consider was kindly shared by Albert et al. (2012). It was collected with a built-in phone accelerometer, which was attached the volunteer subjects' lumbar by a belt strap, positioned such that the accelerometer x, y, and z axes were directed

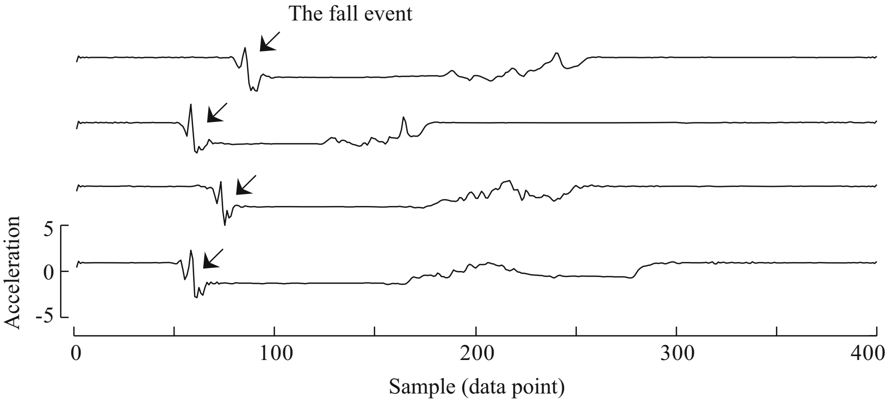
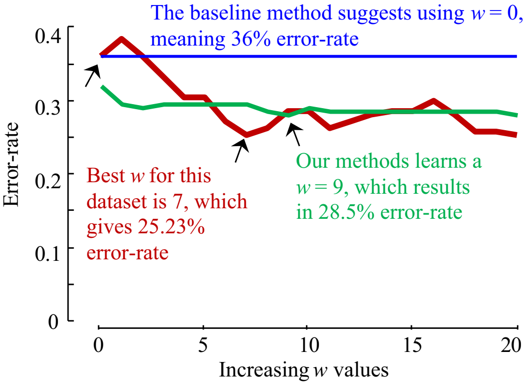


Fig. 28 Four instances of a trip and fall event captured in the x-axis acceleration

Fig. 29 Error-rate of fall classification. The indices of lowest values indicate the best *w*. Our method to learn *w* obtains an 7.5% error-rate reduction compared to the baseline method



upward, left, and behind the subject, respectively. All falls were carried out onto a pad in a controlled lab environment.

We only consider a small subset of the data and only the x-axis acceleration to demonstrate the utility of our method. Each example in our dataset is 400 data points long, representing a 20 s fall event at the sampling rate of 20 Hz (we re-sample the subsequences of uniform length if some are slighter shorter or longer). Figure 28 displays four examples of a trip fall. The data can be considered weakly labeled. The fall does not span the entire 20 s session, but it can be shifted in the time axis by an arbitrary amount ("arbitrary" to us, as we did not collect the data). Visual inspection suggests that this dataset needs warping invariance, and our algorithm helps determine the appropriate amount of warping to allow, as shown in Fig. 29.

Our task is to classify falls into one of two classes: forward orientation (trip and fall) or backward orientation (slip and fall). We randomly sample the data to construct a train set of 20 objects and a test set of 214 objects. Stimulated falls come from five different individuals. We perform stratified sampling, so the number of slip falls, and trip falls are equal, and the contributions from each subject are the same. This training set resembles the classic "cold start" problem. We restrict the train set to 10 objects per class only. Given the data comes from five different people, who possess unique physiques and gaits, we only have two samples of each individual to learn from.

The baseline method leads us to use Euclidean distance ($w = 0$), which gives a classification accuracy of only 64%. However, our method suggests $w = 9$, reducing the error-rate from 36% to only 28.5%. The best warping window width for this dataset is $w = 7$, which corresponds to a 25.23% error-rate. This result is less impressive than the one published in (Albert et al. 2012), but note that we intentionally frame our problem around limited cross-subject training data and we perform classification using only a single dimension.

5 Conclusion

In this work, we have shown that w, the maximum amount of warping allowed by DTW, is a critical parameter for both the classification and clustering of time series under the DTW distance. For most datasets, if this parameter is set poorly, then nothing else matters; it is impossible to produce high-quality results. In many cases, a more careful setting of the value of w can close most or the entire performance gap gained by other more complicated algorithms recently proposed in the literature.

For clustering, we have further proposed the first semi-supervised technique designed to discover the best value for w. Our approach is unique since human involvement is not required up-front as it is in other semi-supervised clustering algorithms. Instead, we seek user annotations *after* the clustering process, and we devise a scoring scheme to ask for only the labels that really matter. This gives our algorithm the desirable anytime algorithm property.

We have also forcefully demonstrated that the choice of warping window width w is critical for accurate DTW-based nearest neighbor classification of time series and proposed a resampling method to learn w in this context. Our method is parameter-free (or equivalently, we hardcoded all parameters). However, experimenting with adaptive parameters may allow others to improve upon our results.

We have tested our algorithms on more than one hundred datasets from diverse domains, showing that it offers statistically significant improvements. We note that the ideas we have proposed are *very simple*. This is not an accident. We hope that the reader sees this simplicity as the strength it is intended to be, not as a weakness; simple ideas are more likely to be widely adopted and widely used.

Our paper has several other observations that are novel, or at least underappreciated. We have shown that w depends not only on the data object shapes, but also on the number data objects considered. This observation has been made for classification before (Ratanamahatana and Keogh 2005), but not for clustering. We have shown that the optimal setting for w for *classification* is not generally the optimal setting for *clustering*, an assumption that has appeared in the literature (Paparrizos and Gravano 2015). Finally, in the last decade, a handful of researchers have argued that warping constraints are not necessary, and that there are "*cases where unconstrained warping is useful*" (Shou et al. 2005), or that research should "*focus on unconstrained DTW*" (Athitsos et al. 2008). While the absence of evidence is not evidence of absence, the extensive nature of our experiments, which failed to find a *single* dataset which requires a value of *w* greater than 20 for either clustering and classification of the UCR Time Series Archive data, suggests that these efforts are likely to be fruitless.

Future work includes a more theoretical treatment of the issues at hand and determining if the basic framework can be extended to other distance measures with tunable parameter(s) (Beecks et al. 2010; Assent et al. 2006; Lee et al. 2008; Vlachoset al. 2002). Finally, we have released all our code and data in a public repository (Supporting Page 2018), to allow others to confirm, extend, and exploit our ideas.

Acknowledgements This material is based upon work supported by the Air Force Office of Scientific Research, Asian Office of Aerospace Research and Development (AOARD) under award number FA2386-16-1-4023. The Australian Research Council under grant DE170100037 and the UK Engineering and Physical Sciences Research Council (EPSRC) under grant number EP/M015807/1 have also supported this work. Finally, we acknowledge the funding from NSF IIS-1161997 II and NSF IIS-1510741. We also wish to take this opportunity to thank the donors of the data to the UCR Time Series Archive.

References

Albert MV, Kording K, Herrmann M, Jayaraman A (2012) Fall classification by machine learning using mobile phones. PLoS ONE 7(5):e36556. https://doi.org/10.1371/journal.pone.0036556

Assent I, Wichterich M, Seidl T (2006) Adaptable distance functions for similarity-based multimedia retrieval. Datenbank Spektrum 19:23–31

Athitsos V, Papapetrou P, Potamias M, Kollios G, Gunopulos D (2008) Approximate embedding-based subsequence matching of time series. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data. ACM, pp 365–378

Bagnall A, Lines J (2014) An experimental evaluation of nearest neighbour time series classification. arXiv Preprint arXiv:1406.4757

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. https://doi.org/10.1007/s10618-016-0483-9

Bagnall A, Lines J, Vickers W, Keogh E (2018) The UEA and UCR time series classification repository. www.timeseriesclassification.com

Basu S, Banerjee A, Mooney R (2002) Semi-supervised clustering by seeding. In: Proceedings of the 19th international conference on machine learning (ICML-2002), pp 19–26

Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. Int Conf Knowl Discov Data Min (KDD). https://doi.org/10.1145/1014052.1014062

Batista GE, Prati RC, Monard MC (2004) A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor Newsl Spec Issue Learn Imbalanced Datasets 6(1):20–29. https://doi.org/10.1145/1007730.1007735

Beecks C, Uysal MS, Seidl T (2010) Signature quadratic form distance. In: Proceedings of the ACM international conference on image and video retrieval. ACM, pp 438–445

Begum N, Ulanova L, Wang J, Keogh E (2015) Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining—KDD'15, pp 49–58. https://doi.org/10.1145/2783258.2783286

Bilenko M, Mooney RJ (2003) Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining—KDD'03, p 39. https://doi.org/10.1145/956755.956759

Cao H, Li XL, Woon DYK, Ng SK (2013) Integrated oversampling for imbalanced time series classification. IEEE Trans Knowl Data Eng 25(12):2809–2822. https://doi.org/10.1109/TKDE.2013.37

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357. https://doi.org/10.1613/jair.953

Chen Y, Hu B, Keogh E, Batista GE (2013) “DTW-D: time series semi-supervised learning from a single example. In: KDD &apos;13: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 383–391. https://doi.org/10.1145/2487575.2487633

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 HA (2018) Supporting page 2018. http://www.cs.ucr.edu/~hdau001/learn_dtw_parameter/

Dau HA, Begum N, Keogh E (2016) Semi-supervision dramatically improves time series clustering under dynamic time warping. In: 25th ACM international conference on information and knowledge management, pp 999–1008. https://doi.org/10.1145/2983323.2983855

Dau HA, Silva DF, Petitjean F, Forestier G, Bagnall A, Keogh E (2017) Judicious setting of dynamic time warping's window width allows more accurate classification of time series. In: IEEE international conference on big data

Demiriz A, Bennett KP, Embrechts MJ (1999) Semi-supervised clustering using genetic algorithms. In: Artificial neural networks in engineering (ANNIE-99), pp 809–814

Deng H, Runger G, Tuv E, Vladimir M (2013) A time series forest for classification and feature extraction. Inf Sci 239:142–153. https://doi.org/10.1016/j.ins.2013.02.030

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. Proc VLDB Endow 1(2):1542–1552. https://doi.org/10.1145/1454159.1454226

Ding R, Wang Q, Dang Y, Fu Q, Zhang H, Zhang D (2015) YADING: fast clustering of large-scale time series data. VLDB Endow 8(5):473–484. https://doi.org/10.14778/2735479.2735481

Esteban C, Hyland SL, Rätsch G (2017) Real-valued (medical) time series generation with recurrent conditional GANs. arXiv Preprint arXiv:1706.02633

Ferreira LN, Zhao L (2016) Time series clustering via community detection in networks. Inf Sci 326:227–242. https://doi.org/10.1016/j.ins.2015.07.046

Forestier G, Petitjean F, Dau HA, Webb GI, Keogh E (2017) Generating synthetic time series to augment sparse datasets. In: 2017 IEEE international conference on data mining (ICDM), pp 865–870. https://doi.org/10.1109/ICDM.2017.106

Geler Z, Kurbalija V, Radovanović M, Ivanović M (2014) Impact of the Sakoe–Chiba band on the DTW time series distance measure for kNN classification. In: International conference on knowledge science, engineering and management. Springer, pp 105–114

Górecki T, Łuczak M (2013) Using derivatives in time series classification. Data Min Knowl Discov 26(2):310–331. https://doi.org/10.1007/s10618-012-0251-4

Górecki T, Łuczak M (2014) Non-isometric transforms in time series classification using DTW. Knowl Based Syst 61:98–108. https://doi.org/10.1016/j.knosys.2014.02.011

Guennec AL, Malinowski S, Tavenard R (2016) Data augmentation for time series classification using convolutional neural networks. In: ECML/PKDD workshop on advanced analytics and learning on temporal data

Guna J, Humar I, Pogačnik M (2012) Intuitive gesture based user identification system. In: 2012 Proceedings of 35th international conference on telecommunications and signal processing, TSP 2012, pp 629–633. https://doi.org/10.1109/TSP.2012.6256373

Ha TM, Bunke H (1997) Off-line, handwritten numeral recognition by perturbation method. IEEE Trans Pattern Anal Mach Intell 19(5):535–539. https://doi.org/10.1109/34.589216

Hayashi A, Mizuhara Y, Suematsu N (2005) Embedding time series data for classification. In: International workshop on machine learning and data mining in pattern recognition, pp 356–365

He H, Bai Y, Garcia EA, Li S (2008) ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: Proceedings of the international joint conference on neural networks, pp 1322–1328. https://doi.org/10.1109/IJCNN.2008.4633969

Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2014) Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series. Data Min Knowl Discov 29(2):358–399. https://doi.org/10.1007/s10618-014-0345-2

Jeong Y-S, Jeong MK, Omitaomu OA (2011) Weighted dynamic time warping for time series classification. Pattern Recogn 44:2231–2240. https://doi.org/10.1016/j.patcog.2010.09.022

Kate RJ (2015) Using dynamic time warping distances as features for improved time series classification. Data Min Knowl Discov 30(2):283–312. https://doi.org/10.1007/s10618-015-0418-x

Kurbalija V, Radovanović M, Geler Z, Ivanović M (2014) The influence of global constraints on similarity measures for time-series databases. Knowl Based Syst 56:49–67. https://doi.org/10.1016/j.knosys.2013.10.021

Lee J-G, Han J, Li X, Gonzalez H (2008) TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. Proc VLDB Endow 1(1):1081–1094. https://doi.org/10.1145/1453856.1453972

Li L, Aditya Prakash B (2011) Time series clustering: complex is simpler! Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 28(1):137–146. https://doi.org/10.1177/1420326X11423163

Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592. https://doi.org/10.1007/s10618-014-0361-2

Liu J, Zhong L, Wickramasuriya J, Vasudevan V (2009) uWave: accelerometer-based personalized gesture recognition and its applications. Pervasive Mob Comput 5(6):657–675. https://doi.org/10.1016/j.pmcj.2009.07.007

Lu S, Mirchevska G, Phatak SS, Li D, Luka J, Calderone RA, Fonzi WA (2017) Dynamic time warping assessment of highresolution melt curves provides a robust metric for fungal identification. PLoS ONE 12(3):e0173320. https://doi.org/10.1371/journal.pone.0173320

Lv Y, Zhai CX (2010) Positional relevance model for pseudo-relevance feedback. In: Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval—SIGIR'10, p 579. https://doi.org/10.1145/1835449.1835546

Masters J (2016) The level of pain and injury from slip and fall accidents. Brain Injury Society. http://www.bisociety.org/level-pain-injury-slip-fall-accidents/

National Council on Aging (NCOA) (2016) Falls prevention facts. https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/

Ng AY (1997) Preventing 'overfitting' of cross-validation data. In: ICML, vol 97, pp 245–253. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.47.6720&rep=rep1&type=pdf%0Ahttp://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.47.6720

Paparrizos J, Gravano L (2015) K-shape: efficient and accurate clustering of time series. ACM Sigmod. https://doi.org/10.1145/2723372.2737793

Paparrizos J, Gravano L (2017) Fast and accurate time-series clustering. ACM Trans Database Syst 42(2):1–49. https://doi.org/10.1145/3044711

Petitjean F, Forestier G, Webb GI, Nicholson AE, Chen Y, Keogh E (2015) Dynamic time warping averaging of time series allows faster and more accurate classification. In: Proceedings of IEEE international conference on data mining, ICDM, pp 470–479. https://doi.org/10.1109/ICDM.2014.27

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—KDD'12, p 262. https://doi.org/10.1145/2339530.2339576

Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850. https://doi.org/10.1080/01621459.1971.10482356

Rani S, Sikka G (2012) Recent techniques of clustering of time series data: a survey. Int J Comput Appl 52(15):1–9. https://doi.org/10.5120/8282-1278

Ratanamahatana CA, Keogh E (2005) Three myths about dynamic time warping data mining. In: Proceedings of the 2005 SIAM international conference on data mining, pp 506–510. https://doi.org/10.1137/1.9781611972757.50

Rodriguez A, Laio A (2014) Clustering by fast search and find of density peaks. Science 344(6191):1492–1496. https://doi.org/10.1126/science.1242072

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans Acoust Speech Signal Process 26(1):43–49. https://doi.org/10.1109/TASSP.1978.1163055

Shokoohi-Yekta M, Wang J, Keogh E (2015) On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In: Proceedings of the 2015 SIAM international conference on data mining, pp 289–297. https://doi.org/10.1137/1.9781611974010.33

Shou Y, Mamoulis N, Cheung D (2005) Fast and exact warping of time series using adaptive segmental approximations. Mach Learn 58(2–3):231–267. https://doi.org/10.1007/s10994-005-5828-3

Silva DF, Batista GE, Keogh E (2017) Prefix and suffix invariant dynamic time warping. In: Proceedings of IEEE international conference on data mining, ICDM, pp 1209–1214. https://doi.org/10.1109/ICDM.2016.107

Silva DF, Giusti R, Keogh E, Batista GE (2018) Speeding up similarity search under dynamic time warping by pruning unpromising alignments. In: Data mining and knowledge discovery. Springer, pp 1–29

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. https://www.francois-petitjean.com/Research/Petitjean2018-SDM-learn-warp-window.pdf

Valsamis A, Tserpes K, Zissis D, Anagnostopoulos D, Varvarigou T (2017) Employing traditional machine learning algorithms for big data streams analysis: the case of object trajectory prediction. J Syst Softw 127:249–257. https://doi.org/10.1016/j.jss.2016.06.016

Vinh NX (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854. https://doi.org/10.1182/blood-2008-03-145946

Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Proceedings of international conference on data engineering, pp 673–684. https://doi.org/10.1109/ICDE.2002.994784

Von Luxburg U (2010) Clustering stability: an overview. Found Trends® Mach Learn 2(3):235–274

Wagstaff K, Cardie C (2000) Clustering with instance-level constraints. In: Proceedings of the national conference on artificial intelligence. http://citeseer.ist.psu.edu/rd/0,307538,1,0.25,Download/http://citeseer.ist.psu.edu/cache/papers/cs/14353/http:zSzzSzwww.cs.cornell.eduzSzhomezSzcardiezSzpaperszSzicml-2000.pdf/wagstaff00clustering.pdf%5Cnhttp://portal.acm.org/citation.cfm?id=658275%5Cnhttp:/

Xi X, Keogh E, Shelton C, Wei L, Ratanamahatana CA (2006) Fast time series classification using numerosity reduction. In: Proceedings of the 23rd international conference on machine learning—ICML'06, pp 1033–1040. https://doi.org/10.1145/1143844.1143974

Zakaria J, Abdullah M, Keogh E (2012) Clustering time series using unsupervised-shapelets. In: Proceedings of IEEE international conference on data mining, ICDM, pp 785–94. https://doi.org/10.1109/ICDM.2012.26

Zhong Y, Liu S, Wang X, Xiao J, Song Y (2016) Tracking idea flows between social groups. In: AAAI, pp 1436–43

Zhou J, Zhu SF, Huang X, Zhang Y (2015) Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning. J Comput Sci Technol 30(4):859–873. https://doi.org/10.1007/s11390-015-1565-7