Semi-Supervision Dramatically Improves Time Series Clustering under Dynamic Time Warping

Hoang Anh Dau Nurjahan Begum Eamonn Keogh
University of California, Riverside
{hdau001, nbegu001, eamonn}@cs.ucr.edu

ABSTRACT

The research community seems to have converged in agreement that for time series *classification* problems, Dynamic Time Warping (DTW), based nearest-neighbor classifiers are exceptionally hard to beat. Obtaining the best performance from DTW requires setting its only parameter, the warping window width (*w*). This is typically set by cross validation in the training stage. However, for *clustering*, by definition we do not have access to such labeled data. This issue seems to have been largely ignored in the literature, with many practitioners simply assuming that "*the larger the better*" for the value of *w*, and using as large a value of *w* as computational resources permit. In this work we show that this is a naive approach which in most circumstances produces inferior clusterings. To address this problem, we introduce a novel semi-supervised technique that allows us to set the best value of *w*. Unlike virtually all other semi-supervised techniques, our ideas are completely independent of the clustering algorithm used, and can be utilized to improve time series clustering under partitional, hierarchical, spectral or density-based clustering. Our approach requires very little human intervention; moreover, we show that in many cases, true human annotation efforts can be replaced with automatically-generated "pseudo" supervision information. We demonstrate our technique by testing with more than one hundred publicly available datasets.

Keywords

Dynamic Time Warping, Time Series, Semi-Supervised Learning.

1. INTRODUCTION

A fundamental task in time series data mining is *clustering*. Clustering may be useful in its own right as an exploratory tool, and it is a subroutine in many higher-level algorithms such as rule-finding, semantic segmentation, anomaly detection, visualization and data editing [16]. Most research efforts to improve time series clustering have either proposed new algorithms [4][11][12][28][30], or new distance measures [15]. The latter is somewhat surprising, since the community seems to have long ago converged on the belief that the Dynamic Time Warping (DTW) distance measure is very hard to beat for *classification* [8][17], and it is not clear why its superiority would not extend to *clustering*.

CIKM'16 , Oct 24 - 28, 2016, Indianapolis, IN, USA.

We believe that the main reason DTW is not considered the "go-to" solution for clustering is because its single parameter, the amount of allowable warping (*w*), critically affects the quality of the returned clusters, regardless of the clustering algorithm used. This is not an issue for classification, where the best value for *w* can be learned by cross validation. This sensitivity to *w* for clustering is illustrated in Figure 1, which shows how changing the warping window width (*w*) affects the quality of clustering on three randomly chosen datasets.

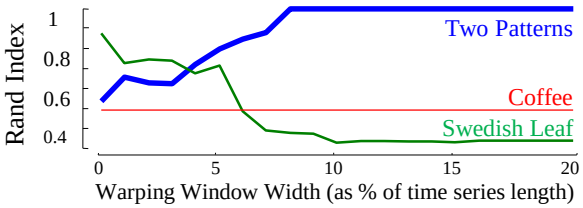

Figure 1: The Rand-Index vs. the warping window width for three datasets, using density-based clustering [21].

The figure shows that changing *w* can affect different datasets in radically different ways. For the Two Patterns dataset, increasing the amount of warping steadily improves the quality of the clustering until it converges at a perfect clustering with *w* = 9. In contrast, for Swedish Leaf, increasing *w* reduces the quality of clustering from a very impressive (for a 15-class problem) Rand-Index of 0.87 to a stunningly low score of 0.32 at *w* = 10%. This is all the more surprising given that allowing some warping slightly improves the *classification* accuracy in this dataset [8].

These results tell us that a practitioner who blindly uses Euclidean distance (i.e. *w* = 0%) will do very badly on some datasets. Likewise, another practitioner, perhaps motivated by the observation that DTW generally helps in classification problems, and who simply clusters with a hard-coded value of *w* set at 10%, will do very poorly on some datasets [15].

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

Given these observations, we are now in a position to state the problem we wish to solve:

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 such *w*.

There are many measures of *clustering quality*; however, measures based on sum-of-squares residual error do not allow meaningful comparisons between clusterings with different values of *w*. Here we wish to optimize the objective "correctness" of the clustering. Normally we will never 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.

How can we choose the best value *w* in the absence of class labels? One possibility is to use semi-supervised clustering [1][2][3][7][26]. 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 had a large impact is its *inefficiency*. Suppose we have a mere 1,415 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, in order to be sure that we get enough actionable annotations to guide the search in the clustering space, we must ask the user to annotate many hundreds or thousands of objects. This is clearly undesirable as the user may be unwilling or unable to provide such effort.

In this work 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 completely 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. This means 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. Our algorithm is maximally respectful of the cost of human effort.
- Because of the above, our approach requires very few annotations; in many cases, sixteen or fewer.
- While we mostly envisage asking a *human* for annotation, at least in some situations these annotations may be gleaned by examining side-information or statistical tests. Our framework can exploit such information.
- Our approach works for both single and multi-dimensional time series.
- Finally, as we shall demonstrate, our approach is very accurate, and robust to mistakes made by the annotator.

The rest of the paper is organized as follows. In Section 2 we review background and related work, before introducing our algorithm in Section 3. In Section 4 we carry out an extensive empirical evaluation with more than 100 time series datasets before offering conclusions and directions for future work in Section 5.

[1] "essentially" because some clustering algorithms are not defined (or loose certain guarantees) for non-metric distance measures.

2. BACKGROUND AND RELATED WORK

We begin by reviewing background material on DTW, then background material on semi-supervised learning, before discussing the most closely-related work.

2.1 Dynamic Time Warping

DTW is a distance measure that originated from within the speech recognition community. Recent work strongly suggests that DTW is the best distance measure for many data mining problems [8]. In [17], 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 [15].

As illustrated in Figure 2.*left*, DTW allows a *one-to-many* mapping between data points, thus enabling 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 Euclidean distance between the point i^{th} of *Q* and j^{th} of *C*. The warping path *W* is a set of contiguous matrix elements that defines the alignment between *Q* and *C*. The k^{th} element of *W* is defined as $w_{k=(i,j)_k}$.

The warping path is subject to several conditions: to 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. This DTW is called unconstrained DTW. Among all the warping paths possible, we are only interested in the path that minimizes the differences between the two time series.

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

Constrained DTW is a variant that imposes a limit on how much the warping path can deviate from the diagonal. This limit is known as the warping window width (*w*). For example, in Figure 2.*right* the warping path cannot visit the grayed out cells.

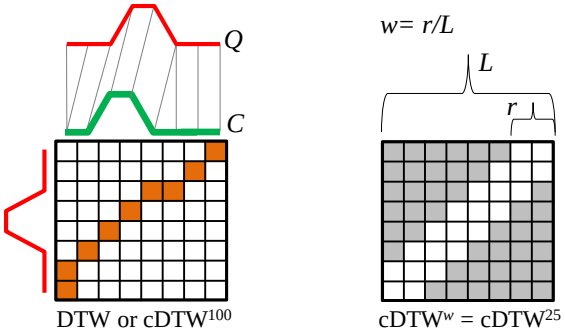

Figure 2: *left*) The unconstrained warping path for time series *Q* and *C*. Such warping paths are allowed to pass through any cell of the matrix. *right*) We can choose to constrain the warping path to avoid passing through cells that are far from the diagonal.

Constrained DTW helps to avoid pathological mappings between two time series, when one point in a 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 beneficial side effect of reducing the computation cost by narrowing down the search for qualified paths. A typical constraint is the Sakoe-Chiba Band, which expresses w as a percentage the time series length. We denote DTW with a constraint of w as cDTW^w.

The Euclidean distance between the two time series is the special case of DTW when the w is set to 0, enforcing a one-to-one mapping between data points. It is denoted as cDTW^0. Unconstrained DTW is denoted as cDTW^{100}. This review is necessarily brief; we refer the interested reader to [8][22] and the references therein for more details.

2.2 Correcting a Common Misunderstanding

Before proceeding we must ward off a possible misunderstanding, and make an original and important observation. To help us do so we will create a synthetic dataset which we will call Single Plateau (SP). This dataset (like all others in this paper) is available at [31]. Each item in the dataset consists of a vector of 500 random numbers taken from a standard Gaussian. In addition, to each exemplar we add a "plateau" of height 100 and with a length randomly chosen in the range five to twenty. If the plateau's location is between 1 to 200 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; Figure 3 shows some examples from each class.

If we cluster this dataset with cDTW^0, we obtain the random clustering shown in Figure 3.*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 Figure 3.*center*). Thus far, these observations agree with the community's intuition. However, what happens when we cluster using cDTW^{100}? We again obtain a clustering that appears random (Figure 3.*right*).

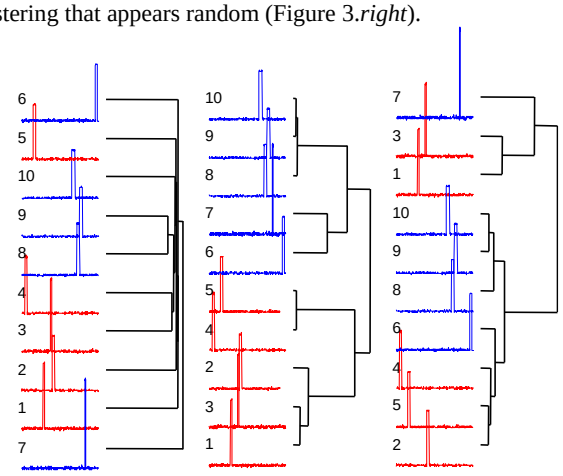

Figure 3: Hierarchical clustering result for the SP dataset. Exemplars in Class A are numbered 1 to 5 and shown in red. Exemplars in Class B are numbered 6 to 10 and shown in blue. *left)* Clustering with cDTW^0 *middle)* Clustering with cDTW^{10} *right)* Clustering with cDTW^{100} .

This idea, that "*a little warping is a good thing, but too much warping is a bad thing"* is known (although perhaps underappreciated [20]) for time series *classification* [6]; however, we believe that this is the first explicit demonstration of the effect for *clustering* (Figures 7, 12 and 13 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 that is the focus of this research.

This observation prevents us from considering a simple (although computationally expensive) solution to the task at hand: simply performing clustering under completely unconstrained warping.

It might also be imagined that we could discover the best warping window width for a given data *type*, and simply use that setting for all future datasets from the domain. For example, we might imagine that for *gesture recognition* cDTW^5 is generally best, but for *heartbeat classification* 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 being clustered. To see this, we can cluster increasing large instances of the SP dataset. For each size, we search over all possible values of w and record the value that maximizes the Rand-Index. Figure 4 shows the results, averaged over 1,000 runs.

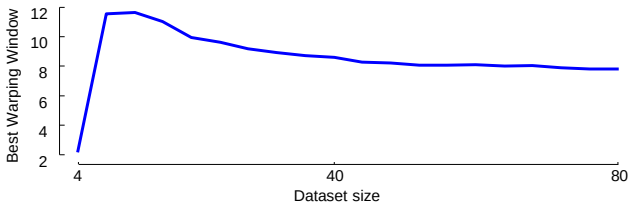

Figure 4: The optimal value of w vs. the dataset size for Single Plateau dataset.

The fact that the best value of w depends on *both* the data size and the data structure bodes ill for any attempt to learn a fixed one-time domain dependent value for it. Note that this size vs. best curve for w is *itself* different for different datasets.

2.3 Semi-Supervised Learning

The semi-supervised learning (SSL) paradigm has drawn significant attention in the data mining and machine learning communities in the last decade, due to its demonstrated utility in many practical applications [1][3][7][26]. 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 towards a more accurate data partitioning. This can be done in several (non-exclusive) ways:

- Enforcing constraints during the clustering process itself [26]. 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, [7] modifies the fitness function of a genetic search algorithm that optimizes clusterings.
- Seeding the clustering using the labeled examples to provide the initial seed clusters [3], 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 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 rarely are [5].

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

2.4 Related Work

Zhou et al. recently introduced a paper entitled "*Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning*" [30]. However, the method is perhaps better seen as *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 do not perform as well as our proposed approach. For example, for Trace they obtain a *best* Normalized Mutual Information (NMI) score of 0.813 whereas, as we will show in Section 4, we can easily obtain a near-perfect NMI of 0.97, without any human annotation.

Beyond this effort, we are not aware of any other work similar to our approach for semi-supervised learning for time series clustering. The general field of semi-supervised time series clustering is vast; we refer the reader to [19] and the references therein. We further briefly review some of the most recent, high-visibility efforts in time series clustering in Section 4.4, before direct empirical comparisons to our proposed algorithm.

3. OUR APPROACH

Without loss of generality we will use Rand-Index in this work, both 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 is thus impossible to optimize in a trivial way. There are some proposed variants including the Adjusted Rand-Index [24]; however, the classic Rand-Index [18] is widely accepted and used. Moreover, at least internally, we are only interested in *relative* improvements in clustering quality.

3.1 Clustering Algorithm

At the risk of redundancy we again emphasize that we are *not* proposing a clustering algorithm in this work. We are proposing a post-hoc measure that will allow us to score candidate clusterings created with different DTW parameters. Nevertheless, we must use *some* clustering algorithm. Without loss of generality we use the TADpole algorithm of [4], which is a specialization of the Density Peaks algorithm of [21] for DTW. This algorithm is suited to DTW because it does not require metric properties, and is particularly amenable to optimization by exploiting both upper and lower bounds to DTW [4].

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 we do not consider this option below for simplicity.

3.2 Choosing Constraints

As we 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 precious resource of 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 correctly clustered or not. A candidate constraint can be seen as a binary vector *C* whose length is the number of values of *w* we are considering. A '0' at the i^{th} position in *C* indicates the pair of time series was not correctly clustered under DTW^i, whereas a '1' indicates it was correctly clustered.

In Figure 5 we can see four candidate constraints. Constraint (A) is vacillating, and is probably of little use to us. We can interpret it as "voting" for a *w* value of 2 or 3 or 6 or 8, etc. Such constraints are very rare and probably indicate a "hybrid" object *just* on the cusp of two distinct clusters.

Constraints (B) and (C) are always/never satisfied respectively. It is easy to see that it is pointless to show such constraints to the user, as they "vote" equally for all values of *w*. In most datasets we consider, the majority (often the *vast* majority) of constraints are these two types. With a little introspection, it is comforting that *most* constraints are non-volatile, as it suggests that the most of the objects being clustered are really in stable clusters. If *all* constraints were highly volatile, it is hard to imagine *any* clustering we could select is meaningful in any sense.

In contrast to the above, constraint (D) seems like an ideal constraint. It can be interpreted as: "*A value for w that is between zero to six is not enough, but anything seven or above works.*"

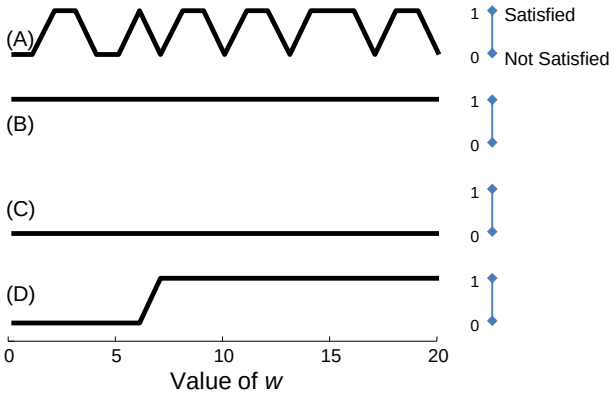

Figure 5: Four representative constraints. (A) represents a vacillating constraint, (B) an *always* satisfied constraint, (C) a *never* satisfied constraint, (D) an ideal constraint.

These observations inform our algorithm design. Constant constraints (types (B) and (C)) should be discarded. Of the remainders, "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 just 1.

$$\text{Simplicity}(C) = \sum_{k=0}^{(\max w)-1} 0, \text{if } C_k = C_{k+1}, \quad \text{else } 1$$

Our algorithm for finding the set of constraints we will ask the user to evaluate is presented in Table 1.

Table 1: Algorithm for Finding the Constraint Set

```
   Input: set of candidate constraints, max 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    ans ← get_user_willingness('Do Another? Y or N')
6    if ans = 'Y'
7        index  ←  index  + 1
8   else
9        index  ←  infinity        // break out of loop
10  end
11 end
```

We begin in line 1 by sorting the constraints, simplest first (breaking ties randomly). At this point, we enter a loop, and while we have some constraints left to annotate, *and* 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 her *must-link*/*cannot-link* annotation.

Figure 6 shows some 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 Figure 6.*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. We would therefore expect the user would annotate this as "*must-link.*"

In contrast, for Figure 6.*right*, we hope the user would recognize that in spite of similarity of the two time series (they have a relatively small Euclidean distance), one time series is missing the short peak that seem to characterize the other sequence.

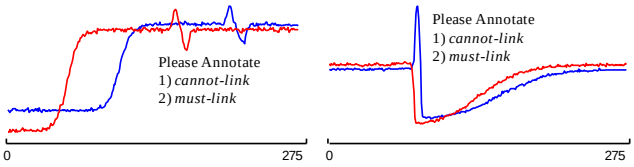

Figure 6: Examples of pairs of time series presented to user for annotation. *left*) Here the correct label is *must-link*. *right*) Here the user should ideally choose *cannot-link*.

Naturally, we desire that our algorithm is insensitive to occasional annotation mistakes. We consider this issue in Section 4.2. 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.

We can see the "anytime" nature of the algorithm by examining the predictions we make for *w* as we obtain increasing numbers of user annotations. Figure 7 show such an example for several datasets. Note that in both cases the "shape" of our prediction vector seems to converge to the shape of the ground truth Rand-Index curve after just sixteen user annotations. However it is important to note that this is *not* necessary for our algorithm to be successful. All we actually require is that the prediction of the best setting for *w* agrees with the ground truth. Recall that this prediction is the location of the maximum value, ties broken by choosing the smallest value.

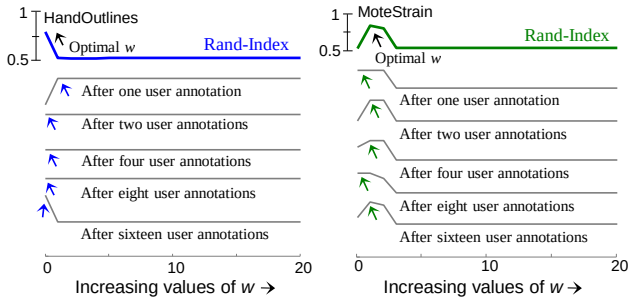

Figure 7: For two datasets HandOutlines and MoteStrain: The ground truth Rand-Index (colored/bold line). The prediction vectors (light/gray lines) learned after 1 to 16 user annotations allow us to estimate *w* (arrows). The shapes of the prediction vectors reflects the ratio of constraints satisfied (correctly linked or not linked) at each *w*.

3.3 Pseudo User Annotation

As the results in Figure 7 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 such situations?

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 automatically generated *pseudo-relevance feedback* [14].

The ambition of these approaches is limited. No one claims that *pseudo*-relevance feedback is as useful as *real* human feedback (if it was, the community would abandon any effort to elicit expensive human feedback). It suffices that it is significantly 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 clusterings, we randomly sample objects from the dataset. For each object O, we create a copy of it that we denote $\bar{O}$. We add some warping to $\bar{O}$, and place it into the dataset with the (*pseudo*) constraint *must-link*(O, $\bar{O}$). The intuition is that because we know that object $\bar{O}$ is just a minor variant of O, we can safely assume that *had* $\bar{O}$ 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 just like those produced by Table 1.

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

A good value for *w* depends not only on the intrinsic variability of the time axis and on the size of the dataset, but on the time series *shapes* themselves. We can illustrate the latter point with a simple experiment. We created two near identical datasets, *Slim Plateau* and *Broad Plateau*, which, as their names hint, differ only in the width of the plateau. In both datasets, one class has a plateau in the first half and the other class has a plateau in the second half. As we look that the leftmost column of Figure 8 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 again them using cDTW^0?

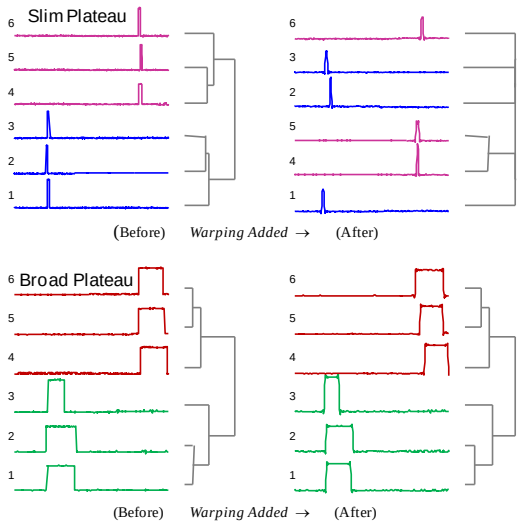

Figure 8: Warping affects different datasets differently under hierarchical clustering. *top*) The clustering of the Slim Plateau dataset is very brittle to the presence of warping in the time axis. *bottom*) In contrast, the Broad Plateau dataset is extremely robust to *identical* levels of warping.

As we can see in the rightmost column of Figure 8, the clustering of *Slim Plateau* becomes essentially random, whereas *Broad Plateau* is basically unaffected.

The take-away message from this experiment is as follows. In this pathological case we can measure exactly how much warping 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, on the dataset size, cf. Section 2.2).

Table 2 outlines algorithm for generating pseudo 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: PA, the set of pseudo annotations for Dnew
1	Dnew ← random_shuffle(D)
2	**for** i = 1 **in** steps of 2 to numberOfInstances(Dnew)
3	$Dnew_{i+1}$ = add_random_warping(D_i) // See Table 3
4	$PA_{(i+1)/2}$ = set_constraint($Dnew_{i+1}$,$Dnew_i$,'must-link')
5	**end**

In line 1 we ensure that the data does not have any arbitrary structure in its ordering. In line 2 we enter a loop which replaces every second data object with a warped version of the data object that preceded 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 Figure 4). The algorithm also outputs PA, a set of pseudo annotations for D*new*. PA is essentially identical to UA produced in Table 1, except its annotations are produced without human interventions. Figure 9 shows some examples of 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 PUA (Pseudo User Annotation) algorithm.

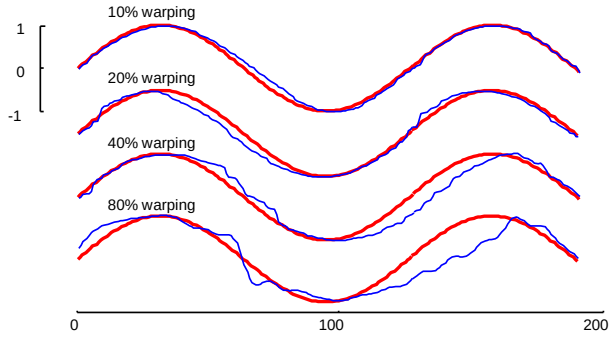

Figure 9: From top to bottom: Increasingly warped versions of a sine wave. The red/bold curve is the original and the blue/fine curves are the ones with warping added.

Table 3: Code to add warping to a time series

```
function [warped_T] = add_warping_one_time_series(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
```

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, we typically only have a tiny fraction of D annotated this way. In contrast, *every* item in D*new* has an annotation, which provides this approach with an advantage, should we choose to use them all. Figure 10 shows how this idea works with Trace and Two Patterns. Here we use 64 out of 1,824 pseudo constraints available for Two Patterns to reach the correct 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).

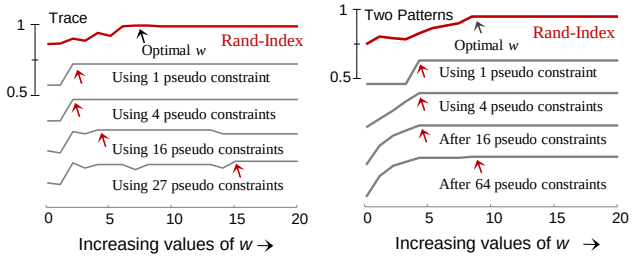

Figure 10: Trace and Two Patterns' prediction vectors using pseudo constraints provided by the PUA algorithm.

The reader may wonder how much warping we should use to obtain good pseudo constraints. The good news is that it makes almost no difference. In this particular case, we tried all 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*.

3.4 Further Reducing Human Effort

There are a handful of techniques we could use to reduce the number of annotations given by the user, many such ideas can be borrowed directly from the information retrieval community [14]. For example, suppose the user decides {7,11} *must-link*, and that {11,27} *must-link*, then there is little point in asking her opinion on {7,27} since she will surely also label this pair as must-link (by transitivity). We do not consider such optimizations here for

brevity, and because, as we will demonstrate, the simplest version of our ideas is already very competitive.

4. EMPIRICAL EVALUATION

In order to ensure that our experiments are reproducible, we have built a website which contains all data/code/raw spreadsheets for the results, in addition to many experiments that are omitted here for brevity. Because we are testing on so many datasets, including all 85 available at [27] plus several more that we introducing with this work, we do not have the space to list all their names and characteristics. We have placed such a summary at [31].

At the risk of redundancy we restate that we are *not* introducing a new clustering algorithm, merely proposing a technique to choose among candidate clusterings that differ in the value of *w* used to create them. Nevertheless, in Section 4.4 we explicitly compare TADpole using learned warping window to five recent state-of-the-art clustering algorithms.

4.1 Preliminary Tests

We denote our algorithm as $cDTW^{ss}$ (DTW Semi-Supervised). We compare to two rivals, clustering with $cDTW^0$ (Euclidean distance), and clustering with $cDTW^{10}$. These two rival methods account for virtually the entire literature, for example [9] uses $cDTW^0$ and [15] uses $cDTW^{10}$. A surprisingly large number of papers neglect to explicitly state what value of *w* was 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 [4], is identical for all approaches, and completely deterministic [21]. 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 agreement with the literature, most datasets do not require *w* greater than 10% [20]). For example, in Figure 1 the maximum Rand-Index is 1.0 for Two Patterns and 0.89 for Swedish Leaf. We can then 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 just sixteen labels; this is about the amount a person can annotate in one minute. With sixteen labeled constraints we achieve success of 46 out of 102 datasets, with $cDTW^0$ 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 requires only a few minutes of user effort, and typically represents less than 0.0001% of the labeled pairs.

In spite of 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 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 got within at least 0.98 of optimal.

However, in some cases we do achieve significantly worse than optimal. Essentially, all such cases can be attributed to very small datasets (or small, relative to the number of clusters). As shown in Figure 11, this tends to result in clusterings that are very unstable to small changes in *w*. The fact that small datasets have poor stability when clustered is well known [25], and the issue is orthogonal to our contributions. In essence, we feel that if the best value of *w* is poorly defined and unstable, it may be impossible for any algorithm to learn *w*. Nevertheless, even in such datasets we do not do worse than the lower scoring of out two rivals

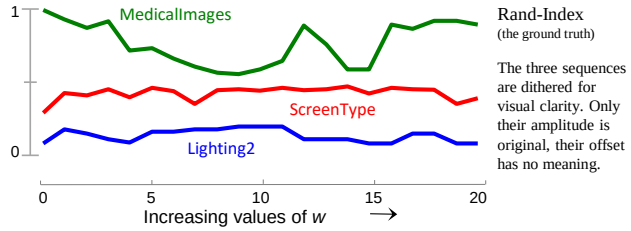

Figure 11: The Rand-Index vs. the warping window width for three small datasets. Contrast the variability of the curves with the relatively smooth curves shown in Figure 1.

4.2 Robustness to Incorrect Constraints

The experiments in the previous section assumed that all the constraints the user gave are correct. However, this assumption may be unwarranted in many circumstances. That is to say, our annotator may indicate that two items *cannot-link*, when in fact 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, this time randomly inverting some fraction of the constraints to be incorrect.

As we can see in Figure 12, for at least ItalyPowerDemand and MiddlePhalanxOutlineAgeGroup datasets, we can achieve near perfect results even if a significant fraction of the constraints are 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*. We then observe the mean best *w* predicted averaged over ten runs. We find it is consistently 0 for ItalyPowerDemand dataset and 1 for MiddlePhalanxOutineAgeGroup, which agrees with the objective ground truth.

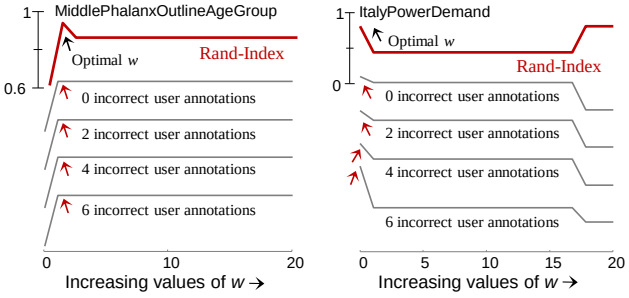

Figure 12: Robustness to incorrect constraints. In each case, 16 pairs of time series are presented for annotation. The annotator may wrongly label a pair that should have been *must-link* as *cannot-link* and vice versa. Our algorithm is robust to these mistakes.

As a practical matter, any system used to garner user feedback should allow three choices (not just two) 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 difficult or ambiguous cases.

4.3 Handling the Multi-Dimensional Case

Thus far we have only considered single dimensional time series; however, the proliferation of sensors from sources such as wearable

devices means that there is increasing interest in multi-dimensional time series data [22]. 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 [22].

In Figure 13 we consider the 4,480 object, three-dimensional UWave dataset [13] which has become something of a benchmark for gesture recognition in the last five years. We also consider the Handwriting Accelerometer using all three dimensions available. Even though all dimensions are not necessary for this task (and in fact can introduce noise to the clusters), we only wish to illustrate that our algorithm can correctly predict a good value for *w*.

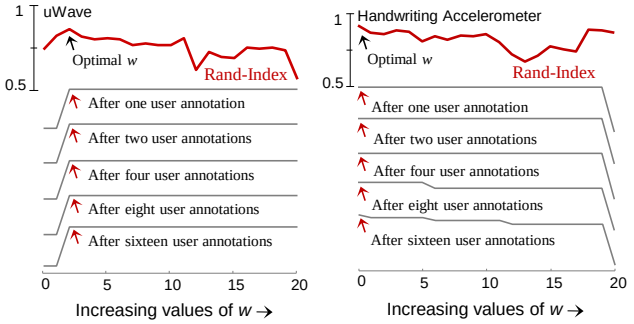

Figure 13: Three-dimensional uWave and Handwriting Accelerometer datasets clustered with DTW_I.

While there are just over one million possible pairwise constraints, our algorithm can find the optimal *w* with just sixteen annotations. Note that here the amount of warping is critical, with too much or too little giving poor results. This fact might go some way to explaining 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.

4.4 Comparisons to Rival Methods

In this section we have two related aims. The first is to satisfy our obligation of comparing to other clustering methods in the literature (in spite of the fact we are 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 generally dwarfs any effect of the choice of clustering algorithm.*

This can also be stated as the essentially equivalent claim:

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

This claim is important because some published research has claimed improvements in creating a clustering algorithm, or in designing an alternative distance measure, with only slight improvements demonstrated in accuracy. 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 largely contradicts recent claims such as "*…the choice of algorithm, .., is as critical as the choice of distance measure*" [15].

We reiterate that we are only offering *some* evidence to support this claim. A more forceful demonstration (that is rigorously fair to all cited work) would require more space than is available here.

In a recent paper [15] the authors introduces k-Shape, a system that combines a novel time series clustering algorithm and a novel distance measure (Shape-Based Distance (SBD)), that are designed to work in conjunction with each other. 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* typically makes improvements that dwarf the claimed improvements of the SDB algorithm. For example, for the Trace dataset they compare five clustering algorithms that use DTW vs. the same five clustering algorithms using SBD. The former achieves Rand-Indexes of {0.87, 0.75, 0.75, 0.83, 0.77} and the latter achieves Rand-Indexes of {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 significantly beat *all* these approaches without any human intervention, as our PUA algorithm can achieve 0.99.

We have similarly large margins improvements for most datasets, for example for Two Patterns, [15] has the DTW based algorithms achieving Rand-Indexes 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.

Similarly, in a recent paper [12] the authors introduce a clustering method called CLDS (complex-valued linear dynamical systems), and claim that the "*approach produces significant improvement in clustering quality, 1.5 to 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 reader 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[2], and they manage to score 0.1015, while $cDTW^{100}$ using K-Means scores significantly worse at 0.4229, 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."* But note that this manipulation is *only* needed for their proposed method; cDTW can handle sequences of different lengths. When we re-ran the experiments, we found that $cDTW^{20}$ gives a perfect conditional entropy of 0 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, TADpole, if forced to use $cDTW^{10}$, scores a slightly worse 0.142.

Note that we are not claiming the work proposed in [12] is without merit. We are simply pointing out that at least on the datasets the original authors used to validate the method, cDTW, using any

[2] For conditional entropy, smaller is better, with 0.0 being perfect.

reasonable choice for *w* with an off-the-shelf clustering method, can be very competitive.

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 [11]. The Rand-Indexes 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 the Trace data, we can significantly beat *all* these approaches without any human intervention, as our PUA algorithm can achieve 0.99.

Another recently published time series clustering technique called YADING is shown to "*provide theoretical proof which... ...guarantees YADING's high performance*" [9]. However, these guarantees are only with respect to Euclidean distance. The only publicly available *real* dataset they test on is StarLightCurves, where they obtain a Normalized Mutual Information (NMI) score of 0.60. However, as shown in Figure 14, with 16 constraints given by the user, we find cDTW[1] to be a good choice and achieve a NMI of 0.79, significantly better (Omitted for brevity: in fact, any number of constraints above four also works this well).

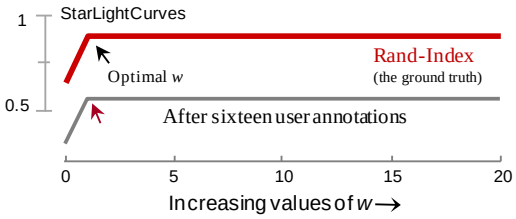

Figure 14: The Rand-Index vs. the warping window width for StarLightCurves. We predict w = 1, obtaining a Rand-Index of 0.83, equivalent to a NMI of 0.79.

Likewise, in an expanded tech report that augments the paper [10], the YADING method achieves a NMI of 0.61 on the CinC_ECG_torso dataset. However on this dataset, our algorithm discovers cDTW[1] to be the best choice for *w*, and NMI of 0.66.

Why did the authors of [9][10] dismiss DTW as a distance measure? They noted that DTW "*is one order of magnitude slower than calculating* [*Euclidian distance*]," and further noted that it only took them a brief 3.1 seconds 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 seconds or taking 30 seconds to do the clustering seems completely inconsequential (but see also Section 4.5). Of course, the authors are correct in noting that there is sometimes a need for great speed and scalability. However, in many domains the tradeoff between speed and accuracy will 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 are able to improve accuracy.

Finally, a paper to appear in AAAI tests four algorithms for time series clustering, two of them based on DTW [29]. These algorithms give 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 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 all other changes inconsequential.

4.5 Scalability of our Algorithm

At first blush our algorithm appears to require a significant overhead in time complexity, given that the Density Peaks algorithm [21] 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. First, 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 able to prune 90%-plus of the cDTW calculations.

In fact, we can do even better than this. Instead of doing twenty-one independent clusterings, we can exploit the fact that for any two time series *Q*,*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 using any $cDTW^w$ calculations actually made, as lower bounds in the next level. Thus, the time overhead for our ideas is only slightly more than a *single* highly optimized clustering.

Finally we note that there is a wide variety of DTW implementations and the efficiency differences between good and bad implementations 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, notes that "*several experiments were unable to return results within 20 days*" [29]. However, we can cluster exactly these same datasets in at most minutes, at least 10,000 times faster.

4.6 Nontransferability of the Best Setting for *w*

We claimed in the introduction that the best setting of *w* for *classification* is generally no indicator of the best setting of *w* for *clustering*. Since this assumption has been explicitly made (but never *tested*) multiple times in the literature [15], we will take the time to show that it is unwarranted. In Figure 15 we show both the Rand-Index and the accuracy for two datasets.

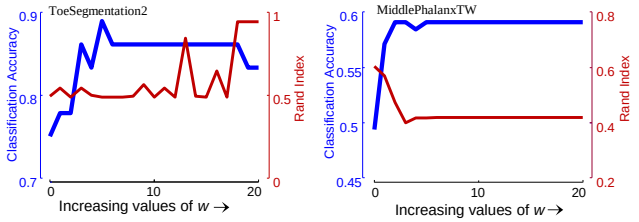

Figure 15: The Rand-Index (red/fine) and the classification accuracy (blue/bold) vs. the warping window width, for two representative datasets.

In retrospect it is not surprising that these values are at best weakly related. For 1NN classification (the most commonly used classification technique in the literature [8][20]) only the distance between the unlabeled exemplar and it's *single* nearest neighbor matters. However, for clustering, the mutual distance among small *groups* of objects matter.

5. CONCLUSIONS AND FUTURE WORK

In this work we have shown that *w*, the amount of warping allowed, is a critical parameter for clustering time series under the DTW distance. For most datasets, if this parameter is badly set, then nothing else matters; it will simply be impossible to produce a high

quality clustering. We have further proposed the first semi-supervised technique designed to discover the best value for *w*.

Our paper has several other observations that are novel, or at least underappreciated. We have shown *w* depends not only on the data object shapes, but on the number data objects considered. This observation has been made for classification before, but not for clustering [20]. 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 [15]. Finally, in the last decade, a handful of researchers have argued that warping constraints are a necessary evil, and that there are "*cases where unconstrained warping is useful*" [23], or that research should "*focus on unconstrained DTW*" [1]. While 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%, suggests that these efforts are likely to be fruitless.

Finally, we have released all our code and data [31], to allow others to confirm, extend and exploit our ideas.

Acknowledgements

We would like to acknowledge funding from NSF IIS-1161997 II and NSF IIS-1510741.

6. REFERENCES

[1] Athitsos, V., et al. Approximate embedding-based subsequence matching of time series. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, 2008, 365-378.

[2] Basu, S., Bilenko, M. and Mooney, R.J. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, 2004, 59-68.

[3] Basu, S., Banerjee, A. and Mooney, R. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning*, 2002.

[4] Begum, N., Ulanova, L., Wang, J. and Keogh, E. Accelerating Dynamic Time Warping Clustering with a Novel Admissible Pruning Strategy. In *Proceedings of the 21st ACM SIGKDD* 2015, 49-58.

[5] Bilenko, M., and Mooney, R. J. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, 39-48.

[6] Chen, Y., Hu, B., Keogh, E.J. and Batista G. E. A. P. A. DTW-D: Time series semi-supervised learning from a single example. In *Proc' of the 19th ACM SIGKDD*, 2013, 383-391.

[7] Demiriz, A., Bennett, K. P. and Embrechts, M. J. Semi-supervised clustering using genetic algorithms. *Artificial neural networks in engineering (ANNIE-99)*, 1999, 809-814.

[8] Ding, H., Trajcevski, G., Scheuermann, P., Wang, X. and Keogh, E. Querying and Mining of Time Series Data: Experimental Comparison of Representations and Distance Measures. *Proc' of the VLDB Endowment*, 2008, 1542-52.

[9] Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., and Zhang, D. YADING: Fast Clustering of Large-scale Time Series Data. *Proc' of the VLDB Endowment*, 2015, *8*(5), 473-484.

[10] Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., and Zhang, D. Evaluation on Real Datasets: YADING. *Microsoft Tech Report*, 2015.

[11] Ferreira, L. N. and Zhao, L. Time Series Clustering via Community Detection in Networks. *Information Sciences*, 2015, Volume 53, 2015, Pages 183–190.

[12] Li, L. and Prakash, B. A. Time series clustering: Complex is simpler!. In *Proceedings of the 28th International Conference on Machine Learning*, 2011, 185-192.

[13] Liu, J., Zhong, L., Wickramasuriya, J. and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 2009, *5*(6), 657-675.

[14] Lv, Y. and Zhai, C. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 2010, 579-586.

[15] Paparrizos, J. and Gravano, L. k-Shape: Efficient and Accurate Clustering of Time Series. In *Proceedings of the 2015 ACM SIGMOD*, 1855-1870.

[16] Petitjean, F., Forestier, G., Webb, G., Nicholson, A. E., Chen, Y. and Keogh, E. Dynamic Time Warping Averaging of Time Series Allows Faster and More Accurate Classification. In *IEEE ICDM* 2014, 470-479.

[17] Rakthanmanon, T., et al. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proc' of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, 262-270.

[18] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 1971, 66(336), 846-850.

[19] Rani, S. and Sikka, G. Recent Techniques of Clustering of Time Series Data: a Survey. *Int. J. Comput. Appl*, 2012, *52*(15), 1-9.

[20] Ratanamahatana, C.A. and Keogh, E.J. Three Myths about Dynamic Time Warping. In *Proceedings of SIAM International Conference on Data Mining*, 2005, 506-10.

[21] Rodriguez, A. and Laio, A. Clustering by fast search and find of density peaks. *Science* 344, no. 6191, 2014, 1492-1496.

[22] Shokoohi-Yekta, M., Wang, J. and Keogh, E. On the Non-Trivial Generalization of Dynamic Time Warping to the Multi-Dimensional Case. In *Data Mining. Proceeding of the 2015 International Conference on*, 2015, 39-48.

[23] Shou, Y., Mamoulis, N. and Cheung, D. W. Fast and Exact Warping of Time Series Using Adaptive Segmental Approximations. *Machine Learning*, 2005, *58*(2-3), 231-267.

[24] Vinh, N. X., Epps, J., & Bailey, J. Information theoretic measures for clusterings comparison. *The Journal of Machine Learning Research*, *11*, 2010, 2837-2854.

[25] Von Luxburg, U. Clustering stability: An overview. Now Publishers Inc, 2010.

[26] Wagstaff, K. and Cardie, C. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Machine Learning*, 2000, 1103–10.

[27] Yanping, C., et al. The UCR Time Series Classification Archive. www.cs.ucr.edu/~eamonn/time_series_data/, 2015.

[28] Zakaria, J., Mueen, A. and Keogh E. Clustering Time Series Using Unsupervised-Shapelets. In *Proc' of the 2012 IEEE 12th International Conference on Data Mining*, 785-794.

[29] Zhong, Y., Liu, S., Wang, X., Xiao, J. and Song, Y. Tracking Idea Flows between Social Groups. *arXiv preprint arXiv:1512.04036*. To appear in AAAI 2016, 2015.

[30] Zhou, J., Zhu, S. F., Huang, X. and Zhang, Y. Enhancing Time Series Clustering by Incorporating Multiple Distance Measures with Semi-Supervised Learning. *Journal of Computer Science and Technology*, 2015, *30*(4), 859-873.

[31] Supporting webpage: sites.google.com/site/dtwclustering/