UNIVERSITY OF CALIFORNIA
RIVERSIDE

Towards More Accurate Time Series Data Mining by Constraining Model's Flexibility

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Hoang Anh Dau

September 2019

Acknowledgments

I am grateful to my academic advisor, Dr. Keogh, without whose help and support, I would not have been here. My finish of this journey raises no other emotion than surprise to my very self.

I thank Dr. Chen, Dr. Lornadi and Dr. Papalexakis for being in my committee, and for your guidance and encouragement whenever I seek. I thank Dr. Tuncel for being a committee member in my oral qualifying exam.

I am thankful to my advisors at RMIT University in Melbourne, who supported me in this adventure to the USA.

I would like to thank all my lab mates, friends and people I have crossed path, for shaping my experience and keeping my heart warm.

Finally, I am indebted to my parents and little brother for their love and patience.

Acknowledgement of previously published material: Chapter 2, 3 and 4 of this dissertation, in part or in full, is a reprint of the material as it appears in *Proceedings of ACM International Conference on Information and Knowledge Management, Proceedings of IEEE International Conference on Big Data*, and *Data Mining and Knowledge Discovery* journal. Chapter 5, in full, is a reprint of the material as it appears in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* Chapter 6, in full, is a reprint of the material as it appears in the e-print archive *arXiv.org.*

To people who care, for all the support.

ABSTRACT OF THE DISSERTATION

Towards More Accurate Time Series Data Mining
by Constraining Model's Flexibility

by

Hoang Anh Dau

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, September 2019
Dr. Eamonn J. Keogh, Chairperson

This dissertation is motivated from enabling various tasks in large scale data mining of time series to produce more accurate and reproducible results, and tailor the solutions to user's specific need when that is favored. To that end, we have explored and contributed to the literature in three parts; each touches an active area of research and unifies under a common theme, reducing errors in time series data mining by learning constraints on model's flexibility.

The first body of work concerns Dynamic Time Warping (DTW), 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. This parameter gives DTW the flexibility to deal with data that can be locally out of phase, however the DTW algorithm sometimes exploits this flexibility to give pathological and unwanted results. We demonstrate the importance of setting DTW's warping window width correctly, to constrain this flexibility, and we propose novel methods to learn this parameter in both supervised and unsupervised settings.

The second body of work concerns time series motif discovery, perhaps the most used primitive for time series data mining. We point out that the current definitions of motif discovery are limited and can create a mismatch between the users intent/expectations, and the motif discovery search outcomes. We explain the reasons behind these issues and introduce a novel and general framework to address them.

The last body of work concerns making more time series data sets and baseline results publicly available for gauging progress and comparison of rival approaches in spirit of reproducible research. We work on expanding the UCR Time Series Archive, an important resource in the time series data mining community, from 85 data sets since the last Summer 2015 release to 128 data sets in Fall 2018. Creating benchmark results for this archive required 61,041,100,000,000 DTW comparisons, greatly more than the number of DTW comparisons that have appeared in all research papers combined. Beyond expanding this valuable resource, we offer pragmatic advice to anyone who may wish to evaluate a new algorithm on the archive.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A time series is an ordered set of observations in real-valued numbers, often sampled at regular time intervals. Time series data are ubiquitous in every aspects of our life. A one-month long time series of how many hours we sleep at night or how many steps we take a day has much to say about our health and well-being. Similarly, massive amount of time series data is generated in medical fields, businesses, industrial manufacturing and whole earth monitoring. Traditionally, these data are laboriously hand-measured and paper-filed. However, recent advances in the Internet, computer hardware and software have enabled us to not only pervasively gather data at scale, but also store and retrieve them efficiently.

Due to their temporal properties, analyzing time series data requires special tool sets, different from techniques designed for data types such as images, audio, text and sequence data. Data mining is the process of examining data to extract pattern, to gain insight and discover knowledge. While the concept is multi-disciplinary in nature and therefore might be open to different interpretations, at large, data mining can be regarded as *"the nontrivial extraction of implicit, previously unknown, and potentially useful information from data"* [41]. Data mining of time series has been an important area of research in computer science and will remain so for two reasons: the availability of massive amount of data and the demand to act upon these data.

This dissertation is motivated from enabling various tasks in large scale data mining of time series to produce more accurate, reproducible results and tailored to the user's specific need when that is favored. To that end, we have explored and contributed to the literature in three parts; each touches an active area of research and unifies under a common theme, reducing errors in time series data mining constraining the model's flexibility. We briefly introduce each part below.

1.1 Constraining Dynamic Time Warping's Flexibility

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 [95]. Both clustering and distance-based classification algorithms depend critically on the availability of a good distance measure [4, 5, 34, 47, 48, 77, 93, 97, 131]. Over the last decade, the time series research community seems to have come to the consensus that Dynamic Time Warping (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 [5, 34].

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 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 [83].

In the supervised case with ample data, w is typically set by cross-validation in the training stage. However, this method is likely to yield sub-optimal 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 data set, 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 data set size. Thus, even if a practitioner knows the best setting for a given data set, 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 Chapter 2, we demonstrate the importance of setting DTW's warping window width correctly, and we also propose novel methods to learn this parameter in both unsupervised setting (Chapter 3) and supervised settings (Chapter 4). 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 data sets. Our forceful results allow us to make a perhaps unexpected claim; an under-appreciated "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.

1.2 Embedding User's Biases into Motif Search

Time series motif discovery [76] has emerged as perhaps the most used primitive for time series data mining, and has seen applications to domains as diverse as robotics, medicine and climatology. There has been recent significant progress on the scalability of motif discovery. However, we believe that the current definitions of motif discovery are limited, and can create a mismatch between the users intent/expectations, and the motif discovery search outcomes. In Chapter 5, we explain the reasons behind these issues, and introduce a novel and general framework to address them. Our ideas can be used with current state-of-the-art algorithms with virtually no time or space overhead, and are fast enough to allow real-time interaction and hypotheses testing on massive data sets. We demonstrate the utility of our ideas on domains as diverse as seismology and epileptic seizure monitoring.

1.3 Making More Data Sets Publicly Available for Reproducible Research

The UCR Time Series Archive - introduced in 2002, has become an important resource in the time series data mining community, with at least one thousand published papers making use of at least one data set from the archive. The original incarnation of the archive had sixteen data sets but since that time, it has gone through periodic expansions. The last expansion took place in the Summer of 2015 [20] when the archive grew from 45 data sets to 85 data sets. Chapter 6 introduces and will focus on the new data expansion from 85 to 128 data sets [28]. Beyond expanding this valuable resource, this Chapter offers pragmatic advice to anyone who may wish to evaluate a new algorithm on the archive. Finally, this Chapter makes a novel and yet actionable claim: of the hundreds of papers that show an improvement over the standard baseline (1-nearest neighbor classification), a large fraction may be mis-attributing the reasons for their improvement. Moreover, they may have been able to achieve the same improvement with a much simpler modification, requiring just a single line of code.

Chapter 2

The Importance of Tuning DTW's Warping Window Width

2.1 Motivation

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). This parameter controls the maximum lead/lag for which points can be mapped to, thus preventing pathological mapping between two time series. 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 2.1 illustrates this sensitivity to w for clustering under DTW. Figure 2.2 similarly demonstrates 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.

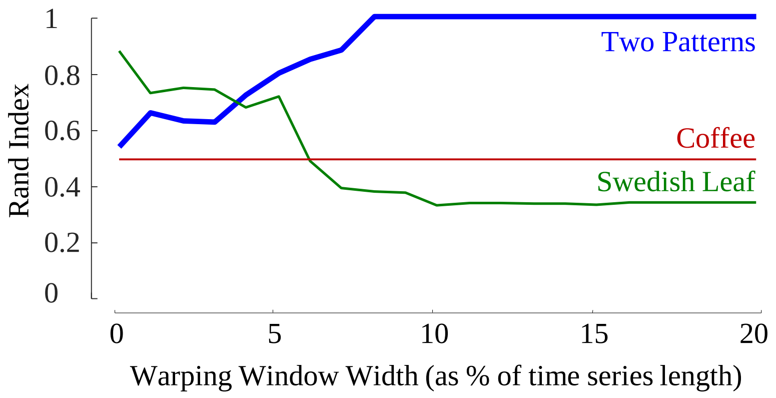

Figure 2.1: Rand-Index vs. warping window (w) width for three data sets, using a density-based clustering algorithm [12]. A larger value of w can make things better, worse or have no effect.

Figure 2.1 shows how changing w affects the quality of clustering on three different data sets. For *TwoPatterns*, increasing the amount of warping steadily improves clustering quality until it reaches perfection with $w = 9$. In contrast, for *SwedishLeaf*, 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 data set slightly [34].

These results indicate that blindly using the Euclidean distance for clustering (i.e. $w = 0$) will yield poor results on some data sets. Likewise, another practitioner, perhaps motivated by the observation that DTW generally helps in classification problems [5, 4, 34], and thus simply clusters with a hard-coded value of w set at 10, will also do poorly on some data sets [93].

The *Coffee* data set is unusual (and empirically rare), since it is virtually unaffected by varying w (in Figure 2.1 it is 0.48 when $w = 0$ and 0.49 everywhere else), but even here we can still avoid 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 [20] (and thus, reflected in hundreds of papers, for example Deng et al. [33] and Gorecki and Luczak [47]). Figure 2.2.*top* demonstrates that on many data sets, this simple method produces favorable results. It predicts the correct optimal value of w for *CinCECG*, and it is only off slightly for *CBF* and *FiftyWords*.

Nevertheless, the results above contrast with the examples in Figure 2.2.*bottom*. 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 *GunPoint*, but $cDTW^2$ would have reduced the rate of mis-classifications 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 [83]. The authors of this study found that DTW is an effective distance measure to classify

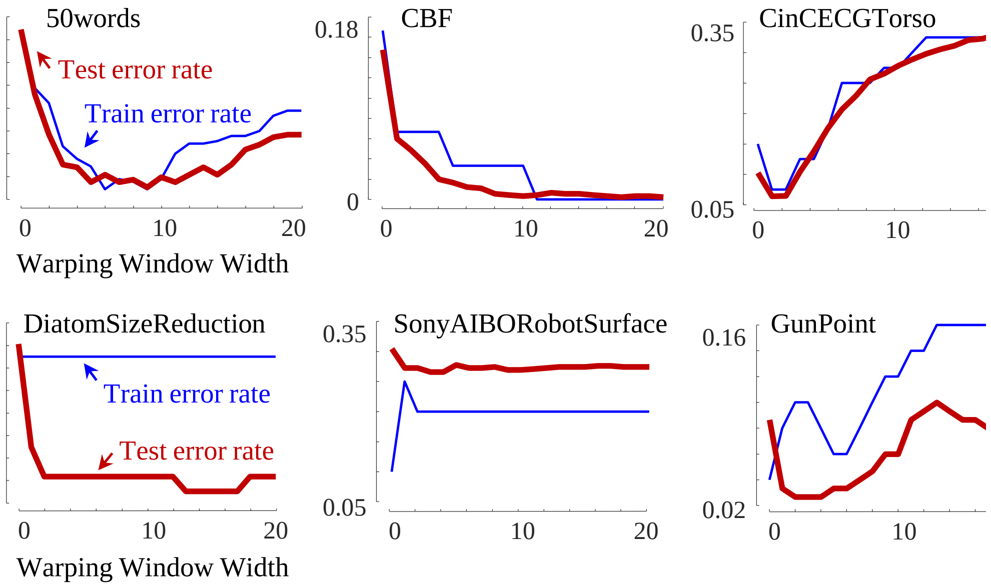

Figure 2.2: *blue/fine*) The leave-one-out error rate for increasing values of warping window w, using DTW-based 1-nearest neighbor classifier. *red/bold*) The holdout error rate. In the bottom-row examples, the holdout accuracies do not track the predicted accuracies.

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 to 2 degrees Celsius. 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. propose a time series forest ensemble method [33]. One of their reported successes is in halving the error rate on *GunPoint* to 4.7%. However, Figure 2.2.*bottom.right* 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, Gorecki and Luczak [47] introduce a new distance measure DDDTW 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 data sets 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 data set, 1-NN DTW can achieve a much better error rate of 23.28%.

In a follow-up paper [48], 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 data sets 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 non-linearity, 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. [64] and Kate [65]. Yet, in most cases, the community has been proposing rather complex methods for relatively modest gains. The results in Figure 2.2.*bottom* 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 optimization that allow them to scale to trillions of data points or to real-time deployment on resource-constrained devices [97].

In what follows, we formally introduce Dynamic Time Warping distance. We then discuss different factors affecting a good choice of warping constraints w and other observations, which lay the foundations for the methods we propose to learn the parameter w in the next chapters.

2.2 Dynamic Time Warping Distance

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 [5, 34]. In an influential paper [97], Rakthanmanon et al. 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 [93]. 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 [20]. While the archive is large and diverse, it reflects only distribution of data sets 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 data sets in the archive.

As illustrated in Figure 2.3.*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 i^{th} point of Q and j^{th} point of C. The warping path W is a set of contiguous matrix elements that define 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. 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.

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

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 Figure 2.3.*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 [104], which expresses w as a percentage of the time series length. We denote DTW with a constraint of w as cDTW$^{\mathrm{w}}$.

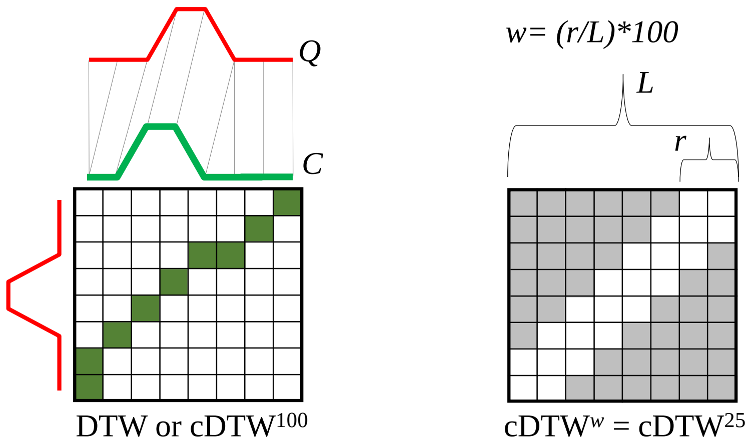

Figure 2.3: *left*) The unconstrained warping path for time series Q and C. Such warping paths are allowed to pass through any cell of the matrix. *right*) A constrained DTW. We can choose to constrain the warping path to avoid passing through cells that are too far from the diagonal.

The Euclidean distance between 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 constrained DTW (for any amount of constraint). Both bounds have been exploited by various clustering/classification algorithms and similarity search algorithms [12]. Interested readers can refer to other surveys [34, 108] and the references therein for more details.

2.3 Classic Learning of Warping Window Width

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 [20], 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 Archives 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 [5, 34, 48, 64, 65, 77, 101], by either explicitly implementing the method, or directly comparing results to the numbers published in the UCR Archive [20].

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 cDTW5, and 10%, for* $cDTW^{10}$*, of the length of the time series of each data set; this second case is more realistic for an unsupervised setting such as clustering"* [93].

However, as our motivating examples demonstrate (recall Figure 2.1 and Figure 2.2), 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/-data sets.

2.4 Factors Affecting the Best Warping Window Width

We note that the detailed discussion below of the factors affecting the best warping window for DTW classification are in the context of 1-nearest neighbor (1-NN) classification only. Undoubtedly the other classifiers that use DTW distances (e.g. some variants of Shapelets and DTW embedding methods [54])) 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 [4, 5, 34].

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 data set, which we call *Single Plateau* (SP). This data set (and all others in this paper) is available at the paper supporting web page [24]. Each item in the data set 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 five to twenty to each exemplar. If the plateaus location falls in the range of 1 to 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; Figure 2.4 shows examples from each class.

Note that while the SP data set is synthetic, it closely models several real data sets, 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 [61]. We will use this SP data set as a running example to demonstrate factors affecting the choice of the maximum warping window width in DTW distance.

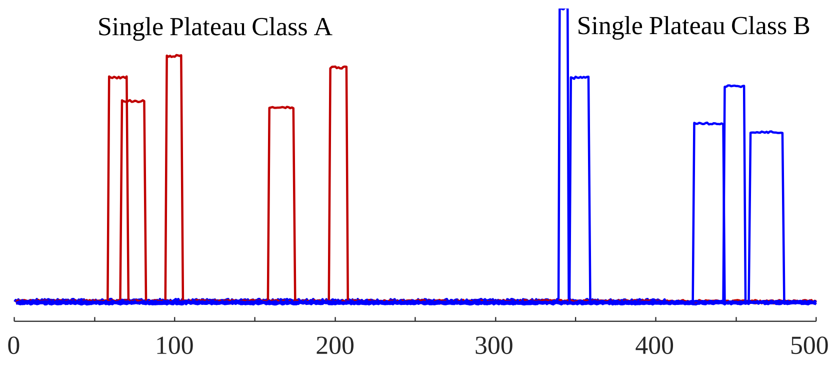

Figure 2.4: Five examples of each class of the *Single Plateau* data set (Class A and B).

2.4.1 The Intrinsic Variability of the Time Axis

If we cluster SP with cDTW^0, we obtain a "random" clustering as shown in Figure 2.5.*left*. This is not surprising, as this is clearly a data set 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 2.5.*middle*). 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 (Figure 2.5.*right*).

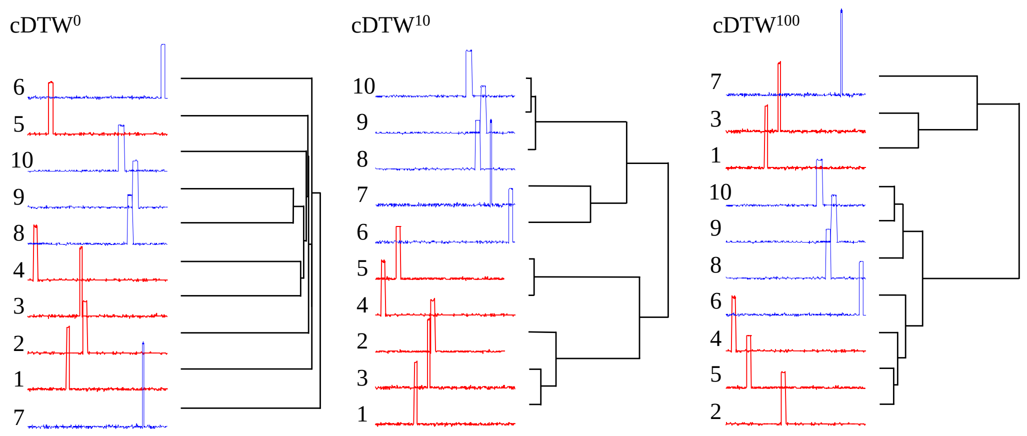

Figure 2.5: A hierarchical clustering result for the *Single Plateau* data set. Exemplars in Class A are numbered 1 to 5 and are shown in red. Exemplars in Class B are numbered 6 to 10, and are shown in blue. *left*) Clustering with cDTW^0 *middle*) Clustering with cDTW^{10} *right*) Clustering with cDTW^{100}.

This notion that *"a little warping is a good thing, but too much warping is a bad thing"* is known (although perhaps under-appreciated [101]) for time series *classification* [19]; however, we believe that this is the first explicit demonstration of the effect for *clustering*. 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.

2.4.2 The Size of the Data Set

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 data sets from the domain. For example, we might imagine that for the *gesture-recognition-for-tall-males* data set, cDTW^5

is generally best, but for the *heartbeat-classification-for-the-elderly* data set, 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 data set. To see this, we can classify increasing large instances of the SP data set. For each size, we search over all possible values of w and record the value that minimizes the error rate of LOO cross-validation. Figure 2.6 shows the result, averaged over 100 runs.

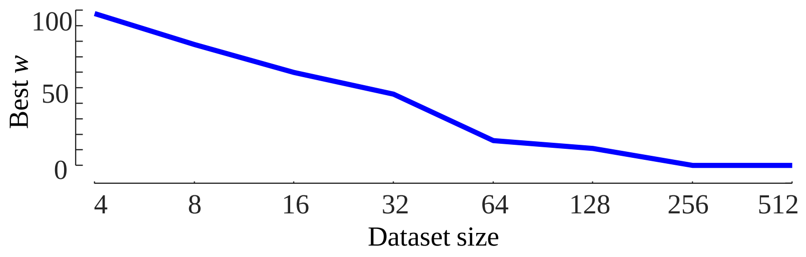

Figure 2.6: Classification of increasingly large instances of *Single Plateau* shows the effect of data set size on the best w.

Consistent with observations by Ratanamahatana and Keogh [101], small data sets 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 data sets. Thus, we cannot generalize the best setting for w on one subset of a data set to a different-size subset of the same data set.

As shown in Figure 2.6, the best value for w on this data set, given that it contains 32 objects, is 46. Let us further consider this particular sized subset of the training set. Figure 2.7 displays the effect of w on the mis-classification rate of the 32-object SP data set. We can see that allowing too much warping is almost as detrimental as too little warping.

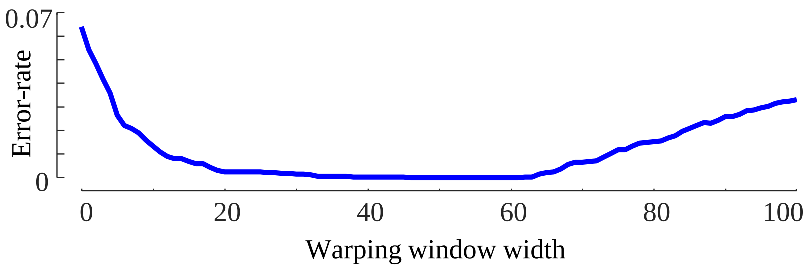

Figure 2.7: Classification of 32-objects *Single Plateau* demonstrates effect of w on LOO error rate. Average result of 100 runs.

In this case, the w vs. 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 mis-classification. However, as Figure 2.7 suggests, this curve can take on more complex shapes, which makes the choice of w more critical.

2.4.3 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 data set, 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 data sets, *Slim Plateau* and *Broad Plateau*, which, as their names suggest, differ only in the width of the plateaus (see Figure 2.8). In both data sets, one class has a plateau in the first half, and the other class has a plateau in the second half.

As shown in the leftmost column of Figure 2.8, 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 data sets and clustered them again using cDTW^0? (We will explain how we can add synthetic warping to a time series in Chapter

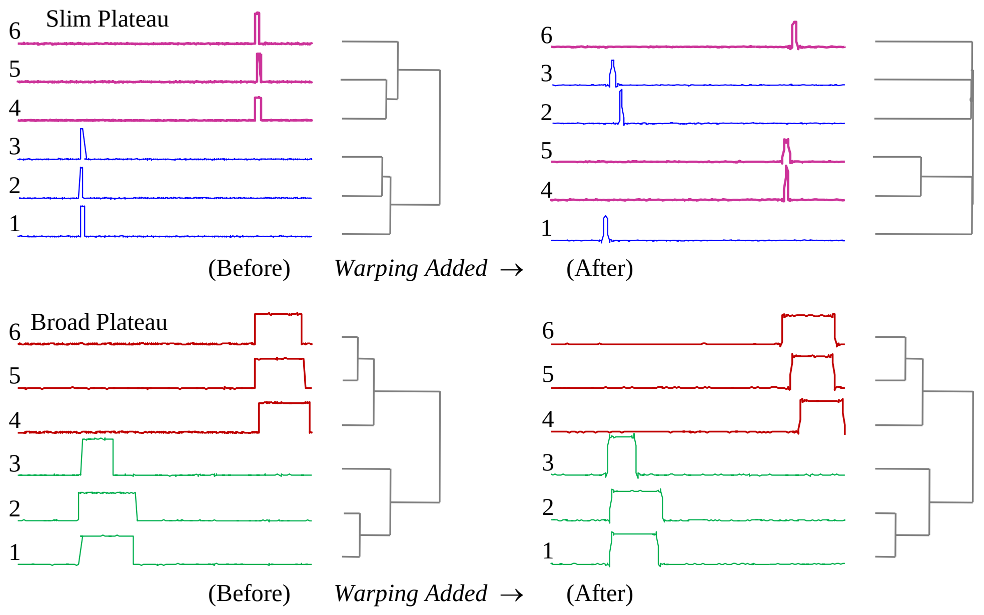

Figure 2.8: Warping affects different data sets differently under hierarchical clustering. *top*) The clustering of the *Slim Plateau* data set is very brittle when the time axis is warped. *bottom*) In contrast, the *Broad Plateau* data set is extremely robust to *identical* levels of warping.

4 - Section 4.4.2). As we can see in the rightmost column of Figure 2.8, 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 data set 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 data set.

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 vs. error rate curve for heartbeats or for gestures. We must learn this curve on a case-by-case basis.

2.4.4 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 [93], we will demonstrate that it is unwarranted. In Figure 2.9, we show both the Rand-Index and the accuracy for two data sets.

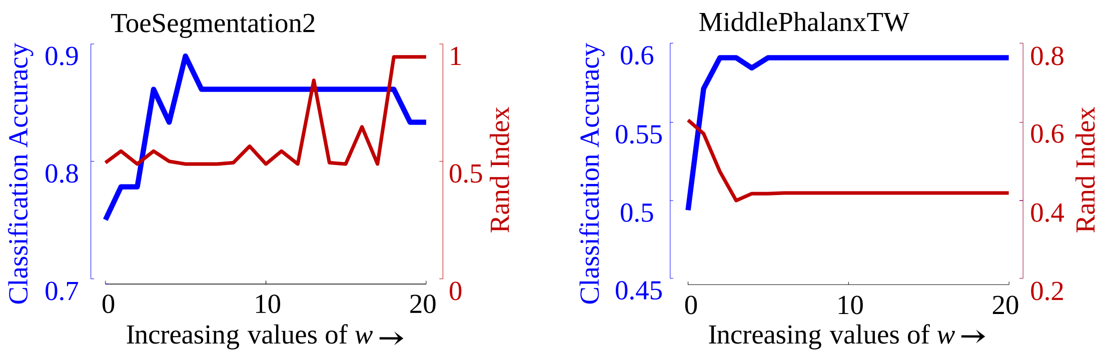

Figure 2.9: The Rand-Index (red/fine) and the classification accuracy (blue/bold) vs. the warping window width for two representative data sets.

In retrospect, it is unsurprising that these values are weakly related. For 1-NN classification (the most commonly used classification technique in the literature [34, 101], 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.

2.5 Conclusions

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 data sets, 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.

We have made several other observations that are novel, or at least under-appreciated. We have shown that w depends not only on the data object shapes, but also on the number of data objects considered. This observation has been previously made for classification [101], but not for clustering. We have also demonstrated that the optimal setting for w for classification is not generally the optimal setting for clustering, an assumption that has appeared in the literature [93].

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 Chapter 3, we offer a semi-supervised method to learn w for time series clustering. In Chapter 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. 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 [24].

Chapter 3

A Semi-Supervised Method to Learn DTW's Warping Window Width for Time Series Clustering

3.1 Introduction

We begin by formalizing the task at hand:

> **Problem Statement**: Given an unlabeled time series data set 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 data sets 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 [3, 8, 7, 31, 124]. 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 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, 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 data set; therefore, we can produce the final clustering using essentially[1] any partitional, hierarchical, spectral, or density-based clustering.

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

- 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.

3.2 Background and Related Work

3.2.1 Semi-supervised Learning

Due to its demonstrated utility in many practical applications, the semi-supervised learning paradigm has drawn in significant attention in the data mining and machine learning communities over the last decade [3, 7, 31, 124]. 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 [124]. 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. [31] 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 [7], 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 are rarely confused [14].

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.

3.2.2 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. [12], which is a specialization of the Density Peaks algorithm [102] for DTW. This algorithm is suited to DTW, since it does not require metric properties, and it is particularly amenable to optimization to its scalability by exploiting both upper and lower bounds of DTW [12].

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 [12] that we do not consider this option for simplicity.

3.2.3 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 [122]; however, the classic Rand-Index [99] 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 data set. 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:

$$\text{Rand-Index} = \frac{(\text{TP+TN})}{(\text{TP+FP+FN+TN})}$$

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.

3.2.4 Related Work

Zhou et al. recently introduced a paper entitled *"Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning"* [135]. 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 data sets 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 Section 3.4, 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 [100] and the references therein. We further briefly review some of the most recent, high-visibility efforts in time series clustering in Section 3.4.4 before the direct empirical comparisons to our proposed algorithm.

3.3 Our Approach

3.3.1 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

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

attention. For every possible pair of time series in our data set, 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 i^{th} position in C indicates that the pair of time series was not linked under $\mathrm{cDTW}^{\mathrm{i}}$, whereas a '1' indicates that it was linked.

In Figure 3.1, 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 most data sets 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 data sets that need warping in-variance, 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

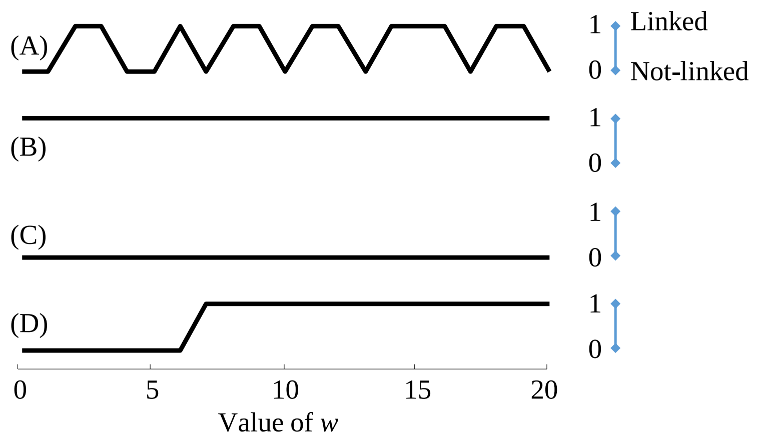

Figure 3.1: Four representative constraints. (A) a vacillating constraint, (B) an *always* linked constraint, (C) a *never* linked constraint, and (D) an ideal constraint.

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.

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

Our algorithm for finding the set of constraints that we will ask the user to evaluate is presented in Algorithm 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 3.2 illustrates examples of time series from the *Trace* data set that are shown to the user. We hope to avail of the users domain knowledge, intuitions, and pattern recognition ability. For Figure 3.2.*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.

Algorithm 1 Algorithm for finding the constraint set

Input: the set of candidate constraints, maximum number of constraints to get annotated

Output: UA, the set of user annotations

```
1: $constraints \leftarrow$ sort_by_simplicity($constraints$)
2: $index \leftarrow 1$
3: while $\neg$ empty($constraints$) AND $loop\_count \leq max\_number\_of\_constraints$ do
4:     $UA_{index} \leftarrow$ get_user_annotation($constraints_{index}$)
5:     $answer \leftarrow$ get_user_willingness()                    ▷ (‘Do Another? Y or N’)
6:     if $answer =$ ‘Y’ then
7:         $index \leftarrow index + 1$
8:     else
9:         $index \leftarrow infinity$                                ▷ break out of loop
10:    end if
11: end while
```

Therefore, we would expect the user to annotate this as *"must-link."* In contrast, for Figure 3.2.*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 Section 3.4.2. One idea would be to add a third option *"skip this annotation"* to the list of possible answers. For simplicity, we ignore this possibility in this work.

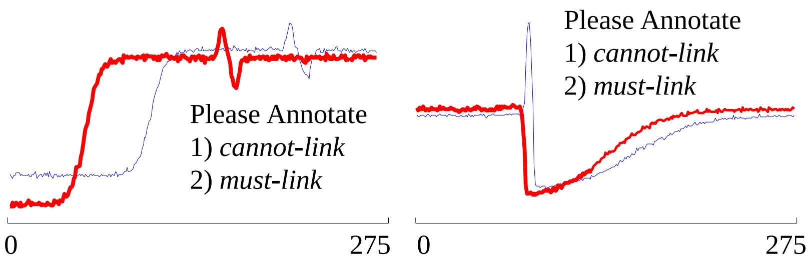

Figure 3.2: Examples of pairs of time series from the *Trace* data set presented to user for annotation. User has to decide whether the two time series should be in a same cluster or not. *left*) The correct label is *must-link* as most of the difference between the two time series is from warping in the time axis. *right*) The user should choose *cannot-link* because one time series is missing the short peak that characterizes the other time series.

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 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 user annotation constraints satisfied (i.e. correctly clustered by our chosen clustering algorithm) over the total number of UA constraints available.

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

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 3.3 shows an example of this for two data sets.

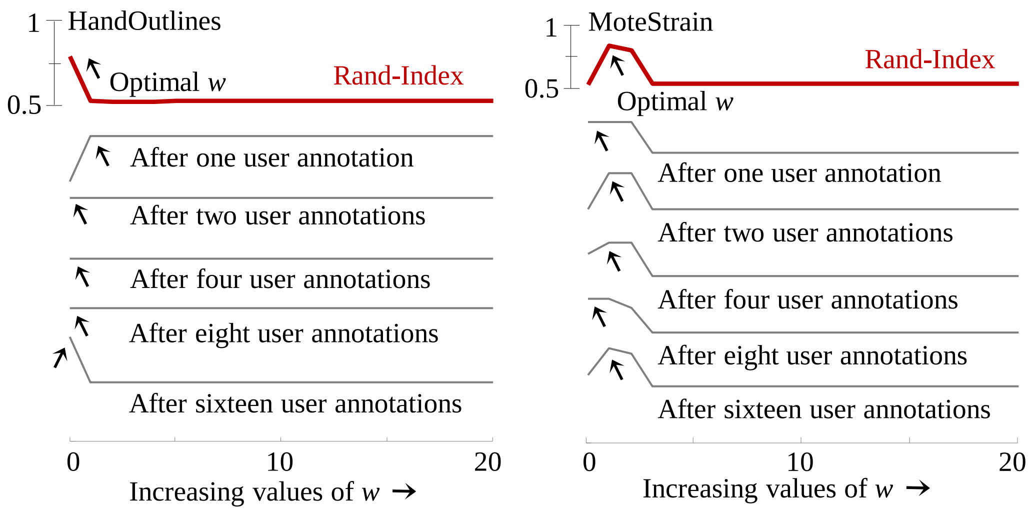

Figure 3.3: For the two data sets *HandOutlines* and *MoteStrain*: The ground truth Rand-Index (colored/bold line). The prediction vectors (light/gray lines) learned after 1 to 16 user annotations allow us to estimate w (arrows). The shapes of the prediction vectors reflect the ratio of constraints satisfied (correctly linked or not linked) at each w.

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, 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.

3.3.2 Pseudo User Annotation

As the results in Figure 3.3 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?

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 [84]. 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 data set. For each object O, we create a copy of it that we denote as $\overline{\mathrm{O}}$. We add some warping to $\overline{\mathrm{O}}$ and place it into the data set with the (pseudo) constraint *must-link*$(\mathrm{O}, \overline{\mathrm{O}})$. Since we know that object $\overline{\mathrm{O}}$ is just a minor variant of O, we can safely assume that if $\overline{\mathrm{O}}$ 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 like those produced in Algorithm 1. This idea seems to have a tautological paradox to it. It seems that if we *add* w amount of warping to the data set, we will *discover* w warping in that data set. However, this is not the case, as discussed Chapter 2 - Section 2.4.3. Algorithm 2 contains procedure for generating pseudo constraints.

Algorithm 2 Algorithm for finding the pseudo constraint set

```
Input: $D$, the data set to be clustered
Input: $M$, the amount of warping to add
Output: $Dnew$, a new version of data set $D$
Output: $PUA$, the set of pseudo user annotations for $Dnew$
1: $Dnew \leftarrow$ random_shuffle($D$)
2: for $i = 1$ in steps of 2 to number_of_instances($Dnew$) do
3:     $Dnew_{i+1} =$ add_random_warping($D_i$)                    ▷ See Table 3.1
4:     $PUA_{(i+1)/2} =$ set_constraint($Dnew_i, Dnew_{i+1}$, '$must-link$')
5: end for
```

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 data set D_{new}, which is the same size as D. This is important, as the size of the data set affects the best setting for w (recall Chapter 2 - Section 2.4.2).

The algorithm also outputs a set of pseudo user annotations (PUA) for D_{new}. PUA is essentially identical to UA produced in Algorithm 1 except its annotations are produced without human interventions. Note that with this method of generating annotations, all PUAs are *must-link* constraints. Figure 3.4 shows some examples of time series with warping added, and Table 3.1 contains the actual MATLAB code used to add warping. We call this variant of our ideas the PUA (Pseudo User Annotation) algorithm.

Table 3.1: 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
```

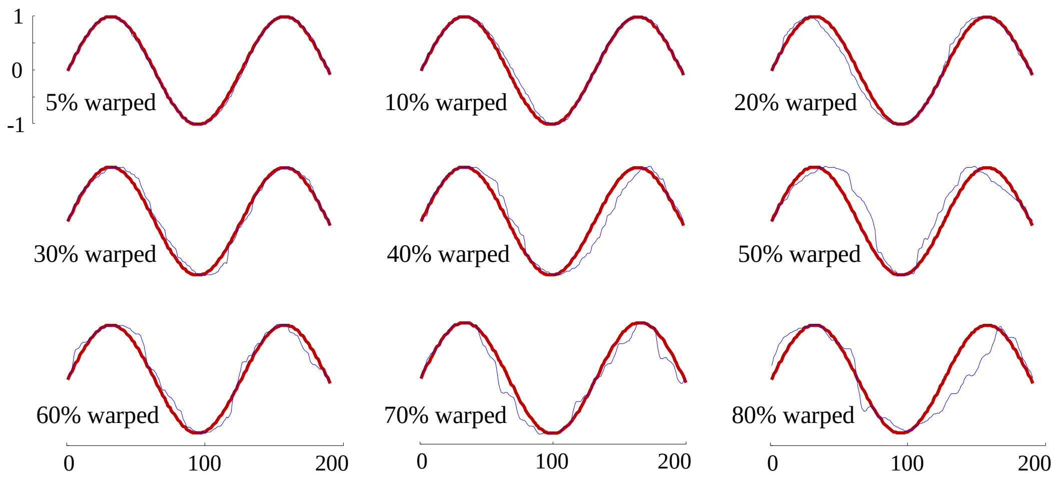

Figure 3.4: From left to right, top to bottom: Increasingly warped versions of a sine wave. The red/bold curve is the original, and the blue/fine curves are the ones with added warping. The "percentages" have no absolute interpretation; they only allow a relative understanding of the amount of warping added.

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 3.5 shows how this idea works with *Trace* and *TwoPatterns*. Here we use 64 out of 1,824 pseudo constraints available for *TwoPatterns* 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 *TwoPatterns*, any warping amount in the range 5 65% allows us to estimate the correct w.

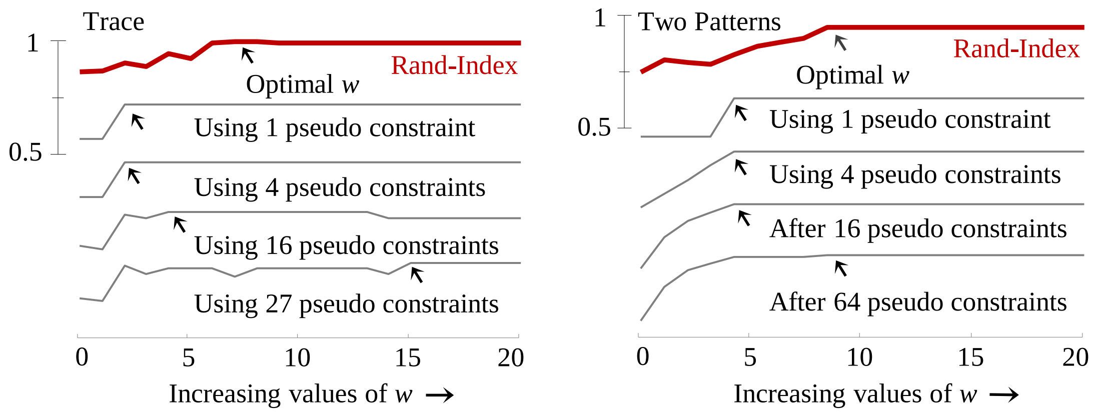

Figure 3.5: *Trace* and *TwoPatterns*'s prediction vectors using pseudo constraints provided by the PUA algorithm.

3.3.3 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 [84]. 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 optimization here for brevity, because the simplest version of our ideas is already very competitive.

3.4 Empirical Evaluation of Using Prediction Vector for Setting w for Time Series Clustering

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 Section 3.4.4, we explicitly compare TADPole by using the learned warping window to five recent state-of-the-art clustering algorithms.

3.4.1 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. use cDTW^0 [35], and Paparrizos and Gravano [93, 94] 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 [12], is identical for all approaches and completely deterministic [102]. Thus, any improvements obtained can be completely attributed solely to our ideas.

We can measure *success* as follows. For each data set, 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 data sets in the UCR Archive do not require w greater than 10% [101]). For example, in Figure 2.1, the maximum Rand-Index is 1.0 for *TwoPatterns* and 0.89 for *SwedishLeaf*. 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 one minute. We summarize the result in Table 3.2. With sixteen labeled constraints, we achieve success of 46 out of 102 data sets, 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 data sets. 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.

Table 3.2: Summary of number of successes on 102 data sets 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

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, CricketY, NonInvasiveFetalECG2*, and *FiftyWords* data sets, we were within at least 0.98 of the optimal.

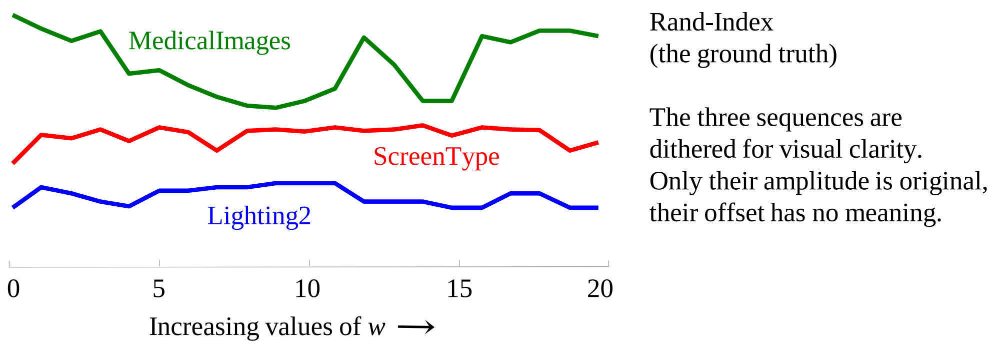

Figure 3.6: The Rand-Index vs. the warping window width for three small data sets. Contrast the variability of the curves with the relatively smooth curves shown in Figure 2.1.

However, in some cases, we do achieve significantly worse than the optimal. Essentially, all such cases can be attributed to very small data sets (or small, relative to the number of clusters). As shown in Figure 3.6, this tends to result in clusterings that are very unstable with small changes in w. The fact that small data sets have poor stability when clustered is well known [123], 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 data sets, we do not do worse than the lower scoring of our two rivals.

3.4.2 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.

As shown in Figure 3.7, for the *ItalyPowerDemand* and *MiddlePhalanxOutlineAge-Group* data set, 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* data set and 1 for the *MiddlePhalanxOutineAgeGroup*, which concurs with the objective ground truth.

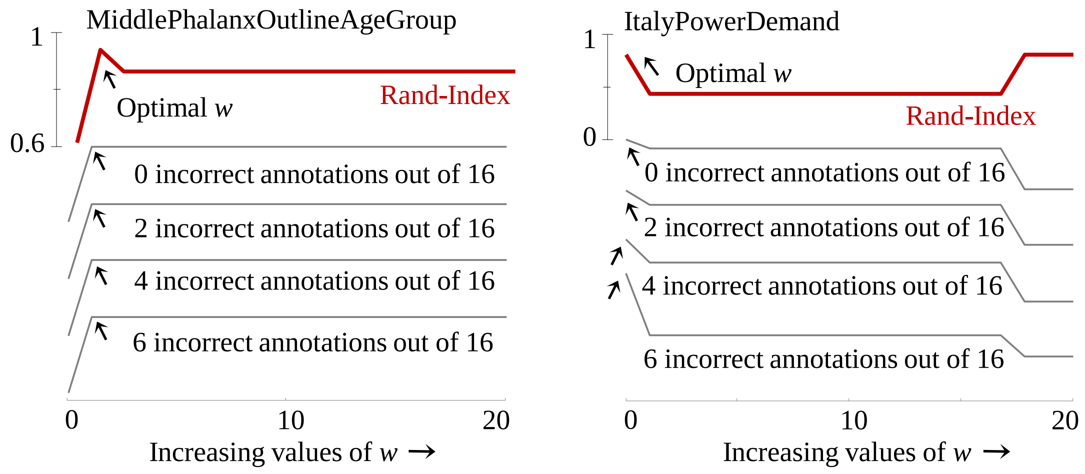

Figure 3.7: Robustness to incorrect constraints. In each case, 16 pairs of time series are presented for annotation. The annotator may incorrectly label a pair that should have been *must-link* as *cannot-link* and vice versa. Our algorithm is robust to these mistakes.

As 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.

3.4.3 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 [108]. 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 $\mathrm{DTW_I}$ [108], which allows each dimension to warp independently. Let Q and C be two multi-dimensional time series of M dimensions. $\mathrm{DTW_I}$ defines their DTW distance as the sum of independent DTW distances between each dimension.

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

In Figure 3.8, we consider the 4,480-objects, three-dimensional *UWave* data set [81], which has become a benchmark for gesture recognition in the last five years. We also consider the *Handwriting Accelerometer* data set 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 data set in the literature. Unfortunately, most papers do not explicitly state the value of w used, but the three most

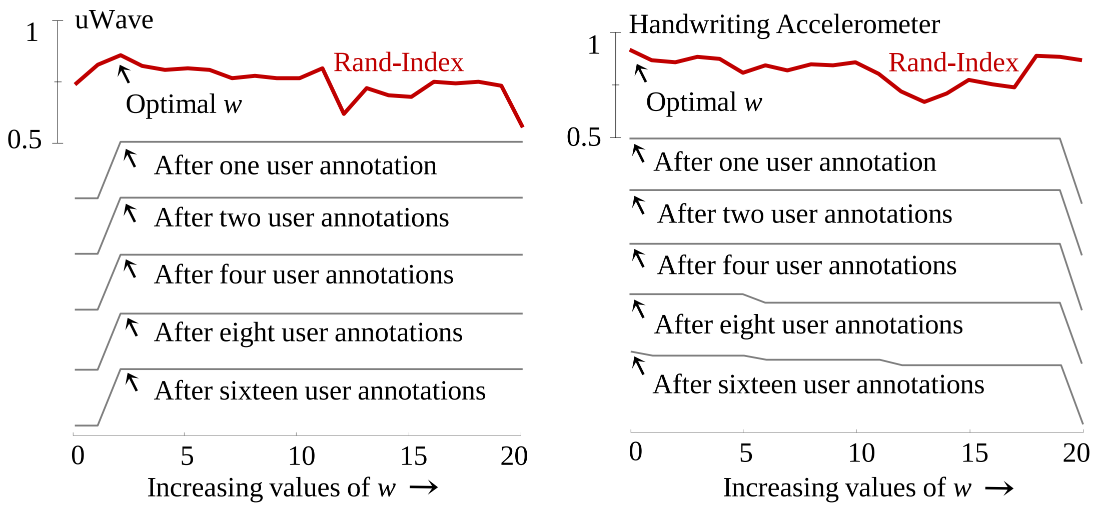

Figure 3.8: Three-dimensional *uWave* and *Handwriting Accelerometer* data set clustered with $\mathrm{DTW_I}$.

common settings, cDTW^0, cDTW^{10}, and cDTW^{100} are all sub-optimal to widely differing degrees.

3.4.4 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"* [93]. 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 [93, 94], 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* data set, they compare five clustering algorithms using DTW vs. 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 *TwoPatterns*. 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 [74], 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 readers to the original paper. The authors demonstrate the utility of their work on the publicly available *MOCAPANG-Subject-35, right-foot-marker* data set. 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."* 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.

[3]For conditional entropy, smaller is better

To be clear, we are not claiming the work proposed by Li and Prakash [74] 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 data sets the original authors used to validate their method.

A recently published work measures the accuracy of eleven carefully optimized clustering algorithms on the *Trace* data set, of which eight use DTW as the distance measure [37]. 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.

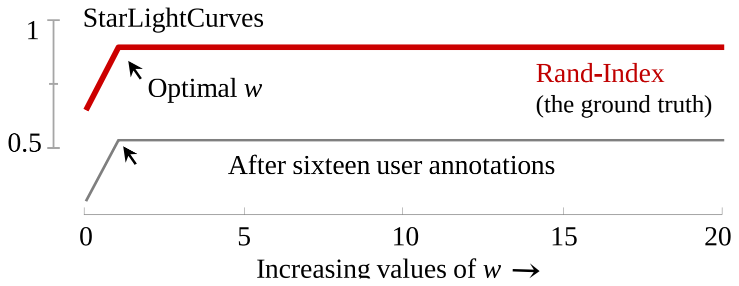

Figure 3.9: The Rand-Index vs. the warping window width for *StarLightCurves*. We predict $w = 1$, obtaining a Rand-Index of 0.83, equivalent to a NMI of 0.79.

Another recently published time series clustering technique called YADING is shown to *"provide theoretical proof which ...guarantees YADINGs high performance"* [35]. However, these guarantees are only with respect to Euclidean distance. The only publicly available real data set they test on is *StarLightCurves*, for which they obtain a NMI score of 0.60. However, as shown in Figure 3.9, 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 seconds to cluster this data set. However, this data set took several years to collect, and many days of careful human effort in preprocessing. Given that, the difference between taking 3.1 seconds and taking 30 seconds to do the clustering seems completely inconsequential (but also see Section 3.4.5). Of course, the authors are correct in noting that there is sometimes a need for better speed and scalability. However, in many domains, the trade-off between speed and accuracy will still favor accuracy. For example, in the UCR Archive, many data sets 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 [134]. These algorithms yield NMI scores of 0.53, 0.45, 0.54, 0.64 for the *Trace* data set, but our PUA algorithm can achieve an almost perfect NMI score of 0.97 (Rand-Index = 0.99) on this same data set.

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.

3.4.5 Scalability

At first, our algorithm appears to require a significant overhead in time complexity, given that the Density Peaks algorithm [102] 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, which is a specialization of the Density Peaks algorithm for DTW that exploits the tight upper and lower bounds for $\mathrm{cDTW^w}$ 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 $\mathrm{cDTW^w(Q, C)}$ is a very tight lower bound for the value of $\mathrm{cDTW^{w+1}(Q, C)}$. Thus, we can perform the clusterings in order, from $w = 0$ to $w = 20$, at each stage by using any $\mathrm{cDTW^w}$ 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. [113] to dismiss unpromising alignments.

Finally, there are a wide variety of DTW implementations, and the efficiency differences between them overshadow the small overhead of our approach. For example, a recent paper that tests a DTW-based clustering on some of the data sets we consider notes: *"several experiments were unable to return results within 20 days"* [134]. However, we can cluster these same data sets in at most minutes, at least 10,000 times faster.

3.5 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 hand-held device [51]. 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 log-in for home sharing Netflix is an example.

The data set was kindly shared by the authors of the paper [51], 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 [24]. The data set 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 way, just as they would normally do in their everyday life. Figure 3.10 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 Figure 3.11.

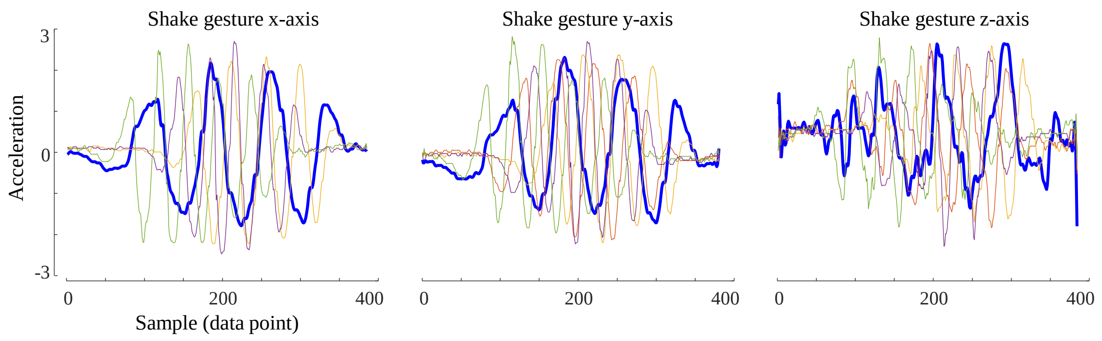

Figure 3.10: Five examples of a shake gesture captured with x, y, and z-axis acceleration. Time series of same color correspond to one specific instance. One instance is highlighted for visual clarity (blue/bold time series). The sampling rate is 100Hz.

We re-sample 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 data set to see how well each time series cluster characterizes an individual user. Figure 3.11.*top* displays the Rand-Index if we have access to the true label. It shows that the highest clustering quality for this data set 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 (Figure 3.11.*bottom*). We count this a success, because the achieved Rand-Index scores 0.99 of the optimal Rand-Index.

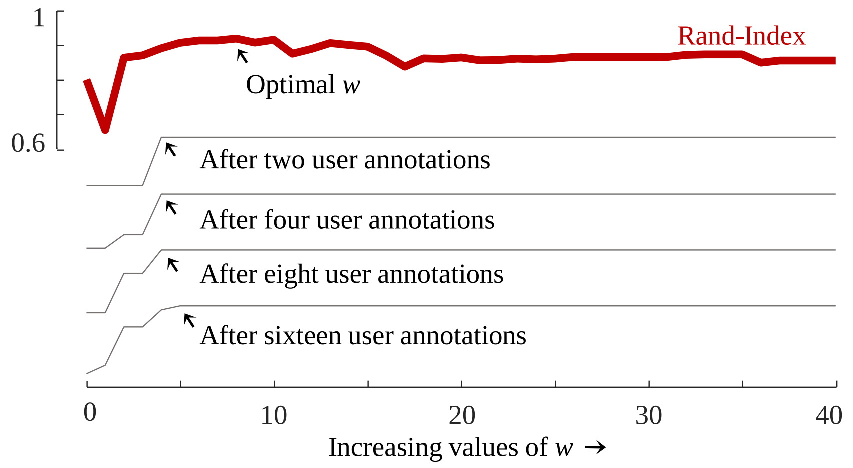

Figure 3.11: Clustering result with TADPole (red/bold) and prediction vectors (grey/thin). With 16 user annotations, the algorithm suggests $w = 5$, which gives a Rand-Index of 0.91, being 99% of optimal.

In retrospect, this is clearly a data set that would benefit from warping in-variance. 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.

3.6 Conclusions

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 data sets, 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.

Two main attractivenesses set our work aside from other methods. First, it requires only very few annotations, because we rank the annotations in order of their importance before soliciting user experts. This contrasts with all other semi-supervised methods that require the labels up-front, often asking user to annotate pairs that will make little or no difference to the search space. Second, our method is clustering algorithm agnostic. We are merely learning the best w for a particular data set; the framework can work with any clustering paradigm.

Finally, we have released all our code and data in a public repository [24], to allow others to confirm, extend and exploit our ideas.

Chapter 4

A Resampling Method to Learn DTW's Warping Window Width for Time Series Classification

4.1 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 data sets.

However, the uneven distribution of classes is not an issue here, since all the data sets 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-nearest neighbor (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 data sets [20]. There are now many independent comprehensive empirical studies demonstrating a strong performance of NN-DTW [4, 34, 77]. 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 though: *"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"* [4] or *"the received wisdom is that DTW is hard to beat"* [5].

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 set. Resampling is normally ill advised in small data sets, 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.

4.2 An Intuition to Our Proposed Approach

To understand the effect(s) of data set 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 *TwoPatterns*, a data set we wish to classify with 1-NN DTW. Because it has 1000 training objects, we will denote it as $TwoPatterns^{1000}$. As shown in Figure 4.1.*left*, $TwoPatterns^{1000}$ is a data set in which we can correctly learn the best maximum warping window with cross-validation.

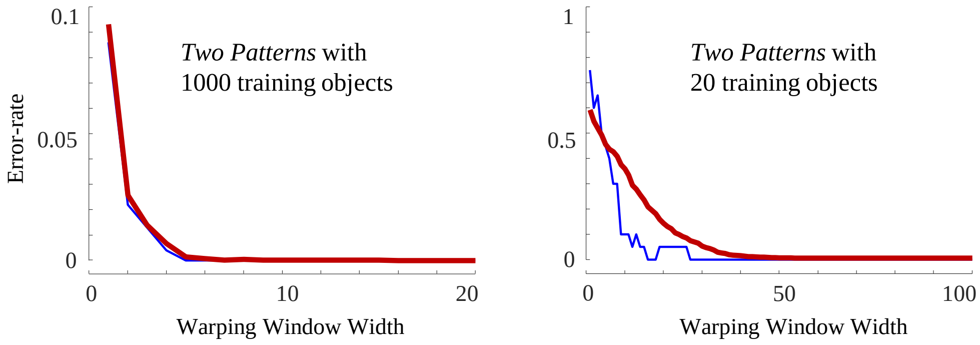

Figure 4.1: The 1-NN DTW LOO error rate (blue/thin) and the holdout error rate (red/bold) for increasing values of w. *left*) $TwoPatterns^{1000}$ data set *right*) $TwoPatterns^{20}$ data set.

Suppose the data set had significantly fewer training instances; we will call this data set $TwoPatterns^{20}$. We would expect that the holdout error rate would increase, and we were advised by Ratanamahatana et al. that we should expect the best value for w to go up slightly [101]. As we can see in Figure 4.1.*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, Figure 4.1.*right* strongly resembles some of the plots shown in Figure 2.2.*top*, and for the same reason, we do not have enough training data.

Let us further suppose that while we are condemned to using $TwoPatterns^{20}$ 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 Figure 23.*left*, we will learn the best value of w for $TwoPatterns^{1000}$, not for $TwoPatterns^{20}$, which is our interest.

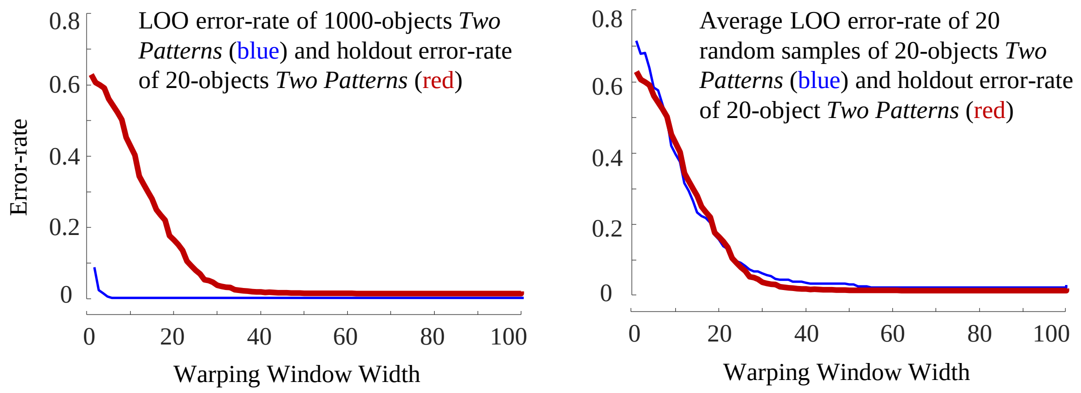

Figure 4.2: *left*) The 1-NN DTW LOO error rate of the $TwoPatterns^{1000}$ data set is a poor predictor of the holdout error on $TwoPatterns^{20}$. right) In contrast, the average LOO error rate of 20 random samples of $TwoPatterns^{20}$ is an excellent predictor of the holdout error on $TwoPatterns^{20}$.

The solution suggests itself. Performing cross-validation with $TwoPatterns^{1000}$ gives us low variance, but it is biased toward the wrong value of w. In contrast, doing cross-validation with $TwoPatterns^{20}$ is biased toward the correct value for w but has high variance. If we resample many subsets of size twenty from $TwoPatterns^{1000}$, do cross-validation on each, and average the resulting w vs. 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 Figure 4.2.*right*, this is exactly the case.

The observations above seem to be non-actionable. In general, we do not have 1,000 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, 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.

4.3 Background and Related Work

4.3.1 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. [71] 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. [45] 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"* [4]. 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.

4.3.2 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:

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

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.

4.3.3 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 [9, 18, 55]. 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 [17], creating synthetic time series data mining under DTW has only been explored recently [39, 95].

A recent paper also generates synthetic exemplars by adding warping to existing objects [49]. However, this is to mitigate convolutional neural network (CNN) weakness *"... that they need a lot of training data to be efficient"* [49]. The synthetic examples do help improve accuracy over the non-augmented data sets; still, it remains unclear if CNNs are generally competitive for time series problems [4], and this issue is orthogonal to the claims of this work.

4.3.4 Related Work

The more general idea of creating synthetic data to mitigate the problems of imbalanced data sets [18] or to learn a distance measure [52] is well-known. However, we are not aware of any other research suggesting a warping 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.

4.4 Our Approach

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 vs. w curve. Use the average of all N curves to predict w.

This procedure, outlined in Algorithm 4, contains a subroutine presented in Algorithm 3. 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 3. These data sets will be used instead of the original training set to learn w. While each data set may produce a noisy error vs. w curve (as in Figure 4.1.*right*), the average of all such curves will be smoother, and it will more closely resemble the true noisy error vs. w curve (as in Figure 4.2.*right*).

Algorithm 3 Algorithm for making augmented training set

```
Input: $D$, the original data set with $n$ objects
Input: $M$, the amount of warping to add
Output: $R$, the ratio of synthetic objects to create
Output: $Dnew$, a new version of data set $D$
1: $real\_objects \leftarrow$ random_sample($D, (1 - R) * n$ objects)
2: $fake\_objects \leftarrow$ random_sample($D, (R * n)$ objects)
3: for $i \leftarrow 1 : 1 :$ number_of_instances($fake\_objects$) do
4:     $fake\_objects_i \leftarrow$ add_warping($fake\_objects_i, M$)
5: end for
6: $D_{new} \leftarrow [real\_objects; fake\_objects]$
```

As shown in Algorithm 3, 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 data sets, we run the risk of only adding warping to one class and possibly skewing the results.

Algorithm 4 Algorithm for finding the warping window width

```
Input: $D$, the original train set
Output: $w$, the predicted best warping window
1: for $i \leftarrow 1:1:number\_of\_iterations$ do
2:     $Dnew \leftarrow$ make_new_train_set($D$)                ▷ See Algorithm 3
3:     for $j \leftarrow 1:1:maximum\_warping\_window$ do
4:         $error\_rate_{i,j} \leftarrow$ run_cross_validation($Dnew$)
5:     end for
6: end for
7: $mean\_of\_all\_iterations \leftarrow$ mean($error\_rate$)
8: $[min\_value, min\_index] \leftarrow$ min($mean\_of\_all\_iterations$)
9: $w \leftarrow min\_index - 1$
```

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 Algorithm 4. 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), 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 $w = 0$.

4.4.1 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 Section 4.1.9 and verified empirically.

4.4.2 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 (Section 3.3.2, Table 3.1). We non-linearly 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 4.3 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

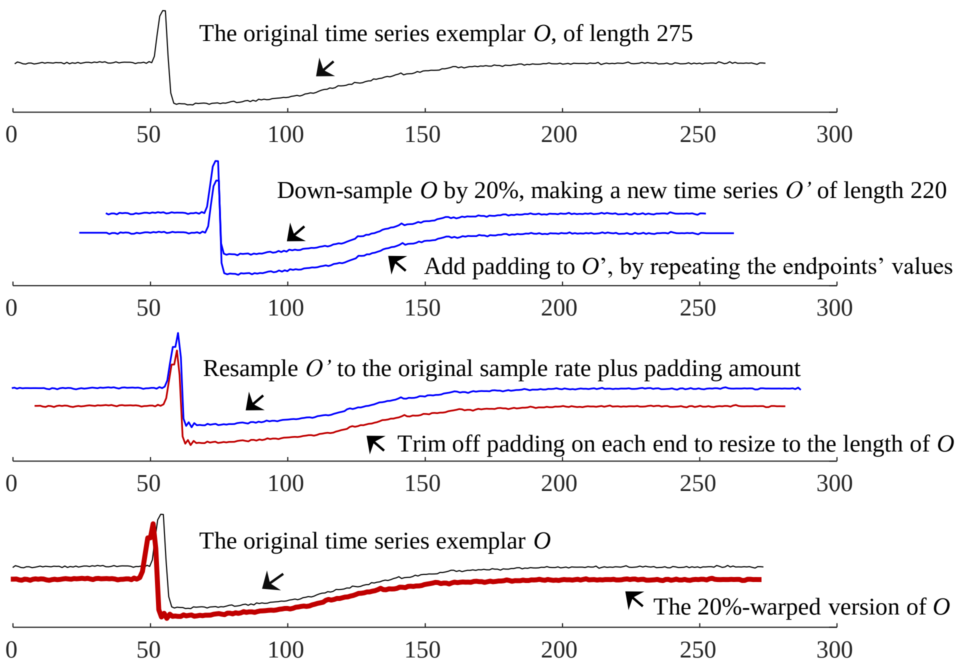

Figure 4.3: Adding 20% warping to an exemplar of *Trace*. Note that in the bottom panel, the generated times series (red/bold) is a slightly warped version of the original time series (black/fine).

the final time series later (Figure 4.3.*middle*). It is important to note, as a recent work indicates, that the endpoints can result in misleading DTW distance [112]. 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 4.1 contains a small modification of Table 3.1 to reflect these changes. Even though "without padding still brings about reasonably good results, findings from our experiments presented in Section 4.5 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 to generate synthetic time series [18, 39, 95]. In brief, there are *dozens* of methods to produce

Table 4.1: "Improved" 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
```

synthetic examples (averaging, grafting, perturbing etc.), and many of these ideas work well [36]. We chose the method shown in Table 4.1, because it is simple and effective.

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

Recall our previous discussion, at first glance, the idea of adding warping seems to have a tautological air to it. However, introducing w amount of warping to a data set does not necessarily mean that we will later discover w warping (see Section 2.4.3). 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* data set. We ran our algorithm (Algorithm 3) on this data set multiple times, changing only M, the amount of warping added to make synthetic exemplars (Algorithm 3 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 Figure 4.4, 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.

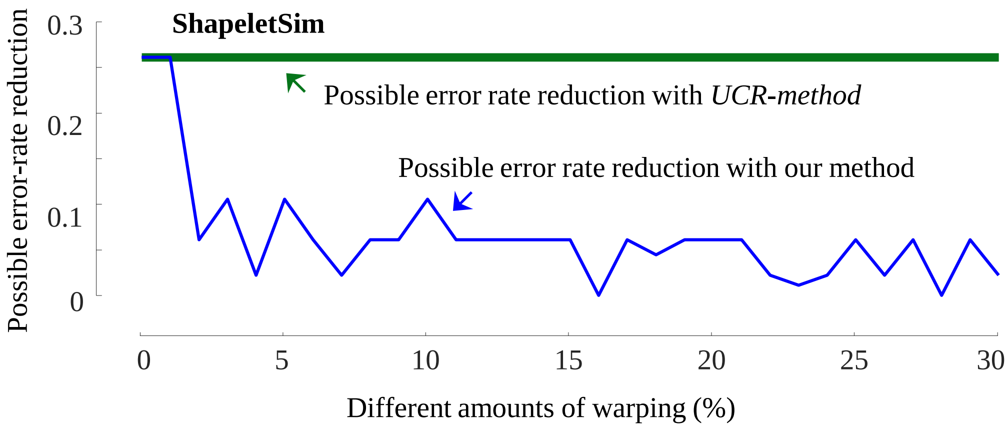

Figure 4.4: Effect of the warping amount on the possible error rate reduction. The vertical axis shows the difference between the error rate achieved by the w learned and the error rate achieved by the best w for this data set. Possible error rate reduction is synonymous with room for improvement. Adding warping helps if the blue/fine line is below the green/bold line.

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 data sets. Making the majority of objects in the newly constructed training set synthetic yields more diversity and variance in each training cycle. This value is hard-coded 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 hard-coded settings for all the data sets in the UCR Archive is an opportunity cost; an adaptive approach *could* be better. However, our strategy guards against over-fitting. 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.

4.4.4 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 [90], 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 data set that is only half the size of the data set we have. As we explained in Section 4.2, 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 trade-off 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 data sets 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 minutes for *GunPoint*, given that we perform 100 iterations for all w from 0-100%. 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 data set, regardless of its size, does not exceed 60. This applies for data sets discussed in Section 4.5.1, which are framed around small training set problem, not the original UCR splits.

4.5 Empirical Evaluation of the Resampling Method to learn w for Time Series Classification

4.5.1 Data Sets

We use the UCR Time Series data sets for our experiments [20]. As of February 2018, the UCR Time Series Archive has 85 data sets from various domains, has served as the benchmark for the time series community, and is widely referenced in the literature. A more comprehensive version of the UCR Archive together more baseline results is hosted by

Bagnall et al. [6]. At the time of this research project, the latest UCR Archive was from its Summer 2015 release, and it is this version we used for the experiments presented below.

As we have demonstrated in Figure 4.1.*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 data sets. Some of the train/test splits in the UCR data sets have large enough training sets that our ideas do not offer any advantages. Rather than ignoring these data sets, 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 data sets do not have enough ten objects per class, we exclude them from the experiment (the excluded are: *OliveOil, FiftyWords and Phoneme*). Therefore, we are left with 82 data sets. These new splits are published in the paper supporting web page [24] 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.

4.5.2 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 Algorithm 4 to learn the warping window size, we classified the holdout test data on the training set with 1-NN. Figure 4.5 and Figure 4.6 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.

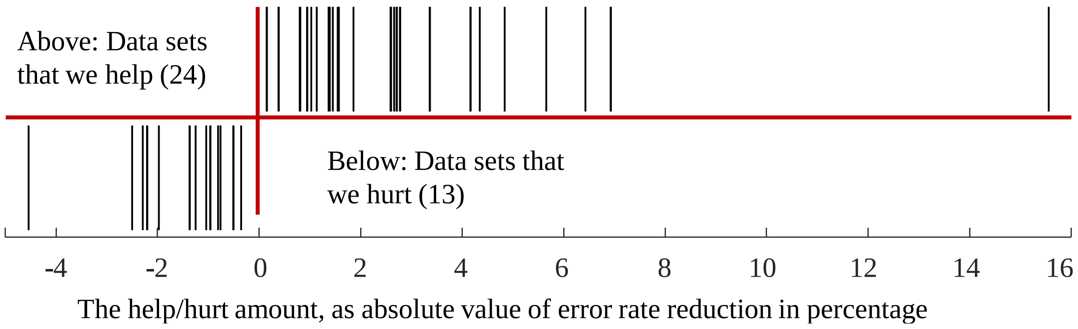

Figure 4.5: Help/hurt amount. The number of data sets that we help is nearly twice the number of data sets that we hurt.

Given this nomenclature, we can say that of the 82 data sets 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 Figure 4.5.

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 data set (found by

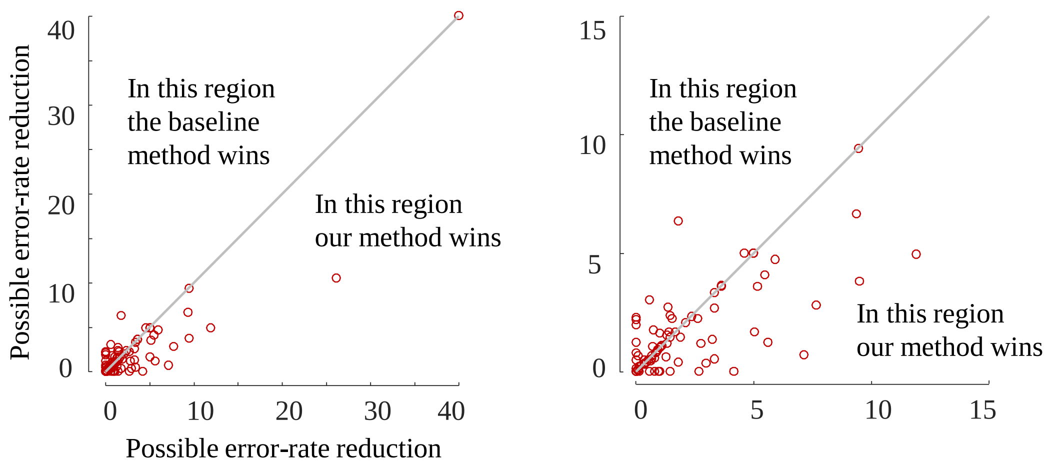

Figure 4.6: Possible error rate reduction (how close a methods error rate to the optimal error rate is) of the baseline method and our proposed method.

exhaustive search). The smaller the difference, the better the method is. This is illustrated in Figure 4.6.

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 supporting web page [24].

4.5.3 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 data sets 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 [117], 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 1000x speed-up than the state-of-the-art [97]. Such algorithms can augment our method.

4.5.4 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 [120]. Nevertheless, it is interesting to ask if our algorithm can improve upon the original UCR Archives 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 Figure 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. [33] can reduce the error rate of *GunPoint* to 4.7%, but our method suggests $w = 5$, which yields an error rate of only

3.3%. Similarly, Gorecki and Luczak [47] 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.

4.6 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 data set/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 [112], averaging exemplars [95], and discarding data [128]. 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 seconds, with one such individual succumbing to their injuries every 19 minutes. The total cost of fall injuries mounted to $34 billion in 2013 in the US alone [92]. The type of fall is highly predictive of the extent of the injuries that the victim sustains [85]. 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 data set we consider was kindly shared by Albert and colleagues [2]. 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 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 data set is 400 data points long, representing a 20 second fall event at the sampling rate of 20Hz (we resample the subsequences of uniform length if some are slighter shorter or longer). Figure 4.7 displays four examples of a trip fall. The data can be considered weakly labeled. The fall does not span the entire 20-second 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 data set needs warping in-variance, and our algorithm helps determine the appropriate amount of warping to allow, as shown in Figure 4.7.

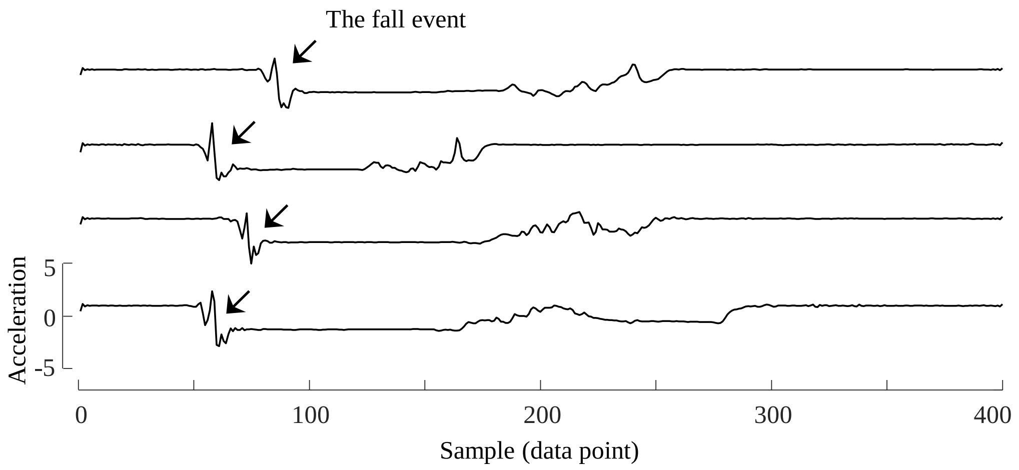

Figure 4.7: Four instances of a trip and fall event captured in the x-axis acceleration

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 come from five different people, who possess unique physiques and gaits, we only have two samples of each individual to learn from. We show the result in Figure 4.8.

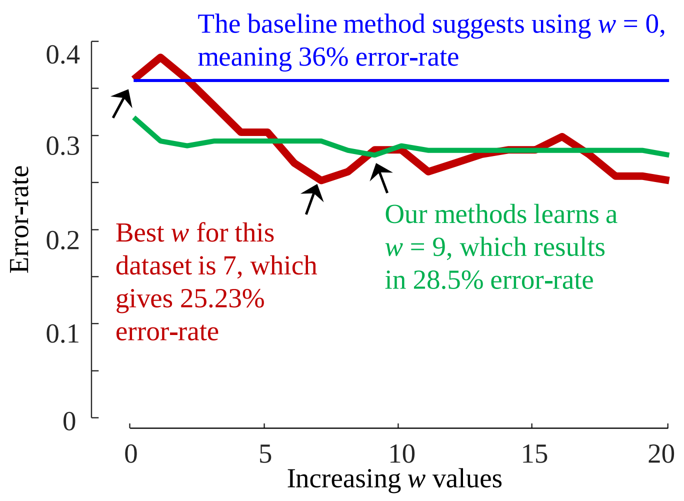

Figure 4.8: Error rate of fall classification. The indices of lowest values indicate the best w. Our method to learn w obtains an 7.5% error rate reduction compared to the baseline method.

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 data set is $w = 7$, which corresponds to a 25.23% error rate. This result is less impressive than the one published by Albert et al. [2], but note that we intentionally frame our problem around limited cross-subject training data and we perform classification using only a single dimension.

4.7 Conclusions

We have demonstrated that the choice of warping window width w is critical for an accurate DTW-based nearest neighbor classification of time series, and proposed a resampling method to learn w in this context. In many cases, a more careful setting of the value of w can close the performance gap gained by other more complicated algorithms recently proposed in the literature. Our method is parameter-free (or equivalently, we hard-coded all parameters). However, experimenting with adaptive parameters may allow others to improve upon our results. In the spirit of reproducible research, we have released all our code and data in a public repository [24], to allow others to confirm and develop upon our ideas.

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"* [109], or that research should *"focus on unconstrained DTW"* [3]. While the absence of evidence is not evidence of absence, the extensive nature of our experiments, which failed to find a single data set 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.

Chapter 5

A Generic Technique to Incorporate Domain Knowledge into Time Series Motif Discovery

5.1 Introduction

The last decade has seen time series motif discovery become a prominent primitive for time series data mining. It has been applied to diverse domains such as climatology, robotics [87], medicine [130] and seismology [136]. Recently there has been significant progress on the scalability of motif discovery, and data sets with lengths in the tens of millions can be routinely searched on conventional hardware [136]. Paradoxically, the ease with which we can now perform motif discovery has revealed some weaknesses in the traditional definition of motifs; there are many situations in which the results of motif search do not align with the user's intent. Below we discuss several examples to help develop the readers intuitions for this fact.

5.1.1 Stop-Word Motif Bias

In some data sets, the most interesting repeated patterns may be rare, and "swamped" by more frequent patterns. By analogy with text, we can ask what is the most common "motif" (i.e. word) in Poe's famous poem, The Raven? One might imagine that it is the eponymous bird; however, that is only the 13th most common word, with 13 occurrences. The most frequent words with their number of occurrences are "the" (56), "and" (38), "I" (32) ... In retrospect, this finding is not surprising, and we recall that stop-words are removed before any text analytic is performed.

A similar problem occurs in time series. Consider the snippet of ECG data shown in Figure 5.1. Note that the signal begins with a slightly noisy saw-toothed wave (called the calibration signal), which is how the ECG apparatus indicates that it is switched-on, but not detecting a biological signal. Unsurprisingly, the saw-tooth "stop-word" is the best motif in this data set.

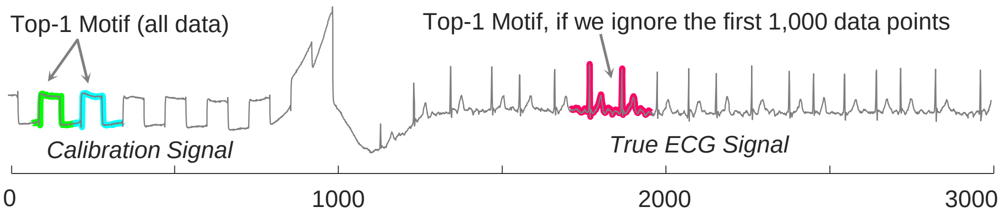

Figure 5.1: A snippet of ECG data from the LTAF-71 Database [96]. The top motifs come from regions of the calibration signal because they are much more similar than the motifs discovered if we search only data that contains true ECGs.

One might imagine that this issue only affects the beginning of signals, and thus could be easily addressed by manual inspection and truncation. Unfortunately, this is not the case. It is very common for sensors to temporarily lose the signal due to poor contact or patient motion, and this issue shows up dozens of times in this 25-hour trace.

It is important to note that the issue of stop-word motifs is not limited to machine artifacts from the recording process. The issue shows up even within "pure" data sources. Again, by analogy, note that a set of text stop-words is context dependent. For example, in a general text search the word "computer" is not a stop-word, but for a search within the ACM portal, it is a stop-word. The exact same situation occurs with time series. Consider the snippet of ECG data shown in Figure 5.2. The eye is drawn to the repeated ventricular contractions highlighted in red, but the best motifs by the classic definitions are a pair of normal heartbeats.

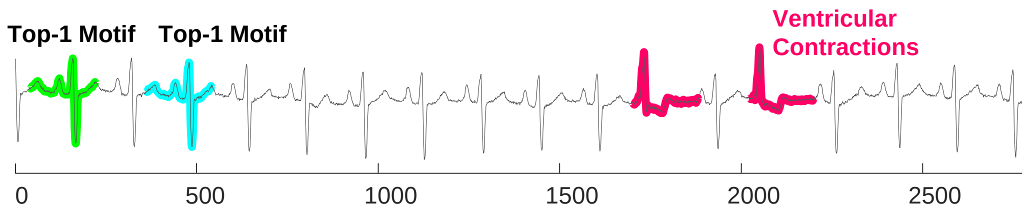

Figure 5.2: A snippet of ECG data from BIDMC Congestive Heart Failure Database (chf02). The top motif pair are the two highlighted normal heartbeats. However, a cardiologist's eye is drawn here to the two repeated ventricular contractions [82].

Even if the rare motifs are more strongly conserved than the background motifs, we should still expect the background motifs to be discovered first. The reason is essentially a real-valued version of the birthday paradox. Consider again the pair of ventricular contractions shown in Figure 5.2. They are better conserved (under the Euclidean distance) than most pairs of normal beats. However, there are fourteen normal beats, therefore there are 91 pairwise combinations that could match. With so many possibilities, it is unsurprising that the closest pair will be reported as the best motif.

The toy problem in Figure 5.2 can be solved by visual inspection, but with time series motif discovery now scaling to data sets of length one-hundred million [136], a more general and automatic solution is required. Finally, we note that this problem is not confined to ECGs or other periodic data. For example, in industrial data the top-K motifs may all correspond to patterns caused by calibration runs or shift changes. These known/uninteresting patterns may obscure a much rarer and unexpected repeated pattern.

5.1.2 Simplicity Bias in Motif Search

Imagine that we have {C1,C2}, a pair of subsequences that are "complicated" (informally for now, this means having many peaks and valleys [10]), and {S1,S2}, a pair subsequences that are "simple". Further imagine that, to your eyes, each pair itself is equally similar. In spite of this imagined subjective equality of similarity, it is almost certain that the dissimilarity, D(C1,C2) is greater than D(S1,S2) under Euclidean distance, Dynamic Time Warping or any other effective distance measure [10]. The practical upshot of this is that it will be difficult to find such complicated motifs, as the top-K will be swamped by many simple motifs. To see this, consider the top motif discovered in the EEG snippet shown in Figure 5.3.

The result is visually jarring. The two motion artifacts are visually similar and do have a small Euclidean distance, yet they are not discovered as the top motif. Instead, two

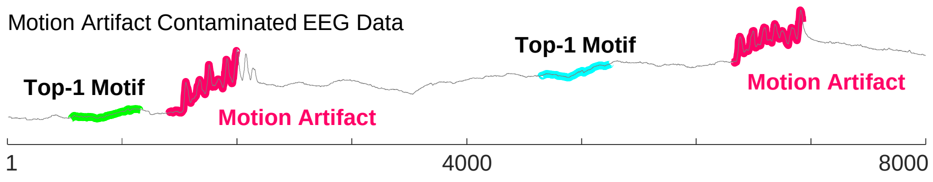

Figure 5.3: A snippet of an EEG time series in which two motion artifacts were deliberately introduced by the attending physician [88]. Surprisingly, the top-motif does not correspond to the motion artifacts, but to simple regions of "drift".

regions of vaguely rising trends are the top motif. The reason for this mismatch between our expectation and the results is understood, but not in this context. In [10], it was demonstrated that the Euclidean distance has a bias toward simple shapes. Quoting [10], *"pairs of complex objects, even those which subjectively may seem very similar to the human eye, tend to be further apart under (Euclidean) distance measures than pairs of simple objects."*

This issue is perhaps the most reported complaint of users of motif discovery tools. Almost all large time series data sets have low complexity regions, which are often uninteresting to the analyst, yet it is virtually certain that all the top-K motifs will come from there.

5.1.3 Actionability Bias

In many cases a domain expert wants to find not simply the best motif (as defined as the subsequence pair that minimizes their mutual distance), but regularities in the data which are exploitable or actionable in some domain specific ways. Such constraints have been explained to us by various domain experts1 with statements such as *"I want to find motifs in this web-click data, preferably occurring on or close to the weekend"*, or *"I want* *to find motifs in this oil pressure data, but they would be more useful if they end with a rising trend"*, or *"I want to find motifs in this PPG time series data, but it would be better if they happened about five to ten minutes after the accompanying RR time series data was relatively high"*.

There is currently no way to support such arbitrary constraints/preferences in motif search. Note that by their very nature, many of these constraints will need to be adjusted in real-time interactive sessions. For example, the user with a preference for a motif that happens near a weekend may be inspired by the quality of the results and wish to sharpen her focus to "strictly on a weekend", or she may have been disappointed by the sparse results and reluctantly weaken her constraint to "preferably not on a Tuesday or Wednesday". As we will show, we can support this need interactively with almost no overheard of time or space.

5.1.4 Summary of the Introductory Materials

We have seen several reasons why classic motif discovery can fail to produce the expected or desired results. Among the three examples that we discussed, stop-word bias and simplicity bias are intrinsic characteristics of classic motif discovery. That is, by definition, we *should* anticipate these to occur. On the other hand, the actionability bias is the users' preference for a result that fulfill their needs. These preferences are from the domain knowledge, and can take on many forms. They can be divided into smaller categories on their own. Current motif discovery tools do not support such preference embedding. Moreover, many practitioner of classic motif discovery are not even aware of its inherent biases toward patterns of simple shapes and super-frequent patterns.

The contribution of this work is to produce a single, real-time, intuitive framework that can handle all the above, and many other domain specific constraints that have yet to occur to us. The basic idea behind our approach is to produce a vector that is “parallel” to the original time series and annotates it with the users constraint(s). As the motif discovery algorithm finds candidate motifs, this annotation vector (AV) is used to re-rank them, allowing the motifs that best balance the fidelity of conservation with the users constraints to rise to the top.

This only leaves the question of how do we create such annotation vectors? We introduce a generic framework that allows a user to create such vectors, typically with just a few lines of code in a scripting language. For concreteness, we explicitly show the annotation functions that allow us to address all the examples above, and illustrate their utility with detailed case studies. Furthermore, our framework is simple and flexible enough to support domains and constraints that have yet to be explored. Despite the flexibility and expressiveness of our system, we will show that it requires an inconsequential overhead of time and space complexity, relative to the state-of-the-art motif discovery [130, 136].

5.2 Related Work and Notation

We begin with a brief review of related work before introducing the notations needed to understand our proposed framework.

5.2.1 Related Work

The literature on general time series motif search is large and growing. We refer the reader to [11, 53, 130] and references therein. While there are many papers on exploiting motifs, and scaling up motif search, to the best of our knowledge there are no research efforts that even explicitly note the issues we tackle, much less address them.

Saria and colleagues noted a limitation of the standard motif definition in that it may not readily discover motifs that have significant temporal warping [106]. Similarly, Yankov et al. noted that in some domains we may need to allow uniform scaling to enable meaningful motif discovery [129]. However, both of these issues are largely orthogonal to the issues at hand. In any case, Mueen et al. have argued that these issues are mostly an artifact of searching small data sets, and as we search larger data sets, these issues simply go away (see Section 4.3.2 of [88]).

Fox et al. proposed a method to mine *contextual motifs*, which are motifs taken into consideration the contexts in which they occur. They argue that this extension to the standard definition of motif increases the discriminating power of the discovered motifs [40]. While we applaud the merit of this work, we think that contextual motif can be subsumed under our work. Our framework for guided motif search is more generic, and can discover contextual motifs and more. In any cases, our approach is faster and simpler.

We do note that in the context of text information retrieval, there is significant work in query-biased (or user-directed) summaries [118]. Such work is in the same spirit as our work, directing the results of a query toward an arbitrary informational need or away from an undesired outcome [82].

5.2.2 Notation

We begin by defining the data type of interest, time series:

Definition 1: A time series $T \in \mathbb{R}^n$ is a sequence of real-valued numbers $t_i \in \mathbb{R}$: $T = [t_1, t_2, , t_n]$ where n is the length of T.

We are typically not interested in the global properties of a time series, but in the local regions known as subsequences:

Definition 2: A subsequence $T_{i,m} \in \mathbb{R}^m$ of a time series T is a continuous subset of the values from T of length m starting from position i. Formally, $T_{i,m} = [t_i, t_{i+1}, ..., t_{i+m-1}]$.

The local property we are interested in in this work is time series motifs:

Definition 3: A time series motif is the most similar subsequence pair of a time series. Formally, $T_{a,m}$ and $T_{b,m}$ is the motif pair $\iff$ $dist(T_{a,m}, T_{b,m}) \leq dist(T_{i,m}, T_{j,m})$ $\forall$ $i, j \in [1, 2, ..., n - m + 1]$ where $a \neq b$ and $i \neq j$, and $dist$ is a function that computes the z-normalized Euclidean distance between two subsequences [11, 21, 88, 129, 130].

We store the distance between a subsequence of a time series with all the other subsequences from the same time series in an ordered array called distance profile.

Definition 4: A distance profile $D \in \mathbb{R}^{n-m+1}$ of a time series T and a subsequence $T_{i,m}$ is a vector stores $dist(T_{i,m}, T_{j,m})$ $\forall$ $j \in [1, 2, ..., n - m + 1]$, where $i \neq j$.

One of the most efficient ways to locate exact time series motifs is to compute the matrix profile [130].

Definition 5: A matrix profile $P \in \mathbb{R}^{n-m+1}$ of a time series T is a meta time series that stores the z-normalized Euclidean distance between each subsequence and its nearest neighbor where n is the length of T and m is the subsequence length. The time series motif can be found by simply locating the two lowest values in P (they will have tying values).

Figure 5.4 illustrates a matrix profile on a small toy data set.

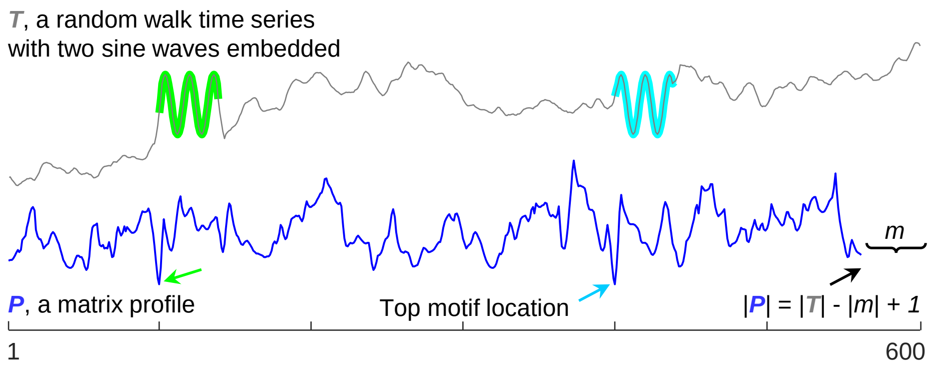

Figure 5.4: A time series T and its self-join matrix profile P. The i^{th} element of P is the z-normalized Euclidean distance between the subsequence starting from position i of T and its nearest neighbor.

To avoid trivial matches [88] in which a pattern is matched to itself, or a pattern that largely overlaps with itself, the matrix profile incorporates an "exclusion-zone" concept, which is a region before and after the location of a given query that should be ignored. The exclusion zone is heuristically set to $m/2$.

The time complexity to compute a matrix profile P is $O(n^2)$. This may seem untenable for time series data mining, but several factors mitigate this concern. First, note that the time complexity is independent of m, the length of the subsequences. Secondly, the matrix profile can be computed with an anytime algorithm, and in most domains, in just $O(c * nlog_n)$ steps the algorithm converges to what would be the final solution [130] (c is a small constant). Finally, the matrix profile can be computed with GPUs, cloud computing and other HPC environments that make scaling to at least tens of millions of data points trivial [136]. Even using standard hardware, all the examples in this paper can be computed much faster than real-time. For example, 30 minutes of ECG sampled at 60Hz takes about four minutes to exactly compute the full matrix profile using STOMP [136]. If that was not fast enough, STAMP can produce a high-quality approximation in under five seconds [130].

Before moving on, we wish to restate the reason for our detailed explanation of the matrix profile. As explained in greater detail in [130], once the matrix profile has been computed, all the top-K motifs are available "for free". For example, the locations of the top-1 motif is simply the (tied) lowest points in the matrix profile. Moreover, all other definitions of motifs (range motifs, top-K motifs etc.) can also be trivially extracted. Thus, we see the computational aspects of motif search largely solved, our contributions are limited to "nudging" the results to be more useful.

5.3 Guided Motif Search

The basic idea behind guided motif search is to produce a vector that is "parallel" to the original time series (and to the matrix profile) that encodes the users domain-dependent bias(es). This vector is then used to modify the matrix profile, changing its shape such that the undesirable solutions are more expensive, and no longer show up in the top-K motifs. We begin by explaining the general domain independent part of our solution in the next section, before showing several examples of how to create domain dependent bias functions.

5.3.1 The Annotation Vector Framework

The Annotation Vector (AV) plays the role of manipulating the motif search. Recall the notation of Matrix Profile (MP) elucidated in Section 5.2.2. The MP is the current state-of-the-art for motif discovery [130, 136]. Our main idea is to leverage this MP to discover more meaningful motifs. We achieve this goal by combining the matrix profile with the annotation vector to produce a new matrix profile. We will refer to this as the "Corrected" MP (CMP), as it correctly incorporates the contextual bias for the problem at hand.

The annotation vector AV is a time series consisting of real-valued numbers between [0 - 1]. A low value indicates the subsequence starting at that index is not a desirable motif, and therefore should be biased against. Conversely, higher values mean the subsequence at that location should be favored for the potential motif pool. Note that the AV has the same length as the matrix profile MP. In Figure 5.5 we show the annotation vector that encodes the bias: *"I want to find motifs in this traffic data, preferably occurring on or close to the weekend."*

There may be some scenarios in which the annotation vector AV consists of just $0s$ and $1s$. For example, this would be the case when the desired motifs are strictly limited to within a certain period, such as, activities between 8am-10am daily, or at a longer time scale, activities that happen only during an Olympic year. For such scenarios, we would assign 0 for AV data points in the un-targeted period, and 1 for targeted ones. However, as shown in Figure 5.5, we allow arbitrary real-number values in the AV, so long as they are in the unit interval.

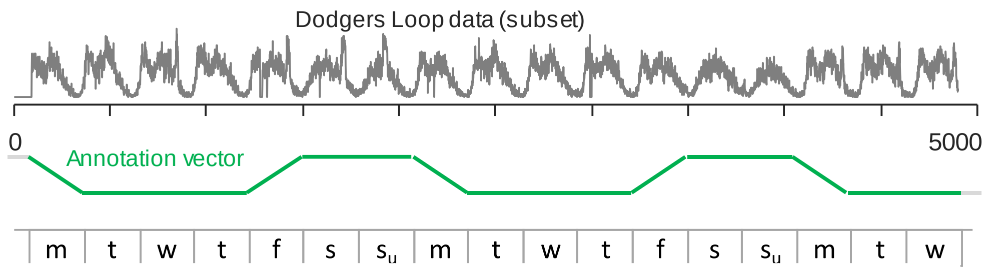

Figure 5.5: *top*) Seventeen days from the *Dodgers Loop* data set. *bottom*) The annotation vector that encodes a preference for motifs occurring on or near the weekend.

Having defined the AV, we can now explain how we use it to correct the undesirable results explained above. To do this, we simply produce a Corrected Matrix Profile (CMP) by combining the annotation vector AV and the original matrix profile MP.

$$CMP_i = MP_i + (1 - AV) * max(MP)$$

In the above equation, CMP_i, MP_i and AV_i denote the corrected MP value, the original MP value and the AV value respectively. The $max(MP)$ term denotes the maximum value of the original MP. The resulting corrected MP can be interpreted as follow: if a region of the time series potentially contains the meaningful motifs, its MP values are left untouched. Otherwise, its MP values are "pushed" higher (increased) to reduce the possibility of any motif in that region appears in the top-K motif list.

In many cases, the annotation vectors can be created by just a few lines of code. Moreover, they can be created in a simple environment such as Microsoft Excel or a scripting language such as MATLAB or Python. This is important, since many of the end-users of motif discovery are biologists, medical doctors, seismologists [130] etc., not computer/data scientists. For example, to produce one week of the AV shown in Figure 5.5, if the data are sampled once a minute and starts at midnight Sunday, we can use a single line of MATLAB:

AV = [(1440:-1:1)/1440, zeros(1,1440*3), (1:1440)/1440, ones(1,1440*2)]

Which we can interpret as:

AV = $[\text{Mon}_{\text{RampDown}}, \text{TueWedThu}_{\text{ConstantLow}}, \text{Fri}_{\text{RampUp}}, \text{Weekend}_{\text{ConstantHigh}}]$

In the following subsections, we will show how the annotation vector is used in action to guide motif search with several motivating case studies.

5.3.2 Case Study: Actionability Bias

In this section, we discuss two problems brought to our attention by medical researchers, and we show that they have trivial fixes using our framework.

Suppressing Motion Artifact

We begin with an example brought up by Dr. Gregory Mason of the UCLA Medical Center. Clinicians often want to find motifs in medical telemetry, for example, in data obtained by brain-imaging technique such as fNIRS shown in Figure 5.6.

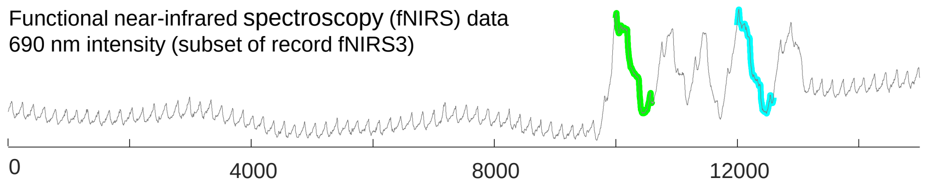

Figure 5.6: A snippet of fNIRS searched for motifs of length 600. The motifs correspond to an atypical region, which (using external data) we know is due to a sensor artifact.

This example highlights a problem ubiquitous in this domain [15, 115]. The motif discovered in Figure 5.6 corresponds to a motion artifact, and has no medical significance. Such motion artifacts plague both clinicians and researchers. For example, an eight-hour sleep study is likely to have dozens or hundreds of such artifacts, as the subject tosses and turns. Note that the example shown in Figure 5.6 is visually obvious for the readers benefit; however, more generally, it is not always easy to differentiate between biologically significant patterns and artifacts.

Nevertheless, we can address this issue for motif discovery in a very simple way, by leveraging the fact that many medical sensors also include an accelerometer. Figure 5.7 shows the synchronization between the fNIRS data and accompanying accelerator data. Given that the motion of the sensors is producing spurious motifs, we can use this motion to produce an annotation vector to suppress them.

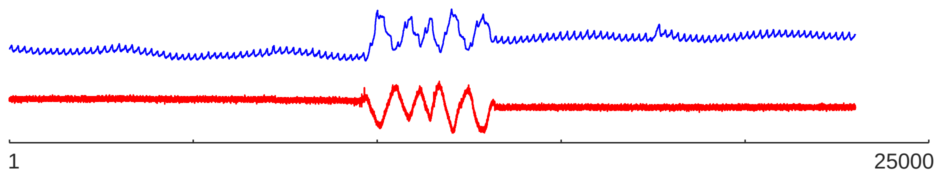

Figure 5.7: The blue time series is fNIRS data. The red time series is the acceleration of the body-worn sensors. This is a dramatic and visually obvious example of this issue. The problem is typically subtler, but still negatively effects motif discovery.

Concretely, we slide a window of length m across the acceleration time series. We compare the standard deviation of each subsequence with the mean of all the subsequences' standard deviations, and assign the AV value to be either 0 or 1 accordingly (see Figure 5.8). Our AV data point is 0 if the corresponding subsequence has its standard deviation equal or greater than the mean, indicating regions of unusually large fluctuation.

To show how little effort this would be for a clinician or researcher, Table 5.1 contains the full MATLAB code used to generate this AV. Naturally, a MATLAB guru could

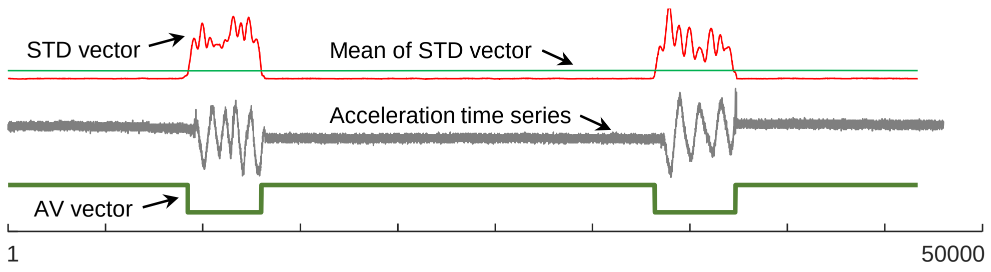

Figure 5.8: Points above the mean of all subsequences' standard deviation are well aligned with regions of motion artifacts.

write even terser code, but our point is simply that writing an AV is typically only a minutes work. For visualization of the results, Figure 5.9.*top* displays the original matrix profile along with the spurious motifs that are discovered with that MP. Figue 5.9.*bottom* shows the MP corrected with the annotation vector produced by Table 5.1, enabling medically meaningful motifs from neuronal activity-related signal regions to be discovered.

Table 5.1: Code to generate the AV for the fNIRS example

```
function AV = make_AV(data, subsequenceLength)
    for i = 1:(length(data) - subsequenceLength + 1)
        stdVector(i) = std(data(i:(i + subsequenceLength - 1)));
    end
    AV(stdVector >= mean(stdVector)) = 0;
    AV(stdVector <  mean(stdVector)) = 1;
end
```

Note that here we created a Boolean AV, which strictly prohibits finding motifs during sensor movement. However, we could also have created a real-valued AV, which

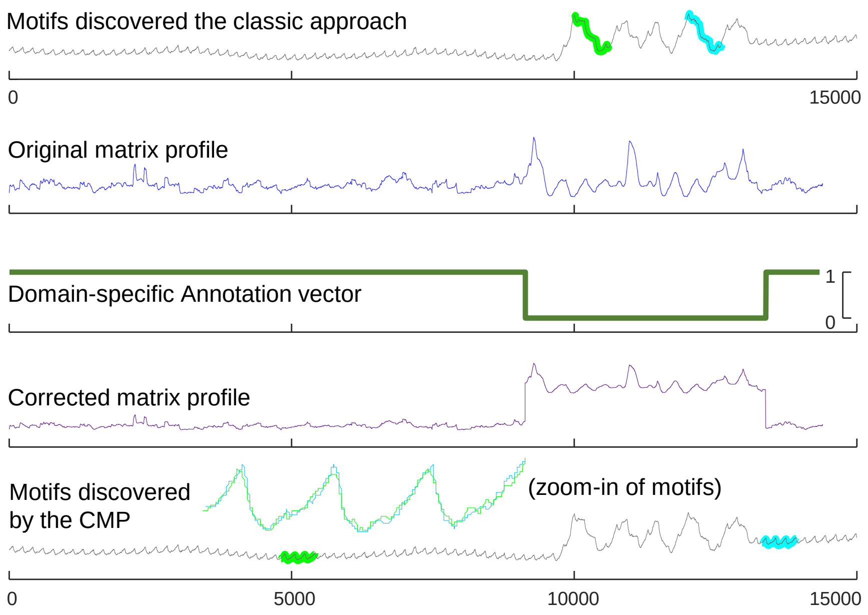

Figure 5.9: (top to bottom) Motifs in fNIRS data discovered using classic motif search tend to be spurious motion artifacts, because the matrix profile is minimized by the highly conserved but specious patterns. If we create an AV using the algorithm in Table 5.1, and use it to correct the MP, then that CMP allows us to find medically significant motifs.

simply biases the motif search away from regions where movement was noted, in proportion to the magnitude of the motion. As it happens, in this case, the Boolean AV was sufficient to solve the problem. In the next section, we will show an example of an issue that does require a real-valued AV.

Suppressing Hard-Limited Artifacts

Here we consider another example of a commonly encountered issue that may prevent us from finding meaningful motifs in medical and industrial data sets. In Figure 5.10.bottom we see that the motif discovered in this Electrooculogram (EOG) has a perfectly flat plateau. This is not reflective of medical reality, but is simply a region where the signal becomes saturated and the physical process fails to make a meaningful measurement [126].

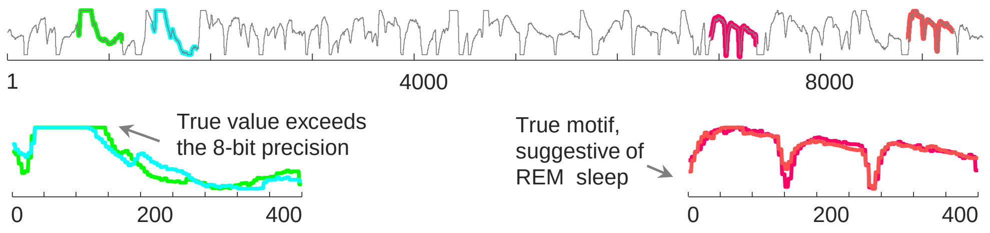

Figure 5.10: *top*) A snippet of a left-eye EOG sampled at 50 Hz, from an individual with sleep disorder. *bottom*) The top motif may be spurious, as it features a region where the time series indicates only the maximum value possible. However, this snippet does have a true motif, with medical significance.

In the many data sets that have this limited precision flaw, we can be guaranteed that most or all the top motifs will feature some of these constant regions, as the flat regions produce little cost in the Euclidean distance calculation that defines motifs. However, as shown in Figure 5.10.*bottom.right*, the data may be replete with more medically meaningful motifs. It is important to note that we do not wish to completely exclude the possibility of returning motifs with some amount of constant regions. It is just that we want to mitigate the strong bias to finding them exclusively.

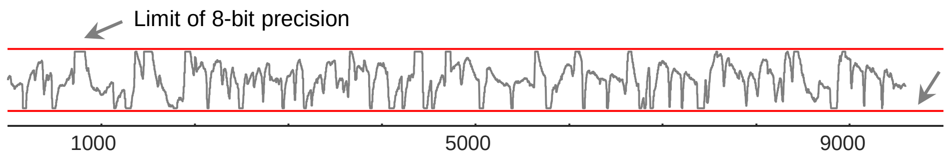

Figure 5.11: The upper and lower bound of the EOG data indicates the regions where the signal becomes saturated.

As the reader will now appreciate after having seen our previous examples, we can easily build an AV to suppress these spurious matches. We begin by recording the maximum and the minimum values of the time series, the constant values touching the red bars shown in Figure 5.11. We slide a window across the time series to extract subsequences, counting the number of constant values (from being hard-limited above or below) in each sequence. This number over the subsequence length is used as the bias function. A higher value hints that the motifs that include this subsequence may be spurious, as the overflow/underflow regions act like a "don't-care" in the motif distance. Figure 5.12 illustrates the results of applying this AV to the problematic example shown in Figure 5.10. Here again the annotation vector helps to bias away from spurious motifs, by leveraging the original matrix profile with domain specific insights.

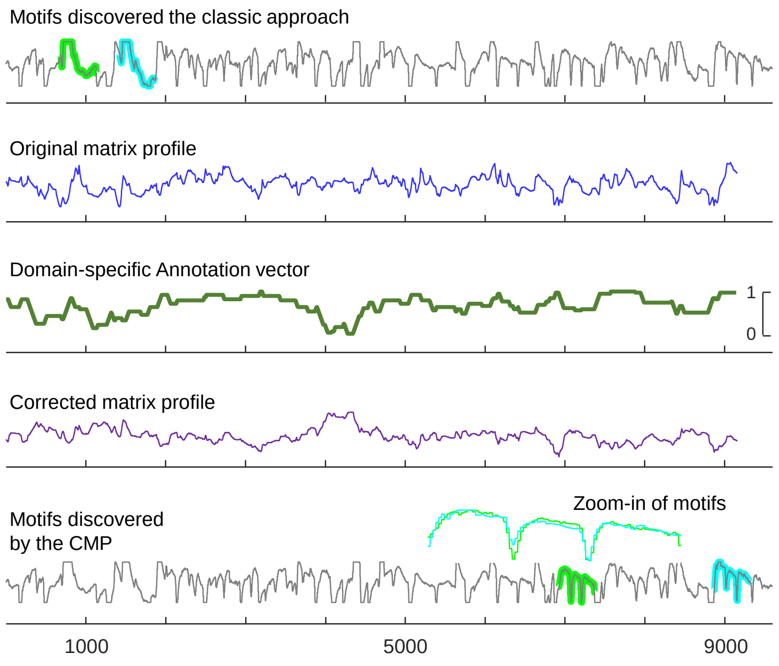

Figure 5.12: (top to bottom) Motifs in the EOG domain that are discovered using classic motif search tend to include hard-limited data, because the matrix profile is minimized by having long constant regions. By creating an AV using the algorithm in Table 5.2 to correct the MP, we can find true motifs corresponding to ponto-geniculo-occipital waves [22].

In Table 5.2, we show the full MATLAB code used to make the annotation vector discussed above. As before, this simple fix requires only a minute of coding effort.

Table 5.2: Code to generate the AV for the EOG example

```
function AV = make_AV(data, subsequenceLength)
    for i = 1: (length(data) - subsequenceLength + 1)
      s = data(i:i + subsequenceLength - 1);
      AV(i) = length(s(s == max(data) | s == min(data)));
    end
    AV = AV - min(AV); % zero one normalization
    AV = AV / max(AV); % zero one normalization
    AV = 1 - AV;      % AV property conformation
end
```

5.3.3 Case Study: Stop-Word Motif Bias

We return to the ECG example presented in Figure 5.1. Recall that the top motifs of this time series do not come from a true medical signal, but regions of the electrodes calibration signal. These signals are much more similar than the "genuine" medically significant motifs, and swamp the top-K motif set.

Leveraging similar ideas in text processing, we propose to treat these spurious patterns as a "stop-word" motifs. Our goal is to design an AV that can discount such motifs. We can make no assumptions about the locations of the stop-word motif, given that they can show up anywhere in the signal.

We assume only that the stop-word motif(s) will be known to the users of our framework, as the stop-words are domain specific. In Figure 5.13.*top*, we show an example of a stop-word motif that we are targeting. Figure 5.13.*bottom* displays the annotation vector that can bias the motif discovery away from that stop-word.

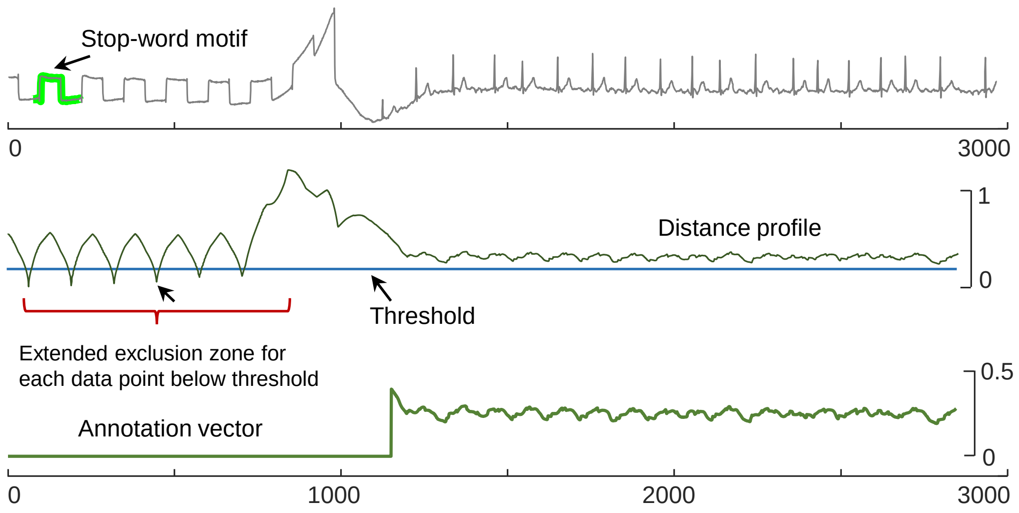

Figure 5.13: *top*) We annotated a single stop-word from the LTAF-71 Database [96]. *middle*) The stop-words distance profile to the entire data set was thresholded to create an exclusion zone, which was used to create a AV (*bottom*).

To create the AV here, we measure the similarity between the stop-word subsequence to all subsequences in the time series T, by sliding window of the stop-words length across T. This gives a distance profile (see Section 5.2.2 Definition 4). We min-max normalize this distance profile to be in range [0 - 1], as shown in Figure 5.13.middle. To convert the normalized distance profile into our final AV vector, we employ a threshold parameter. The threshold chosen indicates how similar the corresponding subsequences are in comparison to the stop-word motif. If we set the threshold to be 0.1, we retrieve the location of subsequences that are 90% stop-word-like.

Having a threshold is generally undesirable, but recall that our framework has completely divorced the computationally trivial AV adjustment from the expensive computation of the matrix profile. Thus, if the threshold is too aggressive or too lax, the user can update it and refresh the results in under a second, even for data sets as large as one million data points.

For each location of stop-word like subsequence, we impose an exclusion zone to avoid retrieving motifs that overlap with the stop-word regions. We set the exclusion zone to be equal to the subsequence length m, and make all the values within each exclusion zone equal to the value at the location of its originating stop-word like subsequence. This effectively means that we are not interested subsequences that have any overlap with the stop-word like pattern, as much as we would like to shun that stop-word like pattern. After this step, we obtain the final AV vector.

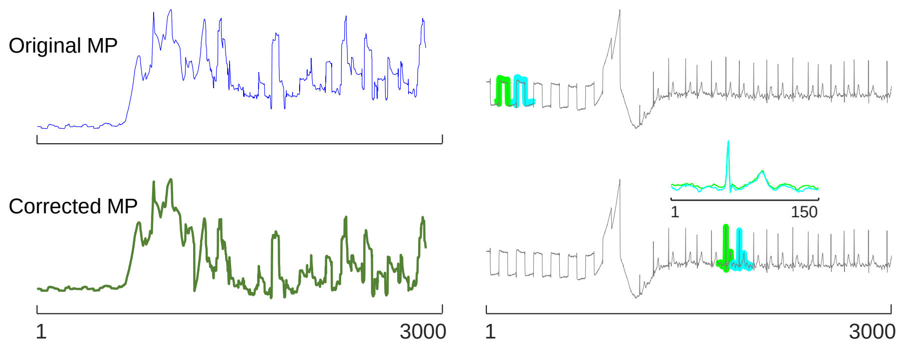

Figure 5.14: By correcting the MP to bias away from stop-word motifs, we can discover medically meaningful motifs.

Figure 5.14.*top* shows the original MP (left) and the top-1 motif that would have been found using that MP (on the right, highlighted in green and cyan). Figure 5.14.*bottom* shows the corrected MP and the corresponding top-1 motif.

5.3.4 Case Study: Simplicity Bias

The issue of simplicity bias has been reported to us by a dozen research groups in the last decade, although not under that name, which we coined for this paper. The problem can be very subtle, but in Figure 5.15 below we show a strikingly obvious example.

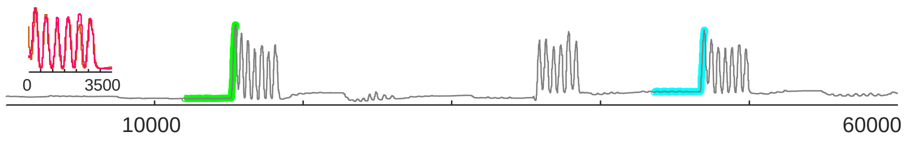

Figure 5.15: A short snippet of a time series of the flexion of a subjects little finger [70]. Subjectively, most people would expect that two occurrences of consecutive multiple flexions to be the top motif (inset). Instead we find the simple "ramp-up" pattern to be the top motif.

The reason why motif discovery fails to match our expectation here was discussed in Section 5.1.2, and at length in [10] (in a very different context). Given our guided motif framework, the problem is trivial to solve. We just need to create an AV that penalizes for simplicity.

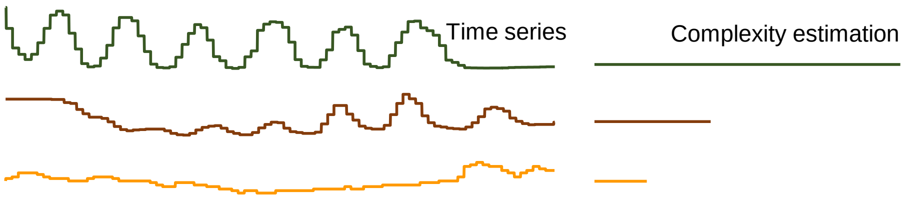

Figure 5.16: A visual intuition of the complexity estimation of three time series subsequences of different complexity levels according to the algorithm proposed in [10].

To do this, we employ the complexity estimation (CE) proposed in [10]. The authors of this work originally embedded CE in a complexity correction factor for the Euclidean distance, making this distance measure complexity-invariant. This complexity estimation is simple (the L2 norm of the differences between consecutive data points), parameter-free and has a natural interpretation. Intuitively, time series can be imagined as “chains” or “ropes”, and have their complexity measured by “stretching” them and measuring the length of the resulting taut lines, as illustrated in Figure 5.16. The more complex the time series is, the longer its corresponding line will be.

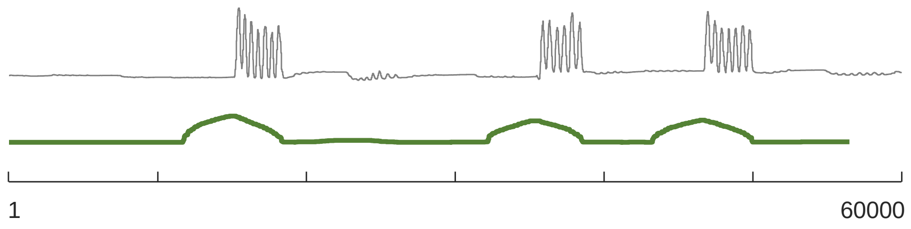

Figure 5.17: The complexity measure (bold/bottom) shown in parallel to the raw data (fine/top).

We slide a window across the time series, measuring the complexity of each subsequence and store them in a complexity vector, as shown by the green (bold) line of Figure 5.17. We simply normalize this complexity vector to be in range $[0-1]$ to obtain the final AV. Table 5.3 contains the MATLAB code used to generate the AV. Figure 5.18 illustrates how the real motifs are uncovered, as opposed to the “ramp-up” patterns that would have dominated if we rely on the classic matrix profile.

To show the ubiquity of this issue, and the generality of our solution, we searched seismology telemetry [136]. Figure 5.19 illustrates how guided motif search correctly discovers two earthquakes from the same fault, occurring thirteen years apart, in Sonoma Country, California. Classic motif search ranks 852 other motifs above this true motif; these higher-ranked motifs are all sensor artifacts [63] that our AV manages to suppress.

Table 5.3: Code to generate AV for the ECoG example

```
function AV = make_AV(data, subsequenceLength)
    for i = 1: (length(data) - subsequenceLength + 1)
        subsequence = data(i:i + subsequenceLength - 1);
        AV(i) = sqrt(sum(diff(subsequence).^2));
    end
    AV = AV - min(AV); % zero one normalization
    AV = AV / max(AV); % zero one normalization
end
```

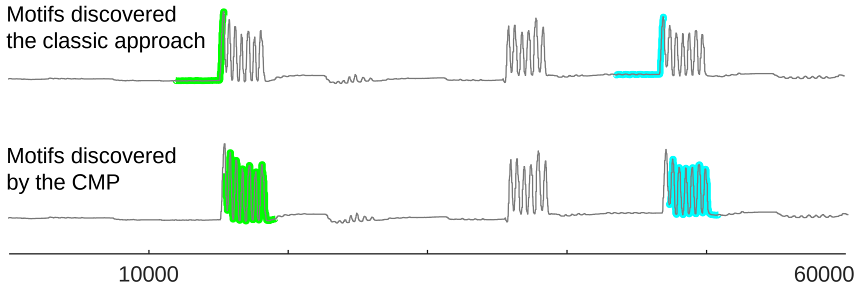

Figure 5.18: By correcting the matrix profile with an AV using the algorithm in Table 5.3, we discover the true motifs of finger flexion pattern in the ECoG signal.

5.4 Empirical Evaluation

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 [23]. This commitment to reproducibility extends to all the examples the previous sections. We note that we do not need to compare to rival algorithms, but a rival definition, the classic motif definition used in several hundred research efforts over the last decade [11, 88, 130, 136].

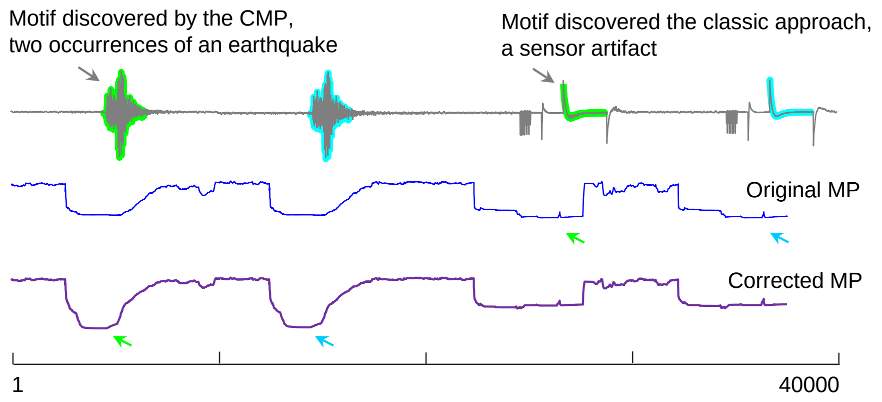

Figure 5.19: The top motif returned by the classic approach is not an earthquake, but a sensor artifact. Using guided motif search framework, we can avoid such various misleading sensor artifacts, which swamp this 17,279,800 data-point earthquake data set. The time series shown was edited for visual clarity.

5.4.1 Preliminary Tests

We begin with an experiment that is designed to closely model the issue shown in Figure 5.3, but at a scale that will allow statistically significant results. We produced 1,000 data sets as follows. We created a random walk of length 20,000, then embedded into it two randomly chosen instances from the eight-class MALLAT data set from the UCR Archive [20]. Figure 5.20 shows that these synthetic data sets are, at least visually, a good proxy for the real motion artifact contamination data.

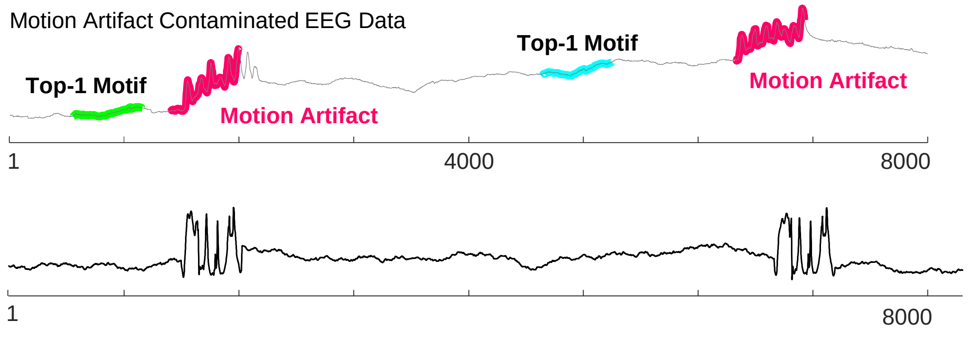

Figure 5.20: A snippet of real motion artifact contamination data (see also Figure 5.3) and a snippet of our synthetic proxy for it.

For each data set, we run motif discovery, and count as a success, any answer in which the top-1 returned motifs overlap with any part of the embedded motifs. We test the performance of three alternatives:

- Classic motif search, to reflect what is currently done in the literature (Classic) [130].
- Guided motif Search: In which the AV is created using complexity bias, as discussed in Section 5.3.4 ($\mathrm{AV}_{\mathrm{complexity}}$).
- Guided motif Search: In which the AV is created by counting the number of zeros-crossings ($\mathrm{AV}_{\mathrm{ZeroCrossings}}$) in the subsequences.

We consider two variants of the AV to test the following informal claim: Once a practitioner understands the issue producing poor motifs, there will almost certainly be many simple fixes possible. Table 5.4 summarizes the results.

Table 5.4: Comparison between classic motif search and guided motif search over 1000 data sets of length 20,000. Both two variants of guided motif search outperform classic motif search in terms of accuracy, with extra time overhead of just 0.09 second per run.

Approach	**Average success rate**	**Time per run**
Classic	16.3%	22.40 seconds
$\mathrm{AV}_{\mathrm{complexity}}$	99.9%	22.46 seconds
$\mathrm{AV}_{\mathrm{ZeroCrossings}}$	38.7%	22.49 seconds

A practitioner that exploited the number of zero-crossings would have seen her hit-rate more than double, yet noted there is still room for improvement. Perhaps she would have noticed that random walk data can sometimes have high zero-crossings by chance and would have turned to a more robust zero-crossings extractor [125]. On the other hand, a more sophisticated practitioner that was aware of complexity bias [10], would have seen essentially perfect results from the first time. In both cases, we see that the simple correction offered by an annotation vector can dramatically improve the utility of motif search. We note in passing these results that they support our claim that the time overhead for guided motif search is inconsequential.

5.4.2 Guided Motif Search for Event Detection

Sometimes motif discovery is the endpoint of data analysis, but more often motif search is a subroutine in a higher-level algorithm. In this section, we show that our framework can also be helpful in such scenarios. We are particularly interested in the problem of building dictionaries from weakly labeled training data. We show that in this case, classic motif search may perform poorly and guided motif search can offer significant improvements.

Figure 5.21 presents an example of a weakly labeled time series. While the time series snippet represents a 71 seconds long episode of mimicked epileptic seizures [121], the actual seizure is only about 30 seconds long, sandwiched in-between states where the actor was preparing-for/recovering-from her role. This presence of spurious data is what is meant by "weakly labeled". Using classic motif search to find the most conserved pattern in this time series returns top motifs in a non-epileptic region, as shown in Figure 5.21.top. However, using the guided motif search framework (CMP), employing a complexity bias, discovers the top-1 motif in the truly representative seizure region, as shown in Figure 5.21.bottom.

We design the experiment as follows. We first run motif search on the weakly labeled training data for subsequence of length 16 (one second) to find the top-1 motif pair. We average this motif pair to get the representative pattern for that class. Next, we slide the representative motif across the test (unseen) time series, measuring its distance to all the subsequences. If the difference between a test subsequence and the representative motif is at most three times the distance between the train top-1 motif pair, we mark that subsequence as epilepsy positive. Our test set consists of five recordings of mimicking epilepsy seizure and three recordings of walking task concatenated (Figure 5.22). We divide the test time series into segments of 36 two-second regions. If we find any hit within each two second segment, and the ground truth agrees with that, we denote this as a true positive.

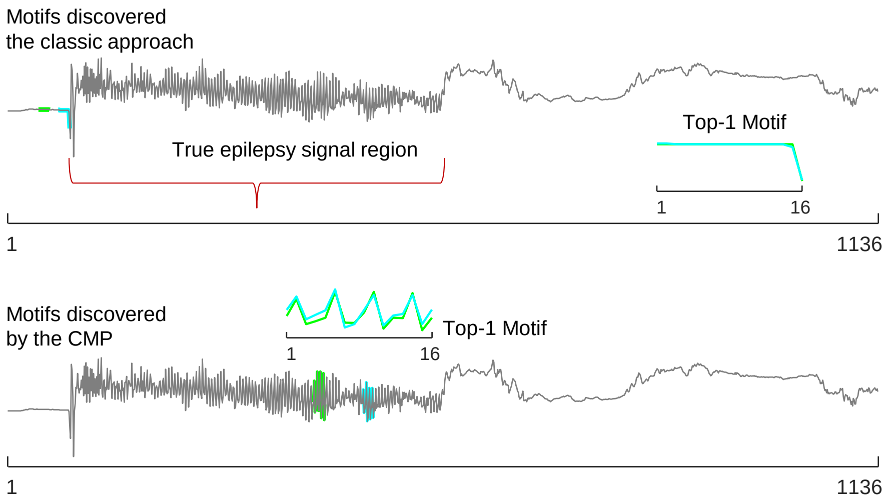

Figure 5.21: A weakly labeled seizure mimicking captured with accelerometer x-axis reading. Classic motif search wrongly finds the top motifs in non-representative event regions (top) whilst CMP correctly uncovers the true epileptic patterns (bottom).

We compare the result of the guided motif search approach using $AV_{complexity}$ with the classic motif search approach. Table 5.5 shows the full contingency matrix. The guided motif search outperforms classic motif search in both metrics, accuracy (95.7% vs. 82.78%)

and F-measure (82.78% vs 2.9%). Figure 5.22 shows a visualization of classification result.

Table 5.5: Contingency matrix. *left*) The classic motif search approach. *right*) The guided motif search approach. E denotes Epilepsy. NE denotes Not Epilepsy.

Classic

Prediction	True class: E	True class: NE
E	1	1
NE	66	321

CMP

Prediction	True class: E	True class: NE
E	52	3
NE	15	319

This issue of weakly labeled data could be avoided by having an expert manually annotating the data set, and we acknowledge that the experiment is somewhat contrived. However, we do not always have the luxury of human expert intervention, and in that case, guided motif search with a little insight might take us a long way.

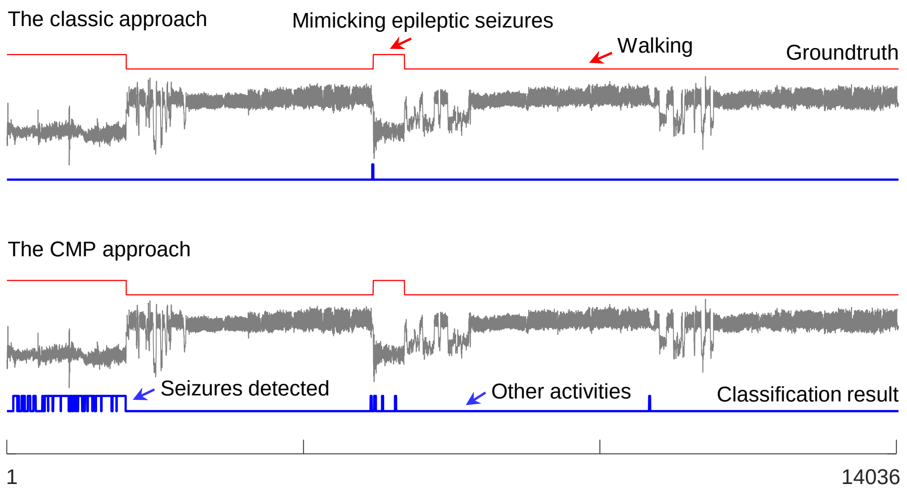

Figure 5.22: Classification result with the dictionary learned using classic motif search (top) and the dictionary built using the guided motif search approach (bottom).

5.4.3 Time and Space Complexity

The time complexity of our proposed guided motif search is $O(n^2)$, and the space complexity is merely $O(n)$, which is the same as classic motif search using state-of-the-art methods [130, 136]. We need $O(n^2)$ for computing the original MP, plus $O(n)$ additional work to compute the AV and produce the corrected MP. To concretely ground these numbers, consider Table 5.4. It took 22.5 seconds to process 20,000 data points. Given that the data was recorded at 50Hz, this is 400 seconds of wall-clock time, meaning we are about eighteen times faster than real-time.

5.5 Conclusions and Future Work

We have shown that direct use of classic time motif search can produce unexpected/undesired results in many circumstances, for a variety of domain dependent reasons. We have presented a novel framework for guided motif discovery, which greatly mitigate these issues. In the spirit of reproducible research, we have released all the code and data at [23], to allow others to confirm, extend and exploit our ideas.

We plan to produce a crowd-sourced website which catalogues the issues effecting motif discovery, together with the suggested AVs that can fix the issue. We envision that this resource will expand as domain experts provide examples, in the manner of: *"If you are looking for repeating stanzas in Byzantine or Turkish folk songs, we suggest you use this AVPitchCorrect, otherwise you may find ..."*. Finally, all the examples that we have given required the practitioner to understand the issue, and come up with an AV to solve it. In future work, we hope to learn the AV simply by observing the user interact with raw data.

Chapter 6

The UCR Time Series Classification Archive

6.1 Introduction

The discipline of time series data mining dates back to at least the early 1990s [1]. As noted in a survey [68], during the first decade of research, the vast majority of papers tested only on a single artificial data set created by the proposing authors themselves [1, 62, 69, 103]. While this is forgivable given the difficulty of obtaining data in the early days of the web, it made gauging progress and the comparisons of rival approaches essentially impossible. Frustrated by this difficulty [68], and inspired by the positive contributions of the more general UCI Archive to the machine learning community [75], Keogh & Folias introduced the UCR Archive in 2002 [67]. The last expansion took place in Summer 2015, bringing the number of the data sets in the archive to 85 data sets [20]. As of Fall 2018, the archive has about 850 citations, but perhaps twice that number of papers use some fractions of the data set unacknowledged[1].

While the archive is heavily used, it has invited criticisms, both in published papers [60] and in informal communications to the lead archivists (i.e. the current authors). Some of these criticisms are clearly warranted, and the 2018 expansion of the archive that accompanies this paper is designed to address some of the issues pointed out by the community. In addition, we feel that some of the criticisms are unwarranted, or at least explainable. We take advantage of this opportunity to, for the first time, explain some of the rationale and design choices made in producing the original archive.

6.2 Setting the Baseline Accuracy

From the first iteration, the UCR Archive has had a single predefined train/test split, and three baseline ("strawman") scores accompany it. The baseline accuracies are from the classification result of the 1-nearest neighbor classifier (1-NN). Each test exemplar is assigned the class label of its closest match in the training set. The notion of "closest match" is how similar the time series are under some distance measures. This is straightforward for Euclidean distance (ED), in which the data points of two time series are linearly mapped i^{th} value to i^{th} value.

However, in the case of the Dynamic Time Warping distance (DTW), the distance can be different for each setting of the warping window width, known as the warping

[1]Why would someone use the archive and not acknowledge it? *Carelessness* probably explains the majority of such omissions. In addition, for several years (approximately 2006 to 2011), access to the archive was conditional on informally pledging to test on *all* data sets to avoid cherry picking (see Section 6.4). Some authors who did then go on to test on only a limited subset, possibly choosing not to cite the archive to avoid bringing attention to their failure to live up to their implied pledge.

constraint parameter w [30]. Figure 6.1 illustrates this idea. For further details, we refer readers to a more thorough introduction of DTW in Chapter 2 - Section 2.2.

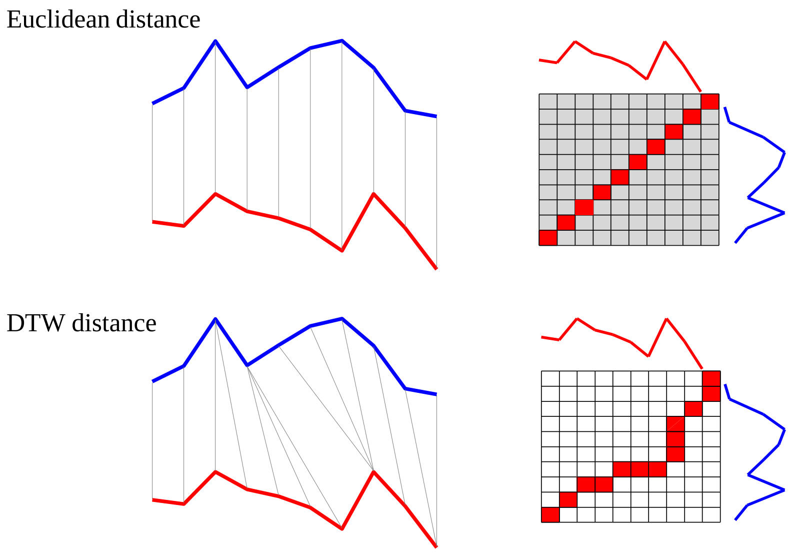

Figure 6.1: Visualization of the warping path. *top*) Euclidean distance with one-to-one point matching. The warping path is strictly diagonal (cannot visit the grayed-out cells). *bottom*) unconstrained DTW with one-to-many point matching. The warping path can monotonically advance through any cell of the distance matrix.

We refer to the practice of using 1-NN with Euclidean distance as 1-NN ED, and the practice of using 1-NN with DTW distance as 1-NN DTW. The UCR Time Series Archive reports three baseline classification results. These are classification error rate of:

- 1-NN Euclidean distance
- 1-NN unconstrained DTW
- 1-NN constrained DTW with learned warping window width

For the last case, we must *learn* a parameter from the training data. The best warping window width is decided by performing Leave-One-Out Cross-Validation (LOO

CV) with the train set, choosing the smallest value of w that minimizes the average train error rate. Generally, this approach works well in practice. However, it can produce poor results as in some situations, the best w in training may not be the best w for testing. (Recall Figure 2.2 in Chapter 2, in which the top row shows some examples where the learned constraint closely predicts the effect the warping window will have on the unseen data, while the bottom row, in contrast, shows some examples where the learned constraint fails to track the real test error rate, thus giving non-optimal classification result on holdout data.) Happily, the former case is much more common [30]. When does learning the parameter fail? Empirically, the problem only occurs for very small training sets; however, this issue is common in real world deployments.

6.3 Criticism of the UCR Archive

In this section we consider the criticisms that have been levelled at the UCR Archive. We enumerate and discuss them in no particular order.

6.3.1 Unrealistic Assumptions

Bing et al. have criticized the archive for the unrealistic assumptions follow [60].

- *There is a copious amount of perfectly aligned atomic patterns.* However, in at least in some domains, labeled training data can be expensive or difficult to obtain.

- *The patterns are all of equal length.* In practice, many patterns reflecting the same behavior can be manifest at different lengths. For example, a natural walking gait cycle can vary by at least plus or minus 10% in time.

- *Every item in the archive belongs to exactly one well-defined class; there is no option to choose an ``unknown" or ``unclassifiable".* For example, in the *Cricket* data sets, each signal belongs to one of twelve classes, representing the hand signs made by an umpire. However, for perhaps 99% of a game, the umpire is not making *any* signal. It can be argued that any practical system needs to have a thirteenth class named `` ``not-a-sign" ``. This is not trivial, as this class will be highly variable, and this would create a skewed data set.

6.3.2 The Provenance of the Data is Poor

Here we can only respond *mea culpa*. The archive was first created as a small-scale personal project for Keogh's lab at University of California, Riverside. We did not know at the time that it would expand so large and become an important resource for the community. In this release, we attempt to document the data sets in a more systematic manner. In fact, one of the criteria for including a new data set in the archive is that it has a detailed description from the data donor or it has been published in a research paper that we could cite.

6.3.3 Data Already Normalized

The time series are already z-normalized to remove offset and scaling (transformed data have zero mean and in unit of standard deviation). The rationale for this step was previously discussed in the literature [98]; we will briefly review it here with an intuitive example. Consider the *GunPoint* data set shown in Figure 6.5. Suppose that we did not z-normalize the data but allowed our classifier to exploit information about the exact *absolute* height of the gun or hand. As it happens, this *would* help a little. However, imagine we collected more test data next week. Further suppose that for this second session, the camera zoomed in or out, or the actors stood a little closer to the camera, or that the female actor decided to wear new shoes with a high heel. None of these differences would affect z-normalized data as z-normalization accounts for offset and scale variance; however, they would drastically (negatively) affect any algorithm that exploited the raw un-normalized values.

Nevertheless, we acknowledge that for some (we believe, *very* rare) cases, data normalization is ill-advised. For the new data sets in this release, we provide the raw data without any normalization when possible; we explicitly state if the data has been normalized beforehand by the donors (the data might have been previously normalized by the donating source, who lost access to original raw data).

6.3.4 The Individual Data Sets are Too Small

While it is true that there is a need for bigger data sets in the era of "big data" (some algorithms specifically target scaling for big data sets), the archive has catered a wide array of data mining needs and lived up to its intended scope. The largest data set is *StarLightCurves* with 1,000 train and 8,236 test objects, covering 3 classes. The smallest data set is *Beef* with 30 train and 30 test objects, covering 5 different classes. Note that in recent years, there have been several published papers that say something to the effect of *"in the interests of time, we only tested on a subset of the archive"*. Perhaps a specialist archive of massive time series can be made available for the community in a different repository.

6.3.5 The Data Sets are not Reflective of Real-world Problems

This criticism is somewhat warranted. The archive is biased toward:

- data sets that reflect the personal interests/hobbies of the principal investigator (PI), Eamonn Keogh, including entomology (*InsectWingbeatSound*), anthropology (*Arrow-Head*) and astronomy (*StarLightCurves*). A wave of data sets added in 2015 reflect the personal and research interests of Tony Bagnall [6], many of which are image-to-time-series data sets. The archive has always had a policy of adding *any* donated data set, but offers of donations are surprisingly rare. Even when we actively solicited donations by writing to authors and asking for their data, we found that only a small subset of authors is willing to share data. The good news is that there appears to be an increasing willingness to share data, perhaps thanks to conferences and journals actively encouraging reproducible research.

- data sets that could be easily obtained or created. For example, fMRI data could be very interesting to study, but the PI did not have access to such a machine or the domain knowledge to create a classification data set in this domain. However, with an inexpensive scanner or a camera, it was possible to create many image-derived data sets such as *GunPoint, OSULeaf, SwedishLeaf, Yoga, Fish* or *FacesUCR*.

- data sets that do not have privacy issues. For many data types, mining the data while respecting privacy is an important issue. Unfortunately, none of the data sets in the UCR Archive motivates the need for privacy (though it is possible to use the data to construct proxy data sets).

6.3.6 Benchmark Results are from a Single Train/Test Split

Many researchers, especially those coming from a traditional machine learning background have criticized the archive for having a single train/test split. The original motivation for fixing the train and test set was to allow *exact* reproducibility. Suppose we simply suggested doing five-fold cross validation. Further suppose, someone claimed to be able to achieve an accuracy of A, on some data sets in the archive. If someone else re-implemented their algorithm and got an accuracy that is slightly lower than A during their five-fold cross validation, it would be difficult to know if that was within the expected variance of different folds, or the result of a bug or a misunderstanding in the new implementation. This issue would be less of a problem if everyone shared their code, and/or had very explicit algorithm descriptions. However, while the culture of open source code is growing in the community, such openness was not always the norm.

With a single train/test split, and a deterministic algorithm such as 1-NN, failure to *exactly* reproduce someone else‘s result immediately suggests an issue that should be investigated before proceeding with research. Note that while performing experiments on the single train/test split was always suggested as an absolute minimum sanity check; it did/does not preclude pooling the two splits and then performing K-fold cross validation or any other more rigorous evaluation.

6.4 How Bad is Cherry Picking?

It is not uncommon to see papers which report only results on a subset of the UCR Archive, without any justification or explanation. Here are some examples.

- *"We evaluated 1D-SAXLSSS classification accuracy on 22 data sets (see Table 2) taken from publicly available UCR repository benchmark"* [116]
- *"Figure 3 shows a performance gain of DSP-Class-SVM and DSP-Class-C5.0 approach in 5/11 data sets compared to another technique that does not use features (1-NN with Euclidean distance)"* [111].
- *"We experiment 48 small-scale data sets out of total 85 problems in the UCR time series archive"* [56]

We have no way to determine if these authors cherry-picked their limited subset of the archive, they may have selected the data on some unstated whim that has nothing to do with classification accuracy. However, without a statement of what that whim might be, we cannot exclude the possibility. Here, we will show how cherry picking can make a vacuous idea look good. Again, to be clear we are not suggesting that the works considered above are in any way disingenuous.

Consider the following section of text (italicized for clarity) with its accompanying table and figure, and imagine it appears in a published report. While this is a fictional report, note that all the numbers presented in the table and figure are *true* values, based on reproducible experiments that we performed.

We tested our novel FQT algorithm on 20 data sets from the UCR Archive. We compared to the Euclidean distance, a standard benchmark in this domain. Table T summarizes the results numerically, and Figure F shows a scatter plot visualization.

Table T: Performance comparison between Euclidean distance and our FQT distance. Our proposed FQT distance wins on all data sets that we consider.

Dataset	ED Error	FQT Error	Error Reduction
Strawberry	0.062	0.054	0.008
ECG200	0.120	0.110	0.010
TwoLeadECG	0.253	0.241	0.012
Adiac	0.389	0.376	0.013
ProximalPhalanxTW	0.292	0.278	0.014
DistalPhalanxTW	0.273	0.258	0.015
ProximalPhalanxOutlineCorrect	0.192	0.175	0.017
RefrigerationDevices	0.605	0.587	0.018
Wine	0.389	0.370	0.019
ProximalPhalanxOutlineAgeGroup	0.215	0.195	0.020
Earthquakes	0.326	0.301	0.025
ECGFiveDays	0.203	0.177	0.026
SonyAIBORobotSurfaceII	0.141	0.115	0.026
Lightning7	0.425	0.397	0.028
Trace	0.240	0.210	0.030
MiddlePhalanxTW	0.439	0.404	0.035
ChlorineConcentration	0.350	0.311	0.039
BirdChicken	0.450	0.400	0.050
Herring	0.484	0.422	0.062
CBF	0.148	0.080	0.068

Note that we used identical (UCR pre-defined) splits for both approaches, and an identical classification algorithm. Thus, all improvements can be attributed to our novel distance measure. The improvements are sometimes small, however, for CBF, Herring and BirdChicken, they are 5% (0.05) or greater, demonstrating that our FQT distance measure potentially offers significant gains in some domains. Moreover, we prove that FQT is a metric, and therefore easy to index with standard tree access methods.

(Returning to the current authors voice) The above results are all true, and the authors are correct in saying that FQT is a metric and is easier to index than DTW. So,

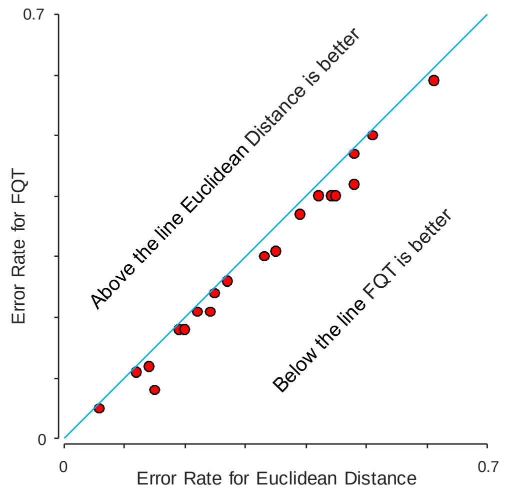

Figure F: The error rate of our FQT method compared to Euclidean distance. Our proposed method clearly outperforms the baseline for all datasets that we consider.

what is this remarkable FQT distance measure? It is simply the Euclidean distance after the first 25% of each time series is thrown away (First Quarter Truncation). Here is how we compute the FQT distance for time series A and B in MATLAB:

```
FQT_dist = sqrt(sum((A(end*0.25:end) - B(end*0.25:end)).^2))
```

If we examine the full 85 data sets in the archive, we will find that FQT wins on 19 data sets, but loses/draws on 66 (if we count a win as at least 1% reduction in error rate). Moreover, the size of the losses is generally more dramatic than the wins. For instance, the "error reduction" for *MedicalImages* is -0.23 (accuracy decreases by 23%).

Simply deleting the first quarter of every time series is obviously not a clever thing to do, and evaluating this idea on all the data sets confirms that. However, by cherry picking the twenty data sets that we chose to report, we made it seem like very good idea. It is true that in a full paper based on FQT we would have had to explain the measure, and it would have struck a reader as simple, unprincipled and unlikely to be truly useful.

However, there are many algorithms that would have the same basic *"a lot worse on most, a little better on a few"* outcome, and many of these could be framed to sound like plausible contributions (cf. Section 6.5.1).

In a recent paper, Lipton & Steinhardt list some *"troubling trends in machine learning scholarship"* [80]. One issue identified is "mathiness", defined as *"the use of mathematics that obfuscates or impresses rather than clarifies"*. We have little doubt that we could "dress up" our proposed FQT algorithm with spurious notation (Lipton and Steinhardt [80] call it *"fancy mathematics"*) to make it sound complex.

To summarize this section, cherry picking can make an arbitrary poor idea look useful, or even wonderful. Clearly, not all (or possibly even, not *any*) papers that report on a subset of the UCR data sets are trying to deceive the reader. However, as an outsider to the research effort, it is essentially impossible to know if the subset selection was random and fair (made *before* any results were computed) or biased to make the approach appear better than it really is.

The reasons given for testing only on a subset of the data (where any reason is given at all) is typically something like *"due to space limitations, we report only five of the …"*. However, this is not justified. A Critical Difference Diagram like Figure 6.3 or a scatter plot like Figure 6.4 require very little space but can summarize an arbitrary number of data sets. Moreover, one can always place detailed spreadsheets online or in an accompanying, cited technical report, as many papers do these days [30, 93].

That being said, we believe that sometimes there are good reasons to test a new algorithm on only a subset of the archive, and we applause researchers who explicitly justify their conduct. For example, Hills et al. stated: *"We perform experiments on 17 data sets from the UCR time-series repository. We selected these particular UCR data sets because they have relatively few cases; even with optimization, the shapelet algorithm is time consuming."* [57].

6.5 Best Practices for Using the Archive

Beating the performance of DTW on some data sets should be considered a *necessary*, but not *sufficient* condition for introducing a new distance measure or classification algorithm. This is because the performance of DTW itself can be improved with very little effort, in at least a dozen ways. In many cases, these simple improvements can close most or all the gap between DTW and the more complex measures being proposed. For example:

- The warping window width parameter of constrained DTW algorithm is tuned by the "quick and dirty" method described in Section 6.2. As Figure 2.2 bottom row shows, on at least some data sets, that tuning is sub-optimal. The parameter could be tuned more carefully in several ways such as by re-sampling or by creating synthetic examples [30].

- The performance of DTW classification can often be improved by other trivial changes. For example, as shown in Figure 6.2.*left*, simply smoothing the data can produce significant improvements. Figure 6.2.*right* shows that generalizing from 1-nearest neighbor to k-nearest neighbors often helps. One can also test alternative DTW step patterns [83]. Making DTW "endpoint invariant" helps on many data sets [112], etc.

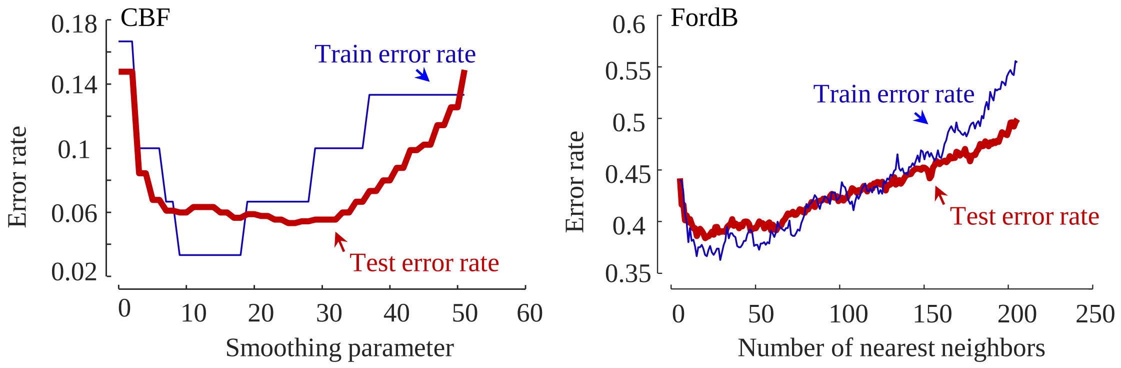

Figure 6.2: *left*) The error rate on classification of the *CBF* data set for increasing amounts of smoothing using MATLAB's default smoothing algorithm. *right*) The error rate on classification of the *FordB* data set for increasing number of nearest neighbors. Note that the leave-one-out error rate on the training data does approximately predict the best parameter to use.

An hour spent on optimizing any of the above could improve the performance of ED/DTW on at least several data sets of the archive.

6.5.1 Mis-attribution of Improvements: a Cautionary Tale

We believe that of the several hundred papers that show an improvement on the baselines for the UCR Archive, a not small fraction is mis-attributing the cause of their improvement and is perhaps *indirectly* discovering one of the low hanging fruits above. This point has recently been made in the more general case by researchers who believe that many papers suffer from a failure *"to identify the sources of empirical gains"* [80]. Below we show an example to demonstrate this.

Many papers have suggested using a wavelet representation for time series classification[2], and have gone on to show accuracy improvements over either the DTW or ED baselines. In most cases, these authors attribute the improvements to the multi-resolution

[2]These works should not be confused with papers that suggest using a wavelet representation to perform dimensionality reduction to allow more efficient indexing of time series.

properties of wavelets. For example (our emphasis in the quotes below):

- *"wavelet compression techniques can sometimes even help achieve higher classification accuracy than the raw time series data, as they better capture essential local features ... As a result, we think it is safe to claim that multi-level wavelet transformation is indeed helpful for time series classification."* [72]
- *"our multi-resolution approach as discrete wavelet transforms have the ability of reflecting the local and global information content at every resolution level."* [72]
- *"We attack above two problems by exploiting the multi-scale property of wavelet decomposition ... extracting features combining the global information and partial information of time series."* [133]
- *"Thus multi-scale analyses give us the ability of observing time series in various views."* [132]

As the quotes above suggest, many authors attribute their accuracy improvements to the multi-resolution nature of wavelets. However, we have a different hypothesis. The wavelet representation is simply smoothing the data implicitly, and all the improvements can be attributed to *just* this smoothing! Figure 6.2 above does offer evidence that at least on some data sets, appropriate smoothing is enough to make a difference that is commensurate with the claimed improvements. However, is there a way in which we could be sure? Yes, we can exploit an interesting property of the Haar Discrete Wavelet Transform (DWT).

Note that if both the original and reduced dimensionality of the Haar wavelet transform are powers of two integers, then the approximation produced by Haar is logically *identical* to the approximation produced by the Piecewise Aggerate Approximation [66]. This means that the distances calculated in the truncated coefficient space are identical for both approaches, and thus they will have the same classification predictions *and* the same error rate. If we revisit the *CBF* data set shown in Figure 6.2, using Haar with 32 coefficients we get an error rate of just 0.05, much better than the 0.148 we would have obtained using the raw data. Critically, however, PAA with 32 coefficients also gets the same 0.05 error rate.

It is important to note that PAA is not in any sense *multi-resolution* or *multiscale*. Moreover, by the definition of PAA, each coefficient being the *average* of a *range* of points, is very similar to the definition of the moving average filter smoothing, with each point being *averaged* with its neighbors within a *range* to the left and the right. We can do one more test to see if the Haar Wavelet is offering us something beyond smoothing. Suppose we use it to classify data that have *already* been smoothed with a smoothing parameter of 32. Recall (Figure 6.2) that using this smoothed data directly gives us an error rate of 0.055. Will Haar Wavelet classification further improve this result? No, in fact using 32 coefficients, both Haar Wavelet and PAA classification produce very slightly worse results of 0.057, presumably because we have effectively smoothed the data twice, and by doing so, over-smoothed it (again, see Figure 6.2, and examine the trend of the curve as the smoothing parameter grows above 30).

In our view, these observations cast significant doubt on the claim that the improvements obtained can correctly be attributed to multi-resolution properties of wavelets. There are two obvious reasons as to why this matters:

- mis-attribution of *why* a new approach works potentially leaves adopters in the position of fruitless follow-up work or application of the proposed ideas to data/tasks for which they are not suited;

- if the problem at hand is really to improve the accuracy of time series classification, and if five minutes spent experimenting with a smoothing function can give you the same improvement as months implementing a more complex method, then surely the former is more desirable, only less publishable.

This is simply one example. We suspect that there are many other examples. For instance, many papers have attributed time series classification success to their exploitation of the "memory" of Hidden Markov Models, or "long-term dependency features" of Convolution Neural Networks etc. However, one almost never sees an ablation study that forcefully convinces the reader that the *claimed* reason for improvement is correct.

In fairness, a handful of papers do explicitly acknowledge that. While they may introduce a complex representation or distance measure for classification, at least some of the improvements should be attributed to smoothing. For example, Schäfer notes: *"Our Bag-of-SFA-Symbols (BOSS) model combines the extraction of substructures with the tolerance to extraneous and erroneous data using a noise reducing representation of the time series"* [107]. Likewise, Li and colleagues [73] revisit their Discrete Wavelet Transformed (DWT) time series classification work [72] to explicitly ask *"if the good performances of DWT on time series data is due to the implicit smoothing effect"*. They show that their previous embrace of wavelet-based classification does not produce results that are better than simple smoothing in a statistically significant way. Such papers, however, remain an exception.

6.5.2 How to Compare Classifiers

The Choice of Performance Metric

Suppose you have an archive of one hundred data sets and you want to test whether classifier A is better than classifier B, or compare a set of classifiers, over these data. At first, you need to specify what you mean by one classifier being "better" than another. There are two general criteria with which we may wish compare classifier ability on data not used in the training process: the prediction ability and the probability estimates.

The ability to predict is most commonly measured by accuracy (or equivalently, error rate). However, accuracy does not always tell the whole story. If a problem has class imbalance, then accuracy may be less informative than a measure that compensates for this skewed classes. Sensitivity, specificity, precision, recall and the F statistic are all commonly used for two-class problems where one class is rarer than the other, such as medical diagnosis trials [91, 119]. However, these measures do not generalize well to multi-class problems or scenarios where we cannot prioritize one class using domain knowledge. For the general case over multiple diverse data sets, we consider accuracy and balanced accuracy enough to assess predictive power. Conventional accuracy is the proportion of examples correctly classified while balanced accuracy is the average of accuracy for each class individually [16]. Some classifiers produce scores or probability estimates for each class and these can be summarized with statistics such as negative log-likelihood or area under the receiver operating characteristic curve (AUC). However, if we are using a classifier that only produces predictions (such as 1-NN), these metrics do not apply.

The Choice of Data Split

Having decided on a comparison metric, the next issue is what data you are going to evaluate the data on. If a train/test split is provided, it is natural to start by building all the classifiers (including all model selection/parameter setting) on the train data, and then assess accuracy on the test data.

There are two main problems with using a single train test split to evaluate classifiers. First, there is a temptation to cheat by setting the parameters to optimize the test data. This can be explicit, for example, by setting an algorithm to stop learning when test accuracy is maximized, or implicit, by setting default values based on knowledge of the test split. For example, suppose we have generated results such as Figure 6.2, and have a variant of DTW we wish to assess on this test data. We may perform some smoothing and set the parameter to a default of 10. Explicit bias can only really be overcome with complete code transparency. For this reason, we *strongly* encourage users of the archive to make their code available to both reviewers and readers.

The other problem with a single train/test split, particularly with small data sets, is that tiny differences in performance can seem magnified. For example, we were recently contacted by a researcher who queried our published results for 1-NN DTW on the UCR archive train/test splits. When comparing our accuracy results to theirs, they noticed that they differ by as much as 6% in some instances, but there was no significant difference for other problem sets. Still, we were concerned by this single difference, as the algorithm in question is deterministic. On further investigation, we found out that our data were rounded to six decimal places, theirs to eight. These differences on single splits were caused by small data set sizes and tiny numerical differences (often *just* a single case classified differently).

These problems can be largely overcome by merging the train and test data and re-sampling each data set multiple times then averaging test accuracy. If this is done, there are several caveats:

- the default train and test splits should always be included as the first re-sample;
- re-samples must be the same for each algorithm;
- the re-samples should retain the initial train and test sizes;
- the re-samples should be stratified to retain the same class distribution as the original.

Even when meeting all these constraints, re-sampling is not always appropriate. Some data are constructed to keep experimental units of observation in difference data sets. For example, when constructing the alcohol fraud detection problem [78], we used different bottles in the experiments and made sure that observations from the same bottle does not appear in both the train and the test data. We do this to make sure we are not detecting bottle differences rather than different alcohol levels. Problems such as these discourage practitioners from re-sampling. However, we note that the majority of machine learning research involves repeated re-samples of the data.

Finally, for clarity we will repeat our explanation in Section 6.3.6 as to why we have a single train/test split in the archive. Publishing the results of both ED and DTW distance on a single deterministic split provides researchers a useful sanity check, before they perform more sophisticated analysis [32, 44, 105].

The Choice of Significance Tests

Whether through a single train/test split or through re-sampling then averaging, you now arrive at a position of having multiple accuracy estimates for each classifier. The core question is, are there significant differences between the classifiers? In the simpler scenario, suppose we have two classifiers, and we want to test whether the differences in average accuracy is different from zero. There are two alternative hypothesis tests that one could go for, a paired two-sample t-test [114] for evidence of a significant difference in mean accuracies or a Wilcoxon signed-rank test [127, 110] for differences in median accuracies. Generally, machine learning researchers favor the latter. However, it is worth noting that many of the problems identified with parametric tests in machine learning derive from the problem of too few data sets, typically twenty or less. With more than 30 data sets, the central limit theorem means these problems are minimized. Nevertheless, we advise using a Wilcoxon signed-rank test with a significance level of α set at or smaller than 0.05 to satisfy reviewers.

What if you want to compare multiple classifiers on these data sets? We follow the recommendation of Demšar [32] and base the comparison on the ranks of the classifiers on each data set rather than the actual accuracy values. We use the Friedmann test [42, 43] to determine if there were any statistically significant differences in the rankings of the classifiers. If differences exist, the next task is to determine where they lie. This is done by forming cliques, which are groups of classifiers within which manifest significant difference. Following recent recommendations in [13] and [44], we have abandoned the Nemenyi post-hoc test [58] originally used by Demšar [32]. Instead, we compare all classifiers with pairwise Wilcoxon signed-rank tests and form cliques using the Holm correction, which adjusts family-wise error less conservatively than a Bonferonni adjustment [59]. We can summarize these comparisons in a critical difference diagram such as Figure 6.3.

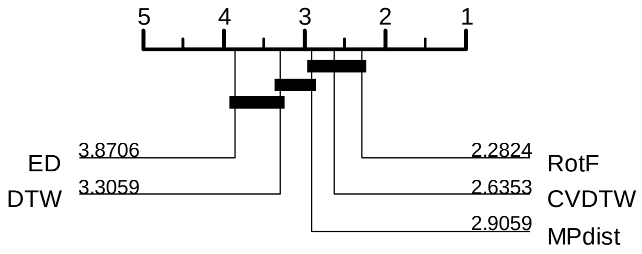

Figure 6.3: Critical difference for MPdist distance against four benchmark distances. Figure credited to Gharghabi et al. [46]. We can summarize this diagram as follow: RotF is the best performing algorithm with an average rank of 2.2824; there is an overall significant difference among the five algorithms; there are three distinct cliques; MPdist is significantly better than ED distance and not significantly worse than the rest.

Figure 6.3 displays the performance comparison between MPdist, a recently proposed distance measure and other competitors [46]. This diagram orders the algorithms and presents the average rank on the number line. Cliques of methods are grouped with a solid bar showing groups of methods within which there is no significant difference. According to Figure 6.3, the best ranked method is Rotation Forest (RotF), however, it is not statistically better than the other two methods in its clique, CVDTW and MPdist.

6.5.3 A Checklist

We propose the following checklist for any researcher who is proposing a novel time series classification/clustering technique and would like to test it on the UCR Archive.

1. Did you test on *all* the data sets? If not, you should carefully explain why not, to avoid the appearance of cherry picking. For example, *"we did not test on the large*

data sets, because our method is slow" or *"our method is only designed for ECG data, so we only test on the relevant data sets"*.

2. Did you take care to optimize all parameters on just the training set? Producing a Texas Sharpshooter plot is a good way to visually confirm this for the reviewers [10]. Did you perform an appropriate statistical significance test? (see Section 6.5.2)

3. If you are claiming your approach is better due to property X, did you conduct an ablation test (lesion study) to show that if you remove this property, the results worsen, and/or, if you endow an otherwise unrelated approach with this property, that approach also improves?

4. Did you share your code? Note, some authors state in their submission, something to the effect of *"We will share code when the paper gets accepted"*. However, sharing of documented code at the time of submission is the best way to imbue confidence in even the most cynical reviewers.

5. If you modified the data in any way (adding noise, smoothing, interpolating, etc.), did you share the modified data, or the code, with random seed generator, that would allow a reader to exactly reproduce the data.

6.6 The New Archive

On January 2018, we reached out about forty researchers soliciting ideas for the new UCR Time Series Archive. They are among the most active researchers in the time series data mining community, who have used the UCR Archive in the past. We raised the question: *"What would you like to see in the new archive?"*. We saw a strong consensus on the following needs:

- Longer time series
- Variable length data sets
- Multi-variate data sets
- Information about the provenance of the data sets

Some researchers also wish to see the archive to include data sets suitable for some specific research problems. For example:

- data sets with highly unbalanced classes
- data sets with very small training set to benchmark data augmentation techniques

Researchers especially raised the need for bigger data sets in the era of big data.

"To me, the thing we lack the most is larger data sets; the largest training data has 8,000 time series while the community should probably move towards millions of examples. This is a wish, but this of course doesn't go without problems: how to host large data sets, will future researchers have to spend even more time running their algorithms, etc." (François Petitjean, Monash University)

Some researchers propose sharing pointers to genuine data repositories and data mining competitions.

"A different idea that might be useful is to add data set directories that have pointers to other data sets that are commonly used and freely available. When the UCR Archive

first appeared, it was a different time, with fewer high quality, freely available data sets that were used by many researchers to compare results. Today there are many such data sets, but you tend to find them with Google, or seeing mentions in papers. One idea would be to pull together links to those data sets in one location with a flexible "show me data sets with these properties or like this data set" function." (Tim Oates, University of Maryland Baltimore County).

While some of these ideas may go beyond the ambition of the UCR Archive, we think that it inspires a wave of effort in making the time series community better. We hope *others* will follow suit our effort in making data sets available for research purposes. In a sense, we think it would be inappropriate for our group to provide all such needs, as this monopoly might bias the direction of research to reflect our interests and skills.

We have addressed *some* of the perceived problems with the existing archive and some of what the community want to see in the new archive. We follow with an introduction of the new archive release.

6.6.1 General Introduction

We refer to archive before the Fall 2018 expansion as the *old* archive and the current version as the *new* archive. The Fall 2018 expansion sees the number of data sets increase from 85 to 128. We adopt a standard naming convention, that is using captions for words and no underscores. We include the provenance of all the data sets. In editing data for the new archive, in most cases, we make the test set bigger than the train set to reflect real-world scenarios, i.e., labeled train data are usually expensive. We keep the data sets as is if they are from published papers and are already in a UCR Archive preferred format.

In Figure 6.4 we use the Texas Sharpshooter plot popularized by Batista et al. [10] to show the baseline result comparison between 1-NN ED and 1-NN constrained DTW on 128 data sets of the new archive. The Texas Sharpshooter plot is introduced to avoid the Texas sharpshooter fallacy [10], that is a simple logic error that seems pervasive in time series classification papers. Many papers show that their algorithm/distance measure are better than the baselines/competitors on some data sets, ties on many and loses on some. They then claim their method works for some domains and thus it has value. However, it is not useful to have an algorithm that are good for some problems unless you can tell *in advance* which problems they are.

The Texas Sharpshooter plot in Figure 6.4 compares ED and constrained DTW distance by showing the expected accuracy gain (based solely on train data) versus the actual accuracy gain (based solely on test data) of the two methods. Note that here, the improvement of constrained DTW over ED is almost tautological, as constrained DTW subsumes ED as a special case. More generally, these plots visually summarize the strengths and weaknesses of rival methods.

6.6.2 Some Notes on the Old Archive

We reverse the train/test split of fourteen data sets to make them consistent with their original release, i.e. when they were donated to the archive, according to the wish of the original donors. These data sets were accidentally reversed during the 2015 expansion of the archive. The train/test split of these data set now agree with the train/test split hosted at the UEA Archive [6], and are the split that was used in a recent influential survey paper

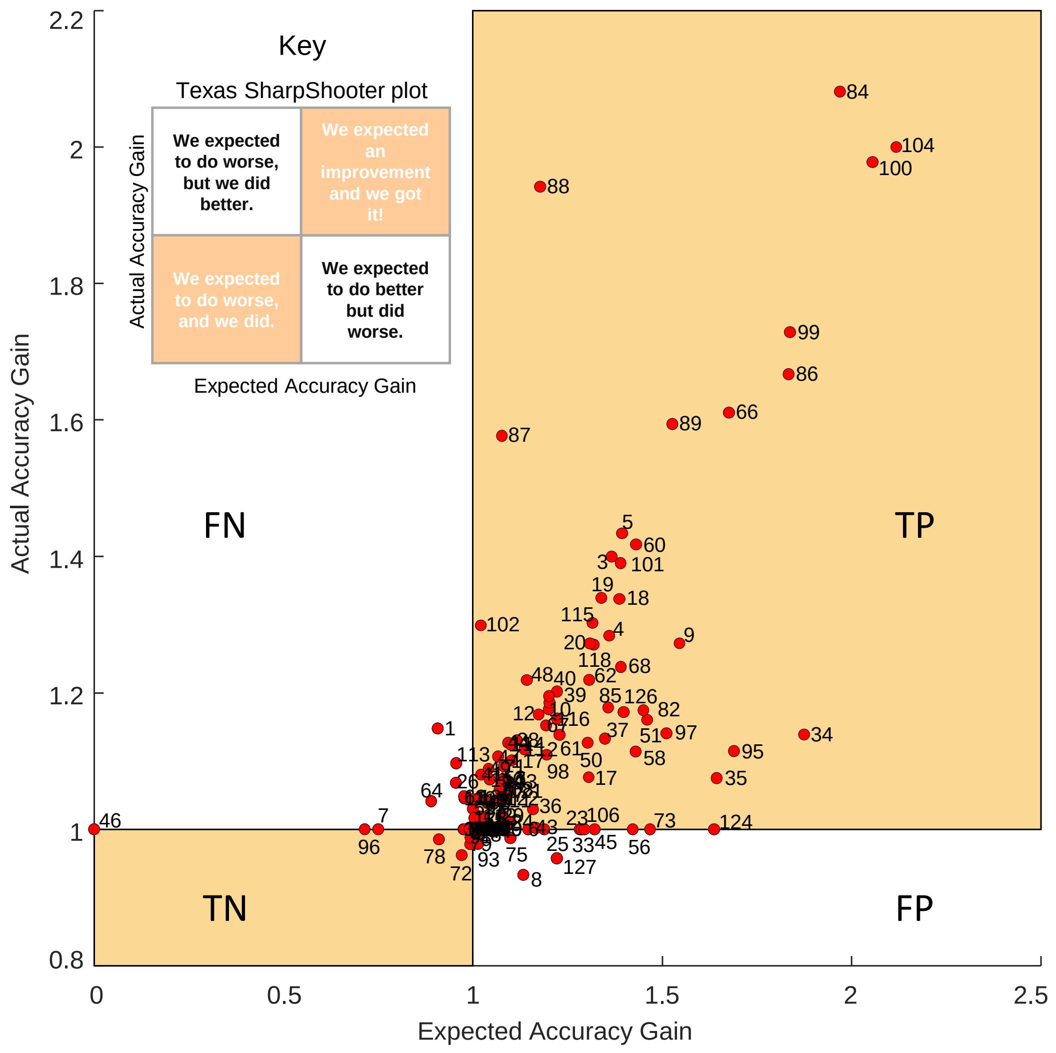

Figure 6.4: Comparison of Euclidean distance versus constrained DTW for 128 data sets. In the Texas Sharpshooter plot, each data set falls into one of four possibilities (True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN)) corresponding to four quadrants (see the key on top left corner). We optimize the performance of DTW by learning a suitable warping window width and compare the expected improvement with the actual improvement. The results are strongly supportive of the claim that DTW is better than Euclidean distance for most problems. Note that some of the numbers are hard to read because they overlap. A higher resolution version of this image is available at the UCR Archive webpage [28].

titled *"The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances"* [5]. We list these data sets in Table 6.1.

Table 6.1: Fourteen data sets that have the train/test split reversed for the new archive expansion

Data set name	
DistalPhalanxOutlineAgeGroup	DistalPhalanxOutlineCorrect
DistalPhalanxTW	Earthquakes
FordA	FordB
HandOutlines	MiddlePhalanxOutlineAgeGroup
MiddlePhalanxOutlineAgeCorrect	MiddlePhalanxTW
ProximalPhalanxTW	Strawberry
Worms	WormsTwoClass

Among the 85 data sets of the old archive, there are twelve data sets that at least one algorithm gets 100% accuracy [79, 6]. We list them in Table 6.2.

Table 6.2: Twelve "solved" data sets, for which at least one algorithm gets 100% accuracy

Type	Data set name	Type	Data set name
Image	BirdChicken	Sensor	Plane
Spectrograph	Coffee	Simulated	ShapeletSim
ECG	ECGFiveDays	Simulated	SyntheticControl
Image	FaceFour	Sensor	Trace
Motion	GunPoint	Simulated	TwoPatterns
Spectrograph	Meat	Sensor	Wafer

6.6.3 Data Set Highlights

GunPoint Data Sets

The original *GunPoint* data set was created by current authors Ratanamahatana and Keogh in 2003. Since then, it has become the "iris data" of the time series community

[38], being used in over one thousand papers, with images from the data set appearing in dozens of papers (see Figure 6.5). As part of these new release of the UCR Archive, we decided to revisit this problem, by asking the two original actors to recreate the data.

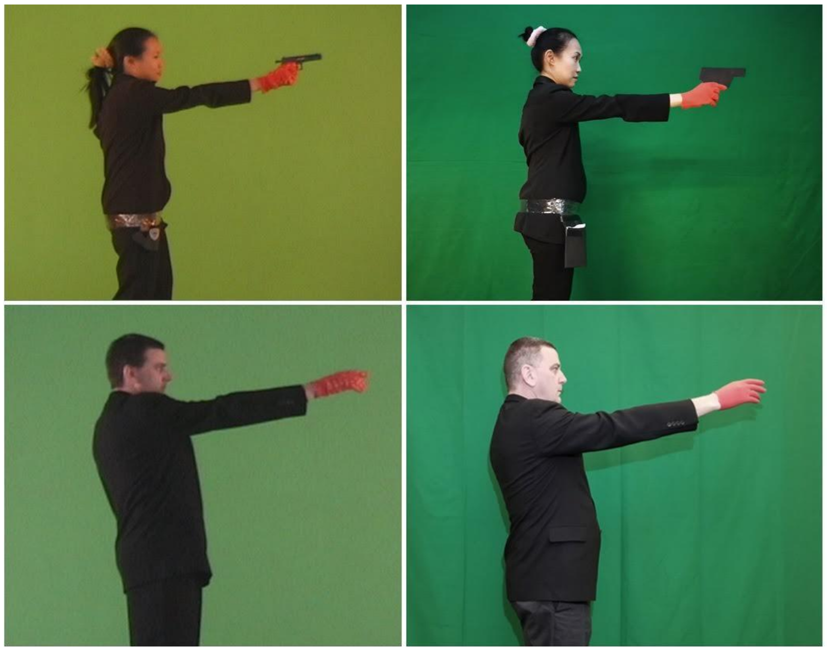

Figure 6.5: *left*) *GunPoint* recording of 2003 and *right*) *GunPoint* recording of 2018. The female and male actors are the same individuals recorded fifteen years apart.

We record two scenarios, "Gun" and "Point". In each scenario, the actors aim at a target at eye level before them. We strived to reproduce in every aspect the recording of the original *GunPoint* data set created 15 years ago. The difference between Gun and Point is that in the Gun scenario, the actor holds a replica gun. They point the gun at the target, return the gun back to the waist holster and then brings their free hand to a rest position to complete an action. Each complete action conforms to a five-second cycle. We filmed with a commodity smart-phone Samsung Galaxy 8. With 30fps, this translates into 150 frames per action. We generated a time series for each action by taking the x-axis location of the centroid of the red gloved hand (see Figure 6.5). We merged the data of this new recording with the old *GunPoint* data to make several new data sets.

The data collected now spans two actors F,M, two behaviors G,P, and two years 03,18. The task of the original *GunPoint* data set was differentiating between the Gun and the Point action: FG03, MG03 vs. FP03, MP03. We have created three new data sets. Each data set has two classes; each class is highly polymorphic with four variants characterizing it.

- *GunPointAgeSpan*: FG03, MG03, FG18, MG18 vs. FP03, MP03, FP18, MP18. The task is to recognize the actions with invariance to the actor, as with *GunPoint* before, but also be invariant to the year of recording.
- *GunPointOldVersusYoung*: FG03, MG03, FP03, MP03 vs. FG18, MG18, FP18, MP18, which asks if a classifier can detect the difference between the recording sessions due to (perhaps) the actors aging, differences in equipment and processing; though as noted above, we tried to minimize such inconsistencies. In this case, the classifier needs to ignore the action and actor.
- *GunPointMaleVersusFemale*: FG03, FP03, FG18, FP18 vs. MG03, MP03, MG18, MP18, which asks if a classifier can differentiate between the build and posture of the two actors.

GesturePebble **Data Sets**

The archive expansion includes several data sets whose time series exemplars can be of different lengths. The *GesturePebble* data set is one of them. For ease of data handling, we pad enough NaNs to the end of each time series, to make it the same length of the longest time series. Some algorithms/distance measures can handle variable-length data directly; other researchers may have to process such data by truncation or re-normalization etc. We deliberately refrain from offering any advice on how to best do this.

The *GesturePebble* data set comes from the paper *"Gesture Recognition using Symbolic Aggregate Approximation and Dynamic Time Warping on Motion Data"* [86]. This work is among the many that study the application of commodity smart devices as a motion sensor for gesture recognition. The data is collected with the 3-axis accelerometer Pebble smart watch mounted to the participants wrist. Each subject is instructed to perform six gestures depicted in Figure 6.6. The data collection included four participants, each of which repeated the gesture set in two separate sessions a few days part. In total, there are eight recordings, which contain 304 gestures. Since the duration of each gestures varies, the time series representing each gesture are of different lengths. Figure 6.7 shows some samples of the data.

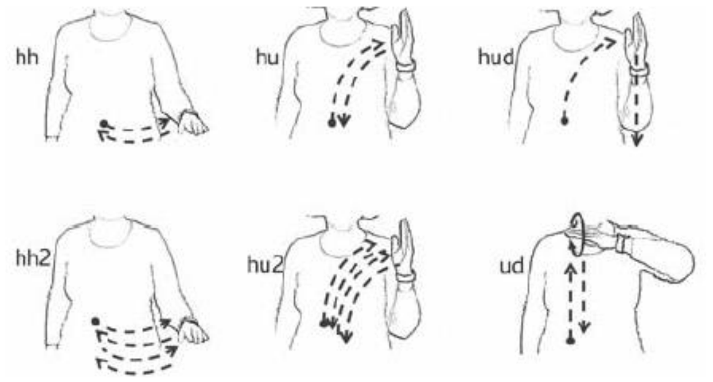

Figure 6.6: The dot marks the start of a gesture. The labels (hh, hu, hud, etc) are used by original authors of the data and may not have any special meaning. The gestures are selected based on criteria that they are characterized by the wrist movements; they simple and natural enough to replicate; and they can be related to commands to control devices [86].

We created two data sets from the original data, both using only the z-axis reading (out of the three channels/attributes available).

- *GesturePebbleZ1*: The train set consists of data of all subjects collected in the first session. The test set consists of all data collected in the second session. This way, data of each subject appear in both train and test set.
- *GesturePebbleZ2*: The train set consists of data of two subjects and the test set consists of data of the other two subjects. This data set is intended to be more difficult than *GesturePebbleZ1* because the subjects in the test set do not appear in the train set (presumably that each participant possesses unique gait and posture and move differently). Baseline results confirm this speculation.

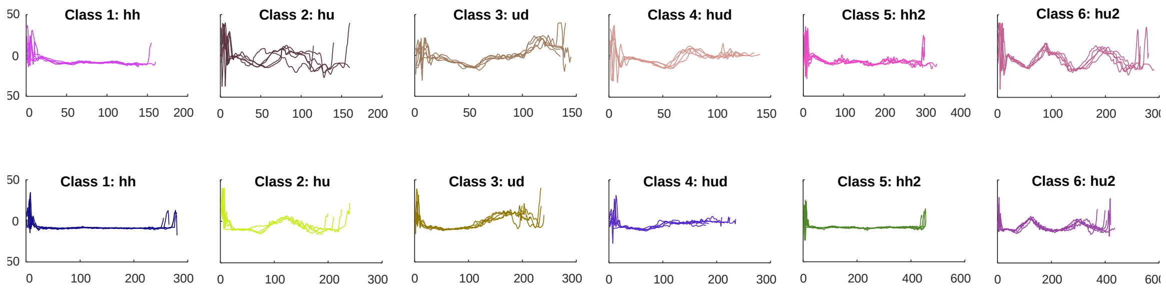

Figure 6.7: Data from recording of two different subjects performing a same set of six gestures (top row and bottom row). The endpoints of the time series contain the "tap event", which are abrupt movements to signal the start and end of the gesture (deliberately performed by the subject). The accelerometer data are also under influence of gravity and the devices orientation during the movement.

Electrical Load Measurement Data - *Freezer* Data Sets

This data set was derived from a multi-institution project entitled Personalized Retrofit Decision Support Tools for UK Homes using Smart Home Technology (REFIT) [89]. The data set includes data from twenty households from the Loughborough area over the period of 2013-2014. All data are from freezers in House 1. This data set has two classes, one representing the power demand of the fridge freezer in the kitchen, the other representing the power demand of the (less frequently used) freezer in the garage.

The two classes are difficult to tell apart globally. Each consists of a flat region (the compressor is off), followed by an instantaneous increase (the compressor is switched on), followed by a slower decease as the compressor builds some rotational inertial. Finally, once the temperature has been lowered enough, there is an instantaneous fall back to a flat region (this part may be missing in some exemplars). The amount of time the compressor is on can vary greatly, but that is not class-dependent. In Figure 6.8 however, if you examine the region just after the fiftieth data point, you can see a subtle class-conserved difference in how fast the compressor builds rotational inertial and decreases its power demand. An algorithm that can exploit this difference could do well on these data sets; however, global algorithms, such as 1-NN with Euclidean distance may have a hard time beating the default rate.

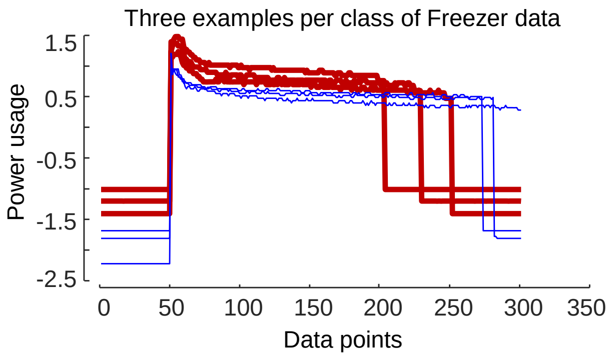

Figure 6.8: Some examples from the two-class *Freezer* data set. The two classes (blue/fine lines vs. red/bold lines) represent the power demand of the fridge freezers sitting in different locations of the house. The classes are difficult to tell apart globally but they differ locally.

Freezer is an example of data sets with different train/test splits (the others are *InsectEPG* and *MixedShapes*). We created two train set versions: a smaller train set and a regular train set (both are accompanied by a same test set). This is to meet the community's demand of benchmarking algorithms that are able to work with little train data, for example, generating synthetic time series to augment sparse data sets [30, 39]. Some algorithms produce favorable results when training exemplars are abundant but deteriorate when the train data are scarce.

InternalBleeding **Data Sets**

The source data set is data from fifty-two pigs having three vital signs monitored, before and after an induced injury [50]. We created three data sets out of this source: *AirwayPressure* (airway pressure measurements), *ArtPressure* (arterial blood pressure measurements) and *CVP* (central venous pressure measurements). Figure 6.9 shows a sample of these data sets. In a handful of cases, data may be missing or corrupt; we have done nothing to rectify this.

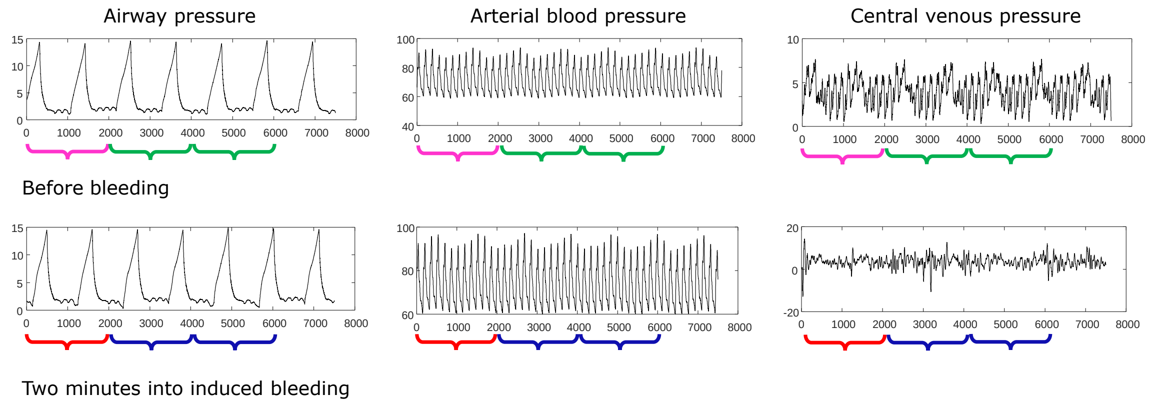

Figure 6.9: *InternalBleeding* data sets. Class i is the i^{th} individual pig. In the training set, class i is represented by two examples, the first 2000 data points of the *"before"* time series (pink braces), and the first 2000 data points of the *"after"* time series (red braces). In the test set, class i is represented by four examples, the second and third 2000 data points of the *"before"* time series (green braces), and the second and third 2000 data points of the *"after"* time series (blue braces).

These data sets are interesting and challenging for several reasons. Firstly, the data are not phase-aligned, which may call for a phase-invariant or an elastic distance measure. Secondly, the number of classes is huge, especially relative to the number of training instances. Finally, each class is polymorphic, having exactly one example from the pig when it was healthy, and one from when it was injured.

EthanolLevel **Data Set**

This data set was produced as part of a project with Scotch Whisky Research Institute into non-intrusively detecting forged spirits (counterfeiting whiskey is an increasingly lucrative crime). One such candidate method of detecting forgeries is by examining the ethanol level extracted from a spectrograph. The data set covers twenty different bottle types and four levels of alcohol: 35%, 38%, 40% and 45%. Each series is a spectrograph of 1,751 observations. Figure 6.10 shows some examples of each class.

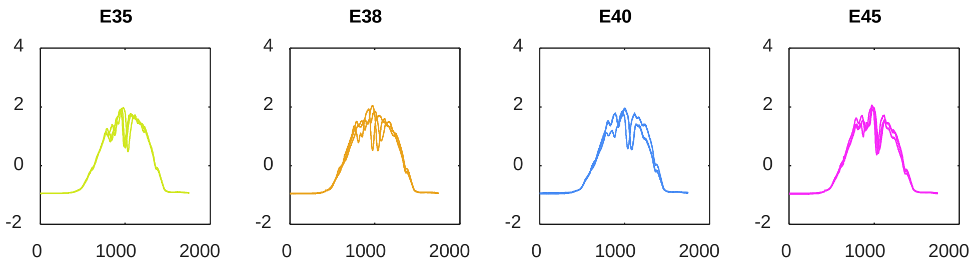

Figure 6.10: Three examples per class of *EthanolLevel* data set. The four classes correspond to levels of alcohol: 35%, 38%, 40% and 45%. Each series is 1,751 data-point long.

This data set is an example of when it is wrong to merge and resample, because the train/test sets are constructed so that the same bottle type is never in both data sets. The data set was introduced in *"HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification"* [78].

6.7 Conclusions and Future Work

We have introduced the 128-data-set version of the UCR Time Series Archive. This resource is made freely available at the online repository in perpetuity [28]. We have further offered advice to the community on best practices on using the archive to test classification algorithms, although we recognize that the community is free to ignore all such advice. Finally, we offered a cautionary tale about how easily practitioners can inadvertently mis-attribute improvements in classification accuracy. We hope this will encourage all users (including the current authors) to have deeper introspection about the evaluation of proposed distance measures and algorithms.

Chapter 7

Conclusions

Motivated from enabling large scale data mining of time series data to produce more accurate, reproducible results and tailed to user's need, we have discussed methods to learn constraints on model's flexibility in various settings. If we let the model over-exploit a flexibility or impose no restriction on the search space, the outcome could be meaningless or even mis-leading. In this context, the contributions of this dissertation are as follows:

- 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 data sets, 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 hard-coded all parameters). However, experimenting with adaptive parameters may allow others to improve upon our results.

- We have shown that the direct application of classic time series motif discovery tool can produce unexpected/undesired results in many circumstances. Some of these issues are intrinsic characteristics of classic motif search, though this fact may not be obvious to many of its practitioners. More fundamentally, current motif search lacks support for user's embedding preferences and domain knowledge, causing useful motifs to be buried among an ocean of fruitless ones. We have presented a novel framework which greatly mitigate these issues. Moreover, our framework is built on top of the state-of-the-art motif discovery tool, the matrix profile [130], thus inherits all of its favorable characteristics such as speed, exactness and anytime property.

- We have introduced the new 128-data-set version of the UCR Archive. Apart from the newly added 43 data sets, this archive release is accompanied with richer documentation and more baseline results. Having more pubic data sets enables gauging progress and comparison of rival approaches. We hope that the UCR Archive will continue to promote the advances and reproducible research among time series data mining community. We have further offered advice to the users of the archive on how to best make use of these resources. We took the opportunity to point out how easily practitioners can inadvertently mis-attribute improvements in classification accuracy or make a lame algorithm appear useful by cherry-picking data sets to test on.

The works presented in this dissertation have been published in several conferences and journals [25, 26, 27, 29, 30], and have attracted hundreds of citations. 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.

Bibliography

[1] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *International Conference on Foundations of Data Organization and Algorithms*, pages 69–84. Springer, 1993.

[2] Mark V. Albert, Konrad Kording, Megan Herrmann, and Arun Jayaraman. Fall classification by machine learning using mobile phones. *PLoS ONE*, 7(5), 2012.

[3] Vassilis Athitsos, Panagiotis Papapetrou, Michalis Potamias, George Kollios, and Dimitrios Gunopulos. Approximate embedding-based subsequence matching of time series. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 365–378. ACM, 2008.

[4] Anthony Bagnall and Jason Lines. An experimental evaluation of nearest neighbour time series classification. *arXiv preprint arXiv:1406.4757*, 2014.

[5] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

[6] Anthony Bagnall, Jason Lines, William Vickers, and Eamonn Keogh. The UEA and UCR Time Series Classification Repository. `www.timeseriesclassification.com`, 2018.

[7] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[8] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1601–1608, 2004.

[9] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets*, 6(1):20–29, 2004.

[10] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius M A De Souza. CID: An efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, 2014.

[11] Nurjahan Begum and Eamonn Keogh. Rare time series motif discovery from unbounded streams. *Proceedings of the VLDB Endowment*, 2014.

[12] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. 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*, pages 49–58, 2015.

[13] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. Should we really use post-hoc tests based on mean-ranks. *Journal of Machine Learning Research*, 17(5):1–10, 2016.

[14] Mikhail Bilenko and Raymond J. Mooney. 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*, page 39, 2003.

[15] Sabrina Brigadoi, Lisa Ceccherini, Simone Cutini, Fabio Scarpa, Pietro Scatturin, Juliette Selb, Louis Gagnon, David A. Boas, and Robert J. Cooper. Motion artifacts in functional near-infrared spectroscopy: A comparison of motion correction techniques applied to real cognitive data. *NeuroImage*, 2014.

[16] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann. The balanced accuracy and its posterior distribution. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 3121–3124. IEEE, 2010.

[17] Hong Cao, Xiao Li Li, David Yew Kwong Woon, and See Kiong Ng. Integrated oversampling for imbalanced time series classification. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2809–2822, 2013.

[18] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[19] Yanping Chen, Bing Hu, Eamonn Keogh, and Gustavo EAPA Batista. DTW-D: Time series semi-supervised learning from a single example. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 383–391. ACM, 2013.

[20] Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive. `https://www.cs.ucr.edu/~eamonn/time_series_data/`, 2015.

[21] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*, 2003.

[22] Subimal Datta. Cellular basis of pontine ponto-geniculo occipital wave generation and modulation. *Cellular and Molecular Neurobiology*, 1997.

[23] Hoang Anh Dau. Supporting page for guided motif search. `https://www.cs.ucr.edu/~hdau001/guided_motif_search/`, 2017.

[24] Hoang Anh Dau. Supporting page for learning DTW's window width. `http://www.cs.ucr.edu/~hdau001/learn_dtw_parameter/`, 2018.

[25] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *arXiv preprint arXiv:1810.07758*, 2018.

[26] Hoang Anh Dau, Nurjahan Begum, and Eamonn Keogh. Semi-supervision dramatically improves time series clustering under dynamic time warping. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 999–1008. ACM, 2016.

[27] Hoang Anh Dau and Eamonn Keogh. Matrix profile V: A generic technique to incorporate domain knowledge into motif discovery. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134. ACM, 2017.

[28] Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping Chen, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The UCR Time Series Classification Archive. `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`, 2018.

[29] Hoang Anh Dau, Diego Furtado Silva, François Petitjean, Germain Forestier, Anthony Bagnall, and Eamonn Keogh. Judicious setting of dynamic time warping's window width allows more accurate classification of time series. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 917–922. IEEE, 2017.

[30] Hoang Anh Dau, Diego Furtado Silva, Francois Petitjean, Germain Forestier, Anthony Bagnall, Abdullah Mueen, and Eamonn Keogh. Optimizing Dynamic Time Warping's window width for time series data mining applications. *Data Mining and Knowledge Discovery*, 2018.

[31] Ayhan Demiriz, Kristin P Bennett, and Mark J Embrechts. Semi-supervised clustering using genetic algorithms. *Art. Neural Net. Eng.*, pages 809–814, 1999.

[32] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[33] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 2013.

[34] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: Experimental comparison of representations and distance measures. *Proc. of the VLDB Endowment*, 1(2):1542–1552, 2008.

[35] Rui Ding, Qiang Wang, Yingnong Dang, Qiang Fu, Haidong Zhang, and Dongmei Zhang. YADING : Fast clustering of large-scale time series data. *VLDB Endowment*, 8(5):473–484, 2015.

[36] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional GANs. *arXiv preprint arXiv:1706.02633*, 2017.

[37] Leonardo N. Ferreira and Liang Zhao. Time series clustering via community detection in networks. *Information Sciences*, 326:227–242, 2016.

[38] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[39] Germain Forestier, Francois Petitjean, Hoang Anh Dau, Geoffrey I. Webb, and Eamonn Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 865–870, 2017.

[40] Ian Fox, Lynn Ang, Mamta Jaiswal, Rodica Pop-Busui, and Jenna Wiens. Contextual motifs: Increasing the utility of motifs using contextual data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM, 2017.

[41] William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57–57, 1992.

[42] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

[43] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[44] Salvador Garcia and Francisco Herrera. An extension on"statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(Dec):2677–2694, 2008.

[45] Zoltan Geler, Vladimir Kurbalija, Miloš Radovanović, and Mirjana Ivanović. 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*, pages 105–114. Springer, 2014.

[46] Shaghayegh Gharghabi, Shima Imani, Anthony Bagnall, Amirali Darvishzadeh, and Eamonn Keogh. Matrix Profile XII: MPdist: a novel time series distance measure to allow data mining in more challenging scenarios. In *To Appear in ICDM 2018*, 2018.

[47] Tomasz Górecki and MacIej Łuczak. Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, 26(2):310–331, 2013.

[48] Tomasz Górecki and Maciej Łuczak. Non-isometric transforms in time series classification using DTW. *Knowledge-Based Systems*, 61:98–108, 2014.

[49] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. Data augmentation for time series classification using convolutional neural networks. *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[50] Mathieu Guillame-Bert and Artur Dubrawski. Classification of time sequences using graphs of temporal constraints. *Journal of Machine Learning Research*, 18:1–34, 2017.

[51] Jože Guna, Iztok Humar, and Matevž Pogačnik. Intuitive gesture based user identification system. In *2012 35th International Conference on Telecommunications and Signal Processing, TSP 2012 - Proceedings*, pages 629–633, 2012.

[52] Thien M Ha and Horst Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 535–539, 1997.

[53] Yuan Hao, Mohammad Shokoohi-Yekta, George Papageorgiou, and Eamonn Keogh. Parameter-free audio motif discovery in large data archives. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2013.

[54] Akira Hayashi, Yuko Mizuhara, and Nobuo Suematsu. Embedding time series data for classification. *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 356—-365, 2005.

[55] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1322–1328, 2008.

[56] Yuanduo He, Jialiang Pei, Xu Chu, Yasha Wang, Zhu Jin, and Guangju Peng. Time series classification via manifold partition learning. In *To Appear in ICDM 2018*, 2018.

[57] Jon Hills, Jason Lines, Edgaras Baranauskas, James Mapp, and Anthony Bagnall. Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, 28(4):851–881, 2014.

[58] Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*, volume 751. John Wiley & Sons, 2013.

[59] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[60] Bing Hu, Yanping Chen, and Eamonn Keogh. Classification of streaming time series under more realistic assumptions. *Data Mining and Knowledge Discovery*, 30(2):403–437, 2016.

[61] Bing Hu, Thanawin Rakthanmanon, Yuan Hao, Scott Evans, Stefano Lonardi, and Eamonn Keogh. Using the minimum description length to discover the intrinsic cardinality and dimensionality of time series. *Data Mining and Knowledge Discovery*, 29(2):358–399, 2014.

[62] Yun-Wu Huang and Philip S Yu. Adaptive query processing for time-series data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 282–286. ACM, 1999.

[63] Havskov Jens and Gerardo Alguacil. Instrumentation in earthquake seismology: Modern approaches in geophysics, 2004.

[64] Young-Seon Jeong, Myong K Jeong, and Olufemi A Omitaomu. Weighted Dynamic Time Warping for time series classification. *Pattern Recognition*, 44:2231–2240, 2011.

[65] Rohit J. Kate. Using Dynamic Time Warping distances as features for improved time series classification. *Data Mining and Knowledge Discovery*, 30(2):283–312, 2015.

[66] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[67] Eamonn Keogh and T Folias. The UCR Time Series Data Mining Archive. `http://www.cs.ucr.edu/~eamonn/TSDMA/index.html`, 2002.

[68] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Mining and knowledge discovery*, 7(4):349–371, 2003.

[69] Edward D Kim, Joyce M W Lam, and Jiawei Han. Aim: Approximate intelligent matching for time series data. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 347–357. Springer, 2000.

[70] JOJWGSJ Kubanek, K J Miller, J G Ojemann, J R Wolpaw, and G Schalk. Decoding flexion of individual fingers using electrocorticographic signals in humans. *Journal of neural engineering*, 6(6):66001, 2009.

[71] Vladimir Kurbalija, Miloš Radovanović, Zoltan Geler, and Mirjana Ivanović. The influence of global constraints on similarity measures for time-series databases. *Knowledge-Based Systems*, 56:49–67, 2014.

[72] Daoyuan Li, Tegawende F Bissyande, Jacques Klein, and Yves Le Traon. Time series classification with discrete wavelet transformed data. *International Journal of Software Engineering and Knowledge Engineering*, 26(09n10):1361–1377, 2016.

[73] Daoyuan Li, Tegawendé François D Assise Bissyande, Jacques Klein, and Yves Le Traon. Time series classification with discrete wavelet transformed data: Insights from an empirical study. In *The 28th International Conference on Software Engineering and Knowledge Engineering (SEKE 2016)*, 2016.

[74] Lei Li and B Aditya Prakash. Time series clustering: Complex is simpler! In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 185–192, 2011.

[75] Moshe Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml/index.php`, 2013.

[76] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*, pages 53–68, 2002.

[77] Jason Lines and Anthony Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 29(3):565–592, 2015.

[78] Jason Lines, Sarah Taylor, and Anthony Bagnall. HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1041–1046. IEEE, 2016.

[79] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(5):52, 2018.

[80] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[81] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[82] Xitong Liu, Hui Fang, and Deng Cai. Towards less biased web search. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 373–376. ACM, 2015.

[83] Sha Lu, Gordana Mirchevska, Sayali S. Phatak, Dongmei Li, Janos Luka, Richard A. Calderone, and William A. Fonzi. Dynamic Time Warping assessment of high-resolution melt curves provides a robust metric for fungal identification. *PLoS ONE*, 12(3), 2017.

[84] Yuanhua Lv and ChengXiang Zhai. 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*, page 579, 2010.

[85] Jacob Masters. The level of pain & injury from slip and fall accidents. `http://www.bisociety.org/level-pain-injury-slip-fall-accidents/`, may 2016.

[86] Antigoni Mezari and Ilias Maglogiannis. Gesture recognition using symbolic aggregate approximation and Dynamic Time Warping on motion data. In *Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 342–347. ACM, 2017.

[87] Abdullah Mueen and Eamonn Keogh. Online discovery and maintenance of time series motifs. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1089–1098. ACM, 2010.

[88] Abdullah Mueen, Eamonn Keogh, Qiang Zhu, Sydney Cash, and Brandon Westover. Exact discovery of time series motifs. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 473–484. SIAM, 2009.

[89] David Murray, Jing Liao, Lina Stankovic, Vladimir Stankovic, Charlie Wilson, Michael Coleman, and Tom Kane. A data management platform for personalised real-time energy feedback. *Eedal*, pages 1–15, 2015.

[90] Andrew Y. Ng. Preventing "overfitting" of cross-validation data. *In ICML*, 97:245–253, 1997.

[91] U M Okeh and C N Okoro. Evaluating measures of indicators of diagnostic test performance: Fundamental meanings and formulars. *J Biom Biostat*, 3(1):2, 2012.

[92] National Council on Aging. Fall prevention facts. `https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/`, 2016.

[93] John Paparrizos and Luis Gravano. k-Shape: Efficient and accurate clustering of time series. *Acm Sigmod*, pages 1855–1870, 2015.

[94] John Paparrizos and Luis Gravano. Fast and accurate time-series clustering. *ACM Transactions on Database Systems*, 42(2):1–49, 2017.

[95] François Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping Chen, and Eamonn Keogh. Dynamic Time Warping averaging of time series allows faster and more accurate classification. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 470–479. IEEE, 2014.

[96] Simona Petrutiu, Alan V Sahakian, and Steven Swiryn. Abrupt changes in fibrillatory wave characteristics at the termination of paroxysmal atrial fibrillation in humans. *Europace*, 9(7):466–470, 2007.

[97] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 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*, page 262, 2012.

[98] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Addressing big data time series: Mining trillions of time series subsequences under Dynamic Time Warping. *Transactions on Knowledge Discovery from Data (TKDD*, 2013.

[99] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[100] Sangeeta Rani and Geeta Sikka. Recent techniques of clustering of time series data: A survey. *International Journal of Computer Applications*, 52(15):1–9, 2012.

[101] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 506–510. SIAM, 2005.

[102] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496, 2014.

[103] Naoki Saito and Ronald R Coifman. Local discriminant bases and their applications. *Journal of Mathematical Imaging and Vision*, 5(4):337–358, 1995.

[104] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[105] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[106] Suchi Saria, Andrew Duchi, and Daphne Koller. Discovering deformable motifs in continuous time series data. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, page 1465, 2011.

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

[108] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. 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*, pages 289–297. SIAM, 2015.

[109] Yutao Shou, Nikos Mamoulis, and David Cheung. Fast and exact warping of time series using adaptive segmental approximations. *Machine Learning*, 58(2-3):231–267, 2005.

[110] Sidney Siegal. *Nonparametric Statistics for the Behavioral Sciences.* McGraw-hill, 1956.

[111] Angelo Silva and Renato Ishii. A new time series classification approach based on recurrence quantification analysis and Gabor filter. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pages 955–957. ACM, 2016.

[112] Diego F. Silva, Gustavo E.A.P.A. Batista, and Eamonn Keogh. Prefix and suffix invariant Dynamic Time Warping. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 1209–1214, 2017.

[113] Diego F Silva, Rafael Giusti, Eamonn Keogh, and Gustavo EAPA Batista. Speeding up similarity search under Dynamic Time Warping by pruning unpromising alignments. *Data Mining and Knowledge Discovery*, pages 1–29, 2018.

[114] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[115] Sungho Tak and Jong Chul Ye. Statistical analysis of fnirs data: a comprehensive review. *Neuroimage*, 85:72–91, 2014.

[116] Mariem Taktak, Slim Triki, and Anas Kamoun. SAX-based representation with longest common subsequence dissimilarity measure for time series data classification. In *Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on*, pages 821–828. IEEE, 2017.

[117] Chang Wei Tan, Matthieu Herrmann, Germain Forestier, Geoffrey I. Webb, and Francois Petitjean. Efficient search of the best warping window for Dynamic Time Warping. *Proceedings of the 2018 SIAM International Conference on Data Mining*, 2018.

[118] Anastasios Tombros and Mark Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 2–10. ACM, 1998.

[119] Selen Uguroglu. *Robust learning with highly skewed category distributions.* PhD thesis, PhD thesis, Carnegie Mellon University, 2013.

[120] Angelos Valsamis, Konstantinos Tserpes, Dimitrios Zissis, Dimosthenis Anagnostopoulos, and Theodora Varvarigou. Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction. *Journal of Systems and Software*, 127:249–257, 2017.

[121] Jose R Villar, Paula Vergara, Manuel Menéndez, Enrique de la Cal, Víctor M González, and Javier Sedano. Generalized models for the classification of abnormal movements in daily life and its applicability to epilepsy convulsion recognition. *International journal of neural systems*, 26(06):1650037, 2016.

[122] Nguyen Xuan Vinh. Information theoretic measures for clusterings comparison : Variants , properties , normalization and correction for chance. *Journal of Machine Learning Research*, 11:2837–2854, 2010.

[123] Ulrike Von Luxburg. Clustering stability: An overview. *Foundations and Trends® in Machine Learning*, 2(3):235–274, 2010.

[124] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints, 2000.

[125] R W Wall. Simple methods for detecting zero crossing. In *Industrial Electronics Society, 2003. IECON'03. The 29th Annual Conference of the IEEE*, volume 3, pages 2477–2481. IEEE, 2003.

[126] Coralyn W Whitney, Daniel J Gottlieb, Susan Redline, Robert G Norman, Russell R Dodge, Eyal Shahar, Susan Surovec, and F Javier Nieto. Reliability of scoring respiratory disturbance indices and sleep staging. *Sleep*, 21(7):749–757, 1998.

[127] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[128] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 1033–1040, 2006.

[129] Dragomir Yankov, Eamonn Keogh, Jose Medina, Bill Chiu, and Victor Zordan. Detecting time series motifs under uniform scaling. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 844–853. ACM, 2007.

[130] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix Profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1317–1322. IEEE, 2016.

[131] Jesin Zakaria, Abdullah Mueen, and Eamonn Keogh. Clustering time series using unsupervised-shapelets. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 785–794, 2012.

[132] Hui Zhang and Tu Bao Ho. Finding the clustering consensus of time series with multi-scale transform. In *Soft Computing as Transdisciplinary Science and Technology*, pages 1081–1090. Springer, 2005.

[133] Hui Zhang, Tu Bao Ho, Mao Song Lin, and Wei Huang. Combining the global and partial information for distance-based time series classification and clustering. *JACIII*, 10(1):69–76, 2006.

[134] Yangxin Zhong, Shixia Liu, Xiting Wang, Jiannan Xiao, and Yangqiu Song. Tracking idea flows between social groups. In *AAAI*, pages 1436–1443, 2016.

[135] Jing Zhou, Shan Feng Zhu, Xiaodi Huang, and Yanchun Zhang. Enhancing time series clustering by incorporating multiple distance measures with semi-supervised learning. *Journal of Computer Science and Technology*, 30(4):859–873, 2015.

[136] Yan Zhu, Zachary Zimmerman, Nader Shakibay Senobari, Chin-Chia Michael Yeh, Gareth Funning, Abdullah Mueen, Philip Brisk, and Eamonn Keogh. Matrix Profile

II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 739–748. IEEE, 2016.