Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm

François I Petitjean, Germain Forestier, Geoffrey I Webb, Ann E Nicholson, Yanping E Chen, Eamonn E Keogh

Faster and More Accurate Classification of Time Series by Exploiting a Novel Dynamic Time Warping Averaging Algorithm

François Petitjean[1], Germain Forestier[2], Geoffrey I. Webb[1],
Ann E. Nicholson[1], Yanping Chen[3] and Eamonn Keogh[3]

[1] Faculty of IT, Monash University, Melbourne, Australia, firstname.lastname@monash.edu
[2] MIPS (EA 2332), Université de Haute Alsace, Mulhouse, France, germain.forestier@uha.fr
[3] Computer Science and Engineering Dpt, University of California, Riverside, USA {ychen053,eamonn}@cs.ucr.edu

***Abstract*—A concerted research effort over the past two decades has heralded significant improvements in both the efficiency and effectiveness of time series classification. The consensus that has emerged in the community is that the best solution is a surprisingly simple one. In virtually all domains, the most accurate classifier is the Nearest Neighbor algorithm with Dynamic Time Warping as the distance measure. The time-complexity of Dynamic Time Warping means that successful deployments on resource constrained devices remain elusive. Moreover, the recent explosion of interest in wearable computing devices, which typically have limited computational resources, has greatly increased the need for very efficient classification algorithms. A classic technique to obtain the benefits of the Nearest Neighbor algorithm, without inheriting its undesirable time and space complexity, is to use the Nearest Centroid algorithm. Unfortunately, the unique properties of (most) time series data mean that the centroid typically does not resemble *any* of the instances, an unintuitive and underappreciated fact. In this paper we demonstrate that we can exploit a recent result by Petitjean *et al.* to allow meaningful averaging of "warped" time series, which then allows us to create super-efficient Nearest "Centroid" classifiers that are at least as accurate as their more computationally challenged Nearest Neighbor relatives. We demonstrate empirically the utility of our approach by comparing it to all the appropriate strawmen algorithms on the ubiquitous UCR Benchmarks, and with a case study in supporting insect classification on resource constrained sensors.**

***Keywords*—time series, averaging, dynamic time warping, classification, data mining**

I. INTRODUCTION

The last decade has seen increasing acceptance that the Nearest Neighbor (NN) algorithm with Dynamic Time Warping (DTW) as the distance measure is *the* technique of choice for most time series classification problems. The NN-DTW algorithm has been shown to be competitive or superior in domains as diverse as pen-based computing, gesture recognition, robotics and ECG classification. Moreover recent comprehensive studies have strongly validated this idea:

- In a near exhaustive empirical study in [1] the authors compared NN-DTW to nearly all of the most highly cited distance measures in the literature on dozens of datasets. They found that no distance measure consistently beats DTW, but DTW almost always outperforms most methods that were originally touted as superior, based on less complete empirical evaluations.
- In [2] (and to a lesser extent [3]) the authors examine the assumption that the Nearest Neighbor classifier is the best technique and consider other classifiers, including kernel methods, neural networks and decision trees. Once again, the evidence strongly suggests that the structure of time series (autocorrelated values, high apparent but low intrinsic dimensionality) lends itself to the Nearest Neighbor algorithm and to NN-DTW in particular.

Because of these findings, most recent research has simply assumed the utility of NN-DTW and concentrated on mitigating the oft-lamented drawback of DTW: its time complexity. There has also been recent significant progress on this, such as Rakthanmanon *et al.*'s result showing that, under reasonable constraints, nearest neighbor queries under DTW can be answered in time that is no worse than twice that of the Euclidean distance [4].

Nevertheless there are still situations where DTW (or for that matter, Euclidean distance) has severe tractability issues. While the accuracy of NN is a function of the size of the training set, unlike eager learners, the classification *time* is also a function of it. Thus, to obtain a required level of accuracy, it may be necessary to compare the incoming exemplar to dozens or hundreds of training objects. While optimizations such as those in [4-6] can mitigate somewhat the time needed, NN-DTW may still be intractable in some situations. This is especially true for resource constrained devices such as wearable computers and embedded medical devices.

One obvious fix is to reduce the size of the training set to the largest size that can be searched at each time interval. Xi et al. [3] showed that by adapting classic data editing techniques it is possible to create a "smart" subset that has an error-rate as low as a much larger random subset. Nevertheless, this result only partly mitigates the problem.

The Nearest Centroid Classifier (NCC) is an apparent solution to this problem. NCC allows us to leverage the strengths of the NN algorithm, while avoiding its substantial space and time requirements. Unfortunately, the centroid is defined only for simple *metrics*, which DTW is not. This is not a trivial semantic point. As Figure 1 shows, even if we consider only instances that have a very low mutual DTW distance, if we attempt standard Euclidean averaging, the resultant centroid will typically resemble *none* of the parent objects.

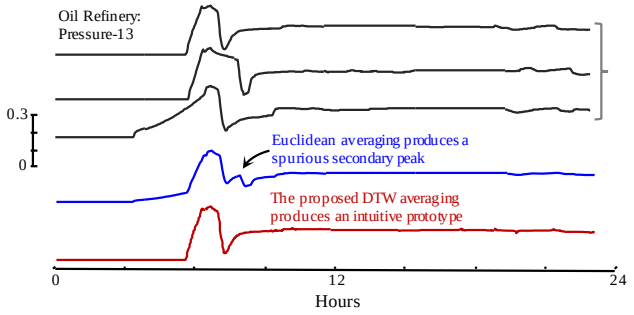


Figure 1: top) Three examples of daily patterns at an oil refinery [7]. middle) When averaged under the Euclidean distance the resulting centroid has an additional peak that is in none of the original time series. bottom) When averaged using the DTW based method proposed in this work, the "centroid" is more intuitive.

The contributions of this paper are as follows:

1. We leverage off and extend a little known recent result that allows us to meaningfully define "centroid" under DTW [8].
2. We show how to use this result to condense large datasets into much smaller (as small as a single instance per class) datasets.
3. We demonstrate that by carefully condensing the dataset, we can build a classification model using centroids that favorably competes with the state-of-the-art NN-DTW, but can classify time series up to 100x faster, which makes it compatible with real-time classification and embedded systems.
4. We show that, in some domains, the condensed datasets allows us to derive a classifier with *greater* accuracy. This less intuitive result arises because the averaging combines evidence from all exemplars to produce prototypes that are more like the class' Platonic ideal than any individual instance.

The remainder of this paper is organized as follows. First, we review related and background work. Then in Section III we introduce the necessary definitions and formally define the problem to be solved, before presenting our solution in Section IV. In Section V we provide strong empirical validation of our claims regarding both accuracy and computational time, then offer conclusions and directions for future work in Section VI.

Note that this paper is an extended version of our paper appearing in IEEE ICDM'14 as [9].

II. Related Work and Background

The idea that the *mean* of a set of objects may be more representative than any *individual* object from that set dates back at least a century to a famous observation of Francis Galton. Galton noted that the crowd at a county fair accurately guessed the weight of an ox when their individual guesses were averaged [10]. Galton realized that the *average* was closer to the ox's true weight than the estimates of most crowd members, and also much closer than any of the separate estimates made by cattle experts. This idea is frequently exploited in machine learning.

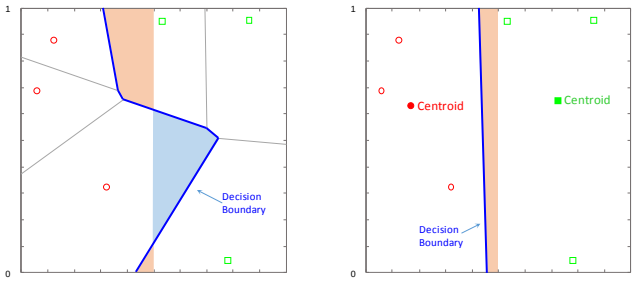


Figure 2: A simple classification problem in which the concept is the left vs. right side of the unit square. This instance of the problem has three points per class (left). Here NN has error-rate of 12.60%, while the Nearest Centroid classifier (right) with the same instances achieves an error-rate of just 5.22%.

For example the Nearest *Centroid* Classifier [11,12,13] generalizes the Nearest *Neighbor* classifier (NN) by replacing the set of neighbors with their centroid. It uses the center of mass of each class as the prototype against which every test instance is compared.

It should be noted that there are two separate motivations for using the Nearest Centroid Classifier. Most obviously it is *faster*, being O(1) rather than O(n).

Because this may be counterintuitive, we will demonstrate it in an intuitive setting. Consider a domain in which all exemplars are uniformly distributed in the unit square, with objects having an X-value less than 0.5 assigned the label **A**, otherwise **B**.

Figure 2 illustrates an example in which there are just three instances per class.

For balanced dataset sizes from 2 to 4,000, we compared the error rates of the NN and the NCC on this domain, each time averaging over 1,000 runs. The results are shown in Figure 3; note that these results assume that the decision boundary is $x = 0.5$, that the samples for each class are uniformly sampled in both half-squares, and that the test samples are uniformly sampled in the unit square.

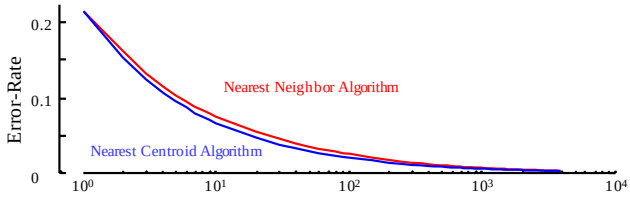


Figure 3: The error rate of two algorithms, NCC (blue) and NN (red) for increasingly large training data sizes of the "left vs. right side of the unit square" problem.

Even without any experiments we know that the two algorithms must agree on the far left side of the figure, and since the centroid of a single point *is* that point, the two algorithms are identical here. A little more introspection tells us that the algorithms will also agree on the far right side of the figure. What is less obvious is that the Nearest

Centroid Classifier is more accurate in between those two extremes. III.[1].

It is important to note that the Nearest Centroid Classifier is *not* guaranteed to be more accurate than the NN classifier in general. For example, consider the "Japanese flag" dataset (adapted from [14]) shown in Figure 4. Here the NN algorithm approaches zero error-rate for large training dataset sizes, while in contrast the Nearest Centroid Classifier steadfastly achieves just the default rate.

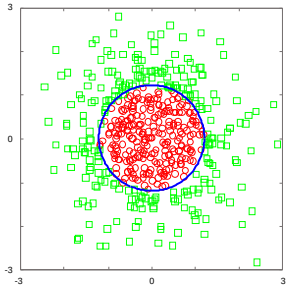


Figure 4: A two-class problem in which objects within 1.2 of the origin are in class **A**, otherwise they are in class **B**. With enough training data the NN classifier can learn this concept very well; however the Nearest Centroid Classifier is condemned to perform at the default rate.

In spite of the existence of such pathological cases, the Nearest Centroid Classifier often outperforms the NN algorithm on real datasets, especially if one is willing (as we are) to generalize it slightly; for example, by using *clustering* to allow a small number of centroids, rather than just one. Thus our claim is simply:

- Sometimes NCC has similar or lower accuracy than NN. In such cases we prefer NCC because it is faster and requires less memory.
- Sometimes NCC can be more accurate than NN. In such cases we prefer NCC because of the accuracy gains, and the reduced computational requirements come "for free".

The above discussion at first may appear to be moot for time series, because the concept of "centroid" for warped time series is ill-defined. It is the central contribution of this paper to show that we can take the "centroid" for warped time series in a principled manner that allows us to achieve both improvements in accuracy and reduced computational requirements at run time.

In the last decade the cognitive science community has presented strong evidence that the visual system's remarkable abilities stem, at least in part, from its ability to represent sets of objects by a "gist" or "ensemble"[2], which may be simply the *average* of the objects [17]. A recent paper notes that the major research direction of the cognitive science community is devoted simply to "*determining how these* (average) *representations are computed, why they are computed and where they are coded in the brain*" [18].

The difficulty faced by the cognitive scientists is similar to the pragmatic difficulty we face here. In some cases *averages* may be well defined, for example, the average height of Norwegian men. However, for some objects it is much less clear how to represent and compute averages. For example, computing an average *face* has been pursued since at least 1883 (again, Francis Galton, using composite photography) but significant progress has only been made in the last decade. Tellingly, this progress in face averaging was exploited to produce dramatic improvements in classification accuracy with a *Science* paper [19] boasting "*100% Accuracy in Automatic Face Recognition*" (this is the paper's actual title).

Compared to the complexity inherent in faces, time series might seem simple to average, however as Figure 1 hints at, the classic definition of centroid for time series usually produces a prototype which is not typical of the data.

IV. Averaging under Time Warping

We start by presenting the problem of creating average centroids that are consistent with the warping behavior of DTW. We then introduce DBA, which is the averaging method that will be used to derive our fast and accurate classifier in the next section.

For our problem, each object in the data set is a *time series*, which may be of different length.

***Definition* 1**: *Time Series.* A time series $T = (t_1, \ldots, t_L)$ is an ordered set of real values. The total number of real values is equal to the length of the time series (L). A dataset $\boldsymbol{D} = \{T_1, \ldots, T_N\}$ is a collection of N such time series.

A. Averaging under time warping – related work

Computational biologists have long known that averaging under time warping is a very complex problem, because it directly maps onto a multiple sequence alignment: the "*Holy Grail"* of computational biology [20]. Finding the multiple alignment of a set of sequences, or its average sequence (often called *consensus sequence* in biology) is a typical chicken-and-egg problem: knowing the average sequence provides a multiple alignment and vice versa. Finding the solution to the multiple alignment problem (and thus finding of an average sequence) has been shown to be NP-complete [21] with the exact solution requiring $O(L^N)$ operations for N sequences of length L. This is clearly not feasible with more than a dozen sequences (just 45 sequences of length 100 would require more operations than the number of particles in the universe).

Finding the average of a set is best seen as an optimization problem, as explained by the definition below.

[1] The source code proving the statistical significance is available at [15]; it performs two-tailed Bonferronni-Dunn test to compare pairs of methods NCC to NN [16].

[2] Note that the cognitive science use of "ensemble" is unrelated to the more familiar machine learning meaning.

***Definition* 2**: *Average object.* Given a set of objects $O = \{O_1, \ldots, O_N\}$ in a space E induced by a measure d, the average object $\bar{o}$ is the object that minimizes the sum of the squares to the set:

$$\arg\min_{\bar{o} \in E} \sum_{i=1}^{N} d^2(\bar{o}, O_i) \quad (1)$$

This definition demonstrates that finding the average of a set is intrinsically linked to the *measure* that is used to compare the data. This means that the average method has to be specifically designed for every measure that is used to compare data.

In our case, the objects are time series and the measure is DTW. We can thus now define what the average sequence should be, in order to be consistent with Dynamic Time Warping.

***Definition* 3**: *Average time series for DTW.* Given a set of time series $\boldsymbol{D} = \{T_1, \ldots, T_N\}$ in a space E induced by Dynamic Time Warping, the average time series $\bar{T}$ is the time series that minimizes:

$$\arg\min_{\bar{T} \in E} \sum_{i=1}^{N} \mathrm{DTW}^2(\bar{T}, T_i) \quad (2)$$

Many attempts at finding an averaging method for DTW have been made since the 1990s [22-25]. Researchers have exploited the idea that the *exact* average of two time series can be computed in $O(L^2)$. These papers have proposed different tournament schemes (the *guide trees* in computational biology) in which the sequences should be averaged first. Interestingly, none of these authors appear to have made the connection with the multiple sequence alignment problem; the most advanced method in 2009, PSA [24], heuristically averages the closest objects first, which corresponds to an idea proposed some 20 years earlier in computational biology [26].

There is a limit, however, to which the comparison between biological sequences and time series can be pushed. Ultimately, time series are sequences of *real-valued* numbers and not of *discrete* symbols like DNA/RNA sequences. While two genes coding for hemoglobin have almost certainly evolved from a common ancestor (although homoplasy can almost never be completely ruled out), no such lineage is present for time series. Nevertheless, we can sometimes imagine a domain in which there is an idealized Platonic prototype, of which we can only see corrupted (i.e. "warped") examples. In this view, DTW based averaging can be seen as an attempt to recover the "ancestor" state. For example, the ideal prototype may be an individual's *internal* (muscle memory) representation of her golf swing or her rendition of a song, of which we can only observe *external* performance approximations.

B. DBA: the best-so-far method to average time series for Dynamic Time Warping

DTW Barycenter Averaging (DBA), introduced in [8], exploits the parallels between time series and computational biology, while taking into account the unique properties of the former. We have shown in [8] that DBA outperforms all existing averaging techniques on all datasets of the UCR Archive [27] available at the time. In particular, it always obtained lower residuals (Equation 2) than the state-of-the-art methods with a typical margin of about 30%, making it the best method to date for time series averaging for DTW.

DBA iteratively refines an average sequence $\bar{T}$ and follows an expectation-maximization scheme:

1. Consider the average sequence $\bar{T}$ fixed and find the best multiple alignment[3] M of the set of sequences $\boldsymbol{D}$ with regard to $\bar{T}$, by individually aligning each sequence of $\boldsymbol{D}$ to $\bar{T}$.
2. Now consider M fixed and update $\bar{T}$ as the best average sequence consistent with M.

Table I gives the pseudocode for DBA.

Algorithm 1 simply finds the initial average sequence $\bar{T}$ and then refines it I times. The medoid sequence is usually a good candidate for the initialization of the algorithm. Note that if computation time is a concern, we have shown that randomly picking any sequence of the set usually gives good results also (see [8] – Section 4.5).

Algorithm 2 describes one iteration of DBA, i.e. one refinement of the current average sequence. Refining an average sequence is composed of two steps. First, every sequence S in $\boldsymbol{D}$**,** the set of sequences to average, is aligned to the to-be-refined average sequence $\overline{T_{init}}$. It is important to note that this process is performed independently for every sequence in the set, and thus does not use any order on the sequences, unlike other state-of-the-art methods. Next, the position of every element of the average sequence $\bar{T}(i)$ is set as the center of the elements of the sequences that had been associated to element i of $\overline{T_{init}}$. When the time series have only one dimension, this is simply performed with the arithmetic mean; for higher-dimensional sequences the position of every element can be updated as the barycenter of the set, i.e. using the arithmetic mean on each dimension separately [29].

Algorithm 3 simply computes DTW between the reference sequence and the set of sequences, and memorizes what elements of the sequences have been associated with each element of the reference sequence.

[3] It actually finds the *compact* multiple alignment [28].

TABLE I. GENERAL ALGORITHM FOR DBA

Algorithm 1. DBA($\boldsymbol{D}$, I)

Require: $\boldsymbol{D}$: the set of sequences to average
Require: I: the number of iterations

```
1:  T̄ = medoid( D ) // get the medoid of the set of sequences D
2:  do I times T̄ = DBA_update( T̄ , D )
3:  return T̄
```

Algorithm 2. DBA_update($\overline{T_{init}}$, $\boldsymbol{D}$)

Require: $\overline{T_{init}}$: the average sequence to refine (of length L)
Require: $\boldsymbol{D}$: the set of sequences to average

```
1:  // Step #1: compute the multiple alignment for T̄_init
2:  alignment = [ ∅, ⋯ , ∅ ] // array of L empty sets
3:  for each S in D do
4:      alignment_for_S = DTW_multiple_alignment ( T̄_init , S )
5:      for i=1 to L do
6:          alignment[i] = alignment[i] ∪ alignment_for_S[i]
7:      done
8:  done
9:  // Step #2: compute the multiple alignment for the alignment
10: let T̄ be a sequence of length L
11: for i=1 to L do
12:     T̄(i) = mean( alignment[i] ) //arithmetic mean of the set
13: done
14: return T̄
```

Algorithm 3. DTW_multiple_alignment (S_{ref} , S)

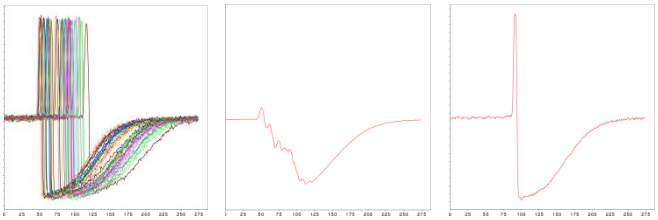
Require: S_{ref}: the sequence for which the alignment is computed
Require: S: the sequence to align to S_{ref} using DTW

```
1:  // Step #1: compute the accumulated cost matrix of DTW
2:  cost = DTWCumulMat( S_ref , S )
3:  // Step #2: store the elements associated with S_ref
4:  L = length( S_ref )
5:  alignment = [ ∅, ⋯ , ∅ ] // array of L empty sets
6:  i = rows( cost ) // i iterates over the elements of S_ref
7:  j = columns( cost ) //j iterates over the elements of S
8:  while (i > 1) && (j > 1) do
9:      alignment[i] = alignment[i] ∪ S(j)
10:     if i == 1 then j = j − 1
11:     else if j == 1 then i = i − 1
12:     else
13:         score = min( cost[i-1][j-1] , cost[i][j-1] , cost[i-1][j] )
14:         if score = = cost[i-1][j-1] then
15:             i = i − 1
16:             j = j − 1
17:         else if score = = cost[i-1][j] then i = i − 1
18:         else j = j − 1
19:         end if
20:     end if
21: done
22: return alignment
```

Algorithm 4. Medoid($\boldsymbol{D}$)

Require: $\boldsymbol{D}$: the set of sequences to find the medoid from

```
1:  minSS = +∞ //minimum sum of squares
2:  for each S_1 in D do
3:      // computing the sum of squares for S_1
4:      tmpSS=0
5:      for each S_2 in D do
6:          tmpSS+= (DTW(S_1,S_2))^2
7:      done
8:      if tmpSS < minSS then
9:          medoid = S_1
10:         minSS = tmpSS
11:     end if
12: done
13: return medoid
```



(a) (b) (c)

Figure 5: Visual comparison of the Euclidean average to our proposed DBA approach. This example highlights the inability of the Euclidean average to preserve the shape of warped time series. (a) One class of the Trace dataset [22]. (b) The average time series produced by the Euclidean average. (c) The average time series produced by DBA.

Note that an implementation of this pseudocode in Matlab and Java is available at [30].

Figure 5 shows that DBA can preserve the shape of warped time series while the traditional Euclidean average provides a prototype that does not resemble any of the time series of the set. Note also that in Figure 1 we showed a similar example of the algorithm's superior output on three examples of a pattern associated with an oil refinery process. This visual comparison is, however, only qualitative. Next we demonstrate the *quantitative* superiority of DBA over other techniques, i.e. its ability to minimize the sum of the residuals expressed in Equation 2.

Our previous work in [8] demonstrated the superiority of DBA over state-of-the-art techniques (Non-Linear Alignment and Averaging Filters – NLAAF – and Prioritized Shape Averaging - PSA) using the UCR archive, then composed of 20 datasets. We complement this evaluation with Appendix B, which quantitatively assesses DBA against both the medoid sequence and the Euclidean average – which had not been included in [8] – over all 44 datasets of the UCR archive [27]. These results show that DBA outperforms by far these two methods on all the datasets in the archive.

In addition, this paper also extends the definition of DBA by providing a proof of its convergence for l2-norm, i.e., that the sum of the squares (Equation 2) always

decreases between two iterations (or refinements). This proof is provided in Appendix A.

V. Our Fast and Accurate Centroid-based Classifier

In recent years there has been an increasing interest in using anytime algorithms for data mining [3,31,39]. However the variant known as *contract algorithms* have received less attention. Contract algorithms are a special type of anytime algorithms that require the amount of run-time to be determined prior to their activation. In other words, contract algorithms offer the anytime tradeoff between computation time and quality of results, but they are not interruptible.

Problem Statement *Contract Time Series Classification*: Given (1) a large time series training dataset, (2) an upper bound on the amount of computational resources that may be consumed for classification, and (3) no limits on the computational resources that may be consumed for training, produce the most accurate classifier possible.

We assume that the computational resource constraint will be *time*, not *space*, and that it will be given to us in the form of the number of CPU cycles available each second. For ease of exposition we assume that the constraint will be given as a positive integer *C*, which is the number of exemplars per class that we can examine when asked to classify a new object. Figure 6 illustrates this problem statement.

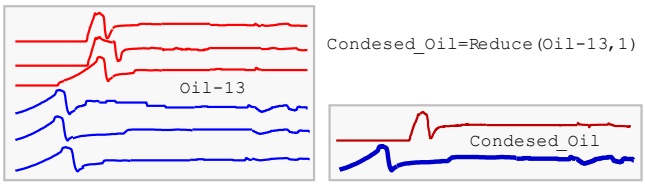


Figure 6: A visual intuition of an instance of our problem statement: Given the Oil-13 time series training dataset (*left*), and a user constraint *C*, here '1'. Produce a new dataset with *C* items per class (*right*), such that the accuracy on future data is maximized.

As we explained in the introduction, based on the consensus of the literature and our own experiments, we believe that the best solution will be a variant of Nearest Neighbor classification. While decision trees and Bayesian classifiers are very efficient, and although time series classification is an active and competitive research area, no competitively accurate classifiers for time series based on these methods have been produced [2,3].

What then is the space of techniques we can explore? After exhausting all known optimization techniques (early abandoning, removing the unnecessary square root calculation, lower bounding, etc.) we can consider manipulating the following:

- Reducing the data cardinality, and doing NN-DTW on the reduced cardinality data. While classification on suitable reduced cardinality data has little effect on accuracy [32], it only helps scalability on specialized hardware. We are aiming for a general solution.
- Reducing the data dimensionality, and doing NN-DTW on the reduced dimensionality data. This idea has been in the literature for at least two decades, and seems to have been rediscovered many times. The idea works well when the raw data is oversampled. For example, some bedside machines report electrocardiograms at up to 4,096Hz, yet there is little evidence that anything above 256Hz is needed for classification. However here we assume that the data we are given is sampled at an appropriate rate.
- Reducing the number of objects the nearest neighbor algorithm must see. This can be done by selecting a subset of the data (which is known as *data editing* or *condensing*) or aggregating the data.

As the reader will have intuited by now, it is the last idea we intend to pursue. There are several obvious ways to reduce the number of objects the nearest neighbor algorithm must see, and several variants of intelligent data *editing* have been proposed [3]. However to the best of our knowledge no one has consistently considered data *aggregation* for NN-DTW. When it has been considered, the artifacts produced by averaging methods for Dynamic Time Warping, such as the one hinted at in Figure 1 and acknowledged in the literature by [8,33,34], suggests that this is an unpromising avenue to explore.

Conversely, as noted above, aggregation methods (including, but not limited to the Nearest Centroid Classifier) have certain properties that seem very desirable. In particular, they provide a condensed model of the aggregated set, allowing speed up, and they weight information from every training instance, potentially improving accuracy. However, as we explain in the next paragraph, simply averaging all the objects in each class is unlikely to work well in most domains, and this motivates a clustering-based data condensing approach.

While it is possible that for some datasets, a single prototype may capture the "essence" of a class, for other datasets it may require a small number of prototypes. Moreover, a single dataset may exhibit both possibilities on a class-by-class basis. For example, for the "Japanese flag" dataset shown in Figure 4, a single centroid is clearly optimal for the circle/red class, but we would need, say eight suitably arranged examples from the green/square class arranged in an octagon to carve out a decision boundary that approximates the true circular decision boundary. To give a more concrete example, consider the case study in insect surveillance we explore in Section VI.A, which appears to be a single class, *Culex stigmatosoma*, the mosquito that spreads West Nile virus. However, this insect, like most mosquitoes, is highly sexually dimorphic. If we try to create a *single* template to represent both males and females we are condemned to have a template that represents neither. However, by

clustering each individual class, we hope to be able to account for any natural polymorphism within the class. In Table II we present the algorithm for such a clustering-based approach to condensing a dataset.

TABLE II. ALGORITHM TO CONDENSE TRAINING DATASET

Algorithm 5. Reduce(*Data, C*)

Require: *Data*: dataset; *C*: The number of exemplars per class

```
1:  // partition the data into C sets of time series
2:  Clusters = do_clustering(Data,C) //for example with K-means
3:  Condensed_Data = ()
4:  for each Cluster in Clusters do
5:          Condensed_Data.add(DBA(Cluster,15))
6:  done
7:  return Condensed_Data
```

It is important to note, however, that we see our main contribution as proposing a *warping-invariant-averaging* based condensation framework, of which Algorithm 4 given in Table II is simply one concrete and straightforward *partitional clustering* example. To further reinforce this notion in our experimental section, we also consider a warping-invariant-averaging *hierarchical clustering* based condensation framework.

VI. EXPERIMENTAL EVALUATION

In this section, we assess the performance of our averaging-based reduction methods for time series classification, over the state-of-the-art data condensing methods (which do *not* average time series). Note that the distance measure used for all experiments is DTW.

We compare the following algorithms; the last two of which exploit our averaging technique:

- **Random Selection**: Here we randomly sample the training data, selecting as many samples as we can use under the contract time.
- **Drop{*X*}**: There has been significant work on data editing (numerosity reduction/condensing) for nearest neighbor classification [35]. All these algorithms create some list of nearest neighbors, of both the same class (associates) and of different classes (enemies), and use a weighted scoring function based on this list to determine the worst exemplar. We compare to three variants; Drop1, Drop2 and Drop3, see [35] for full details on their subtle differences.
- **Simple Rank (SR)**: This method gives to each instance a rank according to its contribution to the classification [36]. A leave-one-out 1-NN classification is performed on the training set, and the rank of the instance is calculated as the following formula:

$$rank(x) = \sum_i \begin{Bmatrix} 1 \; if \; class(x) = class(x_i) \\ -2/(\#classes - 1) \; otherwise \end{Bmatrix}$$

where x_i are associates of x. The ties are broken by sorting the instances according to their distance to their nearest "enemy" (standard terminology).

- **K-Medoids**: This well-known method, also known as "partitioning around medoids", aims at minimizing the intra-cluster sum of squares, by using the proximity of objects to the *medoids* of the clusters formed by the algorithm. Note that the medoid of a set is the object from the set itself, that minimizes the sum of the squares (same objective as Equation 2, with the additional condition that $\bar{T} \in \boldsymbol{D}$). K-medoid thus does not use any average object.

And finally, two methods which instantiate our averaging-based condensing framework:

- **K-Means**: Similar to K-medoids, this well-known method aims at minimizing the intra-cluster sum of squares. The clusters are formed by using the proximity of objects to the *average objects* (or centroids) of the different clusters. We use DBA to perform averaging.
- **AHC with Ward's criterion**: Starting with every object in its own cluster, agglomerative hierarchical clustering (AHC) progressively merges the most similar clusters until all the objects are part of the same cluster. Similar to K-means and K-medoids in its objective, the Ward's criterion ranks the pairs of clusters with regard to the increase in the weighted intra-cluster sum of squares. Here again we use DBA to perform averaging.

We consider situations where we can only visit a small handful of exemplars, as few as just one per class, as this is the defining characteristic of our problem setting. In any case, we expect (and empirically demonstrate) that all algorithms converge as we allow the size of the reduced dataset used to increase. That is to say, if we randomly sample as many time series as there are in the training set, we actually obtain the full training set, which is logically equivalent to the 1-NN classifier. The behavior is similar for the other techniques: the reduced sets of time series all tend to the initial training set as their sizes increase.

Our experiments are divided into three parts:

A. We begin with a case study, to ground the utility of our ideas in the real world.

B. Having shown that average-based methods outperform sampling-based ones on our case study, we further assess the performance of the different methods on a full-scale experiment with 42 datasets. We demonstrate the clear superiority of average-based methods for condensing the model of the class into a handful of exemplars.

C. We show that not only do average-based methods provide better solutions than the state of the art for reducing the size of the training set, but also that they

make it possible to improve on the classification accuracy, compared to the full 1-NN classifier.

Note that the runtimes for the classification phase are directly proportional to the number of prototypes that are used, because the similarity measure is the same and has constant complexity with regard to the length. This means that for a given dataset, and a given number of prototypes per class, the classification time should be exactly the same regardless of the algorithm. Moreover, as explained in the problem statement, we are not interested in the training time.

A. *Case Study in Insect Surveillance*

Recent work has shown that it is possible to classify flying insects with high accuracy by converting the audio of their flight (i.e. the familiar "buzz" of bees) to an amplitude spectrum [37], which, as shown in Figure 7 can essentially be considered a "time series".

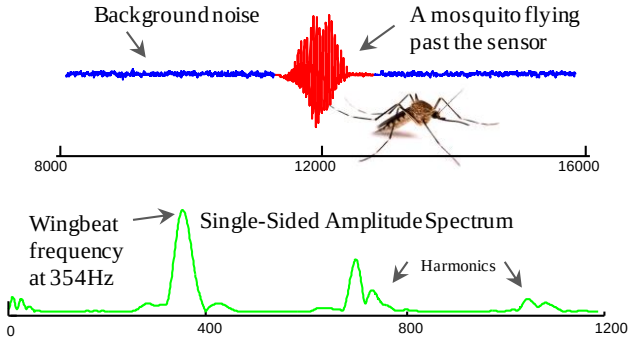


Figure 7: An audio snippet of an insect flight sound (*top*) can be converted into a pseudo time series (*bottom*) and used to allow classification

All previous work on insect classification had assumed that a single feature extracted from the amplitude spectrum, the *wingbeat frequency*, was the *only* useful feature in the amplitude spectrum. However [37] forcefully demonstrates that using the entire spectrum, and treating the problem as a time series classification problem, significantly reduces the error rate. In retrospect this is not surprising. A *G* note on a piano and an open string *G* note on a guitar have the same frequency of 196Hz (about the same frequency as a honey bee), but are easy to tell apart.

The ability to automatically classify insects has potential implications for agricultural and human health, as many plant/human diseases are vectored by insects. The promising results presented in [37] are demonstrated in the laboratory setting, and exploit large training datasets to archive high accuracy. However, field deployments must necessarily be on inexpensive resource-constrained hardware, which may not have the ability to allow nearest-neighbor search on large training datasets, up to hundreds of times a second. Thus we see this situation as an ideal application for our work.

We recorded the flying sound of male and female insects of the species *Culex stigmatosoma*, which is a vector of several diseases such as the West Nile Virus and Western Equine Encephalitis [38]. Being able to classify male vs. female mosquitoes is important because only the females actually spread disease, and different interventions are used to control females (to reduce biting *now*) and males (to reduce biting *one generation hence*).

Using our pseudo-acoustic sensor [31], we recorded about 10,000 flights and created a dataset by randomly choosing 200 examples of each class (male/female). We then randomly split this dataset into two balanced train/test datasets of same size.

As we can see in Figure 8, our algorithm is able to achieve a lower error-rate using just two items per class, than by using the *entire* training dataset. This is an astonishing result. The curves for the other approaches are more typical for data condensing techniques [3,35], where we expect to pay a cost (in accuracy) for the gains in speed.

The error rate for our approach is minimized at 19 items per class, suggesting we can benefit for some diversity in the training data. This diversity probably reflects the diversity of temperatures, as we record 24 hours a day over several days. However even if we kept just one pair of exemplars from each class, we would have an error-rate of just 0.13, which is still better than using all the data. These results are significant in this domain, where a low powered device may have to classify up to hundreds insects per second with limited computational resources.

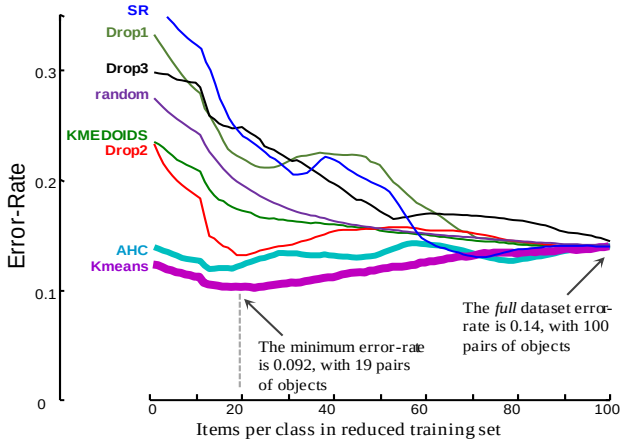


Figure 8: (best viewed in color) The error rate of various data condensing techniques for every output training size from 1 per class to 100 per class. The curves are slightly smoothed for visual clarity; the raw data spreadsheets are available at [15].

We now proceed with the rest of the experiments, in order to assess the generality of the two observations that we have made on this case study:

1. The average-based methods condense better the information about the class than the state-of-the-art methods (detailed in the next sub-section: B).
2. Not only are average-based methods better at reducing the size of the training set, but they can also improve the accuracy of the classifier. This

has been observed in Figure 8 where reducing the training set with the K-means algorithm allows us to derive a classifier that performs better than 1-NN using the full training set (error rate of 0.092 vs 0.14). This observation will be assessed in sub-section C.

Finally, note that all the raw material generated by our experiments (for example, the charts similar to Figure 8 for *all* the datasets, but also the rankings used in the reminder of this section) cannot be completely included in the paper due to space limitations; we provide detailed results for 3 more representative examples in Appendix C; other results are available at [15].

B. *Condensing the model of the class to a handful of exemplars*

To demonstrate that the results in the case study represent typical improvements over the rival methods, we will test on a very diverse collection of datasets. We have compared our approach on all the datasets in the UCR time series archive [27][4]. A description of a representative sample of these datasets is given in TABLE III.

TABLE III: PRESENTATION OF A SAMPLE OF THE DATASETS USED

Name	Length	Size train/test	# classes
Gun-Point	150	50/150	2
Swedish Leaf	128	500/625	15
TwoPatterns	128	1,000/5,000	4
FaceAll	131	560/1,690	14
Coffee	286	28/28	2
Haptics	1,092	155/308	5
Inline Skate	1,882	100/550	7
WordsSyn.	270	267/638	25

We want to compare the performance of the different methods when they are *authorized* (under the "contract") to use, say, 1 prototype per class (or $\#classes$ prototypes for Random, DropX and SimpleRank). To this end, we follow the standard practices for the statistical comparison of classifiers [16] and use the average ranking of each method over all the datasets. This will allow us to assess what algorithm exhibits, on average, the best classification performances under the *contract* restriction.

For every dataset and every algorithm, we compute the error-rate when constrained to use a reduced set of k prototypes per class only. Then, for every dataset, we rank the methods by error-rates: rank 1 is assigned to the best method; rank 8 is assigned to the worst one.[5]

We then compute the average rank for every method (see [34 – Section 3.2.2]). Let r_i^j be the rank of the j^{th} of A algorithms on the i^{th} of N_d datasets. The average rank for algorithm j is computed as $R_j = \frac{1}{N_d}\sum_i r_i^j$.

This gives a direct general assessment of all the algorithms: the lowest rank corresponds to the method that, on average, obtains the lowest error-rate for the considered "contract".

TABLE IV shows the average rank of all algorithms over the datasets of [27] (again, the raw results giving the error rate and rank for every method and every dataset is available at [15]). These results show unanimously that the methods that use an average sequence (K-means and AHC) significantly outperform the prior state of the art.

TABLE IV: AVERAGE RANKING OF THE CONDENSING METHODS FOR 1 TO 5 PROTOTYPES PER CLASS

Algorithm	Average rank R_j using k prototypes per class (or equivalent)				
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Random	4.70	5.06	4.81	5.46	5.01
Drop1	6.38	3.32	6.13	5.71	5.63
Drop2	5.37	5.37	5.32	5.14	5.20
Drop3	6.37	6.62	6.68	6.56	6.80
Simple rank	5.23	5.35	5.42	5.02	5.14
K-medoids	3.67	3.45	3.71	3.82	3.81
K-means	**2.14**	1.96	2.13	2.13	2.36
AHC	**2.14**	**1.92**	**1.98**	**2.08**	**2.13**
χ_F^2	141	166	149	135	128
R_{med}-R_{mean}	1.52	1.49	1.58	1.69	1.45

We first perform a Friedman test [16], in order to assess if the results are significantly different. This test is used to evaluate whether there is enough evidence to confidently state that the different methods have different mean ranks [16, Section 3.2.2]:

$$\chi_F^2 = \frac{12N_d}{A(A+1)}\left[\sum_j R_j^2 - \frac{A(A+1)^2}{4}\right] \quad (3)$$

where N_d is the number of datasets and A is the number of algorithms compared. The values are reported in the second-to-last line of TABLE IV; given that the Friedman test follows a χ^2 distribution with $A-1$ degrees of freedom, these results yield a highly significant difference between the methods ($p < 10^{-16}$).

Having rejected the null hypothesis, we can proceed with a detailed comparison of the methods. Again, we follow standard practices for classifier comparison [16] and perform a two-tailed Bonferronni-Dunn test to compare pairs of methods. Our aim is to show that using the average yields better performance for time series classification than alternative approaches to contract time series classification, rather than trying to establish the prevalence of any

[4] We use 42 datasets, i.e. all but two of the datasets of the archive; we have excluded the StarLightCurve and FetalECG for computational reasons.

[5] In case of ties, we assign the average (or fractional) ranking. For example, if there is one winner, two seconds and a loser [1,2,2,4], then the fractional ranking will be [1,2.5,2.5,4].

algorithm in particular. To this end, we compare K-means to K-medoids. This pair of methods constitutes an excellent test-bed, because K-medoids appears to be the best performing method in the group of methods that do *not* use the average time series, while K-means appears to be the "worst" performing method in the group of methods that do use the average time series. In addition, these two methods are functionally comparable, because they have the same objective function to minimize the intra-cluster sum of squares. In this way, we are comparing the methods in the least advantageous way for averaging-based methods, in order to be extra-conservative in the assessment of average-based methods vs. state-of-the-art methods. Comparing 8 methods over 42 datasets, [16] shows that, to be statistically significant ($\alpha = 0.05$) the critical difference (CD) between the average rankings has to be greater than:

$$\mathrm{CD} = q_{0.05} \cdot \sqrt{\frac{A(A+1)}{6N_d}} = 2.690 \cdot \sqrt{\frac{72}{252}} \approx 1.438.$$

We report the difference between the average rank obtained by K-medoids and the one obtained by K-means over the 42 datasets in the last line of TABLE IV. It shows that the difference is greater than the critical one CD, regardless of the number of prototypes used. As a result, we can confidently conclude that the K-means algorithm is statistically significantly better than K-medoids, and thus that the use of *averaging-based methods yield better results than state-of-the-art methods*.

C. Classifying faster and more accurately

We have seen in the case study on insect surveillance that average-based methods manage, with a reduced set of time series, to outperform the classification accuracy of the 1-NN classifier on the *full* training set. This result may be counterintuitive, so in this section we will assess this phenomenon on a wide variety of datasets.

To this end, we start by performing a standard 1-NN classifier using the full training set for classification. This gives us the reference error-rate against which we compare the results of different methods. We then progressively restrict the allowed size of the reduced set (k), until we find the smallest value of k for which the error-rate is smaller than the full 1-NN algorithm.

Then, for each dataset (and similar to the experiment in the last section), we rank the methods by size of their reduced sets that are able to "beat" the full 1-NN classifier. The results of these experiments are reported in TABLE V; note that for fairness in the ranking, we do not include the *Random sampling* strategy because, on average, it cannot beat the results of the full 1-NN classifier.

A first look at TABLE V shows that average-based methods again outperform the prior state of the art, with the K-means algorithm obtaining an average rank of 1.57 better than the K-medoids algorithm. Moreover, on average, the K-means method is able to condense the training set by 71%. This means that on average over the archive of datasets, our method using the K-means algorithm achieves equal or better performance that the full 1-NN classifier, while only requiring 29% of the computational complexity. Again, this is an extraordinary result.

TABLE V: AVERAGE RANKING OF THE CONDENSING METHODS ON THE SIZE OF THE DATASET REQUIRED TO BEAT THE FULL 1-NN CLASSIFIER

Algorithm	**Average rank R_j**	**Average size of the reduced set** (in % of the training set)
Drop1	5.89	86%
Drop2	5.07	76%
Drop3	5.45	80%
Simple rank	4.31	69%
K-medoids	3.41	52%
K-means	**1.84**	**29%**
AHC	2.73	39%

We can now assess the statistical significance of the superiority of K-means over K-medoids (the best method that does not average time series).

Similar to the last sub-section, we start by computing a Friedman test over the ranking presented in the first column of TABLE V, which yields a highly significant difference between the methods ($\chi_F^2 > 173$ which gives $p < 10^{-18}$).

We can thus proceed with a detailed assessment of the performance of K-means versus the reference K-medoids. The critical difference (CD) for this experiment is:

$$\mathrm{CD} = q_{0.05} \cdot \sqrt{\frac{A(A+1)}{6N_d}} = 2.638 \cdot \sqrt{\frac{56}{252}} \approx 1.244.$$

Moreover, we have:

$$\mathrm{R}_{\mathrm{KMedoids}} - \mathrm{R}_{\mathrm{KMeans}} \approx 1.571 > 1.244$$

As this difference is far greater than the critical value, we can conclude confidently that the *K-means algorithm requires significantly fewer prototypes than the K-medoids algorithm to "beat" the full 1-NN classifier*.

VII. DISCUSSION: WHY CAN WE GET BETTER RESULTS?

We have seen that our approach can provide more accurate predictions for several domains. We wish to complete the intuition that we provided at the start of the paper, with a few elements that can explain this improvement, not only in the speed of the classification, but also in terms of accuracy. We posit that two conjugate elements are responsible for the potential gain in accuracy:

1. Most datasets contain subclasses. Our condensing approach acts as a clustering of the data, which makes it possible to create different sub-models for the different subclasses. Such sub-classes are present in many applications, as it is for example the case for the *Gun Point* dataset, where each class has

recordings associated with people of different heights.

2. NCC has a lower variance than NN. NN can represent much more complex decision boundaries. This greater power comes at the cost of greater capacity to overfit the data. Intuitively, in our example in Figure 2, when choosing the NCC classifier, we are forcing the decision boundary to be a straight line, while the NN's boundary can be a broken-line of high-complexity. This means that for NCC, we only have to estimate two parameters (the equation of the line) with n samples, which leads to a much lower variance than for NN.

VIII. Conclusions and Future Work

We have shown that an obscure result on averaging "warped" time series can be augmented to allow us to create much faster and/or more accurate time series classifiers. Our results may be particularly useful for resource constrained situations, such as wearable devices and "in-sensor" classifiers [36]. We have demonstrated the utility of our approach and ideas on more than 40 datasets, and made all code and data freely available to allow independent confirmation and extensions of our work [16].

Note that the classic data condensing methods such as Drop$\{X\}$ occasionally do reasonably well, at least at some levels of condensation. Further note that the only operator in their search space, the *deletion* of items, is completely orthogonal to our proposed methods. This suggests that we may be able to further improve our search space by expanding our search space to include *deletion*. We propose to consider this avenue in future work.

Acknowledgment

This research was supported by the ARC DP120100553 and DP140100087, the NSF IIS-1161997, the Bill and Melinda Gates Foundation, Vodafone's Wireless Innovation Project, the French-Australia Science Innovation Collaboration Grants PHC Grant #32571NA and by the Air Force Office of Scientific Research, Asian Office of Aerospace Research under contracts FA2386-15-1-4017 and FA2386-15-1-4007.

References

[1] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[2] A. Bagnall and J. Lines, "An experimental evaluation of nearest neighbour time series classification. technical report #CMP-C14-01," Department of Computing Sciences, University of East Anglia, Tech. Rep., 2014.

[3] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Int. Conf. on Machine Learning*, 2006, pp. 1033–1040.

[4] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *Int. Conf. on Knowledge Discovery and Data Mining*, 2012, pp. 262–270.

[5] I. Assent, M. Wichterich, R. Krieger, H. Kremer, and T. Seidl, "Anticipatory DTW for efficient similarity search in time series databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 826–837, 2009.

[6] H. Kremer, S. Günnemann, A.-M. Ivanescu, I. Assent, and T. Seidl, "Efficient processing of multiple DTW queries in time series databases," in *Scientific and Statistical Database Management*. Springer, 2011, pp. 150–167.

[7] D. E. Zhuang, G. C. Li, and A. K. Wong, "Discovery of temporal associations in multivariate time series," *IEEE Transactions on Knowledge and Data Engineering*, 2014.

[8] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[9] F. Petitjean, G. Forestier, G.I. Webb, A.E. Nicholson, Y. Chen and E. Keogh, "Dynamic Time Warping Averaging of Time Series allows Faster and more Accurate Classification," in *Int. Conf. on Data Mining*, IEEE, 2014, pp. 470–479.

[10] F. Galton, "Vox populi," *Nature*, vol. 75, no. 1949, pp. 450–451, 1907.

[11] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu, "Diagnosis of multiple cancer types by shrunken centroids of gene expression," *National Academy of Sciences*, vol. 99, no. 10, pp. 6567–6572, 2002.

[12] J. Gou, Z. Yi, L. Du, and T. Xiong, "A local mean-based k-nearest centroid neighbor classifier," *The Computer Journal*, vol. 55, no. 9, pp. 1058–1071, 2012.

[13] P.E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, no °3, PP.515-516, 1968.

[14] X. Xi, K. Ueno, E. Keogh, and D.-J. Lee, "Converting non-parametric distance-based classification to anytime algorithms," *Pattern Analysis and Applications*, vol. 11, no. 3-4, pp. 321–336, 2008.

[15] "Additional material," http://www.tiny-clues.eu/Research/ICDM2014-DTW/index.php.

[16] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[17] D. Ariely, "Seeing sets: Representation by statistical properties," *Psychological Science*, vol. 12, no. 2, pp. 157–162, 2001.

[18] G. A. Alvarez, "Representing multiple objects as an ensemble enhances visual cognition," *Trends in cognitive sciences*, vol. 15, no. 3, pp. 122–131, 2011.

[19] R. Jenkins and A. Burton, "100% accuracy in automatic face recognition," *Science*, vol. 319, no. 5862, pp. 435–435, 2008.

[20] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997, ch. 14 Multiple String Comparison – The Holy Grail, pp. 332–367.

[21] L. Wang and T. Jiang, "On the complexity of multiple sequence alignment," *Journal of Computational Biology*, vol. 1, no. 4, pp. 337–348, 1994.

[22] L. Gupta, D. L. Molfese, R. Tammana, and P. G. Simos, "Nonlinear alignment and averaging for estimating the evoked potential," *IEEE Transactions on Biomedical Engineering*, vol. 43, no. 4, pp. 348–356, 1996.

[23] K. Wang, T. Gasser *et al.*, "Alignment of curves by dynamic time warping," *The Annals of Statistics*, vol. 25, no. 3, pp. 1251–1276, 1997.

[24] V. Niennattrakul and C. A. Ratanamahatana, "Shape averaging under time warping," in *Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, IEEE, vol. 2, 2009, pp. 626–629.

[25] S. Ongwattanakul and D. Srisai, "Contrast enhanced dynamic time warping distance for time series shape averaging classification," in *Int. Conf. on Interaction Sciences: Information Technology, Culture and Human*, ACM, 2009, pp. 976–981.

[26] D.-F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *Journal of Molecular Evolution*, vol. 25, no. 4, pp. 351–360, 1987.

[27] E. Keogh, X. Xi, L. Wei, and C. A. Ratanamahatana, "The UCR time series classification/clustering homepage," *http://www.cs.ucr.edu/~eamonn/time_series_data/*, 2011.

[28] F. Petitjean and P. Gançarski, "Summarizing a set of time series by averaging: From steiner sequence to compact multiple alignment," *Theoretical Computer Science*, vol. 414, no. 1, pp. 76–91, 2012.

[29] F. Petitjean, J. Inglada and P. Gançarski, "Satellite Image Time Series Analysis under Time Warping," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 50, no. 8, pp. 3081-3095, 2012.

[30] F. Petitjean, "Matlab and Java source code for DBA," doi:10.5281/zenodo.10432, 2014.

[31] P. Kranen and T. Seidl, "Harnessing the strengths of anytime algorithms for constant data streams," *Data Mining and Knowledge Discovery*, vol. 19, no. 2, pp. 245–260, 2009.

[32] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, and E. Keogh, "Discovering the intrinsic cardinality and dimensionality of time series using MDL," in *Int. Conf. on Data Mining*, IEEE, 2011, pp. 1086–1091.

[33] C. A. Ratanamahatana and E. Keogh, "Three myths about dynamic time warping data mining," in *SIAM Int. Conf. on Data Mining*, 2005, pp. 506–510.

[34] V. Niennattrakul and C. A. Ratanamahatana, "Inaccuracies of shape averaging method using dynamic time warping for time series data," in *Int. Conf. on Computational Science*. Springer, 2007, pp. 513–520.

[35] E. Pekalska, R. P. Duin, and P. Paclik, "Prototype selection for dissimilarity-based classifiers," *Pattern Recognition*, vol. 39, no. 2, pp. 189–208, 2006.

[36] K. Ueno, X. Xi, E. Keogh, and D.-J. Lee, "Anytime classification using the nearest neighbor algorithm with applications to stream mining," in *Int. Conf. on Data Mining*, IEEE, 2006, pp. 623–632.

[37] Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh, "Flying insect classification with inexpensive sensors," *Journal of Insect Behavior*, vol. 27, no. 5, pp. 657-677, 2014.

[38] L. B. Goddard, A. E. Roth, W. K. Reisen, T. W. Scott *et al.*, "Vector competence of California mosquitoes for west Nile virus," *Emerging infectious diseases*, vol. 8, no. 12, pp. 1385–1391, 2002.

[39] Y. Yang, G.I. Webb, K. Korb, and K-M. Ting, "Classifying under Computational Resource Constraints: Anytime Classification Using Probabilistic Estimators," *Machine Learning*, vol. 69, no. 1, 2007

APPENDIX A. PROOF OF CONVERGENCE OF DBA

We want to prove that, at each iteration, DBA provides a better average sequence $\bar{T}$, i.e. has a lower sum of squares (Equation 2). DTW guarantees to find the minimum alignment between two sequences, which proves optimality for the first step of DBA (Table I - Algorithm 2 – lines 1 – 8). Proving convergence thus requires showing that for a given multiple alignmen tM, the computed $\bar{T}$ is optimal.

Let $M = DTW\_multiple\_alignment(\bar{T}, \boldsymbol{D})$ (Table I – Algorithm 3) and $M_\ell = M[\ell]$. We start by rewriting the objective function (sum of squares – SS):

$$SS(\bar{T}, \boldsymbol{D}) = \sum_{i=0}^{N} \mathrm{DTW}^2(\bar{T}, T_i) = \sum_{\ell=1}^{L} \sum_{e \in M_\ell} (\bar{T}(\ell) - e)^2 \qquad (4)$$

where e is an element of a sequence of $\boldsymbol{D}$ that has been "linked" to the ℓ^{th} element of $\bar{T}$ by Dynamic Time Warping. Given that this function has no maximum, it is minimized when its partial derivative is 0:

$$\begin{aligned} & \frac{\partial SS(\bar{T}, \boldsymbol{D})}{\partial \bar{T}(\ell)} = 0 \\ \Rightarrow \quad & \sum_{e \in M_\ell} 2 \cdot (\bar{T}(\ell) - e) = 0 \\ \Rightarrow \quad & \bar{T}(\ell) = \frac{1}{|M_\ell|} \sum_{e \in M_\ell} e \end{aligned} \qquad (5)$$

This leads to $SS(\bar{T}, \boldsymbol{D})$ being minimized when every element ℓ of $\bar{T}$ is positioned as the mean of $|M_\ell|$. ■

APPENDIX B. QUANTITATIVE EVALUATION OF DBA

TABLE 6: COMPARISON OF INTRA-CLASS SUM OF SQUARES FOR DYNAMIC TIME WARPING (AS PER EQUATION 2)

Dataset	Intra-class sum of squares		
	Medoid	EUC	DBA
50words	64,451	21,635	**9,315**
Adiac	376	438	**202**
Beef	12.6	13.3	**4**
CBF	33,836	14,738	**12,654**
ChlorineConcentration	76,983	82,174	**62,796**
CinC_ECG_torso	1,844,018	653,815	**145,813**
Coffee	55,177	53,930	**28,916**
Cricket_X	173,688	111,498	**42,149**
Cricket_Y	165,918	91,182	**38,534**
Cricket_Z	171,884	110,940	**41,900**
DiatomSizeReduction	644	513	**468**
ECG200	2,405	1,857	**1,509**
ECGFiveDays	25,876	7,546	**5,919**
FaceAll	93,441	41,152	**32,818**
FaceFour	8,963	4,907	**2,974**
FacesUCR	103,560	48,063	**37,916**
FISH	1,048	697	**510**
Gun_Point	1,558	1,843	**479**
Haptics	27,469	9,376	**6,474**
InlineSkate	203,280	94,621	**16,779**
ItalyPowerDemand	2,930	2,825	**2,470**
Lighting2	25,172	13,811	**9,308**
Lighting7	12,444	7,764	**5,066**
MALLAT	15,924	13,154	**6,860**
MedicalImages	20,997	16,451	**10,767**
MoteStrain	31,594	29,344	**23,751**
OliveOil	0.17	0.20	**0.10**
OSULeaf	84,824	25,376	**10,929**
SonyAIBORobotSurface	5,419	5,102	**4,165**
SonyAIBORobotSurfaceII	18,747	14,096	**11,637**
StarLightCurves	891,254	659,048	**134,463**
Stig	129,079	79,955	**20,025**
SwedishLeaf	8,959	4,045	**2,875**
Symbols	29,905	11,223	**6,224**
synthetic_control	13,899	6,564	**5,472**
Trace	18,157	21,172	**3,294**
TwoLeadECG	2,516	1,644	**1,481**
Two_Patterns	534,606	72,145	**53,696**
uWaveGestureLibrary_X	637,481	385,641	**121,159**
uWaveGestureLibrary_Y	617,871	443,306	**108,913**
uWaveGestureLibrary_Z	685,485	423,927	**127,617**
Wafer	670,529	522,476	**253,702**
WordsSynonyms	84,584	29,555	**11,052**
Yoga	372,221	118,981	**41,132**

APPENDIX C. REPRESENTATIVE SAMPLES OF THE FULL SET OF RESULTS AVAILABLE AT [33]

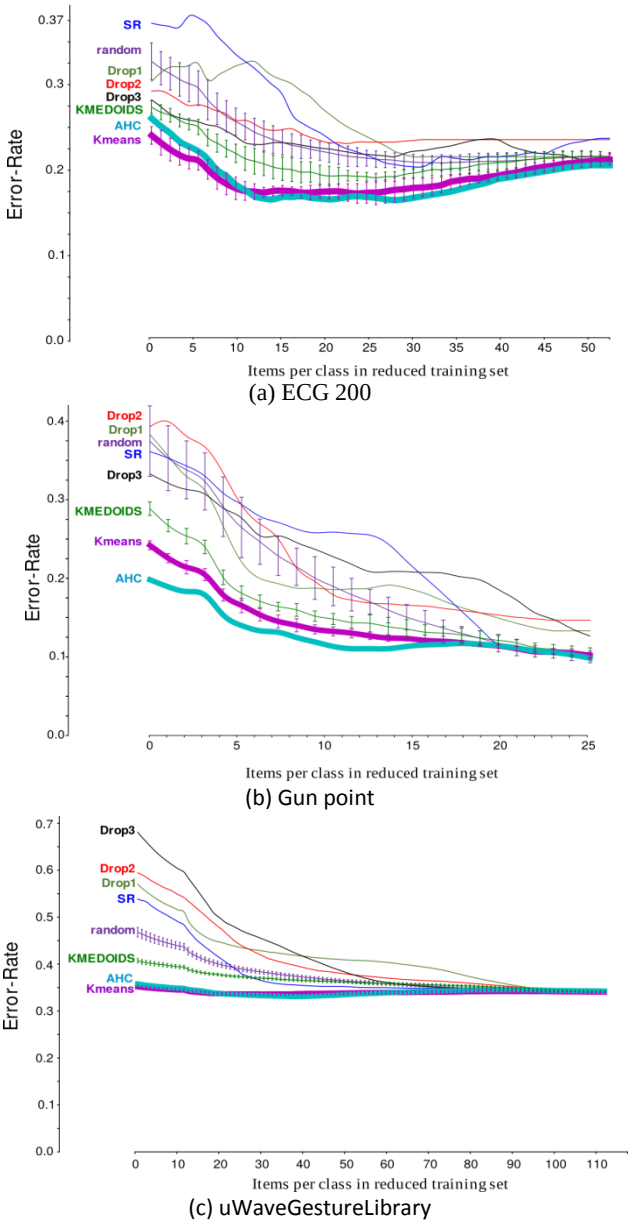


Figure 9: (best viewed in color) The error rate (with standard deviation) of various data condensing techniques for every output training size from 1 per class to 100 per class. The curves are slightly smoothed for visual clarity; the raw data spreadsheets are available at [33].

Figure 9(a) presents the results on the electrocardiograms time series dataset (ECG 200) which show the electrical potential between two points on the surface of the body caused by a beating heart [27]. In this dataset, the proposed condensing methods that make use of the average (KMeans and AHC) outperform all other methods. Similarly, as in our example for insect surveillance (Figure 8), a better overall accuracy can be reached while using a subset of prototypes instead of using the entire training set. The technique based on AHC

reaches an error-rate of 14% with only 16 prototypes per class, while the full 1-NN algorithm requires more than 50 prototypes per class to obtain a 23% error-rate.

Figure 9(b) presents the results on the Gun/NoGun motion capture time series dataset. Here again, our average-based condensing techniques dominate state-of-the-art methods. It is interesting to observe the important reduction of the error-rate with 2 to 5 items per class. This can be explained by the multimodality of the two classes of the dataset, which has been created from recording of movements of people with different heights.

Figure 9(c) presents the results on the uWaveGestureLibrary(Z) time series dataset which contains over 4000 samples of accelerometer readings for gesture recognition. This example shows that one prototype per class makes it possible to "explain" most of the variance in the classes of the dataset. This is another critical example, because gesture recognition systems not only have to be reliable, but also often must perform the recognition very quickly. With one prototype per class on this dataset that is composed of more than 100 training time series for each class, our condensing technique offers a 100-fold speedup, with a loss in the recovery of only 5%. This starkly contrasts with a condensing using the best non-average-based method (K-medoids), for which the error-rate increases by 14% for the same speedup.

BIOGRAPHIES

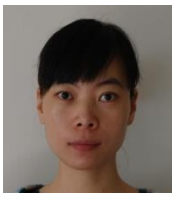
François Petitjean is a Researcher at the Center for Data Science at Monash University. His research area include data mining and machine learning, and focuses on the analysis of large and high-dimensional data. He obtained his PhD in 2012 working for the French Space Agency and then joined Geoff Webb's machine learning team at Monash University.

Germain Forestier obtained the MSc and PhD in Computer Science from the Université de Strasbourg, France, in 2007 and 2010. He was a post-doctoral fellow at INRIA Rennes, France, between 2010 and 2011. He is now an associate professor at Université de Haute-Alsace, France and a member of the MIPS laboratory. His scientific interests include data mining, knowledge and data engineering, semantic web and image processing.

Geoff Webb is Director of the Monash University Center for Data Science. He was editor in chief of Data Mining and Knowledge Discovery from 2005 to 2014. He has been Program Committee Chair of both ACM SIGKDD and IEEE ICDM, as well as General Chair of ICDM. He is a Technical Advisor to BigML Inc, who are incorporating his best of class association discovery software, Magnum Opus, into their cloud based Machine Learning service. He developed many of the key mechanisms of support-confidence association discovery in the 1980s. His OPUS search algorithm remains the state-of-the-art in rule search. He pioneered multiple research areas as diverse as black-box user modelling, interactive data analytics and statistically-sound pattern discovery. He has developed many useful machine learning algorithms that are widely deployed. He received the 2013 IEEE Outstanding Service Award, a 2014 Australian Research Council Discovery Outstanding Researcher Award and is an IEEE Fellow.

Professor **Ann Nicholson** is the Associate Dean Education in the Faculty of Information Technology. After completing her BSc (Hons) and MSc in Computer Science at the University of Melbourne, she completed her doctorate at the University of Oxford in the Robotics Research Group . Ann has worked on many areas within Artificial Intelligence, including probabilistic planning, user modelling and robotics. Her primary research focus is on probabilistic graphical models, specifically Bayesian

networks. Her research includes approximate inference algorithms, causal discovery and elicitation methodologies, and has a strong cross-disciplinary focus, applying Bayesian networks in many domains including meteorology, epidemiology, medicine, education and environmental science.



Yanping Chen is a PhD student of computer science at the University of California Riverside. Her research focused on data mining, machine learning and the applications of these techniques to time series data analysis. She has three papers published in SIGKDD, two in ICDM and one in SDM. She is also doing research on insect detection and classification. Her research in insect study has attracted great attentions from industry, research institutions and popular media.

Eamonn Keogh is a professor of computer science at the University of California Riverside. His research areas include data mining, machine learning and information retrieval, specializing in techniques for solving similarity and indexing problems in time-series datasets. He has authored more than 200 papers. He received the IEEE ICDM 2007 best paper award, SIGMOD 2001 best paper award, and best paper award in SIGKDD 2012. He has given over two dozen well received tutorials in the premier conferences in data mining and databases.