2014 IEEE International Conference on Data Mining

Dynamic Time Warping Averaging of Time Series allows Faster and more Accurate Classification

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, firsname.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*—Recent years have seen significant progress in improving both the efficiency and effectiveness of time series classification. However, because the best solution is typically the Nearest Neighbor algorithm with the relatively expensive Dynamic Time Warping as the distance measure, successful deployments on resource constrained devices remain elusive. Moreover, the recent explosion of interest in wearable devices, which typically have limited computational resources, has created a growing need for very efficient classification algorithms. A commonly used technique to glean the benefits of the Nearest Neighbor algorithm, without inheriting its undesirable time complexity, is to use the Nearest Centroid algorithm. However, because of the unique properties of (most) time series data, the centroid typically does not resemble *any* of the instances, an unintuitive and underappreciated fact. In this work we show that we can exploit a recent result to allow meaningful averaging of "warped" times series, and that this result allows us to create ultra-efficient Nearest "Centroid" classifiers that are at least as accurate as their more lethargic Nearest Neighbor cousins.**

I. INTRODUCTION

There is 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 gesture recognition, robotics and ECG classification [1]. Moreover recent comprehensive studies have validated this idea:

- In [1] we compare 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 test the assumption that the Nearest Neighbor classifier is the best technique and consider other classifiers, including 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 Nearest Neighbor algorithm and to NN-DTW in particular.

These results have meant that the most recent research has simply assumed the utility of NN-DTW and focused on mitigating the oft-lamented drawback of DTW: its time complexity. Here there has also been recent significant progress, with [4] showing that nearest neighbor queries under DTW can be answered in time that is no worse than twice that of the Euclidean distance.

However we argue that there are still situations where DTW (or for that matter, Euclidian distance) has severe tractability issues. The accuracy of NN is a function of the size of the training set, but unlike eager learners, the classification *time* is also a function of the size of the training set. In order to obtain a required level of accuracy, we may have to compare the incoming exemplar to dozens or hundreds of training objects. While the optimizations in [4][5][6] can help mitigate the time needed somewhat, NN-DTW may still be untenable in some circumstances. This is especially true for resource constrained devices such as wearable computers and embedded medical devices.

An obvious fix is to reduce the size of the training set to the largest size we can search at each time interval. In [3] it is shown 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. It allows us to avail of the strengths of the NN algorithm, while bypassing the latter's 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 objects that have a very low mutual DTW distance, if we attempt to average them the result will typically be "*neither fish nor fowl*", resembling *none* of the parent objects.

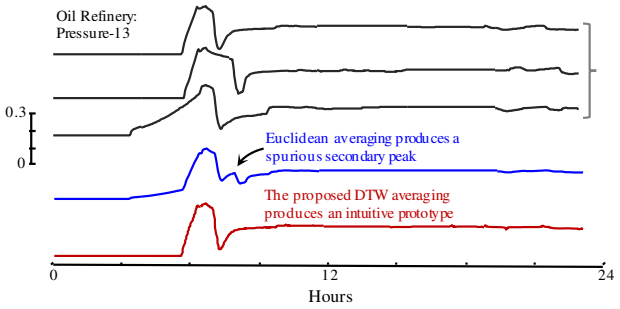

Figure 1: *top*) Three examples of daily patterns at an oil refinery [7]. *middle*) When averaged under the Euclidean distance the resulting centroid has an additional peak that is in none of the original time series. *bottom*) When averaged using the DTW based method proposed in this work, the "centroid"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

In this work we leverage off and extend a little known recent result that allows us to meaningfully define "centroid" under DTW [8]. As we shall show, this allows

1550-4786/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDM.2014.27

us to condense large datasets into much smaller (as small as a single instance per class) dataset that can produce the same accuracy as the original dataset. Less intuitively, in some domains the reduced datasets may allow greater accuracy, because the averaging combines evidence from all exemplars to produce prototypes that are more like the classes platonic ideal than any individual instance.

The rest of this paper is organized as follows. In Section II we review related and background work. In Section III we introduce the necessary definitions and formally define the problem to be solved, allowing us to introduce our solution in Section IV. Section V sees a forceful empirical validation of our claims, and we offer conclusions and directions for future work in Section VI.

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 [9]. 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. For example the Nearest *centroid* classifier [10] generalizes the Nearest *neighbor* classifier by replacing the set of neighbors with their centroid. 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). However, and less intuitively, it is also known that some circumstances, the Nearest *centroid* classifier is *more accurate* than the Nearest *neighbor* classifier (NN) [11].

Because it may be counterintuitive that the nearest centroid classifier can be more accurate than NN, we will demonstrate this 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.

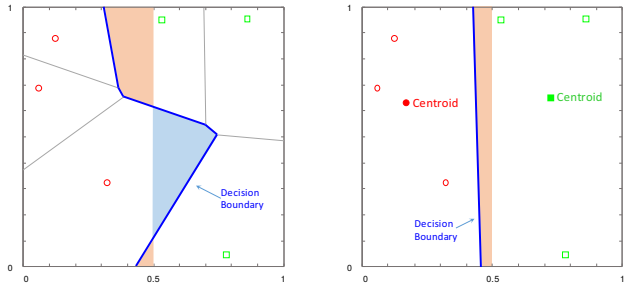

Figure 2: A simple classification problem in which the concept is the left vs. right side of the unit square. This instance of the problem has three points per class. *left*) Here NN has error-rate of 12.60%, while the Nearest Centroid classifier (right) with the same instances achieves an error-rate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

For balanced dataset sizes from 2 to 4,000, we compared the error rates of the NN and the Nearest centroid classifier (NCC) on this domain, each time averaging over 1,000 runs. The results are shown in Figure 3.

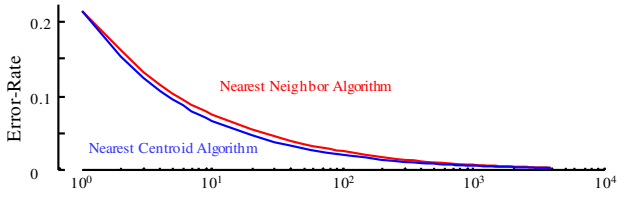

Figure 3: The error rate of two algorithms for increasingly large training data sizes of "left vs. right side of the unit square" problem.

Without any experiments we would realize that the two algorithms must agree on the far left side of the figure, 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. The effect is small, but *is* statistically significant.

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 [35]) shown in Figure 4, here the NN algorithm approaches zero error-rate for large training dataset sizes, in contrast the Nearest centroid classifier steadfastly achieves just the default rate.

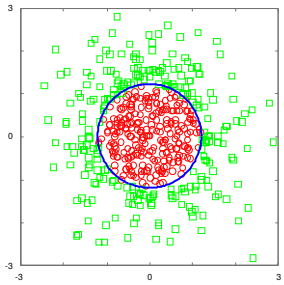

Figure 4: A two-class problem in which objects within 1.2 of the origin are in class **A**, otherwise they are in class **B**. With enough training data the NN classier can learn this concept very well; however the nearest centroid classifier is condemned to perform at the default rate.

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 and NN can have approximately the same accuracy, 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 systems remarkable abilities stem, at least in part, from its ability to represent sets of objects by a "gist" or "ensemble"[1], which may be simply the *average* of the objects [12]. 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*" [13].

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 man. 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 boasting "*100% Accuracy in Automatic Face Recognition*" (this is the paper's title [14]).

Compared to the complexity inherent in faces, time series seem like they would be 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.

III. Definitions and Problem Statement

We present the definitions of key terms that we use in this work. For our problem, each object in the data set is a *time series*, which may be of different length.

A. Definitions

***Definition* 1**: *Time Series.* A time series $T = (t_1, \dots, t_L)$ 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 $\boldsymbol{D} = \{T_1, \dots, T_N\}$ is a collection of N such time series.

B. 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 [15]. 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 [16] 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.

***Definition* 2**: *Average object.* Given a set of objects $O = \{O_1, \dots, O_N\}$ 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:

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

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 to be consistent with Dynamic Time Warping.

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

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

Many attempts at finding an averaging method for DTW have been made since the 1990s [17], [18], [19], [20]. 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 [19], heuristically averages the closest objects first, which corresponds to an idea proposed some 20 years earlier in computational biology [21].

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-*

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

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 platonic 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.

C. 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 account of 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 [22]. 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 $\bar{T}$ fixed and find the best multiple alignment[2] M of the set of sequences $\boldsymbol{D}$ consistently with $\bar{T}$.
2. Now consider M fixed and update $\bar{T}$ as the best average sequence consistent with M.

Table I gives the pseudocode of DBA; an implementation in Matlab and Java is available at [23].

This paper extends the definition of DBA by providing a proof of its convergence, i.e., that the sum of the squares (Equation 2) always decreases between two iterations (or refinements). This proof is provided in Appendix A.

In Figure 1 we showed an example of the algorithm's output on three examples of a pattern associated with an oil refinery process.

IV. Observations and Algorithms

In recent years there has been an increasing interest in using anytime algorithms for data mining [3], [24]. 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 a tradeoff between computation time and quality of results, but they are not interruptible.

[2] It actually finds the *compact* multiple alignment [27].

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

```
T̄ = medoid( D ) // get the medoid of the set of sequences D
do I times T̄ = DBA_update( T̄ , D )
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

```
// Step #1: compute the multiple alignment for T̄_init
alignment = [ ∅, ⋯ , ∅ ] // array of L empty sets
for each S in D do
    alignment_for_S = DTW_multiple_alignment ( T̄_init , S )
    for i=1 to L do
        alignment[i] = alignment[i] ∪ alignment_for_S[i]
    done
done
// Step #2: compute the multiple alignment for the alignment
let T̄ be a sequence of length L
for i=1 to L do
    T̄(i) = mean( alignment[i] ) //arithmetic mean on the set
done
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

```
// Step #1: compute the accumulated cost matrix of DTW
cost = DTW( S_ref , S )
// Step #2: store the elements associated to S_ref
L = length( S_ref )
alignment = [ ∅, ⋯ , ∅ ] // array of L empty sets
i = rows( cumul_cost ) // i iterates over the elements of S_ref
j = columns( cumul_cost ) //j iterates over the elements of S
while (i > 1) && (j > 1) do
    alignment[i] = alignment[i] ∪ S(j)
    if i == 1 then j = j − 1
    else if j == 1 then i = i − 1
    else
        score = min( cost[i-1][j-1] , cost[i][j-1] , cost[i-1][j] )
        if score = = cost[i-1][j-1] then
            i = i − 1
            j = j − 1
        else if score = = cost[i-1][j] then i = i − 1
        else j = j − 1
        end if
    end if
done
return alignment
```

Problem Statement *Contract Time Series Classification*: Given (1) a large time series training dataset, (2) the maximum amount of computation resources available, and (3) as much training time as needed, 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 5 illustrates this problem statement.

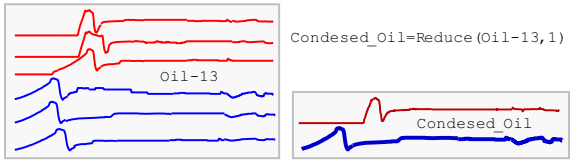

Figure 5: A visual intuition of an instance of our problem statement: Given the Oil-13 time series training dataset (*left*), and a user constraint *C*, here '1'. Produce a new dataset with *C* items per class (*right*), such that the accuracy on future data is maximized.

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, the fact that no competitively accurate classifiers for time series based on these methods have been produced [2], [3], in a research area as active and competitive as time series classification, is very telling.

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 [25], it only helps scalability on specialized hardware. We are hoping 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 considered data *aggregation*. Or rather, it may have been considered, but 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], [27] and particularly by [28], make this an unpromising avenue to explore.

However, 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 allowing improved accuracy.

However, as we explain in the next section, 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.

A. *Why K-Means Based 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 in a domain we explore in this work, consider the case study in insect surveillance in Section V.A. Here we may have what 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 show such a clustering-based approach to condensing a dataset.

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

TABLE II. ALGORITHM TO CONDENSE TRAINING DATASET

Algorithm 1. Reduce(*Data, C*)

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

```
// partition the data into  C sets of time series
Clusters = do_clustering(Data,C) //for example with K-means
Condensed_Data = ()
for each Cluster in Clusters do
        Condensed_Data.add(DBA(Cluster))
done
return Condensed_Data
```

V. 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 [29]. 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 Drop 3, see [29] 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 [30]. 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_j 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, our two proposed methods:

- **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 as the average method associated to DTW.
- **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 as the average method associated to DTW.

We consider situations where we can only visit a small handful of exemplars, as few as just one per class; 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 will be 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.

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 [31], which, as shown in Figure 6 can essentially be considered a "time series".

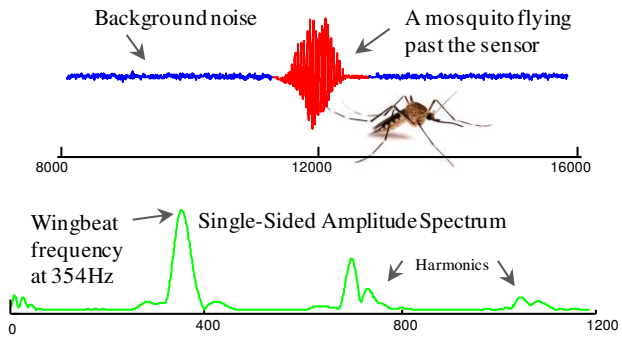

Figure 6: *top*) An audio snippet of an insect flight sound can be converted into a pseudo time series (*bottom*) and used to allow classification

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 [31] 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 [31] are demonstrated in the laboratory settings, 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 [32]. 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 7, 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][29], where we expect to pay a cost (in accuracy) for the gains in speed.

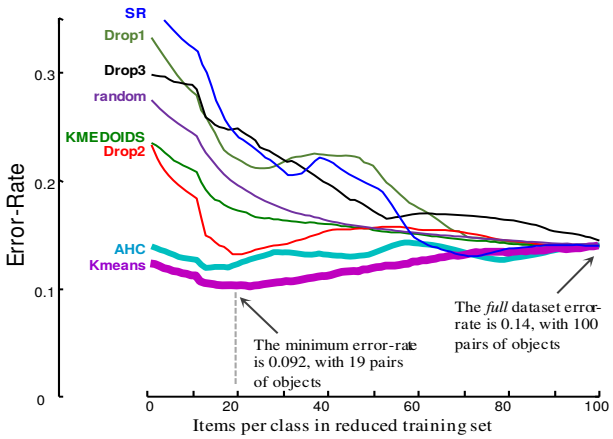

Figure 7: (best viewed in color) The error rate of various data condensing techniques for every output training size from 1 per class to 100 per class. The curves are slightly smoothed for visual clarity; the raw data spreadsheets are available at [33].

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.

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 7 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 7 for *all* the datasets, but also the rankings used in the reminder of this section) cannot be included in the paper due to space limitations, but are available at [33].

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 [22][3]. 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

[3] 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.

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 [34] 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.[4]

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 best error-rate for the considered "contract".

TABLE IV shows the average rank of all algorithms over the datasets of [22] (again, the raw results giving the error rate and rank for every method and every dataset is available at [33]). These results show unanimously that the methods that use an average sequence (K-means and AHC) significantly outperform the prior state of the art.

In addition, we test the statistical significance of these results. We want to assess if 42 datasets is a large enough sample to state that this difference in the ranking is statistically significant.

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

[4] 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].

We first perform a Friedman test [34], in order to assess if the results are significantly different. This test is used to evaluate if there is enough evidence to confidently state that the different methods are not performing equally.

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

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 [34] and perform a two-tailed Bonferronni-Dunn test to compare pairs of methods. Because our aim is not to show the prevalence of any algorithm in particular, but that using the average yields better performance for time series classification, 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, [34] shows that, to be statistically significant ($\alpha = 0.05$) the difference between the average rankings has to be greater than:

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

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 reference 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. 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*.

VI. 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 [30]. We have demonstrated the utility of our ideas on more than 40 datasets, and made all code and data freely available to allow independent confirmation and extensions of our work [33].

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, the NSF IIS-1161997, the Bill and Melinda Gates Foundation and Vodafone's Wireless Innovation Project.

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

[10] 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.

[11] 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.

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

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

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

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

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 to show for a given multiple alignment M, the computed $\bar{T}$ is optimal.

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

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

Note that 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:

$$\frac{\partial \mathrm{SS}(\bar{T}, \boldsymbol{D})}{\partial \bar{T}(\ell)} = 0$$

$$\Rightarrow \quad \sum_{e \in M_\ell} 2 \cdot (\bar{T}(\ell) - e) = 0$$

$$\Rightarrow \quad \bar{T}(\ell) = \frac{1}{|M_\ell|} \sum_{e \in M_\ell} e \qquad (5)$$

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