A global averaging method for dynamic time warping, with applications to clustering

François Petitjean\textsuperscript{a,b,c,*}, Alain Ketterlin\textsuperscript{a,b}, Pierre Gançarski\textsuperscript{a,b}

\textsuperscript{a} University of Strasbourg, 7 rue René Descartes, 67084 Strasbourg Cedex, France
\textsuperscript{b} LSIIT–UMR 7005, Pôle API, Bd Sébastien Brant, BP 10413, 67412 Illkirch Cedex, France
\textsuperscript{c} Centre National d’Études Spatiales, 18 avenue Edouard Belin, 31401 Toulouse Cedex 9, France

\textbf{A R T I C L E  I N F O}

Article history:
Received 13 January 2010
Received in revised form 11 August 2010
Accepted 18 September 2010

Keywords:
Sequence analysis
Time series clustering
Dynamic time warping
Distance-based clustering
Time series averaging
DTW barycenter averaging
Global averaging
Satellite image time series

\textbf{A B S T R A C T}

Mining sequential data is an old topic that has been revived in the last decade, due to the increasing availability of sequential datasets. Most works in this field are centred on the definition and use of a distance (or, at least, a similarity measure) between sequences of elements. A measure called dynamic time warping (DTW) seems to be currently the most relevant for a large panel of applications. This article is about the use of DTW in data mining algorithms, and focuses on the computation of an average of a set of sequences. Averaging is an essential tool for the analysis of data. For example, the \textit{K-MEANS} clustering algorithm repeatedly computes such an average, and needs to provide a description of the clusters it forms. Averaging is here a crucial step, which must be sound in order to make algorithms work accurately. When dealing with sequences, especially when sequences are compared with DTW, averaging is not a trivial task.

Starting with existing techniques developed around DTW, the article suggests an analysis framework to classify averaging techniques. It then proceeds to study the two major questions lifted by the framework. First, we develop a global technique for averaging a set of sequences. This technique is original in that it avoids using iterative pairwise averaging. It is thus insensitive to ordering effects. Second, we describe a new strategy to reduce the length of the resulting average sequence. This has a favourable impact on performance, but also on the relevance of the results. Both aspects are evaluated by the framework. First, we develop a global technique for averaging a set of sequences. This technique is original in that it avoids using iterative pairwise averaging. It is thus insensitive to ordering effects. Second, we describe a new strategy to reduce the length of the resulting average sequence. This has a favourable impact on performance, but also on the relevance of the results. Both aspects are evaluated by the framework.

The article ends by describing the use of averaging in clustering. The last section also introduces a new application domain, namely the analysis of satellite image time series, where data mining techniques provide an original approach.

\textcopyright 2010 Elsevier Ltd. All rights reserved.

1. Introduction

Time series data have started to appear in several application domains, like biology, finance, multimedia, image analysis, etc. Data mining researchers and practitioners are thus adapting their techniques and algorithms to this kind of data. In exploratory data analysis, a common way to deal with such data consists in applying clustering algorithms. Clustering, i.e., the unsupervised classification of objects into groups, is often an important first step in a data mining process. Several extensive reviews of clustering techniques in general have been published \cite{1–4} as well as a survey on time series clustering \cite{5}.

Given a suitable similarity measure between sequential data, most classical learning algorithms are readily applicable. A similarity measure on time series data (also referred to as sequences hereafter) is more difficult to define than on classical data, because the order of elements in the sequences has to be considered. Accordingly, experiments have shown that the traditional Euclidean distance metric is not an accurate similarity measure for time series. A similarity measure called dynamic time warping (DTW) has been proposed \cite{6,7}. Its relevance was demonstrated in various applications \cite{8–13}.

Given this similarity measure, many distance-based learning algorithms can be used (e.g., hierarchical or centroid-based ones). However, many of them, like the well-known \textit{K-MEANS} algorithm, or even Ascendant Hierarchical Clustering, also require an averaging method, and highly depend on the quality of this averaging. Time series averaging is not a trivial task, mostly because it has to be consistent with the ability of DTW to realign sequences over time. Several attempts at defining an averaging method for DTW have been made, but they provide an inaccurate...
2. Dynamic time warping (DTW)

In this section, we recall the definition of the euclidean distance and of the DTW similarity measure. Throughout this section, let \( A = \langle a_1, \ldots, a_T \rangle \) and \( B = \langle b_1, \ldots, b_T \rangle \) be two sequences, and let \( \delta \) be a distance between elements (or coordinates) of sequences.

**Euclidean distance:** This distance is commonly accepted as the simplest distance between sequences. The distance between \( A \) and \( B \) is defined by

\[
D(A, B) = \sqrt{\delta(a_1, b_1)^2 + \cdots + \delta(a_T, b_T)^2}
\]

Unfortunately, this distance does not correspond to the common understanding of what a sequence really is, and cannot capture flexible similarities. For example, \( X = \langle a, b, a, a \rangle \) and \( Y = \langle a, b, a, a \rangle \) are different according to this distance even though they represent similar trajectories.

**Dynamic time warping:** DTW is based on the Levenshtein distance (also called edit distance) and was introduced in [6,7], with applications in speech recognition. It finds the optimal alignment (or coupling) between two sequences of numerical values, and captures flexible similarities by aligning the coordinates inside both sequences. The cost of the optimal alignment can be recursively computed by

\[
D(A_i, B_j) = \delta(a_i, b_j) + \min \left\{ \frac{D(A_i-1, B_j-1)}{D(A_i-1, B_j-1)} \right\}
\]

where \( A_i \) is the subsequence \( \langle a_1, \ldots, a_i \rangle \). The overall similarity is given by \( D(A_1, B_1) = D(A_T, B_T) \).

Unfortunately, a direct implementation of this recursive definition leads to an algorithm that has exponential cost in time. Fortunately, the fact that the overall problem exhibits overlapping subproblems allows for the memoization of partial results in a matrix, which makes the minimal-weight coupling computation a process that costs \(|A| \cdot |B|\) basic operations. This measure has thus a time and a space complexity of \( O(|A| \cdot |B|) \).

DTW is able to find optimal global alignment between sequences and is probably the most commonly used measure to quantify the dissimilarity between sequences [9–13]. It also provides an overall real number that quantifies similarity. An example DTW-alignment of two sequences can be found in Fig. 1: it shows the alignment of points taken from two sinusoids, one being slightly shifted in time. The numerical result computed by DTW is the sum of the heights of the associations. Alignments at both extremities on Fig. 1 show that DTW is able to correctly re-align one sequence with the other, a process which, in this case, highlights similarities that Euclidean distance is unable to capture.

**Algorithm 1. DTW**

**Require:** \( A = \langle a_1, \ldots, a_T \rangle \)

**Require:** \( B = \langle b_1, \ldots, b_T \rangle \)

Let \( \delta \) be a distance between coordinates of sequences

Let \( m[S,T] \) be the matrix of couples (cost,path) \( m[1,1] = (\delta(a_1, b_1), (0,0)) \)

**for** \( i \rightarrow 2 \) to \( S \) **do**

\( m[i,1] \leftarrow (m[i-1,1] + \delta(a_i, b_1), (i-1,1)) \)

**end for**

**for** \( j \rightarrow 2 \) to \( T \) **do**

\( m[1,j] \leftarrow (m[1,j-1] + \delta(a_1, b_j), (1,j-1)) \)

**end for**

**for** \( i \rightarrow 2 \) to \( S \) **do**

**for** \( j \rightarrow 2 \) to \( T \) **do**

minimum \( \leftarrow \min\{m[i-1,j], m[i,j-1], m[i-1,j-1]\} \)

\( m[i,j] \leftarrow (\text{first(minimum)} + \delta(a_i, b_j), \text{second(minimum)}) \)

**end for**

**end for**

return \( m[S,T] \)

\footnote{In fact, the distance \( \delta(a_i, b_j) \) computed in Eq. (1) is the distance between two coordinates without considering the time distance between them.}
Algorithm 2. first

Require: \( p = (a, b) \); couple
return \( a \)

Algorithm 3. second

Require: \( p = (a, b) \); couple
return \( b \)

Algorithm 4. minVal

Require: \( v_1, v_2, v_3 \); couple
if \( \text{first}(v_1) \leq \text{min} \big( \text{first}(v_2), \text{first}(v_3) \big) \) then
return \( v_1 \)
else if \( \text{first}(v_2) \leq \text{first}(v_1) \) then
return \( v_2 \)
else
return \( v_3 \)
end if

3. Related work

In the context of classification, many algorithms require a method to represent in one object, informations from a set of objects. Algorithms like K-MEDOIDS are using the medoid of a set of objects. Some others like K-MEANS need to average a set of objects, some like K-MEDOIDS need to designate the space of all sequences of length \( T \). Given a set of sequences \( S = \{S_1, \ldots, S_N\} \), the average sequence \( \bar{C} \) must fulfill

\[
\forall X \in X, \quad \sum_{n=1}^{N} \text{DTW}^2(C^{(n)}, S_n) \leq \sum_{n=1}^{N} \text{DTW}^2(X, S_n) \quad (2)
\]

Since no information on the length of the average sequence is available, the search cannot be limited to sequences of a given length, so all possible values for \( T \) have to be considered. Note that sequences of \( S \) have a fixed length \( T \). \( C \) has hence to fulfill

\[
\forall t \in [1, +\infty[, \quad \left( \forall X \in X, \sum_{n=1}^{N} \text{DTW}^2(C, S_n) \leq \sum_{n=1}^{N} \text{DTW}^2(X, S_n) \right) \quad (3)
\]

This definition relies in fact on the Steiner trees theory; \( C \) is called Steiner sequence [20]. Note that the sums in Eqs. (2) and (3), is often called Within Group Sum of Squares (WGSS), or discrepancy distance in [21]. We will also use the simple term inertia, used in most works on clustering.

3.2. Exact solutions to the Steiner sequence problem

As shown in [22], when considered objects are simple points in an \( \mathbb{R}^D \) dimensional space, the minimization problem corresponding to Eq. (2) can be solved by using the property of the arithmetic mean. As the notion of arithmetic mean is not easily extendable to semi-pseudometric spaces (i.e., spaces induced by semi-pseudometrics like DTW), we need to detail this Steiner sequence problem, i.e., the problem to find an average sequence. To solve this problem, there are two close families of methods.

The first one consists in computing the global multiple alignment [23] of the \( N \) sequences of \( S \). This multiple alignment is computable by extending DTW for aligning \( N \) sequences. For instance, instead of computing DTW by comparing three values in a square, one have to compare seven values in a cube for three sequences. This idea can be generalized by computing DTW in a \( N \)-dimensional hypercube. Given this global alignment, \( C \) can be found by averaging column by column the multiple alignment. However, this method presents two major difficulties, that prevent its use. First, the multiple alignment process takes \( O(T^N) \) operations [24], and is not tractable for more than a few sequences. Second, the global length of the multiple alignment can be on the order of \( T^N \), and requires unrealistic amounts of memory.

The second family of methods consists in searching through the solution space, keeping those minimizing the sum (Eq. (2)). In the discrete case (i.e., sequences of characters or of symbolical values), as the alphabet is generally finite, this scan is easy. However, as the length of the global alignment is potentially \( T^N - 2T \), scanning the space takes \( O(T^N) \) operations. In the continuous case (i.e., sequences of numbers or of number vectors), scanning all solutions is impossible. Nevertheless, we will explain in the next paragraph how this search can be guided towards potential solutions, even though this strategy also exhibits exponential complexity.

In fact, we need a method to generate all potential solutions. Each solution corresponds to a coupling between \( C \) and each sequence of \( S \). As all coordinates of each sequence of \( S \) must be associated to at least one coordinate of \( C \), we have to generate all possible groupings between coordinates of each sequence of \( S \).
and coordinates of $C$. Each coordinate of $C$ must be associated to a non-empty non-overlapping set of coordinates of $S$. To generate all possible groupings, we can consider that each sequence is split in as many subsequences as there are coordinates in $C$. Thus, the first coordinate of $C$ will be associated to the coordinates appearing in the first subsequences of sequences of $S$, the second coordinate to coordinates appearing in the second subsequences of sequences of $S$, and so on. Then, for $c^{(p)}$ ($C$ of length $p$), the $N$ sequences of $S$ have to be cut into $p$ parts. There are $\binom{N}{p}$ ways to cut a sequence of length $T$ into $p$ subsequences. Thus, for $N$ sequences there are $\left(\binom{N}{1}\right)^N$ possibilities. The time complexity of this scan is therefore in $\Theta(\binom{N}{p})$.

One can note in Eq. (3) that $p > T$ does not make sense. It would mean that sequences have to be split in more subsequences than they contain coordinates. Thus, we can limit the search conditions of $C$ to $p \in [1, T]$. But even with this tighter bound, the overall computation remains intractable.

3.3. Approximating the exact solution

As we explained in the previous subsection, finding the average sequence is deeply linked to the multiple alignment problem. Unfortunately, 30 years of well-motivated research did not provide any exact scalable algorithm, neither for the multiple alignment problem, nor for the consensus sequence problem. Given a sequence standing for the solution, we even cannot check if the potential solution is optimal because we rely on a subset of the search space. A number of heuristics have been developed to solve this problem (see [25–29] for examples). We present in this subsection the most common family of methods used to approximate the average sequence: iterative pairwise averaging. We will also link this family to existing averaging method for DTW.

Iterative pairwise averaging consists in iteratively merging two average sequences of two subsets of sequences into a single average sequence of the union of those subsets. The simplest strategy computes an average of two sequences and iteratively incorporates one sequence to the average sequence. Differences between existing methods are the order in which the merges are done, and the way they compute an average of two sequences.

3.3.1. Ordering schemes

Tournament scheme. The simplest and most obvious averaging ordering consists in pairwise averaging sequences following a tournament scheme. That way, $N/2$ average sequences are created at first step. Then those $N/2$ sequences, in turn, are pairwise averaged into $N/4$ sequences, and so on, until one sequence is obtained. In this approach, the averaging method (between two sequences) is applied $N$ times.

Ascendant hierarchical scheme. A second approach consists in averaging at first the two sequences whose distance (DTW) is minimal over all pairs of sequences. This works like Ascendant Hierarchical Clustering, computing a distance matrix before each average computation. In that way, the averaging method is also called $N - 1$ times. In addition, one has to take into account the required time to compute $N$ times the distance matrix.

3.3.2. Computing the average sequence

Regarding the way to compute an average from two sequences under DTW, most methods are using associations (coupling) computed with DTW.

One coordinate by association. Starting from a coupling between two sequences, the average sequence is built using the center of each association. Each coordinate of the average sequence will thus be the center of each association created by DTW. The main problem of this technique is that the resulting mean can grow substantially in length, because up to $|A| + |B| - 2$ associations can be created between two sequences $A$ and $B$.

One coordinate by connected component. Considering that the coupling (created by DTW) between two sequences forms a graph, the idea is to associate each connected component of this graph to a coordinate of the resulting mean, usually taken as the barycenter of this component. Contrary to previous methods, the length of resulting mean can decrease. The resulting length will be between 2 and $\min(|A|, |B|)$.

3.4. Existing algorithms

The different ordering schemes and average computations just described are combined in the DTW literature to make up algorithms. The two main averaging methods for DTW are presented below.

Nonlinear alignment and averaging filters: NLAAF was introduced in [30] and rediscovered in [15]. This method uses the tournament scheme and the one coordinate by association averaging method. Its main drawback lies in the growth of its resulting mean. As stated earlier, each use of the averaging method can almost double the length of the average sequence. The entire NLAAF process could produce, over all sequences, an average sequence of length $N \times T$. As classical datasets comprise thousands of sequences made up on the order of hundred coordinates, simply storing the resulting mean could be impossible. This length problem is moreover worsened by the complexity of DTW, that grows bi-linearly with lengths of sequences. That is why NLAAF is generally used in conjunction with a process reducing the length of the mean, leading to a loss of information and thus to an unsatisfactory approximation.

Prioritized shape averaging: PSA was introduced in [21] to resolve shortcomings of NLAAF. This method uses the Ascendant hierarchical scheme and the one by connected component averaging method. Although this hierarchical averaging method aims at preventing the error to propagate too much, the length of average sequences remains a problem. If one alignment (with DTW) between two sequences leads to two connected components (i.e., associations are forming two hand-held fans), the overall resulting mean will be composed of only two coordinates. Obviously, such a sequence cannot represent a full set of potentially long sequences. This is why authors proposed to replicate each coordinate of the average sequence as many times as there were associations in the corresponding connected component. However, this repetition of coordinates causes the problem already observed with NLAAF, by potentially doubling the number of coordinates of each intermediate average sequence. To alleviate this problem, the authors suggest using a process in order to reduce the length of the resulting mean.

3.5. Motivation

We have seen that most of the works on averaging sets of sequences can be analysed along two dimensions: first, the way they consider the individual sequences when averaging, and second, the way they compute the elements of the resulting sequences. These two characteristics have proved useful to classify the existing averaging techniques. They are also useful angles under which new solutions can be elaborated.

Regarding the averaging of individual sequences, the main shortcoming of all existing methods is their use of pairwise averaging. When computing the mean of $N$ sequences by pairwise averaging, the order in which sequences are taken influences the quality of the result, because neither NLAAF nor PSA are associative functions. Pairwise averaging strategies are intrinsically sensitive to the order, with no guarantee that a different order would lead to the same result. Local averaging strategies like PSA or NLAAF may let an
initial approximation error propagate throughout the averaging process. If the averaging process has to be repeated (e.g., during K-Means iterations), the effects may dramatically alter the quality of the result. This is why a global approach is desirable, where sequences would be averaged all together, with no sensitivity to their order of consideration. The obvious analogy to a global method is the computation of the barycenter of a set of points in a Euclidean space. Section 4 follows this line of reasoning in order to introduce a global averaging strategy suitable for DTW, and provides empirical evidence of its superiority over existing techniques.

The second dimension along which averaging techniques can be classified is the way they select the elements of the mean. We have seen that a naive use of the DTW-computed associations may lead to some sort of “overfit”, with an average covering almost every detail of every sequence, whereas simpler and smoother averages could well provide a better description of the set of sequences. Moreover, long and detailed averages have a strong impact on further processing. Here again, iterative algorithms like K-Means are especially at risk: every iteration may lead to a longer average, and because the complexity of DTW is directly related to the length of the sequences involved, later iterations will take longer than earlier ones. In such cases, unconstrained averaging will not only lead to an inadequate description of clusters, it will also cause a severe performance degradation. This negative effect of sequence averaging is well-known, and corrective actions have been proposed. Section 5 builds on our averaging strategy to suggest new ways of shortening the average.

4. A new averaging method for DTW

To solve the problems of existing pairwise averaging methods, we introduce a global averaging strategy called DTW barycenter averaging (DBA). This section first defines the new averaging method and details its complexity. Then DBA is compared to NLAADF and PSA on standard datasets [19]. Finally, the robustness and the convergence of DBA are studied.

4.1. Definition of DBA

DBA stands for DTW barycenter averaging. It consists in a heuristic strategy, designed as a global averaging method. DBA is an averaging method which consists in iteratively refining an initially (potentially arbitrary) average sequence, in order to minimize its squared distance (DTW) to averaged sequences.

Let us provide an intuition on the mechanism of DBA. The aim is to minimize the sum of squared DTW distances from the average sequence to the set of sequences. This sum is formed by single distances between each coordinate of the average sequence and coordinates of sequences associated to it. Thus, the contribution of one coordinate of the average sequence to the total sum of squared distance is actually a sum of euclidean distances between this coordinate and coordinates of sequences associated to it during the computation of DTW. Note that a coordinate of one of the sequences may contribute to the new position of several coordinates of the average. Conversely, any coordinate of the average is updated with contributions from one or more coordinates of each sequence. In addition, minimizing this partial sum for each coordinate of the average sequence is achieved by taking the barycenter of this set of coordinates. The principle of DBA is to compute each coordinate of the average sequence as the barycenter of its associated coordinates of the set of sequences. Thus, each coordinate will minimize its part of the total WGSS in order to minimize the total WGSS. The updated average sequence is defined once all barycenters are computed.

Technically, for each refinement i.e., for each iteration, DBA works in two steps:

1. Computing DTW between each individual sequence and the temporary average sequence to be refined, in order to find associations between coordinates of the average sequence and coordinates of the set of sequences.
2. Updating each coordinate of the average sequence as the barycenter of coordinates associated to it during the first step.

Let \( S = \{S_1, \ldots, S_N\} \) be the set of sequences to be averaged, let \( C = \langle C_1, \ldots, C_T \rangle \) be the average sequence at iteration \( i \) and let \( C' = \langle C'_1, \ldots, C'_T \rangle \) be the update of \( C \) at iteration \( i+1 \), of which we want to find coordinates. In addition, each coordinate of the average sequence is defined in an arbitrary vector space \( E \) (e.g., usually a Euclidean space):

\[
\forall t \in [1,T], \quad C_t \in E
\]

We consider the function \( \text{assoc} \), that links each coordinate of the average sequence to one or more coordinates of the sequences of \( S \). This function is computed during DTW computation between \( C \) and each sequence of \( S \). The \( t \)th coordinate of the average sequence \( C'_t \) is then defined as

\[
C'_t = \text{barycenter}(\text{assoc}(C_t))
\]

(see Eq. (6))

\[
\text{barycenter}(X_1, \ldots, X_z) = \frac{X_1 + \cdots + X_z}{z}
\]

(6)

(the addition of \( X_i \) is the vector addition). Algorithm 5 details the complete DBA computation.

Then, by computing again DTW between the average sequence and all sequences of \( S \), the associations created by DTW may change. As it is impossible to anticipate how these associations will change, we propose to make \( C \) iteratively converge. Fig. 2 shows four iterations (i.e., four updates) of DBA on an example with two sequences.

Algorithm 5. DBA

Require: \( C = \langle C_1, \ldots, C_T \rangle \) the initial average sequence
Require: \( S_1 = \langle s_{11}, \ldots, s_{1m} \rangle \) the 1st sequence to average
Require: \( S_n = \langle s_{n1}, \ldots, s_{nm} \rangle \) the \( n \)th sequence to average
Let \( T \) be the length of sequences
Let \( \text{assocTab} \) be a table of size \( T \) containing in each cell a set of coordinates associated to each coordinate of \( C \)
Let \( m[T,T] \) be a temporary DTW (cost,path) matrix

\[
\text{assocTab} = \{0, \ldots, 0\}
\]

for seq in \( S \) do

\[
m = \text{DTW}(C, \text{seq})
\]

\[
i = T
\]

\[
j = T
\]

while \( i \geq 1 \) and \( j \geq 1 \) do

\[
\text{assocTab}(i) \leftarrow \text{assocTab}(i) \cup \text{seq}_j
\]

\[
(i,j) \leftarrow \text{second}(m(i,j))
\]

end while

end for

for \( i = 1 \) to \( T \) do

\[
C'_i = \text{barycenter}(\text{assocTab}(i))\quad \text{[see Eq. (6)]}
\]

end for

return \( C' \)

As a summary, the proposed averaging method for dynamic time warping is a global approach that can average a set of sequences all together. The update of the average sequence
between two iterations is independent of the order with which the individual sequences are used to compute their contribution to the update in question. Fig. 3 shows an example of an average sequence computed with DBA, on one dataset from [19]. This figure shows that DBA preserves the ability of DTW, identifying time shifts.
4.2. Initialization and convergence

The DBA algorithm starts with an initial averaging and refines it by minimizing its WGSS with respect to the sequences it averages. This section examines the effect of the initialisation and the rate of convergence.

Initialization: There are two major factors to consider when priming the iterative refinement:

- first the length of the starting average sequence,
- and second the values of its coordinates.

Regarding the length of the initial average, we have seen in Section 3.2 that its upper bound is \( 1/T \), but that such a length cannot reasonably be used. However, the inherent redundancy of the data lets one except that a much shorter sequence can be adequate. We will observe in Section 4.5 that a length of around \( T \) (the length of the sequences to average) performs well.

Regarding the values of the initial coordinates, it is theoretically impossible to determine the optimal values, otherwise the whole averaging process would become useless. In methods that require an initialisation, e.g., K-Means clustering, a large number of heuristics have been developed. We will describe in Section 4.5 experiments with the most frequent techniques: first, a randomized choice, and second, using an element of the set of sequences to average. We will show empirically that the latter gives an efficient refinement.

Convergence. As explained previously, DBA is an iterative process. It is necessary, once the average is computed, to update it several times. This has the property of letting DTW refine its associations. It is important to note that at each iteration, inertia can only decrease, since the new average sequence is closer (under DTW) to the elements it averages. If the update does not modify the alignment of the sequences, so the Huygens' theorem applies; barycenters composing the average sequence will get closer to coordinates of \( S \). In the other case, if the alignment is modified, it means that DTW calculates a better alignment with a smaller inertia (which decreases in that case also). We thus have a guarantee of convergence. Section 4.6 details some experiments in order to quantify this convergence.

4.3. Complexity study

This section details the time complexity of DBA. Each iteration of the iterative process is divided into two parts:

1. Computing DTW between each individual sequence and the temporary (i.e., current) average sequence, to find associations between its coordinates and coordinates of the sequences.
2. Updating the mean according to the associations just computed.

Finding associations. The aim of Step 1 is to determine the set of associations between each coordinate of \( C \) and coordinates of sequences of \( S \). Therefore we have to compute DTW once per sequence to average, that is \( N \) times. The complexity of DTW is \( \Theta(T^2) \). The complexity of Step 1 is therefore \( \Theta(N \cdot T^2) \).

Updating the mean: After Step 1, each coordinate \( C_i \) of the average sequence has a set of coordinates \( \{p_{1i}, \ldots, p_{ni}\} \) associated to it. The process of updating \( C \) consists in updating each coordinate of the average sequence as the barycenter this set of coordinates. Since the average sequence is associated to \( N \) sequences, its \( T \) coordinates are, overall, associated to \( N \cdot T \) coordinates, i.e., all coordinates of sequences of \( S \). The update step will thus have a time complexity of \( \Theta(N \cdot T) \).

Overall complexity: Because DBA is an iterative process, let us note \( I \) the number of iterations. The time complexity of the averaging process of \( N \) sequences, each one containing \( T \) coordinates, is thus

\[
\Theta(DBA) = \Theta(I(N \cdot T^2 + N \cdot T)) = \Theta(I \cdot N \cdot T^2)
\]

Comparison with PSA and NLAAF: To compute an average sequence from two sequences, PSA and NLAAF require \( \Theta(T) \) operations. However, after having computed this average sequence, it has to be shorten to the length of averaged sequences. The classical averaging process used is uniform scaling which requires \( \Theta(T + 2T + 3T + \cdots + T^2) = \Theta(T^3) \). The computation of the average sequence of two sequences requires \( \Theta(T' + T^2 + T) = \Theta(T^3) \). The overall NLAAF averaging of a set of \( N \) sequences then requires

\[
\Theta((N-1) \cdot (T' + T^2 + T) + N^2 \cdot T^2) = \Theta(N \cdot T^2 + N^2 \cdot T^2)
\]

As \( I < T \), the time complexity of DBA is thus smaller than PSA and NLAAF ones.

4.4. Experiments on standard datasets

Evaluating an average sequence is not a trivial task. No ground truth of the expected sequence is available and we saw in Section 3 that many meanings are covered by the “average” (or consensus) notion. Most experimental and theoretical works use the WGSS to quantify the relative quality of an averaging technique. Thus, to assess the performance of DBA by comparison with existing averaging methods, we compare DBA to NLAAF and PSA in terms of WGSS over datasets from the UCR classification/clustering archive [19] (see Fig. 4).

Let us briefly remind what NLAAF and PSA are. NLAAF works by placing each coordinate of the average sequence of two sequences, as the center of each association created by DTW. PSA associates each connected component of the graph (formed by the coupling between two sequences) to a coordinate of the average sequence. Moreover, to average \( N \) sequences, it uses a hierarchical method to average at first closest sequences.

Experimental settings. To make these experiments reproducible, we provide here the details about our experimental settings:

- all programs are implemented in Java and run on a Core 2 Duo processor running at 2.4GHz with 3 GB of RAM;
- the distance used between two coordinates of sequences is the squared Euclidean distance. As the square function is a strictly increasing function on positive numbers, and because we only use comparisons between distances, it is unnecessary to compute square roots. The same optimization has been used in [31], and seems rather common;
- sequences have been normalized with Z-score: for each sequence, the mean \( \pi \) and standard deviation \( \sigma \) of the coordinate values are computed, and each coordinate \( y_i \) is replaced by

\[
y'_i = \frac{y_i - \pi}{\sigma}
\]
- We compare DBA to NLAAF on all datasets from [19] and to PSA using results given by [21];
- As we want to test the capacity of DBA to minimize the WGSS, and because we do not focus on supervised methods, we put all sequences from both train and test dataset together;
- For each set of sequences under consideration, we report the inertia under DBA and other averaging techniques. To provide quantitative evaluation, we indicate the ratio between the inertia with respect to the average computed by DBA and those computed by NLAAF and/or PSA (see the tables). To provide an overall evaluation, the text also sometimes mentions geometric averages of these ratios.

**Intraclass inertia comparison**: In this first set of experiments, we compute an average for each class in each dataset. Table 1 shows the global WGSS obtained for each dataset. We notice that, for all datasets, DBA reduces/improves the intraclass inertia. The geometric average of the ratios shown in Table 1 is 65%.

Table 2 shows a comparison between results of DBA and PSA. Here again, for all results published in [21], DBA outperforms PSA, with a geometric average of inertia ratios of 65%.

![Sample instances from the test datasets](image.png)

*Fig. 4.* Sample instances from the test datasets. One time series from each class is displayed for each dataset.
Actually, such a decreases of inertia show that old averaging methods could not be seriously considered for machine learning use.

**Global dataset inertia**: In the previous paragraph, we computed an average for each class in each dataset. In this paragraph, the goal is to test robustness of DBA with more data variety. Therefore, we average all sequences of each dataset. This means that only one average sequence is computed for a whole dataset. Thats way, we compute the global dataset inertia under DTW with NLAAF and DBA, to compare their capacity to summarize mixed data.

As can be seen in Table 3, DBA systematically reduces/improves the global dataset inertia with a geometric average ratio of 56%. This means that DBA not only performs better than NLAAF (Table 1), but also more robust to diversity.

### 4.5. Impact of initialisation

DBA is deterministic once the initial average sequence $C$ is chosen. It is thus important to study the impact of the choice of the initial mean on the results of DBA. When used with $K$-MEANS, this choice must be done at each iteration of the algorithm, for example by taking as the initialisation the average sequence $C$ obtained at the previous iteration. However, DBA is not limited to this context.

We have seen in Section 4.2 that two aspects of initialisation have to be evaluated empirically: first, the length of the initial average sequence, and second the values of its coordinates. We have designed three experiments on some of the datasets...
The best scores are shown in boldface.

### Table 1
Comparison of intraclass inertia under DTW between NLAAF and DBA.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Intraclass inertia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLAAF</td>
</tr>
<tr>
<td>50words</td>
<td>11.98</td>
</tr>
<tr>
<td>Adiac</td>
<td>0.21</td>
</tr>
<tr>
<td>Beef</td>
<td>29.90</td>
</tr>
<tr>
<td>CFB</td>
<td>14.25</td>
</tr>
<tr>
<td>Coffee</td>
<td>0.72</td>
</tr>
<tr>
<td>ECG200</td>
<td>11.34</td>
</tr>
<tr>
<td>FaceAll</td>
<td>17.77</td>
</tr>
<tr>
<td>FaceFour</td>
<td>34.46</td>
</tr>
<tr>
<td>Fish</td>
<td>1.35</td>
</tr>
<tr>
<td>GunPoint</td>
<td>7.24</td>
</tr>
<tr>
<td>Lighting2</td>
<td>194.07</td>
</tr>
<tr>
<td>Lighting7</td>
<td>48.25</td>
</tr>
<tr>
<td>OliveOil</td>
<td>0.018261</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>53.03</td>
</tr>
<tr>
<td>Synthetic control</td>
<td>9.71</td>
</tr>
<tr>
<td>Trace</td>
<td>1.65</td>
</tr>
<tr>
<td>Two patterns</td>
<td>9.19</td>
</tr>
<tr>
<td>Wafer</td>
<td>54.66</td>
</tr>
<tr>
<td>Yoga</td>
<td>40.07</td>
</tr>
</tbody>
</table>

### Table 2
Comparison of intraclass inertia under DTW between PSA and DBA.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Intraclass inertia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSA</td>
</tr>
<tr>
<td>Beef</td>
<td>25.65</td>
</tr>
<tr>
<td>Coffee</td>
<td>0.72</td>
</tr>
<tr>
<td>ECG200</td>
<td>9.16</td>
</tr>
<tr>
<td>FaceFour</td>
<td>33.68</td>
</tr>
<tr>
<td>Synthetic control</td>
<td>10.97</td>
</tr>
<tr>
<td>Trace</td>
<td>1.65</td>
</tr>
</tbody>
</table>

The best scores are shown in boldface.

### Table 3
Comparison of global dataset inertia under DTW between NLAAF and DBA.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Global dataset inertia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NLAAF</td>
</tr>
<tr>
<td>50words</td>
<td>51642</td>
</tr>
<tr>
<td>Adiac</td>
<td>647</td>
</tr>
<tr>
<td>Beef</td>
<td>3154</td>
</tr>
<tr>
<td>CFB</td>
<td>21306</td>
</tr>
<tr>
<td>Coffee</td>
<td>61.60</td>
</tr>
<tr>
<td>ECG200</td>
<td>2190</td>
</tr>
<tr>
<td>FaceAll</td>
<td>72356</td>
</tr>
<tr>
<td>FaceFour</td>
<td>6569</td>
</tr>
<tr>
<td>Fish</td>
<td>658</td>
</tr>
<tr>
<td>GunPoint</td>
<td>1525</td>
</tr>
<tr>
<td>Lighting2</td>
<td>25708</td>
</tr>
<tr>
<td>Lighting7</td>
<td>14388</td>
</tr>
<tr>
<td>OliveOil</td>
<td>2.24</td>
</tr>
<tr>
<td>OSULeaf</td>
<td>30293</td>
</tr>
<tr>
<td>Synthetic control</td>
<td>17939</td>
</tr>
<tr>
<td>Trace</td>
<td>22613</td>
</tr>
<tr>
<td>Two patterns</td>
<td>122471</td>
</tr>
<tr>
<td>Wafer</td>
<td>471376</td>
</tr>
<tr>
<td>Yoga</td>
<td>136547</td>
</tr>
</tbody>
</table>

The best scores are shown in boldface.

1. The first experiment starts with randomly generated sequences, of lengths varying from 1 to 2T, where T is the length of the sequences in the data set. Once the length is chosen, the coordinates are generated randomly with a normal distribution of mean zero and variance one. The curves on Fig. 5 show the variation of inertia with the length.

2. Because the previous experiment shows that the optimal inertia is attained with an initial sequence of length in the order of T, we have repeated the computation 100 times with different, randomly generated initial sequences of length T: the goal of this experiment is to measure how stable this heuristic is. Green triangles on Fig. 5 show the inertias with the different random initialisations of length T.

3. Because priming DBA with a sequence of length T seems to be an adequate choice, we have tried to replace the randomly generated sequence with one drawn (randomly) from the dataset. We have repeated this experiment 100 times. Red triangles on Fig. 5 show the inertias with the different initial sequences from the dataset.

Our experiments on the choice of the initial average sequence lead to two main conclusions. First, an initial average of length T (the length of the data sequences) is the most appropriate. It almost always leads to the minimal inertia. Second, randomly choosing an element of the dataset leads to the least inertia on almost all cases. Using some data to prime an iterative algorithm is part of the folklore. DBA is another case where it performs well. We have used this strategy in all our experiments with DBA.

Moreover, in order to compare the impact of the initialisation on DBA and on NLAAF, we perform 100 computations of the average on few datasets (because of computing times), by choosing a random sequence from the dataset as an initialisation of DBA. As NLAAF is sensitive to the initialisation too, the same method is followed, in order to compare results. Fig. 6 presents the mean and standard deviation of the final inertia. The results of DBA are not only better than NLAAF (shown on the left part of Fig. 6), but the best inertia obtained by NLAAF is even worse as the worst inertia obtained by DBA (see the table on the right of Fig. 6).

### 4.6 Convergence of the iterative process

As explained previously, DBA is an iterative process. It is necessary, once the average is computed, to update it several times. This has the property of letting DTW refine its associations. Fig. 7(a) presents the average convergence of the iterative process on the 50words dataset. This dataset is diverse enough (it contains 50 different classes) to test the robustness of the convergence of DBA.

Besides the overall shape of the convergence curve in Fig. 7(a), it is important to note that in some cases, the convergence can be uneven (see Fig. 7(b) for an example). Even if this case is somewhat unusual, one has to keep in mind that DTW makes nonlinear distortions, which cannot be predicted. Consequently, the convergence of DBA, based on alignments, cannot be always smooth.

### 5 Optimization of the mean by length shortening

We mentioned in Section 3.4 that algorithms such as NLAAF need to reduce the length of a sequence. Actually, this problem is more global and concerns the scaling problem of a sequence under time warping. Many applications working with subsequences or...
even with different resolutions require a method to uniformly scale a sequence to a fixed length. This kind of methods is generally called uniform scaling; further details about its inner working can be found in [31].

Unfortunately, the use of uniform scaling is not always coherent in the context of DTW, which computes non-uniform distortions. To avoid the use of uniform scaling with DTW, as done in [31,15,21], we propose here a new approach specifically designed for DTW. It is called adaptive scaling, and aims at reducing the length of a sequence with respect to one or more other sequences. In this section, we first recall the definition of uniform scaling, then we detail the proposed approach and finally its complexity is studied and discussed.

5.1. Uniform scaling

Uniform scaling is a process that reduces the length of a sequence with regard to another sequence. Let A and B be two sequences. Uniform scaling finds the prefix $A_{sub}$ of A such that, scaled up to B, $DTW(A_{sub}, B)$ is minimal. The subsequence $A_{sub}$ is defined by

$$A_{sub} = \arg \min_{i \in [1,T]} DTW(A_{1:i}, B)$$

(11)

uniform scaling has two main drawbacks: one is directly linked to the method itself, and one is linked to its use with DTW. First, while uniform scaling considers a prefix of the sequence (i.e., a subsequence), the representativeness of the resulting mean using such a reduction process can be discussed. Second, uniform scaling is a uniform reduction process, whereas DTW makes non-uniform alignments.
5.2. Adaptive scaling

We propose to make the scaling adaptive. The idea of the proposed adaptive scaling process is to answer to the following question: “How can one remove a point from a sequence, such as the distance to another sequence does not increase much?” To answer this question, adaptive scaling works by merging the two closest successive coordinates.

To explain how adaptive scaling works, let us start with a simple example. If two consecutive coordinates are identical, they can be merged. DTW is able to stretch the resulting coordinate and so recover the original sequence. This fusion process is illustrated in Fig. 8. Note that in this example, DTW gives the same score in Fig. 8(a) as in Fig. 8(b).

This article focuses on finding an average sequence consistent with DTW. Performances of DBA have been demonstrated on an average sequence of length arbitrarily fixed to T. In this context, the question is to know if this average sequence can be shortened, without making a less representative mean (i.e., without increasing inertia). We show in the first example, that the constraint on inertia is respected. Even if uniform scaling could be used to reduce the length of the mean, an adaptive scaling would give
better results, because DTW is able to make deformations on the time axis. Adaptive scaling is described in Algorithm 6.

**Algorithm 6. Adaptive Scaling**

```
Require: A = \langle A_1, \ldots, A_T \rangle
while Need to reduce the length of A
\( (i,j) \leftarrow \) successive coordinates with minimal distance
merge \( A_i \) with \( A_j \)
end while
return A
```

Let us now explain how the inertia can also decrease by using adaptive scaling. Fig. 9 illustrates the example used below. Imagine now that the next to last coordinate \( C_T/C_0 \) of the average sequence is perfectly aligned with the last \( z \) coordinates of the \( N \) sequences of \( S \). In this case, the last coordinate \( C_T \) of the average sequence must still be, at least, linked to all last coordinates of the \( N \) sequences of \( S \). Therefore, as the next to last coordinate was (in this example) perfectly aligned, aligning the last coordinate will increase the total inertia. This is why adaptive scaling is not only able to shorten the average sequence, but also to reduce the inertia. Moreover, by checking the evolution of inertia after each merging, we can control this length reduction process, and so guarantee the improvement of inertia. Thus, given the resulting mean of DBA, coordinates of the average sequence can be successively merged as long as inertia decreases.

5.3. Experiments

Table 4 gives scores obtained by using adaptive scaling after the DBA process on various datasets. It shows that adaptive scaling alone always reduces the intraclass inertia, with a geometric average of 84%. Furthermore, the resulting average sequences are much shorter, by almost two thirds. This is an interesting idea of the minimum necessary length for representing a tim behaviour.

In order to demonstrate that adaptive scaling is not only designed for DBA, Fig. 10 shows its performances as a reducing process in NLAAF. Adaptive scaling is here used to reduce the length of a temporary pairwise average sequence (see Section 3.4). Fig. 10 shows that adaptive scaling used in NLAAF leads to scores similar to the ones achieved by PSA.

5.4. Complexity

Adaptive scaling (AS) consists in merging the two closest successive coordinates in the sequence. If we know ahead of time the number \( K \) of coordinates that must be merged, for example using AS for NLAAF to be scalable, it requires a time complexity of \( \Theta(K \cdot T) \), and is thus tractable.

One could, however, need to guarantee that adaptive scaling does not merge too many coordinates. That is why we suggest to control the dataset inertia, by computing it and by stopping the adaptive scaling process if inertia increases. Unfortunately, computing the dataset inertia under DTW takes \( \Theta(N \cdot T^2) \). Its complexity may prevent its use in some cases.

Nevertheless, we give here some interesting cases, where the use of adaptive scaling could be beneficial, because having shorter sequences means spending less time in computing DTW. In databases, the construction of indexes is an active research domain. The aim of indexes is to represent data in a better way while being fast queryable. Using adaptive scaling could be used here, because it correctly represents data and reduces the DTW complexity for further queries. The construction time of the index is here negligible compared to the millions of potential queries.
Another example is (supervised or unsupervised) learning, where the learning time is often negligible compared to the time spent in classification. Adaptive scaling could be very useful in such contexts, and can be seen in this case as an optimization process, more than an alternative to uniform scaling.

6. Application to clustering

Even if DBA can be used in association to DTW out of the context of K-MEANS (and more generally out of the context of clustering), it is interesting to test the behaviour of DBA with K-MEANS because this algorithm makes a heavy use of averaging. Most clustering techniques with DTW use the K-MEANs algorithm, which does not require any computation of an average [15–18]. However, K-MEANs has some problems related to its use of the notion of median: K-MEANs is not idempotent, which means that its results can oscillate. Whereas DTW is one of the most used similarity on time series, it was not possible to use it reliably with well-known clustering algorithms.

To estimate the capacity of DBA to summarize clusters, we have tested its use with the K-MEANS algorithm. We present here different tests on standard datasets and on a satellite image time series. Here again, result of DBA are compared to those obtained with NLAAF.

6.1. On UCR datasets

Table 5 shows, for each dataset, the global WGSS resulting from a K-MEANS clustering. Since K-MEANS requires initial centers, we place randomly as many centers as there are classes in each dataset. As shown in the table, DBA outperforms NLAAF in all cases except for Fish and OliveOil datasets. Including these exceptions, the inertia is reduced with a geometric average of 72%.

Let us try to explain the seemingly negative results that appear in Table 5. First, on OliveOil, the inertia over the whole dataset is very low (i.e., all sequences are almost identical; see Fig. 4), which makes it difficult to obtain meaningful results. The other particular dataset is Fish. We have seen, in Section 4.4, that DBA outperforms NLAAF provided it has meaningful clusters to start with. However, here, the K-MEANS algorithm tries to minimize this inertia in grouping elements in “centroid form”. Thus, if clusters to identify are not organized around “centroids”, this algorithm may converge to any local minima. In this case, we explain this exceptional behaviour on Fish as due to non-centroid clusters. We have shown in Section 4.4 that, if sequences are averaged per class, DBA outperforms all results, even those of these two datasets. This means that these two seemingly negative results are linked to the K-MEANS algorithm itself, that converges to a less optimal local minimum even though the averaging method is better.

6.2. On satellite image time series

We have applied the K-MEANS algorithm with DTW and DBA in the domain of satellite image time series analysis. In this domain, each dataset (i.e., sequence of images) provides thousands of relatively short sequences. This kind of data is the opposite of sequences commonly used to validate time sequences analysis. Thus, in addition to evaluate our approach on small datasets of long sequences, we test our method on large datasets of short sequences.

Our data are sequences of numerical values, corresponding to radiometric values of pixels from a satellite image time series (SITS). For every pixel, identified by its coordinates \((x, y)\), and for a sequence of images \(I_1, \ldots, I_n\), we define a sequence as \(\langle I_1(x, y), \ldots, I_n(x, y)\rangle\). That means that a sequence is identified by coordinates \(x\) and \(y\) of a pixel (not used in measuring similarity), and that the values of its coordinates are the vectors of radiometric values of this pixel in each image. Each dataset contains as many sequences as there are pixels in one image.

We have tested DBA on one SITS of size 450 x 450 pixels, and of length 35 (corresponding to images sensed between 1990 and 2006). The whole experiment thus deals with 202,500 sequences of length 35 each, and each coordinate is made of three radiometric values. This SITS is provided by the KALIDES database [32] (see Fig. 11 for an extract).

We have applied the K-MEANS algorithm on this dataset, with a number of clusters of 10 or 20, chosen arbitrarily. Then we computed the sum of intraclass inertia after five or 10 iterations. Table 6 summarizes results obtained with different parameters.

We can note that scores (to be minimized) are always ordered as NLAAF > DBA > DBA + AS, which tends to confirm the behaviour of DBA and adaptive scaling. As we could expect, adaptive scaling
also provided a symbolic description of this particular behaviour. Combining DTW growth behaviour (appearance of new buildings and urban densification). This is illustrated on Fig. 12. Combining DTW and one or two coordinates can be sufficient to represent the time series. This sequence corresponds to the thematical behaviour of the urban area.

Let us try to explain why the results of DBA are close to those of NLAAF. One can consider that when clusters are close to each other, then the improvement is reduced. The most likely explanation is that, by using so short sequences, DTW has not much material to work on, and that the alignment it finds early have little chance to change over the successive iterations. In fact, the shorter the sequences are, the closer DTW is to the Euclidean distance. Moreover, NLAAF makes less errors when there is no re-alignment between sequences. Thus, when sequences are small, NLAAF makes less errors. For that reason, even if DBA is in this case again better than NLAAF, the improvement is smaller.

Let us now explain why adaptive scaling is so useful here. In SITS, there are several evolutions which can be considered as random perturbations. Thus, the mean may not need to represent these perturbations, and we think that shorter means are sometimes better, because they can represent a perturbed constant subsequence by a single coordinate. Actually, this is often the case in SITS. As an example, a river can stay almost the same over a SITS and one or two coordinates can be sufficient to represent the evolution of such an area.

From a thematic point of view, having an average for each cluster of radiometric evolution sequences highlights and describes typical ground evolutions. For instance, the experiment just described provides a cluster representing a typical urban growth behaviour (appearance of new buildings and urban densification). This is illustrated on Fig. 12. Combining DTW with DBA has led to the extraction of urbanizing zones, but has also provided a symbolic description of this particular behaviour. Using euclidean distance instead of DTW, or any other averaging technique instead of DBA, has led to inferior results. Euclidean distance was expected to fail somehow, because urbanization has gone faster in some zones than in others, and because the data sampling is nonuniform. The other averaging methods have also failed to produce meaningful results, probably because of the intrinsic difficulty of the data (various sensors, irregular sampling, etc.), which leads to difficulty separable objects. In such cases, less precise averaging tends to blur cluster boundaries.

### 7. Conclusion

The DTW similarity measure is probably the most used and useful tool to analyse sets of sequences. Unfortunately, its applicability to data analysis was reduced because it had no suitable associated averaging technique. Several attempts have been made to fill the gap. This article proposes a way to classify these averaging methods. This “interpretive lens” permits to understand where existing techniques could be improved. In light of this contextualization, this article defines a global averaging method, called Dtw barycenter averaging (DBA). We have shown that DBA achieves better results on all tested datasets, and that its behaviour is robust.

The length of the average sequence is not trivial to choose. It has to be as short as possible, but also sufficiently long to represent the data it covers. This article also introduces a shortening technique of the length of a sequence called adaptive scaling. This process is shown to shorten the average sequence in adequacy to DTW and to the data, but also to improve its representativity.

Having a sound averaging algorithm lets us apply clustering techniques to time series data. Our results show again a significant improvement in cluster inertia compared to other techniques, which certainly increases the usefulness of clustering techniques.

Many application domains now provide time-based data and need data mining techniques to handle large volumes of such data. DTW provides a good similarity measure for time series, and DBA complements it with an averaging method. Taken together, they constitute a useful foundation to develop new data mining systems for time series. For instance, satellite imagery has started to produce satellite image time series, containing millions of sequences of multi-dimensional radiometric data. We have briefly described preliminary experiments in this domain.

We believe this work opens up a number of research directions. First, because it is, as far as we know, the first global approach to the problem of averaging a set of sequences, it raises interesting questions on the topology of the space of sequences, and on how the mean relates to the individual sequences.

Regarding Dtw barycenter averaging proper, there are still a few aspects to be studied. One aspect could be the choice of the initial sequence where sequences to be averaged do not have the same length. Also we have provided an empirical analysis of the rate of convergence of the averaging process. More theoretical or empirical work is needed to derive a more robust strategy, able to adjust the number of iterations to perform. An orientation of this work could be the study of the distribution of coordinates contributing to a coordinate of the average sequence.

Adaptive scaling has important implications on performance and relevance. Because of its adaptive nature, it ensures that average sequences have “the right level of detail” on appropriate sequence segments. It currently considers only the coordinates of the average sequence. Incorporating averaged sequences may lead to a more precise scaling, but would require more computation time. Finding the right balance between cost and precision requires further investigation.

When combining DBA with adaptive scaling, e.g., when building reduced average sequences, we have often noticed that...
they provide short summaries, that are at the same time easy to visualize and truly representative of the underlying phenomenon. For instance, the process of length reduction builds an average sequence around the major states of the data. It thus provides a sampling of the dataset built from the data themselves. Exploiting and extending this property is a promising research direction.

References


Francois Petitjean is a Ph.D. student in Computer Science at the University of Strasbourg, France. His research interests include complex time series clustering and remote sensing image analysis.

Alain Ketterlin holds a Ph.D. in Computer Science from the University Louis Pasteur, Strasbourg, France. Since 1995, he is Assistant Professor at University of Strasbourg. His research interests include data mining and remote sensing image processing.

Pierre Cançarski received his Ph.D. degree in 1989 and his Habilitation in 2007 in Computer Science from the University Louis Pasteur, Strasbourg, France. He is currently an Associate Professor at the Department of Computer Science from the University of Strasbourg. His current research interests include collaborative multistratified clustering with applications to complex data mining and remote sensing image analysis.