An Average-Compress Algorithm for the Sample Mean Problem under Dynamic Time Warping

Brijnesh Jain, Vincent Froese, and David Schultz
Technische Universität Berlin, Germany
e-mail: brijnesh.jain@gmail.com

Computing a sample mean of time series under dynamic time warping (DTW) is NP-hard. Consequently, there is an ongoing research effort to devise efficient heuristics. The majority of heuristics have been developed for the constrained sample mean problem that assumes a solution of predefined length. In contrast, research on the unconstrained sample mean problem is underdeveloped. In this article, we propose a generic average-compress (AC) algorithm for solving the unconstrained problem. The algorithm alternates between averaging (A-step) and compression (C-step). The A-step takes an initial guess as input and returns an approximation of a sample mean. Then the C-step reduces the length of the approximate solution. The compressed approximation serves as initial guess of the A-step in the next iteration. The purpose of the C-step is to direct the algorithm to more promising solutions of shorter length. The proposed algorithm is generic in the sense that any averaging and any compression method can be used. Experimental results show that the AC algorithm substantially outperforms current state-of-the-art algorithms for time series averaging.

1. Introduction

Time series such as stock prices, climate data, energy usages, sales, biomedical measurements, and biometric data are sequences of time-dependent observations that often vary in temporal dynamics, that is in length, speed, and shifts in phase. For example, the same word can be uttered with different speaking speeds. Similarly, monthly temperature or precipitation extremes of certain regions can differ in duration and may occur out of phase for a period of a few weeks.

To account for temporal variations in proximity-based time series mining, the dynamic time warping (DTW) distance is often the preferred choice of proximity measure [1, 3, 4]. An intricate problem in DTW-based time series mining is time series averaging. The problem consists in finding a typical representative that summarizes a sample of time series. Different forms of time series averaging have been applied to improve nearest neighbor classifiers [14, 19], to accelerate similarity search [23], and to formulate k-means clustering in DTW spaces [12, 19, 22].

A standard approach to time series averaging under DTW is based on an idea by Fréchet [11]: Suppose that \( S = \{x_1, \ldots, x_N\} \) is a sample of \( N \) time series. A sample mean of \( S \) is any time series \( \mu \) that minimizes the Fréchet function

\[
F: \mathcal{U} \rightarrow \mathbb{R}, \quad z \mapsto \frac{1}{N} \sum_{i=1}^{N} \text{dtw}(z, x_i)^2,
\]

where \( \text{dtw}(x, y) \) is the DTW-distance and \( \mathcal{U} \) is a set of time series of finite length. The search space \( \mathcal{U} \) typically takes two forms: (i) \( \mathcal{U} \) is the set of all time series of finite length and (ii) \( \mathcal{U} \) is the set of all time series of length \( n \). We refer to (i) as the unconstrained and to (ii) as the constrained sample mean problem.

A sample mean is guaranteed to exist in either case but may not be unique [13]. In addition, computing a sample mean is NP-hard [7]. Consequently, there is an ongoing research on devising heuristics for
minimizing the Fréchet function. Most contributions focus on devising and applying heuristics for the constrained sample mean problem. State-of-the-art algorithms are stochastic subgradient methods [21], majorize-minimize algorithms [12, 13], and soft-DTW [10]. In contrast, only few work has been done for solving the unconstrained sample mean problem. One algorithm is an (essentially optimal) dynamic program that exactly solves the unconstrained problem in exponential time [6]. A second algorithm is a heuristic, called adaptive DBA (ADBA) [17]. This algorithm uses a majorize-minimize algorithm (DBA) as a base-algorithm and iteratively refines subsequences to improve the solution quality.

Currently, there is no clear understanding of the characteristic properties, advantages, and disadvantages of both types of sample means. We can approach the sample mean problem theoretically and empirically. A prerequisite for an empirical approach towards a better understanding of the sample mean problem are sufficiently powerful averaging algorithms. Compared to the constrained sample mean problem, algorithms for the unconstrained sample mean problem are underdeveloped.

In this work, we propose an average-compress (AC) algorithm for the unconstrained sample mean problem. The algorithm repeatedly alternates between averaging (A-step) and compression (C-step). The A-step requires a time series as initial guess, minimizes the Fréchet function, and returns an approximate solution as output. The C-step compresses the approximation of the A-step to obtain an improved solution. The compressed solution of the C-step serves as initial guess of the A-step in the next iteration. Compression is motivated by empirical observations that an unconstrained sample mean is typically shorter than the sample time series to be averaged [6]. In principle, any averaging algorithm and any compression method can be applied. Here, we propose a compression method that minimizes the squared DTW error between original and compressed time series. Empirical results suggest that the AC scheme substantially outperforms state-of-the-art heuristics including ADBA.

This article is organized as follows: Section 2 describes the AC algorithm. In Section 3 we present and discuss empirical results. Finally, Section 4 concludes with a summary of the main findings and an outlook for future research.

2. Average-Compress Algorithm

In this section, we develop an average-compress (AC) algorithm for approximately solving the unconstrained sample mean problem. To this end, we first introduce the DTW-distance (Section 2.1), the concept of a sample mean under DTW (Section 2.2), and compressions (Section 2.3). Thereafter, we describe the AC algorithm in Section 2.4.

2.1. Dynamic Time Warping

For a given $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. A time series is a sequence $x = (x_1, \ldots, x_n)$ with elements $x_i \in \mathbb{R}$ for all $i \in [n]$. We denote the length of time series $x$ by $|x| = n$, the set of time series of length $n$ by $T_n$, and the set of all time series of finite length by $T$. Consider the $(m \times n)$-grid defined as

$$[m] \times [n] = \{(i, j) : i \in [m], j \in [n]\}.$$ 

A warping path of order $m \times n$ and length $\ell$ is a sequence $p = (p_1, \ldots, p_{\ell})$ through the grid $[m] \times [n]$ consisting of $\ell$ points $p_l = (i_l, j_l) \in [m] \times [n]$ such that

1. $p_1 = (1, 1)$ and $p_{\ell} = (m, n)$
2. $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for all $l \in [\ell - 1]$.

The first condition is the boundary condition and the second condition is the step condition of the DTW-distance. We denote the set of all warping paths of order $m \times n$ by $\mathcal{P}_{m,n}$. Suppose that $p = (p_1, \ldots, p_{\ell}) \in \mathcal{P}_{m,n}$ is a warping path with points $p_l = (i_l, j_l)$ for all $l \in [\ell]$. Then $p$ defines an expansion (or warping) of the time series $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_n)$ by the length-$\ell$ time series $\phi_p(x) = (x_{i_1}, \ldots, x_{i_\ell})$ and $\psi_p(y) = (y_{j_1}, \ldots, y_{j_\ell})$. By definition, the length $\ell$ of a warping path satisfies $\max(m, n) \leq \ell \leq m + n$.

The cost of warping time series $x$ and $y$ along warping path $p$ is defined by

$$C_p(x, y) = \|\phi_p(x) - \psi_p(y)\|_2^2 = \sum_{(i, j) \in p} (x_i - y_j)^2,$$
where $\| \cdot \|$ denotes the Euclidean norm and $\phi_p$ and $\psi_p$ are the expansions defined by $p$. The DTW-distance of $x$ and $y$ is
\[
dtw(x,y) = \min \left\{ \sqrt{C_p(x,y)} : p \in \mathcal{P}_{m,n} \right\}.
\]
A warping path $p$ with $C_p(x,y) = \dtw^2(x,y)$ is called an optimal warping path of $x$ and $y$. By definition, the DTW-distance minimizes the Euclidean distance between all possible expansions that can be derived from warping paths. Computing the DTW-distance and deriving an optimal warping path is usually solved via dynamic programming [20, 25].

2.2. Sample Means under DTW
Let $\mathcal{S} = \{x_1, \ldots, x_N\}$ be a sample of $N$ time series $x_i \in \mathcal{T}$. Note that $\mathcal{S}$ is a multiset that allows multiple instances of the same elements. A sample mean of $\mathcal{S}$ is any time series that minimizes the Fréchet function [11]
\[
F : \mathcal{U} \to \mathbb{R}, \quad z \mapsto \frac{1}{N} \sum_{i=1}^{N} \dtw(z,x_i)^2,
\]
where $\mathcal{U} \subseteq \mathcal{T}$ is a subset of time series. The value $F(z)$ is the Fréchet variation of sample $\mathcal{S}$ at $z$. The infimum $\inf_z F(z)$ serves as a measure of variability of $\mathcal{S}$. Here, the search space $\mathcal{U}$ takes one of the following two forms: (i) $\mathcal{U} = \mathcal{T}$ and (ii) $\mathcal{U} = \mathcal{T}_m$. We refer to (i) as the unconstrained and to (ii) as the constrained sample mean problem. Note that the constrained formulation only restricts the length of the candidate solutions, whereas there is no length restriction on the sample time series to be averaged.

A sample mean exists in either case but is not unique in general [13]. This result implies that $F$ attains its infimum (has a unique minimum). However, computing a sample mean is NP-hard [7]. The implication is that we often need to resort to heuristics that return useful solutions within acceptable time.

We briefly describe two state-of-the-art algorithms for the constrained sample mean problem: a stochastic subgradient method (SSG) [21] and a majorize-minimize algorithm (DBA) [12, 18]. For a detailed description of both algorithms, we refer to [21].

To present the update rule of both algorithms in a compact form, we introduce the notions of warping and valence matrix as proposed by [21]. Suppose that $p \in \mathcal{P}_{m,n}$ is a warping path. The warping matrix of $p$ is the zero-one matrix $W = (w_{ij}) \in \{0,1\}^{m \times n}$ with elements
\[
w_{ij} = \begin{cases} 1 : (i,j) \in p \\ 0 : \text{otherwise} \end{cases}.
\]
The valence matrix of warping path $p$ is the diagonal matrix $V = (v_{ii}) \in \mathbb{N}^{m \times m}$ with positive diagonal elements
\[
v_{ii} = \sum_{j=1}^{n} w_{ij}.
\]
Suppose that $z$ and $x$ are time series of length $|z| = m$ and $|x| = n$. Then $W$ warps $x$ onto the time axis of $z$. Each diagonal element $v_{ii}$ of $V$ counts how many elements of $x$ are warped to element $z_i$.

**Stochastic Subgradient Algorithm.** Subgradient methods for time series averaging have been proposed by [21]. Algorithm 1 outlines a vanilla version of the SSG algorithm with constant learning rate $\eta$. In practice, more sophisticated stochastic subgradient variants such as Adam [16] are preferred. The input of Algorithm 1 are a learning rate $\eta$, a length-parameter $m$ of the constrained search space, and a sample $x_1, \ldots, x_N$ of time series to be averaged. The output is a time series with lowest Fréchet variation that has been encountered during optimization.

3
Algorithm 1 Stochastic Subgradient Method

1: procedure SSG(η, m, x₁, ..., xₙ)  
2: initialize solution z ∈ Tₘ 
3: initialize best solution z* = z  
4: repeat  
5: reshuffle order of sample time series  
6: for i ← 1 to N do  
7: compute optimal warping path pᵢ of z and xᵢ  
8: compute valence matrix Vᵢ of pᵢ  
9: compute warping matrix Wᵢ of pᵢ  
10: update solution z according to the rule  
11: z ← z − 2η(∑ᵢ Vᵢz − Wᵢxᵢ)  
12: record best solution z* = argmin{F(z*), F(z)}  
13: until termination  
14: return z

Majorize-Minimize Algorithm. Majorize-minimize algorithms for time series averaging have been proposed in the 1970s mainly by Rabiner and his co-workers with speech recognition as the primary application [19, 27]. The early approaches fell largely into oblivion and were successively rediscovered, consolidated, and improved in a first step by Abdulla et al. [2] in 2003 and then finalized in 2008 by Hautamaki et al. [12]. In 2011, Petitjean et al. [18] reformulated, explored, and popularized the majorize-minimize algorithm by Hautamaki et al. [12] under the name DTW Barycenter Averaging (DBA).

Algorithm 2 describes the DBA algorithm. It takes a length-parameter m and a sample of time series as input and returns the candidate solution of the last iteration as output. The DBA algorithm terminates after a finite number of iterations in a local minimum of the Fréchet function [21]. □

Algorithm 2 DBA Algorithm

1: procedure DBA(m, x₁, ..., xₙ)  
2: initialize solution z ∈ Tₘ 
3: repeat  
4: //*** Majorize  
5: for i ← 1 to N do  
6: compute optimal warping path pᵢ of z and xᵢ  
7: compute valence matrix Vᵢ of pᵢ  
8: compute warping matrix Wᵢ of pᵢ  
9: //*** Minimize  
10: update solution z according to the rule  
11: z ← (∑ᵢ Vᵢ)⁻¹(∑ᵢ Wᵢxᵢ) 
12: until termination  
13: return z

2.3. Compressions

Let x ∈ T be a time series of length n. A compression of x is a time series x’ of length m ≤ n that maintains some desirable problem-specific properties of x. By definition, x is also a compression of itself. A compression chain of x is a sequence C(x) = (x’₁, ..., x’ₖ) of k ∈ [n] compressions x’ᵢ of x such that

1 ≤ |x’₁| < |x’₂| < ... < |x’ₖ| ≤ n.

There are numerous compression methods such as principal component analysis, discrete Fourier transform, discrete wavelet transform, and many more. Here, we consider two simple methods: adaptive scaling (ADA) and minimum squared DTW error (MSE).

Adaptive Scaling. Algorithm 3 describes ADA. The procedure takes a time series x = (x₁, ..., xₙ) as input and returns a compression chain C(x) consisting of n compressions of x of length 1 to n. To compress a time series x’ₖ₊₁ of length k + 1 to a time series x’ₖ of length k, ADA merges two consecutive
elements with minimal distance. The merge subroutine in Algorithm 3 replaces these two consecutive time points by their average.

Finding the smallest distance in Line 4 takes \( O(|x'|) \) time. In each iteration, the length of \( x' \) is reduced by one. Thus, the complexity of computing all \( n \) compressions is \( O(n^2) \).

**Minimum Squared DTW Error Compression.** The second compression method computes a time series of a given length such that the squared DTW error is minimized. Let \( x \in \mathcal{T} \) be a time series of length \( n \) and let \( m < n \). We call each

\[
x' \in \arg\min \left\{ \text{dtw}(x, z)^2 : z \in \mathcal{T}_m \right\}
\]

an MSE compression of \( x \) of length \( m \). Observe that the MSE compression problem for \( x \) is the constrained sample mean problem of the sample \( S = \{x\} \). Algorithm 4 outlines MSE compression.

It is not hard to see that for a compression \( x' \), an optimal warping path \( p \) between \( x \) and \( x' \) warps every element of \( x \) to exactly one element of \( x' \), that is, \( \phi_p(x) = x \). Thus, we can write

\[
p = ((1, 1), \ldots, (i_1, 1), (i_1 + 1, 2), \ldots, (i_1 + i_2, 2), \ldots, (n - i_m), \ldots, (n, m)),
\]

where \( \sum_{i=1}^m i_i = n \). Let \( d_l = \sum_{j=1}^l i_j, l \in [m] \), and \( d_0 = 0 \). The squared DTW error of \( x' \) is

\[
dtw(x, x')^2 = \sum_{l=1}^m \sum_{i=d_{l-1}+1}^{d_l} (x_i - x_i')^2.
\]

MSE compression is also known as *adaptive piecewise constant approximation* [8] and as *segmentation problem* [24]. It can be solved exactly via dynamic programming in \( O(n^2m) \) time [5]. Moreover, the dynamic program allows to find all \( n \) compressions (for each length \( m = 1, \ldots, n \)) in \( O(n^3) \) time by running it once for \( m = n \) (as it is done in Algorithm 4).

Interestingly, MSE compression is also related to one-dimensional \( k \)-means clustering. To see this relationship, consider an optimal warping path between the compression \( x' \) and the original time series \( x \) as in Eq. (1). Then, the squared DTW error is minimal for \( x'_i = (x_{d_{i-1}+1} + \cdots + x_{d_i})/i_i \). Thus, finding an MSE compression \( x' \) of length \( k \) can also be seen as a one-dimensional \( k \)-means clustering problem, where every cluster consists of a consecutive subsequence of elements in \( x \). Indeed, the dynamic program in Algorithm 4 is the same as for one-dimensional \( k \)-means [26] (without previously sorting the elements in \( x \)).

To reduce the computational complexity, several heuristics and approximations for MSE compression have been proposed [8, 13, 24, 28]. Also ADA compression can be regarded as a heuristic for MSE compression since it greedily averages two consecutive elements.

To conclude, with MSE compression, we consider an exact solution method to a sound compression problem and with ADA compression, we consider a fast heuristic. Among the various heuristics, we have
Algorithm 4 MSE Compression

1: procedure MSE(x)
2:   initialize tables $D$, $C$
3:   $C[1,1] \leftarrow (x_1)$
4:   $D[1,1] \leftarrow 0$
5:   for $i \leftarrow 2$ to $|x|$ do
6:     for $m \leftarrow 1$ to $i$ do
7:       $\mu \leftarrow (\sum_{l=j}^{i} x_l) / (i - j + 1)$
8:       $d \leftarrow \sum_{l=j}^{i} (x_l - \mu)^2$
9:       if $D[j-1, m-1] + d < D[i, m]$ then
10:          $D[i, m] \leftarrow D[j-1, m-1] + d$
11:          $C[i, m] \leftarrow C[j-1, m-1]$.append($\mu$)
12:     end for
13:   $C(x) \leftarrow \{C[i, m] : 1 \leq m \leq |x|\}$
14: return $C(x)$

chosen ADA compression because it has been successfully tested for improving approximate solutions of
the constrained sample mean problem [13].

2.4. The Average-Compress Algorithm

In this section, we assemble the pieces of the previous sections and propose a generic average-compress
(AC) algorithm for approximately solving the unconstrained sample mean problem.

AC Algorithm. The AC algorithm alternates between averaging (A-step) and compression (C-step). For
this purpose, any averaging algorithm and any compression method can be used. Algorithm 5 depicts
the generic procedure. The input of the algorithm is a sample $S$ of time series and an initial guess $z \in T$.
It then repeatedly applies the following steps until termination:

1. A-step: approximate sample mean
   $$ z \leftarrow \text{average}(S, z). $$

2. C-step: compute a compression chain
   $$ C(z) \leftarrow \text{compress}(z). $$

3. Evaluate solution:
   a) Select the shortest compression $z_* \in C(z)$ such that $F(z_*) \leq F(z')$ for all $z' \in C(z)$.
   b) If $F(z_*) < F(z)$, set $z \leftarrow z_*$ and go to Step 1, otherwise terminate.

Line 7 computes the complete compression chain $C(z)$ that consists of all $|z|$ compressions of $z$ of lengths
1 to $|z|$. To accelerate the algorithm at a possible expense of solution quality, sparse compression chains
can be considered.

In the following, we explain why and under which conditions compression is useful. To simplify our
argument, we assume that AC uses an averaging algorithm for the constrained sample mean problem
(such as SSG or DBA). In this case, the length $m$ of the initial guess restricts the search space of AC
to the set $T_{\leq m}$ of all time series of maximum length $m$. The choice of the length-parameter $m$ via the
initial guess is critical. If $m$ is too small, the search space $T_{\leq m}$ may not contain an unconstrained sample
mean. For a given sample $S$, the Reduction Theorem [13] guarantees the existence of an unconstrained
sample mean of a length at most $m_S = \sum_{x \in S} |x| - 2(|S| - 1)$. Consequently, we can safely constrain the
search space to $T_{\leq m_S}$ for solving the unconstrained sample mean problem. Then, a naive approach to
minimize the Fréchet function on $T_{\leq m_S}$ is to solve $m_S$ constrained sample mean problems on $T_{m_S}, \ldots, T_1$
and then to pick the solution with lowest Fréchet variation. When using state-of-the-art heuristics for
the $m_S$ constrained problems, the naive approach is computationally infeasible.

6
Algorithm 5 Average-Compress Algorithm

1: procedure AC(X, z)
2:   \( z_* \leftarrow z \) // best solution found so far
3:   \( f_* \leftarrow F(z_*) \) // variation of \( z_* \)
4:   \( \ell_* \leftarrow |z_*| \) // length of \( z_* \)
5: repeat
6:   A-Step: \( z \leftarrow \text{average}(X, z) \)
7:   C-Step: \( C(z) \leftarrow \text{compress}(z) \)
8:   //*** Evaluate solution
9:   \( z \leftarrow \text{argmin} \{ F(z') : z' \in C(z) \cup \{z\} \} \)
10:  if \( F(z) < f_* \) or \((F(z) = f_* \text{ and } |z| < \ell_*)\) then
11:     \( f_* \leftarrow F(z) \)
12:     \( \ell_* \leftarrow |z| \)
13:   \( z_* \leftarrow z \)
14: until convergence
15: return \( z_* \)

The purpose of compression is to substantially accelerate the intractable naive approach at the expense of solution quality. Instead of solving all \( m_S \) constrained problems, the AC algorithm uses compressions to select a few promising search spaces \( T_{m_0}, T_{m_1}, \ldots, T_{m_k} \) with \( m_0 > m_1 > \cdots > m_k \geq 1 \). Starting with \( T_{m_S} = T_{m_0} \), the solution \( z_{i-1} \) found in \( T_{m_{i-1}} \) is compressed in order to determine the next search space \( T_{m_i} \). The length-parameter \( m_i \) of the next search space \( T_{m_i} \) corresponds to the length of the compression \( z_i \) of \( z_{i-1} \) with lowest Fréchet variation. Obviously, this idea only accelerates the naive approach if the length \( m_i \) of the best compression is substantially smaller than the length \( m_{i-1} \) of the previous solution.

The theoretical upper bound \( m_S \) provided by the Reduction Theorem [13] is usually very large such that existing state-of-the-art heuristics for solving the constrained problem on \( T_{\leq m_S} \) are computationally intractable. In this case, also the AC algorithm using such a heuristic would be infeasible. However, empirical results on samples of two time series of equal length \( n \) suggest that the length of an unconstrained sample mean is more likely to be less than \( n \) [6]. Similar results for larger sample sizes are unavailable due to forbidding running times required for exact sample means. For solving constrained sample mean problems, it is common practice to choose \( m \) within the range of the lengths of the sample time series [18]. Within this range, experimental results showed that an approximate solution of a constrained sample mean can be improved by reducing its length using adaptive scaling [18]. These findings suggest to choose the length-parameter \( m \) within or slightly above the range of lengths of the sample time series.

3. Experiments

Our goal is to assess the performance of the proposed AC algorithm. For our experiments, we use the 85 data sets from the UCR archive [9]. Appendix A.1 summarizes the parameter settings of the mean algorithms used in these experiments.

3.1. Comparison of ADA and MSE

We compared the performance of ADA and MSE as compression subroutines of the AC algorithm. We applied the following configurations:

| Acronym    | Algorithm                        | I  |
|------------|----------------------------------|----|
| DBA-ADA    | DTW Barycenter Averaging [12]    | 1  |
| DBA-MSE    | DTW Barycenter Averaging [12]    | 1  |
| DBA-ADA_1  | DBA with ADA compression [18]    | 1  |
| DBA-MSE_1  | DBA with MSE compression         | 1  |
| DBA-ADA_2  | DBA with ADA compression         | *  |
| DBA-MSE_2  | DBA with MSE compression         | *  |

Column \( I \) refers to the number of iterations of the repeat-until loop of the AC Algorithm. Compression schemes with \( * \) iterations run until convergence. We applied DBA and the four AC algorithms to approximate the class means of every UCR training set[^1].

[^1]: The UCR data sets have prespecified training and test sets.
Table 1: Results of ADA and MSE compressions.

| Percentage deviations | DBA | DBA-ADA | DBA-MSE | DBA-ADA | DBA-MSE |
|------------------------|-----|---------|---------|---------|---------|
| avg                    | 0.0 | 4.6     | -6.0    | -5.6    | -7.0    |
| std                    | 0.0 | 4.8     | 5.9     | 5.6     | 6.6     |
| min                    | 0.0 | -43.1   | -43.9   | -47.1   | -47.7   |
| max                    | 0.0 | 0.0     | 0.0     | 0.0     | 0.0     |

| Ranking distribution   | Rank | DBA | DBA-ADA | DBA-MSE | DBA-ADA | DBA-MSE |
|------------------------|------|-----|---------|---------|---------|---------|
| avg                    | 5.0  | 3.8 | 2.4     | 2.5     | 1.0     |
| std                    | 0.3  | 0.6 | 0.6     | 0.7     | 0.2     |

| Space-saving ratios    | DBA | DBA-ADA | DBA-MSE | DBA-ADA | DBA-MSE |
|------------------------|-----|---------|---------|---------|---------|
| avg                    | 0.00| 0.34    | 0.44    | 0.36    | 0.45    |
| std                    | 0.00| 0.21    | 0.24    | 0.22    | 0.24    |
| min                    | 0.00| 0.00    | 0.00    | 0.00    | 0.00    |
| max                    | 0.00| 0.96    | 0.99    | 0.96    | 0.99    |

To assess the performance of the mean algorithms, we recorded the percentage deviations, ranking
distribution, and space-saving ratios. Here, we used the solutions of the DBA algorithm as reference.
The percentage deviation of a mean algorithm $A$ is defined by

$$p_{dev}(A) = 100 \cdot \frac{F(z_A) - F(z_{DBA})}{F(z_{DBA})},$$

where $z_{DBA}$ is the solution of the DBA algorithm and $z_A$ is the solution of algorithm $A$. Negative
(positive) percentage deviations mean that algorithm $A$ has better (worse) Fréchet variation than DBA.
The ranking distribution summarizes the rankings of every mean algorithm over all samples. The best
(worst) algorithm is ranked first (last). The space-saving ratio of algorithm $A$ is

$$\rho_{ss}(A) = 1 - \frac{|z_A|}{|z_{DBA}|},$$

A positive (negative) space-saving ratio means that the solution $z_A$ is shorter (longer) than $z_{DBA}$.

Table 1 summarizes the results. The top table shows the average, standard deviation, minimum, and
maximum percentage deviations from the Fréchet variation of the DBA algorithm (lower is better). The
table in the middle shows the distribution of rankings and their corresponding averages and standard
deviations. The best (worst) algorithm is ranked first (fifth). Finally, the bottom table shows the average,
standard deviation, minimum, and maximum space-saving ratios (higher is better).

All AC variants improved the solutions of the DBA baseline by 4.6% to 7.0% on average and 45%($\pm$2%)
in the best case. By construction, an AC solution is never worse than a DBA solution. The best method
is DBA-MSE with average rank 1.0 followed by DBA-MSE$_2$ and DBA-ADA with average ranks 2.4
and 2.5, respectively. These three methods clearly outperformed DBA-ADA$_1$ proposed by Petitjean et
al. [18]. The main improvement of DBA-MSE and DBA-ADA occurs at the first iteration.
Table 2: Results of the AC algorithms.

| Percentage deviations | DBA | SSG | ADBA | DBA-MSE | SSG-MSE | ADBA-MSE |
|-----------------------|-----|-----|------|---------|---------|----------|
| avg.                  | 0.0 | -7.4| -6.4 | -7.0    | -11.6   | -9.4     |
| std                   | 0.0 | 16.1| 10.4 | 6.6     | 13.8    | 10.5     |
| min.                  | 0.0 | -79.7| -81.8| -47.7   | -83.3   | -82.0    |
| max.                  | 0.0 | 188.1| 17.7 | 0.0     | 50.0    | 13.3     |

| Ranking distribution |
|----------------------|
| Rank | DBA | SSG | ADBA | DBA-MSE | SSG-MSE | ADBA-MSE |
|-------|-----|-----|------|---------|---------|----------|
| 1     | 1.9 | 5.1 | 1.1  | 9.5     | 80.6    | 8.9      |
| 2     | 2.1 | 37.6| 1.7  | 22.9    | 10.5    | 21.9     |
| 3     | 2.4 | 20.8| 8.6  | 25.9    | 2.1     | 41.4     |
| 4     | 8.1 | 19.5| 33.7 | 14.0    | 0.5     | 21.0     |
| 5     | 11.4| 11.1| 36.7 | 27.8    | 6.3     | 6.8      |
| 6     | 74.1| 5.9 | 18.3 | 0.0     | 0.0     | 0.0      |
| avg   | 5.5 | 3.1 | 4.6  | 3.3     | 1.4     | 2.9      |
| std   | 1.1 | 1.3 | 1.0  | 1.3     | 1.0     | 1.0      |

| Space-saving ratios |
|---------------------|
| avg.                | DBA | SSG | ADBA | DBA-MSE | SSG-MSE | ADBA-MSE |
|---------------------|-----|-----|------|---------|---------|----------|
| avg.                | 0.00| 0.00| 0.31 | 0.45    | 0.43    | 0.45     |
| std.                | 0.00| 0.00| 0.21 | 0.24    | 0.24    | 0.24     |
| min.                | 0.00| 0.00| -0.04| 0.00    | 0.00    | 0.00     |
| max                 | 0.00| 0.00| 0.91 | 0.99    | 0.99    | 0.97     |

We considered the lengths of the approximated means. Recall that all mean algorithms started with the same initial guess (medoid). The length of a DBA solution corresponds to the length of its initial guess, whereas solutions of AC algorithms are likely to be shorter by construction. The bottom table shows that solutions of ADA compression save about a third (0.34, 0.36) of the length of DBA solutions on average and solutions of MSE compressions close to a half (0.44, 0.45). As for the Fréchet variation, most of the space-saving occurs in the first iteration of an AC algorithm.

### 3.2. Comparison of AC Algorithms

The goal of the second experiment is to compare the performance of the following mean algorithms:

| Acronym | Algorithm | \( J \) |
|---------|-----------|---------|
| DBA     | DTW Barycenter Averaging \[12 \[13\] | – |
| SSG     | stochastic subgradient method \[21\] | – |
| ADBA    | adaptive DBA \[17\] | – |
| DBA-MSE | DBA with MSE compression | * |
| SSG-MSE | SSG with MSE compression | * |
| ADBA-MSE| adaptive DBA with MSE compression | * |

Table 2 summarizes the results using the same legend as in Table 1. The percentage deviations and rankings suggest that the three AC variants DBA-MSE, SSG-MSE, and ADBA-MSE performed substantially better than the corresponding base algorithms DBA, SSG, and ADBA, respectively. The SSG-MSE algorithm performed best with an average rank of 1.4, followed by ADBA-MSE (2.9), SSG (3.1), and DBA-MSE (3.3). Interestingly, ADBA performed worse than DBA-MSE. Both, ADBA and DBA-MSE, are based on the DBA algorithm. The difference between both algorithms is that ADBA compresses and expands selected subsequences of the current DBA solution, whereas DBA-MSE only compresses...
Table 3: Results of $k$-means clustering.

| Percentage deviations | DBA | SSG | ADBA | DBA-MSE | SSG-MSE | ADBA-MSE |
|-----------------------|-----|-----|------|---------|---------|----------|
| avg                   | 20.0| 8.3 | 8.1  | 11.5    | 1.5     | 3.8      |
| std                   | 58.1| 10.8| 4.8  | 58.1    | 6.2     | 3.5      |
| max                   | 491.8| 55.6| 26.5 | 488.5   | 44.4    | 17.1     |

| Ranking distribution | Rank | DBA | SSG | ADBA | DBA-MSE | SSG-MSE | ADBA-MSE |
|----------------------|------|-----|-----|------|---------|---------|----------|
| avg                  | 5.5  | 3.7 | 4.5 | 2.9  | 1.4     | 2.9     |          |
| std                  | 1.1  | 1.4 | 0.9 | 1.4  | 1.0     | 1.0     |          |

the current DBA solution. The results indicate that simple MSE compression on the entire sequence appears to be a better strategy than ADBA’s compression and expansion schemes on selected subsequences. Notably, SSG performed best among the three base averaging algorithms DBA, SSG, ADBA, and performed even better than DBA-MSE. These results are in contrast to those presented in [17], where ADBA outperformed SSG (and also DBA). Our findings confirm that the performance of SSG substantially depends on a careful selection of an optimizer (such as Adam) and a proper choice of the initial learning rate.

Next, we examine the length of the solutions. Note that SSG also does not alter the length of its initial guess such that $\rho_{ss}(DBA) = \rho_{ss}(SSG) = 0$. The bottom table shows that MSE compression schemes reduce the length of the solutions obtained by their corresponding base algorithm (DBA, SSG, and ADBA). The space-saving ratios of the AC variants are roughly independent of the particular base algorithm for mean computation (0.43–0.45). Notably, the base algorithm ADBA is more likely to compress rather than to expand the DBA solutions. This finding is in line with the hypothesis that an exact mean is typically shorter than the length of the sample time series [6].

3.3. Application: k-Means Clustering

In this experiment, we investigated how the quality of a mean algorithm affects the quality of a $k$-means clustering. Let $S = \{x_1, \ldots, x_n\} \subseteq T$ be a set of $n$ finite time series. The goal of $k$-means is to find a set $Z = \{z_1, \ldots, z_k\}$ of $k$ centroids $z_j \in T$ such that the $k$-means error

$$J(Z) = \frac{1}{n} \sum_{i=1}^{n} \min_{z \in Z} dtw(x_i, z)^2$$

is minimized. We used DBA, SSG, ADBA, DBA-MSE, SSG-MSE, and ADBA-MSE for computing the set $Z$ of centroids. We applied the six variants of $k$-means to 70 UCR data sets and excluded 15 UCR data sets due to overly long running times (see Appendix A.2). We merged the prespecified training and test sets. The number $k$ of clusters was set to the number of classes and the centroids were initialized by the class medoids.

Table 3 summarizes the results. The top table presents the average, standard deviation, minimum, and maximum percentage deviations from the respective minimum $k$-means error (lower is better). The percentage deviation of $k$-means algorithm $A$ for data set $D$ is defined by

$$\rho_{dev}(A,D) = 100 \cdot \frac{J(Z_A) - J(Z_D)}{J(Z_D)},$$
where $Z_A$ is the set of centroids returned by algorithm $A$ and $Z_D$ is the best solution obtained by one of the six $k$-means algorithms. The bottom table shows the distribution of rankings and their corresponding averages and standard deviations. The best (worst) algorithm is ranked first (sixth).

The results show that the AC approach substantially improved all $k$-means variants using one of the base averaging methods (DBA, SSG, ADBA). Notably, SSG-MSE performed best with an average percentage deviation of 1.5% and an average rank of 1.4, followed by ADBA-MSE (3.8% and 2.9). The average percentage deviations of DBA and DBA-MSE are substantially impacted by the results on a single data set (DiatomSizeReduction). Removing the DiatomSizeReduction data set yields an average percentage deviation of 13.2 for DBA and 4.6 for DBA-MSE, whereas the other average percentage deviations remain unchanged up to ±0.1%. These findings confirm the hypothesis raised by Brill et al. [6] that better mean algorithms more likely result in lower $k$-means errors.

4. Conclusion

We proposed a generic average-compress algorithm for the unconstrained sample mean problem in DTW spaces. Starting with an initial guess of sufficient length, the AC algorithm alternates between averaging and compression. In principle, any averaging and any compression algorithm can be plugged into the AC scheme. The compression guides the algorithm to promising search spaces of shorter time series. This approach is theoretically justified by the Reduction Theorem [13] that guarantees the existence of an unconstrained sample mean in a search space of bounded length. Experimental results show that the AC algorithm substantially outperforms state-of-the-art heuristics for time series averaging. In addition, we observed that better averaging algorithms yield lower $k$-means errors on average. Open research questions comprise empirical analysis of alternative compression methods for the AC algorithm and reducing its computational effort.

A. Experimental Settings

A.1. Hyperparameter Settings

In all experiments, we selected the sample medoid as initial guess of a mean algorithm. The DBA algorithm terminated after convergence and latest after 50 epochs (cycles through a sample). The ADBA algorithm terminates subsequence optimization when the sum of the scaling coefficients changes its sign and latest after 50 iterations. The SSG algorithm terminated after 50 iterations without observing an improvement and latest after max (50, 5000/n) epochs. As optimization scheme, SSG applied Adam [16] with $\beta_1 = 0.9$ as first and $\beta_2 = 0.999$ as second momentum. To cope with the problem of selecting an initial learning rate, we used the procedure described in Algorithm 6. The input is a sample $S$ of size $n$. The output is the best solution found. The algorithm terminates if the solution did not improve for two consecutive learning rates and latest if $\sqrt{n}/2^i \leq 10^{-6}$.

Algorithm 6 SSG with learning rate selection

1: procedure SSG($S$)
2: \hspace{1cm} $n \leftarrow |S|$
3: \hspace{1cm} $i \leftarrow 1$
4: repeat
5: \hspace{1cm} test SSG with learning rate $\sqrt{n}/2^i$
6: \hspace{1cm} record best solution $z_*$ found so far
7: \hspace{1cm} $i \leftarrow i + 1$
8: until convergence
9: return $z_*$

A.2. Data Sets Excluded From $k$-Means Experiments

The following list contains all UCR data sets excluded from $k$-means clustering due to computational reasons:
References

[1] A. Abanda, U. Mori, and J.A. Lozano. A review on distance based time series classification. *Data Mining and Knowledge Discovery*, 33(2):378–412, 2019.

[2] W.H. Abdulla, D. Chow, and G. Sin. Cross-words reference template for DTW-based speech recognition systems. *Conference on Convergent Technologies for Asia-Pacific Region*, 2003.

[3] S. Aghabozorgi, A.S. Shirkhorshidi, and T.-Y. Wah. Time-series clustering – A decade review. *Information Systems*, 53:16–38, 2015.

[4] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

[5] R. Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

[6] M. Brill, T. Fluschnik, V. Froese, B. Jain, R. Niedermeier, and D. Schultz. Exact Mean Computation in Dynamic Time Warping Spaces. *Data Mining and Knowledge Discovery*, 33(1):252-291, 2019.

[7] L. Bulteau, V. Froese, and R. Niedermeier. Hardness of Consensus Problems for Circular Strings and Time Series Averaging. *CoRR*, abs/1804.02854, 2018.

[8] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani. Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. *ACM Transactions on Database Systems*, 27(2):188–228, 2002.

[9] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G.E. Batista. The UCR Time Series Classification Archive. [www.cs.ucr.edu/~eamonn/time_series_data/](http://www.cs.ucr.edu/~eamonn/time_series_data/), Accessed: 08/2018.

[10] M. Cuturi and M. Blondel. Soft-DTW: A Differentiable Loss Function for Time-Series. *Proceedings of the 34th International Conference on Machine Learning*, 70:894–903, 2017.

[11] M. Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l’institut Henri Poincaré*, 215–310, 1948.

[12] V. Hautamaki, P. Nykanen, P. Franti. Time-series clustering by approximate prototypes. *International Conference on Pattern Recognition*, 1–4, 2008.

[13] B. Jain and D. Schultz. On the Existence of a Sample Mean in Dynamic Time Warping Spaces. *CoRR*, abs/1610.04460, 2016.

[14] B. Jain and D. Schultz. Asymmetric learning vector quantization for efficient nearest neighbor classification in dynamic time warping spaces. *Pattern Recognition*, 76:349–366, 2018.

[15] E. Keogh, K. Chakrabarti, S. Mehrotra, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Journal of Knowledge and Information Systems*, 3(3):263–286, 2001.

[16] D. P. Kingma and J. L. Ba. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

[17] Y. Liu, Y. Zhang, and M. Zeng. Adaptive Global Time Sequence Averaging Method Using Dynamic Time Warping. *IEEE Transactions on Signal Processing*, 67(8):2129–2142, 2019.

[18] F. Petitjean, A. Ketterlin, and P. Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44(3):678–693, 2011.

[19] L.R. Rabiner and J.G. Wilpon. Considerations in applying clustering techniques to speaker-independent word recognition. *The Journal of the Acoustical Society of America*, 66(3):663–673, 1979.

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

[21] D. Schultz and B. Jain. Nonsmooth analysis and subgradient methods for averaging in dynamic time warping spaces. *Pattern Recognition*, 74:340–358, 2018.

[22] S. Soheily-Khah, A. Douzal-Chouakria, and E. Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp. *Pattern Recognition Letters*, 75:63–69, 2016.

[23] C. W. Tan, G.I. Webb, and F. Petitjean. Indexing and classifying gigabytes of time series under time warping. *Proceedings of the 2017 SIAM International Conference on Data Mining*, 282–290, 2017.
[24] E. Terzi and P. Tsaparas. Efficient algorithms for sequence segmentation. *Proceedings of the 2006 SIAM International Conference on Data Mining*, 316–327, 2006.

[25] T.K. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics*, 4(1):52–57, 1968.

[26] Haizhou Wang and Mingzhou Song. Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming. *The R Journal*, 3(2):29–33, 2011.

[27] J.G. Wilpon and L.R. Rabiner. A Modified K-Means Clustering Algorithm for Use in Isolated Work Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(3):587–594, 1985.

[28] B.-Y. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary $L_p$ norms. *International Conference on Very Large Databases*, 385–394, 2000.