Improved time series clustering based on new geometric frameworks

Clément Pealat, Guillaume Bouleux, Vincent Cheutet

Highlights

Improved Time Series Clustering Based on New Geometric Frameworks

Clément Péalat, Guillaume Bouleux, Vincent Cheutet

- We use the geometrical information of the time series via Takens' embedding.
- We analyze the geometrical information obtained by the embedding on the Stiefel, the unit sphere and the $\mathbb{R}^{n \times p}$ manifolds.
- We point out the gain obtained by such an embedding with respect to traditional time series clustering approaches.
- We analyze over 79 times series databases different frameworks
- The advocated framework is the Stiefel embedding followed by the UMAP and HDBSCAN algorithms.

Improved Time Series Clustering Based on New Geometric Frameworks

Clément Péalat, Guillaume Bouleux*, Vincent Cheutet

Univ. Lyon, INSA-LYON, EA 4570, DISP, F-69621, Villeurbanne, France

Abstract

Most existing methods for time series clustering rely on distances calculated from the entire raw data using the Euclidean distance or Dynamic Time Warping distance. In this work, we propose to embed the time series onto higher-dimensional spaces to obtain geometric representations of the time series themselves. Particularly, the embedding on $\mathbb{R}^{n\times p}$, on the Stiefel manifold, and on the unit sphere are analyzed for their performances with respect to several yet well-known clustering algorithms. The gain brought by the geometrical representation for the time series clustering is illustrated through a large benchmark of databases. We particularly exhibit that, firstly, the embedding of the time series on higher dimensional spaces gives better results than classical approaches and, secondly, that the embedding on the Stiefel manifold, in conjunction with UMAP and HDBSCAN clustering algorithms - is the recommended framework for time series clustering.

Keywords:
Clustering, Time series, Delayed coordinate embedding, Embedding, Stiefel Manifold, UMAP, HDBSCAN

1. Introduction

Clustering is one of the most famous unsupervised machine learning methods [45]. This set of data mining techniques gives insight of the structure of the data. The goal is to determine from data with multiple unlabeled observations, groups

*Corresponding author

Email addresses: clement.pealat@insa-lyon.fr (Clément Péalat), guillaume.bouleux@insa-lyon.fr (Guillaume Bouleux), vincent.cheutet@insa-lyon.fr (Vincent Cheutet)

(clusters) of elements with the closest behaviour. In order to do that, the clustering algorithms create clusters aiming to maximize similarities for the elements inside the same cluster while minimizing similarities between elements of different clusters. The similarities obtained are closely related to the specific characteristics of the data. The datasets studied in this work are one-dimensional time series. This data structure is studied in several domains such as aviation [25], meteorology [27], industry [16], speech or music processing [8], mechanical diagnosis [4], healthcare [6, 10] and many others. The time series clustering corresponds to the application of the clustering methods to this special kind of data. One way to do it is to directly apply clustering algorithms to the raw data [22, 44]. Most of the studies that work directly on raw data use Dynamic Time Warping distance [30] or Euclidean distance [46]. But, in some cases, it does not give accurate results, as it was the case in one of our precedent works on medical data [33]. Some works have studied clustering with other distances but from the angle of high-dimensional time series. For example, the motion of a rigid body over time [38] which is characterized by a vector of $\mathbb{R}^3$ evolving over time, can be modeled by high-dimensional time series. In this case, it has been demonstrated in the literature that manifold learning and subspace-based clustering methods give both an accurate description of the data and an improved clustering with respect to Euclidean distances [24, 42, 11]. If the literature is vast concerning the clustering of high-dimensional data, such as clustering of images, or rigid body trajectories to name a few, there is a huge gap applying manifold-based approaches for one-dimensional data clustering. For example, reviews on one-dimensional time series clustering [44, 2, 19] do not address these approaches at all. Consequently, the present work explores and evaluates some well-known manifold-based approaches in the case of one-dimensional time series and aims to bridge the gap between one-dimensional data and their higher-order geometrical features useful for improving the clustering.

Extracting intrinsic geometrical properties from a time series can be processed in different ways. We will quote for example the class of works that use second order statistics [31], the class of works that propose a direct description of the time series on a Lie group [9, 5] or finally the class of works that use the famous delay coordinate embedding [29, 40, 39] in order to obtain a description of the time series in the phase space. Obviously these classes do not represent the exhaustiveness of the approaches for the geometrical characterization of time series, but the references given may help the reader, or at least help him to start, to enrich his state of the art.

In this work, we want to cluster time series via the extraction of geometrical characteristics obtained by the delay coordinate embedding. This so-familiar embedding is intimately relied with the Takens' theorem. The idea behind this theorem [29, 40, 39]

is to consider a time series as a measure of a hidden dynamic system of larger dimension. It states that it is possible to recover an accurate representation of the hidden system once some parameters, the delay and the dimension embedding, have been estimated. Embedding a time series in such a way leads to describing the time series by its trajectory on the phase space. The induced geometry of the system (of the trajectory) may be therefore studied with several ways [31, 12] by considering the matrix associated with the trajectory. In order to fully exploit the geometrical information brought by those matrices, we propose here to project them onto three manifolds, directly $\mathbb{R}^{n\times p}$, the Stiefel manifold and the unit sphere. This way, well-known clustering algorithms can be adapted to account for the distance defined on these manifolds. In particular, we propose to compare the clustering performances of the agglomerative hierarchical method, and K-means algorithm when they perform the clustering on the three manifolds proposed. Based on the good results obtained in [32], we have also decided to test the HDBSCAN algorithm [7] enhanced by UMAP [26].
Our point of view being to use the geometrical information of time series trajectories obtained by the delay coordinate embedding, we have first tested the relevance of such a hypothesis through time series generated from dynamic systems well known in the literature. Afterwards, we have tested all the couples (manifold-type, clustering algorithm) on 79 databases available in the UCR Time Series Classification Archive, where the 'true' labels of the time series are known. This process is summarized in fig.1.
The paper organizes as follow. In section 2, we present the delay coordinate embedding and the resultant embedding into the manifolds. Section 3 introduces important notions of Riemannian geometry that allows to determine the distances as well as the Fréchet means of each manifolds in section 4. In section 5, the clustering algorithms are presented in detailed and in section 6, we present the clustering results obtained on known time series models. Next, an exhaustive analysis of the frameworks proposed in this work are displayed in section 7 with the application of the frameworks on the 'UCR time series'. Finally, a conclusion as well as elements of perspectives are given in the section 8.

Notations

- D is a database of time series of length l : $D = (Y_i(t), t = 0, ...l-1, i = 0, ..., n-1)$, with each time series characterised by a label.
- $O(n)$ is the orthogonal group of dimension n defined by $O(n) = \{A \in \mathbb{R}^{n\times n} : A^T A = I_n\}$ with I_n the identity matrix of dimension n.

– $V_{n,p} = O(n)/O(n-p)$ is the Stiefel manifold. It is a subspace of $\mathbb{R}^{n\times p}$ defined by $V_{n,p} = \{A \in \mathbb{R}^{n\times p} : A^T A = I_p\}$

– S_n is the unit sphere of $\mathbb{R}^n$.

– $T_X M$ is the tangent space at X of the manifold M

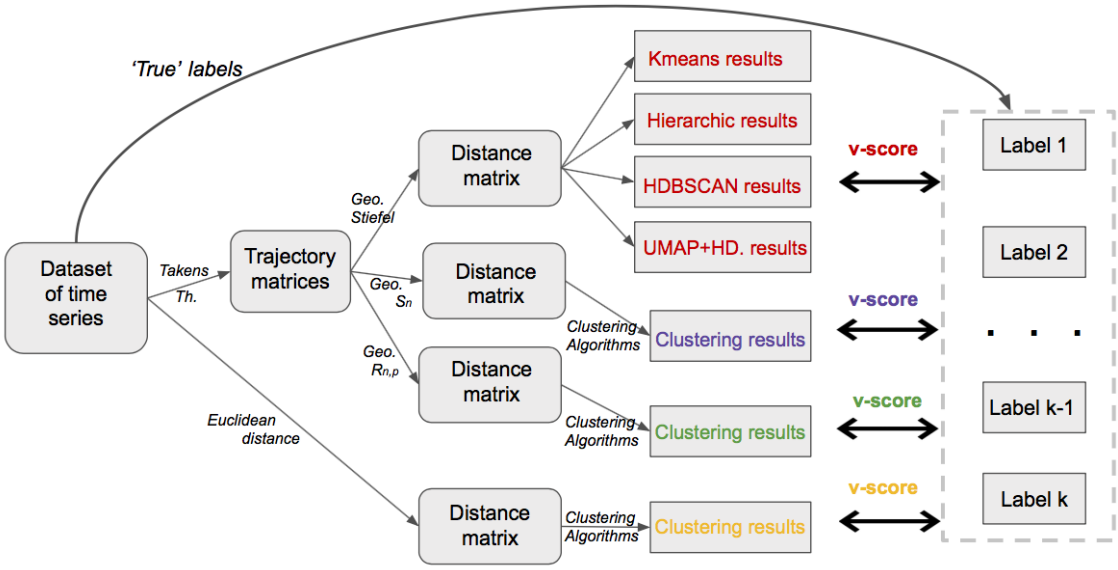


Figure 1: Summary of our benchmark. For a database, 16 clustering results are obtained depending on both the geometrical representation and the clustering algorithm. Then, using the v-measure-score, those results are compared to the expected labels.

2. Representation Manifold

One-dimensional time series can be seen as a discrete measurement (observation) of a dynamic system. The underlying dynamic system is in most situations of large dimension, obviously unknown a priori. Takens' theorem [39, 29, 40] states that it is possible, from this observation, to reconstruct the trajectory (the attractor) of the dynamic system by a basic delay coordinate embedding of the time series. From a time series $y = (y(0), ..., y(l-1))$, we subdivide it into vectors of size m defined by $(y(k), y(k+\tau), ..., y(k+(m-1)\tau))$, $k = 0, \ldots, l-1-(m-1)\tau$. By concatenating those vectors for all k, we obtained a trajectory matrix [43, 31] T_y of the time series

y defined as:

$$T_y = \begin{bmatrix} y(0) & y(\tau) & \cdots & y((m-1)\tau) \\ y(1) & y(2+\tau) & & \vdots \\ \vdots & \vdots & & \vdots \\ y(k) & y(k+\tau) & \cdots & y(k+(m-1)\tau) \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad (1)$$

Each row of the trajectory matrix represents a state of the time series in the embedded space which is now of larger dimension. Moreover, the trajectory matrix is an element of $\mathbb{R}^{n\times p}$ with $n = l-1-(m-1)\tau$ and $p = m$. From the previous relation, it is clear how the two parameters, m, the dimension of embedding and τ, standing for the delay, have a great importance in the geometric characterisation of the attractor for the time series.
Once the trajectory matrices have been determined from the time series, several possibilities exist to treat those elements of $\mathbb{R}^{n\times p}$. We studied here three geometric representations: a direct one ($\mathbb{R}^{n\times p}$), the Stiefel manifold, and the unit sphere which allows to deal with Stiefel manifolds of different dimensions.
By treating basically the trajectory matrix T_y of (1) as an element of $\mathbb{R}^{n\times p}$ leads to straightforwardly consider the Frobenius norm defined by the following

$$d(A,B) = (\sum_{k=0}^{n-1}\sum_{l=0}^{p-1}(a_{k,l}-b_{k,l})^2)^{\frac{1}{2}} \ \forall\ A, B \in\ \mathbb{R}^{n\times p}.$$

as the distance between the different trajectory matrices. In this context, the mean admits the basic expression $\overline{A} = \frac{1}{k}\sum_{i=1}^{k} A_i, \forall\ A_1, ..., A_k\ \in \mathbb{R}^{n\times p}$.
If we now go a little further in the information carried by the trajectory matrix, instead of using the Euclidean distance between the matrices, *i.e.* the Frobenius distance, we propose to measure the distance between the trajectory matrices by the distance of the subspace they span. This is equivalent to projecting the trajectory matrices on the Stiefel manifold.
The Stiefel manifold is a subspace of $\mathbb{R}^{n\times p}$ defined by $V_{n,p} = \{A \in \mathbb{R}^{n\times p} : A^T A = I_p\}$. For sufficiently small values of m and τ, we have directly $p < n$. It is thus enough to orthogonalize, via a QR decomposition, the trajectory matrices to see them as elements of $V_{n,p}$. We precise that we only dealt with the $p < n$ case. However, we can note that in the $p > n$ case, it is possible to apply a dimension reduction to return to $p < n$ [33]. So far we have deliberately not discussed the values of m and τ, which we have implicitly assumed to be equal for each of the embedded time

series (inside a database). Of course, in order to make the best use of the geometric information of the time series attractor, these values may vary and thus be different for each of the trajectory matrices to be compared. To cope with this, the trajectory matrices can be projected onto the unit sphere [41]. Let $U_1, U_2, ..., U_k$ be elements of $(V_{n,p_1} \times V_{n,p_2} ... \times V_{n,p_k})$. To compare those elements, we compute the projection matrix $P_{U_i} = U_i U_i^T$ which then becomes an element of $\mathbb{R}^{n \times n}$ with the particularity to be a symmetric matrix. This matrix is then embedded as a vector of $\mathbb{R}^{n(n+1)/2}$ composed of all the different elements of the symmetric matrix only once. To fix the ideas, in $\mathbb{R}^{2\times 2}$, we have:

$$\begin{pmatrix} a & b \\ b & d \end{pmatrix} \rightarrow \begin{pmatrix} a \\ b \\ d \end{pmatrix}. \tag{2}$$

Thanks to the orthogonality of the matrix $U_i, i = 0, ..., k$, all the associated vector v_i of $\mathbb{R}^{(n(n+1)/2)}$ are on the same sphere of a radius r and center c in $\mathbb{R}^{(n(n+1)/2)}$. To simplify, the vector on the unit sphere of $\mathbb{R}^{(n(n+1)/2)}$ is put back using $v_i' = \frac{v_i - c}{||v_i - c||}$. Thus, we can determine the distance and the mean using the geometry of $S^{(n(n+1)/2)}$. This will be discussed in section 2.
The Stiefel manifold and the unit sphere are particular spaces. They are not flat spaces like the euclidean space is. To determine the distance and the mean with respect to these manifolds, we propose in the next part a presentation of the Riemannian geometry tools.

3. Generalities about Riemannian geometry

A manifold is a set of points that only locally looks like a Euclidean space. Without going too much into the abstract details of the definition of a manifold, we will nevertheless place ourselves in the case where the manifold is a subspace of a larger ambient Euclidean space. For example, the circle is a one-dimensional manifold which can be represented on $\mathbb{R}^2$. This manifold is locally similar to a straight line and thus, similar to $\mathbb{R}^1$; this explains the dimension of the circle manifold. A little more rigorously we still need to have the existence of a diffeomorphic map, *i.e.* a function f which is C^k (with f^{-1} also C^k), realizing a bijection between the local part of the manifold and the smaller Euclidean space.

3.1. Tangent Space

We consider a smooth parametric curve $\gamma : \mathbb{R} \rightarrow M$ passing by p at an initial time t_0. We can define a tangent vector v of γ at p by $v = \frac{d\gamma}{dt})_{t_0}$. The tangent space

T_pM at the manifold M at p is given by all the tangent vectors at p coming from all the smooth parametric curve passing by p. The tangent space allows us to determine the metric on a manifold.

3.2. Riemannian metric

For each point p of the manifold M, we define the function $g_p : T_pM \times T_pM \to \mathbb{R}$ as the inner product between elements of the tangent space. This defines g as the family of inner product of the tangent space at the different points of the manifold. Considering g invariant allows to defined it as the Riemannian metric and (M, g) becomes a Riemannian manifold. Then, a distance can be defined with respects to this metric.

3.3. Distance

Let's call $\gamma : [a; b] \to M$ a smooth curve on the Riemannian manifold (M, g). The length of this curve is defined by $L(\gamma) = \int_a^b \sqrt{g(\gamma'(t), \gamma'(t)}\, \mathrm{d}t = \int_a^b ||\gamma'(t)||\, \mathrm{d}t$. Let X,Y be two elements of the manifold M. The distance between X and Y is defined by the minimization of L(γ) with γ such that $\gamma(0) = X$ and $\gamma(1) = Y$. The curve γ realizing the minimization is called the geodesic. In fig.2, from [42], the several elements of the Riemannian geometry are summarized.
Those notions of Riemannian geometry allow us to define the distance between the time series with respect to the geometrical representations used.

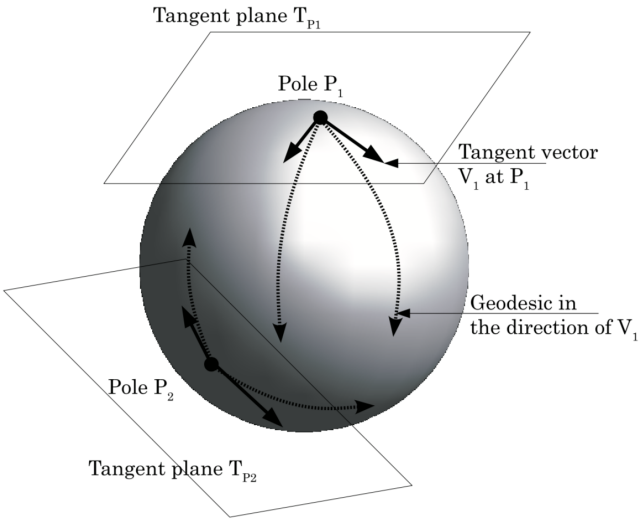


Figure 2: Illustration of the Riemannian geometry on the sphere manifold in $\mathbb{R}^3$.

3.4. Riemannian exponential and logarithm

From a starting point X on a manifold M, the Riemannian exponential (or exponential map) runs along the geodesic of origin X with respect to the direction of an element of the tangent space. For an element X of the manifold M, and Y an element of the tangent space $T_X M$ of M at X, we have only one geodesic γ that verifies such as $\gamma(0) = X$ and $\dot{\gamma}(0) = Y$. Then, the Riemannian exponential function is defined by $Exp_X(Y) = \gamma(1)$. Thus, the Riemannian logarithm function is defined as the inverse function of the Riemannian exponential. For example, on $\mathbb{R}^{n\times p}$, the geodesic γ such as $\gamma(0) = X$ and $\dot{\gamma}(0) = Y$ is directly defined by $\gamma(t) = X + tY$ and $Exp_X(Y) = X + Y$. Thus, $Log_X(Y) = Y - X$

In fig.2, from [42], two tangent vectors at P are projected into the manifold using the Riemannian exponential. We note Exp (resp. Log) the Riemannian exponential (resp. logarithm) function and exp ((resp. log) is the classic exponential (resp. logarithm) for scalars and matrices.
The Riemannian exponential and logarithm are useful tools to determine a mean on a manifold.

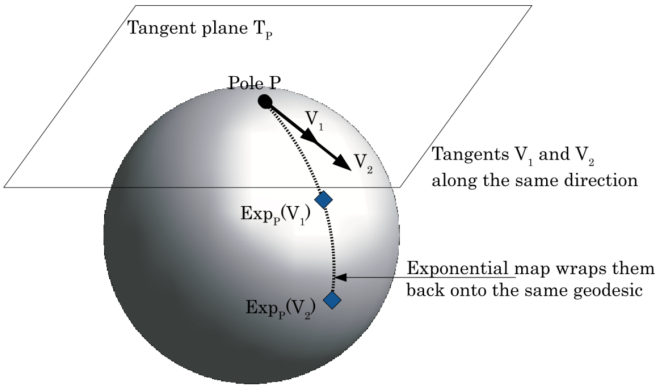


Figure 3: Riemannian Exponential on the sphere manifold in $\mathbb{R}^3$. We determine $Exp_P(V_1)$ from the value of the geodesic, that starts from P in the direction of V_1, after one unit of time.

3.5. Karcher mean

Once the distance is defined, a mean with respect to this distance, must be also defined. It is the Karcher mean, defined by

$$\overline{U} = \arg\min_{X\in M} \sum_{i=1}^{n} d_M(X, U_i)^2, \ \forall\, U_1, ...U_n \in M. \tag{3}$$

To determine this mean, we can use a gradient descent algorithm. To do so, we derive the equation (3), as proposed by [20], and we obtain a new definition for the karcher mean $\overline{X}$:

$$\sum_{i=1}^{n} Log_{\overline{X}}(X_i) = 0, \ \forall X_1, ...X_n \in M. \tag{4}$$

This equation uses the Riemannian logarithm, the inverse function of the Riemannian exponential.

Algorithm 1: Karcher means

Input : $X_1, X_2, ...X_N \in M$;
$\mu_0 = X_1$
while $\mu_k \neq \mu_{k+1}$ **do**
  $\Delta\mu = \frac{1}{N}\sum_{i=1}^{N} Log_{\mu_k}(X_i)$;
  $\mu_{k+1} = Exp_{\mu_k}(\Delta\mu)$;
end
Output : μ

With this equation (4), we can deduce our gradient descent algorithm (see algorithm 1). In fig.4, an example of the Gradient descent algorithm for four elements is illustrated. The function *Log* gives us the direction to follow on the tangent space, and the *Exp* function put this vector back on the manifold. For example, on $\mathbb{R}^{n\times p}$, we directly have the classic mean of the section 2.2 on the first iteration. Indeed, for $X_1, X_2, ..., X_N \in \mathbb{R}^{n\times p}$, $\Delta\mu = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_0)$. Then, $\mu_1 = \mu_0 + \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_0) = \frac{1}{N}\sum_{i=1}^{N} X_i = \overline{X}$.

4. Geometry of the proposed manifolds

4.1. Geometry of the Stiefel manifold

We can notice that for two elements A, B of $V_{n,p}$, an element C of $O(n)$ (the orthogonal group of dimension n) exists such as $CA = B$. Moreover, if A, B represent the same space, an element O of $O(n-p)$ (the orthogonal group of dimension $n-p$) exists such as $AD = B$. This allows us to see the Stiefel manifold as an homogeneous space defined as $V_{n,p} \simeq O(n)/O(n-p)$.
Two metrics can be used, they are defined on the tangent space of $V_{n,p}$ with either the euclidean metric [47]:

$$< A, B >= Tr(A^T B) \tag{5}$$

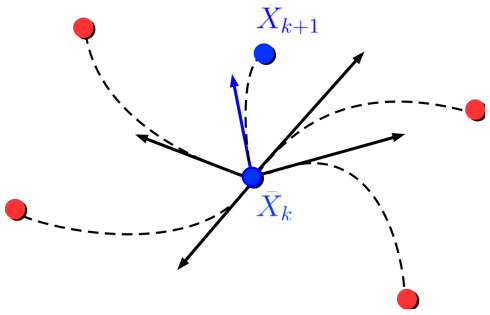


Figure 4: Example of one iteration of the gradient descent algorithm. The tangent vectors (in black) from X_k to the four elements are summed and give the resultant tangent vector (in blue). Then, the Riemannian exponential of this tangent vector put back on the manifold and gives the next value X_{k+1} of the estimated mean

.

coming from the definition of $V_{n,p}$ as a submanifold of $\mathbb{R}^{n\times p}$ and the canonical metric for $Q \in V_{n,p}$:

$$< A, B >= Tr\left(A^T\left(I - \frac{1}{2}QQ^T\right)B\right) \tag{6}$$

coming from the homogeneous space definition of $V_{n,p}$. The benefice of using the Stiefel manifold embedding is the correspondence of these two metrics regarding the geodesic distance between the points of $V_{n,p}$. Indeed, both yields to the same geodesic distance given by the principal angles of their subspaces in such a way that for any $A, B \in V_{n,p}$, we have:

$$distance(A, B) = \left(\sum_i^d \theta_i^2\right)^{\frac{1}{2}} \tag{7}$$

Figure 5 illustrates an example of the distance between two elements $A = (a_1|a_2), B = (b_1|b_2)$ of $V_{3,2}$. A base of the subspace $\mathcal{R}(A)$ spanned by A is defined by the two vectors of $\mathbb{R}^{\not\vdash}$ a_1, a_2 (in green in the figure). For B, the subspace $\mathcal{R}(B)$ is defined by the two vectors b_1, b_2 (in red in the figure). Then, the two principal angles θ_1, θ_2 between A, B are defined by $\theta_1 = (a_1, b_1), \theta_2 = (a_2, b_2)$.

As we have seen, p rotations (so p angles) are needed for $V_{n,p}$ to go from one element to another. To ensure that we deal with principal angles, we look for the rotations of smallest energy. To do so between two elements A, B of $V_{n,p}$ [28], we compute the singular values $\lambda_i, i = 1, .., p$ of A^TB. Then, for i in $1, ...p$, $\theta_i = arccos(\lambda_i)$ are the principal angles between A and B. The quick algorithm 2 gives the computational point of view of the distance on the Stiefel manifold.

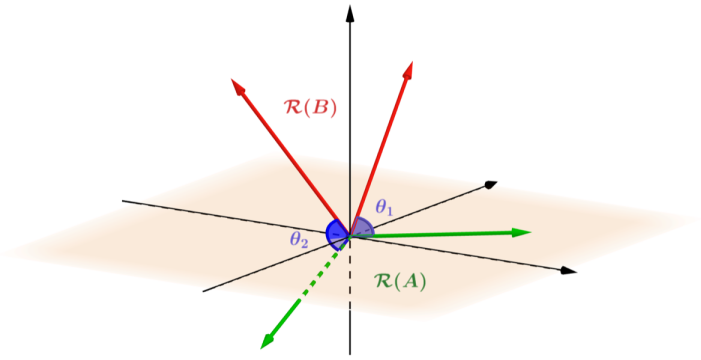


Figure 5: Example of Principal Angle between two elements A and B of $V_{3,2}$. The principal angles characterize the distance between the subspaces spanned by the columns of A and B, defined as $\mathcal{R}(A)$ and $\mathcal{R}(B)$ respectively. Only two rotations of angles θ_1 and θ_2 are needed to go from $\mathcal{R}(A)$ to $\mathcal{R}(B)$.

Algorithm 2: Computation distance on the Stiefel manifold

Input : $A, B \in V_{n,p}$

$A^T B = U \Sigma V$ (Singular Value Decomposition)

$\theta_i = arccos(\lambda_i), i = 1, \ldots, p$

Output : $d(A, B) = (\sum_{i=1}^{p} \theta_i^2)^{\frac{1}{2}}$

Using this distance, we can apply agglomerative hierarchical, HDBSCAN and UMAP algorithms, but we need the associated average to use the K-means algorithm. The algorithms 3 and 4 from [48] allow us to compute those two functions.

Algorithm 3: Stiefel Exponential

Input : $U \in V_{n,p}$, $\Delta \in T_U V_{n,p}$

Decomposition of Δ: $\Delta = UU^T\Delta + (I - UU^T)\Delta$

Decomposition QR of $(I - UU^T)\Delta$: $\Delta = UA + Q_E R_E$

$$\begin{bmatrix} M \\ N_E \end{bmatrix} = exp\left(\begin{bmatrix} A & -R_E^T \\ R_E & 0 \end{bmatrix}\right)\begin{bmatrix} I_p \\ 0 \end{bmatrix}$$

Output : $Exp_U^S t(\Delta) = UM + Q_E N_E$;

Algorithm 4: Stiefel Logarithm

Input : $U \in V_{n,p}$, $\overline{U} \in V_{n,p}$, threshold τ ;

Initialization :

$M = U^T\overline{U}$

$QN = \overline{U} - UM$

$$V_0 = \begin{bmatrix} M & X_0 \\ N & Y_0 \end{bmatrix}$$

$$log(V_0) = \begin{bmatrix} A_0 & -B_0^T \\ B_0 & C_0 \end{bmatrix}$$

while $\|C_k\|_2 \leq \tau$ **do**

$\phi_k = exp(-C_k)$ $V_{k+1} = V_k W_k$ where $W_k = \begin{bmatrix} I_p & 0 \\ 0 & \phi_k \end{bmatrix}$

end

Output : $Log_U^{St}(\overline{U}) = UA_k + QB_k$

4.2. Geometry of S_n

The unit sphere [15] on $\mathbb{R}^n$ is defined by $S_n = \{X \in \mathbb{R}^n : X^T X = 1\}$. For a point X of S_n, the tangent space at X is all the vectors of $\mathbb{R}^n$ perpendicular at X. So, $T_X S = \{U \in \mathbb{R}^n :< U, X >= U^T X = 0\}$. The distance between X, Y of S_n is defined by

$$d(X, Y) = arccos(X^T Y)$$

Table 1: Different computational formulas for each manifold

Geometry	$\mathbf{V_{n,p}}$	$\mathbf{S_n}$	$\mathbb{R}^{\kappa \times \iota}$
Distance	See algo 2	$d(X,Y) = arccos(X^T P)$	$d(X,Y) = (\sum_{k=0}^{n-1} \sum_{l=0}^{p-1} (x_{k,l} - y_{k,l})^2)^{\frac{1}{2}}$
Karcher mean	Gradient descent algorithm (algo 1)		$X_1, ..., X_n \in \mathbb{R}^{\kappa \times \iota}, \overline{X} = \frac{1}{k} \sum_{i=1}^{k} X_i$
Riemannian exponential	See algo 3	$Exp_X(v) = cos(\lvert v \rvert) \times x + \frac{sin(\lvert v \rvert)}{\lvert v \rvert} \times y$	$Exp_X(v) = X + v$
Riemannian logarithm	See algo 4	$Log_X(Y) = \frac{d(X,Y)}{\sqrt{1-(X^T Y)^2}}(Y - (X^T P)X$	$Log_X(Y) = Y - X$

We can also remark that : for all $(X, H) \in S_n \times \mathbb{R}^n, < X, H - X^T HX >= X^T H - (X^T X)(X^T H) = 0$. So, the projector into the tangent space of X is defined by $P_X(H) = H - X^T H$, for all $H \in \mathbb{R}^n$.
For $(X, U) \in S_n \times T_X S_n$, we define the smooth curve γ on $[0, 1]$ by $\gamma(t) = cos(t\|U\|)X + \frac{sin(t\|U\|)}{\|U\|}U$. We can remark that for all t, $\gamma(t)^T \gamma(t) = 1$ (so, it is on the manifold) and $\gamma(0) = X, \gamma'(0) = U$. So, we know from the definition that the Riemannian exponential is defined by:

$$Exp_X(U) = \gamma(1) = cos(\|U\|)X + \frac{sin(\|U\|)}{\|U\|}U$$

Moreover, as the inverse function, we have

$$\forall X, Y \in S_n, Log_X(Y) = \frac{d(X,Y)}{\|P_X(Y - X)\|} P_X(Y - X)$$

We can then determine the mean on the sphere using the same algorithm that we use for the Stiefel manifold.
We propose on the table 1, a summary of the different elements needed to apply the clustering algorithms for each manifold.

5. Clustering Algorithm

Once the distances and the means on the different geometrical representations have been determined, clustering algorithms are applicable. To determine the clusters, we proposed to use HDBSCAN (density-based clustering algorithm) [7] enhanced by UMAP [26], K-means and Agglomerative Hierarchical clustering.

5.1. K-means and Agglomerative Hierarchical Clustering

K-means is one of the most famous algorithms of clustering. It needs beforehand to determine the number of clusters. For a set of n time series $(Y_i, i = 1, ..., n)$, the K-means algorithm initializes randomly a number k of center-clusters (k is the number of clusters) and each time series is assigned to its closest center clusters. All time series are affected to one and only one cluster. The center-clusters are then recalculated with the new composition of the clusters, they become the mean of the time series in a cluster. This iteration is repeated until the composition of the clusters stop changing [18]. The Agglomerative Hierarchical algorithm [17] is an other classic algorithm of clustering. It starts by considering each time series as one distinct cluster. Then, the distance between each cluster is calculated (according to the distance used). The nearest clusters to each other are then merged together to form a new cluster. So, the distance between each cluster needs to be updated. Some methods exist to tackle this problem and we chose to use the average method (one of the most popular choice). The average method defines the distance between two clusters as the mean of the distances between each pair of observations of the clusters. The iteration goes on until all the time series have been merged in the same cluster. The final result is a dendogram where each merging corresponds to a node [14].
For both clustering algorithms, the number of clusters needs to be determined beforehand. To do so, we propose to use the Silhouette score [21].

5.1.1. Silhouette score

From a clustering results, the silhouette score is calculated using the mean intra-cluster distance (a_i) and the mean nearest-cluster distance (b_i) for the time series x_i. The silhouette score is determined by $\frac{1}{n}\sum_{i=1}^{n}\frac{b_i - a_i}{max(a_i, b_i)}$. The silhouette score is also a indicator of the performance of the algorithm, it is bounded between -1 for incorrect clustering and 1 for good clustering.
So, we can use it to determine the correct number of cluster [36]. Indeed, we compute the results of the K-means and Agglomerative Hierarchical algorithms for several values of number of clusters k. We keep the value of k such that the silhouette score is maximized.

5.2. HDBSCAN

HDBSCAN is a prolongation of the clustering algorithm Density-Based Spatial Clustering of Applications with noise (DBSCAN). We start by explaining DBSCAN [37, 23].
DBSCAN is one of the most used density-based clustering algorithms. A cluster

determined by this kind of clustering concurs to a region of high density on the data surrounded by region of low density [23]. Moreover, DBSCAN has the ability to detect noise points. Two parameters must be chosen beforehand: the threshold ϵ and the number of neighbors m. A point A of the data is density reachable from an other point B, if a path from A and B through the elements of the data exists such that the distance between two elements of the path is smaller than ϵ. For each point of the datasets, DBSCAN determines its category:

- Core point: If a point of the dataset has in its ϵ-neighbourhood more than m other points of the dataset, it is a core point.
- Border point: A point of the dataset is a border point if it is not a core point and is density reachable by at least one core point.
- Noise point: A point is considered as noise if it is neither core point nor border point

Then, the clusters are defined such that there is at least one core point, and all the elements are density reachable from one another. Figure 6, from [37], is an example of DBSCAN clustering with m (number of neighbors) equal to two. The core points in red creates a cluster. B and C are not core points (not enough neighbors), but are density reachable from the red points, so they belong to the same cluster. N is not reachable, so it is a noise point.

However, in this algorithm, the threshold ϵ is chosen by the user, and there is no correct answer on how to choose the threshold on real data. Moreover, if for two groups of our dataset, the densities are not the same, ϵ can consider a valid group as noise. To tackle this problem, we use the HDBSCAN algorithm. This algorithm computes the results of DBSCAN for all thresholds in the range of $]0; +\infty[$. The clustering starts with one big cluster (meaning a high value of ϵ). Then, when ϵ decreases, the cardinal of a cluster is diminishing until it splits into two clusters or when all the elements become noises. It gives us a birth and death of each cluster in function of ϵ. So, for each cluster, a score of stability is given depending on the ϵ of creation and the ϵ of disappearance. A cluster has a high score of stability if it exists for a large range of value of ϵ. Then, the clusters with a good stability score are kept. So, the HDBSCAN algorithm gives accurate clusters, even if there is a difference of density, and the user does not have to chose the value of a threshold.
The values of the number of neighbors m is kept by default at 5 (value proposed on `https://hdbscan.readthedocs.io/en/latest/index.html`).

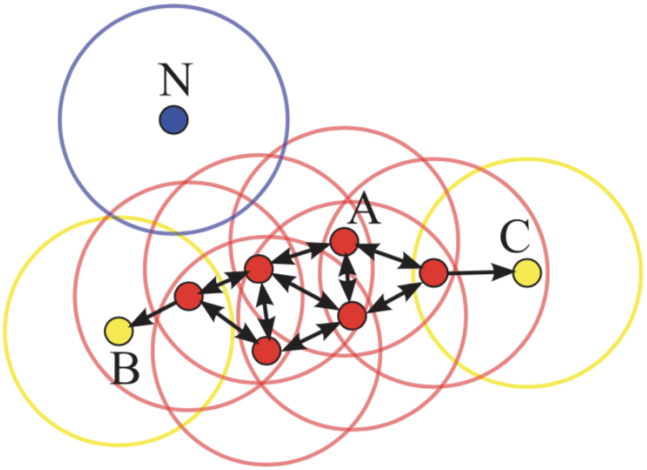


Figure 6: Example of clustering by DBSCAN. The dots corresponds to the elements of the datasets. The ϵ-neighbourhood is the circles. The blue is for noise points, the red for core points, and yellow for border points. A, B and C are in the same cluster.

5.3. HDBSCAN enhanced by UMAP

To increase the quality of the density algorithm of clustering, we use beforehand Uniform Manifold Approximation and Projection (UMAP) [26]. This decision is motivated by the fact that in some cases the density is not clear enough to use directly HDBSCAN. Indeed, in a lot of our applications, lot of elements of the data are considered as noise. The clustering algorithm decides that they are errors in the dataset. So, UMAP allows, by embedding in a lower dimension the data, to increase the gap between groups. There are still some concerns with this method because UMAP does not completely keep the density of the data. So, it can create pseudo-groups inside one 'true' cluster. However, in [32], we have shown that a reduction of dimension through UMAP improved the clustering performances of the HDBSCAN algorithm on time series from real data. Using the databases available at `www.cs.ucr.edu/~eamonn/time_series_data/`, we compared the clustering results of 6 clustering algorithms on 85 databases and the combination UMAP+HDBSCAN has clearly shown better results. UMAP realizes a reduction of dimension which clarifies the structure of the data for HDBSCAN. For example, in the case of the embedding into Stiefel manifold $V_{n,p}$, the time series belongs to an ambient space of dimension $d_{stiefel} = pn - \frac{p(p+1)}{2}$. So, UMAP realizes a reduction dimension from $d_{stiefel}$ to a selected dimension. We present more clearly below how UMAP works. The theory behind this algorithm assumes that the data is uniformly distributed. It is a strong assumption, but a correct choice of a parameter σ (see eq.(8)) allows to

make it true. It starts by giving a graphical directed weighted representation. Each element is a vertex, and the distance determines the edges. Indeed, for a given k, and an element Y, the k-th closest neighbor $Y_1, ..., Y_k$ (sorted out by distance d to Y) are linked to Y. The weights, degrees of membership, are then computed between Y and $Y_1, ..., Y_k$. The closest element Y_1 to Y as a membership of 1. Let's call λ the distance between Y_1 and Y. The weight between Y and $Y_{0<i\leq k}$ is then defined by :

$$w_i = exp(-\frac{d(Y, Y_i) - \lambda}{\sigma})$$

The choice of this heat kernel for the weight is justified in [3]. The parameter σ is defined by eq.(8). This parameter ensures that for each element Y of the data, the density of the circle of center Y and of radius equal to the distance to the k-neighbors are sensibly the same for each Y.

$$\sum_{i=1}^{k} w_i = log_2(k) \tag{8}$$

Once the graph G has been determined, the reduction of dimension into $\mathbb{R}^m$ is done. To do so, the Laplacian eigenmaps is used. The i-th element of the data is represented as a vector of $\mathbb{R}^m$ of coordinates $(f_1(i), ..., f_m(i))$ with $f_0, .., f_m$ the eigenvectors of the Laplacian associated with G (with the eigenvalues associated such that $\lambda_0 < \lambda_1 < ... < \lambda_m$). Then, it defines a graph G' from the elements of $\mathbb{R}^m$, the same way that it did for G except for the weights. Indeed, the weight between $Y, Y' \in \mathbb{R}^m$ is now defined by :

$$w(X, Y) = \frac{1}{1 + a(\|Y - Y'\|_2^2)^b}$$

The parameters a and b are chosen such that the function ψ realizes a smooth approximation of Ψ with ψ and Ψ defined by :

$$\mathbb{R}^m \times \mathbb{R}^m \to [0, 1]$$

$$\psi(X, Y) = \frac{1}{1 + a(\|X - Y\|_2^2)^b}$$

$$\Psi(X, Y) = exp(-\|X - Y\|_2)$$

This smooth approximation allows to derived the cross-entropy between G and G'. This determines attractive and repulsive forces, and a forced directed graph layout algorithm is computed. So, each element of $\mathbb{R}^m$ acts as a physical point under those two forces until a physical equilibrium is obtained. The cross-entropy is now

minimized between the two graphs G and G'. So, from the dataset lying on the manifold M, UMAP returns elements of $\mathbb{R}^m$ with respect to the cross-entropy between a graph on the manifold and a graph on $\mathbb{R}^m$. To summarize :

- UMAP creates a graph G with respect to the distances on the manifold and to the k-neighborhood of each element.
- A graph G' on $\mathbb{R}^m$ is obtained by a Laplacian eigenmaps reduction of dimension.
- The graph G' is modified by a forced directed graph layout algorithm so that the cross-entropy between G and G' is minimized.

To realize the clustering, a global view of the structure of the data is needed. So, we fixed the parameters k (number of neighbors to determine the graph) at 30 throughout the rest of the article (or 10 if the concerned database has not enough samples) and the dimension of reduction is fixed at 10 (values proposed on `https://umap-learn.readthedocs.io/en/latest/`.

6. Clustering Results from known models

To get a first glimpse of the efficiency of the framework proposed in this work, we used four known dynamic systems from which we selected one coordinate to obtain a time series. We used then :

- The Lorenz system. it is defined by the system of equations (9), and is plotted in figure 7a. This figure is the trajectory of a dot submitted by the equations (9) with $\rho = 28$, $\sigma = 10$ and $\beta = 8/3$.

$$\begin{cases} \dot{x}(t) = & \sigma(y(t) - x(t)) \\ \dot{y}(t) = & x(t)(\rho - z(t)) - y(t) \\ \dot{z}(t) = & x(t)y(t) - \beta.z(t) \end{cases} \tag{9}$$

- The Rössler system. It is a dynamic system of $\mathbb{R}^3$. The movement is determined by a set of 3 coupled differential equations (10).

$$\begin{cases} \dot{x}(t) = & -y(t) - z(t) \\ \dot{y}(t) = & x(t) + ay(t) \\ \dot{z}(t) = & b + z(t)(x(t) - c) \end{cases} \tag{10}$$

  The figure 7b is obtained with $a = 0.1$, $b = 0.1$ and $c = 14$.

- The Henon system. This dynamic system is defined by the following equations

$$\begin{cases} x_{n+1} = y_n + 1 - a.x_n^2 \\ y_{n+1} = \quad b.x_n \end{cases} \tag{11}$$

  It is a two-dimensional system, and we traced the trajectory of 100 dots in figure 7d with $a = 1.4$ and $b = 0.3$. We can notice that the equations are discrete.

- The Ikeda system. It is initially a discrete-time dynamical system on the complex map. A 2D example is given by the following equations

$$\begin{cases} x_{n+1} = 1 + u(x_n \cos(\theta_n) - y_n \sin(\theta_n)) \\ y_{n+1} = \quad u(x_n \sin(\theta_n) + y_n \cos(\theta_n)) \\ \theta_n = \quad 0.4 - \frac{6}{1+x_n^2+y_n^2} \end{cases} \tag{12}$$

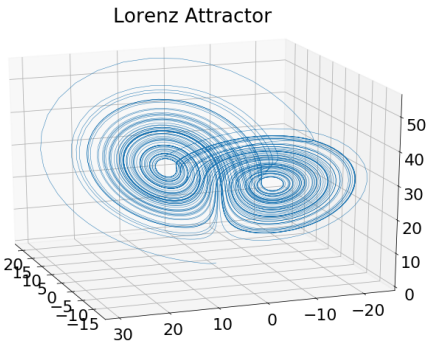
  The figure 7c is the trajectory of 100 dots, with random initial conditions and $u = 0.7$.

We used next these dynamic systems for creating known time series clusters.

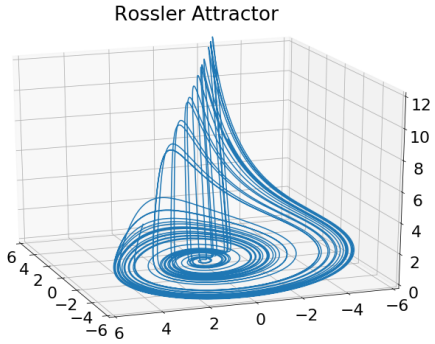
6.1. clustering results from a unique known model

In this scenario, we propose to create a first database from the Lorenz system. Depending on the values of ρ, the trajectory obtained by the Lorenz equations is very different. In the figure 8, we have plotted four trajectories got from four different values of ρ, the other parameters remaining unchanged. For each value of ρ, 100 trajectories according to the coordinates x, y and z have been calculated. For that, the differential equations have been discretized with a time step of 0.01. The difference between the trajectories comes from the initial conditions, randomly chosen between 0 and 1. Then, we kept only the x component and we therefore built 400 one-dimensional time series each belonging to one of the four known cluster.

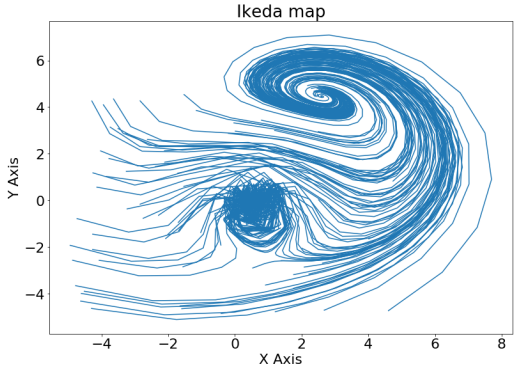
The delay parameter τ used for the delay coordinate embedding was kept to the value 10 and we fixed m at the value 3. Then, the embedding into the Stiefel manifold, via the QR decomposition, into the unit sphere and into $\mathbb{R}^{n\times p}$ are computed as well as the distance between the time series with respect to the geometrical representations. Finally, the clustering was done by the UMAP and HDBSCAN clustering algorithm. To check the quality of the results, we compared the results of the clustering to the 'true' clusters generated. The relevance indicator we chose was the v-measure-score [35]. It is an entropy-based clustering evaluation which realizes a harmonic mean



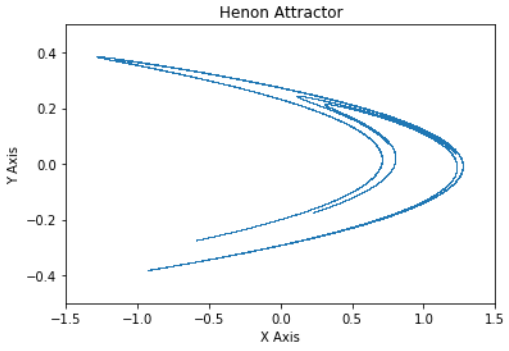
(a) Lorenz attractor with $\rho = 28$, $\sigma = 10$, $\beta = 8/3$



(b) Rössler attractor with $a = 0.1$, $b = 0.1$,$c = 14$

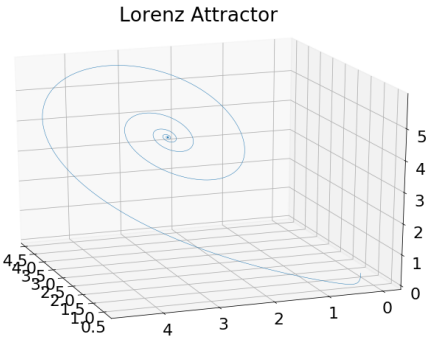


(c) Ikeda Map with $a = 1.4$,$b = 0.3$, trajectories of 100 dots in $\mathbb{R}^2$

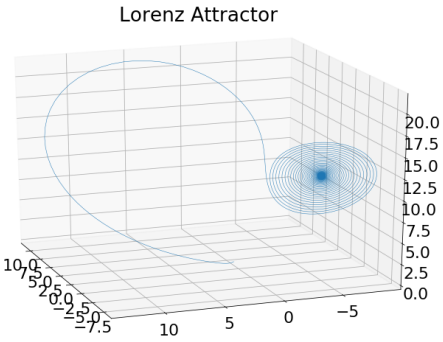


(d) Hénon Map with $u = 0.7$, trajectories of 100 dots in $\mathbb{R}^2$

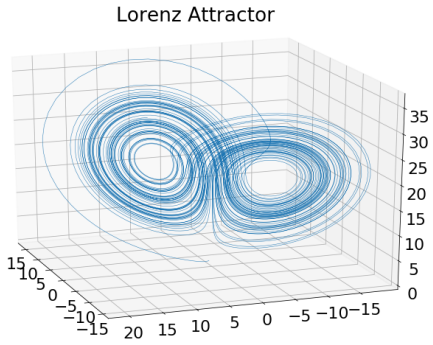
Figure 7: Representation of the four dynamic systems.



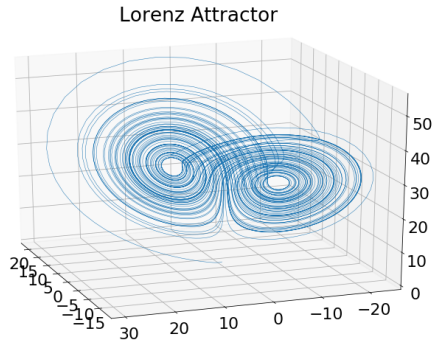
(a) Lorenz system with $\rho = 5$



(b) Lorenz system with $\rho = 13$



(c) Lorenz system with $\rho = 20$



(d) Lorenz system with $\rho = 28$

Figure 8: Trajectory of the Lorenz system for several values of ρ

between the completeness and homogeneity of the expected labels (true clusters) and the clustering results. Homogeneity gives a best score of 1 when all the elements of each cluster have the same 'true' cluster. By symmetry, completeness is at a maximum when all elements of each 'true' cluster are in the same cluster. The homogeneity and the completeness must be balance since if all the elements are in the same cluster the completeness is maximized, and if each element has its own cluster the homogeneity is maximized, hence the harmonic mean between homogeneity and completeness.
For the scenario studied, we obtained a v-measure-score of 1 for the Stiefel and unit Sphere manifolds embedding of the trajectory matrices and a v-measure-score of 0.721 when trajectory matrices are embedded in $\mathbb{R}^{n\times p}$. The lowest performance was obtained by using the euclidean distance between the time series with a v-measure-score of 0.648. For this toy example, we thus conclude that the Stiefel and the unit sphere embedding seems to completely exploit the geometrical intrinsic properties of the attractor of the time series, giving in conclusion the best performances with no error at all.

6.2. Clustering Results across several known models

As we did in the previous section, we created a database of time series. For that scenario, the time series are divided into four clusters according to the dynamic system from which they are obtained. We computed for each dynamic system 100 trajectories with the same parameters as given previously, and the initial conditions randomly chosen between 0 and 1. From this, we extracted the x coordinate of the trajectories. It gave us a database of 400 one-dimensional time series evenly distributed between four clusters. Alike the previous scenario, the parameters used for the Lorenz system are fixed to $\tau = 10$ and m=3. The delay parameter for the Ikeda and Henon systems is fixed to $\tau = 1$, whereas the delay parameter for the Rössler system is fixed to $\tau = 100$. The embedding parameter m is kept equal to 3 for all the systems. The fixed parameters given have been estimated to fully retrieve the geometrical properties of the associated attractors. We can notice that since the values of τ are not the same across all the databases, the trajectory matrices are of different dimensions, indeed $p = m$ and $n = l-(m-1)\tau$, $l = 5000$. Consequently, we have decided to keep only the first $l - 1 - (m - 1)100$ vectors of $\mathbb{R}^m$. In this way, all the matrices will belong to the same space $\mathbb{R}^{n\times p}$. We obtained a v-measure-score of 0.8456 for the Stiefel manifold, a v-measure-score of 0.837 for S_n, of 0.471 for $\mathbb{R}^{n\times p}$ and once again, the worst clustering performance was obtained by the Euclidean distance on the raw data with a v-measure-score of 0.460.
This second scenario confirms the good capacities of the framework composed by a

delay coordinate embedding followed by an embedding of the trajectory matrices on the Stiefel manifold to cluster the time series. Even if the clustering results are less exceptional than the previous scenario, a v-measure-score of near 0.85 is significant. Clearly this preprocessing framework, as the first stage before employing clustering algorithms, shows the importance of the intrinsic geometrical property of the time series for the clustering. But, shows that the representation manifold of the time series attractors needs to be carefully chosen as well.

7. Clustering Results on UCR Time Series Classification Archive datasets

To realize our benchmark, we used the 79 UCR Time Series Classification Archive databases available at UCR Time Series Classification Archive. A brief summary of those databases are presented in table 2. The number of clusters in a database varies from two (only two labels) to 60. Moreover, some databases have a large number of time series to cluster whereas others have a weaker one. For a given database, all its time series have the same length. The length of the time series varies from 24 samples to a maximum of 2709 samples.

Table 2: Presentation of the databases

Number of databases	79		
	Min	**Mean**	**Max**
Nb of clusters	2	7.5	60
Nb of time series	16	432	8926
Length of time series	24	422	2709

7.1. How we did it

We recall and specify our methodology here to create the benchmark. From open access databases, we have compared 4 different geometries and used 4 clustering algorithms. This gave us 16 different clustering results per database which we then compared to the expected labels (cf. fig.1). Three of our geometries were linked to the delay coordinate embedding. The time series were then transformed into a trajectory matrix which depended on two parameters m and τ (as already exposed in the previous sections). After some exploratory works, we have found that the embedding parameter m was not determinant and could be kept to a fixed value we chose equal to 5. On contrary, we found that the delay parameter τ was important for the clustering. To determine it, we have decided to use the real database labels. We have varied the values of τ between 1 and 20 and computed the distance matrix between each

elements for each geometry. Then, from the distance matrix, the silhouette score have been deduced with the true labels. For each of the three geometries, the τ values which maximized the silhouette score was kept.
Two of our clustering algorithms needed to determine the number of clusters in advance. For this, we have performed the clustering and computed the associated silhouette score for a number of clusters varying between 2 and the number of labels + 10. The number of clusters that maximized the silhouette score was kept.

7.2. Analysis of the Clustering results

On table 3, we present:

- The mean of the v-measure-score for each method of clustering across the 79 databases
- The standard deviation of the v-measure-score
- The number of databases for which a method is the best one. For example, K-means on $\mathbb{R}^{n\times p}$ has the best results for 3 databases (among 79 databases). We can notice that in some cases, the best score for a database is obtained by several methods.

The complete results are available in the appendix.

Table 3: Main characteristics of the v-measure-score obtained. For 4 geometries and 4 clustering algorithms, this table gives the mean and the standard deviation along the 79 databases. Also, the number and percentage of databases for which a method is the best are given.

Geometry	$\mathbb{R}^{n\times p}$				S_n			
Clustering algo.	**K-means**	**Hier.**	**HDB.**	**UMAP+HDB.**	**K-means**	**Hier.**	**HDB.**	**UMAP+HDB.**
Mean	0.240	0.224	0.217	0.300	0.303	0.254	0.227	0.322
Std	0.241	0.253	0.232	0.272	0.260	0.290	0.250	0.270
Times best results	3	2	3	9	7	10	6	12
Percentage best results	3.8%	2.5%	3.8%	11.4%	8.9%	12.7%	7.6%	15.2%
Geometry	$V_{n,p}$				Euclidean, $\mathbb{R}^l$			
Clustering algo.	**K-means**	**Hier.**	**HDB.**	**UMAP+HDB.**	**K-means**	**Hier.**	**HDB.**	**UMAP+HDB.**
Mean	0.245	0.220	0.190	0.376	0.281	0.242	0.215	0.293
Std	0.228	0.263	0.219	0.271	0.256	0.260	0.233	0.259
Times best results	7	5	4	21	8	6	3	4
Percentage best results	8.9%	6.3%	5.1%	26.6%	10.1%	7.6%	3.8%	5.1%

Analysing the results, the best method is clearly the application of UMAP and HDBSCAN framework on the Stiefel manifold. Indeed, it presents the best average v-measure-score (0.376) and gives the best results for 21 databases out of 79. It is quite clearly superior to the second best average of 0.322 obtained for the combination

UMAP+HDBSCAN on the unit Sphere which is the best method for 12 out of 79 databases. Overall, the other methods propose sightly inferior mean values, between 0.2 and 0.3.
From the point of view of the geometric representation, we have

- the Stiefel manifold which is the best geometrical representation for 46.9% of the databases. This means that for 37 databases across the 79, one of the 4 clustering algorithms used on the Stiefel manifold gives the best results.
- The unit Sphere which is the best for 44.3% of the databases (35/79).
- The space $\mathbb{R}^{n\times p}$ which is the best for 21.5% of the databases (17/79).
- The Euclidean on raw data which arrives at the last place with only 26.6% of databases well clustered (21/79).

Euclidean on raw data is clearly a less efficient method than the metrics induced by the Stiefel manifold or the unit sphere representations. It validates once again the interest of using the delay coordinate embedding to obtain geometric features of the time series through the trajectory matrix. Moreover, it is interesting to work on the geometrical representation of this trajectory matrix rather than using directly the Frobenius norm which places us on $\mathbb{R}^{n\times p}$. Indeed, the Stiefel manifold and the unit sphere present much better results. Finally, the unit sphere is an interesting alternative to the Stiefel manifold, because it is only slightly less good as the Stiefel and allows more flexibility by allowing to compare Stiefel manifold elements of different dimensions.
Once the geometrical representation of the time series analyzed, we let the reader gives itself an idea on which could be the best clustering framework to use on the Stiefel manifold. In this view we have the following results

- UMAP+HDBSCAN is the best clustering framework for 58.2% of the databases (46/79). It has the best v-measure-score average of 0.323.
- K-means is able to well clusters the databases in 31.6% of the cases (25/79) and has a mean value performance of 0.267.
- hierarchical algorithm managed to cluster 29.1% of the databases (23/79) with a mean v-measure-score value of 0.235.
- HDBSCAN alone obtained a ratio of 20.2% good clustering (16/79) with still less mean value of 0.212.

Part of the conclusion is that the HDBSCAN algorithm alone seems to be the worst clustering algorithm. We further point out that for nearly 16% of the databases (13/79), all 4 HDBSCAN results are zero. However, using it with UMAP seems really very interesting because the results are greatly improved. The use of UMAP as a pre-processing step of the HDBSCAN algorithm therefore seems very important. We can also notice that a few parameters are used for the UMAP+HDBSCAN clustering algorithms. It was not possible to optimize those elements for all the databases, but for a user with one database, those parameters can be manipulated to have better clustering results depending on what the user is looking for.
To represent the results, we propose to compare two methods by plotting the v-measure-score of one method in function of the v-measure-score of the other one. The databases are represented by points, and the red line separates into two fields, when one of the two methods is better than the other. The figure 9 gives a visual representation of the good results obtained by UMAP+HBDSCAN on the Stiefel manifold. Indeed, when plotted against a more classic method (K-means on raw data), we clearly see the efficiency of the method.

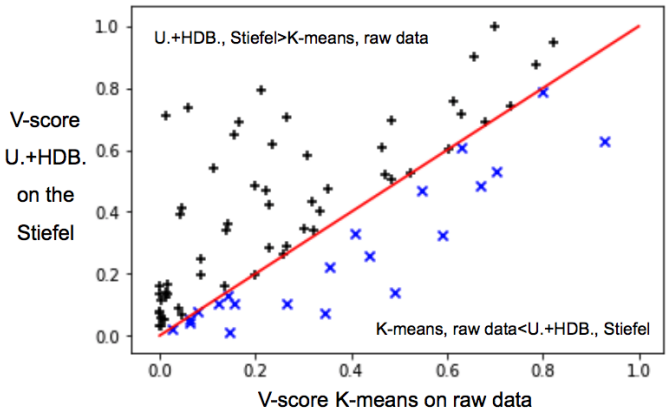
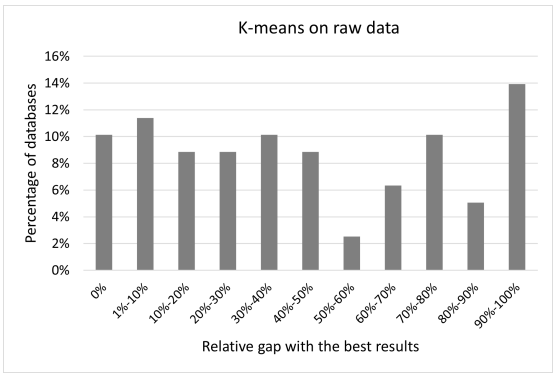


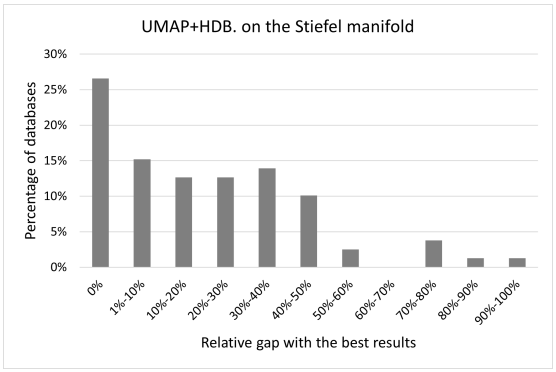
Figure 9: Databases represented by v-measure-score with UMAP+HDBSCAN on the Stiefel in function of v-measure-score with K-means on raw data

A large part of the databases are above the red line (when UMAP+HDBSCAN on the Stiefel is better than K-means on raw data). Moreover, the blue points are in general closer to the red line. Indeed, when K-means on raw data is better (this is the case for 21 databases out of 79), it is for a small amount. In average on the 21/79 databases, K-means on raw data has a higher v-measure-score with 0.117. The median is even lower with a value of 0.08.
The figure 10 gives an other visualisation of the efficiency of UMAP+HDBSCAN on the Stiefel compared to K-means on raw data. For each databases, the relative

gap $e = \frac{v_{max} - v_{method}}{v_{max}}$ between the results of a given method (fig. 10a: K-means on raw data, fig. 10b: UMAP+HDBSCAN. on the Stiefel) and the best results across the 16 methods is computed. Then, for each interval of 10% of the relative gap, the percentage of concerned databases is plotted. For example, for 10% of the databases, K-means on raw data gives results far away from the best results possible with a relative gap between 70% and 80%. This shows that UMAP+HDBSCAN on the Stiefel manifold is much more often the best method (first column). Moreover, when this method is not the best, it does not give completely wrong results either. Indeed, the number of databases falls largely when the relative gap increases. On the contrary, for the K-means method on raw data the number of databases with a very high relative gap is important, *i.e.* 14% of databases for which K-means on raw data is completely off.



(a) Comparison between the K-means on raw data and the best method



(b) Comparison between the UMAP+HDBSCAN on the Stiefel and the best method

Figure 10: The relative gap between the clustering of a given method and the best results is computed for each database. Then, the percentage of concerned databases by interval of relative gap is plotted for two methods : K-means on raw data, and UMAP+HDBSCAN on the Stiefel

The authors of [19] performed a similar benchmark work on the same databases. They tested two distances, the euclidean distance and the Time Wrapping distance as well as 5 clustering algorithms, the K-means, K-medoids, Fuzzy C-means, Density Peaks and Agglomerative. In addition, in their work, the authors have fixed the number of clusters of each databases with respect to the true labels given. The K-means method with the euclidean distance on raw data appears to be the best clustering method of their benchmark (see fig. 5). Since we show that the framework UMAP+HDBSCAN on the Stiefel manifold clearly presents better results than K-means on raw data, this framework outperforms the best one proposed in [19]. As a

conclusion, none of the clustering methods proposed and tested in [19] seem better than the geometrical frameworks we propose in this work.

8. Conclusion

In this article, we were interested in one-dimensional time series clustering. More precisely, we proposed to apply delay coordinate embedding methods to embed the time series on a higher-dimensional space. We used for that the delay coordinate embedding, also known Takens' embedding, which extracts geometrical features of the times series by representing them on the so-called phase space. The induced trajectory matrices can next be analyzed under the cover of different geometries. Each geometry leads to the definition of a particular distance and a particular mean which respect the structure of this geometry. Once these elements had been determined, clustering algorithms could be applied.

Since clustering is an unsupervised method, we thought it was important to carry out a complete benchmark to determine in advance which method is the best. To do this, we used 79 databases with known expected results. The good results obtained by geometric methods on high-dimensional time series, motivated us to carry out this work. We proposed to apply similar methods to one-dimensional time series using delay coordinate embedding. In addition, as the literature on one-dimensional time series clustering makes little mention of this type of methods, it appeared important to us to carry out this benchmark.

We have compared 4 coupled geometries with 4 clustering algorithms. The three geometries induced by the embedding on higher-dimensional space gave significantly better results than using an Euclidean method on raw data. Moreover, the Stiefel manifold allowed better comparison and thus clustering of time series than the unit sphere and then $\mathbb{R}^{n\times p}$.

If we look only at the clustering algorithms, a fairly clear ranking has emerged. The combination of UMAP and HDBSCAN has been the best algorithm in front of K-means, Hierarchical and HDBSCAN alone. In particular, the combination UMAP+HDBSCAN on the Stiefel manifold gave the best results.

However, two parameters came into play during the delay coordinate embedding. if m, the embedding dimension appeared as a parameter with little influence on clustering, the τ parameter seemed to be of prime importance. For the different simulation scenario, we have determined this optimal parameter for the clustering with respect to the 'true' labels, leading consequently to a small flavor of supervision for the methods analyzed. An undergoing work is then to find an optimal way to determine τ.

In the literature, several methods are proposed to determine m and τ for the Takens' theorem. In general, for m, the false nearest neighbor method is quite widely used [1, 34]. For τ this is more complicated. Some methods like the one proposed in [13] and which is an entropy-based method, allow to determine m and τ at the same time. However, all these methods become complicated to implement in cases of noisy and 'short' time series like the ones available at 'UCR time series'. Moreover, the best τ for the Takens' theorem is not necessarily the best for the clustering. Indeed, we don't necessarily want a perfect representation of the dynamic system attractor, but rather to highlight the difference between clusters through specific geometrical features. Determining τ properly for real time series clustering therefore remains an open question.

References

[1] Henry D. I. Abarbanel and Matthew B. Kennel. Local false nearest neighbors and dynamical dimensions from observed chaotic data. *Phys. Rev. E*, 47(5):3057–3068, May 1993. Publisher: American Physical Society.

[2] Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering – A decade review. *Information Systems*, 53:16–38, October 2015.

[3] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396, 2003.

[4] G. Bouleux. Oblique projection pre-processing and tls application for diagnosing rotor bar defects by improving power spectrum estimation. *Mechanical Systems and Signal Processing*, 41(1):301 – 312, 2013.

[5] G. Bouleux, M. Dugast, and E. Marcon. Information topological characterization of periodically correlated processes by dilation operators. *IEEE Transactions on Information Theory*, 65(10):6484–6495, 2019.

[6] G. Bouleux, E. Marcon, and O. Mory. Early index for detection of pediatric emergency department crowding. *IEEE Journal of Biomedical and Health Informatics*, 19(6):1929–1936, 2015.

[7] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell,

Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7819, pages 160–172. Springer Berlin Heidelberg, 2013.

[8] E. Coviello, A. B. Chan, and G. Lanckriet. Time series models for semantic music annotation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):1343–1359, 2011.

[9] M. Dugast, G. Bouleux, and E. Marcon. Representation and characterization of nonstationary processes by dilation operators and induced shape space manifolds. *Entropy*, 20:717, 9 2018.

[10] M. Dugast, G. Bouleux, O. Mory, and E. Marcon. Improving health care management through persistent homology of time-varying variability of emergency department patient flow. *IEEE Journal of Biomedical and Health Informatics*, 23(5):2174–2181, 2019.

[11] Ehsan Elhamifar and Rene Vidal. Sparse Subspace Clustering: Algorithm, Theory, and Applications. *arXiv:1203.1005 [cs, math, stat]*, February 2013. arXiv: 1203.1005.

[12] Zhongke Gao and Ningde Jin. Complex network from time series based on phase space reconstruction. *Chaos*, 19(3):033137, 2009.

[13] T. Gautama, D.P. Mandic, and M.M. Van Hulle. A differential entropy based method for determining the optimal embedding parameters of a signal. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 6, pages VI–29, April 2003. ISSN: 1520-6149.

[14] Cyril Goutte, Peter Toft, Egill Rostrup, Finn Å. Nielsen, and Lars Kai Hansen. On Clustering fMRI Time Series. *NeuroImage*, 9(3):298–310, March 1999.

[15] Sigmundur Gudmundsson. An Introduction to Riemannian Geometry. page 130.

[16] Hossein Hassani, Saeed Heravi, and Anatoly Zhigljavsky. Forecasting UK Industrial Production with Multivariate Singular Spectrum Analysis. *Journal of Forecasting*, 32(5):395–408, 2013.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[18] Anil K. Jain. Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, June 2010.

[19] A. Javed, B. Suk Lee, and D.M. Rizzo. A benchmark study on time series clustering. *Machine Learning with Applications*, 1:100001, 2020.

[20] H. Karcher. Riemannian center of mass and mollifier smoothing. *Comm. Pure Appl. Math.*, 30(5):509–541, September 1977.

[21] Leonard Kaufman and Peter J Rousseeuw. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis*, 344:68–125, 1990.

[22] V. Kavitha and M. Punithavalli. Clustering Time Series Data Stream - A Literature Survey. *International Journal of Computer Science and Information Security*, 8(1), 2010.

[23] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *WIREs Data Mining Knowl Discov*, 1(3):231–240, 2011.

[24] Rui Li, Tai-Peng Tian, and Stan Sclaroff. Simultaneous Learning of Nonlinear Manifold and Dynamical Models for High-dimensional Time Series. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, 2007. IEEE.

[25] Bryan Matthews, Santanu Das, Kanishka Bhaduri, Kamalika Das, Rodney Martin, and Nikunj Oza. Discovering anomalous aviation safety events using scalable data mining algorithms. *Journal of Aerospace Information Systems*, 10(10):467–475, 2013.

[26] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2018.

[27] A. Mellit, A. Massi Pavan, and M. Benghanem. Least squares support vector machine for short-term prediction of meteorological time series. *Theoretical and applied climatology*, 111(1):297–307, 2013.

[28] Jianming Miao and Adi Ben-Israel. On Principal Angles between Subspaces in Rn. *Linear algebra and its applications*, 171(92):81–98, 1992.

[29] Lyle Noakes. The Takens embedding theorem. *International Journal of Bifurcation and Chaos*, 01(04):867–872, 1991.

[30] Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering Time Series with Hidden Markov Models and Dynamic Time Warping. In *Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning*, pages 17–21, 1999.

[31] Colin O'Reilly, Klaus Moessner, and Michele Nati. Univariate and Multivariate Time Series Manifold Learning. *Knowledge-Based Systems*, 133:1–16, 2017.

[32] Clement Pealat, Guillaume Bouleux, and Vincent Cheutet. Improved Time-Series Clustering with UMAP dimension reduction method. In *2014 22nd International Conference on Pattern Recognition*, 2020.

[33] Clément Pealat, Guillaume Bouleux, and Vincent Cheutet. Extracting Most Impacting Emergency Department Patient Flow By Embedding Laboratory-confirmed and Clinical Diagnosis on The Stiefel Manifold. In *2019 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 1–4, May 2019. ISSN: 2641-3590.

[34] Carl Rhodes and Manfred Morari. False-nearest-neighbors algorithm and noise-corrupted time series. *Phys. Rev. E*, 55(5):6162–6170, May 1997. Publisher: American Physical Society.

[35] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[36] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, November 1987.

[37] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3):1–21, 2017.

[38] Leonid Sigal, Alexandru Balan, and Michael Black. HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision*, 87:4–27, 2010.

[39] J. Stark, D.S. Broomhead, M.E. Davies, and J. Huke. Takens embedding theorems for forced and stochastic systems. *Nonlinear Analysis: Theory, Methods & Applications*, 30(8):5303–5314, 1997.

[40] Floris Takens. Detecting strange attractors in turbulence. In David Rand and Lai-Sang Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, volume 898, pages 366–381. Springer Berlin Heidelberg, 1981.

[41] Brian St Thomas, Lizhen Lin, Lek-Heng Lim, and Sayan Mukherjee. Learning Subspaces of Different Dimension. *arXiv:1404.6841 [math, stat]*, April 2014. arXiv: 1404.6841.

[42] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical Computations on Grassmann and Stiefel Manifolds for Image and Video-Based Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2273–2286, November 2011.

[43] Zhiguang Wang and Tim Oates. Imaging time-series to improve classification and imputation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[44] T. Warren Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11):1857–1874, 2005.

[45] Rui Xu and Don Wunsch. *Clustering*. John Wiley & Sons, 2008.

[46] Xiaohang Zhang, Jiaqi Liu, Yu Du, and Tingjie Lv. A novel clustering method on time series data. *Expert Systems with Applications*, 38(9):11891–11900, September 2011.

[47] Alan Edelman, Tomás A. Arias, and Steven T. Smith. The Geometry of Algorithms with Orthogonality Constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[48] Ralf Zimmermann. A matrix-algebraic algorithm for the Riemannian logarithm on the Stiefel manifold under the canonical metric. *arXiv:1604.05054 [math]*, April 2016. arXiv: 1604.05054.

Acknowledgement

This study was supported by a grant from the French Ministry of Health (PREPS-15-000668).

Author biographies

Prof. Vincent CHEUTET is full professor since 2014 at INSA Lyon. He is deputy director of the DISP Laboratory (Decision and Information System for Production). His main research topics are industrial information systems (especially PLM and ERP), knowledge management, industrial and mechanical engineering management.

Guillaume Bouleux received the Msc. And PhD degrees in 2005 and 2007 respectively. He is the co-leader of the operation management and optimisation team of the DISP laboratory. His field of application focuses on Healthcare systems and geometry along with the topology for stochastic processes are of his principal theoretical concern.

Clément PEALAT obtained his engineering degree in 2018. He then continued with a thesis in the DISP laboratory. His thesis is in the medical field in partnership with the CHU of Saint-Etienne. This led him to specialize in time series clustering.

	Frobenius				Sphere				Stiefel				Raw data			
Databases	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.
50words	0.442	0.621	0.187	0.546	0.190	0.036	0.298	0.545	0.208	0.045	0.245	0.532	0.704	0.695	0.186	0.531
Adiac	0.162	0.060	0.306	0.208	0.186	0.005	0.311	0.580	0.137	0.092	0.465	0.651	0.154	0.060	0.315	0.534
ArrowHead	0.390	0.051	0.378	0.451	0.445	0.540	0.553	0.489	0.499	0.051	0.319	0.607	0.463	0.051	0.000	0.371
Beef	0.265	0.265	0.376	0.104	0.182	0.238	0.214	0.104	0.265	0.265	0.160	0.104	0.265	0.265	0.376	0.104
BeetleFly	0.174	0.206	0.000	0.000	0.356	0.399	0.000	0.243	0.397	0.423	0.000	0.347	0.302	0.206	0.000	0.000
BirdChicken	0.313	0.307	0.000	0.221	0.170	0.283	0.000	0.158	0.067	0.283	0.000	0.342	0.320	0.313	0.000	0.233
Car	0.149	0.277	0.000	0.139	0.448	0.576	0.440	0.303	0.391	0.040	0.000	0.343	0.139	0.116	0.000	0.303
CBF	0.649	0.211	0.000	0.281	0.274	0.420	0.000	0.274	0.283	0.431	0.000	0.467	0.221	0.281	0.000	0.264
Chlo...tion	0.005	0.015	0.018	0.021	0.005	0.015	0.020	0.012	0.018	0.032	0.029	0.030	0.005	0.006	0.023	0.017
CinC...orso	0.599	0.655	0.388	0.485	0.902	0.902	0.687	0.902	0.663	0.902	0.662	0.902	0.655	0.637	0.388	0.198
Coffee	0.812	0.524	0.476	1.000	1.000	0.812	0.871	1.000	0.695	0.842	0.000	1.000	0.700	0.110	0.464	0.871
Computers	0.018	0.009	0.018	0.037	0.011	0.008	0.024	0.062	0.072	0.054	0.066	0.070	0.046	0.086	0.003	0.052
CricketX	0.049	0.062	0.135	0.122	0.198	0.011	0.060	0.381	0.142	0.296	0.098	0.413	0.047	0.071	0.132	0.315
CricketY	0.174	0.095	0.167	0.110	0.119	0.005	0.140	0.311	0.091	0.021	0.102	0.364	0.141	0.174	0.162	0.191
CricketZ	0.043	0.041	0.143	0.164	0.285	0.022	0.109	0.326	0.130	0.036	0.114	0.391	0.041	0.038	0.116	0.235
Diat...tion	0.648	1.000	0.000	0.789	0.641	0.884	0.000	0.606	0.875	0.504	0.000	0.629	0.928	1.000	0.000	0.000
Dist...roup	0.194	0.012	0.258	0.282	0.185	0.279	0.311	0.207	0.207	0.334	0.374	0.282	0.229	0.176	0.262	0.282
Dist...rect	0.051	0.009	0.065	0.024	0.046	0.009	0.120	0.023	0.053	0.070	0.115	0.021	0.028	0.009	0.078	0.012
Dist...nxTW	0.474	0.521	0.437	0.521	0.521	0.521	0.523	0.521	0.367	0.551	0.562	0.521	0.470	0.521	0.439	0.521
Earthquakes	0.006	0.022	0.008	0.020	0.000	0.007	0.000	0.009	0.044	0.019	0.007	0.043	0.064	0.077	0.008	0.025
ECG200	0.228	0.289	0.314	0.289	0.350	0.224	0.000	0.268	0.242	0.319	0.239	0.423	0.230	0.289	0.299	0.185
ECG5000	0.672	0.005	0.669	0.601	0.760	0.819	0.782	0.692	0.024	0.005	0.640	0.691	0.680	0.740	0.671	0.559
ECG....ays	0.000	0.015	0.000	0.002	0.028	0.231	0.000	0.059	0.073	0.015	0.003	0.136	0.015	0.015	0.000	0.007
FaceAll	0.055	0.004	0.456	0.559	0.727	0.121	0.549	0.772	0.515	0.136	0.224	0.737	0.059	0.054	0.152	0.565
FaceFour	0.381	0.537	0.000	0.588	0.687	0.736	0.000	0.601	0.044	0.813	0.000	0.698	0.484	0.486	0.000	0.417
FacesUCR	0.115	0.022	0.252	0.416	0.251	0.102	0.253	0.637	0.209	0.102	0.185	0.607	0.629	0.058	0.326	0.188

Table .4: Complete results for 16 methods and 79 databases (1/3)

	Frobenius				Sphere				Stiefel				Raw data			
Databases	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.
FISH	0.125	0.124	0.113	0.479	0.456	0.309	0.299	0.336	0.434	0.011	0.232	0.542	0.112	0.042	0.114	0.433
FordA	0.000	0.000	0.019	0.042	0.187	0.016	0.016	0.028	0.072	0.030	0.047	0.135	0.000	0.010	0.015	0.051
FordB	0.001	0.005	0.013	0.023	0.036	0.010	0.020	0.009	0.010	0.002	0.012	0.077	0.000	0.001	0.010	0.023
GunPoint	0.363	0.352	0.315	0.429	0.208	0.253	0.078	0.358	0.287	0.013	0.438	0.475	0.351	0.379	0.305	0.013
Ham	0.020	0.018	0.000	0.103	0.020	0.018	0.000	0.046	0.017	0.018	0.000	0.103	0.155	0.018	0.000	0.066
Haptics	0.098	0.017	0.070	0.156	0.079	0.017	0.081	0.182	0.018	0.017	0.000	0.106	0.121	0.017	0.090	0.091
Herring	0.003	0.021	0.000	0.005	0.017	0.021	0.000	0.005	0.043	0.039	0.000	0.030	0.001	0.021	0.000	0.005
InlineSkate	0.082	0.101	0.083	0.230	0.024	0.064	0.074	0.054	0.056	0.064	0.074	0.159	0.135	0.133	0.083	0.186
Inse...ound	0.520	0.548	0.360	0.293	0.475	0.423	0.288	0.290	0.200	0.324	0.279	0.467	0.547	0.557	0.357	0.552
Ital...emand	0.014	0.083	0.035	0.551	0.458	0.474	0.308	0.390	0.226	0.491	0.205	0.713	0.014	0.083	0.402	0.607
Lighting2	0.039	0.039	0.000	0.051	0.035	0.019	0.022	0.059	0.117	0.026	0.000	0.051	0.007	0.039	0.000	0.067
Lighting7	0.138	0.198	0.297	0.198	0.442	0.033	0.210	0.291	0.115	0.198	0.152	0.198	0.198	0.198	0.297	0.198
MALLAT	0.800	0.697	0.700	0.781	0.644	0.716	0.774	0.698	0.529	0.796	0.751	0.791	0.800	0.697	0.700	0.781
Meat	0.734	0.707	0.691	0.734	0.734	0.707	0.691	0.721	0.392	0.031	0.617	0.740	0.734	0.707	0.691	0.734
Med...ages	0.290	0.004	0.126	0.140	0.070	0.012	0.084	0.078	0.244	0.298	0.112	0.222	0.355	0.015	0.131	0.143
Midd...roup	0.019	0.112	0.139	0.129	0.051	0.139	0.139	0.139	0.007	0.113	0.078	0.139	0.013	0.111	0.139	0.111
Midd...rect	0.023	0.003	0.002	0.003	0.091	0.027	0.005	0.010	0.039	0.003	0.022	0.115	0.002	0.016	0.003	0.003
Midd...nxTW	0.461	0.466	0.454	0.506	0.506	0.506	0.506	0.506	0.485	0.424	0.477	0.506	0.484	0.470	0.443	0.506
MoteStrain	0.204	0.081	0.000	0.097	0.179	0.081	0.000	0.196	0.137	0.275	0.000	0.258	0.438	0.475	0.000	0.293
NonI...rax1	0.261	0.001	0.075	0.774	0.249	0.001	0.478	0.658	0.114	0.001	0.159	0.708	0.264	0.001	0.075	0.765
NonI...rax2	0.214	0.001	0.683	0.816	0.253	0.001	0.616	0.788	0.557	0.001	0.609	0.794	0.211	0.001	0.696	0.839
OliveOil	0.516	0.200	0.589	0.486	0.583	0.200	0.568	0.486	0.683	0.679	0.459	0.486	0.200	0.200	0.589	0.486
OSULeaf	0.072	0.322	0.273	0.317	0.372	0.485	0.440	0.402	0.263	0.390	0.248	0.434	0.318	0.424	0.304	0.285
Ohal...rect	0.005	0.009	0.065	0.003	0.000	0.002	0.071	0.001	0.003	0.007	0.072	0.057	0.003	0.002	0.048	0.002
Phoneme	0.505	0.500	0.000	0.254	0.528	0.010	0.138	0.255	0.267	0.026	0.097	0.327	0.589	0.569	0.091	0.290
Plane	0.796	0.841	0.893	0.891	0.798	0.923	0.929	0.948	0.788	0.870	0.853	0.948	0.823	0.841	0.893	0.831

Table .5: Complete results for 16 methods and 79 databases (2/3)

	Frobenius				Sphere				Stiefel				Raw data			
Databases	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.	Kmeans	Hier.	HDB.	U.+HDB.
Prox...roup	0.524	0.005	0.516	0.524	0.524	0.524	0.525	0.524	0.524	0.524	0.524	0.524	0.524	0.522	0.516	0.524
Prox...rect	0.056	0.012	0.066	0.033	0.038	0.057	0.097	0.034	0.014	0.050	0.088	0.055	0.063	0.012	0.055	0.034
Prox...nxTW	0.602	0.599	0.599	0.602	0.581	0.581	0.599	0.602	0.617	0.602	0.602	0.602	0.602	0.599	0.599	0.602
Refr...ices	0.025	0.030	0.010	0.021	0.019	0.004	0.054	0.059	0.002	0.045	0.004	0.086	0.038	0.063	0.010	0.041
ShapeletSim	0.278	0.042	0.000	0.000	0.126	0.287	0.000	0.000	0.054	0.027	0.000	0.012	0.147	0.165	0.000	0.000
ShapesAll	0.169	0.465	0.514	0.634	0.717	0.660	0.493	0.666	0.610	0.007	0.478	0.693	0.166	0.165	0.512	0.636
Smal...nces	0.056	0.043	0.053	0.082	0.024	0.005	0.085	0.091	0.019	0.065	0.072	0.076	0.079	0.063	0.053	0.068
Sony...face	0.115	0.157	0.000	0.070	0.115	0.157	0.000	0.025	0.099	0.416	0.000	0.140	0.490	0.157	0.000	0.038
Star...rves	0.610	0.604	0.736	0.656	0.760	0.760	0.471	0.750	0.504	0.760	0.525	0.760	0.612	0.607	0.736	0.760
Strawberry	0.084	0.062	0.224	0.143	0.145	0.017	0.056	0.293	0.031	0.048	0.164	0.249	0.086	0.004	0.228	0.195
SwedishLeaf	0.234	0.042	0.218	0.550	0.262	0.008	0.357	0.631	0.356	0.008	0.313	0.620	0.234	0.004	0.218	0.546
Symbols	0.000	0.651	0.000	0.000	0.691	0.872	0.000	0.547	0.840	0.654	0.000	0.875	0.787	0.810	0.000	0.580
ToeS...ion1	0.002	0.032	0.042	0.003	0.072	0.163	0.129	0.031	0.210	0.202	0.000	0.132	0.144	0.055	0.010	0.035
ToeS...ion2	0.280	0.360	0.017	0.043	0.593	0.430	0.000	0.616	0.341	0.126	0.000	0.585	0.309	0.317	0.059	0.125
Trace	0.669	0.669	0.600	0.669	0.014	0.457	0.599	0.237	0.485	0.588	0.462	0.484	0.669	0.669	0.669	0.498
TwoPatterns	0.012	0.017	0.170	0.209	0.036	0.020	0.022	0.168	0.000	0.002	0.015	0.126	0.013	0.031	0.174	0.198
TwoLeadECG	0.086	0.086	0.000	0.086	0.082	0.082	0.175	0.082	0.254	0.252	0.000	0.197	0.086	0.086	0.000	0.086
uWav...aryX	0.261	0.271	0.246	0.476	0.321	0.003	0.119	0.205	0.130	0.002	0.098	0.290	0.264	0.218	0.249	0.438
uWav...aryY	0.257	0.406	0.336	0.411	0.214	0.002	0.074	0.269	0.082	0.003	0.068	0.261	0.257	0.200	0.350	0.415
uWav...aryZ	0.383	0.447	0.410	0.548	0.320	0.007	0.057	0.237	0.063	0.011	0.113	0.329	0.408	0.292	0.405	0.540
uWav...yAll	0.537	0.668	0.583	0.800	0.673	0.656	0.304	0.643	0.497	0.699	0.198	0.719	0.630	0.758	0.582	0.755
Wafer	0.000	0.000	0.225	0.143	0.263	0.006	0.268	0.143	0.110	0.042	0.214	0.161	0.000	0.000	0.257	0.123
Wine	0.001	0.091	0.019	0.034	0.193	0.091	0.016	0.009	0.087	0.091	0.004	0.071	0.344	0.091	0.001	0.023
Wor...yms	0.326	0.345	0.176	0.356	0.395	0.514	0.158	0.337	0.265	0.037	0.228	0.400	0.335	0.344	0.221	0.368
Worms	0.003	0.199	0.190	0.116	0.209	0.031	0.141	0.136	0.280	0.270	0.149	0.166	0.018	0.252	0.144	0.095
Wor...lass	0.000	0.061	0.092	0.004	0.086	0.037	0.000	0.062	0.059	0.033	0.000	0.076	0.001	0.080	0.032	0.009
Yoga	0.008	0.014	0.056	0.041	0.020	0.000	0.054	0.035	0.005	0.006	0.066	0.051	0.008	0.010	0.054	0.040

Table .6: Complete results for 16 methods and 79 databases (3/3)