Improved Time-Series Clustering with UMAP dimension reduction method

Clément Pealat, Guillaume Bouleux, Vincent Cheutet

Improved Time-Series Clustering with UMAP dimension reduction method

Clément Pealat *Univ Lyon, INSA-Lyon*
DISP EA4570
Villeurbanne, France
clement.pealat@insa-lyon.fr Guillaume Bouleux *Univ Lyon, INSA-Lyon*
DISP EA4570
Villeurbanne, France
guillaume.bouleux@insa-lyon.fr Vincent Cheutet *Univ Lyon, INSA-Lyon*
DISP EA4570
Villeurbanne, France
vincent.cheutet@insa-lyon.fr

***Abstract*—Clustering is an unsupervised machine learning method giving insights on data without early knowledge. Classes of data are return by assembling similar elements together. Giving the increasing of the available data, this method is now applied in a lot of fields with various data types. Here, we propose to explore the case of time series clustering. Indeed, time series are one of the most classic data type, and are present in various fields such as medical or finance. This kind of data can be pre-processed by of dimension reduction methods, such as the recent UMAP algorithm. In this paper, a benchmark of time series clustering is created, comparing the results with and without UMAP as a pre-processing step. UMAP is used to enhance clustering results. For completeness, three different clustering algorithms and two different geometric representation for the time series (Classic Euclidean geometry, and Riemannian geometry on the Stiefel Manifold) are applied. The results are compared with and without UMAP as a pre-processing step on the databases available at UCR Time Series Classification Archive www.cs.ucr.edu/~eamonn/time_series_data/.**

I. Introduction

Considering unlabelled data, clustering is one of the most famous unsupervised machine learning methods[1]. It is a data mining method that will put in the same groups, named clusters, elements with similar behaviour. To do so, clustering algorithms aim to create clusters by maximizing similarities between the elements inside while minimizing similarities between the elements outside.

Several methods have been proposed in the literature. However, it can be difficult to determine if the results are reliable since the labels are not available. So, those methods must be evaluated beforehand on data with known labels.

Hence, we develop a benchmark based on time series data available at UCR Time Series Classification Archive[2][3]. The databases are composed of one-dimensional labelled time series. This datatype exists in various fields such as aviation[4], weather[5], industrial [6],...

To determine the similarity between two one-dimensional time series, a distance needs to be defined. The Euclidean distance can be used directly on the time series (seen as vectors). An other possibility to deal with one-dimensional time series is to embed the time series into their phase-space[7][8] using the Takens theorem[9]. It defines a matrix for each time serie and the similarity between two time series is now the distance between those matrices. To determine this distance, we apply the Riemannian geometry.

The main objective of this benchmark is to evaluate the efficiency of UMAP[10] as a pre-processing step for clustering algorithms. UMAP is a non-linear dimension reduction method that aims to keep the structure of high-dimensional data in a smaller dimension. This method is new, and its efficiency for clustering algorithms has not yet been clearly shown. Indeed, this algorithm raises some controversies for clustering because it can create false clusters due to noises.

Here, we propose to check the effects of UMAP on three different clustering algorithms: K-means[11], Agglomerative Hierarchic[12], and HDBSCAN[13]. After computing a metric (Euclidiean or Riemannian) between all the time series of a database, we can embed the time series into a small dimensional space by applying UMAP with respect to the distances. We apply those clustering algorithms with and without the pre-processing UMAP step and we compare the results.

Organization of the paper: In section II, we present the UMAP algorithm. Then, in section III, the clustering algorithms are detailed. In section IV, the geometric representation of the time series is explained. We use general results on Riemannian geometry to determine the distance, and the Karcher mean on the Stiefel manifold. With this two elements, we can then apply UMAP and clustering algorithms. Then, in section V, the results on the Euclidean geometry and the Stiefel manifold are presented.

II. UMAP: Uniform Manifold Approximation and Projection

A. Presentation

UMAP[10] realizes a reduction of dimension using the distance between each elements of a dataset. 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.(1)) 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 neighbour $Y_1, ..., Y_k$ (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 [14]. The parameter σ is defined by eq.(1). This parameter ensures that for each element Y of the data, the density of the circle of centre Y and of radius equal to the distance to the k-neighbours are sensibly the same for each Y.

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

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 element 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-neighbourhood of each element.
- A graph G' on $\mathbb{R}^m$ is obtained by a Laplacian eigenmaps dimension reduction method.
- The graph G' is modified by a forced directed graph layout algorithm so that the cross-entropy between G and G' is minimized.

B. UMAP for clustering

The first goal of UMAP is to visualize high-dimensional data[15]. To do so, the dimension is reduced at 2 or 3. But, since UMAP is a dimension reduction method, it can be used to clarify the data, and so, to increase the results of clustering algorithms. However, this is not compulsory. Indeed, UMAP cannot completely keep the density of the data when the dimension is reduced. It can create pseudo-groups that will disturb the clustering algorithms. Indeed, an article[16] shows it on T-SNE a similar reduction of dimension. But, it is in some particular cases, and we decide to check it on real databases.

To do so, we use particular parameters. Indeed, we want to keep the global structure of the data. So, the number of neighbours should be high, we put it at 30. This way, we take into account for each elements the action of its 30th closest neighbours, it limits the impact of noise that can create false clusters when only a small number of neighbours is kept. Moreover, since we are not interested by visualization, we decide to keep 10 components for the data.

III. Clustering algorithms

To be exhaustive, we decide to choose clustering algorithms of three distinct types: the K-means[11] algorithm among the top-down clustering algorithms, the Agglomerative Hierarchical[12] among the bottom-up methods (Hier.) and HDBSCAN[13] (Density-Based Clustering Based on Hierarchical Density Estimates) among the density-based methods.

A. K-means

For a given k, the K-means algorithm initializes randomly k pseudo-centres and each element of the dataset is affiliated with its closest pseudo-centre. Then, the k means of the k clusters created are the new pseudo-centre, and each element of the data is affiliated with those new means. This iteration is repeated until the results of clustering stop changing. So, to apply the K-means algorithm, we need to initially select a distance, a mean, and a number of clusters.

B. Agglomerative Hierarchical (Hier.)

At the beginning, Hier. algorithm considers each element of the dataset as one distinct clusters. Then, the nearest clusters are merged together and form a new cluster. This operation goes on until the wanted number of clusters is obtained.

C. Silhouette Score

Both K-means and Hier. need to know beforehand the number of clusters k. To determine this number without using the labels of the data, we use the silhouette score[17]. The silhouette score for a clustering calculates a score that represents the quality of the clustering. If the distance intra-clusters is low, and the distance inter-clusters is high, the

silhouette score gives a good value. So, we compute the clustering algorithms for several values of k, and we keep the number of clusters to optimize the value of the silhouette score.

D. HDBSCAN

HDBSCAN is a prolongation of the clustering algorithm Density-Based Spatial Clustering of Applications with noise (DBSCAN). We start by explaining DBSCAN[18][19].

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[19]. Moreover, DBSCAN has the ability to detect noise points. Two parameters must be chosen beforehand: the threshold ϵ and the number of neighbours m. A point A of the data is density reachable from another 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, DBSCAN determines the category of each point of the datasets:

- 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 1, from [18], is an example of DBSCAN clustering with m (number of neighbours) at two. The core points in red creates a cluster. B and C are not core points (not enough neighbours), 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.

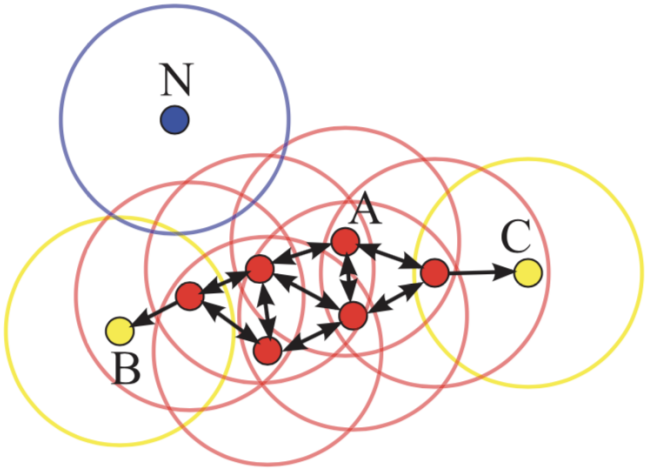

Fig. 1: Example of clustering by DBSCAN: N is a noise point, A (core point), B and C (border points) are in the same cluster

However, in this algorithm, the threshold ϵ is chosen by the user, and there is no correct answer on how to choose it 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 issue, we prefer 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 choose the value of a threshold.

IV. Geometric representations of the time series

A. Euclidean distance

One way to do it is to directly apply clustering algorithms to the data[20] [11]. It uses the Euclidean geometry : For a database D of n time series $(Y_i(t), t = 0, ...l-1, i = 0, ..., n-1)$ of length l, we consider each time series Y_i as a vector of $\mathbb{R}^l$. So, the distance is defined by: for $Y_i, Y_j \in \mathbb{R}^l, d(Y_i, Y_j) = (\sum_{k=0}^{l-1}(Y_i(k) - Y_j(k))^2)^{\frac{1}{2}}$ and the Karcher mean is the classical mean defined by: for $Y_1, ..., Y_k \in \mathbb{R}^l, \overline{Y} = \frac{1}{k}\sum_{i=1}^{k} Y_i$.

But, for time series data, it can be accurate to look at the phase space of the time series.

B. Phase space

In the case of high-dimensional time series (for example movement of a body through time[21]), one way to do the clustering is to determine the true space of the data[22][23].

Several articles show great results with this method[24][22]. We wanted to apply a similar method to our 1-D time series. To do so, the time series can be embedded in a structure similar to a high-dimensional time series, it defines a phase space[25][7]. The Takens Embedding theorem[9][26][27] confirms that the phase space is an accurate representation of the time series. There are several ways to realise this embedding, proposed for example in [7] and [8].

C. Stiefel Manifold

1) Embedding: For a time series $Y = (Y(0), ..., Y(m))$, we determine the trajectory matrix as proposed in figure 2. To have a good representation, we choose $n - m$ such that $n - m = \lfloor m/2 \rfloor$[28]

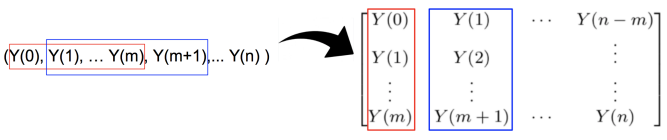

Fig. 2: Example of construction of the trajectory matrix

Those trajectory matrices are rectangular matrix, and have an important noise. We operate a reduction of dimension with UMAP. UMAP realizes a non-linear reduction of dimension. Then, we orthogonalize the reduced matrices to clarify the space generated by the trajectory. So, each time series of the databases are now embedded into the Stiefel manifold defined by : $V_p(\mathbb{R}^n) = \{A \in \mathbb{R}^{n\times p} : A^TA = Id\}$. The Stiefel manifold has not an Euclidean geometry, we need to define the Riemannian geometry to determine the distance and the Karcher mean on the Stiefel manifold with respect to the curvature of this manifold.

2) Generality about manifold: A manifold M is a curve space that can be seen locally as an Euclidean space. The geometry associated is called the Riemannian geometry[29].

The distance on a manifold between two elements is seen as the length of the shortest curve γ that goes from one to another (see equ.(2)).

$$\forall U_1, U_2 \in M, d_M(U_1, U_2) = \min_{\gamma(0)=U_1, \gamma(1)=U_2} L(\gamma) \quad (2)$$

The tangent space at A of M is defined by the tangent at A of all the curves passing by A. In figure 3, from [22], the manifold is a sphere. For two elements P_1 and P_2, the tangent vectors define the tangent plane T_{P1} and T_{P2}.

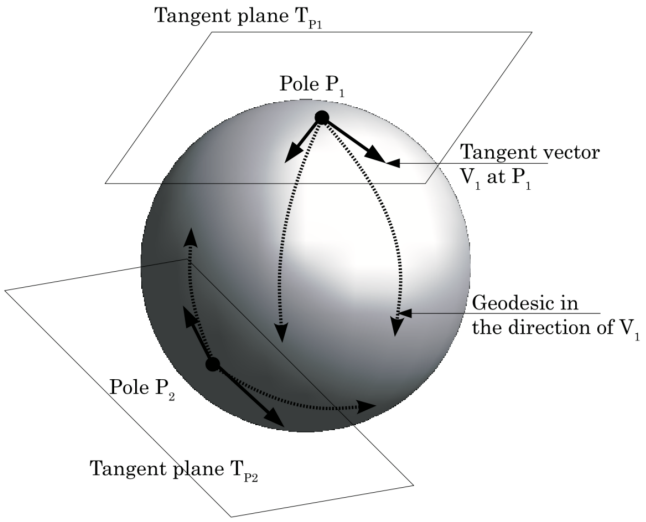

Fig. 3: Illustration on a sphere of the geodesics and the tangent planes

The Riemannian exponential projects a tangent vector to the closest point on the manifold. In figure 4, from [22], two tangent vectors at P are projected into the manifold using the Riemannian exponential.

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

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

To determine this mean from eq.(3), we can use a gradient descent algorithm. To do so, we derivate the equation (3),

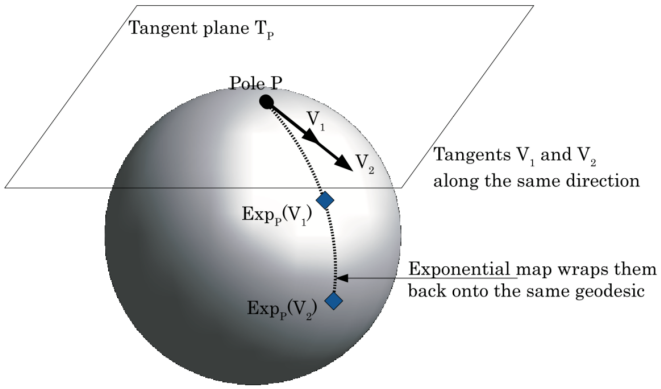

Fig. 4: Riemannian Exponential on a sphere

as proposed by [30], and we obtain a new definition for the Karcher mean $\overline{X}$:

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

3) Geometry of the Stiefel manifold: The Stiefel manifold[31][32] is a submanifold of $\mathbb{R}^{n\times p}$ defined by $V_p(\mathbb{R}^n) = \{A \in \mathbb{R}^{n\times p} : A^TA = Id\}$. For two elements A, B of $V_p(\mathbb{R}^n)$, it exists an element of $O(n)$ (orthogonal group of dimension n), such that $CA = B$. It allows us to see the distance between two elements of $V_p(\mathbb{R}^n)$ as the norm of their principal angles. Figure 5 illustrates an example of the distance between two elements A, B of $V_{3,2}$. A base of A is defined by the two vectors a_1, a_2 in blue and the base B is defined by the two vectors b_1, b_2 in blue. Then, the two principal angles θ_1, θ_2 between A, B are defined by $\theta_1 = (a_1, b_1), \theta_2 = (a_2, b_2)$. To

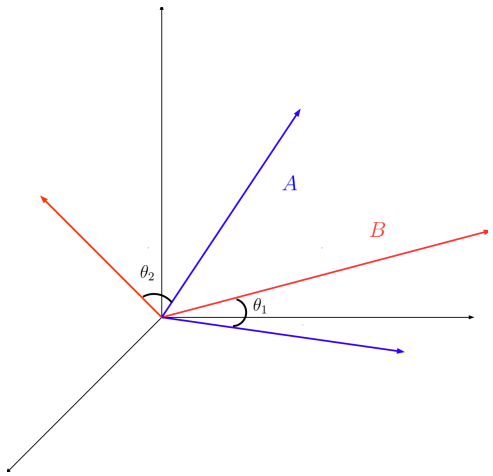

Fig. 5: Example of Principal Angle on $\mathbb{R}^3$

determine the principal angles between A, B, we compute the $arccos$ of the singular values of A^TB.

To determine the Karcher mean on the Stiefel manifold, we need to compute the Riemannian exponential and Riemannian logarithm associated at the Stiefel manifold. The algorithms [33] **??** and **??** allow us to compute those two functions.

Algorithm 1: 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 $(I - UU^T)\Delta$:
$\Delta = UA + Q_E R_E$

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

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

Algorithm 2: 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: $Log_U^{St}(\overline{U}) = UA_k + QB_k$

V. Results

To determine the quality of the clustering, we use the v-measure score [34]. The v-measure score compares the expected labels and the clustering results, using an evaluation based on entropy. It computes the harmonic mean between the homogeneity and the completeness score. Homogeneity and Completeness give a score between 0 and 1. For the homogeneity, the score of 1 is given if all the elements of each cluster have the same 'true' labels. By symmetry, completeness score is at maximum if all elements of each 'true' label are in the same cluster. The v-measure score realizes a balance between those two indicators. It is necessary because, for example, the completeness is at 1 if each element has its own clusters.

A. Databases

At UCR Time Series Classification Archive, 85 databases are available. A brief summary of those databases are presented in table I. The number of clusters in a database varies from 2 (only two labels) to 60. Moreover, some databases have a great number of time series to cluster and others are quite small. For a given database, all the time series have the same length (from 24 min. to 2709 max.)

B. Results on Euclidean geometry

On table II, we present:

- The mean of the v-measure score for each method of clustering across the 85 databases
- The standard deviation of the v-measure score
- The number of databases for which a method is the best one. For example, HDBSCAN alone has the best results for 12 databases (among 85 databases)
- For a given algorithm the percentage of databases for which the v-measure score is enhanced with UMAP (ex: UMAP with k-means is better than k-means for 61% of the databases.

First, we can remark that UMAP is a good pre-processing step whatever the clustering algorithm is. Indeed, UMAP increases the v-measure score in a majority of databases for the 3 algorithms, and this is confirmed by the mean of the v-measure score. However, the standard deviation increases a bit for K-means and hierarchic algorithm. So, the results are less stable. But, for HDBSCAN, UMAP clearly improves the results for all of those indicators.

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 method. 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.

Those figures 6,7 and 8 give a visual representation of the good results obtained by UMAP. On figure 6, we see that UMAP clearly enhances HDBSCAN in a vast majority of cases. Moreover, for some databases, UMAP + HDBSCAN gives good results whereas HDBSCAN is completely off.

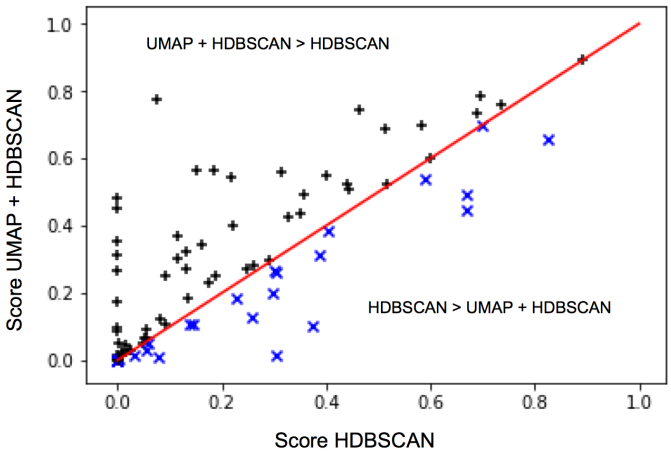

Fig. 6: Databases represented by v-measure score with UMAP+HDBSCAN in function of v-measure score with HDBSCAN

UMAP is particulary efficient for HDBSCAN, a density-based clustering algorithms. It was expected, since UMAP also uses the data density to reduce the dimensions.

The results does not seem to be intrinsic at the databases. Indeed, only 26 databases presents better results with UMAP for the 3 clustering algorithms. We can take also the example of k-means and hierarchical algorithm. UMAP presents better results for k-means on 52 databases, and on 56 databases for hierarchical algorithm. Only 35 databases are in common.

At last, across the six methods, UMAP with HDBSCAN presents the best results. It has the best mean and has the best score for 25 of the 85 databases. We propose in figure 9 a

Number of databases	85		
	Min	**Mean**	**Max**
Nb of clusters	2	7.5	60
Nb of time series	16	432	8926
Length of time series	24	422	2709

TABLE I: Presentation of the databases

	Kmeans	**Hier.**	**HDB.**	**UMAP+Kmeans**	**UMAP+Hier.**	**UMAP+HDB.**
Mean	0.253	0.212	0.218	0.284	0.273	0.292
Best with UMAP				61%	66%	75%

TABLE II: Summary of the Euclidean results

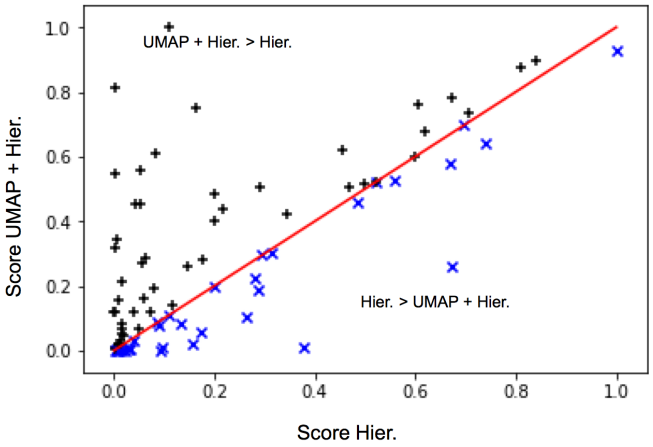

Fig. 7: Databases represented by v-measure score with UMAP+HDBSCAN in function of v-measure score with HDBSCAN

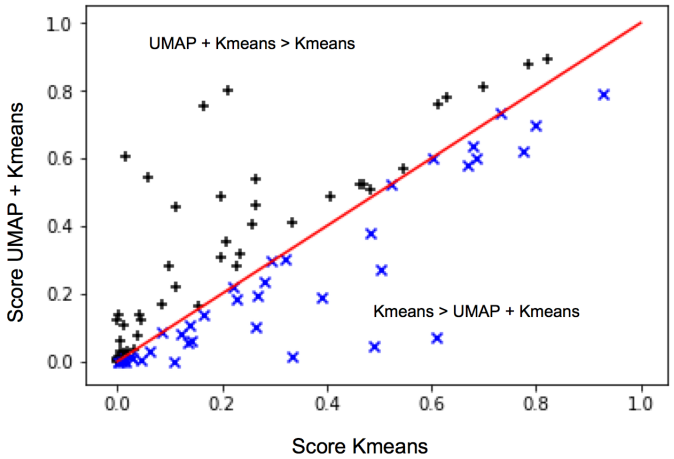

Fig. 8: Databases represented by v-measure score with UMAP + K-means in function of v-measure score with K-means

comparison with the most classic/used clustering algorithms the k-means. UMAP with HDBSCAN has a better mean than k-means and gives better results for 62% of the databases.

C. Results on the Stiefel Manifold

For the Stiefel Manifold, the computation of the Karcher mean takes some time, as the rest of the calculus. So, we

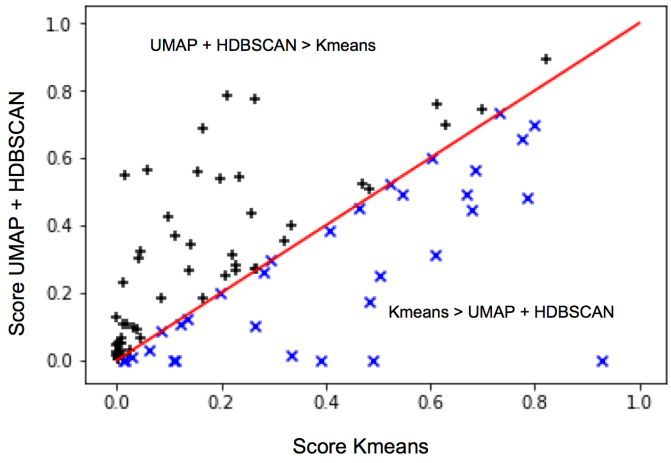

Fig. 9: Databases represented by v-measure score with UMAP + HDBSCAN in function of v-measure score with K-means

decide to apply the data transformation only on 6 databases. The results on the Stiefel Manifold are summarized on table III. It gives similar results: UMAP enhances the clustering score, and the duo UMAP + HDBSCAN gives the best results.

VI. Conclusion

Clustering algorithms are unsupervised method. An evaluation before a general utilisation is needed. So, on this paper, we propose a benchmark on time series clustering to evaluate the efficiency of UMAP to enhance clustering results. Using 85 labelled databases, we compute the results with and without UMAP. UMAP shows great results. Indeed, on the Euclidean geometry, UMAP increases the score of clustering in a large majority of the cases. Moreover, UMAP seems to be particulary efficient with density-based clustering algorithms.

On a few cases with a Riemannian geometry, the results have been consistent: UMAP gives an improvement.

So, UMAP enhances in general the clustering score, and the best method to use with is HDBSCAN. Compared to k-means (the most known method), UMAP coupled with HDBSCAN gives a better results for 62% of the databases.

References

[1] R. Xu and D. Wunsch, *Clustering*, en. John Wiley & Sons, Nov. 2008, Google-Books-ID: kYC3YCyl_tkC, ISBN: 978-0-470-38278-3.

	K-means	Hier.	HDBSCAN	UMAP + K-means	UMAP + Hier.	UMAP + HDBSCAN
Mean	0.203	0.173	0.182	0.292	0.213	0.301
Std	0.202	0.179	0.186	0.227	0.256	0.270
Times best results	1	1	0	2	1	2
Pre-processing UMAP best results				50%	67%	83%

TABLE III: Summary of the Stiefel Manifold results

[2] M. Corduas and D. Piccolo, "Time series clustering and classification by the autoregressive metric," en, *Computational Statistics & Data Analysis*, vol. 52, no. 4, pp. 1860–1872, Jan. 2008, ISSN: 0167-9473. DOI: 10.1016/j.csda.2007.06.001.

[3] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 9, pp. 2522–2535, Sep. 2015, ISSN: 2326-3865. DOI: 10.1109/TKDE.2015.2416723.

[4] B. Matthews, S. Das, K. Bhaduri, K. Das, R. Martin, and N. Oza, "Discovering anomalous aviation safety events using scalable data mining algorithms," *Journal of Aerospace Information Systems*, vol. 10, no. 10, pp. 467–475, 2013.

[5] A. Mellit, A. M. Pavan, and M. Benghanem, "Least squares support vector machine for short-term prediction of meteorological time series," en, *Theor Appl Climatol*, vol. 111, no. 1, pp. 297–307, Jan. 2013, ISSN: 1434-4483. DOI: 10.1007/s00704-012-0661-7.

[6] H. Hassani, S. Heravi, and A. Zhigljavsky, "Forecasting UK Industrial Production with Multivariate Singular Spectrum Analysis," en, *Journal of Forecasting*, vol. 32, no. 5, pp. 395–408, 2013, ISSN: 1099-131X. DOI: 10.1002/for.2244.

[7] C. O'Reilly, K. Moessner, and M. Nati, "Univariate and Multivariate Time Series Manifold Learning," en, *Knowledge-Based Systems*, vol. 133, pp. 1–16, Oct. 2017, ISSN: 09507051. DOI: 10.1016/j.knosys.2017.05.026.

[8] Z. Gao and N. Jin, "Complex network from time series based on phase space reconstruction," en, *Chaos*, vol. 19, no. 3, p. 033 137, Sep. 2009, ISSN: 1054-1500, 1089-7682. DOI: 10.1063/1.3227736.

[9] L. Noakes, "The takens embedding theorem," *Int. J. Bifurcation Chaos*, vol. 01, no. 04, pp. 867–872, Dec. 1991, ISSN: 0218-1274. DOI: 10.1142/S0218127491000634.

[10] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," en, *arXiv:1802.03426 [cs, stat]*, Dec. 2018, arXiv: 1802.03426.

[11] T. Warren Liao, "Clustering of time series data—a survey," en, *Pattern Recognition*, vol. 38, no. 11, pp. 1857–1874, Nov. 2005, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2005.01.025.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," en, *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, Sep. 1999, ISSN: 03600300. DOI: 10.1145/331499.331504.

[13] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia conference on knowledge discovery and data mining*, 2013, pp. 160–172.

[14] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," en, *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976603321780317.

[15] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using UMAP," en, *Nature Biotechnology*, vol. 37, no. 1, pp. 38–44, Jan. 2019, Number: 1 Publisher: Nature Publishing Group, ISSN: 1546-1696. DOI: 10.1038/nbt.4314.

[16] E. Schubert and M. Gertz, "Intrinsic t-Stochastic Neighbor Embedding for Visualization and Outlier Detection," en, in *Similarity Search and Applications*, C. Beecks, F. Borutta, P. Kröger, and T. Seidl, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, pp. 188–203, ISBN: 978-3-319-68474-1. DOI: 10.1007/978-3-319-68474-1_13.

[17] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. DOI: 10.1016/0377-0427(87)90125-7.

[18] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN," en, *ACM Trans. Database Syst.*, vol. 42, no. 3, pp. 1–21, Jul. 2017, ISSN: 03625915. DOI: 10.1145/3068335.

[19] H. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," en, *WIREs Data Mining Knowl Discov*, vol. 1, no. 3, pp. 231–240, May 2011, ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.30.

[20] V. Kavitha and M. Punithavalli, "Clustering Time Series Data Stream - A Literature Survey," *arXiv:1005.4270 [cs]*, May 2010, arXiv: 1005.4270.

[21] L. Sigal, A. Balan, and M. Black, "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion," *International Journal of Computer Vision*, vol. 87, pp. 4–27, Mar. 2010. DOI: 10.1007/s11263-009-0273-6.

[22] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical Computations on Grassmann and Stiefel Manifolds for Image and Video-Based Recognition," en, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2273–2286, Nov. 2011, ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2011.52.

[23] A. B. Chan and N. Vasconcelos, "Modeling, Clustering, and Segmenting Video with Mixtures of Dynamic Textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, May 2008, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2007.70738.

[24] R. Li, T.-P. Tian, and S. Sclaroff, "Simultaneous Learning of Nonlinear Manifold and Dynamical Models for High-dimensional Time Series," en, in *2007 IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, Brazil: IEEE, 2007, pp. 1–8, ISBN: 978-1-4244-1630-1. DOI: 10.1109/ICCV.2007.4409044.

[25] Z. Wang and T. Oates, "Imaging Time-Series to Improve Classification and Imputation," en, *arXiv:1506.00327 [cs, stat]*, May 2015, arXiv: 1506.00327.

[26] F. Takens, "Detecting strange attractors in turbulence," en, in *Dynamical Systems and Turbulence, Warwick 1980*, D. Rand and L.-S. Young, Eds., vol. 898, Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 366–381, ISBN: 978-3-540-11171-9 978-3-540-38945-3. DOI: 10.1007/BFb0091924.

[27] J. Stark, D. Broomhead, M. Davies, and J. Huke, "Takens embedding theorems for forced and stochastic systems," en, *Nonlinear Analysis: Theory, Methods & Applications*, vol. 30, no. 8, pp. 5303–5314, Dec. 1997, ISSN: 0362546X. DOI: 10.1016/S0362-546X(96)00149-6.

[28] G. Bouleux, E. Marcon, and O. Mory, "Early index for detection of pediatric emergency department crowding," *IEEE journal of biomedical and health informatics*, vol. 19, no. 6, pp. 1929–1936, 2015.

[29] S. Gudmundsson, "An Introduction to Riemannian Geometry," *Lecture Notes version*, pp. 1–235, 2004.

[30] H. Karcher, "Riemannian center of mass and mollifier smoothing," en, *Comm. Pure Appl. Math.*, vol. 30, no. 5, pp. 509–541, Sep. 1977, ISSN: 00103640, 10970312. DOI: 10.1002/cpa.3160300502.

[31] P.-A. Absil, R. Mahony, and R. Sepulchre, "Riemannian Geometry of Grassmann Manifolds with a View on Algorithmic Computation," en, *Acta Applicandae Mathematicae*, vol. 80, no. 2, pp. 199–220, Jan. 2004, ISSN: 0167-8019. DOI: 10.1023/B:ACAP.0000013855.14971.91.

[32] A. Edelman, T. A. Arias, and S. T. Smith, "The Geometry of Algorithms with Orthogonality Constraints," en, *SIAM Journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, Jan. 1998, ISSN: 0895-4798, 1095-7162. DOI: 10.1137/S0895479895290954.

[33] R. Zimmermann, "A matrix-algebraic algorithm for the Riemannian logarithm on the Stiefel manifold under the canonical metric," en, *arXiv:1604.05054 [math]*, Apr. 2016, arXiv: 1604.05054.

[34] A. Rosenberg and J. 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)*, 2007, pp. 410–420.