Riemannian Principal Component Analysis for Any Type of Data

Oldemar Rodríguez Rojas

Abstract This paper introduces an innovative extension of Principal Component Analysis (PCA), moving beyond the traditional assumption of data residing in Euclidean space to accommodate data on Riemannian manifolds. The primary challenge addressed is the lack of vector space operations on such manifolds. Fletcher et al., in their work *Principal Geodesic Analysis for the Study of Nonlinear Statistics of Shape*, proposed Principal Geodesic Analysis (PGA) as a geometric approach to analyze data on Riemannian manifolds, particularly effective for structured datasets like medical images, where the manifold's intrinsic structure is apparent. However, PGA's applicability is limited when dealing with general datasets that lack an implicit local distance notion. In this work, we introduce a generalized framework, termed *Riemannian Principal Component Analysis (R-PCA)*, to extend PGA for any data endowed with a local distance structure. Specifically, we adapt the PCA methodology to Riemannian manifolds by equipping data tables with local metrics, enabling the incorporation of manifold geometry. This framework provides a unified approach for dimensionality reduction and statistical analysis directly on manifolds, opening new possibilities for datasets with region-specific or part-specific distance notions, ensuring respect for their intrinsic geometric properties.

Keywords: Riemannian Manifold, Riemannian Principal Component Analysis (R-PCA), Riemannian Statistics, Local Distance Notion, Dimension Reduction Geometric Structures, Riemannian Statistics.

Oldemar Rodríguez Rojas
School of Mathematics, CIMPA, University of Costa Rica, San José, Costa Rica, e-mail: oldemar.rodriguez@ucr.ac.cr

1 Introduction

Each point in a data table can be imagined as a star or planet in the universe, especially when dealing with big data issues. In the universe, due to the infinitely different sizes of constellations, there are vastly different perceptions of distances between celestial bodies. For example, two constellations or galaxies that appear to be the same size from a distance (from Earth, for example) could be infinitely different, and one could even fit inside the other in a very small portion or empty space within it. For this reason, especially in problems involving Big Data, *thinking that the data is in Euclidean space is just as wrong as thinking that the earth is flat*.

Similarly, in data, there are local notions of distance corresponding to different regions of the data, and this should be considered when calculating indices or statistical models, as Principal Component Analysis. To address this, we propose considering that the data exists within a Riemannian manifold, where these local notions of distance can be effectively taken into account.

In [2] *Fletcher et al.*, the authors had proposed the Principal Geodesic Analysis on Riemannian manifolds through the use of geometry. This concept works particularly well when analyzing data derived from images, such as medical images, where the intrinsic Riemannian manifold structure is evident. However, this idea is not readily applicable to general data where there are no implicit notions of local distance. The *Riemannian Principal Component Analysis* that we propose go beyond of what was mentioned in the previous paragraph. The core concept is to impart a Riemannian manifold structure to any given set of data. This approach enables the assignment of local notions of distance to the data, thereby enhancing our ability to capture the internal structure of the data. This, in turn, leads to a significant improvement in the results of various statistical analyses as well as their interpretability.

The diagram in Figure 1 illustrates the transformation of data analysis methodologies from a Euclidean framework to a Riemannian geometric framework. It begins with a data table residing in Euclidean space, where classical techniques such as statistics, machine learning, and artificial intelligence are traditionally applied, relying on global distance metrics and vector space operations. By applying a specific method or algorithm, the data is endowed with local metrics that capture intrinsic geometric properties, enabling a transition to a Riemannian space. In this Riemannian space, data analysis leverages curvature, local distances, and local structures, transforming classical methodologies into Riemannian Statistics, Riemannian Machine Learning, and Riemannian AI. This paradigm shift allows for more accurate and meaningful analyses of data that naturally resides on curved manifolds, such as shapes or other structured datasets, by respecting their underlying geometric nature.

In this work, we propose to transform the Euclidean space into a Riemannian manifold using the UMAP (Uniform Manifold Approximation and Projection) algorithm, see McInnes et al. [5]. As described in the paper *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, the transformation is based on the construction of local metrics derived from a graph representation of the data. Specifically, UMAP methodology, detailed in Section 3.1 Graph Construction of the referenced paper, k-nearest neighbor graph where local distances between data

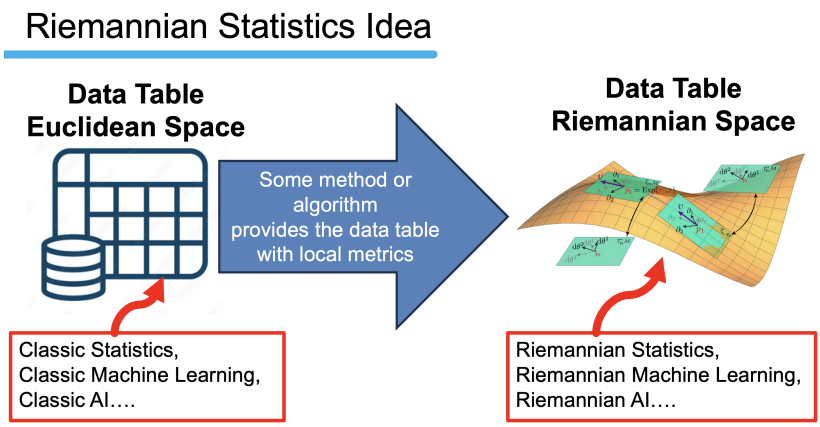

Fig. 1 Transforming an Euclidean into a Riemannian Space.

points are computed and normalized. These distances are then used to define local Riemannian metrics, effectively approximating a Riemannian structure for the data manifold. By leveraging the graph local connectivity properties and its edge weights, UMAP provides an efficient and scalable way to construct Riemannian metrics, enabling the analysis of complex datasets in their intrinsic geometric space. This transformation allows for the application of Riemannian statistics and machine learning methods, capturing both the local and global structures of the data.

UMAP is a novel technique for manifold learning and dimension reduction. Utilizing simplicial complexes, Čech complexes, and the Nerve theorem, UMAP gains additional benefits from this Riemannian metric-based approach. It generates a local metric space associated with each point, allowing for meaningful distance measurements. Consequently, the algorithm can assign weights to edges in a graph (simplicial complex), signifying the local metric-based separation between the original points. So the idea that we proposed in this paper is to use the local notions of distance that the UMAP algorithm generates in any data table to provide it with local distance. In this way, the data table can be conceptualized as a Riemannian manifold, incorporating these local distance.

UMAP, as a successor to t-SNE method, inherits a controversy associated with the t-SNE method. The challenge with t-SNE lies in its inability to preserve distances and density effectively. It only partially maintains the concept of *nearest-neighbors*. Though the distinction may seem subtle, it has implications for any clustering algorithm based on density or distance. This issue is somewhat controversial, and should be approached with caution. A comprehensive discussion on this topic can be found at https://umap-learn.readthedocs.io/en/latest/clustering.html.

Despite these concerns, there are still valid reasons to utilize UMAP as a preprocessing step for clustering. As highlighted in the discussion, when applied to real high-dimensional datasets such as `MNIST` data [1] or `cell RNA-seq` data [3], and with appropriate parameterization, both t-SNE and UMAP yield significantly

better clustering results than other algorithms. Regardless, for Riemannian Principal Component Analysis, the crucial aspect is that UMAP maintains the concept of *nearest-neighbors* in the low-dimensional representation of the dataset. This is very important as it provides the data table with local distance notions, enhancing the utility of the UMAP algorithm in this context.

2 Providing to a classical data table with a Riemannian manifold structure

UMAP method was designed to improve the main limitations of the t-SNE method. t-SNE means t-distributed Stochastic Neighbor Embedding and it was proposed by Laurens van der Maaten, see all the detail of this method in [4]. UMAP algorithm is competitive with t-SNE for visualization quality and it improves t-SNE limitations. UMAP (Uniform Manifold Aproximation and Projection) is an algorithm for dimension reduction based on algebraic topology, topological data analysis and Riemannian geometry. It was proposed by the Mathematician Leland McInnes in [5]. UMAP works in a similar way to t-SNE, it finds distances in a space with many variables and then tries to reproduce these distances in a low-dimensional space. But UMAP does it very differently because more than distances it tries to reproduce the topology, not necessarily the geometry. UMAP assumes that data is distributed along a Riemannian manifold. A manifold is a uniform n-dimensional geometric shape in which, for each point of this manifold, there is a neighborhood around that point that looks like a flat two-dimensional plane. Riemannian manifolds admit local notions of distances, area and angles. To explain the UMAP method we need to define the notion of k-simplex and simplicial complexes.

Let $\{x_0, \ldots, x_k\}$ be points in $\mathbb{R}^n$. We will assume that these points satisfy the condition that the set of vectors in $\mathbb{R}^n$ represented by the differences with respect to x_0, that is $\{x_1 - x_0, x_2 - x_0, \ldots, x_k - x_0\}$ are linearly independent.

Definition 1 The k-simplex generated by the points $\{x_0, \ldots, x_k\}$ is the set of all points $z = \sum_{i=0}^{k} a_i x_i$, where $\sum_{i=0}^{k} a_i = 1$. For a given z, we refer to a_i as the i-th barycentric coordinate.

Simplicial complexes are generalizations of graphs. A simplicial complex S in $\mathbb{R}^n$ is a set of simplices such that every face of a simplex in S is also a simplex in S. The intersection of two simplices in S is a face of each of them. Given data set presented as a finite metric space, we need to produce a simplicial complex such that the algebraic invariants of the simplicial complex reflect the shape of the data. To do that, we need to make the connection between clustering and components precise, via single-linkage clustering, which works as follows.

1. Choose a parameter ϵ.
2. Assign two points x and y to the same group if they are connected by a path of points (for some k) $x = x_0, x_1, x_2, \ldots, x_{k-1}, x_k = y$ such that each point x_i is at a distance ϵ from x_{i+1}. See the Figure 2.

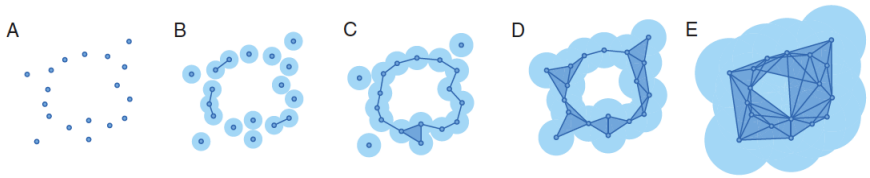

Fig. 2 As ϵ increases, more and more simplices are added to the simplicial complex and topological features emerge. In panels C and D, a circle can be detected.

The Nerve Theorem and its corollary are the fundamental theoretical basis that allows us to go from topological spaces to simplicial complexes and then to data. The Čech complex allows us to demonstrate that there exists a homeomorphism between the union of balls (determined by the parameter ϵ) and the nerve and therefore we will have a bijection between the data and the simplicial complexes.

Definition 2 The nerve $N(\mathcal{U})$ of a cover $\mathcal{U} = \{U_i\}$ of topological space X is the simplicial complex with vertices corresponding to the sets $\{U_i\}$ and a k-simplex $[j_0, j_1, \ldots, j_k]$ when the intersection $U_{j_0} \cap U_{j_1} \cap U_{j_2} \cap \cdots \cap U_{j_k} \neq \emptyset$.

Definition 3 Let $X \subset \mathbb{R}^n$ be a finite subspace and fix $\epsilon > 0$. The Čech complex $C_\epsilon(X, \partial_X)$ is the simplicial complex with vertices the points of X, and a k-simplex $[v_0, v_1, \ldots, v_k]$ when a set of points $\{v_0, v_1, \ldots, v_k\} \subset X$ satisfies $\bigcap_i B_\epsilon(v_i) \neq \emptyset$.

Theorem 1 (Nerve Theorem) *Let X be a topological space. Let $\mathcal{U} = \{U_i\}$ be an open cover of X such that all non-empty finite intersections $U_{j_1} \cap U_{j_2} \cap \cdots \cap U_{j_k}$ are contractible (homotopy equivalent to a point). Then the nerve (the geometric realization) $N(\mathcal{U})$ is homotopy equivalent to X.*

Corollary 1 *Let $X \subset \mathbb{R}^n$ be a finite subspace and fix $\epsilon > 0$. There exists a homeomorphism: $\bigcup_{x \in X} B_\epsilon(x) \cong |C_\epsilon(X, \partial_X)|$ between the union of balls and the nerve $N(\mathcal{U})$ (the geometric realization) of the Čech complex.*

The above guarantees that there exists a homeomorphism between the union of balls and the nerve, so, there is relation one-to-one (bijection) between data and Čech complex, as it is illustrated in the Figure 3.

To apply these ideas, UMAP choose a radius from each point, connecting points when those radii overlap, then we can create a simplicial complex using 0, 1, and 2 simplexes as points, lines, and triangles. Choosing this radius is critical, too small choice will lead to small, isolated clusters, while too large choice will connect everything together. UMAP overcomes this challenge by choosing a radius locally, based on the local distance to each point to the k-th nearest neighbor. To do that, Riemannian Geometry is used.

Definition 4 Fixed x, a **Riemannian metric** is defined by a scalar products $\langle \cdot, \cdot \rangle_x$ on each tangent space $T_x\mathcal{M}$ at points x of the manifold. For each x, each such scalar product is a positive definite bilinear map $\langle \cdot, \cdot \rangle_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$. The inner product gives a norm $\| \cdot \|_x : T_x\mathcal{M} \to \mathbb{R}$ by $\|v\|^2 = \langle v, v \rangle_x$

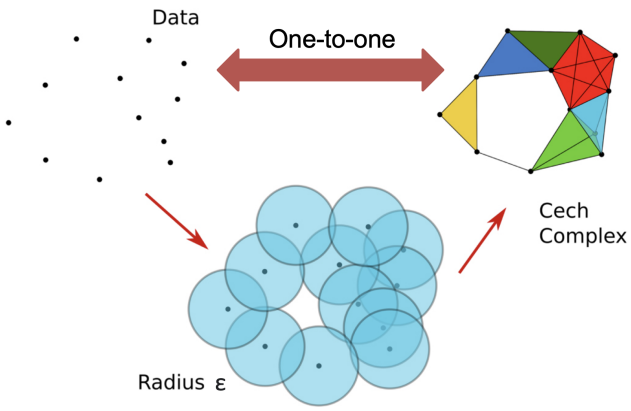

Fig. 3 Relation one-to-one between data and Čech complex.

The choice of k determines how locally we wish to estimate the Riemannian metric. A small choice of k means we want a very local interpretation, while, choosing a large k means our estimates will be based on larger regions. *This is very important, because it means that the UMAP algorithm provides the data table with local distance notions.* To the graph construction, UMAP algorithm begins by constructing a weighted k-nearest neighbor graph from the dataset $X = \{x_1, \ldots, x_n\}$, where a distance metric d defines the distances between points. The process can be described as follows:

1. The dataset X consists of points in a high-dimensional space, and a distance metric d (e.g., Euclidean distance) is used to compute the similarity between points.
2. The algorithm identifies the k-nearest neighbors using the distance metric d. This step produces a directed graph where each point has outgoing edges to its k-nearest neighbors.
3. The local parameters, ρ_i and σ_i, are defined as follows:

   - ρ_i: Local Connectivity. For each x_i, ρ_i is the minimum distance to its nearest neighbor that is greater than zero:

   $$\rho_i = \min\{d(x_i, x_{i_j}) \mid 1 \leq j \leq k,\ d(x_i, x_{i_j}) > 0\}.$$

   This ensures that x_i connects to at least one other point with an edge of weight 1.
   - σ_i: Local Scaling. σ_i normalizes the distances of all neighbors, ensuring consistency in the local metric. It is computed by solving:

   $$\sum_{j=1}^{k} \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right) = \log_2(k).$$

4. To constructing the weighted directed graph we use ρ_i and σ_i, and the weight of each directed edge (x_i, x_{i_j}) is computed as:

$$w((x_i, x_{i_j})) = \exp\left(-\frac{\max(0, d(x_i, x_{i_j}) - \rho_i)}{\sigma_i}\right).$$

5. To symmetrized the graph we convert the directed graph into an undirected graph, the adjacency matrix A is symmetrized using:

$$B = A + A^\top - A \circ A^\top,$$

where A is the adjacency matrix, $\circ$ denotes the Hadamard product, and $\top$ represents the transpose matrix. The resulting symmetric graph represents the unified global manifold structure of the dataset.

6. The symmetrized graph G forms a *fuzzy simplicial set* that captures both the local and global topological structure of the data. Also, this symmetrized graph G allows us to define a local similarity as $S_{\text{UMAP}}(x_i, x_j) = B_{ij}$. In the UMAP algorithm, the values in the distance graph represent the normalized weights of the connections between points in the k-nearest neighbor (k-NN) graph. These values range from 0 to 1, as UMAP normalizes these relationships to compute probabilistic affinities (similarities). A value of 0 indicates no connection between the points in the graph, while a value greater than 0 signifies a connection, with the value representing the strength of the connection. A value close to 1 indicates a strong connection.

To generalize Principal Component Analysis it will be important to understand how UMAP connects points from different neighborhoods because this will have implications on how to project individuals belonging to different Riemannian submanifolds. In UMAP, connections between points that belong to different neighborhoods (i.e., points that the k-nearest neighbors (k-NN) algorithm did not place in the same neighborhood) are handled during the *graph symmetrization step* explained above in 5. If two points x_i and x_j may belong to different neighborhoods, for example, x_i may not consider x_j one of its k-nearest neighbors, and vice versa, then it may be no direct edge between x_i and x_j in the initial directed graph. To address this, UMAP symmetrizes the directed graph by combining local neighborhood information from both points. The symmetrization step combines forward and backward edge weights, creating an undirected edge between points in different neighborhoods. The edge weights are symmetrized as:

$$w_{ij} = w_{ij}^{\rightarrow} + w_{ij}^{\leftarrow} - w_{ij}^{\rightarrow} \cdot w_{ij}^{\leftarrow}, \tag{1}$$

where $w_{ij}^{\rightarrow}$ and $w_{ij}^{\leftarrow}$ are the forward and backward affinities between x_i and x_j, computed from their respective local neighborhoods. This ensures points in separate neighborhoods can still be connected if their neighborhoods overlap or have similar affinities, capturing both local and global structures of the data. Also, this ensures that redundant weights from overlapping neighborhoods are not double-counted.

3 Riemannian Principal Component Analysis for any type of data

In the paper [2], they aim to generalize Principal Component Analysis (PCA) to Principal Geodesic Analysis (PGA) a generalization of principal component analysis to manifolds using geodesic distances and geodesic submanifolds.

Let be $x_1, \ldots, x_n \in \mathbb{R}^p$ with zero mean. Principal component analysis find an orthonormal basis $\{v_1, \ldots, v_p\}$ of $\mathbb{R}^p$, which satisfies the recursive relationship:

$$v_1 = \arg\max_{\|v\|=1} \sum_{i=1}^{n} (v \cdot x_i)^2, \tag{2}$$

$$v_s = \arg\max_{\|v\|=1} \sum_{i=1}^{n} \sum_{j=1}^{s-1} (v_j \cdot x_i)^2 + (v \cdot x_i)^2, \quad s = 2, \ldots, p. \tag{3}$$

Then, the subspace $V_s = \text{span}(\{v_1, \ldots, v_s\})$ is the s-dimensional subspace that maximizes the variance of the data projected onto that subspace. As is well known, the basis $\{v_s\}$ is computed as the set of ordered eigenvectors of the sample covariance matrix of the data.

The lower-dimensional subspaces in PCA are linear subspaces, in [2] for general manifolds H, they extend the concept of a linear subspace to a geodesic submanifold. A geodesic is a curve that is locally the shortest path between points. In this way, a geodesic is the generalization of a straight line. Thus, it is natural to use a geodesic curve as the one-dimensional subspace, the analog of the first principal direction in PCA.

Let M be a manifold, the projection of a point $x \in M$ onto a geodesic submanifold H of M is defined as the point on H that is nearest to x in geodesic distance. So, the projection operator $\pi_H : M \to H$ is:

$$\pi_H(x) = \arg\min_{y \in H} d(x, y)^2. \tag{4}$$

Since projection is defined by a minimization, there is no guarantee that the projection of a point exists or that it is unique. However, by restricting to a small enough neighborhood about the mean, the projection is unique for any geodesic submanifold at the mean and the projection onto a geodesic submanifold can be approximated linearly in the tangent space of M, that is, in 4 to ensure that $\pi_H(x)$ is actually within H, an approximation of the projection is used, then they find a sequence of nested geodesic submanifolds that maximize the projected variance of the data. Finally, given a dataset $x_1, x_2, \ldots, x_n \in M$, they compute the intrinsic mean μ, which minimizes the sum of squared Riemannian distances, $\mu = \arg\min_{x \in M} \sum_{i=1}^{n} d(x, x_i)^2$. Each data point is mapped to the tangent space using $u_i = x_i - \mu$. The covariance matrix, $S = \frac{1}{n} \sum_{i=1}^{n} u_i u_i^T$, is constructed, and eigendecomposition yields principal directions v_k and variances are λ_k, see [2] for the details.

To generalize these ideas to any data table, we define a Local Manifold Approximation, let $X = \{x_1, x_2, \ldots, x_n\}$ be a dataset embedded in a high-dimensional space

$\mathbb{R}^p$. For each data point x_i, UMAP defines a local neighborhood N_i consisting of its k-nearest neighbors under a metric d. The k-nearest neighbors of x_i are defined as the set $\mathcal{N}(x_i)$ such that:

$$\mathcal{N}(x_i) = \left\{ x_j \in X \setminus \{x_i\} \;\middle|\; d(x_i, x_j) \leq d(x_i, x_k),\ \forall x_k \notin \mathcal{N}(x_i) \right\},$$

where:

- $d(x_i, x_j)$ is the distance between x_i and x_j.
- k is the number of neighbors (a hyperparameter).

Thus, $\mathcal{N}(x_i)$ is the set of the k closest points to x_i in $\mathbb{R}^p$. The *Patch* $\mathcal{P}_i$ is defined as a *fuzzy simplicial set* that approximates the local geometry of the manifold around x_i. Formally:

$$\mathcal{P}_i = \left\{ (x_i, x_j, w_{ij}) \,\middle|\, x_j \in N_i,\ w_{ij} > 0 \right\},$$

where:

- x_i is the anchor point.
- $x_j \in \mathcal{N}_i$ are the k-nearest neighbors of x_i.
- w_{ij} is the weight (or affinity) associated with the edge between x_i and x_j, defined in the previews section.

The individual patches $\mathcal{P}_i$ for all $x_i \in X$ can be combined to form the global fuzzy simplicial set o graph G, representing the entire manifold:

$$G = \bigcup_{i=1}^{n} \mathcal{P}_i.$$

Because the data resides on a Riemannian manifold, with local distances it is necessity to define something akin to a Riemannian correlation, requiring a Riemannian mean, and, more broadly, necessitating the development of *Riemannian Statistics*. By leveraging the one-to-one relationship given by the Nerve Theorem in 1 and its corollaries, we define the vector subtraction the Riemannian correlation on manifold as follows.

Definition 5 Let x_α and x_β rows of X, we define the subtraction induced by the UMAP algorithm as $x_\alpha \ominus x_\beta = \rho_{\alpha\beta}(x_\alpha - x_\beta)$, where $\rho_{\alpha\beta} = 1 - S_{\text{UMAP}}(x_\alpha, x_\beta)$.
Then, local distances generated by the UMAP algorithm can be defined as:

$$d_{\text{UMAP}}(x_\alpha, x_\beta) = \|x_\alpha \ominus x_\beta\| = \sqrt{\langle x_\alpha \ominus x_\beta, x_\alpha \ominus x_\beta \rangle},$$

where $\langle \cdot, \cdot \rangle$ is an inner product on the vector space $\mathbb{R}^p$. We also define the the addition induced by the UMAP algorithm as $x_\alpha = x_\beta \oplus x_\gamma$ if $x_\gamma = x_\alpha \ominus x_\beta$.

In the subsequent definition, we generalize Fréchet mean:

Definition 6 Let $X \in M_{n\times p}$ the data table. We denote by $x_1, \ldots, x_n \in \mathbb{R}^p$ the rows of X and by $y_1, \ldots, y_p \in \mathbb{R}^n$ the columns of X. Each vector x_i can be also considered

a point in the Riemannian manifold M induced by the simplicial complex. The *Riemannian mean* is the minimizer of the sum-of-squared distances to the data:

$$g = \arg\min_{x \in M} \sum_{i=1}^{n} d_{\text{UMAP}}\,(x, x_i)^2\,.$$

Definition 7 We defined the variance-covariance matrix $S \in M_{p\times p}$ of X as $S = \frac{1}{n}\sum_{i=1}^{n}(x_i \ominus g)(x_i \ominus g)^t$, where $(x_i \ominus g) = \rho_{i\lambda}\begin{bmatrix} x_{i1} - g_1 \\ \vdots \\ x_{ip} - g_p \end{bmatrix}$, note that g must be equal to x_λ for some λ. Then, we can define *Riemannian correlation* between y_i and y_j columns of X, that are in $\mathbb{R}^n$, as follows $R(y_i, y_j) = \frac{S_{ij}}{\sqrt{S_{ii}S_{jj}}}$.

We can say that each patch $\mathcal{P}_i$ induces a *Riemannian Submanifold* H_i with the local metric d_{UMAP} generated by the UMAP algorithm, defined in previous paragraphs. We are now ready to define *Riemannian Principal Component Analysis* (R-PCA) for any data $x_1, \ldots, x_n \in \mathbb{R}^p$. Our goal, analogous to PCA, is to find a sequence of subspaces S_i de $\mathbb{R}^p$ that maximize the projected variance of the data, but, which also takes into account the local distances of each of the submanifods H_i that we have in the data. We find a sequence of subspaces, not a a sequence submanifolds, because our total space is finite, then requiring the projections to be in there would greatly degrade the result, something that even happens in the PGA proposed in [2].

Taking into account that $G = \bigcup_{i=1}^{n} \mathcal{P}_i$ and the following properties of patches: 1) $\mathcal{P}_i$ is a weighted graph over the local neighborhood $\mathcal{N}_i$, where the weights $w_{ij} \in [0, 1]$ define the strength of the connection between x_i and x_j, 2) $\mathcal{P}_i$ represents the local structure of the manifold around x_i, capturing relationships with its neighbors, 3) the weight w_{ij} can be interpreted as a probability or affinity that x_i and x_j are connected within the underlying manifold, the *Riemannian Principal Component Analysis* (R-PCA) that respects the local metrics of the submanifolds H_i, can be defined by the Algorithm 1.

De debe aclarar acá como se calcula D y g citar definición, y que las componentes y las correlaciones entre variables y componentes también son Riemannianas.

In practice, in algorithm 1 it is recommended to use at least $k = \left\lfloor \frac{n}{c} \right\rfloor$ (the whole part) where c is the number of clusters that the data table is suspected to have, or the number of clusters that you want to study, that is, k is at least the average number of individuals that each cluster has.

Algorithm 1 Riemannian Principal Component Analysis (R-PCA)

Input: $x_1, \ldots, x_n \in \mathbb{R}^p$ and the number k of nearest neighbors in UMAP algorithm.
Output: Principal directions $v_s \in S_i$, variances $\lambda_s \in \mathbb{R}$.
Generate symmetric UMAP graph G.
Compute the matrix of UMAP similarities $S_{\text{UMAP}}(x_i, x_j)$ with $i, j = 1, \ldots, n$. calculate_umap_graph_similarities_V1
Compute the matrix $P = \rho_{ij}$ con $i, j = 1, \ldots, n$, as in definition 5. calculate_rho_matrix ????
Compute the Riemannian distance matrix D. calculate_umap_distance_matrix
Compute the Riemannian mean g using D.
Compute the Riemannian variance-covariance matrix $S \in M_{p\times p}$ as $S = \frac{1}{n}\sum_{i=1}^{n}(x_i \ominus g)(x_i \ominus g)^t$. riemannian_covariance_matrix
Compute the Riemannian correlation matrix $R \in M_{p\times p}$ where $R_{ij} = \frac{S_{ij}}{\sqrt{S_{ii}S_{jj}}}$. correlation_matrix
Extract eigenvectors and eigenvalues of R: $\{v_s, \lambda_s\}$.
Compute the Riemannian Principal Components. riemannian_components_from_data_and_correlation
For the correlation circle, the Riemannian correlations between the original variables and the principal components are calculated. correlation_variables_components

4 Applications with simulated data and real data

4.1 Description of the simulated data in `Data10D.csv`

The `Data10D.csv` file contains data structured with 2900 rows and 10 variables, such that the first two columns determine five clusters within the dataset. These clusters are explicitly identified in the `cluster` column. Additionally, the file includes eight extra columns that were generated following a specific process to ensure they do not alter the cluster assignments.

The purpose of this process was to extend the original dataset by adding new variables that provide additional information without modifying the existing classifications. Eight additional columns, named `var1` to `var8`, were created with values generated independently using a standard normal distribution $N(0, 1)$. This means that each new variable has a mean of 0 and a standard deviation of 1. These variables were generated randomly and independently of the first two columns and the clusters defined in the `cluster` column.

The process of generating these new variables began by determining the total number of rows in the original dataset. For each row, eight random values were generated using NumPy's `np.random.normal(0,1,n)` function, where n is the number of rows in the dataset. Each generated value was stored in one of the new columns, ensuring that all values followed the same distribution and were uncorrelated with one another or with the original variables.

The final result is a file that preserves the structure and cluster assignments of the original data while being enriched with eight new independent variables. The preservation of the `cluster` column ensures that the groupings defined by the first two columns remain intact, while the new variables provide additional information that can enhance analysis.

The data in the file `Data10D.csv` corresponds to the 2D plot in Figure 4, where the first two columns (`x` and `y`) define the positions of the points and generate

the clustering structure observed in the graph. These columns are responsible for the distinct shapes of the clusters, such as Cluster 1 and Cluster 2 are concentric, where Cluster 2 is nested within Cluster 1, forming a circle-within-a-circle structure. Cluster 3 and Cluster 4 exhibit a similar nested structure but are located to the right of the first group, with Cluster 3 inside Cluster 4. Cluster 5 is located at the bottom of the graph, forming a distinct parabolic shape that is separated from the others. The `cluster` column in the file assigns each point to one of the five clusters, which is reflected in the graph through color coding: each cluster is represented by a unique color (e.g., blue for Cluster 1, orange for Cluster 2). While the additional eight variables (`var1` to `var8`) in the file were generated independently following a normal distribution $N(0,1)$, they are not used in the 2D plot and do not influence the visualization. The plot accurately represents the clustering structure defined by the first two columns and the `cluster` column in the file, validating the data generation process and demonstrating the clear separability and arrangement of the clusters in the dataset.

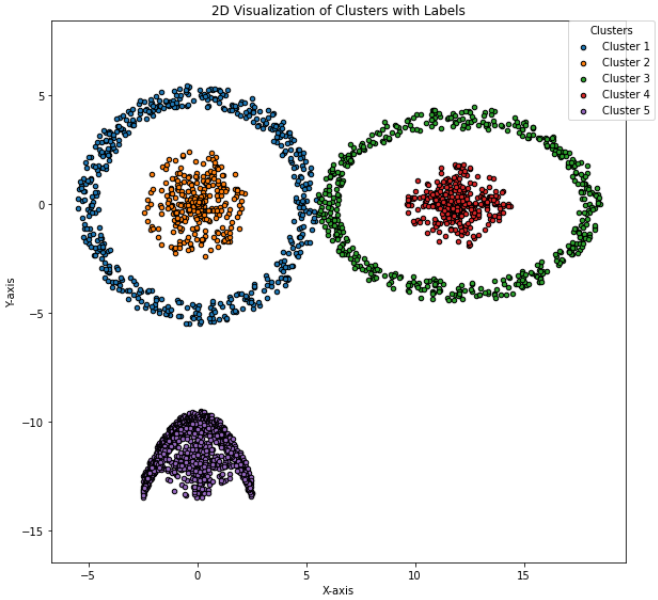

Fig. 4 Plotting the data using only the first two variables.

The images 5, 7, 6 and 8 represent visualizations of the Principal Component Analysis (PCA) and the Riemannian Principal Component Analysis (R-PCA) applied to the dataset `Data10D.csv`. The plots 5 and 7 correspond to the principal plane and the correlation circle of the PCA, respectively. Similarly, figures 6 and 8 represent the principal plane and the correlation circle of the R-PCA. Here we are computing R-PCA with $k = \left\lfloor \frac{2900}{5} \right\rfloor = 580$.

The principal plane of the R-PCA in Figure 6 preserves the structure of the original five clusters in `Data10D.csv` more effectively than the standard PCA. The nested circular clusters (Clusters 1, 2, 3, and 4) and the parabolic cluster (Cluster 5) are clearly separated and retain their geometric shapes. In contrast, the principal plane of the PCA in Figure 5 shows more diffuse clusters, with less distinct shapes. Especially Cluster 5 (orange) which is a circle inside Cluster 1 (blue), is better captured in the R-PCA than the PCA showing the orange points mostly enclosed in the blue points. Similarly it happens between clusters 3 and 4 in green and red respectively.

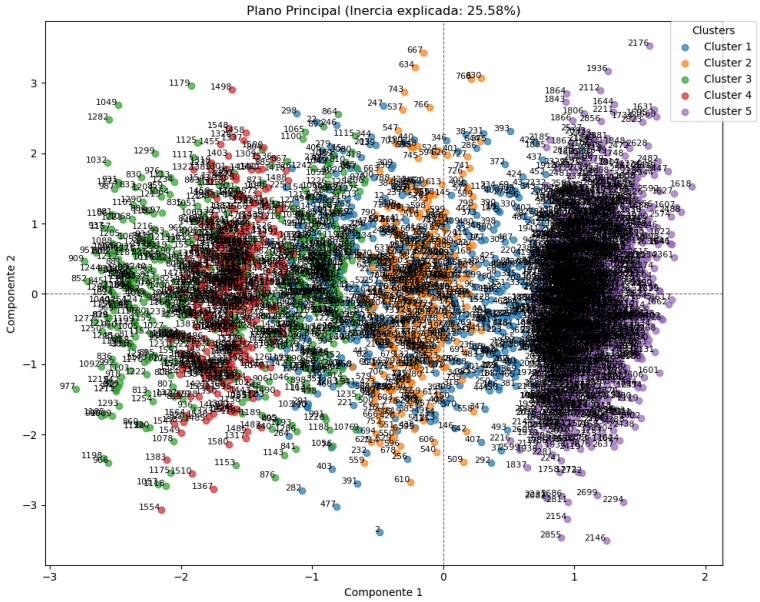

Fig. 5 PCA Plane of `Data10D.csv`.

The correlation circle of the R-PCA in Figure 8 provides a better interpretation of the relationships among the original variables, particularly highlighting the correlations for `x` and `y`. These variables are more prominently and accurately represented in the R-PCA correlation circle. In contrast, the PCA correlation circle in Figure 7 captures some variable relationships but is less effective at emphasizing these critical correlations.

The R-PCA achieves an explained variance (inertia) of **43.38%**, which is significantly higher than the **25.58%** explained by the PCA. This indicates that the R-PCA captures a larger proportion of the variability in the dataset, making it a more effective tool for dimensionality reduction in this context.

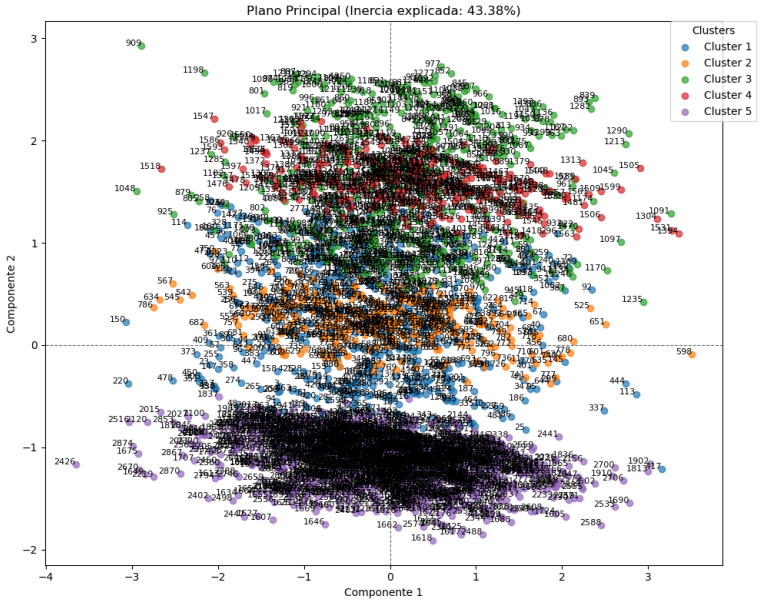

Fig. 6 R-PCA Plane of `Data10D.csv`.

5 Conclusions and Future Work

In conclusion, the R-PCA outperforms the standard PCA in preserving the geometric structure of the clusters, such as the nested circles and the parabolic cluster, in the principal plane. It also provides a more meaningful interpretation of variable correlations in the correlation circle. The higher explained variance of the R-PCA further demonstrates its superiority in capturing the underlying structure of the data.

In this paper, we successfully extend the ideas proposed by *Pennec et al.* in [7], broadening the scope to compute Riemannian statistical indices and Riemannian data analysis models to any data table. Unlike previous approaches, our methodology is not restricted to data with an intrinsic Riemannian manifold structure. This advancement opens up a new field of research, where diverse methods like regression, k-means, and more, can be generalized for broader applicability.

Currently, we are actively engaged in implementing these novel ideas in both **R** and **Python**, ensuring practical adoption and seamless integration across different computational platforms.

References

1. Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine, 29(6), 141–142.

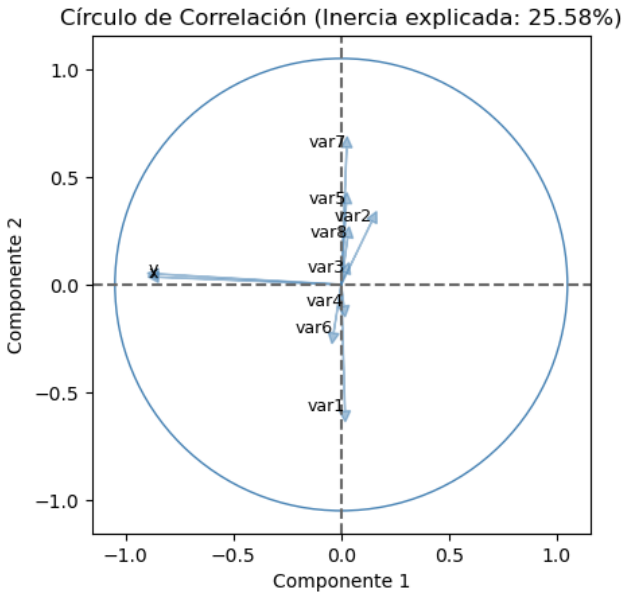

Fig. 7 PCA Correlations Circle of `Data10D.csv`.

2. Fletcher, P. T., Lu, C., Pizer, S. M., & Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. IEEE Transactions on Medical Imaging, 23(8), 995–1005. https://doi.org/10.1109/TMI.2004.831793
3. Karthik Shekhar, Sylvain W. Lapan, Irene E. Whitney, Nicholas M. Tran, Evan Z. Macosko, Monika Kowalczyk, Xian Adiconis, Joshua Z. Levin, James Nemesh, Melissa Goldman, Steven A. McCarroll, Constance L. Cepko, Aviv Regev, Joshua R. Sanes. (2016). Comprehensive Classification of Retinal Bipolar Neurons by Single-Cell Transcriptomics. In *Comprehensive Classification of Retinal Bipolar Neurons by Single-Cell Transcriptomics*, Volume 166, Issue 5, Pages 1308-1323.e30, ISSN 0092-8674. https://doi.org/10.1016/j.cell.2016.07.054.
4. Maaten, L. van der, and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9 (Nov), 2579-2605. Submitted 5/08; Revised 9/08; Published 11/08.
5. McInnes, L., Healy, J., and Melville, J. (2018). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *cite arxiv: 1802.03426 Comment: Reference implementation available at http://github.com/lmcinnes/umap/ 6*, 352–357. http://arxiv.org/abs/1802.03426
6. Oudot, S. Y. (2016). Persistence Theory: From Quiver Representations to Data Analysis (Mathematical Surveys and Monographs, Vol. 209). American Mathematical Society.
7. Pennec X. Sommer S. and Fletcher T. (Eds). (2020). *Riemannian Geometric Statistics in Medical Image Analysis.* Academic Press, Elsevier.
8. Rabadán, R., and Blumberg, A. J. (2020). Topological Data Analysis for Genomics and Evolution: Topology in Biology. Columbia University, New York, y University of Texas, Austin. Cambridge University Press.

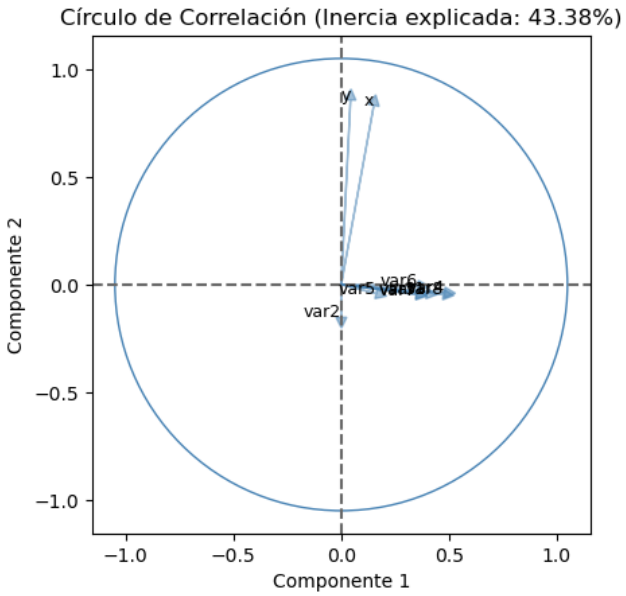

Fig. 8 R-PCA Correlations Circle of `Data10D.csv`.