Uniform Manifold Approximation and Projection

RD - S2

Pierre LAGUE & Ilian VANDENBERGHE - Université de Lille 1

Abstract

UMAP, or Uniform Manifold Approximation and Projection, was introduced in a research paper titled "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction" by Leland McInnes and John Healy. The paper was published in the journal "arXiv" on February 2, 2018.

Section 1: Introduction

Uniform Manifold Approximation and Projection (UMAP) [3] is a dimension reduction technique that can be used for visualization similarily to t-distributed Stochastic Neighbor Embedding (t-SNE), but also for general non-linear dimension reduction. Before diving into the experiments, it is important to contextualize the underlying concepts UMAP is based on. UMAP is constructed from a theoretical framework based on Riemannian geometry and algebraic topology.

Riemannian geometry [2] is a branch of differential geometry that deals with curved spaces as seen in Figure 1. In UMAP, the high-dimensional data is considered to lie on a manifold, which can be seen as a curved space embedded within the high-dimensional Euclidean space. UMAP leverages Riemannian geometry to model the local and global structure of the data manifold, which is crucial for preserving the geometric relationships between data points in the low-dimensional embedding.

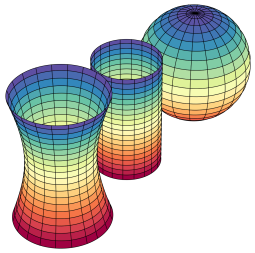

Figure 1: Curvature of Riemannian Manifolds - Wikipedia

The riemannian geometrical objects are then studied and analyzed using **algebraic topology** [1]. UMAP utilizes these concepts to understand the connectivity and structure of the data manifold. In particular, UMAP constructs a weighted graph representation of the data manifold, where nodes represent data points and edges represent the connections between neighboring points. By understanding the topological properties of the data manifold, UMAP can create a low-dimensional embedding that preserves the essential topological features of the original data.

Finally, **manifold learning** is a subfield of machine learning that focuses on understanding and representing high-dimensional data lying on low-dimensional manifolds.

Section 2: UMAP Algorithm

The UMAP algorithm is founded on three assumptions about the data :

- The data is uniformly distributed on the riemannian manifold
- The riemannian metric is locally constant (or can be approximated as such) [1]

[1] A Riemannian metric is a mathematical structure that defines how distances and angles are measured on a smooth manifold.

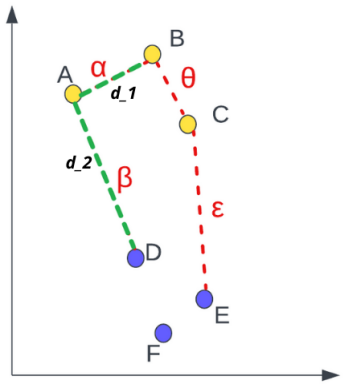

Figure 2: Here are displayed 6 points {A,B,C,D,E,F} forming 2 distinct clusters. Some distances have been displayed between the points. We will refer to this as the high-dimension representation of the points.

- The manifold is locally connected [2]

We will now explain the procedural steps involved in achieving a lower-dimensional representation of high-dimensional data. In this study, we focus on 2D data and illustrate a 1D dimension reduction. The initial step in the UMAP algorithm is the computation of similarity scores to discern clustered data points, to preserve the inherent clustering structure within the resultant low-dimensional graph. UMAP utilizes a specified number of neighbors to identify these clusters, often opting for 15 or more on extensive datasets. In this illustrative example, we limit the neighborhood size to 3 for the sake of simplicity, facilitating the visualization of distinct clusters (Figure 2).

This distance metric is referred to as the "High-Dimensional Similarity Score". UMAP computes this score for each pair of high-dimensional points, capturing their proximity in the original feature space. Subsequently, UMAP initiates the first dimension reduction step, wherein each point is positioned along an axis as reference (in our example, the reference is point A in Figure 3, there are as many similarity curves as there are points) based on its distance from the studied point (point A). Upon arranging the points along the axis, a "similarity curve" is traced. This curve is calculated using the following formula:

$$f : \exp^{-(d\_1 - d\_2)/\sigma}$$

Here, d_1 represents the distance between the studied point and the closest neighbor point (w.r.t. the initial cluster), d_2 denotes the distance to the nearest non-neighbor point (both can be seen on Figure 2). σ serves as a scaling factor that adjusts the characteristics of the "similarity curve" (depicted as the green curve in Figure 3, denoted as $f(x)$, x being the point on which the score is computed). The parameter σ, in the similarity function f, is fine-tuned such that, for all points in our dataset $S = A, B, C, D, E, F$, with k representing the number of neighbors (here, 3) :

$$\sum_{i=1}^{card(S)} f(S_i) = \log_2(k)$$

In other words, the sum of the values obtained by applying the function f (representing the "similarity curve") to each point equals the logarithm, base 2, of the number of neighbors. As depicted in Figure 3, the computed values of the function showcase that $f(B) + f(C) + f(D) + f(E) + f(F) = 1.6$, which corresponds to $\log_2(3)$. This step underscores the significance of two key hyperparameters: the number of neighbors (inclusive of the point itself) and the parameter σ, which adjusts the curve to align with the data distribution. Notably, the similarity curve and scores vary from one point to another. However, UMAP scales these curves such that regardless of the proximity or distance between neighboring points, the sum of the similarity scores remains consistent at $\log_2(k)$, where k represents the specified number of neighbors.

It is essential to highlight that due to the variability of similarity scores across different points, it becomes imperative to symmetrize these results. This symmetrization is crucial for maintaining consistency with the underlying theory of topology and fuzzy sets [3], which form the basis of UMAP. By ensuring symmetry in the

[2] "locally connected" refers to the property that every point on the manifold has a neighborhood that is connected. In other words, if you pick any point on the manifold and zoom in close enough, the portion of the manifold around that point will be connected; there won't be any gaps or separations.

[3] In a fuzzy set, instead of a binary membership function, elements are assigned membership degrees on a continuous scale between 0 and 1, indicating the degree of belongingness to the set.

similarity scores, UMAP adheres more closely to the principles of topology and fuzzy sets, thus facilitating a more robust and interpretable dimensionality reduction process (score $A-B$ has to be same as score $B-A$). This process differs from the t-SNE technique where t-SNE averages the two scores. For two different high-dimension scores, $s1$ and $s2$, we can make them symmetrical by computing the following result :

$$f_u(s_1, s_2) = (s1 + s2) - s1 \times s2$$

This new score is then assigned to both points in order to reach the perfect fit on the curve.

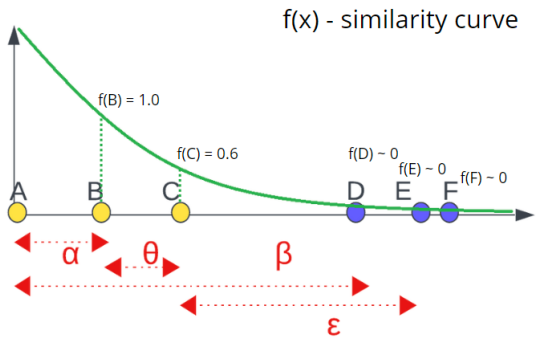

Figure 3: Similarity Curve

Following the initialization of a low-dimensional graph utilizing the similarity scores derived from the curve illustrated in Figure 3, UMAP employs a technique known as Spectral Embedding [4]. However, this graph does not initially represent the high-dimension clusters as seen on Figure 4. To rectify this, UMAP chooses two low-dimension points that should be put closer together. This is achieved by randomly selecting a pair of points in a cluster proportionally to their high-dimensional score (here point A and B have a high-dimensional score in the first cluster in Fig 2). Initially, the two high-dimension clusters are not represented in the low dimension clusters. Point B needs to be closer to point A. That is done by iteratively moving B closer to A to maximize its low-dimensional score using the t-distribution on the right. And moving B away from E, because they don't belong to the same high-dimension neighborhood, by minimizing its low-dimensional score using the t-distribution on the left.

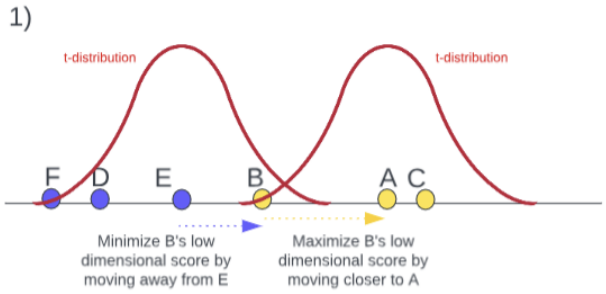

Figure 4: First step, representing the points in low dimension and choosing the points to compute the low dimension score $Z(d)$.

The low-dimensional score Z is given by the following computation :

$$Z(p) = \frac{1}{1 + \alpha \times d^{2\times\beta}}$$

With p being the distance between the moving point (in our example B) and the reference point (in our example A). Parameters α and β control how tightly packed the points are at the end of the iterations. This way, UMAP gives more control than t-SNE over how packed low-dimension points are at the end.

The second step represented on Figure 5 illustrates the intermediate step where B is closer to A and the two clusters start to appear in low-dimension representation.

[4]Spectral embedding is a dimensionality reduction technique that leverages the spectral properties of a similarity or affinity matrix derived from the data

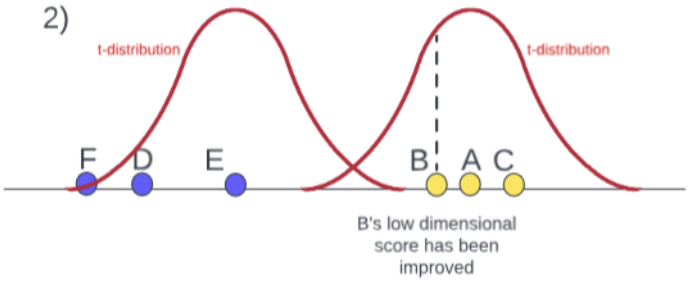

Figure 5: Second step, iteratively moving B maximizing it's low dimensional score on the A t-distribution and minimizing it's low dimensional score on the E t-distribution.

In order to know where B should go, and when to stop, UMAP introduces a cost function optimized with Stochastic Gradient Descent[5]. It is known as cross-entropy and it is defined as follows:

$$\text{Loss} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(P_{ij}\log\left(\frac{P_{ij}}{Q_{ij}}\right) + (1-P_{ij})\log\left(\frac{1-P_{ij}}{1-Q_{ij}}\right)\right)$$

- N is the total number of data points.
- P_{AB} is the value of the membership strength between data points A and B in the original high-dimensional space.
- Q_{AB} is the value of the membership strength between data points A and B in the low-dimensional space.

Each value P_{AB} and Q_{AB} are computed with Z, previously introduced. This loss is used on points E (resp. A) and B as well in order to move B "away" from E (resp. "closer" to A). Cross-entropy loss is used in UMAP because it effectively captures the difference between pairwise similarities of data points in the original high-dimensional space and their representations in the low-dimensional space. By minimizing the cross-entropy loss through stochastic gradient descent, UMAP can learn a low-dimensional embedding that preserves the local structure of the data while reducing dimensionality. The SGD iterative process uses this loss function and places B such that the loss is minimized.

Finally, the third step in Figure 6 illustrates the final low-dimension representation after each point has been moved to maximize it's low dimension score. **After a certain amount of iterations, we have a low-dimension representation of the high-dimension clusters.**

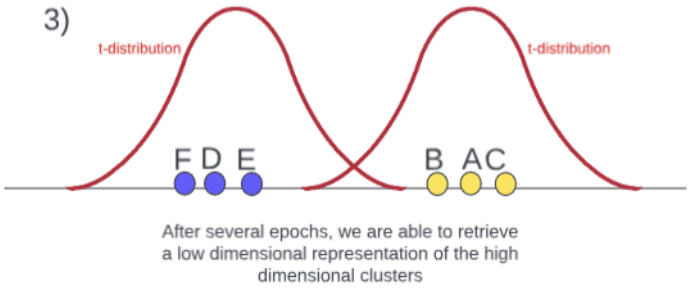

Figure 6: Final state of the low-dimension representation.

A simplified version of the UMAP pseudo-algorithm is described in pseudo-algorithm 1

[5]Stochastic Gradient Descent, is a fundamental optimization algorithm commonly used in machine learning for minimizing a loss function.

Algorithm 1 UMAP Algorithm

Input: High-dimensional data $\mathbf{X}$, number of dimensions $n_{\text{components}}$, number of nearest neighbors k
Compute high-dimensional scores.
Optimize σ to have a perfect fit on the similarity curve
Initialize low-dimensional embedding Y randomly
for $t = 1$ **to** maximum iterations **do**
  Compute fuzzy set Φ based on Y
  Update learning rate α_t
  for all data point pairs $[a, b]$ **do**
    Compute gradient $\nabla(\log(\Phi))$ and $\nabla(\log(1 - \Phi))$
    Update y_a using stochastic gradient descent: $y_a \leftarrow y_a + \alpha_t \cdot \nabla(\log(\Phi))(y_a, y_b)$
  end for
end for
return Low-dimensional embedding Y

Section 3: UMAP Practical Experiments

In this study, we aimed to evaluate the efficacy of UMAP (Uniform Manifold Approximation and Projection) dimensionality reduction technique on benchmarked datasets. The experimental setup followed a rigorous protocol to assess the quality of dimensionality reduction based on the score of classification algorithms. We selected benchmarked datasets from diverse domains (tabular data and images) to ensure a comprehensive evaluation. These datasets encompassed a range of characteristics, including varying dimensions, sample sizes, and class distributions, thus providing a robust testbed for our analysis.

Throughout our experiments we will try to observe the effect of various hyper-parameters as depicted in 2 :

- k : the number of neighbors initially chosen, `n_neighbors`=15 by default in python.
- α and β : used to control how tightly packed the points will be in low-dimension α=`a`=0 and β=`b`=0 by default in python.

(1) UMAP vs Structured Data The Palmer Penguin dataset is a comprehensive collection of morphological measurements and demographic data pertaining to three species of penguins inhabiting the Palmer Archipelago, Antarctica. The dataset comprises measurements from 344 penguins, encompassing three distinct species: Adelie (Pygoscelis adeliae), Chinstrap (Pygoscelis antarctica), and Gentoo (Pygoscelis papua). These measurements include various morphological attributes such as bill length, bill depth, flipper length, and body mass, along with categorical variables like sex and species.

The goal here is to try and reduce the dimension from 4 (number of penguins' physical attributes) to 2 in order to have a single plot keeping the high-dimensional informations. By applying UMAP on this dataset, we are able to reach such result, as depicted on Figure 7. We can see that the packing parameters not only clarify the low dimension information, they also improve it. However, the third plot on Figure 7 shows that the blue and orange clusters are mixed together, which shows such hyper-parameter should be dealt with carefully. Here, the number of neighbors did not significantly change the result.

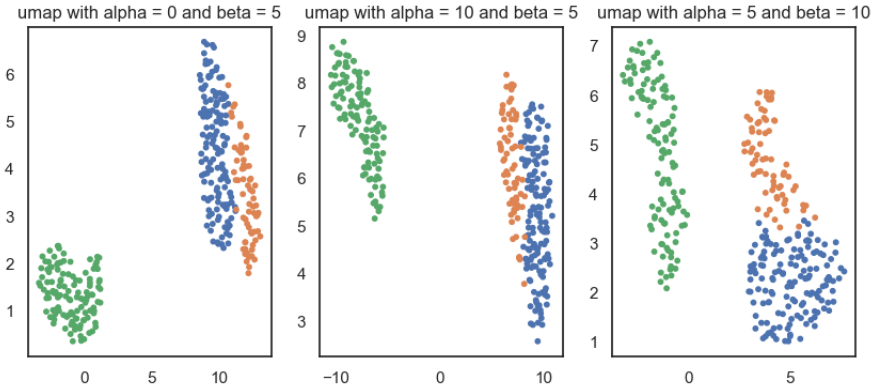

Figure 7: Low Dimension projection of the penguin dataset with different values for packing parameters a and b. $k = 15$ in all experiments.

The plots indicate that optimal parameters, to maximise comprehension and scoring would be $\alpha = 10$ and $\beta = 5$ with $k = 15$. This way, we have a representation that has linear properties, with clusters that do not overlap. This experiment shows that UMAP can achieve significant performance on keeping the high-dimensional information throughout it's dimension reduction process on structured data. Using the score of supervised algorithm classification methods gives us an insurance in the quality of the dimension reduction. We aim to find a similar score before and after the reduction. We can use the reduced-dimensional representation obtained by UMAP as input features and train a classifier (such as kNN or SVC) to predict the labels. We can then evaluate the classifier's performance using metrics like accuracy. This evaluation provides insight into how well the reduced-dimensional representation preserves the discriminative information present in the original high-dimensional data. Both SVC (C-Support Vector Classification) and KNN (K-nearest neighbors) scores displayed in Table 1 show that the high-dimensional information are well represented in low dimension.

Algorithm score/Projection	SVC	KNN
High-Dimension	0.71	0.76
Low-Dimension	0.70	0.75

Table 1: Evaluation of the low-dimensional projections on the Penguins Dataset compared to high-dimensional score.

(2) UMAP vs Unstructured Data The MNIST Digit dataset, consisting of handwritten digit images, and the Olivetti Faces dataset, comprising grayscale facial images, represent two diverse yet prominent datasets frequently employed in machine learning and computer vision research on unstructured data. Leveraging UMAP, we aimed to uncover meaningful low-dimensional representations of these datasets while preserving their inherent structures. Both datasets being unstructured data, their initial high dimensional representations are extremely complex. In fact their initial shapes are (1797, 64) for the `MNIST Digits` dataset and (400, 4096) for the `Olivetti Faces` dataset. Reducing the dimension to 2D and applying an inverse transform to assess the quality of the information that was reduced is a way of evaluating UMAP efficiency.

When applying UMAP on the MNIST dataset for a low-dimensional representation in 2D we obtain the following result :

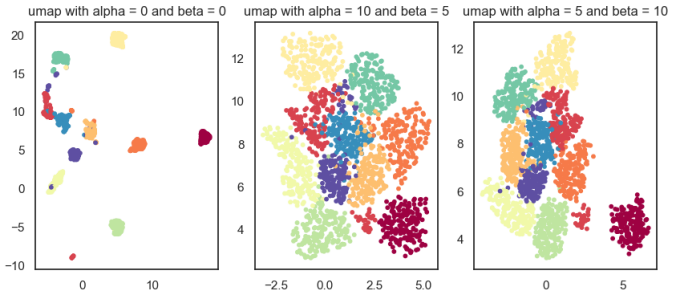

Figure 8: Low-Dimension 2D projection of the MNIST Digit dataset with different values for packing parameters a and b and for the number of neighbors k

The results showcased in Figure 8 illustrate one of cases where the dimension reduction is too low. In fact, each of the plots showcase clusters that are overlapping. In that case, we can't understand how the data is clustered. UMAP allows us to choose the dimension in which we project our low-dimensional data (parameter `n_components` in python). We will keep only the first plot in Figure 8, since it has the most clear representation of the clusters. It seems the parameters by default in the UMAP instantiation give optimal packing representation. By adapting the `n_components` parameter from 2 to 3 for a 3D representation, we can obtain a clearer representation with only 1 additional dimension as shown in Figure 9, where we can clearly see the different cluster each corresponding to a digit. UMAP allows us to grasp the structure of high-dimensional data in low dimension, this step can greatly improve data complexity before feeding it to a machine learning model (supervised or unsupervised learning). We've managed to reduce the number of attributes to 2 instead of 64, with a slight decrease in clustering score, as shown in Table 2, this highlights the UMAP effectiveness to create a low-dimensional embedding that preserves the essential topological features of the original data.

When applying UMAP on the Olivetti Faces dataset for a low-dimensional representation in 2D we obtain the same nature of results as for the MNIST dataset (Figure 8, fuzzy information). In 2D, the dimension is too low to fully grasp the complex nature of the images via a plot. However, we can apply an inverse transformation and see what result is returned in order to assess the quality of the dimension reduction. The inverse transform operation consists in producing a high dimension representation from a low dimension embedding. This can be

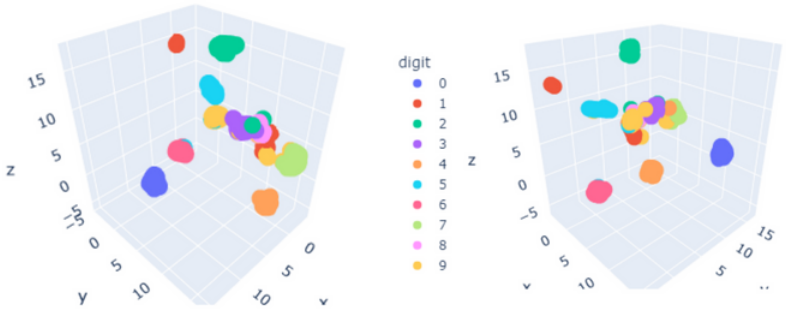

Figure 9: Low-Dimension 3D projection of the MNIST Digit dataset with values a=0, b=0 and k=15.

Algorithm score/Projection	SVC	KNN
High-Dimension	0.95	0.98
Low-Dimension	0.94	0.97

Table 2: Evaluation of the low-dimensional projections on the MNIST Dataset compared to high-dimensional score.

done by fitting the data with UMAP using only `fit` and not `fit_transform` to retain the trained model for later generating new digits based on samples from the embedding space. Each point on the generated grid point is a two dimensional point lying somewhere in the embedding space. The `inverse_transform` method will convert this into an approximation of the high dimensional representation that would have been embedded into such a location. If the inverse transformation from the low-dimensional representation returns clear faces (or handwritten digits) as shown in Figure 10, then we can be sure that the dimension reduction process successfully describe the high-dimensional information.

Figure 10: Inverse transform operation on the Olivetti Faces Dataset and MNIST Digit Dataset. From the low-dimensional representation we can still see that the image information is roughly preserved.

For each of these datasets, we've been able to reduce the dimension from above 60 to 3 or even 2 dimension representations that still hold a majority of the information contained in the high-dimensional projection. We've seen that the number of neighbors and packing parameters also influence the quality of the projection as well as the time needed to compute the low-dimensional representation.

Section 4: Efficiency comparison with other techniques

In the context of this study, we decided to compare other common dimension reduction techniques to UMAP. This comparison will be done on the same dataset (Fashion-MNIST, complex data). The algorithms are the following : PCA (Principal Component Analysis) is a linear dimensionality reduction technique that aims to capture the maximum variance in the data by projecting it onto orthogonal axes, reducing the dimensionality while preserving the global structure. MDS (Multidimensional Scaling) constructs a lower-dimensional representation of data by preserving pairwise distances or dissimilarities between data points, aiming to maintain the global structure of the data. t-SNE (t-distributed Stochastic Neighbor Embedding) focuses on preserving local similarities between data points in the high-dimensional space by embedding them into a lower-dimensional space, often used for visualizing clusters and preserving local structures.

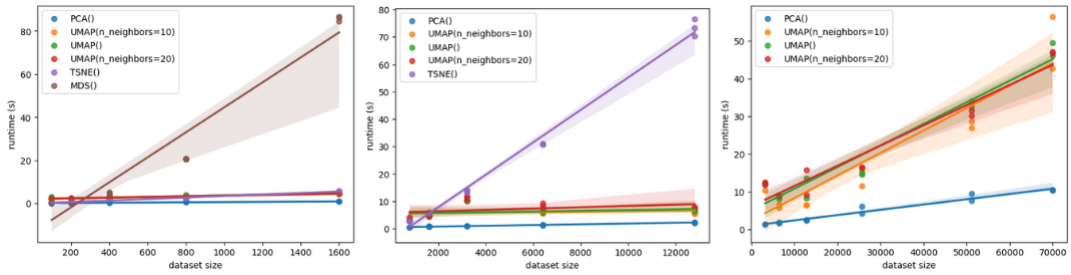

Figure 11: Comparison of three common dimension reduction techniques : PCA, T-SNE, UMAP and MDS. Plots showcase the difference of efficiency between UMAP and T-SNE, but show a similarity between UMAP and PCA.

The results depicted in Figure 11 exhibit a comparative analysis of dimensionality reduction techniques across increasing dataset sizes. Initially, PCA, T-SNE, and UMAP demonstrate comparable efficiency relative to MDS as the dataset sizes escalate. However, as the dataset size further increases, T-SNE exhibits diminished efficiency, thereby positioning UMAP and PCA in the subsequent plot. Within this context, PCA demonstrates a slight superiority over UMAP, particularly noticeable with a substantial dataset size. For large datasets, MDS and T-SNE are, in conclusion, not the most efficient, where as UMAP and PCA show some convincing results in terms of dimension reduction on complex data.

Section 5: Conclusion

In conclusion, this study contributes to a deeper understanding of UMAP's role in dimensionality reduction, thereby enhancing our analytical capabilities for interpreting high-dimensional datasets effectively.

References

[1] Allen Hatcher. *Algebraic topology.* , 2005.

[2] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

[3] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.