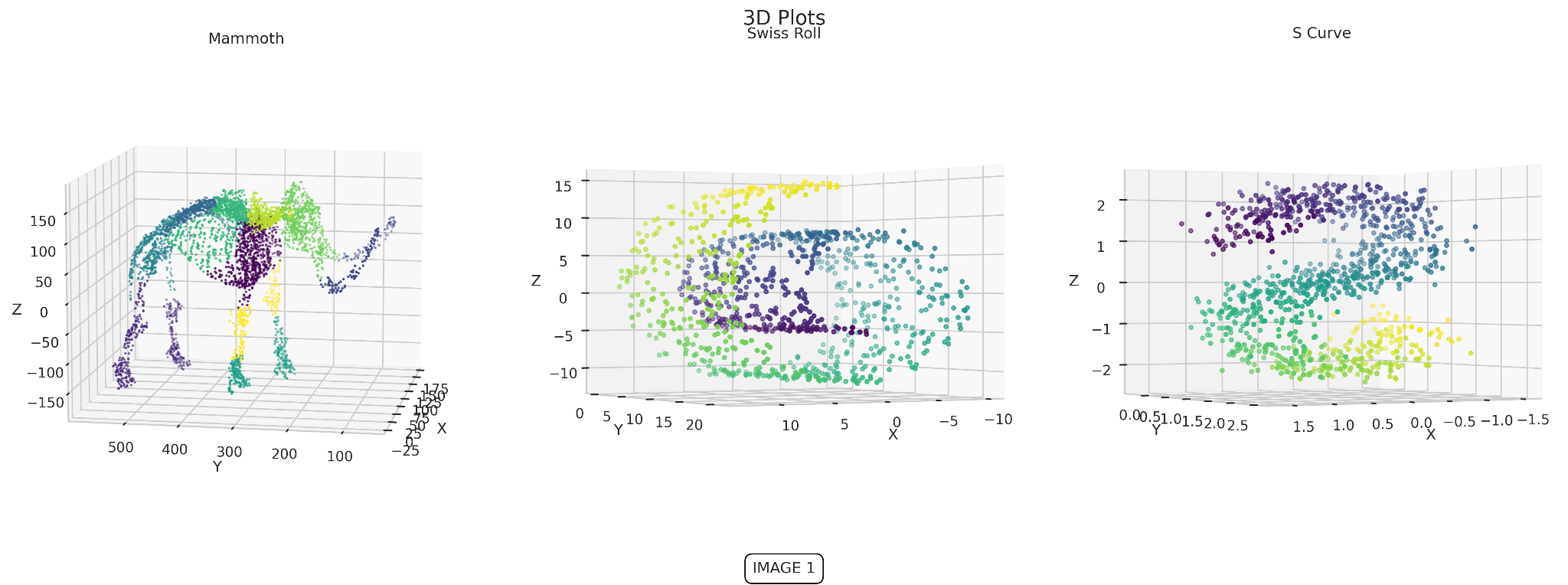
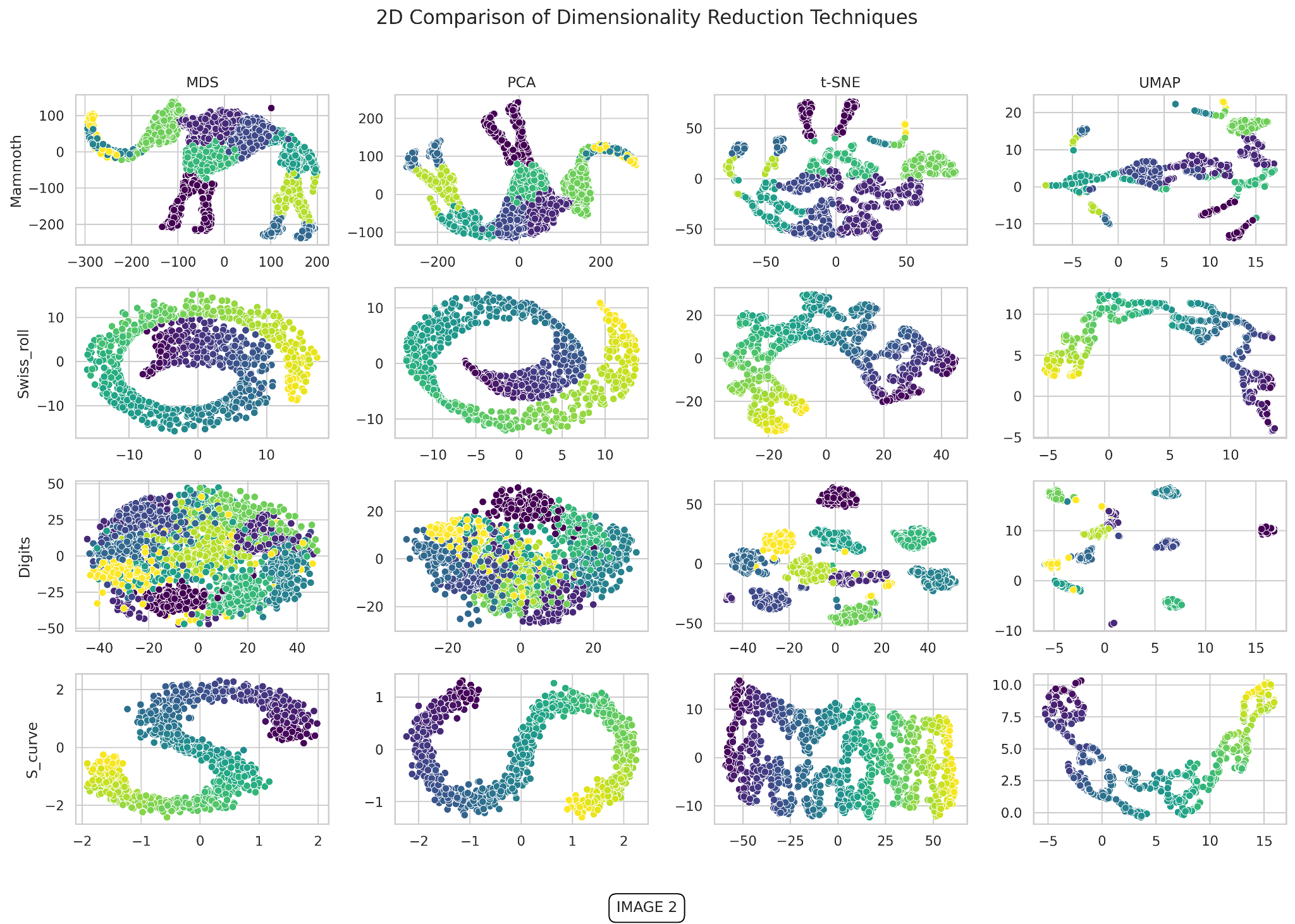
****

****

**Understanding Dimension Reduction: Navigating High-Dimensional Spaces**

In the world of data, the concept of dimension reduction is essential for making sense of complex, high-dimensional datasets. At its core, dimension reduction is about transforming data from a high-dimensional space into a lower-dimensional space while retaining the most meaningful properties of the original data.

This technique is crucial for various applications, including filtering, compression, regression, classification, feature analysis, and visualization.

The four techniques that have been reviewed: **PCA, MDS, T-SNE, UMAP**

**Table of content:**

1. Toy Example
2. Forces in High-Dimensional Spaces
3. Dimension Reduction Formalism
4. Dimension Reduction Methods: PCA, MDS, T-SNE, UMAP

**Toy Example**

First, let’s look at toy examples to gain insight into the differences between the methods.

Considering these datasets:

* **Mammoth:** A 3D scanned model of a woolly mammoth skeleton, included in the Smithsonian Institution's collection of digital 3D models. The dataset comprises 1 million data points, of which 5,000 have been randomly selected for analysis.
* **Swiss Roll:** A 3D dataset in which points are organized in a spiral, helical shape resembling a rolled-up surface. It is created using parametric equations that define a spiral: 𝑥 = 𝑡 ⋅ cos ⁡ ( 𝑡 ) , 𝑦 = ℎ , 𝑧 = 𝑡. Where 𝑡 represents the distance along the spiral and ℎ is a vertical component. Additional Gaussian noise is added to introduce variability. A total of 1000 data points were sampled.
* **Digits:** A dataset containing 1,797 samples of handwritten digits (0-9), represented as 8x8 pixel grayscale images. Each image is converted into a 64-dimensional vector, corresponding to the pixel intensities. This dataset is part of the UCI Machine Learning Repository.
* **S Curve:** A 2-dimensional (2D) surface embedded in a 3-dimensional (3D) space. The points in the dataset are arranged in an "S" shape, with the curve bending back and forth along two of the three axes, resulting in a non-linear form. This dataset is created by mapping a 2D grid onto a 3D space using the following mathematical function: 𝑥 = t, y = sin( 𝑡 ), 𝑧 = ℎ. A total of 1000 data points were sampled.

We can visualize the raw **Mammoth**, **Swiss Roll** and **S Curve** datasets, because their data are in 3D - **[IMAGE 1]**

The projection into 2D dimensions - **[IMAGE 2]**

Insights:

* **Mammoth**, **Swiss Roll** and **S Curve**: **PCA** and **MDS** preserve the global structure of the data. However, **T-SNE** and **UMAP** might introduce artifacts, resulting in misleading clusters or artificial islands.
* **Digits**: **PCA** and **MDS** produce cluttered visualizations. This is because they focus on global structure preservation, which may not sufficiently capture the local variations and distinct features of each digit. Consequently, the resulting plots can be less informative, with overlapping digit representations.
* **PCA** & **MDS** - Best suited for preserving the global structure of the data, providing a broad overview. They may not capture local details as effectively, which can result in less detailed visualizations, especially for datasets with complex local structures.
* **T-SNE and UMAP**: Excellent at preserving local structures and revealing small clusters or patterns within the data. They might introduce artifacts, and their performance is sensitive to hyperparameter settings. Proper tuning of these parameters is crucial to achieving accurate and interpretable results.

**Forces in High-Dimensional Spaces**

When working with high-dimensional data, we encounter two opposing forces:

1️⃣ **The Curse of Dimensionality:** As the number of dimensions in a dataset grows, the data becomes sparse, and the distance between points becomes nearly uniform. To illustrate, imagine a D-dimensional hypercube containing a D-dimensional hypersphere. As the dimensions increase, the volume of the hypersphere shrinks toward zero while the hypercube's volume remains constant. This leads to data points becoming more uniformly distributed, diminishing the differences between distant points. Furthermore, to maintain the same density of data points within a fixed volume, the number of data points would need to increase exponentially.

2️⃣ **The Blessing of Non-Uniformity:** Fortunately, real-world data often has a latent structure, existing on a lower-dimensional manifold. This means the data can be compressed into fewer dimensions without losing significant information. By leveraging this underlying structure, dimensionality reduction techniques can counteract the curse of dimensionality, preserving meaningful insights.

**Dimension Reduction Formalism:**

Optimising while trying to preserve:

Let’s dive into the four techniques: **PCA, MDS, T-SNE, UMAP**

**PCA** (**P**rincipal **C**omponent **A**nalysis), 1901

The general idea

* PCA is a statistical procedure that transforms a set of correlated variables into a set of uncorrelated variables called principal components. This is achieved through an orthogonal transformation, projecting the data onto straight lines (principal components) in Euclidean space.
* The technique assumes that the most important information is captured by the variance in the data; hence, features with higher variance carry more information.
* PCA is particularly effective in preserving global information.
* Time complexity:
  1. Eigenvalue decomposition: O(n³), for n × n matrix.
  2. Singular Value Decomposition (SVD): O(mn²), for m × n matrix.

PCA Algorithm

1. **Data Preprocessing:**
   1. Center the data by subtracting the mean.
   2. Scale the data by dividing by the standard deviation.
2. **Finding Principal Components:**

There are two main approaches:

* 1. **Closed-form solution:**
     1. Compute the covariance matrix.
     2. Perform Eigenvalue decomposition or SVD on the covariance matrix.
  2. **Iterative process:**
     1. **Power Iteration Method:**
        1. Start with an initial random vector.
        2. Multiply the covariance matrix by the vector, normalize the result.
        3. Repeat until convergence to find the eigenvector corresponding to the largest eigenvalue.
        4. For additional components, subtract the contribution of the found principal components and repeat.
     2. **Expectation-Maximization (EM) Approach:**
        1. Expectation Step: Estimate the projection of the data onto the current principal components. Optimization can be done by either minimizing the distance between data points and the principal component line or maximizing the distance from the origin.
        2. Maximization Step**:** Recalculate the principal components based on the updated projections.
        3. Repeat until convergence.

Additional notes

* The iterative process is especially useful for large datasets where the closed-form solution is computationally expensive.
* SVD is often preferred over Eigenvalue decomposition because it avoids computing the covariance matrix directly, which can be time-consuming. However, for kernel PCA, where the data is not linearly separable, Eigenvalue decomposition is necessary.
* Loading scores - These indicate the contribution of each variable to a principal component.
* Eigenvalues - Measure the amount of variance captured by each principal component, which can be visualized using a Scree plot.
* Singular Values (from the SVD) **-** These are the square roots of the eigenvalues.
* The number of principal components is limited by either the number of variables or the number of samples, whichever is smaller.

**MDS** (**M**ulti**D**imensional **S**caling), 1960

The general idea

* Attempts to maintain the **overall** geometric relationships between points in a lower-dimensional, which is considered a global structure.
* We have a distance matrix, now we are trying to find that leads to that distance matrix (we can first calculate the distance matrix from the data).
  + If the original data comes from Euclidean space, then we can find exact solution that (Where , and ).
  + If there is another distance metric, then there is approximation: .
  + The solution is not unique.
* Time complexity:
  + Classical MDS - , for the n × n matrix.
  + Metric/Non-Metric MDS:-

Algorithm  
 There are 3 types of MDS:

* Classic MDS (PCoA - **P**rincipal **Co**ordinate **A**nalysis):
  + Similar to PCA but focuses on distances rather than correlations.
  + Involves constructing a distance matrix (Gram matrix), performing eigenvector decomposition, and projecting to lower dimensions.
* Metric MDS:
  + Assumes distances satisfy the triangle inequality.
  + Defines a Stress (**St**andardized **re**siduals **s**um of **s**quares) function to measure the error between original and lower-dimensional distances.
  + No convex solution, unlike in the classical one. Iterative techniques can be applied like “gradient descent”. Another possibility is to use the SMACOF algorithm (Basically a surrogateconvex function that iteratively goes through minimization).
* Non-Metric MDS:
  + Does not assume the triangle inequality.
  + Deals with ordinal data by finding a monotonic function (using isotonic regression) and optimizing the stress function, typically using the SMACOF algorithm.

**SNE/T-SNE** (**T**-Distributed **S**tochastic **N**eighbour **E**mbedding), 2008

The general idea

* Two points in the Euclidean space may be considered close. For example “Swiss roll” shape with two points from different layers. Those points may not be appropriately considered close from a geodesic distance representation and close neighbours may be after description of the shape.
* The algorithm initializes low dimensional points and then moves the low dimensional points around until they form clusters that have the same relationship as in the high-dimensional data.
* Try to preserve local structure.
* Time complexity , but the Barnes-Hut approximation reduces it to - ).

Algorithm

* Hyperparameter is specified, which approximately represents the number of neighbours - **Perplexity**. Defined as: , where P - is conditional probability distribution (which is the Gaussian described below), and H - is the Shannon entropy (Higher value - Higher variance, less certainty).
  1. Low Perplexity - Thinner gaussian, lesser points considered as neighbours.
  2. High Perplexity - Wider gaussian, higher points considered as neighbours.
* **The original paper (SNE)**
  1. Calculating the high dimensional data distribution:
     1. Find the Perplexity value (the same Perplexity to all points) - Binary search algorithm is applied to search from a list of variances (calculated from the dataset) value that has the closest perplexity value to that specified hyperparameter (the variance is part of the Perplexity formula).
     2. An individual normal Gaussian curve is created above each point (adjusting the variance), so each point will have approximately the same number of nearest neighbours (same Perplexity).
  2. low-dimension representation is initialized (randomly or with PCA). The distribution above each point in high and low dimensions needs to be similar - The low dimension distribution is adjusted. A comparison between the two distributions is done with Kullback–Leibler (KL) divergence.
  3. An iterative process of gradient descent on the KL divergence is used, which can be formalized as a balance of attractive and repulsive forces.
* **Improvement (T-SNE)**
  1. Symmetric Probabilities - Since the score from A->B and B->A is different (different curves), their values are averaged, speeding up computation.
  2. Early exaggeration - For the first couple iterations the attractive forces are exaggerated in order to escape local minima.
  3. Student's t-Distribution - Instead of gaussian, a Student's t-distribution is used only in the lower dimension. This has the following benefit as it prevents the “crowding problem” - This issue occurs when too many points in the high-dimensional space are mapped too closely together in the low-dimensional space. A broader neighbourhood in the low dimensional space which allow for a greater error.

Additional information

* Shepard diagram - Is a good diagnostic tool which is used to assess how well the lower-dimensional representation of data preserves the pairwise distances between points as compared to their distances in the original high-dimensional space. It is basically a scatter plot of the distances between points in lower dimensional space against the distances between points in higher dimensional space. Suitable for: T-SNE, UMAP, MDS.
* Its flexibility makes it tricky to interpret.

**UMAP** (**U**niform **M**anifold **A**pproximation and **P**rojection), 2018

The general idea

* Initializes low-dimensional points and iteratively adjusts them to form clusters that reflect the relationships present in the high-dimensional data.
* Aims to preserve both global and local structures, though the degree to which this is achieved may depend on the algorithm's initialization and parameter settings.
* Time Complexity:

Algorithm

1. **Hyperparameter (#NB):** Defines the number of neighbours for each point.
2. We specify the hyperparameter for the number of neighbours for each point (#NB).

High #NB = captures the global structure. Low #NB = focuses on local structure.

1. Calculating *Similarity scores*:
   1. For each point, calculate the log(#NB).
   2. Construct a decaying exponential curve for each point (), so the sum of the values (Y-Axis) for all other points needs to be equal to log(#NB).
   3. All those values are registered as the strength of the edges between the points.  
      Since the score from A->B and B->A is different (different curves), UMAP makes a single symmetrical score by *fuzzy union operation*: SAB​=(SA​+SB​)−SA​×SB​.
2. Those *Similarity scores* translated in the lower dimension to a relative distance between points.
3. UMAP initializes a low-dimensional graph using spectral embedding, which provides a consistent starting point for the iterative process.
4. The iterative process:
   1. Two points are selected (E.g. **A, B**) relative to their high dimensional score in order to move them together (one of those points is selected as the ference, for example **A**). Another point is selected in order to increase the distance from it, for example **C**.
   2. Two distributions based on the t-distribution are centered at **A** and **C**, in order to decrease the distinction between **A** and **B** and Increase the distance between **C** and **A**.
   3. The distribution is: ( , yields similarity scores similar to those in T-NSE), where:
      1. - hyperparameters.
      2. d – Is the distance between points.
      3. If . yields similarity scores similar to those in T-NSE. So UMAP gives more control over the spread of points in the low-dimensional space.
   4. The cost function helps determine how much to move **B** by its derivatives.
   5. This process iterates over a small subset of points, and due to the stochastic nature of gradient descent, each final graph may differ slightly.

Differences to T-SNE:

In terms of the main ideas of how UMAP and T-SNE work they are essentially the same and most of the differences are very subtle, however there are some important differences:

* T-SNE always starts with a **random** initialization of the low-dimensional graph, in contrast UMAP uses spectral embedding to initialize the low-dimensional graph, resulting in a consistent starting graph.
* T-SNE moves every single point a little bit each iteration. In contrast, UMAP can move just one point or a small subset of points each time (scale well with large data sets).
* Similarity scores are determined by different distributions.
* The different similarity scores, A->B and B->A are combined into one value: UMAP - fuzzy union operation. T-NSE - Average them.

| **Method** | **Purpose** | **Preserves** | **Complexity** | **Strengths** | **Weaknesses** |
| --- | --- | --- | --- | --- | --- |
| **PCA** (Principal Component Analysis) | Dimensionality Reduction | Global structure | O(n³) for Eigenvalue Decomposition, O(mn²) for SVD | Simple, fast, good for linear relationships | Can miss complex structures, may not handle non-linearity well |
| **MDS** (Multidimensional Scaling) | Dimensionality Reduction | Global and local structure | O(n²) for classical MDS | Can capture both global and local relationships | Computationally expensive for large datasets, may struggle with non-linearity |
| **T-SNE** (t-Distributed Stochastic Neighbor Embedding) | Non-linear Dimensionality Reduction | Local structure | O(n²) for large datasets | Excellent at preserving local structure, revealing clusters | Computationally intensive, may produce misleading artifacts or non-global structure |
| | **UMAP** (Uniform Manifold Approximation and Projection) | | --- |  |  | | --- | |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Resources:

* <https://en.wikipedia.org/wiki/Dimensionality_reduction>
* https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/
* <https://www.nature.com/articles/s41587-020-00809-z>
* <https://drops.dagstuhl.de/storage/01oasics/oasics-vol027-vluds2012-irtg1131/OASIcs.VLUDS.2011.135/OASIcs.VLUDS.2011.135.pdf>
* <https://www.youtube.com/watch?v=eN0wFzBA4Sc>
* https://www.youtube.com/watch?v=jc1\_yPYmspk&list=PLV8yxwGOxvvoJ87mFL27k7XSDq\_lF3pD5&index=1
* <https://www.youtube.com/watch?v=NEaUSP4YerM&t=196s>
* https://en.wikipedia.org/wiki/Principal\_component\_analysis
* <https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm>
* <https://en.wikipedia.org/wiki/Power_iteration>
* <https://www.youtube.com/watch?v=GEn-_dAyYME>
* <https://distill.pub/2016/misread-tsne>
* <https://www.youtube.com/watch?v=YevCnRd61f8>