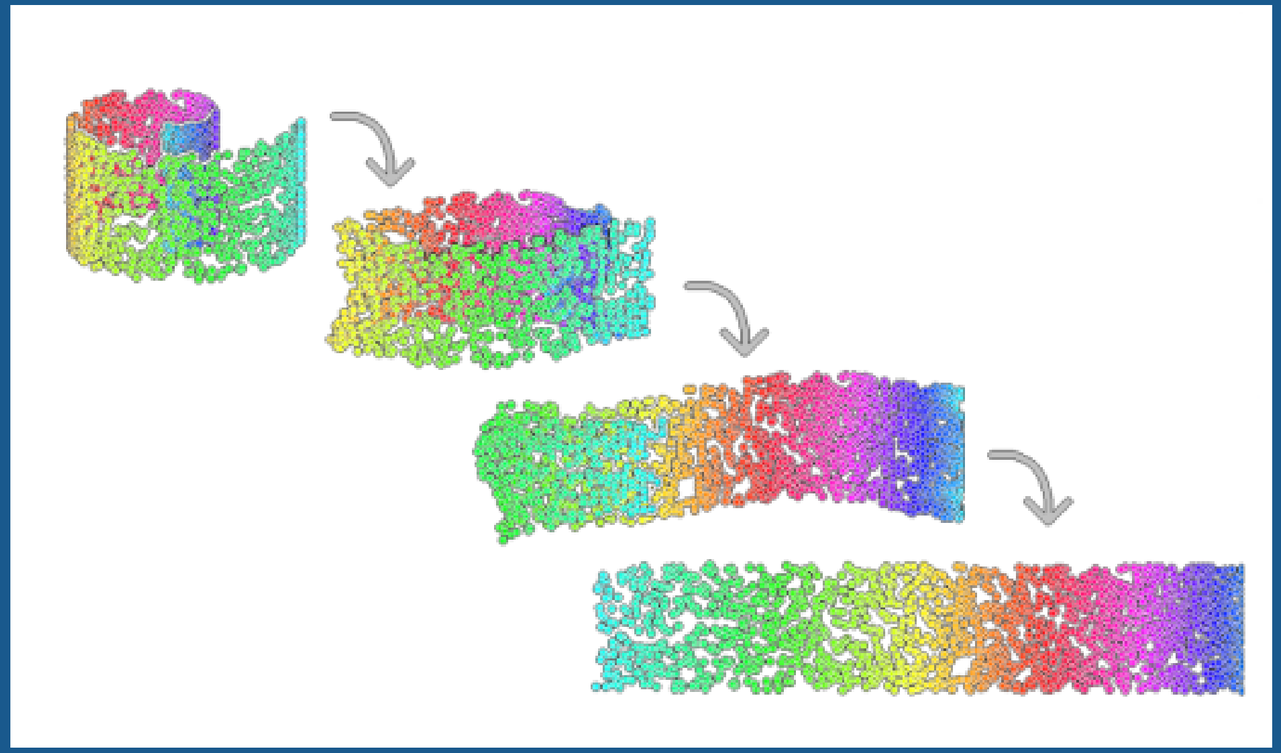
Manifold Sculpting

An iterative Non-Linear Dimensionality reduction method



mat. SM3800008

Presented By : Valentinis Alessio

Problem Statement

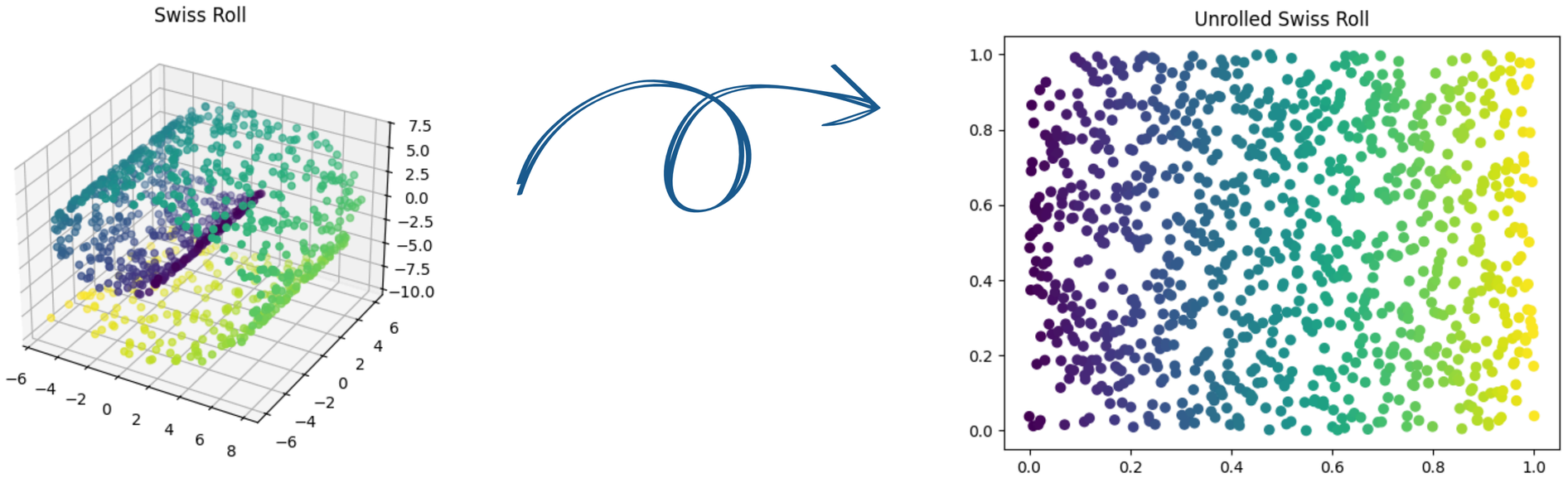
Nowadays, we can have access to lots of information, which is mainly gathered in high-dimensional datasets.

One of the main tasks of Unsupervised Learning is to try and reduce the dimensionality of such data, while preserving almost the same characteristics of the original data, i.e. trying to minimize the information loss.

There is a huge variety of algorithms developed with this scope, but each of them comes with its own strengths and limitations.

In the proposed paper, a new algorithm is proposed, called Manifold Sculpting, to face the problem of non-linear dimensionality reduction (i.e. suited for non-linear manifolds).

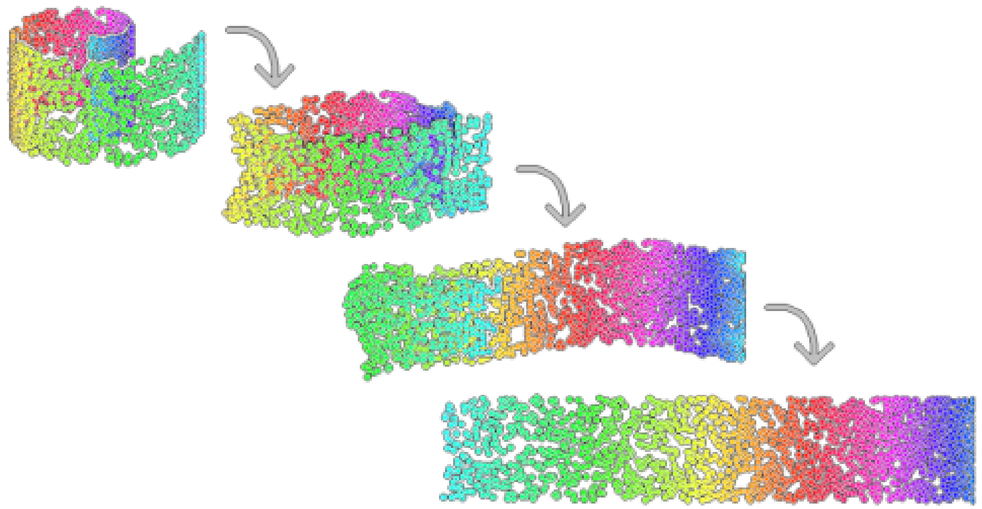
Problem Statement

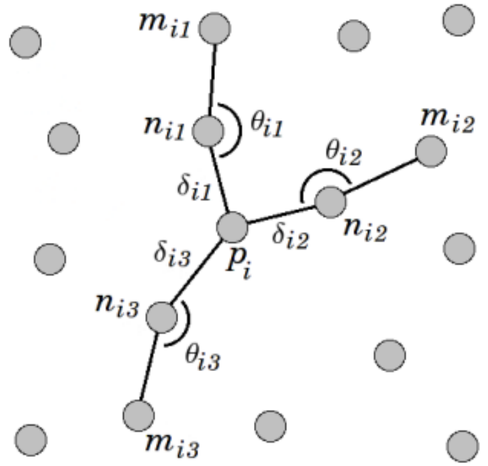


The algorithm

This method aims to slowly “unfold” the complex manifold in which the data lie, just like you would unfold a crumpled piece of paper.

In order to achieve this, we should provide the data a graph-like structure composed of neighbors, which the algorithm will aim to preserve locally.





The algorithm (in detail)

- **Step 1: Find the *k* nearest neighbors for each point.** We will create a matrix of neighbors N such that n_{ij} will represent the j-th neighbor of the point p_i
- **Step 2: Compute relationships between neighbors.** Now we will calculate, for each of the neighbors n_{ij} of each point, the euclidean distance δ_{ij} and the angle θ_{ij}, which is the angle formed between p_i, n_{ij} and m_{ij}, which is the most collinear neighbor of n_{ij} with p_i. Also the average distance δ_{avg} will be computed. These are the properties that the algorithm will try to preserve.
- **Step 3: Preprocess data (optional).** Here, optionally you can preprocess data with the transformation step of PCA (i.e. align the manifold to its principal components). This step has been shows to speed-up convergence. This simply aligns the manifold with the first $|D_{pres}|$ principal components, in order to let the algorithm handle only the non-linear part.

The algorithm (in detail)

- **Step 4: Transform the data.** Data now is iteratively transformed untill some stopping-criterion has been met. One effective technique is to stop when the sum of changes of all points has fallen below a predefined threshold.
  - **4a: Scale the data.** In this step, all the values in D_{scal} are scaled by a constant factor $0 < \sigma < 1$. At the end of the algorithm, these dimension will have very little information left, and will be discarded.
  - **4b: Restore relationships.** This step will aim to restore the original relationships that are distorted during scaling in the remaining dimensions. Intuitively, this step simulates tension in the surface of the manifold. An heuristic error value is used to evaluate the current relationships w.r.t. the original ones.

$$\epsilon_{p_i} = \sum_{j=0}^{k} w_{ij} \left(\left(\frac{\delta_{ij} - \delta ij_0}{\delta_{avg}} \right)^2 + \left(\frac{\theta_{ij} - \theta ij_0}{\pi} \right)^2 \right)$$

The algorithm (in detail)

Warning: In order to optimize that value, we don't compute the gradient, as it would require O(D^3) computations, and the main objective of this algorithm is to be "light-weight".
With this purpose, we change the coordinates to be preserved following a local hill-climbing technique, so moving by a learning rate (**lr**) step in each dimension to preserve sequentially to minimize the error. In which order do we perform this optimization? Starting from a random point in the manifold, we follow breadth-first the graph of neighbors.

- **Step 5: Project the data.** Now we simply drop the D_{scal} dimensions, that are now close to zero.

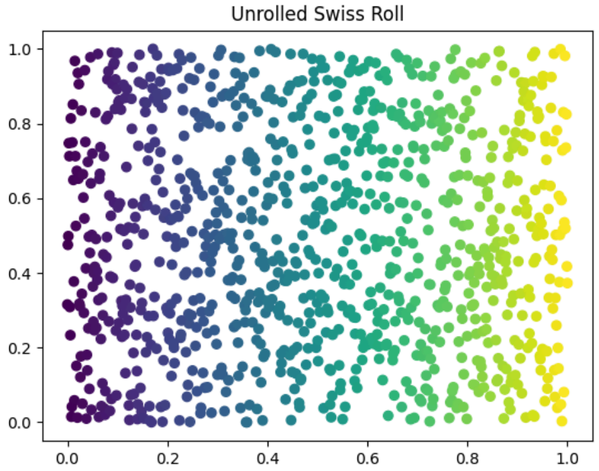
The algorithm: parameters

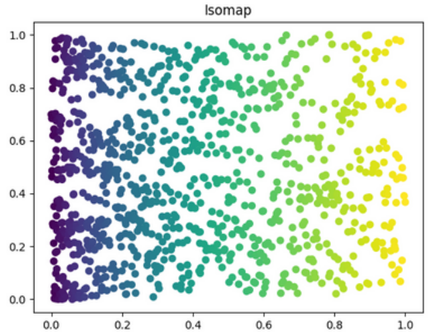
This algorithm relies on many parameters:

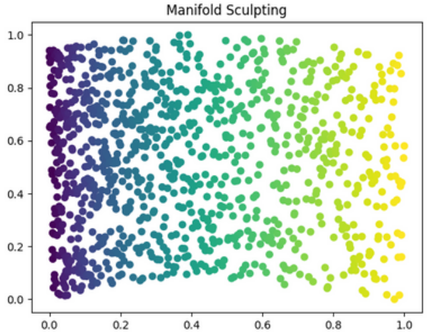
- **k**: number of neighbors of each point
- **sigma** : scaling parameter
- **lr**: learning rate
- **n_iters**: number of iterations to be performed
- **err:** error threshold to consider in order to stop
- **patience** (optional)**:** number of iterations without improvement

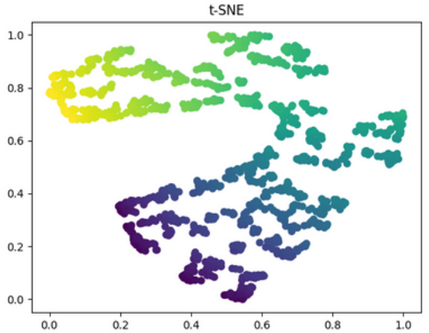
Comparison (qualitative)

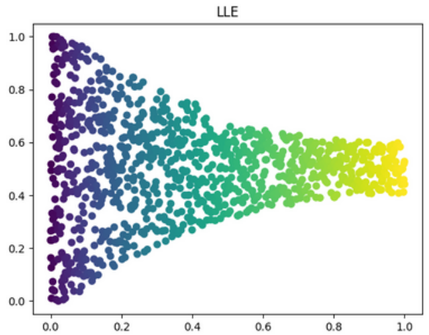
In order to conduct sensible comparison, I roto-traslated each unfolded manifold to the principal components and min-maxed the results.

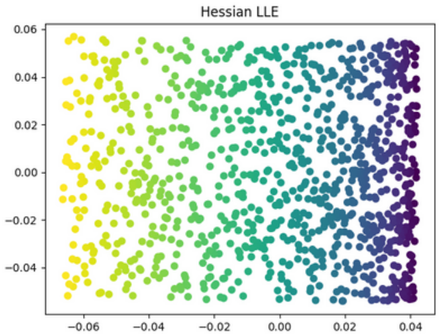


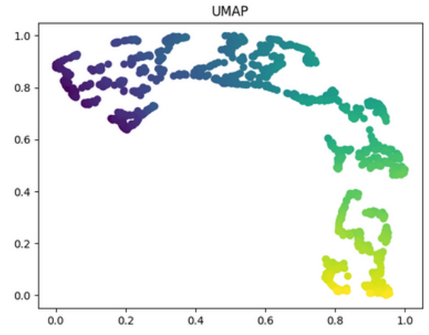






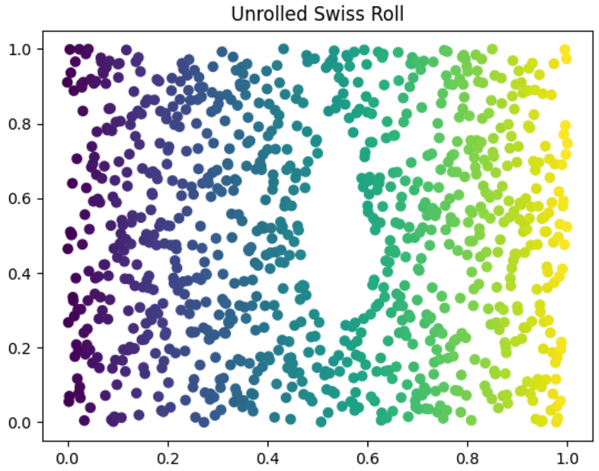


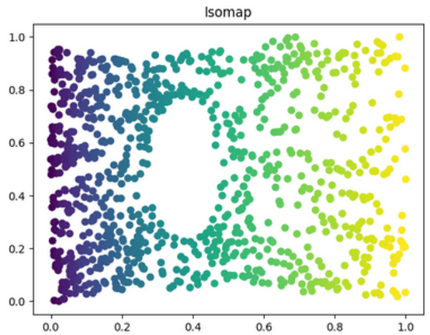


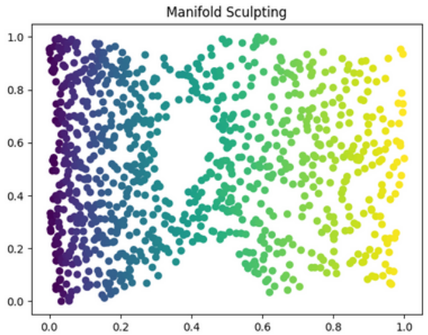


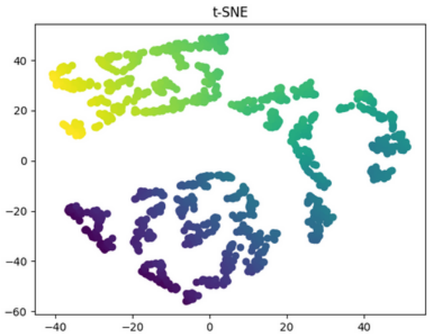
Comparison (qualitative)

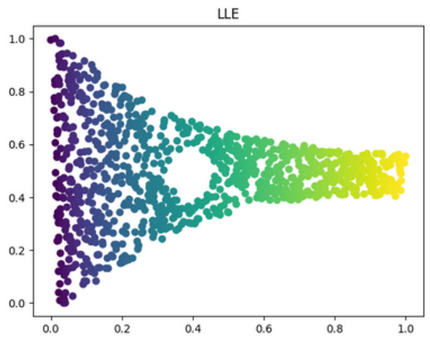
Now a qualitative comparison on a “holed” manifold

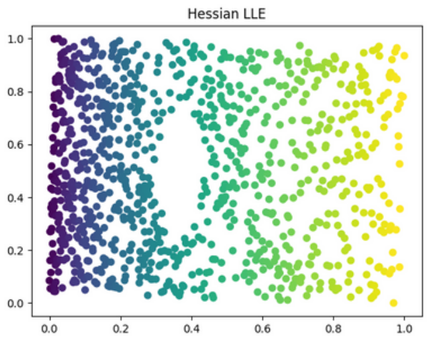


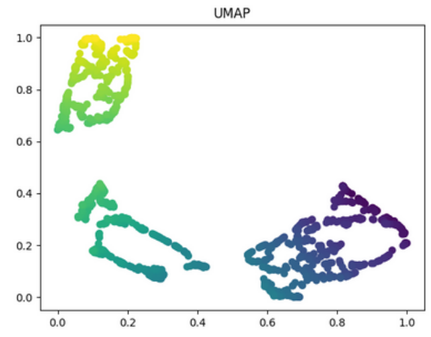






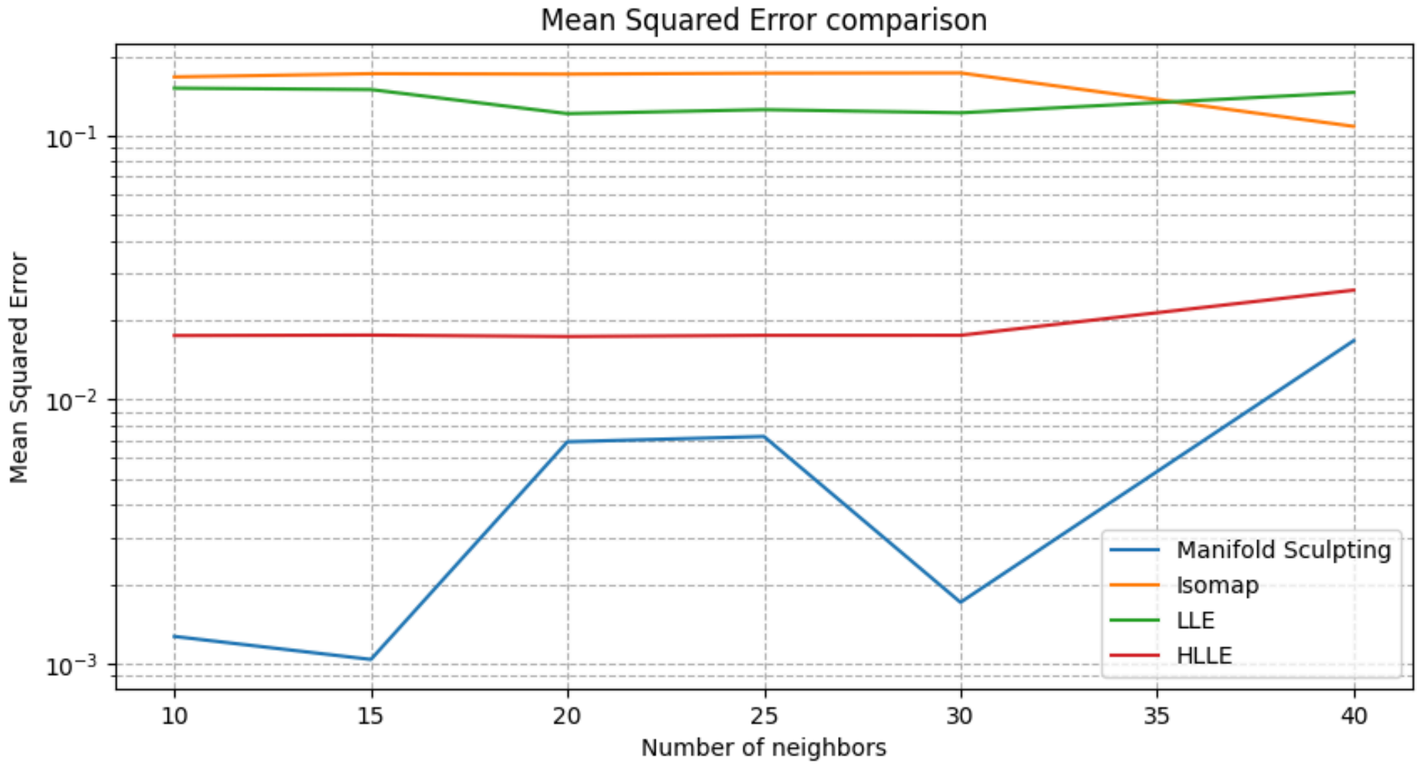






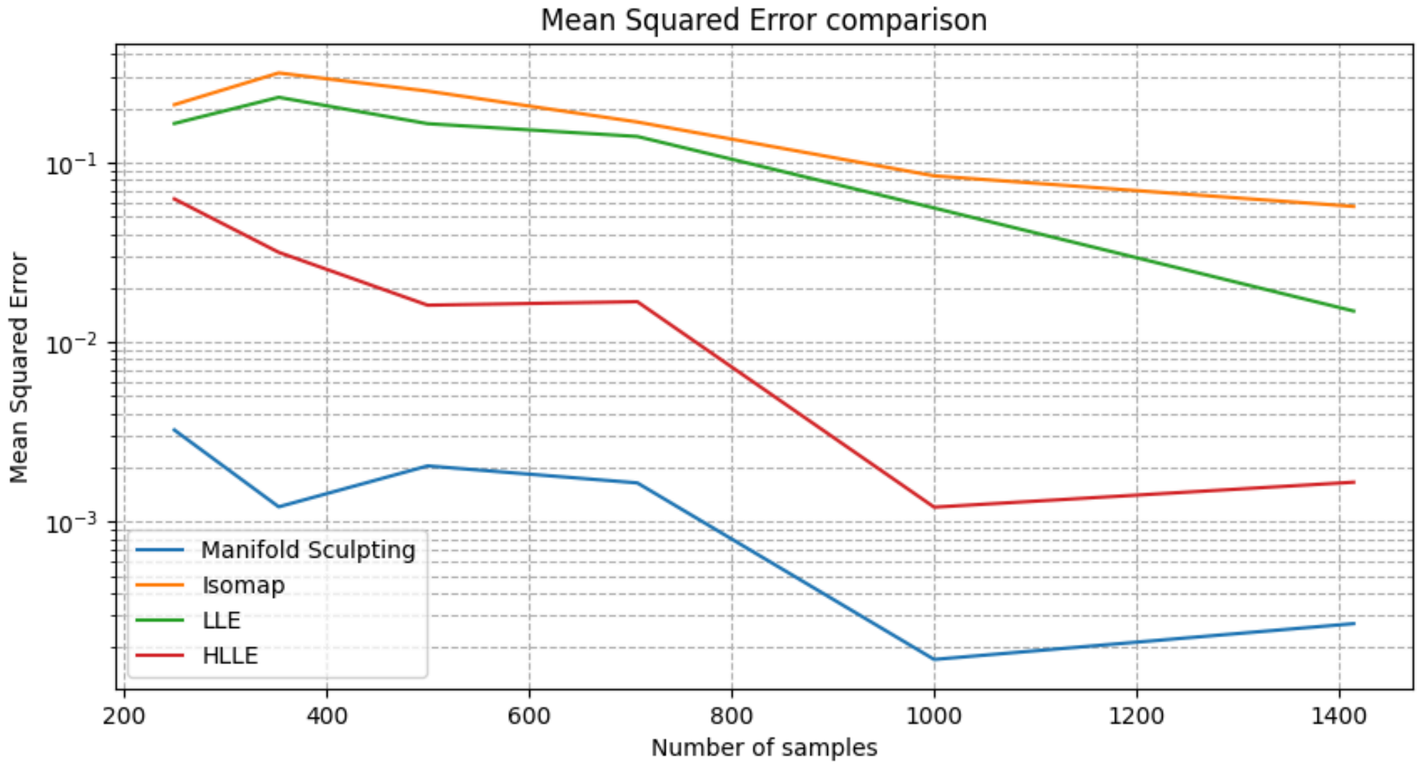
Comparison (quantitative)

First comparison:
Apply different Dimensionality reduction techniques on a Swiss Roll manifold of 1000 samples, with different neighborhood size.
Both the original manifold and the ones obtained using dimensionality reduction are centered, aligned with the principal components and min-maxed, in order to obtain the fairer comparison possible.



Comparison (quantitative)

Second comparison:
Apply different Dimensionality reduction techniques on an s-curve manifold, on a neighborhood parameter of 10 and varying sample size.
Both the original manifold and the ones obtained using dimensionality reduction are centered, aligned with the principal components and min-maxed, in order to obtain the fairer comparison possible.



Conclusions

- This algorithm is not guaranteed to converge to a global minimum, mainly due to the optimization technique. This risk, however, is mitigated by the preliminar alignment with the principal components of the data. Furthermore, I tried to apply some further stochasticity to the process, by scaling at each iteration the learning rate. This attempt to reduce the risk to fall in a local minimum worked in the majority of the times, but still there were some runs in which the unfolding got stuck.
- It is very sensitive to hyperparameters (which are many) like the neighborhood size, the scaling factor, or what I called “patience” parameter i.e. the maximum number of iterations without global improvement.
- Although it is cheaper in terms of memory, it resulted by orders of magnitude slower, and considering that at each “unfolding” iteration it uses a breadth-first search, it is almost completely serial, reducing parallelization and vectorization techniques almost ineffective.
- It has the advantage to “look into the computations”, by seeing at each iteration the shape of the manifold.

Thank you for your attention!