Advanced Machine Learning Comparison between LLE and HLLE

Havy Thomas
School of Engineering (STI), Microengineering
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: thomas.havy@epfl.ch

Weber Antoine
School of Engineering (STI), Microengineering
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: a.weber@epfl.ch

***Abstract*—In this report we discuss the differences of two manifold learning algorithms which are Locally Linear Embedding and Hessian Locally Linear Embedding. The different effects of their hyperparameters is studied on two different classes of datasets, the first class being different toy datasets to assess qualitatively the performances of the algorithm and the second class being real high dimensional datasets. It was noticed that these two manifold learning algorithms should be used carefully as they strongly depend on their hyperparameters, mainly the number of output dimensions. Moreover, no intrinsic metrics are available to measure their performances making the choice of hyperparameters quite difficult. It was noticed that HLLE generally performs better embeddings than LLE, however its constraint on the minimum number of neighbors to approximate the hessian functional makes it impractical for high dimensional datasets.**

I. Introduction

Considering the curse of dimensionality, i.e the computational cost of ML algorithms growing at least linearly with the number of dimensions of the data, being able to reduce the dimension of the data while keeping its relevant information is essential. Finding robust algorithms able to reduce the computational load of a problem while keeping a good performance is then naturally of high importance. Moreover, it might be important to retain the structure of the data present in its original space when embedding the data in a way that reflects this structure in the lower dimensional space.

In this report we will investigate two algorithms: Locally Linear Embedding (LLE) and Hessian Locally Linear Embedding (HLLE), which are two manifold learning algorithms that could potentially solve problems of dimensionality reductions. To put it simply, a D-dimensional manifold is a structure living in a N-dimensional that can locally be described with a lower number of dimensions D; for instance, a surface is a 2D-manifold as it lives in a 3D space but can locally described with two coordinates. The goal of manifold learning (and of LLE/HLLE) is to find an underlying manifold in the data space such as the similarities between datapoints in the original space are preserved in the lower dimensional space.

II. Methods

A. Algorithms principle

1) LLE: The data living on a potentially high dimensional space may still possess a neighborhood structure like if it was sampled from a smooth manifold. The LLE algorithm works by processing a few main steps. First it computes the $k-nearest$ neighbors of all the data points in the original space considering local metrics (using the Euclidean distance, one can see the $k-nearest$ neighbors being the k closest points encapsulated in a hypersphere). Then, after defining an adjacency matrix A, one can construct a weight matrix W to define the similarities between the data points. These latter weights are defined to minimize the reconstruction error in the original space. Using the $k-nearest$ neighbors, each data points should be more or less reconstructed using a linear combination of it's neighbors as the datapoints do not necessarily define a simplex, i.e

$$x_j \approx \sum_{i \in N(j)} W_{ij} x_i$$

where $N(j)$ defines the neighborhood of x_j. Now the goal of the algorithm is to find y_j best reconstructed by the same W_{ij} minimizing the l_2 reconstruction loss in a lower dimensional space. It hence conserves a neighborhood mapping of the data.

2) HLLE: The HLLE algorithm can be compared to the LLE algorithm as they both try to minimize a reconstruction loss. While LLE simply minimizes the l_2 reconstruction error, HLLE defines the local Hessian functional on the manifold and tries to minimize it.
The algorithm also starts by defining the $k-nearest$ neighbors of each datapoints, but then by defining tangent coordinates on the local dataset, being a point and its k neighbors, it constructs the local Hessian functional by orthonormalizing the matrix $\mathbf{V^a}$ defined as

$$\mathbf{V^a} = [\mathbf{1}, \mathbf{V^i}, \mathbf{Q^i}]$$

where the columns of $\mathbf{V^i}$ are tangent coordinates function on the local dataset and $\mathbf{Q^i} = [\mathbf{v_i} \boxtimes \mathbf{v_j}]_{1 \leq i \leq j \leq d}$, d being the dimension of the initial data and $\boxtimes$ being the Hadamard product [3]. The result of the orthonormalization gives $[\mathbf{1}, \mathbf{V^i}, \tilde{\mathbf{Q}}^\mathbf{i}]$. Hence one can define the local Hessian function as $\mathbf{W_i} = (\tilde{\mathbf{Q}}^\mathbf{i})' \tilde{\mathbf{Q}}^\mathbf{i}$ [3]. Then, as in LLE, a Kernel is constructed and decomposed using spectral methods to create the lower dimensional embedding.

B. Hyperparameters & Computational Costs

Both two manifold learning algorithms have the same hyperparameters which are the number of neighbors k to be considered and the dimension d of the output space. The number of neighbors is a critical hyperparameters as it defines the number of points to be taken into account to reconstruct a target point.
Considering the dimensionality of the output space, it would depend on the user intention. Embedding a dataset into 2 or 3 dimensions allows to visually assess the results which is convenient.
The general constraint on these hyperparameters is that n being the initial dimensionality of the dataset and k have to be lower bounded by d, i.e $min(k, n) > d$. This means that the output dimension should be strictly smaller than the input dimension.

Considering the number of neighbors, it also has to be strictly greater than the dimensionality of the embedded space. Indeed, taking p neighbors spans only a space of dimension $p - 1$. It has been shown that taking a margin between k and d permits to perform better embedding [2]. This constraint has to be strictly respected for HLLE and LLE. However, the minimum number of neighbors for HLLE depends on the implementation of the Hessian matrix as it needs to approximate second derivatives.
Using *Scikit-Learn* [9], the method required $n_{neighbors} > \frac{d(d+3)}{2}$, d being the dimensionality of the output space.

The different costs of both algorithms varies whether a $sparse$ strategy is used or not. For LLE, the computational cost of the standard version scales with $\mathcal{O}(dN^2)$ where N is the number of datapoints and d the dimensionality of the output space. By using sparse LLE, this complexity can be reduced to beeing subquadratic in N [8].
For HLLE, the computational complexity of the standard version scales with $\mathcal{O}(dN^2)$. By using the sparse version, the computational complexity remains the same, meaning $\mathcal{O}(dN^2)$ [5], [9].
Both algortihms engender more or less the same computational complexity even if their loss function is not identically computed. This is due to their same critical step being the final eigen-decomposition to determine the output space, fixing the critical computational cost. Using sparse methods, LLE surpasses HLLE with a slightly smaller computational cost.

III. Results

A. Effect of hyper parameters and perturbations

We will start by presenting some qualitative results using toy datasets that we generated. Those datasets are three-dimensional, thus reducible only to 2D or 1D.

Let's start with a very simple dataset: the "mountain" dataset (Figure 1, left).

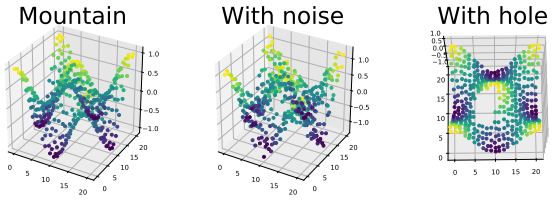

Fig. 1: Mountain dataset

Given the 3D distribution of points, both LLE and HLLE are able to reduce the dimension to 2D with very satisfying results (Figure 2). Even though the distribution is non linear, simpler algorithms such as PCA could obtain as good results, as projecting the data onto a plane preserves the similarity between the data points. Using different number of neighbors does not influence a lot the result.

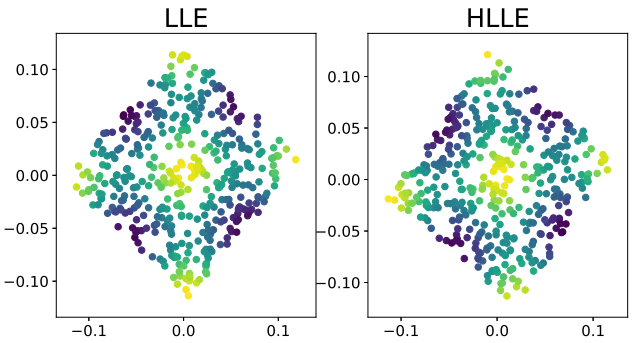

Fig. 2: Result LLE/HLLE on mountain dataset (15 neighbors)

A good point to note is that both LLE and HLLE obtain good result on the dataset in presence of noise (Dataset: Figure 1 (center), Results: Figure 3). Additionally, both methods are satisfying when the dataset has a hole in it (dataset: Figure 1 (right), result: Figure 4)

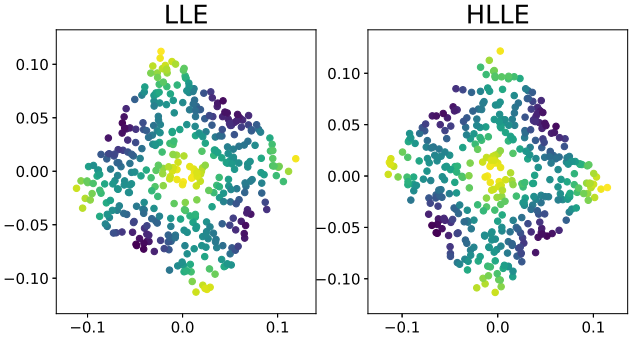

Fig. 3: Result LLE/HLLE on mountain dataset with noise

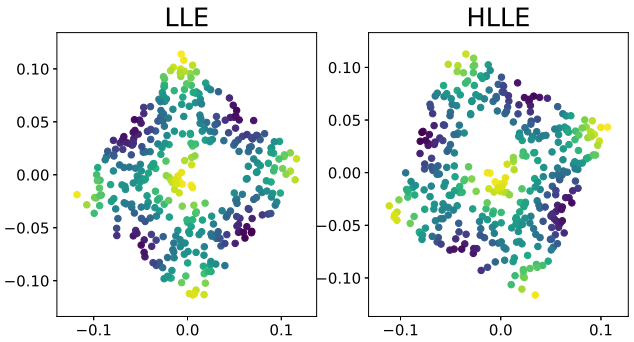

Fig. 4: Result LLE/HLLE on mountain dataset with hole

The mountain dataset is rather simple, both LLE and HLLE are able to achieve good results even with noise or holes. The superiority of Hessian LLE over standard LLE can be observed when applying these methods to the ”twist” dataset (Figure 5, left).

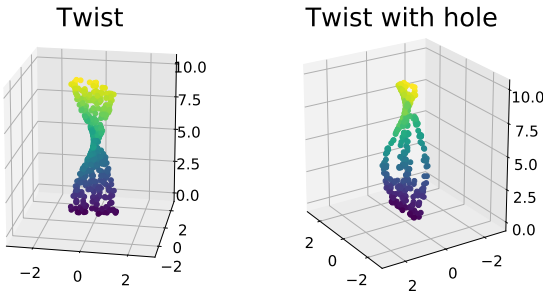

Fig. 5: Twist dataset

Because of the rotation in the surface, the LLE reduction is not satisfying around the twisting region. HLLE on the other hand is able to seamlessly project the 3D surface onto a 2D space. The number of neighbors is of course of the utmost importance, as too few of them yields poor results with the loss of the original structure of the data. Figure 6 and 7 show interesting results with different number of neighbors for both methods.

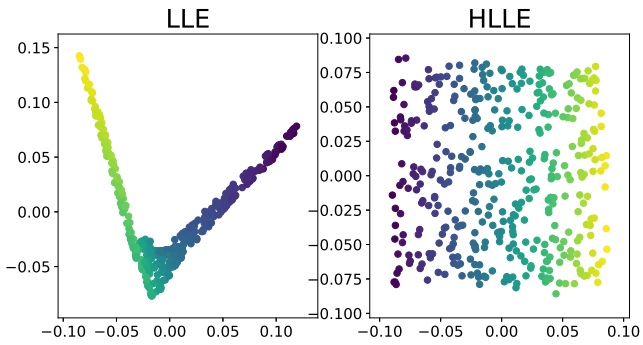

Fig. 6: Result of LLE and HLLE on twist dataset (7 neighbors)

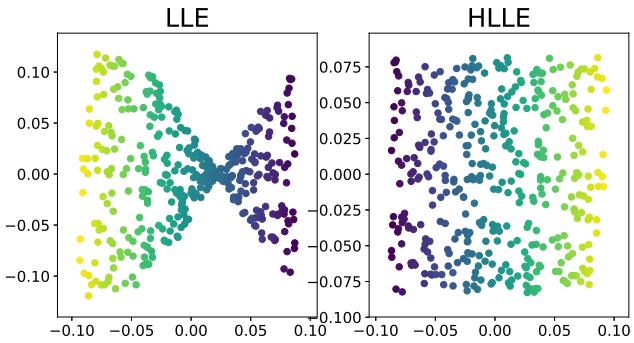

Fig. 7: Result of LLE and HLLE on twist dataset (50 neighbors)

Figure 6 and 7 illustrates both robustnesses to a change in the number of neighbors. Indeed, LLE drastically changed its representation of the data when changing the numbers of neighbors while HLLEs representation did not change at all. Moreover, LLE did not manage to correctly unfold the dataset while HLLE succeeded in doing so. Hence, for this task, HLLE surpasses LLE. More impressive results are obtained when removing parts of the twist dataset as shown in Figure 5 (right). Again HLLE does not have any problem projecting the data onto a 2D plane, and the twisted surface is nicely unfolded. LLE on the other hand does not succeed to provide a satisfactory projection. These results can be seen on Figure 8.

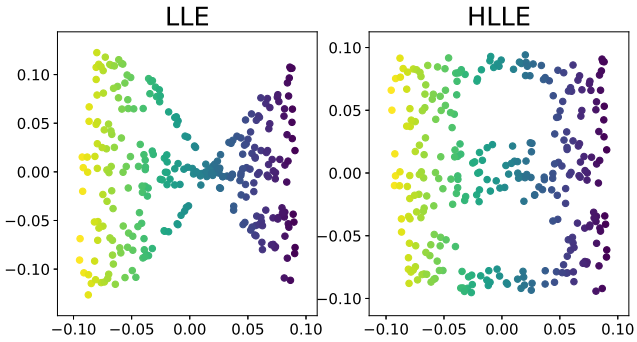

Fig. 8: Result of LLE and HLLE on twist dataset with holes (50 neighbors)

Let’s now examine a new toy dataset: the (wavy) cylinder dataset (Figure 9) that will highlight the problems that can arise when dealing with closed manifolds (that is a manifold with no boundaries such as a sphere). We use two variations of this dataset, the first one is a full cylinder while the second one is the same cylinder with a slit in it.

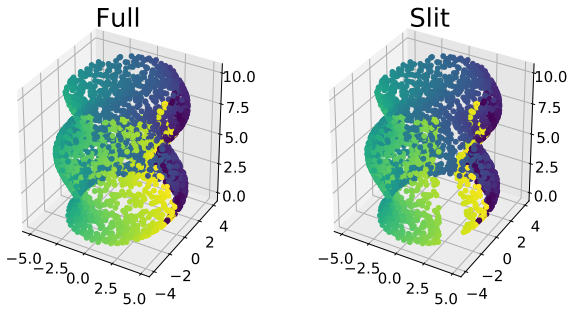

Fig. 9: Cylinder dataset

If we observe the result of LLE and HLLE for the full cylinder (Figure 10), one can see that the embedding is not satisfying as it seems that the cylinder was flattened and not unfolded. The simple explanation is that since the surface of the cylinder is connected, there is no ”edge” around which the data could be unfolded. This can be explained more tangibly by how the algorithms work. The goal of these algorithm is to preserve the structure of the data from the original space to the reduced space (refer to Section II-A for the principles used to do so). Thus, points that are close in the original space should be close in the reduced space (and similarly for points that are far away). However, every point on the cylinder has neighbors to its left and to its right in a very similar way (except towards the top and bottom edges). This makes the final step - when the new coordinates in the reduced space are computed - very difficult. This step being a minimization step on the reconstruction error of the data *in the reduced space*, it will find the projection (in our case in 2D) of the data points that minimizes this error. Although this solution is optimal with respect to the error, it is qualitatively not good as it is impossible to have all the points adequately close to their neighbors due to the circular nature of the data in the original space.

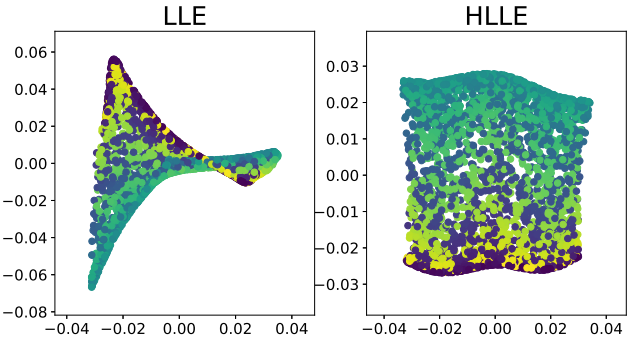

Fig. 10: Result of LLE and HLLE on full cylinder (15 neighbors)

This problem can be solved by opening the surface of the cylinder with a slit. Once, this is done, it is possible for LLE and HLLE to unfold the surface of the cylinder. Indeed, a point on one side of the slit will be unlikely to have neighbors on the other side of the slit, thus the points on the two sides of the slits can be far away in the reduced space. The results of LLE/HLLE on the cylinder with a slit can be observed on Figure 11.

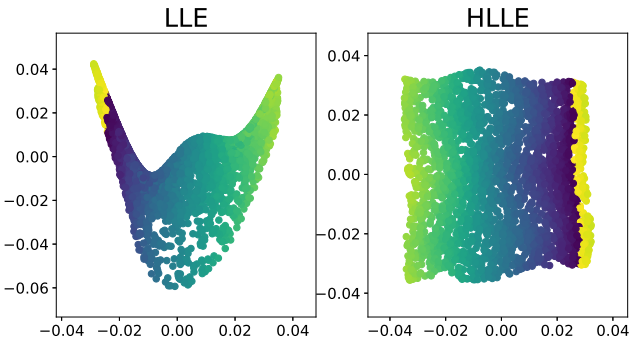

Fig. 11: Result of LLE and HLLE on cylinder with slit (15 neighbors)

B. Real high dimensional dataset

1) Caltech101 Silhouette: To complete the analysis on the toy dataset, an implementation of both algorithms was performed on a real high dimensional dataset. To best illustrate the potential of these algorithms, the image *Caltech101 Silhouette* [1] dataset was picked. It comes from the initial *Caltech101* dataset to which an edge detector followed by a binary threshold was applied. Figure 12 illustrates two images of the *plane* label.

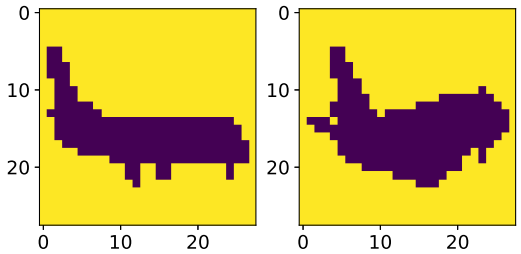

Fig. 12: Two images of the *Caltech101 Silhouette* dataset representing the *plane* label

The dataset contains a total of 8671 binary 28x28 images segmented into 101 different labels. As these images are characterized by 784 pixels, each image represent a single point in a 784D Hilbert space. Only 10 classes from the initial 101 were retained to permit shorter computational time while keeping a descent amount of data.

To have a first glance at the difficulty of the task being to group the different classes together while decreasing the dimensionality of the dataset, an embedding to a 3D space was performed. Figure 13 illustrates the resulting embedding using 300 neighbors for both methods.

[1] https://people.cs.umass.edu/m̃arlin/data.shtml

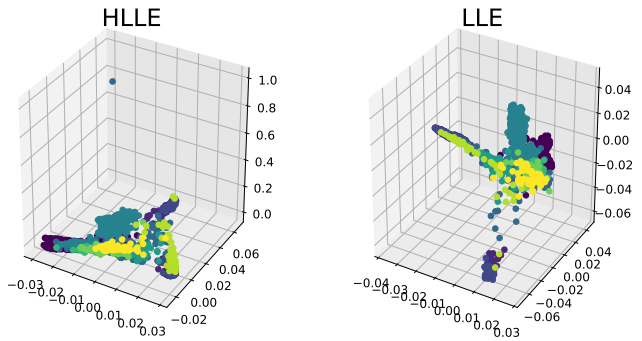

Fig. 13: Embedding of the dataset into a 3D space.

It can be observed that the different colors representing the 10 different classes are nicely grouped. To quantitatively assess the reduction of dimensionality, a classification was performed using a *Ridge Regression* technique. Table I summarizes the obtained results.

	Initial space (784D)	HLLE (3D)	LLE (3D)
Test accuracy [%]	76.35	52.54	59.76

TABLE I: Test classification accuracy using a *Ridge Regression* method on the different spaces using a train/test ratio of 60%

Indeed the accuracy decreases with the dimensionality. However, the initial dimensionality being 784, embedding the dataset into a 3D space is useful only to have a glance at the task.

As said in section II-B, the HLLE implementation of *Scikit-Learn* requires a number of neighbors being quite large with respect to the dimensionality of the embedded space. Hence, considering the number of points of the dataset, it was impossible to embed the data in more than 50D as we did not have enough points to respect the constraint. As the results of the classification after the embedding in a 50D space were still around $60\,\%$ of accuracy, the following analysis was performed only on LLE.

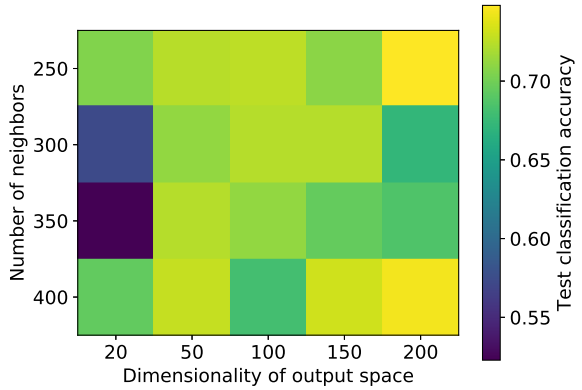

Fig. 14: Different test classification accuracies using a train/test ratio of 60% with respect to the number of neighbors and the dimensionality of the output space of an LLE embedding.

Looking at Figure 14, one can observe that the parameter having the highest impact is, not surprisingly, the dimension of the embedded space compared to the number of neighbors. After an embedding to a 200D space, the classification accuracy reached the same accuracy as in the original space. Moreover, the time gain to perform the classification is ≈ 50 meaning that the classification was performed approximately 50 times faster in the embedding than in the original space while conserving the same accuracy.

2) Wisconsin Breast Cancer: The results of the manifold learning with the previous dataset were not very conclusive, no clear manifold structure could be visualized either in 2D or 3D. In order to demonstrate better results, we also looked at the *Wisconsin Breast Cancer* dataset [2]. This dataset consists of 569 data points with 30 dimensions describing various measurements of tumors (radius, texture, area, smoothness, ...). We will not perform here a quantitative analysis of LLE/HLLE on this dataset, as such analysis has been done for the previous dataset and remains difficult (as explained in Section IV. We will instead showcase results for which we can clearly visualize the structure present in the data.

The results of applying LLE and HLLE on this dataset to embed it in 3D are shown in Figure 15.

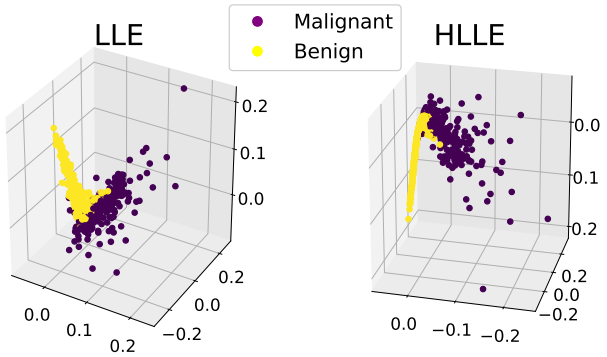

Fig. 15: Embedding of the Wisconsin Breast Cancer in 3D space (50 neighbors)

There are two points that are worst noticing on these results that apply to both LLE and HLLE. Firstly, the data seem to be distributed on a surface, with a noticeable "elbow" trend in the data. This seems like a proper manifold. Secondly, one can clearly notice the difference of distribution between the malignant (yellow) and benign (purple) tumors, with the malignant tumors being much tightly distributed (thus very similar in the original 30D space) while the benign tumors are more spread on a plane. The "elbow" occurs right at the separation between malignant and benign tumors reflecting a very clear intrinsic difference in the data, even though neither LLE nor HLLE had any information about the two classes.

The Wisconsin Breast Cancer dataset is almost linearly separable in the original space. Simple algorithms can yield very good classification results: for instance we ran a logistic

[2]http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)

regression on the original 30D data and obtained around 94% of prediction accuracy (for a train/test ratio of 66%). Embedding the data in lower dimensions does improve the results of the classification, but lowers it to around 60%, when varying the embedding dimension from 2D to 10D. It is important to emphasize that the purpose of these algorithm is not to do classification, so there is no reason for them to improve the results unless the original data lies on a manifold that can be linearly separated once learned.

Although the classification results after embedding are not better than prior to it, this example still showcases the strength of LLE/HLLE for structure discovery, as the result of both algorithm show a clear difference between malignant and benign tumors. They are also a very good tool for visualizing data, as it is difficult to make sense of a 30-dimensional data compared to 3D data that can be plotted.

IV. Discussion

Although LLE and HLLE are powerful methods for manifold learning and dimensionality reduction, one must be aware of their limitations and the problems that might occur when using it. We will address some of these issues in this section.

A. Effect of grid sampling

Something to be aware of is the way the data is sampled on the original space when using HLLE/LLE. If we sample the mountain (Figure 1, left) toy dataset on a grid we obtain a very singular result shown in Figure 16. However, with real datasets, it is unlikely that the sampling will be this precise and regular, so the result will look more like Figure 2 where the same mountain shape was randomly sampled. Such a difference does not occur when using LLE.

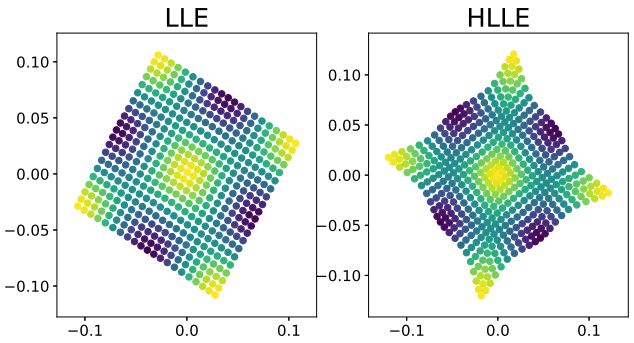

Fig. 16: Effect of grid sampling

B. Effect of non-uniform sampling

If a too regular sampling is rarely a problem with real datasets, the opposite problem - non uniform sampling - certainly is in many cases. LLE and HLLE still provide good results with non-uniformly sampled data. For instance, let's take the mountain manifold and sample it non-uniformly as such: we sample half of the points on two edges (each edge has a width of one eighth of the total domain) and half of the points on the full domain (see Figure 17).

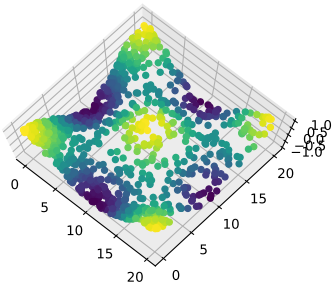

Fig. 17: Mountain dataset with non uniform sampling

The result of LLE and HLLE are as good as they are when the sampling is uniform. In the 2D embedding, the points are not mixed up (as indicated with the color) and the peak and valleys come out clearly. Furthermore, changing the number of neighbors can help improving the quality of the embedding. Some representative results are shown on Figure 18. It is good to note that even with few neighbors HLLE is capable of preserving the overall geometry whereas LLE introduces some distortion in the embedding because of the difference of density.

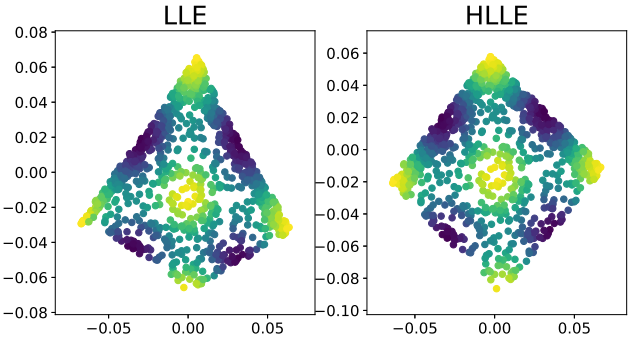

Fig. 18: Result of LLE and HLLE on non-uniformly sampled mountain dataset (10 neighbors)

C. Dataset comprising multiple manifolds

A more limiting problem of LLE and HLLE arises when the data in reality does not lay on a single manifold but on multiple. For instance if we have the data of Figure 19, which is composed of 2 parabolas.

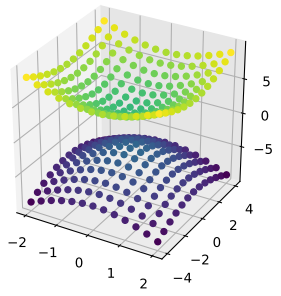

Fig. 19: Two parabolas dataset

In this case, neither LLE or HLLE is able to correctly transform the data from 3D to 2D. The result using LLE is visible on Figure 20, where one parabola was projected onto a line and the other onto a single point. This problem is due to the fact that the closest neighbors of a point in one parabola are likely also in the same parabola, thus the two sets of points in each parabola don't share common neighbors and the dimensionality reduction cannot be done properly. When there are disconnected sets of points in the neighbor connectivity graph, it may be necessary to run LLE/HLLE on each of these sets and obtain several manifold [1].

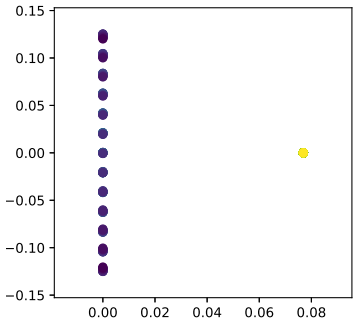

Fig. 20: Result of LLE with 20 neighbors on the two parabolas dataset

D. Assessing the results of LLE/HLLE

It is generally noted that LLE embeds distorsion of the initial dataset into the lower dimensional space. Such a drawback is not present when dealing with HLLE. Indeed this effect may be problem for some cases but also may not be of huge importance for other cases. Looking at Figure 6, the created distorsion by LLE is handicaping in this case as the spread along the horizontal axis of the initial dataset is lost. However, looking at Figure 18, the embedded distorsion should not be of huge importance.
This effect should still be considered as a drawback of LLE as, as mentioned previously, in some cases it can be a cause of information loss. Moreover, HLLE do not embeds such behaviors.

Another drawback of LLE and HLLE is the difficulty to assess the quality of the embedding obtained with either one of the methods. For low-dimensional datasets such as the toy datasets discussed in this report, a visual inspection can be performed to qualitatively assess the fitness of the embedding. However, even in these simple cases it is difficult to decide which embedding is better than another (when varying the number of neighbors used for instance). If there is a subsequent application to be performed on the data (such as clustering or classification), the performance metrics used for the said application can be used to have an idea of the quality of the embedding. However, there is no intrinsic measure of the quality of the result, but some perspectives are explored in [6].

E. Finding the true dimension of an underlying manifold

The problem of the quality of the embedding is even more important in manifold learning, when the actual dimension of the manifold that needs to be learned is unknown, or when the very existence of the manifold is speculated. In this case, great care has to be taken to select the dimension of the embedding, prior to trying different number of neighbors as both parameters influence a lot the results.

A good example to illustrate this is the spiral dataset (Figure 21). This is a curve in a 3D space, which is obviously a 1D manifold.

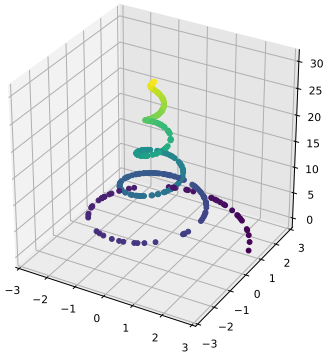

Fig. 21: Spiral dataset

When using LLE or HLLE to embed this data in one dimension, both methods yield satisfying results (see Figure 22).

Fig. 22: Result of LLE and HLLE on the spiral dataset embedded in 1D (20 neighbors)

However, when embedding this data in 2D, the number of neighbors chosen influences a lot more the results which can be either fine (Figure 24) or terrible (Figure 23). Indeed

with a sufficiently large number of neighbors each point will have neighbors not only to their side (really close) but also neighbors further away on different section of the spiral. The reason HLLE provides better result in Figure 24 is because it considers the Hessian when finding the embedding, allowing it to take more into account the variation of curvature in different neighbor directions.

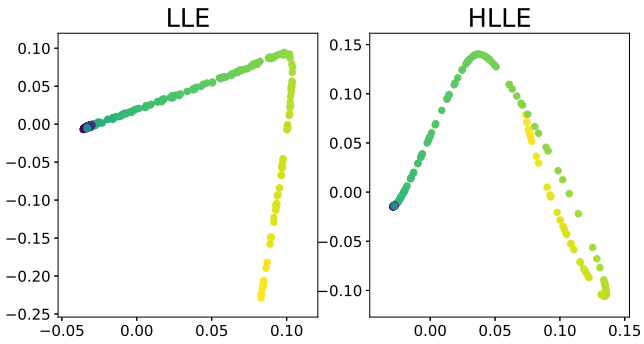

Fig. 23: Result of LLE and HLLE on the spiral dataset embedded in 2D (20 neighbors)

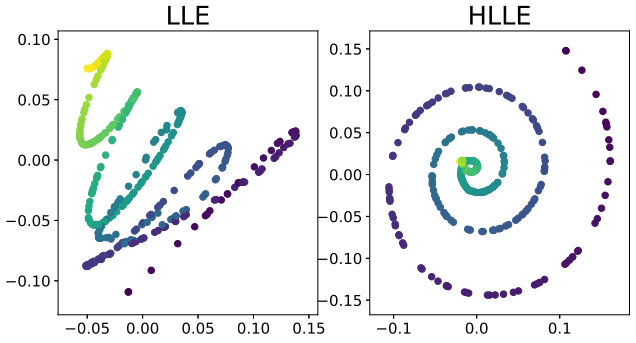

Fig. 24: Result of LLE and HLLE on the spiral dataset embedded in 2D (60 neighbors)

F. Offline learning

An additional drawback of both LLE and HLLE is that in can only be used for offline learning. The learning must be done on the entire dataset at once, and cannot be improved with new data without having to run the algorithm again. This is a limitation as it requires to have a large static dataset ready, and neither algorithm can be used when the data is acquired continuously.

For the different computational costs, even if their general expression do not mention the number of neighbors, it was observed than the more neighbors are chosen, the slightly longer the computations.

G. Constraint on the number of neighbors for HLLE

The major drawback of HLLE is the constraint on the required number of neighbors with respect to the dimensionality of the output space. Indeed, it was observed that an embedding using HLLE quickly becomes unpractical due to the lack of datapoints needed for approximated the second derivative. Please refer to Section II-B for more information.

H. Discussion about the real Datasets

Considering the *Caltech101 Silhouette* dataset, it was observed that the classification accuracy was not increasing after the implementation of the manifold learning algorithms. Such an observation may come from the fact that the initial images are binary and not continuous. Hence, defining manifolds with such a dataset may be difficult, resulting in poor embeddings using manifold learning. In this case, changing the number of neighbors did not drastically change the results.

The quantitative analysis could not be performed on this dataset using HLLE. This illustrates again the major drawback of HLLE being the constraint on the minimum number of neighbors required to approximate the hessian functional. LLE not being affected by such a hard constraint, it was possible to embed the data in a lower dimensional space while keeping a descent accuracy. Hence, LLE surpasses HLLE for this problem.

Considering high dimensional datasets, manifold learning algorithms do not have a metric to assess their performances. Hence, the cases where the data cannot be embedded in a 2D or 3D space cannot be directly analyzed. To quantify the quality of the embedding, a classifier has to be trained. This drawback stands for all the dimensionality reduction algorithms.

For the *Wisconsin Breast Cancer* dataset, it was observed that both LLE and HLLE algorithms succeeded in embedding the data in a lower dimensional space while keeping the two tumors separated while embedding it in a 3 dimensional space. This way, the quality of the embedding could be visually assessed.

V. Conclusion

The different behaviors with respect to hyperparameters and perturbations has been assessed with custom toy-datasets. The main outcome is that HLLE was generally a better choice when dealing with low dimensional dataset, however its constraint on the minimum number of neighbors makes it impractical for high dimensional dataset, as it imposes a low-dimensional embedding. It has been noted than HLLE generally possess a better robustness regarding changes in its hyperparameters than LLE. Moreover, LLE may embed distorsions whenever the hyperparameters are not chosen wisely while HLLE is adapting better to variations of shape/curvature in the data.

Both algorithms have been applied to 2 real high dimensional dataset: the Wisconsin Breast Cancer dataset and the Calltech101 silhouette dataset. The former yielded satisfying results while for the later the results were not encouraging. These example highlight the difficulty of assessing the correctness or quality of the outcome of LLE/HLLE.

VI. Appendix

The source code for generating the toy datasets, figures and results of this report can be found on the following GitHub repository: https://github.com/AntoineWeber/AML-Project

References

[1] Ethem Alpaydın, *Introduction to Machine Learning*, The MIT Press, Second Edition, 2009

[2] Lawrence K. Saul and Sam T. Roweis, *An Introduction to Locally Linear Embedding*

[3] Wang J., *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*. Springer, Berlin, Heidelberg, 2012

[4] Lawrence K. Saul and Sam T. Roweis, *Nonlinear Dimensionality Reduction by Locally Linear Embedding*

[5] David L. Donoho and Carrie Grimes, *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*. Springer, Berlin, Heidelberg, 2012

[6] Peng Zhang, Yuanyuan Ren, Bo Zhang, *A new embedding quality assessment method for manifold learning*. Neurocomputing (Volume 97), 2012

[7] Lori Ziegelmeier1, Michael Kirby, and Chris Peterson *Sparse Locally Linear Embedding* Colorado State University, Fort Collins, Colorado, USA

[8] Yasuhiro Fujiwara, Naoki Marumo, Mathieu Blondel, Koh Takeuchi, Hideaki Kim, Tomoharu Iwata, Naonori Ueda *Scaling Locally Linear Embedding*

[9] F.Pedregosa, G.Varoquaux, A.Gramfort, V.ichel, B.Thirion, O.Grisel, M.Blondel, P.Prettenhofer, R.Weiss, V.Dubourg, J.Vanderplas, A.Passos, D.Cournapeau, M.Brucher, M.Perrot, E.Duchesnay *Scikit-learn: Machine Learning in Python*