Advanced Machine Learning “LLE vs MLLE”

Arnaud Guibbert
School of Engineering (STI)
École polytechnique fédérale de Lausanne (EPFL)
Lausanne, Switzerland
Email: arnaud.guibbert@epfl.ch

***Abstract*—Dimensionality reduction is a wide branch of Machine Learning. Such methods are able to map high dimensional data to a lower dimensional space while preserving meaningful data. These techniques are really useful to extract the data of interest and are widely used in field such as image processing, data visualization ... They are even more important in today’s society, as the acquired data are increasingly high dimensional. In this paper two popular dimensionality reduction methods are presented and compared on a real data set.**

I. Introduction

This paper aims at comparing two dimensionality reduction methods: Locally Linear Embedding (LLE) and one of its variants Modified Locally Linear Embedding (MLLE). These two methods belong to the class of the spectral non-linear dimensionality reduction methods. We recall that a dimensionality reduction technique is a method that maps data points from a high dimensional $\mathbb{R}^n$ space to a lower dimensional space $\mathbb{R}^d$ (d is chosen such that $d \ll n$). The lower dimensional space is commonly called the embedding space. The goal of the dimensionality reduction methods is to encapsulate the meaningful information of the data into the embedding space. This paper will then assess how well LLE and MLLE achieve this task on a real data set.

In this article, the main aspects and characteristics of these algorithms will be recalled in section II. Then, in section III, the ability of these algorithms to encapsulate the data of interest into a lower dimensional space will be assessed on a real data set. A broad description of these data sets is provided in section III-A. Finally the results obtained will be discussed in the sections IV and V.

In the rest of this paper, the initial dimension of the data points (input dimension) will be denoted by n, the output dimension by d and the number of data points by M.

II. Methods

A. Theory

As the theory will support and underpin the results presented in section III, a short recap about the key steps of these two algorithms is necessary. However as it is not the main purpose of this paper, this recap does not intend to be exhaustive and is only focused on the main results, not on the derivations of them. An exhaustive description of these algorithms is available in these three articles [1] & [2] & [3].

As it was underlined in the introduction, the two methods have a common structure: actually MLLE derives from LLE. Therefore, the LLE algorithm will be tackled first, then it will be shown how to move from LLE to MLLE. Most of the elements of the theory discussed below are taken from these to papers [2] & [3].

Locally Linear Embedding:
LLE aims at preserving the local structure of the data points. In other words, LLE aims at preserving the neighborhood of each data point. To do so, LLE starts by computing the neighborhood of each data point using the k-nearest-neighbors algorithm. As it will be tackled later, the number of neighbors k is a hyperparameter of both methods. Then each data point is approximated by a linear combination of its neighbors. This task is achieved by minimizing the following quantity over the weights $W = [w_1, w_2, ..., w_M]^\top \in \mathbb{R}^{M\times k}$:

$$\begin{gathered} \min_{w_1,\dots w_M} \sum_{i=1}^{M} \| x_i - \sum_{j=1}^{k} w_{ij} x_j^{(i)} \|^2 \\ \text{subject to} \ \sum_{j=1}^{k} w_{ij} = 1 \ \ \forall i \end{gathered} \tag{1}$$

Where $X = [x_1, ..., x_M]^\top \in \mathbb{R}^{M\times n}$ are the M data points and $x_j^{(i)}$ is the j^{th} neighbor of the data point x_i. This optimization problem can be rewritten as:

$$\begin{gathered} \min_{w_1,\dots w_M} \sum_{i=1}^{M} w_i^\top G_i w_i \\ \text{subject to} \ \ \mathbf{1}^\top w_i = 1 \ \ \forall i \\ \text{where} \ \ G_i = (x_i\mathbf{1}^\top - X_{J_i})^\top (x_i\mathbf{1}^\top - X_{J_i}) \in \mathbb{R}^{k\times k} \end{gathered} \tag{2}$$

Where $X_{J_i} = [x_i^{(1)}, x_i^{(2)} ..., x_i^{(k)}]^\top$. One can show that the minimizer [1] of this problem is given by:

$$\forall i \ \ w_i = \frac{G_i^{-1}\mathbf{1}}{\mathbf{1}^\top G_i^{-1}\mathbf{1}} \tag{3}$$

This form assumes that the G_i matrix is invertible. However one can notice, from (2), that the rank of the G_i matrix is upper bounded by $\min(k, n)$. Then if $k > n$, the G_i matrix

[1] The full derivation is available in this paper [3]

is singular, meaning that the problem does not have a unique global minimizer. In LLE this problem is solved by adding a regularization term $\gamma > 0$ such that G_i is replaced by $G_i + \gamma I$, and then the problem has a unique minimizer. On the contrary, MLLE will exploit the non-uniqueness of the solution.

The second step in LLE is to reconstruct the neighborhood of each data point in a lower dimension space. LLE aims at finding latent vectors $y_i \in \mathbb{R}^d$ that are preserving the neighborhood of each data point. This task is achieved by minimizing the following quantity over the output $Y = [y_1, y_2, ..., y_M]^\top \in \mathbb{R}^{M\times d}$:

$$\begin{aligned} \epsilon_{LLE} = \min_{y_1,\dots y_M} \quad & \sum_{i=1}^{M} \parallel y_i - \sum_{j=1}^{k} w_{ij} y_j^{(i)} \parallel^2 \\ \text{subject to} \quad & \sum_{i=1}^{M} y_i = 0 \ \text{ and } \ \sum_{i=1}^{M} y_i y_i^\top = I \end{aligned} \tag{4}$$

To write the problem in a more compact form, one needs to define an extension of the matrix W. This extension $\hat{W}$ is defined as follow:

$$\hat{w_{ij}} = \begin{cases} w_{ij} & \text{if } x_j \text{ is a neighbor of } x_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

One can notice that the matrix $\hat{W}$ is sparse. Using $\hat{W}$, this minimization problem can then be factorized and rewritten as follow:

$$\begin{aligned} & \min_{Y} \quad \text{Tr}(Y^\top B Y) \\ \text{subject to} \quad & \sum_{i=1}^{M} y_i = 0 \ \text{ and } \ \sum_{i=1}^{M} y_i y_i^\top = I \\ \text{where} \quad & B = (I - \hat{W})^\top (I - \hat{W}) \ \in \mathbb{R}^{M\times M} \end{aligned} \tag{6}$$

One can show that this problem can be solved using eigen-decomposition. Indeed the minimizer is given by $Y = [u_2, u_3..., u_{d+1}]$ where u_i are the d eigenvectors of B associated to the d smallest eigenvalues[2]. Thus the reconstruction error may be rewritten as a sum of eigenvalues:

$$\min_{Y} \quad \text{Tr}(Y^\top B Y) = \sum_{i=2}^{d+1} \lambda_i \tag{7}$$

Where $\lambda_2, \lambda_3 ... \lambda_{d+1}$ are the d smallest eigenvalues of B.

Modified Locally Linear Embedding:
MLLE is performing the same first optimization problem defined in equation (2). It has been shown that this problem may have multiple solutions. MLLE exploits this non-uniqueness and will reconstruct the points in the embedding using multiple weights. Thus instead of minimizing the quantity defined in (1), MLLE minimizes the following quantity:

$$\begin{aligned} \epsilon_M LLE = \min_{y_1,\dots y_M} \quad & \sum_{i=1}^{M} \sum_{l=1}^{s_i} \parallel y_i - \sum_{j=1}^{k} w_{ij}^{(l)} y_j^{(i)} \parallel^2 \\ \text{where} \quad & w_i^{(l)} = (1-\alpha_i) w_i + V_i J_i(:, l) \end{aligned} \tag{8}$$

[2]The first eigenvector associated to the smallest eigenvalue is not retained, since this eigenvector is equal to $\mathbf{1}$ because of the constraints of equation (4)

This quantity is minimized under the same constraints defined in (4). We will not delve into the detail of the expression of $w_i^{(l)}$, since the key idea to keep in mind is that each data point is reconstructed not only by one weight vector but by s_i weight vectors[3]. In MLLE, the number of weight vectors s_i used to reconstruct each data point is upper bounded by $k - d$, and the algorithm is designed such that half of the data points have $s_i = k - d$. This last information must be kept in mind to analyze the results. Finally one can show that the optimization problem, as for LLE, can be rewritten as an eigenvalue problem. The minimizer Y is given by $[u_2, u_3..., u_{d+1}]$ where u_i are the eigenvectors of a matrix $F_d \in \mathbb{R}^{M\times M}$ associated to its d smallest eigenvalues[2]. As for LLE, the reconstruction error can be rewritten as a sum of eigenvalues:

$$\min_{Y} \quad \text{Tr}(Y^\top F_d Y) = \sum_{i=2}^{d+1} \lambda_{i,d} \tag{9}$$

One can remark that the eigenvalues depend on the output dimension. This comes from the fact that the number of weight vectors s_i used to reconstruct the data points depends on d since $0 \leq s_i \leq k - d$.

B. Hyperparameters

These two algorithms have only one hyperparameter that is the number of neighbors k used to build the neighborhood of the data points. The latter is crucial: if k is too small, it will not acquire enough information to reconstruct the local structure in the embedding space. On the contrary if k is too high there is a risk to embed the global structure of the data points instead of the local one. Furthermore, regarding MLLE, half of the data points will be reconstructed with $k - d$ different weight vectors. Hence when k becomes high, MLLE tries to fit the neighborhood with a high number of weight vectors. Then one risks fitting the neighborhood with an overly complex model. As it will be shown in section III, this will lead to a poor fit.

In addition to this hyperparameter, one can play on the modularity of the algorithm by modifying the distance norm used to construct and reconstruct the neighborhood. Nevertheless, this last point will not be tackled in this paper.

C. Computational cost

To fairly assess the performance of an algorithm, the complexity of this one must be taken into account. The complexity of LLE is given by:

$$O(\underbrace{n \log(k) M \log(M)}_{1} + \underbrace{M k^3}_{2} + \underbrace{d M^2}_{3}) \tag{10}$$

The first term corresponds to the search of the k nearest neighbors. The second one corresponds to the inversion of the G_i matrices and the third one is the cost associated with the partial eigenvalue decomposition. The complexity of MLLE is equal to the complexity of LLE plus a computational cost associated with the partial eigenvalue decomposition of the matrix G_i

[3]The full derivation is available in this paper [2]

to further determine the s_i weight vectors. The additional computational cost is equal to $O(M(k-d)k^2)$. This additional cost induces significant difference with LLE complexity only when $O(k^3M)$ dominates $O(nlog(k)log(M)M + dM^2)$. In the other cases, the complexity of the two algorithms is roughly the same.

In short, one should keep in mind that the computational cost for both algorithms grows linearly with the input and output dimensions (n & d), quadratically with the number of data points M and cubically with the number of neighbors k. These statements will be further confirmed by results presented in section III-E.

D. Metrics

To assess the performances of these algorithms a set of different metrics has been chosen. These metrics will help to first determine the best hyperparameter, then the dimension of the embedding space, and finally compare the performance of the two methods.

The first metric is the neighborhood reconstruction error defined in (4) for LLE and (8) for MLLE. It will be denoted by ϵ and as described in section II-A, it can be rewritten as a sum of eigenvalues for both algorithms. This intrinsic metric gives an insight on how well the neighborhood is reconstructed in the embedding space. Unfortunately the form of this metric differs from one algorithm to another[4]. Thus a fairly comparison based on this metric cannot be performed. Actually one should expect that $\epsilon_{LLE} \leq \epsilon_{MLLE}$ due to (8). This statement will be confirmed in section III. Therefore this metric will essentially help to find the best k and d.

This brings us to use other metrics to be able to compare the algorithms. Furthermore one is looking for metrics that give a measure of how well the data of interest is encapsulated. Indeed ϵ does not give any information about how well the meaningful data is extracted. To do so, a common metric used in dimensionality reduction is the residual variance of the pairwise distances denoted σ_R^2. Mathematically it is defined as:

$$\sigma_R^2 = 1 - \rho_{D_X D_Y}^2$$

$$\rho_{D_X D_Y} = \frac{\text{Cov}(\parallel X_i - X_j \parallel, \parallel Y_i - Y_j \parallel)}{\sigma_{\parallel X_i - X_j \parallel} \; \sigma_{\parallel Y_i - Y_j \parallel}} \quad i \neq j \tag{11}$$

Where X_i and Y_i are the i^{th} data point respectively in the input and embedding spaces. Concretely it computes the correlation between the pairwise distances in the input dimension and the one in the output dimension. In other words, it assesses to what extent the distances between points are preserved in the embedding space. This extrinsic metric has the good property to be normalized into $[0, 1]$ that allows to fairly perform the comparison. Besides many papers show that this metric is suitable for determining the best hyperparameter k.

[4]Indeed equations 4 and 8 show that for LLE only one reconstruction error is considered for each data point while for MLLE s_i reconstruction errors are summed for each data point

To objectively assess the performances of these algorithms, the time complexity will be evaluated by monitoring the elapsed time. In addition to these four metrics a visualization of the data in the embedding space will be provided through a pairplot, one can directly have a look on it on figure 2.

Before diving into the results, given the theory, let's discuss the expected shapes of the curves for each metric. Regarding the reconstruction error, one may expect low values for small k and a high slope. This intuition comes from equations (4) and (8): as the number of neighbors is growing up, the neighborhood considered for each data point is growing up (more weights), and this leads having more constraints on the latent vectors y_i. Then it becomes more and more difficult to find latent vectors that completely fit with their neighborhood. Besides, regarding MLLE, the reconstruction error should grow up faster with respect to k as half of the data points are reconstructed with $k - d$ weight vectors. Regarding the residual variance, it is quite difficult to guess anything as it is an extrinsic metric and thus does not explicitly appear in the theory. Nevertheless for low values of k one should expect a high residual variance, as the pairwise distances are not likely to be encapsulated using small neighborhoods. One should also expect that as k increases, this residual variance decreases as more information is encapsulated, thus pairwise distances are more likely to be preserved. Considering time complexity, expectations have been already tackled in section II-C.

III. Results

A. Data set & Pre-processing

The performances of these methods will be assessed on a e-mail data set[5]. This data set consists of 4051 e-mails. Each mail is characterised by 57 continuous/integer features:

- 48 features correspond to the occurrence percentage of well chosen words in the e-mail.
- 6 features correspond to the occurrence percentage of well chosen characters in the e-mail.
- the last three features are relative to statistics about capital letters in the e-mail.

Each mail is then classified as spam (label = 1) or not-spam (label = 0). The class distribution is such that 39,4% of the data points are labeled 1, therefore the data set is slightly unbalanced. Of course, these labels will not be seen by the dimensionality reduction algorithm, as we want to assess the ability of these methods to extract the data of interest. Then LLE and MLLE will take as input the 57 continuous/integer features.

To be able to use these data, some slight modifications shall be applied to the initial data set. The data set does not present any missing values, however as one can remark the features are not expressed in the same range. The first 54 features belong to $[0, 100]$ (percentage) while the last three ones belong to $[0, +\infty]$. As the first step of LLE and MLLE is to find neighbors using the euclidean distance, some issues may occur. To normalize the data, one could classically subtract the mean

[5]Spambase data set, besides a wide documentation is available in the code

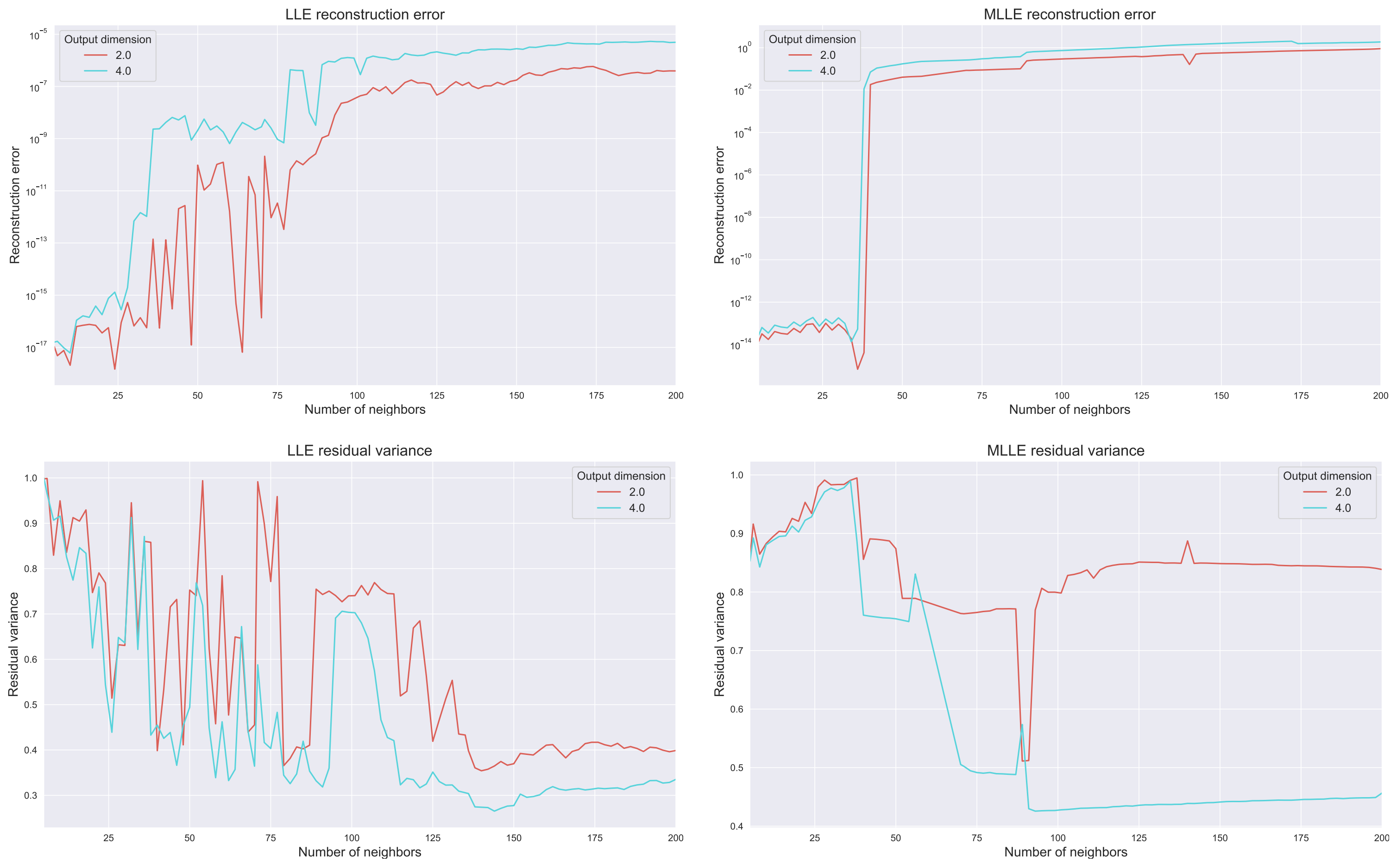

Fig. 1: On top graphs, the evolution of the reconstruction error with respect to the number of neighbors is plotted for two different values of d. On graphs at bottom, the residual error is plotted in the same way. These results are obtained using the **whole data set**

and divide by the standard deviation. Nevertheless dividing by the standard deviation would erase some information from the initial data set: indeed the first 54 features are expressed in the same unit and the difference of values/variance between features is information that shall be retained[6]. To do so, only the last three features are rescaled in $[0, 100]$ such that features are comparable. In addition, the data are centered by subtracting the mean.

B. Experimental set-up

This section is describing the parameters chosen for the hyperparameters search, the way to compute the metrics and the main content of the graphs.

Regarding the hyperparameter k, a grid search is performed to determine the best k. Furthermore, as one aims at finding the dimension of the embedding space, a similar grid search is performed over the output dimension d. The grid search parameters are summed up in the table I.

[6]for instance, if a word has an occurrence percentage of 40% and an other word has an occurrence percentage of 10% then one shall preserve the fact that the first word is four times more frequent than the second one. Dividing by the standard deviation will erase this information

	Number of neighbors k	Embedding space dimension d
Number of values	100	4
Search range	$[5, 200]$	$[1, 4]$

TABLE I: Grid search parameters

The search range for k is a parameter that should be well chosen. The one chosen here is the result of several attempts: we first started on a wider range with a lower granularity to determine the region of interest. Once it was done, the size of the search range around the region of interest was chosen such that the granularity was high enough (granularity of 2). As it will be shown in the results, this high granularity is necessary to determine the best hyperparameter.

Four different graphs are plotted to be able to visualize the evolution of the metrics with respect to k and d. A brief description of their content is provided below.

- The reconstruction error and the residual variance are plotted with respect to k. Different curves are plotted so that each curve corresponds to a different value of the output dimension d. To preserve the readability of the

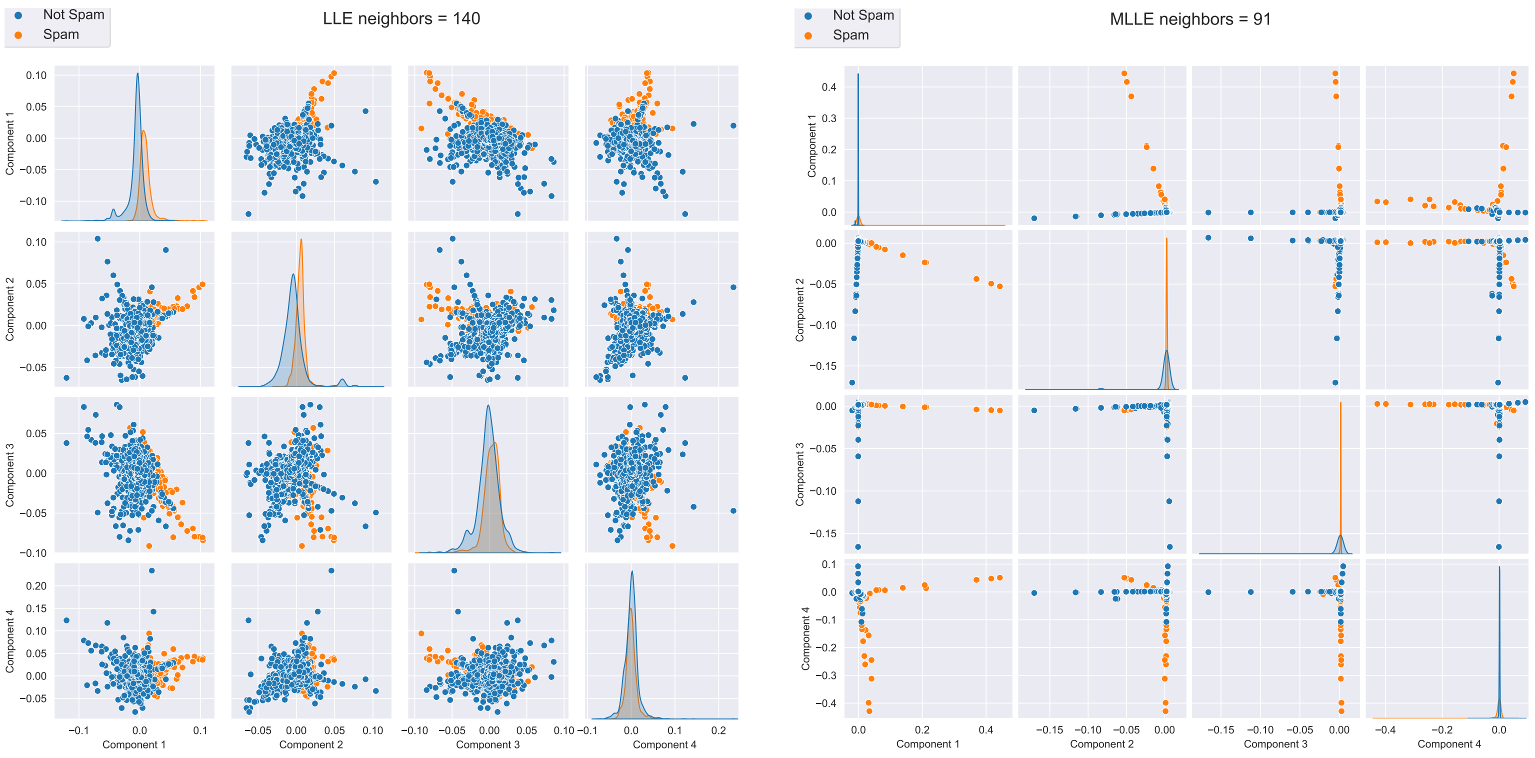

Fig. 2: Data points mapped into the embedding space. The four first components of each latent vectors are plotted on the right for LLE, and on the left for MLLE. The choice of number of neighbors comes from the previous hyperparameter analysis. This results were obtained using the **half of the data set**.

graphs only two curves (instead of four) are plotted[7]. An example of these graphs can be visualized on figure 1.

- Once the best hyperparameter has been found, the output data using this latter will be visualized using a pairplot. Only the first four components will be plotted to preserve the readability of the graph. Such a graph can be visualized on figure 2.
- In addition, the time performances of LLE and MLLE are plotted with respect to the number of neighbors. Different curves are plotted: they correspond to different number of data points. Such a graph is available on figure 5.

C. *Performance*

This section aims at describing and explaining the results obtained on the real data set, determining what is the best hyperparameter and identify the strengths and weaknesses of both algorithms. Besides that, the sensitivity of the methods to the number of data points M will be evaluated by running the same experiments on half of the data set.

Let's start on the full data set with the reconstruction error and the residual variance. These metrics are plotted on figure 1. As expected in section II-D, the reconstruction error increases with k and $\epsilon_{LLE} \leq \epsilon_{MLLE}$. One can remark that ϵ_{LLE} is sensitive to k as it oscillates before reaching a steady-state for $100 \leq k$. Regarding MLLE The reconstruction error curve is split into two distinct areas. These two areas are separated by a high slope around $k = 30$. The theory can to explain this result: according to section II-A MLLE is modeling the neighborhood of each data points with s_i weight vectors. For half of the data points $s_i = k - d$. Qualitatively, each time k is incremented, s_i is incremented by the same amount. When s_i and k are small it is more likely to find latent vectors y_i that satisfy each weight vector, however, as k and s_i increase, it becomes unlikely to find such vectors. Then for a certain value of s_i some distances in the sum over s_i in equation (8) explodes: this leads to the famous steep slope. This behavior can also be observed for LLE but it not as blatant as it is for MLLE. This comes from the fact that LLE is selecting only one weight vector.

Regarding the residual variance (again on figure 1), a similar behavior to the reconstruction error is observed with some oscillations. For low values of k the residual variance is quite high. This is due to the fact that the number of neighbors is too small to well encapsulate the local structure, hence, the pairwise distances are not preserved as well as the neighborhood. On the contrary, for high values of k, the global structure is encapsulated: the pairwise distances will be preserved. This last sentence explains the low residual variance for $140 \leq k$. For MLLE the same arguments hold. Moreover, one can identify three steep slopes: the first one is located around $k = 30$ as it was the case for the reconstruction error.

[7]Actually the number of curves plotted is justified by the almost Brownian shapes of these. Nevertheless the code provided allows you to plot the four curves

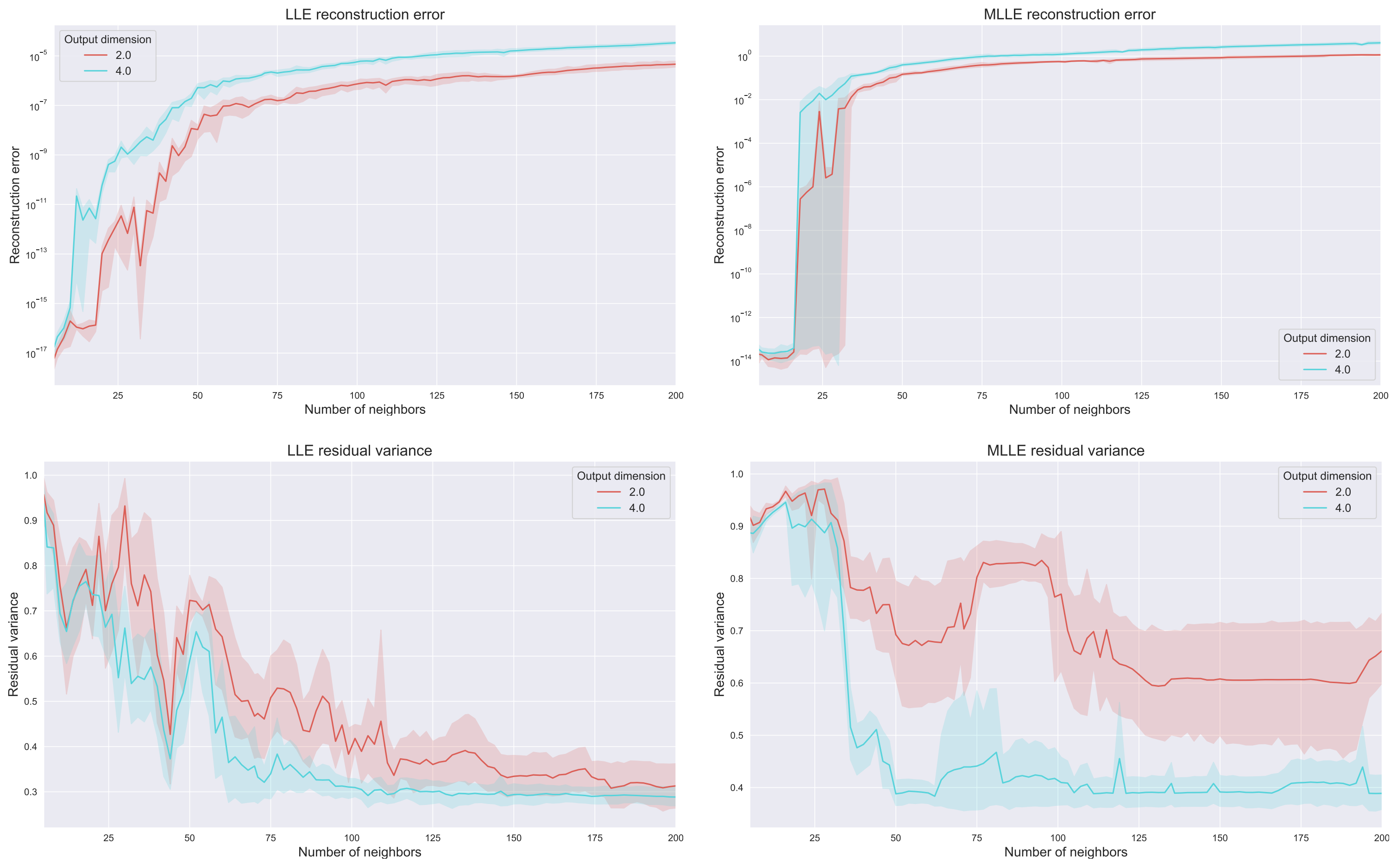

Fig. 3: On top graphs, the evolution of the reconstruction error with respect to the number of neighbors is plotted for two different values of d. On graphs at bottom, the residual error is plotted in the same way. These results are obtained using **half of the data set**

One can interpret these slopes as the adding of a weight vector (among the s_i ones) that encapsulate better pairwise distances.

Given the previous analysis and the results, the best hyperparameter k can be determined. The real question is which metric should be taken precedence over the other one. The second metric seems to be more suitable according to **put a ref** and also because it is an extrinsic metric contrary to the reconstruction error. However there are no strict rules. Therefore to determine the best hyperparameter, one can follow these steps:

- For each metric determine a small subset of candidates to be the best hyperparameter $C_{metric} = \{k_1, k_2, ...k_p\}$, p should not exceed a given limit.
- Merge the subsets of all unions into a set gathering all the potential candidates $C_{cand} = \bigcup C_{metric}$.
- Determine the best hyperparameter using a third metric computed for each candidate of C_{cand}.

One can choose p equal to the number of global minimizer that is often low, nevertheless this number should be kept low to avoid having too many candidates. Regarding the third metric to use, we would like to have a metric that gives a measure of how well the meaningful data is encapsulated in the embedding space. Here the meaningful data is the type of e-mail: spam or not spam. To do so we will use pair plots. Because pair plots are heavy and large graphs only the one for the best hyperparameter will be plotted[8].

According to the data acquired[9], the potential candidates for k_{LLE}^{best} and k_{MLLE}^{best} are:

$$C_{cand}^{LLE} = \{24, 140, 200\} \qquad C_{cand}^{MLLE} = \{36, 91, 200\}$$

It results that $k_{LLE}^{best} = 140$ and $k_{MLLE}^{best} = 91$ seem to be a good choice. The spread of the data in the embedding space for these values can be visualized on the figure 2.

Regarding LLE, the data in the embedding space are spread in the whole space contrary to MLLE. The distributions of the data points on components 1 and 2 between classes 1 and 2 are slightly differ, nevertheless this difference is not sufficient to easily segregate the two classes. Hence LLE does not succeed

[8]Nevertheless the pair plots for the other good candidates can be easily generated with the code provided.

[9]All the data used to plot the graphs are stored in csv files and available in the *src/data* folder.

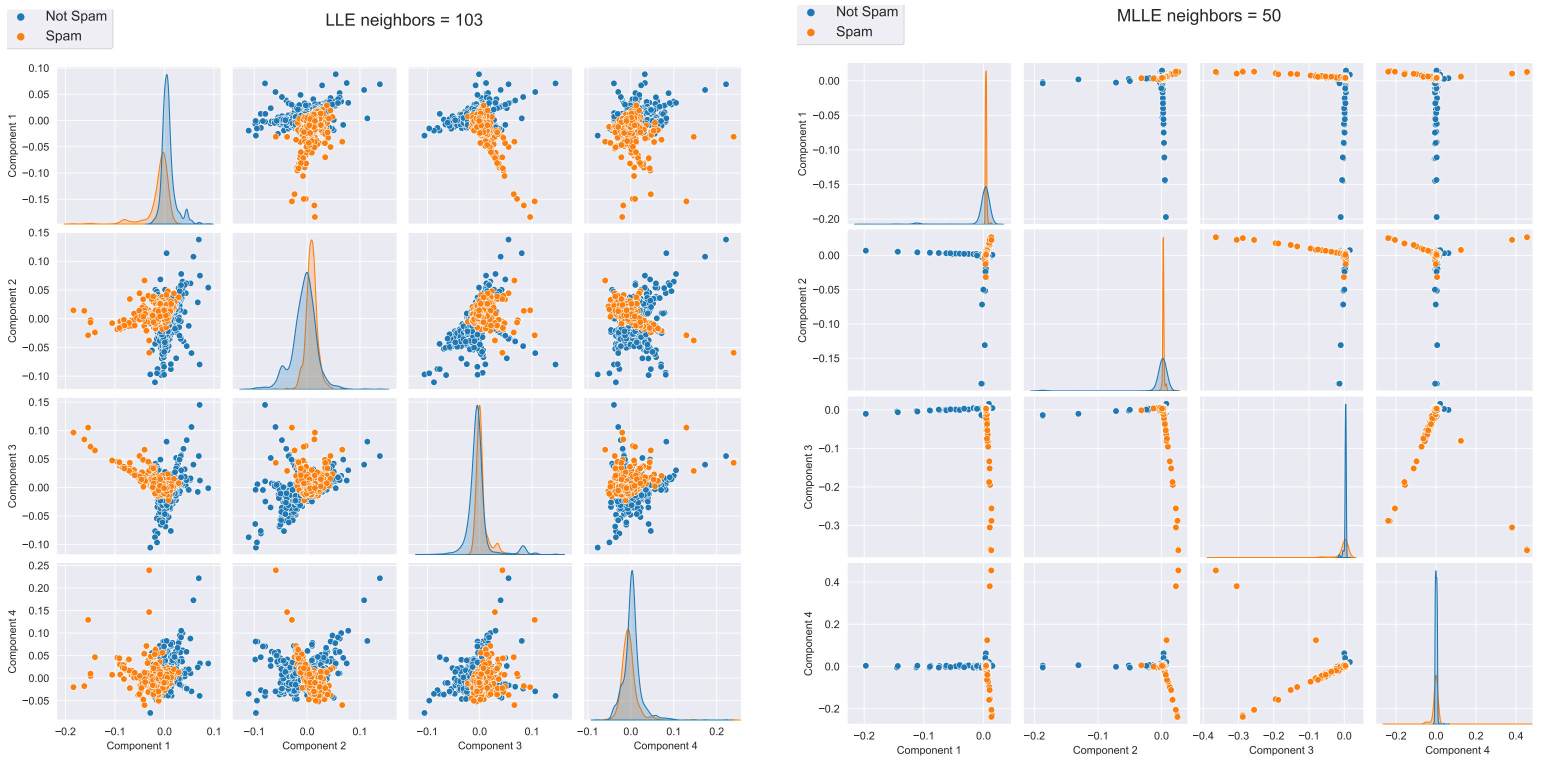

Fig. 4: Data points mapped into the embedding space. The four first components of each latent vectors are plotted on the right for LLE, and on the left fro MLLE. The choice of number of neighbors comes from the previous hyperparameter analysis. This results were obtained using the **whole data set**.

in separating the two classes and a fortiori extracting the meaningful data.

Regarding MLLE, the spread of the data in the embedding space is completely different: most of the data points have sparse coordinates. Furthermore, from a visual point of view, MLLE segregates well the two classes, and one is able to easily separate them only using the two first components.

D. Sensitivity to the number of data points

A second set of experiments was run to evaluate the sensitivity of these algorithms with respect to the number of data point M. It is well known that the performances of machine learning algorithms decrease as the M decreases. Moreover one should expect that this statement is even more valid for LLE and MLLE. This expectation comes from the fact that these methods are first using k-NN algorithm to construct the neighborhood, and unfortunately it is very sensitive to the number of data points. In other words, using fewer data points implies a less accurate neighborhood identification. Let's see if these expectations are confirmed by experimental results.

To do so, the series of experiments conducted on the whole data set was repeated on half of the data set. To have a true estimation of the performances on half of the data set, the experiments are re-conducted on four different data sets of size $\frac{M}{2}$. Then the mean is taken as the final value for each metric. In addition to this, the standard deviation is computed to have a notion of confidence.

The reconstruction error as well as the residual variance are plotted on figure 3. The shapes of the curves are similar to the one seen on the whole data set, however one can remark that reconstruction error curves are shifted towards the left. The steep slope is now located around $k = 25$ for MLLE and around $k = 15$ for LLE. This shift can be explained by the fact that the relevant neighborhood of each data point is smaller than the one on the whole data set, as the number of data points is less important.

To find the best hyperparameter, the process described in section III-C was followed. According to the acquired data the potential candidates for k_{LLE}^{best} and k_{MLLE}^{best} are:

$$C_{cand}^{LLE} = \{34, 103, 200\} \qquad C_{cand}^{MLLE} = \{16, 50, 131\}$$

The spread of the data in the embedding space for these values can be visualized in the figure 2. The graphs presented are actually very similar to the one obtained on the whole data set. However one can remark that the distributions of the data points between the two classes in the embedding space found by LLE are not as well segregated as in the case on the whole data set. This observation is still true for MLLE but not as blatant as it is for LLE. Hence, LLE seems to be more sensitive to the number of data points.

A recap of the performances of the experiments conducted so far is provided in table II. To give more relief to these results, some information was added about the ability to extract meaningful data. This information is the F1-measure of a K-

NN classifier[10] where the data set was the data points projected onto the embedding space. The F1-measure was computed using cross-K-validation with $K = 5$.

	Full data set		Semi data set	
Algorithms	LLE	MLLE	LLE	MLLE
k_{best}	140	91	103	50
Residual Variance	0.27	0.43	0.31	0.39
Reconstruction error	2.31e-6	0.65	6.25e-6	0.40
KNN mean F1 measure	0.87	0.76	0.85	0.76

TABLE II: Results recap

As one can remark, contrary to the visual impression one can get from the pair plots, the KNN classifier gives better results on the embedding space found by LLE than the one found by MLLE. However this information shall be taken with a grain of salt, since this metric depends a lot on the classifier used here, far different results can be obtained with other classifiers. Finally regarding the sensitivity to the number of data points, most of the metrics show that MLLE is clearly less sensitive than LLE.

E. Time performances

The wall clock time is plotted with respect to the number of neighbors. When k is high one can remark that the curve is tending to have a cubic shape. According to section II-C it corresponds to the term k^3M. When k is small the cubic shape is not anymore relevant, this typically true for LLE on the whole data set. This comes from the fact that for low values of k the term $O(k^3M)$ is dominated by the term $O(nlog(k)log(M)M + dM^2)$. Therefore the cubic shape is replaced by a logarithm shape that is clearly visible for LLE.

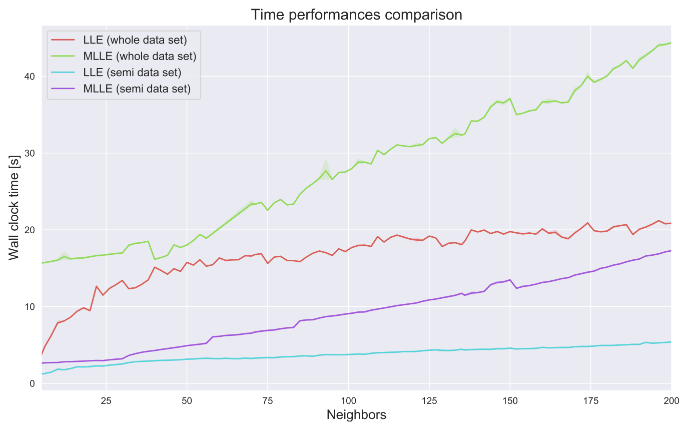

Fig. 5: Evolution of the time complexity with respect to k.

Regarding MLLE the reasoning for small values of k is not valid since there is an additional cost $O((k-d)k^2M)$, hence on $O(nlog(k)log(M)M + dM^2)$ does not dominate anymore the cubic term. This statement is true only if the input dimension $n \ll M$ that is not necessarily the case [11]. Finally one can also remark that, as the theory predicts it, if the number of data points is multiplied by two then the time complexity is multiplied by four.

IV. Discussion

This section tackles the further experiments that can be conducted to deeply evaluate the two algorithms. To fairly assess these two methods, needless to say several diverse data sets are needed: one should for instance select data sets with a higher number of data points and a higher number of features n. Furthermore one could use more metrics than the two used here, many examples of such metrics are tackled in these papers [4] & [5]. Finally one could also study the influence of the distance norm used in equation (4). This paper was limited to the euclidean distance, however use different distances such as the L_p norm could be an interesting point.

V. Conclusion

Given the theory of the results, the strengths and the weaknesses of the two algorithms can be well-identified. Theoretically, MLLE can be considered as an enhancement of LLE since it has the exact same base but uses several weights instead of only one. The experiments show that MLLE can indeed retrieve the neighborhood with less number of neighbors k and is also less sensitive to the number of data points. However if k is too high then MLLE performances start crashing while LLE ones are maintained.

According to section II-C, the computational cost is roughly the same for both algorithms if $O(nlog(k)log(M)M + dM^2)$ dominates $O(k^3M)$. If it is not the case then the computational cost of MLLE is almost twice higher than LLE one. This last point was confirmed by the experiments conducted.

To conclude, MLLE is providing in general better performances than LLE, however the hyperparameter must be chosen carefully for MLLE to avoid any issues. Therefore, these higher performances come at some costs:

- Lower granularity for the grid search to well determine k.
- Additional computational cost that is significant when $O(k^3M)$ dominates $O(nlog(k)log(M)M + dM^2)$.

References

[1] S. Roweis and L Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290: 2323–2326, 2000.
[2] Zhang, Zhenyue Wang, Jing. MLLE: Modified Locally Linear Embedding Using Multiple Weights. *Adv Neural Inf Process Syst*. 19. 1593-1600. 2006.
[3] Benyamin Ghojogh and Ali Ghodsi and Fakhri Karray and Mark Crowley. Locally Linear Embedding and its Variants: Tutorial and Survey. 2011.10925. 2020.
[4] Kouropteva, Olga Okun, Oleg Pietikäinen, Matti. Selection of the Optimal Parameter value for the Locally Linear Embedding Algorithm. FSKD. 359-363. 2002
[5] Valencia-Aguirre J., Álvarez-Mesa A., Daza-Santacoloma G., Castellanos-Domínguez G. Automatic Choice of the Number of Nearest Neighbors in Locally Linear Embedding. In: Bayro-Corrochano E., Eklundh JO. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2009

[10]The K-NN classifier was used with 10 neighbors as parameter
[11]One can typically think about images.