# Exploration of Manifold Learning methods for face dataset

**Qichao Li**
Department of MAE
HKUST
qlich@connect.ust.hk

**Haohan Huang**
Department of Mathematics
HKUST
hhuangbu@connect.ust.hk

**Qiqi Zhou**
Department of CSE
HKUST
qzhouam@connect.ust.hk

**Jin Wu**
Department of ECE
HKUST
jwucp@connect.ust.hk

## 1 Introduction

Manifold learning is a family of unsupervised machine learning techniques that aim to uncover the low-dimensional structure hidden within high-dimensional data. The central idea is that real-world high-dimensional data often lies on or near a lower-dimensional manifold, which is a smooth and continuous surface embedded in the high-dimensional space. Manifold learning algorithms seek to find this underlying manifold and represent the data in the reduced dimensions while preserving its intrinsic structure or relationships.

These techniques are particularly useful for tasks such as dimensionality reduction, data visualization, and noise reduction. By discovering the low-dimensional representation of the data, manifold learning can help reveal meaningful patterns, simplify complex data, and improve the performance of various machine learning tasks. Some well-known manifold learning methods include: t-Distributed Stochastic Neighbor Embedding (t-SNE), ISOMAP, Locally Linear Embedding (LLE), Diffusion Maps, Local Tangent Space Alignment (LTSA), and Multidimensional Scaling (MDS). In this project, we implement these methods to the classical face images dataset, trying to find some characteristics of these images and compare their performance on feature extraction of these certain high-dimensional data.

## 2 Dataset

We select the classic face dataset, which contains 33 faces of the same person ($Y \in \mathbb{R}^{112\times92\times33}$) in different angles, available on the website

`https://github.com/yao-lab/yao-lab.github.io/raw/master/data/face.mat`.

These images, shown in Fig. 1, are characterized by different aspects of the faces, varying a little bit gradually from right to left. We implement several manifold learning methods on this dataset to compare their performance on detecting this little difference between each face image from the same person. It is worth noting that we observed an incorrect ordering of the face images in the raw data. Fig. 1 displays the corrected order after manual adjustment, which we consider as the ground truth for subsequent evaluations.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Figure 1: All the tested face images.

## 3 Methodology

In this project, a variety of manifold learning methods are implemented to find the key characteristics behind a set of face images from one woman. Details of each ways are shown below.

### 3.1 Diffusion Maps

Diffusion Maps is a powerful non-linear dimensionality reduction method that extracts low-dimensional non-linear structures from high-dimensional data. It is based on a distance metric between the data points and diffuses the similarity between them to map high-dimensional data to a low-dimensional space.

The key advantage of Diffusion Maps is its ability to handle non-linear structures that other dimensionality reduction methods cannot. Additionally, it is sensitive to the local structures of the data, making it ideal for preserving the local features of the data, especially for data with local structures. Diffusion Maps can efficiently handle large-scale data through matrix decomposition.

### 3.2 MDS

MIDS (Minimum Inner-Distance Separation) is often used for analyzing high-dimensional data.

The optimization problem for MIDS can be written as:

$$\min_{\mathbf{C}} \sum_{i=1}^{n} \sum_{j=1}^{n} ||\mathbf{x}_i - \mathbf{x}_j||_2^2 c_{ij} \tag{1}$$

subject to:

$$\sum_{j=1}^{n} c_{ij} = 1 \quad \text{for all } i$$

$$c_{ij} \in \{0, 1\} \quad \text{for all } i, j$$

$$\sum_{i=1}^{n} c_{ij} = 1 \quad \text{for all } j$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ are data points in the dataset, $c_{ij}$ is the binary variable indicating whether or not $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster, and $n$ is the number of data points.

The MIDS algorithm works as follows:

1. Initialize the matrix $\mathbf{C}$ with random binary values.
2. Calculate the pairwise distances between all data points.

3. For each data point, assign it to the cluster with the closest point.
4. Update the $\mathbf{C}$ matrix by setting $c_{ij} = 1$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ are in the same cluster, and $c_{ij} = 0$ otherwise.
5. Repeat steps 2-4 until convergence.

### 3.3 ISOMAP

Given a set of data points $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathbb{R}^D$, the goal of ISOMAP is to find a lower-dimensional representation $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$, where $\mathbf{y}_i \in \mathbb{R}^d$ and $d < D$, such that the geodesic distance between any two points is preserved as much as possible.

The geodesic distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ on a manifold is defined as the shortest path between them on the manifold. Let $d_{ij}$ be the geodesic distance between $\mathbf{x}_i$ and $\mathbf{x}_j$, and let $\|\cdot\|$ denote the Euclidean norm. The goal of ISOMAP is to find the lower-dimensional representation $\mathbf{Y}$ that minimizes the following objective function:

$$\sum_{i,j=1}^{N} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2, \tag{2}$$

subject to the constraint that the geodesic distance $d_{ij}$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ is approximated by the Euclidean distance $\|\mathbf{y}_i - \mathbf{y}_j\|_2$, where $w_{ij}$ is a weight that depends on the adjacency of the data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the original high-dimensional space.

### 3.4 LLE/MLLE

Locally Linear Embedding (LLE) and its extension, the Modified Locally Linear Embedding (MLLE), are based on the idea that a low-dimensional manifold can be approximated by a set of linear relationships between neighboring points in high-dimensional space.

Let $X$ be an $n \times d$ matrix, where each row corresponds to a $d$-dimensional data point. We can represent the manifold using a weight matrix $W$, where $w_{ij}$ represents the weight of the $j$-th nearest neighbor of point $i$. The reconstruction error for a point $x_i$ is given by:

$$\epsilon_i = \|x_i - \sum_{j=1}^{n} w_{ij} x_j\|^2 \tag{3}$$

The goal of LLE is to find a lower-dimensional representation $Y$ of $X$ such that the reconstruction error is minimized. This is achieved by solving the following optimization problem:

$$\min_{Y} \sum_{i=1}^{n} \epsilon_i \text{ s.t. } YY^T = I \tag{4}$$

where $I$ is the identity matrix and $Y$ is an $n \times k$ matrix, where $k$ is the desired dimensionality of the lower-dimensional representation.

MLLE extends LLE by introducing a penalty term that encourages the preservation of local geometry. The optimization problem becomes:

$$\min_{Y} \sum_{i=1}^{n} \epsilon_i + \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \|y_i - y_j\|^2 \text{ s.t. } YY^T = I \tag{5}$$

where $\lambda$ is a regularization parameter.

### 3.5 LSTA

Local Tangent Space Alignment (LSTA) is based on the concept of tangent spaces, which are local linear approximations of the underlying manifold at each data point.

The key idea behind LSTA is to align the tangent spaces of neighboring data points in the high-dimensional space and map them to a common low-dimensional space. Specifically, LSTA first estimates the tangent space at each data point by computing the principal components of the neighboring data points. It then aligns the tangent spaces of neighboring data points using a rotation matrix and maps them to a common low-dimensional space using classical multidimensional scaling (MDS).

The objective function for LSTA can be written as:

$$\min_{Y} \sum_{i<j} (d(Y)_{ij} - d(X)_{ij})^2 \tag{6}$$

where $d(Y)_{ij}$ is the Euclidean distance between the $i$th and $j$th data points in the low-dimensional space, and $d(X)_{ij}$ is the distance between the tangent spaces of the $i$th and $j$th data points in the high-dimensional space.

The rotation matrix is computed by minimizing the difference between the tangent spaces of neighboring data points in the high-dimensional space. This is achieved by minimizing the following cost function:

$$\min_{V} \sum_{i<j} (V_{ij} - V_{ji})^2 \tag{7}$$

where $V_{ij}$ is the rotation matrix that aligns the tangent spaces of the $i$th and $j$th data points in the high-dimensional space.

LSTA can handle non-linear structures and is robust to noise and outliers. It can also preserve the local geometry of the data and can uncover hidden relationships and patterns that are difficult to see in the original high-dimensional space. However, LSTA can be computationally expensive, as it requires computing the tangent spaces and the rotation matrix for all pairs of neighboring data points.

In summary, LSTA is a non-linear dimensionality reduction technique that aligns the tangent spaces of neighboring data points in the high-dimensional space and maps them to a common low-dimensional space. It is useful for analyzing complex datasets with non-linear structures and can provide insights into the underlying geometry of the data.

### 3.6 2D t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) was introduced by van der Maaten and Hinton in 2008. The primary objective of t-SNE is to represent a high-dimensional dataset in a lower-dimensional space while preserving the similarity between data points.

Given a high-dimensional dataset $X = \{x_1, x_2, \ldots, x_n\}$ with $x_i \in \mathbb{R}^D$, t-SNE aims to find a low-dimensional representation $Y = \{y_1, y_2, \ldots, y_n\}$ with $y_i \in \mathbb{R}^d$ ($d \ll D$) that best represents the pairwise similarities $p_{ij}$ in the high-dimensional space. The steps of the t-SNE algorithm are as follows:

1. Compute pairwise similarities $p_{ij}$ in the high-dimensional space using the Gaussian kernel:

$$p_{j|i} = \frac{\exp\left(-||x_i - x_j||^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-||x_i - x_k||^2 / 2\sigma_i^2\right)}, \tag{8}$$

   and then symmetrize the conditional probabilities to obtain the joint probabilities:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{9}$$

2. Initialize the low-dimensional data points $Y^{(0)} = \{y_1^{(0)}, y_2^{(0)}, \ldots, y_n^{(0)}\}$ randomly.
3. Compute pairwise similarities $q_{ij}$ in the low-dimensional space using the Student's t-distribution with one degree of freedom (i.e., Cauchy distribution):

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}}. \tag{10}$$

4. Minimize the Kullback-Leibler divergence between $P$ and $Q$ using gradient descent:

$$C(Y) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{11}$$

with the gradient:

$$\frac{\delta C}{\delta y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1}. \tag{12}$$

5. Repeat steps 3 and 4 until convergence.

# 4 Experiments

## 4.1 Experimental settings

We use a reduction space dimension of $d = 2$ for all methods. To obtain orders from the 2-dimensional space, we explored two approaches in our experiments. The first approach involved following the order of one of the $d$ components (selected based on observations from the visualization results). The second approach consisted of determining a starting point (also based on visualization observations) in the $d$-dimensional space and sequentially appending the nearest next point to the order sequence until a complete order was obtained.

In addition to the visualization results, we also utilized the concept of Inversion from discrete mathematics [1] as a numeric metric to evaluate the quality of the inferred order using different reduction methods and ordering approaches. Inversion is a smaller-is-better metric, with zero indicating a perfect order.

For reduction methods, we use the implementation from *sklearn* [2] package. And in the experiment of Diffusion Maps, we did some post-processing to the embedding, which can be referred to in our code.

## 4.2 Visualization Results

### 4.2.1 Diffusion Maps

We first test the performance of the diffusion map. The embedding results and ordered results are shown in Figure 2, Figure 3a, and Figure 3b.

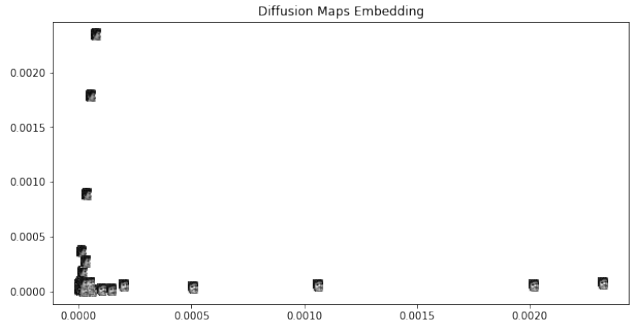


Figure 2: Distribution of the face images based on Diffusion Maps embedding.


(a) Face images sorted by Diffusion Maps one component ordering.

(b) Face images sorted by Diffusion Maps nearest neighbor ordering.

Figure 3: Comparison of face images sorted by Diffusion Maps.

### 4.2.2 MDS

Then, the MDS-embedding is utilized to reduce the data dimension of the 33 face images to a lower one. The distribution of this set of figures based on the first two eigenvectors can be shown in Fig. 4, where we can find that the orientation of the face in each image gradually transitions from looking towards the right to facing forward with the increase of the first eigenvector, while nothing change can be found based on the variation of the second eigenvector.

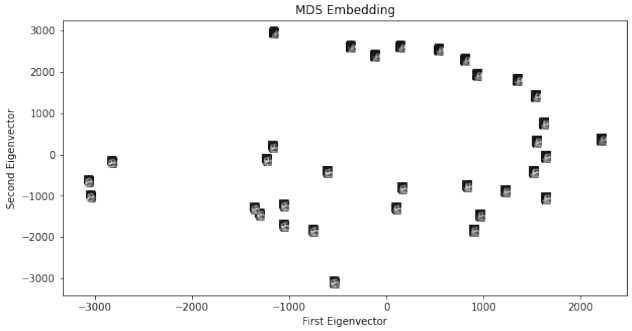


Figure 4: Distribution of the face images based on MDS-embedding.

Therefore, we sort the 33 images according to the top first eigenvector in Fig. 5, intended to represent the faces in a sequence that preserves the pairwise distances between the faces in the high-dimensional space. As can be seen in Fig. 5, the faces generally transition from one angle to another smoothly, while some discontinuities in the ordering can still be observed. This might be due to the limitations of the MDS method in preserving the exact pairwise distances in the lower-dimensional space.


(a) Face images sorted by MDS one component ordering.

(b) Face images sorted by MDS nearest neighbor ordering.

Figure 5: Comparison of face images sorted by MDS.

### 4.2.3 ISOMAP

In this experiment, the ISOMAP-embedding of these 33 faces on the $k = 5$ nearest neighbor graph is tested using the sklearn Python library, to compare against the MDS results. Fig. 6 is a scatter plot showing the distribution of the faces in the 2D space formed by the top two eigenvectors, where similar but inverse pattern related to the vary of the orientation of faces can be found compared to MDS.

The second visualization, Fig. 7, shows the 33 faces sorted based on the top first eigenvector from the ISOMAP embedding. Similar to MDS, the order is intended to represent the faces in a sequence that preserves the geodesic distances between the faces in the high-dimensional space. By comparing the ISOMAP results with the MDS results, it can be observed that the ISOMAP method provides a smoother transition between the face angles, which suggests that the ISOMAP technique captures the underlying nonlinear structure of the data more effectively than MDS.

### 4.2.4 LLE/MLLE

Since ISOMAP-embedding has already achieved acceptable results to capture the underlying nonlinear structure of the data according to previous experiments, we consider it as the baseline to check whether other methods, like LLE/MLLE in this test, can achieve good performance to find the

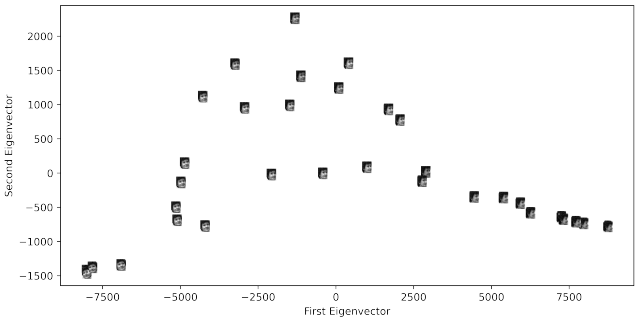


Figure 6: Distribution of the face images based on top two eigenvectors with ISOMAP-embedding.


(a) Face images sorted by ISOMAP one component ordering.

(b) Face images sorted by ISOMAP nearest neighbor ordering.

Figure 7: Comparison of face images sorted by MDS.

underlying structures behind the high-dimension data. We explore the LLE/MLLE-embedding in this test on the $k = 5$ nearest neighbor graph and compare it against the ISOMAP. The distribution of the face images based on the top two eigenvectors from LLE/MLLE-embedding can be seen in Fig. 8. Similar to ISOMAP, the transition of the face angles is mainly captured and represented by the top first eigenvector. The main difference lies in the finding that the second eigenvector from LLE/MLLE also reflects this variation to a certain extent, while no meaningful information can be found from that of ISOMAP.

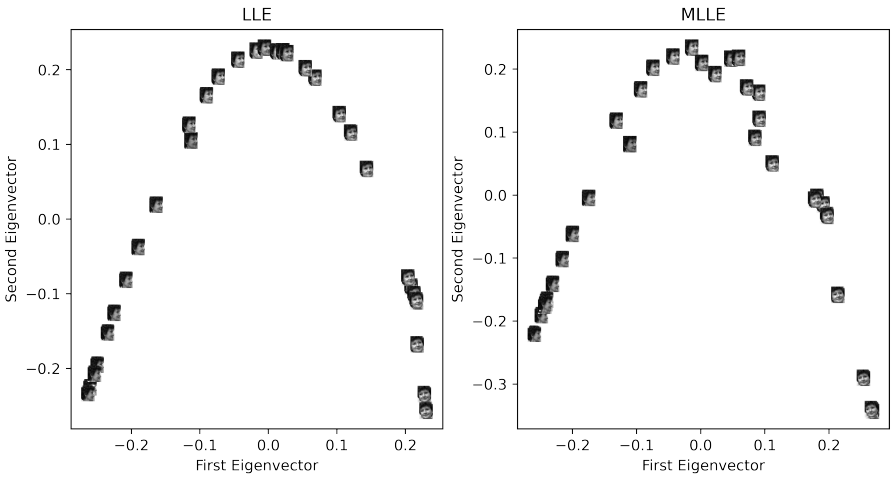


Figure 8: Distribution of the face images based on top two eigenvectors with LLE(left)/MLLE(right)-embedding.

#### 4.2.5 LSTA

We next explore the LSTA-embedding of the 33 faces on the $k = 5$ nearest neighbor graph, with the distribution shown in Fig. 9. The results show that, with LSTA-embedding, the structure behind the face images dataset can be clearly captured within the top first eigenvector.

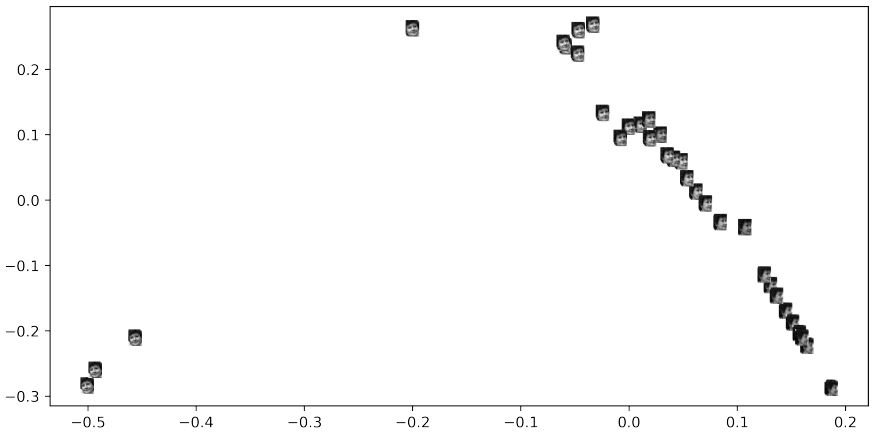


Figure 9: Distribution of the face images based on top two eigenvectors with LTSA-embedding.

#### 4.2.6 2D t-SNE

The last manifold learning method we test to uncover the low-dimensional structure hidden within high-dimensional face images dataset is t-SNE. As can be seen in Fig. 10, where the distribution of face images is illustrated using the top two eigenvectors, the faces transition from one angle to another smoothly. Therefore, the t-SNE embedding can also be considered as a good manifold learning method to capture the structure behind this dataset.

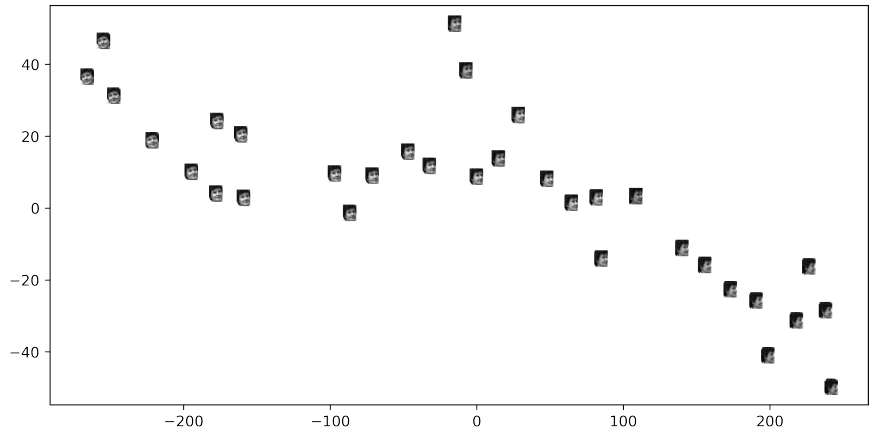


Figure 10: Distribution of the face images based on top two eigenvectors with t-SNE-embedding.

**Discussion** In Sections 4.2.4, 4.2.5, and 4.2.6(cherry-picked), we omitted the results of the grid of faces to show the order, because we observed that all three methods detected clear structure in

the human face dataset from the embedding visualization. And later, the quantitative evaluation will confirm that we can infer near-perfect order from the reduction results.

### 4.3 Quantitative Evaluation

To quantitatively compare the performance of these manifold learning methods to uncover the low-dimension structures behind this face images dataset, we provide inversion value $\text{Iv}_x$ of one-component ordering method and $\text{Iv}_n$ of nearest-neighbor ordering method.

Table 1: $\text{Iv}_x$ and $\text{Iv}_n$ evaluation for different methods

| | Diffusion Maps | MDS | ISOMAP | LLE | MLLE | LSTA | t-SNE |
|---|---|---|---|---|---|---|---|
| $\text{Iv}_x$ | 135 | $28.7 \pm 17.0$ | 6 | 3 | 3 | 1 | $62.8 \pm 50.7$ |
| $\text{Iv}_n$ | 46 | $52.1 \pm 58.9$ | 7 | 2 | 0 | 12 | $49.6 \pm 49.4$ |

Because of the randomness in MDS and t-SNE, their result is unstable in different running. We report 10 times avg $\pm$ std for them in Table 1. And the results in the visualization section are cherry-picked results.

In Table 1, we can observe that the result stays consistent with our visualization observation and gives a more accurate comparison between different methods. The concrete conclusion is declared in Section 5.

**Supplementary data:** For unstable methods MDS and t-SNE, we increase the times of running and also provide box plots for the inversion of the orders to show their performance better.

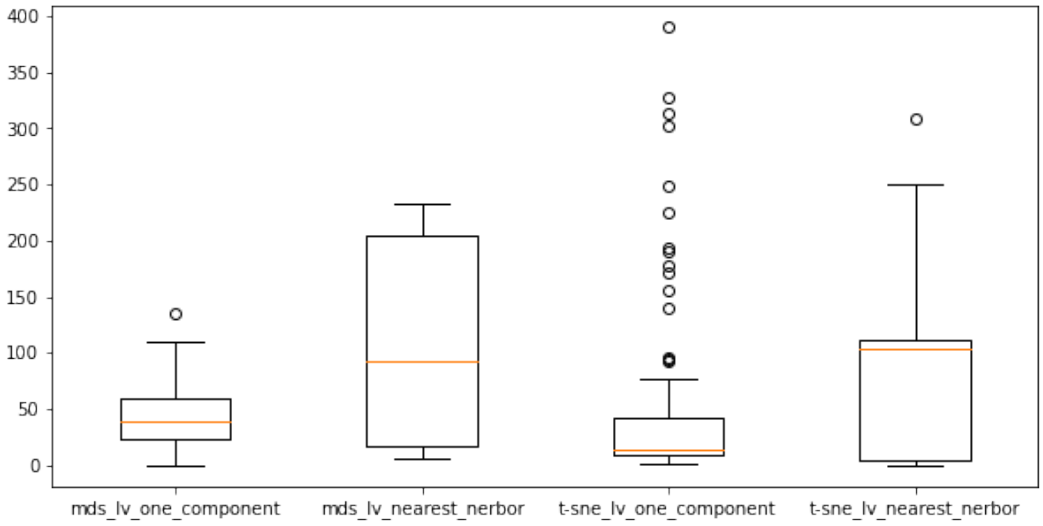


Figure 11: Box plots of 100 times runs for unstable methods.

## 5 Conclusion

In this mini-project, we investigate the performance of different kinds of manifold learning methods to uncover the low-dimensional structure hidden within high-dimensional data using a small dataset characterized by the variation of the face orientation. Our experiments show that ISOMAP, LLE, MLLE, and LSTA all show near-perfect performance, but in this face image dataset, MLLE with nearest neighbor ordering performs best, which is perfect. Cherry-picked MDS or t-SNE also can give a near-perfect result, but they work unstably.

## Contribution

- Haohan Huang wrote the methodology part of the paper.

- Qichao Li wrote most of the other parts of the paper.
- Qiqi Zhou wrote the Quantitative Evaluation part of this paper and did some revisions on the Experiment and Conclusion part. Qiqi implemented LLE/MLLE, LSTA, and t-SNE.
- Jin Wu implemented Diffusion Maps, MDS, and ISOMAP.
- PPT and video presentation are completed under the collaboration of Qiqi, Qichao, and Haohan.

## References


[1] Inversion (discrete mathematics). Wikipedia, 2023. https://en.wikipedia.org/wiki/Inversion_(discrete_mathematics).

[2] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. scikit-learn: Machine learning in python, 2011–. https://scikit-learn.org.