Order the Faces by Manifold Learning

Zhenghui CHEN
Department of Chemical and Biological Engineering
The Hong Kong University of Science and Technology
zchenef@connect.ust.hk

Abstract

In this project, we intended to sort 33 different face images of same person from left face direction to right face direction. We applied different manifold learning methods and utilized python machine learning package scikit-learn [1] to achieve this task. We found that MDS and TSNE perform poorly by the order of first eigenvector. Diffusion map, ISOMAP, LLE, LSTA perform well according to the first eigenvector. However, all these six methods can be well interpreted in their 2D embedding graph. We also explored the model performance affected by different number of nearest neighbours. It turns out that different model has its diverse best neighbour number.

1 Introduction

Face pose estimation is an interesting research topic. It helps us to infer the person's attention and detect related behaviour. In this report, we try to order the face orientation of 33 images from the same person by using different manifold learning methods including Diffusion map, MDS, ISOMAP, LLE, LSTA and TSNE.
We obtained the embedding results from different methods and ordered the face based on first eigenvector size. We also plotted the scatter graph of two eigenvectors, visualized the face images and explored the relation between the face orientation and data distribution. Then we compared the results of different manifold learning methods. Finally, we would like to find the best nearest neighbour number for ISOMAP, LLE, LSTA.
The following sections are organized as below. We will introduce the dataset and show ground truth rank of face image in section 2. We will briefly describe the six manifold learning methods in Section 3. We will analyse the rank results of first eigenvector and 2D embedding graph in Section 4. We will draw a conclusion in Section 5.

2 Dataset

The face dataset contains 33 faces of the same person in different angles, which is represented by the data matrix $Y \in R^{112\times92\times33}$. The dataset can be downloaded from the following website.
https://github.com/yao-lab/yao-lab.github.io/blob/master/data/face.mat
We used Python to read the dataset including the grayscale image and its corresponding id. However, we found that the id didn't match with the face direction. Therefore, we adjusted the order of the image from the left to the right and labelled with the rank number.

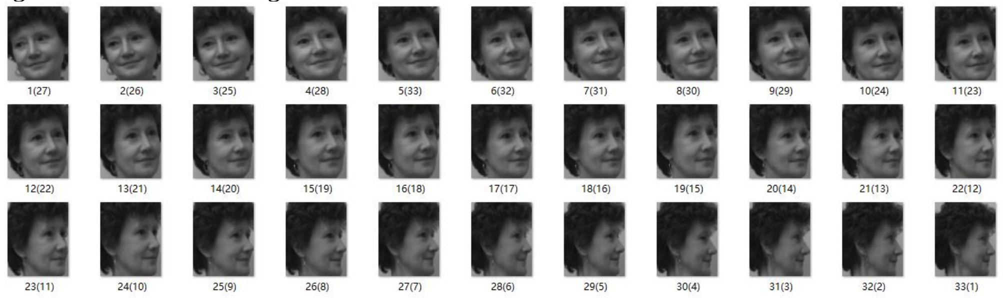


Figure 1: The ordered face image from left to right

Figure 1 shows the dataset with face direction in order. Table 1 shows the rank of face image from left to right, which serves as the ground truth label in this experiment.

Table 1: The rank label of face image

image	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
rank	9	13	19	32	6	18	28	7	17	1	5	16	12	10	4	21	22

image	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
rank	26	33	11	2	24	3	27	29	23	14	30	31	20	15	25	8

3 Methods

3.1 Diffusion map

Diffusion map [2] discovers the potential manifold structure of the dataset. It has low computation cost and is robust to noise.
Given a high dimensional dataset $\{x_i : i = 1, \dots, n\}$. Define a Gaussian kernel $k(x, y)$, the kernel function has symmetric properties.

$$k(x, y) = \exp\,(-\frac{||x - y||^2}{\epsilon})$$

Then create the diffusion matrix $L_{i,j} = k(x_i, x_j)$ and form the normalized matrix $M = D^{-1}\mathrm{L}$, where D is a diagonal matrix and $D_{i,i} = \sum_j L_{i,j}$. Finally, we compute the eigenvalues of M and the corresponding eigenvector to get the embedding.

3.2 MDS

Multidimensional scaling (MDS) [3] is a means of visualizing the level of similarity of individual cases of a dataset. It uses the pairwise dissimilarities of Euclidean distance to construct a low-dimensional space.
Given a set of data $x_1, x_2, \dots, x_n \in \mathbb{R}$, the distance between i^{th} and j^{th} data point is defined as $d_{i,j}$. The distance matrix D is composed of these distances.

$$D = \begin{pmatrix} d_{1,1} & \cdots & d_{1,n} \\ \vdots & \ddots & \vdots \\ d_{n,1} & \cdots & d_{n,n} \end{pmatrix}$$

The squared proximity matrix $D^{(2)} = d_{i,j}^2$. Then calculate $B = -\frac{1}{2}JD^{(2)}J^T$, where J is the centering matrix. The eigenvalues and eigenvectors of B can be computed from $B = U\Lambda U^T$.

$$\Lambda = diag(\lambda_1, \dots, \lambda_n) \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$$

We determine the m largest eigenvalues and corresponding eigenvectors.

$$U_m = [u_1, \dots, u_m] \;\; \Lambda_m = diag(\lambda_1, \dots, \lambda_m) \;\; \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$$

Finally, we form the coordinate matrix Y_m by following equations.

$$Y_m = U_m \Lambda_m^{\frac{1}{2}}$$

3.3 ISOMAP

ISOMAP is an extension of MDS [4], which incorporates the geodesic distances imposed by a weighted graph. It defines the geodesic distance to be the sum of edge weights along the shortest path between two nodes.
Given a set of data $x_1, x_2, \dots, x_n \in \mathbb{R}$, we firstly determine the neighbours of each data point and construct a neighborhood graph $G = (V, E)$, where $V = \{x_i : i = 1, \dots, n\}$, $E = \{(i, j)\}$ if j is a neighbor of i.
Then we compute the shortest path $d_{i,j} = dist(x_i, x_j)$ between node x_i and node x_j by Dijkstra's algorithm. Finally, we repeat the MDS steps and use $dist(x_i, x_j)$ as the input of MDS to get the embedding results.

3.4 LLE

Locally linear Embedding (LLE) [5] focuses on preserving the local linearity of the sample when reduce data dimensionality.
Given graph $G = (V, E)$, the first step is to compute the distance from x_i to x_j and find the k nearest neighbours of x_i by KNN.

$$N_i = KNN(x_i, k), N_i = [x_{1i}, \dots, x_{ki}]$$

For each x_i and its neighbours $x_j \in \mathcal{N}_i$, solve the reconstruction weights matrix w.

$$\min \varepsilon(w) = \sum_{i=1}^{n} \left\| x_i - \sum\nolimits_{j=1}^{k} w_{ij} x_j \right\|^2 \qquad \sum_{j=1}^{k} w_{ij} = 1$$

We create sparse matrix $M = (I - W)^T (I - W)$ where

$$W_{ij} = \begin{cases} w_{ij}, & j \in \mathcal{N}_i \\ 0, & else \end{cases}$$

We find bottom $d + 1$ nonzero eigenvalues from λ_{n-1} to λ_{n-d} and their corresponding eigenvectors of M because we discard the bottom eigenvector with eigenvalue zero.

$$\min \varepsilon(Y) = \sum_{i=1}^{n} \left\| y_i - \sum\nolimits_{j=1}^{k} w_{ij} x_j \right\|^2 = tr(YMY^T)$$

The final embedding coordinate is defined as $Y_d = U_d \Lambda_d^{\frac{1}{2}}$ where

$$U_d = [u_{n-d}, \dots, u_{n-1}] \quad \Lambda_d = diag(\lambda_{n-d}, \dots, \lambda_{n-1}) \quad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1} > \lambda_n = 0$$

3.5 LSTA

Local Tangent Space Alignment (LTSA) is a modified version of LLE. It is based on the intuition that all tangent hyperplanes to the manifold will be aligned when a manifold is correctly unfolded. The basic algorithm is illustrated as follows.
1. Given a set of data $x_1, x_2, \dots, x_n \in \mathbb{R}$, compute the k-nearest neighbours of x_i and do local SVD in neighbourhood of x_i.

$$\tilde{X}^{(i)} = [x_{i_1} - \mu_i, \dots, x_{i_k} - \mu_i]^{p \times k} \quad x_{i_j} \in \mathcal{N}(x_i) \quad \mu_i = \sum_{j=1}^{k} x_{i_j}$$

$$\tilde{X}^{(i)} = \tilde{U}^{(i)} \tilde{\Sigma} (\tilde{V}^{(i)})^T$$

2. Optimize to find an embedding that aligns the tangent space.

$$K^{n \times n} = \sum_{i=1}^{n} S_i W_i W_i^T S_i^T \qquad W_i^{k \times k} = I - G_i G_i^T$$

$$[x_1, \dots, x_n] S_i^{n \times k} = \left[x_{i_1}, \dots, x_{i_k}\right] \qquad G_i = [\frac{1}{\sqrt{k}}, \tilde{V}_1^{(i)}, \dots, \tilde{V}_d^{(i)}]^{k \times (d+1)}$$

3. Do eigenvalue decomposition $K = U \Lambda U^T$. Find smallest $d + 1$ eigenvectors and drop the smallest one. Obtain d dimensional embedding of data points from the remaining d eigenvectors.

3.6 TSNE

T-distributed Stochastic Neighbour Embedding (TSNE) [6] is derived from SNE algorithm and treat the coordinates in the lower dimension as the t-distribution. It increases the distance between the clusters with large distances and solves the crowding problem.

4 Results and Analysis

4.1 Rank results

We applied different manifold learning methods to sort the direction of face image from left to right. The results are shown in Table 2. The row of Rank from 1 to 33 represents the trend of face orientation from left to right. The row of Image represents the ground truth label of image corresponding to the rank. The remaining rows represents the prediction outcome of different manifold learning methods.

Table 2: The rank results of different methods

Rank	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Image	10	21	23	15	11	5	8	33	1	14	20	13	2	27	31	12	9
Diffusion Map	10	21	23	15	11	5	8	1	33	14	20	13	2	27	31	12	9
MDS	10	23	21	19	4	20	14	29	28	11	5	33	25	13	8	7	1
ISOMAP	10	21	23	15	5	11	1	33	14	8	20	13	2	27	31	12	9
LLE	10	21	23	15	5	11	8	1	33	14	20	13	2	27	31	12	9
LSTA	10	21	23	15	11	5	33	8	1	14	20	13	2	31	27	12	9
TSNE	3	8	6	30	13	16	29	2	20	21	7	9	5	18	1	22	19

Rank	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Image	6	3	30	16	17	26	22	32	18	24	7	25	28	29	4	19
Diffusion Map	6	3	30	16	17	22	26	32	18	24	7	25	29	28	4	19
MDS	24	27	18	2	15	32	9	26	17	12	6	31	16	30	22	3
ISOMAP	6	3	30	16	17	26	22	32	18	24	7	25	28	29	4	19
LLE	6	3	30	16	17	22	26	32	18	24	7	25	28	29	4	19
LSTA	6	3	30	16	17	26	22	32	18	24	7	25	28	29	4	19
TSNE	26	33	14	15	17	27	23	32	11	10	31	28	12	4	25	24

4.2 Evaluation metrics

In order to better evaluate the ranking performance of different methods, we proposed the metrics to calculate the difference of rank position for the same image. The results of this absolute error are shown in Table 3.

Table 3: The absolute error of rank position

Methods	Diffusion Map	MDS	ISOMAP	LLE	LSTA	TSNE
Absolute Error	6	308	8	6	4	204

From the absolute error result, we can see that MDS and TSNE method have much larger error. This indicates that these two methods don't work well in this task. Other methods like Diffusion Map, ISOMAP, LLE and LSTA have very small error. The LSTA method achieves the best performance with the absolute error of 4.

4.3 Visualization results

4.3.1 Diffusion map

We implemented the diffusion map with default settings and visualized the embedding result on the first two eigenvectors. Figure 2 shows the embedding graph with face image. We can observe that face order changes with the increase of the first eigenvector. The scatter plot is a V shape and the face orientation is regular as the head gradually turn right from left. The boundary between left face and right face is in the middle part of the graph. Diffusion map has a good performance on this face order task.

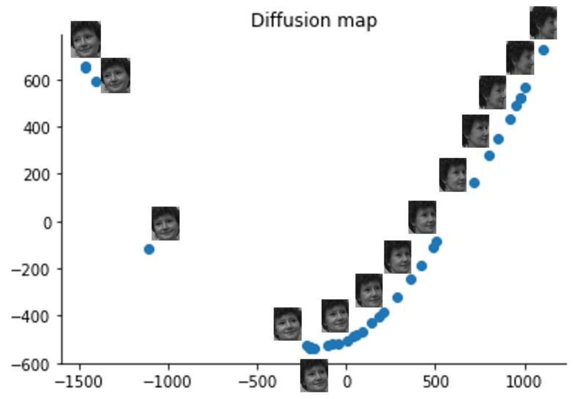


Figure 2: Embedding graph by Diffusion map

4.3.2 MDS

Figure 3 shows the face order according to the first eigenvector by MDS. We can see that the face image is in chaos. The face of left orientation and the face of right orientation are mixed. This suggests that MDS performs badly in the first eigenvector because it calculates the points similarity by euclidean distance.

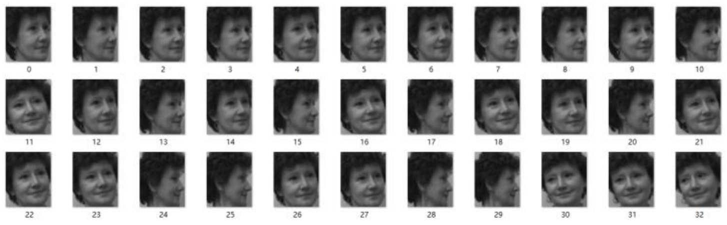


Figure 3: Order the faces by MDS

Figure 4 shows the 2D embedding results by MDS. We can see that face images change the angle from right to left. It is quite easy to find the pattern between the region and face order on this 2D graph. The face orientation of images in green region is towards right. The face orientation of images in yellow region is close to middle. The face orientation of images in blue region is towards left. This suggests that the two eigenvectors of MDS can better help us to interpret the face order.

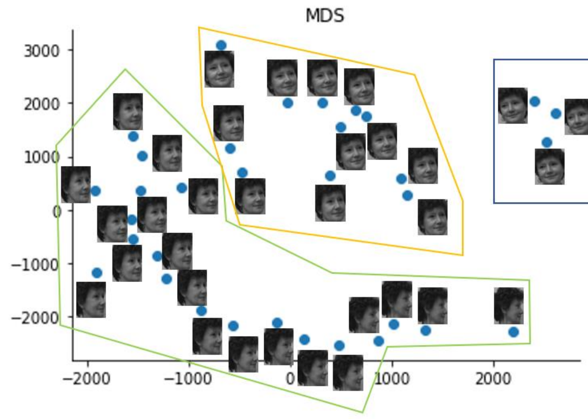


Figure 4: Embedding graph by MDS

4.3.3 ISOMAP

We implemented the ISOMAP with 5 nearest neighbor. Figure 5 shows the face order by ISOMAP. The face order by first eigenvector is correct from left to right.

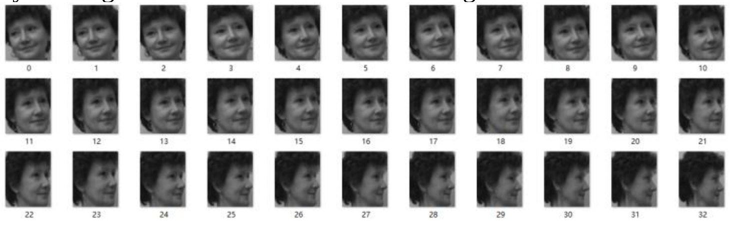


Figure 5: Order the faces by ISOMAP

Figure 6 shows the 2D embedding graph by ISOMAP. It is obvious that ISOMAP has a better data distribution on 2D graph than MDS as it replaces euclidean distance with geometric distance.

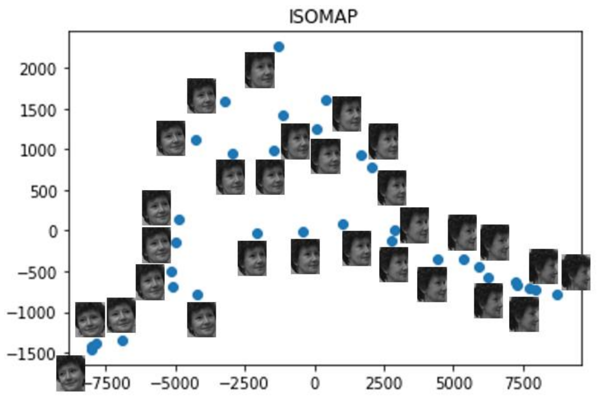


Figure 6: Embedding graph by ISOMAP

4.3.4 LLE

Figure 7 shows the embedding results with 5 nearest neighbor by LLE. The face orientation is also correct according to the first eigenvector. The angle of face gradually changed from right to left. In order to get face rank from left to right. We can sort the face order from lagre first eigenvector to small first eigenvector. The face images towards left is separated from other face images towards right on 2D graph. Compared with ISOMAP and Diffusion map, the LLE curve has a more uniform distribution because of the data approximation by local linear interpolations.

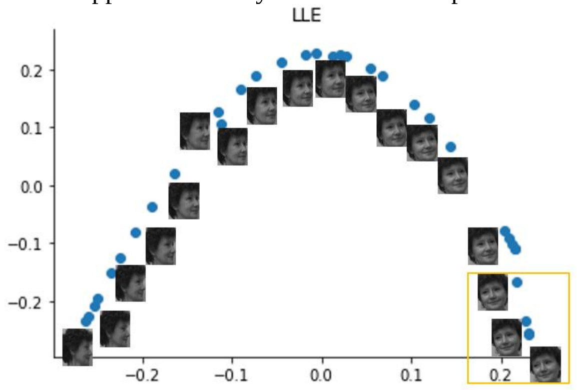


Figure 7: Embedding graph by LLE

4.3.5 LSTA

Figure 8 shows the embedding results with 5 nearest neighbor by LSTA. The LSTA curve is smoother than ISOMAP curve. The face images with left orientation in yellow region are more distinguishable from the face images with right orientation.

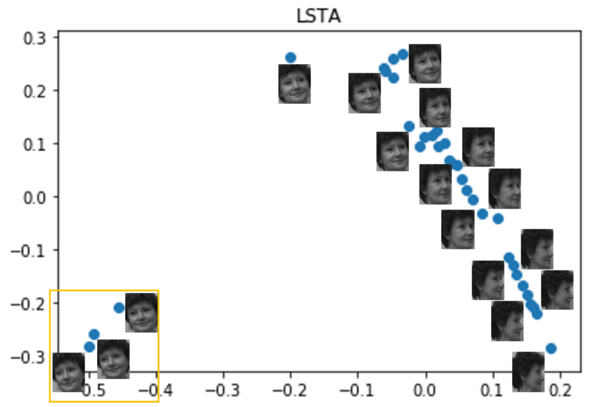


Figure 8: Embedding graph by LSTA

4.3.6 TSNE

Figure 9 shows the embedding results by TSNE. Although the face order on first eigenvector is not good, the 2D embedding graph can be used to distinguish different face images. As we can see in the figure, the face images with left orientation are in middle bottom yellow region. The face images with different angle of right orientation are separated well in red, purple, blue, green region.

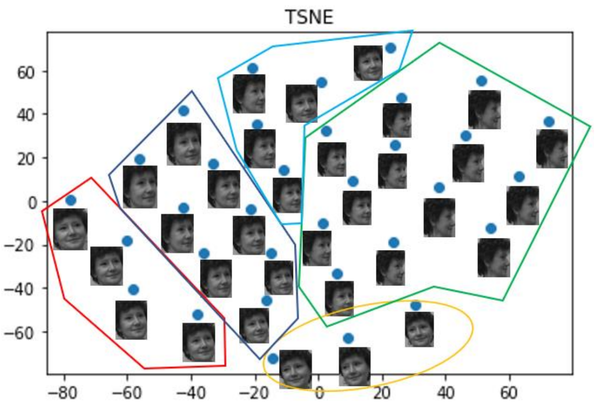


Figure 9: Embedding graph by TSNE

4.4 Parameter study

For ISOMAP, LLE and LSTA, we noticed that the number of nearest neighbor will influence the performance of the model. Therefore, we would like to explore the effect by adjusting the parameter of these three methods.

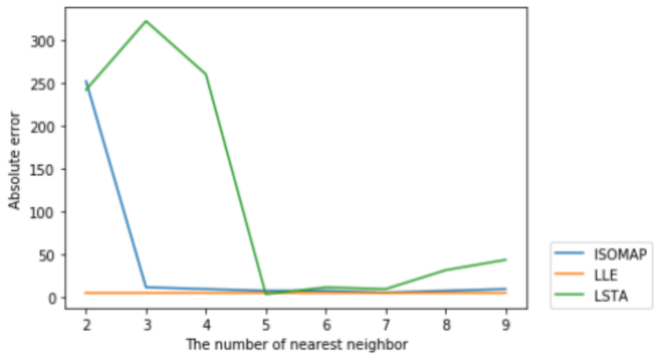
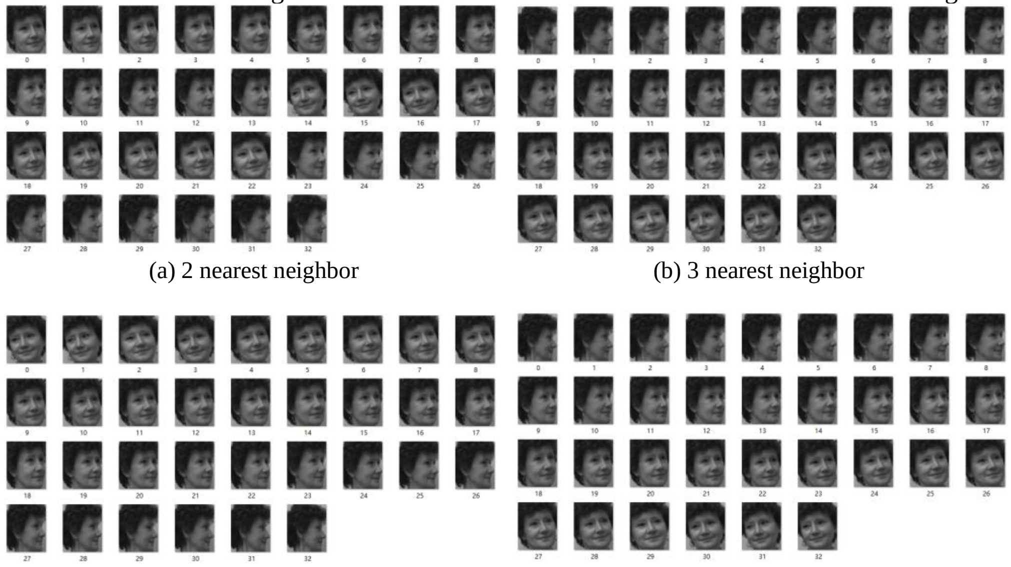


Figure 10: The effect of different nearest neighbor setting

Figure 10 shows the results in three different manifold learning methods. We can see that LLE is more stable and robust with different number of nearest neighbor. It achieves absolute error of 6. With the increase of nearest neighbor, LSTA and ISOMAP have less error. LSTA achieves absolute error of 4 with 5 nearest neighbor. ISOMAP achieves absolute error of 6 with 7 nearest neighbor.



(a) 2 nearest neighbor

(b) 3 nearest neighbor

(c) 7 nearest neighbor

(d) 8 nearest neighbor

Figure 11: Face order results of different nearest neighbor by ISOMAP

During the parameter study, we found an interesting phenomenon that the initial face direction of the smallest first eigenvector is different for LSTA and ISOMAP. Figure 11 shows some face order results of first eigenvector from small to large with different nearest neighbor by ISOMAP. We can see that the face order of 3 and 8 nearest neighbor is from right to left. The face order of 2 and 7 nearest neighbor is from left to right.

5 Conclusion

In summary, we perform several manifold learning methods (Diffusion map, MDS, ISOMAP, LLE, LSTA, TSNE) to order the face direction from left to right. We calculate the absolute error between prediction results and ground truth ranking label to quantify the sorting performance. MDS and TSNE perform poorly by the order of first eigenvector while Diffusion map, ISOMAP, LLE, LSTA have a good performance.
However, MDS and TSNE have better visualization results by adding the second eigenvector, which can be seen from the 2D embedding graph. The 2D scatter plots of Diffusion map, ISOMAP, LLE, LSTA are very intuitive. The two-dimensional graphs of these methods are very similar with the shape of '∨' and '∧'. The LLE curve has a more uniform distribution while the curves of Diffusion map and LSTA are more sensitive to distinguish the left and right orientation of face.
We also explored the influence of different nearest neighbor number. The best number of nearest neighbor is unique for different methods. LLE seems to be robust and works well with nearest neighbor number from 2 to 9. ISOMAP works best with 7 nearest neighbor and LSTA achieves best performance with 5 nearest neighbor.

6 References

[1] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR, 2011.
[2] An introduction to diffusion maps, Porte, J. D. La et al., 2008.
[3] Modern Multidimensional Scaling, I. Borg, P. Groenen, Springer Series in Statistics, 1997.
[4] A global geometric framework for nonlinear dimensionality reduction, J.B. Tenenbaum, V. De Silva et al., Science, 2000.
[5] Nonlinear dimensionality reduction by locally linear embedding, S. Roweis, L. Saul, Science, 2000.
[6] Visualizing High-Dimensional Data Using TSNE, L.J.P. van der Maaten, G. Hinton, Journal of Machine Learning Research, 2008