CSIC 5011 Final Project: Order the Faces via Manifold Learning

Di LIU
Department of Mechanical and Aerospace Engineering
The Hong Kong University of Science and Technology
dliuah@connect.ust.hk

Meilan WANG
Department of Civil and Environmental Engineering
The Hong Kong University of Science and Technology
mwangau@connect.ust.hk

Xu HAN
Department of Physics
The Hong Kong University of Science and Technology
xhanah@connect.ust.hk

Abstract

This project is intended to order 33 images of the same person but with different face directions. Several manifold learning techniques were employed (e.g. Diffusion Map *etc.*). After comparing the results, we found that locally linear embedding performed better than others in which number of components and neighbors are 2 and 5, respectively. We also conducted experiments to compare the performance depending on hyper-parameter (number of components), by changing the number of neighbors from 5 to 10. It was discovered that 5 neighbors provided the best model performance.

1 Introduction

Face pose determination is an important area of research in human computer interaction (HCI). An important problem in HCI is to determine one's focus of attention (inattention or lack of attention). This can be inferred from the person's head orientation and gaze direction. Head directions can be estimated from one's face orientation. Moreover, a person's state of mind and/or level of vigilance can also be deduced from his/her face orientation. For example, tracking one's face orientation through multiple image frames allows us to detect the nodding behavior, which can be used to infer one's fatigue level. In summary, face pose estimation plays an important role in HCI [1].

Face pose estimation methods include *model-based* and *face property-based* (or *appearance-based*) two categories. Model-based approaches assume a three-dimensional (3D) model of face which could be recovered by establishing 2-3D features correspondences. Property-based approaches assume there is causal-effect relationship between 3D face pose and certain properties of the facial image.

In this report, we try to explore how manifold learning methods contribute to appearance-based methods so as to order head directions. The experiments were conducted on a small dataset, 33 images which are 33 angles of the same person. The goal of our experiment is to order the relative head directions from left to right correctly matching the ground truth voted by our group members.

Preprint. Work in progress.

More than following the calculation steps suggested by the project requirement, we tested 9 prevailing manifold learning methods to compare their performance on the relative head direction estimation. And we also tested how hyper-parameter influenced on their performance.

The rest of the paper's structure is as follows. In next section, we will briefly introduce the dataset. In Section 3, we will describe the eight manifold learning methods which we have learned in class. In Section 4, we will present the experiment results and evaluation results. In Section 5, there is a brief summary and future work discussion.

2 Dataset

The given dataset contains 33 faces of the same person (Y $\in \mathbb{R}^{112\times 92\times 33}$) in different angles which can obtained from the following website,

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

We then created a data matrix X $\in \mathbb{R}^{n\times p}$ where n = 33, p = 112 $\times$ 92 = 10304.

3 Methods

3.1 Multi-dimensional scaling (MDS)

MDS is a method which aims to recover Euclidean coordinates in given pairwise distance metrics or dissimilarities [2]. Given a set of data points x_1, x_2, ..., $x_n \in \mathbb{R}^p$, let

$$X = [x_1, x_2, ..., x_n]^{p\times n}.$$

The distance between different samples is denoted as $d_{ij}^2 = ||x_i - x_j||^2$ and then a squared distance matrix D_{ij} = d_{ij}^2 is created.

Afterwards, we compute $-\frac{1}{2}HDH^T$ as B where H is the Householder centering matrix. By using eigenvalue decomposition $B = U\Lambda U^T$ with $\Lambda = diag(\lambda_1, ..., \lambda_n)$ where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$, we can choose the top k eigenvalues and corresponding eigenvectors to form the embedding data point $\tilde{T_k} = U_k\Lambda_k^{\frac{1}{2}}$ where

$$U_k = [u_1, ..., u_k], u_i \in \mathbb{R}^n$$
$$\Lambda_k = diag(\lambda_1, ..., \lambda_k)$$

with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_k > 0$.

3.2 ISOMAP

ISOMAP is a extended method of MDS in that it uses pairwise geodesic distances between data points and graph shortest path distances to reconstruct the data.

The first step is to construct a neighborhood graph $G = (V, E, d_{ij})$ where $V = \{x_i : i = 1, ..., n\}$, $E = \{(i, j)$: if j is a neighbor of i $\}$ and $d_{ij} = d(x_i, x_j)$.

Next, we compute the graph shortest path distance

$$d_{ij} = min_{P=(x_i,...x_j)}(||x_i - x_{t_1}|| + ... + ||x_{t_{k-1}} - x_j||)$$

which connects i and j.

Then the following steps are the same as MDS,

(1) Compute $K = -\frac{1}{2}HDH^T$(D:=[d_{ij}^2]), where H is the Householder centering matrix.

(2) Computer eigenvalue decomposition $K = U\Lambda U^T$ with $\lambda = diag(\lambda_1, ..., \lambda_n)$ where $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$;

(3) Choose top k nonzero eigenvalues and corresponding eigenvectors as embedding coordinates

$$Y_k = U_k\Lambda_k^{\frac{1}{2}}$$

3.3 Locally linear embedding (LLE)

The central point of LLE is that any data point in a high dimensional space can be a linear combination of data points in its neighborhood.

Given a graph $G = (V, E)$, one can first do linear fitting. Namely, for each x_i and its neighbors $\mathcal{N}_i$, solve

$$\min_{\sum_{j\in\mathbb{N}_i} w_{ij}=1} \left\| x_i - \sum_{j\in\mathcal{N}_i} w_{ij} x_j \right\|^2$$

by $\hat{w}_i(\mu) = (C_i + \mu I)^{-1} \mathbf{1}$ for some regularization parameter $\mu > 0$ and $w_i = \hat{w}_i / \hat{w}_i^T \mathbf{1}$.

The next step is to do global alignment by computing $K = (I - W)^T (I - W)$ where

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

The last step is to do eigenvalue decomposition $K = U\Lambda U^T$ with $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$ where $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_{n-1} > \lambda_n = 0$. Then choose bottom $k + 1$ nonzero eigenvalues and their corresponding eigenvectors with dropping the 0-constant eigenvalue, such that

$$U_d = [u_{n-d}, \ldots, u_{n-1}], \quad u_j \in \mathbb{R}^n$$
$$\Lambda_d = diag(\lambda_{n-d}, \ldots, \lambda_{n-1})$$

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

3.4 Local Tangent Space Alignment(LTSA)

LTSA is a modified version of LLE. Its first step is to do local PCA, i.e. computing local SVD on neighborhood of x_i, $x_{i_j} \in \mathcal{N}(x_i)$,

$$\tilde{X}^{(i)} = [x_{i_1} - \mu_i, \ldots, x_{i_k} - \mu_i]^{p\times k} = \tilde{U}^{(i)} \tilde{\Sigma} \left(\tilde{V}^{(i)}\right)^T$$

where $\mu_i = \sum_{j=1}^k x_{i_j}$.

The second step is to do tangent space alighment,

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

wehere $S_i^{n\times k} : [x_{i_1}, \ldots, x_{i_k}] = [x_1, \ldots, x_n] S_i^{n\times k}$ and $G_i = \left[1/\sqrt{k}, \tilde{V}_1^{(i)}, \ldots, \tilde{V}_d^{(i)}\right]^{k\times(d+1)}$.

Finally, we do eigenvalue decomposition $K = U\Lambda U^T$ and find the smallest $d + 1$ eigenvectors of K with dropping the smallest one. The remaining d eigenvectors will give rise to d-mebedding coordinates.

3.5 Modified LLE (MLLE)

MLLE is another method to improve the performance of LLE. It uses multiple weight vectors projected from orthogonal complement of local PCA in each neighborhood. It replaces the weight vector above by a weight matrix $W \in \mathbb{R}^{k_i\times s_i}$, a family of s_i weight vectors using bottom s_i eigenvectors of C_i, $V_i = [v_{k_i-s_i+1}, ..., v_{k_i}] \in \mathbb{R}^{k_i\times s_i}$, such that

$$W_i = (1 - \alpha_i)\, \omega_i(\mu) 1_{s_i}^T + V_i H_i^T$$

where $\alpha_i = \left\|V_i^T 1_{k_i}\right\|_2 / \sqrt{s_i}$ and H_i is a Householder centering matrix. Equipped with this weight matrix, one can set the objective function by simultaneously minimizing the residue over all reconstruction weights:

$$\min_Y \sum_i \sum_{l=1}^{s_i} \left\| y_i - \sum_{j\in N_i} W_i(j,l) y_j \right\|^2 = \sum_i \left\| Y\widehat{W}_l \right\|_F^2 = trace\left[Y \left(\sum_i \widehat{W}_\iota \widehat{W}_l^T \right) Y^T \right]$$

where $\widehat{W_l}$ is the embedding of $\mathrm{W}_i \in \mathbb{R}^{k_i \times s_i}$ into $\mathbb{R}^{n \times s_i}$,

$$\widehat{W_\imath}(j,:) = \begin{cases} -1^T_{s_i'}, & j = i \\ W_i, & j \in N_i \\ 0, & \text{otherwise} \end{cases}$$

3.6 Hessian LLE (HLLE)

HLLE is also a method to solve the regularization problem of LLE. It revolves around a hessian-based quadratic form at each neighborhood. Donoho and Grimes [3] proposed Hessian LLE (Eigenmap) in search of

$$\min_{y \perp 1} \int \|\mathcal{H}_y\|^2, \quad \|y\| = 1$$

The basic algorithmic idea is as follows,

1. G is incomplete, often k-nearest neighbor graph.

2. Local SVD on neighborhood of x_i, for $x_i \in \mathcal{N}_i$,

$$\tilde{X}^{(i)} = [x_{i_1} - \mu_i, \ldots, x_{i_k} - \mu_i]^{p \times k} = \tilde{U}^{(i)} \tilde{\Sigma} \left(\tilde{V}^{(i)}\right)^T$$

where $\mu_i = \sum_{j=1}^k x_{i_j} = \frac{1}{k} X_i 1$.

3. Null Hessian estimation: define

$$\mathrm{M} = \left[1, \tilde{V}_1, \ldots, \tilde{V}_d, \tilde{V}_1^2, \tilde{V}_1 \tilde{V}_2, \ldots, \tilde{V}_{d-1} \tilde{V}_d, \tilde{V}_d^2\right] \in \mathbb{R}^{k \times (1+d+(d+1))}$$

where $\tilde{V}_i \tilde{V}_j = \left[\tilde{V}_{ik} \tilde{V}_{jk}\right]^T \in \mathbb{R}^k$.

Then perform a Gram-Schmidt Orthogonalization procedure on M get $\widetilde{\mathrm{M}}$ and the null Hessian $\left[H^{(i)}\right]^T = \left[\text{last} \begin{pmatrix} d+1 \\ 2 \end{pmatrix} \text{columns of } \tilde{M}\right]_{k \times \frac{d+1}{2}}$ as the first $d+1$ columns of $\widetilde{\mathrm{M}}$ consists an orthonormal basis for the kernel of Hessian together with the constant vector.

Define a selection matrix $S^{(i)} \in \mathbb{R}^{n \times k}$ which selects those data in $\mathcal{N}(x_i)$, and then the kernel matrix is defined to be

$$K = \sum_{i=1}^n S^{(i)} H^{(i)T} S^{(i)T} \in \mathbb{R}^{n \times n}$$

.

Find smallest $d+1$ eigenvectors of K and drop the smallest eigenvector, the remaining d eigenvectors will give rise to a d dimensional embedding of data points

3.7 Diffusion map

Give a data set $x_i \in \mathbb{R}^d$, one can define a undirected weighted graph $G = (V, E, W)$. V and E are the same as before but W is a symmetric matrix,

$$W_{ij} = W_{ji} = exp(-\frac{d(x_i, x_j)^2}{t})$$

for $i \in \mathcal{N}_i$, otherwise 0.

Then, let $d_i = \sum_{j=1}^n W_{ij}$ and $D = diag(d_i)$. A random walk on graph G can be defined through Markov matrix,

$$L = D^{-1} W - I$$

.

Finally, we do eigenvalue-decomposition of L and obtain the embedding coordinates.

3.8 t-distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a method which uses Student t-distribution or Cauchy distribution kernel instead of Gaussian distribution kernel. The following are the same.

The experiments we conducted were implemented in Matlab and Python. The codes in Matlab for ISOMAP and LLE were from Ref. [4], [5], respectively. Experiments for other methods were conducted by using the sklearn.manifold library [6] in Python.

4 Results and Discussions

4.1 Ground truth

In order to evaluate the performance of different methods, a ground truth is needed since the original dataset is out of order. At first, we voted for the order of which the woman's head in the photo turns from the left to the right as shown in Figure 1.

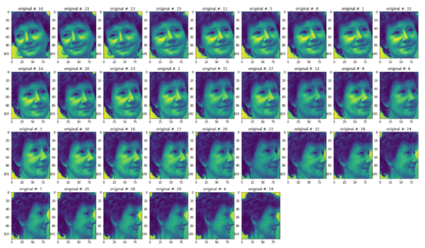


Figure 1: Ground truth of the ordered image.

4.2 Performance evaluation

Table 1: The performance evaluation metrics of all the methods without error analysis.

Original#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Ground truth	8	13	19	32	6	18	28	7	17	1	5	16	12	10	4	21
Diffusion Map	9	11	19	31	5	18	28	6	17	1	7	16	12	10	4	21
MDS	9	10	19	32	5	18	30	6	17	1	7	16	12	11	4	21
ISOMAP	7	13	19	32	5	18	28	10	17	1	6	16	12	9	4	21
LLE	8	13	19	32	5	18	28	7	17	1	6	16	12	10	4	21
MLLE	8	12	19	32	6	18	28	7	17	1	5	16	11	10	4	21
HLLE	8	13	19	32	6	18	28	7	17	2	5	16	12	10	4	21
Spectral Embedding	6	13	18	32	8	19	28	7	17	3	5	16	12	10	4	21
LTSA	8	13	19	32	6	18	28	7	17	2	5	16	12	10	4	21
t-SNE	8	12	20	30	14	22	23	10	17	1	18	16	5	2	6	9

Original#	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Ground truth	22	26	33	11	2	24	3	27	29	23	15	30	31	20	14	25	9
Diffusion Map	22	26	33	13	2	23	3	27	30	24	14	32	29	20	15	25	8
MDS	22	26	28	13	2	23	3	27	31	24	14	33	29	20	15	25	8
ISOMAP	22	26	33	11	2	24	3	27	29	23	14	30	31	20	15	25	8
LLE	22	26	33	11	2	23	3	27	29	24	14	30	31	20	15	25	9
MLLE	22	26	33	13	2	23	3	27	29	24	14	31	30	20	15	25	9
HLLE	22	26	33	11	1	23	3	27	29	24	14	31	30	20	15	25	9
Spectral Embedding	22	26	33	11	2	24	1	27	29	23	14	30	31	20	15	25	9
LTSA	22	26	33	11	1	23	3	27	29	24	14	31	30	20	15	25	9
t-SNE	33	25	24	27	15	11	4	32	29	21	19	28	31	13	7	26	3

For the sake of quantifying the similarity between each method's result and ground truth, absolute error was used. First, we labelled the original images according to the column index of the matrix from the given dataset. Next, we recorded each photo's position in the ground truth and the results from different methods. Then for each original image, the absolute error was calculated as the difference between the image's position in ground truth and a specific method.

The entire performance evaluation metrics can be seen in Table 1. Then, total absolute error (TAE) which sums up the absolute error for each image is shown here. The TAE for each method can be obtained in Table 2.

Table 2: TAE of different methods. The first four values are based on Matlab codes while the latter five values are from Python codes.

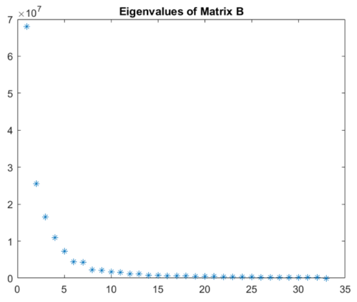
Methods	Diffusion Map	ISOMAP	LLE	MDS	HLLE	Spectral Emebedding	LTSA	MLLE	t-SNE
TAE	20	10	6	30	8	12	8	10	164

Basically, we can observe that t-SNE has a much larger error than other methods, which suggests that this method may not be suitable for this face ordering task. Apart from t-SNE, MDS has the largest TAE among all the other methods. Then, the other 7 methods have relatively small TAE. Apparently, LLE shows the best performance, which has a TAE of 6.

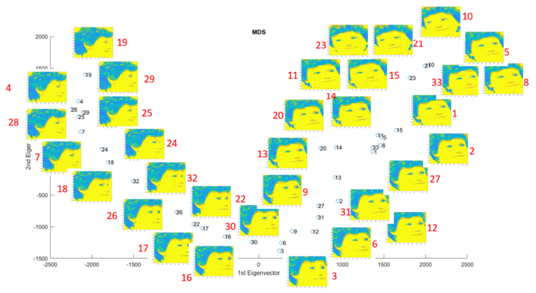
4.2.1 Results of MDS and ISOMAP

Considering that the methods we used are too many, we only take the results of MDS and ISOMAP for examples. For other methods, we did the same process.

Figure 2(a) shows that the first two eigenvalues has explained a large portion of the total variation. The embedding result on the first two eigenvectors is shown Figure 2(b).



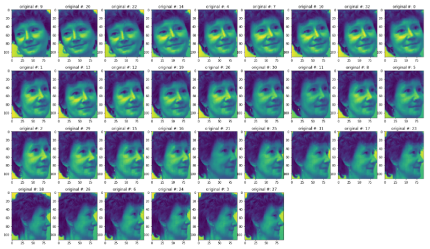
(a)



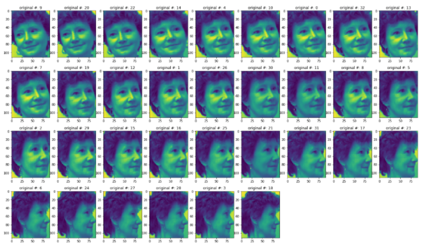
(b)

Figure 2: Analysis of MDS. (a)Explained variance as a function of number of components in MDS. (b) 2D embedding graph.

As can be seen, the scatter plot exhibits a ‘v’ shape, one possible explanation for eigenvector 2 is that it measures the yellow area on the top left and right corner. When the head slowly rotates from right to left, those yellow areas are filled by blue hair. But as the head continue to rotate to the left, the yellow areas reappear. The result sorted by the first eigenvector is shown in Figure 3(a).



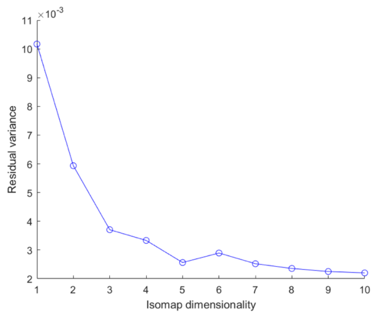
(a)



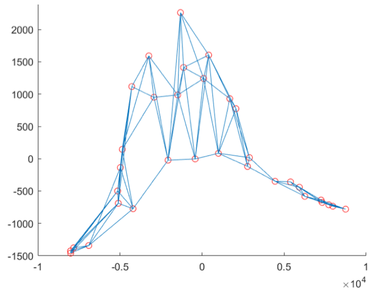
(b)

Figure 3: Ordered face image of (a)MDS and (b)ISOMAP.

The ISOMAP was applied by using Tenenbaum's Matlab code with $k = 5$ nearest neighbors graph. The residual variance is shown in Figure 4(a) and two dimensional Isomap embedding result with neighbor graph is shown in Figure 4(b). Figure 3(b) illustrates the order sorted by the first eigenvector.



(a)



(b)

Figure 4: Analysis of ISOMAP. (a)Residual variance as a function of number of components in MDS. (b)2D embedding graph.

4.2.2 Hyper-parameter study

Moreover, we explored the effect of number of neighbors (k) on the sorting performance for each method. We tested for k =5 to 10, the methods were all applied with sklearn.manifold library. Compared with the other methods, LLE tends to be affected by k more than others. In terms of performance versus number of neighbors, we can find that 5 tends to be the best for most methods.

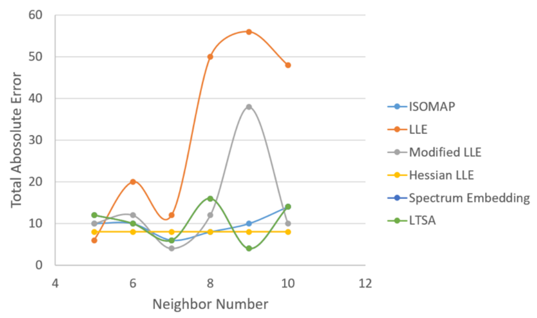


Figure 5: TAE as a function of number of neighbors in different methods.

4.3 2D embedding graph

Afterwards, we compared the 2D embedding graph for different methods based on sklearn.manifold library. We assigned the number of neighbors to be 5 in our experiments. The results can be seen in Figure 6. Obviously, LLE and Spectral embedding seem to better than others in terms of x-axis which represents the first component. With the results before taken into consideration (i.e. TAE of different methods), we may know that LLE should be the best.

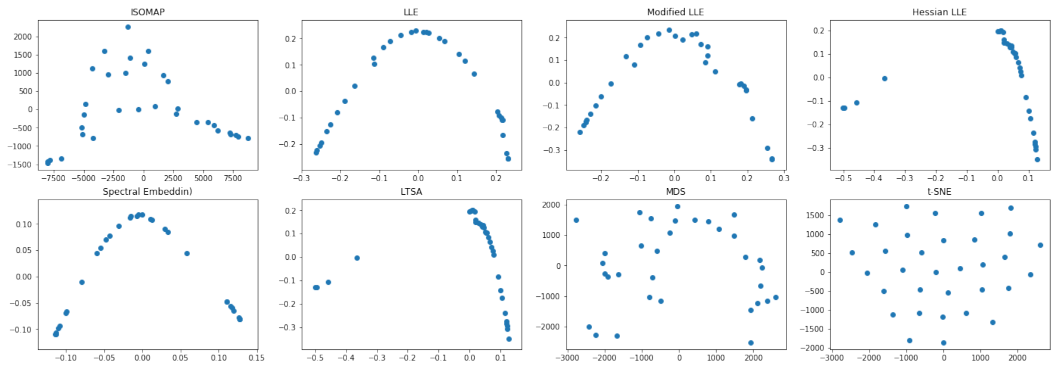


Figure 6: Manifold learning with 33 data points, 5 neighbors.

Inspired by the results in Figure 5, we also studied the effect of number of neighbors on the 2D embedding graph. Here, we only take LLE for an example (see Figure 7). One interesting phenomenon is that the shape of the graph can be both ’V’ and ’Λ’. However, this phenomenon makes no difference to the conclusion. The focus here is the discrepancy between different points in the figure. As the number of neighbors increases, more and more points tend to be indistinguishable. This provides us with another way to understand that the number of neighbors of 5 should behave the best.

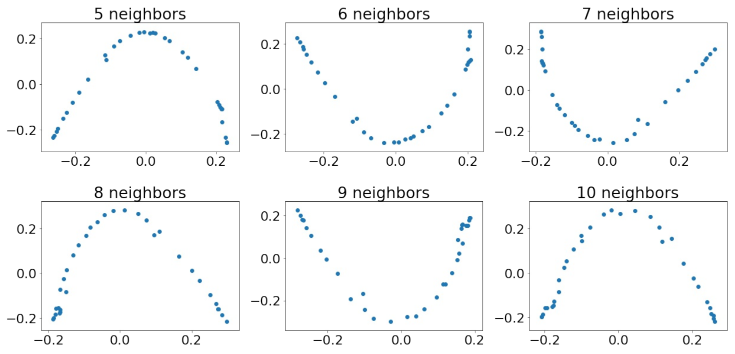


Figure 7: Spectral embedding with different number of neighbors.

5 Conclusion and future work

In summary, almost all the methods we explored show reasonable sorting order except t-SNE. We quantified the sorting performance by calculating the total absolute error between different methods’ results and ground truth and found that LLE exhibited the best performance. What’s more, a parameter study was performed on the number of nearest neighbor k and 5 proved to be the best.

For future work, these methods can be applied to a more complicated dataset, (e.g. head images with both turning and nodding motion). By adding one degree of freedom, more eigenvectors may be involved in deciding the order. Last but not least, more techniques can be combined with the methods in this report to achieve emotion detection and so on.

Contributions

Di LIU: code in Matlab (Diffusion map, MDS, ISOMAP, LLE), ground truth collection and performance metrics decision, result organization, report (4.1, 4.2 and 5), ppt & presentation.

Meilan WANG: code in Python (8 methods, hyper-parameter test and performance evaluation), report (abstract, 1, 2, 3.5, 3.6, 4.2 and 4.3), ppt & presentation.

Xu HAN: code in Python (6 methods), report (3 (exclude 3.5, 3.6) and 4.3, proofreading and format modifying)

References

[1] Ji, Qiang. *3D face pose estimation and tracking from a monocular camera*, Image and vision computing, 20.7 (2002): 499-511.

[2] G. Young and A. S. Householder, *A note on multidimensional psycho-physical analysis*, Psychometrika **6** (1941), 331–333.

[3] David L. Donoho and Carrie Grimes. *Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data*, Proceedings of the National Academy of Sciences of the United States of America 100 (2003), no. 10, 5591-5596.

[4] `https://github.com/yao-lab/yao-lab.github.io/blob/master/data/isomapII.m`

[5] `https://github.com/yao-lab/yao-lab.github.io/blob/master/data/lle.m`

[6] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.