Homework 7 for Pattern Recognition

Fan JIN (2015011506)

May 31, 2018

Question 1

Notations

The multidimensional scaling (MDS) seeks to find a centered configuration $X = [x_1, \cdots, x_n]$, where $x_1, \cdots, x_n \in \mathbb{R}^2$, such that its distance matrix $\hat{D} = (\hat{d}_{ij}) \approx D$ where

$$\hat{d}_{ij}^2 = (x_i - x_j)^T (x_i - x_j)$$

for $1 \leq i \leq n$ and $1 \leq j \leq n$.

Define the Gram matrix

$$B = (b_{ij}) = X^T X,$$

and it follows that

$$\hat{d}_{ij}^2 = x_i^T x_i + x_j^T x_j - 2x_i^T x_j = b_{ii} + b_{jj} - 2b_{ij}.$$

By algebra manipulations[1], we can express b_{ij} in terms of d_{ij}:

$$\hat{B} = -\frac{1}{2} J \hat{D} J,$$

where

$$J = I_n - \frac{1}{n} 1 \cdot 1^T.$$

Thus, we take B as an estimate of $\hat{B}$, by reducing the dimension using the PCA approach. Apply eigen decomposition and we obtain

$$B = V * D * V^T,$$

where the eigenvalues are sorted in descending order. Retain the first two eigenvalues:

$$\hat{B} = \hat{V} * \hat{D} * \hat{V}^T,$$

where $\hat{V}$ is the first two columns of V, and $\hat{D}$ is the first two eigenvalues of the diagonal matrix D. Then, we have

$$X = \hat{D}^{1/2} \hat{V}^T$$

which satisfies

$$B = X^T X.$$

[1]https://en.wikipedia.org/wiki/Multidimensional_scaling

Visualization

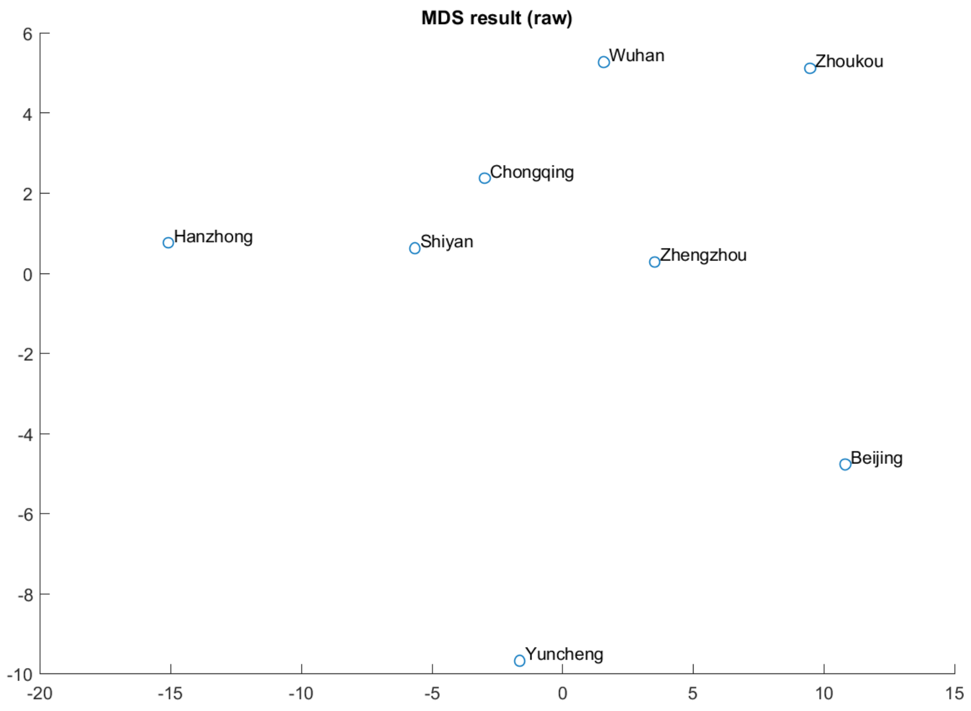


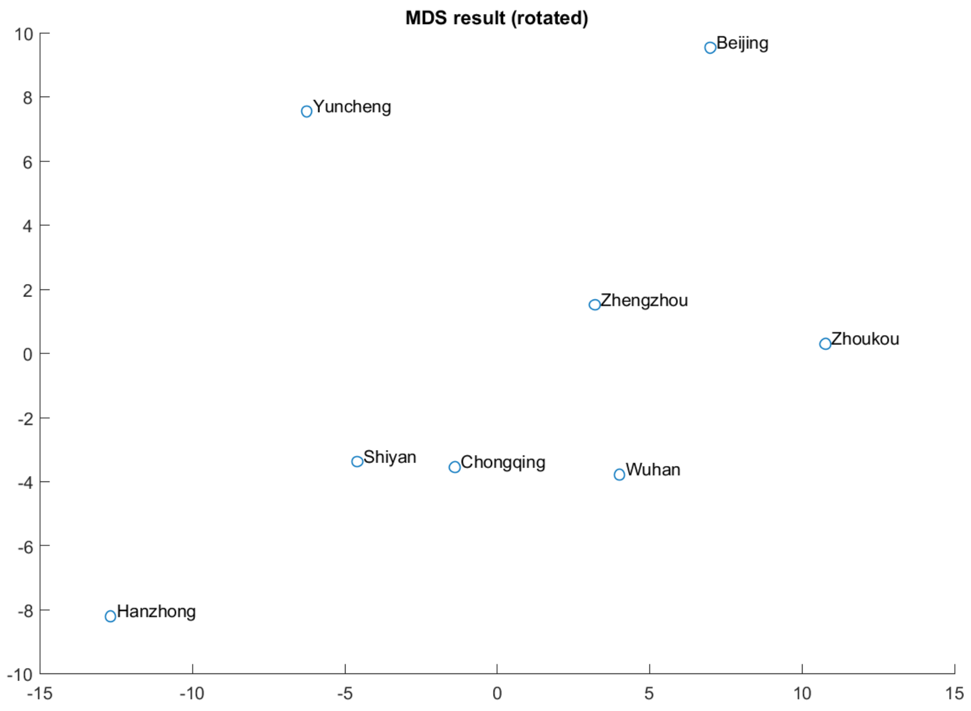
Figure 1: Raw result

To compare it with the actual satellite maps, we rotate it such that Zhengzhou is located roughly on the north of Wuhan.

We see the cities in main lines (e.g. Beijing, Zhengzhou, Wuhan) preserve their relative location better, compared to those in branch lines. This is mainly because the linearity between distance and traveling duration differs in the railway level. For main line railways, the traveling duration is usually linear with the distance on map, for they have

- routemaps much more straightforward,
- fewer stops,
- faster train speed.

It is noticed that the city Hanzhong seems like an outlier. This city, located in the middle of Mount Qinling and Mount Daba, was connected to other cities by a very slow branch line railway, before the construction of highspeed railways. This is why it is far away from other cities in the MDS map, although it is actually not so on the fringe.



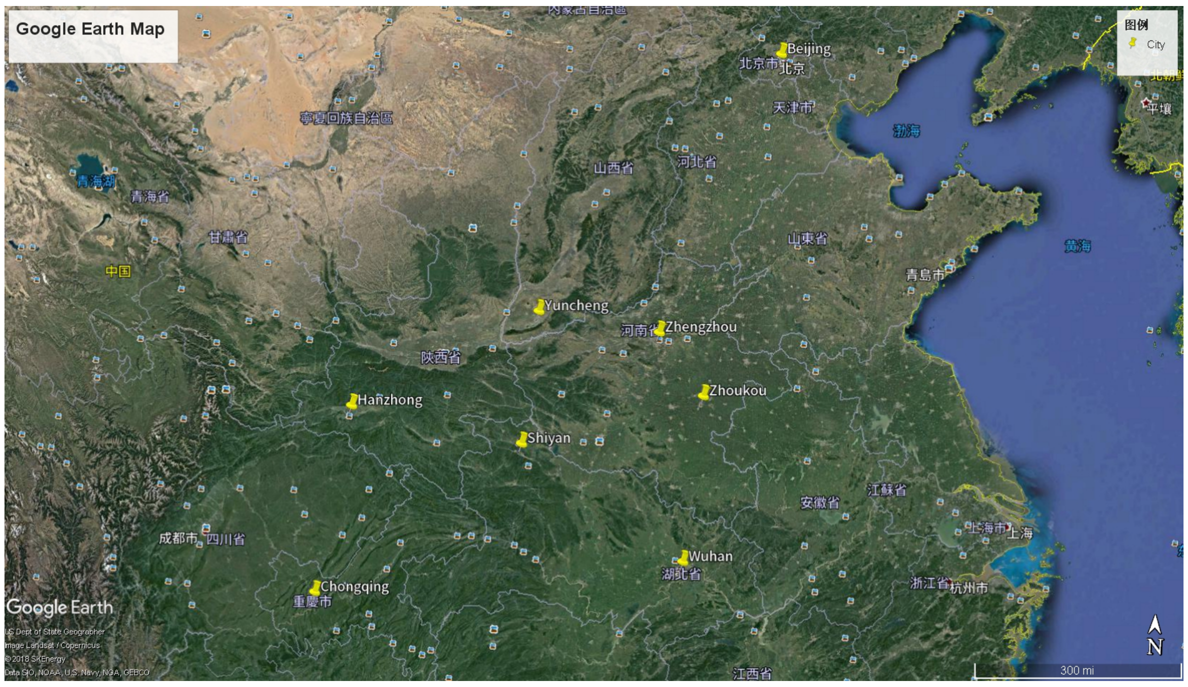


Figure 2: Comparison

Question 2a

PCA

The PCA (principal component analysis) applies eigen decomposition to the covariance matrix (unnormalized) or correlation matrix (normalized) to the original data in P dimensions, and only retains the first p largest eigenvalues and their corresponding eigenvectors, which reconstructs the compressed data in p dimensions.

t-SNE

The basic idea of t-SNE (t-distribution stochastic neighbor embedding) is to map the data to a probability distribution by affine transformation.

Given N high dimensional objects $x_1, \cdots, x_N$, the t-SNE algorithm first computes a conditional probability, which indicates the dissimilarity between two objects,

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

where the bandwidth of the Gaussian kernels σ_i is set such that the perplexity of the conditional distribution equals a predefined perplexity. This can be done using the bisection method. [2]

Given the conditional distributions, the joint distribution of two objects is computed as follows.

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}.$$

Unlike the classical SNE algorithm, the joint probabilities in low dimension space are computed in t-SNE using the t-distribution (heavy-tailed).

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|y_i - y_j\|^2)^{-1}}.$$

The final step is to train the low dimensional objects $y_1, \cdots, y_N$ such that the KL distance below between the two distributions is minimized:

$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

LLE

The basic idea of LLE (locally linear embedding) is to map the data to a lower dimensional space while retaining the nonlinear manifold. [3] It hypothesize that each data point is a weighted linear combination of its neighboring objects, and that such weights are shared in both original space and lower dimensional space. It consists of 3 steps:

- Calculate the k nearest neighbors using KNN algorithms.
- Minimize the reconstruction loss

$$\epsilon(W) = \sum_i \|X_i - \sum_j W_{ij} X_j\|^2$$

[2] https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
[3] https://cs.nyu.edu/ roweis/lle/algorithm.html

subject to

$$\sum_j \frac{\sum_k C^{-1}_{jk}}{\sum_{lm} C^{-1}_{lm}} = 1$$

where the local convariance matrix

$$C_{jk} = (x - \eta_j)^T (x - \eta_k)$$

with η is the nearest neighbors of x.

Question 2b

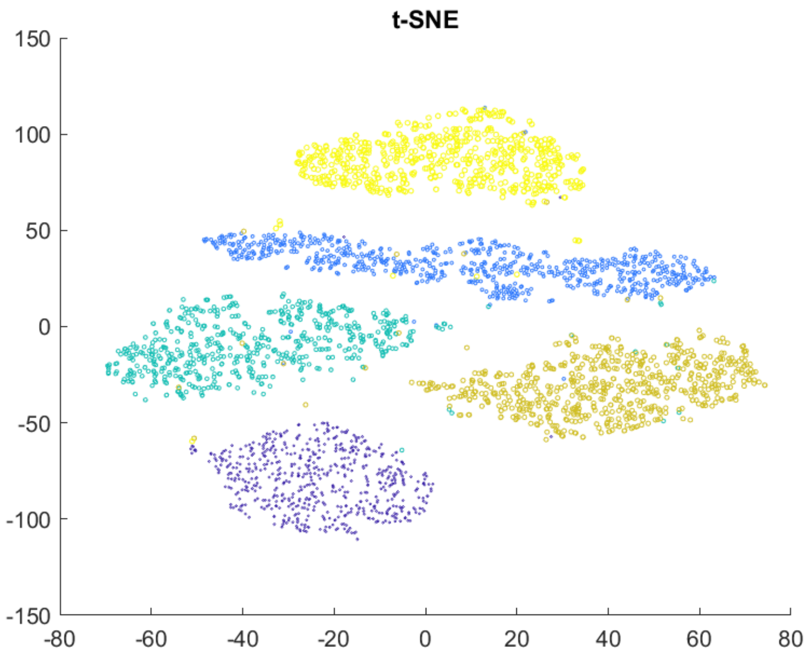


Figure 3: 2D mapped data by t-SNE

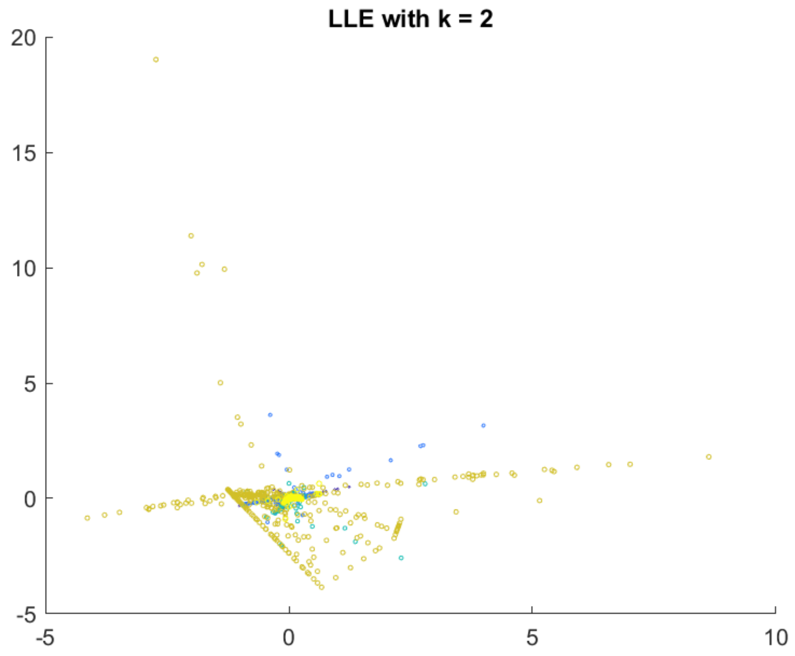


Figure 4: 2D mapped data by LLE with k=2

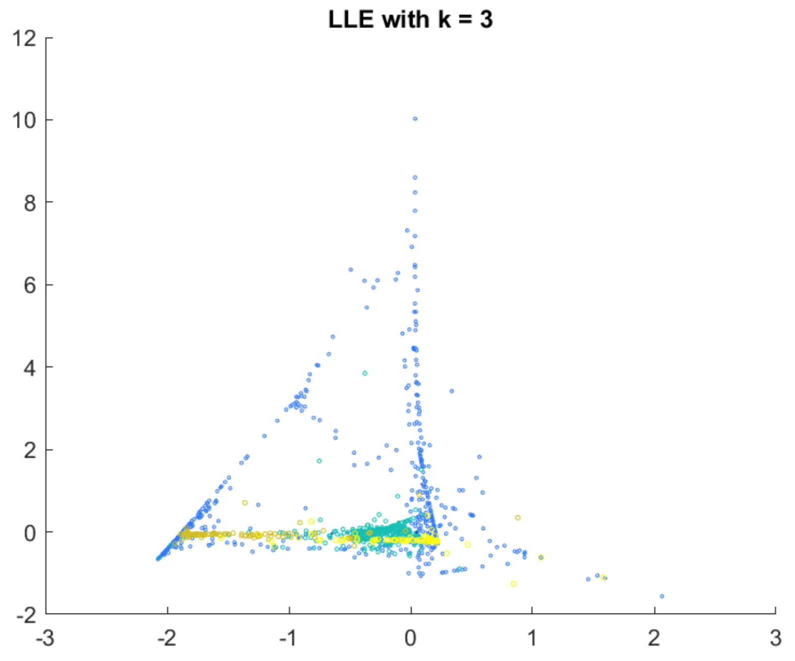


Figure 5: 2D mapped data by LLE with k=3

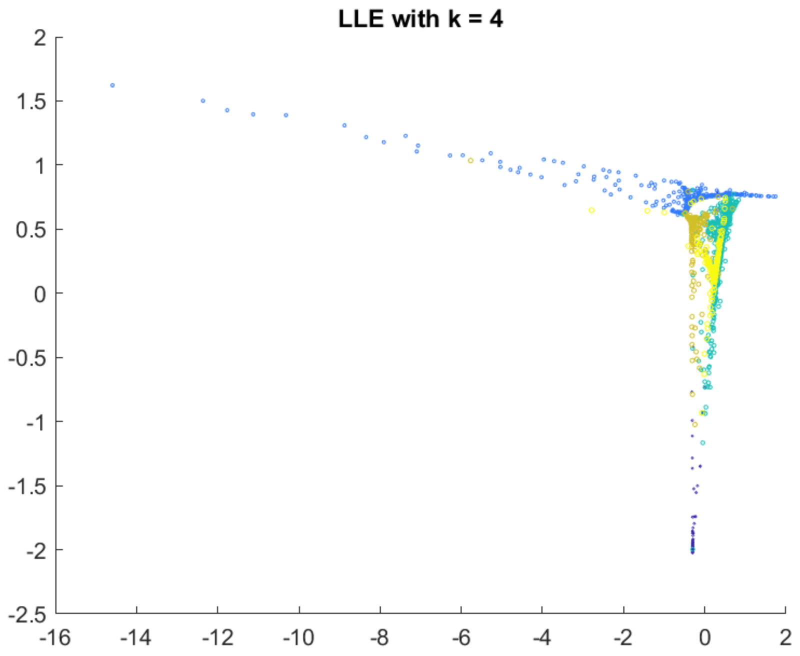


Figure 6: 2D mapped data by LLE with k=4

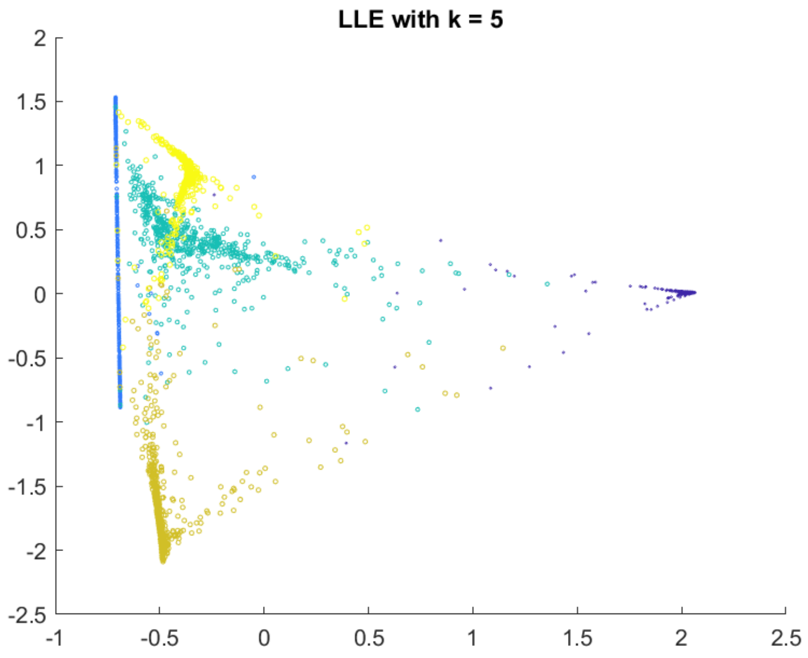


Figure 7: 2D mapped data by LLE with k=5

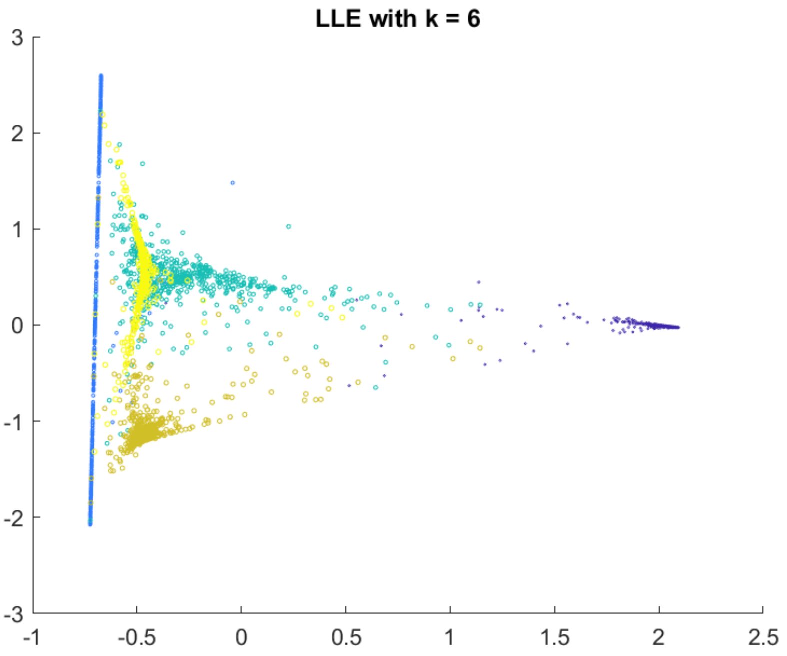


Figure 8: 2D mapped data by LLE with k=6

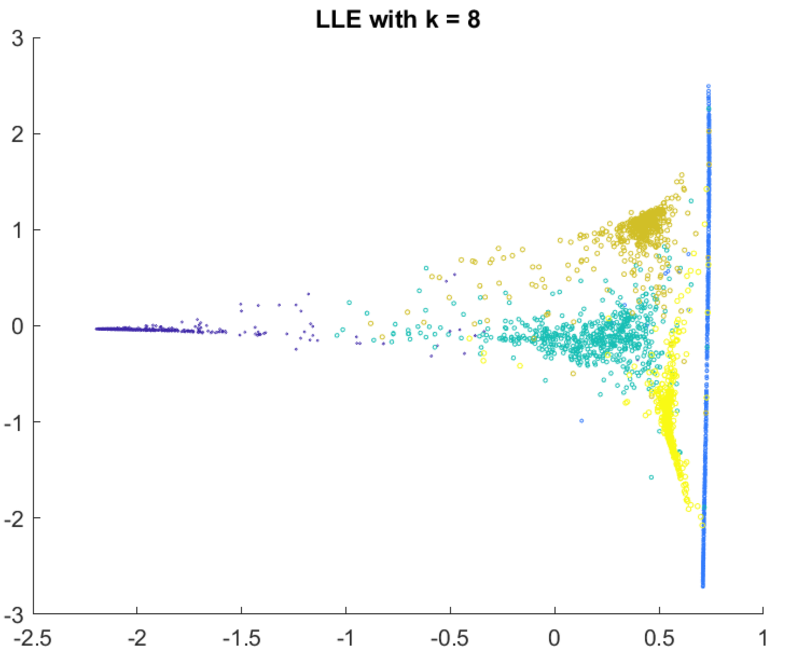


Figure 9: 2D mapped data by LLE with k=8

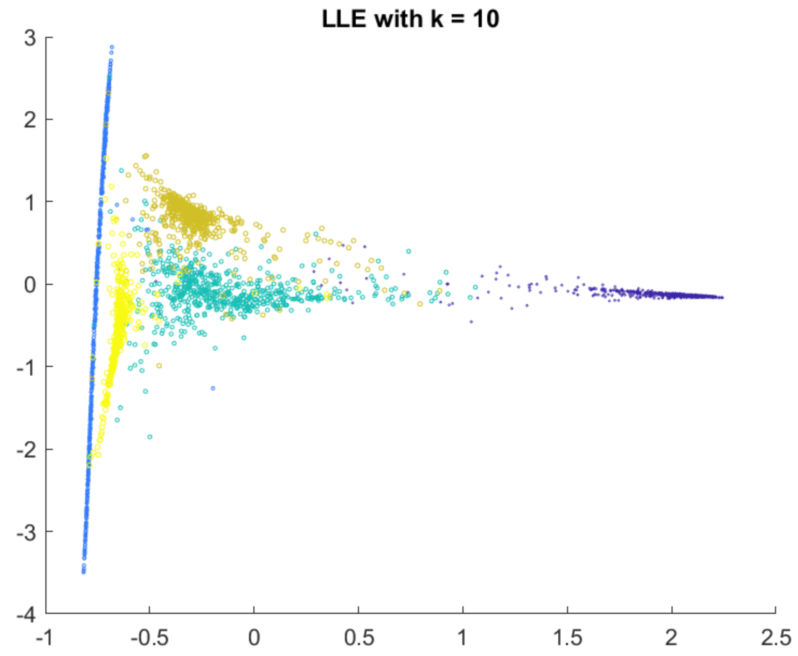


Figure 10: 2D mapped data by LLE with k=10

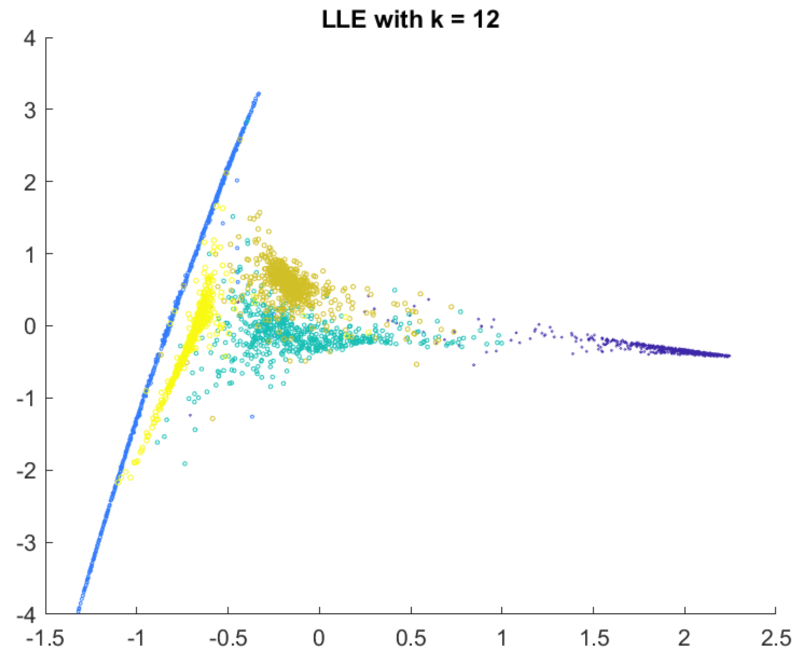


Figure 11: 2D mapped data by LLE with k=12

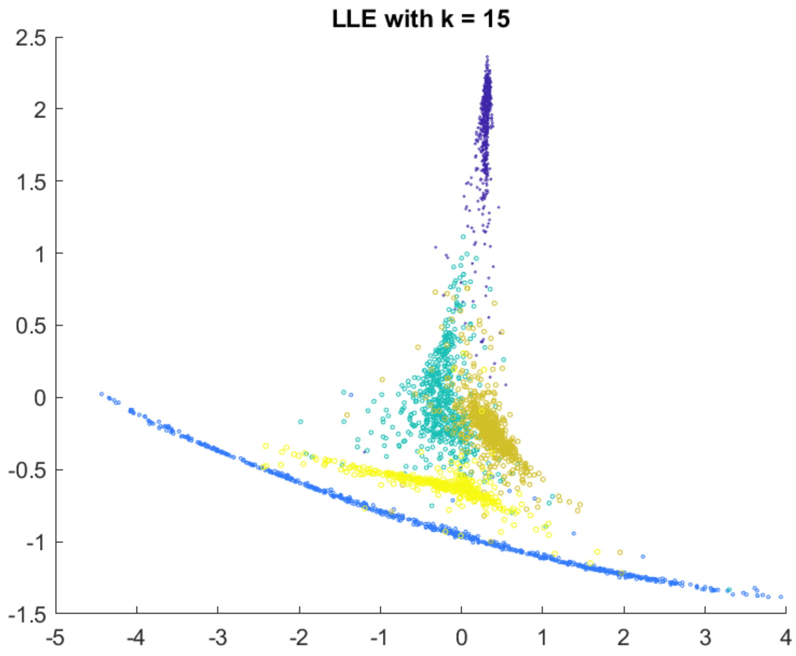


Figure 12: 2D mapped data by LLE with k=15

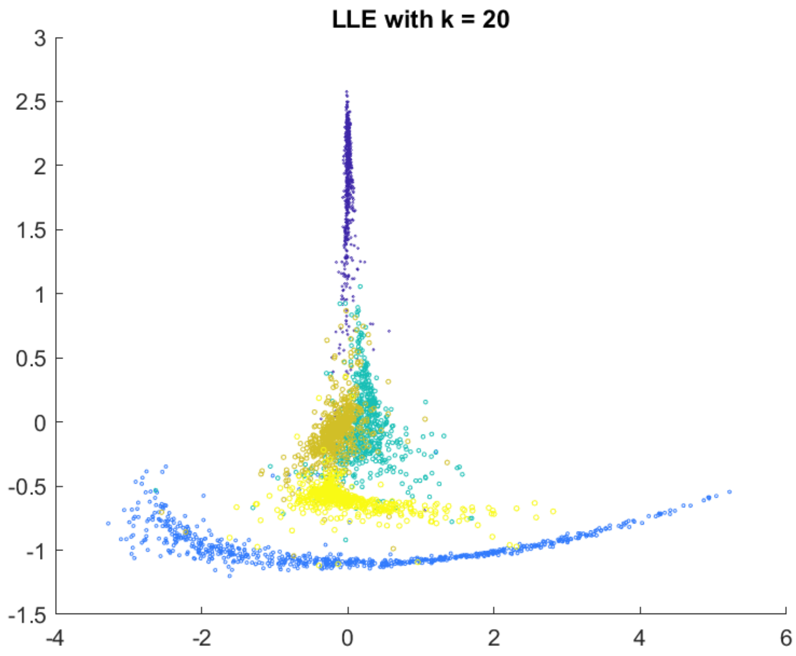


Figure 13: 2D mapped data by LLE with k=20

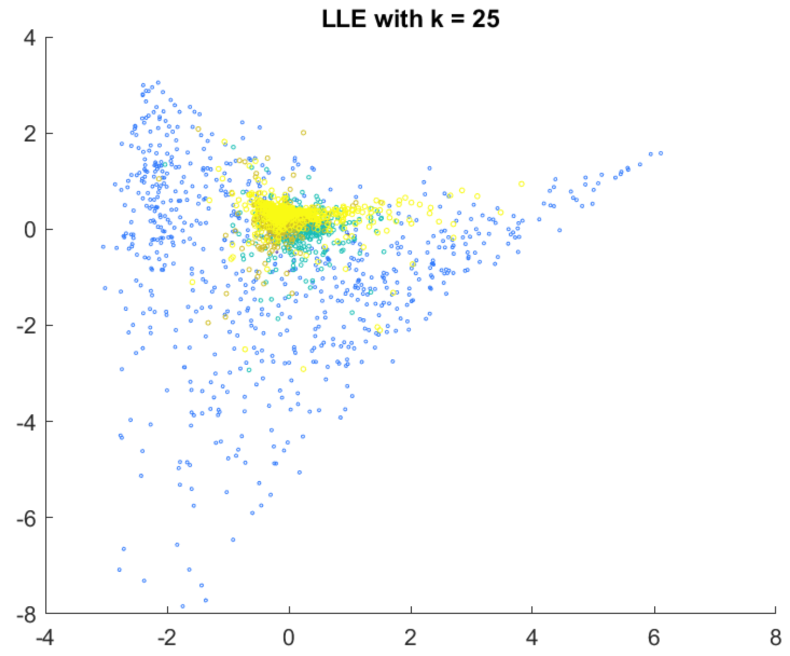


Figure 14: 2D mapped data by LLE with k=25

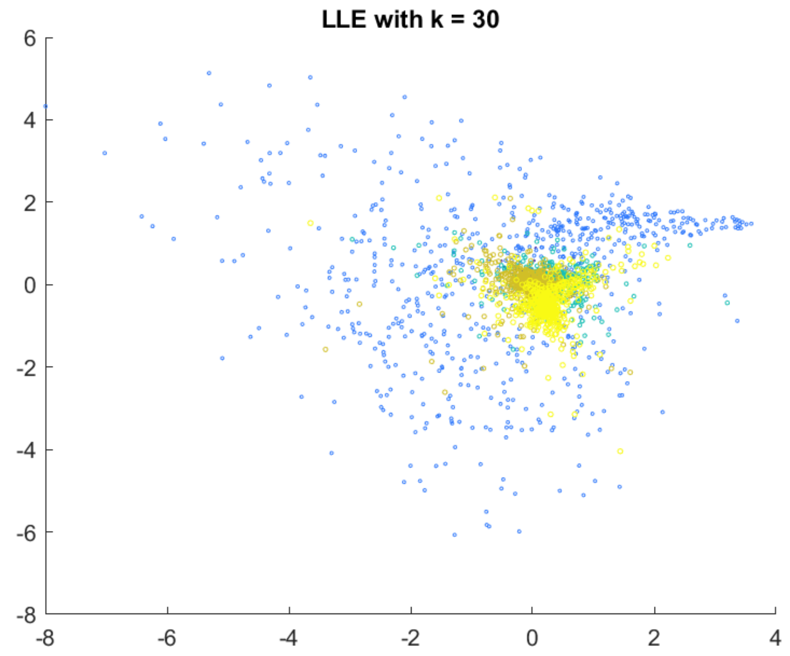


Figure 15: 2D mapped data by LLE with k=30

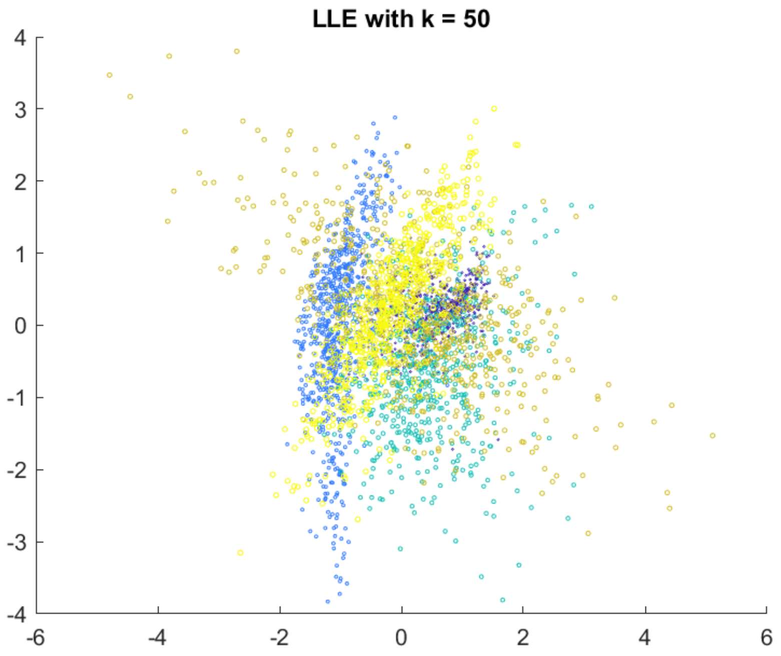


Figure 16: 2D mapped data by LLE with k=50

Question 2c

According to the author [4], the incoming test points cannot be embedded in the t-SNE map. There are two workarounds.

- Re-run t-SNE on the full dataset. But the classifier we trained on training set would not work, since the re-run t-SNE map would differ from the previous one.
- Train a multivariate regressor to predict the map location.

In this project, we choose the latter workaround. We use the KNN classifier provided by the Classification Learner Toolbox in MATLAB. And we achieve a testing accuracy of 0.983051 in the reduced model in 2 dimensional space, while it is 0.947034 in the full model in 784 dimensional space.

Question 2d

In section 2b, we experimented the t-SNE algorithm, and the LLE algorithm with various hypermaramter ks.

- The LLE requires a hyperparamter k, and the performance is quite sensitive to k. This means hypertuning is necessary, and a proper k is crucial to the success. However, the output is separable within a range from $k = 8$ to $k = 20$. Many hypervalues are available for the algorithm.

[4]Once I have a t-SNE map, how can I embed incoming test points in that map?
t-SNE learns a non-parametric mapping, which means that it does not learn an explicit function that maps data from the input space to the map. Therefore, it is not possible to embed test points in an existing map (although you could re-run t-SNE on the full dataset). A potential approach to deal with this would be to train a multivariate regressor to predict the map location from the input data. Alternatively, you could also make such a regressor minimize the t-SNE loss directly, which is what I did in this paper.

- The t-SNE has no hyperparamters. But it has some weakness in that it is impossible to embed incoming test points to a pretrained map. When a test point comes in, we have to re-train the model with the training set combined with the testing point, which is time consuming.

Question 3

- Classifier: Medium KNN (Classification Toolbox in MATLAB)
- Feature Selection by: KL Divergence
- Features to select from: 1000
- Validation: 10-fold
- Features to select: 20. We experimented from $p = 2$ to $p = 30$, and $p = 20$ has the best performance in terms of validation accuracy.

The features selected are

361	425	662	166	219
329	592	957	663	804
542	817	395	281	778
720	626	206	844	48

Table 1: Features selected

	Full features	20 selected features
Validation accuracy	0.697500	0.797500

Table 2: Comparison with 10-fold

By feature selection with KL divergence, the accuracy of Medium KNN increases from 69.75% to 79.75%. What an improvement!

Source Code

Please download the souece code from http://39.106.23.58/files/PR7_2015011506.7z

For Question 2, please cd into "Q1" directory and run "main.m".

For Question 2, please cd into "Q2" directory run "main.m". It may take *less than a minute* to train the network, but the result is reproducible because of the random seed.

For Question 3, please cd into "Q3" directory run "main.m".

For each model, I clicked "Generate code" button to transcript my operations into MATLAB codes, and stored each of them in the corresponding ".m" file. These files include:

- trainClassifierFull.m
- trainClassifierReduced.m

Thus, the steps above can be easily reproduced without using the GUI of the toolbox.