SLT coding exercise #1

Locally Linear Embedding

https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises

Due on Monday, March 6th, 2017

Zhejun Zhang
16-930-612

Contents

The Model

The model section is intended to allow you to recapitulate the essential ingredients used in Locally Linear Embedding. Write down the *necessary* equations to specify Locally Linear Embedding and and shortly explain the variables that are involved. This section should only introduce the equations, their solution should be outlined in the implementation section.
Hard limit: One page

LLE is used to reduce the dimensionality of data set. Given a set of N points $\mathbf{x}_i \in \mathbb{R}^D, i = 1, .., N$, LLE will find the corresponding points $\mathbf{y}_i \in \mathbb{R}^d, i = 1, .., N, d \ll D$ in a lower dimensional space. Basically, LLE contains the following steps.

(1) Find matrix $\mathbf{W}$ such that it satisfies

$$\sum_j w_{ij} = 1 \tag{1}$$

and minimizes

$$\varepsilon(\mathbf{W}) = \sum_i \left\| \mathbf{x}_i - \sum_{j \in \mathcal{N}(\mathbf{x}_i)} w_{ij} \mathbf{x}_j \right\|^2, \tag{2}$$

where $\mathcal{N}(\mathbf{x}_i)$ is the index set for the neighboring points of point $\mathbf{x}_i$.

(2) Find $\mathbf{y}_i \in \mathbb{R}^d, i = 1, .., N$ such that they satisfy

$$\sum_i \mathbf{y}_i = \mathbf{0} \tag{3}$$

$$\sum_i \mathbf{y}_i \mathbf{y}_i^\top = \mathbb{I} \tag{4}$$

and minimize

$$\varepsilon(\mathbf{y_1}, ..., \mathbf{y_N}) = \sum_i \left\| \mathbf{y}_i - \sum_{j \in \mathcal{N}(\mathbf{y}_i)} w_{ij} \mathbf{y}_j \right\|^2, \tag{5}$$

where w_{ij} are entries of matrix $\mathbf{W}$ founded in step (1). In principle, the neighborhood sets $\mathcal{N}(\mathbf{x}_i)$ and $\mathcal{N}(\mathbf{y}_i)$ are the same.

The Questions

This is the core section of your report, which contains the tasks for this exercise and your respective solutions. Make sure you present your results in an illustrative way by making use of graphics, plots, tables, etc. so that a reader can understand the results with a single glance. Check that your graphics have enough resolution or are vector graphics. Consider the use of GIFs when appropriate.
Hard limit: Two pages

(a) Get the data

For this exercise we will work with the MNIST data set. In order to learn more about it and download it, go to http://yann.lecun.com/exdb/mnist/.

(b) Locally linear embedding

Implement the LLE algorithm and apply it to the MNIST data set. Provide descriptive visualizations for 2D & 3D embedding spaces. Is it possible to see clusters?

(c) Cluster structure

Investigate the cluster structure of the data. Can you observe block structures in the M matrix (use matrix plots)? Also plot the singular values of M. Do you notice something? Can you think of ways to determine the optimal embedding dimension?

(d) Nearest Neighbors

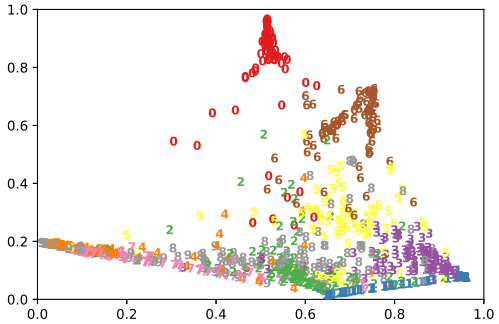
Investigate the influence of the choice of how many nearest neighbors you take into account. Additionally, try different metrics to find the nearest neighbors (we are dealing with images!).

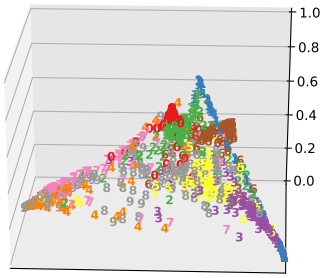
(e) Linear manifold interpolation

Assume you pick some point in the embedding space. How can you map it back to the original (high dimensional) space? Investigate how well this works for points within and outside the manifold (does it depend on the dimensionality of the embedding space?) Try things like linearly interpolating between two embedding vectors and plot the sequence of images along that line. What happens if you do that in the original space?

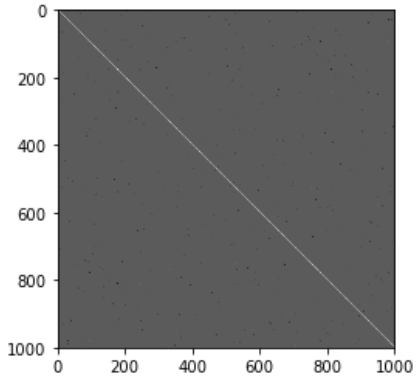
The Questions continued on next page...

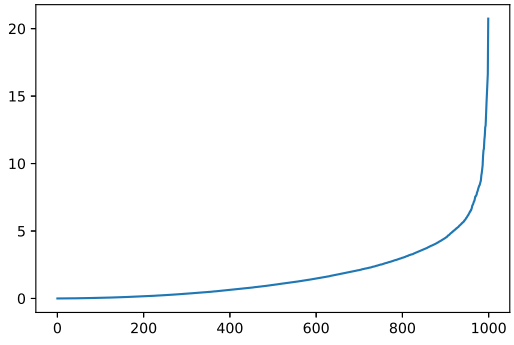
(b) The left figure demonstrates 2D embedding space. The right figure shows 3D embedding space. The data sets include 1000 images. In both figure the 1-norm and five neighborhoods are used. Clusters can be distinguished easily in both 2D and 3D embedding space. Clusters that cannot be distinguished in 2D embedding space, like 4 and 7, are possible to be separated in a 3D embedding space along the third additional direction.



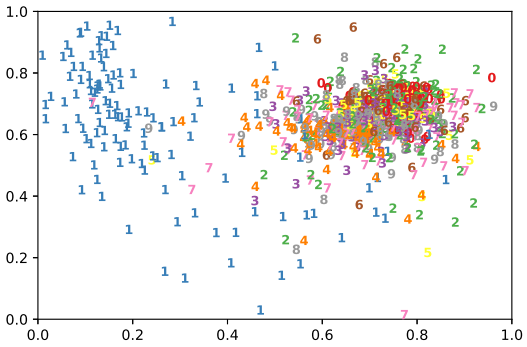
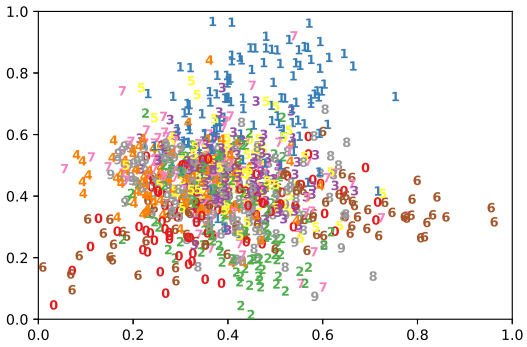


(c) It is hard to see on the image but M has a diagonal structure and is very sparse. But we cannot observe obvious block structures in the M matrix. Since M is a symmetric, positive definite matrix, its singular values are identical to its eigenvalues. The eigenvalues of M are plotted on the right figure. We notice after 800, the eigenvalue begins to increase dramatically. But most eigenvalues are small and close to zero. Greater eigenvalues lead to larger reconstruction errors and are thus less informative. This implies us to choose the optimal embedding dimension d such the dth greatest eigenvalue is still close to zero. Therefore the optimal embedding dimension should be smaller than 600.

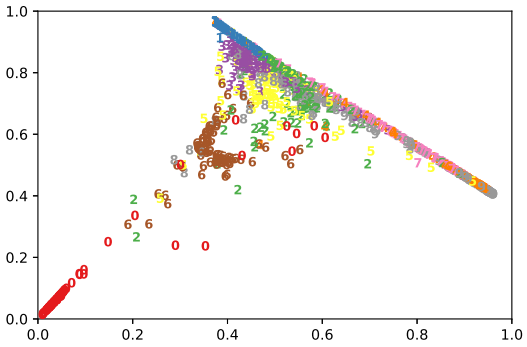
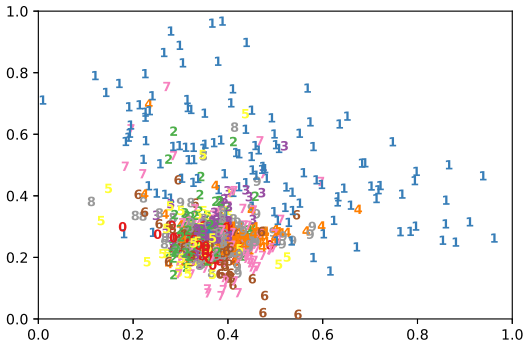




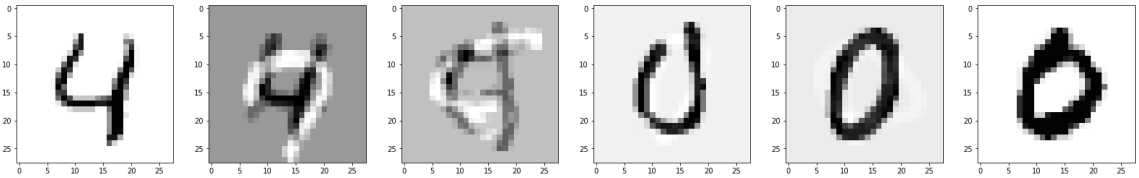
(d) The following two figures show the 2D embedding space if we use 30 neighbors (left) and 50 neighbors (right). The used metric is still 1-norm. Clusters can still be distinguished but more of them overlapped with each other. It seems for a lower dimensional embedding space, considering more neighbors in the computation doesn't necessarily make it easier to separate clusters.

Before we used 1-norm. Now we try 2-norm, the Euclidean norm. The results are shown in the following figures. Left with 5 neighbors and right with 30 neighbors. It seems for MNIST dataset, 1-norm gives rise to more distinguishable clusters. But the results of 2-norm are not meaningless. Clusters still exist.

(e) A naive approach is to use the training data as attractors. Given an arbitrary point y in the embedding space, we can find a trained point y_{train} which is nearest to y. For y_{train} we already now its inverse mapping is the image x_{train} in the higher dimensional space. Now we can let x_{train} to be the inverse mapping of the given point y. Another approach could be find two nearest neighbors $y_{train,1}$ and $y_{train,2}$ and express y as $k_1 \cdot y_{train,1} + k_2 \cdot y_{trainm,2}$. Then the corresponding inverse map of y is $k_1 \cdot x_{train,1} + k_2 \cdot x_{train,2}$. In general, we can use any interpolation method to interpolate between the training data in the embedding space.



These figures show the results of using two nearest neighbor to find inverse mapping. In above figures, the coefficients for the linear interpolation between number four and number zero are 0, 0.2, 0.4, 0.6, 0.8 and 1 respectively. From these figures we notice if we use the two nearest neighbors to find the inverse mappings of points between two embedding vectors, the results will be rather noisy. This is because the clusters are not well separated. There are many other attractors between both given embedding vectors (here 4 and 0). On the other hand, in the original space linear interpolation between two images means the outcome will slowly change from one image to another. It has a smooth fade in fade out effect.

The Implementation

In the implementation section you give a concise insight to the practical aspects of this coding exercise. It mainly mentions the optimization methods used to solve the model equations. Did you encounter numerical or efficiency problems? If yes, how did you solve them? Provide the link to your git branch of this coding exercise.
Hard limit: One page

Find solution to equation (2):
As stated in exercises sheet, the matrix $\mathbf{W}$ can be found as

$$w_{ij} = \frac{\sum_k C_{jk}^{(i)-1}}{\sum_{lk} C_{lk}^{(i)-1}}, \tag{6}$$

where $C_{jk}^{(i)} = (\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_k)$.

Find solution to equation (5):
The optimization problem defined in equation (5) is equivalent to find $\mathbf{u}_k = (y_{1k}, .., y_{Nk})$ for $k = 1, .., d$ such that they satisfy

$$\sum_i \mathbf{u}_{ki} = 0 \tag{7}$$

$$\mathbf{u}_k^\top \mathbf{u}_l = \delta_{lk} \tag{8}$$

and minimize

$$\sum_k \mathbf{u_k}^\top \mathbf{M} \mathbf{u}_k, \tag{9}$$

where $\mathbf{M} = (\mathbb{I} - \mathbf{W})^\top (\mathbb{I} - \mathbf{W})$. And the optimal solution to this problem are eigenvectors of the matrix $\mathbf{M}$ associated with the smallest $d + 1$ eigenvalues and discarding the first eigenvector associated with the eigenvalue 0.

Efficiency problem:
In the original formulation of LLE stated on the exercise sheet, both neighborhood sets $\mathcal{N}(\mathbf{x}_i)$ and $\mathcal{N}(\mathbf{y}_i)$ are the complete data set. If we are facing a large data set, this approach is clearly inappropriate because the most time consuming part of the program deals with inversing matrix C. And matrix the number of neighbors determines the size of matrix C. Therefore the number of neighbors need to be predefined as a constant number.
Another issue is find n nearest neighbors. If a naive approach is used, like build up the distance matrix between each pair of points, this part of LLE will cost a lot of time. In my implementation I have used ckdtree as suggested by someone on github. And it is very efficient.

Code: LLE is implemented based on the equations in the exercise sheet.
https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/16-930-612/1_locally_linear_embedding