SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Renfei Liu
16-949-364

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

Consider a data set of N points X_i, i = 1,..,N in a D dimensional space. The goal is to embed these points in a low d dimensional space with d ¡¡ D. We calculate the k nearest neighbors for each data point according to the Euclidean distance. And for each nearest neighbor of each data point, we intend to find the weights W_{ij} such that we could minimize the reconstrution error

$$\mathcal{E}(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2$$

. We have 2 constraints for this minimization problem: 1. The rows of the W matrix is normalized (sum to 1). 2. Each data point is only constructed by its neighbors($W_{ij} = 0$ if X_j is not a neighbor). In the end, we use d-dimensional reduced data point Y_i to represent the original D-dimensional data point X_i. This is done by minimizing the embedding cost function

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2$$

.

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?

Your Answer

After implementing the LLE algorithm, we have got the results as the graphs shown bellow: Visualize in 2D Embedding:

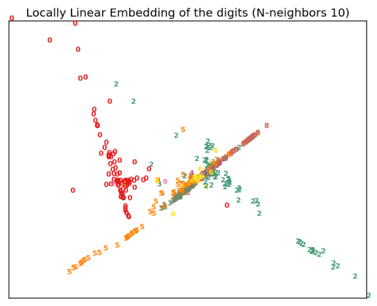

Figure 1: Plotting part of the 2D visualization of the data points

Visualize in 3D Embedding:

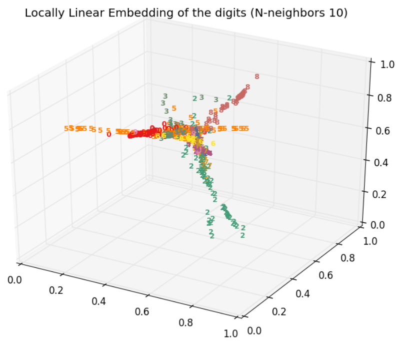

Figure 2: Plotting part of the 3D visualization of the data points

As we can observe from the graphs, there are some cluster structures among the data points (the label are shown in different colors). In both 2D and 3D visualization of the structure, it is quite obvious that the dimensionally reduced data are quite well clustered in 2D and 3D spaces. As for the M matrices of the algorithm, it is shown bellow: 2D case:

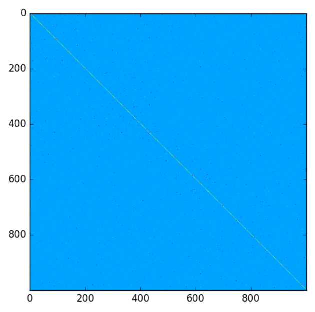

Figure 3: Plotting part of the matrix M in 2D case

3D case:

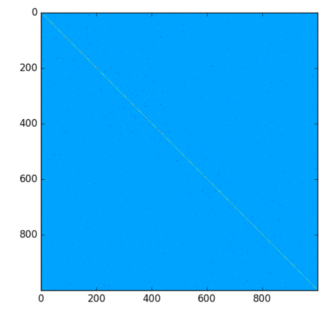

Figure 4: Plotting part of the matrix M in 3D case

We could see that the diagnal elements of the M matrices are non-zero. About the way of determining the best dimension for embedding, I think Singular Value Decomposition is a good way to try. As for the number of nearest neighbors we take into account in the algorithm, I have tried the following numbers of neighbors (apart from neighbors = 10 which has been shown above): Number of neighbors = 9:

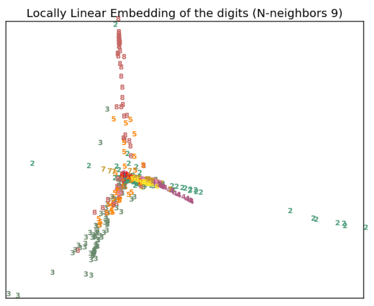

Figure 5: Plotting part of the 2D visualization while neighbors = 9

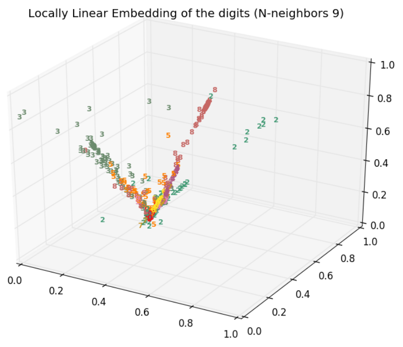

Figure 6: Plotting part of the 3D visualization while neighbors = 9

Numbers of neighbors = 8:

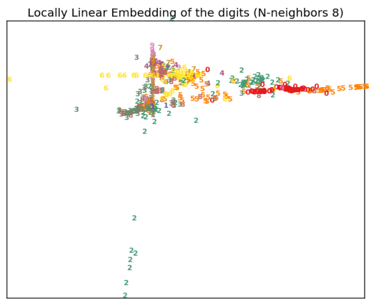

Figure 7: Plotting part of the 2D visualization while neighbors = 8

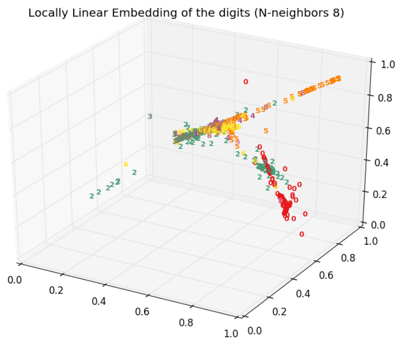

Figure 8: Plotting part of the 3D visualization while neighbors = 8

Numbers of neighbors = 7

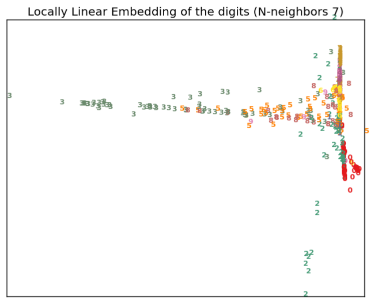

Figure 9: Plotting part of the 2D visualization while neighbors = 7

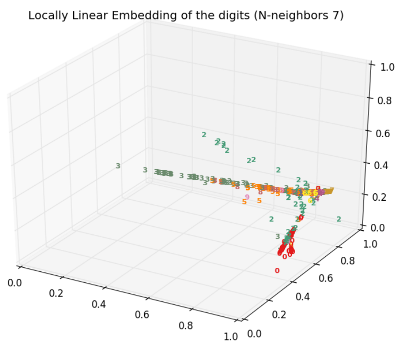

Figure 10: Plotting part of the 3D visualization while neighbors = 7

As we can see, while numbers of neighbors = 10, the result is better. And as the number of neighbors decrease, the result became worse.
And as for the metrics of measuring the distance among points, we could also use Chebyshev distance to measure it. As for linear manifold interpolation, my idea is that the nearest neighbors of a data point in the reduced space is also the nearest neighbors in the original space. And the structure of the data point is similar in the 2 spaces. Just assume that they are the same, and assume that each point is the only unknown point in the original space. Iterate the process and we will get the mapped data points in the original space.

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

Your Answer

In the coding exercise, I firstly tried the python function manifold.LocallyLinearEmbedding to get a initial result the the whole algorithm. Then I implemented the LLE algorithm myself toe compare the results. In the process, I found out that if I take the whole dataset into account, it takes forever to run the algorithm. So I just took a subset of the dataset to implement the algorithm.
And here I will show some results from the python library function from sklearn:

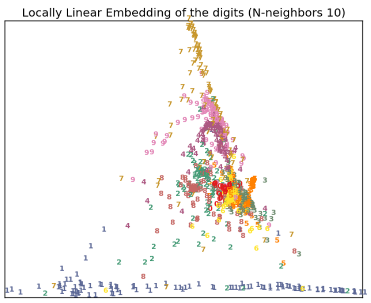

Figure 11: Plotting part of 10 neighbors plot

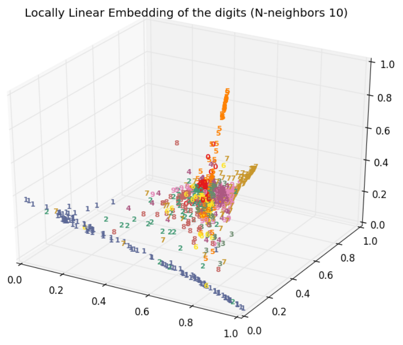

Figure 12: Plotting part of 10 neighbors plot (in 3D)

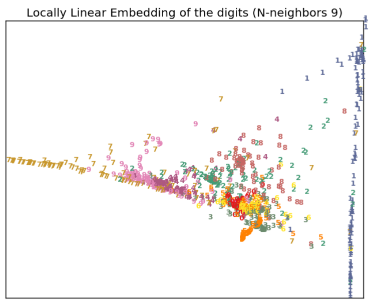

Figure 13: Plotting part of 9 neighbors plot

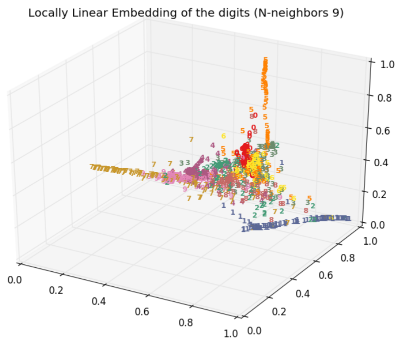

Figure 14: Plotting part of 9 neighbors plot (in 3D)

Your Page

Your page gives you space to include ideas, observations and results which do not fall into the categories provided by us. You can also use it as an appendix to include things which did not have space in the other sections.
No page limit.

Your Answer
16-949-364/1$_locally_linear_embedding$