SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Leonard Adolphs
16-947-699

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

In Locally Linear Embeddingwe reduce the dimensionality of the input vectors by finding a new embedding space with d < D. The algorithm is performed in three steps [a]:

1. Find for each vector X_i in the input the k nearest neighbors
2. Compute the weights W_{ij} that best reconstruct X_i from its neighbors. We achieve this by minimizing the recontruction error:

$$\mathcal{E}(W) = \sum_i \left| X_i - \sum W_{ij} X_j \right| \tag{1}$$

With the constraints:

$$\sum_j W_{ij} = 1 \tag{2}$$

$$W_{ij} = 0 \quad \text{if } X_j \text{ is not a neighbor of } X_i \tag{3}$$

3. Fix the weights W and find the embedding vectors Y by minimizing the following objective:

$$\Phi(Y) = \sum_i \left| Y_i - \sum W_{ij} Y_j \right| \tag{4}$$

$$= \sum_{ij} M_{ij} (Y_i \cdot Y_j) \tag{5}$$

With the constraints:

$$\sum_i Y_i = 0 \tag{6}$$

$$\frac{1}{N} \sum_i Y_i Y_i^T \tag{7}$$

[a] An Introduction to Locally Linear Embedding by Saul et al.

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!).

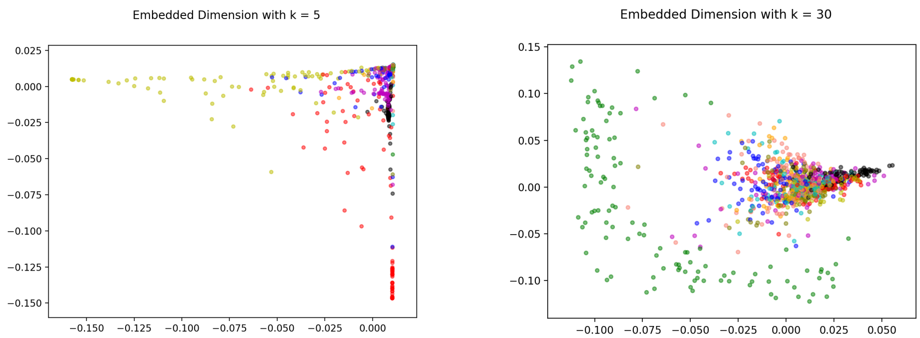
(e) Linear manifold interpolation

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) Locally linear embedding

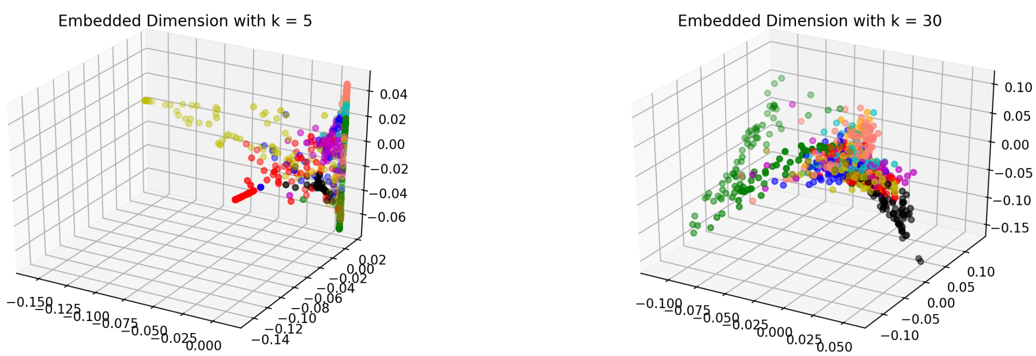
I fixed the number of nearest neighbors to either $k = 5$ or $k = 30$. In figure 1 you can see the results for the 2D embedded vector space and in figure 2 for the 3D case. The colors in the plot represent the labels of the images, i.e. different digits. We can see, that the cluster structure becomes better with $k = 30$ and the higher dimension, but still is not perfectly seperable. Neither for the 2D nor for the 3D case.



(a) k= 5 Nearest Neighbours

(b) k= 30 Nearest Neighbours

Figure 1: Embedding space with dimensionality 2.



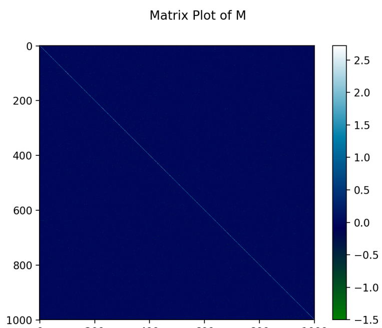
(a) k= 5 Nearest Neighbours

(b) k= 30 Nearest Neighbours

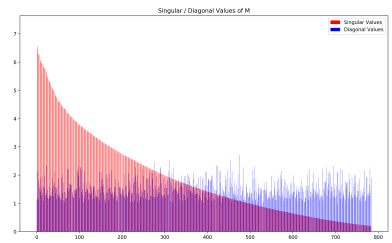
Figure 2: Embedding space with dimensionality 3.

(c) Cluster structure

To investigate the structure of the embedded vectors I plotted the values of the matrix plot in figure 3a. As we can see in this plot the matrix is really sparse. The important (non-zero) values are on the diagonal of M. In figure 3b I plotted the diagonal values from M, as well as its singular values. The closer an eigenvector is to zero, the more information we gain if we add the corresponding eigenvector as an embedded vector. So to obtain the best embedding dimension we should only add a vector, if its corresponding eigenvalue is almost zero.



(a) Matrix plot of $M = (I - W)^T(I - W)$



(b) Comparison between singular and diagonal values of M.

Figure 3: Properties of the Matrix $M = (I - W)^T(I - W)$.

(d) Nearest Neighbors

In the figures 4 and 5 I plotted the embedded space for different neighbor values k = [2, 30, 100] and different distance metrics m = [Euclidean, Manhattan], respectively. For both metrics we can see that two neighbours are too few to obtain a good clustering. The change in metric gives back different clustering results, since the vectors have probably different neighbors for the different properties.

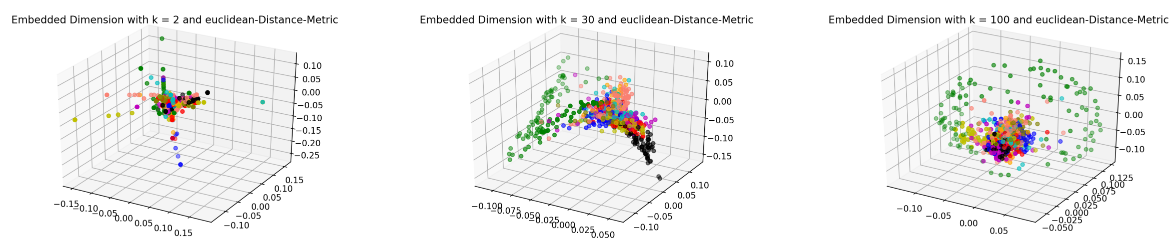


Figure 4: Embedded vectors for k = 2, 30, 100 and *Euclidean* distance-metric

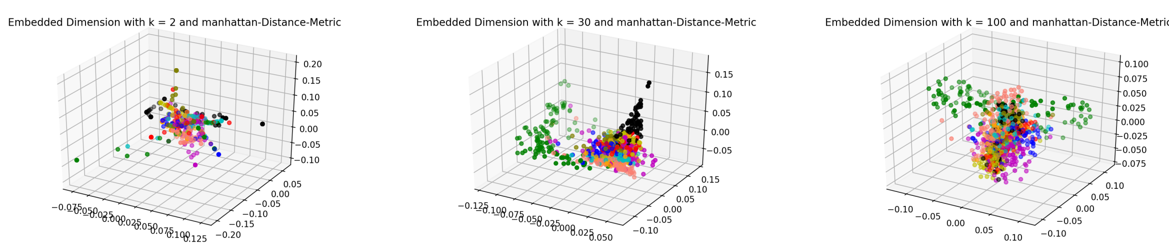


Figure 5: Embedded vectors for k = 2, 30, 100 and *Manhattan* distance-metric

(e) Linear manifold interpolation

We are able to reconstruct the original vector by the embedded vector and the knowledge about the number of neighbors k we used. To reconstruct X_i from the embedded vector Y_i we need to execute the following steps:

1. Find the k nearest neighbors of Y_i in the embedded space.
2. Let V be the $k \times d$ (d is dimension of embedded space) matrix corresponding to the embedded vectors of the nearest neighbors.
3. Solve the following equation for w:

$$V^T w = Y_i \tag{8}$$

4. Let S be the $k \times D$ (D is dimension of original space) matrix corresponding to the original vectors of the nearest neighbors. Compute the reconstructed vector X_{rec} as:

$$X_{rec} = w^T S \tag{9}$$

In figure 6 we can see the difference between the reconstructed image (first row) and the original image (second row). The columns represent different pi values for the interpolation between two vectors. So, the first and the last column show a vector which is assigned to class '0' and '7' only. The columns inbetween are linear interpolations between these two vectors.
The plot shows, that the reconstruction of the vector from the small (3D) to the original space (784D) is possible without much loss of information.

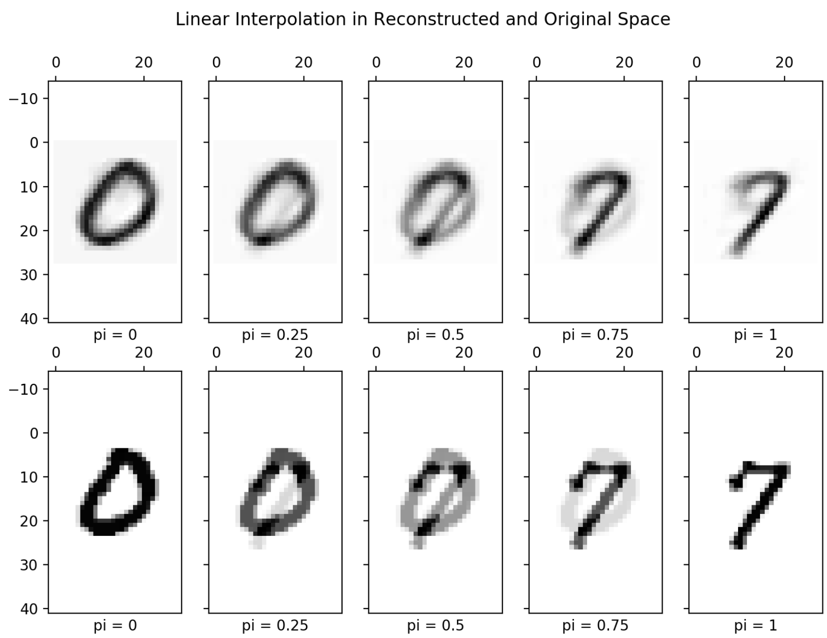


Figure 6: Interpolation between two random vectors ('0' to '7') in the reconstructed and the original space, respectively.

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

To implement my solution of LLE, I used big parts of the manifold library provided by scikit-learn[a]. If the number of input images is too large we encounter efficiency problems, since the runtime complexity is dependant on N^2. For < 1500 input images this is not a problem, so I chose $N = 1000$ for the experiments. My code for this exercise is in the *mnist.py* file. To see the plots for the different sections you just have to uncomment the according line.

[a] http://scikit-learn.org/stable/modules/manifold.html

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.

My Git Branch: 16-947-699/1_locally_linear_embedding