SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Julien Lamour
13-820-907

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

Cost function for reconstruction errors:

$$\mathcal{E}(W) = \sum_i \left| \vec{X_i} - \sum_j W_{ij} \vec{X_j} \right|^2 \tag{1}$$

where $\vec{X_i}$ is the point to be reconstructed from a weighted combination ($W_{ij}$for$j = 1...n$) of its neighbors $\vec{X_j}$. $\sum_{ij} W_{ij} = 1$ and $W_{ij} = 0$ if $\vec{X_i}$ and $\vec{X_j}$ are not neighbors.

Cost function for low dimensional coordinates:

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

where W_{ij} are the exact same weights are described earlier and $\vec{Y_i}$ is the lower dimensional vector corresponding to $\vec{X_i}$ in the initial dimension.

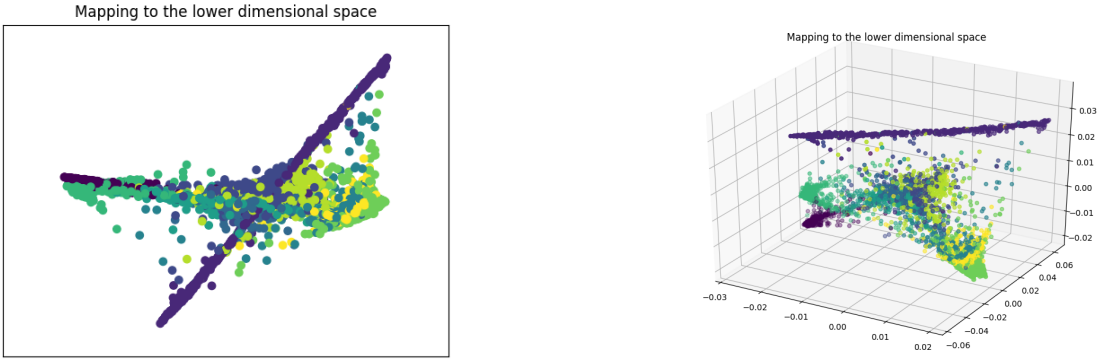
The Questions

(a) Get the data

I used a subset of the training set of size $N = 5000$ for the exercise in order to get acceptable runtime. I made a few tests with the whole dataset ($N = 60000$) but the results were similar.

(b) Locally linear embedding

We can observe that both in 2D and 3D it is possible to see clusters in the data. Since the influence on the number of neighbors is studied in a later question, these plots were done using the 10 nearest neighbors for each points.

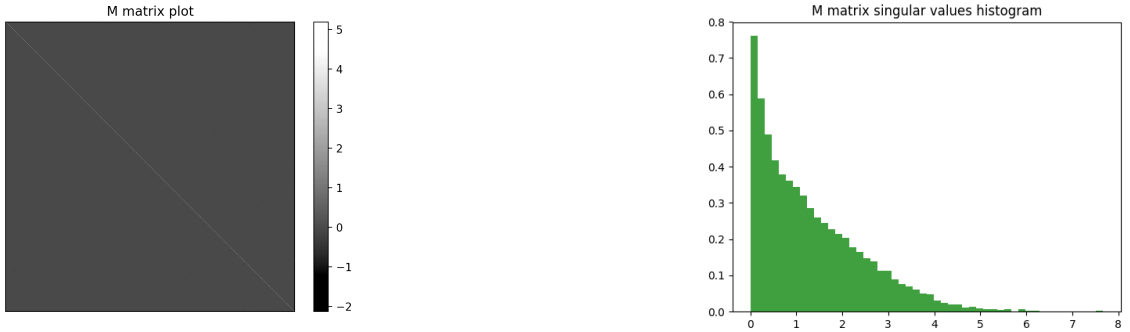


(a) 2D **(b)** 3D

Figure 1: Visualizations for 2D (a) and 3D (b-d) embedding spaces.

(c) Cluster structure

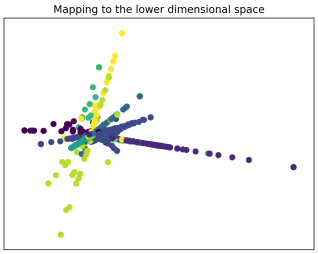
As we can observe on the following plots, the M matrix has values very close to zero everywhere but in the diagonal and singular values are concentrated close to 0. Since the optimal embedding is found by computing the botton $d + 1$ eigenvectors of the matrix M, one could start from 1D and stop adding eigenvectors (i.e. increasing the embedding dimension) until there is no significant gain in the reconstruction error.



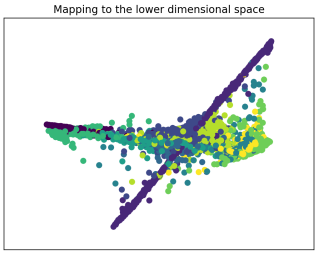
(a) M Matrix plot **(b)** Histogram for M matrix singular values

Figure 2: Cluster structure of the data.

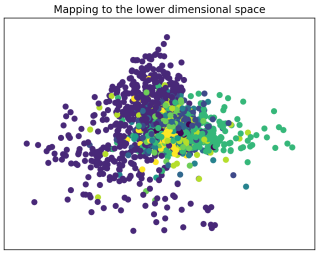
(d) Nearest Neighbors



(a) 3 neighbors

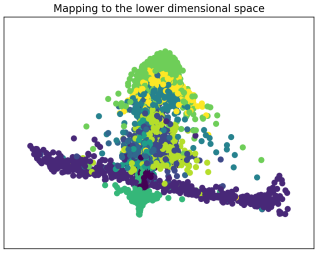


(b) 10 neighbors

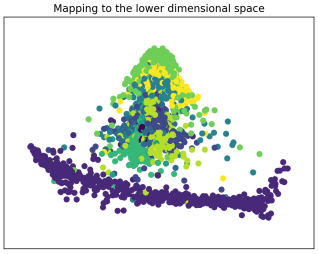


(c) 35 neighbors

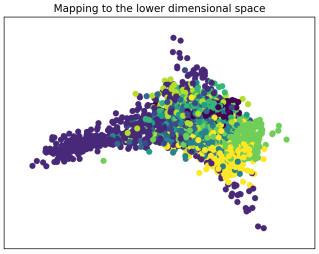
Figure 3: Influence of the number of neighbors k (using Euclidean norm).



(a) L-1 norm



(b) L-2 norm



(c) L-∞ norm = $max(abs(x))$

Figure 4: Usage of different norms with fixed number of neighbors (15).

As we can observe of the above plots, the number of neighbors k strongly influences the mapping in the lower dimensional space. Literature shows that it is usually common to pick k by cross validation. Different norms also leads to different results, which can be trivially explained by the facts that different neighbors are going to be picked, but it has less influence than the number of neighbors.

(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 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

I used Numpy for system solving as well as for eigenvalues computation, which is already optimized and therefore didn't lead to numerical neither efficiency problems. I used Matplotlib for plotting. I could have used a library for faster nearest-neighbors computation but found it more interesting to have it written by myself since it is fairly simple and makes the code more readable.

Git branch: `https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/13-820-907/1_locally_linear_embedding`

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
https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/13-820-907/1_locally_linear_embedding