SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Sveinn Plsson
16-931-149

Contents

The Model

Suppose the data consists of N real valued vectors X_i, each of dimensionality D. For each data point we find its K nearest neighbors as measured by Euclidean distance. For each data point X_i we want to find weights W_{ij} for each of the K nearest neighbor such that we minimize the reconstruction error function

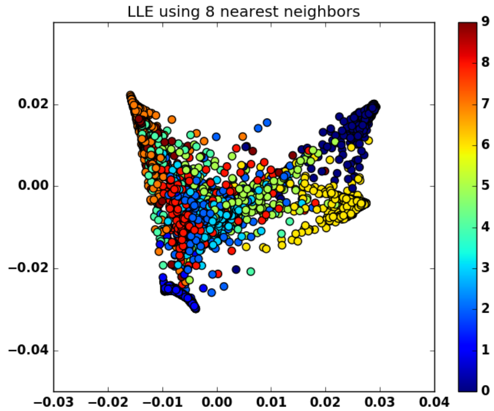
$$\mathcal{E}(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2$$

subject to two constraints: first, that each data point is reconstructed only from its neigbors, enforcing $W_{ij} = 0$ if X_j does not belong to this set; second, that the rows of the weight matrix sum to one.
We choose a $d << D$ and find d dimensional coordinates Y_i for each data point X_i. We do this by minimizing the embedding cost function

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

The Questions

Answer b: Yes, some clusters appear. See examples in figure 1 below.



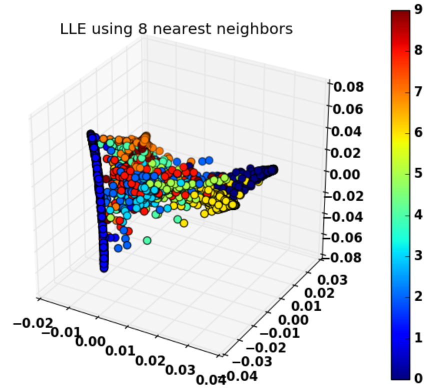


Figure 1: There are some visible clusters in the results.

Answer c: We can see that the diagonal elements are all non-zero.

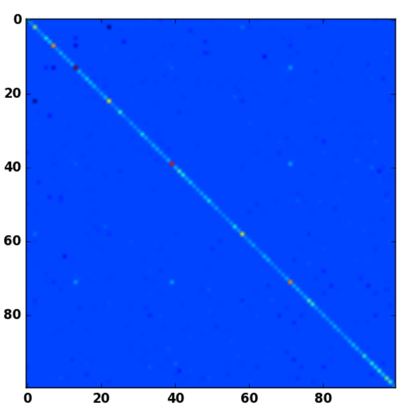


Figure 2: Plotting part of the matrix M

Answer d: When we change K, the number of nearest neighbors we can see some differences. If we only use 1 neighbor for each data point we can see that no visible clusters form (figure 3). Then as we increase the number of nearest neighbors we start seeing clusters even with K=3 (figure 4). The clusters seem to become more clear when we increase K.

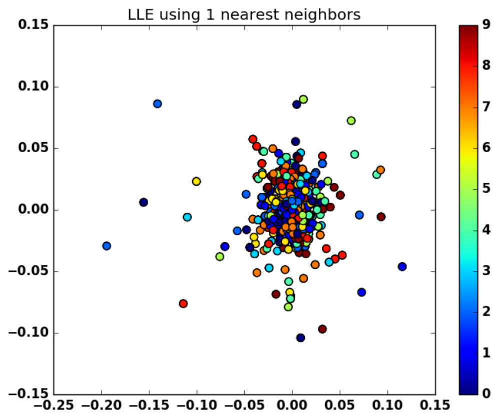


Figure 3: Using only 1 nearest neighbor

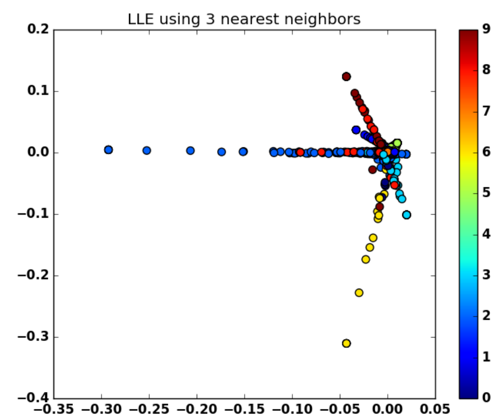


Figure 4: Using 3 nearest neighbor

Figures 5,6 and 7 show the use of 7 nearest neighbors with Euclidean, Hamming and Chebyshev distance metrics respectively.

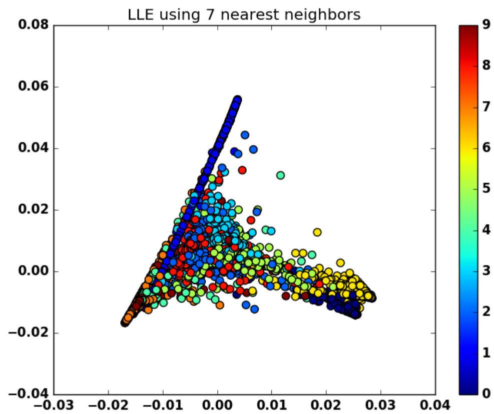


Figure 5: K=7, Euclidean distance

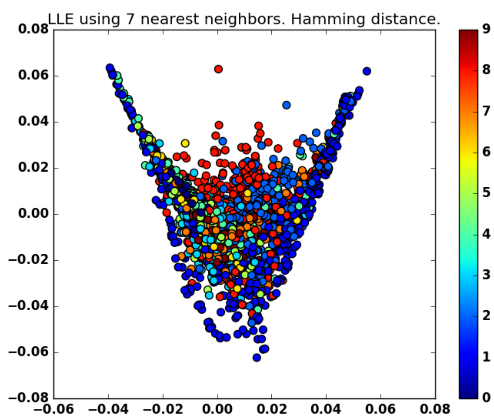


Figure 6: K=7, Hamming distance

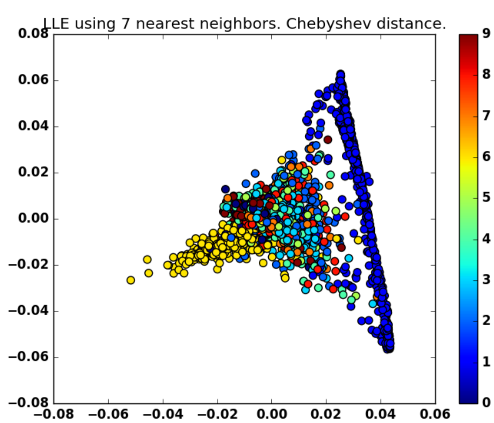


Figure 7: K=7, Chebyshev distance

Answer e: One method would be to find the point's K nearest neighbors in the embedding space and find the reconstruction weights as in the high dimensional space. Then use these K nearest neighbors and weights in the high dimensional space to reconstruct the data-point. This depends highly on the dimensionality of the embedding space, it works better for higher dimensions. See figure 8 where a data point is reconstructed using this method. The figures also show the original image for comparison.

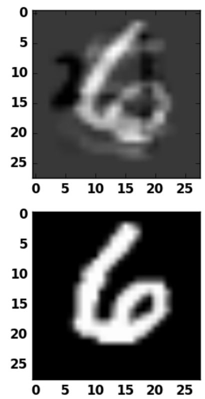


Figure 8: Reconstruction of a data point. Upper image shows the reconstruction, the lower image shows the original

The Implementation

The implementation involves finding eigenvectors of a very large matrix. For the MNIST dataset this could mean a 60000 by 60000 matrix which my implementation would take a very long time to compute. When answering the questions in the previous questions I used only 5000 samples which my implementation of LLE took only about 10 seconds to solve. Link to my git branch: https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/16-931-149/1_locally_linear_embedding

Your Page

https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/16-931-149/1_locally_linear_embedding