SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Lucio Fernandez-Arjona
16-741-928

Contents

The Model

In LLE the original data points are reconstructed by a linear combination, given by the weight matrix W_{ij}, of its neighbors. The reconstruction error is given by the cost function E(W).

$$E(W) = \sum_i |\mathbf{X}_i - \sum_j \mathbf{W}_{ij}\mathbf{X}_j|^2$$

The weights W_{ij} refer to the contribution the point X_j has while reconstructing the point Xi. The cost function is minimized under two constraints: (a) Each data point Xi is reconstructed only from its neighbors, thus enforcing W_{ij} to be zero if point X_j is not a neighbor of the point Xi and (b) The sum of every row of the weight matrix equals 1.
$\sum_j \mathbf{W}_{ij} = 1$
The original data points are collected in a D dimensional space and the goal of the algorithm is to reduce the dimensionality to d such that $D >> d$. The same weights W_{ij} that reconstruct the ith data point in the D dimensional space will be used to reconstruct the same point in the lower d dimensional space. Each point X_i in the D dimensional space is mapped onto a point Y_i in the d dimensional space by minimizing the cost function

$$\Phi(Y) = \sum_i |\mathbf{Y}_i - \sum_j \mathbf{W}_{ij}\mathbf{Y}_j|^2$$

In this cost function, unlike the previous one, the weights W_{ij} are kept fixed and the minimization is done on the points Y_i to optimize the coordinates. This minimization problem can be solved by re-expressing $\Phi(Y)$ as

$$\Phi(Y) = \sum_{ji} \mathbf{M}_{ij}(\mathbf{Y_i Y_j})$$

with

$$\mathbf{M}_{ij} = \delta_{ij} - \mathbf{W}_{ij} - \mathbf{W}_{ji} + \sum_k \mathbf{W}_{ki}\mathbf{W}_{kj}$$

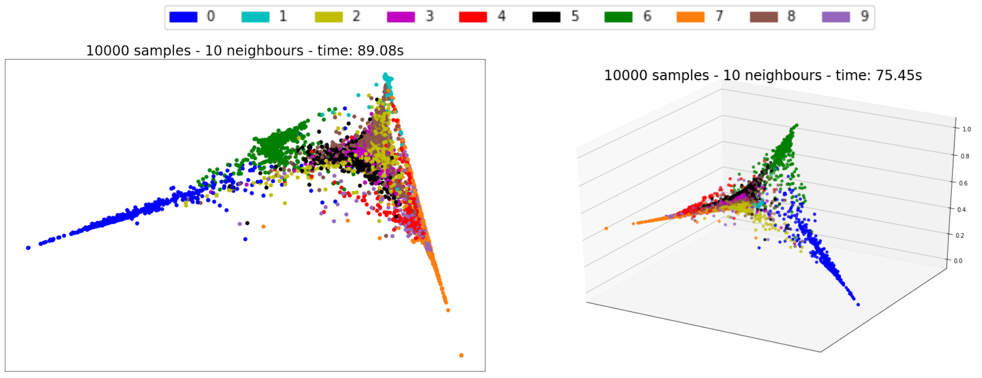
Taking the d+1 eigenvectors of $\mathbf{M}$ associated with the smallest d+1 eigenvalues and dropping the one associated with the smallest, we obtain the d eigenvectors that provide the coordinates Y_i that minimize $\Phi(Y)$

The Questions

Hard limit: Two pages

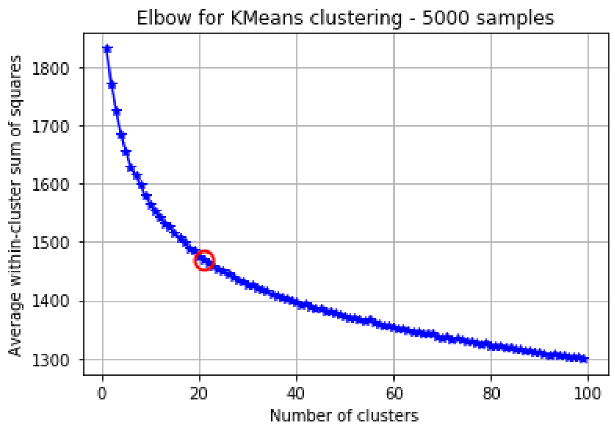
(b) Locally linear embedding

For this exercise we apply the LLE algorithm to the MNIST dataset. As the dataset is quite large, I have worked with subsets of the data to shorten runtimes. After applying the algorithm, it is possible to see clear clusters in the low-dimensional representation of the data, corresponding to the different digits. Not all digits stand out in the same way. "0" (blue) is very clear in many of charts below for example.



(c) Cluster structure

The data in its original high-dimensional space seems to present a certain number of clusters but none is massively dominant. Using the elbow method one could choose about 20 clusters.



There are no obvious block structures in M. The chart below shows M for 100 samples. M is sparse, with 3,934 non-zero values out a total of 10,000. The only visible values are those in the diagonal, the rest are evenly distributed as seen in Figure 1.

The Questions [(c) Cluster structure] continued on next page. . .

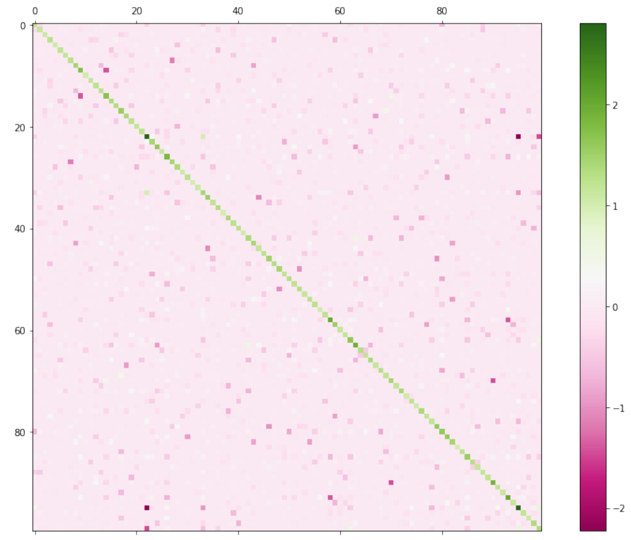
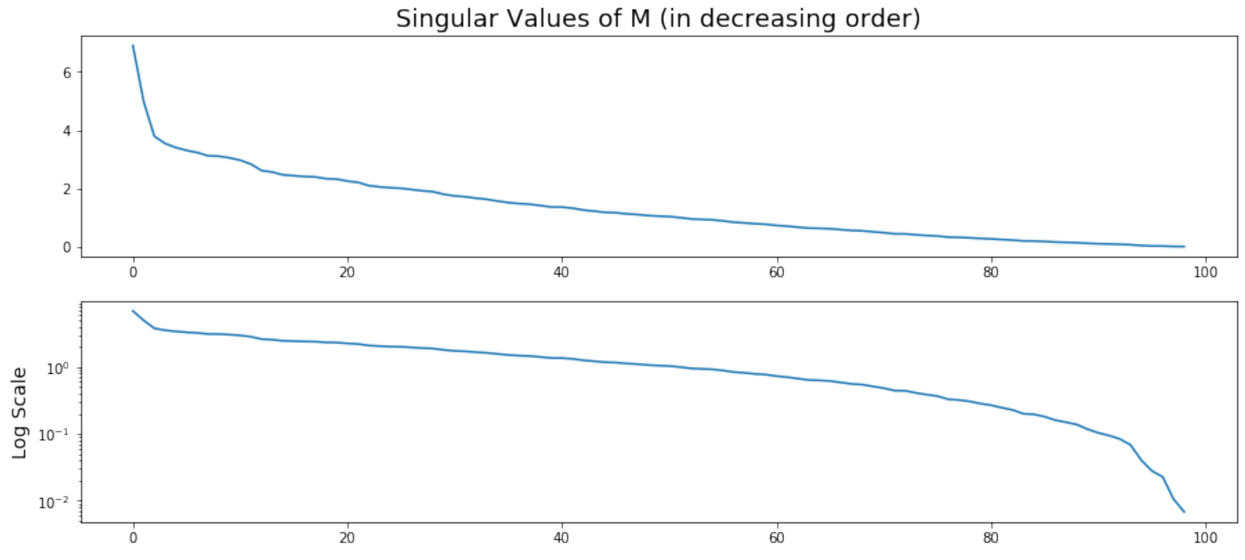


Figure 1: Plot of M matrix

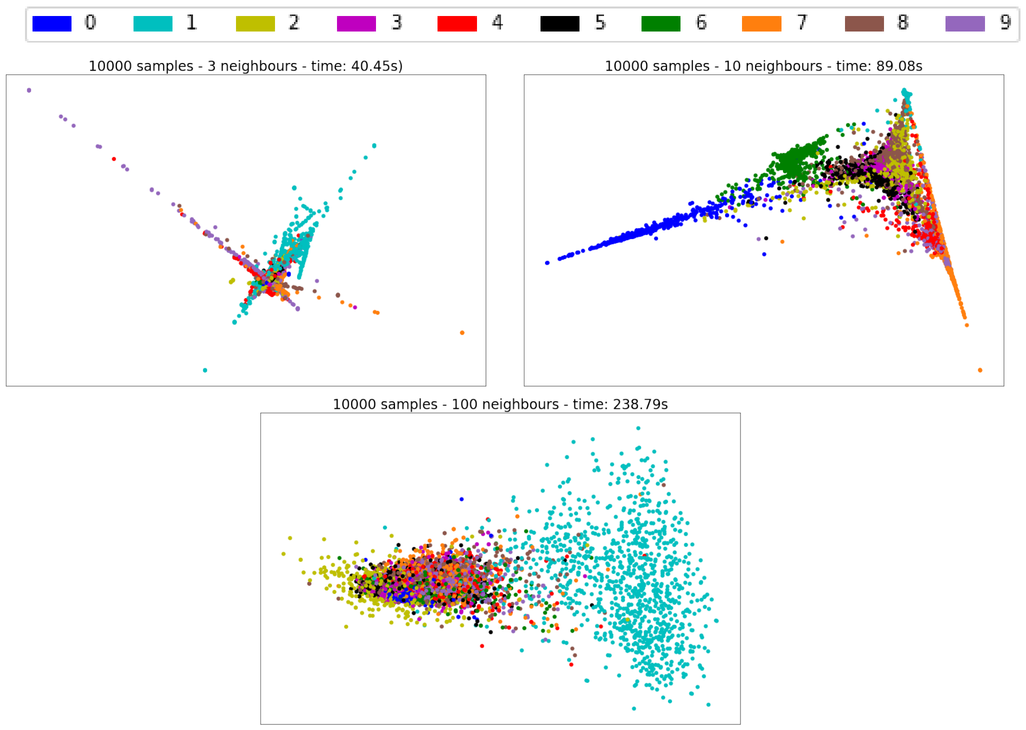
The following plot shows the singular values of M (ignoring the smallest one) in linear and logarithmic scales. I would not be able to determine an optimal embedding, as it is not clear to me what optimal would be in this context. In cases like PCA, optimality is determined by the minimization of the reconstruction error from the low-dimensional $\hat{X}$ to the high-dimensional X. In the case of LLE, I am not sure such a metric would be possible (due to the selection of a limited number of neighbours, $k << n$). Intuitively, one possible candidate is the inflexion point of the singular value curve in log scale around the 8th smallest singular value. It would make sense if there were between 5 and 10 basic operations that determine the manifold dimensions.



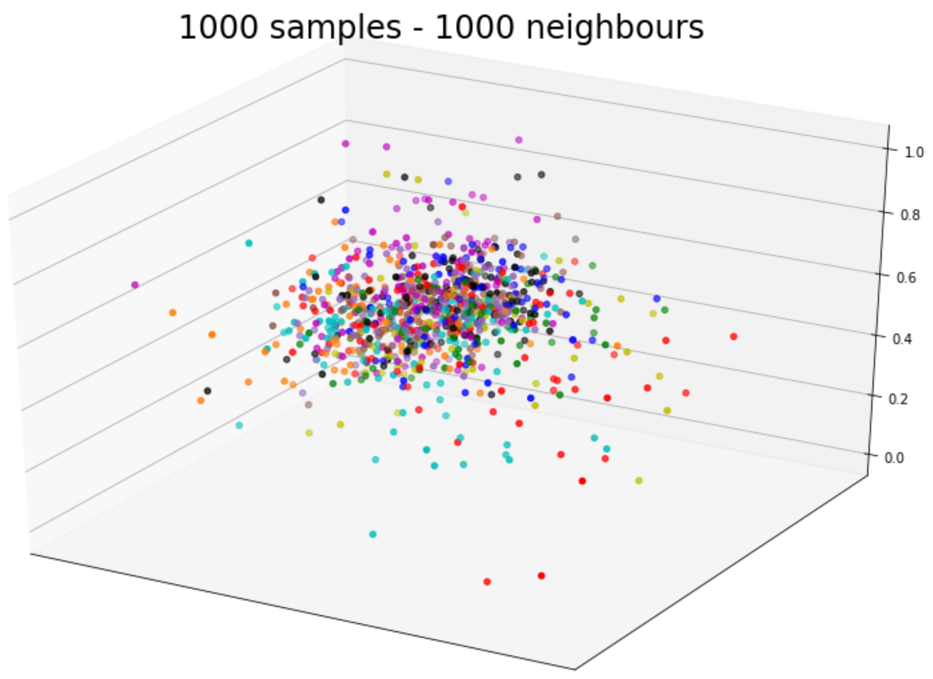
(d) Nearest Neighbors

A low number of neighbors leads to sharp edges on the manifold, while higher numbers lead to rounder shapes. The following figure shows results for 3, 10 and 100 neighbors.

The Questions [(d) Nearest Neighbors] continued on next page. . .



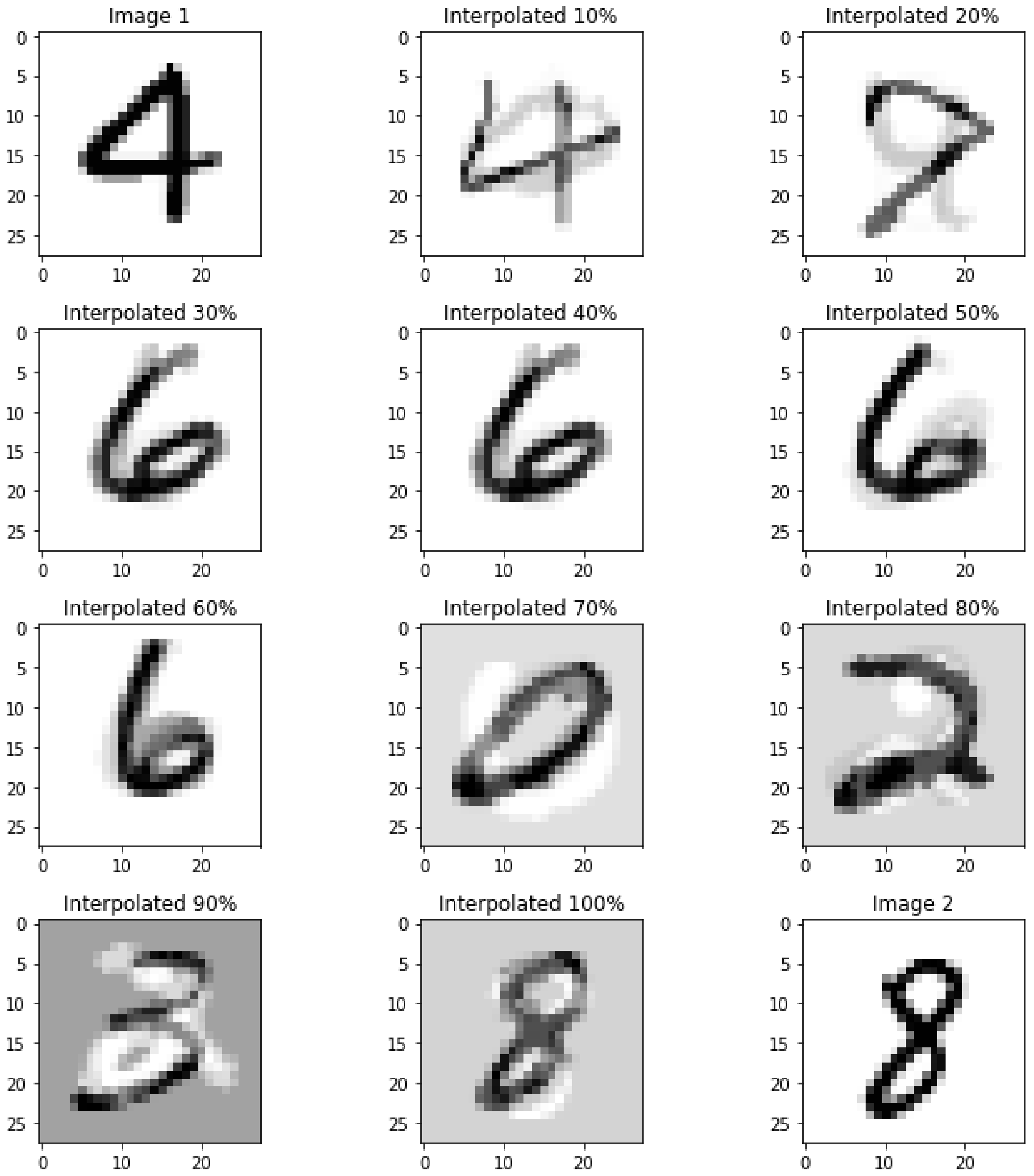
The following is the result using the mutual information between the vectors as the distance. There does not seem to be any clustering which makes me think there is an error in the implementation (which I have not been able to find).



Considering that the inputs are images, I think a measure of similarity that treats nearby pixels differently than far away pixels would be beneficial. Maybe using a 2-D convolution of the image to keep a measure of locality in the pixels, ie if I had three images with one black pixel each (and the rest all white), I would like the distance to be proportional to the cartesian distance of the pixels in the image.

(e) Linear manifold interpolation

To map back a point Y^* from the embedding space to the original space, one should find the closest neighbor and from the original W matrix choose d neighbors of that point (d being the embedding dimension) of such point. With those d neighbors calculate the weights that minimize the reconstruction error of Y^* in the embedding space and apply those weights in the original space. The following is the sequence of the convex combinations (in intervals of 0.1) between two images.



The results are different if one interpolates in the original space:

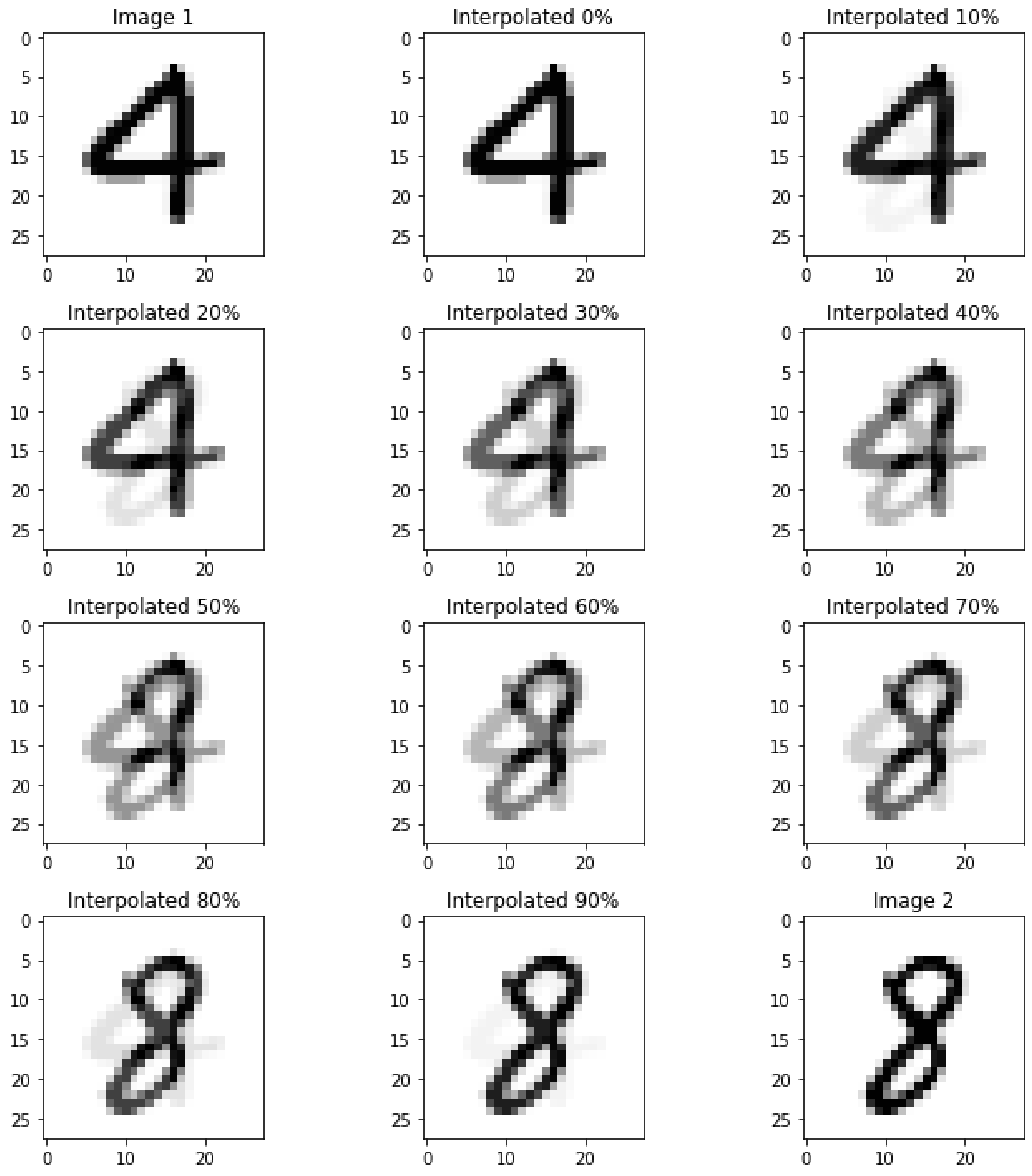


Image 1
Interpolated 0%
Interpolated 10%
Interpolated 20%
Interpolated 30%
Interpolated 40%
Interpolated 50%
Interpolated 60%
Interpolated 70%
Interpolated 80%
Interpolated 90%
Image 2
0
5
10
15
20
25
0
10
20

The Implementation

https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/16-741-928/1$_locally_linear_embedding$
LLE exercise-Copy1.ipynb .
Hard limit: One page

One of the biggest problems was the size of the dataset and the complexity of the algorithm $((O[D\log(k)N\log(N)]+O[DNk^3]+O[dN^2])$. To solve the problem, I used a subset of the data (between 100 and 10,000). The number of nearest neighbors is the most sensitivity parameter (k^3 term in the formula above) so I did not run with more than 100 neighbors.
There was a problem with the covariance matrix being singular which is extremely common with large covariance matrices. I solved it in the way described in the reference paper (applying a small delta to the matrix).
Another problem was that my implementation had a bug I could not find and had to construct a new version from the Scipy implementation to be able to read the matrix M and change the distance measure.

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 16-741-928//$1_locally_linear_embedding$