SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Michiels Joran
16-948-549

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 Embedding the goal is to represent a data set of N points $\mathbf{x}_i \in \Re^D$ in a low dimensional space $\Re^d$ ($\mathbf{y}_i \in \Re^d$) with $d \ll D$. It is assumed that the data points (dimension D) lie (approximately) on a manifold of dim d.

First the nearest neighbours for every $\mathbf{x}_i$ have to be found (denoted by $\mathbf{x}_j$). Out of these points, $\mathbf{x}_i$ can be approximately reconstructed. The best weights w_{ij} can be found by minimizing the folowing error:

$$E(\mathbf{W}) = \sum_i \|\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j\|$$

under the constraints $\sum_j w_{ij} = 1$. w_{ij} is zero when $\mathbf{x}_j$ is not a nearest neighbour of $\mathbf{x}_i$.

Note that these weights are invariant to rotations, rescalings and translations of the data point and it's neighbours. These weights represent the local geometry of the data space. When the data (approximately) lies on a manifold of dimension d, it is expected that the local geometry of small patches will be equal to the local geometry of the original data space. Therefore the weights w_{ij} can be reused.

The points $\mathbf{y}_i \in \Re^d$ can then be found by minimizing the cost function:

$$E(\mathbf{Y}) = \sum_i \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|.$$

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!).

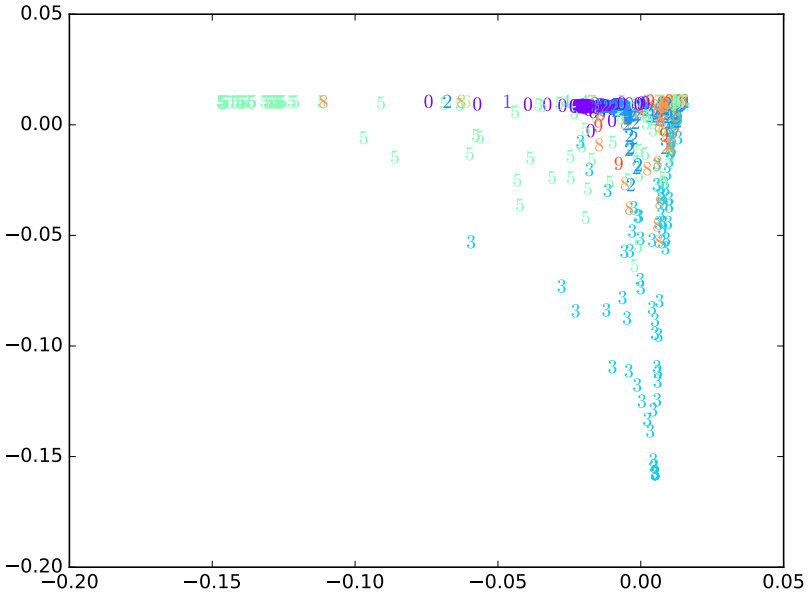
(e) Linear manifold interpolation

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?

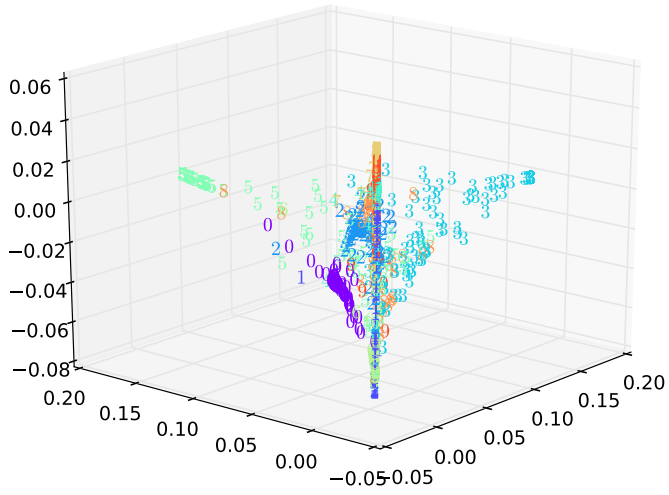
The Questions continued on next page...

(b) Locally linear embedding

Scatter plot of 1000 data points in a 2D embedding space (5 neighbours, Euclidean distance)



Scatter plot of 1000 data points in a 3D embedding space (5 neighbours, Euclidean distance)



The above results are for 5 neighbours using the Euclidean distance. It is possible to see a bit of clustering, although it is easier for some numbers (labels) than for others (e.g. the number 5 vs 1). In the 3D embedding space some clusters are better to see (e.g. the number 0).

(c) Cluster structure

The structure of the matrix M is symmetric with a dominant diagonal. This follows from its construction:

$$M = (I - W)^T(I - W) \iff M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki}w_{kj}$$

where δ_{ij} is 1 if $i = j$ and 0 otherwise. The matrix is also very sparse.

The eigenvalues, (or singular values, they are the same for this type of matrix) are all positive since M is a positive semi-definite matrix. One eigenvalue is equal to zero (or equal to machine precision when it is computed). The eigenvector corresponding with this eigenvalue is the unit vector with all equal components. The plot of eigenvalues is not shown because it didn't seem interesting.

When dealing with a supervised problem one can choose the dimension of the embedding space that gives the lowest test error.

(d) Nearest Neighbors

As can be seen in the pictures in the appendix, taking more neighbours seems to increase the spread of the data points in the embedding space (it is less concentrated around a particular region). Since we are dealing with images, it might be better to work with extracted features for the image, instead of the complete image. When working with the complete image, the image first has to be flattenend to a 1D-array before it can be fed into the LLE-algorithm. Then we can use different distance metrics like the Euclidean or Manhattan distance. The Manhattan distance gives very different results (see appendix).

(e) Linear manifold interpolation

A intiutive approach would be to try to describe that point (let's call it $\mathbf{y}_t$) with the help of nearby points (of which you know the corresponding points in the D-dimensional space). In a 2D-space this can be done by representing $\mathbf{y}_t$ by two linearly independent other points $\mathbf{y}_s$ and $\mathbf{y}_r$ (preferably the 2 nearest neighbours):

$$\mathbf{y}_t = a * \mathbf{y}_s + b * \mathbf{y}_r.$$

$\mathbf{x}_t \in \Re^D$ can then be found as:

$$\mathbf{x}_t \approx a * \mathbf{x}_s + b * \mathbf{x}_r.$$

The above method does assumes a 2D-space. A similar method in 3D would require the 3 nearest neighbours.

I don't understand the question about comparing for points within or outside the manifold, since we are already working within the manifold when picking some point in the embedding space. Linear interpolation between two embedding vectors corresponds to linear interpolation between the original images of these embedding vector, according to my way of mapping described above. This gives us a transition from one image to another as can be seen in the appendix.

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 the methods as described in https://www.cs.nyu.edu/ roweis/lle/papers/lleintro.pdf. The nearest neighbours are found by using a sklearn method for nearest neighbours.

They minimizing weights w_{ij} are found by using this formule:

$$w_{ij} = \frac{\sum_k C_{jk}^{(i)-1}}{\sum_{lk} C_{lk}^{(i)-1}}$$

with $\mathbf{C}^{(i)}$ the matrix with element $C_{jk}^{(i)} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_k)$ and $\mathbf{C}^{(i)-1}$ its inverse. $\mathbf{x}_j$ and $\mathbf{x}_k$ are nearest neighbours.

Then a matrix $\mathbf{M}$ is computed as:

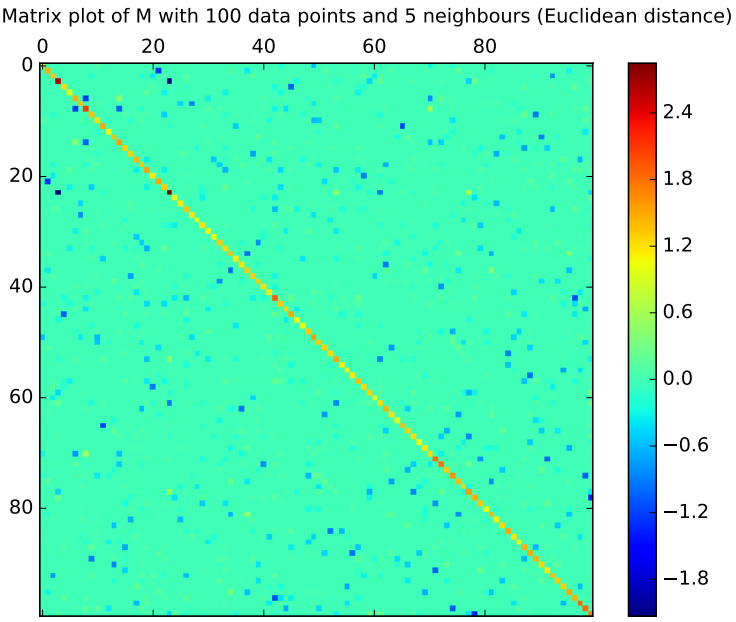
$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^T * (\mathbf{I} - \mathbf{W})$$

The top d of the bottom $d+1$ eigenvectors of this matrix represent the d embedding coordinates.

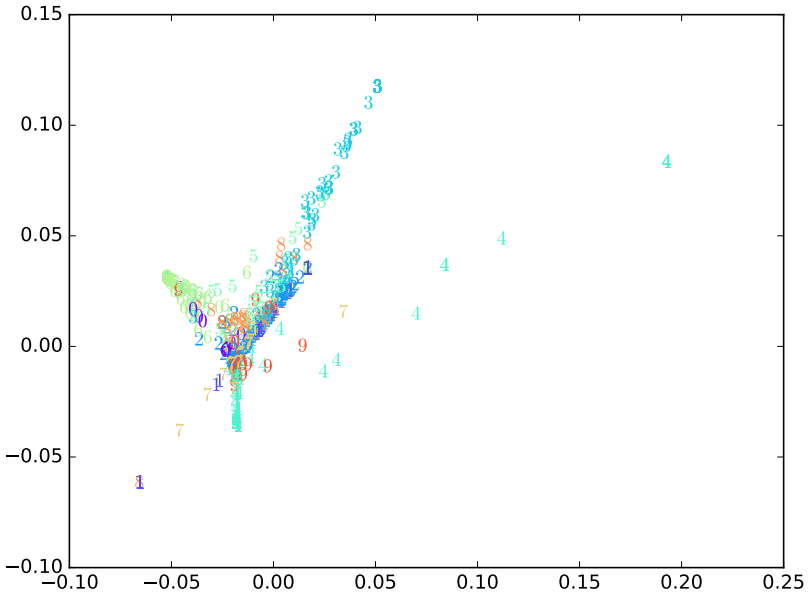
I had no efficiency or numerical problems. I restricted myself to 1000 data points because the visualizations didn't need more than that. One thing to take into account is that the zero eigenvalue of matrix M will be found as the machine precision.

Your Page

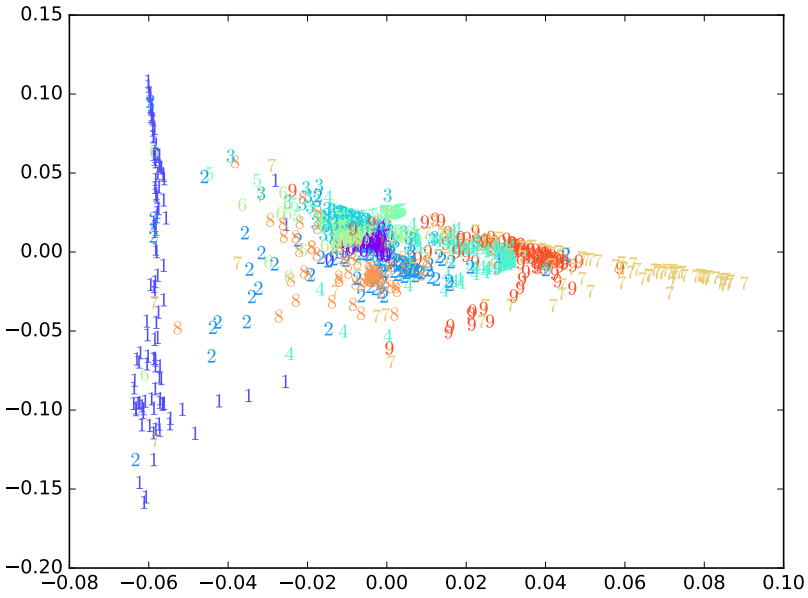
Your page gives you space to include ideas, observations and results which do not fall into the categories provided by us. You can also use it as an appendix to include things which did not have space in the other sections.
No page limit.



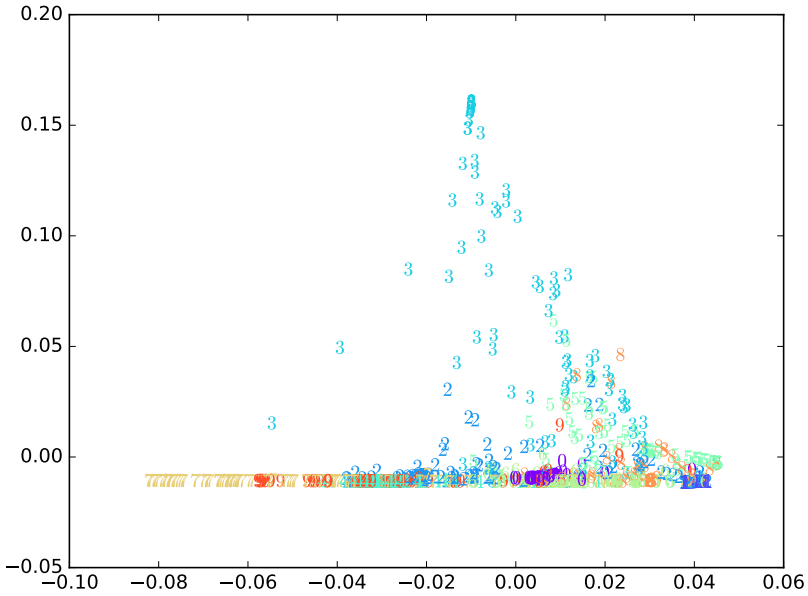
Scatter plot of 1000 data points in a 2D embedding space (3 neighbours, Euclidean distance)



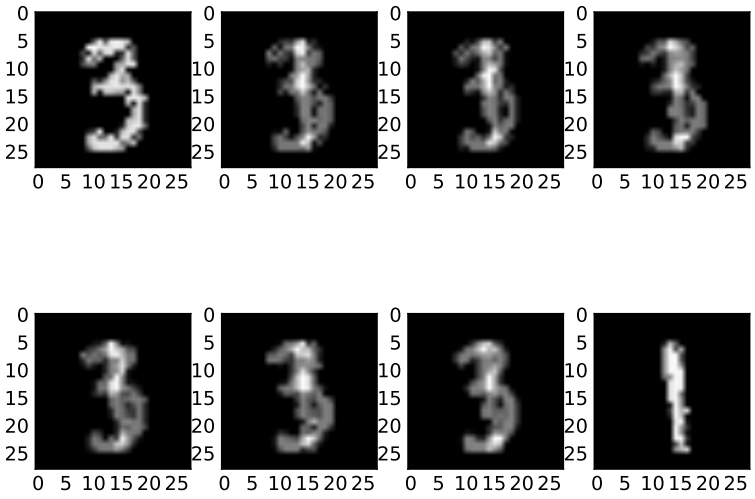
Scatter plot of 1000 data points in a 2D embedding space (10 neighbours, Euclidean distance)



Scatter plot of 1000 data points in a 2D embedding space (5 neighbours, Manhattan distance)



Linear interpolation between two images (labels 3 and 1)



16-948-549/1_locally_linear_embedding