SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Lidia Freitas
16-947-285

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 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

The main goal of Locally Linear Embedding is to generate highly nonlinear embeddings while reducing the dimensionality of data points. We expect each data point and its neighbours to lie on or close to a locally linear patch of the manifold and in this way we can rely on the neighbours to reconstruct a data point.
To do this we specify a reconstruction error:

$$\varepsilon(W) = \sum_i |\overrightarrow{X_i} - \sum_j W_{ij}\overrightarrow{X_j}\,|^2$$

Our goal is now to minimise the reconstruction error based on two assumptions:

- $\overrightarrow{X_i}$ is reconstructed only from its neighbours so the weight should be enforced $W_{ij} = 0$ when $\overrightarrow{X_j}$ is not a neighbour of $\overrightarrow{X_i}$.
- The rows of the weight matrix should sum to 1, so $\sum_j W_{ij} = 1$ for any $i = 1, ..., n$.

After figuring out the best values for W_{ij} we now try to find the best representation ($\overrightarrow{Y_i}$) of the data points in a lower dimensional space $d << D$.

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

The difference in this case from the previous is that now we fix the weights W and try to figure out Y, while in the previous we had X fixed and figured out W.

Variables and nomenclature:
- $\overrightarrow{X_i}$ - data points, for i=1,...,n.
- $\overrightarrow{Y_i}$ - representation in a lower dimension of the data point $\overrightarrow{X_i}$.
- W_{ij} - weight of neighbour $\overrightarrow{X_j}$ of the datapoint $\overrightarrow{X_i}$, or respectively $\overrightarrow{Y_j}$ and $\overrightarrow{Y_i}$.

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 visualisations 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 Neighbours

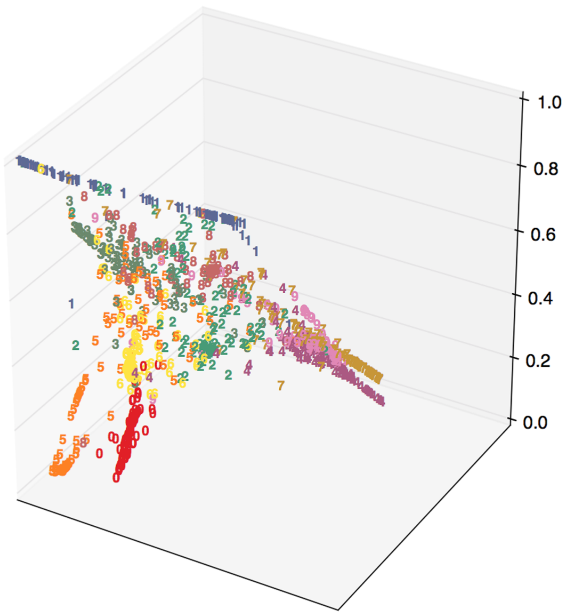
Investigate the influence of the choice of how many nearest neighbours you take into account. Additionally, try different metrics to find the nearest neighbors (we are dealing with images!).

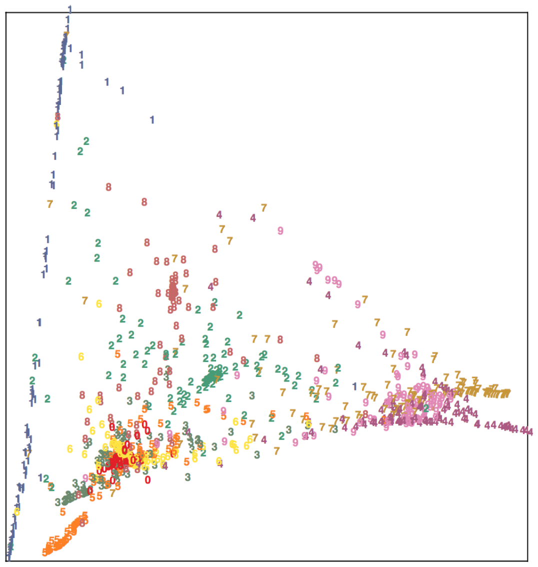
(e) Linear manifold interpolation

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...

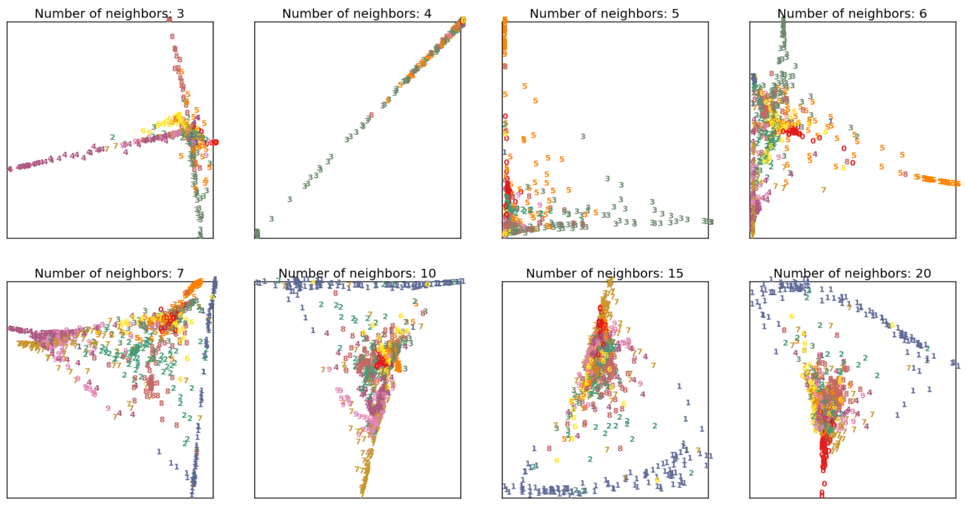
(b) Here we can see the embeddings in 3 dimensions and in 2 dimensions for a fixed number of neighbours (7) with 1000 data points. We can clearly see clusters of digits and some clear distinction on the edges of the plots (each colour is a different digit), even though not all of them all well separable.





(c) To minimise the embedding error we must ensure that we have at least $d+1$ small eigenvalues of M that should be close to zero. By plotting the singular values of M we can decide upon how many dimensions d are good based on the number of small singular values, we could even check if the sum of the chosen eigenvalues is still small.

(d) To analyse the best value of neighbours in this given problem I decided to experiment with different values and see how clustered the data was. The best value would be one that could provide enough information so the classifier wouldn't underfit but also not so much or the classifier would start to pick up noise and overfit. Some of the plots can be seen in the image below:



The Implementation

In the implementation section you give a concise insight to the practical aspects of this coding exercise. It mainly mentions the optimisation 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

It was too costly to use all the data set for this exercise (60000 data points), so I decided to only go with 1000 data points. To implement the algorithm I used scikit-learn and numpy.

source: 16-947-285/1_locally_linear_embedding