SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Giacomo Giuliari
16-932-121

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

The fundamental idea of this algorithm relies on the minimization of the two reconstruction errors, defined as:

$$E(W) = \sum_{i=1}^{N} \parallel x_i - \sum_{i=1}^{N} w_{ij} x_j \parallel^2$$

$$E(y_1 \dots y_n) = \sum_{i=1}^{N} \parallel y_i - \sum_{i=1}^{N} w_{ij} y_j \parallel^2$$

However, the explicit formulation of the error is never used in the actual implementation.
An important role in the algorithm is played by the matrix C_i for each datapoint x_i:

$$C_i = Z \cdot Z^T$$

Where $Z = N - X_i$ and N is the matrix containing as columns the k-nearest-neighbors of x_i.
The weights w are found by solving the linear system of equations

$$C_i \cdot w = \mathbf{1}$$

Finally, the y points in the embedding are the eigenvectors of the matrix

$$M = (I - W)^T \cdot (I - W)$$

associated with the $k + 1$ smallest eigenvalues (after dropping the eigenvector associated with eigenvalue 0).

The Answers

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 Answers continued on next page...

(a) The dataset was loaded from the supplied website and was then parsed to access images and labels.

(b) In the plots, expecially the 3D ones, the clusters are clearly visible. It seems that more numbers are "recognised" (gathered in the same cluster) better tant the others.

(c) No relevant box structures seem to appear in the matrix plot of M.
The matrix appears to be symmetric, sparse and with higher numbers on the diagonal.
By plotting the singular values in order it can be noticed that there are a few of them with a high absolute value. Since the singular values are associated with the energy of the projection on the lower-dimensional space, a way to determine the optimal embedding space would be to choose the number of dimendions of the space equal to the number singular values (in order of decreasing magnitude) that have the biggest difference in value from the others (i.e. before their value starts to decrease slowly). This can be regarded as an "elbow method".

(d) The lowest error is achieved by using 4-nearest-neighbors. For values greater than 4 the error increases, for lower values the matrices can be singular.
The L1-norm (Manhattan distance) was tried instead of the usual L2. No changes in the results were noticed. More sophisticated image-related metrics were not investigated due to lack of time.

(e) The mapping is done by first finding the k-nearest-neighbors of the point amongst the transformed ones. Then the local weights need to be calculated and used to find the corresponding point in the higher-dimensional space by applying the same weights to the transformations of the neighbors.
The linear interpolations in the embedding and in the original space are shown in figure.

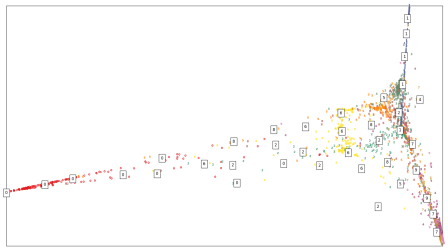

Figure 1: figure
2D Embedding

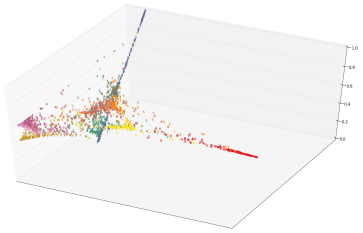

Figure 2: figure
3D Embedding

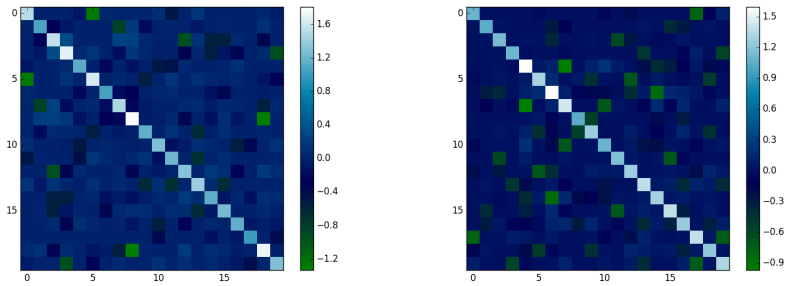

Figure 3: figure
Matrix plots of a subset of indices on the diagonal of the M matrix.

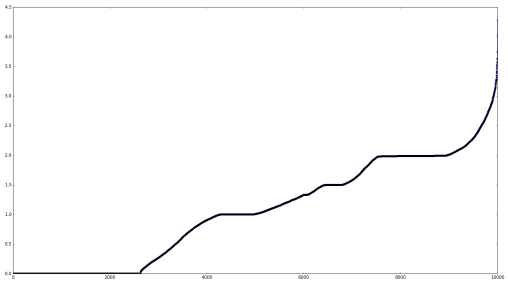

Figure 4: figure
All the singular values

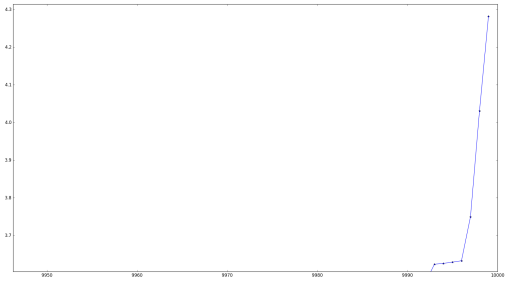

Figure 5: figure
The top 7 singular values. Notice the change in the 'gradient' after the third.

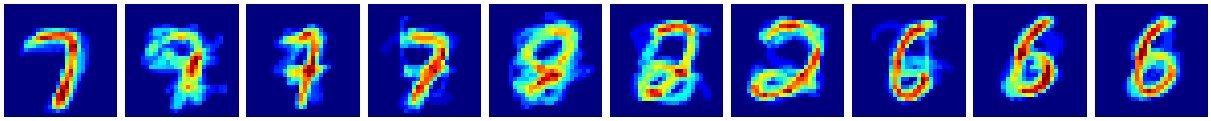

Figure 6: Interpolation (linear) in the 2-dimensional embedding space, mapped to the original space.

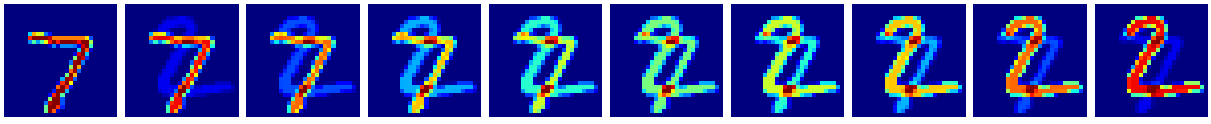

Figure 7: Interpolation (linear) in the original 784-dimensional space.

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 implemented the algorithm 3 times, one using matrix inversion, another using the linear system solution and the last one changing the way the M and C matrices were handled.
The implementation problems encountered were in practice the ones listed in the paper presenting the LLE algorithm. In particular:

- The efficiency issue of inverting the C matrix was solved by instead finding the results of the linear system and then normalizing the weights.
- The problem with singular C matrices, i.e. no solution of the linear system, was solved by adding a term to the diagonal that is small compared to the trace.
- The possibly huge dimensions of the M matrix were handled leveraging on the fact that it is sparse and symmetric, enabling efficient storage and computation.
- To find efficiently the nearest neighbors in case the total number of datapoints is "small enough" k-d trees can be used.

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
16-932-121/ 1_locally_linear_embedding