SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

SILVAN WEDER
13-914-643

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 the Locally Linear Embedding problem, a set of N points $\mathbf{x}_i$ with D dimensions (e.g. $\mathbf{x}_i \in \mathbb{R}^D$). With the locally linear embedding the embedding of these points in a low dimensional space $\mathbb{R}^d$ is possible ($d << D$). In this low dimensional space each point $\mathbf{x}_i$ is represented by a point $\mathbf{y}_i \in \mathbb{R}^d$.

The formulas for the implementation of the locally linear embedding are originally derived from its reconstruction error in this problem. Therefore, the reconstruction error is presented first.

$$\mathcal{E}(\mathbf{W}) = \sum_i \|\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j\|^2 \tag{1}$$

The points $\mathbf{x}_j$ are the j closest neighbors of the point $\mathbf{x}_i$ and w_{ij} are the weights, that correspond to each of this neighbors.

Then the reconstruction error is minimized using the constraint $\sum_j w_{ij} = 1$. This minimization yields the equation for the weights that give the optimal reconstruction with respect to the minimum of the reconstruction error given in 1.

$$w_{ij} = \frac{\sum_k C^{(i)-1}_{jk}}{\sum_{lk} C^{(i)-1}_{lk}} \tag{2}$$

The matrix $\mathbf{C}^{(i)}$ has the elements $C^{(i)}_{jk} = (\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{x}_i - \mathbf{x}_k)$ and $\mathbf{C}^{(i)-1}$ is its inverse.

Having now the weights $\mathbf{W}$, one can define the embedding error for the new points $\mathbf{y}_i \in \mathbb{R}^d$.

$$\mathcal{E}(\mathbf{y}_1, ..., \mathbf{y}_N) = \sum_i \|\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j\|^2 \tag{3}$$

This equation 3 can also be written as

$$\mathcal{E}(\mathbf{y}_1, ..., \mathbf{y}_N) = \sum_k \mathbf{u}_k^T \mathbf{M} \mathbf{u}_k \tag{4}$$

where $\mathbf{u}_k = (y_{1k}, ..., y_{Nk})$ for $k = 1, ..., d$ and $M = (\mathbb{I} - \mathbf{W})^T(\mathbb{I} - \mathbf{W})$. It can be shown that the u's are found with the computation of the eigenvectors of $\mathbf{M}$. The correct u's are the ones that are associated with the smalles $d + 1$ eigenvalues skipping the one with eigenvalue 0.

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...

(a) Get the data

The MNIST data set is used for the Locally Linear Embedding problem. The data set has 60000 images of handwritten numbers from 0 to 9.

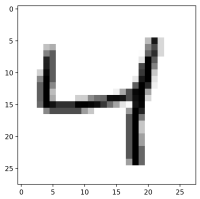

Figure 1: Gray-scale image of number 4

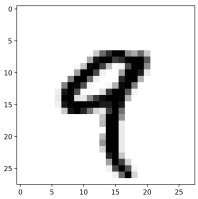

Figure 2: Gray-scale image for number 9

(b) Locally linear embedding

The images of the handwritten figures are transformed into D-dimensional vectors and then embedded in a space of dimension 2 and 3.

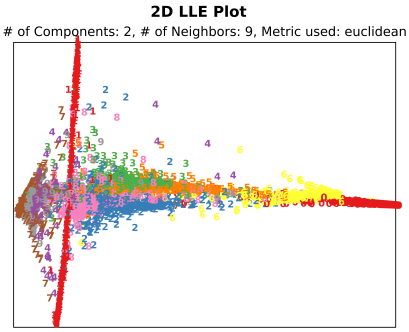

Figure 3: Locally linear embedding of 5000 MNIST images into a 2 dimensional space

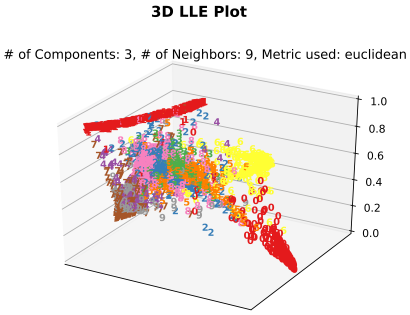

Figure 4: Locally linear embedding of 5000 MNIST images into a 2 dimensional space

In figures 3 and 4, the LLE into 2- or 3-dimensional space is shown. One can clearly see the cluster of the different numbers in the MNIST sample. For this LLE, 9 neighbors have been used for the nearest neighbor computation and the Euclidean vector norm has been used as a metric.

(c) Cluster structure

In order to study the structure of the matrix M that is used for the LLE, M is plotted as it is shown in figure 5.
In this plot, one can observe that the matrix M has a block-diagonal structure. For each number from 0 to 9 there is a corresponding block in the matrix. Looking at the singular values when the labels of the training data are sorted, one can easily see that the singular values are strictly decreasing. Therefore, I would propose to introduce a LLE with 10 coordinates (e.g. for each label one coordinate).

(d) Nearest Neighbors

In order to analyze the influence of the number of neighbors chosen in the nearest neighbor computation, two plots with different numbers of neighbors are presented in figure 7 and 8.

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

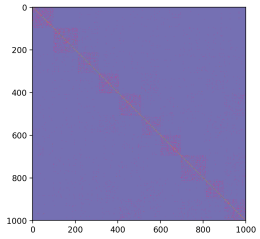

Figure 5: Matrixplot of matrix M

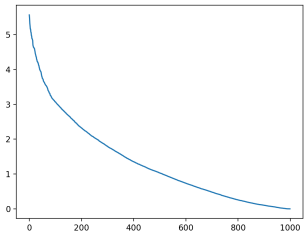

Figure 6: Plot of singular values of matrix M

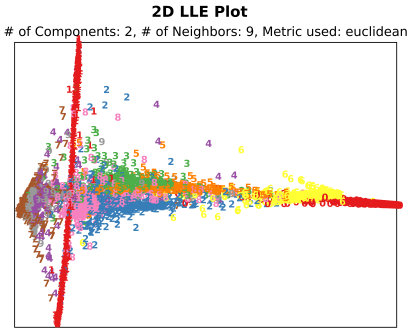

Figure 7: LLE with 7 neighbor points

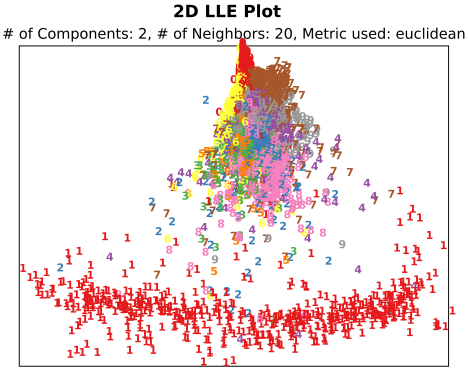

Figure 8: LLE with 20 neighbor points

From these plots, it can be derived that the an increasing number of neighbors used in the computation does not improve the structure of the clusters in the result. However, a number of neighbors that is too low is also not recommendable because then, the clusters are seperable.
If one changes the metric that is used for the nearest neighbor computation, the results look quite similar. Although there is a slight rotation in the plots around the center of the plot.

(e) Linear manifold interpolation

The reconstruction of points from inside the manifold works quite well as it is shown in the images below. However, if the point is not part of the manifold, the reconstruction is not working well anymore. Although, there are still some features of a number visible, it is hard to tell what number it is.

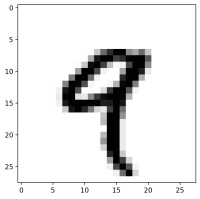

Figure 9: Original gray-scale image

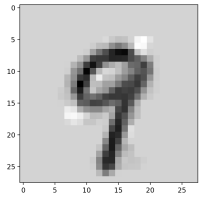

Figure 10: Reconstructed gray-scale image from the embedded space

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

For this problem I mainly used the numpy and scipy libraries for python. The LLE algorithm I implemented from scratch. However, for the nearest neighbors computation I used the NearestNeighbors class from the sklearn library in order to achieve a proper efficiency of my code. As the nearest neighbors computation and the solver for the eigenvalues/eigenvectors scale with N^2, the code was efficient enough to run on my computer. However, I restricted the number of data to 5000 in order to get results in a reasonable time.

Algorithm

1. Nearest Neighbor Computation using NearestNeighbors-functions from the sklearn library. For the Computation the kd-tree algorithm is used.
2. Computation of the weights using simple linear algebra tools from numpy.
3. Computation of matrix $\mathbf{M}$ and solving for the $d+1$ smallest eigenvalues. For the eigenvalue computation the function $eigh()$ has been used.

Git Branch 13-914-643/1_ locally_ linear_ embedding

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.