SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Vincent Stettler
12-932-661

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

Locally linear embedding is a non-linear dimension reduction algorithm that, as opposed to methods like PCA or MDS, can be used to preserve the internal, low-dimensional structure of a non-linear, smooth manifold living in a high dimesnional space.
Given are N vectors $x_i \in \mathbb{R}^D$ that are expected to lie on some manifold. The idea is to use the fact that neighboring points should then, approximately, lie on a locally linear patch of the manifold. Each data point is expressed as a linear combination of its neighbors, and the so-called *reconstruction error* is then defined as:

$$\mathcal{E}(W) = \sum_i |x_i - \sum_j w_{ij} x_j|^2$$

The optimal weights are found by minimizing the reconstruction error under the constraints that (i) $w_{ij} = 0$ if x_j is not considered to be a neighbor of x_i, and (ii) $\forall i. \sum_j w_{ij} = 1$. Note that these weights then are invariant under rotation, rescaling, and translation of the data points and its neighbors. This implies that the wheights need to characterize intrinsic properties of each neighborhood, and do not depend on a particular frame of reference.
The next step of the LLE algorithm then maps each $x_i \in \mathbb{R}^D$ to a $y_i \in \mathbb{R}^d$, where $d << D$ is the internal dimension of the manifold. This is done by choosing y_i such that the *embedding error* is minimized:

$$\Phi(Y) = \sum_i |y_i - \sum_j w_{ij} y_j|^2$$

Note that the weights w_{ij} are the optimized weights of the first step. By our assumptions we can expect that the same weights w_{ij} that reconstruct x_i in the D dimensional space, also reconstruct its embedded manifold coordinates in d dimensions. The above problem can be solved by solving a sparse N $\times$ N eigenvector problem.

The Questions

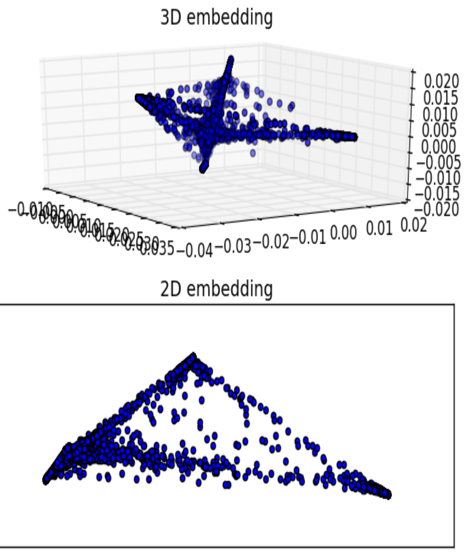
This is the core section of your report, which contains the tasks for this exercise and your respective solutions. Make sure you present your results in an illustrative way by making use of graphics, plots, tables, etc. so that a reader can understand the results with a single glance. Check that your 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?



There are some clusters, yes.

(c) Cluster structure

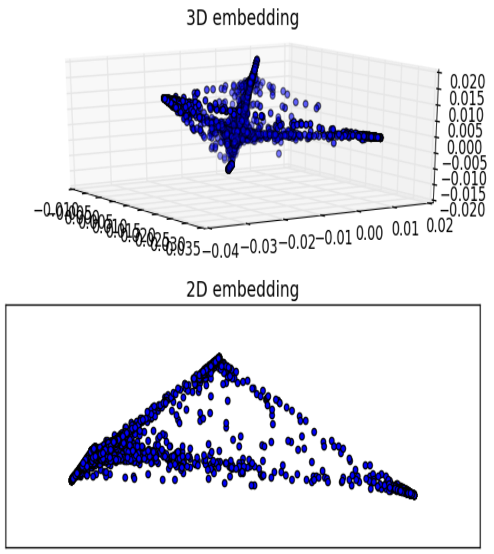
Investigate the cluster structure of the data. Can you observe block structures in the M matrix (use matrix plots)? Also plot the singular values of M. Do you notice something? Can you think of ways to determine the optimal embedding dimension?

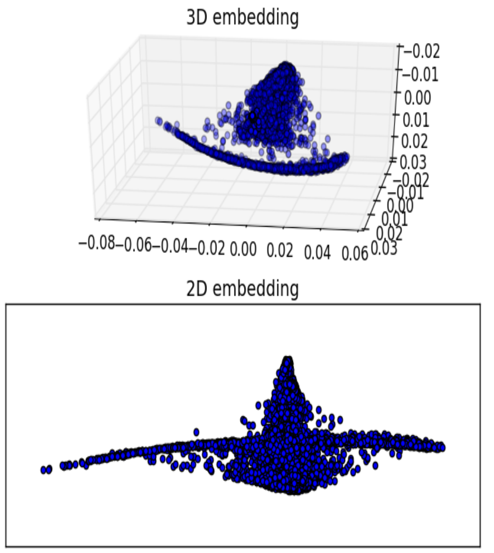
Unfortunately, I am not able to investigate the M matrix, because I am using the scikit implementation of the LLE algorithm and there is no simple way to access the matrix. I further explain my issues in the implementation section.
The optimal embedding in the LLE algorithm is found by computing the d+1 eigenvectors of the M matrix that correspond to the smallest d+1 eigenvalues. After dropping the smallest eigenvector, the remaining d vectors are then used as embedding coordinates. So I think that the optimal embedding dimsension can be determined by taking all eigenvectors that are small in some sense. Another way I could think of, is to simply try out a range of candidate dimensions and take that number, after which the reconstruction error improves only slightly.

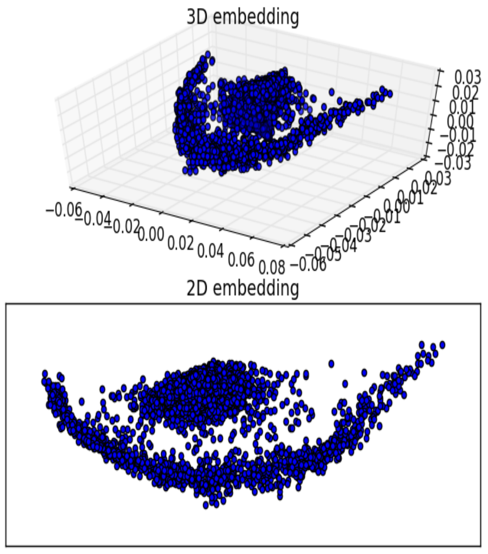
(d) Nearest Neighbors

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!).

Again, I am not able to try different metrics, as the scikit interface has no simply way to do use different metrics. The following figures show the embeddings using 5, 15, and 25 neighbors (Sorry for the formatting):







I notice that the reconstruction error decreases when the number of neighbors decreases.

(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?

Unfortunately, I had no time left to do this part of the project.

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

https://gitlab.vis.ethz.ch/vwegmayr/slt-coding-exercises/tree/12-932-661/1_locally_linear_embedding
I started by implementing the LLE algorithm using the scikit library. After reading the question about the M matrix, I decided to implement the algorithm without the scikit implementation, using only numpy and the scikit nearest neighbor function. I tried to implement the algorithm as it is described on the following page: https://www.cs.nyu.edu/ roweis/lle/algorithm.html. As it is often more efficient to replace for loops with matrix operations, I tried to completely avoid the for loops. After having written some parts, I tested the implementation and encountered an out of memory error. Unfortunately, I didn't had enough time left to fix this bug, so I finished the project using the scikit implementation. The code producting the out of memory exception is inside the lle(X, k) function in the lle.py file. I would try to fix the bug by using for loops instead of matrix operations at some places.
As the number of samples in the MNIST dataset is quite large, I decided to only use the samples of the test set.

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.

In the next project I will invest more time. 12-932-661/1_locally_linear_embedding