SLT coding exercise #1

Locally Linear Embedding

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

Due on Monday, March 6th, 2017

Yu-chen Tsai
16-929-747

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

Your Answer

$$\mathcal{E}(W) = \sum_i |\vec{X}_i - \sum_j W_{ij}\vec{X}_j|^2$$

$$\sum_j W_{ij} = 1$$

$$\Phi(Y) = \sum_i |\vec{Y}_i - \sum_j W_{ij}\vec{Y}_j|^2$$

$$\sum_i \vec{Y}_i = 0$$

$$\frac{1}{N}\sum_i \vec{Y}_i\vec{Y}_i^{\,T} = I$$

1. For each $\vec{X}_i$, one identifies the neighbors $\vec{X}_j$ of it either by K nearest points in Euclidean distance or other sophisticated rules based on local metrics.
2. Because the reconstruction error should be invariant to translation, the rows of the weight matrix W have to sum to one.
3. The cost function of the mapping is defined as sum of the reconstruction error of mapped vectors.
4. The embedding should be invariant of translation and scaling, so one can remove two degrees of freedom by making Y has zero mean and unit variance.

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!).

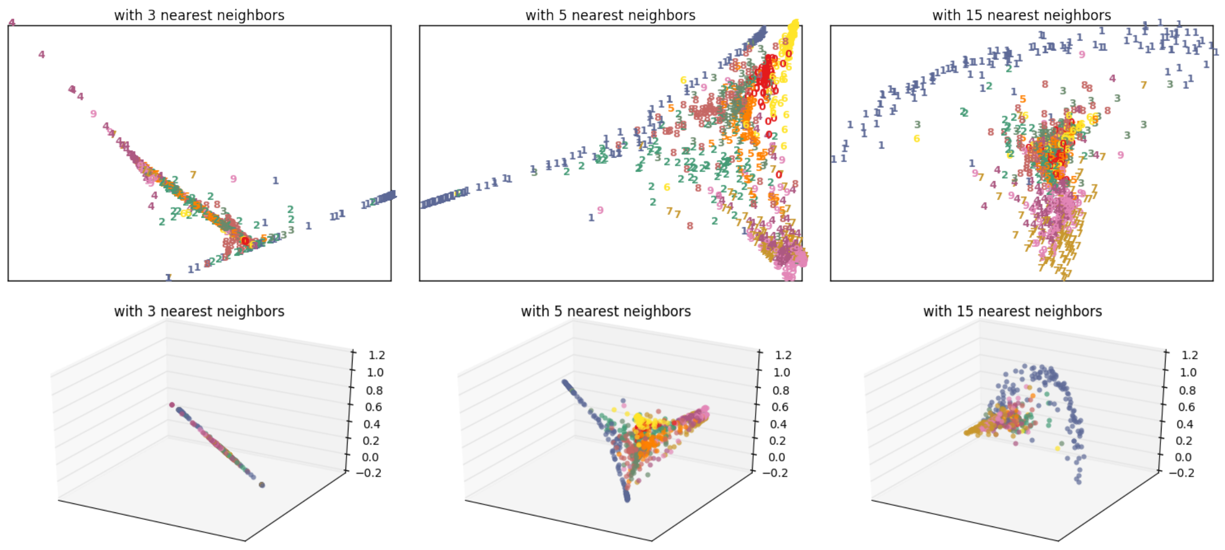
(e) Linear manifold interpolation

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?

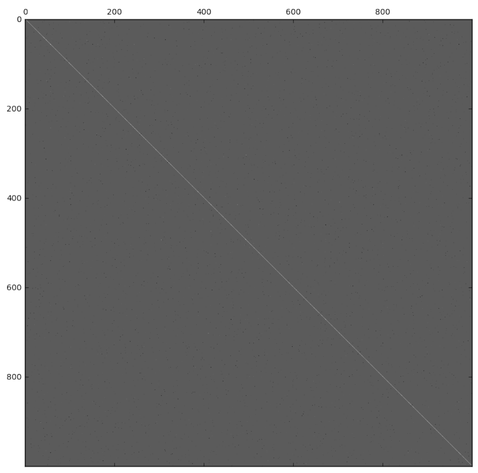
The Questions continued on next page. . .

(b) Locally linear embedding

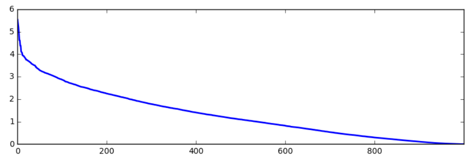
Yes, as the number of neighbors used for computing the embedding increases, we can see it gets easier to separate the clusters of different digits. And it is worth to mention that, as one can see, the 2 dimensional embedding is actually a projection of the 3 dimensional embedding.



(c) Cluster structure



Matrix plot of M

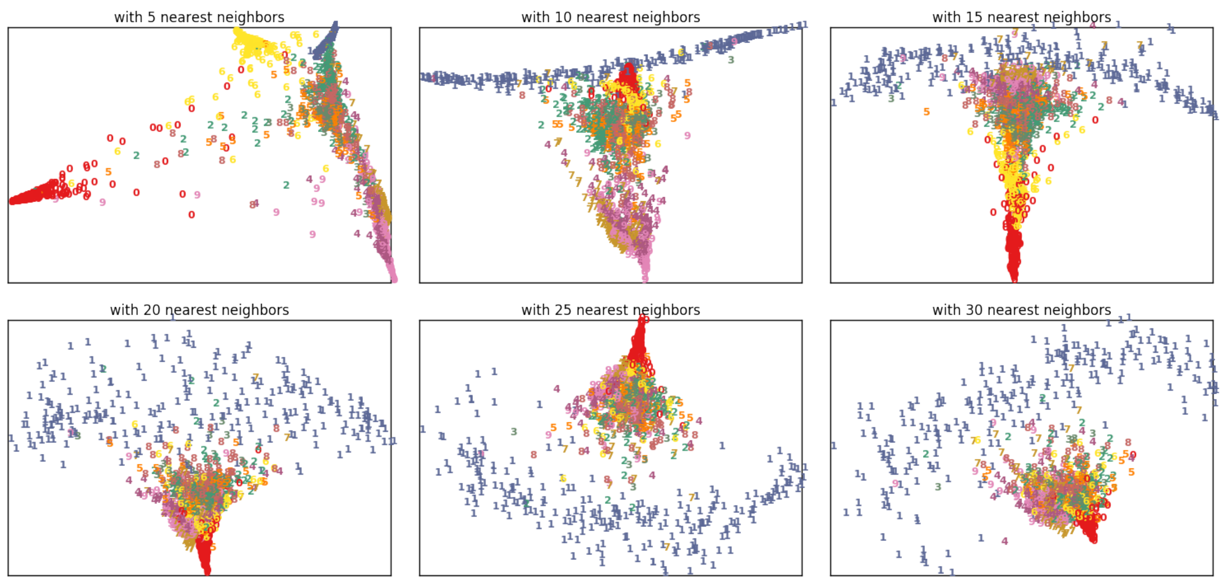


Eigenvalues of M (sorted)

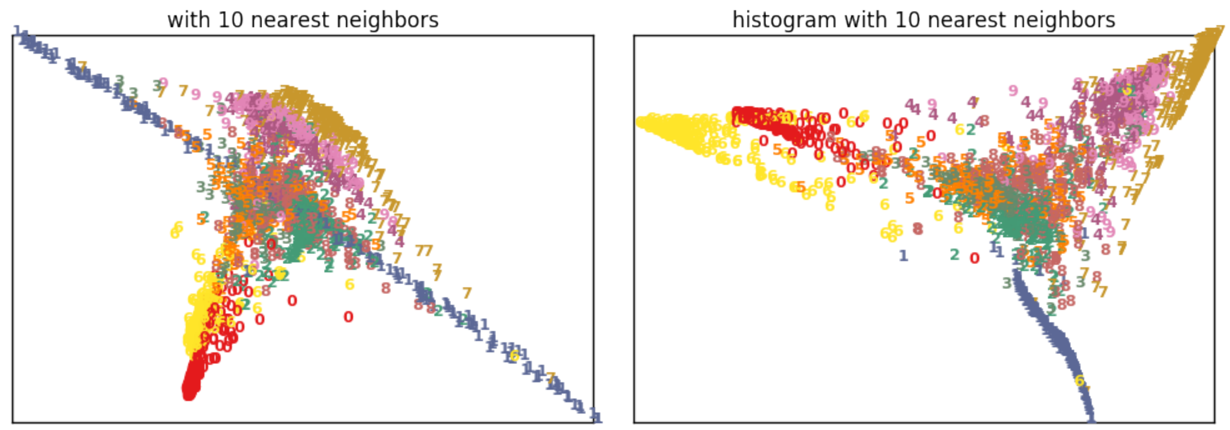
Honestly, I didn't observe any "block" structures in the matrix plot. But I think it depends on the dataset. If the space of the data can be partitioned into m subspace, then maybe we can observe m blocks on the diagonal of the matrix plot indicating that the problem can be divided into smaller embedding problems.

One can observe that there is a turning point in the growth of eigenvalues. Because the value of the cost function $\Phi(Y)$ is actually the sum of smallest d + 1 eigenvalues (including at least one zero), we can determine the embedding dimension by observing the growth of eigenvalues(sorted from lowest to highest) and then pick a d that is smaller than the point where eigenvalue starts to grow rapidly.

(d) Nearest Neighbors



We can see that increasing the number of neighbors from 5 to 15 does make the clusters of each digit easier to separate, but more than 15 neighbors doesn't improve the result a lot. Maybe it's because the number of points I used here is 2000, so when we consider too many neighbors, we may include points of different digits. And this will only hinder the embedding algorithm to find a good embedding where digits are visually separated.



Because we are dealing with images here, it might helps to use histogram of sliding windows instead of raw image. Here I tried to compute the 8-bin histogram of 7x7 sliding window of step size 3, and concatenated them together as our new feature vector. In total it has 1352 dimensions. As we can see in the figures above, using histogram really improve the embedding result. Windows of proper size and step size can keep the important information and ignore the noise, that's why the result is better.

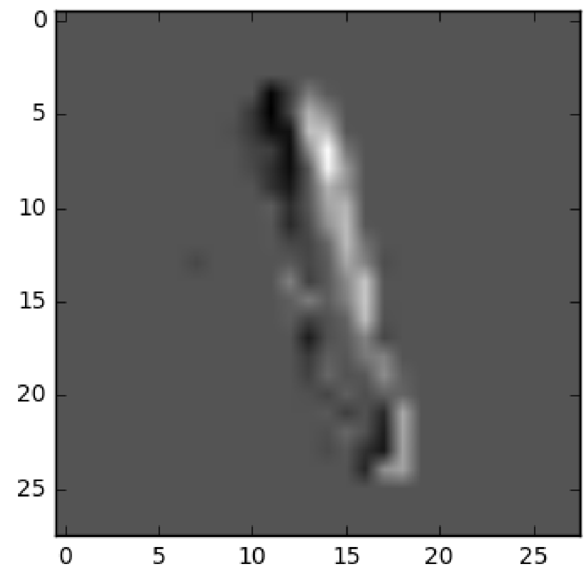
(e) Linear manifold interpolation

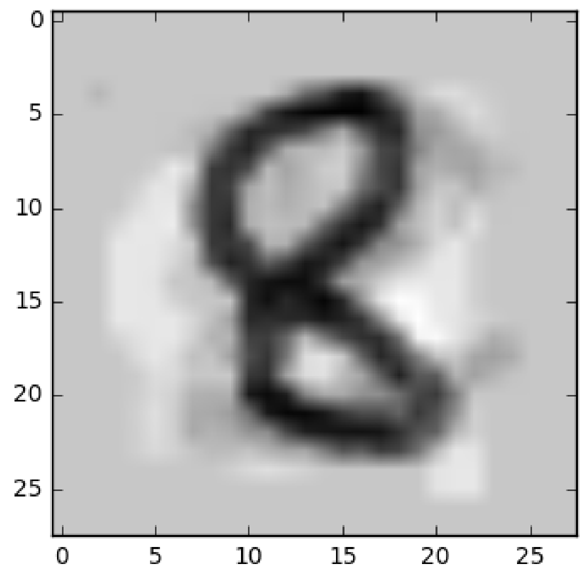
Given a point y_i in the embedding space, we first find the nearest neighbors of it in the embedding space, and computer the weights w_j of them by

$$w_j = \frac{\sum_k C_{jk}^{-1}}{\sum_{lk} C_{lk}^{-1}}$$

where $C_{jk} = (y_i - y_j)^T (y_i - y_k)$. And then, construct x_i in the original space by

$$x_i = \sum_j w_j x_j$$





The left figure is a combination of neighbors of digit 1, the right figure is a combination of digit 2,3,8, and 9. The problem of this method is clear: when the picked point is in the embedding space but outside the manifold, the nearest neighbors are likely come from different clusters(because they are all far away). And the corresponding reconstruction may be totally meaningless.

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

1. Sometimes C is singular or nearly singular, the program report error message at the matrix inversion step. And this typically arises when K is large. One way to avoid the numerical difficulty is adding small values on all the diagonal terms of C as regularization.
2. All the calculations have closed form solution, so no optimization used to solve the equations. The only mathematical libraries used is the inverse matrix solver and eigenvalue decomposition solver in numpy.
3. Because of the limit of memory and time, all the experiments only run on less than 2000 points randomly selected from the original 70000 points.

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
16-929-747/1 locally linear embedding