Introducing Locally Linear Embedding (LLE) as a Method for Dimensionality Reduction

Jennifer Chu
Math 285 – Fall 2015

Introduction

In many areas of study, large data sets often need to be simplified and made easier to visualize. For example, images of faces or spectrograms of speech are complex and need to be preprocessed before the underlying patterns within these data can be seen. Thus there is a need for dimensionality reduction.

In this short paper, we will investigate Locally Linear Embedding (LLE) as a method for dimensionality reduction. LLE was originally introduced by Sam T. Roweis and Lawrence K. Saul in 2000, for the purpose of addressing the problem of reducing dimensions for non-linear data structures. We will take a look at the algorithm used, and compare the effectiveness of LLE to other reduction methods we have learned in class, namely PCA, Kernel PCA, and Isomap.

LLE Method Explained

We introduce Locally Linear Embedding (LLE) as an **unsupervised** method for non-linear dimensionality reduction that can discover **non-linear structures** in the data set, and also **preserve the distances within local neighborhoods**.

Let us assume a data set X with a non-linear structure. This data set is in D-dimensions. We want to map this data onto a smaller d-dimensional data set Y. We want to find Y while preserving as much of original pattern in the data as possible.

$$X \in R^D \to Y \in R^d$$

LLE first finds the k-nearest neighbors of the points. Then, it approximates each data vector as a weighted linear combination of its k-nearest neighbors. Finally, it computes the weights that best reconstruct the vectors from its neighbors, then produce the low-dimensional vectors best reconstructed by these weights. Thus, LLE preserves the local distances between the data points and will successfully detect non-linear structures in the data.

A way to visualize the process of LLE is by **imagining locally linear patches in the manifold**. Although the underlying structure of the data set is curved, we can assume each data point and its closest neighbors to be on the same linear plane. We can imagine that a data point is lying on a flat patch of circle, and has close neighbor points that are also lying on the same flat patch of circle.

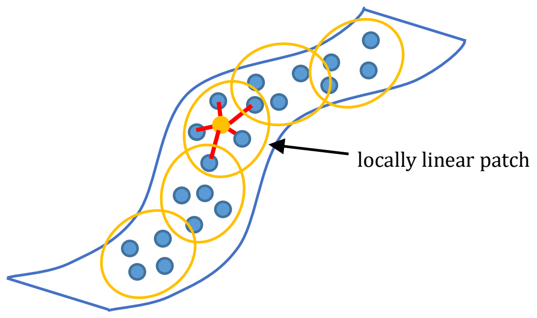

Figure 1: A manifold

Then, we have many connecting patches along the manifold curve. The transplantation of each patch involves no more than one transformation (translation, rotation, etc.), so that the linear weights of the points are invariant, and we can use the same weights to reconstruct data points in the low-dimensional space. The local distances between the points are preserved this way.

Algorithm: Locally Linear Embedding (Roweis & Saul)

Input: X_{Dxn}, k

Output: Y_{dxn}

(1) Find the k-nearest neighbors of each data point x_i.

(2) Compute weights W_{ij} that best reconstructs x_i from its neighbors.

$$min\ E(W) = \sum_i \left| \vec{X_i} - \sum_j W_{ij} \vec{X_j} \right|^2$$

(3) Compute embedding vectors Y_i using W_{ij}.

$$min\ \Phi(Y) = \sum_i \left| \vec{Y_i} - \sum_j W_{ij} \vec{Y_j} \right|^2$$

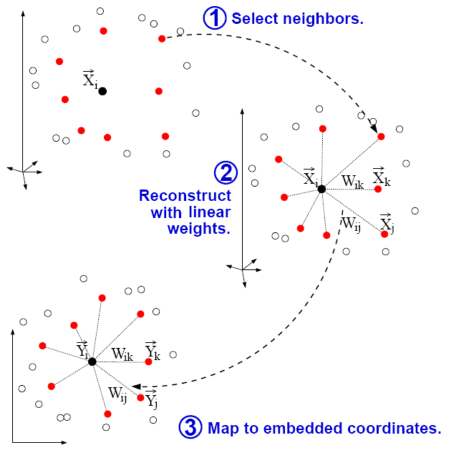

Figure 2: LLE algorithm
(picture from Lawrence K. Saul at al. (2002))

Now let us apply the LLE technique along with PCA, Kernel PCA, and Isomap to some data sets.

The Swiss Roll Example

Here, our toy data set is a Swiss roll figure, which is a manifold with smooth curves (data and tools from http://lvdmaaten.github.io/drtoolbox/). We want to reduce dimensions and "unroll" the Swiss roll so that the points are not overlapped with any other points, in other words, we want to map the points onto the lower-dimension space while preserving the original local distances between points.

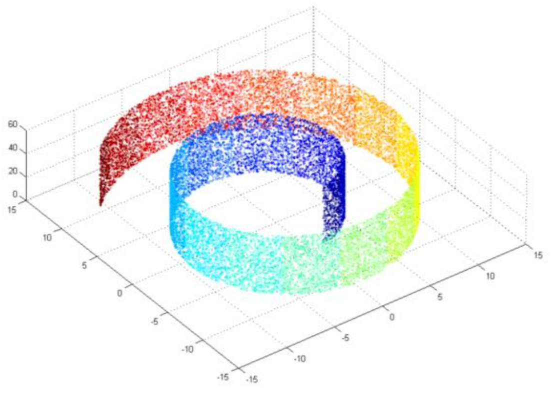

Figure 3: A Swiss roll in 3-dimensions

There are many ways to do dimensionality reduction on X to get Y. A popular method for dimensionality reduction is **Principle Component Analysis (PCA).** For PCA, the goal to is to project data from a high-dimension to a low-dimension so that the **variance of the data in the lower dimension is maximized.** PCA computes the singular value decomposition (SVD) of the covariance matrix, and take the largest of the eigenvectors (directions where there is most variance in the data). However, PCA is limited in that it is only good for linear structures; it cannot properly detect any non-linear patterns. Many people use this method because it is fast and easy to implement, however it would not make sense to apply PCA here since we have a non-linear figure.

Let's try applying PCA:

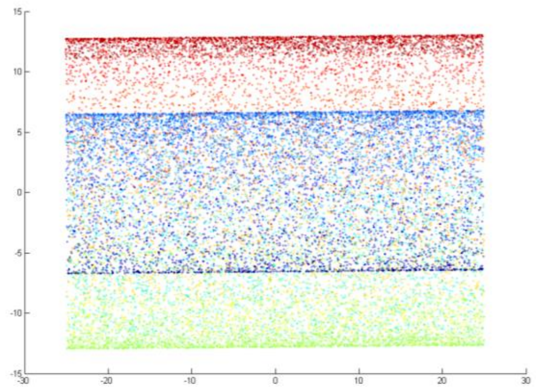

Figure 4: Applying PCA to the Swiss roll

PCA does not do a very good job of detecting the underlying curved pattern. PCA maps faraway data points to nearby points in the plane, and we can see from the result that some points are overlapped in the low-dimensional space.

An extension of PCA is **Kernel PCA**. Kernel PCA does have the capability to model non-linear manifolds. This method maps the original data set to a larger dimensional feature space using a kernel equation, then apply regular PCA. In other words, it **first reduces non-linear structure to linear, then apply PCA to further reduce dimensions.** Here we are using the Gaussian kernel equation that calculate the squared L2-norm of the distances between two points: $\kappa(x_i, x_j) = exp(\|x_i - x_j\|_2^2/2\sigma^2)$. We apply the equation to every pair of points in the data set, then obtain the eigenvectors of the centered kernel matrix.

We apply Kernel PCA to the Swiss roll:

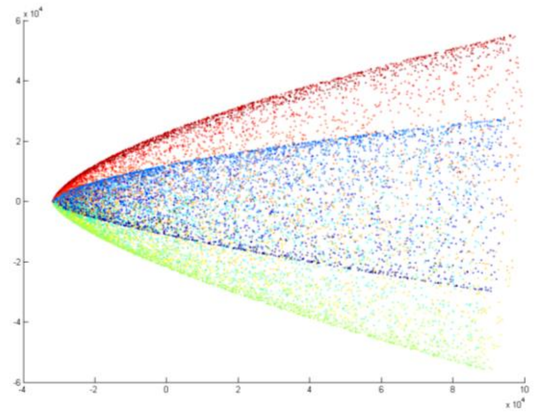

Figure 5: Applying Kernel PCA to the Swiss roll

Kernel PCA also does not work well here. We can see different colored points still overlap each other.

Another non-linear dimensionality reduction method we learned is **Isomap**, which **combines geodesic distance and MDS to preserve the distances between pairs of data points.** MDS only preserves pairwise distances between data points using Euclidean distances, while Isomap uses geodesic distances in its calculations, which enables detection of non-linear or curved structures. Similar to LLE, Isomap first constructs the neighborhood graph using k-nearest neighbors. It computes the shortest path graph search, and then finally constructs the d-dimensional space using MDS. The difference between Isomap and LLE: Isomap take a global approach and maintains the geodesic distances between all points; nearby points are nearby and faraway points are faraway. LLE takes a local approach and maintains distances in local neighborhoods; only nearby points should be nearby.

We apply Isomap to the Swiss roll:

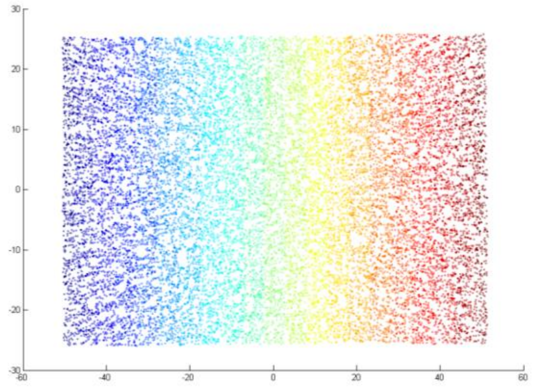

Figure 6: Applying Isomap to the Swiss roll

Isomap works pretty well here. There are no overlaps with the different colored points.

Finally, we apply **LLE** to the Swiss roll:

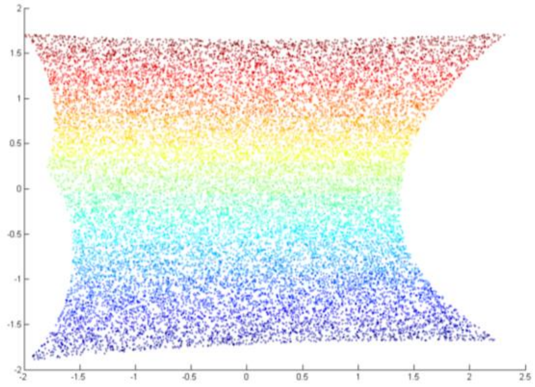

Figure 7: Applying LLE to the Swiss roll

Like Isomap, LLE gave good results and has fully "unrolled" the Swiss roll. We can see that LLE successfully preserved the local neighborhoods of the data points when mapping to a lower-dimensional space. Let's look at the computational time it took for each method:

	PCA	Isomap	Kernel PCA	LLE
Processing time (in seconds)	0.4829	11.2933	1.7236	1.0172

We can see that **LLE is computationally much less expensive than Isomap.** Computations are fast because it tends to accumulate sparse matrices, which saves calculation time and also computer space.

The MNIST Digits Example

We now apply the LLE method to a real data set, the MNIST digits data set – 1'st digits only (data set from http://yann.lecun.com/exdb/mnist/index.html). For comparison, we will also apply PCA, Kernel PCA, and Isomap to the same data. The data set contains 6,742 images of 28x28 size; we only used 1,000 images in this test.

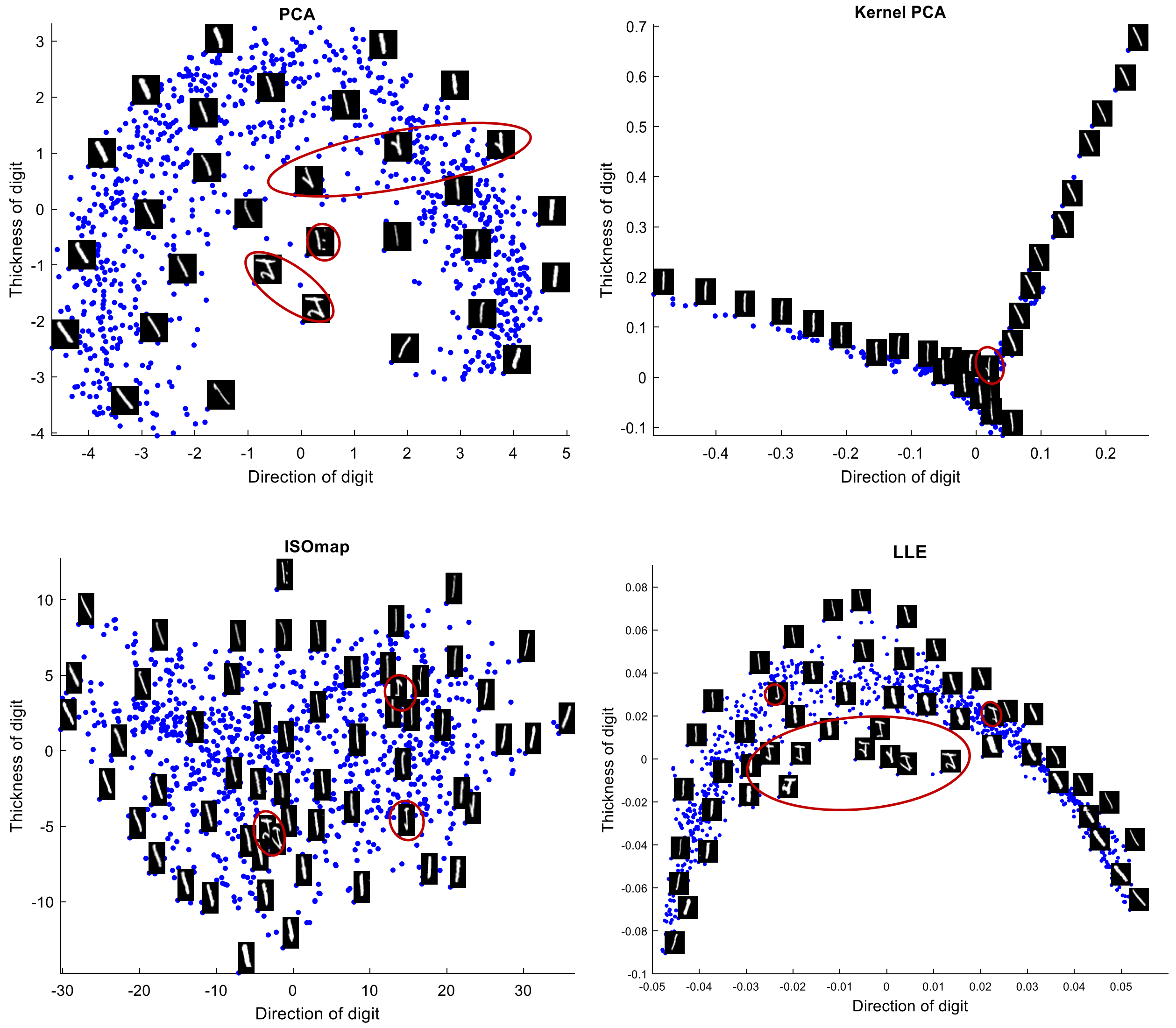

Figure 8: Dimensionality reduction of the MNIST digits data set- 1's digits.
(Top left) PCA. (Top right) Kernel PCA. (Bottom left) Isomap. (Bottom right) LLE.

PCA did okay in mapping the digits with different characteristics. The "1"s that look like "J"s are grouped in the middle, and we can see that there are a couple of "1"s with hooks near the right-middle of the graph. There is a general pattern of the digits being tilted to the left moving to being tilted to the right on the x-axis, and the digits going from thin to thick on the y-axis, but the patterns are not perfect and some images are still misplaced.

Kernel PCA did fine in mapping the images by direction of digits, but did not really separate the digits with thicker strokes from digits with thinner strokes. The plot looks like an open "V", or two orthogonal lines. We can see that from the left side of the graph, the direction of the digit is centered, and on the right side of the graph, the direction of the digits is tilted to the left. One weird image "1" with a hook is found in the vertex of the "V", and it is assumed that the other weird "1"s are also in this neighborhood.

Isomap did pretty well in mapping the digits both by direction and by thickness. We can see that going from left to right of the graph, the digits generally go from tilted to the left to tilted to the right. Here, we can also see that the weird "1"s that look like "J"s are grouped together in the center, and the weird "1"s with a hook are in the same area on the right side of the graph. The digits with weak strokes are near the top center.

LLE did the best in mapping the digits in the reduced dimension. The reduced structure looks like an upside down "V" shape. We can see that the digits on the left side are straight and tilted right, while the digits on the right side are tilted left. And keeping in accordance to the direction of the digits, the digits on the outer part of the "U" shape have thinner strokes, while the digits near the inner part of the "V" shape have thicker strokes. The weird "1"s with hooks tend to be near the inner part of the "U". And on the inner side of the vertex of the "V" structure, we have the weird "1"s that look like "J"s. There is a very clear pattern of where the images are placed.

Let's take a look at the computational time:

	PCA	Isomap	Kernel PCA	LLE
Processing time (in seconds)	0.5938	18.8594	2.7344	1.2188

LLE only took 1.2188 seconds to process, while Isomap (with the second best result) took 18.8594 seconds to process.

For non-linear data structures, LLE is the best choice for dimensionality reduction among the four methods here.

Advantages of LLE

There are many advantages for using LLE:

- The method is simple and easy to use.
- There is only 1 free parameter, k, the number of nearest neighbors.
- Optimizations do not involve local minima.
- Local distances of the original high dimensional space are preserved.
- Computations are fast since LLE tends to accumulate sparse matrices, reducing calculation time and computer space.

Disadvantages of LLE and Future Work

There are still some problems with LLE that can be improved in the future:

- LLE is sensitive to noise or outliers.
- For LLE methods that use k-nearest neighbors search by Euclidean distances, it is possible to have short-circuit problems occur (a point's neighbor is mixed with different types of points and this results in points being grouped as neighbors, but they do not actually lie on the same locally linear patch).
- LLE is an unsupervised method and assumes that data points are on or near a manifold, which may not be the case in certain situations, such as in multi-class classification problems.

References

L. van der Maaten. Matlab Toolbox for Dimensionality Reduction. http://lvdmaaten.github.io/drtoolbox/.

Y. LeCun, C. Cortes, and C. Burges. The MNIST Database. http://yann.lecun.com/exdb/mnist/index.html.

S. T. Roweis and L. K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290, 2323. 2000.

Ali Ghodsi. Dimensionality Reduction a Short Tutorial. Department of Statistics and Actuarial Science University of Waterloo, Waterloo, Ontario, Canada, 2006.

Joshua Tennenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science, vol. 290, pp. 2319-2323, December 2000.

Multidimensional Scaling, Chapter 3. University of Bristol. http://www.bristol.ac.uk/media-library/sites/cmm/migrated/documents/chapter3.pdf.

Dan Ventura. Manifold Learning Examples - PCA, LLE and ISOMAP. Department of Computer Science, Brigham Young University, Provo, UT, USA, October 2008.

L.J.P. van der Maaten, E.O. Postma, and H.J. van den Herik. Dimensionality Reduction:A Comparative Review. MICC, Maastricht University, Maastricht, The Netherlands, January 2008.

Quan Wang. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models. Rensselaer Polytechnic Institute, Troy, NY, USA, August, 2014.