67750: Advanced Practical Machine Learning 2017/8

Exercise 1 — Manifold Learning

Dr. Matan Gavish *TA: Daniel Gissin*

1 Theoretical Questions

Note that this section is not graded and you do not have to hand it in.

1.1 PCA

In PCA we diagonalize the empirical covariance matrix of our data, $S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$. Assume the data is centered, meaning $\bar{x} = 0$.

1. Show that the data sits on a d-dimensional subspace $V \subset \mathbb{R}^n$ if and only if S is of rank d.
2. Show that the new coordinates are the result of an isometry on the subspace V.

1.2 MDS

1. Show that if the data sits on a d-dimensional subspace of $\mathbb{R}^n$, then the matrix that MDS diagonalizes is of rank d. Assume the data is centered, meaning $\bar{x} = 0$. (Hint: show that we are actually diagonalizing XX^T)
2. Show that the new coordinates are the result of an isometry on the subspace V.

1.3 LLE

In LLE, we obtain an n by n matrix, W, where ω_{ij} is the weight of data point j we use to reconstruct data point i. We also enforce the constraint that each row of W sum to 1. We then minimize

$$\Phi(Y) = \Sigma_{i=1}^{n}(y_i - \Sigma_{j=1}^{n}\omega_{ij}y_j)^2$$

where Y is the n by 1 matrix (we will restrict ourself to the 1 dimensional reduction for simplicity). In class, we showed this to be equivalent to minimizing

$$\Phi(Y) = Y^T M Y$$

where

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

Also, we saw in class that minimizing RSS_i is equivalent to minimizing $w_i^T G w_i$ (slide 21).

1. Show that $\Phi(Y) = \Phi(Y + c1)$, where c is any constant.

2. Why do we need to impose the constraint that $\frac{1}{n}\Sigma_{i=1}^{n} y_i^2 = 1$ and that $\sum_{i=1}^{n} y_i = 0$?

3. Show that G is invertible iff the k neighbors of x_i are linearly independent.

4. Assuming that is the case, solve the lagrangian to find the optimal weights. Show that $w_i = \frac{\lambda}{2} G_i^{-1}\mathbf{1}$

2 Practical Exercise

Please submit a single tar file named ”ex1_<YOUR_ID>.tar.gz”. This file should contain your code, along with an ”Answers.pdf” file in which you should write your answers and provide all figures/data to support your answers (write your ID in there as well, just in case). Your code will be checked manually so please write readable, well documented code.

The exercises in our course are written with Python3 in mind, but if you wish to write your code in another language, please contact the TA.

If you have any constructive remarks regarding the exercise (e.g. question X wasn't clear enough, we lacked the theoretical background to complete question Y...) we'll be happy to read them in your Answers file.

2.1 Exercise Requirements

1. Implement the MDS algorithm (10 points)
2. Generate scree plots and distance scatter plots to learn more about MDS (10 points)
3. Implement the LLE algorithm (25 points)
4. Implement the Diffusion Map algorithm (25 points)
5. Explore different data sets using the implemented algorithms (30 points + 5 bonus points)

2.2 Data sets

In this exercise we will use the algorithms we learned in class to reduce the dimensionality of several data sets, while trying to preserve their innate structure.

2.2.1 MNIST Digits

The MNIST data set is a collection of hand written digits (0 through 9) and is one of the most frequently used data sets for testing machine learning algorithms. We will attempt to visualize it in two dimensions while still maintaining the clusters of the original labels.

We will use a lower resolution MNIST, which can be accessed using the scikit-learn Python package. An example for loading the data can be found in the complimentary code (the digits_example function).

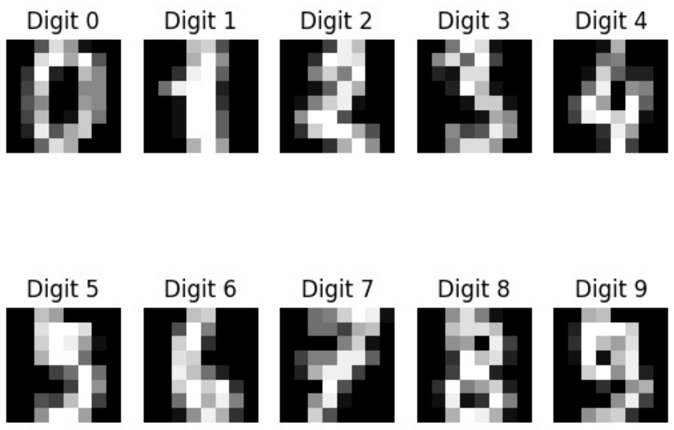

Figure 1: An example of the low resolution MNIST data set

2.2.2 Swiss Roll

The Swiss Roll is an artificially generated data set of two dimensional data (a rectangle) embedded in three dimensions as a spiral. If we are able to successfully approximate the geodesic distances in the original plane, we should be able to embedd the data in two dimensions as the original rectangle.

This data set can also be generated easilly using scikit-learn, and an example of using and plotting the swiss roll can be found in the complimentary code (the swiss_roll_example function).

2.2.3 Faces

The data set consists of simple 64x64 pictures of a face from different angles and lighting. Since the changes in lighting and angle are gradual, we should be able to detect this local structure and the highly non linear but low dimensional degrees of freedom in the data can be embedded well in low dimension.

This data is given to you in this exercise in the file "faces.pickle". An example of loading the data can be found in the complimentary code (the faces_example function).

2.3 Review Of The Algorithms

Just to make sure we aren't lost, here is a quick review of the steps of the algorithms:

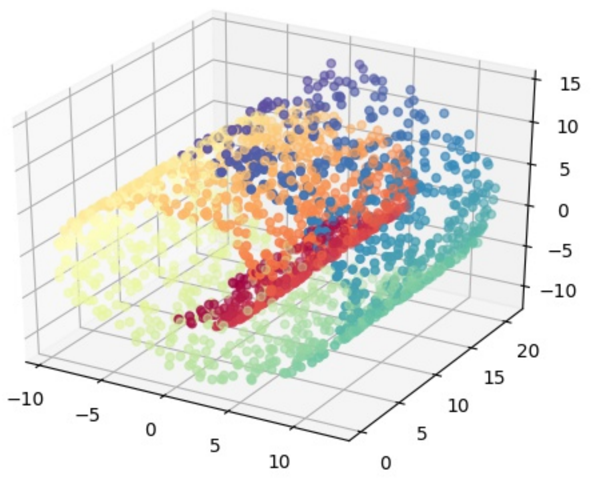

Figure 2: The Swiss Roll data set - a rectangle embedded non linearly in 3D

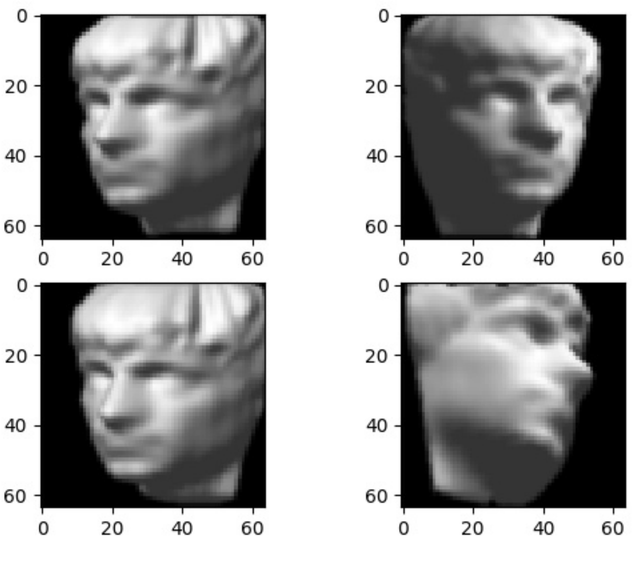

Figure 3: The Faces data set

2.3.1 MDS

Given a data matrix:

1. Compute the squared euclidean distance matrix $\Delta_{i,j} = ||x_i - x_j||^2$
2. From the distance matrix, form the matrix $S = -\frac{1}{2}H\Delta H$. ($H = I - \frac{1}{n}11^T$)
3. Diagonalize S to form $S = U\Lambda U^T$
4. Return the $n \times d$ matrix of columns $\sqrt{\lambda_i}u_i$ for $i = 1...d$.

2.3.2 LLE

Given a data matrix:

1. Compute the KNN graph of the data matrix (you may use the naive $O(n^2p^2)$ approach of calculating pairwise distances).
2. Compute W using the inverse of the Gram Matrix of each points nearest neighbors (note - the rows of W can be calculated independently from one another).
3. Decompose $M = (I - W)^T(I - W)$ into its eigenvectors and return the ones corresponding to the 2...$(d + 1)$ lowest eigenvalues (the lowest one has eigenvalue 0 and is discarded).

2.3.3 Diffusion Map

Given a data matrix:

1. Create a kernel similarity matrix K, using the heat kernel.
2. Normalize the rows of K to form the Markov Transition Matrix A.
3. Decompose A into its eigenvectors and select only the ones corresponding to the 2...$(d + 1)$ highest eigenvalues.
4. Return those eigenvectors, where the i^{th} eigenvector is multiplied by λ_i^t.

2.4 Implementing The Algorithms

Implement MDS, LLE and Diffusion Maps as instructed in "Manifold_Learning.py".

2.4.1 Scree Plot

As we discussed in class, a common way for deciding which dimension we should reduce our data to when using PCA or MDS, is looking at the eigenvalues of the eigendecomposition of the matrix in the algorithm (covariance matrix for PCA, centered distance matrix for MDS). When we see a point where the eigenvalues become low compared to previous eigenvalues, that's where we decide to stop.

Create a random 2-dimensional data set and embed it in a higher dimension using a random rotation matrix, then add Gaussian noise to your new data set. Test varying degrees of noise and see how the eigenvalues of MDS change as the noise increases.

A reminder - you can obtain a random rotation matrix by creating a random Gaussian matrix, then performing a QR decomposition. The Q matrix you get from the QR decomposition will be your rotation matrix.

2.4.2 Lossy Compression of Distances Information

We want to see how our information about the distances between points in the data set changes as we reduce the dimensions. Create a random data matrix drawn from a Gaussian distribution. Plot the $\binom{n}{2}$ pair-wise distances of your data set for different dimensions.

2.4.3 MNIST

Compare the results of the three algorithms on the MNIST data set. Discuss your results.

2.4.4 Swiss Roll

Compare the results of the three algorithms on the Swiss Roll data set.

Discuss the reason for the poor results of MDS.

Between LLE and DM, which algorithm required less parameter tweaking to reach good results? Which parameters are most important for an artificial data set like this?

2.4.5 Faces

Compare the results of the three algorithms on the Faces data set.

Were you able to extract the lower dimensional degrees of freedom of the data?

Show plots and discuss the embeddings. You may use the supplied "plot_with_images" function to show the original pictures superimposed on a scatter plot of the lower dimensional data extracted from the algorithm.

2.4.6 Parameter Tweaking

Discuss how the k-neighbors parameter of LLE effects the results of the algorithm. What is the problem with choosing a k which is too large or too small?

Show plots to support your claims.

2.4.7 Bonus (5 points)

Use the supplied Pac Man picture (or any other small, simple picture you prefer) and create a data set of several hundred pictures by rotating it by different angles.

Show that you can successfully obtain the one dimensional intrinsic structure of the data by using the algorithms learned in class (you can plot your results in either 2D or 1D).

2.4.8 Main Function

Finally, please write a basic main function which demonstrates your implementation.

2.5 Feedback

We would be happy to hear constructive comments about this exercise in your Answers.pdf file. Specifically:

1. How long did it take to complete the exercise?
2. Were there parts where you felt that you were working too hard on something that didn't teach you enough?
3. Did you feel that you lacked background for any of the questions?