67750: Advanced Practical Machine Learning 2020/21

Exercise 4 — Manifold Learning

Dr. Matan Gavish *TA: Omri Bloch*

1 Preface

Hi all

I am glad to present the third exercise of the course!

As a warm-up you will solve few theoretical questions, implement MDS and Diffusion Map and apply them to demo-datasets. The main part of this exercise is to use the algorithms you have learned in order to visualize and explore the Netflix Prize Dataset (this is the main part also in the grade). the third question in the theoretical part is time consuming and I recommend approaching it only if you have time left after you finished the rest of the exercise.

our code-sharing guidelines had changed a little in this exercise. If you have some file or function for reading/filtering/visualizing/other non-core thing and you want to share it with others, please send me (Omri) a message and I will approve sharing it in the Piazza.

This time You are limited to four pages in your submission, not including the theoretical part. Pay attention that playing with parameters of the algorithms can consume any amount of time and it is OK to submit visualizations that are not perfect.

good luck!

Part I

Theoretical Questions - Manifold Learning

2 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 S is a PSD matrix.
2. Show that the data sits on a d-dimensional subspace $V \subset \mathbb{R}^n$ if and only if S is of rank d. hint: show that S and X have the same rank.
3. Show that the new coordinates are the result of an isometry on the subspace V.

3 LLE

In this question we will walk through the derivation of the second stage of LLE (finding the W matrix). In class, we saw that the goal of the second stage of LLE is to describe every point x_i as an affine combination of its k nearest neighbors:

$$W_{i,:} = \underset{w}{argmin} ||x_i - \sum_{j \in N(i)} w_j x_j||^2 = \underset{w}{argmin} || \sum_{j \in N(i)} w_j z_j||^2$$

Where we define $z_j = x_j - x_i$.

Define G to be the Gram matrix of the z vectors, namely $G_{a,b} = z_a^T z_b$.

1. Show that $|| \sum_{j \in N(i)} w_j z_j||^2 = w^T G w$.
2. The above means that we can find w by minimizing the quadratic form $w^T G w$, under the constraint that $\sum_i w_i = 1$. Formulate the Lagrangian and derive the solution:

$$w = \frac{\lambda}{2} G^{-1} \mathbf{1}$$

Note that we do not have to analytically derive λ - it is only there to make sure that the constraint holds up (that w sums to 1). This means we can calculate $w = G^{-1}\mathbf{1}$ and then just normalize the vector to sum to one...

4 Diffusion Maps

Read the first four pages of "Diffusion Maps, Spectral clustering and Eigenfunctions of Flokker-Planck Operator" and repeat here the proof of the diffusion map theorem.

Part II

First practical part - getting to know the algorithms

Please submit a single tar file named "ex4_<YOUR_ID>". 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.

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.

Requirements

1. Implement the MDS algorithm
2. Generate scree plots to learn more about MDS
3. Implement the Diffusion Map algorithm
4. Demonstrate MDS, LLE and Diffuison Map on demo data sets. use LLE implementation from sklearn.

Datasets

In this exercise you will use the algorithms we learned in class to reduce the dimensionality of several data sets, while trying to preserve their innate structure. As warm-up you will play with a few toy-dataset, and after that you will get to the Netflix Prize dataset.

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 embed the data in two dimensions as the original rectangle.

This data set can also be generated easily 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).

Diffusion map will probably fail on this dataset.

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

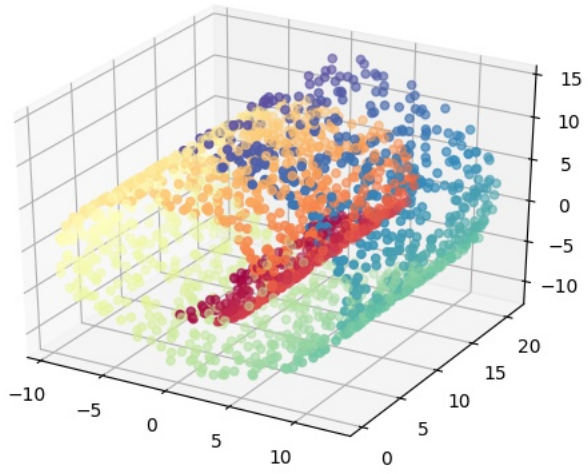

Figure 1: The Swiss Roll data set - a rectangle embedded non linearly in 3D

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).

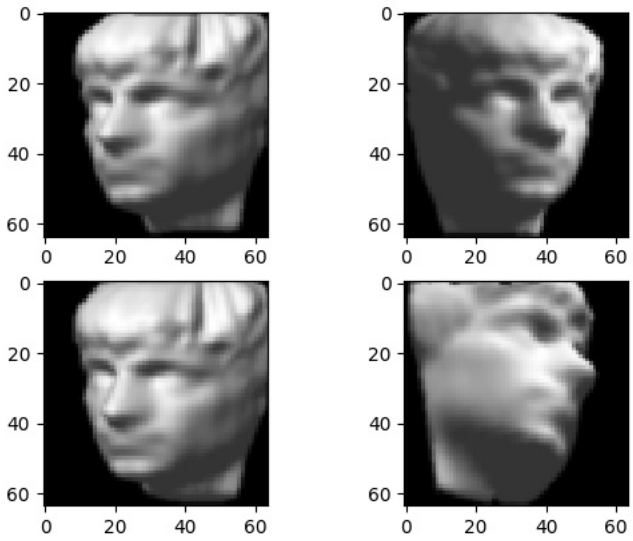

Figure 2: The Faces data set

Review Of The Algorithms

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

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 $i = 1...d$ (the eigenvectors corresponding to the largest eigenvalues).

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.

5 Implementing The Algorithms

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

6 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.

7 demonstrate your algorithms on the following problems

7.1 Swiss Roll

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

Discuss the reason for the poor results of MDS.

7.2 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.

Part III

Main practical part - Netflix Prize Dataset

8 Netflix Prize Dataset

Demonstrate the usage of manifold learning algorithms on the Netflix Prize Dataset. you are welcome to use other algorithms and other implementations of the same algorithms. We expect high variability in the answers, and failures are acceptable.

You can share code of pre-processing and visualization after getting approval from me. You are welcome to edit and add sections to the code your friends shared.

8.1 I expect to see at least:

1. Two visualizations of the same data using two manifold learning techniques, explained, with comparison. If you manage to extract some interesting intrinsic structure of the data, I expect you to look at the movies names and try to understand what this structure means.
2. Clustering above one of these visualizations, including explanation for how you chose the number of clusters.
3. Explained pre-processing pipeline.

8.2 A few tips and suggestions:

1. I highly recommend changing pycharm settings to work with interactive plots.
2. I recommend using the library pydiffmap for diffusion maps, since they have both automatic calibration of epsilon and out-of-sample embedding.
3. sklearn are great
4. You don't have to use all the data. It is recommended to filter the data in a meaningful way and then to sub-sample.
5. Pre-processing tips: Plot some histograms and basic statistics before you start.
6. PCA the data before using non-linear algorithms, the dimension is just too high.

9 Bonus

you can get small bonus for:

1. sharing useful code in the piazza (after getting my approval)

2. implementing diffusion maps above nearest neighbors
3. explaining in the piazza how out-of-sampling embedding is implemented
4. explaining in the piazza how the bgh method for choosing epsilon works