ISOMAP-Copy1

September 2, 2021

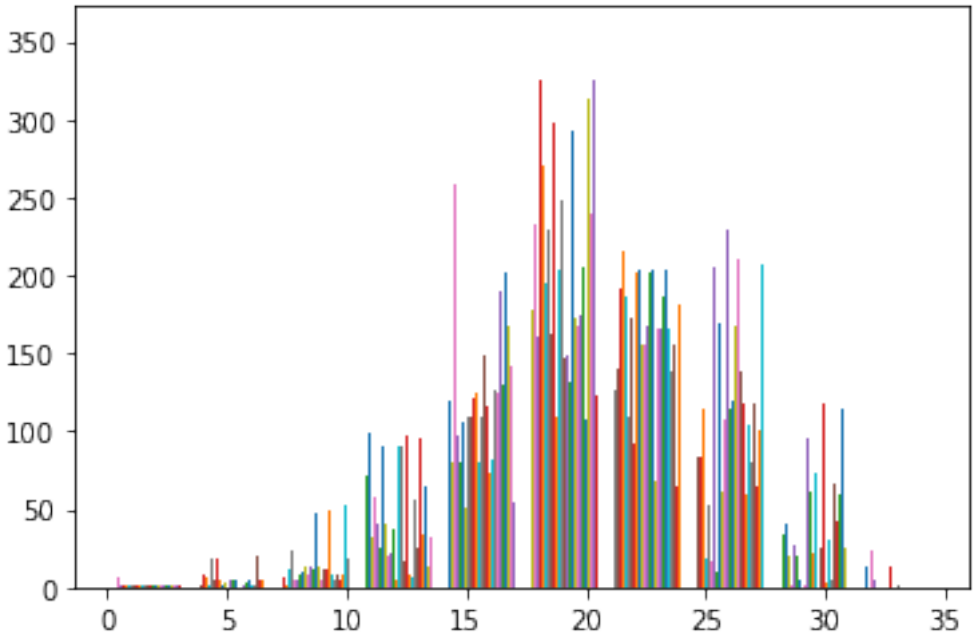
0.0.1 Implementation of ISOMAP algorithm and comparing results with PCA for ISOMAP image dataset

Function definition to plot images in a scatter plot

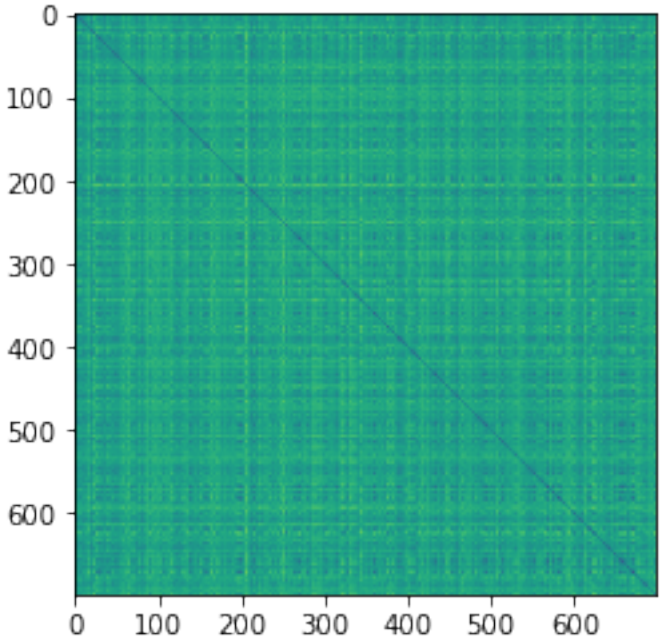
Visualizing the images in the isomap dataset in a nearest neighbor graph format and Adjacency matrix First, I am importing the isomap.dat data and analyzing the shape. This image file is a (4096, 698) matrix, where there are 698 images and each image is represented by 4096 features. To create the adjacency matrix, I am using scipy.spatial.distance.cdist library using "Euclidean" distance as the metric to find the distance between each data point (as we will be creating an Epsilon-ISOMAP algorithm).

After tuning, the Epsilon value of 11 gives the best performance. The points that are within epsilon are connected and the points that are beyond epsilon are not connected. It was observed that increasing epsilon value created dense graph with more connected nodes.

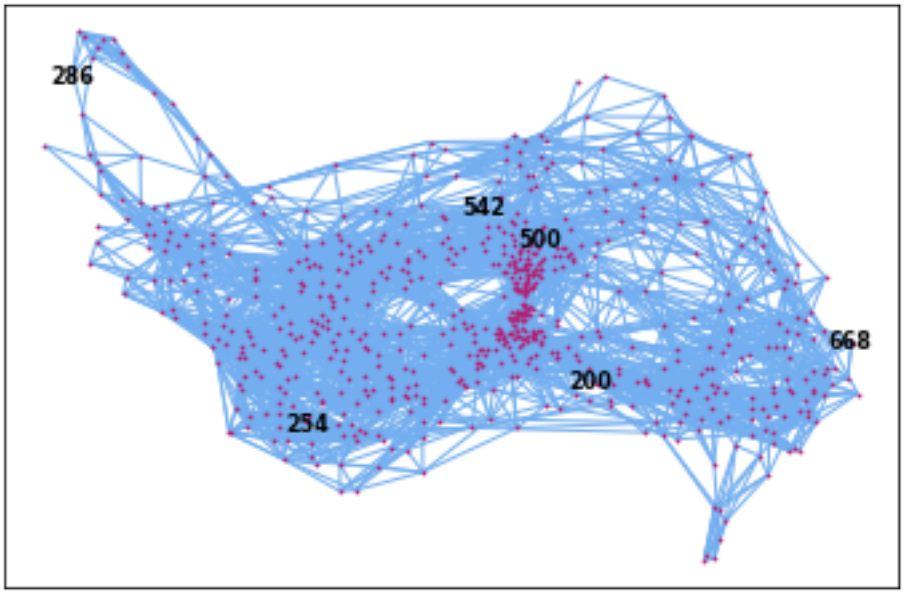
```
(array([[  1.,   3.,  20., …,  13.,   2.,   0.],
        [  1.,   1.,   4., …,  40.,   5.,   0.],
        [  2.,   5.,  16., …,  14.,   1.,   0.],
        …,
        [  4.,  19.,  49., …, 123.,  59.,  13.],
        [  1.,   1.,   6., …, 134.,  78.,   4.],
        [  2.,   1.,  18., …,  97.,  25.,   0.]]),
 array([ 0.        ,  3.46871685,  6.9374337 , 10.40615054, 13.87486739,
        17.34358424, 20.81230109, 24.28101794, 27.74973479, 31.21845163,
        34.68716848]),
 <a list of 698 Lists of Patches objects>)
```



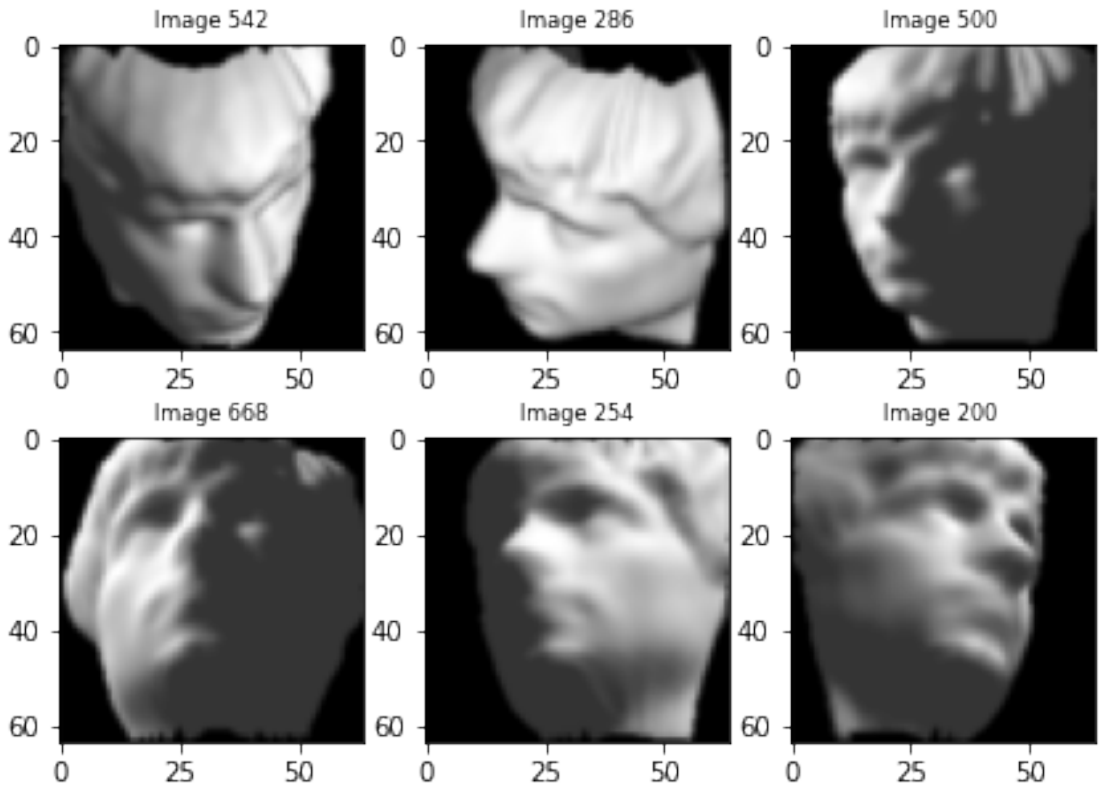

```
<matplotlib.image.AxesImage at 0x251654d6310>
```



Below is a graph representation of the adjacency matrix (using network), and a few data points that are numbered in the graph displayed in cell 7 results

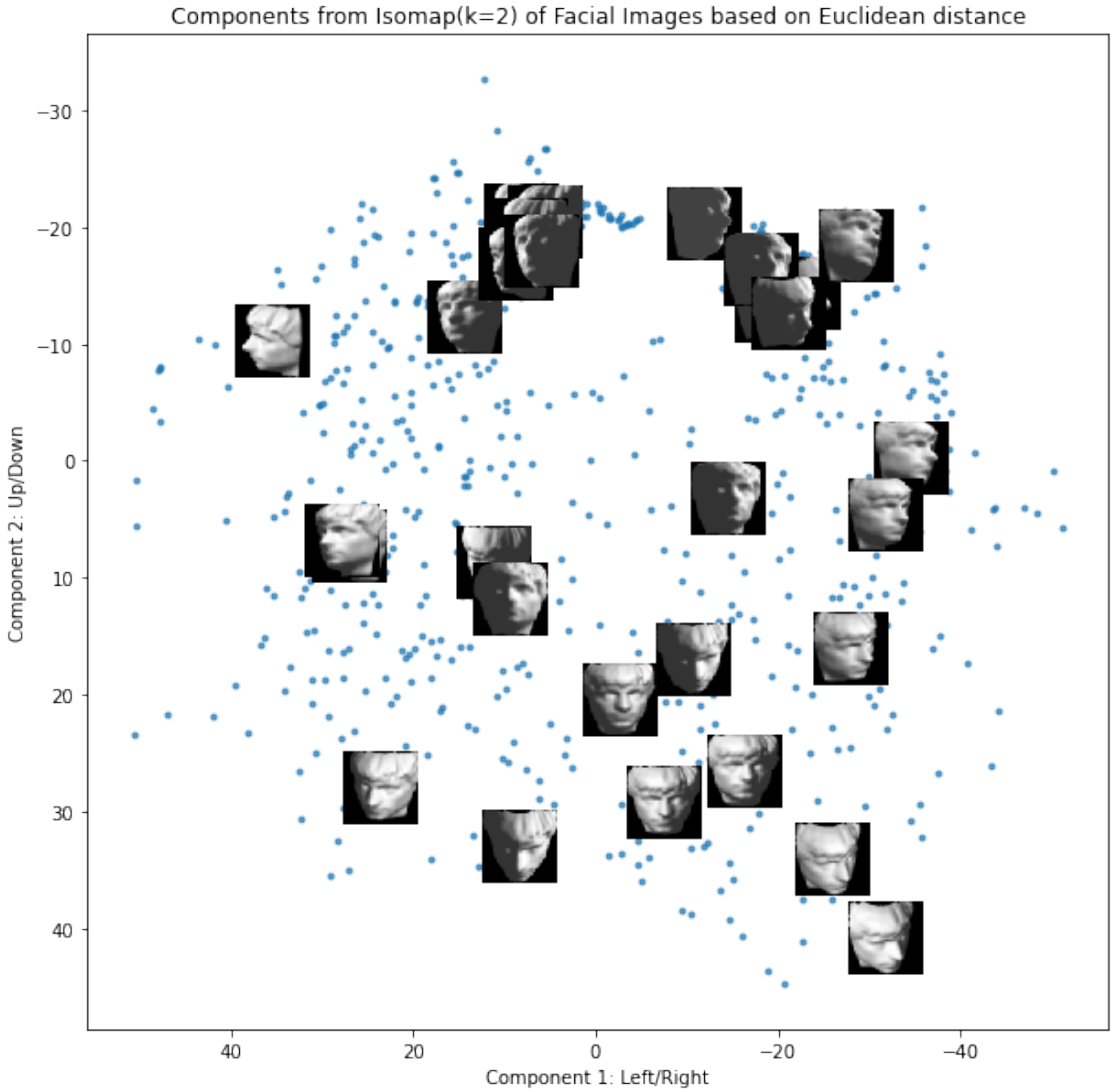


Understanding the different orientations of input images in the dataset We see that the most common orientations are images facing left, right up and down



Implementing ISOMAP algorithm using Euclidean distance metric To implement the ISOMAP algorithm, shortest distance matrix is created from the adjacency matrix of first step. After this, the centering matrix is computed which helps to calculate the C matrix, on which Eigen decomposition can be applied. He we are creating a 2-dimensional embedding of the image, which can be plotted on a scatter plot to understand any specific similarity in the data.

Below is the scatter plot of the 2D embedding space created after performing ISOMAP (Note that the images are displayed on the plot using random function and the data points will change for every run).

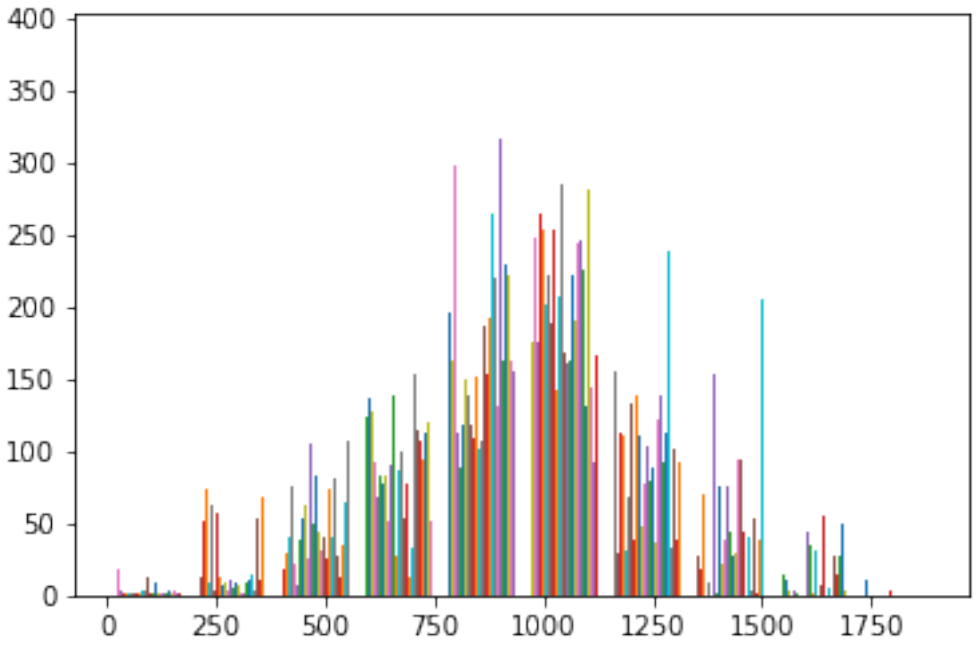


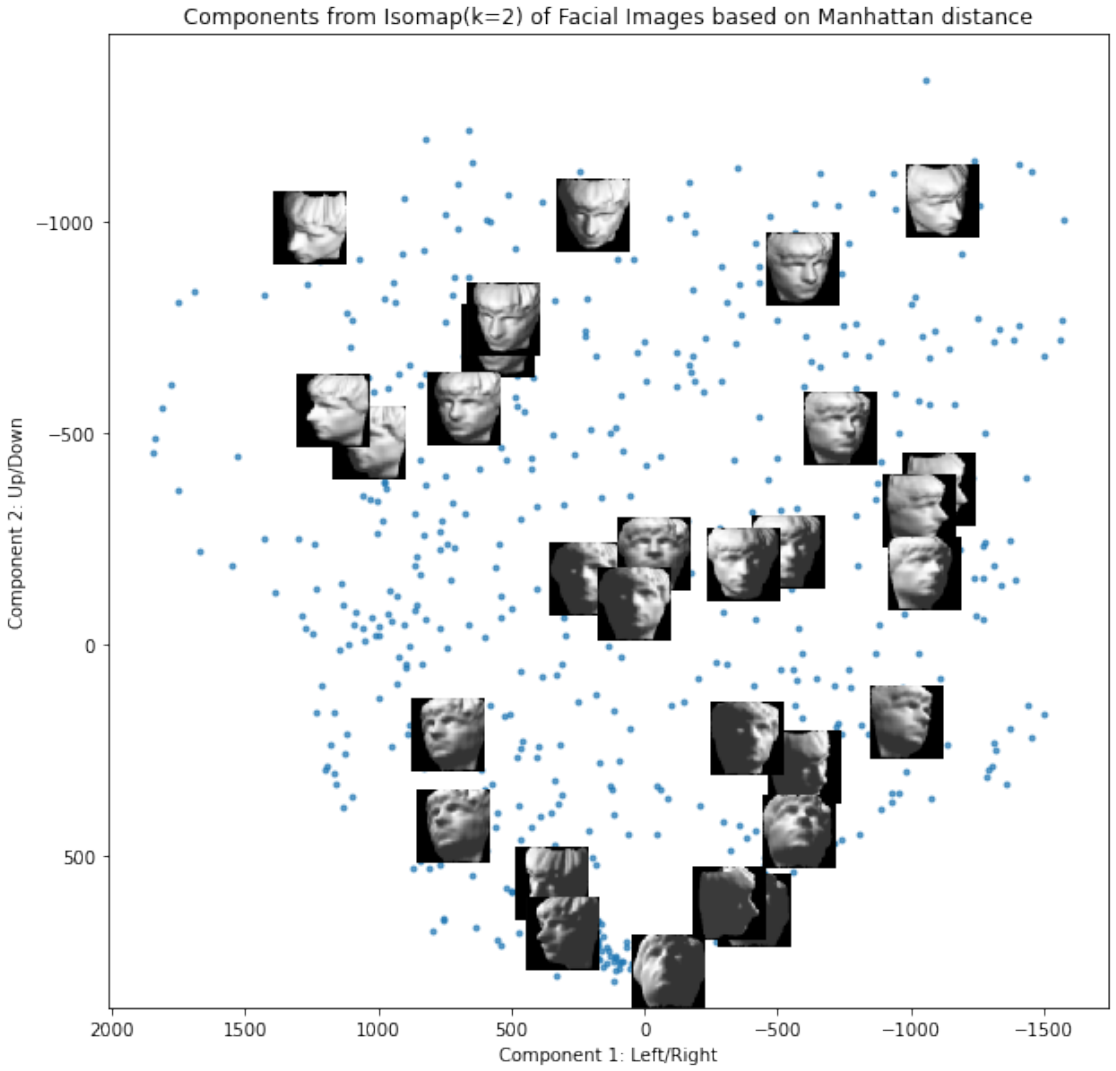
Observation : We can observe from the direction of the scatter plot that in the X-axis, the images that face left are in one direction and the images that face right are in another direction. Similarly, in Y-axis, the images that face up are in one direction and the images that face down are in the other direction. We can also observe data is clustered closely (in the top) where features are along the same direction (facing left) or facing right. This can be related to the fact that the orientation pattern or similarity in the data is captured by this algorithm.

Implementing ISOMAP algorithm using Manhattan distance metric Replacing the L2 distance with Manhattan (L1 distance), I have re-run the epsilon ISOMAP to obtain the 2-Dimensional embedding. Below is the scatter plot of the first 2 Eigen vectors, labeled with their corresponding images.

The Epsilon cut of distance value for this had to be increase to be in the range of the Manhattan distances of the data points (Epsilon = 550).

```
(array([[  1.,  13.,  68., …,   4.,   0.,   0.],
        [  2.,   6., 115., …,  16.,   0.,   0.],
        [  2.,  13.,  65., …,   4.,   0.,   0.],
        …,
        [ 20.,  59.,  83., …,  73.,  23.,   1.],
        [  1.,   5.,  33., …,  71.,  42.,   2.],
        [  3.,  29., 108., …,  55.,   6.,   0.]]),
 array([   0.        ,  190.11951593,  380.23903186,  570.35854779,
         760.47806373,  950.59757966, 1140.71709559, 1330.83661152,
        1520.95612745, 1711.07564338, 1901.19515931]),
 <a list of 698 Lists of Patches objects>)
```





Comparing results from Euclidean distance and Manhattan distance The plot looks similar to the plot obtained by using Euclidean distance. In the X-axis, the images that face left are in one direction and the images that face right are in another direction. Similarly, in Y-axis, the images that face up are in one direction and the images that face down are in the other direction.

Some differences we can observe is that, the scale is larger in Manhattan distance (which is reasonable, since Euclidean distance is the shortest distance between two points). Also, the data points seem to be more scattered than the previous plot. With the Euclidean distance, the points were closer, with some clusters visible.

Implementing PCA on the image dataset and plotting the Principal components in 2-dimensions Now, we perform PCA on the same data and observe the 2-Dimensional scatter plot with the top 2 principal components. We use the same algorithm used in the first problem for this (calculating the covariance matrix, and performing eigen decomposition.

Below is the scatter plot of the top 2 principal components and their corresponding images.

```
<matplotlib.collections.PathCollection at 0x19e8a0a9a00>
```

Observation of PCA results We can see from the scatter plot that the data points are not meaningful using PCA. To the left, on the X axis, we can see both left and right orientation of the image. Similarly, on the Y-axis to the top, we can see both up/down facing images. We cannot observe any pattern or similarity in the data. ISOMAP helps us to identify patterns and some clusters, whereas PCA does not accomplish this since it only assumes linear projections when variables are linearly correlated and not with non-linear structures in the data.

0.0.2 Reference

Code from below link was used as a reference for scatter plot representation http://benalexkeen.com/isomap-for-dimensionality-reduction-in-python/