# Problem Set 6

This homework you implement 3 algorithms. ISOMAP, PCA and MDS

Your code must be all written within the file HW6.py provided to you. The drawing part is also provided in the file.

**Part 1 (5 points) Implement ISOMAP**

Implement the classical MDS algorithm.

This algorithm takes as input a distance matrix $D = \{d_{ij}\}$ (k by k matrix) and number of component d<n and returns a collection of points $\{y_i\}_{i=1}^{k}$ in the space $R^d$. The steps of the algorithm are given as follows:

1-Construct the matrix of squares of the distances $P^{(2)} = [d_{ij}^2]$.

2. Apply the double centering: $\mathrm{B} = -\frac{1}{2}JP^{(2)}J$ where $J = I - \frac{1}{n}11'$ , where N is the number of elements. The definition of
$1\ and\ 1'\ can\ be\ found\ in\ the\ lecture\ (see\ the\ example\ in\ the\ lecture)$.

3. Extract the largest d positive eigenvalues $\lambda_1 \ldots \lambda_d$ of $B$ and the corresponding m eigenvectors $e_1 \ldots e_d$.

4. Construct the matrix $X = E_d\Lambda_{\mathrm{d}}^{\frac{1}{2}}$ , where $E_d$ is the matrix of d eigenvectors and $\Lambda_d$ is the diagonal matrix of d eigenvalues of $B$, respectively

An example of the above steps is given in the lecture so you may refer back to the lecture and use the example as a guide when you do the above implementation.

Note 1: you will need to use an eigensolver for to do step 3 in the above algorithm. One common choice is here. Also, you can use the following function to do step 2.

Note 2. Use the data set given in this example here as a testing data set. Use d=2 when do the testing.

**Part 2 (4 points) Implement ISOMAP and MDS**

Just like the previous algorithm, this algorithm takes as input a collection of points **X**=$\{x_i\}_{i=1}^{k}$ in $R^n$ and number of component d<n, and the number of K nearest neighbors in the K-NN graph and returns a collection of points $\{y_i\}_{i=1}^{k}$ that represent the points $\{x_i\}_{i=1}^{k}$ in the space $R^d$. The steps of the algorithm are given as follows:

**1- Construct the neighborhood graph of the data X using the *K-NN neighborhood* graph we studied earlier in the course.**

**2- Use the Dijekstra algorithm or the Floyd–Warshall algorithm to find the distance between nodes on the graph.**

**3- Apply MDS on the distance matrix above and extract the coordinates with the desired dimension (part one of the algorithm). In step 1 and 2 you may use the function in sklearn and networkx that we have utilized before in earlier homeworks.**

Note 2. Use the data set given in this example here as a testing data set. Use d=2 when do the testing. Use K=10.

**Part 3 (3 points) Compare your results with the results in** here

The following code imports a points cloud with the shape "S" that we studied many times during the lecture :

```
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

from matplotlib.ticker import NullFormatter


from sklearn import manifold, datasets
```

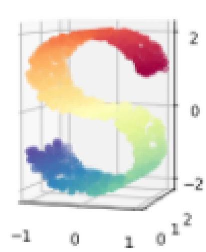
```
# Next line to silence pyflakes. This import is needed.
Axes3D

n_points = 1000
X, color = datasets.samples_generator.make_s_curve(n_points, random_state=0)
n_neighbors = 10
n_components = 2

fig = plt.figure(figsize=(15, 8))
plt.suptitle("Manifold Learning with %i points, %i neighbors"
             % (1000, n_neighbors), fontsize=14)


ax = fig.add_subplot(251, projection='3d')
ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=color, cmap=plt.cm.Spectral)
```

**It will also plot the data X for you.**




**Use the data X above as input for your MDS and ISOMAP functions and compare your results to the results obtained in the sklearn example [here](#).**

**Remark : MDS takes as input the distance matrix of X and not X itself so make sure you pass the distance matrix of X instead of X itself when you do your computation in this example.**

**Part 4 (3 points) PCA**

Use PCA to reduce the dimension of the data set “S” to 2 dimension. Plot the result point cloud and explain your observation. (does PCA do a good job here ? why ? )

PCA can be found.