ISYE 6740 Fall 2025
Homework 2
(100 points + 5 bonus points)

Provided Data: Questions marked with GS must be submitted to Gradescope. You must still include all your results and explanations in this PDF, and include your code in your submitted zip. Failure to pass all gradescope tests will result in losing 50% of the points for that question.

- (GS) Q3: Order of faces using ISOMAP [isomap.dat, isomap.mat]
- (GS) Q4: Eigenfaces and simple face recognition [yalefaces]

All results that are present only in code/notebooks, but not in your PDF report will not be accepted for points.
Handwritten solutions will not be accepted for any reason
For any code that requires randomness, please set your seed as 6740

1. Conceptual questions [30 points].

1. (5 points) Please prove the first principle component direction v corresponds to the largest eigenvalue of the sample covariance matrix:

$$v = \arg \max_{w:\|w\|\leq 1} \frac{1}{m} \sum_{i=1}^{m} (w^T x^i - w^T \mu)^2.$$

   You may use the proof steps in the lecture, but please write them logically and cohesively.

2. (5 points) Based on your answer to the question above, explain how to further find the second and third largest principle component directions.

3. (5 points) Consider the diagonal matrix $A = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$. Find two distinct eigenvalue decompositions of A, and prove mathematically that both are valid. This demonstrates that eigendecomposition is not unique. **Note:** Your eigenvalue decompositions must be done mathematically, not programatically.

4. (5 points) Explain the three key ideas in ISOMAP (for manifold learning and non-linear dimensionality reduction).

5. (5 points) Explain how to decide k, the number of principle components, from data.

6. (5 points) How do outliers affect the performance of PCA? You can create numerical examples to study and show this.

2. Graph Laplacian and GNN [20 points].

This question is designed to show how Graph Laplacian and the principle of spectral clustering is used in the context of deep learning, in particular, in developing graph neural networks (GNN). In GNN, such as ChebNet (one of the most popular GNNs), filters are defined as *polynomials of the graph Laplacian*, to extract graph features. This question is meant to build the understanding of how properties from Graph Laplacian can be used in GNN. You can refer to the paper referenced below for additional context.
Ref: M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. NeurIPS, 2016.

Formally, if L is the graph Laplacian and $T_k(\cdot)$ is the Chebyshev polynomial of order k, then

$$g_\theta(L)\,x \;\approx\; \sum_{k=0}^{K} \theta_k\, T_k(\tilde{L})\,x, \qquad \tilde{L} = \tfrac{2}{\lambda_{\max}} L - I.$$

where $\lambda_{\max}$ is the largest eigenvalue of the Graph Laplacian. Here, "a polynomial applied to the graph Laplacian" means taking powers of L (e.g., $I, L, L^2, \dots$) and combining them with learnable coefficients θ_k. Recall from lecture that $L = D - A$, where D is the degree matrix and A is the adjacency matrix.

Example (Chebyshev, small K). Take $K = 2$ and use $T_0(x) = 1$, $T_1(x) = x$, $T_2(x) = 2x^2 - 1$. Then

$$g_\theta(L)\,x \;\approx\; \theta_0\, T_0(\tilde{L})\,x \;+\; \theta_1\, T_1(\tilde{L})\,x \;+\; \theta_2\, T_2(\tilde{L})\,x.$$

For a toy 3-node path with

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad L = D - A = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix},$$

one computes

$$L^2 = \begin{bmatrix} 2 & -3 & 1 \\ -3 & 6 & -3 \\ 1 & -3 & 2 \end{bmatrix}, \qquad T_2(L) = 2L^2 - I = \begin{bmatrix} 3 & -6 & 2 \\ -6 & 11 & -6 \\ 2 & -6 & 3 \end{bmatrix}.$$

Thus, a single ChebNet layer with $K = 2$ applies a learnable combination of $\{x,\ Lx,\ T_2(L)x\}$ to produce updated node features—no eigenvectors required.

How this is used in a GNN layer.

- *Input:* a matrix $X \in \mathbb{R}^{n \times d}$ of node features (one row per node).
- *Operation:* build $\{T_k(\tilde{L})X\}_{k=0}^{K}$ (each mixes information up to k hops), linearly combine them with learnable weights, and pass through a nonlinearity.
- *Output:* new node features $X' \in \mathbb{R}^{n \times d'}$ that aggregate information from multi-hop neighborhoods.

Questions. In this exercise, we will show how GNN is constructed based on the idea of graph Laplacian, the same idea we exploited in spectral cluster.

(1) (10 Points) **Neighborhood aggregation.**

(a) Show that

$$(Lx)_i = d_i x_i - \sum_{j \sim i} x_j,$$

so multiplying by L mixes node i's value with its **1-hop neighbors**.

(b) Explain why L^2x includes terms that traverse **paths of length up to 2** (so information propagates up to 2-hops).

(c) Conclude: why does applying a polynomial in L aggregate information from **multiple hops** in one shot?

> Hint
>
> Recall: the off-diagonal entries of L are negative when nodes are connected, and diagonal entries contain the degree. Multiplying by L subtracts neighbor contributions and scales by the degree. Multiplying again by L compounds this mixing, reaching farther neighbors.

(2) (10 Points) **Small example calculation.** Using the 3-node path above, compute $L = D - A$. Then, with $T_0(x) = 1$, $T_1(x) = x$, $T_2(x) = 2x^2 - 1$, write down $T_2(L)$ explicitly.

3. Order of faces using ISOMAP [25 points]

This question aims to reproduce the ISOMAP algorithm results in the original paper for ISOMAP, J.B. Tenenbaum, V. de Silva, and J.C. Langford, Science 290 (2000) 2319-2323 that we have also seen in the lecture as an exercise (isn't this exciting to go through the process of generating results for a high-impact research paper!)

The file isomap.mat (or isomap.dat) contains 698 images, corresponding to different poses of the same face. Each image is given as a 64×64 luminosity map, hence represented as a vector in $\mathbb{R}^{4096}$. This vector is stored as a row in the file. (This is one of the datasets used in the original paper.) In this question, you are expected to implement the ISOMAP algorithm by coding it up yourself. You may find the shortest path (required by one step of the algorithm), using `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csgraph.shortest_path.html`.

Using Euclidean distance (i.e., in this case, a distance in $\mathbb{R}^{4096}$) to construct the ϵ-ISOMAP (follow the instructions in the slides.) You will tune the ϵ parameter to achieve the most reasonable performance. Please note that this is different from K-ISOMAP, where each node has exactly K nearest neighbors.

1. (5 points) Visualize the nearest neighbor graph through either an image of your adjacency matrix, or visualizing the graph similar to the lecture slides using graph packages such as Gephi (https://gephi.org). Additionally, include a few of the face images that correspond directly to nodes on different parts of your visualization.

2. (10 points) Implement the ISOMAP algorithm yourself to obtain a two-dimensional low-dimensional embedding. Plot the embeddings using a scatter plot, similar to the plots in lecture slides. Find a few images in the embedding space and show what these images look like and specify the face locations on the scatter plot. Comment on do you see any visual similarity among them and their arrangement, similar to what you seen in the paper? Come up with a way to tune the kernel bandwidth to have desired result.

3. (10 points) Perform PCA (you can utilize a PCA package for this) on the images and project them into the top 2 principal components. Again show them on a scatter plot. Explain whether or not you see a more meaningful projection using ISOMAP than PCA.

4. Eigenfaces and simple face recognition [25 points].

This question is a simplified illustration of using PCA for face recognition. We will use a subset of data from the famous Yale Face dataset.

Remark: You will have to perform downsampling of the image by a factor of 4 to turn them into a lower resolution image as a preprocessing (e.g., reduce a picture of size 16-by-16 to 4-by-4). In this question, you can implement your own code or call packages.

First, given a set of images for each person, we generate the eigenface using these images. You will treat one picture from the same person as one data point for that person. Note that you will first vectorize each image, which was originally a matrix. Thus, the data matrix (for each person) is a matrix; each row is a vectorized picture. You will find weight vectors to combine the pictures to extract different "eigenfaces" that correspond to that person's pictures' first few principal components.

1. (10 points) Perform analysis on the Yale face dataset for Subject 1 and Subject 2, respectively, using all the images EXCEPT for the two pictures named subject01-test.gif and subject02-test.gif. **Plot the first 6 eigenfaces for each subject.** When visualizing, please reshape the eigenvectors into proper images. Please explain can you see any patterns in the top 6 eigenfaces?

2. (10 points) Now we will perform a simple face recognition task.

   Face recognition through PCA is proceeded as follows. Given the test image subject01-test.gif and subject02-test.gif, first downsize by a factor of 4 (as before), and vectorize each image. Take the top eigenfaces of Subject 1 and Subject 2, respectively. Then we calculate the *projection residual* of the 2 vectorized test images with the vectorized eigenfaces:

   $$s_{ij} = \|(\text{test image})_j - (\text{eigenface}_i)(\text{eigenface})_i^T(\text{test image})_j\|_2^2$$

   Report all four scores: s_{ij}, $i = 1, 2$, $j = 1, 2$. Explain how to recognize the faces of the test images using these scores.

3. (5 points) Comment if your face recognition algorithm works well and discuss how you would like to improve it if possible.

5. To subtract or not to subtract, that is the question [Bonus: 5 points].

In PCA, we have to subtract the mean to form the covariance matrix

$$C = \frac{1}{m}\sum_{i=1}^{m}(x^i - \mu)(x^i - \mu)^T$$

before finding the weight vectors, where $\mu = \frac{1}{m}\sum_{i=1}^{m} x^i$. For instance, we let

$$Cw^1 = \lambda_1 w^1$$

where λ_1 is the largest eigenvalue of C, and w^1 is the corresponding largest eigenvector.

Now suppose Prof. X insisting not subtracting the mean, and uses the eigenvectors of

$$\tilde{C} = \frac{1}{m}\sum_{i=1}^{m} x^i x^{iT}$$

to form the weight vectors. For instance, she lets $\tilde{w}^1$ to be such that

$$\tilde{C}\tilde{w}^1 = \tilde{\lambda}_1 \tilde{w}^1$$

where $\tilde{\lambda}_1$ is the largest eigenvalue of $\tilde{C}$.

Now the question is, are they the same (with and without subtract the mean)? Is w^1 equal or not equal to $\tilde{w}^1$? Use mathematical argument to justify your answer.