Important Note

You must write your code independently. Do not copy the code from any source. To receive full credit, your code must be well-documented with comments. Your script **must** be uploaded as either a .py or .ipynb file to Brightspace. The evaluation will be 10-point scale.

Project Requirements

Due Date

Sunday, 11:59 PM EST, November 16th, 2025

- 1-2 page report, includes:
  - work done
  - results
  - discussion of results
  - 500 words of text max (excluding figure captions)
- One ZIP file containing commented, executable code

1 PageRank Algorithm (20 Points)

The PageRank algorithm, used by search engines like Google, ranks websites based on the number of links pointing to them. You can represent a simplified web network as a stochastic matrix where each entry $M[i, j]$ is the probability that a user on page j will click a link to page i.

Given the following matrix representing a small web network:

$$M = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

1. Use `scipy.linalg.eig` to compute the dominant eigenvector of the matrix M. This eigenvector represents the PageRank scores of the pages.
2. Start with an initial rank vector v (of ones) and iterate until convergence (i.e., until the difference between consecutive rank vectors is small).
3. Normalize the resulting eigenvector so that the sum of all ranks is 1. What do the entries of the eigenvector represent?
4. Based on the final PageRank scores, which page is ranked the highest, and why?

2 Dimensionality Reduction via PCA (20 points)

Principal Component Analysis (PCA) is widely used in data analysis for dimensionality reduction, often to compress data while preserving the most important patterns. You are given a dataset containing height and weight measurements for 100 people (already standardized):

$$\text{Data} = \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ \vdots & \vdots \\ h_{100} & w_{100} \end{bmatrix}$$

1. Compute the covariance matrix of the data.
2. Perform eigenvalue decomposition on the covariance matrix using `scipy.linalg.eigh`.
3. Identify the principal components and explain how they relate to the variance in the dataset.
4. Reduce the dataset to 1D by projecting it onto the principal component that captures the most variance. Plot the original data and the 1D projection.

3 Linear Regression via Least Squares (30 points)

A real estate company wants to predict house prices based on features like square footage, number of bedrooms, and age of the house. The company has the following dataset:

$$X = \begin{bmatrix} 2100 & 3 & 20 \\ 2500 & 4 & 15 \\ 1800 & 2 & 30 \\ 2200 & 3 & 25 \end{bmatrix}$$

The corresponding house prices (in \$1000s) are:

$$y = \begin{bmatrix} 460 \\ 540 \\ 330 \\ 400 \end{bmatrix}$$

1. Set up the system as a least-squares problem $X\beta = y$, where β represents the coefficients (weights) for square footage, bedrooms, and age.

2. Solve for β using `scipy.linalg.lstsq`.
3. Use the resulting model to predict the price of a house with 2400 square feet, 3 bedrooms, and 20 years old.
4. Discuss the role of the least-squares method in this prediction task and compare its performance to an alternative method (such as a direct solution via `scipy.linalg.solve`).

4 Gradient Descent for Minimizing Loss Function (30 points)

Given a matrix $X \in \mathbb{R}^{100\times 50}$, implement gradient descent to minimize the following loss function:

$$f(X) = \frac{1}{2}\sum_{i,j}(X_{i,j} - A_{i,j})^2$$

where A is a known matrix of the same size as X. This is essentially a mean squared error (MSE) loss function, where the goal is to make X as close as possible to a given matrix A.

1. Define the loss function:
$$f(X) = \frac{1}{2}\sum_{i,j}(X_{i,j} - A_{i,j})^2$$
2. The gradient of the loss function is:
$$\nabla f(X) = X - A$$
3. Implement gradient descent using `scipy.optimize.minimize` to minimize the loss function.
4. Stop the algorithm when the difference between the loss values in two consecutive iterations is smaller than a threshold (e.g., 10^{-6}), or after 1000 iterations.
5. Initialize the matrix X and A with random values.
6. Track the loss values over each iteration and visualize the results.