Statistical Learning and Data Analysis - Exercise 2 (SVD and PCA)

Q.1

Objective

In this question, you will implement the power method to estimate the dominant right singular vector of a matrix $A \in \mathbb{R}^{n \times m}$, where $n > m$. You will explore how the choice of initialization affects convergence and visualize the error across iterations. Your implementation will be compared against the true singular vectors obtained from the SVD.

(A) Generating a Matrix with a Singular Value Gap

To test the power method, you will construct a matrix with a large singular value gap.

```
import numpy as np

np.random.seed(0)
n, m = 100, 50  # A will be n x m

# Generate U and V from a random matrix SVD
A_rand = np.random.randn(n, m)
U, _, Vt = np.linalg.svd(A_rand, full_matrices=False)

# Define singular values with a gap
s = np.linspace(10, 1, m)
s[1:] *= 0.1  # Compress all except the top singular value
S = np.diag(s)

# Construct matrix A
A = U @ S @ Vt
```

Note: We construct $A \in \mathbb{R}^{n \times m}$ using its singular value decomposition:

$$A = USV^T$$

Here, U and V are obtained from the SVD of a random matrix, and we manually define S to have a large gap between the first and remaining singular values. **Try to explain why?**

Plot the singular values of A.

(B) Implementing the Power Method

Implement the power method on $A^TA \in \mathbb{R}^{m\times m}$ to estimate the top right singular vector.

1. Write a function `power_method(A, v0, num_iter)` which:
   - Starts with an initial vector $v_0 \in \mathbb{R}^m$
   - Iteratively computes $v_{k+1} = A^TAv_k$, and normalize the result
   - Returns the iterates v_k at each step
2. Briefly explain our expectations for this algorithm's outcomes.

(C) Initializations

1. Generate two vectors that will be used for initialization:
   - A random unit vector $v_{\text{rand}} \sim \mathcal{N}(0, I)$
   - A vector v_{orth} that is almost orthogonal to the main singular vector. Think about how to do this effectively.
2. Plot them on the same figure.

(D) Measuring Convergence

1. For each iterate v_k, compute the alignment with the true singular vector v_1 using the cosine distance:
$$\text{error}_k = 1 - |\langle v_k, v_1\rangle|$$
2. Plot this error as a function of iteration k, for both initializations.
3. Observe how convergence differs between the random and orthogonal starts. Discuss the difference between them.
4. At each iteration k, compute the Rayleigh quotient:
$$\rho_k = \|Av_k\|$$
5. Plot ρ_k as a function of k, and compare it to the top singular value σ_1. Explain your results.

Q.2

Objective

In this question, you will use Principal Component Analysis (PCA) to explore the structure of facial image data. You will visualize the principal components (also known as **eigenfaces**), reconstruct images using different numbers of PCs, and interpret what information the leading components capture.

Dataset: Olivetti Faces

We use the Olivetti Faces dataset from the `sklearn.datasets` module. It consists of 400 grayscale face images of size 64×64, each flattened into a 1D vector of length 4096. There are 40 unique individuals, each with 10 different images.

```
from sklearn.datasets import fetch_olivetti_faces
faces = fetch_olivetti_faces()
X = faces.data  # shape (400, 4096)
images = faces.images  # shape (400, 64, 64)
```

(A) PCA Decomposition

1. Standardize the data (center it by subtracting the mean image).
2. Apply PCA and compute the first $K = 100$ components.
3. Plot the cumulative explained variance ratio as a function of K.

(B) Visualizing Eigenfaces

1. Visualize the first 10 principal components as 64×64 grayscale images, as subplots.
2. These are called **eigenfaces**, representing dominant patterns of variation across faces. Discuss: What types of features appear in the first few components? (e.g., lighting, face orientation)

(C) Face Reconstruction

1. Choose a few test images from the dataset (2-3).
2. Reconstruct them using different numbers of components (e.g., 5, 10, 25, 50, 100).
3. For each reconstruction, compute and plot the reconstruction error.
4. Visualize the original vs. reconstructed images side-by-side (again, use subplots for that).