DSC 210 Numerical Linear Algebra, Fall 2025

Homework Problems for Topic 3: *Eigenvalue Problems*

Student Name (PID):

Write your solutions to the following problems by typing them in LaTeX. Unless otherwise noted by the problem's instructions, show your work and provide justification for your answer. Homework is due via Gradescope at **25th November 2025, 11:59 PM**.
Late Policy: If you submit your homework after the deadline we will apply a late penalty of 10% per day.

Guidelines for Homework Related Questions:

(a) As a general rule, we can help you understand the homework problems and explain the material from the corresponding lectures, but we cannot give you the entire solution.

(b) Regarding debugging programming questions: We ask you to do some debugging on your own first, including printing out intermediate values in your algorithms, trying a simpler version of the problem, etc.

(c) We will not be pre-grading the homework, i.e. we won't confirm if the answer you have is correct.

AI Usage Policy:

(a) Code: You may use LLMs to debug your code; however, you may not use LLMs to generate your entire code, and code must be reviewed and tested.

(b) Writing: You may use LLMs to correct grammar, style and latex issues; however, you may not use LLMs to generate entire solutions, sentences or paragraphs. All writing must be in your own voice.

Academic Integrity Policy:

The UC San Diego Academic Integrity Policy (formerly the Policy on Integrity of Scholarship) is effective as of September 25, 2023 and applies to any cases originating on or after September 25, 2023. The university expects both faculty and students to honor the policy. For students, this means that all academic work will be done by the individual to whom it's assigned, without unauthorized aid of any kind. If violations of academic integrity occur, the same Sanctioning Guidelines apply regardless of which policy was effective for that case.

For more information on how the policy is implemented, refer to the most current procedures. Remember: When in doubt about what constitutes appropriate collaboration or resource use, please ask TAs before proceeding. It's always better to clarify expectations than to risk an academic integrity violation. Academic integrity violations can have serious consequences for your academic record, and you will get zero grades.

You can access the Homework Template using the following link: `https://www.overleaf.com/read/vfhcmsppvskp`

Question 1: Singular value decomposition (25 points)
Determine the SVD of the following matrices by hand calculation. Make sure to explicitly list Σ, U, and V in your answer.

(a) $\begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$ (4 points)

(b) $\begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}$ (4 points)

(c) $\begin{bmatrix} 0 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$ (4 points)

(d) $\begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$ (4 points)

(e) $\begin{bmatrix} 2 & 2 \\ 3 & 2 \end{bmatrix}$ (4 points)

(f) Write Python code to compute SVD of the matrices in parts (a) to (e) and verify that the above results match what you got from code. (5 points)
Hint: You can use the numpy library function.

Solution:

```python
import numpy as np

matrix_a = [[5, 0], [0, 2]]
matrix_b = [[2, 0], [0, 5]]
matrix_c = [[0, 3], [0, 0], [0, 0]]
matrix_d = [[1, 0], [1, 0]]
matrix_e = [[2, 2], [3, 2]]

svd_a = np.linalg.svd(matrix_a)
print(f'svd_a: {svd_a}')

svd_b = np.linalg.svd(matrix_b)
print(f'svd_b: {svd_b}')

svd_c = np.linalg.svd(matrix_c)
print(f'svd_c: {svd_c}')

svd_d = np.linalg.svd(matrix_d)
print(f'svd_d: {svd_d}')

svd_e = np.linalg.svd(matrix_e)
print(f'svd_e: {svd_e}')
```

```
svd_a: SVDResult(U=array([[1., 0.],
       [0., 1.]]), S=array([5., 2.]), Vh=array([[1., 0.],
       [0., 1.]]))

svd_b: SVDResult(U=array([[0., 1.],
       [1., 0.]]), S=array([5., 2.]), Vh=array([[0., 1.],
       [1., 0.]]))

svd_c: SVDResult(U=array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]]), S=array([3., 0.]), Vh=array([[ 0.,  1.],
```

```
12          [-1.,  0.]]))
13
14 svd_d: SVDResult(U=array([[-0.70710678, -0.70710678],
15          [-0.70710678,  0.70710678]]), S=array([1.41421356, 0.        ]), Vh=array
       ([[-1., -0.],
16          [ 0.,  1.]]))
17
18 svd_e: SVDResult(U=array([[-0.61541221, -0.78820544],
19          [-0.78820544,  0.61541221]]), S=array([4.56155281, 0.43844719]), Vh=array
       ([[-0.78820544, -0.61541221],
20          [ 0.61541221, -0.78820544]]))
```

Question 2: Eigenvalues and Eigenvectors (25 points)

$$\mathbf{A} = \begin{bmatrix} -5 & 3 \\ -6 & 6 \end{bmatrix}$$

(a) Write down the characteristic equation for matrix $\mathbf{A}$. Use the characteristic equation to solve the eigenvalues and normalized eigenvectors of matrix $\mathbf{A}$. (10 points)

(b) Write python code to verify your answers for the eigenvalues and normalized eigenvectors from Part (a). **Hint**: You can use numpy to solve this problem. (5 points)

(c) Prove that if a real matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ has unique eigenvalues, then the eigenvectors $\mathbf{x}_i$ are linearly independent. **Hint:** Prove by contradiction, start with $n = 2$ case. (10 points)

Solution:

Question 3: Power method (20 points)

Solve the problem in the google colab. Please make a copy of the notebook and solve it there.

(a) Power Method: Write function `power_method(A,x)`, which takes as input matrix A and a vector $\mathbf{x}$, and uses the power method to calculate eigenvalues and eigenvectors.
Get the largest **(in absolute value)** eigenvalue and the corresponding eigenvector for matrix A using the above function.

$$A = \begin{bmatrix} 2 & 2 & 1 \\ 1 & 3 & 2 \\ 2 & 4 & 1 \end{bmatrix}$$

Start with initial eigenvector guesses: $[-1, 0.5, 3]$ and $[2, -6, 0.2]$. For each of the vectors, iterate until convergence.

i. Plot how the eigenvalue changes with respect to iterations.
ii. Report the number of steps it took to converge, for both the eigenvalue and eigenvector.
iii. Report the final eigenvalue and eigenvector. Match your output with the results generated by the numpy API: `numpy.linalg.eig`.

Note: You only need to look at magnitudes of eigenvalues. Use an absolute tolerance of 10^{-6} between eigenvalue output of previous and current iteration as stopping criteria. You may also need to normalize the final eigenvector to match with output of numpy API `numpy.linalg.eig`. (10 points)

(b) Inverse Power Method:
Write a function `inverse_power_method(A,x)`, which takes as input matrix $\mathbf{A}$ and a vector $\mathbf{x}$, and uses inverse power method to calculate the smallest (in absolute value) eigenvalue and corresponding eigenvector. Solve for the smallest (in absolute value) eigenvalue and corresponding eigenvector for the matrix from (a). Use the same initial eigenvector guesses as (a).

i. Plot the computed/estimated eigenvalue with respect to iterations.
ii. Report how many iterations do you need for it to converge to the smallest eigenvalue.
iii. Report the final eigenvalue and eigenvector you get. Match your answer with the results generated by the numpy API `numpy.linalg.eig`.

Note: You only need to look at magnitudes of eigenvalues. Use an absolute tolerance of 10^{-6} between eigenvalue output of previous and current iteration as stopping criteria. You may also need to normalize the final eigenvector to match with output of numpy API `numpy.linalg.eig`. (10 points)

Solution:

Question 4: Face Recognition with Eigenfaces (30 points + 10 Bonus Points)

Solve the problem in the google colab. Please make a copy of the notebook and solve it there.

Goal: Perform face recognition on the *Labeled Faces in the Wild* dataset using PyTorch.

Dataset Information: Labeled Faces in the Wild dataset consists of face photographs designed for studying the problem of unconstrained face recognition. The original dataset contains more than 13,000 images of faces collected from the web.

Tasks:

- First, perform Principal Component Analysis (PCA) on the image dataset.
- Using PCA, extract the Top k principal components (eigenvalues).
- Reconstruction of faces from these **eigenvalues** will give us the **eigen-faces** which are the most representative features of most of the images in the dataset.
- **BONUS**: Finally, train a simple PyTorch Neural Network model on the modified image dataset. This trained model will be used for prediction and evaluation on a test set.

Problem Description

(a) Preprocessing: Using the `train_test_split` API from `sklearn`, split the data into train and test dataset in the ratio 3:1. Use `random_state=42`.

For better performance, normalize the features which can have different ranges with huge values. (As all our features here are in the range [0, 255], it is not explicitly needed here. However, it is a good exercise.)

Use the `StandardScaler` class from `sklearn` and use that to normalize `X_train` and `X_test`. Validate and show your result by printing the first 5 features of 5 images of `X_train` (This result can vary from PC to PC). (10 points)

(b) Dimensionality reduction: Use the PCA API from `sklearn` to extract the top 100 principal components of the image matrix and fit it on the training dataset.

Visualize some of the top few components as an image (eigenfaces). (10 points)

(c) Face reconstruction: Reconstruct an image from its point projected on the principal component basis. Project the first three faces on the eigenvector basis using PCA models trained with varying number of principal components. Using the projected points, reconstruct the faces, and visualize the images.

Your final output should be a (3×5) image matrix, where the rows are the data points, and the columns correspond to original image and reconstructed image for n_components = $[10, 100, 150, 500]$. (10 points)

(d) Prediction **(BONUS)**: Train a neural network classifier in **PyTorch** on the transformed dataset. Note: For PyTorch reference see documentation (10 points)

Solution: