The University of Alabama
College of Engineering · Computer Science

Assignment 3

Submission Instructions

- Submit your solutions as a **separate PDF file**.
- Clearly label each problem number and part (e.g., Problem 1a, Problem 2c).
- For Python problems, include screenshots of your code and output.
- Ensure your work is legible and well-organized. Show all steps for full credit.

Problem 1: Eigenvalue Properties (15 Points)

Let $A = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$.

(a) Compute $\text{tr}(A)$ and $\det(A)$.

(b) Using the trace-determinant relationships, find the eigenvalues of A **without** expanding the full characteristic polynomial.

Hint: For a 2×2 *matrix,* $\lambda_1 + \lambda_2 = \text{tr}(A)$ *and* $\lambda_1 \cdot \lambda_2 = \det(A)$.

(c) Verify one of your eigenvalues by showing that $\det(A - \lambda I) = 0$.

Solution:

■

Problem 2: Eigenvectors, Geometry, and Diagonalization **(25 Points)**

Consider the matrix $A = \begin{pmatrix} 5 & 4 \\ 1 & 2 \end{pmatrix}$.

(a) Find the eigenvalues and eigenvectors of A.

(b) Describe geometrically what A does to a vector in the direction of each eigenvector.

(c) What happens when A is applied to the vector $\mathbf{v} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$? Is $\mathbf{v}$ an eigenvector? Why or why not?

(d) Form the matrix C (columns are eigenvectors) and diagonal matrix D such that $A = CDC^{-1}$.

(e) Explain in words why A^{100} is easy to compute using diagonalization.

Solution:

■

Problem 3: Building Symmetric Matrices **(10 Points)**

Let $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$ be a 3×2 matrix.

(a) Compute $A^T A$ and AA^T. What are the dimensions of each?

(b) Verify that both $A^T A$ and AA^T are symmetric matrices.

(c) Explain why $A^T A$ is always symmetric for **any** matrix A (not just this example).
Hint: What is $(A^T A)^T$ *?*

Solution:

■

Problem 4: Understanding SVD (20 Points)

The Singular Value Decomposition states that any $m \times n$ matrix A can be written as $A = U\Sigma V^T$.

(a) Explain why eigenvalue decomposition cannot be applied directly to a 3×5 matrix, but SVD can.

(b) Given a matrix A with singular values $\sigma_1 = 10$, $\sigma_2 = 3$, $\sigma_3 = 0.01$, $\sigma_4 = 0$:

- What is the rank of A?
- Which singular value contributes most to the matrix? Which contributes least (besides zero)?

(c) In the decomposition $A = U\Sigma V^T$, what are the dimensions of U, Σ, and V^T if A is 100×50?

(d) The relation $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$ is fundamental. Explain in plain English what this equation says about how A transforms the right singular vector $\mathbf{v}_i$.

Solution:

■

Problem 5: Low-Rank Approximation **(15 Points)**

Consider using SVD for image compression.

(a) A grayscale image is represented as a 1000×800 matrix. How many numbers are needed to store the original image?

(b) If we use a rank-k SVD approximation, we store: U (first k columns), Σ (first k singular values), and V^T (first k rows). How many numbers are stored for a rank-50 approximation?

(c) What is the compression ratio (original size / compressed size) for the rank-50 approximation?

(d) Explain why keeping the **largest** singular values (and discarding the smallest) gives the best approximation in terms of the Frobenius norm.

Solution:

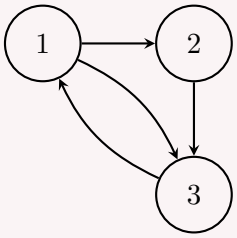
■

Problem 6: PageRank and the Dominant Eigenvector (15 Points)

Google's PageRank algorithm finds the importance of web pages by computing a specific eigenvector.

(a) In PageRank, we solve $A\mathbf{r} = \mathbf{r}$. What eigenvalue does this correspond to?

(b) **(Coding)** Consider the following web graph where arrows indicate links:



- Build the column-stochastic transition matrix A from this graph.
- Implement power iteration in Python: start with $\mathbf{r}_0 = (1/3, 1/3, 1/3)^T$, iterate 20 times.
- Compare your result with `np.linalg.eig`.

(c) Why is power iteration more practical than computing eigenvectors directly (using Gaussian elimination) for a matrix with billions of rows?

Solution:

■

Problem 7: Python: SVD Image Compression **(Bonus - 10 Points)**

Write Python code to:

(a) Create or load a grayscale image matrix (you can use a simple test matrix or load an actual image).

(b) Compute the SVD using `np.linalg.svd`.

(c) Reconstruct the image using only the top k singular values for $k = 5, 20, 50$.

(d) Print the relative reconstruction error: $\frac{\|A - A_k\|_F}{\|A\|_F}$ for each k.

Paste your Python code and output below.
Hint: Use `np.linalg.norm(A, 'fro')` to compute the Frobenius norm.

Solution:

```python
import numpy as np

# Your code here
```

■