Work Integrated Learning Programmes Division
M. Tech. in AI & ML

Assignment 1

AIML ZC416 - Mathematical Foundations for Machine Learning

Instructions

1. By random entries, I mean a system generated random number. No marks would be awarded for deterministic entries.
2. All entries should have 8 decimals.
3. This is not a group activity. Each student should do the problems and submit individually.
4. The code snippets and outputs can be copied and pasted to a file and upload as single pdf file with name BITSID.pdf
5. Submissions beyond 11th of Sept, 2025 23.59 hrs would not be graded
6. Assignments sent via email would not be accepted
7. Copying is strictly prohibited. Adoption of unfair means would lead to disciplinary action.

Answer all the questions

Q1) Consider a dataset $X \in \mathbb{R}^{50\times 6}$ constructed as follows: the first four features f_1, f_2, f_3, f_4 are generated as random features sampled from a standard normal distribution (read about this). The fifth and sixth features are defined by the relations

$$f_5 = 2f_1 + f_2, \qquad f_6 = f_3 - f_4.$$

Perform the following tasks in sequence:

1. Write a python code to generate the dataset

   $$X = [f_1, f_2, f_3, f_4, f_5, f_6].$$

   Comment on round off errors if any. (0.5 mark)

2. Write a python code which computes the rank of X and display the output for the dataset X generated in step 1. (0.5 mark)

3. **Numerical Experiment with the Power Method**

   Read about the power method for finding the dominanat eigenvalue and its corresponding eigenvector and perfrom the follwoing tasks.

(a) Write a python code to compute the covariance matrix

$$C = \frac{1}{n} X^T X.$$

where n is number of data points, for our data set. (0.5 mark)

(b) Implement the power method in python to approximate the largest eigenvalue λ_1 and its corresponding eigenvector v_1 of C. Show the code and also the outputs. (0.5 mark)

(c) Write a python code to obtain the next largest eigenvalue λ_2 and its corresponding eigenvector v_2 by applying power method on $C - v_1 v_1^T C$. Having found out $v_1, v_2, \ldots, v_{k-1}$, one can find λ_k and its corresponding eigenvector v_k by applying power method on $C - \sum_{j=1}^{k-1} v_j v_J^T C$. Give the code and also display the obtained eigenvalues and the corresponding eigen vectors. (1 mark)

(d) Find all the eigenvalues and eigenvector using the builtin python function and compare with the obtained result in (c). (0.5 mark)

(e) Compare the number of iterations required to get an accuracy of 10^{-8} using the power method. The actual values can be taken as the one obtained in (d). (0.5 mark)

Q2) Consider an image represented as a matrix:

$$A \in \mathbb{R}^{150 \times 100} \quad \text{(grayscale image)}$$

We aim to use the Singular Value Decomposition (SVD) for image compression and analysis. The task is divided into several parts for clarity.

(a) Perform SVD

Compute the Singular Value Decomposition of A:

$$A = U \Sigma V^T$$

where U and V are orthogonal matrices, and Σ is a diagonal matrix containing singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$, where r is rank of A. (0.5 mark)

(b) Reconstruction with Multiple Ranks

Construct rank-k approximations of A for $k \in \{5, 10, 20, 40, 60\}$:

$$A_k = U_k \Sigma_k V_k^T$$

where U_k, Σ_k, V_k correspond to the top-k singular values and their corresponding singular vectors. Display the original image and each approximation side by side. (1 mark)

(c) Error Analysis

For each k, compute the Frobenius norm of reconstruction error:

$$E_k = \|A - A_k\|_F$$

Plot E_k as a function of k. (1 mark)

(d) Energy Preservation

Compute the proportion of variance (energy) preserved by the top-k singular values:

$$\text{Energy}(k) = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{r} \sigma_i^2}$$

where $r = \text{rank}(A)$. Plot Energy(k) vs. k. (0.5 mark)

Q3) Consider the matrix:

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

This type of symmetric tridiagonal matrix appears in many machine learning applications.

Tasks(Implement in Python)

(a) **Eigenvalue Computation.** Compute the characteristic polynomial of M and determine its eigenvalues. Verify your results using Python (`numpy.linalg.eig`). (0.5 mark)

(b) **Eigenvectors.** Compute the eigenvectors of M corresponding to each eigenvalue. Normalize them and check numerically in Python that $Mv = \lambda v$. (0.5 mark)

(c) **Diagonalizability.** Determine whether M is diagonalizable with justification. If diagonalizable, write $M = PDP^{-1}$, where D is diagonal. (1 mark)

(d) **Generalization** Compute the Eigen values for the general case of M being a $n \times n$ tridiagonal matrix with 2 replaced by a and 1 replaced by b in the equation (1). (1 mark)