DA5400 Foundations of Machine Learning - Assignment 3

PCA and K-means Clustering

Roll Number: DA24C026
Name: Rishabh Garg

Problem Statement

1. Implement PCA from scratch to analyse the dimensionality of image data, visualise principal components, assess explained variance, and reconstruct data with different numbers of principal components.
2. Implement K-means clustering with Lloyd's algorithm, observe the effect of initializations, visualise clusters with Voronoi regions, and explore alternative clustering methods for non-linear clusters.

Part I - Principal Component Analysis (PCA) on MNIST Dataset

Principal Component Analysis (PCA) is a dimensionality reduction technique that aims to reduce the number of variables in a dataset while retaining as much variance as possible. This technique is useful for high-dimensional datasets like MNIST, where each image has 784 pixels.

Data preperation:

- Dataset: MNIST handwritten digit dataset. 784 dimensional data points (when converted to row vectors).
- Sampling: I randomly sampled 1000 images from the dataset, with 100 samples per digit (0–9) to maintain a balanced representation.

PCA: Implementation:

1. Centering the Data:

    - For PCA, the data must be centered. This involves subtracting the mean of each feature to ensure the data is centered around zero.

    - Formula for mean centering is given as:

$$\text{Centered data } X' = X - \text{mean}(X)$$

2. Covariance Matrix:

    - The covariance matrix, Σ, measures the variance and relationships between features. The diagonal values represent the variance of individual features, while off-diagonal values show the covariance between pairs of features.

      Covariance Matrix is calculated as:

$$\Sigma = \frac{1}{n} X'^T X'$$, where X' is centered X.

3. Eigen Decomposition:

    - I performed eigen decomposition on the covariance matrix to obtain eigenvalues and eigenvectors. Eigenvectors represent the directions of maximum variance (principal components), while eigenvalues represent the amount of variance along each component.

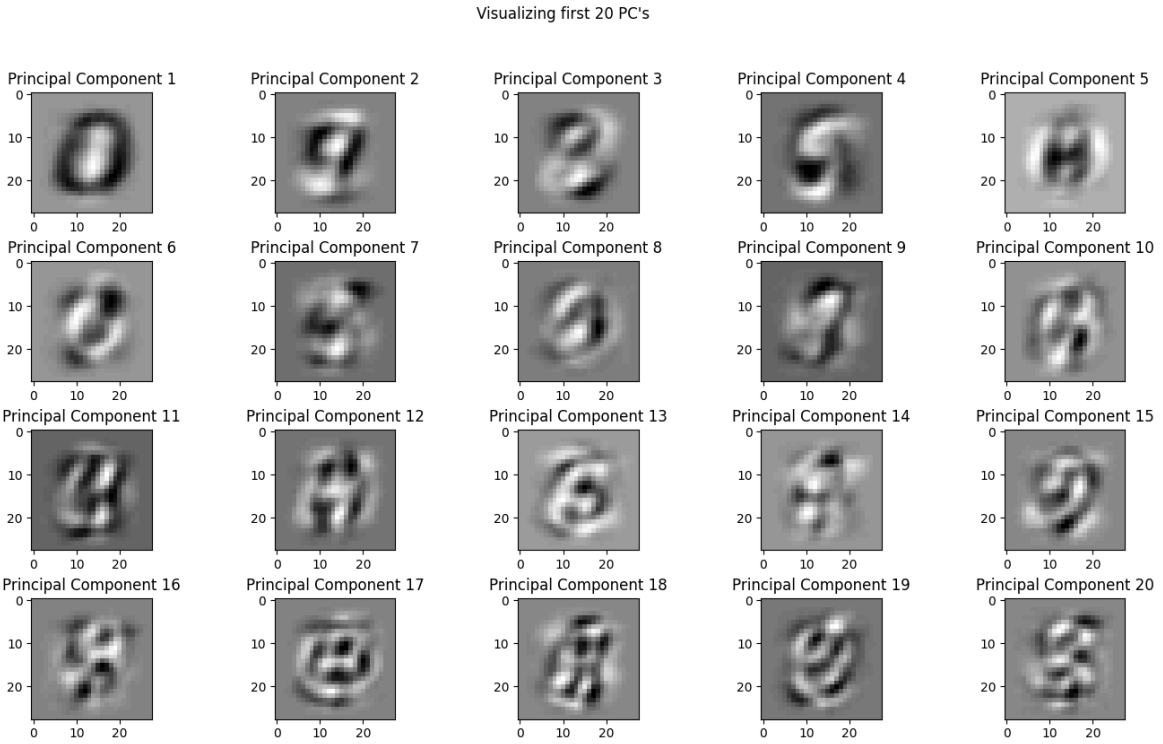
      $\Sigma v = \lambda v$, where 'v' is an eigenvector, 'λ' is its corresponding eigenvalue.

4. Principal Components:

    - I sorted eigenvectors by their eigenvalues in descending order, with the largest eigenvalues representing the most significant components. The top k eigenvectors form the principal components for reducing the dataset dimensionality to k dimensions.

Task (i): Visualization of Principal Components

The first 20 principal components (PCs) were visualized. Each PC reveals distinct patterns in pixel intensity that capture significant visual characteristics of the digits. The patterns obtained highlight edges, strokes, and shapes important for distinguishing between different digits.



Explained Variance by top 20 principal components:

- Explained variance indicates the proportion of the total dataset variance captured by each principal component.
- Explained variance of a component *i* is given by:

$$\mathrm{EV}_i = \frac{\lambda_i}{\sum_{j=1}^{d} \lambda_j}$$, where λ values are the sorted eigenvalues.

Observation: - The first few PCs captured the most variance, with diminishing returns as more components are added. The cumulative explained variance for the top 20 PCs was approximately **65.38%**

Result of variance explained by each of the first 20 PCs.

- Explained variance of Principal Component 1: 0.098861
- Explained variance of Principal Component 2: 0.073117
- Explained variance of Principal Component 3: 0.061343
- Explained variance of Principal Component 4: 0.054319
- Explained variance of Principal Component 5: 0.050405
- Explained variance of Principal Component 6: 0.044210
- Explained variance of Principal Component 7: 0.031867
- Explained variance of Principal Component 8: 0.029912
- Explained variance of Principal Component 9: 0.027919
- Explained variance of Principal Component 10: 0.025332
- Explained variance of Principal Component 11: 0.022116
- Explained variance of Principal Component 12: 0.018992
- Explained variance of Principal Component 13: 0.018194
- Explained variance of Principal Component 14: 0.017309
- Explained variance of Principal Component 15: 0.015836
- Explained variance of Principal Component 16: 0.014121
- Explained variance of Principal Component 17: 0.013503
- Explained variance of Principal Component 18: 0.012656
- Explained variance of Principal Component 19: 0.012065
- Explained variance of Principal Component 20: 0.011715

Task (ii): Reconstruction and Dimensionality Selection

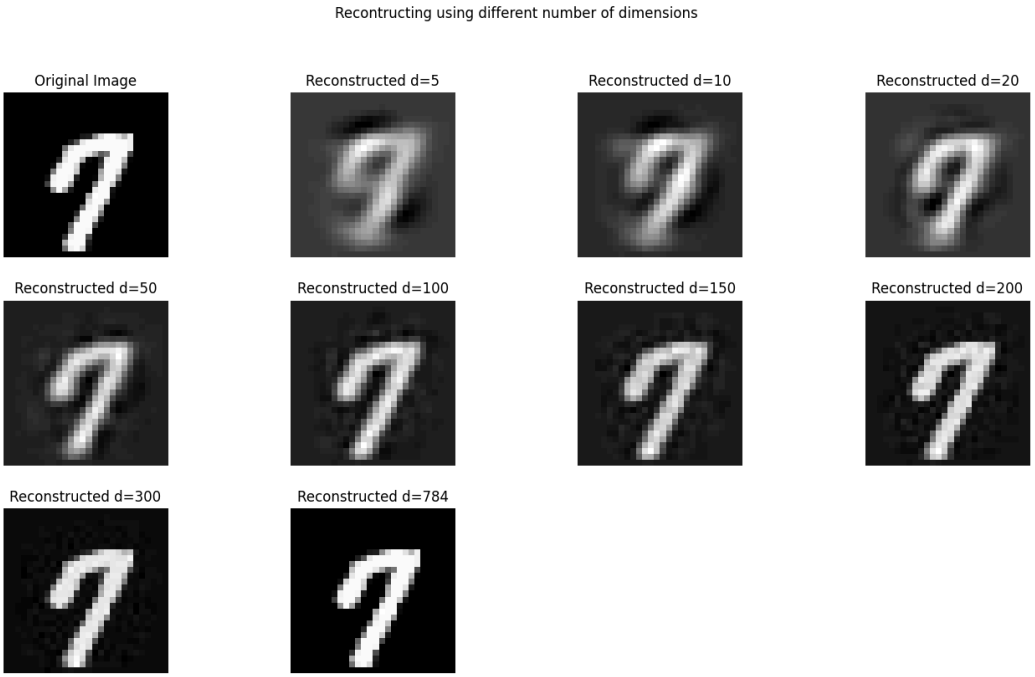
1. Reconstruction:

    - I reconstructed images by projecting them onto the top *d* principal components and then transforming them back to the original 784-dimensional space.

      $$X_{\text{reconstruct}} = (X_{\text{proj}} V^T) + \text{mean},$$

      where V is the matrix of top *d* eigenvectors.

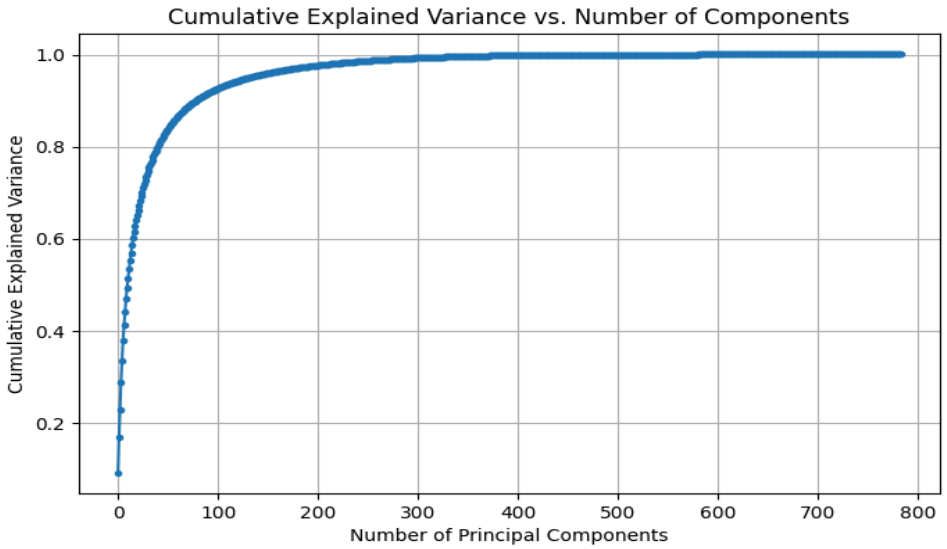
Results of Reconstruction using different number dimensions:



2. Quality of Reconstruction:

- As the number of dimensions *d* increased, the reconstructed images became clearer. The reconstructed images at d = 150 were visually close to the originals, indicating that this dimensionality captures more than 95% of variance and is sufficient for downstreaming this task.

The cumulative variance plot helped to select *d* by showing the percentage of variance captured as *d* increases.



Based on the cumulative variance plot, using the top 150 components captured over 95% of the variance, making **150 dimensions** an optimal choice for retaining sufficient amounts of critical information.

Part II. K-means Clustering on 2D Dataset

K-means clustering aims to partition data into k clusters by minimizing the distance between data points and their corresponding cluster centers. Here, I applied Lloyd's algorithm to a 2D dataset to explore clustering performance with different initializations and cluster counts.

K-means Clustering: Implementation

1. Lloyd's Algorithm:
    - Initialization: Randomly initialize *k* cluster centers in the data space.
    - Assignment Step: Each data point is assigned to the nearest cluster center.
    - Update Step: The mean of the data points in each cluster is calculated and used to update the cluster center.
    - Convergence: The algorithm iterates until the cluster centers stabilize (no significant changes between iterations).
2. Error Function:
    - The objective is to minimize the within-cluster sum of squares (WCSS), calculated as the sum of squared Euclidean distances between data points and their assigned cluster centers.

$$\text{WCSS} = \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2,$$

where C_i is the set of points in cluster *i* with center μ_i.

Task (i): Lloyd's Algorithm with k=2 and Multiple Initializations

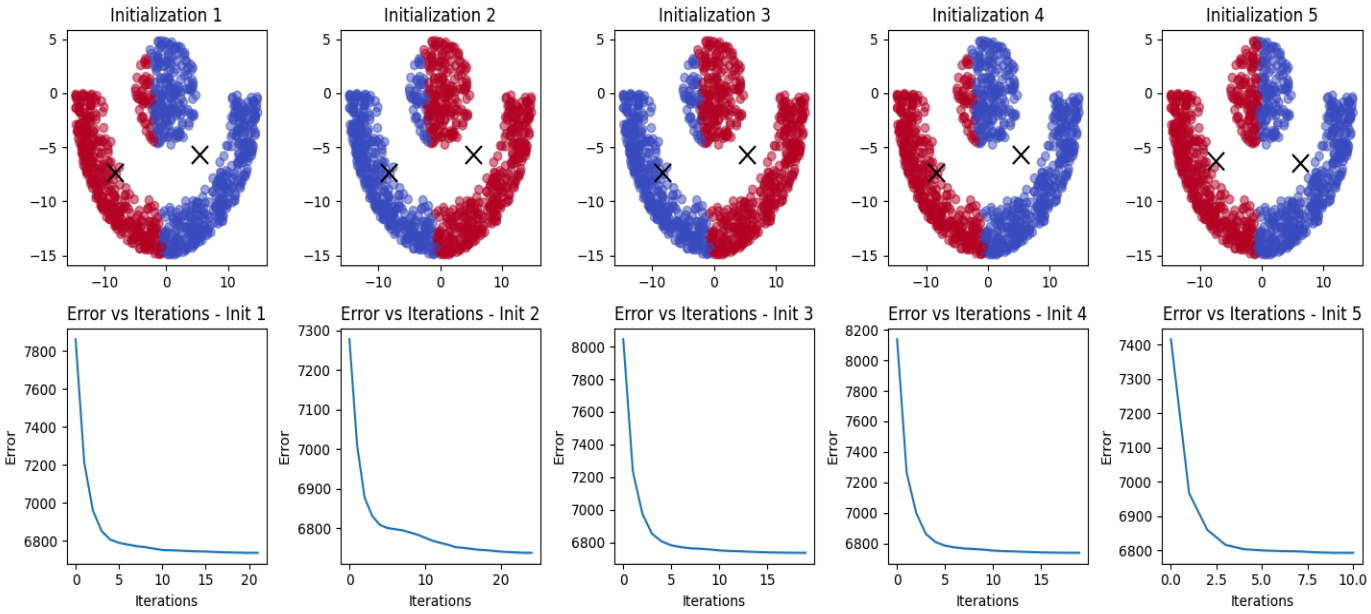
The K-means algorithm was implemented from scratch using Lloyd's algorithm. I set *k*=2 and repeated the clustering five times with different random initializations. To determine when the algorithm has converged, I used **np.allclose** method to check if

the current cluster centers are sufficiently close to the centers from the previous iteration. If the change in centers is within a small tolerance, indicating minimal movement, the loop terminates early, signaling that the algorithm has reached a stable state and further iterations are unlikely to significantly change the clustering results.

Results obtained from different initializations:

Each initialization converged to slightly different clusters (for maximum iterations set to 200), demonstrating K-means' sensitivity to initialization.

For each initialization, the error function (WCSS) decreased, indicating convergence to local optima.



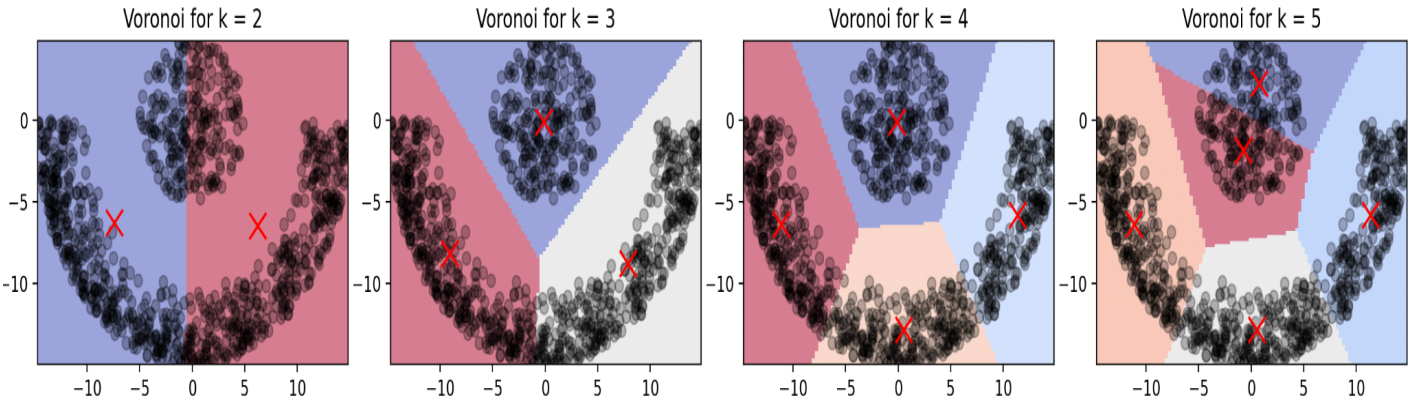
As it can be inferred from the error plots, different initializations follow different convergence patterns, but they all converge within 25 iterations.

Task (ii): Voronoi Regions with Fixed Initializations for Different *k* Values

For each *k* in (2, 3, 4, and 5), I fixed the initial cluster centers and visualized the Voronoi regions, which represent the areas closest to each cluster center.

Observation: As *k* increased, the Voronoi regions became more refined, allowing for a more detailed partitioning of the data space.

Results for Voronoi regions obtained for different k values:



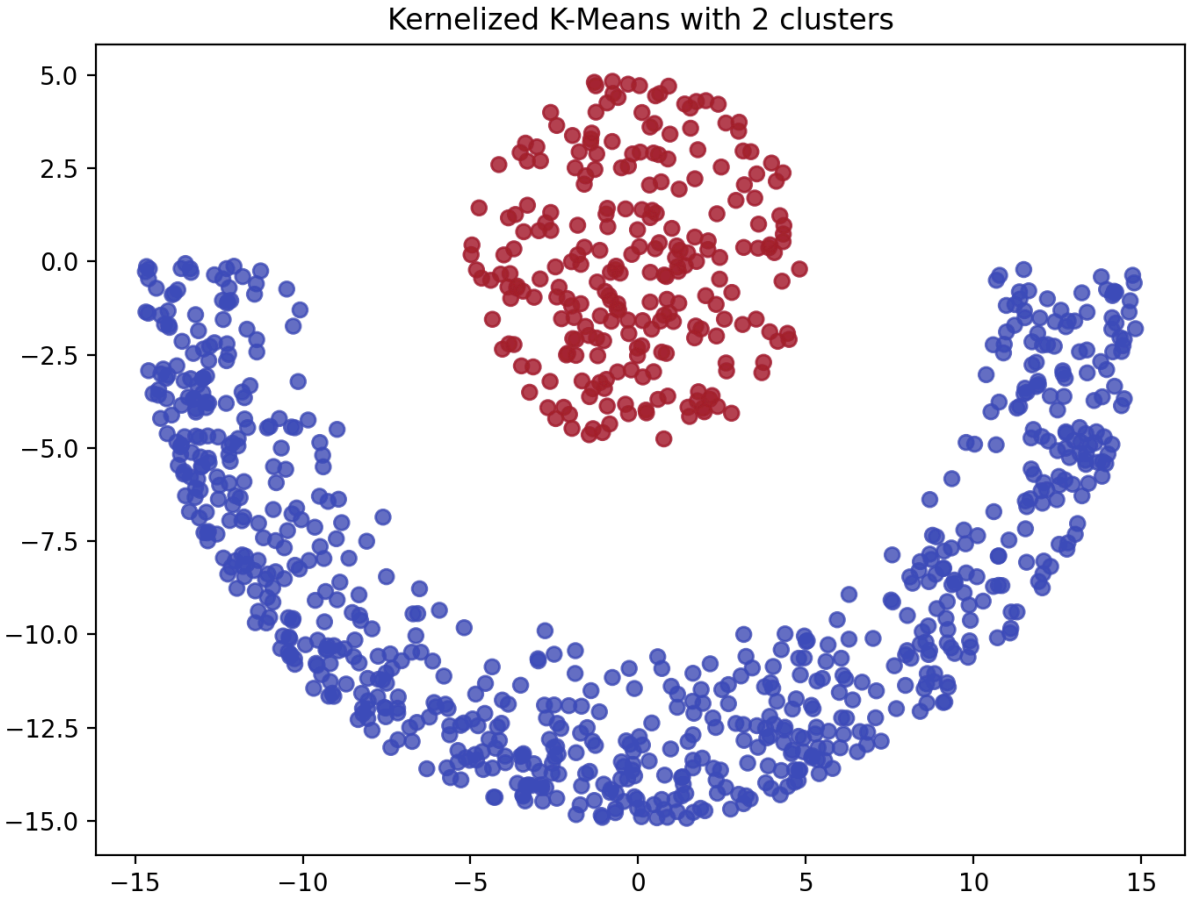
Task (iii): Analysis of dataset and Recommendations for Clustering

1. Challenges with Lloyd's Algorithm:
    - Standard K-means, using Euclidean distance, struggled with non-linear cluster shapes, as seen in the Voronoi regions. Lloyd's algorithm is limited to detecting convex clusters, which may **not represent** all patterns in **this dataset**.

2. **Kernelized K-means** for Non-linear Clusters:
    - Kernelized K-means: To address non-linear clusters, I implemented K-means with a Gaussian (RBF) kernel. This method uses a similarity matrix instead of Euclidean distances, allowing detection of complex clusters.

Observation: For non-linear data distributions, kernelized K-means or spectral clustering is more effective than standard K-means, as these methods can capture complex shapes. Using kernels, we transform non-linear data from lower dimension to a higher dimension. In higher dimensions, we apply standard k - means on the transformed dataset.

Results of the kernelized approach for (k=2):



Summary

PCA effectively reduced dimensionality for MNIST, with 150 principal components capturing over 95% variance. In K-means clustering, kernelized approaches were recommended for non-linear clusters, as standard K-means was limited to convex clusters. These analyses demonstrate the trade-offs and considerations in applying PCA and clustering for diverse datasets.

End of report