Krithika Jain G#01462514

Report on Image Clustering – HW3

ID: fkrithik Rank: 10 Public Score: 0.80

Overview:

The given dataset consists of 10,740 handwritten images of digits (0-9) with 784 dimensions which corresponds to 28x28 pixels in two dimensions. The objective is to cluster similar images together. The following steps were performed to achieve this.

1. Data Analysis
2. Data Preprocessing
3. Dimensionality Reduction
4. K-Means Clustering
5. Evaluation

Data Analysis

The dataset consists of images in dimensions of 784, where each feature represents values ranging 0-255 corresponding to the original 28x28 pixel image. To gain insights into the images, I reconstructed it into grey scale to view different images achieved by normalization and flattening the pixels. It is observed that there are different styles of the same digit existing in the dataset. Moreover, it is also observed that numerous pixels in the images have a value 0. This indicates the importance of addressing high prevalence of zero values, as discussed further.

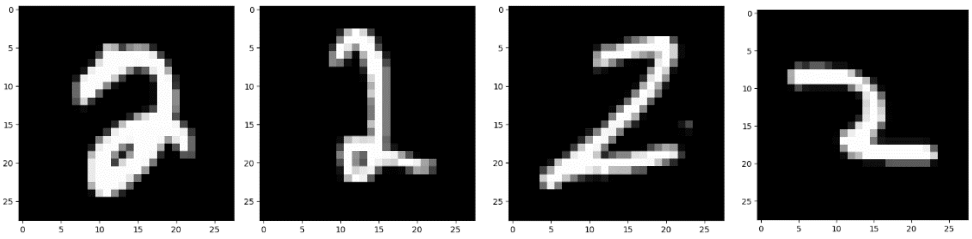

Figure 1: Different style of the same digits in the dataset

Data Preprocessing

After analysis it is clear that directly plotting pixel values in high dimensional space does not add any value for a clustering algorithm due to variability in pixel values arising from different styles of the same digit. To address this, an approach is adopted to enhance similarity among images of the same digit, with the goal of reducing intra-cluster distances for improved clustering.

To achieve similarity, specific features that contribute to the distinct styles are ignored. By doing so, the algorithm becomes less biased towards certain features and focuses on the overall construction of images. Averaging is employed as a technique to achieve this similarity. This involves averaging pixel values in an nxn grid matrix across the entire image, resulting in smoother images where fine-grained details are lost. Gaussian smoothing is also applied to further reduce noise and enhance generalization.

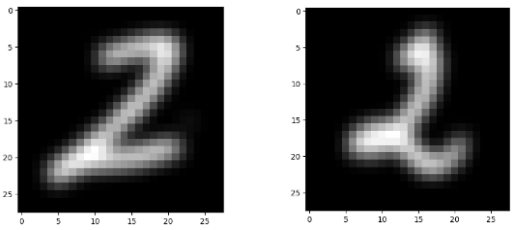

Figure 2: Blurred images

Dimensionality Reduction

Understanding the complexity and the given high dimensionality nature of the data as observed before, it is essential to select appropriate techniques and fine-tuning its parameters that optimise the clustering of images.

1. *PCA (Principal Component Analysis)*
   It is employed to address the redundancy in features, a dimensionality reduction technique that projects the dataset onto a hyperplane while keeping the maximum amount of variance. Instead of specifying a fixed number of components, the parameter *n_components is set to 0.99*, indicating a hyperplane that preserves 99% of the variance of the original dataset. This approach was chosen to avoid losing any significant details. After applying PCA, the dimension of the dataset was effectively reduced from 784 to 65.

2. *Learning Manifolds (t-SNE)*
   When dealing with complex datasets that are difficult to analyze using PCA, manifold learning is a more effective approach. In this case, t-SNE (t-Distributed Stochastic Neighbor Embedding) is used to reduce the dimensions to 2D.
   For t-SNE, the perplexity parameter, set to 18, determines the balance between local and global attention by considering k-neighbors. Additionally, the early_exaggeration parameter, set to 28, amplifies the distance between clusters, ensuring that distant clusters are well-separated and aiding in the clustering process. Typically, the perplexity parameter falls between 5-50 and is an essential factor in the t-SNE algorithm.

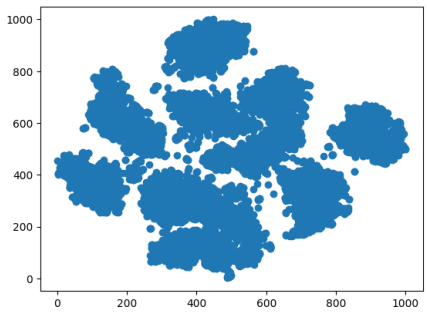

Figure 3:

This shows clusters on a scaled plot for k value = 10 after reducing dimensions.

K Means Clustering

The K-means Clustering algorithm implemented in the code consists of four essential functions. The pseudocode for the same is discussed below:

generateRandomKCentroids (data,k)

- *Selecting k random indices from data as centroids using numpy random generator*

assignClusters (data, centroids)

- *Computes numpy euclidean distances from each datapoint to all centroids*
- *Assigns each datapoint to the cluster of minimum distance from centroid*

updateCentroids (data, cluster_assignments, k)

- *Initializing updated_centroids np.array with zeros*
- *For each cluster in K:*
  - *Selects datapoints belonging to the cluster*
  - *If the cluster is not empty:*
    - *Updates the centroid to the mean of selected datapoints*
      Else
    - *Chooses a new random point as a new centeroid*

kMeans (data, numClusters, maxIterations=500, tolerance=1e-4)

- *generateRandomKCentroids ()*
- *Repeat for each in range(maxIterations)*
  - *assignClusters ()*
  - *updateCentroids ()*
  - *If centroids below tolerance (checks for convergence) break*

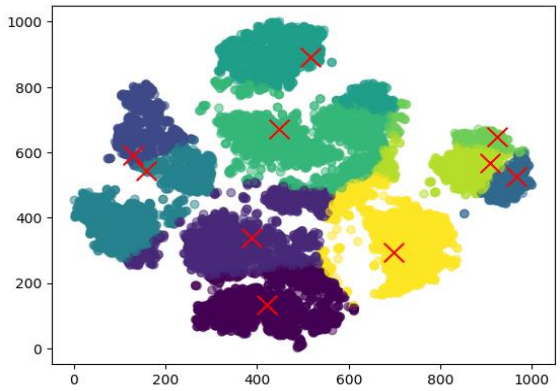

Figure 3: random centroids generated for k=10

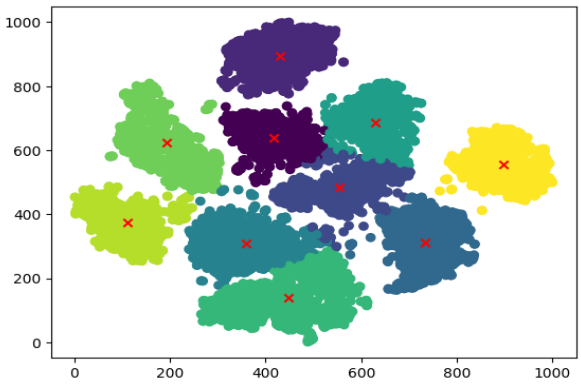

Figure 4: assigned clusters for k=10

Evaluation

To determine the optimal number of clusters for a given range (here k=2 to 20 with step 2), it is common to plot the values of the evaluation metrics. Two metrics are used, inertia (sum of squared distances) that measures the compactness of the clusters, which is indicated by smaller values. The "elbow" in the inertia plot, where the rate of decrease sharply changes, often indicates a suitable k value as seen in the figure 5, the values becomes almost consistent after k = 10. The silhouette score is another metric used for assessing the quality of clusters, it considers both the cohesion within clusters and separation between clusters. A higher silhouette score indicates better-defined clusters.

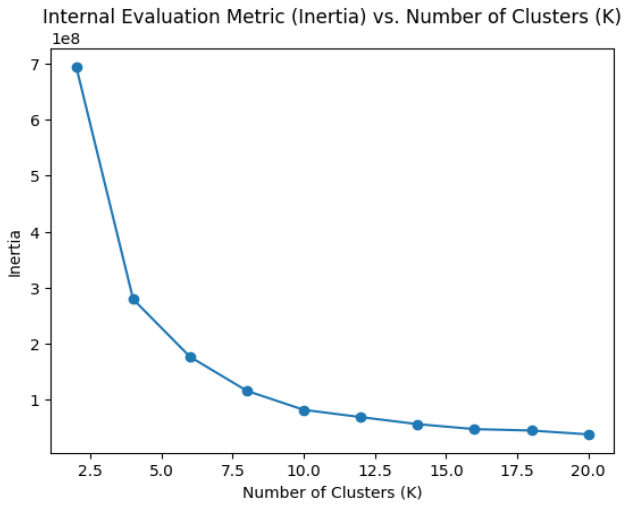

Figure 5: Inertia graph with rapid decrease as cluster increases, from k=10 the values are around the same range.

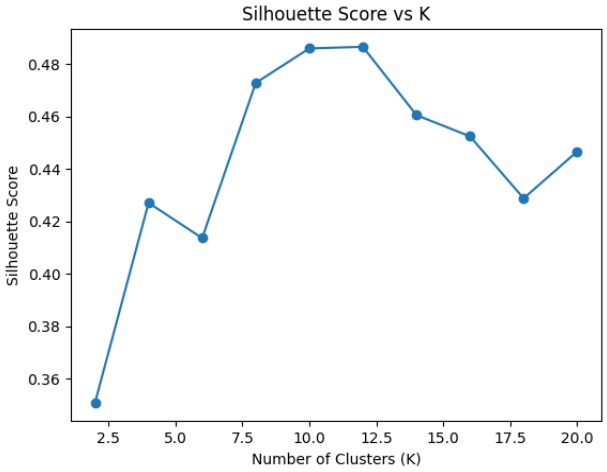

Figure 6: Silhouette scores for k values, higher value noticed around k =10