ML Assignment-2 Report

Harsh Vishwakarma

MTech, CSA

21532

K-MEANS Clustering:

K-means clustering is a machine learning algorithm used for clustering or grouping similar data points in a dataset. It is an unsupervised learning algorithm that works by partitioning a dataset into k clusters, where k is a predetermined number of clusters specified by the user.

Methodology:

- The **__init__** method initializes the number of clusters and the convergence threshold.
- The **euclid** method calculates the Euclidean distance between two points.
- The **classify** method assigns a point to a cluster based on the distance to the nearest mean.
- The **initialise_means** method randomly selects initial means from the data.
- The **recompute_means** method updates means based on the assigned points.
- The **fit** method fits the K-Means model to the data using the previously mentioned methods.
- The **predict** method returns the assigned cluster for each data point.
- The **fit_predict** method fits the K-Means model and predicts the assigned cluster for each data point.
- Finally, the **replace_with_cluster_centers** method replaces each data point with its assigned mean.

The **fit** method iteratively assigns points to clusters and re-computes means until convergence. The **max_iter** argument determines the maximum number of iterations before terminating the algorithm. If the means stop changing, the algorithm terminates early. The **predict** method assigns each data point to the nearest cluster using the computed means. The **fit_predict** method combines the **fit** and **predict** methods into a single call. The **replace_with_cluster_centers** method replaces each data point with its assigned mean.

The implementation uses the **NumPy** and **math** libraries for mathematical calculations, and the **tqdm** library for displaying the progress of the fitting process.

Some Modifications to make standard K-Means Optimal:

- Use the **NumPy** library to vectorize calculations wherever possible, as it is much faster than looping over data points.
- Use a mini-batch K-Means implementation that updates mean using a small subset of the data rather than the entire dataset. This can reduce the computational cost while still producing reliable results.
- Use a faster initialization method like K-Means++ to select initial means.
    - Brief overview of K-Means++: K-Means++ is an algorithm used to initialize the centroid values in K-means clustering
        - Choose the first centroid randomly from the data points.
        - For each data point, calculate the minimum distance to the nearest centroid that has already been chosen.
        - Choose the next centroid randomly from the data points, with probability proportional to the square of the distance to the nearest centroid.
        - Repeat step 2 and step 3 until all k centroids have been chosen.

Results:

Resultant images where pixels are replaced by the nearest cluster centres:

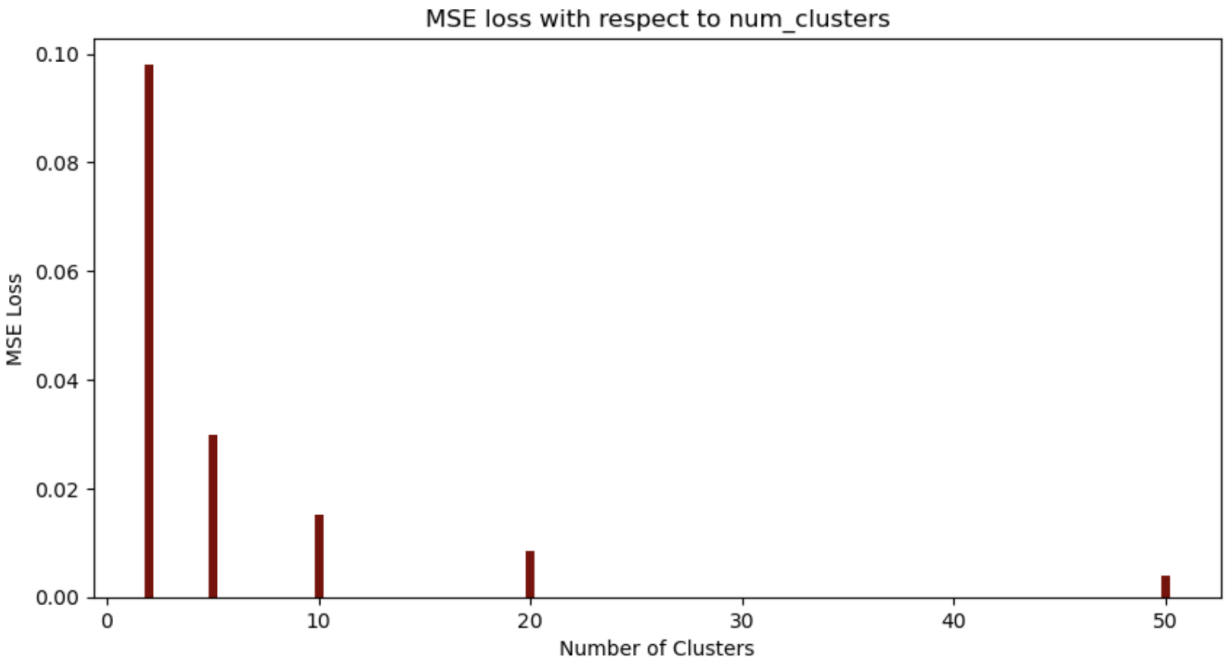
K=2 K=5 K=10



K=20 K=50 Original Image

Plot of Mean Squared Error as a function of the number of clusters:

```
  8%|█████                                            | 8/100 [00:21<04:03,  2.65s/it]
CLUSTER2
MSE: 0.09793520205971341
 28%|███████████████                                  | 28/100 [02:31<06:29,  5.41s/it]
CLUSTER5
MSE: 0.029992877201005547
 69%|█████████████████████████████████████            | 69/100 [11:33<05:11, 10.04s/it]
CLUSTER10
MSE: 0.015347217275378952
 75%|█████████████████████████████████████████        | 75/100 [24:26<08:08, 19.56s/it]
CLUSTER20
MSE: 0.008604219129227896
 90%|█████████████████████████████████████████████████| 90/100 [1:11:59<07:59, 47.99s/it]
CLUSTER50
MSE: 0.004135898666352226
```


MSE loss with respect to num_clusters
MSE Loss
0.10
0.08
0.06
0.04
0.02
0.00
0
10
20
30
40
50
Number of Clusters