PCA Implementation:

Dataset:

The MNIST dataset is utilized for the task. It consists of 70,000 grayscale images of handwritten digits (0 - 9), each in 28 x 28 pixel format. A random subset of 1000 images are considered as the dataset, 100 from each class 0 – 9.

Data Preparation:

The subset of images (1000) are read as numpy arrays wherein each image would be read as 2d array of dim(28, 28). The arrays are flattened to obtain a 1d array representation of the image of dim(784,). The dataset of 1000 images is represented by a numpy array of dimension (1000, 784) indication 1000 rows corresponding to 1000 images and 784 columns i.e., each row corresponds to one flattened image.

Implementation:

The Principal component analysis is implemented as a class which accepts the number of components as input when initialized. The number of components should be less than or equal to 784. By default the number of components is set as 784. The init constructor method initializes the number of components.

The fit method:

It accepts the data as input. Mean centering is performed on the dataset i.e., the origin is moved to the mean position. The covariance matrix is computed on the mean centered data.

The eigen values and the eigen vectors of the covariance matrix is obtained. The eigen values indicate the amount of variance captured in each of the principal component direction. The direction of each eigen vectors is adjusted according to sign of the element with the maximum absolute value. The index of the element with max absolute value is found. The sign at the index is determined and each element of the eigen vector is multiplied with the sign such that the sign of element with max absolute value is positive.

When reconstructing data, maintaining a consistent orientation of the eigenvectors across components is essential. If the directions are arbitrary, reconstructed data points have variations, as the direction of each component could vary unpredictably across different runs. By setting the direction of each eigenvector based on a specific

rule such as aligning the sign with the largest absolute element in the eigenvector, ensures stabilized reconstruction.

The eigen value and corresponding eigen vector pair is sorted in decreasing order of the eigen values. The explained variance and the cumulative explained variance is computed of each components is determined.

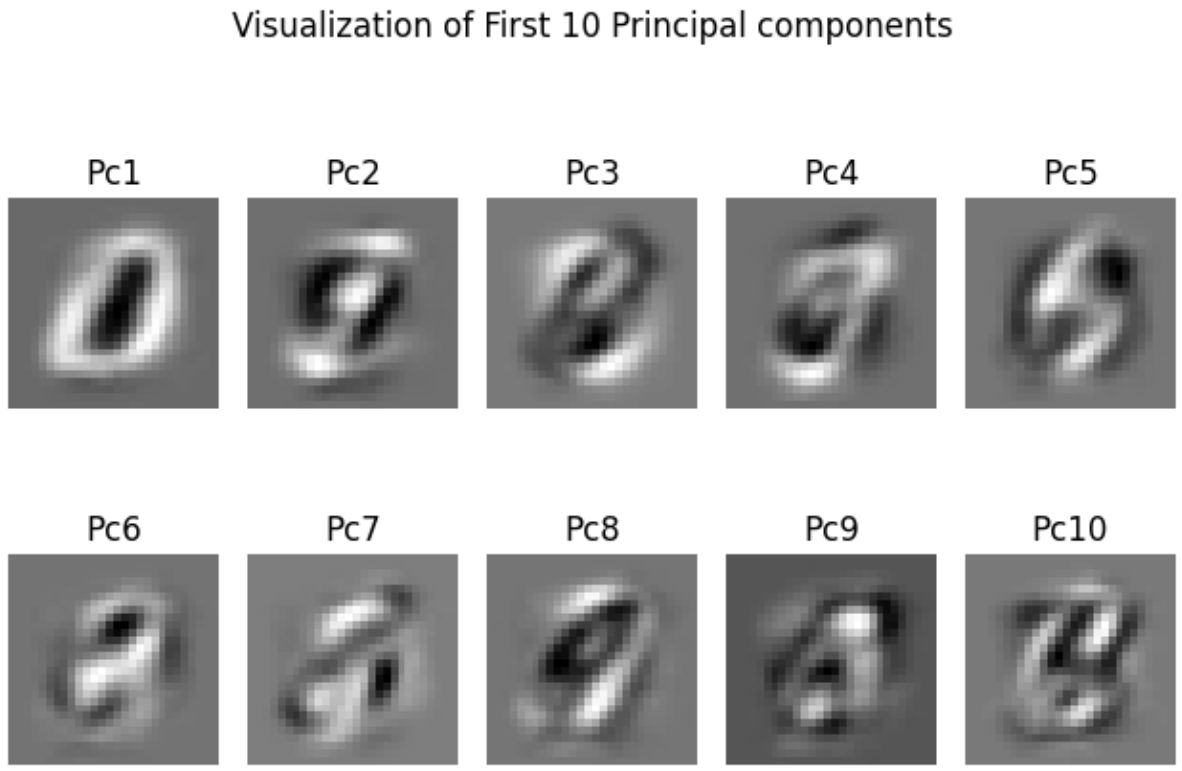
The Transform method:

The method accepts the data as input, mean centering on the data is performed. The modified data is then projected onto the principal components.

The Inverse Transform method:

The method accepts the transformed data as input and the data is projected onto the components and their corresponding means are added to remove the mean centering component to obtain the reconstructed data.

Visualization of the first 10 principal components:



The amount of variance explained by first 25 PCA components individually is:

The variance explained by component 1: 0.0969

The variance explained by component 2: 0.0744

The variance explained by component 3: 0.0692

The variance explained by component 4: 0.0544

The variance explained by component 5: 0.0488

The variance explained by component 6: 0.0457

The variance explained by component 7: 0.0349

The variance explained by component 8: 0.0303

The variance explained by component 9: 0.028

The variance explained by component 10: 0.0215

The variance explained by component 11: 0.0203

The variance explained by component 12: 0.0191

The variance explained by component 13: 0.0183

The variance explained by component 14: 0.0172

The variance explained by component 15: 0.0164

The variance explained by component 16: 0.0151

The variance explained by component 17: 0.013

The variance explained by component 18: 0.0127

The variance explained by component 19: 0.0124

The variance explained by component 20: 0.0116

The variance explained by component 21: 0.0108

The variance explained by component 22: 0.0101

The variance explained by component 23: 0.0096

The variance explained by component 24: 0.0095

The variance explained by component 25: 0.009

The cumulative variance explained by first 25 components:

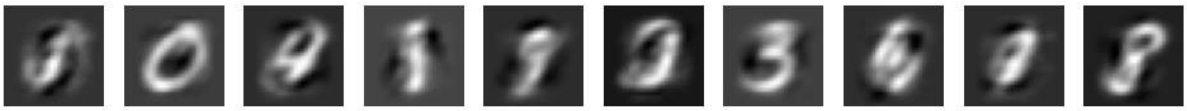
0.09686582, 0.17124482, 0.2404878, 0.29485035, 0.34365902, 0.38933237, 0.4242533
0.45456899, 0.48257209, 0.5040934, 0.52440704, 0.54346536, 0.56172905, 0.57888647

0.59524684, 0.61034748, 0.62333752, 0.63604561, 0.64846497, 0.66009486, 0.67087915
0.68098181, 0.69062208, 0.70010023, 0.70909706.

Approx 70% of the variance is captured by the first 25 PCA components.

Reconstruction:

Reconstructed Images using 10 dimension PCA representation



Reconstructed Images using 100 dimension PCA representation

Reconstructed Images using 140 dimension PCA representation

Reconstructed Images using 200 dimension PCA representation

Reconstructed Images using 250 dimension PCA representation

Reconstructed Images using 500 dimension PCA representation

The above images indicate the digits reconstructed using different dimensional PCA components. As indicated by the images, the digits reconstructed using less number of PCA components are vague and it is difficult to identify the digits by the images because the components captures lesser amount of variance in the data(information). 10 PCA components capture only 50% of variance present in the data. The digits reconstructed using Higher number of PCA components have increased clarity and would be sufficient to perform down stream tasks such as digit classification.

The digits reconstructed using 140 PCA components is sufficient to perform down stream tasks such as digit classification. The reconstructed images are of fine clarity and the 140 PCA components capture around 95.57% of variance present in the data.

Lloyd's algorithm implementation:

Data Preparation:

The csv file contains 1000 rows and 2 columns. Each row can be viewed as a (x, y) coordinate of the point in the 2d space. The csv file is read as pandas data frame and is converted into a numpy array for easier dimension manipulations.

Implementation:

The Kmeans algorithm is implemented as a class. It accepts number of clusters as input when initialized. By default the number of clusters is set to 2.

The init constructor method :

It initializes the number of clusters. It also initializes the centroids container to **None** which would contain the cluster means and label container to **None** which would contain the label(cluster indicator) of each of the datapoint in the dataset.

The fit method:

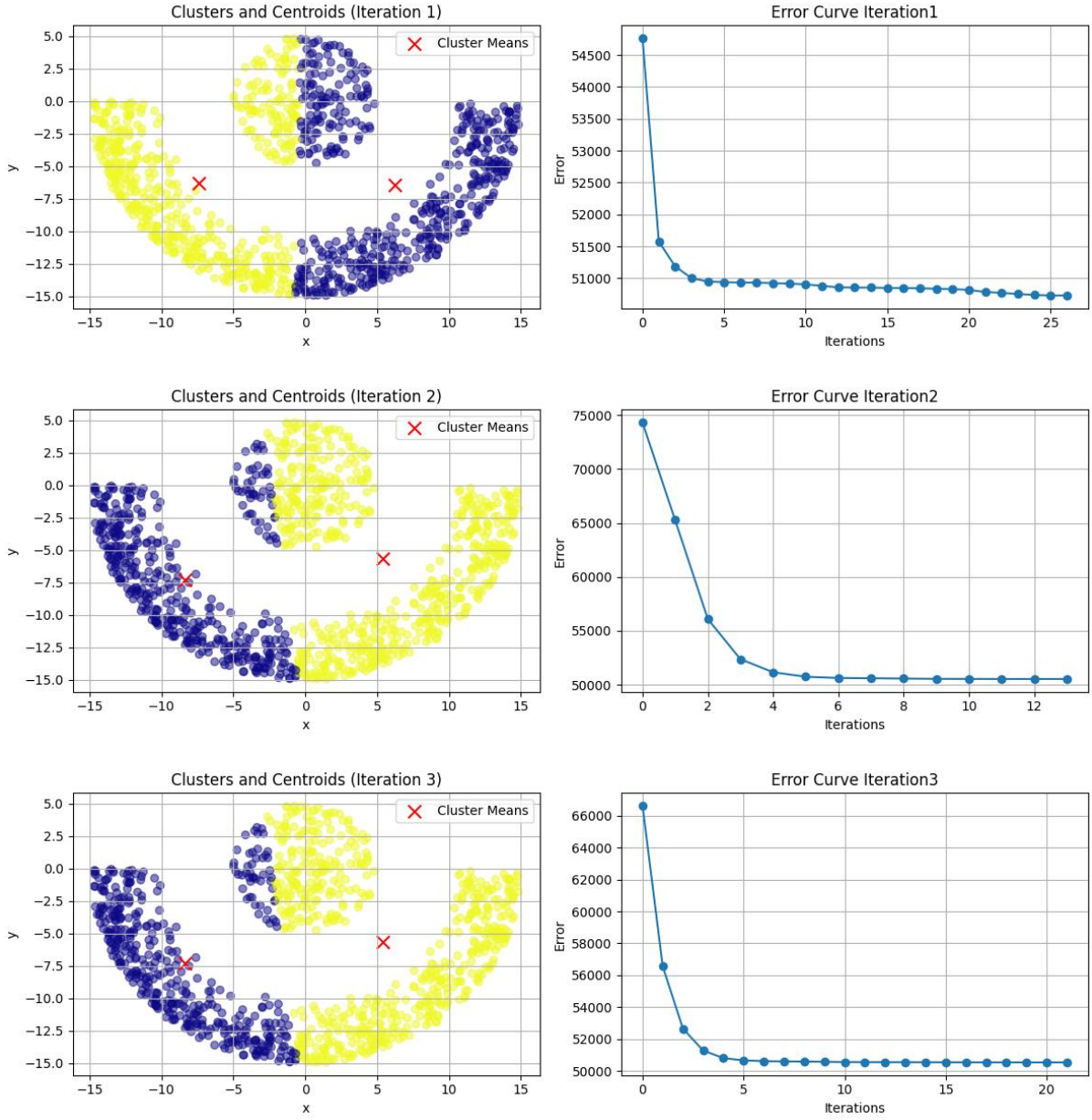
It accepts data as input. It randomly selects k points to be the k cluster means. It assigns the points to each cluster based on the distance of each point with each of the cluster centroid. It assigns point to that cluster whose distance with respect to cluster mean is minimum. It updates the cluster centroids means based on the points assigned to a particular cluster. It computes the error that is the difference b/n the point and the cluster mean. If the error computed is equal to the error computed in

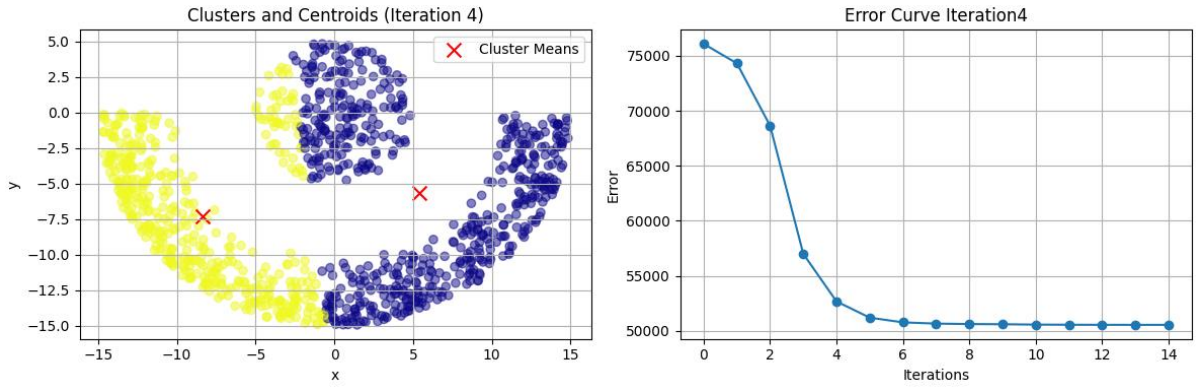
the previous iteration then the algorithm has converged and it stops else the algorithm continues the above mentioned steps.

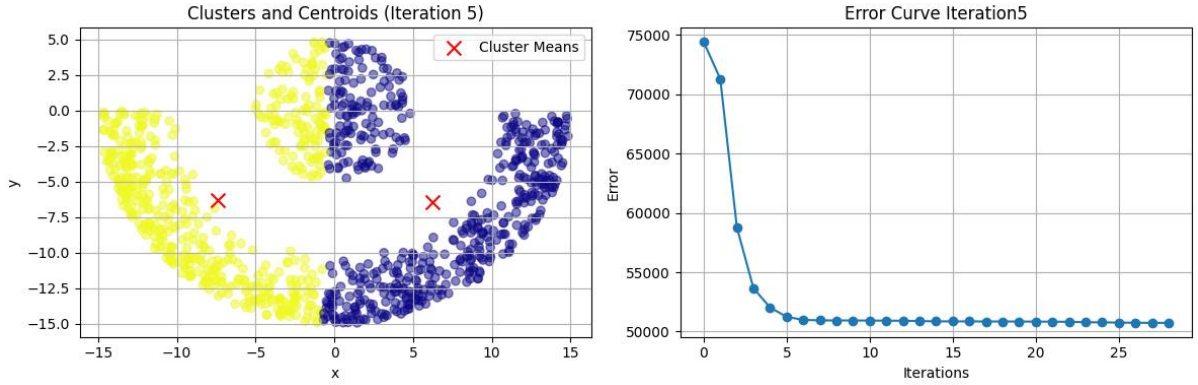
The predict method:

It accepts the data as input, it computes the Euclidean distance of each point with respect to each of the cluster means and assigns labels based on the distance computed. The point gets assigned to the cluster whose distance with respect to the cluster mean is minimum.

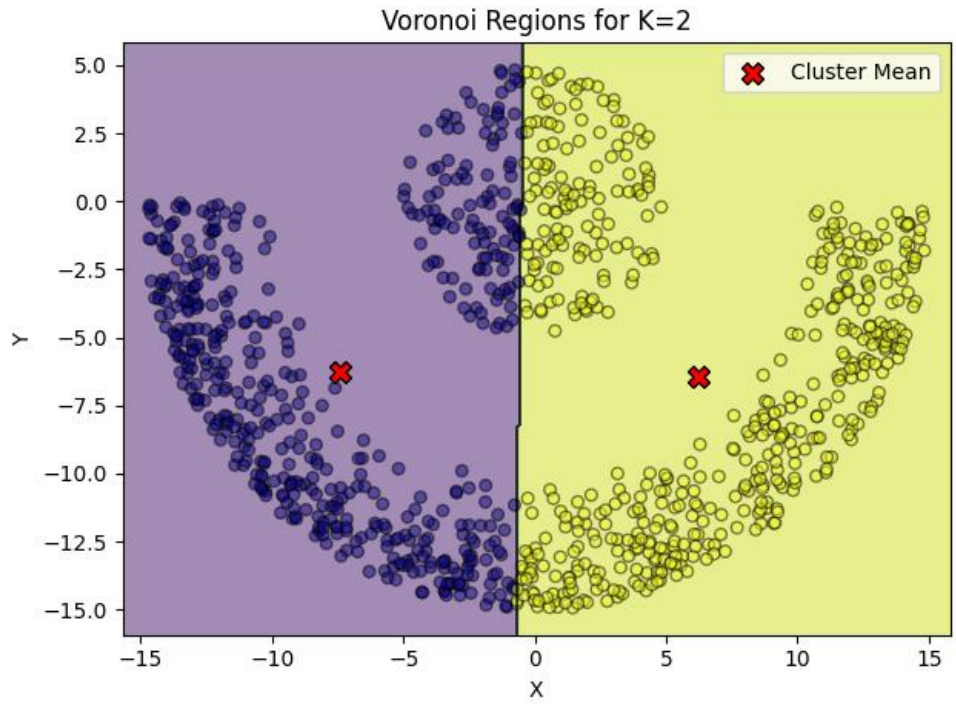
The error values and the clusters obtained for five different random initialization:

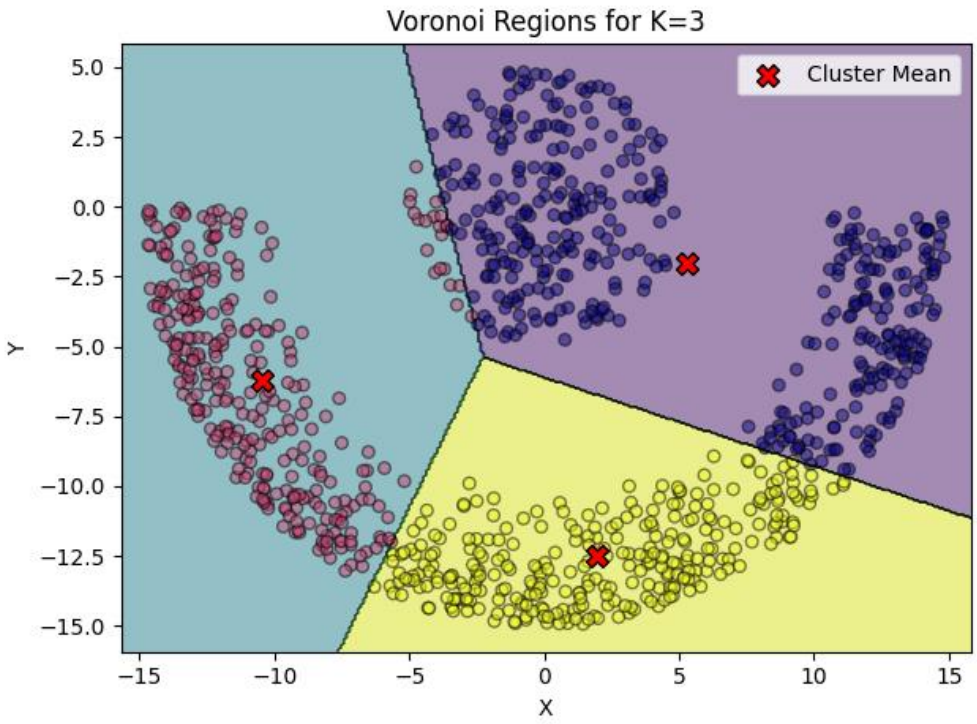


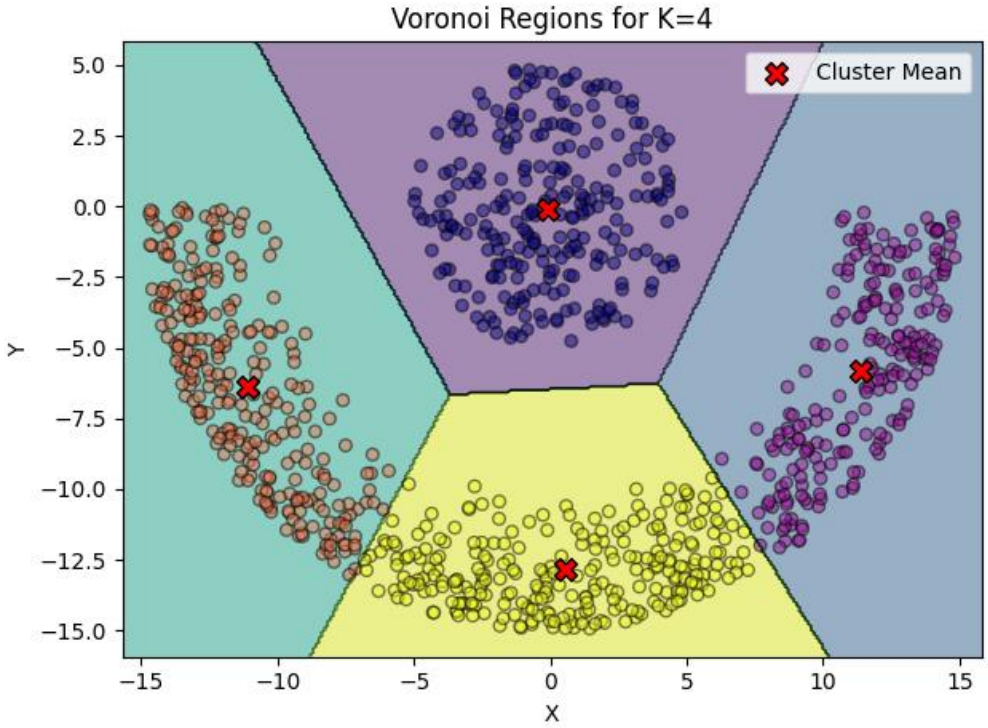


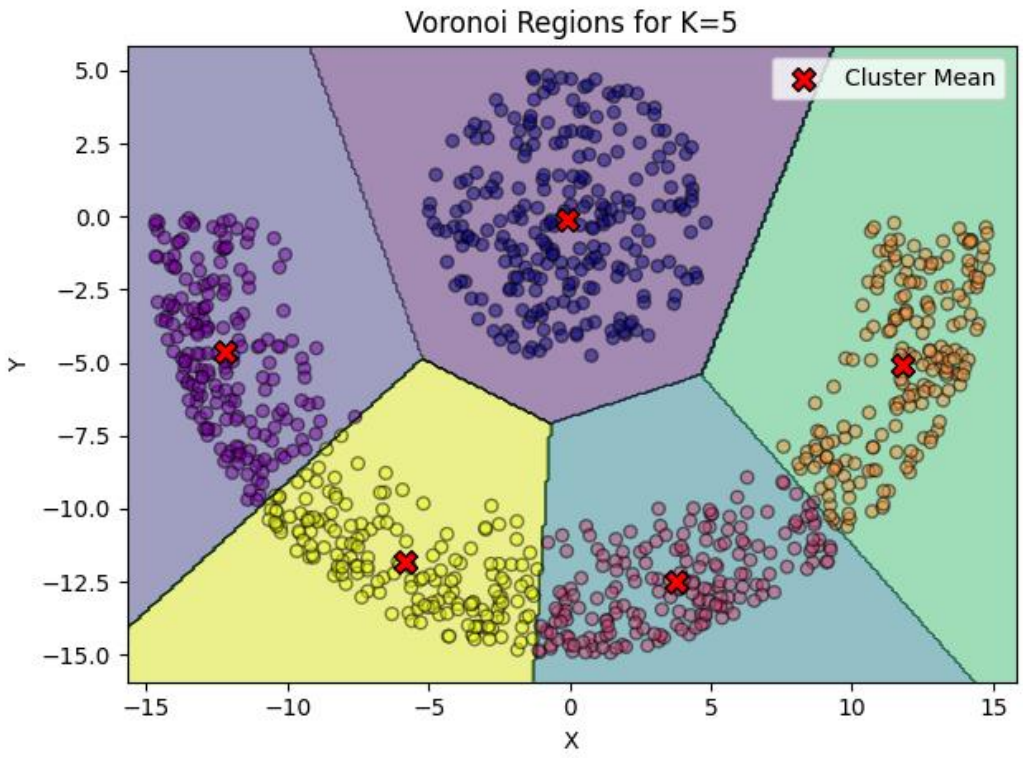


For a fixed arbitrary initialization the Voronoi regions obtained for different values of K:




Voronoi Regions for K=3
Cluster Mean
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
Y
−15
−10
−5
0
5
10
15
X


Voronoi Regions for K=4
Cluster Mean
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−15.0
Y
−15
−10
−5
0
5
10
15
X



The Lloyd algorithm is not a good procedure to cluster the above dataset. The algorithm produces Voronoi regions which clusters the dataset. Voronoi regions are finite intersections of affine half spaces, from the images it is evident that there exists two clusters present in the data. The points belonging to the circular region form one class and all the points in the U/ cup shaped region belongs to the second class. The Lloyd's algo with k=2 tries to fit a line which divides the data into two clusters and it is not possible to fit a line which would capture the inherent structure required to cluster the data. The algorithm is limited to linear cluster boundaries.

To capture the non linear cluster boundary **Spectral clustering** can be utilized. It uses the graph of nearest neighbours to compute a higher-dimensional representation of the data, and then assigns labels using a *k*-means algorithm. It computes the eigen vectors of the kernel matrix , row normalizes the data(eigen vector matrix) and runs Lloyd algorithm considering each row as the new data. This algorithm would capture the non linear cluster boundary and thus classify the two clusters present in the data.

Submitted by,

Manoj Kumar.CM

DA24S018