CS5691 – Pattern Recognition in Machine Learning

Programming Assignment 1

Lab Report

Name: - Jay Jitendra Prajapati

Roll No.: - ME21B143

TABLE OF CONTENTS

1) PCA Algorithm on MNIST Dataset

- Loading the MNIST Dataset from Hugging face using load_dataset library.
- Extracting 1000 images from dataset by maintaining a count array
- Convert the images into NumPy arrays and flatten them.

1.1 PCA Algorithm

1.1.1 Code for Algorithm

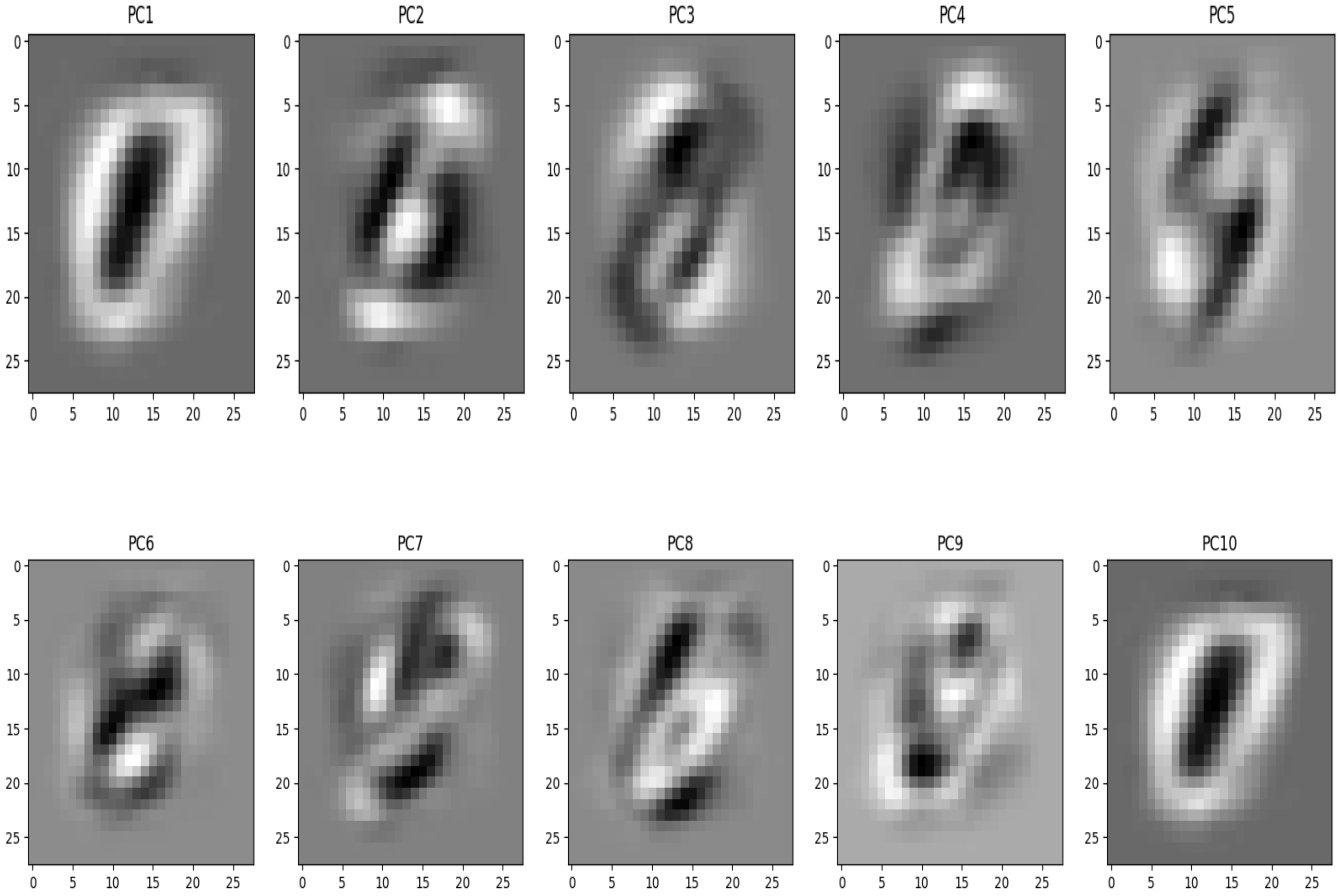
- Step 1: - Center the dataset by calculating the mean and subtracting it from every data point.

- Step 2: - Calculate the Covariance matrix using the formula.

$$C = (\sum x_i * x_i^T) / n$$

- Step 3: - Calculate the eigenvalues and eigenvectors of Covariance matrix with NumPy.linalg.eigh library.

1.1.2 Images of Principal Components



1.1.3 Variance in the dataset

- The eigenvalues are the measure of variance in the dataset explained by each principal component.

- Variance is calculated by dividing each eigenvalue by the total sum of all eigenvalues.

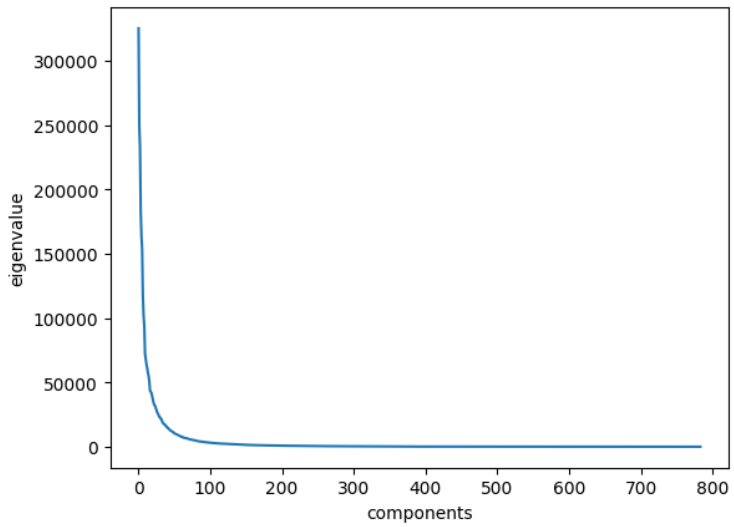
Var[i] = eigenvalue[i]/∑eigenvalues ∀ i

- The variance of the first 10 components is as follows. (Rest values can be checked from code)

  Var [] = [9.68658170e-02, 7.43790038e-02, 6.92429787e-02, 5.43625495e-02, 4.88086687e-02, 4.56733529e-02, 3.49209319e-02, 3.03156851e-02, 2.80030978e-02, 2.15213188e-02]

1.2 Reconstructing the dataset using different representations

1.2.1 Dimension k for downstream task



- So, from the above graph, we can conclude that only a small number of eigenvalues are required. Its better to store less data for given dataset.

- So, for a downstream task we can use the rule of thumb.

$$\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{n} \lambda_i \geq 0.95$$

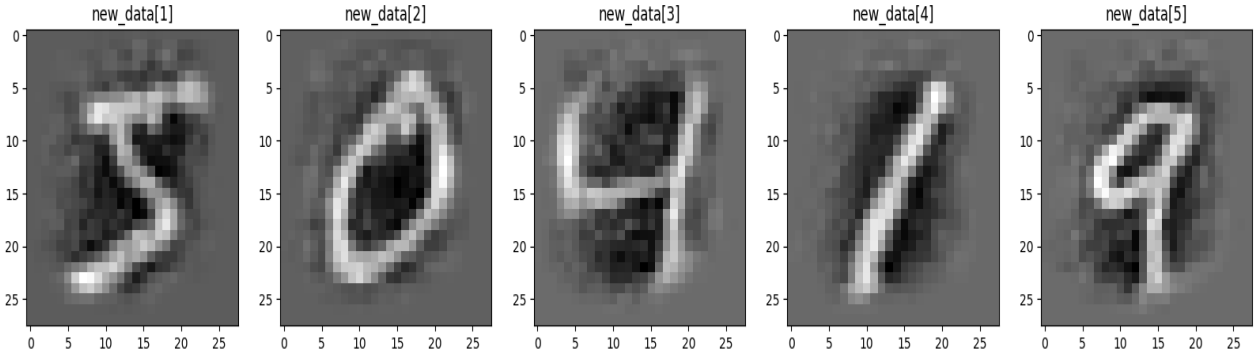
- On solving, we get **k = 130**.

1.2.2 Images of new dataset

- Every data is a linear combination of top k eigenvectors.

$$x_i = \sum_{j=1}^{k} \left(x_i^T . w_j \right) * w_j \qquad \forall \ \mathbf{i}$$

- Reconstructed first five images are shown below.



1.3 Kernel PCA

- Step 1: - Compute the kernel matrix from a given kernel function as follows.

$$\boldsymbol{K(i,j) = k(x_i, x_j)}$$

- Step 1.5: - Center the kernel matrix

- Step 2: - Eigen decompose the matrix K and get eigenvectors beta and eigenvalues.

$$\boldsymbol{\{\beta_1, \beta_2, \ldots . \beta_n\}}$$

$$\boldsymbol{\{\lambda_1, \lambda_2, \ldots . . \lambda_n\}}$$

- Step 3: - Compute alphas from betas as follows.

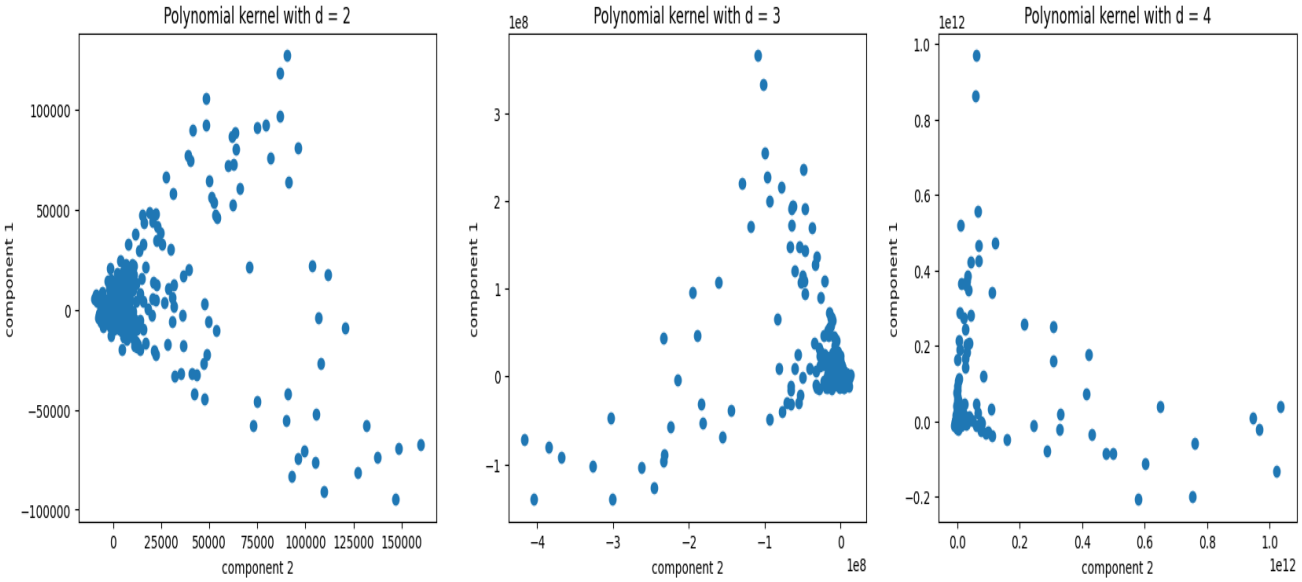
$$\boldsymbol{\alpha_k = \beta_k / \sqrt{n\lambda_k}} \qquad \forall \ \mathbf{k}$$

- Step 4: - Compute $\phi^T(x)*w_i$ for every i.

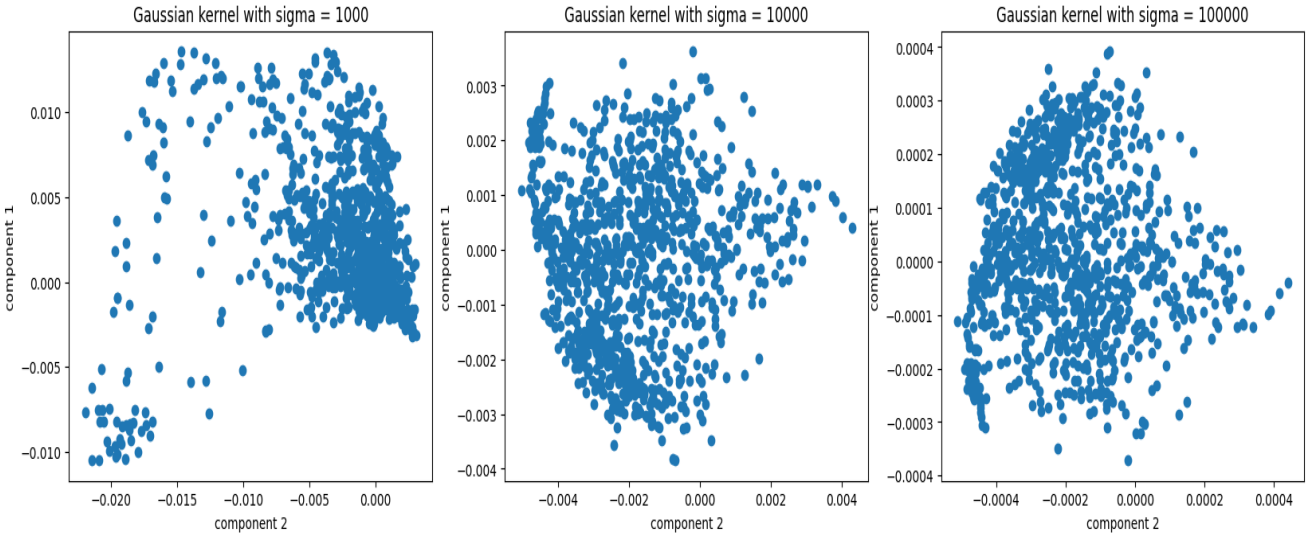
$$\boldsymbol{\phi^T(x) * w_i = \sum_{j=1}^{n} \alpha_{ij} * k(x, x_j)} \quad \forall \ \mathbf{i}$$

- Step 5: - Plot the projection of each point in the dataset onto the top-2 components.

1.3.1 Polynomial kernel



1.3.2 Gaussian kernel



- The values of sigma chosen are {1000,10000,100000} because for these values points are more scattered around in comparison to smaller values of sigma.

- So, for better understanding of data higher value sigma is preferable.

1.4 Best suited Kernel for this dataset

- The **Gaussian kernel** is better suited for this dataset because from the plots we can see that in gaussian kernel's plot, all data points are spread out whereas in polynomial kernel's plot many points are merging. For better understanding, points must be spread out.

- The Gaussian kernel with higher value of sigma is also preferred as points are more spread out.

- If points are merging some information might be lost, so better to have all points spread out. That's the reason gaussian kernel is preferred.

2) Clustering Algorithm on given dataset

- To import the dataset, panda library is used.

2.1 K-Means Algorithm with different random initialization (k=2)

- Step 1: - Initialize z (cluster indicator array) with some random value (0 or 1).

- Step 2: - Calculate the means of all clusters.

$$\boldsymbol{\mu_k = \sum_{i=1}^{n} x_i * 1(z_i = k) / \sum_{i=1}^{n} 1(z_i = k)}$$

$\forall$ **k**

- Step 3: - (Reassignment step) Reassign all data points according to the following equation.

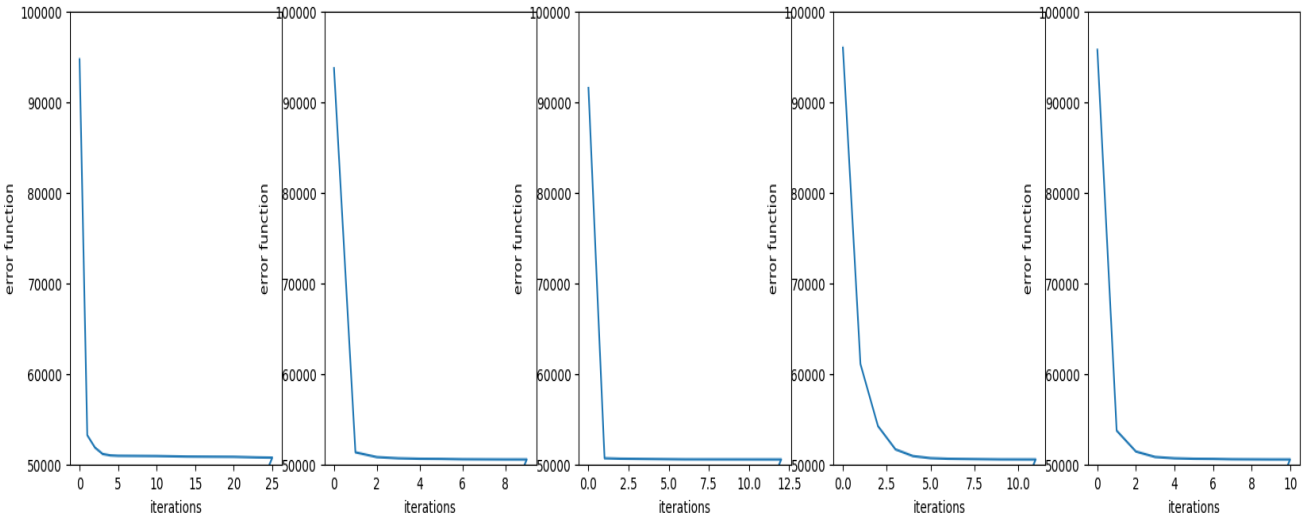
$$\boldsymbol{z_i^{t+1} = arg\ min\ k \left\| (x_i - \mu_i^t) \right\|^2}$$

- Step 4: - Check if there is no change in any cluster indicator then stop the algorithm and it means every point is in its desired cluster.

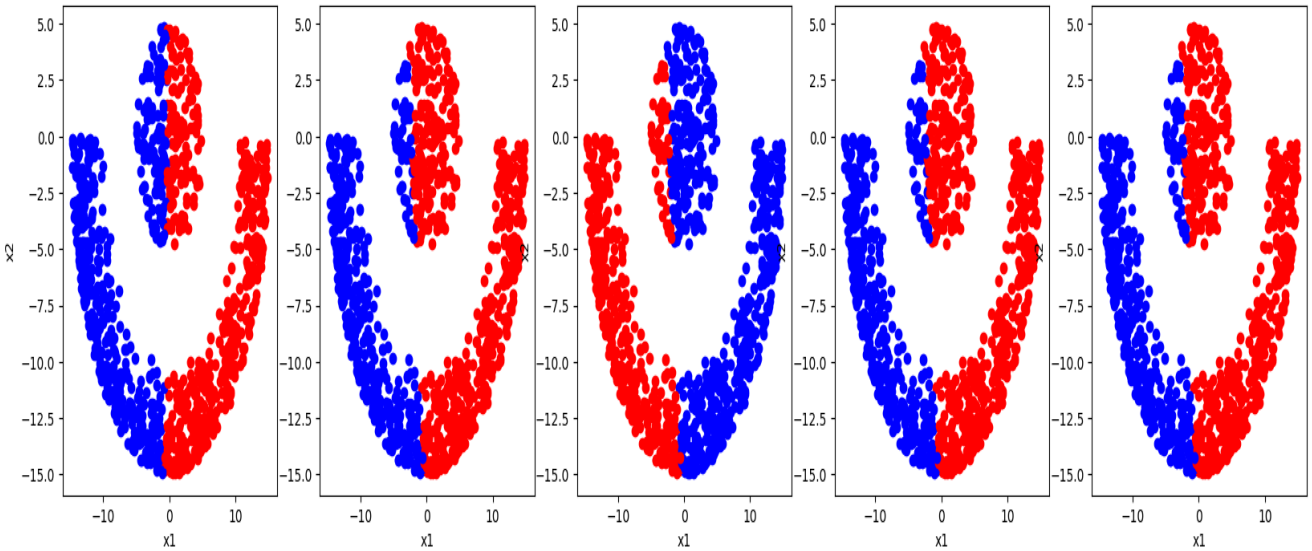
- Error calculations: -

$$\boldsymbol{min} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \left\| (\boldsymbol{x_i} - \boldsymbol{\mu_{z_i}}) \right\|^{\boldsymbol{2}}$$

Error vs iterations curve for 5 different random initializations

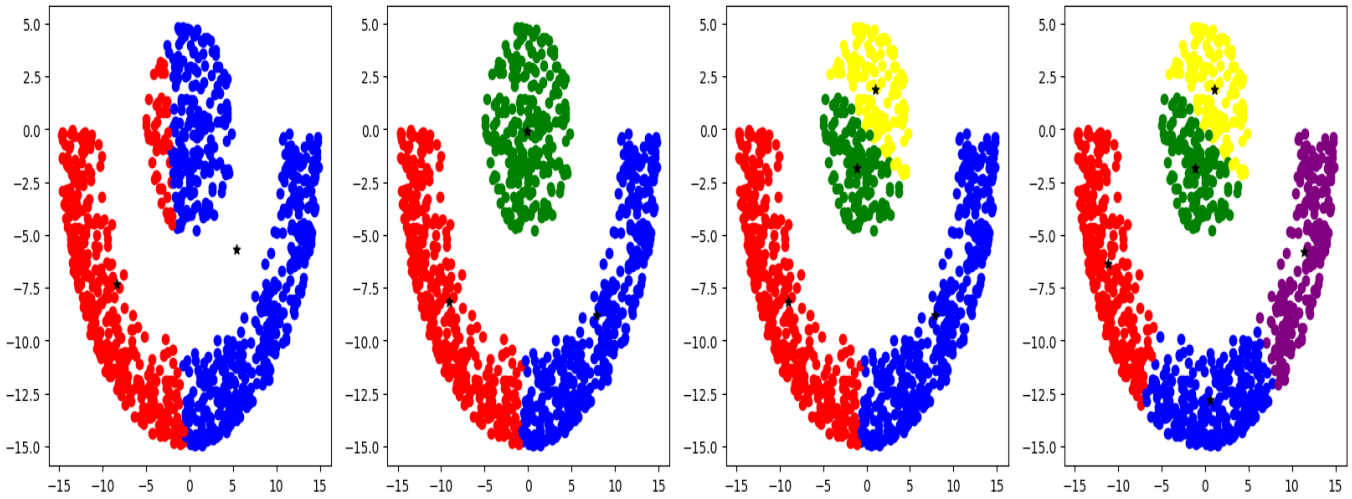


Clustered dataset



2.2 K-Means Algorithm with K = {2,3,4,5}

- Fix a random initialization and run K-Means algorithm for k = {2,3,4,5}.



- Black stars represent cluster centers and different colors represent Voronoi regions for cluster centers.

2.3 Spectral Clustering

- Step 1: - Compute the kernel matrix from kernel function.

$$K(i,j) = k(x_i, x_j)$$

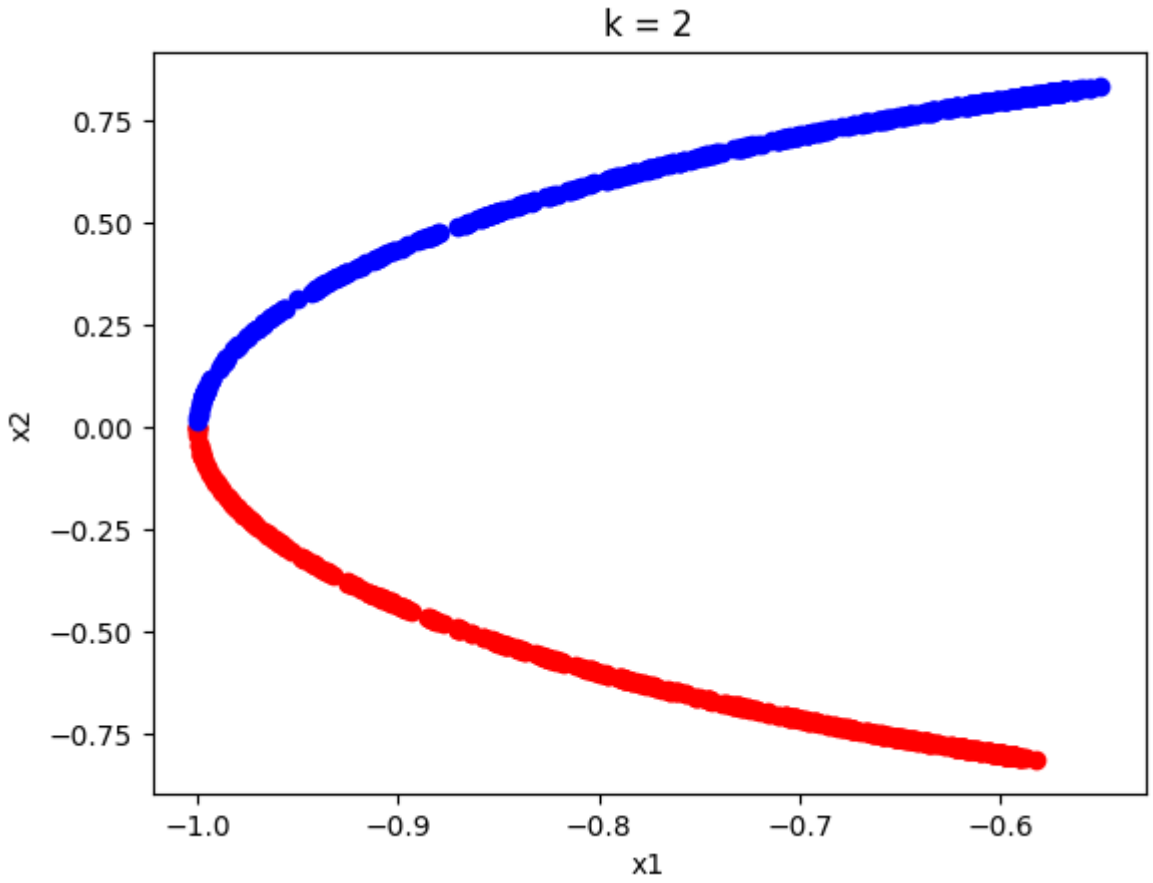
- Step 2: - Compute the H matrix which consists of top k eigenvectors of kernel matrix K.

$$\mathbf{H = [h_1\ h_2 \ldots h_k]}$$

- Step 3: - Normalize the rows of H matrix.

- Step 4: - Run the K-Means Algorithm assuming each row as new data points.

Dataset after spectral clustering algorithm



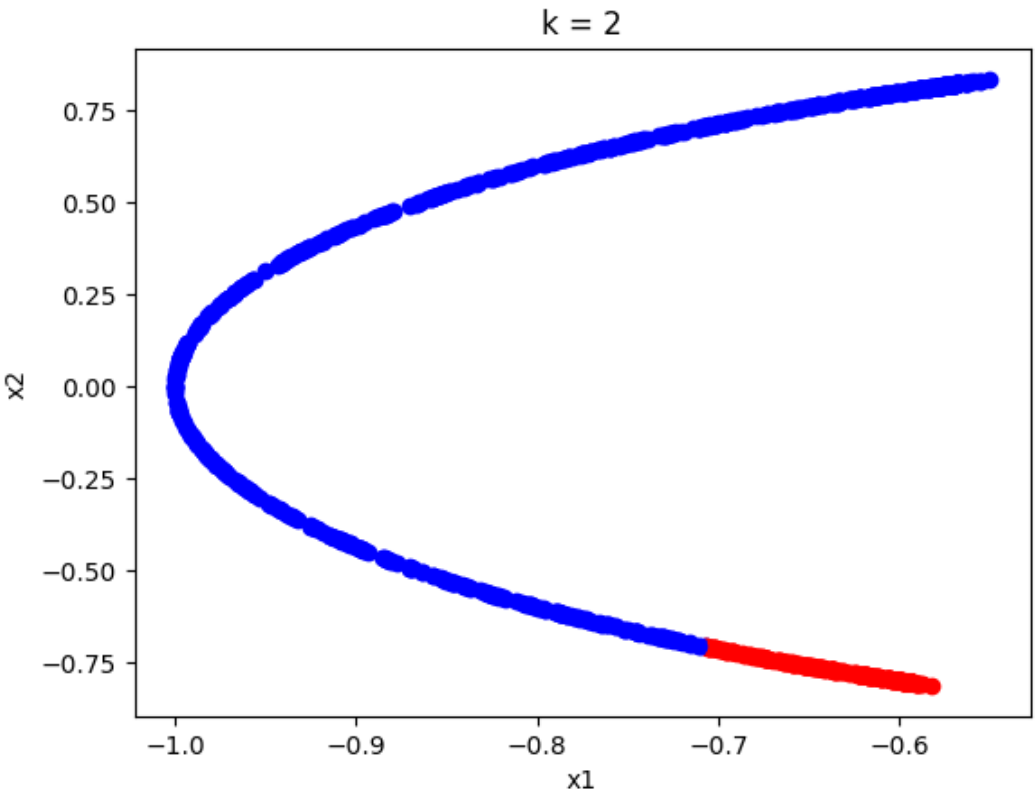
2.4 Variation in spectral clustering algorithm

- Repeat the first three steps of spectral clustering algorithm.

- Assign data point I to cluster l whenever.

$$l = arg\ max_{j=1,..k}\ v_i^j$$

- Where v^j is eigenvector of kernel matrix associated with the j-th largest eigenvalue.



- Calculate the error in both cases spectral clustering and here. Below are the results for it.

  Spectral clustering: - Error = 53769.29337547361

  Variations in Spectral clustering: -
  Error = 54001.96669171018

- So, our main aim was to minimize error so this variations did not help us much as error is still lesser in spectral clustering.

- So overall, this mapping still did a good job as it created similar results as spectral clustering.