ASSIGNMENT 1:- PCA and K-Means

CS5691:Pattern Recognition and Machine

Course Instructor : Arun Rajkumar

Pragalbh Vashishtha **MM19B012**

February 2022

1 Introduction

In this assignment we look over Principal Component Analysis, Kernel-PCA, clustering using K-means, Spectral relaxation of K-means using Kernel-PCA, and a few modification experiments.

2 Q1) Principal Component Analysis and Kernel PCA

2.1 Synopsis

Principal Component Analysis is a linear dimensionality reduction algorithm. It does so by creating new uncorrelated variables that successively maximize variance.

The steps involved in PCA are;-

- The data-set is centered by calculating mean of each variable.
- The covariance matrix C is computed
- Eigen-decomposition of C is done to get Eigen-values and Eigen-vectors.
- The Eigen-values correspond to the contribution of the Principal Component to variance, while the Eigen-vectors correspond to the direction of the Principal Components

2.2 Q1) i) PCA Algorithm

Given the data-set in 1000x2 dimensions, we take transpose of the data-set to get the standard dxn matrix. We then compute the covariance matrix as $C = XX^T$, which returns a 2x2 dim matrix. We find out the eigen-values and eigenvectors by employing *np.linalg.eigh* method in python, which returns normalized eigenvectors in order.

We see the following from running PCA on the dataset:-

$$Eigenvalue_1 = 17.13191440244437,$$
$$Eigenvalue_2 = 14.489604749330642.$$
$$\text{Eigenvector1} = [0.323516, 0.9462227]^T$$
$$\text{Eigenvector2} = [-0.9462227, 0.323516]^T$$

We need amount of variance, we know,

$$Cw_1 = \lambda w_1$$

$$w_1^T C w_1 = \lambda w_1$$

$$w_1^T (\tfrac{1}{n} \textstyle\sum_{i=1}^{n} x_i x_i^T) w_1 = \lambda \text{ hence,}$$

$$\lambda = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} (x_i^T w_1)^2$$

We can therefore look at eigenvalues directly. The first PC contributes 54.17% to variance, while the second PC contributes 45.83%

2.3 Q1) ii) PCA Algorithm without centering

The observed values of mean for the data-set are [4.075e-07, 2.227e-07], which is very small compared to the scale of the values of X[i] in the data-set. This is because the data is mostly elliptically distributed around the origin.

Hence, when performing PCA without centring, no significant change was observed. The eigenvalues and eigenvectors,i.e., the principal components, are same as if centred.
This could happen due to two possibilities:-
The mean lies on one of the eigenvectors with the other component being minuscule, or the mean is minuscule compared to the data as a whole.

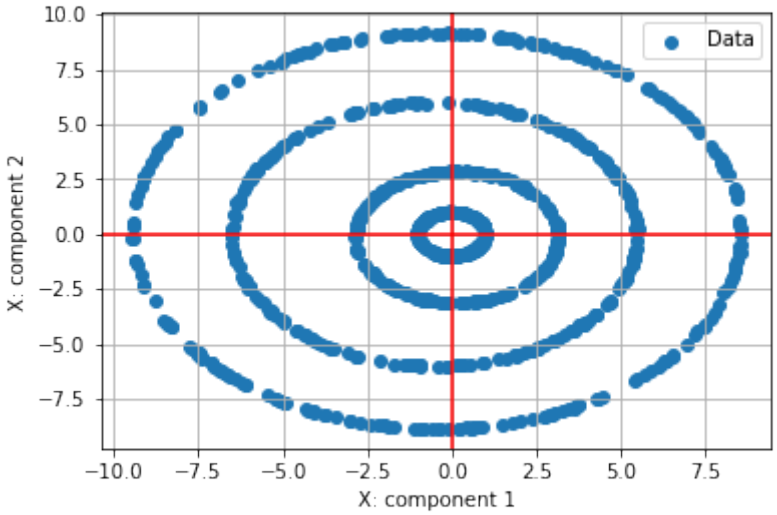


Figure 1: Data with red cross corresponding to the mean

Let us plot the Mean.

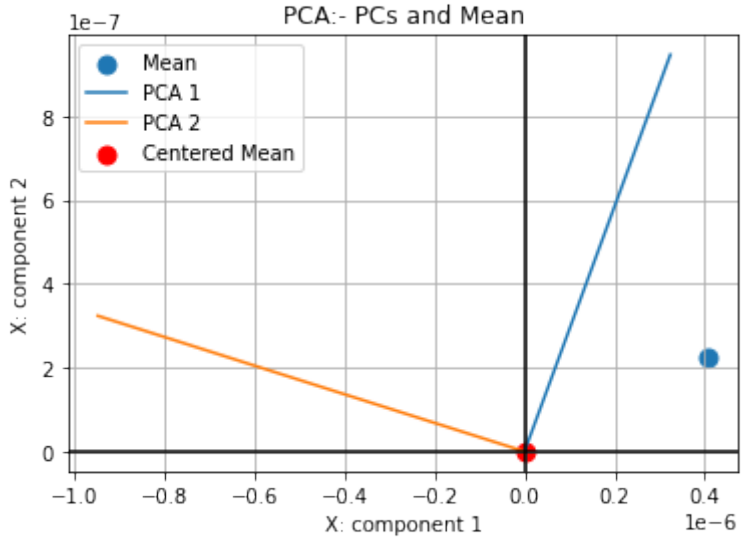


Figure 2: Mean is plotted on a scale of 10^{-7}, with scaled eigenvectors.

Note the mean is so minuscule compared to the rest of the data, centring plays no role.
In this case, as the mean is so minuscule, PCA preforms identical whether centered or not. Centering however is essential for most data-sets, whose mean have significant deviation form the origin. Not centering the data-set will result in the covariance matrix being faulty, which in-turn leads to erroneous results.

2.4 Q1) iii) Kernel PCA Algorithm

PCA is a linear method, It does an excellent job in data compression for linear data-sets, but it does not capture non-linearity in compression. Hence by using a kernel we map the data vectors to a higher dimension and hope to get a linearly separable region.

Kernel function are used to get generate a Kernel matrix K, which is of dimension nxn. The kernel function is essentially a dot product in the higher dimension, and hence the validity of the Kernel matrix can be ascertained using the Mercer's Theorem, which ensures symmetry and positive semi-definiteness.

Two kernels have been employed:-

- Polynomial Kernel, degree=2: $k(x, y) = (1 + x^T y)^2$
- Polynomial Kernel, degree=3: $k(x, y) = (1 + x^T y)^3$
- Radial Basis Function Kernel: $k_{\text{RBF}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$ for various σ values.

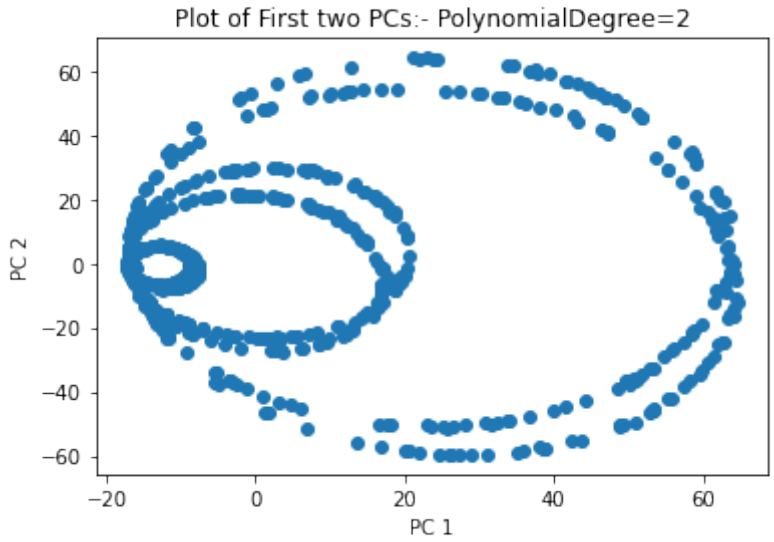


Figure 3: Kernel = $k(x, y) = (1 + x^T y)^2$

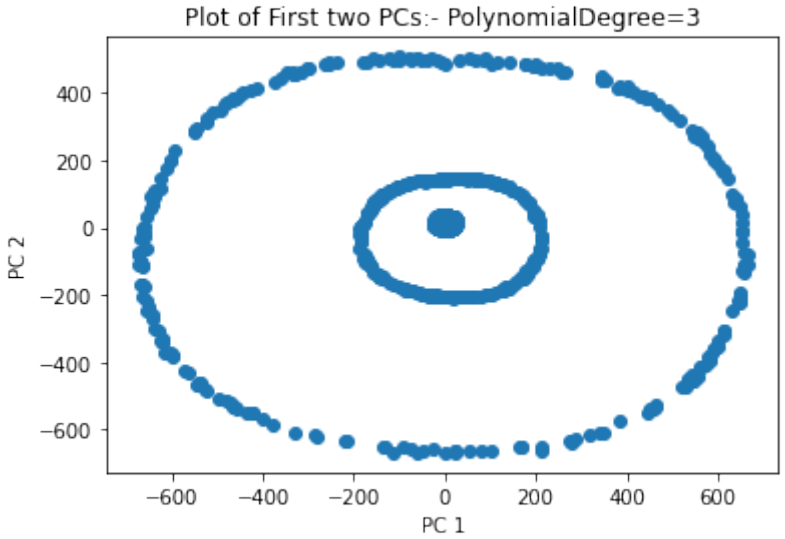
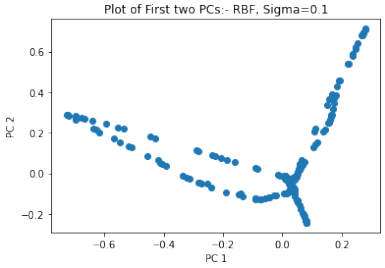
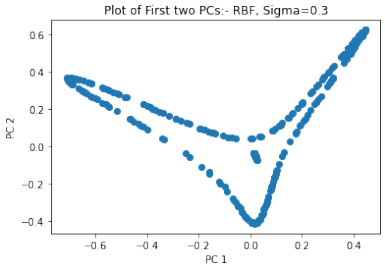


Figure 4: Kernel = $k(x, y) = (1 + x^T y)^3$

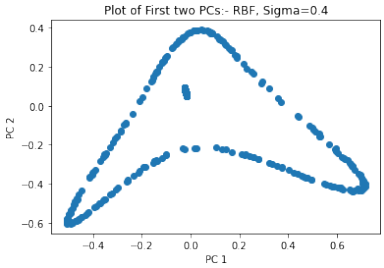


(a) RBF Kernel, $\sigma = 0.1$

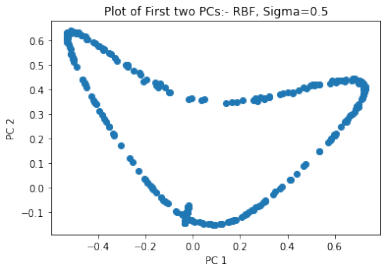
(b) RBF Kernel, $\sigma = 0.2$



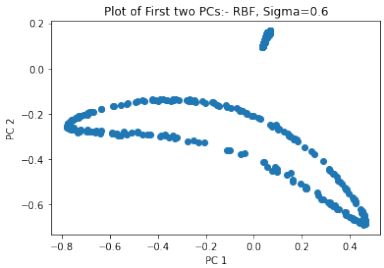
(c) RBF Kernel, $\sigma = 0.3$



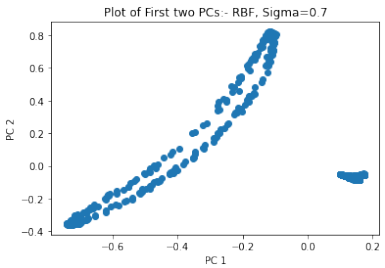
(d) RBF Kernel, $\sigma = 0.4$



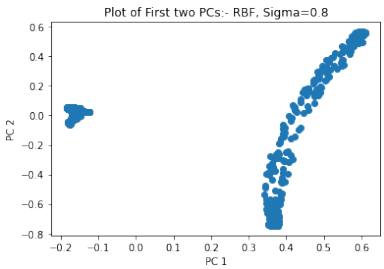
(e) RBF Kernel, $\sigma = 0.5$



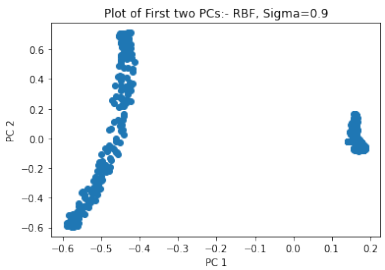
(f) RBF Kernel, $\sigma = 0.6$



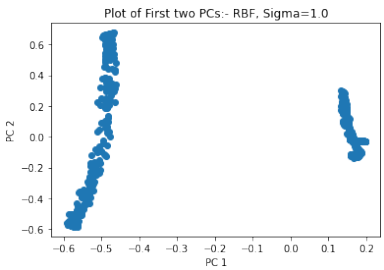
(g) RBF Kernel, $\sigma = 0.7$



(h) RBF Kernel, $\sigma = 0.8$



(i) RBF Kernel, $\sigma = 0.9$



(j) RBF Kernel, $\sigma = 1.0$

Figure 5: RBF Kernel: $k_{\text{RBF}}(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$, with various σ values, Note the linear separable regions

2.5 Q1) iv) Kernel PCA comparison

There is no standard definition of performance metric for Kernel PCA, but depending on the downstream task, two such metrics are proposed:-

- Data separation:- For downstream class separation, looks at the linear separability of data
- Data Compression:- The amount of variance explained by first few (in our case 2) Principal components

For the polynomial kernels $k(x,y) = (1+x^T y)^2$ and $k(x,y) = (1+x^T y)^3$ don't create linearly separable regions. While $K_{\text{RBF}}(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}\right)$ for σ values of 0.6,0.7 and more pronounceable for σ values of 0.8,0.9,1.0
Hence the best kernel for data separation is the RBF:- Radial Basis Function kernel.
For data compression, Polynomial kernel of degree 2 and degree 3 explains 68.53% and 73.41% respectively. The RBF Kernel explains only $2.45\% - 15.6\%$ for various values of sigma. Actually, the best in this regard is simple PCA which explains 100% of variance using 2 Principal components. Hence for pure data compression, Kernel PCA is not viable.

3 Conclusion:-

PCA helps us in data compression and representation. PCA gives two principal components with % variance contribution of 54.17 % and 45.83 % respectively.
Kernel PCA is employed to produce linearly separable regions. Polynomial Kernels do not produce linearly separable region, however, radial basis function kernel do for σ values of 0.8,0.9,1.0.

4 Q2) Clustering:- K-means, Spectral Clustering

4.1 Synopsis

K-means clustering is an iterative algorithm that aims to partition n data-points to k disjoint clusters, in which each data-point belongs to the cluster with the nearest mean, where the sum of squared distance is used as a measure of closeness.
The steps of the algorithm are the following:-

- Choose 'k' random points from the data-set to be the initial means for the 'k' clusters. Points are assigned to the cluster which has the nearest mean
- Till Convergence:- Find mean of each cluster at that iteration, reassign points to clusters with the nearest new means.
- Convergence occurs when the clusters assignments do not change, this also implies the error does not reduce further, hence the name, convergence.

4.2 Q2) i) Kmeans-Number of clusters:- 4

The K-means algorithm is run with k=4. to get random initialization, *np.random* is employed With 6 random initializations we observe the following:-

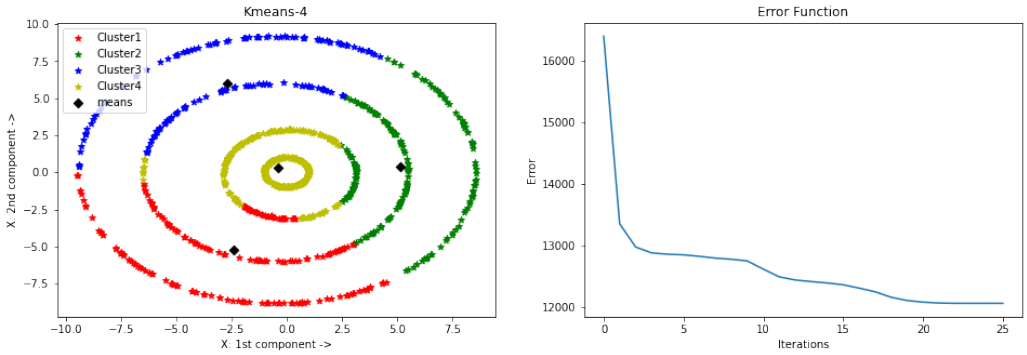


Figure 6: Random Initialization 1

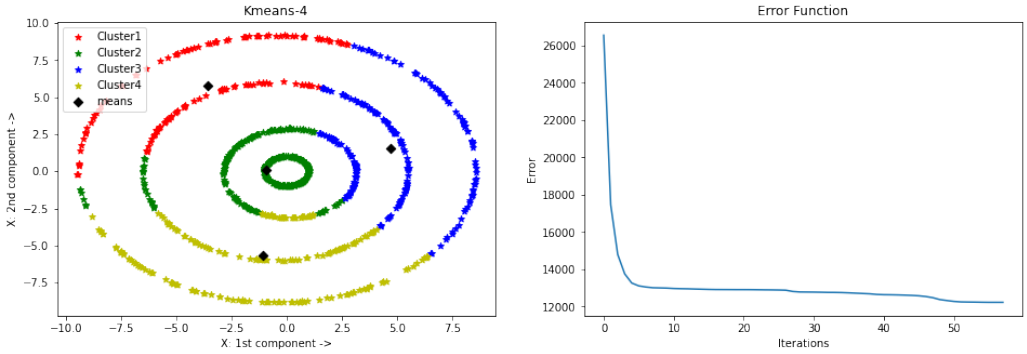


Figure 7: Random Initialization 2

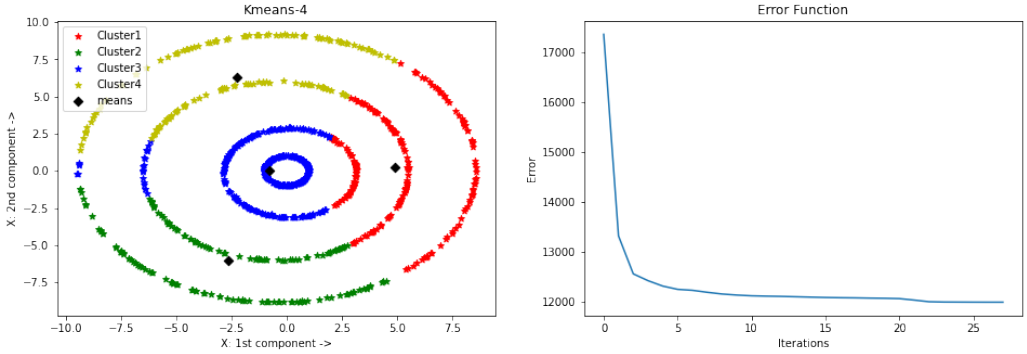


Figure 8: Random Initialization 3

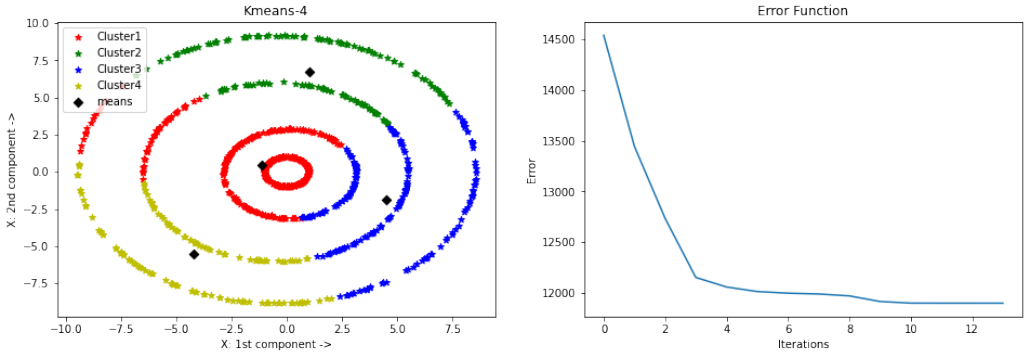


Figure 9: Random Initialization 4

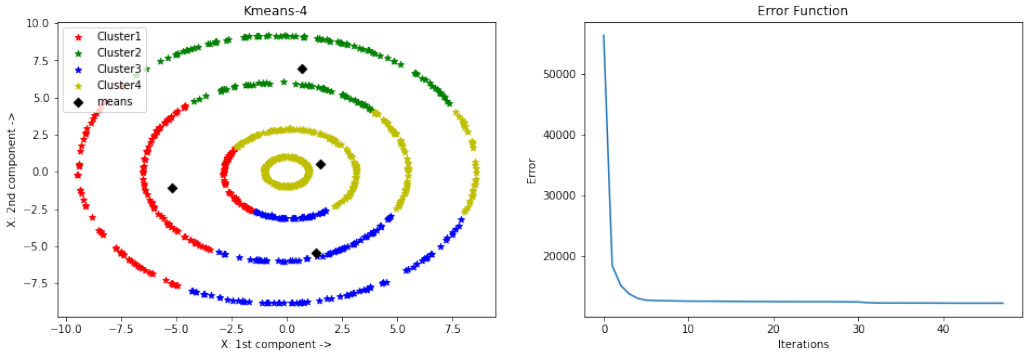


Figure 10: Random Initialization 5

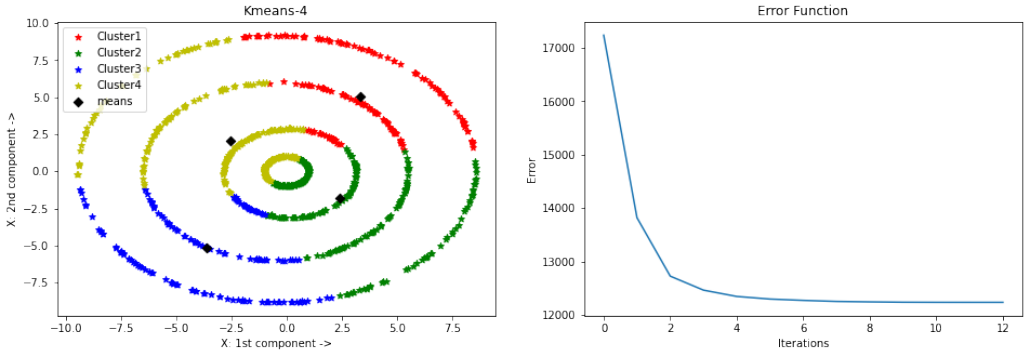


Figure 11: Random Initialization 6

The error here is defined by the function:- $Error = \left(\frac{\sum_{i=1}^{n} \|\mathbf{x_i} - \mu_{\mathbf{z_i}}\|^2}{n}\right)$. Here μ_{z_i} is the mean of the cluster x_i is assigned to. We note that although the algorithm converges, the error is quite high

4.3 Q2) ii) Kmeans:- Number of clusters with Voronoi

The initialization of the means are fixed. 5 means are generated. When the number of clusters is 2, the first two means are taken, when the number of clusters is 3, the first three are taken and so on.

The Voronoi regions of a cluster are the points which are closest to the mean of that cluster. The line that separates them, i.e., the decision boundary is where points are equidistant from two or more means.

Implementation:- *scipy.spatial* is emplyed to get voronoi regions, the colours for the regions are generated by creating a grid of all points in the x,y plane and assigning them to clusters obtained at convergence.

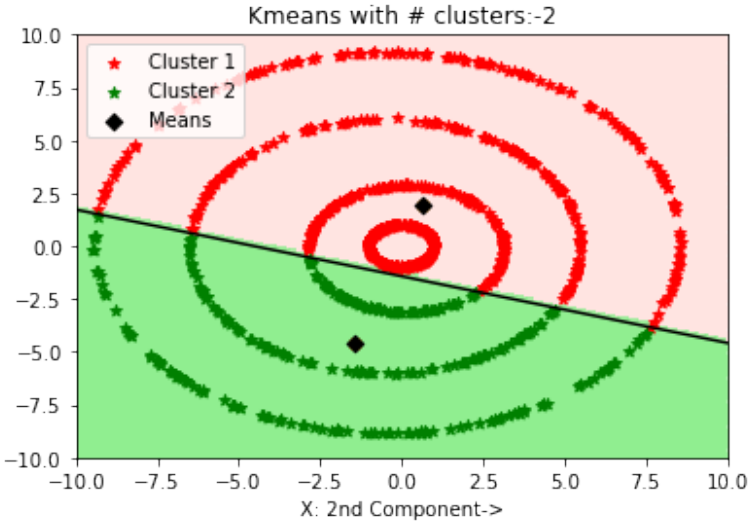


Figure 12: Kmeans 2 Clusters

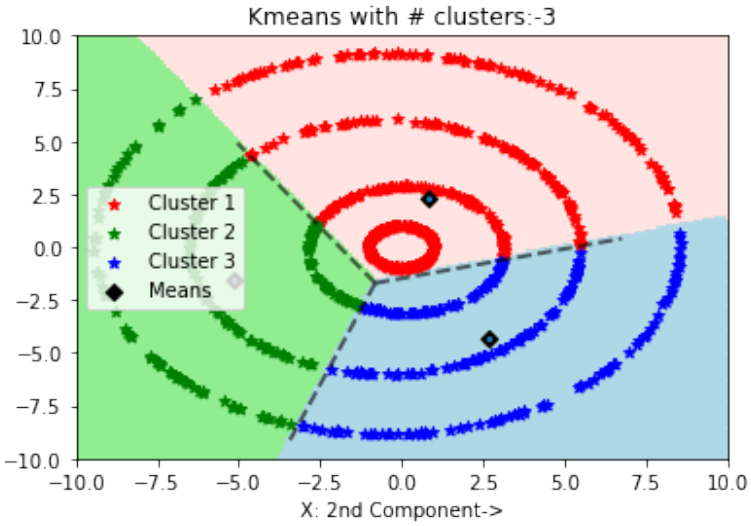


Figure 13: Kmeans 3 Clusters

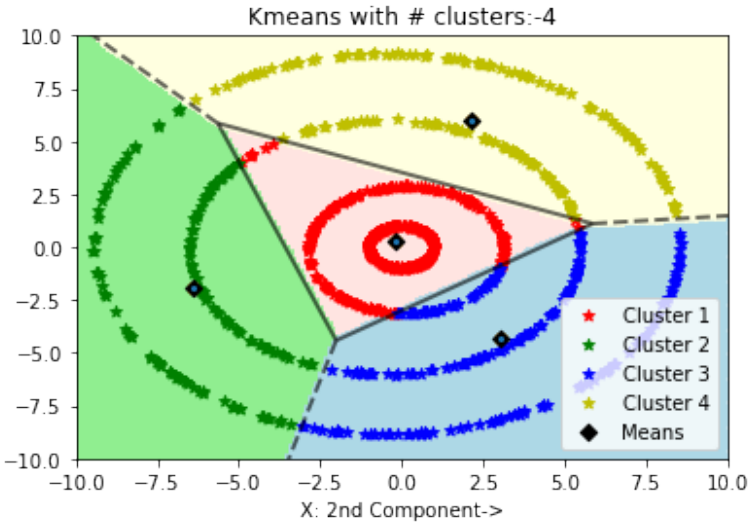


Figure 14: Kmeans 4 Clusters

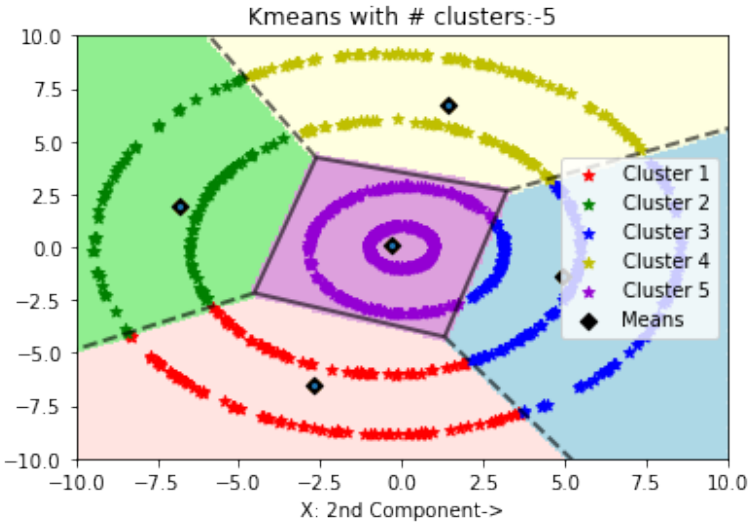
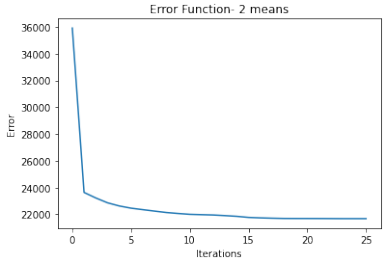
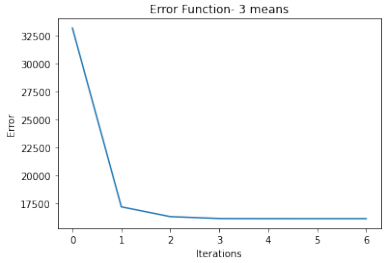


Figure 15: Kmeans 5 Clusters

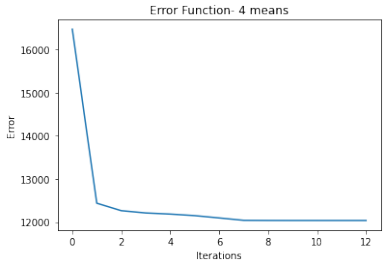
Now, looking at the error plots,



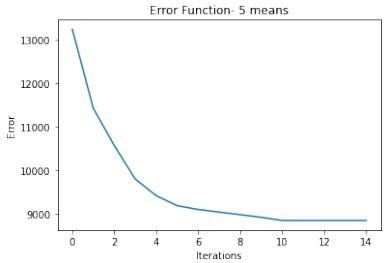
(a) Error Kmeans 2 clusters



(b) Error Kmeans 3 clusters



(c) Error Kmeans 4 clusters



(d) Error Kmeans 5 clusters

The error obtained after convergence for k=2,3,4 and 5 respectively are 21678.99850693509, 16172.47977918421, 12035.904320472775, 8857.12464461779.

The clustering regardless of the number of means is not performing well on the data-set, hence Spectral Clustering will be focused on.

4.4 Q2) iii)Spectral relaxation of K-Means using Kernel PCA

As seen above, Kmeans does not produce natural clusters. hence to combat this kernels are employed because the clusters are convex, this prevents the formation of natural clusters in the original dimension of the data. Hence we need to map the data to a higher dimension using a kernel function. The kernel function must sufficiently bring out the differences in the data by mapping it to an appropriate dimension.

The process of Kernel relaxation of K-means using Kernel PCA:-

- Given the data-set X, generate the K matrix using the chosen kernel function
- Compute H^* which is the matrix constructed by the top k eigenvectors of the kernel matrix, K
- Normalize the rows of H^* and run standard Lloyd's algorithm by treating each row as new data

In view of this first polynomial kernels were tried out, however as noted from the figure, the performance of the clustering is poor and the clusters produce don't reflect the natural truth

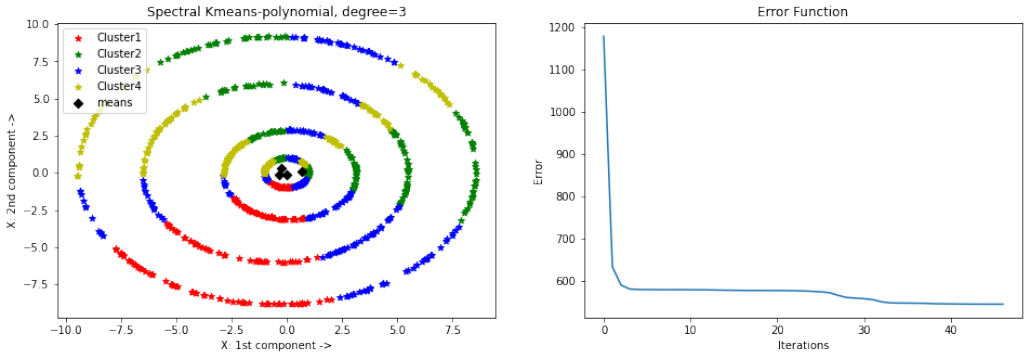


Figure 17: Spectral Clustering Polynomial Kernel, degree=3

Hence the Radial Basis Function was used. Various values of sigma are tried out. The following clusters with errors are observed:-

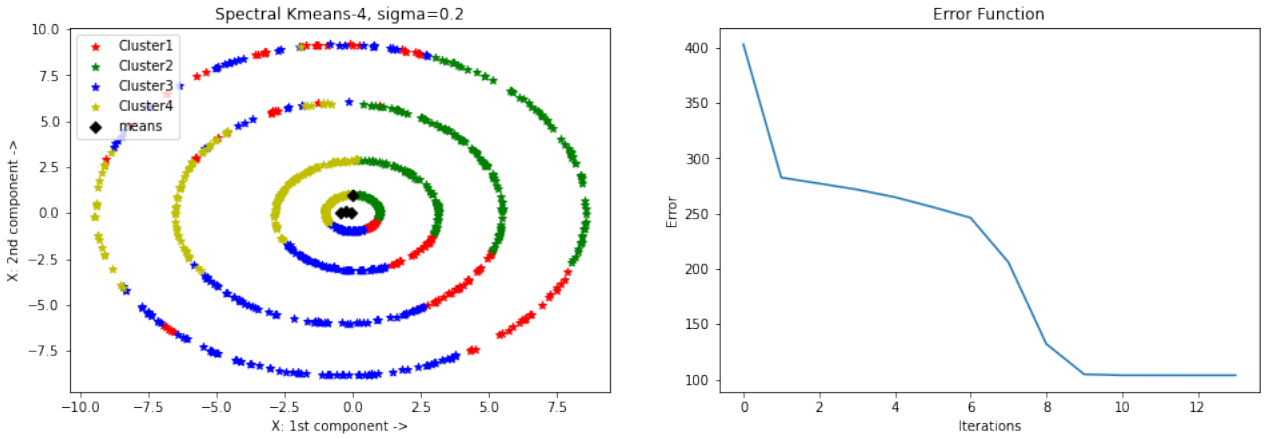


Figure 18: Spectral Clustering RBF Kernel, σ=0.2

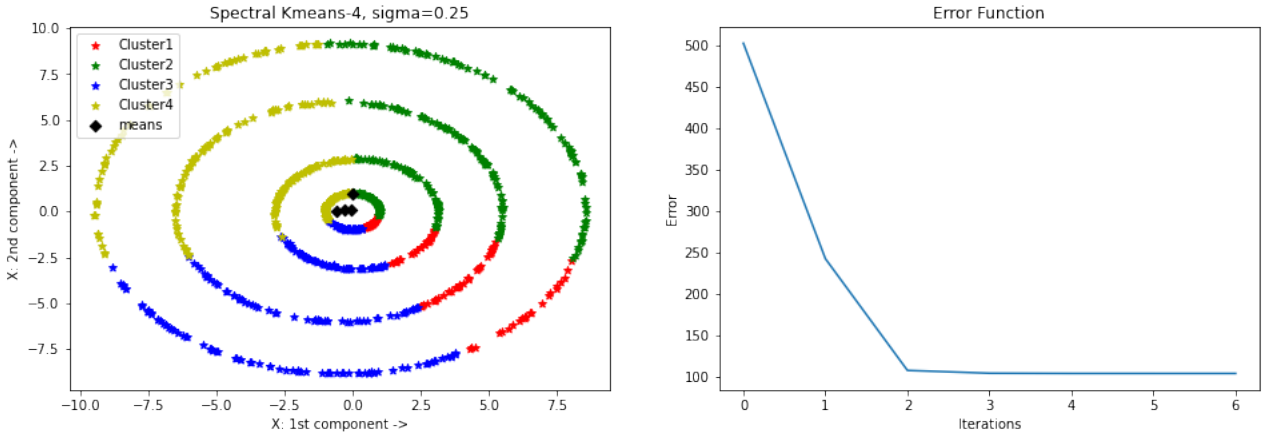


Figure 19: Spectral Clustering RBF Kernel, σ=0.25

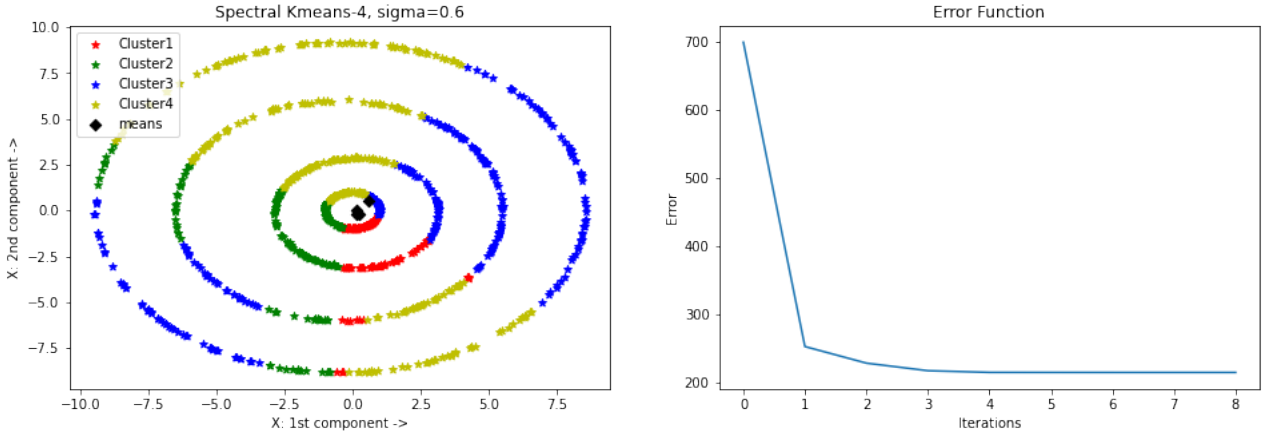


Figure 20: Spectral Clustering RBF Kernel, σ=0.6

For any σ values, we note the algorithm still does not produce natural kernels, hence we define our own kernel .

Look at the following kernel function:- $f(x, y) = [norm(x) * norm(y) + 1]^2$, where norm is the distance from origin.
This kernel satisfies the Mercer's Theorem, as it is symmetric and positive semi-definite.

This was what was seen in the graphs:-

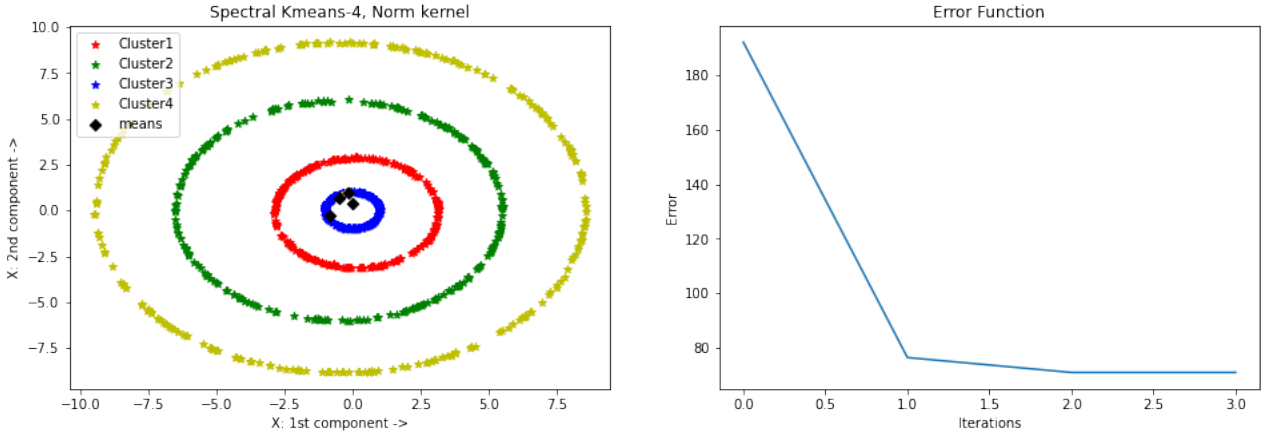


Figure 21: Custom Kernel:- Trial 1

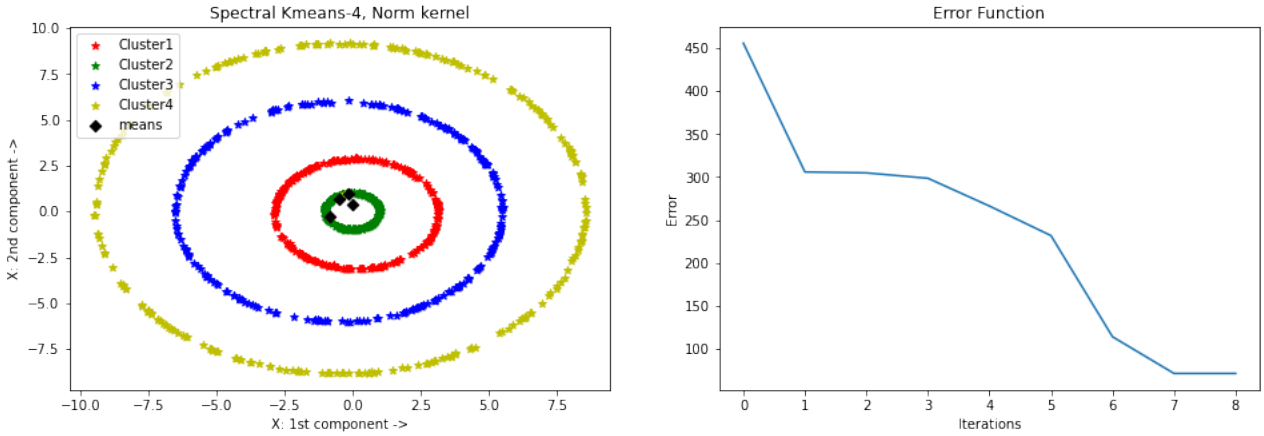


Figure 22: Custom Kernel:- Trial 2

[H]

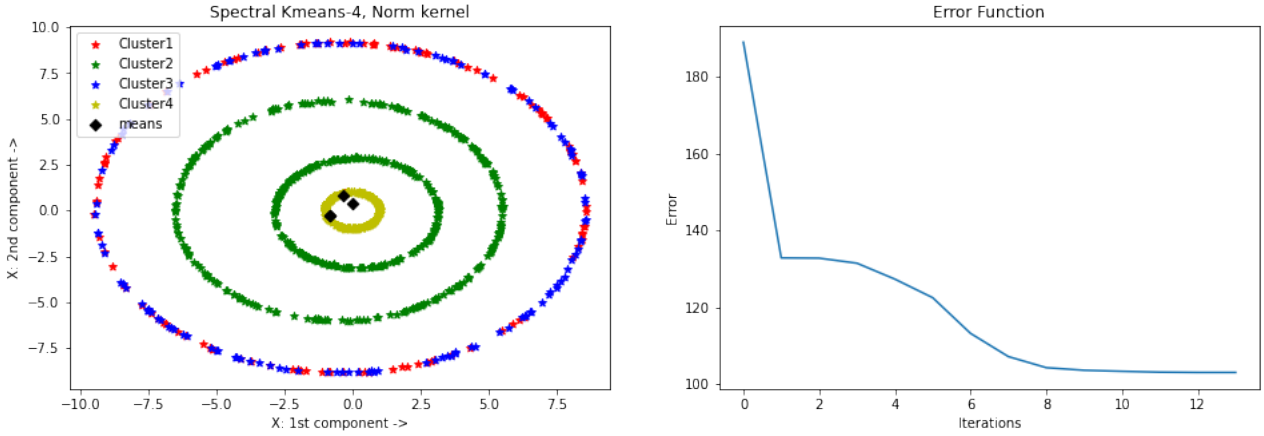


Figure 23: Custom Kernel:- Trial 3

The error defined as $Error = \left(\frac{\sum_{i=1}^{n} \|\phi(\mathbf{x_i}) - \mu_{\mathbf{z_i}}\|^2}{n}\right)$

For the custom kernel we get the error as approximately 70.97539259569791. Hence we note that the custom kernel of the norm, which has a radial length property, is the best kernel. Note the dependence on the random

initialization. So, Sometimes the clusters formed are not natural. This is because there is no initialization method like Kmeans++ for spectral clustering, hence there is no minimum guarantee for Spectral clustering algorithm, making it sensitive to intial conditions.
However, most of the time the custom kernel performs well!

4.4.1 Why this kernel?

As is evident from the data, the clusters are roughly circular around the origin, with their 'radius' effecting their cluster assignment in the natural case. Hence, using the norm makes sense.
Due to the circular nature of the data, a ower of two was tried out, and natural clusters formed

4.5 Kernel - Assign to argmax v_j^i

Instead of mapping eigenvectors to cluster assignments, we use the argmax of the eigenvector of the Kernel matrix associated with the j-th largest , v_j^i. Effectively, the data-point is assigned to the cluster which corresponds to the the eigen-vector to which the data-point contributes the largest component towards. This is what is was observed

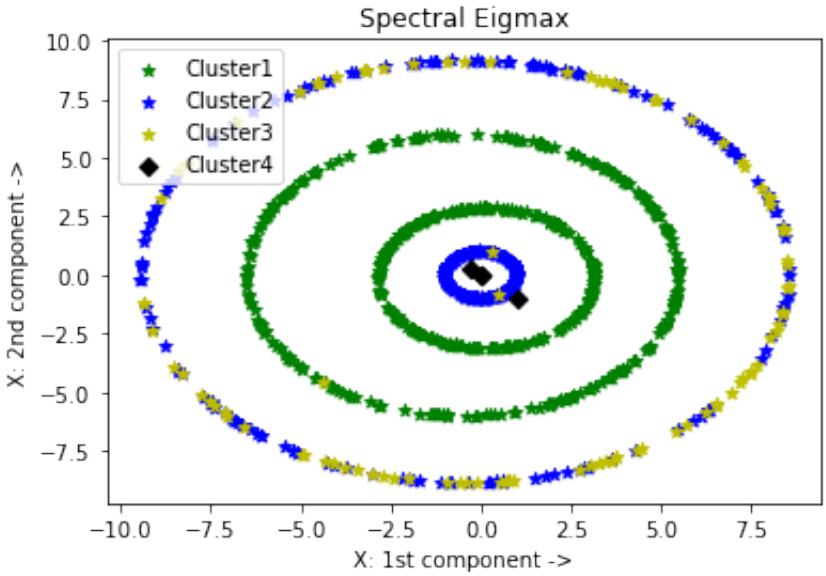


Figure 24: Eigmax of custom kernel

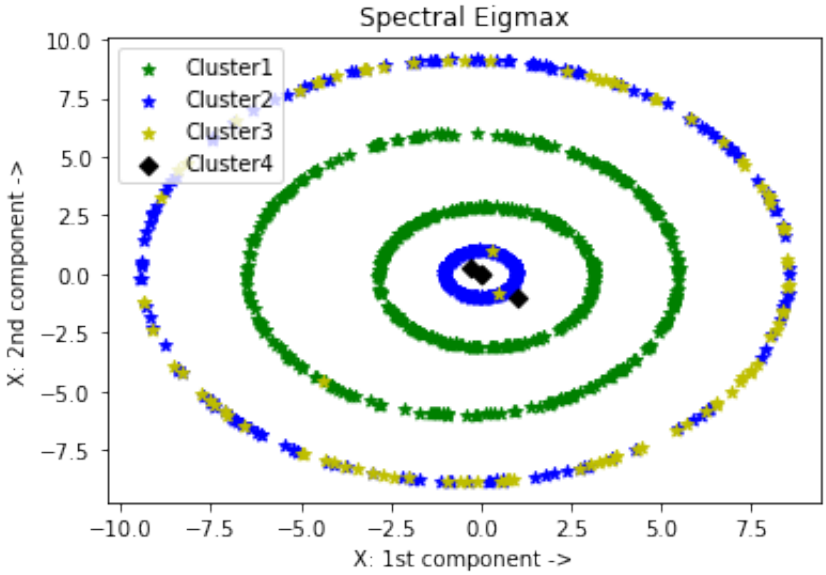


Figure 25: Eigmax of custom kernel

We notice the clustering is not accurate.The error is also higher:- around 250.4
Note the formation of only three clusters, even when 4 means are initialized, this means a cluster is empty! Hence doing Kmeans on the kernel relaxation is important

5 Conclusion:-

K-means (aka Lloyd's) gives us clusters that do not capture the nature of the data,regardless of tuning the number of clusters. This is due to the non-convexity of the data set, hence we kernelize, and relax $ZL^{1/2}$ to H using kernel relaxation and run spectral clustering on the data. We try polynomial, which performs poorly. Hence we try the radial basis functions, which still performs poorly. So we define a custom kernel, it works very well and captures the nature of the clusters perfectly.
When taking the argmax of the eigenvector of the Kernel matrix associated with the j-th largest , v_j^i, we note that the clustering quality is hit significantly.

THANK YOU