CS5691 Pattern recognition and machine learning Assignment 1 Solutions |

Kalyani Umesh Burande | ED21S014 | 02/03/2022

(1) You are given a data-set with 1000 data points each in R2.

i. Write a piece of code to run the PCA algorithm on this data-set. How much of the variance in the data-set is explained by each of the principal components?

Solution:

Please refer source code with file name: Q1_i_amount_of_variance_020322.py

- **Result from source code:**

```
PCA without centering dataset-->

Mean of dataset = [4.075e-07 2.227e-07]
Covariance matrix:
 [[14.76615576  0.80885904]
 [ 0.80885904 16.85536339]]

Printing Eigen vectors and Eigen values of covariance matrix-->

Eigen vector with largest eigen value or First Principal Component:
 w1 =  [-0.323516  -0.9462227] with eigen value lambda 1 = 17.131914402444487

Eigen vector with second largest eigen value or Second Principal Component:
 w2 =  [-0.9462227  0.323516 ] with eigen value lambda 2 = 14.489604749330738

Amount of variance in dataset explained by w1 =  54.1780245288522  %
Amount of variance in dataset explained by w2 =  45.8219754711478  %
Total variance explained by w1 and w2 in %: 100.0
```

- **Explanation:**

- Since the dimension of dataset is 2, we obtain 2x2 covariance with 2 eigen vectors.
- We observe that the first principal component of dataset corresponds to the eigen vector of the covariance matrix with the largest eigen value of covariance matrix.
- The second principal component of dataset corresponds to the eigen vector of the covariance matrix with the second largest eigen value of covariance matrix.
- PCA algorithm explains that the variance in the dataset as explained by the first principal component of the dataset is the maximum.
- The variance explained by the following principal components of dataset are lesser than the variance explained by first principal component of dataset.
- Here, w1 being the first principal component of the dataset (eigen vector of covariance matrix with largest eigen value), the projection of all the datapoints on this vector gives measure of the amount of variance explained by this w1 vector for given dataset.
- Hence, w1 being the first principal component of dataset contributes to about 54.178% of the total variance in the dataset.
- Also, w2 being the second principal component of dataset contributes to about 45.8219% of the total variance in the dataset.
- We notice, the sum of variance contributions by w1 and w2 amount to 100% implying that w1 and w2 vectors are the only two and sufficient vectors in given 2D space to represent the dataset in a compressed form while preserving the variance in dataset.

(1) You are given a data-set with 1000 data points each in R2.

ii. Study the effect of running PCA without centering the data-set. What are your observations? Does Centering help?

Solution:

Please refer source code with file name: Q1_ii_PCA_without_centering_020322.py

- **Result from source code:**

```
PCA without centering dataset-->

Mean of dataset = [4.075e-07 2.227e-07]
Covariance matrix:
 [[14.76615576  0.80885904]
 [ 0.80885904 16.85536339]]

Printing Eigen vectors and Eigen values of covariance matrix-->

Eigen vector with largest eigen value or First Principal Component:
 w1 =  [-0.323516  -0.9462227] with eigen value lambda 1 = 17.131914402444487

Eigen vector with second largest eigen value or Second Principal Component:
 w2 =  [-0.9462227  0.323516 ] with eigen value lambda 2 = 14.489604749330738

Amount of variance in dataset explained by w1 =  54.1780245288522  %
Amount of variance in dataset explained by w2 =  45.8219754711478  %
Total variance explained by w1 and w2 in %: 100.0
```

- **Explanation:**

- The mean of the dataset is [4.075e-07 2.227e-07] which is a very small value.
- Hence, the elements of covariance matrix are not affected much by not centering the dataset.
- Since the covariance matrix elements are not changed (as compared to covariance matrix in centered dataset), the eigen vectors, eigen values and hence the corresponding principal components are also the same as those obtained for centered dataset.
- Hence, for the given dataset, mean being of small value, not centering the dataset also yields the same principal components and their associated variances as obtained for centered dataset.

(1) You are given a data-set with 1000 data points each in R2.

iii. Write a piece of code to implement the Kernel PCA algorithm on this dataset.

Use the following kernels :

A. $\kappa(x, y) = (1 + x^T y)^d$ for $d = \{2, 3\}$

B. $\kappa(x, y) = \exp \frac{-(x-y)^T(x-y)}{2\sigma^2}$ for $\sigma = \{0.1, 0.2, \ldots, 1\}$

Plot the projection of each point in the dataset onto the top-2 components for each kernel. Use one plot for each kernel and in the case of (B), use a different plot for each value of sigma.

Solution:

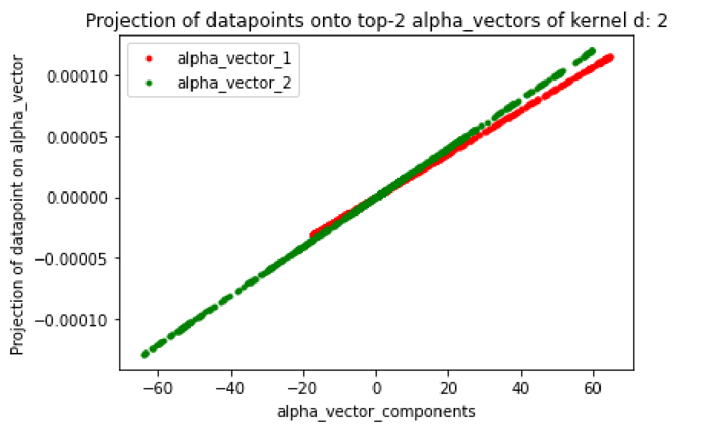
Please refer source code with file name: Q1_iii_Kernel_PCA_plots_020322.py

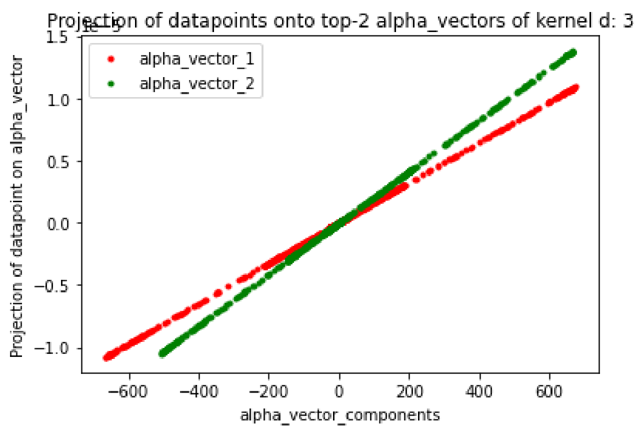
- **Result from source code:**

Kernel PCA on dataset for Polynomial and Radial Basis Kernels -->
Projections of all datapoints on top 2 components of each kernel is plotted on scatter plots.

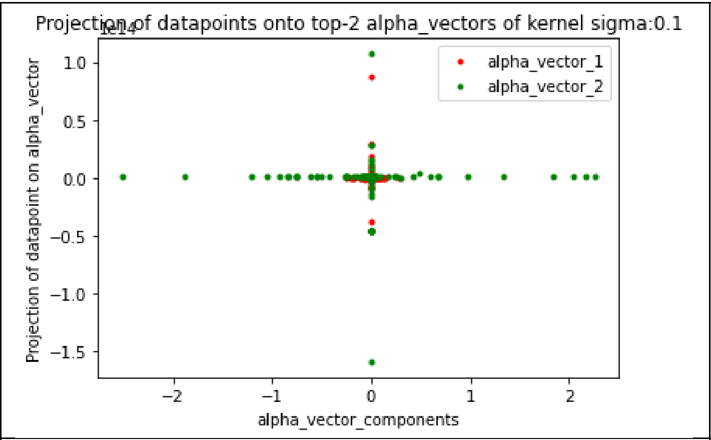
- **Plots:**

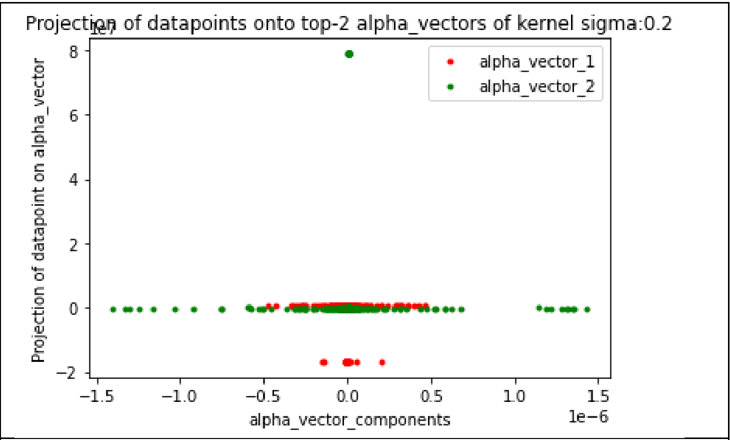
A. Polynomial kernel with d = 2 and d = 3

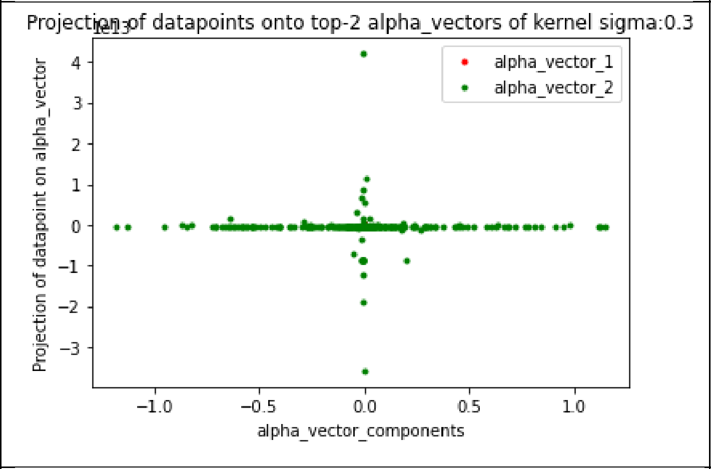


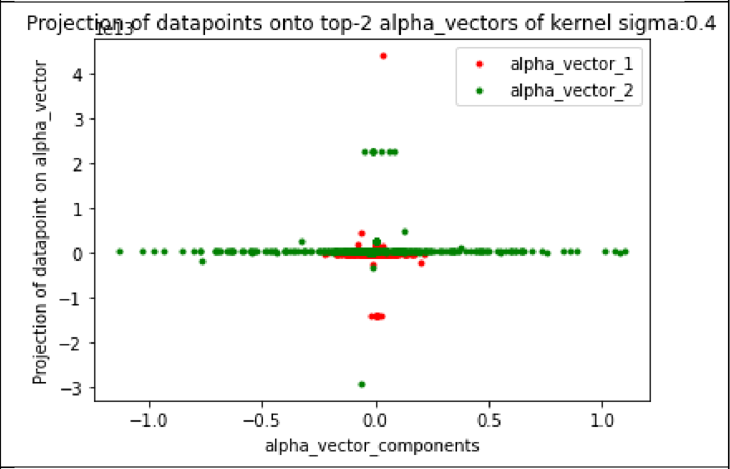


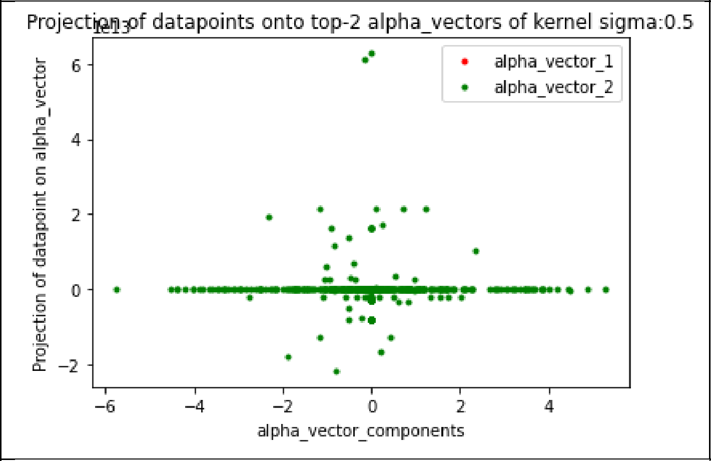
B. Radial Basis kernel with sigma = {0.1, 0.2, ..1}

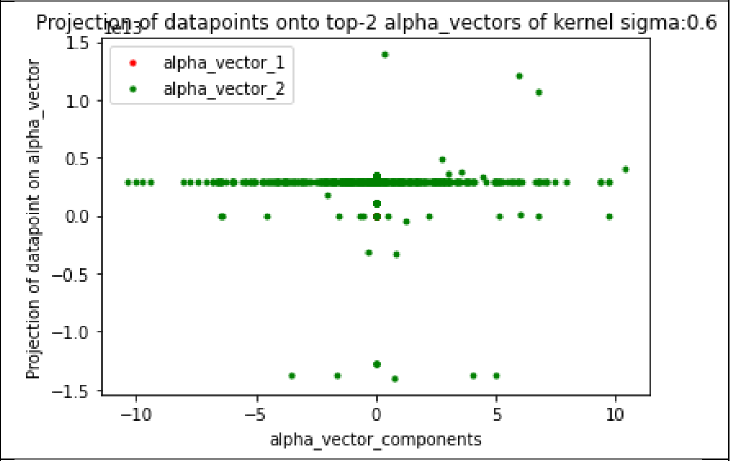


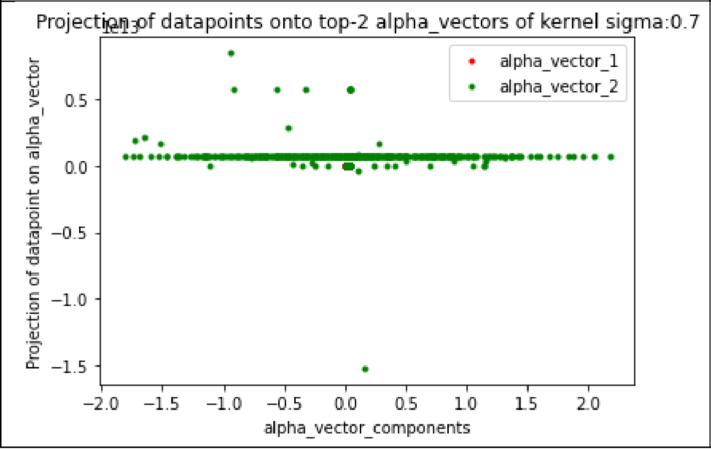


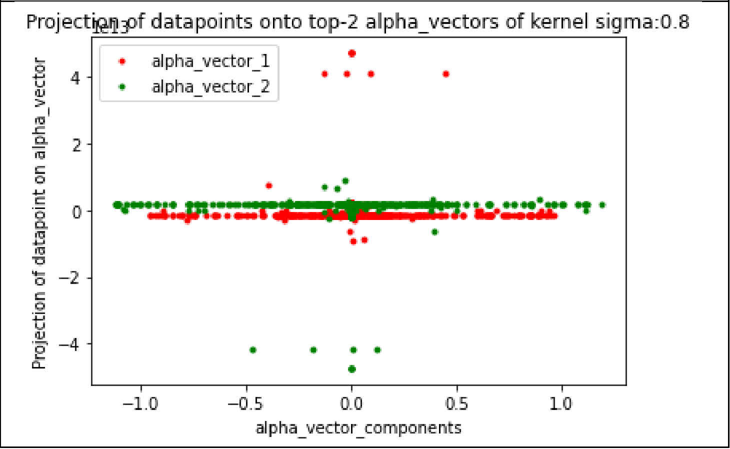


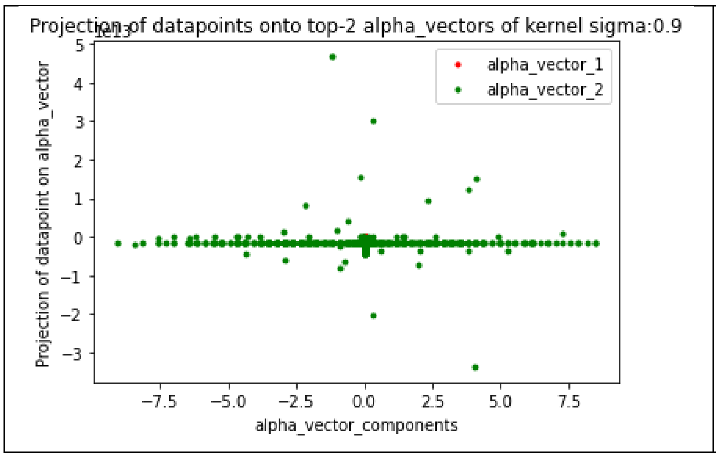


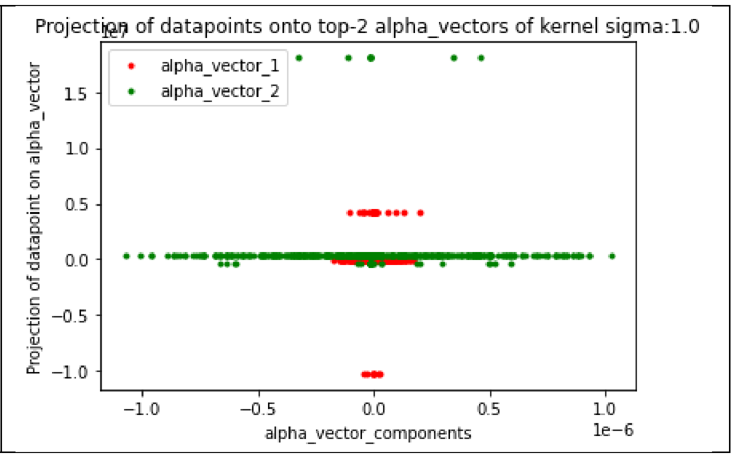












iv. Which Kernel do you think is best suited for this dataset and why?

- **Explanation:**

- We observe that the projections of datapoints on the top 2 vector components of the kernel matrix is distinctly observable for the polynomial kernel with d = 2,3 than for any of the RBF kernels.
- Further, the top 2 eigen vectors of kernel matrix with d = 3 seem to distinctly depict the variance in dataset.
- In all the plots for RBF with given sigma values (expect sigma = 0.8), projection of all datapoints along both the top 2 vector components of kernel matrix seem to overlap and hence the datapoints projections along individual alpha_vectors cannot be distinguished clearly.
- For RBF projection plot along alpha_vectors for sigma = 0.8, although most of the projections on each alpha_vetors seem to be separable but there are few projections that overlap on both the alpha_vactors. Hence, using this kernel, we will be unable to separate the datapoints distinctly in higher dimension.
- Hence, the best choice for kernel for this dataset will be a polynomial kernel with d= 3 since the projections of all the datapoints along both the alpha_vectors appear distinctly on the projection plots.

(2)You are given a data-set with 1000 data points each in R2.

i. Write a piece of code to run the algorithm studied in class for the K-means problem with k = 4 . Try 5 different random initialization and plot the error function w.r.t iterations in each case. In each case, plot the clusters obtained in different colors.

Solution:

Part 1: Plotting error function vs. iteration count for 5 different initial cluster coordinates initialization.

Please refer source code with file name: Q2_i_K_Means_error_function_plots_020322.py

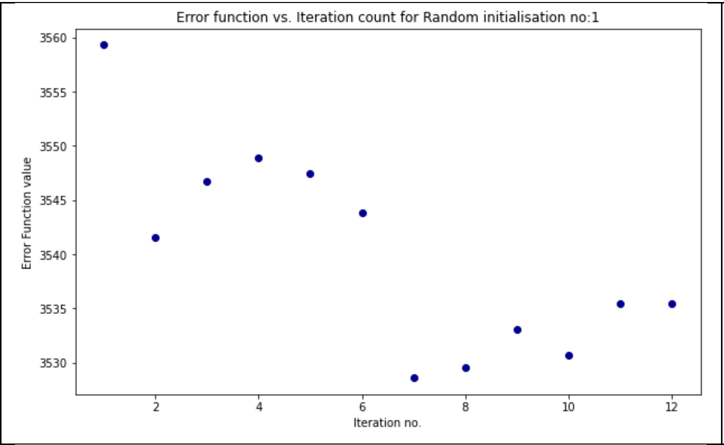
- **Result from source code:**

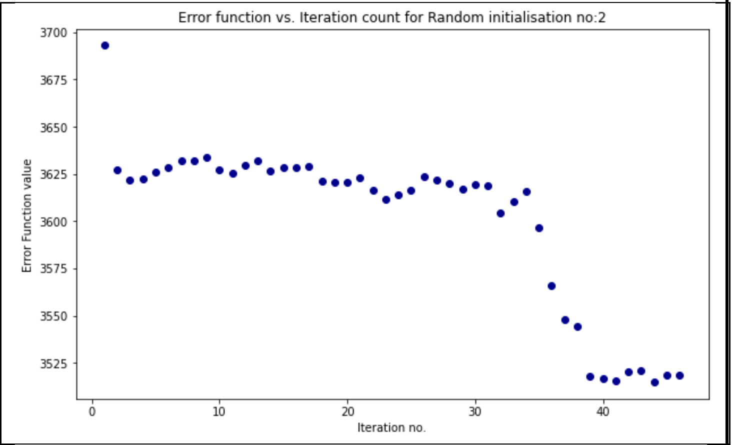
For k=4 plotting the error function w.r.t iterations in each case for 5 different initialisations.

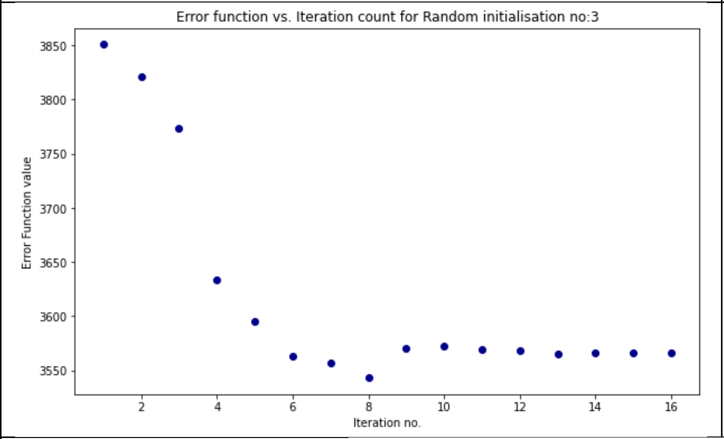
```
Random centroid coordinates initialisation round no. : 1
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[5.756580849468129, 4.813552433432495], [-1.8983669644045245, -4.173716460623965], [-
0.26022289472224713, -1.544060131828], [4.66169441726251, -3.3741804534993802]]
iter_no when no change in centroid coordinates: 12
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5425700318746594

Final K-means Centroid Coordinates for k = 4 : [[1.0156661506849314, 6.710788965517245], [-
4.345007794326241, -5.472855785714283], [-1.1127009084362143, 0.4966394335390945],
[4.427893788546253, -1.9884811982378858]]

Random centroid coordinates initialisation round no. : 2
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-7.067414569207978, 6.425003655250009], [4.3000515060271365, -4.243010901653608], [-
0.5462946095760017, -0.7416255160066587], [2.2739948838594586, 5.367676575269684]]
iter_no when no change in centroid coordinates: 46
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5253211232369814

Final K-means Centroid Coordinates for k = 4 : [[-3.5636105255474435, 5.7831857352941185], [-
1.0583282748815168, -5.669682857142857], [-0.9039476957040571, 0.08973698019093086],
[4.679297854077253, 1.559512738197425]]

Random centroid coordinates initialisation round no. : 3
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[7.772356967455785, 8.233014779949233], [-8.472754023169971, -7.30812326478395],
[5.595427488456922, 4.417630727132849], [2.6015659152111077, -3.287308907985585]]
iter_no when no change in centroid coordinates: 16
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5737537419467182
```

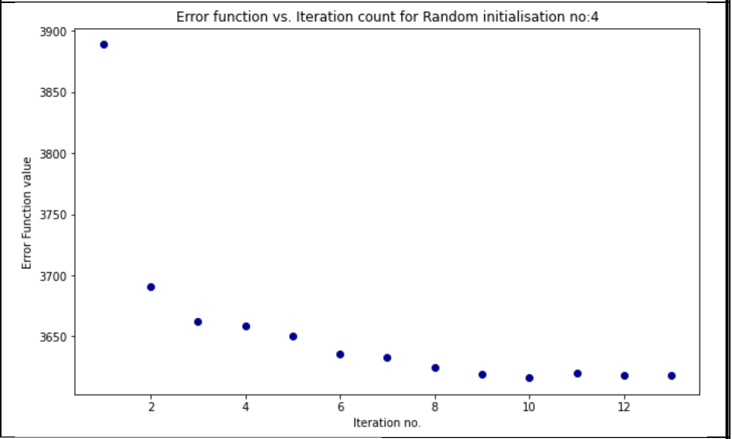
```
Final K-means Centroid Coordinates for k = 4 : [[4.140592404371582, 3.926456617486338], [-
4.046317753731342, -5.833121052631576], [-3.5858763268292693, 3.564682843627453],
[1.0869941914225938, -1.4081289506276151]]
Random centroid coordinates initialisation round no. : 4
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-5.196339852406896, 0.9645329641998561], [-2.812524714558415, 2.0395167590865846],
[1.8067215523435092, -7.656477668178359], [-9.256279488523194, 6.245548193291567]]
iter_no when no change in centroid coordinates: 13
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.625405095862684

Final K-means Centroid Coordinates for k = 4 : [[-4.402206505376343, -3.7192917736559137],
[1.0538704630390143, 1.1091417117043123], [3.4571089347826076, -4.127362956284155], [-
2.311413580419581, 6.3647169014084515]]

Random centroid coordinates initialisation round no. : 5
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-5.230953475297064, -6.978662256332194], [-2.341090109329796, -6.045673380665233], [-
8.292804114478766, -1.604267101935391], [7.084627055685747, 5.578906488558314]]
iter_no when no change in centroid coordinates: 10
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.58169399311765

Final K-means Centroid Coordinates for k = 4 : [[-4.046317753731342, -5.833121052631576],
[1.1020382536534443, -1.4092766981210856], [-3.6225624384236466, 3.5682059118226617],
[4.0745154239130414, 3.9527265628415296]]

K means algorithm complete and cluster coordinates are found.
```

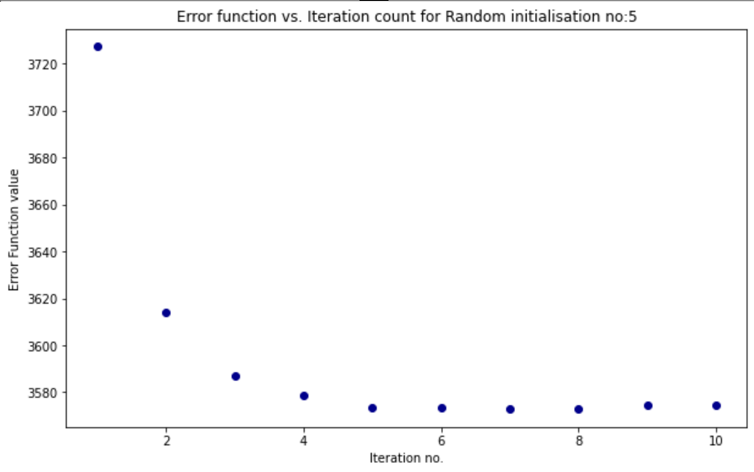
- **Plots:**











- **Explanation:**

- The K-means algorithm is bound to converge since the assignment of datapoint to new cluster happens every iteration as per the objective function to minimize the distance between all datapoints to the respective assigned cluster centroids over all possible initial cluster assignments; since the objective function is a monotonically decreasing function.
- The no. of iterations taken by the algorithm to minimize the objective function varies strongly as per the initial centroids assigned to each datapoint.
- The cluster assignment obtained from this algorithm will be a valid assignment but it may or may not be guaranteed to be optimum cluster assignment due to the random nature of initial assignment of cluster to datapoints.

(2)You are given a data-set with 1000 data points each in R2.

i. Write a piece of code to run the algorithm studied in class for the K-means problem with k = 4 . Try 5 different random initialization and plot the error function w.r.t iterations in each case. In each case, plot the clusters obtained in different colors.

Solution:

Part 2: Plotting colored clusters for 5 different initial cluster coordinates initialization.

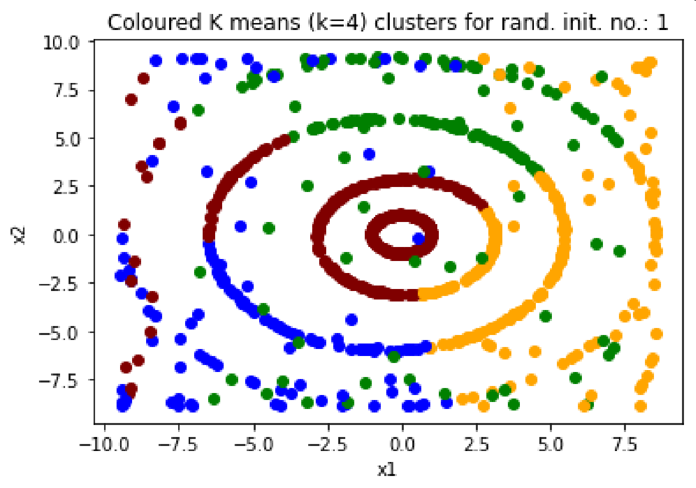
Please refer source code with file name: Q2_i_K_Means_coloured_clusters_020322.py

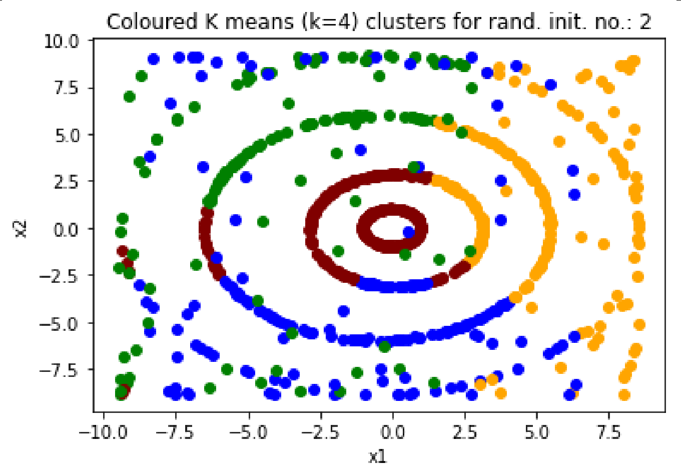
- **Result from source code:**

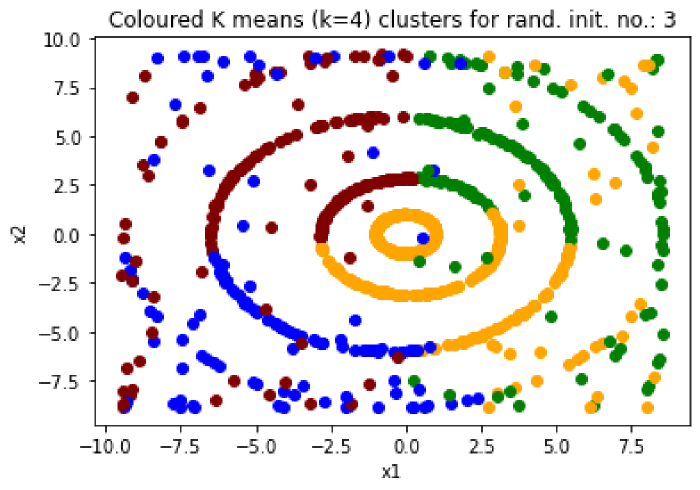
```
For k=4 plotting the colourful clusters for 5 different initialisations.

Random centroid coordinates initialisation round no. : 1
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[5.756580849468129, 4.813552433432495], [-1.8983669644045245, -4.173716460623965], [-
0.26022289472224713, -1.544060131828], [4.66169441726251, -3.3741804534993802]]
iter_no when no change in centroid coordinates: 12
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5425700318746594

Centroid Coordinates for k = 4 : [[1.0156661506849314, 6.710788965517245], [-4.345007794326241, -
5.472855785714283], [-1.1127009084362143, 0.4966394335390945], [4.427893788546253, -
1.9884811982378858]]

Random centroid coordinates initialisation round no. : 2
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-7.067414569207978, 6.425003655250009], [4.3000515060271365, -4.243010901653608], [-
0.5462946095760017, -0.7416255160066587], [2.2739948838594586, 5.367676575269684]]
iter_no when no change in centroid coordinates: 46
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5253211232369814

Centroid Coordinates for k = 4 : [[-3.5636105255474435, 5.7831857352941185], [-1.0583282748815168, -
5.669682857142857], [-0.9039476957040571, 0.08973698019093086], [4.679297854077253,
1.559512738197425]]

Random centroid coordinates initialisation round no. : 3
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[7.772356967455785, 8.233014779949233], [-8.472754023169971, -7.30812326478395],
[5.595427488456922, 4.417630727132849], [2.6015659152111077, -3.287308907985585]]
iter_no when no change in centroid coordinates: 16
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.5737537419467182
```

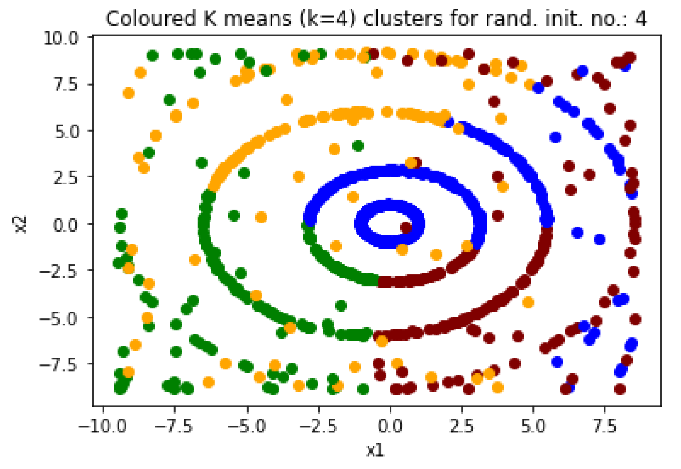
```
Centroid Coordinates for k = 4 : [[4.140592404371582, 3.926456617486338], [-4.046317753731342, -
5.833121052631576], [-3.5858763268292693, 3.564682843627453], [1.0869941914225938, -
1.4081289506276151]]

Random centroid coordinates initialisation round no. : 4
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-5.196339852406896, 0.9645329641998561], [-2.812524714558415, 2.0395167590865846],
[1.8067215523435092, -7.656477668178359], [-9.256279488523194, 6.245548193291567]]
iter_no when no change in centroid coordinates: 13
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.625405095862684

Centroid Coordinates for k = 4 : [[-4.402206505376343, -3.7192917736559137], [1.0538704630390143,
1.1091417117043123], [3.4571089347826076, -4.127362956284155], [-2.311413580419581,
6.3647169014084515]]

Random centroid coordinates initialisation round no. : 5
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']

k_means_centroid_cordinates random intiialisations at start:
 [[-5.230953475297064, -6.978662256332194], [-2.341090109329796, -6.045673380665233], [-
8.292804114478766, -1.604267101935391], [7.084627055685747, 5.578906488558314]]
iter_no when no change in centroid coordinates: 10
K means coordinates found, breaking from while loop
Cost for k = 4 : 3.58169399311765

Centroid Coordinates for k = 4 : [[-4.046317753731342, -5.833121052631576], [1.1020382536534443, -
1.4092766981210856], [-3.6225624384236466, 3.5682059118226617], [4.0745154239130414,
3.9527265628415296]]

K means algorithm complete and cluster coordinates are found.
```

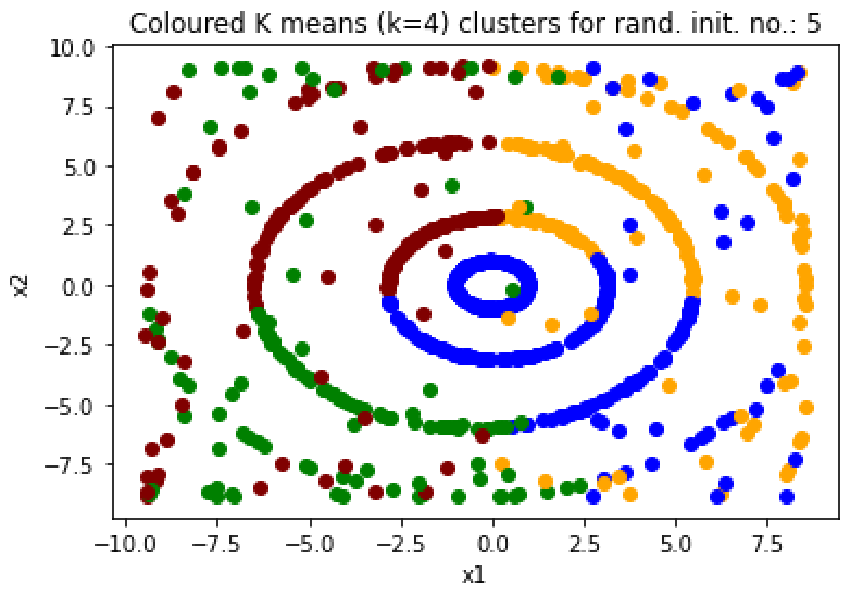
- **Plots:**











- **Explanation:**

- Based on the initial coordinates assigned to each cluster, the final cluster assignment for each datapoint changes.
- The final cluster assignment is a valid assignment but may or may not be the most optimum assignment due to random initialization of cluster coordinates.

(2)You are given a data-set with 1000 data points each in R2.

ii. Fix a random initialization. For K = {2; 3; 4; 5}, obtain cluster centers according to K-means algorithm using the fixed initialization. For each value of K, plot the Voronoi regions associated to each cluster center. (You can assume the minimum and maximum value in the data-set to be the range for each component of R2).

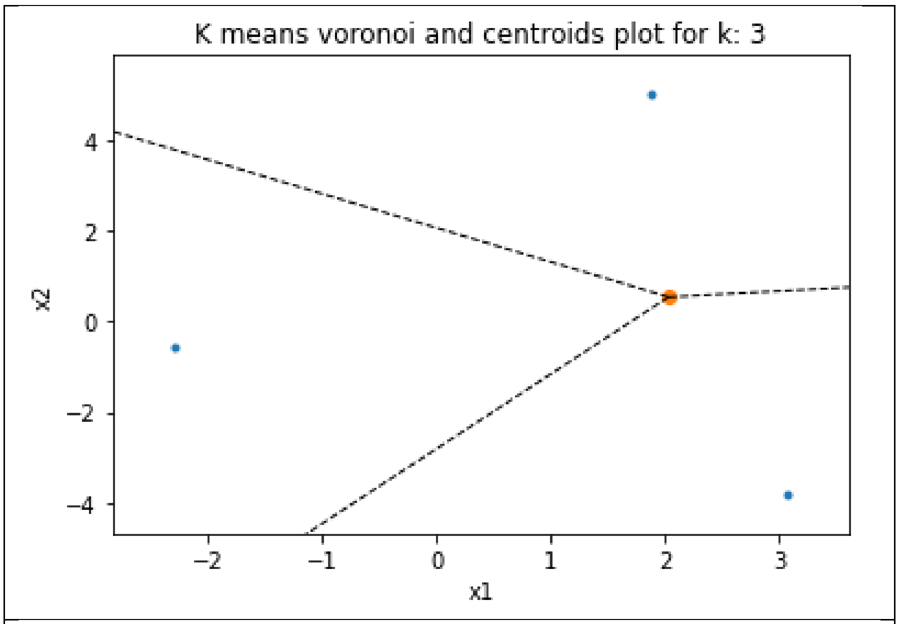
Solution:

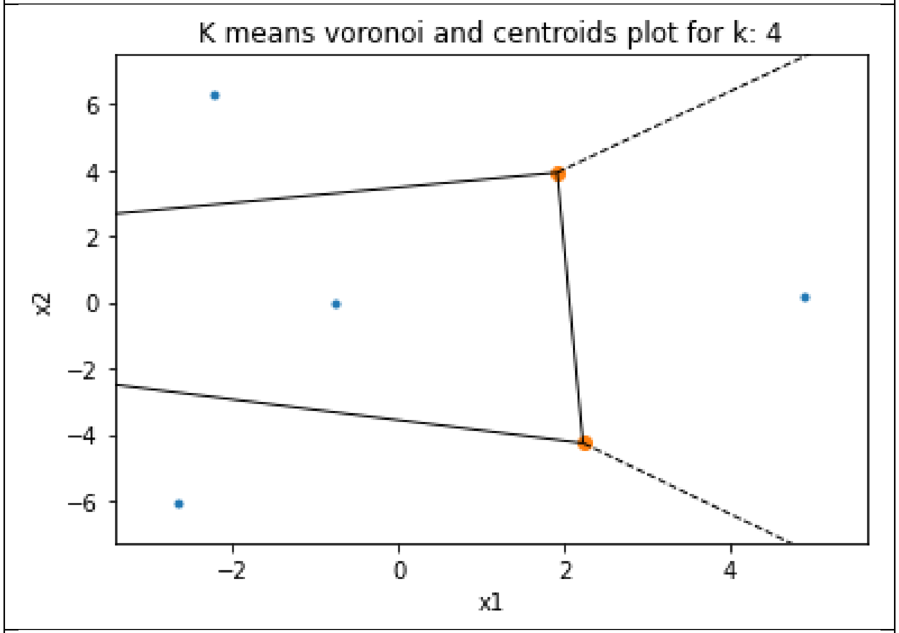
Please refer source code with file name: Q2_ii_K_Means_Voronoi_regions_plots_020322.py

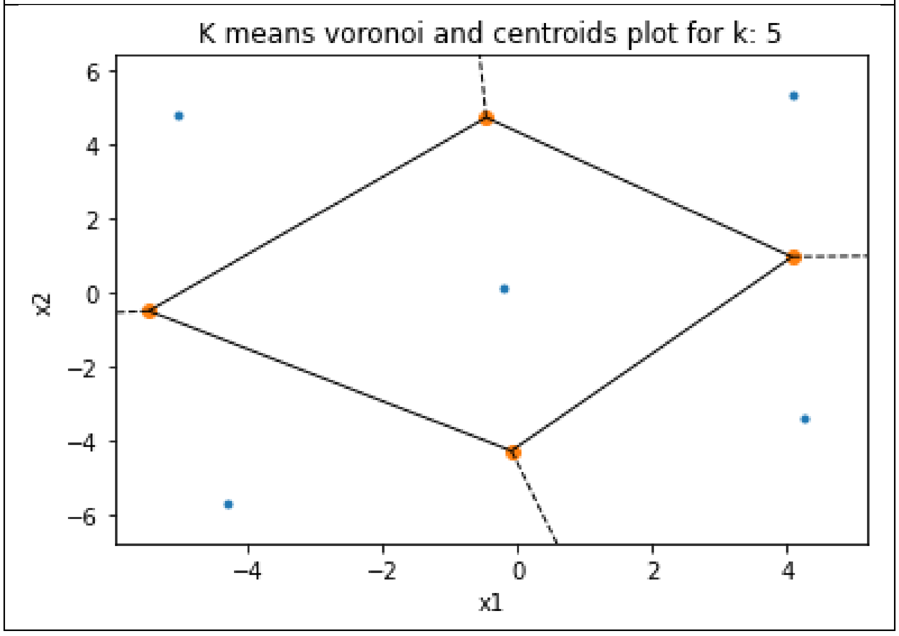
- **Result from source code:**

```
For k={2; 3; 4; 5} plotting Voronoi regions associated to each cluster center for fixed initialisation.

Number of classes = K = 2
k_means_centroid_cordinates at start: [[6.86220537154993, 3.5222883647252914], [4.349440424537535, 7.454814750026806]]
iter_no when no change in centroid coordinates: 18
K means coordinates found, breaking from while loop
Centroid Coordinates for k = 2 : [[-0.17612262884333818, -2.0978062570381213], [0.37947054574132494, 4.517563259493672]]

Number of classes = K = 3
k_means_centroid_cordinates at start: [[6.86220537154993, 3.5222883647252914], [4.349440424537535, 7.454814750026806], [-4.801486384625333, 2.6123143309081804]]
iter_no when no change in centroid coordinates: 23
K means coordinates found, breaking from while loop
Centroid Coordinates for k = 3 : [[3.0721759502074675, -3.8048026333333342], [1.8811809291666672, 5.003905020920502], [-2.2964882842003846, -0.550933613294798]]

Number of classes = K = 4
k_means_centroid_cordinates at start: [[6.86220537154993, 3.5222883647252914], [4.349440424537535, 7.454814750026806], [-4.801486384625333, 2.6123143309081804], [6.84967171852394, 6.869770332696232]]
iter_no when no change in centroid coordinates: 39
K means coordinates found, breaking from while loop
Centroid Coordinates for k = 4 : [[-2.6642628597560973, -6.046244171779142], [-2.2337890198675496, 6.254868000000001], [-0.7684418494505494, -0.00814185780219783], [4.8864465217391295, 0.20807942608695 65]]

Number of classes = K = 5
k_means_centroid_cordinates at start: [[6.86220537154993, 3.5222883647252914], [4.349440424537535, 7.454814750026806], [-4.801486384625333, 2.6123143309081804], [6.84967171852394, 6.869770332696232], [0.8534949569755703, -5.786236153338187]]
iter_no when no change in centroid coordinates: 21
K means coordinates found, breaking from while loop
Centroid Coordinates for k = 5 : [[4.2501576047904175, -3.378873766467067], [-5.051982719298245, 4.811431149122807], [-0.2019555684782609, 0.1599096754347826], [4.0799194375, 5.3276771653543324], [-4.29908403816794, -5.673527384615382]]

K means algorithm complete and voronoi region per cluster is plotted.
```

- **Plots:**


K means voronoi and centroids plot for k: 3
x2
x1
4
2
0
−2
−4
−2
−1
0
1
2
3


K means voronoi and centroids plot for k: 4
x2
x1
6
4
2
0
−2
−4
−6
−2
0
2
4


K means voronoi and centroids plot for k: 5
x2
x1
6
4
2
0
−2
−4
−6
−4
−2
0
2
4

(2)You are given a data-set with 1000 data points each in R2.

iii. Run the spectral clustering algorithm (spectral relaxation of K-means using Kernel PCA) k = 4. Choose an appropriate kernel for this data-set and plot the clusters obtained in different colors. Explain your choice of kernel based on the output you obtain.

Solution:

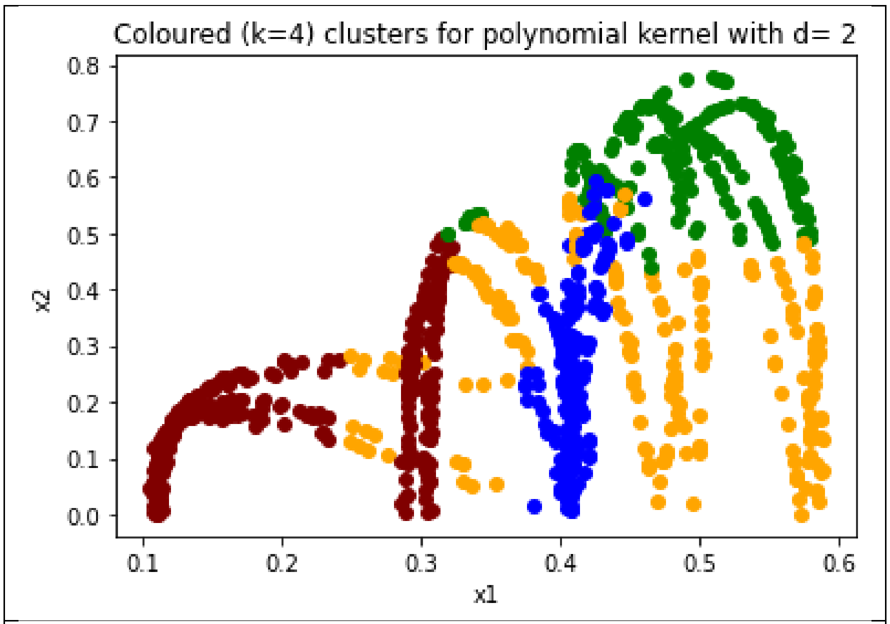
Please refer source code with file name:
Q2_iii_Spectral_Clustering_Kernel_Selection_MAIN_file_020322.py

Note: 1) Q2_iii_Spectral_Clustering_Kernel_Selection_import_file_020322.py file is already imported into above main file as a module and is kept in the same folder as the main file. 2) Kindly execute only the Q2_iii_Spectral_Clustering_Kernel_Selection_MAIN_file_020322.py

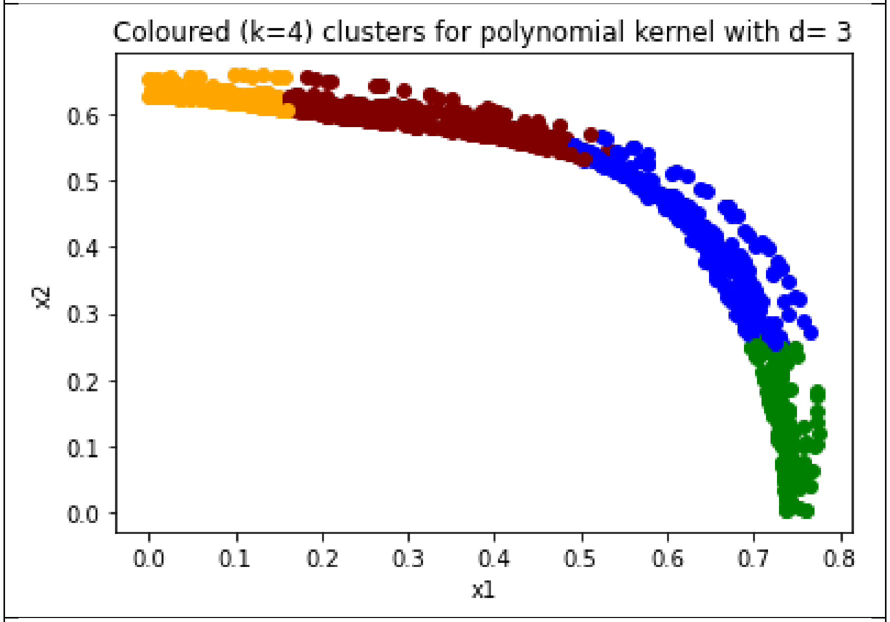
- **Result from source code:**

```
Reloaded modules: Q2_iii_Spectral_Clustering_Kernel_Selection_import_file_020322
Plotting coloured clusters (k=4) for fixed rand. init. for kernels and choosing best kernel for dataset
Random centroid coordinates initialisation round no. : 1
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']
iter_no when no change in centroid coordinates: 14
K means coordinates found, breaking from while loop
Cost for k = 4 : 0.19204959146193398

Centroid Coordinates for k = 4 : [[0.47982477727610545, 0.6308541431568679, 0.22338293742893378,
0.5292030826169819], [0.4094610065571004, 0.24671328600964473, 0.834647928310195,
0.1746832548211348], [0.20398483160515032, 0.19359361656407834, 0.14815240744904318,
0.9300585676962466], [0.4526059491014522, 0.30379996339159454, 0.48954892336550665,
0.6387967269945762]]

Random centroid coordinates initialisation round no. : 1
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']
iter_no when no change in centroid coordinates: 7
K means coordinates found, breaking from while loop
Cost for k = 4 : 0.16575618336281064

Centroid Coordinates for k = 4 : [[0.7350514693799003, 0.1266411772775784, 0.2461394743305102,
0.59577145390353], [0.6268761073113401, 0.4245927270943095, 0.579362206935734,
0.2423233570227152], [0.32121115728897776, 0.5919656175397435, 0.271305318836024,
0.6624410405012492], [0.07590017892612322, 0.6280146168418799, 0.6941210567721088,
0.2876606596264178]]

Random centroid coordinates initialisation round no. : 1
Number of classes = K = 4
K-means class labels are: ['0', '1', '2', '3']
iter_no when no change in centroid coordinates: 3
K means coordinates found, breaking from while loop
Cost for k = 4 : 0.25763667378873506

Centroid Coordinates for k = 4 : [[0.9118998661260861, 0.17997491729388226, 0.014105513987241868,
0.01721982676830494], [0.54034480869238, 0.4098907725856137, 0.7153612909975652,
0.017049813269830662], [0.14207141845777735, 0.37582276895002215, 0.38160254229586,
0.7863471867069178], [0.19920833397431773, 0.7429214596051319, 0.5425990441979377,
0.1456716779707883]]

Spectral clustering algorithm implemented and coloured clusters for kernels are plotted.
```

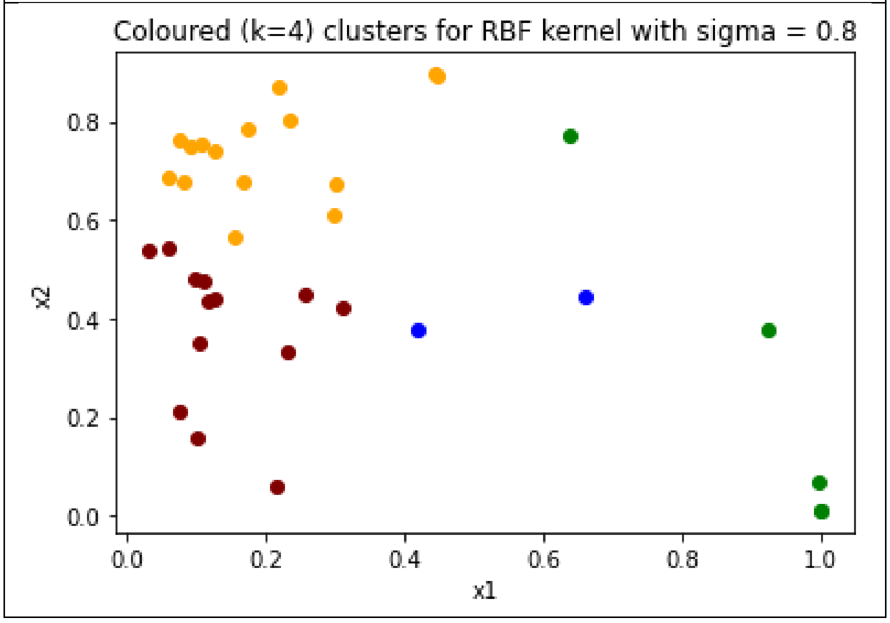
- **Plots:**



Coloured (k=4) clusters for polynomial kernel with d= 2
x2
x1



Coloured (k=4) clusters for polynomial kernel with d= 3
x2
x1



Coloured (k=4) clusters for RBF kernel with sigma = 0.8
x2
x1

- **Explanation:**

- Three kernels were employed and implemented: polynomial kernel with d=2, polynomial kernel with d=3, Radial basis kernel with sigma = 0.8
- We observe that the Polynomial kernel with d=3 clearly categorizes the datapoints into 4 different clusters. Hence the polynomial kernel with d = 3 is the best kernel to choose.
- When we use the RBF kernel with sigma = 0.8, we notice that some datapoints are lost in higher dimension. Also, the datapoints are not clustered uniquely. Hence, this kernel cannot be used for given dataset.
- For polynomial kern el matrix with d = 2, all the datapoints are mapped to the feature space but clusters cannot be observed distinctly, hence this cluster is also not a suitable cluster.

(2)You are given a data-set with 1000 data points each in R2.

iv. Instead of using the method suggested by spectral clustering to map eigenvectors to cluster assignments, use the following method: Assign data point i to cluster ℓ whenever

$$\ell = \arg \max_{j=1,\ldots,k} v_i^j$$

where $v^j \in \mathbb{R}^n$ is the eigenvector of the Kernel matrix associated with the j-th largest eigenvalue. How does this mapping perform for this dataset?. Explain your insights.

Solution:

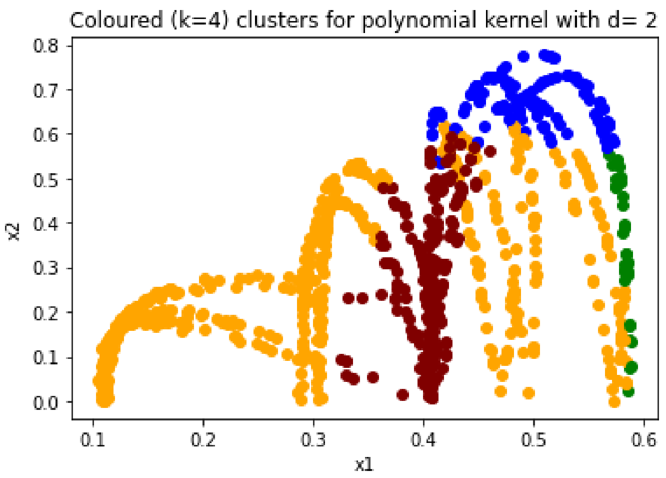
Please refer source code with file name: Q2_iv_Spectral_Clustering_ARG_MAX_MAIN_file_020322.py

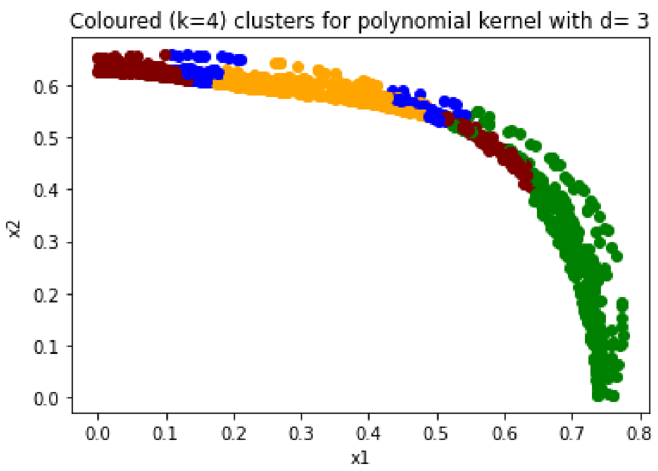
Note: 1) Q2_iv_Spectral_Clustering_ARG_MAX_import_file_020322.py file is already imported into above main file as a module and is kept in the same folder as the main file. 2) Kindly execute only the Q2_iv_Spectral_Clustering_ARG_MAX_MAIN_file_020322.py

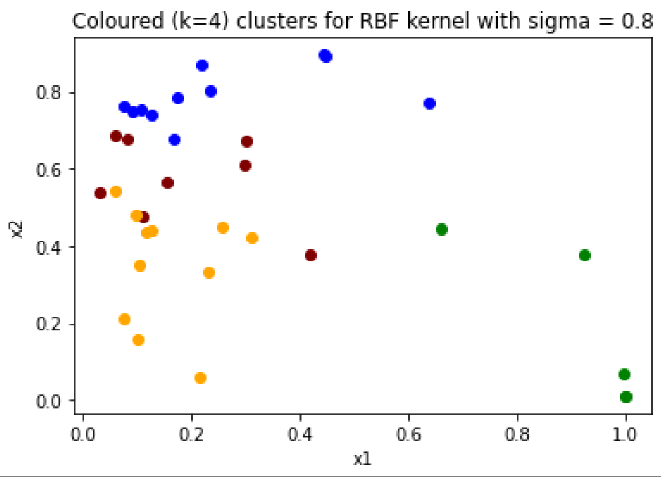
- **Result from source code:**

Plotting coloured clusters (k=4) for fixed rand. init. for ARG MAX based cluster assignment and ARG MAX assignment checking performance
Spectral clustering algorithm implemented and coloured clusters for kernels are plotted.

- **Plots:**







- **Explanation:**

- Three kernels were employed and implemented: polynomial kernel with d=2, polynomial kernel with d=3, Radial basis kernel with sigma = 0.8
- As compared to part iii of Q2, we observe that the clustering pattern obtained in part iv of Q2 (i.e. using arg max of given expression) is exactly same since we have employed the same set of kernels in both part iii and iv of Q2.
- However, we note that the cluster assignment in part iv. Is different from the cluster assignment obtained in part iii.
- This implies the method by which we assign clusters to the datapoints once we get the H (proxy for Z(^1/2) matrix will determine which cluster will the datapoint be finally assigned to.
- We observe that the Polynomial kernel with d=3 manages to categorizes the datapoints into 4 different clusters to some extent but the categorization is not well defined since some red colour dots appear in other cluster regions.
- Hence, this method proposed in part iv of Q2. Of assigning clusters to datapoints could be considered as an approximate cluster assignment to get a first hand approximate of distribution of datapoints in different clusters but it is certainly not the most optimized / reliable way of cluster assignment for datapoints.