ECE 544- Homework 4

Ahmadreza Eslaminia (Ae15)

Questions:

Problem 1:

ECE544. HW 4

Ahmadreza Eslaminia (AE15)

1. [k. Means]

$$\min \sum \frac{1}{2} r_{n,k} \|x - \mu_k\|_2^2 \quad \text{s.t.} \begin{cases} r_{n,k} \in \{0,1\} & \forall x \in D,\ k \in \{1,\dots k\} \\ \sum_{k \in \{1,\dots k\}} r_{n,k} = 1 & \forall x \in D \end{cases}$$

a) Since μ_k is cluster center of 2D points $x \in R^2 \Rightarrow \mu_k \in R^2$

b) If cluster centers μ_k are fixed, we can reduce minimization to:

$$r_{n,k} = \begin{cases} 1 & \text{if } k = \underbrace{\operatorname{argmin} \|x - \mu_k\|_2^2}_{\text{for } k \in \{1,\dots k\}} \\ 0 & \text{otherwise} \end{cases}$$

when cluster centers are fixed, then we chose for each datpoint "x" we choose $r_{n,k}$ s.t $\|x - \mu_k\|_2^2$ in minimized and thus the cost function which is the summation of all $\|x - \mu_k\|_2^2$ are minimum, then for $r_{n,k}$ achived with abovementioned formula is optimal.

c) if $r_{n,k}$ are fixed we have: cost function: $F = \min_{\mu_k} \sum_{\substack{x \in D \\ k \in \{1,\dots k\}}} \frac{1}{2} r_{n,k} \|x - \mu_k\|_2^2$

It's a Quadratic function which has a min at: μ_k^* where $\nabla F_{\mu_k^*} = 0$

$$\Rightarrow \nabla_{\mu_k} F = \sum_{x \in D} r_{n,k} (x - \mu_k) = 0 \Rightarrow \boxed{\mu_k^* = \frac{\sum_{x \in D} r_{n,k}\, x}{\sum_{x \in D} r_{n,k}}}$$

1. d) Pseudo-code:

- Initialize μ_k
- Iterate until stopping criteria:
  - update $r_{n,k}$ by $r_{n,k} = \begin{cases} 1 & \text{if } k = \underset{k \in \{1 \dots k\}}{\operatorname{argmin}} \|x_n - \mu_k\|_2^2 \\ 0 & \text{otherwise} \end{cases}$
  - update μ_k by $\mu_k = \dfrac{\sum_{x \in D} r_{n,k}\, x}{\sum_{x \in D} r_{n,k}}$

- yes it's granted to converg since in each step cost decreases since we analytically find the minimum. also cost has lower bound which is Zero. so we can say it does converg.

however, we cannot say that converg to global optimum since the Problem is not an Convex function.

e) two epoch

- cost: 4.55997
- Centers: 1: $\begin{cases} 1.9163 \\ -1.9143 \end{cases}$ 2 $\begin{cases} -2.0952 \\ 2.0540 \end{cases}$

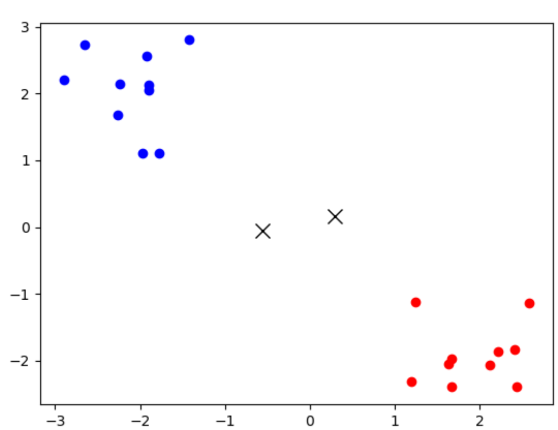
P2

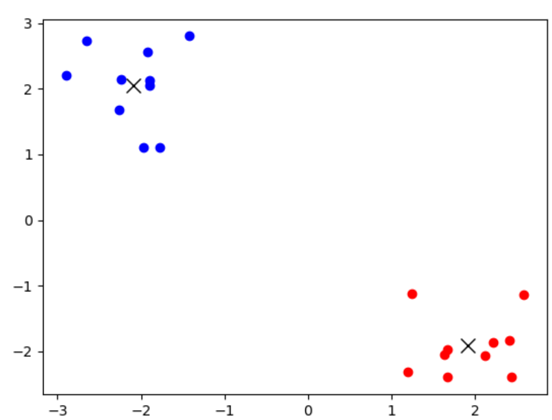
Code:

A7_KMeans.py × | PVA_fourbar_solved.m | PVA_inclass.m

C: > DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > hw4 > homework4 > A7_KMeans.py > ...

```python
import torch
import numpy as np
import matplotlib.pyplot as plt

N = 10
std = 0.5
torch.manual_seed(1)
x = torch.cat((std*torch.randn(2,N)+torch.Tensor([[2],[-2]]), std*torch.randn(2,N)+torch.Tensor([[-2],[2]])),1)

def Plot(c):
    plt.plot(x[0,:N].numpy(), x[1,:N].numpy(), 'ro')
    plt.plot(x[0,N:].numpy(), x[1,N:].numpy(), 'bo')
    l = plt.plot(c[0,:].numpy(), c[1,:].numpy(), 'kx')
    plt.setp(l, markersize=10)
    plt.show()

c = torch.Tensor([[2, -2],[2, -2]])
ctmp = c.transpose(0,1).view(2,2,1)

Plot(c)
dist = torch.zeros(2,20)
for iter in range(10):
    ###############################
    ## compute the distance between points and cluster centers
    ## Dimensions: dist (2x20)
    ###############################
    dist[0,:] = 0.5*torch.norm(x-ctmp[0,:,:],dim=0)**2
    dist[1,:] = 0.5*torch.norm(x-ctmp[1,:,:],dim=0)**2


    val,assign = dist.min(0)
    print("Cost: %f" % torch.sum(val))
    for k in range(ctmp.size()[0]):
        mn = torch.mean(x[:,assign==k],1)
        ctmp[k,:,:] = mn.view(-1,1)

    Plot(c)
```





Problem 2:

2. [GMM and kmean]

GMM. K components → μ_k, σ_k^2, π_k $D=\{x_i\}$ $x_i \in R$ z_{ik} latent

a) log likelihood → $\log \prod_{x_i\in D} p(x_i|\pi,\mu,\sigma) = \sum_{x_i\in D}\log\sum_{k=1}^{k}\pi_k N(x_i|\mu_k,\sigma_k)$ s.t. $\sum\pi_k=1$

$$\Rightarrow \text{log likelihood} = \underbrace{\sum_{x_i\in D}\log\sum_{k=1}^{k} \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}}\exp\left(\frac{-(x_i-\mu_k)^2}{2\sigma_k^2}\right)}\quad \text{s.t. } \sum_{k=1}^{k} \pi_k=1$$

b) $$K=1 \to \max \sum_{x_i\in D}\log\left(\overbrace{\pi_1 \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{2\sigma_1^2}\right)}^{*}\right)$$

$$\Rightarrow \frac{\partial}{\partial\mu_1} = \sum_{x_i\in D}\frac{\pi_1\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{\sigma_1 2}\right)(+2)(x_i-\mu_1)}{\pi_1\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{\sigma_1 2}\right)2\sigma_1^2} = 0 \Rightarrow \mu_1 = \frac{\sum x_i}{|D|}$$

$$\frac{\partial}{\partial\sigma_1} = \sum_{x_i\in D}\frac{\left(-\frac12\right)(*) + \pi_1\exp\left(-\frac{(x_i-\mu_1)^2}{2\sigma_1^2}\right)\times\frac{1}{\sqrt{2\pi\sigma_1^2}}\times(+2)\left(\frac{(x_i-\mu_1)^2}{2\sigma_1^3}\right)}{*}$$

$$= \sum_{x_i\in D}\left(\frac{(x_i-\mu_1)^2}{\sigma_1^3}-\frac{1}{\sigma_1^2}\right)=0 \Rightarrow \sum_{x_i\in D}\sigma_1 = \sum(x_i-\mu_1)^2 \Rightarrow \sigma_1 = \frac{\sum_{x_i\in D}(x_i-\mu_1)^2}{|D|}$$

$$\frac{\partial}{\partial\pi_k} = \sum_{x_i\in D}\frac{\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{2\sigma_1^2}\right)}{\pi_1\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{2\sigma_1^2}\right)} + \lambda = 0 \Rightarrow \pi = \frac{-|D|}{\lambda}$$

$$\to \max \sum \log\left(\frac{-|D|}{\lambda}\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-(x_i-\mu_1)^2}{2\sigma_1^2}\right)\right)\right) + \lambda\left(\sum\frac{-|D|}{\lambda}-1\right)$$

$$\Rightarrow \frac{\partial}{\partial\lambda} = \sum\frac{-1/\lambda}{1/\lambda} = -1 \to \frac{|D|}{-\lambda}=1 \Rightarrow \lambda=-|D| \Rightarrow \pi = \frac{-|D|}{\lambda} = \frac{-|D|}{-|D|}=1$$

$P_{3/4}$

2. c) it's called multinomial distribution:

$$p(z_{i1}, z_{i2}, \dots, z_{ik}) = \prod_{k=1}^{k} \pi_k^{z_{ik}}$$

$$p(z_{ik}=1) = \pi_k \qquad \sum_{k=1}^{k} z_{ik} = 1$$

d) $$p(z_{ik}=1 \mid x^{(i)}) = \frac{p(z_{ik}=1)\, p(x^{(i)} \mid z_{ik}=1)}{\sum_{k=1}^{k} p(z_{ik}=1)\, p(x^{(i)} \mid z_{ik}=1)} = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \sigma_k)}{\sum_{k=1}^{k} \pi_k \mathcal{N}(x^{(i)} \mid \mu_k, \sigma_k)}$$

e) 1. zero temprature of ~~GMM~~ GMM

2. Same variance for all GM

3. $\forall k \to \pi_k = 1 \to$ changing to uniform distribution

f) Assume σ's are equal $\Rightarrow \sigma_k =$ constant, $\pi_k = 1$

objective $$\Rightarrow \min_{\mu} - \sum_{x_i \in D} \log \sum_{k=1}^{k} \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x_i-\mu_k)^2}{2\sigma_k^2}\right) \equiv \min_{\mu} - \sum_{x_i \in D} \log \sum_{k=1}^{k} \exp\left(-(x_i-\mu_k)^2\right)$$

approximation $$\Rightarrow \min_{\mu} - \sum_{x_i \in D} \varepsilon \log \sum_{k=1}^{k} \exp\left(-(x_i-\mu_k)^2/\varepsilon\right) \overset{\text{hopital}}{=\!=} \lim_{\varepsilon \to 0} \sum_{k=1}^{k} \frac{\exp\left(-(x_i-\mu_k)^2/\varepsilon\right)(x_i-\mu_k)^2}{\sum_k \exp\left(-(x_i-\mu_k)^2/\varepsilon\right)}$$

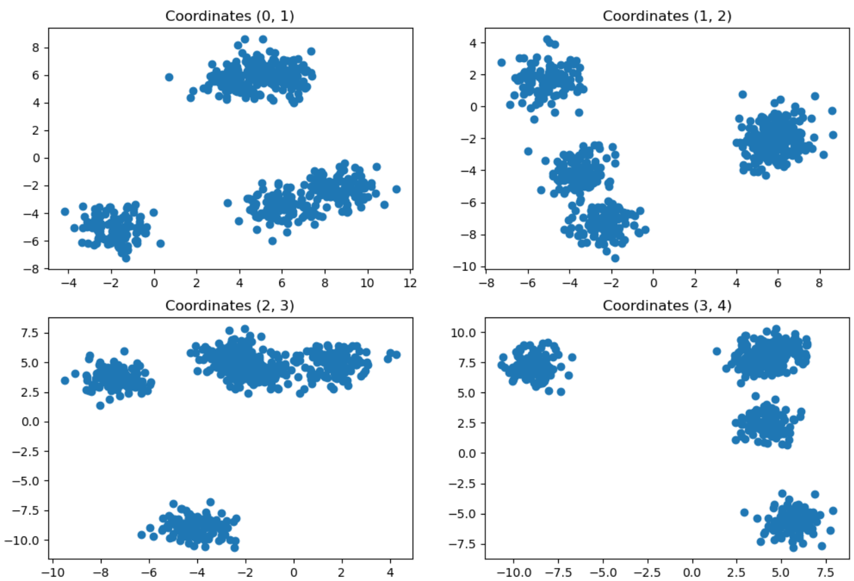
$$= \sum_{x_i \in D} \sum_{k=1}^{k} \mathbb{1}\left[k = \arg\min_k (x_i-\mu_k)^2\right](x_i-\mu_k)^2$$

P_I

Problem 3:

a) Plot:

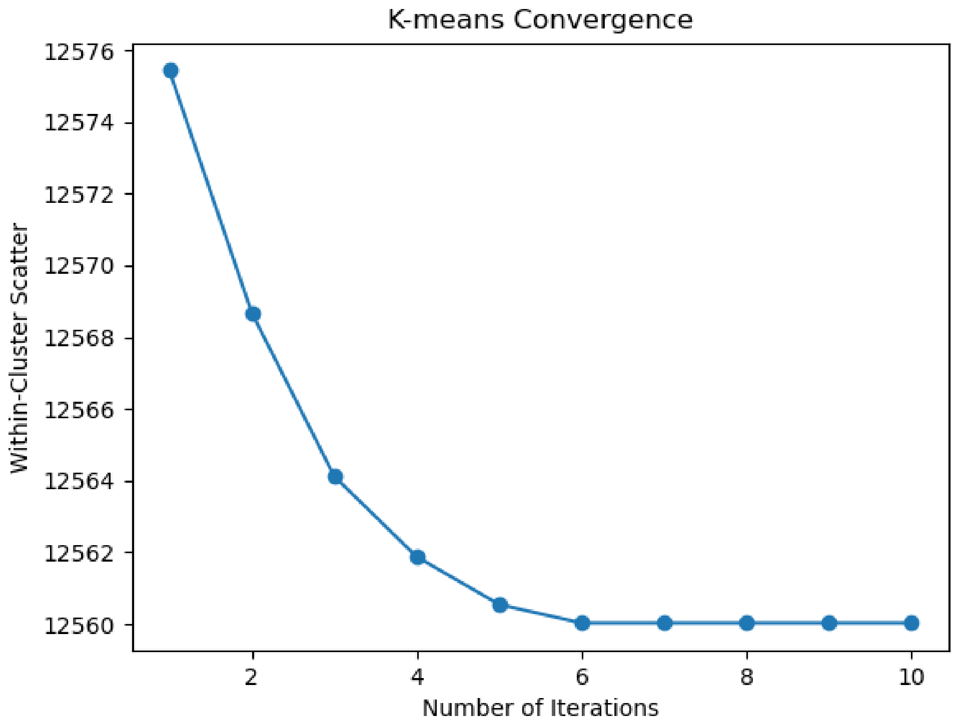
As you can see in some figures there are 4 clusters and in other figures there are 5 clusters.



Code:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Load the dataset
data = np.load('dataset_problem3.npy')

# Define pairs of coordinates for visualization
coordinate_pairs = [(0, 1), (1, 2), (2, 3), (3, 4)]

# Create subplots for visualization
plt.figure(figsize=(12, 8))

for i, (x, y) in enumerate(coordinate_pairs, start=1):
    plt.subplot(2, 2, i)
    plt.scatter(data[:, x], data[:, y])
    plt.title(f'Coordinates ({x}, {y})')

plt.show()

```

b) Plot:



Code:

DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > hw4 > homework4 > Problem3-b.py > ...

```
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Load the dataset
data = np.load('dataset_problem3.npy')

inertias = []

for i in range(10):
    if i == 0:
        kmeans = KMeans(n_clusters=5, init='random', random_state=0, max_iter=1, n_init=1
    else:
        kmeans = KMeans(n_clusters=5, init=kmeans.cluster_centers_, max_iter=1, n_init=1)
    kmeans.fit(data)
    inertias.append(kmeans.inertia_)

# Plot within-cluster scatter versus iterations
plt.plot(range(1, 11), inertias, marker='o')
plt.xlabel('Number of Iterations')
plt.ylabel('Within-Cluster Scatter')
plt.title('K-means Convergence')
plt.show()
```

c) Taking the 5 as the number of clusters according to previous scatter plots:

a. Random Initialization

```
Number of Iterations (Random Initialization): 28
Means of Clusters (Random Initialization):
 [[ 4.99041084  5.83811952 -2.08267398  4.86948203 -1.59008579]
 [ 5.85889425 -3.49425405 -4.11521301 -8.91851391  7.26676872]
 [ 8.71919112 -2.26401712 -7.33702562  3.66736492  7.76584753]
 [-1.86823866 -5.09980967  1.6649104   4.90742049  8.49422576]
 [ 4.72843789  6.37134526 -1.53628356  5.10471725 -1.81689547]]
```

b. K-Means initialization

```
  warnings.warn(
Number of Iterations (K-means Initialization): 9
Means of Clusters (K-means Initialization):
 [[-1.1353878  -4.12666116  0.56506425  4.33074235  8.63541772]
 [ 7.28904268 -2.87913559 -5.72611931 -2.6255745   7.51630812]
 [ 3.93433287  5.7162665  -2.52442182  5.61608643 -5.71115552]
 [-1.92418217 -5.17409674  1.74886917  4.95144227  8.48344762]
 [ 5.9880891   6.07884064 -1.51912333  4.1753169   2.48042293]]
```

As we can see using the initialization from K-means algorithm enables the algorithm to converge sooner (28 decreased to 9 iterations). This is because the K-means algorithm tries to calculate the clusters which is supposed to be better than the random guess.

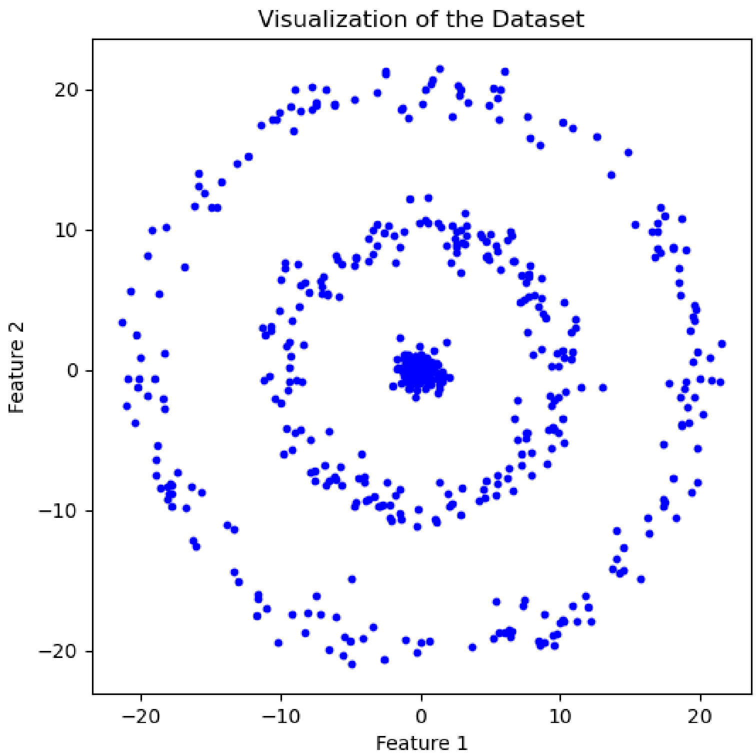
Code:

C: > DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > hw4 > homework4 > Problem3-c.py > ...

```python
import numpy as np
from sklearn.mixture import GaussianMixture

# Load the dataset
data = np.load('dataset_problem3.npy')

# Fit a GMM with random initialization
gmm_random_init = GaussianMixture(n_components=5, init_params='random', n_init=1, random_state=0) # Random state set for reproducibility
gmm_random_init.fit(data)

# Get the number of iterations for convergence with random initialization
iterations_random_init = gmm_random_init.n_iter_

# Report the mean values of each cluster for random initialization
means_random_init = gmm_random_init.means_

print("Number of Iterations (Random Initialization):", iterations_random_init)
print("Means of Clusters (Random Initialization):\n", means_random_init)

# Initialize the EM with centers obtained using K-means
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=5, init='random', n_init=1, max_iter=1)
kmeans.fit(data)
kmeans_centers = kmeans.cluster_centers_

gmm_kmeans_init = GaussianMixture(n_components=5, init_params='kmeans', n_init=1, means_init=kmeans_centers, random_state=0)
gmm_kmeans_init.fit(data)

# Get the number of iterations for convergence with K-means initialization
iterations_kmeans_init = gmm_kmeans_init.n_iter_

# Report the mean values of each cluster for K-means initialization
means_kmeans_init = gmm_kmeans_init.means_

print("Number of Iterations (K-means Initialization):", iterations_kmeans_init)
print("Means of Clusters (K-means Initialization):\n", means_kmeans_init)
```

Problem 4:

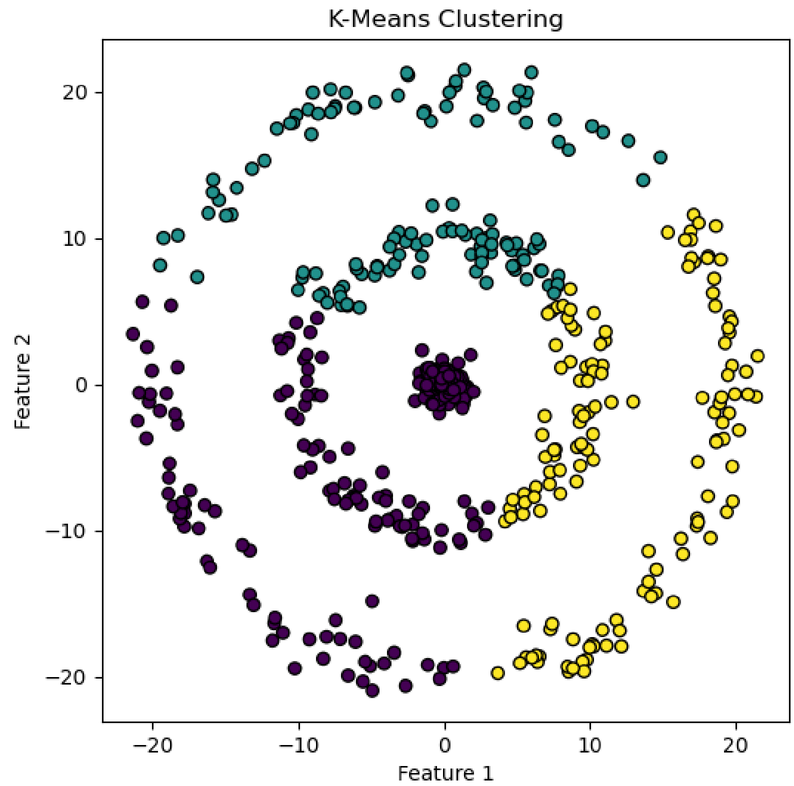
Visualization:



According to the visualization we consider three different clusters for this dataset.

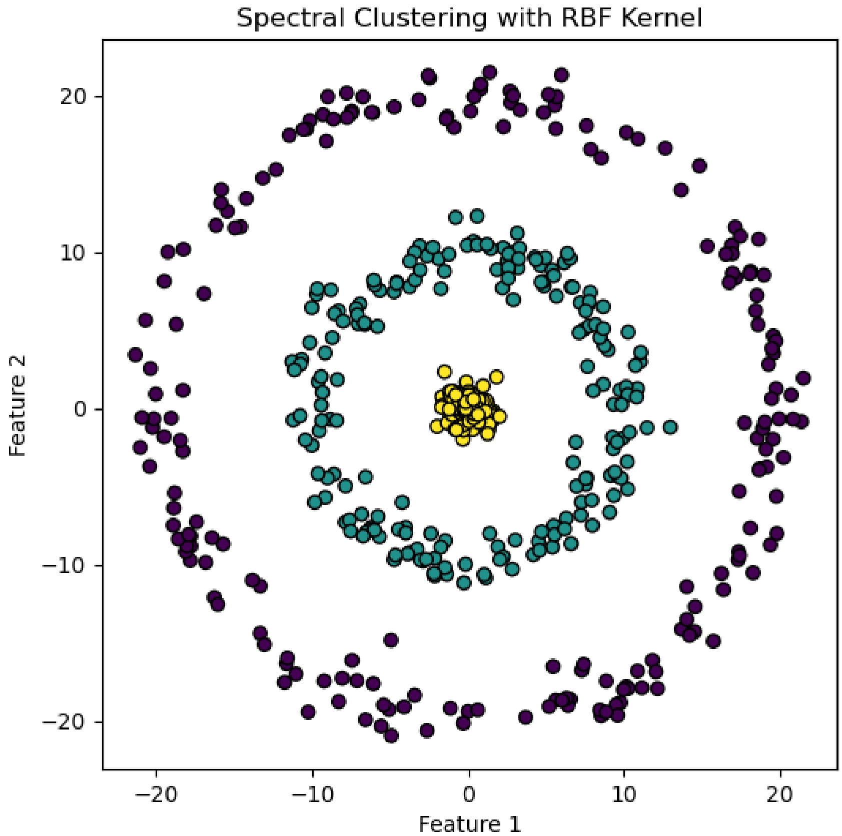
```
DriveA > UIUCcourses > Fall 2023 > ECE 544 pattern recognition > HWS > hw4 > homework4 > Probler
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, SpectralClustering
from sklearn.metrics import pairwise_distances
from sklearn.neighbors import kneighbors_graph

# Load the dataset
data = np.load('dataset_problem4.npy')

# Visualize the dataset
plt.figure(figsize=(6, 6))
plt.scatter(data[:, 0], data[:, 1], s=10, c='b', label='Data Points')
plt.title('Visualization of the Dataset')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
```

a) As you can see K-means clustering seem to have issue with clustering the radial shape clusters.




```
# (a) Using K-Means to cluster the dataset
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(data)
kmeans_clusters = kmeans.labels_

# Plot K-Means clustering results
plt.figure(figsize=(6, 6))
plt.scatter(data[:, 0], data[:, 1], c=kmeans_clusters, cmap='viridis', edgecolors='k')
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

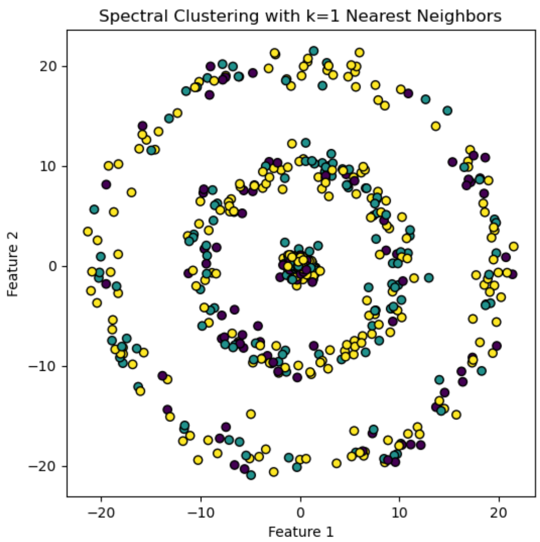
b) spectral clustering:



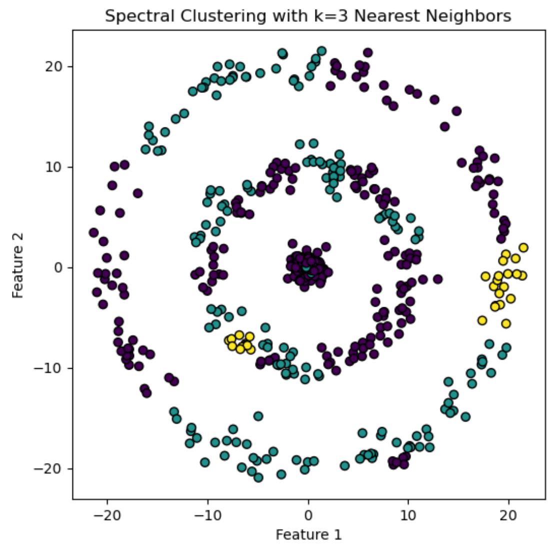

```
# (b) Spectral Clustering with RBF Kernel
affinity_matrix_rbf = np.exp(-pairwise_distances(data, squared=True))
spectral_rbf = SpectralClustering(n_clusters=3, affinity='precomputed', n_init=100, random_state=0)
spectral_clusters_rbf = spectral_rbf.fit_predict(affinity_matrix_rbf)

# Plot Spectral Clustering results with RBF Kernel
plt.figure(figsize=(6, 6))
plt.scatter(data[:, 0], data[:, 1], c=spectral_clusters_rbf, cmap='viridis', edgecolors='k')
plt.title('Spectral Clustering with RBF Kernel')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

c) KNN:As you can see by increasing the K from 1 to 7 the model gets better. However, from K=7 to bigger K it does not differ till K=11 (**getting similar results compared to section b**) . However, it is obvious if we increase the K so much it will assign the group with the most number of datapoint to all the datapoints so it get to perform worse.

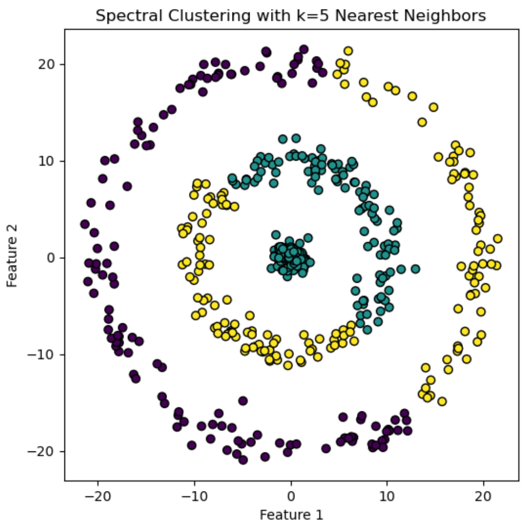
K=1



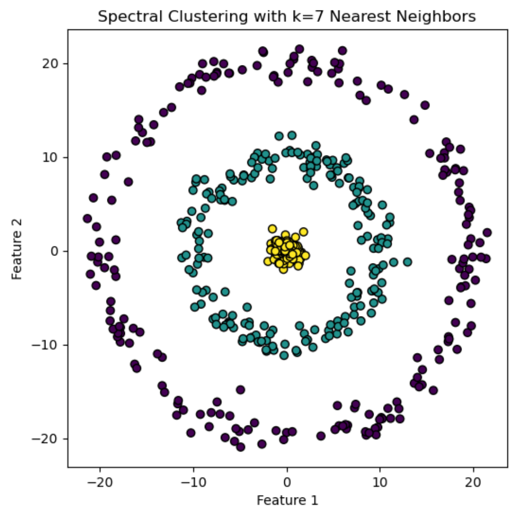
K=3



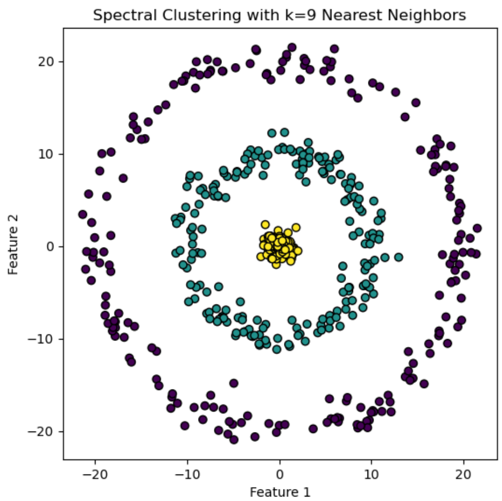
K=5

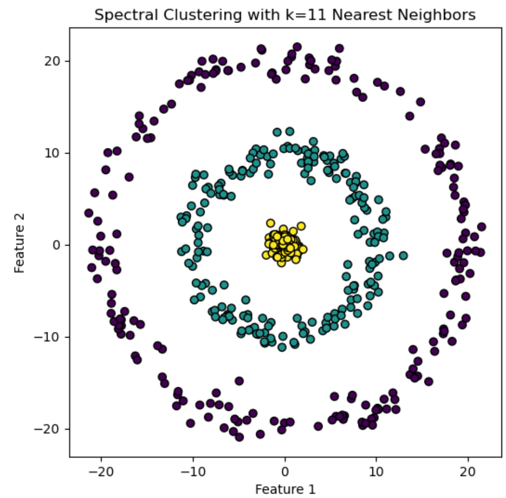


K=7



K=9 K=11






```python
# (c) Spectral Clustering with 'nearest neighbors' affinity
k_values = range(1, 21, 2)  # Vary k from 1 to 21, incrementing by 2

for k in k_values:
    affinity_matrix_knn = kneighbors_graph(data, k, mode='connectivity', include_self=True)
    spectral_knn = SpectralClustering(n_clusters=3, affinity='precomputed', n_init=100, random_state=0)
    spectral_clusters_knn = spectral_knn.fit_predict(affinity_matrix_knn)

    # Plot Spectral Clustering results with k-Nearest Neighbors
    plt.figure(figsize=(6, 6))
    plt.scatter(data[:, 0], data[:, 1], c=spectral_clusters_knn, cmap='viridis', edgecolors='k')
    plt.title(f'Spectral Clustering with k={k} Nearest Neighbors')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.show()
```