DA5400: Foundation of Machine Learning

Assignment 3

Nandhakishore C S (DA24M011)

November 17, 2024

1 Details regarding the files in submitted *.zip* file

The given submission **Solutions_DA24M011.zip** file contains three main types of files:

- **Python script(s)** files with ***.py*** extension, where all the **source code** is written. The classes for Principal Component Analysis, K-Means Clustering and it's corresponding helper functions are written in separate files.
- The file ***data_loader.py*** reads the dataset and checks for missing values. If the data is clean, it returns a *NumPy* array for using in computations.
- **Text file** with *.txt* extension, which contains my **details**.
- A **PDF file** named ***'Report.pdf'***(i.e.) this file, with the explanation for the solutions for the above question. The latex file for the report can be found here. There is also another file named ***rough_work.pdf*** which has the rough work and miscellaneous code which were tried during implementing the solutions for assignment.
- **To run** the python scripts, keep the datasets, ***.py*** files in the same directory and run *python3 main.py* in terminal to see the results.
- The **plots** are displayed while executing the code and also saved in the same directory as the code files.

2 Questions

1. Download the MNIST dataset from `https://huggingface.co/datasets/mnist`. Use a random set of 1000 images (100 from each class 0-9) as your dataset

   The MNIST Dataset contains images of photos of handwritten numbers from 0-9 in gray scale of size 28×28. The images are flatted to get a vector in $\mathbb{R}^{784}$, then PCA is applied to the vector to get top $'k'$ principal components. By default, the code calculates top 15 principal components for a dataset. A subset of this dataset is created by randomly sampling 100 images from each class. The subset contains 1000 images in total with 100 images in each class. For the ease of handling data, the datasets library from sci-kit learn is used.

   (a) Write a piece of code to run the Principal Component Analysis (PCA) algorithm on this data-set. Visualize the images of the principal components that you obtain. How much of the variance in the data-set is explained by each of the principal components?

   The pseudo code for PCA is given below. The source code keeps track of the size of dataset and calculates the Principal components using Left and Right Singular Matrices from Singular Value Decomposition (SVD). **Principal Component Analysis**

   Procedure PCA($X = [x_1, ..., x_n], X \in \mathbb{R}^{d \times n}, k \in \mathbb{R}_+ - \{0\}$)
   if $(d >> n)$ **then**
   COMPUTE $K = X^T X \in \mathbb{R}^{n \times n}$
   EIGEN DECOMPOSE(K)
   GET $\{\beta_1, ..., \beta_k\}$ FOR $\{n\lambda_1, ..., n\lambda_k\}$, WHERE $n\lambda_1 >= n\lambda_2 >= ... >= n\lambda_k$, λ ARE THE EIGEN VALUES AND β ARE THE EIGEN VECTORS OF K
   COMPUTE $\alpha_k = \frac{\beta_k}{\sqrt{n\lambda_k}, \forall k}$
   RECONSTRUCTION $w_k = X\alpha_k, \forall k w_k \in \mathbb{R}^d, ||w_k||^2 = 1$
   else
   COMPUTE $C = \frac{1}{n} X X^T \in \mathbb{R}^{d \times d}$
   EIGEN DECOMPOSE(C)
   GET $\{\alpha_1, ..., \alpha_k\}$ FOR $\{\lambda_1, ..., \lambda_k\}$, WHERE $\lambda_1 >= \lambda_2 >= ... >= \lambda_k$, λ ARE THE EIGEN VALUES AND α ARE THE EIGEN VECTORS OF C

12: RECONSTRUCTION $w_k = X\alpha_k, \forall k w_k \in \mathbb{R}^d, ||w_k||^2 = 1$
13: **end if**
14: **End Procedure**

Refer the source code files *decomposition.py* to see the source code. The visualization of the top 16 principal components of the given dataset is given in Figure 1. For the signal and noise directions of the dataset refer Figure 2. For the signal and noise components for the MNIST dataset, refer figure 3. From the image it is evident that, the eigen vectors for the last $k-1$ values, the components correspond to noise and they are not contributing much to the dataset.

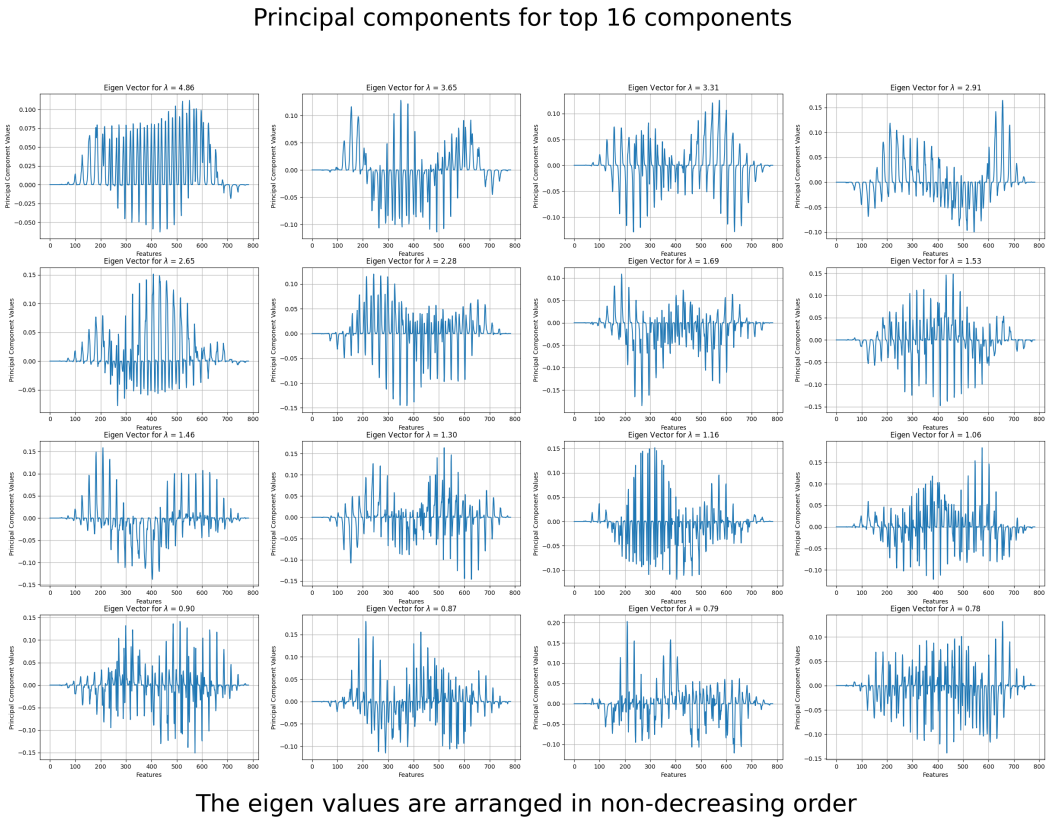

Figure 1: Top 16 Eigen vectors of MNIST dataset

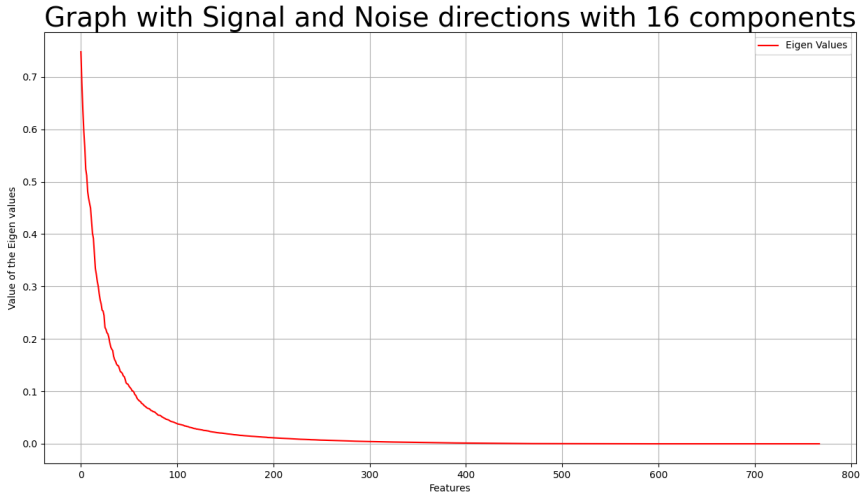

Figure 2: Signal and Noise directions for MNIST dataset.

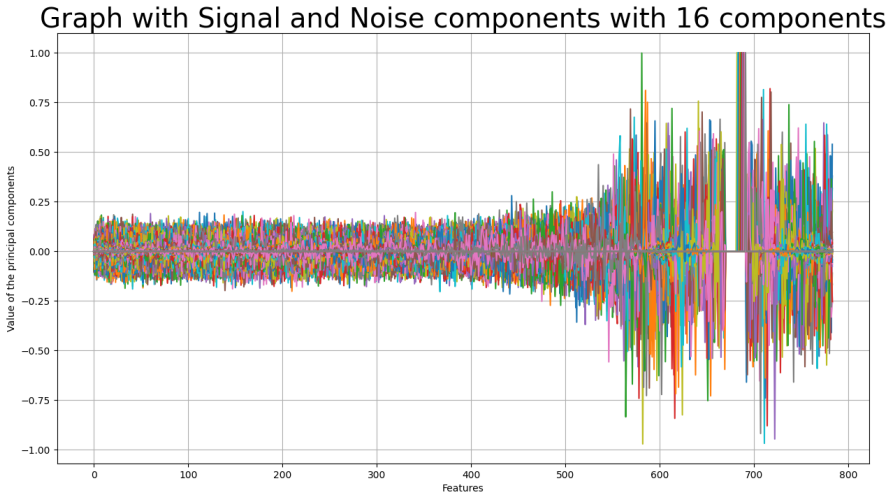

Figure 3: Signal and Noise Components for MNIST dataset. Each line in the plot is a principal component

(b) Reconstruct the dataset using different dimensional representations. How do these look like? If you had to pick a dimension 'd' that can be used for a down stream task where you need to classify the digits correctly, what would you pick and why? With $[10, 20, 50, 75]$ principal components of MNIST, the images were reconstructed. (Refer figure: 4), and for $d = 75$, we get good representation of the dataset, where the structure of the images are clearly visible. This can be used for downstream tasks, such as classification. Note that, for better visualization of pictures, the color map is set 'viridis' not gray scale. Actual images are in gray scale.

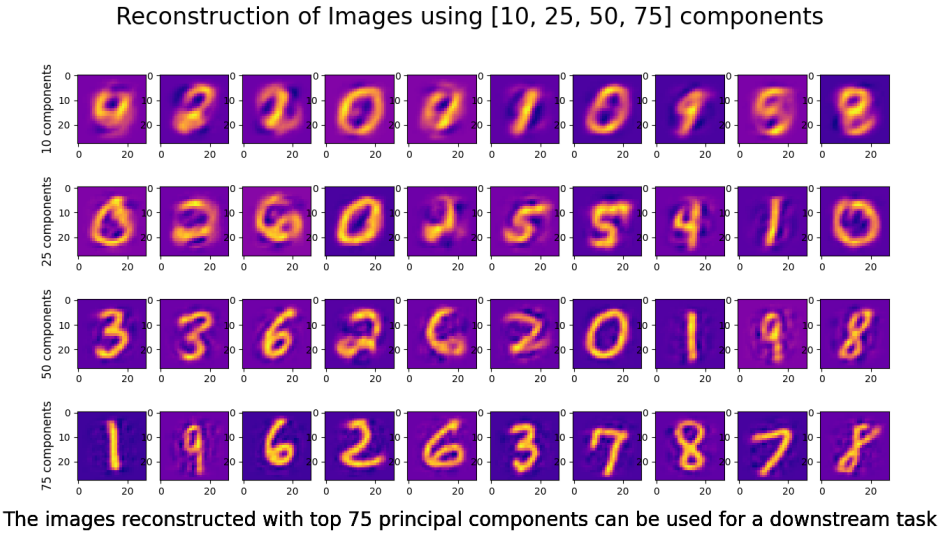

Figure 4: Reconstructed images with various number of Principal Components

2. You are given a data-set with 1000 data points each in $\mathbb{R}^2$ (cm_dataset2.csv).

The visualization of the given dataset, explains the structure / positioning of the points in $\mathbb{R}^2$ (Refer figure 5). From the image, we can clearly see that, there are two clusters in the dataset.

(a) Write a piece of code to implement the Llyod's algorithm for the K-means problem with k = 2. 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

The pseudo code for KMeans is given below. The source code keeps track of square of the distance between the data points and the cluster mean and reassigns data points to clusters when the distance between the current cluster and other cluster is more.

The given data set has two features (renamed as x1 and x2) for ease of handling data.

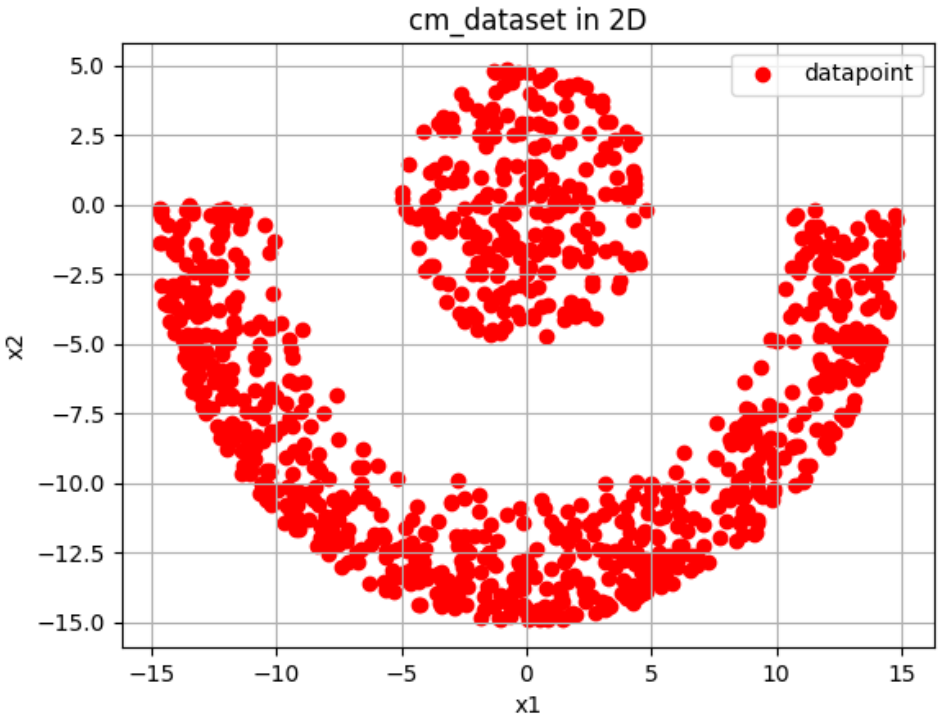

Figure 5: *cm_dataset.csv* in 2D scatter plot

Lloyd's Algorithm - K Means Clustering

1: **Procedure** K MEANS($X = [x_1, ..., x_n], X \in \mathbb{R}^{d \times n}, k \in \mathbb{R}_+ - \{0\}$)
2: **Assign Cluster Initialization** $Z = \{z_1^0, z_2^0, ...z_n^0\}$ **where** $z_i \in [1, k]$
3: **Until Convergence do** $\forall k$**, iterate T times where** $\mu_k^t = \frac{\sum_{i=1}^{n} x_i 1(z_i^t = k)}{\sum_{i=1}^{n} 1(z_i^t = k)}$
4: **Reassignment** $z_i^{t+1} = argmin_k ||x_i - \mu_k^t||^2$
5: **End Procedure**

When the clustering is done for random 5 initializations, we can clearly see that, even though the initial error is high, but all the models converge to similar clusters. Refer figure 6. Plots of 5 different clusters for random initializations (Refer figure 7 to 11).

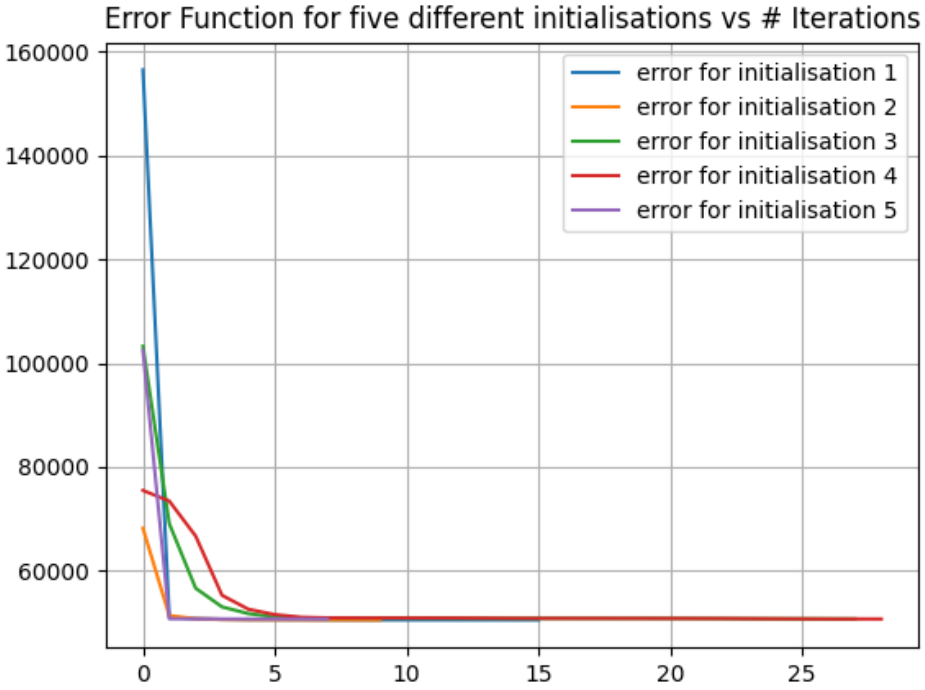

Figure 6: Error for K Means with K = 2, for 5 different initializations

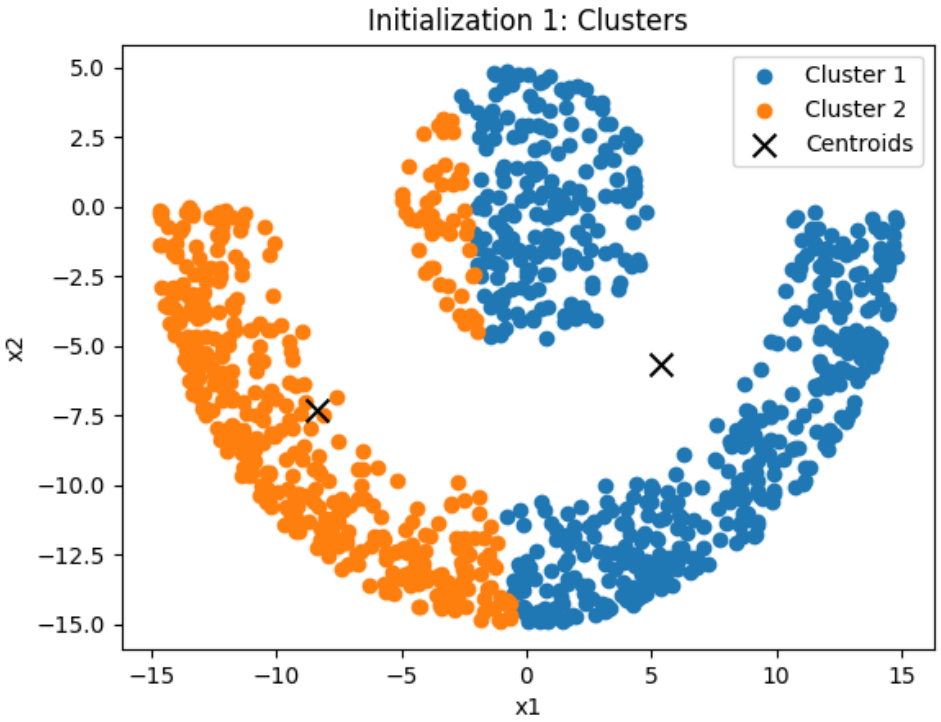

Figure 7: Clusters for K = 2 for Initialization 1

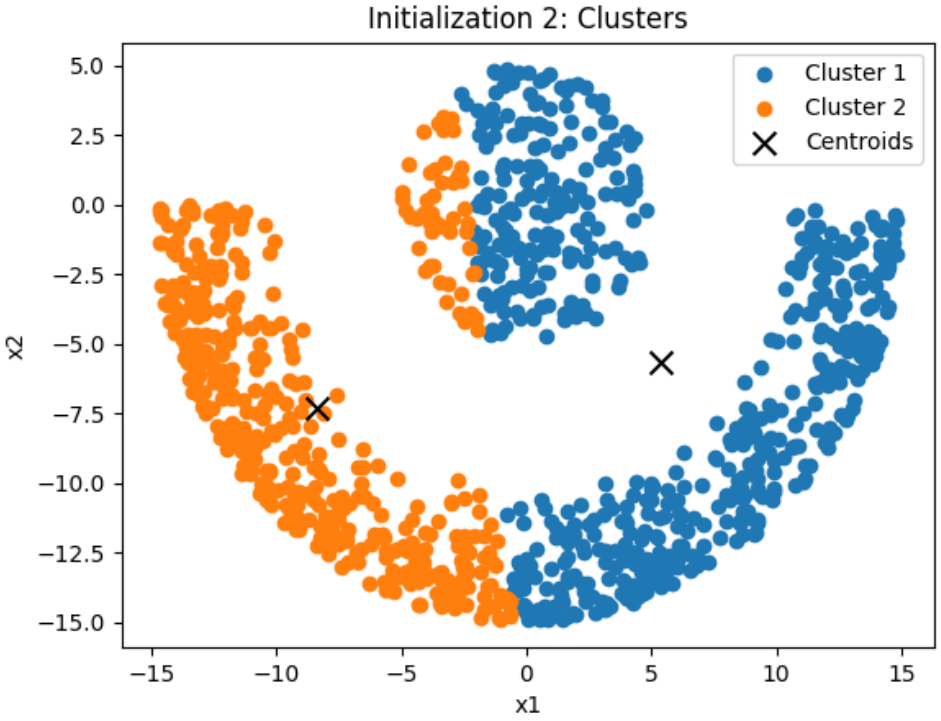

Figure 8: Clusters for K = 2 for Initialization 2

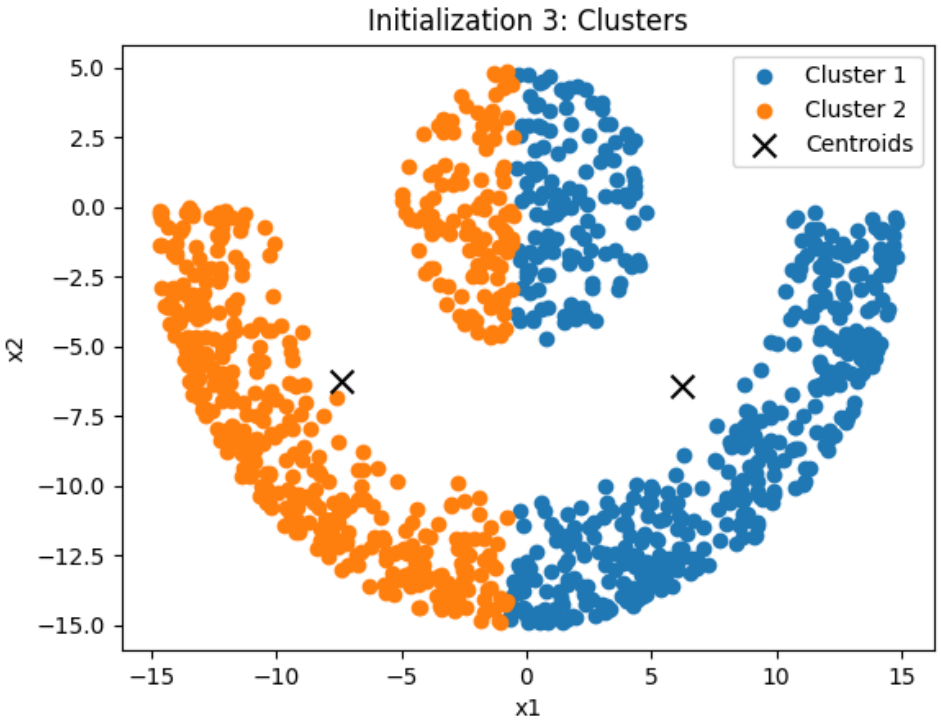

Figure 9: Clusters for K = 2 for Initialization 3

(b) For each K=2,3,4,5, Fix an arbitrary initialization and 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 $\mathbb{R}^2$)

The Voronoi regions are plotted using the mesh grid function. In the KMeans implementation, each cluster is assigned with a number (i.e.) cluster identifier and the voronoi regions are plotted based on the number of identifiers assigned to a cluster of data points. Refer helper functions file in the source code for detailed implementation. Figures 12 - 15 describe the clusters using K means for k in [2, 3, 4, 5] and their voronoi regions.

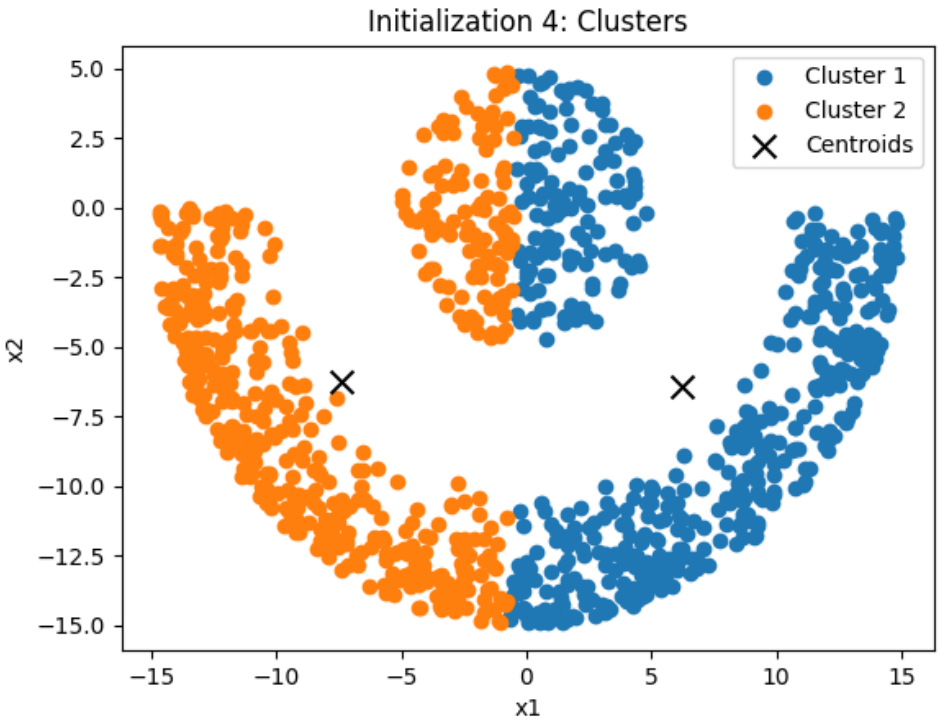

Figure 10: Clusters for K = 2 for Initialization 4

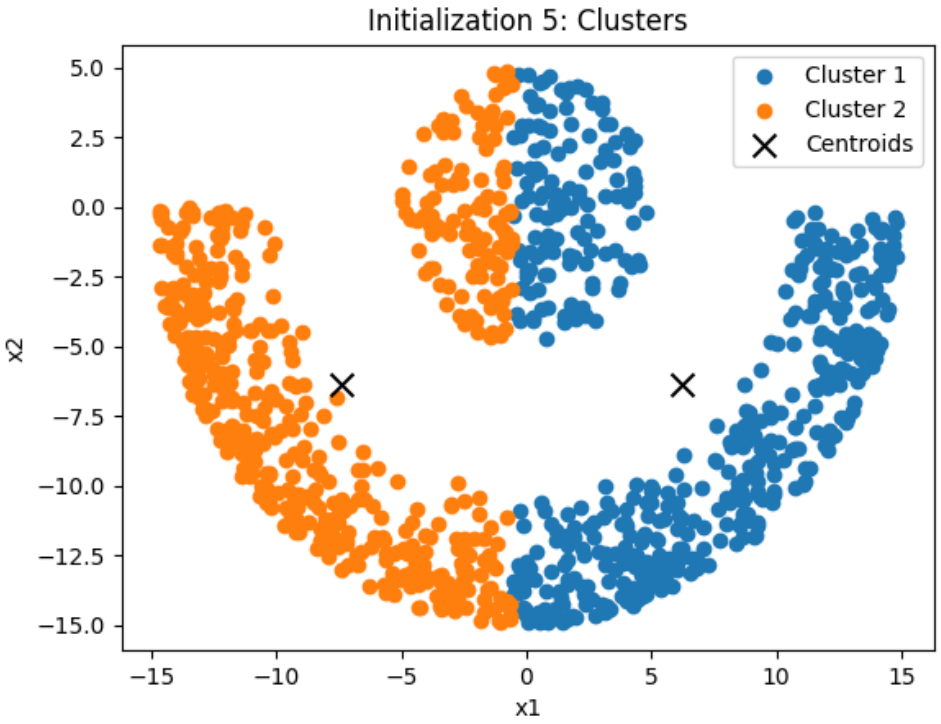

Figure 11: Clusters for K = 2 for Initialization 5

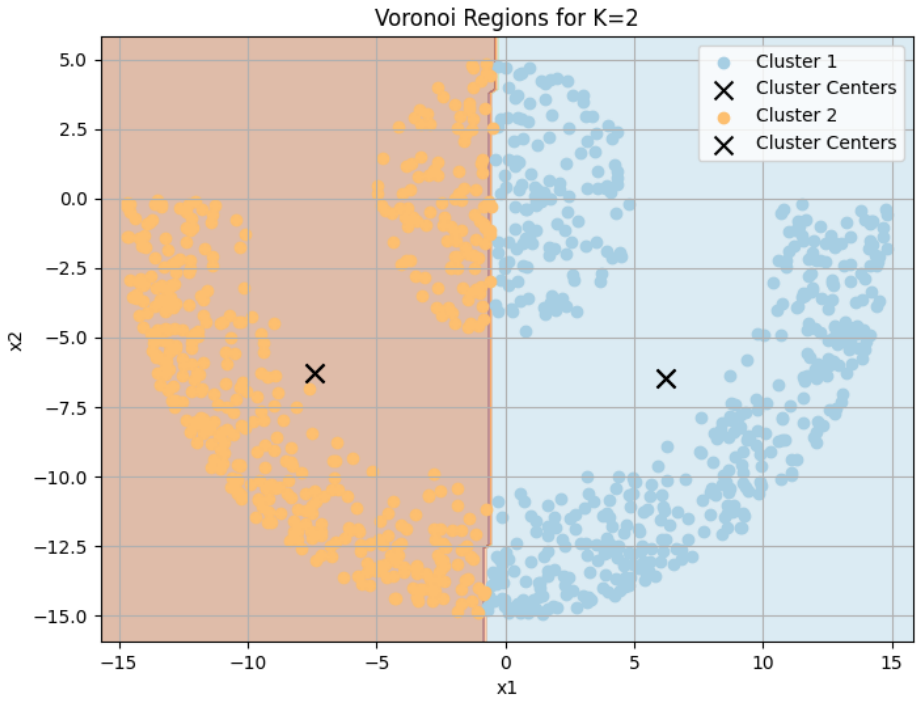

Figure 12: Voronoi Region for K Means Algorithm for K = 2 in cm_dataset

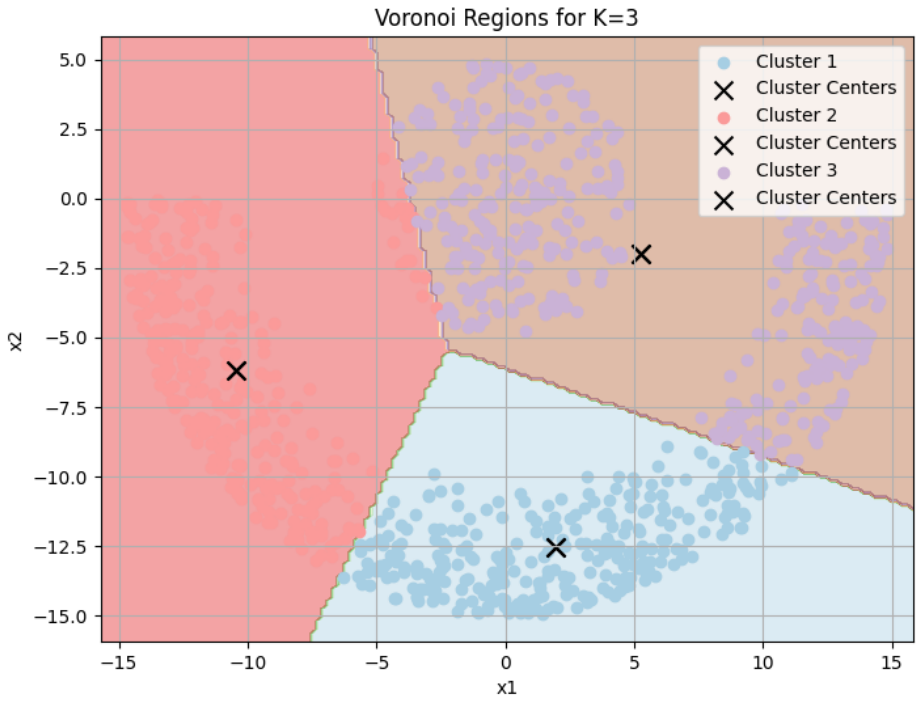

Figure 13: Voronoi Region for K Means Algorithm for K = 3 in cm_dataset

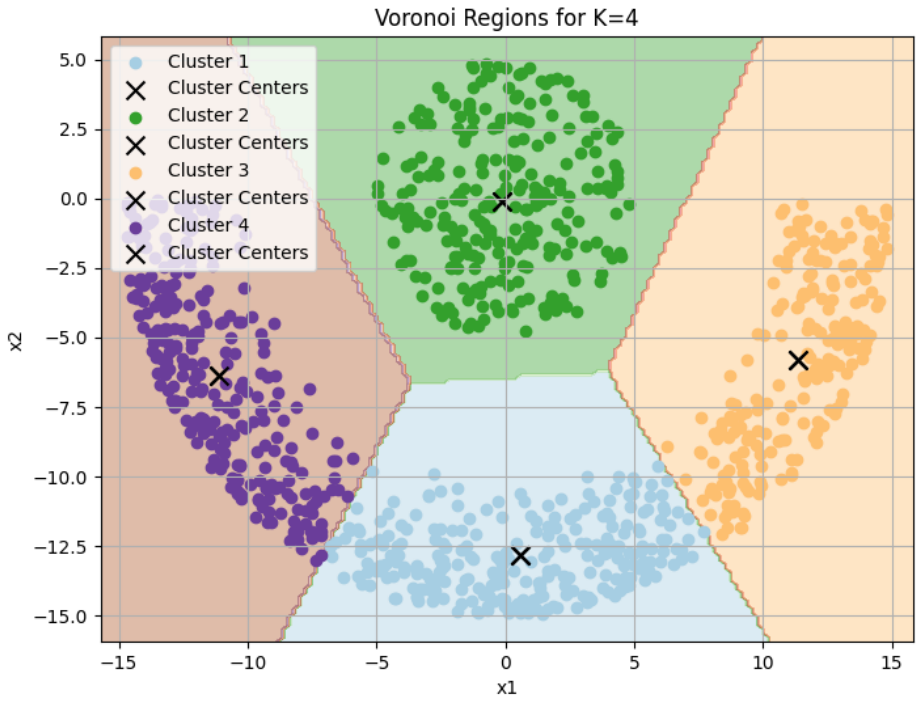

Figure 14: Voronoi Region for K Means Algorithm for K = 4 in cm_dataset

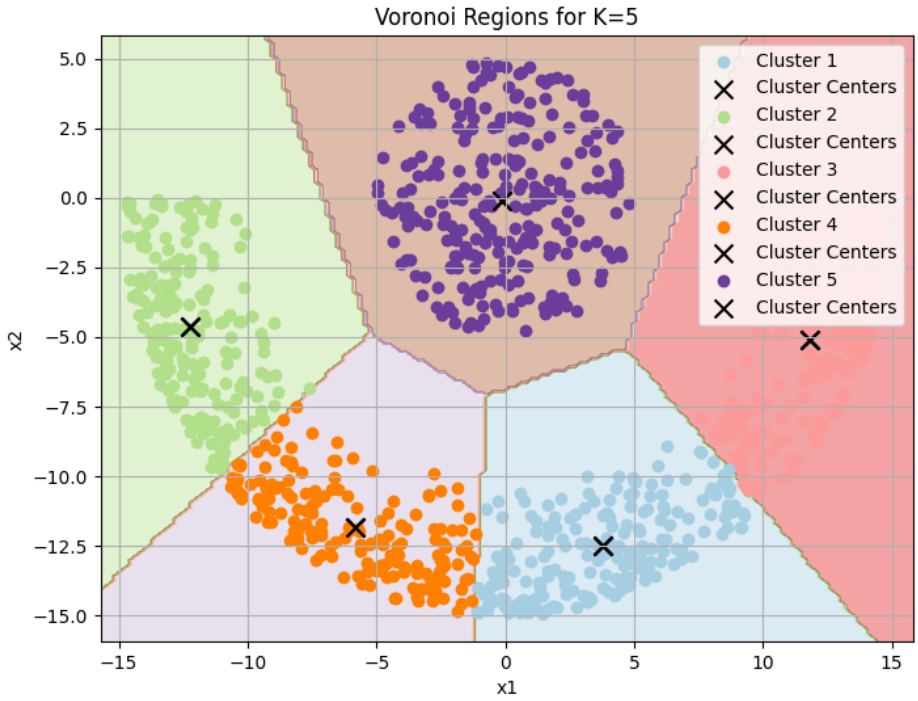

Figure 15: Voronoi Region for K Means Algorithm for K = 5 in cm_dataset

(c) Is the Llyod's algorithm a good way to cluster this dataset? If yes, justify your answer. If not, give your thoughts on what other procedure would you recommend to cluster this dataset?

The Answer is **no**. The vanilla version of the Lloyd's algorithm is not a good method to do clustering. When spectral clustering (Kernel version of Lloyd's algorithm with the kernel as the pseudo inverse of Laplacian matrix of data), the clusters are nicely separated. Refer figure 16 for the results of spectral clustering.

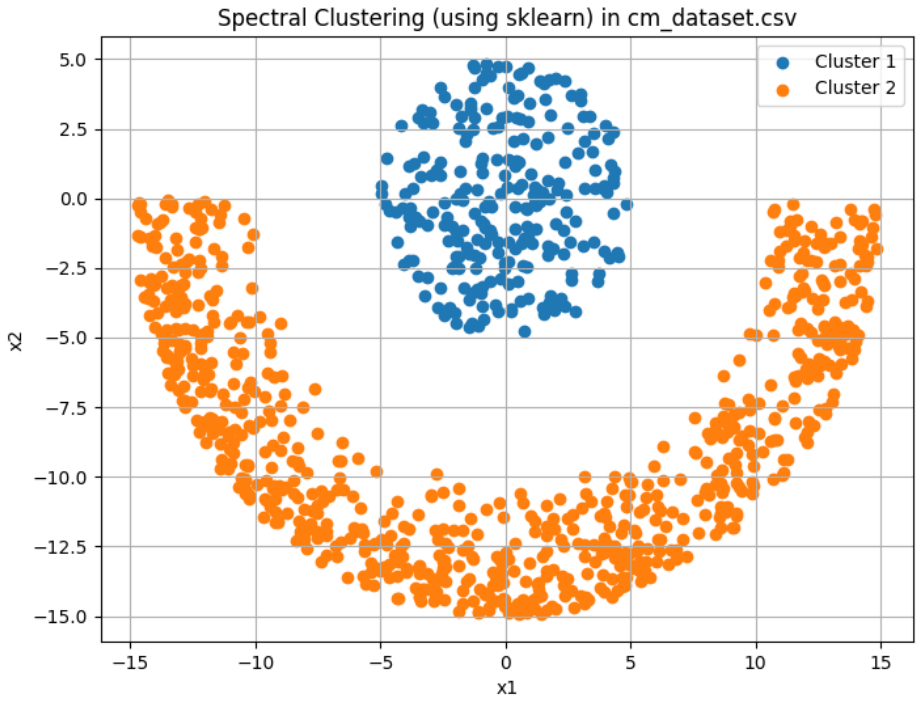

Figure 16: Voronoi Region from spectral clustering for cm_dataset when k = 2

3 References

i All the equations and mathematical concepts for Principal Component Analysis, KMeans and related concepts are referred from the book: *Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.*

ii Programming Language used: *Python3 - Van Rossum, G. Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.*

iii Python Libraries used:

- *Numpy - Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2*
- *Pandas - McKinney, W. others, 2010. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference. pp. 51–56.*
- *Matplotlib - J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science Engineering, vol. 9, no. 3, pp. 90-95, 2007*
- *SciPy - Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.*
- *Scikit-learn: Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.*