1 CS5691:Pattern Recognition and Machine Learning

1.1 Assignment 1

By: Sukriti Shukla
Roll no: ED20B067

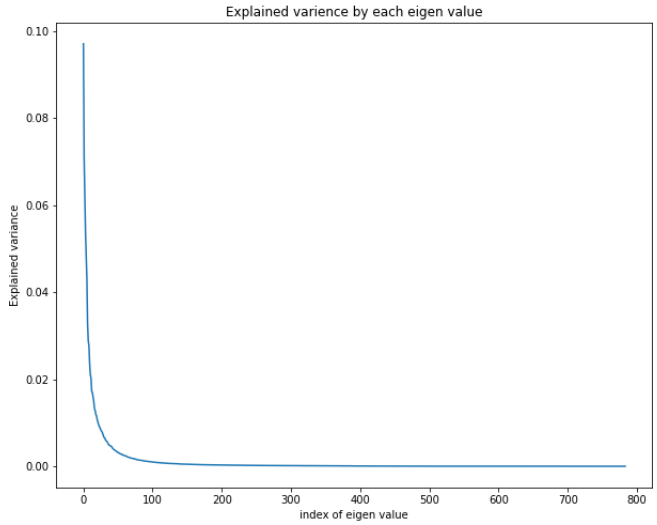
Question 1

i) I loaded the dataset using datasets library in python, I appended the 60000 train images to an array of size (60000, 784). I divided each data by 255.0 and centred the final train data. $X\_Centred = X\_train - X\_mean$

The code required 6 steps after that :

1. Calculating Covariance matrix.
2. Calculating eigen values and eigenvectors of the covariance matrix obtained.
3. Sort the top K eigen values and get the corresponding K eigen vectors. This gives us the top K principal components.
4. Variance experienced by each component is $variance = eigenvalue/sum(eigenvalues)$

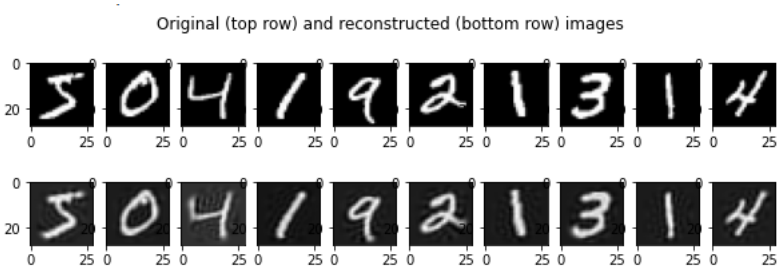
Figur 1: 5 principal components for 10 images



Figur 2: Variance explained by each eigen vector

ii) For second part I reconstructed 10 images from the dataset using different number of dimensions, after calculation cumulative variance for different number of dimensions,taking 150 top principal components gave variance nearly about 0.95, which is the desired value.

$$PC = X\_Centred * Selected\_Eigen\_Vectors$$
$$Reconstructed\_Images = PC[:N\_images] * Selected\_Eigen\_Vectors^T + X\_Mean$$



Figur 3: Reconstructed images for top 150 components

iii) I took only top 600 images in the Xtrain dataset, due to large execution time of the kernel PCA code. Then I did the following steps:

1. First, we calculate the values of the K matrix for different values of d using the following formula:

$$k_{i,j} = (1 + \mathbf{X}train[:,i]^T \mathbf{X}train[:,j])^d$$

where $\mathbf{X}_{train}$ represents the training dataset, and i and j represent the indices of the columns being considered.

2. Next, we center the K matrix by subtracting the mean of each column from each element in that column.

3. Then, we calculate the top 2 eigenvalues and corresponding eigenvectors of the centered K matrix. Let λ_1 and λ_2 be the top 2 eigenvalues, and $\mathbf{v}_1$ and $\mathbf{v}_2$ be the corresponding eigenvectors.

4. We calculate α using the following formula:

$$\alpha = \frac{\mathbf{v}}{\sqrt{\lambda}}$$

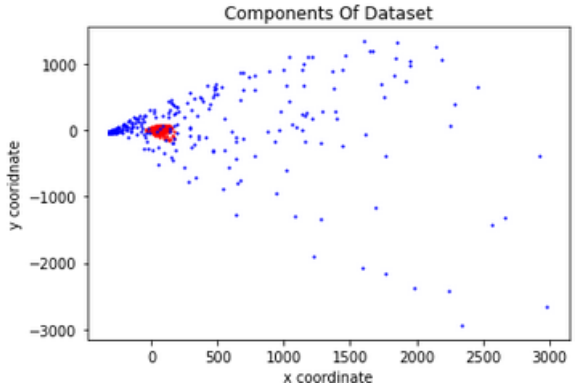
where $\mathbf{v}$ is the eigenvector and λ is the corresponding eigenvalue.

5. We calculate the components of the data set using the following formula:

$$\mathbf{c} = \alpha^T \mathbf{K}_c$$

where $\mathbf{K}_c$ is the centered K matrix.

6. Finally, we plot the data for different values of d.

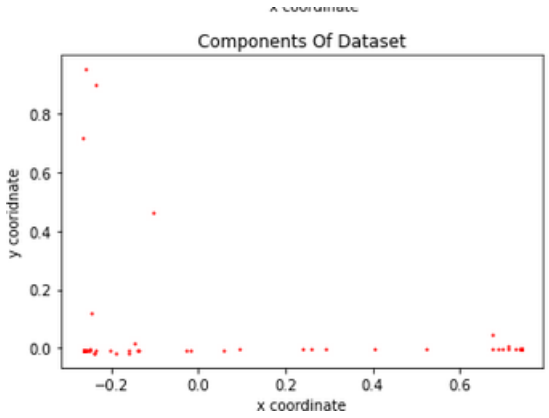


Figur 4: Polynomial Kernel

7. Then I repeated the above steps for the next kernel function:

$$k_{i,j} = \exp\left(-\frac{(\mathbf{X}train[:,i] - \mathbf{X}train[:,j])^T (\mathbf{X}train[:,i] - \mathbf{X}train[:,j])}{\sigma^2}\right)$$

for different values of σ



Figur 5: RBF kernel for $\sigma = 0.3$

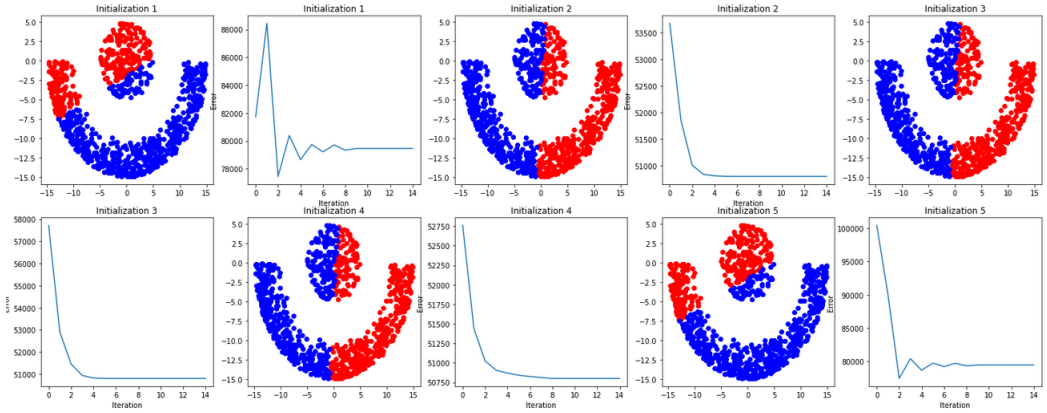
iv) As we can see in the above 2 graphs, it is evident that using the Gaussian kernel or Radial Basis Function (RBF) results in dense clustering of data for lower values of sigma, which then splits along the axes for higher sigma values. However, this does not provide any meaningful insight into the data. On the other hand, using a polynomial kernel results in data being widely spread out in the plot, which gives the impression that it could be a useful method for separating the data.

Question 2

i) I made 4 functions to perform various steps of lloyds algorithm:
Initialization: To randomly initialize the k centroids, we select k data points from our dataset as the initial centroids.
$C_1^{(0)}, C_2^{(0)}, \ldots, C_k^{(0)}$ where X is our dataset.
Reassign_clusters: To reassign clusters to the data points based on their distances with the current centroids, we calculate the Euclidean distance between each data point $x^{(i)}$ and each centroid C_j and assign the data point to the nearest centroid. $c^{(i)} = \arg\min_j ||x^{(i)} - C_j||^2$.
Reassign_centroids: To recalculate the centroid of each cluster taking the mean of all the data points in that cluster, we calculate the mean of all data points in each cluster.
Plot_clusters: To plot the different clusters in different colors, we can use a scatter plot with each cluster colored differently.

To run the k-means algorithm, we perform the following steps:

1. Initialize the centroids using the Initialization function.
2. Get clusters for that initial centroid using Reassign_clusters.
3. Reassign centroids according to the new clusters using Reassign_centroids.
4. Repeat steps 2 and 3 for num_iterations times to perform the Lloyd's algorithm.



Figur 6: Clustering and errors for 5 random initializations

The number of iterations, *num_iterations*, can be determined by testing various values and observing the convergence of the algorithm. We can also use an infinite loop to run the algorithm as long as it does not converge, testing if consecutive clusters are not the same, but this can take a lot of time for some initializations and may not produce better results than using a fixed number of iterations.

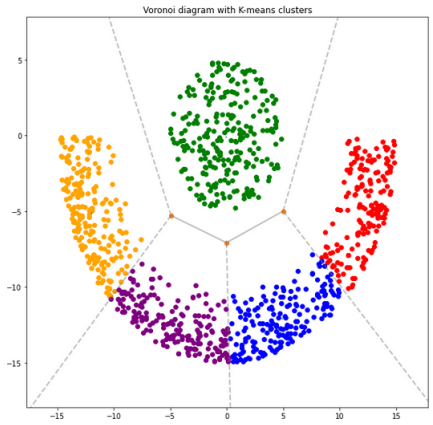
For the error function I used the following steps

1. Initialize error to 0.
2. For each centroid C_j:
   (a) For each data point x in cluster j:
   (b) Compute the distance between x and C_j.
   (c) Add the squared distance to the error.
3. Set objective[i] to the computed error.

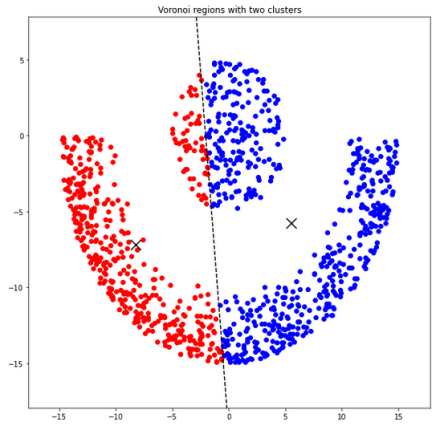
Finally, after getting the final clusters, we can plot them using the Plot_clusters function.

ii) I plotted Voronoi regions using the Voronoi class and voronoi_plot_2d function from the scipy.spatial library. The plot_clusters_vornoi function takes in the clusters, centroids, and the number of clusters (k), and generates a Voronoi diagram with the clusters plotted in different colors.

However, the voronoi_plot_2d function doesn't support drawing regions for 2 clusters, so I wrote a separate function plot_voronoi_regions_2 to handle this case. This function takes in the clusters and centroids, and computes the perpendicular bisector of the line connecting the centroids. It then plots the clusters, centroids, and the perpendicular bisector using the matplotlib library.



Figur 7: Vornoi regions for 5 clusters

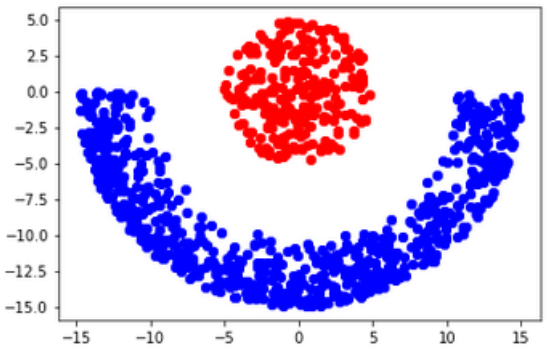


Figur 8: Vornoi regions for 2 clusters

iii) I tried using polynomial kernel but results were not satisfactory, so I moved to use RBF or the Gaussian kernel. I tried plotting data for different values of sigma but the result was not satisfactory.I did trails for various values of σ but the clustering more or less turned out to be random. So, I tried replicating the approach shown in the research paper https://ai.stanford.edu/ ang/papers/nips01-spectral.pdf.
The approach in the paper presents a new algorithm for clustering data based on the eigenvectors of the data's Laplacian matrix. The authors prove that the eigenvectors corresponding to the smallest eigenvalues of the Laplacian matrix can be used to cluster the data. They also show that the clustering can be improved by using a normalized Laplacian matrix.

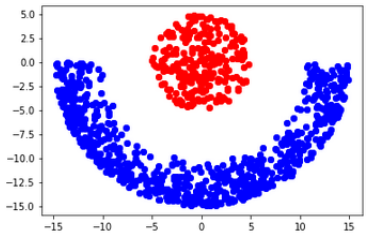
The algorithm consists of three steps:

1. construct the similarity matrix of the data points.
2. construct the Laplacian matrix of the similarity matrix
3. compute the eigenvectors of the Laplacian matrix and cluster the data based on the eigenvectors.



Figur 9: Clusters obtained using the new algorithm

iv) Algorithm first finds the index of the maximum value of h in each row, which indicates the index of the centroid to which the data point belongs. This is done using np.argmax(h, axis=1), which returns a 1D array of indices. Next, it creates an empty dictionary cluster to



Figur 10: Clusters obtained using the new algorithm

store the data points belonging to each cluster. It loops through the indices and for each data point, it appends the data point to the corresponding cluster list in the dictionary.

Finally, it calls the plot_clusters function to visualize the clusters. The parameters passed are the cluster dictionary and the values of 0 and 2 indicate the indices of the two dimensions to be plotted. The final results obtained are very similar to results obtained in part (iii). In this case the the plot we got was satisfactory, but its not guaranteed to work on all the datasets, because the H matrix we obtained might not be as seperable as in this case.