CMPE462 - Project 3
K Means & PCA

Sadullah Gültekin - 2014400066
Bahadır Hocamoğlu - 2014400117
Yusuf Başpınar - 2014400042

Spring Term, 2020

1 Introduction

In this project we implemented K-Means which is a supervised learning algorithm and PCA which is a unsupervised dimensionality reduction algorithm. Unsupervised learning is to extract information from data that does not have any labels, however in the supervised learning we have labels. Although we have labels in our given data, both algorithms that are implemented doesn't require any label in them. These labels are being used only in sanity checks. Unsupervised learning can also be considered a way to extract a higher level representation of the data. K-Means is doing classification among data by clustering groups based on similarity of objects. On the other hand PCA does dimensionality reduction by mapping the original features space into a smaller space. Due to the curse of high dimensioanlity models, effciency degrades rapidly as the dimension increases. Therefore PCA solves this issue by reducing dimensionality in the data. PCA generates new variables called Principle Components (PC) that will be generated in this project.

2 K-Means

2.1 Implementation Details

In K-Means implementation we started by downloading kmeans_data.zip first plotted the labels as can be seen in Figure 1.

We wrote K-Means class with functions that implements K-Means algorithm for any cluster number. This implementation uses euclidean distance (L-2 Norm) to calculate the distance between data points.

- **initialize function:** Initializes the centroids and belonging array which shows which datapoint belongs to which centroid. This array is initialized with zeros which mean all data points are belong to the first class. However, at the first iteration of the training, proper values will be assigned to the array.

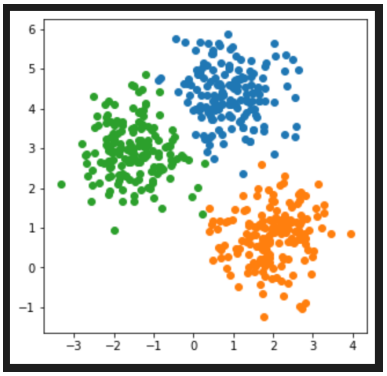

Figure 1: Clusters

- **visualize_data function:** Visualize the current clustering information into a scatter plot. If the axis, N, and title parameters are not given, single scatter plot that shows all clusters will be generated. Otherwise, generated scatter plot will be placed to the given axis object, using the given title.

- **assign_clusters function:** Calculates the distance between each centroid and each data point, and assings new belongings to each datapoint.

- **recompute_centroids function:** For each cluster calculates a new centroid by taking the mean of each cluster. After calculating new centroids, calculates euclidian distance for each data point and sums them up to get overall distance. Returns the calculated overall distance.

- **print_distance function:** Logging function.

- **stop_distance_criterion function:** Distance related stopping criterion for training loop.

- **stop_iteration_criterion function:** Iteration number related stopping criterion for training loop.

- **fit function:** Training function of K-Means clustering algorithm. Takes only the data and tries to split it into N cluster which is given in the initialization function. Runs until one of the following condiitons fulfilled:
  - Number of iteration gets equal to number of iteration defined by the user
  - Number of iteration reaches to the maximum number of iteration
  - Overall change in the distance gets smaller than the tollerance value

2.2 Evaluation

We run implemented k-means 9 times with number of iterations (N) =0,1,2,...,8 . Plotted the final clustering assignments as a scatter plot for each run as can be seen in

Figure 2 with seed 1 and Figure 3 with seed 8. We observed that after iteration 3 there is not much of a change in the clusters. Also with different seeds, K-Means may result with different clusters. Which shows the algorithm doesn't reach to optimal clustering results.

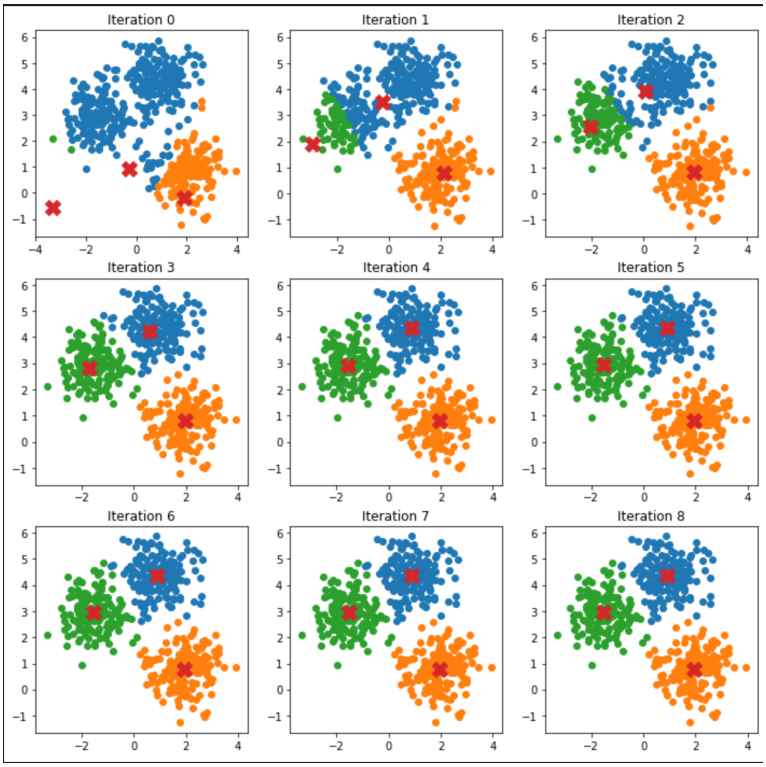

Figure 2: Seed 1

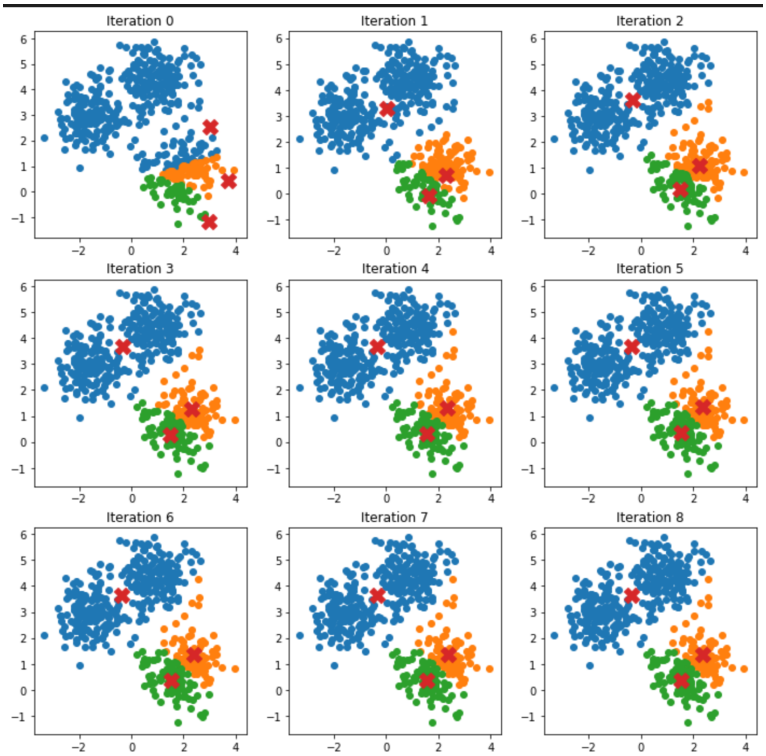

Figure 3: Seed 8

3 PCA

We first wrote PCA class that implements principal component analysis. It has functions to transform, inverse_transform, normalize and fit. Fit function is a training function for PCA algorithm. Takes the data and finds a transformation matrix which is assigned to a class variable. Number of principal components should be given in the initialization part. This algorithm can be written in a more clear way if class structure is not used. However for the ease of use and for good practice, we decided writing it in this way. In our implementation, user needs to create a PCA object by also giving the number of cluster. Then call the fit function giving only the data. The class will calculate the transformation matrix and save it into a class variable. If it is desired, it can be reached from there. After calling fit function, the user needs to call transform function using the same data. If different data is used, an error may occur because of the difference between the shape of transformation matrix and the shape of the data.

3.1 Image Transformation

After implementing the PCA class. We ran the algorithm with different d values, which corresponds to the number of principal components that will be used. As alternative d values, we used 50, 100, 150, 200, 256 values. because the image has 256 pixel, final d value results with the original image. We collected the resulted PCA objects and corresponding transformed data into a list which will be used in the next step

3.2 Image Reconstruction

We iterated all PCA objects and their corresponding transformed data and applied inverse transform on datas and collected reconstructed images.

3.3 Evaluation

We visualized the reconstructed images for the images at indices i= 0,500,1000,2000 for d = 50,100,150,200,256 . Then we created a 5x4 subplot in Figure 4 where the rows correspond to images at each index, first four columns correspond to reconstructed images using each d and the last column which is d = 256 is the raw image before PCA.

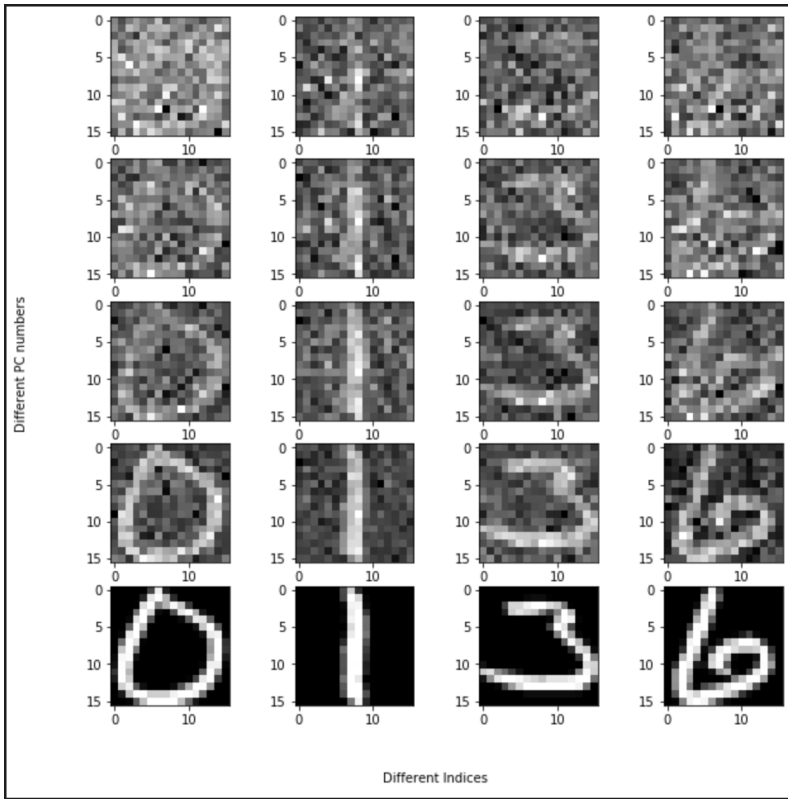

Figure 4: PCA runs with different d values