CMPE 462: Machine Learning

PROJECT 3

Implementing K-Means & PCA

Prepared By:

Fish & Fishers

Burak Ömür – 2016400186

Hasan Ramazan Yurt – 2016400078

Ömer Faruk Deniz – 2016400003

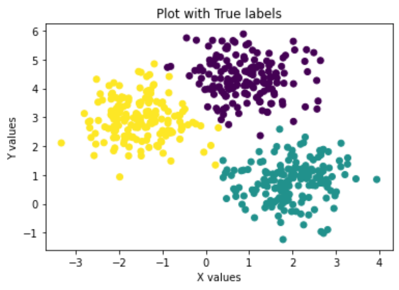
Overview

In this project, we are expected to implement two unsupervised learning algorithms. In the first task, we are expected to implement k-means, partitional clustering, and apply it to the given **kmeans_data**. In the second task, we are expected to implement principal component analysis and apply it to the images of digits in the **USPS.mat** dataset.

Task 1

K-means is a partitional clustering method, it is an unsupervised technique to learn from data. By arranging initial conditions, cluster number and iteration number we can obtain good results. It works best with globular shaped data.

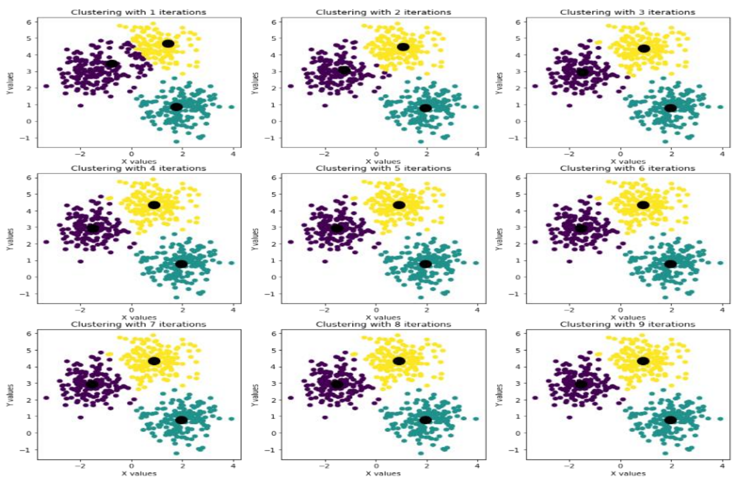
In this step, we import necessary libraries and load the labels and data that are given to us. Then, we plot the data with given labels to visualize what the real clustering looks like.



Then, we implement k-means clustering with the formula that has seen in lectures.

- Repeat n times:
  - Make random seed 1 to obtain pseudorandom numbers.
  - Randomly select k points and assign them as centroids of each cluster.
  - Compare centroids for each of the other points and assign points to closest cluster in terms of L2 – Frobenius norm.
  - Calculate and assign new centroids by averaging the points in the clusters.

We run algorithm with k=3, and n= {1,2…9} and plot the resulting clustering. The purpose of it to see how many steps are required to converge the intended result. However, starting conditions have important role here. If points are selected from edges of the graph in the random initialization, we may get worse results. And a solution for this would be making more than one initialization and run. Luckily, seed 1 gives good results. Our code converges with 2-3 iterations or so.



Task 2

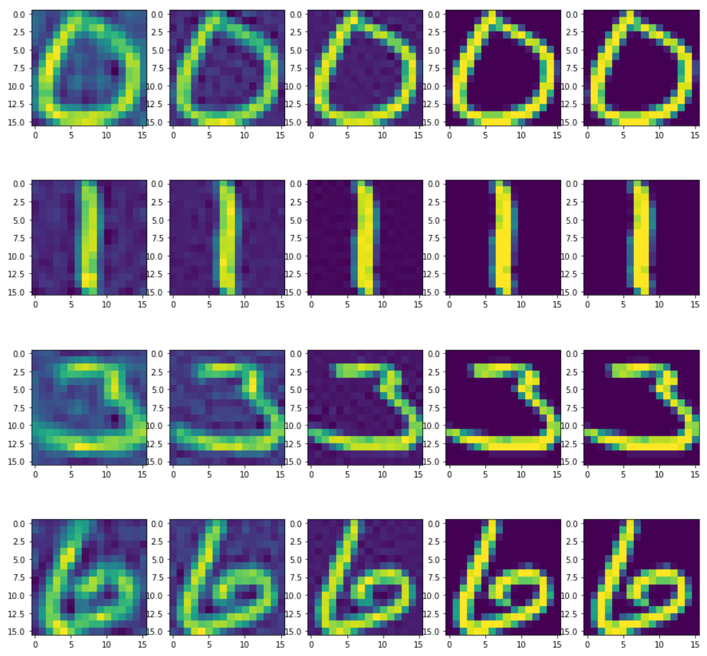
Principal component analysis helps us to see the most valuable features by maximizing the variance. Most valuable features can be obtained by sorting eigenvalues descending order and taking their corresponding eigenvectors.

To implement, firstly, we import libraries and load images from **USPS.mat** and plot the one of the images to see what the data looks like.

We have implemented the PCA using the formulas in the lecture slides.

- Load the data (3000x256)
- Normalize the data by subtracting mean and dividing to standard deviation.
- Calculate covariance matrix by dotting data matrix with itself and dividing to feature number.
- Return the eigenvalues-eigenvector tuples in descending sorted order of eigenvalues.
- Get n eigen vectors from start of tuples.
- Reconstruct the image matrix by dotting n eigenvector matrix with n eigenvector matrix dotted with original data matrix.

Then, we plot the reconstructed data with n = {50,100,200,256} in 4x5 subplots as below.



50 pca,   100 pca,   200 pca,   256 pca – original image

Comments on PCA and Results:

- Getting first 50 principal components gave us a decent amoung of information about the image. We can see and distinguish the number by looking at it because the most varianced feature covers the most information about image.
- We can see that 200 and 256 are not so different from each other which also shows that variance is small among the lowest values eigenvectors.
- We learned that using PCA, we can store the data in much less amount of features while storing a decent amount of variation at the same time.
- We believe that this algorithm will prove very useful to us when working with high dimensional data and will save us from storage & computational costs.