CMPE 462 Machine Learning
Department of Computer Engineering
Bogazici University

PROJECT 3

K-MEANS CLUSTERING AND PCA IMPLEMENTATION

Group Name: Hacı Kolonyası
Student ID1: 2015300084
Student ID2: 2015401183

14.06.2020

Contents

1. K-Means Clustering

In this section we implemented clustering which is one of the unsupervised learning methods.

1.0.1 Plot of Ground Truth Clusters

First, we plotted the data with given labels.

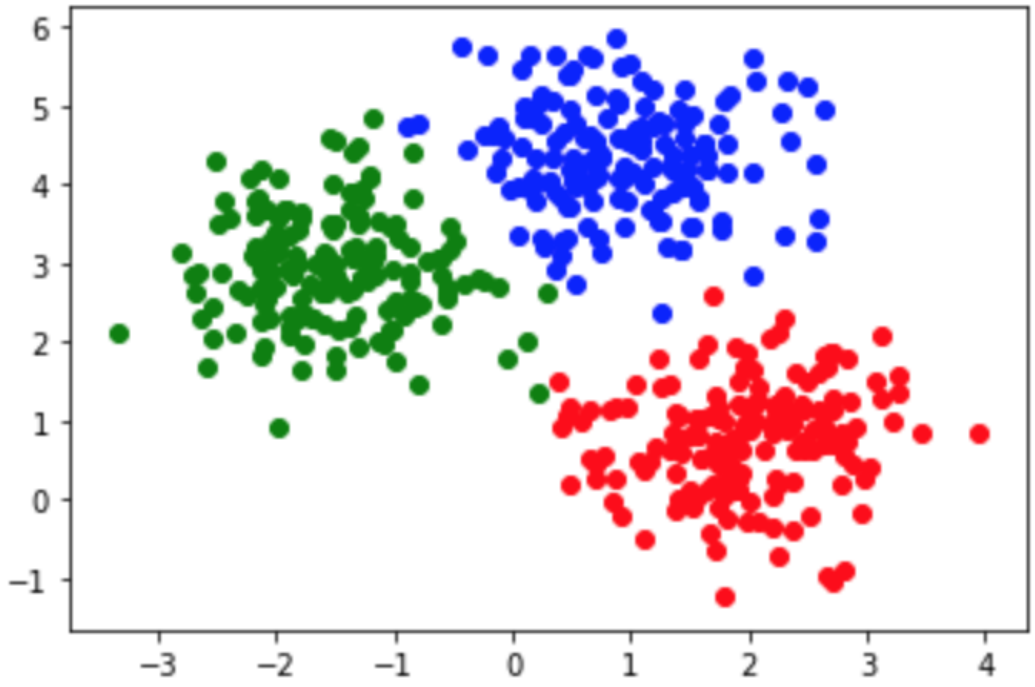

Figure 1.1: Original clusters in the data

1.0.2 K-Means Implementation

We implemented the k-means algorithm as given in the slides. We used iteration number as stopping condition for the algorithm.

- First, we chose given k points as our initial centers.
- Second, we calculated L2 norms of a data point to all k centers and assigned it to a closest one. We did it for data points.

- In the third step after assigning all data points to their closest centers, we recalculated each center by taking average of all points in a particular cluster by the following formula.

$$new \;\; center = \frac{\text{summation of all vector in a particular cluster}}{\text{number of the total data points in the cluster}} \tag{1.1}$$

- We repeated step 2 and step 3 for the number of iterations.

1.0.3 Evaluation

In this last task we run k-means 9 times with number of iterations $(N) = \{1, 2, ..., 9\}$. We used 3 as the number of clusters. Here is the plot of 9 different iterations.

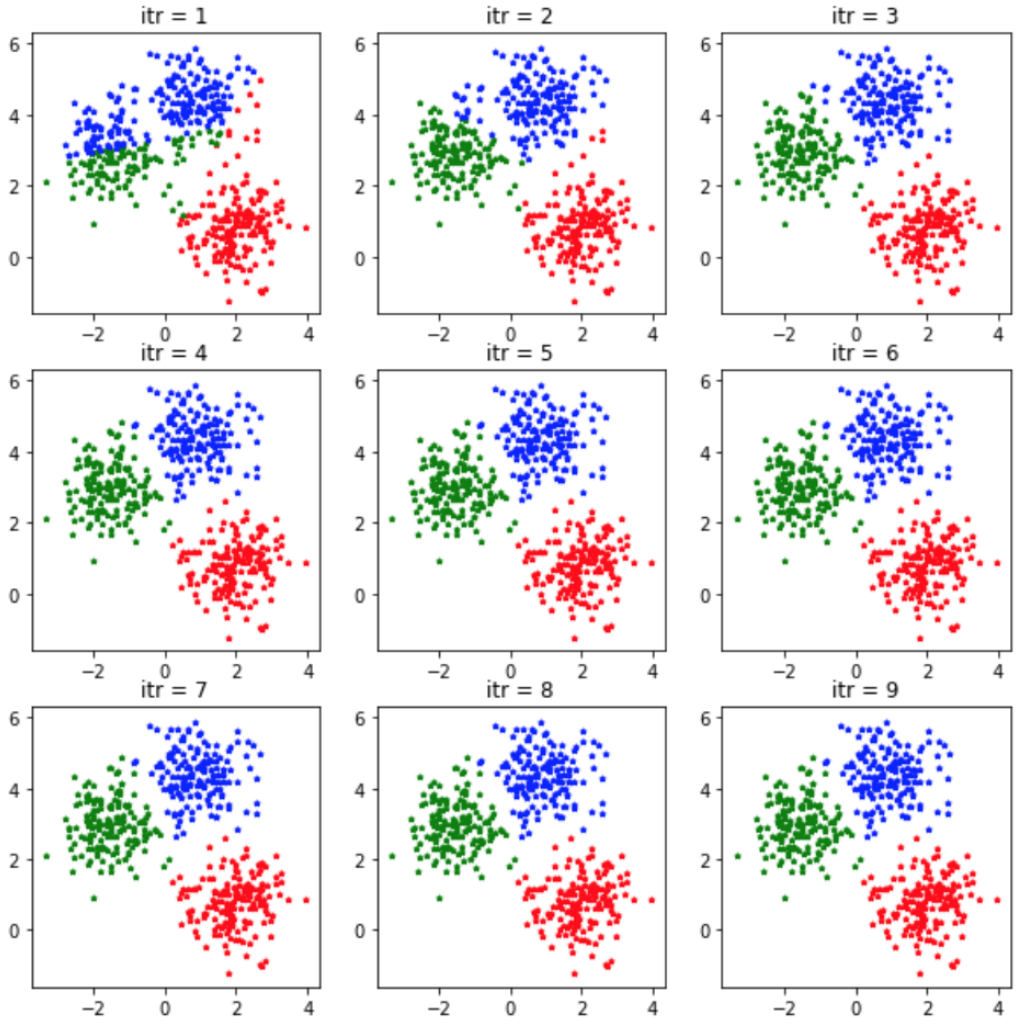

Figure 1.2: Plot of different iterations

Conclusion: Model converges after iteration 4. After that number it doesn't change with the number of iterations.

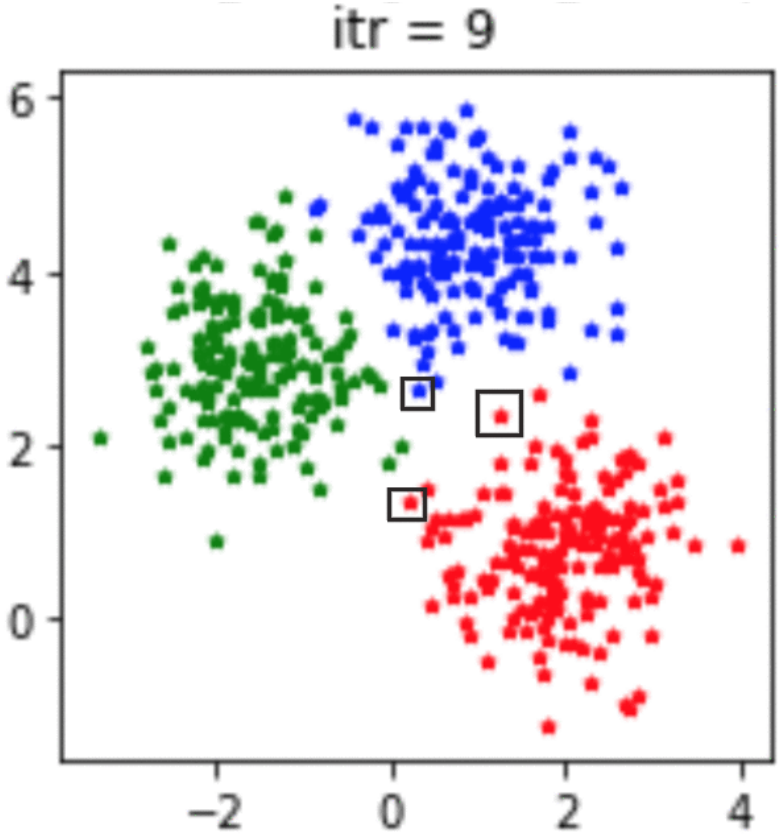

Figure 1.3: Comparison with original labels

Our model predicted true labels with error approximately 0.006. It misclassified only 3 data points.

2. PCA Implementation

- Training Data Standardization

  training data: $X \epsilon R^{nxp}$ for n observations and p features

  mean features vector: $\bar{x} \epsilon R^{px1} = \frac{1}{n} \sum_{j=1}^{n} x_j$ where x_j is observation j at the training data.

  mean features matrix: $\bar{X} \epsilon R^{nxp} = \begin{bmatrix} \bar{x}^T \\ \bar{x}^T \\ \vdots \\ \bar{x}^T \end{bmatrix}$

  standardized training matrix: $\tilde{X} \epsilon R^{nxp} = X - \bar{X}$

- Covariance Matrix: $S \epsilon R^{pxp} = \frac{1}{n} \tilde{X}^T \tilde{X}$

  We used numpy cov matrix rather than this formula.

- Transformation Matrix: $G = \epsilon R^{pxd} = \begin{bmatrix} a_1 & a_2 & \dots & a_d \end{bmatrix}$ where $a_k \epsilon R^{px1}$ vector is the eigenvector of the kth largest eigenvalue of the covariance matrix.

- Constructed Training Matrix $X^{'} \epsilon R^{nxd} = XG$

- Reconstructed Training Matrix $X^{''} \epsilon R^{nxp} = X^{'} G^T = XGG^T$

We implemented these formulas to calculate constructed data and reconstructed data. We calculated matrices for d = 50, 100, 200 and 256. After calculations, we checked images at indices 0, 500, 1000, 2000. For lower d, the images more blurred. The images for d = 256 and original images are same. Also, it's difficult to see any differences between images for d = 200 and original images.

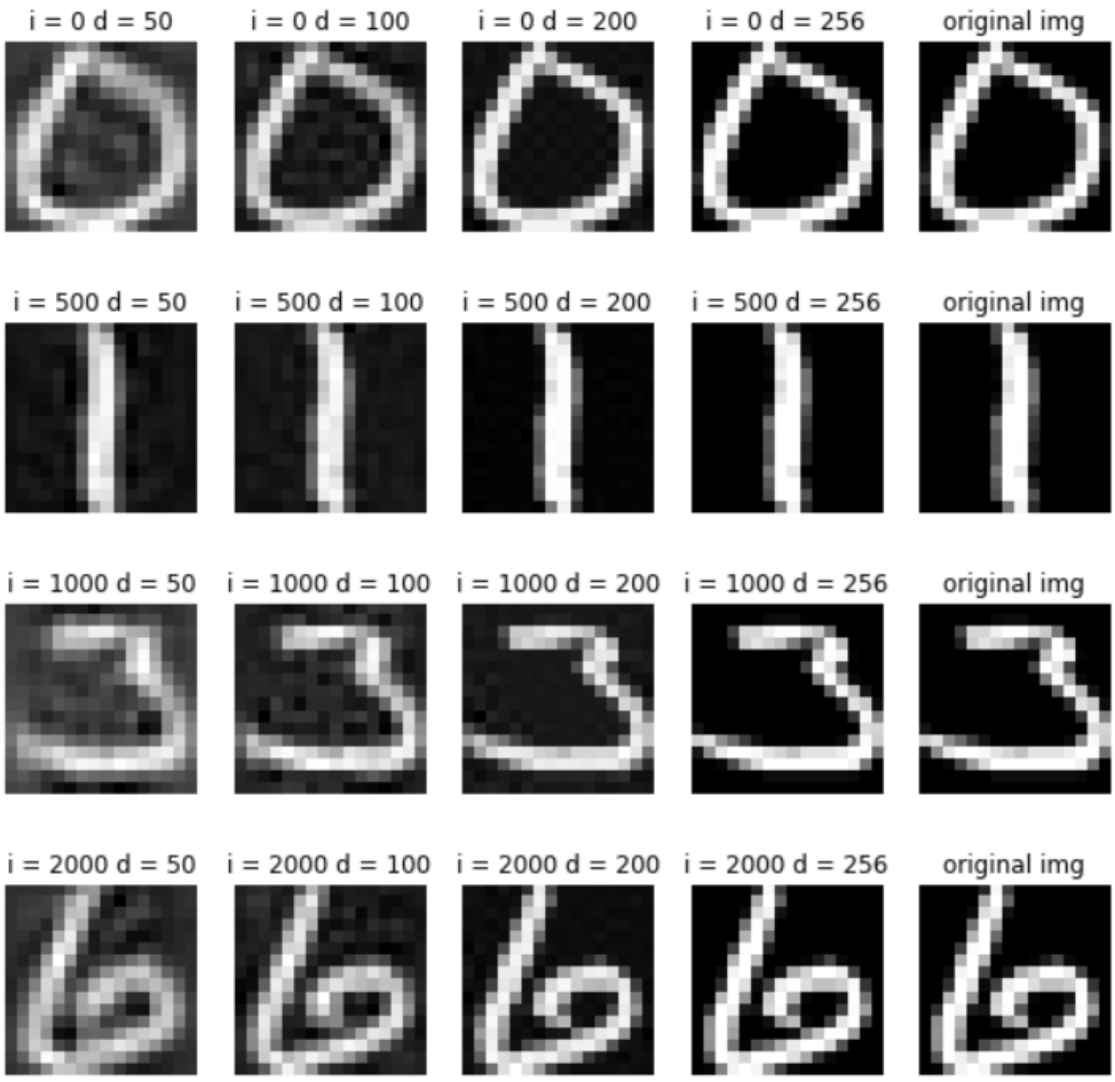

Figure 2.1: Reconstructed and Original Images

Bibliography

[1] Baytaş, İ., 2020. Clustering Slide.CMPE 462 Machine Learning Bogazici University.

[2] Baytaş, İ., 2020. Unsupervised Dimensionality Reduction Slide.CMPE 462 Machine Learning Bogazici University.