NAME: DUONG DUY CHIEN

K-means Clustering and Principal Component Analysis

I. Theory:

1. K-Means Algorithm

The K-Means Algorithm is the most popular and widely used algorithm for automatically grouping data into coherent subsets.

1. Randomly initialize two points in the dataset called the cluster centroids.
2. Cluster assignment: assign all examples into one of two groups based on which cluster centroid the example is closest to.
3. Move centroid: compute the averages for all the points inside each of the two cluster centroid groups, then move the cluster centroid points to those averages.
4. Re-run (2) and (3) until we have found our clusters.

2. PCA Algorithm

1. Data preprocessing
2. Compute “covariance matrix”
3. Compute “eigenvectors” of covariance matrix $\sum$
4. Take the first k columns of the U matrix and compute z

II. Results:

1. K-means

1.1 Implementing K-means

After finding closest centroids and Computing centroids mean, I can see the result after 10 iterations on the example dataset as shown in Fig 1

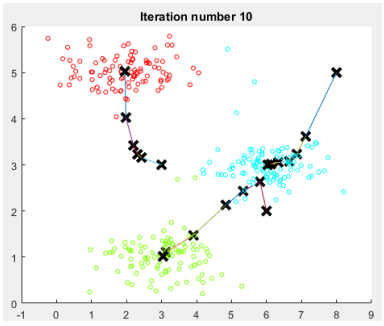

Figure 1 The output of K-means on example dataset

Because K-means algorithm is often stuck at local minimum. In practice a good strategy for K-means to approach global minimum by random initialization.

1.2 Image compression with K-means

The original image includes RGB color, in the range from 0-255 (use 8 bit to represent). Now, we apply K-means algorithm to group all color into 16 groups of color (now we just need to use 4 bits to represent a color component).

The original image required 24 bits for each one of the 128_128 pixel locations, resulting in total size of 128x128x24 = 393,216 bits. The new representation requires some overhead storage in form of a dictionary of 16 colors, each of which require 24 bits, but the image itself then only requires 4 bits per pixel location. The final number of bits used is therefore 16x_ 24 + 128 x128x4 = 65, 920 bits, which corresponds to compressing the original image by about a factor of 6. The result shown in Fig2

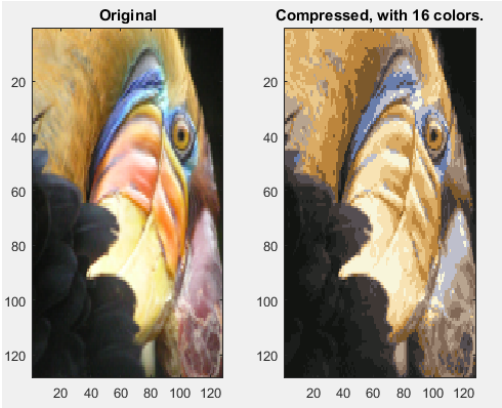

Figure 2 The result of compressed image

2. Principal Component Analysis

2.1 Implementing PCA

In order to find the principal components, I perform feature normalization then find the covariance matrix of the data. Final apply SVD to find the Eigen vector and Eigen value, to find the principle components of data. The result shown in figure 3.

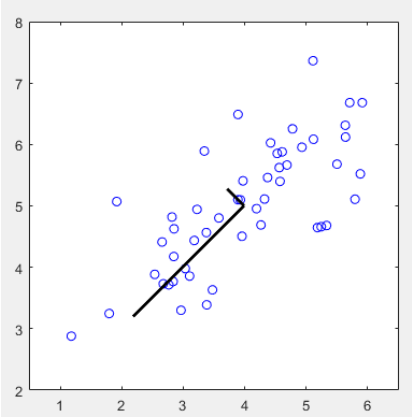

Figure 3 Computed eigenvectors of dataset

2.Dimensionality Reduction with PCA

We can reduce the feature dimension of our dataset by projecting each example onto a lower dimensional space which is our principle component (as shown in figure 4)

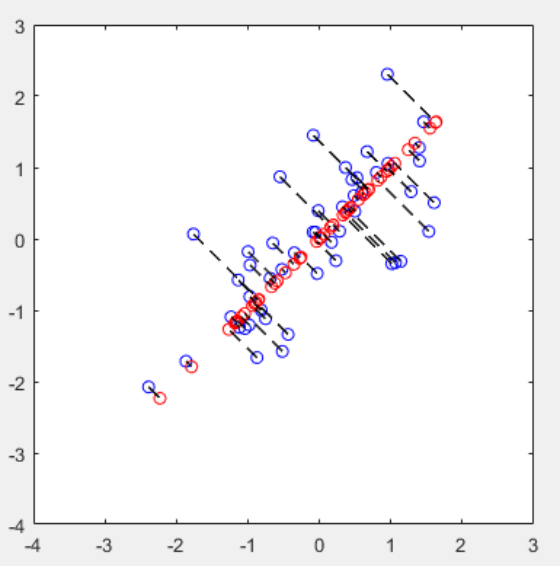

Figure 4 The normalized and projected data after PCA

We can approximately recover the data by projecting them back onto the original high dimensional space. My result after projecting back the first example

```
Approximation of the first example: -1.047419 -1.047419
```

2.3 Face Image Dataset

In this part, I will apply PCA to reduce the original 1024 dimensions of image onto only first 100 principal components (as shown in figure 5). This is a remarkable reduction (more than 10x).

Figure 5 Principal components on the face dataset

In order to understand what is lost in dimension reduction, I will recover the data using only the projected dataset then compare with original dataset (as shown in figure 6)

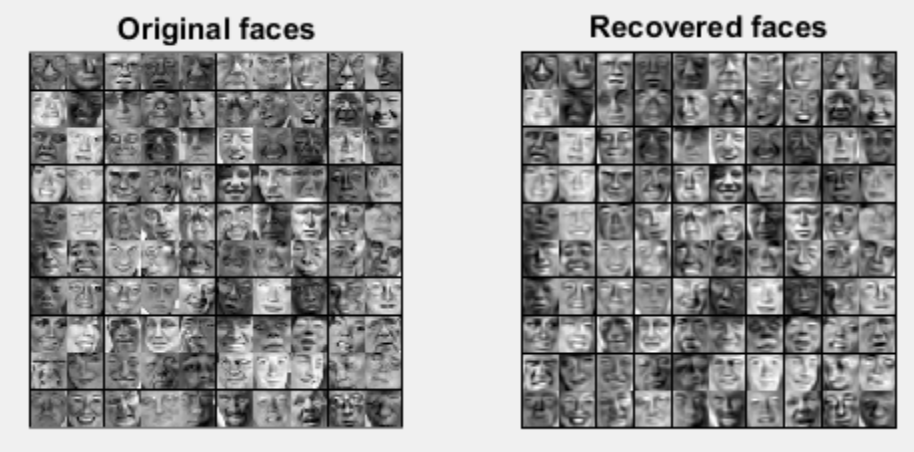

Figure 6 Original images of faces and ones reconstructed from only the top 100 principal components

2.4 PCA for visualization

PCA is often used to reduce the dimensionality of data for visualization purposes. The Figure 6 and 7 shown the result after project 3D space into the "best view" 2D

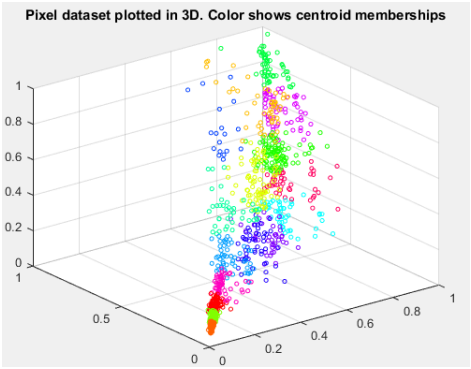

Figure 7 Original data in 3D

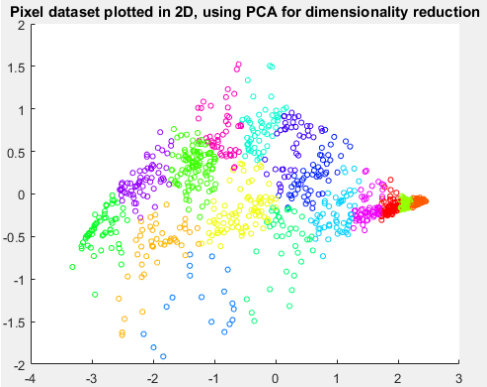

Figure 8 2D visualization produced using PCA