|  |
| --- |
| CSE 440 |
| Assignment 8 |
| Coursera – week 8 |

Submitted by

Mahbuba Tasmin

1610064042

Submitted to

M Ehsanul Karim

Week 8

Objective : In this exercise, we will implement the K-means clustering algorithm and  
apply it to compress an image. In the second part, we will use principal  
component analysis to find a low-dimensional representation of face images.

First part : We will first start on an example 2D dataset that will help you gain an intuition of how the K-means algorithm works. After that, we willl use the K-means algorithm for image compression by reducing the number of colors that occur in an image to only those that are most common in that image.

Intuition : The K-means algorithm is a method to automatically cluster similar data  
examples together. Concretely, we are given a training set fx(1); :::; x(m)g(where x(i) 2 Rn), and want to group the data into a few cohesive \clusters".  
The intuition behind K-means is an iterative procedure that starts by guessing the initial centroids, and then refines this guess by repeatedly assigning  
examples to their closest centroids and then recomputing the centroids based  
on the assignments.

1st step : finding the closest centroid

In the \cluster assignment" phase of the *K*-means algorithm, the algorithm  
assigns every training example *x*(*i*) to its closest centroid, given the current  
positions of centroids.

Each centroid defines one of the clusters; in each step, each data point is assigned to it’s nearest centroid , based on the squared Euclidean distance.

Closest centroids for the first 3 examples : 1 3 2( as expected output).

2nd step : Centroid update step

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid’s cluster. The algorithm iterates between steps one and two until a stopping criteria is met(i.e. no data points change clusters, the sum of the distance is minimized, or some maximum number of iterations is reached).

Centroids computed after initial finding of closest centroids:

2.428301 3.157924

5.813503 2.633656

7.119387 3.61666

K means clustering on example dataset

Throughout 10 iterations on the given dataset, the k means code will produce a visualization that steps through the progress of the algorithm ; how each step of the K- means algorithm changes the centroids and cluster assignments.

From step 1 to 9 , repeated assignment of centroid and update of centroid points.

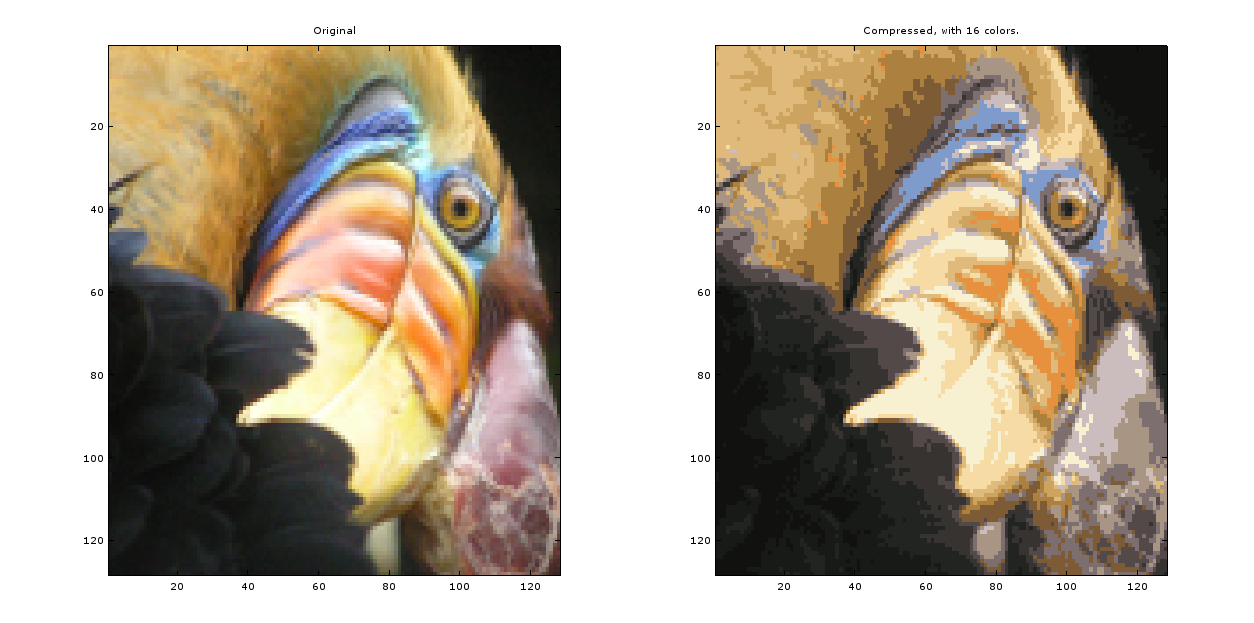
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  | | |

End result after 10 iteration on the dataset

**Image compression with K means :**

In this part , k- means will be applied in image compression . From an image that contains thousands of colors, the number of colors will be reduces to k( any real number ) . Hence, K- means algorithm will be used to select the k number of colors that will be used to represent the compressed image. As in , each pixel in the original image will be treated as a data example and k- means algorithm will be used to find the best k number of colors that best group (cluster) the pixels in the 3 dimensional RGB space. Once the clusted centroids are calculated, the K number of colors will replace the pixels in the original image.

Before using k –means algorithm, centroids are initialized randomly since k – means tends to perform better when centroids are seeded in such a way that doesn’t clump them together in a space.



For K = 16; max\_iters = 10; the output comes as above:

Some of the colors are discarded in the compressed image apparently , which is the result of k – means algorithm ; they are clustered in 16 group of colors from thousands of colors in the original image.

For k = 50 , max\_iters = 100; the output is as below :

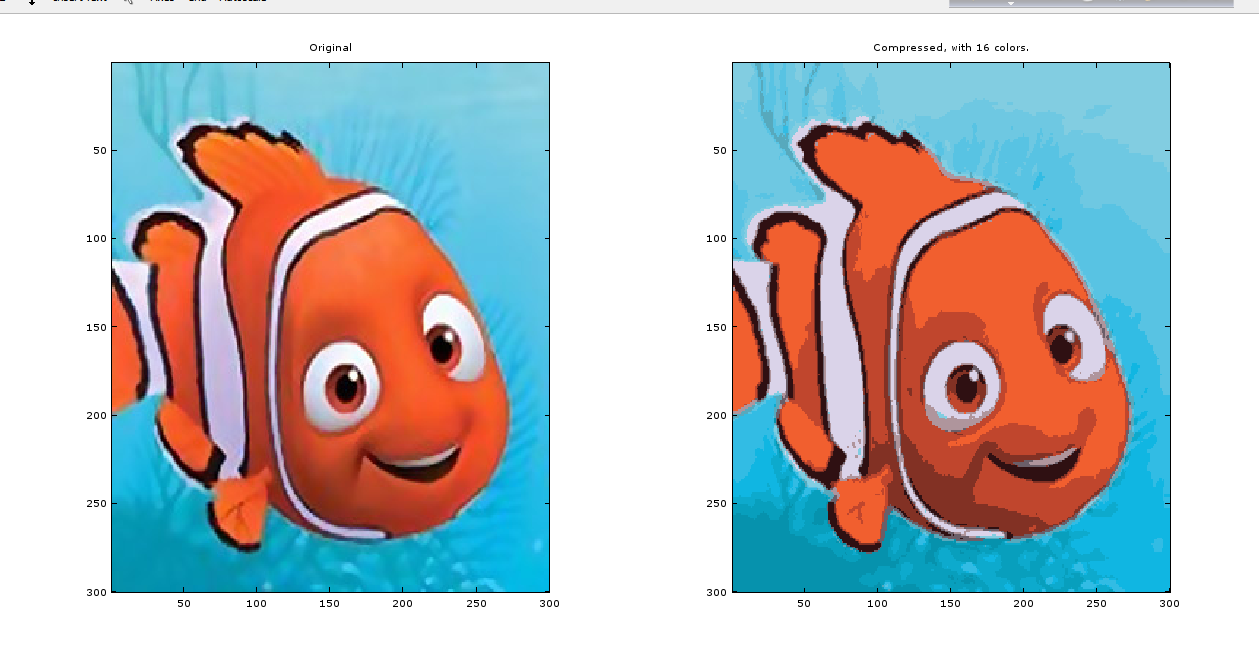


Since the number of K is greater here, hence the variation of color in the compressed image is also greater than the previous one.

Using own image (optional task ):

Here , I have used an image which is larger than the supplied image. Hence , the k –means algorithm gets slow even though K and max\_iters remain same.

For K = 16; max\_iters = 10; the output comes as below :

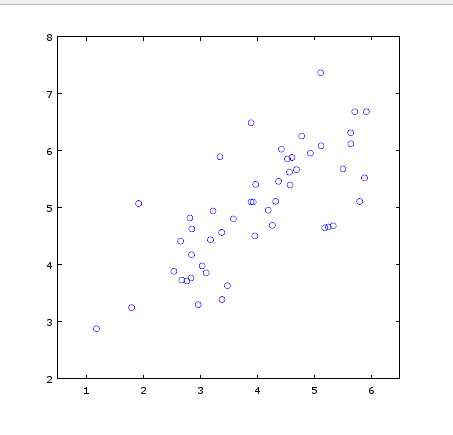


Part 2 : Principal Component Analysis

Objective : In this exercise, principal component analysis (PCA) will be used to perform dimensionality reduction. We will first experiment with an example 2D  
dataset to get intuition on how PCA works, and then use it on a bigger  
dataset of 5000 face image dataset.

Visualizing dataset :

To understand work mechanism of PCA, let us first start with a 2D dataset which has one direction of large variation and one of lesser variation.



Implementing PCA:

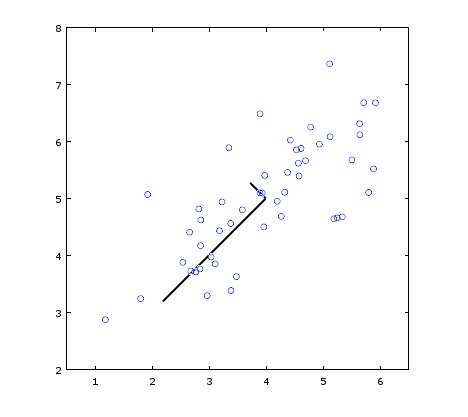
PCA consists of two computational steps; first , the covariance matrix of the data will be computed and then, octave’s SVD function will compute the eigenvectors U1,U2,…..Un, these will correspond to the principal components of variation in the data.

Figure : Computed Eigenvectors of the dataset

**Dimensionality Reduction with PCA, Projection and Reconstruction of the data :**After computing the principal components, it can be used to reduce the  
feature dimension of given dataset by projecting each example onto a lower  
dimensional space, (e.g., projecting the data from 2D to 1D of the original data.).  
Specifically, given a dataset X, the principal components U, and the desired number of dimensions to reduce to K. Each example should be projected in X onto the  
top K components in U. After projecting the data onto the lower dimensional space, the data can approximately be recovered by projecting them back onto the original high  
dimensional space.

Visualizing the projection :

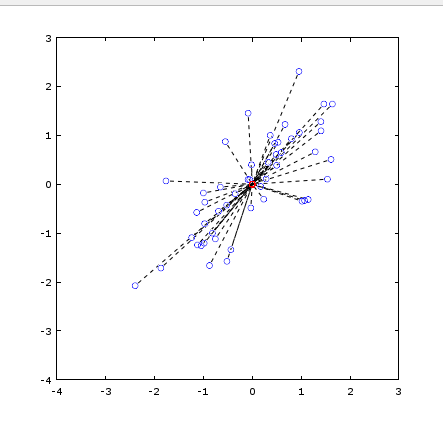
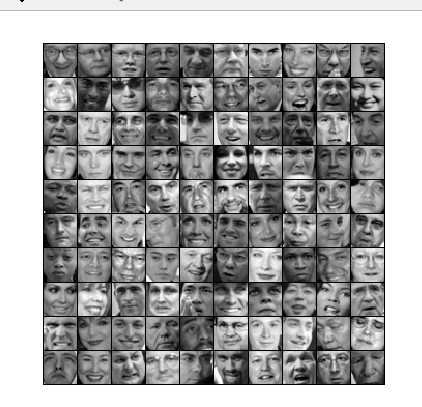


Figure : The normalized and projected data after PCA

In Figure , the original data points are indicated with the blue circles, while the projected data points are indicated with the red circles. The projection effectively only retains the information in the direction given by *U*1

Face image dataset :

PCA will be applied on face images to see how it can be used in practice for dimension reduction.



To run PCA on the face dataset, we first normalize the dataset by subtracting the mean of each feature from the data matrix X. After running PCA, we will obtain the principal components of the dataset.

**Dimensionality Reduction**Now that we have computed the principal components for the face dataset,we can use it to reduce the dimension of the face dataset. This allows to use learning algorithm with a smaller input size (e.g., 100 dimensions) instead of the original 1024 dimensions.

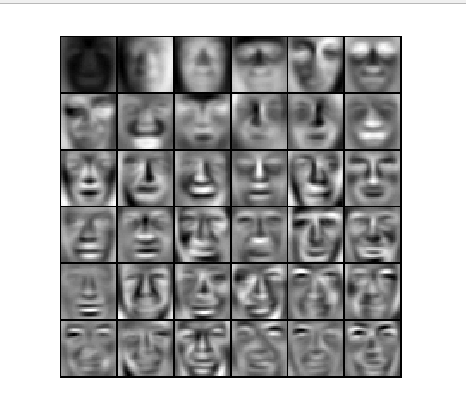


Figure : Principal components on face dataset(36)



Figure : Principal components on face dataset(100 )

From the reconstruction, we can observe that the general structure and appearance of the face are kept while the fine details are lost. This is a remarkable reduction (more than 10*×*) in the dataset size that can help speed up learning algorithm significantly.

|  |  |
| --- | --- |
|  |  |

Optional task : PCA for visualizing data

To facilitate better visualizing data, often PCA is used to reduce dimension of data in 3d dataset into 2d form , though in the cost of losing some information.