Machine Learning HW4

Gaumart Siméon . 0845209

Problem 1 :

a)

We train a kmean algorithm using scikit learn library on the mnist training set of 60000 images. We use 10 clusters (number of different labels in mnist data set).

Then we calculate the score (completeness score and homogeneity score) with comparing the real labels of the training set and the labels given by kmean during the training (because kmean is a unsupervised machine learning clustering algorithm).

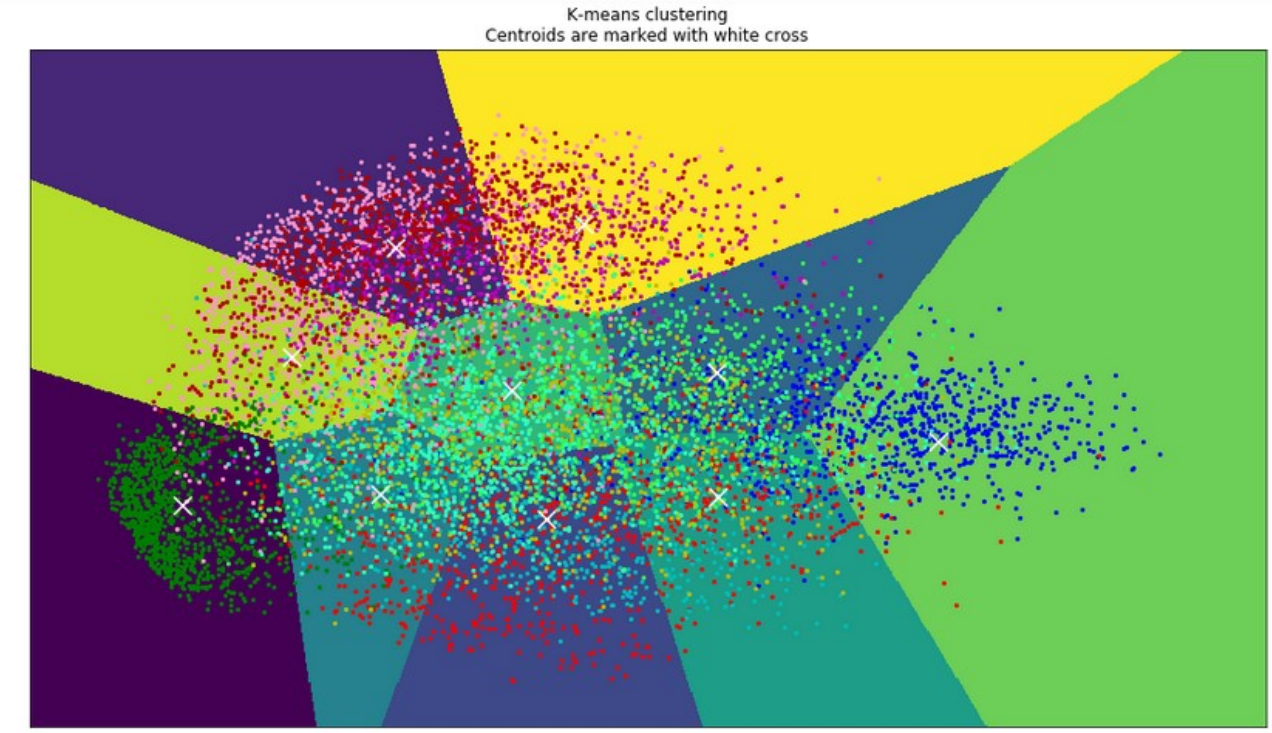
I find that homogeneity score is around 0,499 (49,9%)
and completeness score is around 0,503 (50,3%).

b)

Now we want to see the clustered data distribution and the original test data distribution. For this we have to reduce data features from 784 to 2 (to be able to plot data on a 2D graph).

Then we train a kmean algorithm on reduced training data (but where accuracy is lower) and plot the boundary of each cluster and the centroids. Then we add to the plot the reduced test data distribution (with plotting each data) and giving a color to each depending on its real label.

We got the following graph :



Problem 2 :

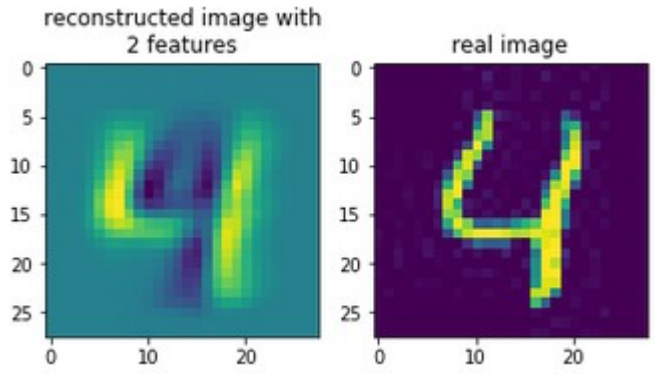
a)

We write our own PCA (reduction to k) with following these steps :
#1 : take X the original matrix with each data (line) and their features (column).
#2 : make Xc = X – Xmean where Xmean is the mean of X by column.
#3 : calculate the covariance Matrix : Mcov = (1/m)*Xc*Xct where Xct is the transpose matrix of Xc and m the number of column of X.
#4 : calculate the eigen values and eigen vectors of Mcov and take the real part (because it's complex numbers).
#5 : take the k eigen vectors where the eigen values are the highest
#6 : we return the k eigen vectors (in a matrix k_eig_vec) and we return X*k_eig_vec (matrix which represents the each data (line) and their k features after reduction (column)).

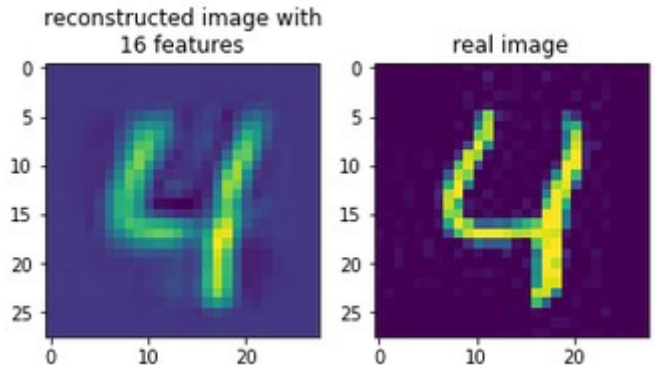
(PS : We return -k_eig_vec and -X*k_eig_vec to have the same rotation in question b) when we plot but the result is the same for reconstruction)

So the reconstructed images are :

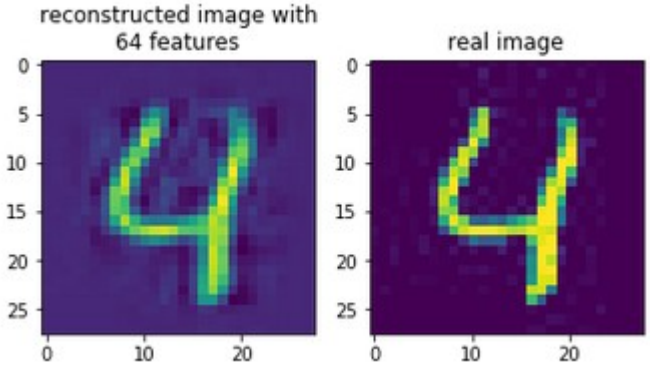
For dimension reduced to 2 :



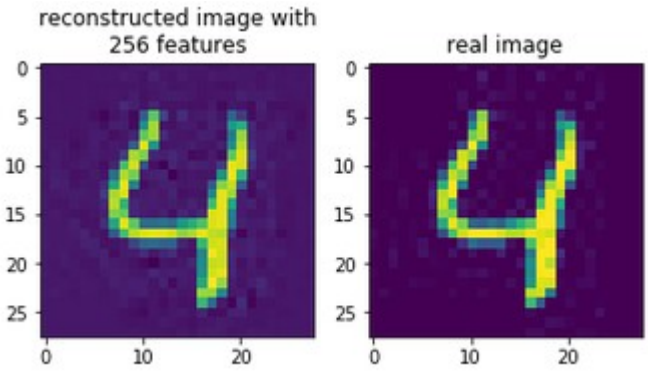
For dimension reduced to 16 :



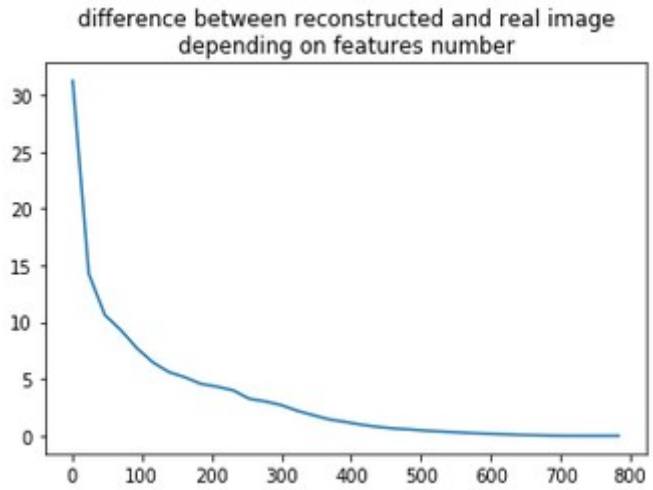
For dimension reduced to 64 :



For dimension reduced to 256 :

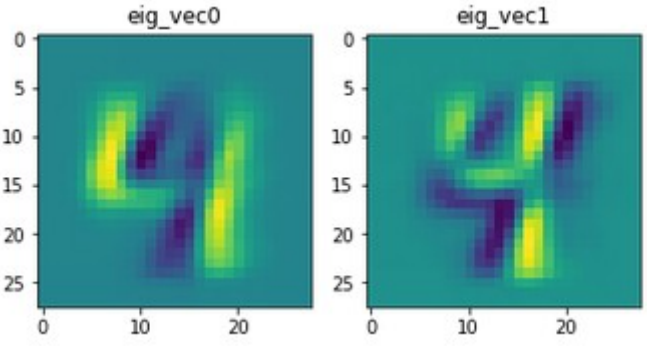


We can see that when we have more features, the reconstructed image is closer to the original one. Indeed, if we plot the difference (it takes the difference between each corresponding pixel (| pixel_k_im1-pixel_k_im2|) and divide by the number of pixels) between the real image and the reconstructed image in function of the number of features, we got this :



x axis is the number of features
y axis is the difference (difference of 0 mean that the images are the same).

We can explain this because, to reconstruct an image, we perform X*k_eig_vec*k_eig_vect where k_eig_vect is the transpose of k_eig_vec. And we take the line corresponding to the image in this matrix.
Indeed, each vectors in k_eig_vec is an image which best fit with the whole data set. It means k_eig_vec are k images which best represents the whole data set (and with no correlation between each vector). For example, if we show the eigen vectors for a reduction of 2, we get :



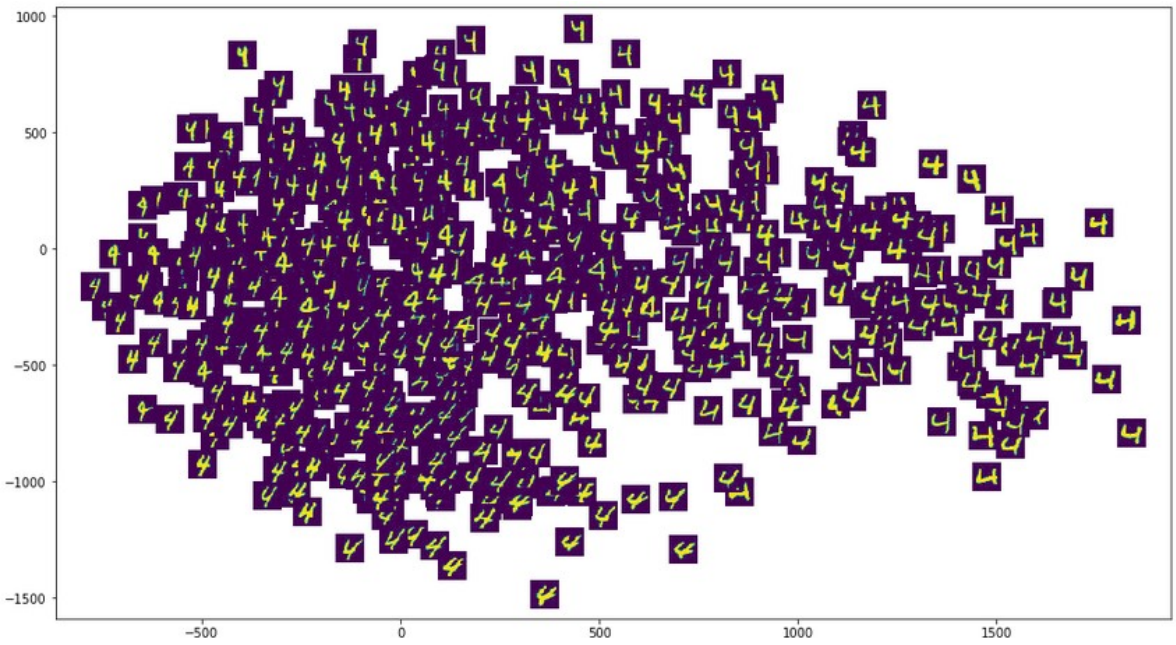
Furthermore, X*k_eig_vec represents a matrix where we have, for each image, the k features that best represent the image.

So to reconstruct the image, we perform a combination of the k eigen vectors, using the features as coefficient (ex with 2 features/eigen_vectors : reconstructed_image=feature0*eig_vec0+feature1*eig_vec1).

So the more eigen_vectors (and features so) we have, the more the reconstructed image will be close to the real image, because we add more precision to the reconstructed image.

b)

We perform PCA with a reduction to 2 dimension and plot the images where the first feature is x and the second feature is y:



c)

LDA is very useful to find dimensions which aim at seperating cluster, thus you will have to know clusters before. LDA is not neccesarily a classifier, but can be used as one. Thus LDA can only be used in supervised learning

PCA is a general approach for denoising and dimensionality reduction and does not require any further information such as class labels in supervised learning. Therefore it can be used in unsupervised learning.

However, both the algorithms tell us which attribute or feature is contributing more in creating the new axes, and both try to reduce the dimensions.

d)

(we don’t have the course for LDA so I don’t know the mathematical difference and correlation)