Lab 6 K-Means and PCA

1. **K-Means Clustering**

Overview of algorithm

1. Randomly choose K centroids
2. Calculate the distance of all instances to the K centroids and assign instances to closest centroid
3. Calculate new centroid for each of the K clusters
4. Repeat Step 2 and 3 until clusters' assignments are stable or centroids are not changing

The MNIST dataset is a dataset of 28×28 images of hand-written digits (http://yann.lecun.com/exdb/mnist/). To read these images in Python, you can use the following script.

```
from sklearn.datasets import fetch_openml
        mnist = fetch_openml('mnist_784', version=1)
X = mnist["data"]
```

Since the dataset is quite large, restrict yourself to the first 2000 training images. The data should be a 2000×784 matrix.

Requirements

a. Write a function **my_kmeans** to perform a k-means clustering of the 2000 images of digits.
b. Your function should take 3 arguments, the data matrix, the number of clusters K, and the number of initializations N.
    i. Your code should consist of 3 nested loops.
    ii. The outermost (from 1 to N) cycles over random centroids initializations (i.e. you will call k-means N times with different initializations).
    iii. The second loop is the actual k-means algorithm for that initialization, and cycles over the iterations of k-means.
    iv. Inside this are the actual iterations of k-means-- look for the convergence conditions as defined above. Each iteration can have 2 successive loops: the first assigns observations to each cluster and the second recalculates the means of each cluster.
c. Your function should return:

    (a) the K centroids and cluster assignments for the best solution with the lowest loss function (recall that the k-means loss function is the sum of the squared distances of observations from their assigned means)

    (b) the sequence of values of the loss-function over k-means iterations for the best solution (this should be non-increasing)

    (c) the set (of size N) terminal loss-function values for all initializations

d. Run your code on the 2000 digits for K = 10 and N = 15. Plot the sequence of values of the loss-function over all the iterations provided by the best k-means solution.

e. Plot the N terminal loss-function values for all initializations.

2. Principal Component Analysis (PCA)

Principal components analysis (PCA) is a data reduction technique that allows to compress high-dimensional data sets into very low dimensions.

With Scikit-Learn, PCA is really trivial. It even takes care of mean centering for you. Below is an example of reducing the data dimensionality to 2.

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
```

Requirements

a. Plot the 2000 MNIST digit images to the 2 and 3 dimensional spaces respectively after applying PCA.
b. Use “pca.explained_variance_ratio_” to see how much variances of the data have been explained by the principal components.

Constraints

Please submit your work in the form of a Jupyter Notebook.

Only Numpy, Matplotlib, and the functions explicitly outlined above are allowed.