Homework Assignment 4:
k-means and hierarchical clustering

Objective: The objective of this exercise is to understand how the unsupervised learning methods k-means clustering and hierarchical clustering work. Upon completing the exercise you should also understand how the choice of number of clusters, distance metrics and linkage functions can impact the solutions obtained and further be able to interpret dendrograms and measures of cluster validity.

Material: Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, section 8.1-8.3 and 8.5.7

Additional tools: For this homework, you need to install/load script that provided, namely *clusterPlot.py* and *clusterVal.py*

Important: The following points are how you hand-in the homework assignment:

- Provide clear and complete answers to the questions below (not hidden somewhere in your source code), and make sure to explain your answers / motivate your choices. Please make as PDF file.
- Source code, output graphs, derivations, etc., should be included, and zipped together with the PDF file.
- Hand-in: upload to Blackboard.
- Include name, student number, assignment (especially in filenames)
- When working in pairs both of you should upload the assignment, and report the name of your partner in the upload comments.
- For problems or questions: use the BB discussion board or email.

In previous exercises we considered supervised learning, i.e., we were given both input data $\mathbf{X}$ and output values $\mathbf{y}$. We now move on to unsupervised learning where we are only provided input data $\mathbf{X}$. The aim is here to find common patterns in the data such as groups of observations that are similar in some sense. In this exercise we will consider two clustering approaches for unsupervised learning: k-means clustering and hierarchical clustering.

4.1 *k*-means clustering

In this part of the exercise we will investigate *k*-means clustering. In *k*-means each of the data points are assigned to the cluster in closest proximity according to some measure of distance between cluster centers and data points. When the distance is given by the squared euclidian distance, the centers are also called as centroids. Once the data points have been assigned, each cluster center is updated to be placed at the center of the data points that are assigned to the cluster. This continues iteratively, usually until the assignment of data points to centers no longer change or until a maximal number of iterations is reached.

4.1.1 (Point = 2) Load the `synth1` data into Python using the `loadmat` function. Cluster the data into $K = 4$ clusters using the *k*-means algorithm. Make a scatter plot of the data and the clustering using the `clusterPlot()` function that will be provided.

Hints:

- In Python, you can use the function `k_means()` from the package `sklearn.cluster` to compute *k*-means clustering. Import the function and type `help(k_means)` to learn how to use the function.
- Type `clusterPlot(X,clusters,centroids,y)` to plot the data and the clustering.

Try also to cluster and plot the data sets `synth2`, `synth3`, and `synth4` with the same number of K. Does the clustering coincide with the true classes? Answer explicitly for every data set.

Rather than using the error rate we will consider the supervised measures of cluster validity described in *Introduction to Data Mining* section 8.5.7, in particular the entropy, purity, rand statistic, and Jaccard coefficient. Carefully review these measures in the book and make sure you understand how they are calculated.

4.1.2 (Point = 2) Repeat Exercise 4.1.1, but this time perform *k*-means clustering for $K = 1, \ldots, 10$ clusters. For each value of K compute the four cluster validity measures mentioned above. Plot the cluster validity measures as a function of K.

Hints:

- Use the function `clusterval` to compute the cluster validity. Read carefully the description of the function which is written as comments.

How can the cluster validity measures be used to select the best number of clusters? What happens when more than four clusters are used to model the data?

k-means clustering has many different applications, one of which is data compression. A data set can be compressed by performing k-means clustering and then representing each data object by its cluster center. Thus, the only data that need to be stored is the K cluster centers and the N cluster indices.

4.1.3 (Point = 2) We will consider a subset of the wild faces data described in [1]. Load the wildfaces data, `Data/wildfaces` using the `loadmat` function. Each data object is a $40 \times 40 \times 3 = 4800$ dimensional vector, corresponding to a 3-color 40×40 pixels image. Compute a k-means clustering of the data with $K = 10$ clusters. Plot a few random images from the data set as well as their corresponding cluster centroids to see how they are represented.

Hints:

- You can plot an image by the command `imshow(np.reshape(X[k,:],(c,x,y)).T)` which reshapes an image vector to a 3-dimensional array and plots it. Similarly, you can plot the cluster centroids.
- Running k-means on a large data set can be slow. If you type `k_means(X, K, verbose=True, max_iter=X, n_init=S)` it will provide information about the iterations as they run. `n_init` as before constrains the number of initial repeated centroid seeds. Type `help(kmeans)` to read more about these options.

How well is the data represented by the cluster centroids? Are you able to recognize the faces in the compressed representation? What happens if you increase or decrease the number of clusters?

4.1.4 (Point = 2) Repeat the exercise with the digits data set. Load the digits data set from `load Data/digits`. Each data object is a $16 \times 16 = 256$ dimensional vector, corresponding to a gray scale 16×16 pixels image.

Hints:

- You can change the color map to black-on-white grey-scale by adding the parameter `cmap=cm.binary` to the function `imshow()`.

Why does running k-means with $K = 10$ not give you 10 clusters corresponding to the 10 digits 0..9? How many clusters do you need to visually represent the 10 different digits? Are there any digits that the clustering algorithm seems to confuse more than others?

4.2 Hierarchical clustering

We will in this part of the exercise consider hierarchical clustering based on the functions from the package `scipy.cluster.hierarchy`. Function `linkage()` forms a

sample to sample distance matrix according to a given distance metric, and creates the linkages between data points forming the hierarchical cluster tree. Function `dendrogram` creates a plot of the generated tree. Function `fcluster` extracts the cluster from a linkage matrix w.r.t. a given criterion. Use `help` for the three functions and inspect what distance metrics and linkage functions are implemented.

4.2.1 (Point = 2) Load the data set from `Data/synth1`. Cluster the data using hierarchical clustering with single linkage using the Euclidean distance measure. Cluster the data into 4 clusters by cutting off the dendrogram at a threshold. Plot a dendrogram and a scatter plot of the clusters.

Hints:

- The function `linkage()` computes the hierarchical clustering, resulting in a matrix representing the hierarchy of clusterings. Type `help(linkage)` to learn how to use it.
- You can e.g. type `Z = linkage(X, method='single', metric='euclidean')` to use single linkage with the Euclidean distance measure.
- To compute a clustering, you can use the function `fcluster()`. For example, type `cls = fcluster(Z, criterion='maxclust', t=4)` to get a maximum of 4 clusters. Type `help(fcluster)` to learn more about what this function does.
- To plot a dendrogram, you can use the `dendrogram()` function.
- Again, you can use the function `clusterplot()` to plot a scatter plot of the clustering.

Changing the linkage methods (single, complete, average) and explain how the changes of the dendrogram. Apply also (plot a dendrogram and a scatter plot of the clusters) to the data sets `synth2`, `synth3`, and `synth4`.

References

[1] Tamara L Berg, Alexander C Berg, Jaety Edwards, and DA Forsyth. Who's in the picture. *Advances in neural information processing systems*, 17:137–144, 2005.