EPFL
ENAC TRANSP-OR
**Prof. M. Bierlaire**
Gael Lederrey & Nikola Obrenovic

Decision Aids
Spring 2018

# DATA MINING I - CLUSTERING - EXERCISES

## Exercise 1

In this exercise, you will implement the k-means clustering algorithm. You can find the main pseudo-code for the algorithm in Algorithm 1. You will also find other functions that are used by Algorithm 1 in Algorithms 2 and 3.

**Algorithm 1:** k-means clustering

**Result:** Compute the $k$ clusters using k-means.
**Input :** *data, k, maxIters, threshold*
**Output:** *centroids, assignments*

Initialize the clusters by randomly selecting $k$ samples as init clusters
Initialize an empty list to store the losses: *loss*
**foreach** $i \leftarrow 1$ ***to*** *maxIters* **do**
    `/* See Algorithm 2 for pseudo-code */`
    Compute the updated k-means parameters
    Compute the average loss and add it the list *loss*
    `/* Stopping criterion. Be careful when you have only one value in` $loss$`. */`
    **if** $|loss_i - loss_{i-1}| < threshold$ **then**
        break
    **end**
    Update the centroids
**end**

**Algorithm 2:** Update k-means parameters

**Result:** Update the centroids, compute the loss and the new assignments.
**Input :** *data, old_centroids*
**Output:** *losses, assignments, new_centroids*

`/* See Algorithm 3 for pseudo-code */`
Build the distance matrix between the centroids and the samples
Compute the assignments bases on minimum distance between samples and centroids
**foreach** $k \leftarrow 1$ ***to*** *#clusters* **do**
    Get the indices corresponding to all samples assigned to cluster $k$
    Compute the new centroid of cluster $k$ using the average position of all samples belonging to cluster $k$
**end**

**Algorithm 3:** Build Euclidian distance matrix

**Result:** Build a distance matrix between all samples and centroids
**Input :** *data, centroids*
**Output:** *distance_matrix*

```
Create an empty matrix D
foreach k ← 1 to #clusters do
    Compute the sum of square between all the points in data and the centroid of cluster k
    Add it to the matrix D;
end
Make sure the matrix D has the right shape.
```

Here are the steps you need to follow to implement the k-means clustering algorithm:

1. Run the file `exercise0.m` and analyze the data that we are giving you. More information on this dataset can be found on Wikipedia. Choose the dimensions along which you would like to cluster and add them in the file `exercise1.m` on lines 11-12.

2. Code the different parts of the k-means algorithm in the following files in the directory `functions`:

   - `intialize_cluster.m`
   - `build_distance_matrix.m`
   - `update_kmeans_parameters.m`
   - `kmeans.m`

   Do not forget to define the different parameters for the k-means clustering on lines 18 to 21 in the file `exercise1.m`. Test your implementation by running this file.

   *Hints:*

   - *You should code the files in the order they are given above.*
   - *It can be helpful to test the functions you're implementing. So, do not hesitate to create new scripts to test your implementations.*

3. Test your implement against ours using the function `kmeans_vizu`. (The code is not shared, but you can call the function.) This function is also used to show how the clusters are being updated. To do so, comment line 24, uncomment line 33 and run `exercise1.m`

## Exercise 2

In this exercise, you will implement the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm. You can find below the pseudo-code in Algorithm 4.
Here are the steps you need to follow to implement the DBSCAN algorithm:

1. Report the dimensions you used from Exercise 1 in the file `exercise2.m` on lines 11-12.

2. Implement the algorithm based on the pseudo-code given earlier.

   *Hint: It may be useful to have a look at your course material and on Internet to find more information about this algorithm and how it should behave.*

3. Test your algorithm by running the file `exercise2.m`. But first, you will have to find the two parameters for this algorithm: $\varepsilon$ for the range and the minimum number of neighbours $N_{min,neigh}$ to be a cluster point. Here is how to do it:

**Choice of $N_{min,neigh}$:** A low number for this parameter means that DBSCAN will create more clusters. Aside from this information, the choice is kind of arbitrary or expert-needed. Play with it to see how the clusters are computed based on this parameter.

**Choice of $\varepsilon$:** You can use multiple methods to choose this algorithm. One of them is the *k-distance plot*. The code is provided for the exercises. Indeed, in a clustering with $N_{min,neigh} = k$, we expect that the core points will all be in a certain range while noise points can be much further away. Therefore, you will observe a **knee point** in the k-distance plot, as shown in Figure 1 (Sometimes, you will not find any knee point, therefore, you will have to use other methods.)

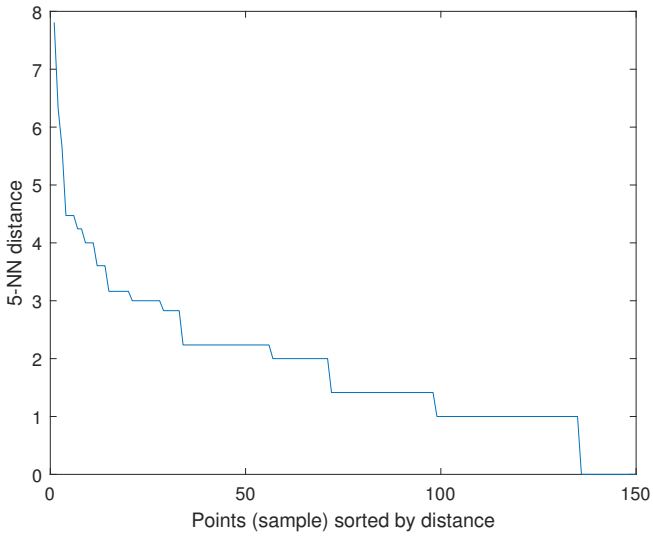


Figure 1: k-distance plot for $k = N_{min,neigh} = 5$. The knee point is around 2-2.5. But other values are also acceptable since it is not a very clear knee point.

To run the k-distance plot, you can uncomment line 23 in the file `exercise2.m` and run it.

4. Compare the results of this algorithm with the results you obtained with the k-means algorithm. What are the main differences? Which one do you think is better on this dataset and why?

**Algorithm 4:** DBSCAN

**Result:** Compute clusters and noise based on two parameters: $\varepsilon$ and $N_{min,neigh}$
**Input :** *data,* $\varepsilon$, $N_{min,neigh}$
**Output:** *cluster_labels* (for all the points, a label is given)

Initialize `classified`, a vector of zeros containing the information if a point has been classified or not.
Initialize `nbr_neigh_eps`, a vector of zeros for the number of neighbours to a sample within range $\varepsilon$.
Initialize `clust_labels`, a vector of NaN's to give the nature of the cluster for each sample.
Initialize `num_clust` the integer for the current cluster.
Compute the matrix `D` of Euclidian distance between all the points. **foreach** $i \leftarrow 1$ *to #Samples* **do**
    `/* You start coding from here. The initialization is given to you. */`
    **if** *i is not classified* **then**
        Use the matrix `D` to **find** all points that are in the range $\varepsilon$ of sample $i$.
        Update the vector `nbr_neigh_eps` with the number of neighbours to $i$. (Without taking i into account)
        **if** *nbr_neigh_eps(i)* $< N_{min,neigh}$ **then**
            `/* This sample is a noise */`
            This point is now classified and its label is 0.
        **else**
            `/* This sample belongs to a cluster */`
            Update the current cluster and update the label of sample $i$. In addition, this point is classified.
            Create an array of neighbours `neigh` to $i$ without $i$ inside.
            **while** *neigh is not empty* **do**
                Get the first neighbour, call it $j$, in the list `neigh`, find all of its neighbours that are in the range of $\varepsilon$ and compute the number of neighbours for this point.
                **if** *nbr_neigh_eps(j)* $< N_{min,neigh}$ **then**
                    `/* j is also a core point. We need to do the same thing as for i. */`
                    Find all the unclassified neighbours of $j$ and add this array to the array `neigh`.
                    Get the neighbours of $j$ that were considered as noisy
                    Change the label of the unclassified neighbours and noisy ones to be the same as $j$ and $i$ and say that all the neighbours of $j$ are classified.
                **end**
                Remove the first element of `neigh`.
            **end**
        **end**
    **end**
**end**

## Exercise 3

In this exercise, you do not have to implement anything. In the file `exercise3.m`, we are comparing the two algorithms you've implemented as well as the Hierarchical clustering algorithm implemented in Matlab on different datasets. You can find an example how it is used on the Mathworks website.

Once you've implemented the first two exercises, you can run the file `exercise3.m`. This will give you 4 figures:

- Figure 1 shows you the artificial data composed of non-linear clusters as well as some noise.
- Figure 2 shows you the results of the k-means algorithm with $k = 6$.
- Figure 3 shows you the results of the Hierarchical clustering.
- Figure 4 shows you the results of the DBSCAN algorithm.

Your task is to understand the behaviour of these algorithms on this particular dataset and answer the following questions:

1. In your opinion, which algorithm performs better? And why?
2. Compared to the results of exercises 1 and 2 (as done in point 4 of Exercise 2) where k-means and DBSCAN were clustering on the Iris dataset, do you observe the same ranking between the two algorithms, *i.e.* are there any differences with your answer from point 4 of Exercise 2? When you try to rank algorithms, you can use the accuracy as the metric. Were you expecting the results you got? If no, why? Can you think of a way to fix the ranking and get the same on both datasets?
3. If you encounter a new dataset, which of these three algorithms will you use first, and why?