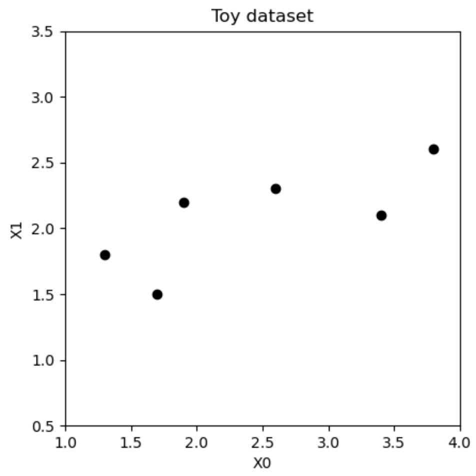
INF264 - Exercise 6

Pierre Gillot **Natacha Galmiche**[*]

Week 40 - 2021

1 Clustering techniques

In this first section, we get familiar with two different clustering algorithms, namely K-means and hierarchical clustering. For both clustering tasks, we consider the following toy dataset consisting of 6 two-dimensional points:



X_0	X_1
1.7	1.5
1.3	1.8
1.9	2.2
2.6	2.3
3.4	2.1
3.8	2.6

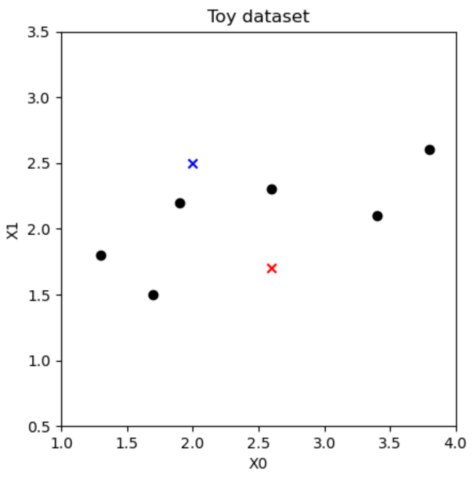
1.1 K-means

In K-means, the task is to split the data into k cluster, where k is fixed and given. Each cluster is represented by its centroid defined as the mean of the data points forming the cluster. A possible heuristic for K-means (implemented in section 2) is to iteratively refine the clusters as follows:

[*]Not a TA for this course this year.

0. The centroids are randomly initialized.
1. Each data point is assigned to the cluster whose centroid is the closest (we consider the euclidian norm).
2. Centroids are updated as the mean of the data points in their corresponding cluster.
3. Steps 1 and 2 are repeated until the stopping criterion is met.

We now simulate an update of K-means with $k = 2$ on the toy dataset. We assume the blue and red centroids were randomly initialized as $(2.0, 2.5)$ and $(2.6, 1.7)$ respectively, as depicted below:



Answer the following questions on paper:

1. Perform two iterations of K-means on the toy dataset. You can draw the data points in red or blue to visualize their assigned cluster.
2. Draw the likely localization of the two centroids and their respective cluster after convergence.
3. Assume $k = 3$, draw the likely localization of the three centroids and their respective cluster after convergence.

1.2 Hierarchical clustering

In hierarchical clustering, the objective is to produce nested clusters organized in a hierarchy tree. There are 2 types of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). In this exercise we will do an agglomerative hierarchical clustering using single linkage. This algorithm is then as follows:

1. Compute pairwise distances between all data points in the dataset D: $Dist(x, z) \quad x, z \in D$
2. Put every point in a separate cluster D_i
3. Initialize the linkage matrix as the pairwise distance matrix $L_{single} = Dist$
4. As long as there are more than one cluster:
   (a) Find the 2 closest clusters $\text{argmin}_{x \in D_i, z \in D_j} L_{single}(D_i, D_j)$
   (b) Merge them into one cluster $D_{new} = D_i \cup D_j$
   (c) Update L_{single} accordingly, i.e:

$$L_{single}(D_i, D_j) = \min_{x \in D_i, z \in D_j} Dist(x, z)$$

Answer the following questions on paper:

1. Compute the pairwise distances
2. Apply the agglomerative hierarchical clustering using single linkage to the toy dataset
3. What are the successive dimensions of L_{single}?
4. Visualize your nested clusters using Venn diagrams
5. Visualize your nested clusters using dendograms (the y axis is the distance between the merged clusters and the x axis represents the different datapoints)

2 Lloyd's algorithm

In this second section, we will implement the famous Lloyd's algorithm, a well known heuristic solving K-means. The purpose of the algorithm is to partition a set of observations into k clusters, where an observation should belong to a cluster if this cluster is the closest from the observation. To evaluate how far an observation is from a cluster, the distance between the observation and the centroid of the cluster is measured using a Euclidian norm.

It is known that this partitioning problem is difficult (in fact it is NP-hard even with only 2 clusters and even in the plane), but there exist heuristics that converge quickly to a local minimum. Lloyd's algorithm is one such heuristic which, while a bit "naive", has been widely used and successfully modified into efficient algorithms achieving state-of-the-art performance in clustering problems.

The pseudo-code for Lloyd's algorithm is as follows:

Lloyd's algorithm

Input:

- A dataset $\mathcal{D}$ represented by a matrix X of size $n \times d$, consisting of n observations in $\mathbb{R}^d$
- An integer $k \geq 1$, representing the number of clusters
- An integer $r \geq 1$, representing the number of restarts
- A float $p \geq 0$, determining the precision of the optimization (defaults to 0)

Output:

- A partition of $\mathcal{D}$ into k clusters $\mathcal{D}_1, \dots, \mathcal{D}_k$ represented by their centroids $\mu_1, \dots, \mu_k$, where an observation $x_i = X[i,:]$ belongs to the cluster $\mathcal{D}_l$ if μ_l is the closest centroid to x_i
- The cost of the partition: $\frac{1}{n}\sum_{j=1}^{k} \sum_{x \in \mathcal{D}_j} \|\mu_j - x\|_{L_2}^2$

```
1. For restart = 1 : r
    2. Initialize $k$ clusters $\mathcal{D}_1, \dots, \mathcal{D}_k$, i.e. create $k$ centroids $\mu_1, \dots, \mu_k$ and set them to $k$ distinct observations in $X$ picked randomly
    3. While |previous_cost − new_cost| $> p$
        4. Find the new partition, i.e for each observation $x_i = X[i,:]$ find the cluster $\mathcal{D}_l$ whose centroid $\mu_l$ is the closest to $x_i$, that is such that $l = \underset{j}{\operatorname{argmin}} \|\mu_j - x_i\|_{L_2}^2$, then place $x_i$ inside $\mathcal{D}_l$
        5. Compute the cost of the new partition: $\frac{1}{n}\sum_{j=1}^{k} \sum_{x \in \mathcal{D}_j} \|\mu_j - x\|_{L_2}^2$
        6. If |previous_cost − new_cost| $> p$ update the clusters, i.e. for each cluster $\mathcal{D}_j$, update its centroid $\mu_j$ to become the mean of the observations assigned to this cluster, that is $\mu_j \leftarrow \underset{i}{\operatorname{mean}}\{x_i \in \mathcal{D}_j\}$
    End
    7. If the cost is smaller than all the costs obtained in previous restarts, store the clusters and the cost
End
8. Return the stored clusters and cost
```

Once you have carefully read this pseudo-code, do the following:

1. Implement Lloyd's algorithm. **You can follow exactly the above pseudo-code, but this is not mandatory. You will find many other pseudo-codes on the internet**.
2. Load the Iris dataset using the 'sklearn.datasets.load_iris' function. Store the three features in a matrix X and the labels in a vector y.

3. Perform clustering on X, for different values of k. Note that for this unsupervised learning task, we do not split X into train/validation/test sets, nor do we do cross-validation.

4. Plot the "elbow graph", that is the curve of the cost as a function of the number of clusters k. You can set the number of restarts to 5 and the precision to 0.

5. The obtained curve should have an elbow shape. A common heuristic is to select the value of k at the angle of the elbow. Based on your curve, which value of k would you select ? Is this value of k close to the number of distinct classes in the target labels y ?

6. Perform once more clustering on X with $k = 3$ (i.e. create as many clusters as there are classes in the Iris dataset). How well does the obtained partition $\mathcal{D}_1, \ldots, \mathcal{D}_3$ matches the target labels y ?