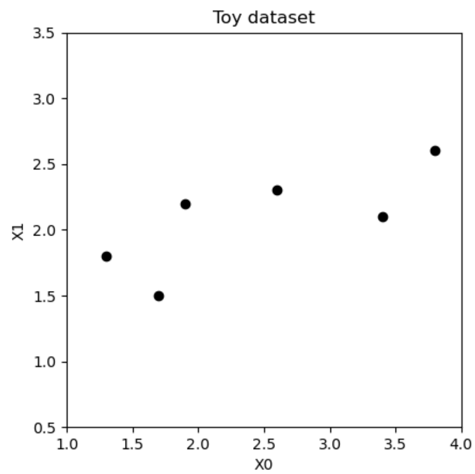
INF264 - Exercise 6

Natacha Galmiche

1 Clustering techniques

In this section, we get familiar with hierarchical clustering. 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 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 $argmin_{x \in D_i, z \in D_j} L_{single}(D_i, D_j)$
   (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 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 Euclidean 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:

Algorithm 1: Lloyd's algorithm

Input: A matrix `X` $\in \mathbb{R}^{n \times d}$ representing a dataset $\mathcal{D}$ of n observations $x_{i=1..n}$ in $\mathbb{R}^d$
Input: An integer `k` ≥ 1, representing the number of clusters
Input: An integer `restarts` ≥ 1, representing the number of restarts
Input: A float `precision` ≥ 0, determining the precision of the optimization
Output: `k` centroids, the corresponding partition of `X`, and the cost of the partition

---

for $r = 1..\mathit{restarts}$ **do**
  Initialize `k` centroids $\mu_1, \dots, \mu_k$ by randomly picking `k` distinct observations x_i from `X`;
  Partition `X`, i.e. for each observation $x_i =$ `X`$[i,:]$:
    a) find the cluster $\mathcal{C}_l$ whose centroid μ_l is the closest to x_i (i.e. $l = \underset{j}{\operatorname{argmin}} \|\mu_j - x_i\|_2^2$)
    b) append x_i to $\mathcal{C}_l$;
  Compute cost of the partition: $\frac{1}{n}\sum_{j=1}^{k} \sum_{x \in \mathcal{C}_j} \|\mu_j - x\|_{L_2}^2$;
  while *continue* **do**
    Update centroids, defined as the mean of each cluster according to the current partition;
    Partition `X` using the new centroids;
    Compute cost of the new partition;
    if *the new cost is better than the previous one (with a precision* `precision`*)* **then**
      Keep looping;
    else
      Go back to previous partition, previous centroids and previous cost and store them;
      Exit while loop;
return the list of `k` centroids that yielded the lowest cost among all the restarts, its corresponding partition of `X` and its cost

---

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

1. Implement Lloyd's algorithm.
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. Visualizing the "angle" of the elbow is not always easy, but it can be computed, using for instance `KneeLocator()` from the `kneed` library. In our specific case, the keyword arguments should be the following: `curve="convex", direction="decreasing"`.

7. How well does your obtained partition matches the target labels `y`?