# K-Means Clustering

A clustering algorithm is an algorithm designed with the purpose of grouping, into an arbitrary number of classes, a set of points. Such classes are created by trying to group together points that have common features. Thus, the goal of this category of algorithms is to build knowledge about the data provided as input that makes it possible to find hidden relationships among the data, or to classify (into one of the classes created in the training stages) a point whose class is not known a priori.

# Standard (or Lloyd) Algorithm

In this section, the standard algorithm for implementing k-means clustering is introduced.
First, it is important to define the initial condition (i.e., the input data) on which the algorithm will be run. Considering the case in two dimensions, the input is a collection of points that are randomly placed in 2D space (Figure 1).

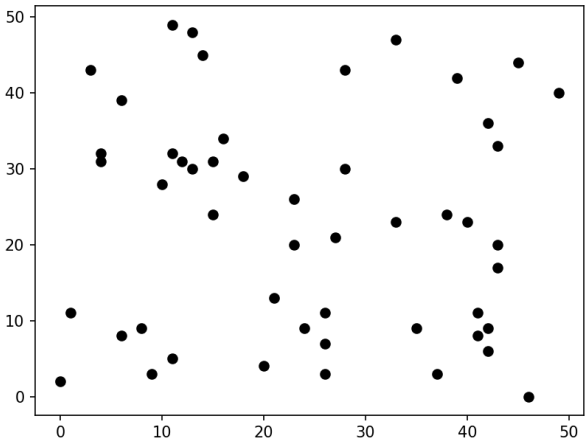


***Figure 1:*** *Initial distribution of random points*

The first step is to generate an arbitrary number of **centroids**, i.e., points in space representing the classes to which points will be assigned according to a criterion of maximum closeness.

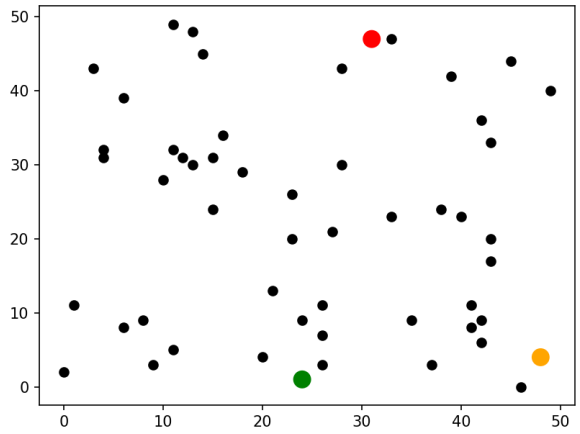


***Figure 2:*** *Random initial position of centroids*

Figure 2 shows three centroids representing 3 classes (red, green, orange). The generation of centroids can be done in 2 ways: by choosing as centroids some of the points provided as input, or by randomly generating them in space. After that the centroids are generated, each point is assigned a class according to the principle of maximum closeness.

That is, given a point P = (x, y) in space its membership class is the class represented by the centroid whose distance from the point is less than that of all other centroids. The choice of how to measure the distance between two points in space affects the final result of the classification. Therefore it is necessary to specify which measure is being used, in this case the **Euclidean distance**. Given two points A = $(a_x, a_y)$ and B = $(b_x, b_y)$ the Euclidean distance is calculated as

$$d(A, B) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \tag{1}$$

Figure 3 shows the application of the above to assign a class to each point.

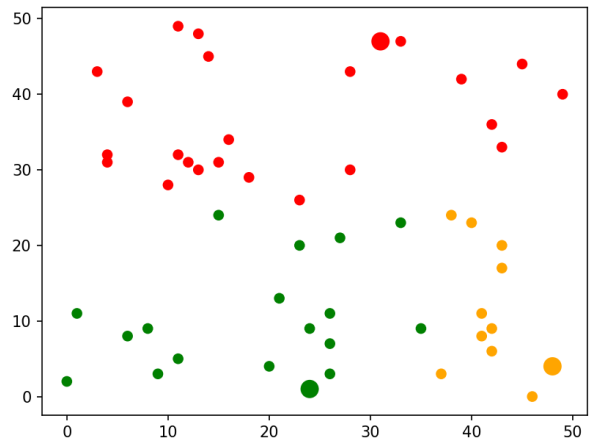


***Figure 3:*** *Punti dopo l'assegnazione delle classi*

The next step is to reposition the centroids by calculating the average of all points belonging to its class. If $C_i = (C_i^x, C_i^y)$ is the i-th centroid, its new position $(\tilde{C}_i)$ can be calculated as

$$\tilde{C}_i^x = \frac{1}{N_i} \cdot \sum_{j=1}^{N_i} P_{ij}^x \tag{2}$$

$$\tilde{C}_i^y = \frac{1}{N_i} \cdot \sum_{j=1}^{N_i} P_{ij}^y \tag{3}$$

Where $N_i$ is the number of points belonging to the i-th class and $P_{ij}$ is the j-th point belonging to the i-th class. Figure 4 shows the centroids after being repositioned.

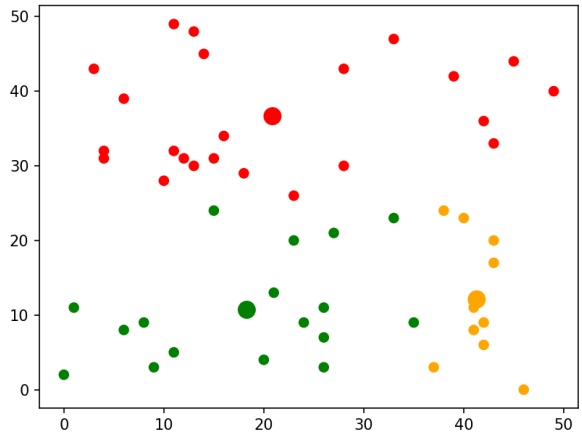


***Figure 4:*** *Punti dopo l'assegnazione delle classi*

At this point, the steps seen so far are repeated, that is, classes are reassigned to the points and the centroids are shifted

until a convergence criterion is reached, which for example could be when the centroids do not shift more than a certain threshold, or simply after a certain number of iterations. This latter method could be usefull when comparing two different algorithm (like lloyd vs hamerly algorithm which is described later) because the two could converge after a different number of iterations therefore by using a fixed number of repetitions it is easy to compare the result after the same number of cycles. Because of this, for this report, the second approach has been used.

# Number of iterations

The iterations required during a single cycle are

$$N \cdot K \cdot d$$

where $N$ is the number of points, $K$ is the number of centroids and k is the dimensionality of the data. This is because for each point it is necessary to calculate the distance to each centroid to know which is the least, and to calculate the distance it is necessary to sum k terms (one for each dimension).
So if m is the number of cycles, the iterations needed are in total

$$m(N \cdot K \cdot d) \tag{4}$$

**Algorithm 1** k-means pseudo-code

```
Let P be the set of all points P_i
Let C be the set of all centroids C_i
for l < iterations do
                                        ▷ Point assignation
    for P_i in P do
        d_min = d(C_1, P_i)
        classe = C_1
        for C_{j≠1} in C do
            d = d(C_j, P_i)
            if d ¡ d_min then
                d_min = d
                classe = C_j
            end if
        end for
    end for
                                        ▷ Centroid update
    Let P_Ci be the set of points with class C_i
    for C_i in C do
        C̃_i^x = 0
        C̃_i^y = 0
        for p in P_Ci do
            C̃_i^x = C̃_i^x + p^x
            C̃_i^y = C̃_i^y + p^y
        end for
        C_i^x = C̃_i^x / size(P_Ci)
        C_i^y = C̃_i^y / size(P_Ci)
    end for
end for
```