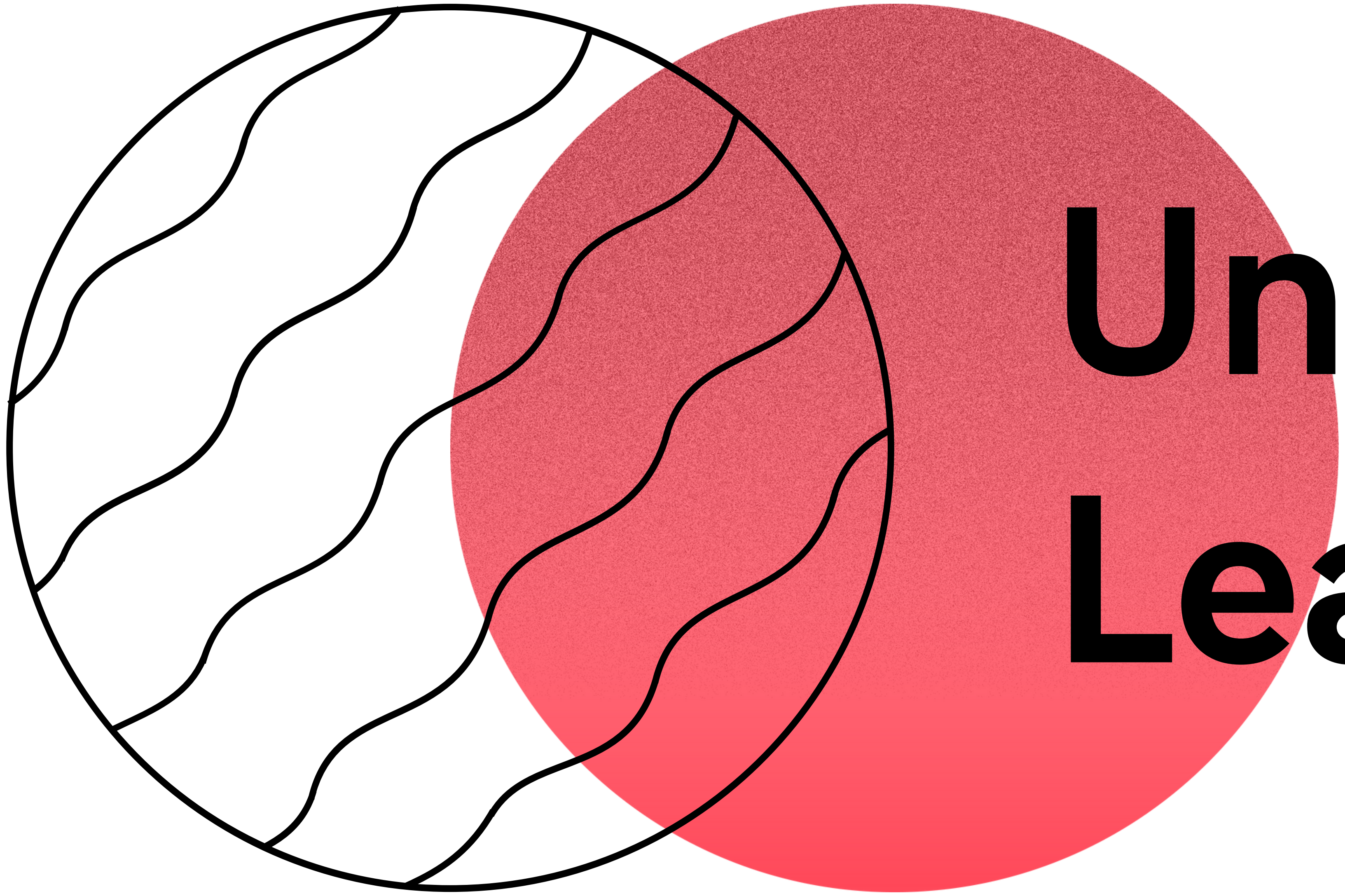
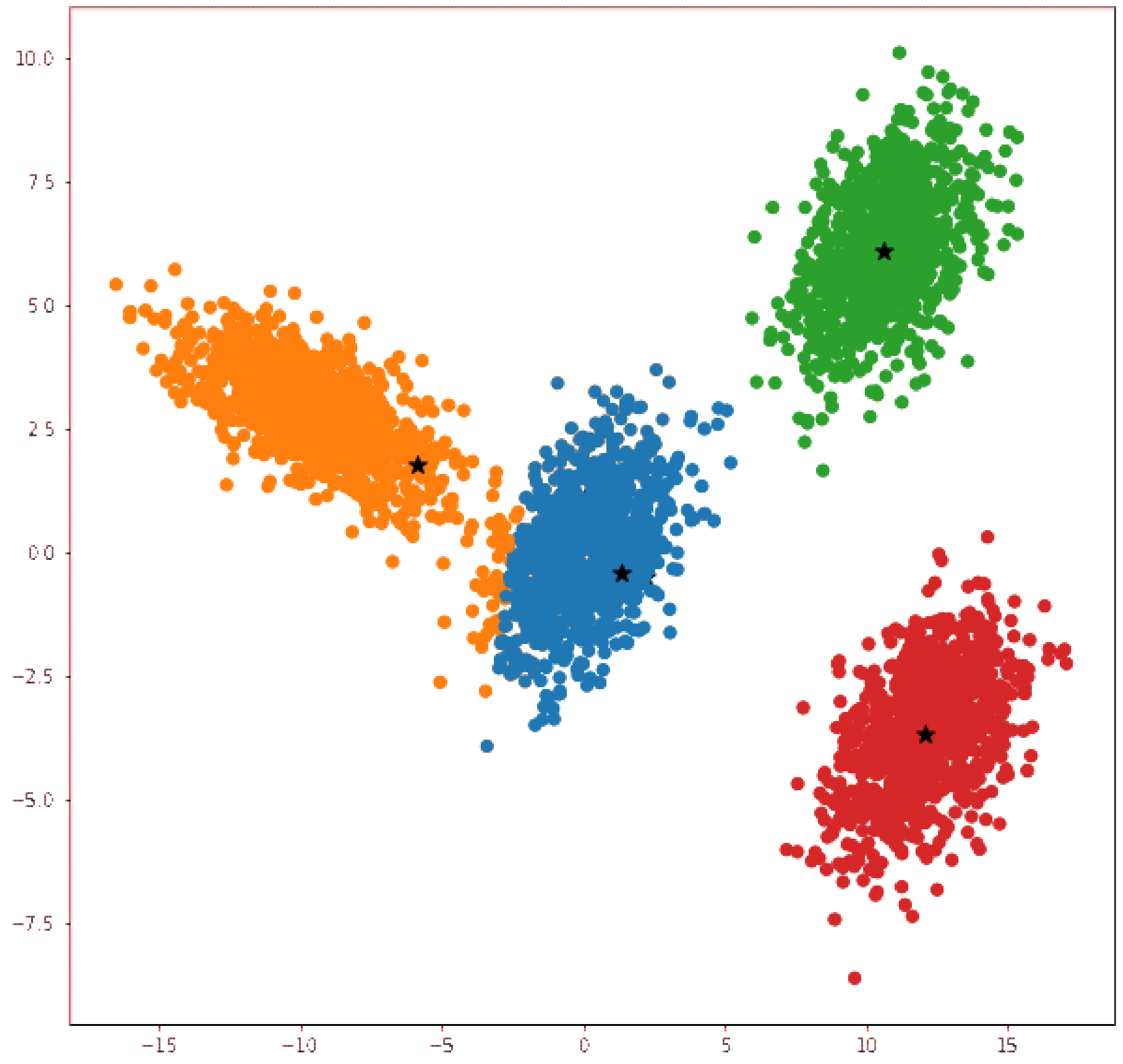


Unsupervised Learning

Unsupervised learning deals with problems in which your dataset doesn't have labels. This property is what makes it very problematic for many practical applications. The absence of labels that represent the desired behavior for your model means the absence of a solid reference point to judge the quality of your model.



KMeans

K-means is a popular unsupervised learning algorithm used to cluster data points into a specified number of clusters (k). The goal of k-means clustering is to minimize the sum of squared distances between data points and their corresponding cluster centers. Here's how the algorithm works:

- Initialize the centroids: Randomly select k data points from the dataset to act as the initial cluster centroids.

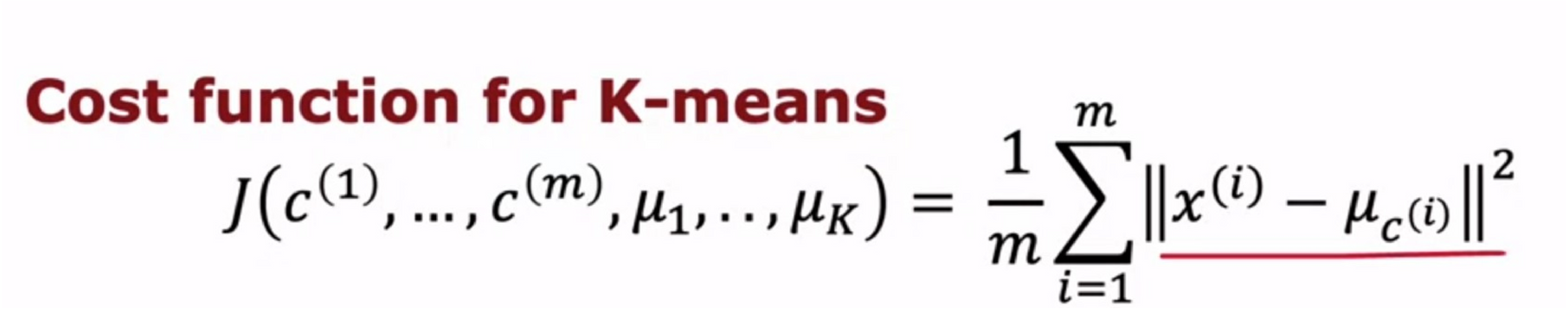
- Assign data points to clusters: For each data point in the dataset, calculate the distance to each centroid. Assign the data point to the cluster with the nearest centroid.

- Update centroids: Calculate the mean of all the data points assigned to each cluster. Move the centroid of each cluster to the mean of its assigned data points.

- Repeat steps 2 and 3 until convergence: Repeat steps 2 and 3 until the assignment of data points to clusters no longer changes or the maximum number of iterations is reached.

- Output the clusters: The final output of the k-means algorithm is a set of k clusters, each represented by its centroid.

- Notably, k-means clustering is sensitive to the initial choice of centroids, and the algorithm may converge to a suboptimal solution. To overcome this, k-means can be run multiple times with different initial centroid selections, and the solution with the lowest sum of squared distances can be chosen.

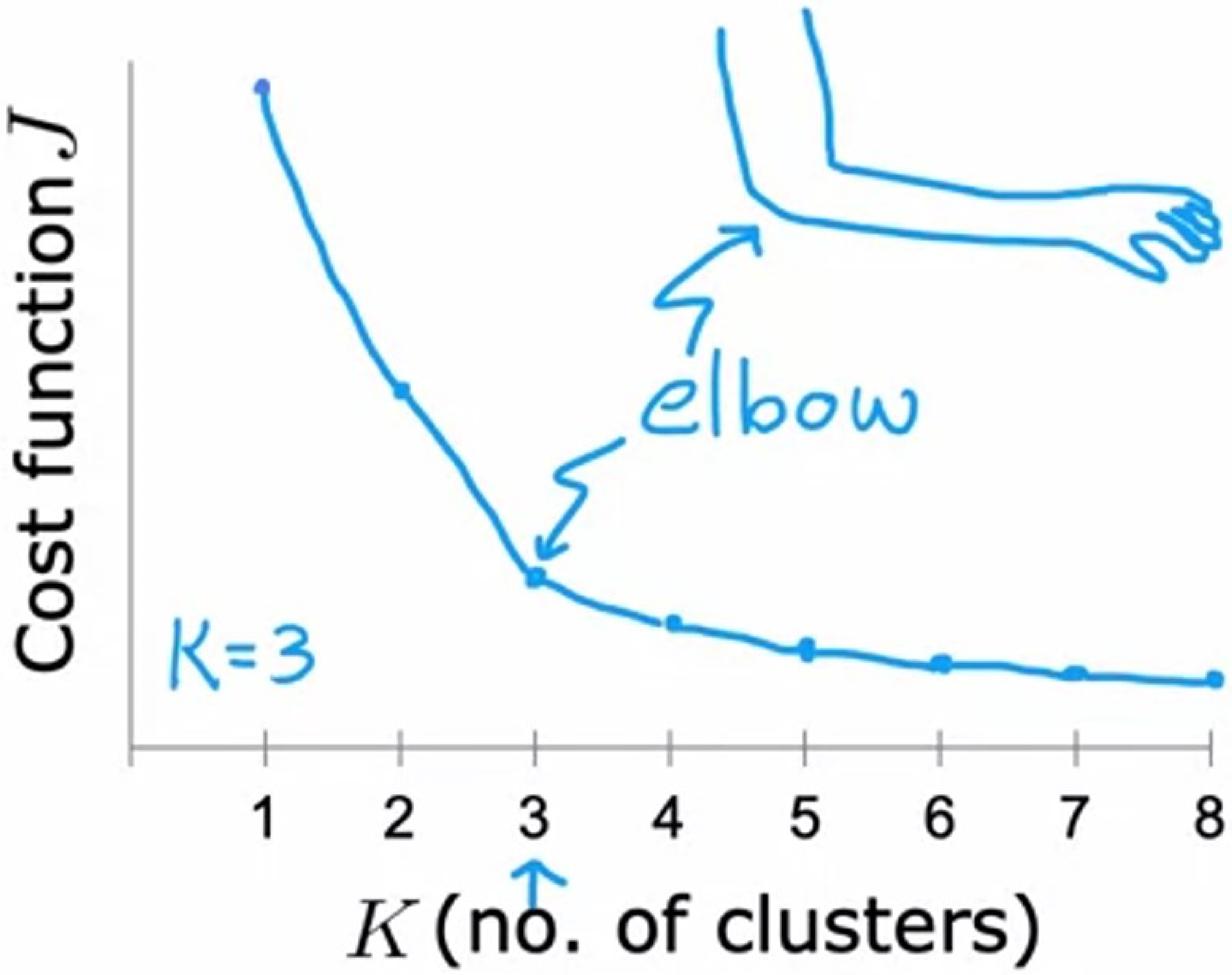


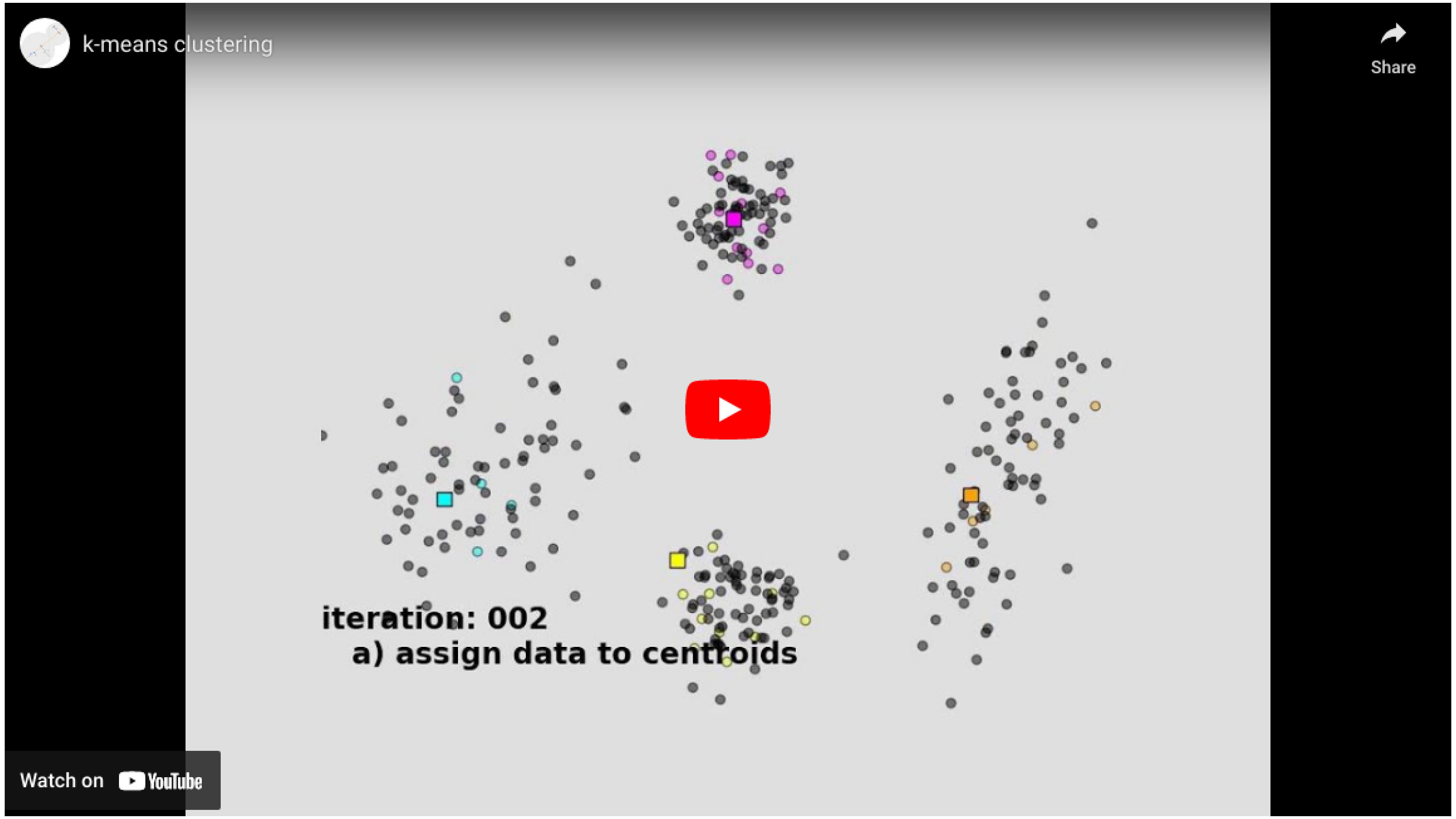
Cost function for K-means

$$J\left(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K\right) = \frac{1}{m} \sum_{i=1}^{m} \underline{\left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2}$$

Choosing the value of K

Elbow method





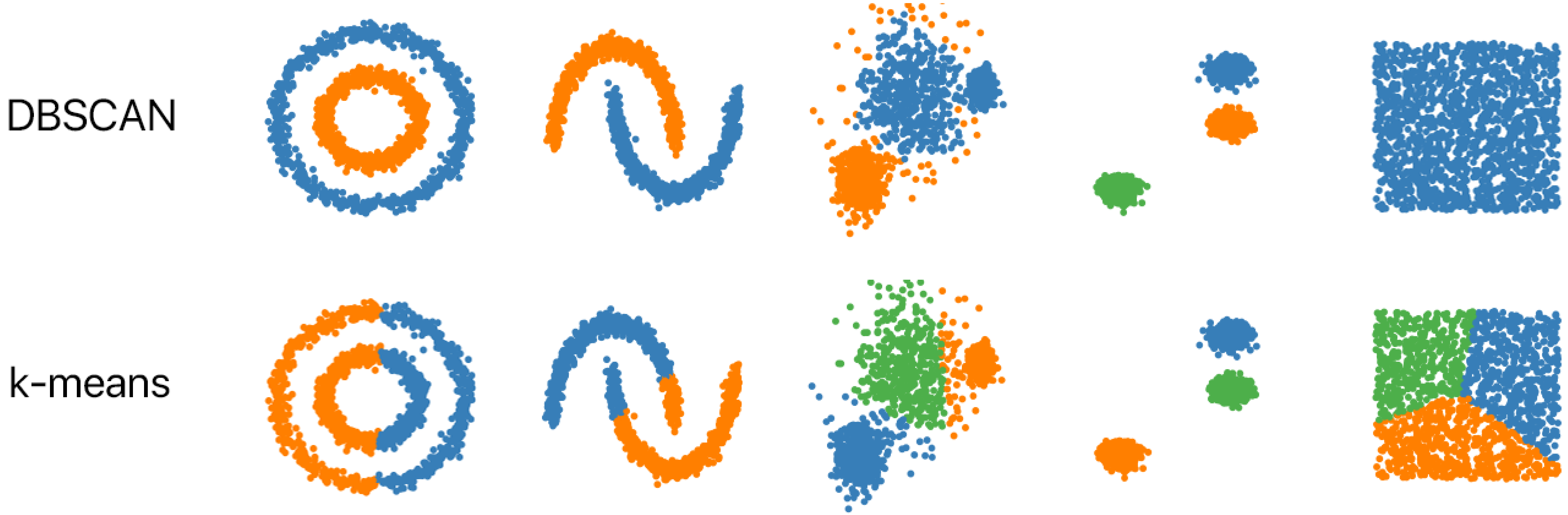
k-means clustering
Share
iteration: 002
a) assign data to centroids
Watch on YouTube

DBSCAN

Why do we need a Density-Based clustering algorithm like DBSCAN when we already have K-means clustering?

K-Means clustering may cluster loosely related observations together. Every observation becomes a part of some cluster eventually, even if the observations are scattered far away in the vector space. Since clusters depend on the mean value of cluster elements, each data point plays a role in forming the clusters. A slight change in data points might affect the clustering outcome. This problem is greatly reduced in DBSCAN due to the way clusters are formed. This is usually not a big problem unless we come across some odd shape data.

What's nice about DBSCAN is that you don't have to specify the number of clusters to use it. All you need is a function to calculate the distance between values and some guidance for what amount of distance is considered "close". DBSCAN also produces more reasonable results than k-means across a variety of different distributions. Below figure illustrates the fact:



The DBSCAN algorithm uses two parameters:

min_samples: The minimum number of points (a threshold) clustered together for a region to be considered dense.

eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point.

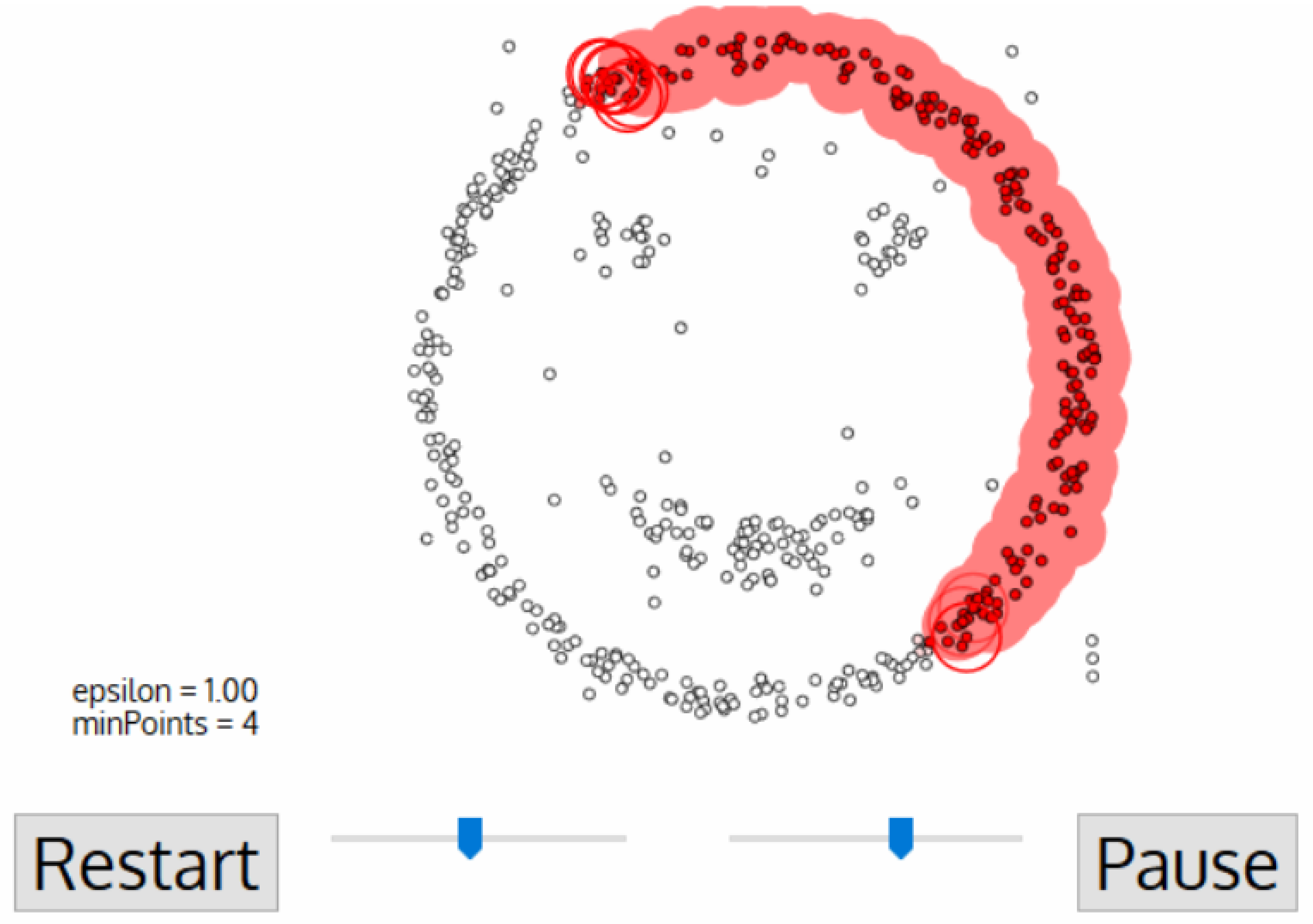
These parameters can be understood if we explore two concepts called Density Reachability and Density Connectivity.

Reachability in terms of density establishes a point to be reachable from another if it lies within a particular distance (eps) from it.

Connectivity, on the other hand, involves a transitivity based chaining-approach to determine whether points are located in a particular cluster. For example, p and q points could be connected if p->r->s->t->q, where a->b means b is in the neighborhood of a.

If your data has more than 2 dimensions, choose min_samples *at least* = 2*dim, where dim= the dimensions of your data set.

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least ‘min_samples’ points within a radius of ‘ε’ to the point, then we consider all these points to be part of the same cluster.
- The clusters are then expanded by recursively repeating the neighborhood calculation for each neighboring point



epsilon = 1.00
minPoints = 4