Unsupervised Learning

Introduction: In contrast to the Supervised Learning, Unsupervised Learning is about making decisions without any given output labels i.e. in Unsupervised Learning, the training data is provided without any target labels/ desired output. Unsupervised Learning algorithms try to find hidden patterns in the input data.

Clustering: Clustering is one of the most important unsupervised learning problems, wherein we try to optimally distribute the data into Clusters (groups) of data points, which are similar in some way within that particular Cluster. A Cluster is, therefore, a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other Clusters. In the figure below, the data points are organised into three different clusters (groups).

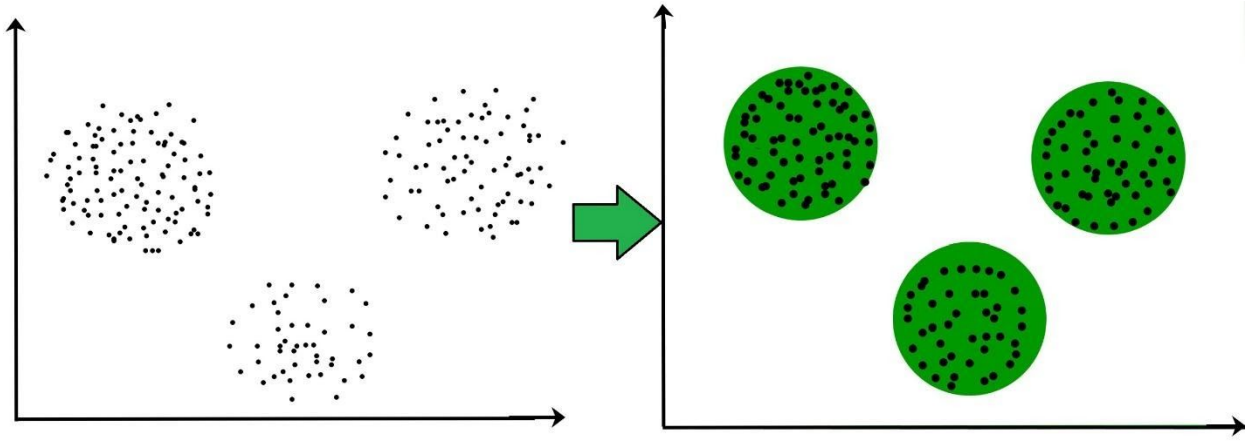


Image source: https://media.geeksforgeeks.org/wp-content/uploads/merge3cluster.jpg

Now Clustering can be divided into two categories:

1. **Flat Clustering**: In this, the complete data is divided into some (say K) clusters such that, if a particular data point belongs to a definite cluster then it could not be included in another cluster. (see above figure)
2. **Hierarchical Clustering**: In this, initially all data points are put into a single cluster (say root cluster) and then this root cluster can further be divided into sub-clusters and so on (based on a termination condition for when to stop dividing further) in a tree-like structure. Unlike Flat Clustering, here, each data point can belong to many clusters.

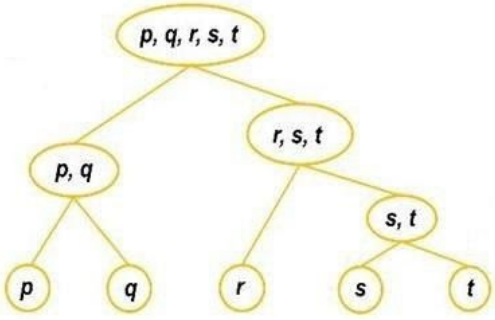
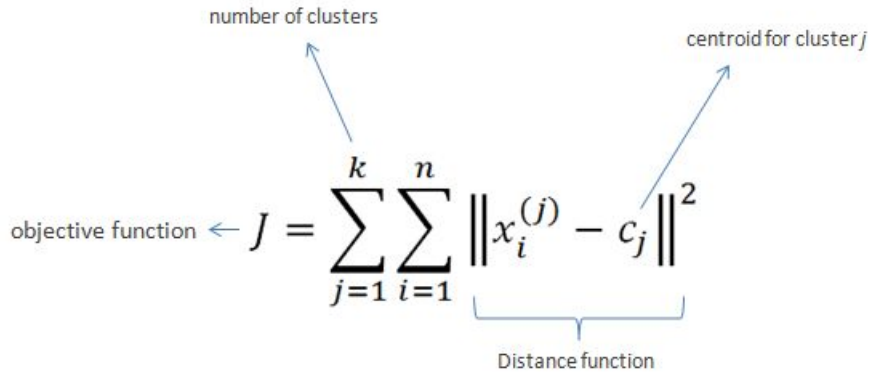


Image Source: https://www.researchgate.net

KMeans for Flat Clustering:

As the name suggests, it has to do something with K (no. of clusters to make) and the Mean values. In this algorithm, the data points are divided into K clusters such that distances of all points in a particular cluster with the mean value (centroid) of that cluster (not necessarily a real data point) will be much smaller than their distances with the mean values (centroids) of other clusters.

To put it mathematically, say we have reached a state, where we have divided the data into K clusters, then this algorithm tries to minimise the following objective function:



: i-th data point of j-th cluster $x_i^{(j)}$

KMeans Algorithm:

1. Randomly pick k mean values (centroids) representing k clusters.
2. Assign each data point one of these k clusters based on their distances with these centroids.
3. Calculate the new mean values for the clusters.
4. Go back to step 2 and repeat the same for certain no. of iterations or if there is no more considerable change in mean values.

(Note: See code files provided in the lecture videos for implementation.)

It is worth noting that the KMeans algorithm does not necessarily find the most optimal clusters. It is significantly sensitive to the initial randomly selected cluster centroids. The algorithm can be run multiple times to reduce this effect.

How to choose the optimal value of 'K' (number of clusters):

While choosing the optimal value for ‘k’, we would like to have the distribution of data such that the distance among points in the same cluster is less as compared to their distances from the points in different clusters.

For this, there is a metric called ‘Silhouette’. The silhouette value is a measure of how similar an object is to its cluster compared to other clusters.

Let’s say

a_i is the average distance of i-th point from all other points in the same cluster as the i-th point

b_i is the average distance of i-th point from all other points in a neighboring cluster

then, we define

$S_i = (b_i - a_i)/Max(b_i , a_i)$

Where,

$-1 <= S_i <= 1$

(a value close to 1 indicates that i-th point is close to the points in its cluster as compared to the points in other clusters)

Thus, this value can be found for all the points and take an average of all, which will give Silhouette score for the clustering in context. This way, we can find a Silhouette score for different values of ‘k’ (number of clusters) and choose the optimal ‘k’ which gives higher value of Silhouette score.

K-Medoids Clustering:

This is similar to the K-Means except for some differences. One significant difference between K-Means and K-Medoids is that, in K-Medoids, instead of taking mean values, we need to take actual points from the data as our Medoids (plays the same role as Centroid (Mean) in K-Means). Other difference is that in K-Medoids, l1 distance is used instead of l2 distance (used in K-Means), which can make it K-Medoids more robust to noise and outliers in the dataset as compared to K-Means.

K-Medoids Algorithm:

1. Randomly pick k data points as Medoids.
2. Assign each data point one of these k clusters based on their distances with these Medoids.
3. Find replacements for existing Medoids:
    a) For each Medoid ‘m’, go to each data point which is not a medoid and calculate the new score of clustering considering that point as Medoid in place of ‘m’
    b) If the new score is better than the previous score (i.e. with ‘m’ as Medoid), then make this point as the new Medoid in place of ‘m’
4. Go back to step 2 and repeat the same for certain no. of iterations.

Hierarchical Clustering: As explained above, in this clustering, each data point can belong to many clusters. It is useful in many applications, where we want to target some specific data points (at lower levels in the tree structure) as well as much broader set of data points (at higher levels in the tree structure).

Now this tree like structure can be built in two ways:

a) Top – Down
b) Bottom-Up

a) **Top – Down/ Divisive Approach**: In this, we will first consider all the data points in one cluster and then keep on dividing it further down in the tree-like structure. For dividing clusters in smaller clusters, we can use any of the clustering algorithm i.e. K-Means or K-Medoids.
b) **Bottom – Up/ Agglomerative Approach**: In this, we will first consider all data points as different clusters of their own. Afterwards, these smaller clusters are joined to make new bigger clusters based on distance between the smaller clusters.

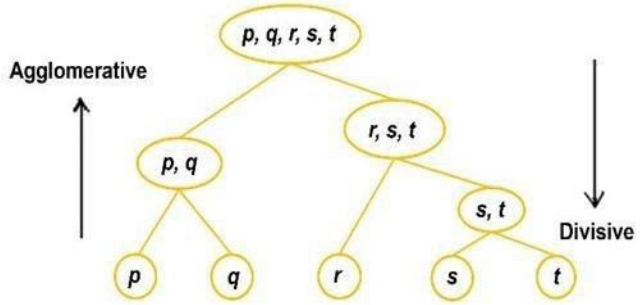


Image source: https://www.researchgate.net