Clustering: K-Means

Introduction

Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.

Clusters are loosely defined as groups of data objects that are more similar to other objects in their cluster than they are to data objects in other clusters. In practice, clustering helps identify two qualities of data:

- Meaningfulness
- Usefulness

Clustering Techniques

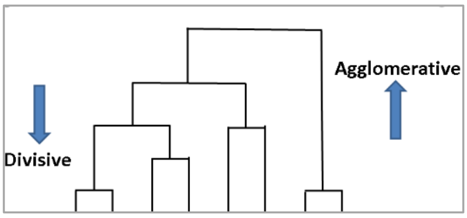
Partitional Clustering

divides data objects into nonoverlapping groups. In other words, no object can be a member of more than one cluster, and every cluster must have at least one object.

These techniques require the user to specify the number of clusters, indicated by the variable k. Two examples of partitional clustering algorithms are **k-means** and **k-medoids.**
Advantage: They work well when clusters have a spherical shape.
Dis-Advantage: They're not well suited for clusters with complex shapes and different sizes.

Hierarchical Clustering

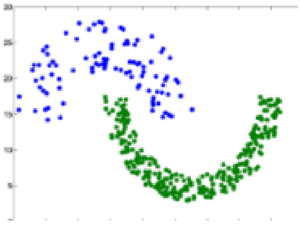
- Agglomerative clustering is the bottom-up approach
- Divisive clustering is the top-down approach



Advantage: They often reveal the finer details about the relationships between data objects. They provide an interpretable dendrogram.
Dis-Advantage: They're computationally expensive with respect to algorithm complexity

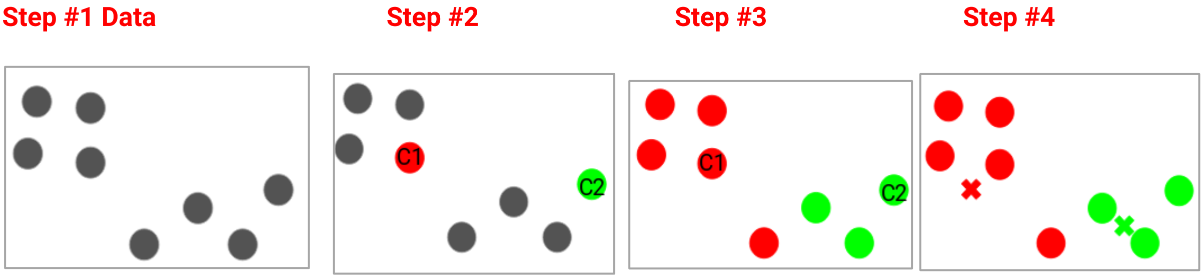
Density-based Clustering

Determines cluster assignments based on the density of data points in a region.
Advantage: They excel at identifying clusters of **no spherical shapes**.
They're resistant to **outliers**.
Dis-Advantage: They aren't well suited for clustering in high-dimensional spaces. They have trouble identifying clusters of varying densities.



K-Means Algorithm

K-means is a centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid. Recall the first property of clusters – it states that the points within a cluster should be similar to each other. So, our aim here is to minimize the distance between the points within a cluster.



Step 1: Choose the number of clusters k

Step 2: Select k random points from the data as centroids

Randomly select the centroid for each cluster. Let's say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid: Here, the red and green circles represent the centroid for these clusters.

Step 3: Assign all the points to the closest cluster centroid

Once we have initialized the centroids, we assign each point to the closest cluster centroid:
You can see that the points which are closer to the red point are assigned to the red cluster whereas the points which are closer to the green point are assigned to the green cluster.

Step 4: Recompute the centroids of newly formed clusters

Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters: Here, the red and green crosses are the new centroids.

Step 5: Repeat steps 3 and 4

The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration. But wait – when should we stop this process? It can't run till eternity, right?

Stopping Criteria for K-Means Clustering

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations are reached

Choosing the Appropriate Number of Clusters

Commonly used to evaluate the appropriate number of clusters are:

- The elbow method
- The silhouette coefficient

Elbow method

To perform the elbow method, run several k-means, increment k with each iteration, and record the sum of squared error (SSE).

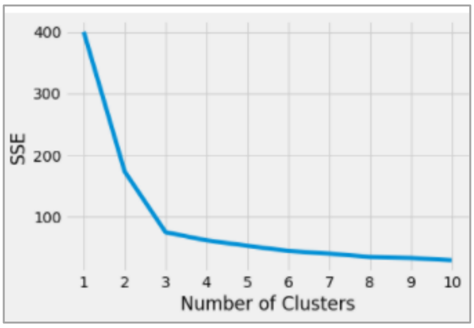
When you plot SSE as a function of the number of clusters, notice that SSE continues to decrease as you increase k. As more centroids are added, the distance from each point to its closest centroid will decrease.

There's a sweet spot where the SSE curve starts to bend known as the **elbow point**. The x-value of this point is thought to be a reasonable trade-off between error and number of clusters. In this example, the elbow is located at x=3:

$$\text{WCSS} = \sum_{C_k}^{C_n}\Big(\sum_{d_i\, in\, C_i}^{d_m} distance(d_i, C_k)^2\Big)$$

Where,

C is the cluster centroids and d is the data point in each Cluster.



Silhouette coefficient

The silhouette coefficient is a measure of cluster **cohesion** and **separation**. It quantifies how well a data point fits into its assigned cluster based on two factors:

1. How close the data point is to other points in the cluster (**Cohesion**)
2. How far away the data point is from points in other clusters (**Separation**)

Distance Measures:

Euclidean distance measure:

The most common case is determining the distance between two points.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Manhattan distance measure:

The Manhattan distance is the simple sum of the horizontal and vertical components.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$