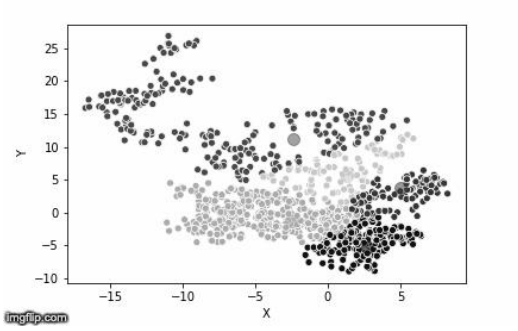


K-Mean Clustering

K-Means Cluster

- K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters.
- Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on..

K-Means Cluster

- It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each datapoints belongs only one group that has similar properties.

K-Means Cluster

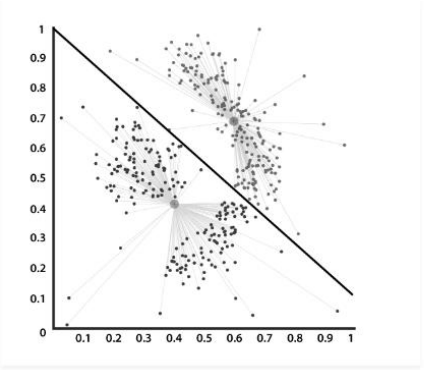
- It is a centroid-based algorithm, where each cluster is associated with a centroid.
- The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

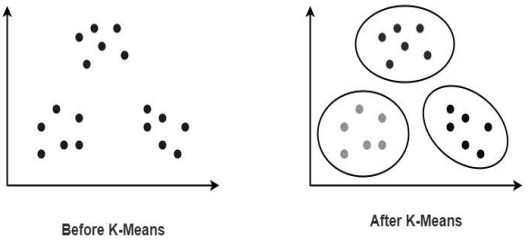
K-Means Cluster

- The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters.
- The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

- Determines the best value for K center points or centroids by an iterative process.
- Assigns each data point to its closest k-center.
- Those data points which are near to the particular k-center, create a cluster.





It partitions the data set such that-

- Each data point belongs to a cluster with the nearest mean.
- Data points belonging to one cluster have high degree of similarity.
- Data points belonging to different clusters have high degree of dissimilarity.

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
(5) **until** no change;

K-Means Clustering Algorithm-

- Step-01:

- Choose the number of clusters K.

Step-02:

Randomly select any K data points as cluster centers or centroids.

- Select cluster centers in such a way that they are as farther as possible from each other.

Step-03:

- Calculate the distance between each data point and each cluster center.
- The distance may be calculated either by using given distance function or by using euclidean distance formula.

Step-04:

- Assign each data point to some cluster.
- A data point is assigned to that cluster whose center is nearest to that data point.

Step-05:

- Re-compute the center of newly formed clusters.
- The center of a cluster is computed by taking mean of all the data points contained in that cluster.

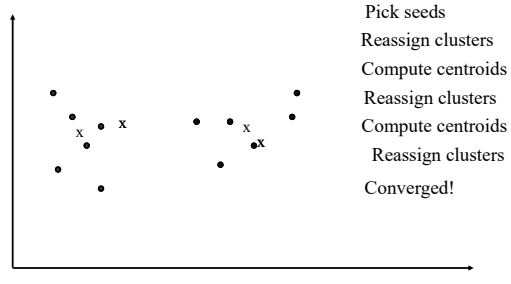
Step-06:

- Keep repeating the procedure from Step-03 to Step-05 until any of the following stopping criteria is met-

- Center of newly formed clusters do not change
- Data points remain present in the same cluster
- Maximum number of iterations are reached

Sec. 16.4

K Means Example
(*K*=2)



How to choose the value of "K number of clusters" in K-means Clustering?

- The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms.
- But choosing the optimal number of clusters is a big task.
- There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K.

Elbow Method Contd.,

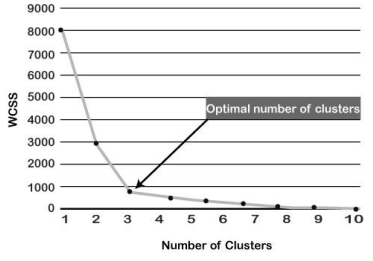
- The Elbow method is one of the most popular ways to find the optimal number of clusters.
- This method uses the concept of WCSS value.
- WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

Elbow Method Contd.,

- WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2+∑Pi in CLuster3 distance(Pi C3)2
- In the above formula of WCSS,
- ∑Pi in Cluster1 distance(Pi C1)2: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

Elbow Method Contd.,

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Distance Measure K-Means

- The formula for Manhattan distance between two points on a plane is (d=|x2-x1|+|y2-y1|)
- This method is called "Manhattan distance" because, like a taxi driving through the grid-like streets of Manhattan, it must travel along the grid lines. city block,
- The Euclidean distance formula is used to find the distance between two points on a plane. This formula says the distance between two points (x1 , y1 1) and (x2 , y2) is d = √[(x2 – x1)2 + (y2 – y1)2]

Sec. 16.4

Termination conditions

- Several possibilities, e.g.,
  - A fixed number of iterations.
  - Doc partition unchanged.
  - Centroid positions don't change.

Does this mean that the docs in a cluster are unchanged?

Sec. 16.4

Convergence

- Why should the *K*-means algorithm ever reach a *fixed point*?
  - A state in which clusters don't change.
- *K*-means is a special case of a general procedure known as the *Expectation Maximization (EM) algorithm*.
  - EM is known to converge.
  - Number of iterations could be large.
    - But in practice usually isn't

K Mediods Clustering

- K-Medoids clustering is an unsupervised machine learning algorithm used to group data into different clusters.
- It is an iterative algorithm that starts by selecting **k data points** as medoids in a dataset.
- After this, the distance between each data point and the medoids is calculated.
- Then, the data points are assigned to clusters associated with the medoid at the minimum distance from each data point.
- Here, the medoid is the most **centrally located point** in the cluster.
- Once we assign all the data points to the clusters, we calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. We term the sum of distances as the cost.

K-Medoids Clustering Algorithm

- First, we select **K** random data points from the dataset and use them as medoids.
- Now, we will calculate the distance of each data point from the medoids. You can use any of the Euclidean, Manhattan distance, or squared Euclidean distance as the distance measure.
- Once we find the distance of each data point from the medoids, we will assign the data points to the clusters associated with each medoid. The data points are assigned to the medoids at the closest distance.
- After determining the clusters, we will calculate the sum of the distance of all the non-medoid data points to the medoid of each cluster. Let the cost be Ci.
- Now, we will select a random data point Dj from the dataset and swap it with a medoid Mi. Here, Dj becomes a temporary medoid. After swapping, we will calculate the distance of all the non-medoid data points to the current medoid of each cluster. Let this cost be Cj.
- If Ci>Cj, the current medoids with Dj as one of the medoids are made permanent medoids. Otherwise, we undo the swap, and Mi is reinstated as the medoid.
- Repeat 4 to 6 until no change occurs in the clusters.

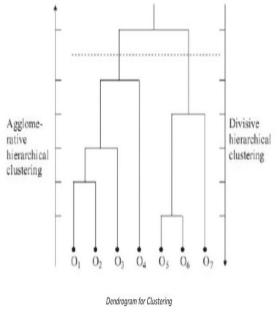
Hierarchical Methods

- Data is grouped into a tree like structure. There are two main clustering algorithms in this method:

Hierarchical Methods

- A. Divisive Clustering(**DIANA**): It uses the top-down strategy, the starting point is the largest cluster with all objects in it and then split recursively to form smaller and smaller clusters. It terminates when the user-defined condition is achieved or final clusters contain only one object.
- B. Agglomerative Clustering(**AGNES**): It uses a bottom-up approach. It starts with each object forming its own cluster and then iteratively merges the clusters according to their similarity to form large clusters. It terminates either
- When certain clustering condition imposed by user is achieved or
- All clusters merge into a single cluster

- A dendrogram, which is a tree like structure, is used to represent hierarchical clustering.
- Individual objects are represented by leaf nodes and the clusters are represented by root nodes. A representation of dendrogram is shown in this figure



Dendrogram for Clustering

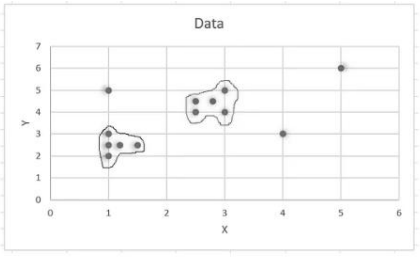
DBSCAN

- DBSCAN stands for Density-Based Spatial Clustering Application with Noise.
- It is an unsupervised machine learning algorithm that makes clusters based upon the density of the data points or how close the data is.
- That said, the points which are outside the dense regions are excluded and treated as noise or outliers.
- This characteristic of the DBSCAN algorithm makes it a perfect fit for outlier detection and making clusters of arbitrary shape.

- The DBSCAN algorithm takes two input parameters.
- Radius around each point (eps) and the minimum number of data points that should be around that point within that radius (MinPts).

DBSCAN

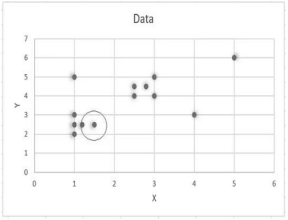
X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Logic and Steps:

- For example, consider the point (1.5,2.5),
- if we take eps = 0.3, then the circle around the point with radius = 0.3, will contain only one other point inside it (1.2,2.5)

X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3



Algorithm in action

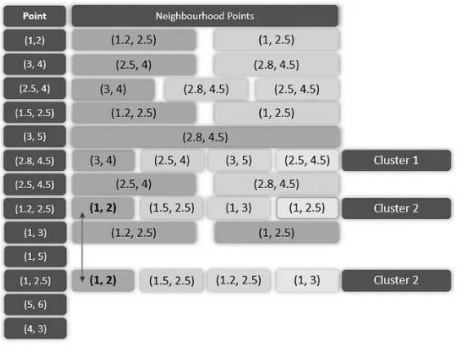
- Let's choose eps = 0.6 and MinPts = 4.
- Let's consider the first data point in the dataset (1,2) & calculate its distance from every other data point in the data set.

X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

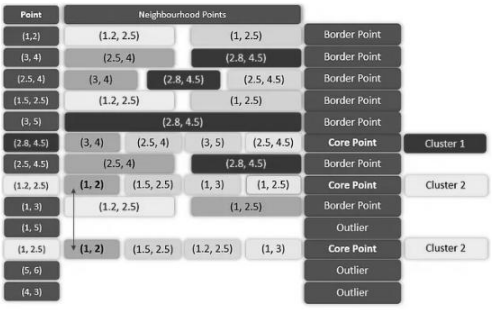
- As evident from the above table the point (1, 2) has only two other points in its neighbourhood (1, 2.5), (1.2, 2.5) for the assumed value of eps, as its less than MinPts, we can't declare it as a **core point**.

X	Y	Distance from (1,2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	0.53
1	3	1
1	5	3
1	2.5	0.5
5	6	5.6
4	3	3.1

- Let's repeat the above process for every point in the dataset and find out the neighbourhood of each. The calculations when repeated can be summarized



- There are three points in the data set, (2.8, 4.5) (1.2, 2.5) (1, 2.5) that have 4 neighbourhood points around them, hence they would be called core points.
- Hence, (2.8, 4.5) is assigned to a new cluster, Cluster 1 and so is the point (1.2, 2.5), Cluster 2. Also observe that the core points (1.2, 2.5) and (1, 2.5) share at least one common neighbourhood point (1,2) so, they are assigned to the same cluster



- There are three types of points in the dataset as detected by the DBSCAN algorithm, core, border and outliers.

Cluster 1	Cluster 2	Outliers
(3,4)	(1, 2)	(1, 5)
(2.5, 4)	(1.5, 2.5)	(5, 6)
(3,5)	(1.2, 2.5)	(4, 3)
(2.8, 4.5)	(1, 3)	
(2.5, 4.5)	(1, 2.5)	