Unsupervised Learning Techniques

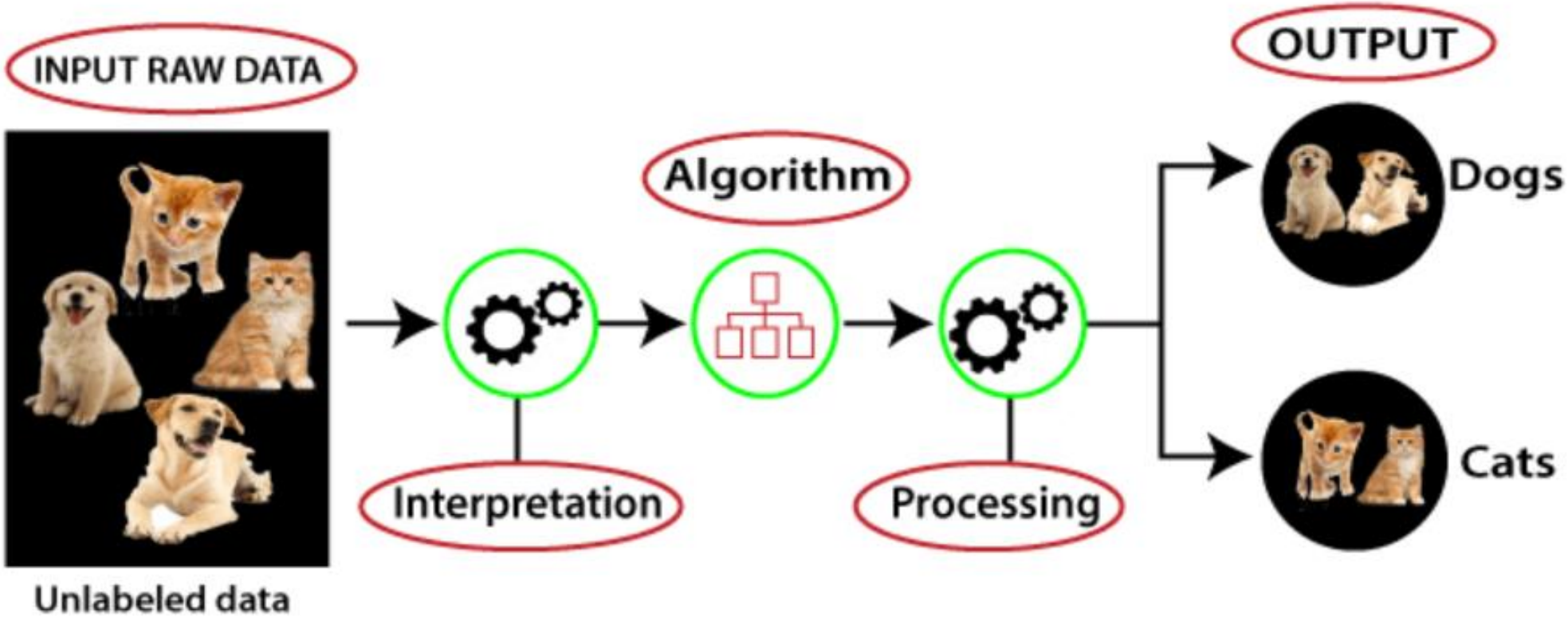
By Dr. Srabana Pramanik & Ms. Mary Divya Shamili
Asst. Professor
Department of SCSE
Presidency University

Unsupervised Machine learning

- Every Machine Learning engineer wants to achieve accurate predictions with their algorithms. Such learning algorithms are generally broken down into two types - supervised and unsupervised.
- In Unsupervised learning the unlabeled and unclassified information is analyzed to discover the hidden knowledge. Here the algorithms work without prior training but they can identify the patterns and similarity present in the dataset.

Definition

- Unsupervised learning is a type of machine learning where the model is trained on a dataset without any labeled output. The goal of unsupervised learning is to find patterns, structures, or relationships in the data that can be used to gain insights, make predictions, or classify new data.
- By finding patterns and relationships in large and complex datasets, unsupervised learning can provide valuable insights and help solve real-world problems.
- Unsupervised learning has a wide range of applications, including customer segmentation, image and speech recognition, anomaly detection, and natural language processing.


INPUT RAW DATA
Algorithm
OUTPUT
Dogs
Cats
Interpretation
Processing
Unlabeled data

There are several common techniques used in unsupervised learning, including

- Clustering: Clustering is a technique used to group similar data points together based on some similarity metric. The most common clustering algorithms are k-means, hierarchical clustering, and density-based clustering.
- Dimensionality Reduction: Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset while preserving the most important information. Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) are common dimensionality reduction techniques.

Techniques

- Anomaly Detection: Anomaly detection is a technique used to identify data points that are significantly different from the majority of the data. Isolation Forest and Local Outlier Factor (LOF) are common anomaly detection algorithms.
- Association Rule Mining: Association rule mining is a technique used to identify patterns in data that occur very frequently in the data set. Apriori and FP-Growth are common association rule mining algorithms.
- Generative Models: Is used to classify data into different categories based on the probability distribution. Gaussian Mixture Models (GMM) and Variational Autoencoders (VAE) are common generative models.

1. GMMs can be used to estimate the probability that a new data point belongs to each cluster. Then based on maximum probability it will select the cluster. Gaussian mixture models can be used in different areas, including finance, marketing, etc.

K means techniques

- K-Means clustering is a popular unsupervised learning algorithm that aims to partition a set of data points into K clusters based on the similarity between them. The algorithm works by iteratively assigning each data point to its nearest cluster centroid and updating the centroids based on the mean of the assigned data points.
- There are two main variants of the K-Means algorithm: simple K-Means and mini-batch K-Means.
- In simple K-Means, the algorithm starts by randomly selecting K data points as the initial centroids. Then, it iteratively assigns each data point to its nearest cluster centroid and updates the centroids based on the mean of the assigned data points. The process continues until the centroids no longer move significantly or a maximum number of iterations is reached.

K means algorithm with updating centroids incrementally

K-Means to update the centroids incrementally, we can use the following steps:

- Step1:Initialize the centroids: Randomly select K data points as the initial centroids.
- Step2:Assign data points to centroids: For each data point, compute its distance to each centroid and assign it to the nearest centroid.
- Step 3: Update the centroids: For each centroid, compute the mean of the assigned data points and use it as the new centroid.(for normal K-means)

  or

- Step 3: Incremental update: When a new data point is added to the cluster, we can update the centroids incrementally by only computing the mean of the new data point and the current centroid. This avoids the need to recompute the mean of all the data points assigned to the centroid.(for update the centroids incrementally)
- Step 4:Repeat steps 2-3 until the centroids no longer move significantly or a maximum number of iterations is reached.

Mini Batch K-Means

- Mini-batch K-Means is a variant of K-Means that is designed to handle large datasets more efficiently. Instead of using all the data points in each iteration, mini-batch K-Means randomly selects a small subset (batch) of the data points to update the centroids. This reduces the computational complexity of the algorithm and allows it to handle large datasets in a reasonable amount of time.
- Overall, K-Means clustering is a powerful technique for partitioning data into clusters based on their similarity. The algorithm can be further improved by using mini-batch K-Means to handle large datasets and updating the centroids incrementally to handle streaming data.

Drawbacks of K-Means:

- Although K-Means is a popular and effective clustering algorithm, it has several limitations, including

1. Sensitivity to initial centroids: The quality of the clustering solution can be highly dependent on the initial randomly selected centroids
2. Difficulty in determining the optimal number of clusters: Choosing the number of clusters can be a challenging task, as it requires manual intervention or the use of heuristic techniques.
3. Sensitive to outliers: K-means is sensitive to outliers, which can have a significant impact on the resulting clusters.
4. Not able to form cluster of any shape.

K-Means++ :

- K-Means++ is an improvement over the original K-Means algorithm that addresses the sensitivity to the initial centroids problem.
- Instead of randomly selecting the initial centroids, K-Means++ uses a smarter initialization strategy that selects the initial centroids with a higher probability of being far from each other.
- This helps to improve the quality of the clustering solution and reduce the number of iterations needed to converge.

K- Medoids technique

- **K-Medoids** (also called Partitioning Around Medoid) algorithm was proposed in 1987 by Kaufman and Rousseeuw. A medoid can be defined as a point in the cluster, whose dissimilarities with all the other points in the cluster are minimum. The dissimilarity of the medoid(Ci) and object(Pi) is calculated by using E = |Pi – Ci|

- *The cost in K-Medoids algorithm is given as*

$$c = \sum_{Ci} \sum_{Pi \in Ci} |Pi - Ci|$$

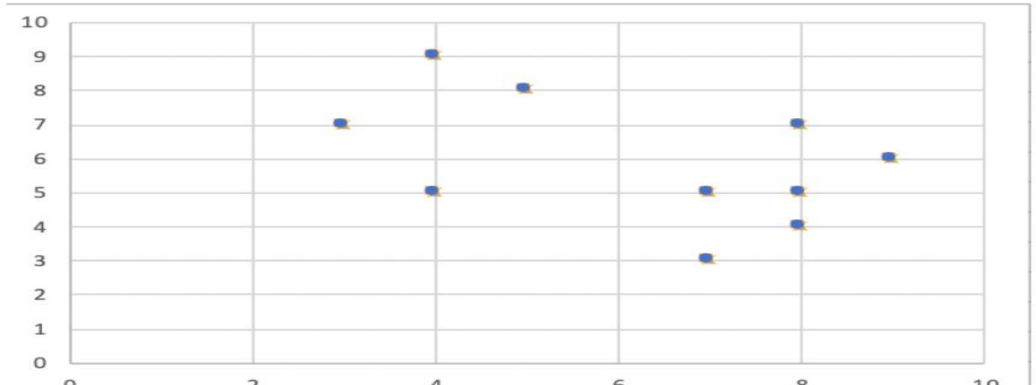
Algorithm

1. Initialize: select *k* random points out of the *n* data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.
3. While the cost decreases: For each medoid m, for each data o point which is not a medoid:
    1. Swap m and o, associate each data point to the closest medoid, and recompute the cost.
    2. If the total cost is more than that in the previous step, undo the swap.

	X	Y
0	8	7
1	3	7
2	4	9
3	9	6
4	**8**	**5**
5	5	8
6	7	3
7	8	4
8	7	5
9	**4**	**5**

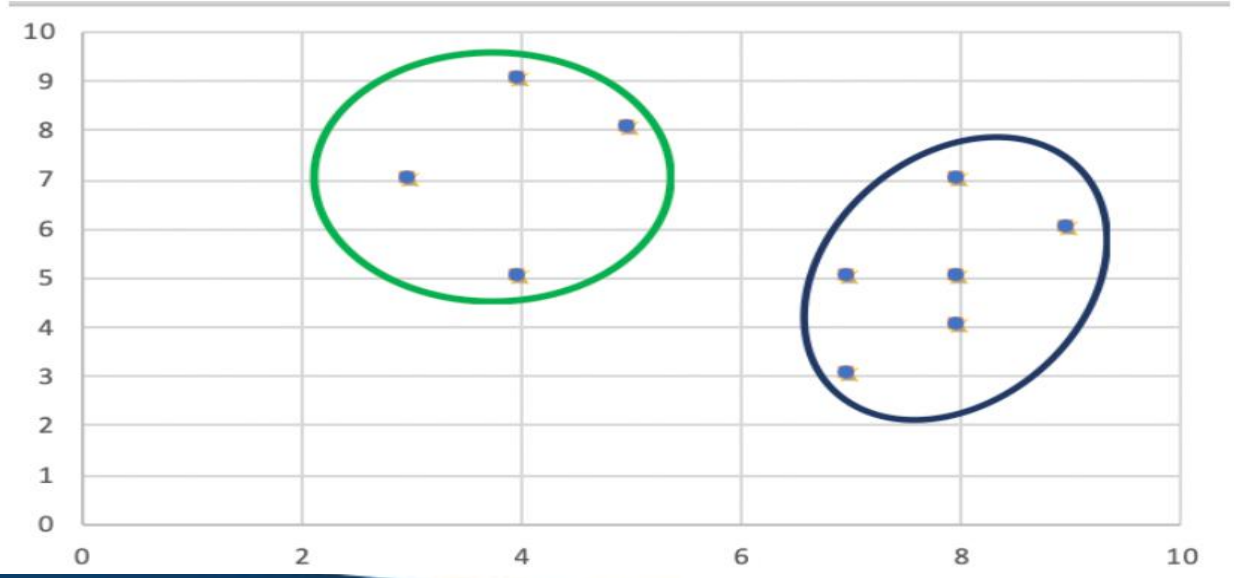
Let's consider the following example: If a graph is drawn using the above data points, we obtain the following:

- **Step 1:** Let the randomly selected 2 medoids, so select k = 2, and let **C1 -(4, 5)** and **C2 -(8, 5)** are the two medoids.
- **Step 2: Calculating cost.:** The dissimilarity of each non-medoid point with the medoids is calculated and tabulated:
- Here we have used Manhattan distance formula to calculate the distance matrices between medoid and non-medoid points. That formula tell that **Distance = |X1-X2| + |Y1-Y2|.**
- Each point is assigned to the cluster of that medoid whose dissimilarity is less. Points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The Cost = (3 + 4 + 4) + (3 + 1 + 1 + 2 + 2) = 20
- **Step 3: randomly select one non-medoid point and recalculate the cost.** Let the randomly selected point be (8, 4). The dissimilarity of each non-medoid point with the medoids – C1 (4, 5) and C2 (8, 4) is calculated and tabulated.



	X	Y	Dissimilarity from C1	Dissimilarity from C2
0	8	7	6	3
1	3	7	3	8
2	4	9	4	9
3	9	6	6	3
4	8	5	4	1
5	5	8	4	7
6	7	3	5	2
7	8	4	-	-
8	7	5	3	2
9	4	5	-	-

- Each point is assigned to that cluster whose dissimilarity is less. So, points 1, 2, and 5 go to cluster C1 and 0, 3, 6, 7, 8 go to cluster C2. The New cost = (3 + 4 + 4) + (2 + 2 + 1 + 3 + 3) = 22 Swap Cost = New Cost – Previous Cost = 22 – 20 and **2 >0** As the swap cost is not less than zero, we undo the swap. Hence (4, 5) and (8, 5) are the final medoids. The clustering would be in the following way.

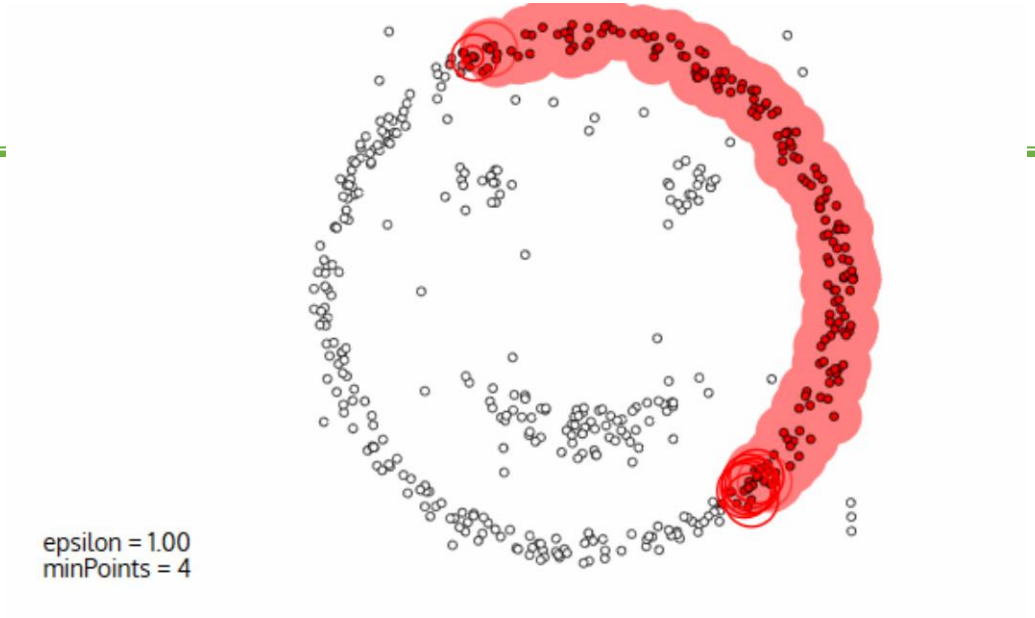


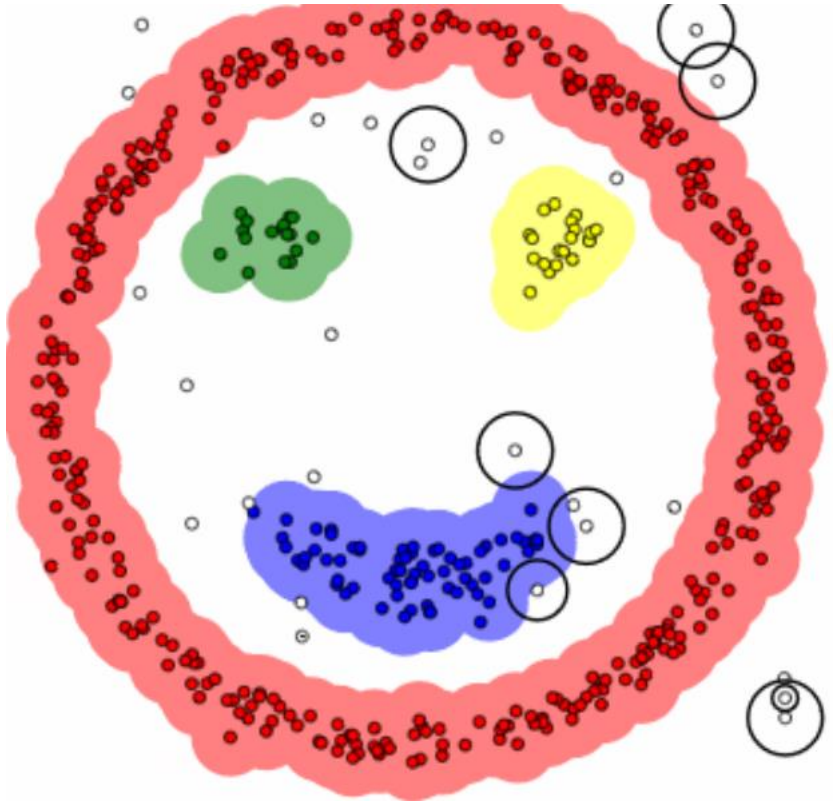
Density Based Spatial Clustering Of Applications with Noise

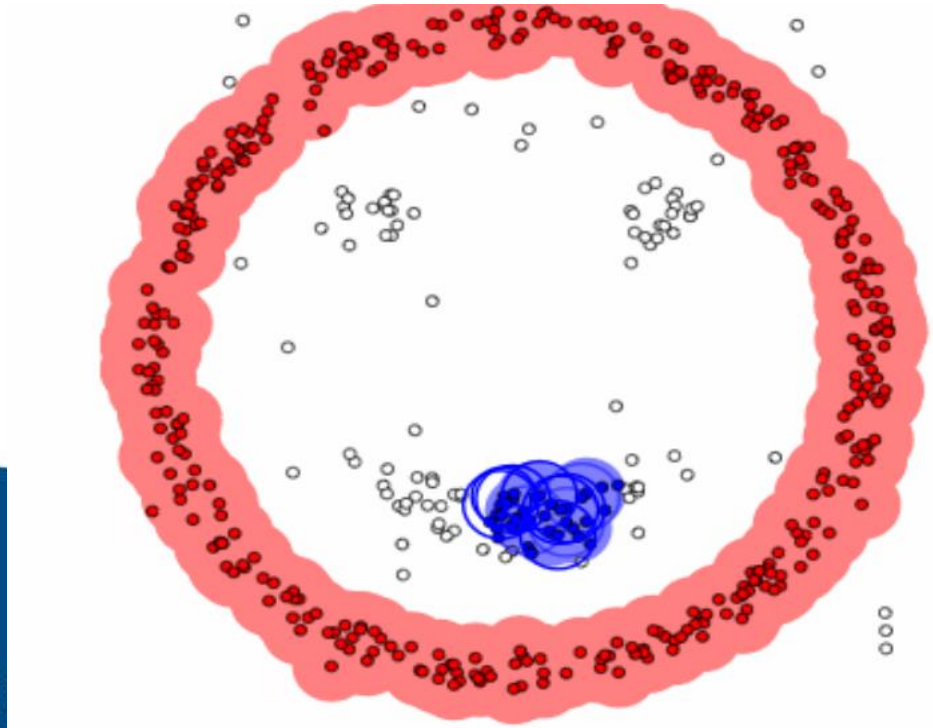
- Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that can be used to identify clusters of arbitrary shape.
- The algorithm works by identifying regions of high density in the data and grouping the data points that belong to these regions into clusters.
- 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.
- DBSCAN has two parameters: epsilon ϵ, which determines the radius of the neighborhood around each data point, and minPts, which determines the minimum number of data points required to form a cluster.

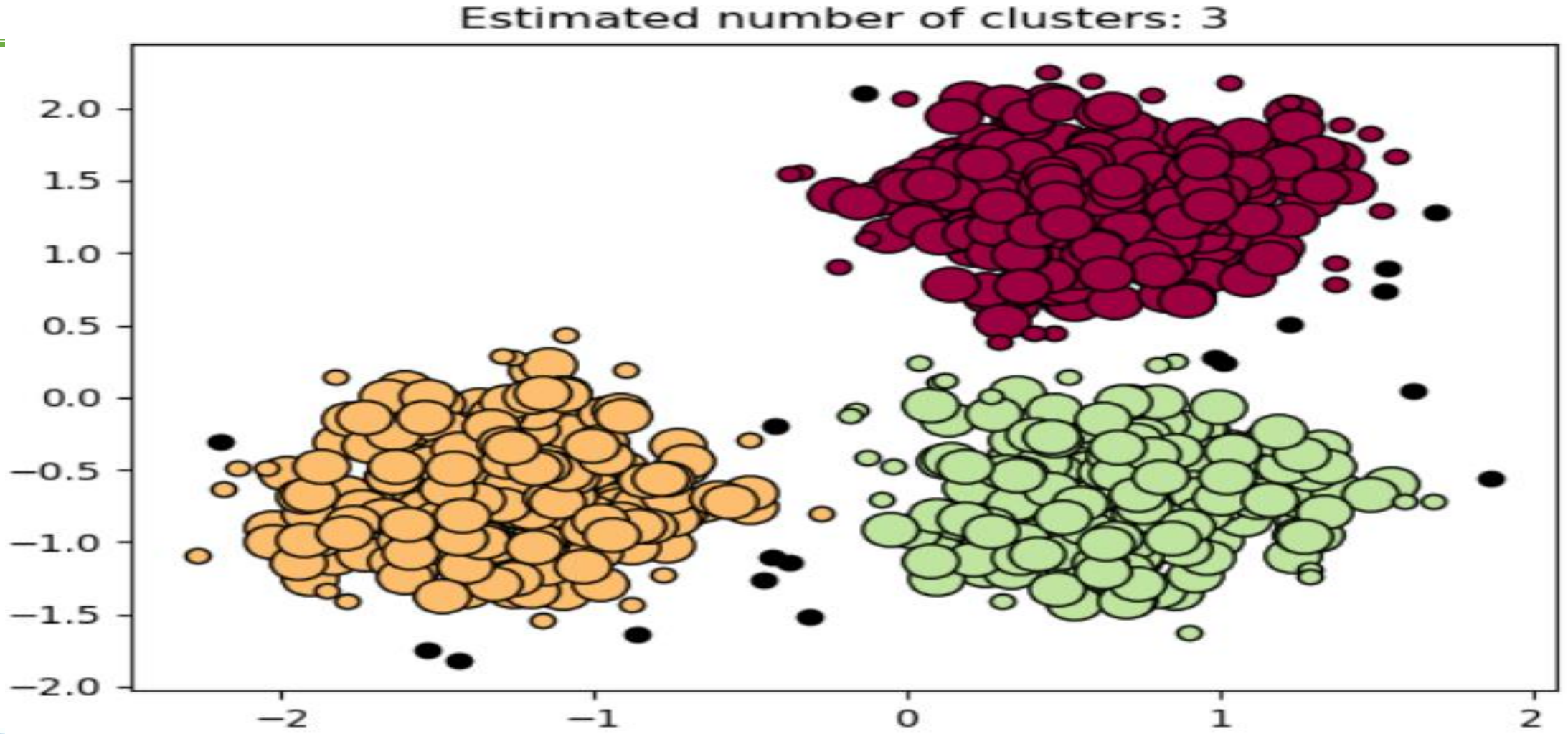
Algorithmic steps for DBSCAN clustering

- The algorithm proceeds by arbitrarily picking up a point in the dataset (until all points have been visited).
- If there are at least ‘minPoint’ 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





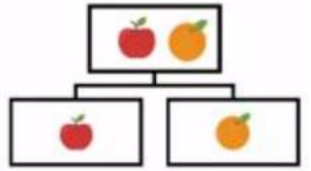

Estimated number of clusters: 3
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2
−1
0
1
2

Types of Clustering

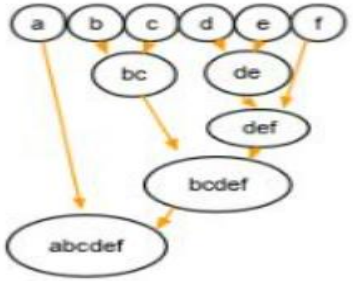
- **Clustering:**
- Clustering is a technique used to group similar data points together based on some similarity metric. The most common clustering algorithms are k-means, hierarchical clustering, and density-based clustering.
- **Type of Clustering**
- Hierarchical clustering
- Partitioning clustering
- Hierarchical clustering is further subdivided into:
- Agglomerative clustering
- Divisive clustering
- Partitioning clustering is further subdivided into:
- K-Means clustering
- Fuzzy C-Means clustering

Hierarchical Clustering

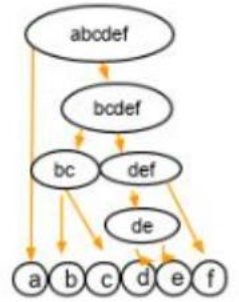
- Hierarchical clustering uses a tree-like structure, like so.



- In **agglomerative clustering**, there is a bottom-up approach. We begin with each element as a separate cluster and merge them into successively more massive clusters.

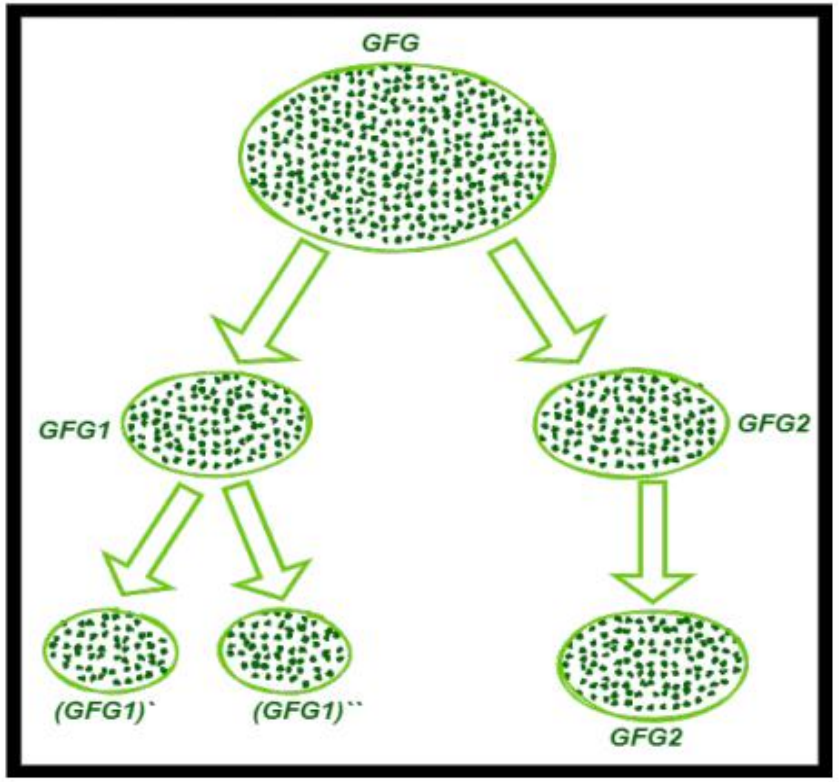


- **Divisive clustering** is a top-down approach. We begin with the whole set and proceed to divide it into successively smaller clusters, as you can see.
- Divisive hierarchical clustering is a type of hierarchical clustering algorithm that works by recursively dividing a dataset into smaller and smaller subsets until each subset contains only one data point.
- Two commonly used methods for performing divisive hierarchical clustering are bisecting K-Means and clustering using Minimum Spanning Tree (MST).



Bisecting K-Means:

- Bisecting K-Means is a type of divisive hierarchical clustering that works by recursively dividing the dataset into two subsets using K-Means clustering.
- The algorithm starts by treating the entire dataset as one cluster and applies K-Means clustering to it.
- The resulting clusters are then bisected into two subsets by running K-Means clustering again on each of the clusters.
- The process continues until the desired number of clusters is reached.



Clustering using Minimum Spanning Tree (MST):

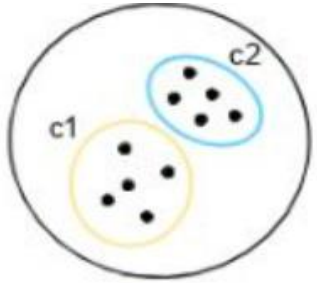
- Clustering using Minimum Spanning Tree (MST) is a type of divisive hierarchical clustering that works by constructing a Minimum Spanning Tree of the dataset and recursively dividing it into smaller subtrees.
- The algorithm starts by constructing a Minimum Spanning Tree of the dataset, which is a tree that connects all the data points with the minimum total edge weight.
- The tree is then recursively bisected into two subtrees using a clustering criterion such as K-Means.
- The process continues until the desired number of clusters is reached.

Bisecting K-Means and clustering using MST

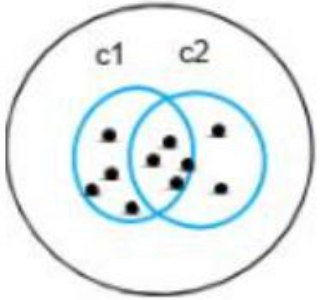
- Both bisecting K-Means and clustering using MST have their advantages and disadvantages.
- Bisecting K-Means is a faster algorithm and can handle large datasets, but it may not always produce the best clustering solution.
- Clustering using MST, on the other hand, produces more accurate clustering solutions but can be computationally expensive for large datasets.
- The choice of algorithm depends on the specific characteristics of the dataset and the desired level of accuracy and computational efficiency.

Partitioning Clustering

- Partitioning clustering is split into two subtypes - K-Means clustering and Fuzzy C-Means.
- In k-means clustering, the objects are divided into several clusters mentioned by the number 'K.' So if we say K = 2, the objects are divided into two clusters, c1 and c2. Here, the features or characteristics are compared, and all objects having similar characteristics are clustered together.



- Fuzzy c-means is very similar to k-means in the sense that it clusters objects that have similar characteristics together.
- In k-means clustering, a single object cannot belong to two different clusters. But in Fuzzy c-means, objects can belong to more than one cluster.



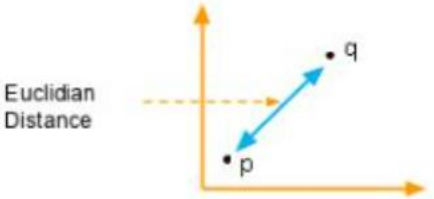
Distance Measure

- Distance measure determines the similarity between two elements and influences the shape of clusters.
- K-Means clustering supports various kinds of distance measures, such as. Euclidean distance measure

1. Manhattan distance measure,
2. squared euclidean distance measure,
3. Cosine distance measure, etc.

- **Euclidean Distance Measure**
- The most common case is determining the distance between two points.
- If we have a point P and point Q, the euclidean distance is an ordinary straight line.
- It is the distance between the two points in Euclidean space. The formula for distance between two points is shown below:
-

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



Evaluation methods

Evaluation methods are used to measure the performance of clustering algorithms.

- **Sum of Squared Errors (SSE):** This measures the sum of the squared distances between each data point and its assigned centroid.
- **Within cluster sum of squares (WCSS)** This measures the sum of the squared distance between each member of the cluster and its centroid.

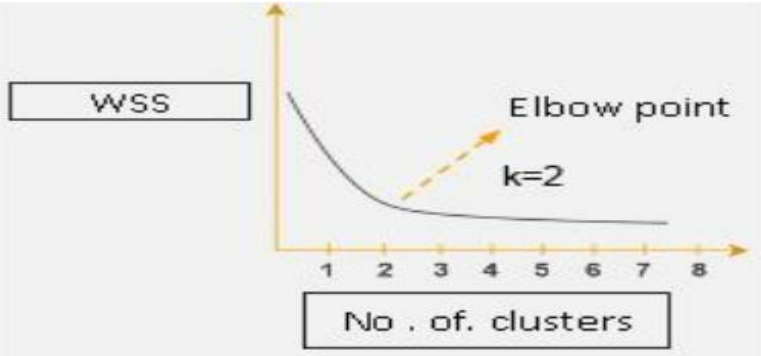
Technique to find optimum no of clusters

- Finding the optimal number of clusters in K-Means clustering is an important task, as it can greatly impact the accuracy of the clustering. Two commonly used methods for determining the optimal number of clusters are the Elbow method and the Silhouette coefficient.

The Elbow method

- The Elbow method is a heuristic technique that calculates the Within cluster sum of squares (WCSS) against the number of clusters, K.
- WCSS is the sum of the squared distances between each data point of the cluster and its assigned centroid.
- The idea is to choose the number of clusters, K, at the ”elbow” of the curve, which is the point where the reduction in WCSS starts to diminish significantly.

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$



Silhouette Coefficient:

- Silhouette coefficient measures how close a point is to other points of its own cluster, compared to points in the next cluster.
- The Silhouette coefficient ranges from -1 to 1, with higher values indicating better clustering solutions.
- You have to measure the Silhouette coefficient of all point.

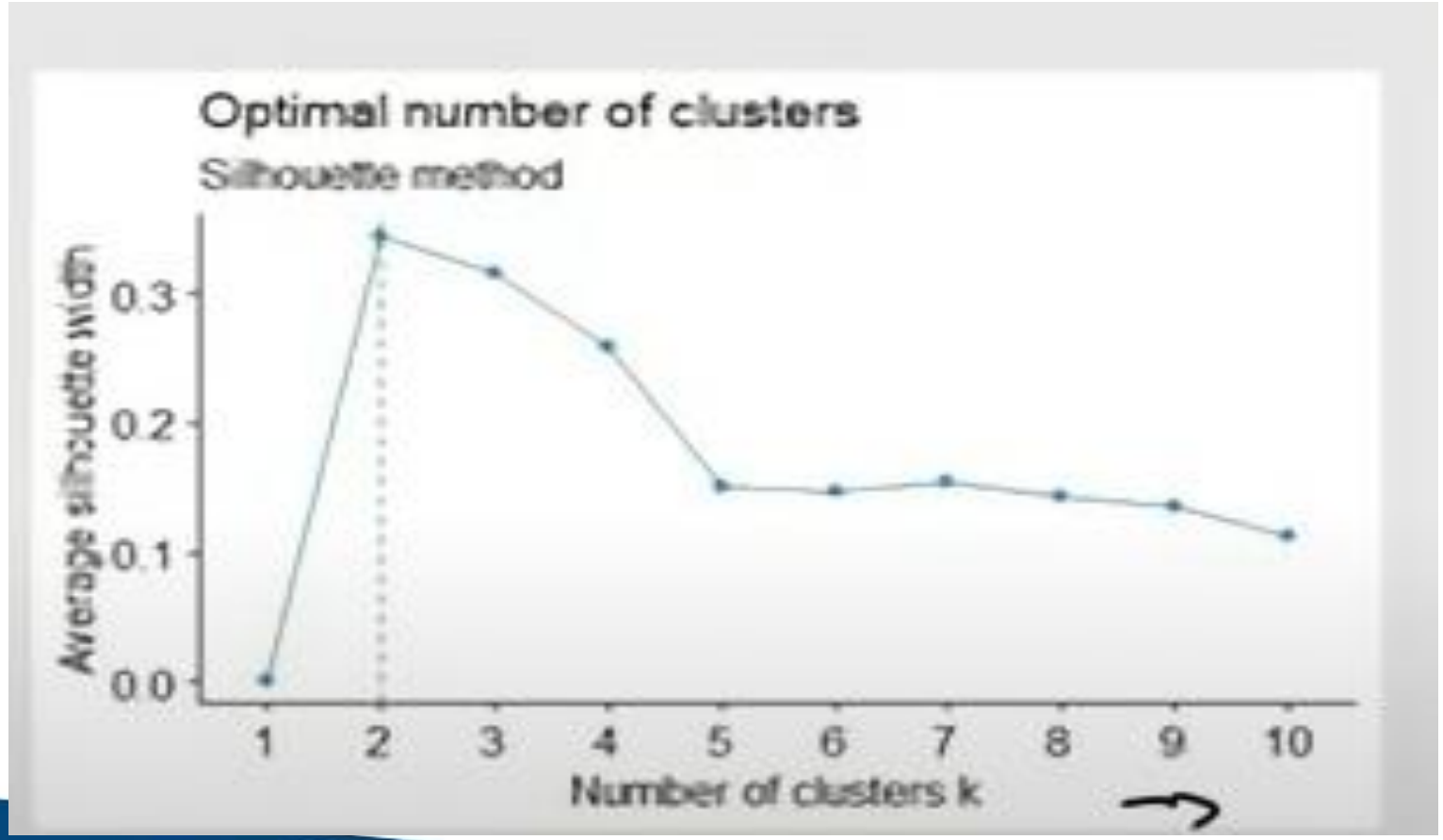
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

a(i) average distance between point i and all other points in same cluster

b(i) average distance between point I and all other points in nearest cluster

Average Silhouette Score for cluster K = Mean(s(i))

- To find the optimal number of clusters using the Silhouette coefficient, we can calculate the coefficient for each value of K and choose the highest average coefficient value.


Optimal number of clusters
Silhouette method
Average silhouette width
0.0
0.1
0.2
0.3
1
2
3
4
5
6
7
8
9
10
Number of clusters k


ALL THE BEST