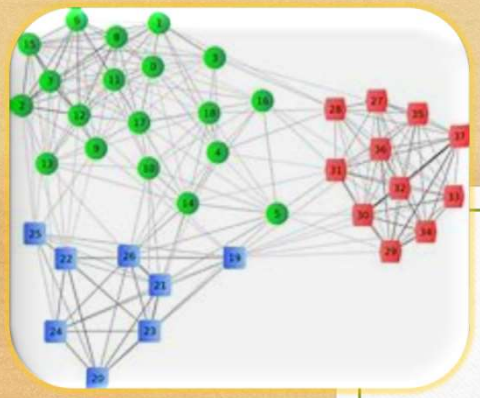


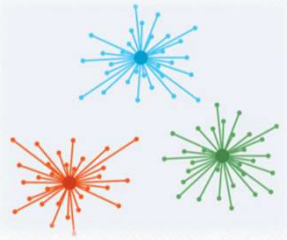
Machine Learning
Clustering

All Clustering algorithms definitions
with Advantage and Disadvantage
Python Codes included

Save this for interview purpose

Affinity Propagation Clustering

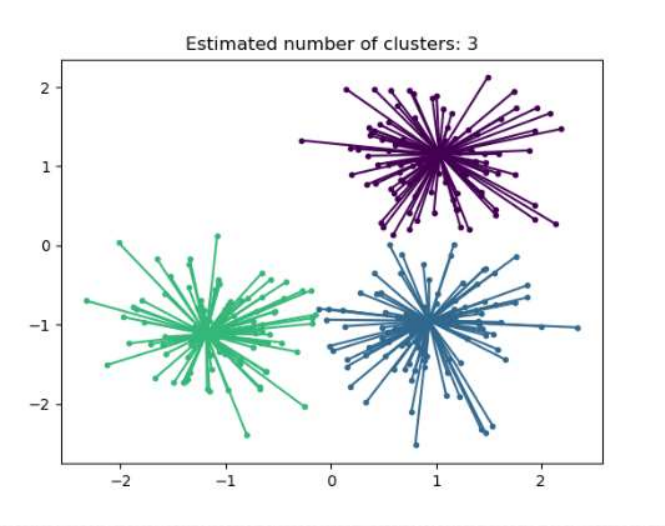
```
from sklearn.cluster import AffinityPropagation
aff = AffinityPropagation(random_state=5)
y_aff=aff.fit_predict(x)
```



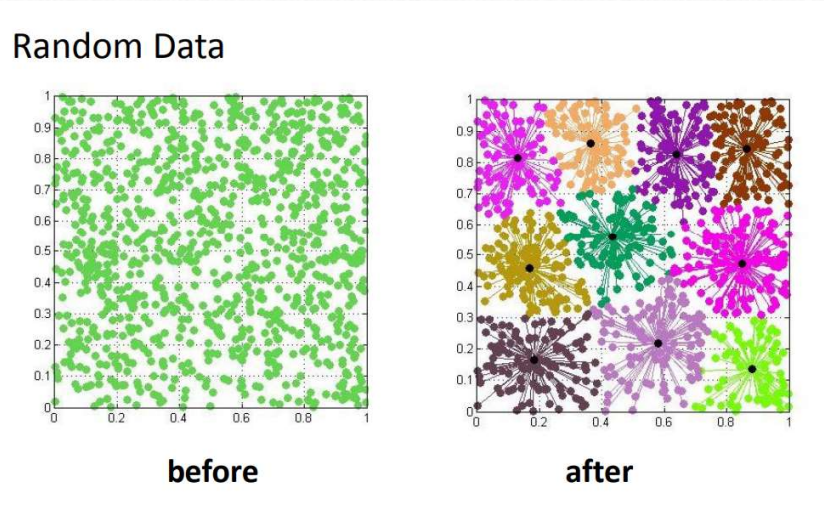
Affinity Propagation creates clusters by sending messages between pairs of samples until convergence.

A dataset is then described using a small number of exemplars, which are identified as those most representative of other samples.

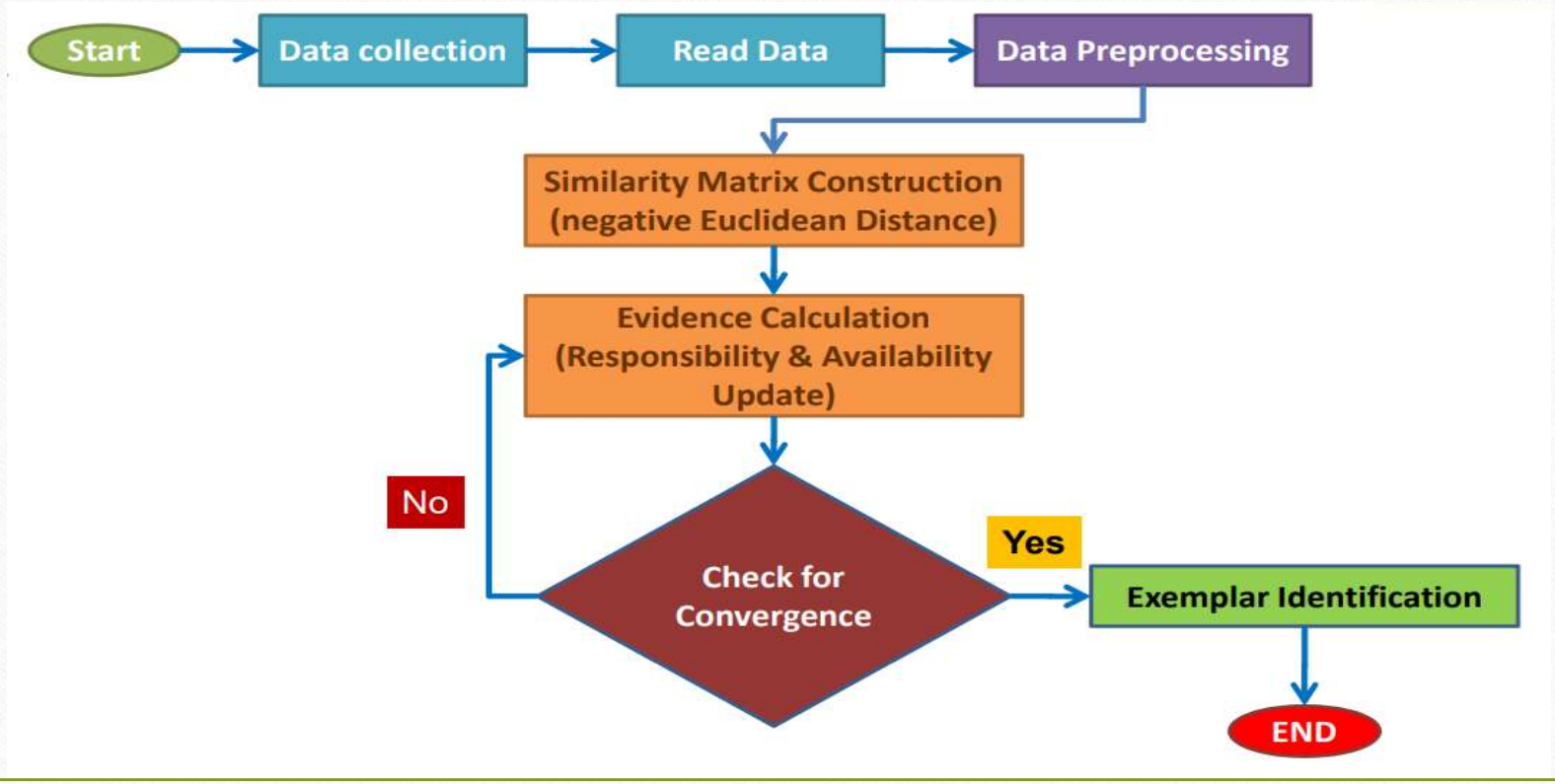
The messages sent between pairs represent the suitability for one sample to be the exemplar of the other, which is updated in response to the values from other pairs.



- Affinity Propagation works based on similarities between data points
- In Affinity Propagation there are two kinds of message exchanged between data points, they are Responsibility and Availability
- Considers all data points as potential cluster center
- Identifies a set of cluster center that represents the dataset
- Takes as input a collection of similarities matrix
- Deterministic which is its clustering results do not depend on initialization



Steps of Affinity Propagation



There are some of the developments related to the affinity propagations. They are

- **Adaptive Affinity Propagation**
- **Partition Affinity Propagation**
- **Soft Constraint Affinity Propagation Algorithm**
- **Fuzzy Statistic Affinity Propagation**

According to the testing results, it can be inferred that among several affinity propagation development, **Partition Affinity Propagation** is **slightly faster** since it uses an approach to divide matrix into several part to reduce the iteration numbers.

On the other hand **Adaptive Affinity Propagation** is much **more tolerant to errors**, it can remove the oscillation when it occurs where the occupance of oscillation will bring the algorithm to fail to converge. Adaptive Affinity propagation is **more stable** than the other since it can deal with error which the other can not.

And **Fuzzy Statistic Affinity Propagation** can **produce smaller number of cluster** compared to the other since it produces its own preferences using fuzzy iterative methods.

For future work, in order to produce a better algorithm of affinity propagation, consider to try **combining two or three algorithm** such as adaptive and partition affinity propagation or any other affinity propagation. So it can run **faster and stabler** in generating cluster center from massive data.

Advantages and Disadvantages of Affinity propagation Clustering

Advantages

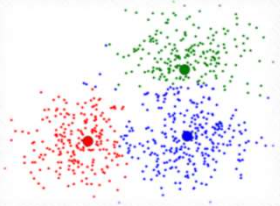
- Has better performance and lower clustering error

Disadvantages

- It is quite slow and memory-heavy, making it difficult to scale to larger datasets. We do not have any direct control on the number of clusters but in some applications, we need a specific number of clusters.
- It also assumes the true underlying clusters are globular

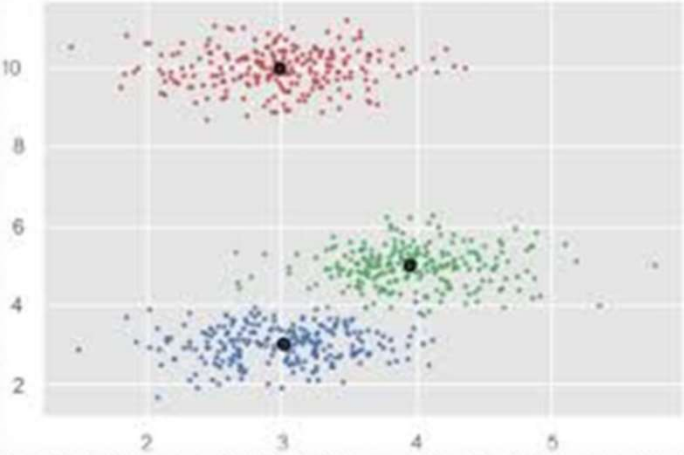
Mean Shift Clustering

```
from sklearn.cluster import MeanShift
MS = MeanShift(bandwidth=2).fit(x)
y_MS=MS.fit_predict(x)
```



Mean-shift algorithm basically assigns the datapoints to the clusters iteratively by shifting points towards the highest density of datapoints i.e. cluster centroid.

The difference between K-Means algorithm and Mean-Shift is that later one does not need to specify the number of clusters in advance because the number of clusters will be determined by the algorithm w.r.t data.



Working of Mean-Shift Algorithm

We can understand the working of Mean-Shift clustering algorithm with the help of following steps

---

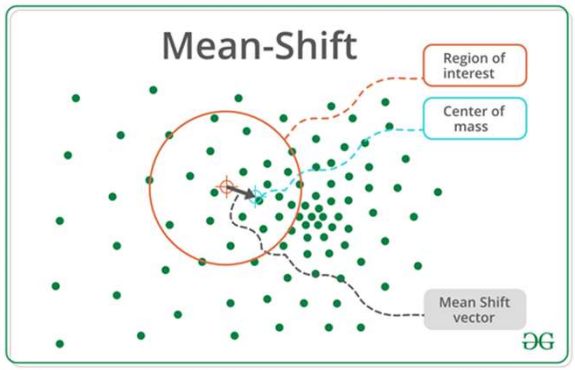
Step 1 − First, start with the data points assigned to a cluster of their own.

Step 2 − Next, this algorithm will compute the centroids.

Step 3 − In this step, location of new centroids will be updated.

Step 4 − Now, the process will be iterated and moved to the higher density region.

Step 5 − At last, it will be stopped once the centroids reach at position from where it cannot move further.



Advantages and Disadvantages of Mean Shift Clustering

Advantages

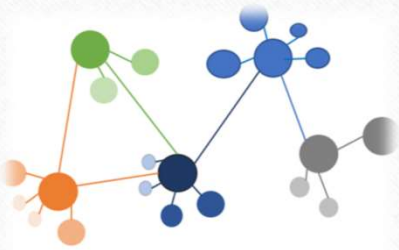
- It does not need to make any model assumption as like in K-means or Gaussian mixture.
- It can also model the complex clusters which have nonconvex shape.
- It only needs one parameter named bandwidth which automatically determines the number of clusters.
- There is no issue of local minima as like in K-means.
- No problem generated from outliers.

Disadvantages

- Mean-shift algorithm does not work well in case of high dimension, where number of clusters changes abruptly.
- We do not have any direct control on the number of clusters but in some applications, we need a specific number of clusters.
- It cannot differentiate between meaningful and meaningless modes.

Spectral Clustering

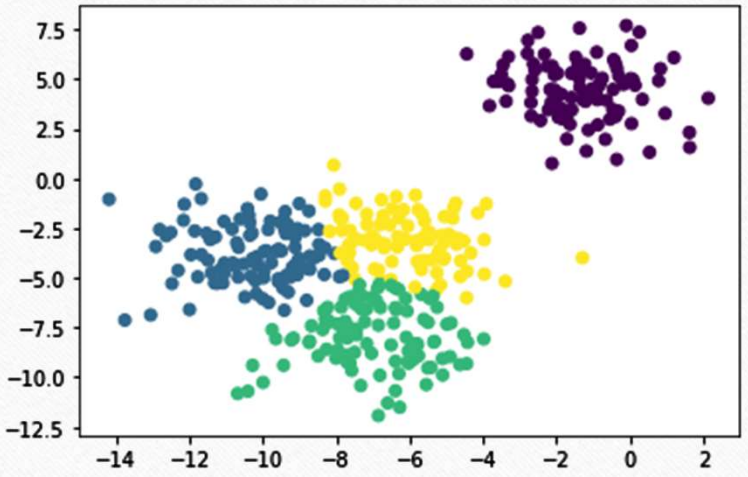
```
from sklearn.cluster import SpectralClustering
SC = SpectralClustering(n_clusters=2, assign_labels='discretize', random_state=0).fit(x)
y_SC=SC.fit_predict(x)
```



In spectral clustering, data points are treated as nodes of a graph. Thus, spectral clustering is a graph partitioning problem.

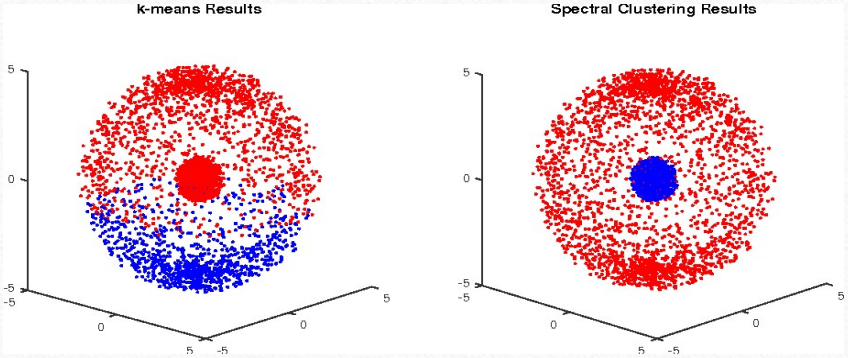
The nodes are then mapped to a low-dimensional space that can be easily segregated to form clusters. No assumption is made about the shape/form of the clusters.

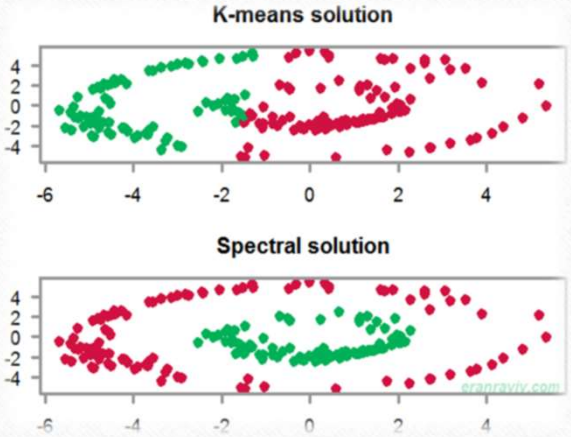
The goal of spectral clustering is to cluster data that is connected but not necessarily compact or clustered within convex boundaries.



Difference between K means and Spectral Clustering

K means Clustering	Spectral Clustering
K-means algorithm generally assumes that the clusters are spherical or round i.e. within k-radius from the cluster centroid	Spectral clustering helps us overcome two major problems in clustering: one being the shape of the cluster and the other is determining the cluster centroid
In K means, many iterations are required to determine the cluster centroid	In Spectral, the clusters do not follow a fixed shape or pattern





Advantages and Disadvantages of Mean Shift Clustering

Advantages

- **Applicable for high dimensional datasets.** One of the main advantages that spectral clustering has over other clustering algorithms is that it can be used on high-dimensional datasets with many features.

- **Not strong assumptions about cluster shape.** Spectral clustering does not make strong assumptions about the shape of the clusters in the data. That means that it is appropriate to use spectral clustering even when you suspect the clusters in your data may be irregularly shaped.

- **Can sometimes handle categorical variables.** Some implementations of spectral clustering can handle cases where you have mixed data types, such as cases where you have categorical variables in your data. This is in part because spectral clustering uses similarity metrics rather than distance metrics to determine which points have more in common.

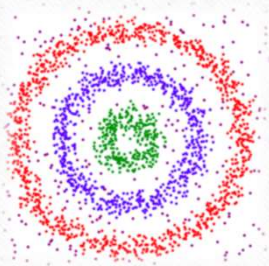
Disadvantages

- **Relatively slow.** One disadvantage of spectral clustering is that it is relatively slow compared to other clustering algorithms like k-means clustering. If you have a dataset with many, many data points then you may be better off using a faster algorithm.

- **Need to select the number of clusters.** As with many other clustering algorithms, spectral clustering requires you to select the number of clusters that should be used for your dataset. This can be difficult to do if you do not have strong intuition about the true number of clusters in the data.

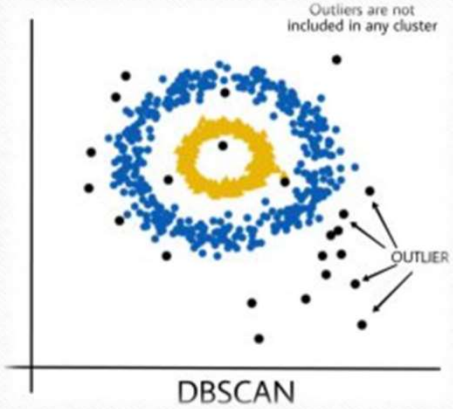
DBSCAN Clustering

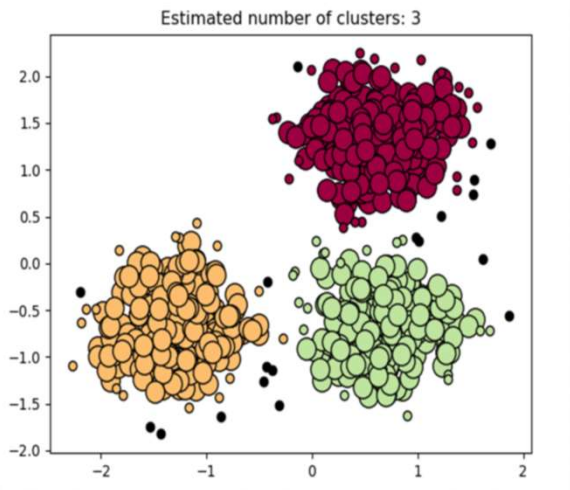
```
from sklearn.cluster import DBSCAN
DB = DBSCAN(eps=3, min_samples=2).fit(x)
y_DB=DB.fit_predict(x)
```



Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise - **(DBSCAN)** clustering.

DBSCAN is a density-based clustering algorithm that works on the assumption that clusters are dense regions in space separated by regions of lower density. It groups 'densely grouped' data points into a single cluster.

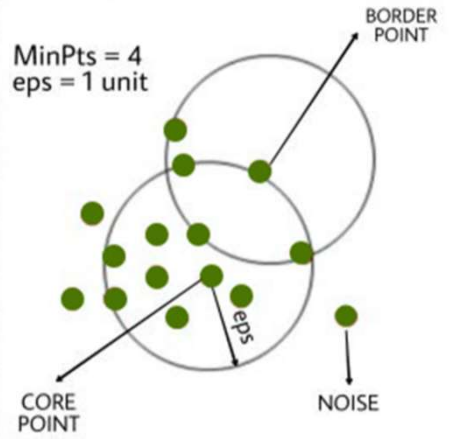




DBSCAN algorithm can be abstracted in the following steps:

1. Find all the neighbor points within **eps** and identify the core points or visited with more than **MinPts** neighbors.

eps: It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the **k-distance graph.**



MinPts: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

2. For each core point if it is not already assigned to a cluster, create a new cluster.

3. Find recursively all its density connected points and assign them to the same cluster as the core point.

4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Advantages and Disadvantages of DBSCAN Clustering

Advantages

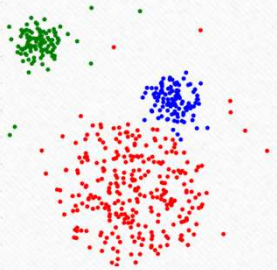
- DBSCAN is great at separating high-density clusters from low-density clusters,
- DBSCAN can be used to detect clusters that are oddly or irregularly shaped, such as clusters that are ring-shaped.
- DBSCAN is used to handle clusters of multiple sizes and structures and is not powerfully influenced by noise or outliers.

Disadvantages

- DBSCAN struggles with clusters of similar density.
- Struggles with high dimensionality data. If given data with too many dimensions, DBSCAN suffers.

OPTICS Clustering

```
from sklearn.cluster import OPTICS
OP=OPTICS(min_samples=2).fit(x)
y_OP=OP.fit_predict(x)
```

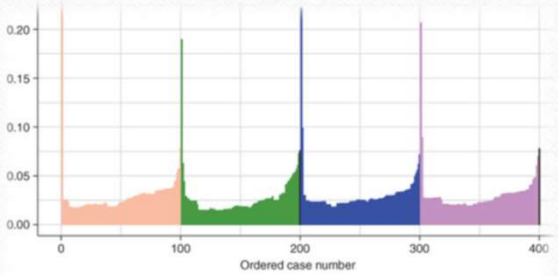


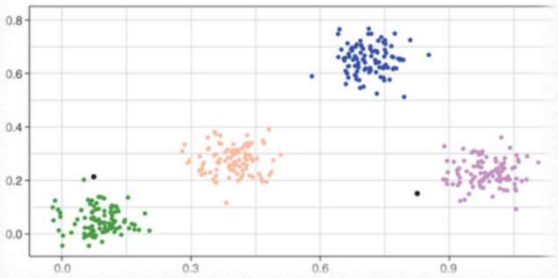
Ordering Points To Identify Cluster Structure - (OPTICS)

OPTICS is a density-based clustering algorithm, similar to DBSCAN, but it can extract clusters of varying densities and shapes.

The main idea behind OPTICS is to extract the clustering structure of a dataset by identifying the density-connected points.

The algorithm builds a density-based representation of the data by creating an ordered list of points called the **reachability plot**. Each point in the list is associated with a reachability distance, which is a measure of how easy it is to reach that point from other points in the dataset. Points with similar reachability distances are likely to be in the same cluster.





OPTICS Clustering v/s DBSCAN Clustering:

1. Memory Cost : The OPTICS clustering technique requires more memory as it maintains a priority queue (Min Heap) is used to find Reachability Distance. Where as DBSCAN requires less memory space.

2. Handling varying densities: OPTICS can identify clusters of different sizes and shapes more effectively than DBSCAN in datasets with varying densities.

3. Noise handling: OPTICS may be less effective when compared to DBSCAN at identifying small clusters that are surrounded by noise points, as these clusters may be merged with the noise points in the reachability distance plot.

4. Runtime complexity: The runtime complexity of OPTICS is generally higher than that of DBSCAN

5. Fewer Parameters: OPTICS has fewer parameters when compared to DBSCAN

Advantages and Disadvantages of OPTICS Clustering

Advantages

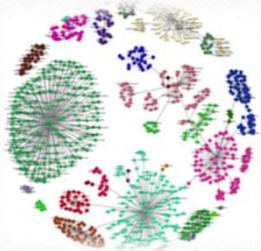
- OPTICS clustering doesn't require a predefined number of clusters in advance
- Clusters can be of any shape, including non-spherical ones

Disadvantages

- It fails if there are no density drops between clusters
- It is also sensitive to parameters that define density (radius and the minimum number of points) and proper parameter settings require domain knowledge.

BIRCH Clustering

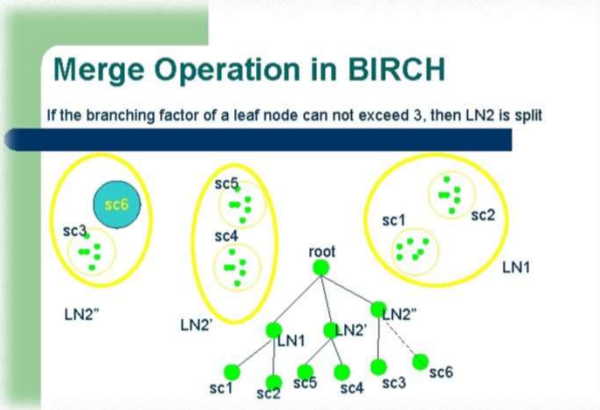
```
from sklearn.cluster import Birch
brc=Birch(n_clusters=None)
y_brc=brc.fit_predict(x)
```



Balanced Iterative Reducing and Clustering using Hierarchies - (BIRCH)

Clustering algorithms like K-means clustering do not perform clustering very efficiently and it is difficult to process large datasets with a limited amount of resources. So, regular clustering algorithms do not scale well in terms of running time and quality as the size of the dataset increases. This is where BIRCH clustering comes in.

BIRCH clustering algorithm can cluster large datasets by first generating a small and compact summary of the large dataset that retains as much information as possible. This smaller summary is then clustered instead of clustering the larger dataset.

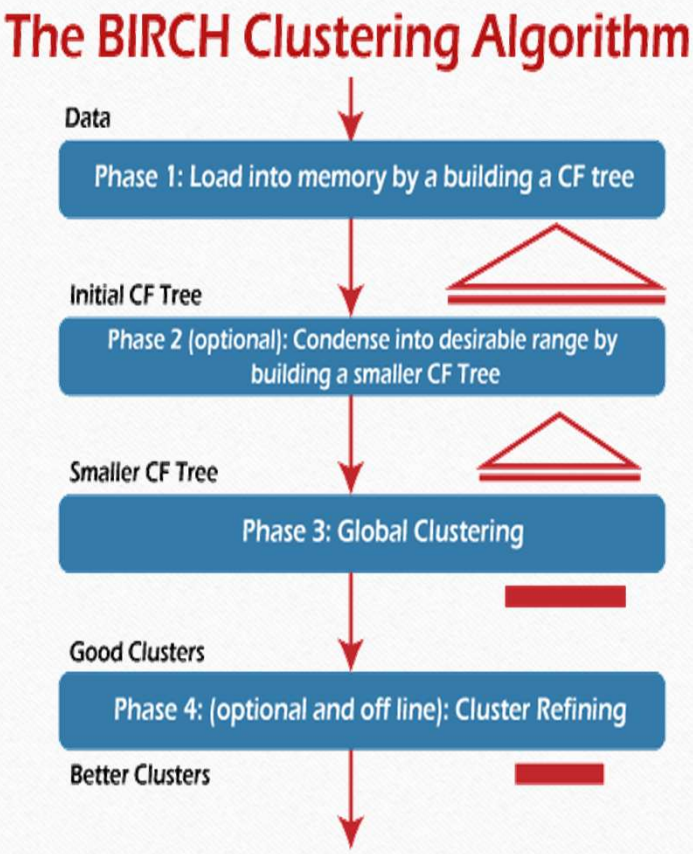


The **BIRCH** clustering algorithm consists of two stages:

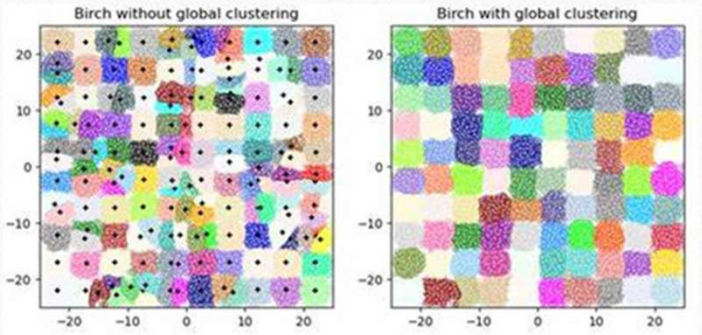
1.Building the CF Tree: BIRCH summarizes large datasets into smaller, dense regions called Clustering Feature (CF) entries. Formally, a Clustering Feature entry is defined as an ordered triple (N, LS, SS) where 'N' is the number of data points in the cluster, 'LS' is the linear sum of the data points, and 'SS' is the squared sum of the data points in the cluster. A CF entry can be composed of other CF entries. Optionally, we can condense this initial CF tree into a smaller CF.

2.Global Clustering: Applies an existing clustering algorithm on the leaves of the CF tree. A CF tree is a tree where each leaf node contains a sub-cluster. Every entry in a CF tree contains a pointer to a child node, and a CF entry made up of the sum of CF entries in the child nodes. Optionally, we can refine these clusters.

Due to this two-step process, BIRCH is also called **Two-Step Clustering.**



Advantages and Disadvantages of BIRCH Clustering



Advantages

- BIRCH is useful for performing precise Clustering on large datasets
- An main advantage of BIRCH is its ability to incrementally and dynamically cluster incoming, multi-dimensional metric data points to produce the best quality clustering for a given set of resources (memory and time constraints). In most cases, BIRCH only requires a single scan of the database.

Disadvantages

- BIRCH has one major drawback, it can only process metric attributes. A metric attribute is an attribute whose values can be represented in Euclidean space, i.e., no categorical attributes should be present.