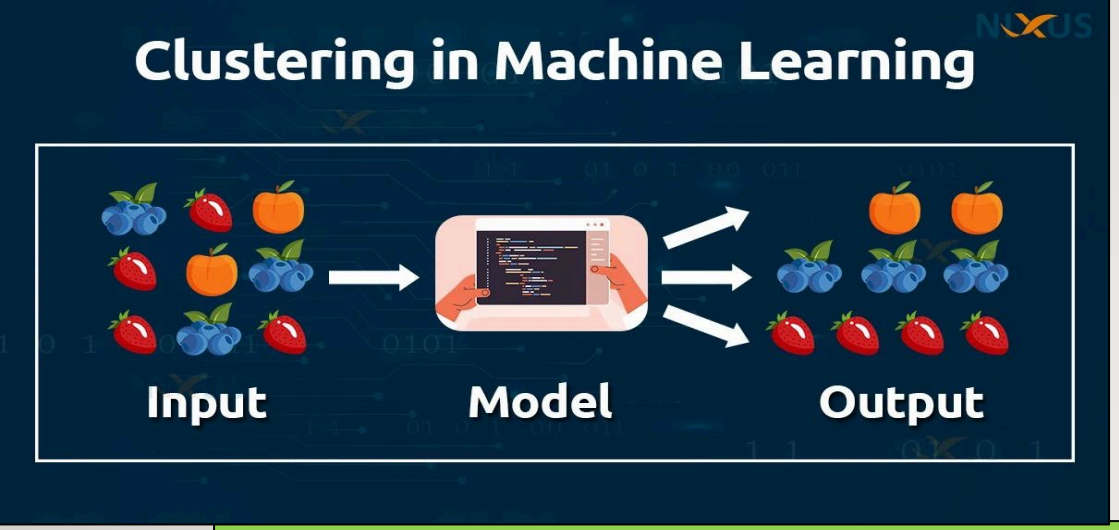
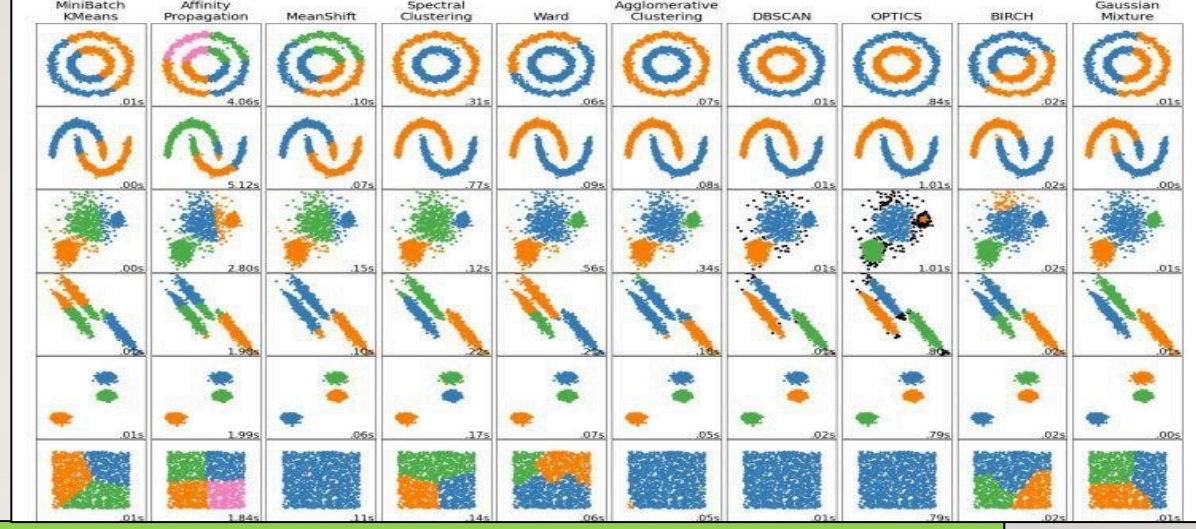
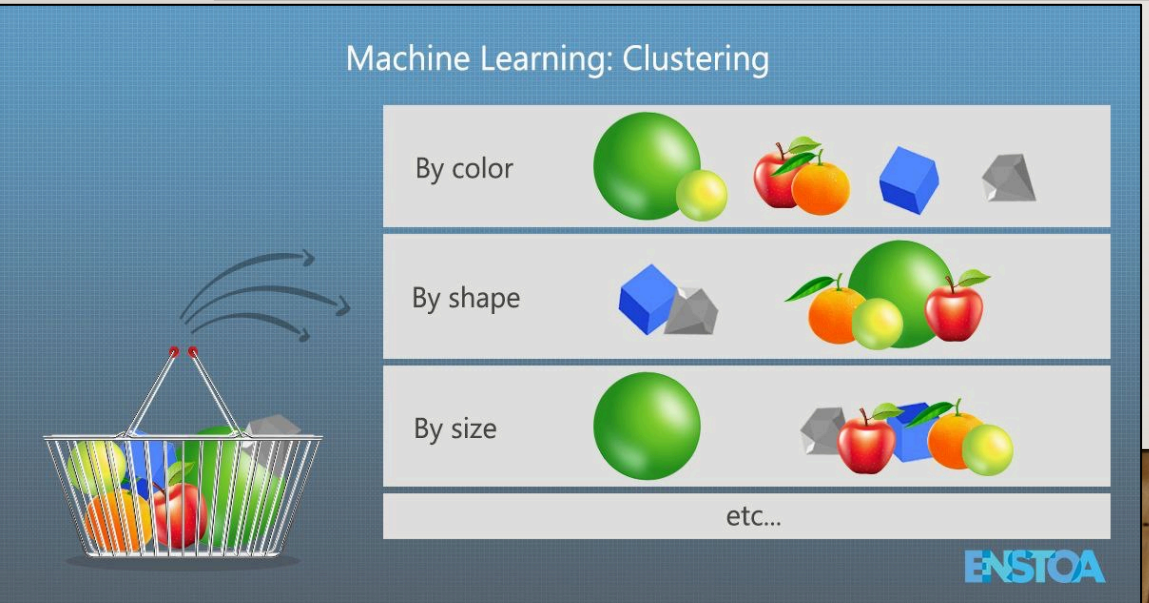
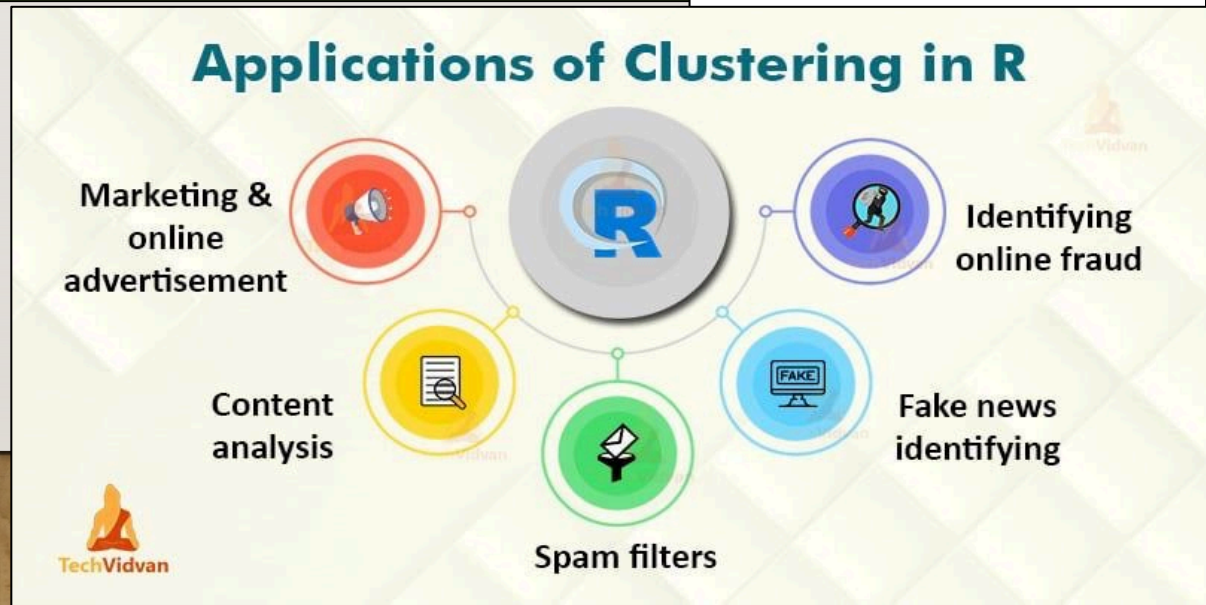




CLUSTERING TYPES

முத்து வசுமதி
20/09/23





PRESENTATION FLOW

- Overview
- Inductive & Transductive
- Dataset Geometry
- ~~K-Means~~
- Affinity Propagation
- Mean Shift
- Spectral
- DBSCAN
- ~~Hierarchical~~
- OPTICS
- BIRCH

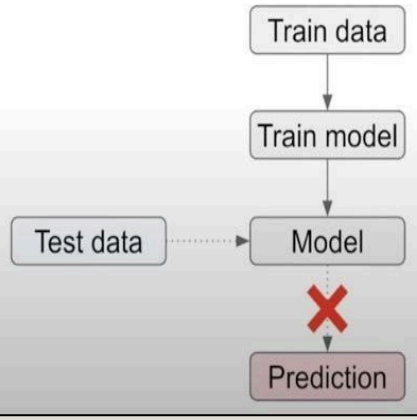
OVER VIEW

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, transductive	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points

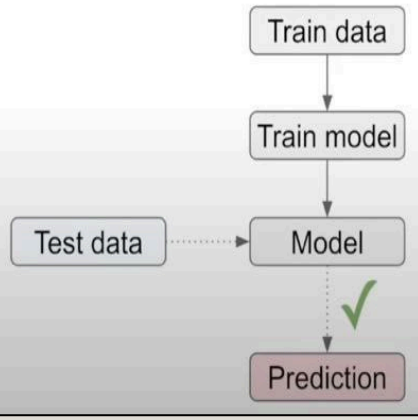
INDUCTIVE & TRANSDUCTIVE

Inductive vs. transductive

"A **transductive** (contrasted with inductive) machine learning method is designed to model a specific dataset, but **not to apply that model to unseen data**" - *Scikit-learn*



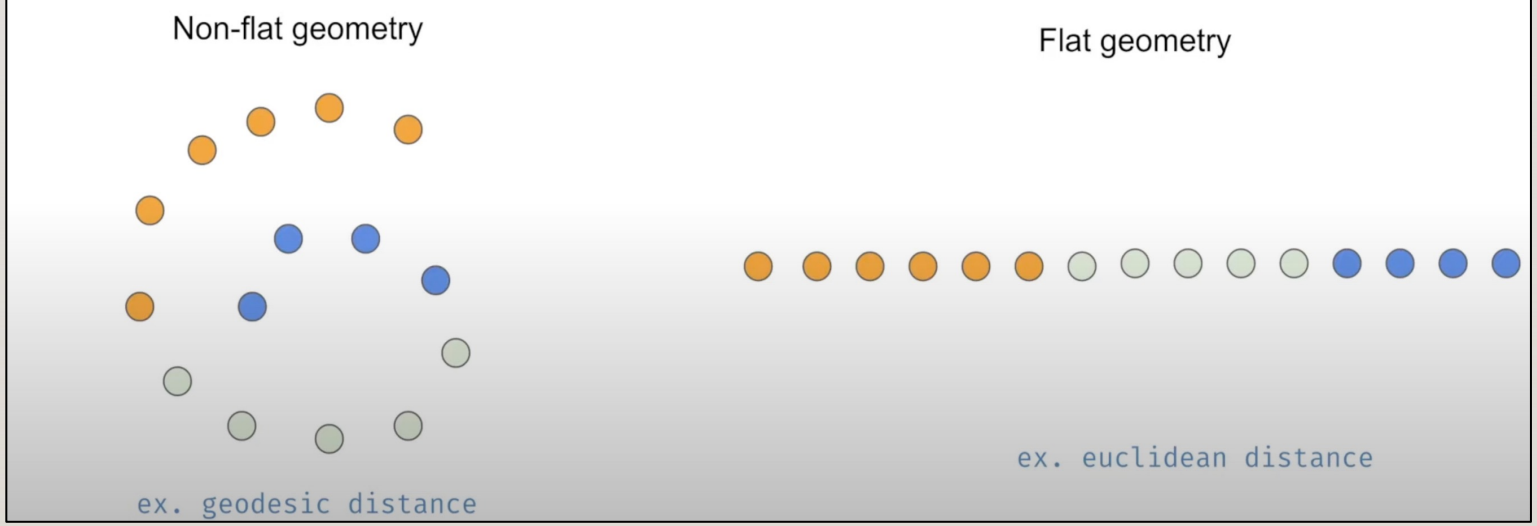
"**Inductive** (contrasted with transductive) machine learning builds a model of some data that can then be applied to new instances. Most estimators in Scikit-learn are inductive, having predict and/or transform methods." - *Scikit-learn*



DATASET GEOMETRY

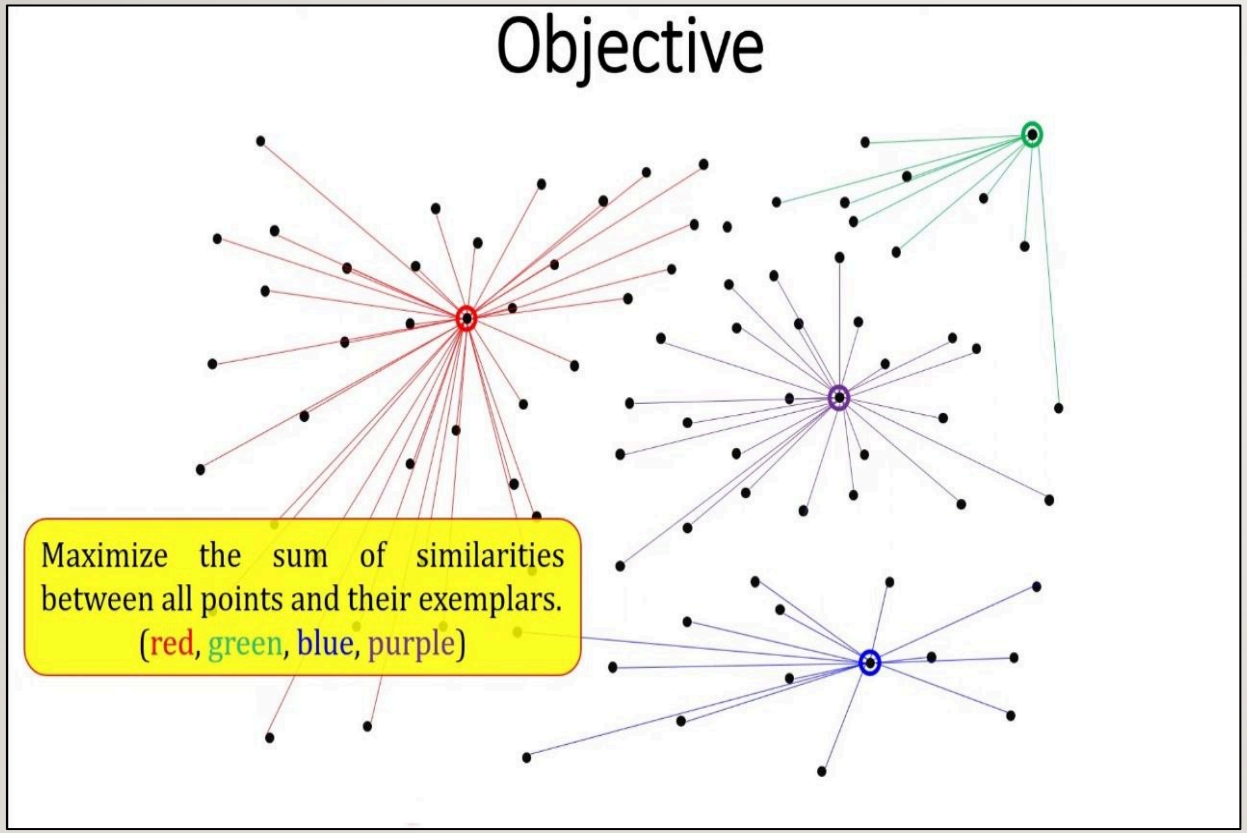
Flat vs. non-flat geometry

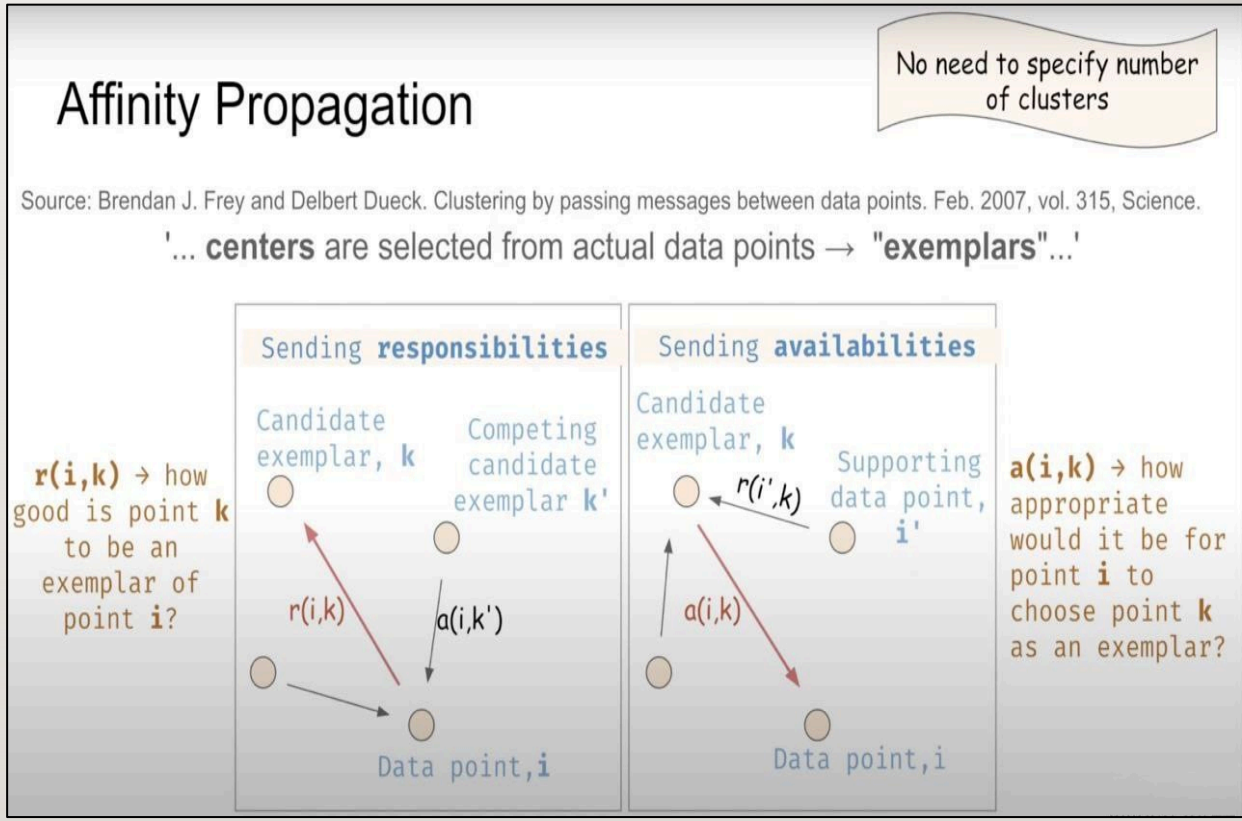
"Non-flat geometry clustering is useful when the clusters have a specific shape, i.e. a non-flat manifold, and the standard euclidean distance is not the right metric." - *Scikit-learn*



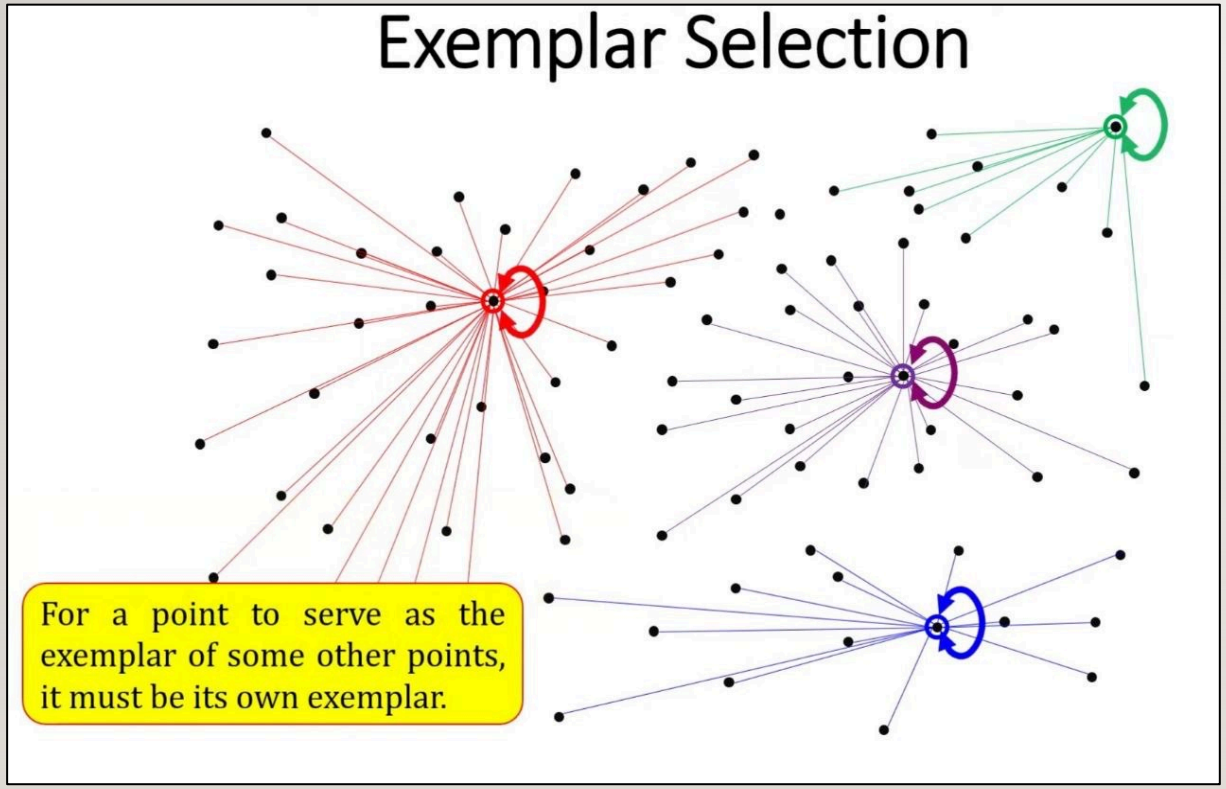
AFFINITY PROPAGATION

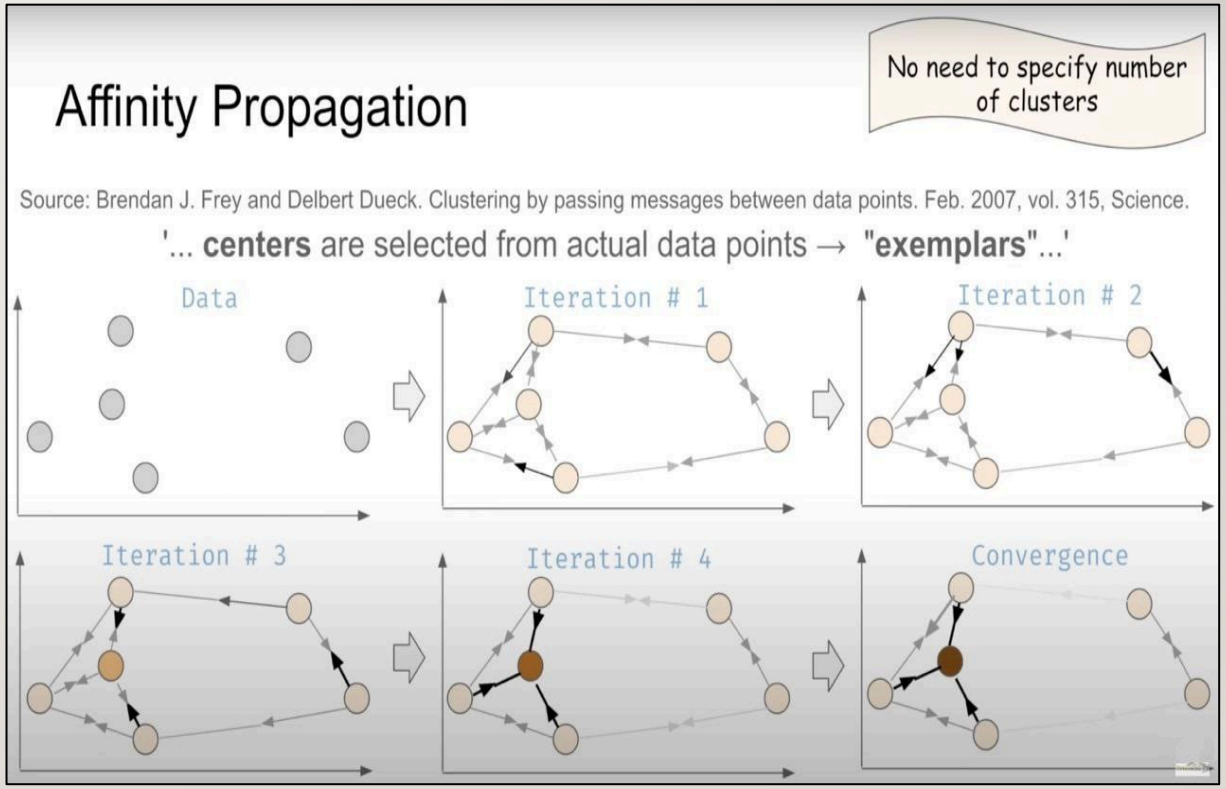
Affinity Propagation is a clustering algorithm that identifies a set of exemplars among the data points and forms clusters around these exemplars. Unlike other clustering methods that require the user to specify the number of clusters in advance, it automatically determines the number of clusters based on the data.





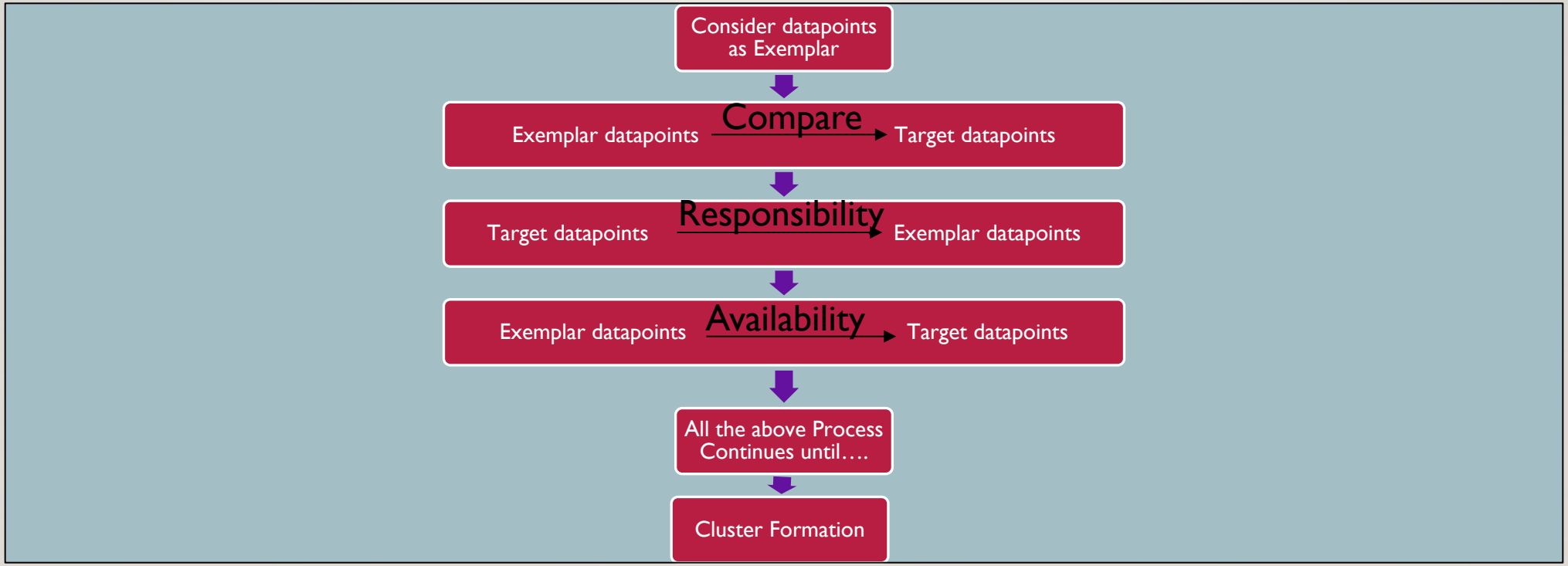
AFFINITY PROPAGATION





முத்து வசுமதி

AFFINITY PROPAGATION



AFFINITY PROPAGATION

Affinity Propagation

Responsibility matrix Similarity matrix

$$r(i,k) \leftarrow s(i,k) - max[a(i,k') + s(i,k') \forall k' \neq k]$$

Responsibility of sample **k** to be exemplar of sample **i**

Similarity between samples **i** and **k**

$$a(i,k) \leftarrow min[0, r(k,k) + \sum_{i' \ s.t. \ i' \notin \{i,k\}} r(i',k)]$$

Availability of sample **k** to be exemplar of sample **i**

Damping factor — to avoid numerical oscillations when updating messages

$$r_{t+1}(i,k) = \lambda \cdot r_t(i,k) + (1-\lambda) \cdot r_{t+1}(i,k)$$

$$a_{t+1}(i,k) = \lambda \cdot a_t(i,k) + (1-\lambda) \cdot a_{t+1}(i,k)$$

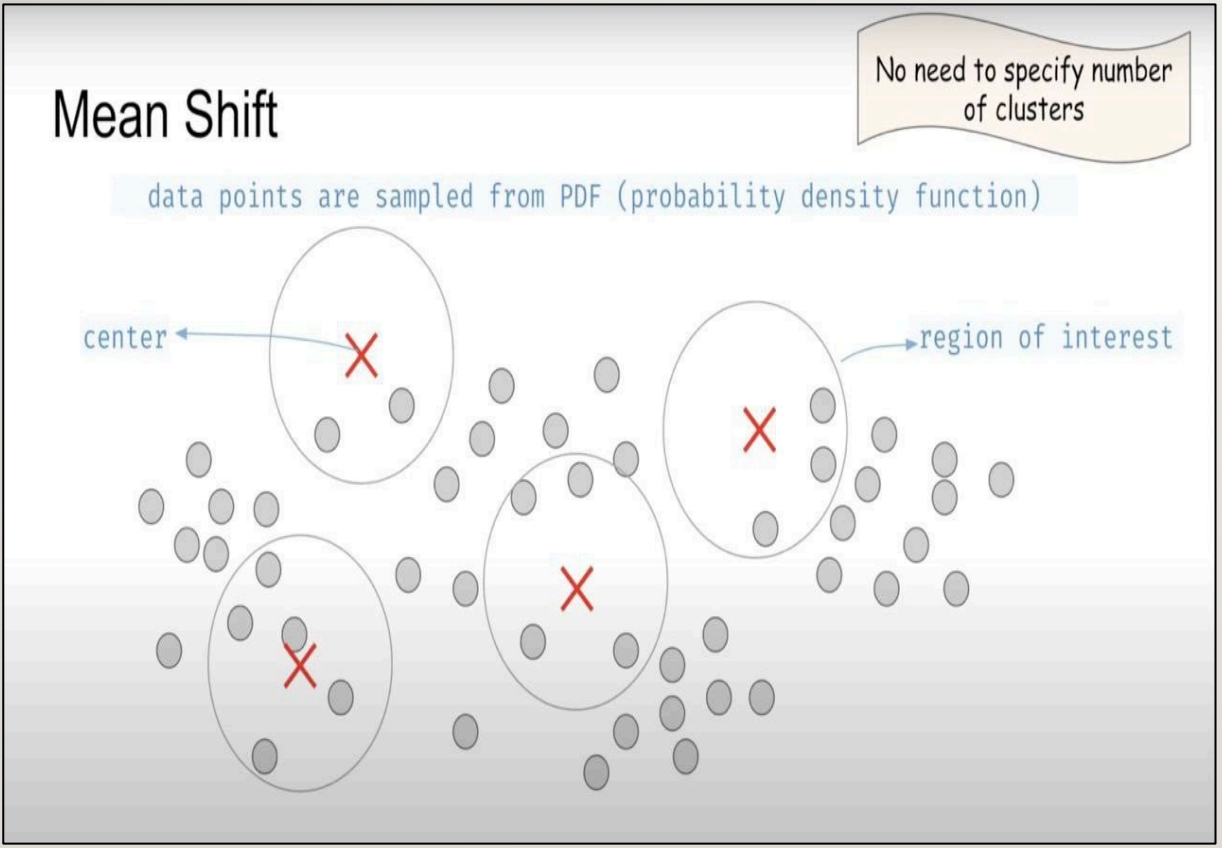
AFFINITY PROPAGATION- STRENGTH & WEAKNESS

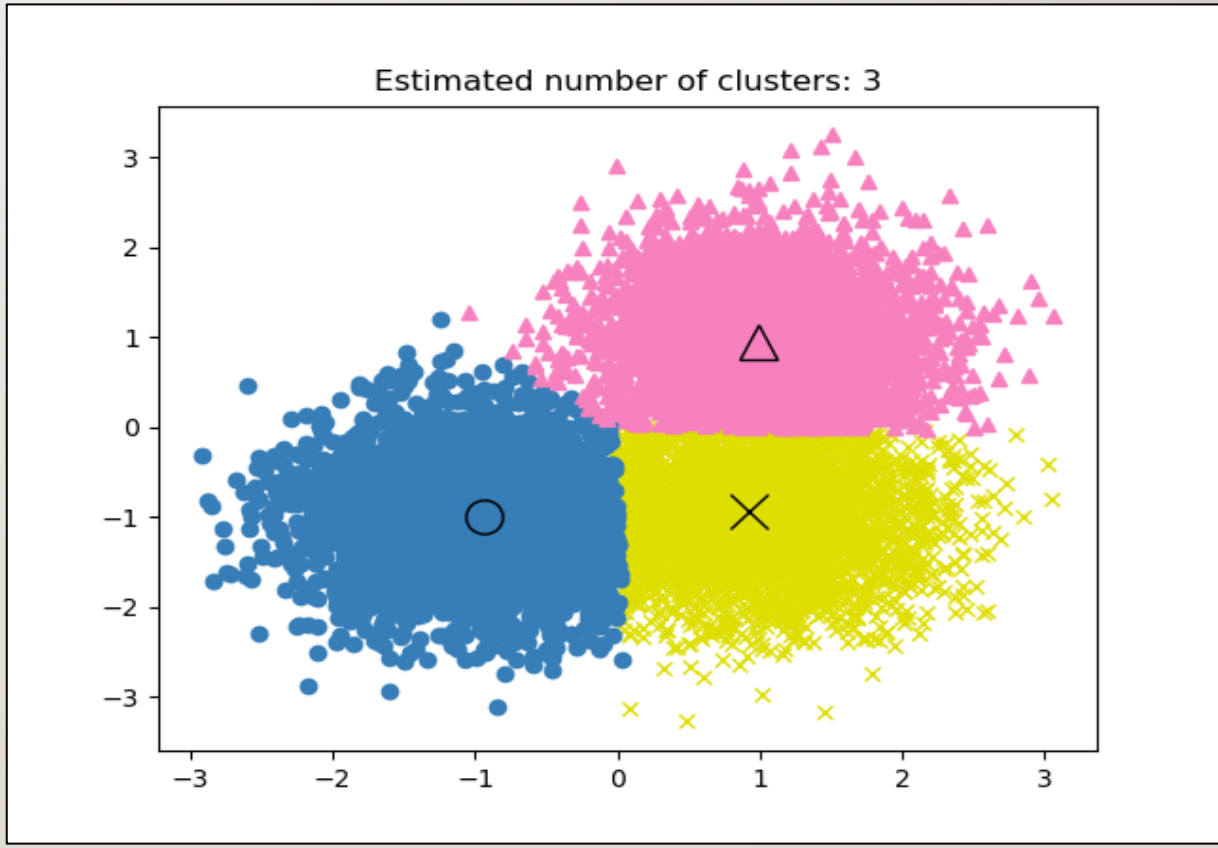
- **Strengths**: The user doesn't need to specify the number of clusters (but does need to specify 'sample preference' and 'damping' hyperparameters).

- **Weaknesses:** Quite slow and memory-heavy, making it difficult to scale to larger datasets.

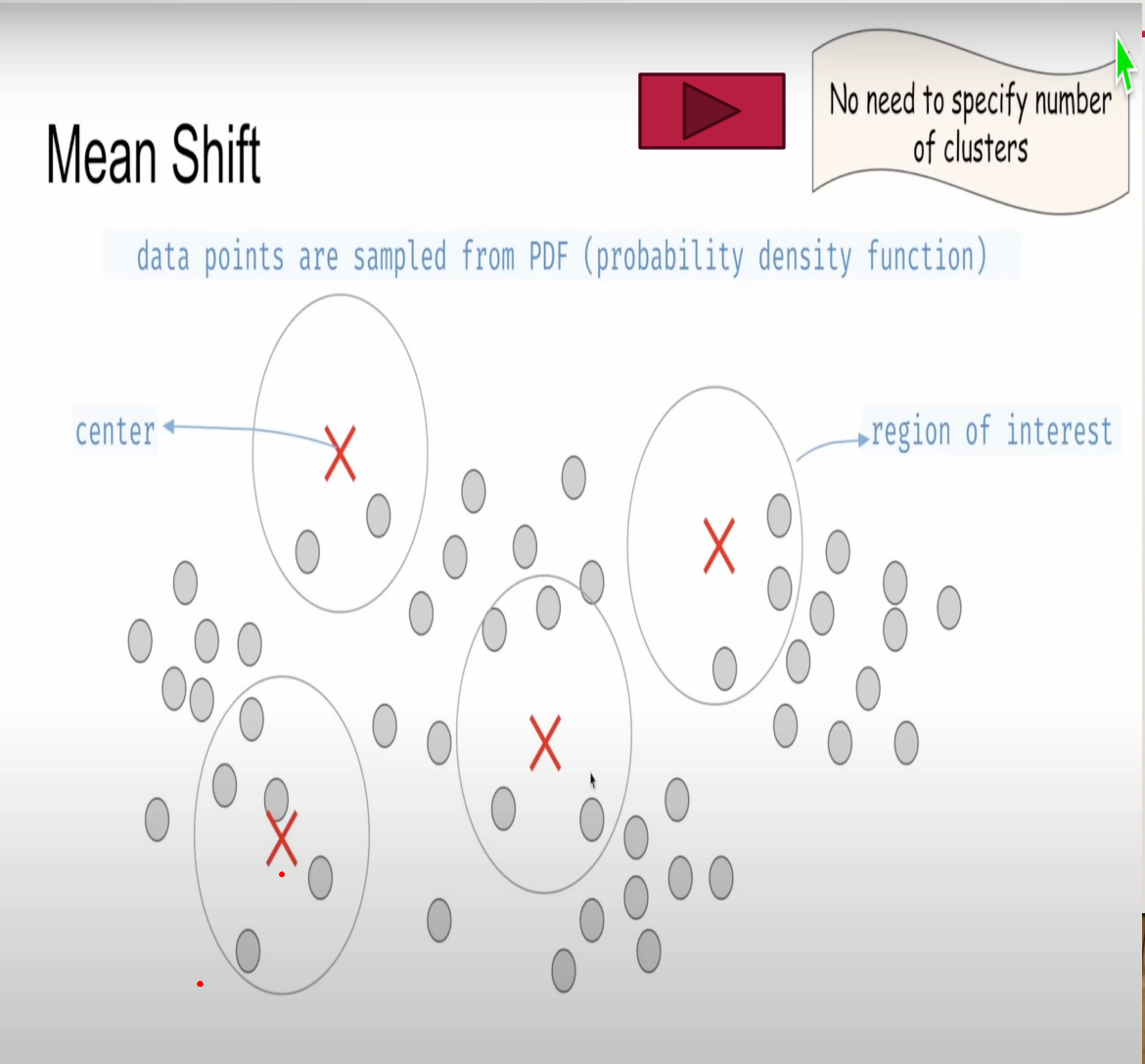
MEAN SHIFT

Mean shift clustering is probability density based clustering algorithm. Idea is to find attraction basin for each point and points which belong to same attraction basins are in same cluster.

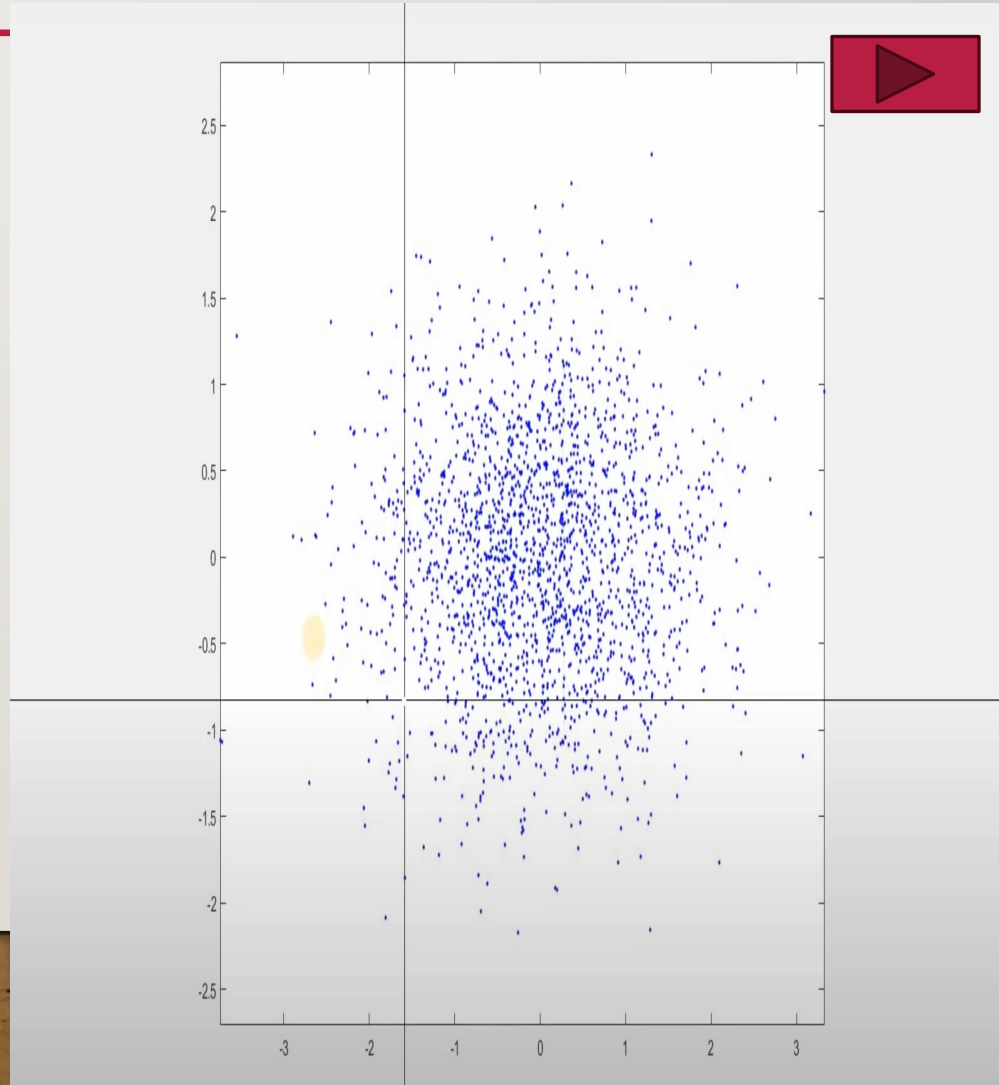




MEAN SHIFT



முத்து வசுமதி



MEAN SHIFT

Mean Shift

Equation to update candidate:

$$x_i^{t+1} = m(x_i^t)$$

iteration

candidate centroid

mean shift vector

$$m(x_i) = \frac{\sum_{x_j \in N(x_i)} K(x_j - x_i) x_j}{\sum_{x_j \in N(x_i)} K(x_j - x_i)}$$

kernel

MEAN SHIFT – PROS & CONS

Pros

•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.

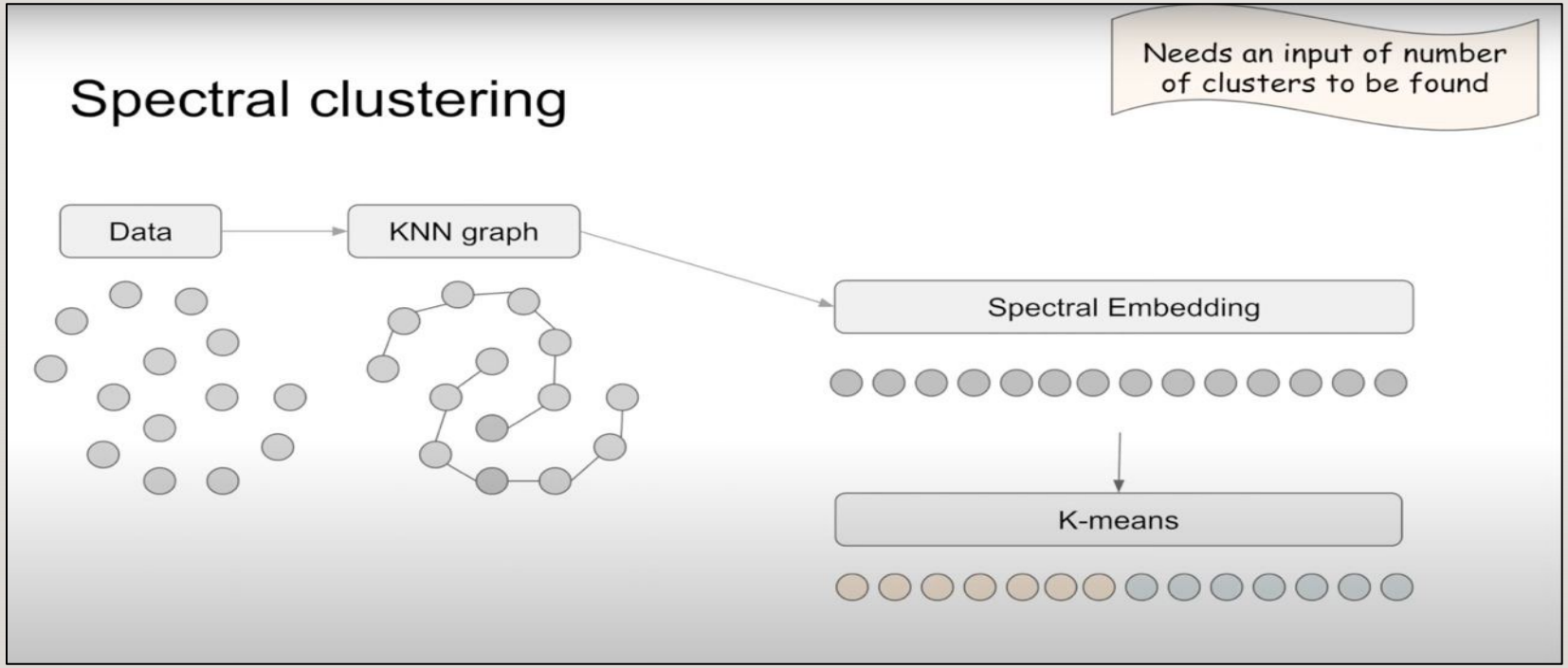
Cons

•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

"Spectral clustering" implies, the purpose of this algorithm is to perform clustering for given data as per spectrum (eigenvalues).

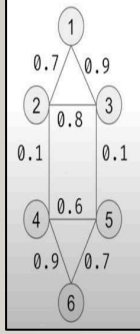
i.e - Spectral Clustering is a variant of the clustering algorithm that uses the connectivity between the data points to form the clustering. It uses eigenvalues and eigenvectors of the data matrix to forecast the data into lower dimensions space to cluster the data points.



SPECTRAL CLUSTERING

Spectral clustering — Step:1

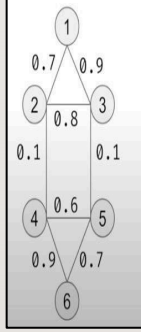
Data → Adjacency matrix A → Degree matrix D → Laplacian matrix L=D-A → Eigenvalues Eigenvectors → Partition




```
    1     2     3     4     5     6
1 [[0. , 0.7, 0.9, 0. , 0. , 0. ],
2  [0.7, 0. , 0.8, 0.1, 0. , 0. ],
3  [0.9, 0.8, 0. , 0. , 0.1, 0. ],
4  [0. , 0.1, 0. , 0. , 0.6, 0.9],
5  [0. , 0. , 0.1, 0.6, 0. , 0.7],
6  [0. , 0. , 0. , 0.9, 0.7, 0. ]]
```

Spectral clustering — Step:2

Data → Adjacency matrix A → Degree matrix D → Laplacian matrix L=D-A → Eigenvalues Eigenvectors → Partition

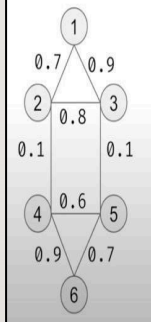



```
    1     2     3     4     5     6
1 [[1.6, 0. , 0. , 0. , 0. , 0. ],
2  [0. , 1.6, 0. , 0. , 0. , 0. ],
3  [0. , 0. , 1.8, 0. , 0. , 0. ],
4  [0. , 0. , 0. , 1.6, 0. , 0. ],
5  [0. , 0. , 0. , 0. , 1.4, 0. ],
6  [0. , 0. , 0. , 0. , 0. , 1.6]]
```

Spectral clustering — Step:3

Data → Adjacency matrix A → Degree matrix D → Laplacian matrix L=D-A → Eigenvalues Eigenvectors → Partition

- positive diagonal elements,
- zero or negative off-diagonal elements
- sum of rows equals zero




```
    1     2     3     4     5     6
1 [[ 1.6, -0.7, -0.9,  0. ,  0. ,  0. ],
2  [-0.7,  1.6, -0.8, -0.1,  0. ,  0. ],
3  [-0.9, -0.8,  1.8,  0. , -0.1,  0. ],
4  [ 0. , -0.1,  0. ,  1.6, -0.6, -0.9],
5  [ 0. ,  0. , -0.1, -0.6,  1.4, -0.7],
6  [ 0. ,  0. ,  0. , -0.9, -0.7,  1.6]]
```

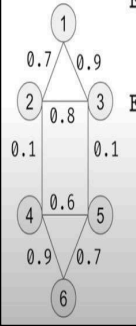
not connected

connected

self

Spectral clustering — Step:4

Data → Adjacency matrix A → Degree matrix D → Laplacian matrix L=D-A → Eigenvalues Eigenvectors → Partition

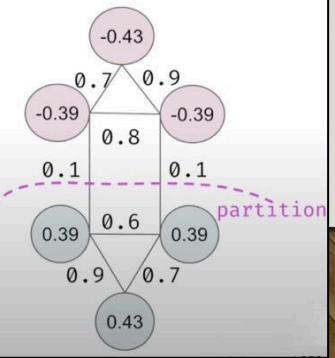



```
Eigenvalues:
 [0.    0.13  2.655 2.519 2.289 2.008]  Graph spectrum
    connectivity  spectral radius
    (2nd largest) (largest)
Eigenvectors:
 [[-0.408 -0.431 -0.492 -0.163 -0.614 -0.036]
 [-0.408 -0.395 -0.294  0.155  0.733  0.171]
 [-0.408 -0.398  0.806  0.046 -0.101 -0.116]
 [-0.408  0.398  0.104 -0.651  0.053  0.487]
 [-0.408  0.394 -0.102 -0.103  0.151 -0.796]
 [-0.408  0.431 -0.02   0.716 -0.223  0.291]]
```

Spectral clustering — Step:5

Data → Adjacency matrix A → Degree matrix D → Laplacian matrix L=D-A → Eigenvalues Eigenvectors → Partition

```
Eigenvalues:
 [0.    0.13  2.655 2.519 2.289 2.008]
          2nd eigenvector
Eigenvectors:  smaller eigenvalue → better partitioning
 [[-0.408 -0.431 -0.492 -0.163 -0.614 -0.036]
 [-0.408 -0.395 -0.294  0.155  0.733  0.171]
 [-0.408 -0.398  0.806  0.046 -0.101 -0.116]
 [-0.408  0.398  0.104 -0.651  0.053  0.487]
 [-0.408  0.394 -0.102 -0.103  0.151 -0.796]
 [-0.408  0.431 -0.02   0.716 -0.223  0.291]]
```



முத்து வசுமதி

SPECTRAL CLUSTERING- PROS & CONS

Pros

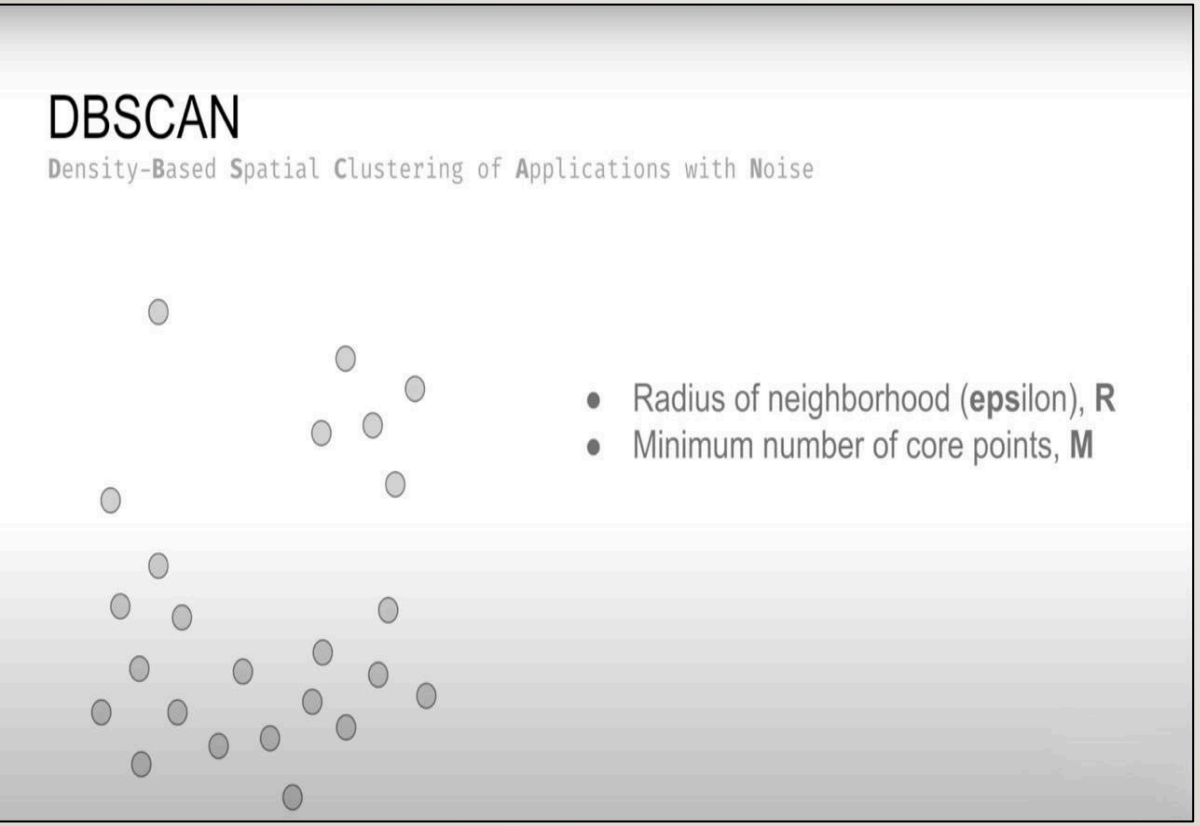
- Applicable for high dimensional datasets.
- No strong assumptions about cluster shape.
- Can sometimes handle categorical variables.

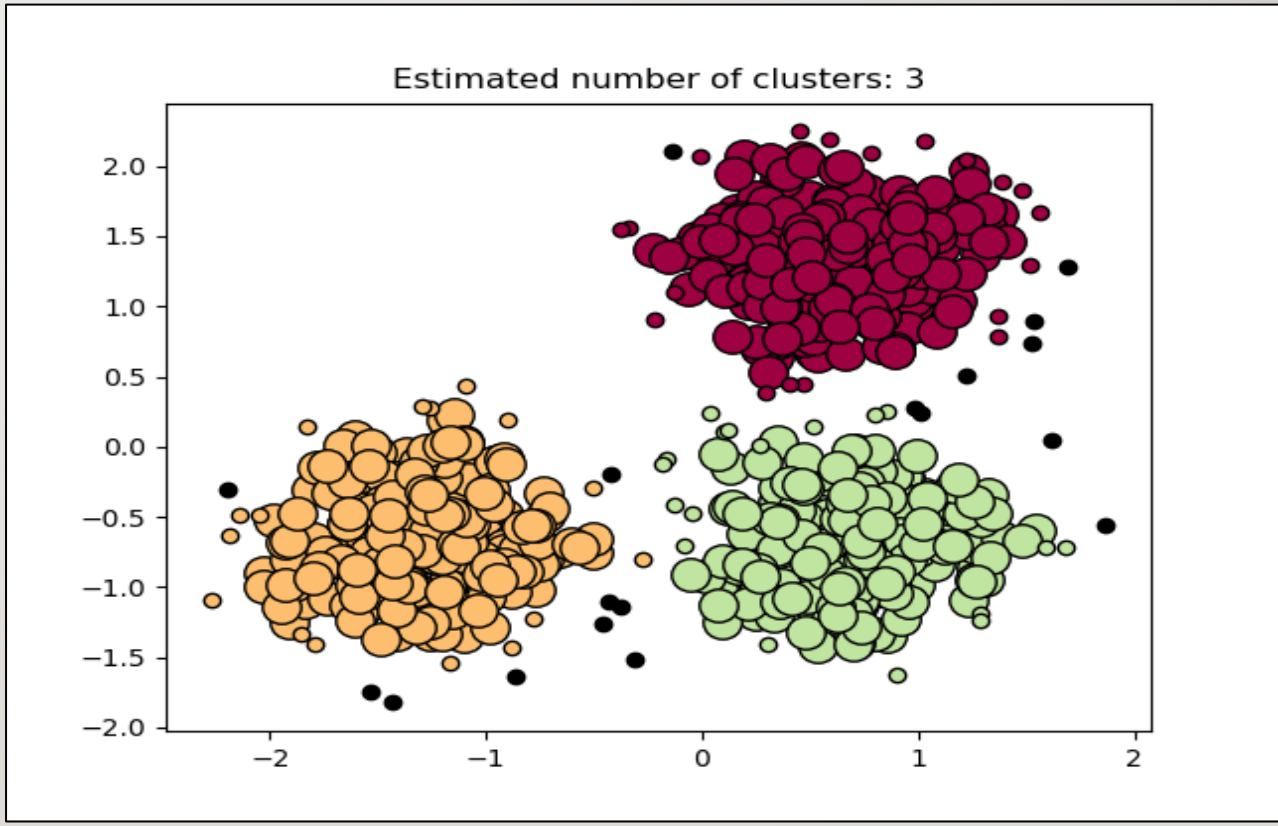
Cons

- Relatively slow.
- Less common.
- Not intuitive or easy to explain.
- Sensitive to seed.
- Need to select the number of clusters.

DBSCAN

Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”. The key idea is that for each point of a cluster, the neighbourhood of a given radius has to contain at least a minimum number of points.



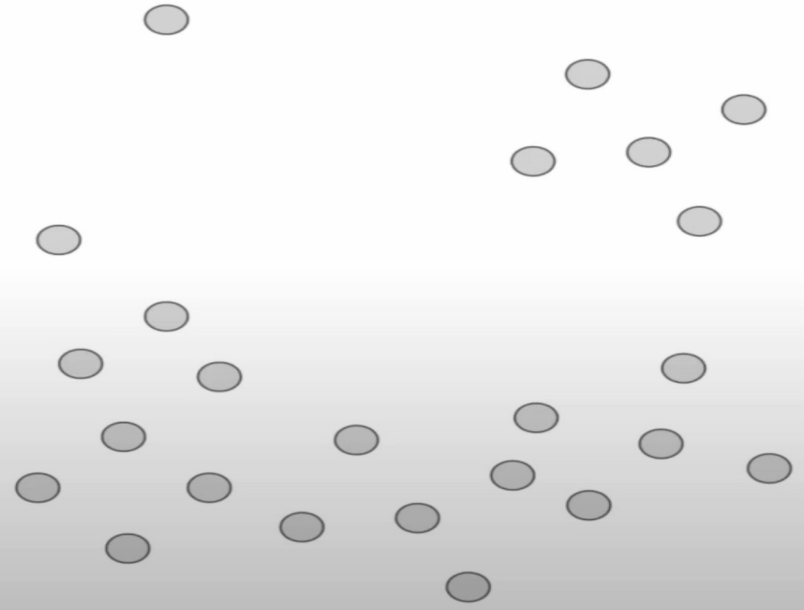


முத்து வசுமதி

DBSCAN

DBSCAN

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise



- Radius of neighborhood (**eps**ilon), **R**
- Minimum number of core points, **M**

முத்து வசுமதி

DBSCAN – PROS & CONS

Pros

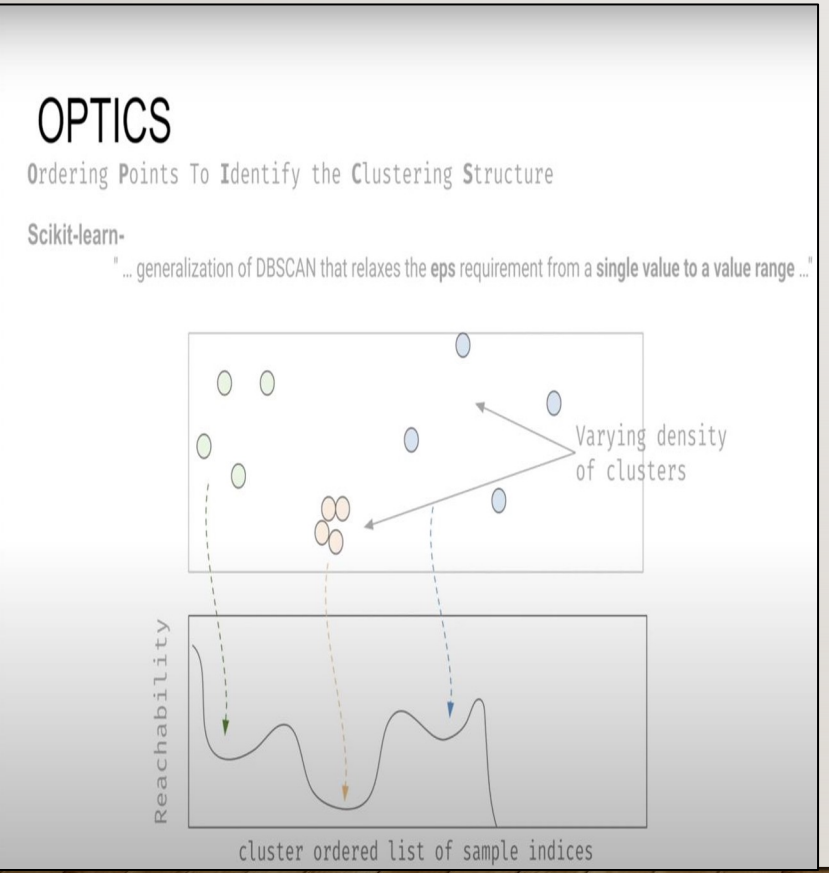
- Capable of identifying clusters of any shape.
- Robust to noise.
- Parameter-free clustering algorithm.
- DBSCAN is efficient on large datasets.

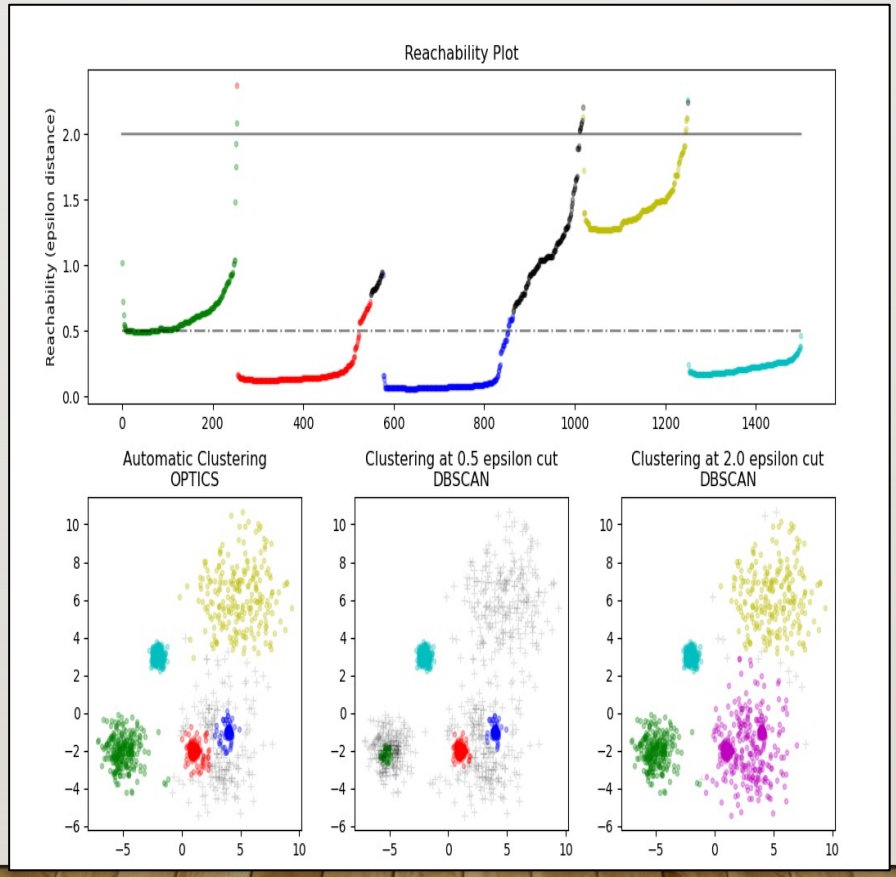
Cons

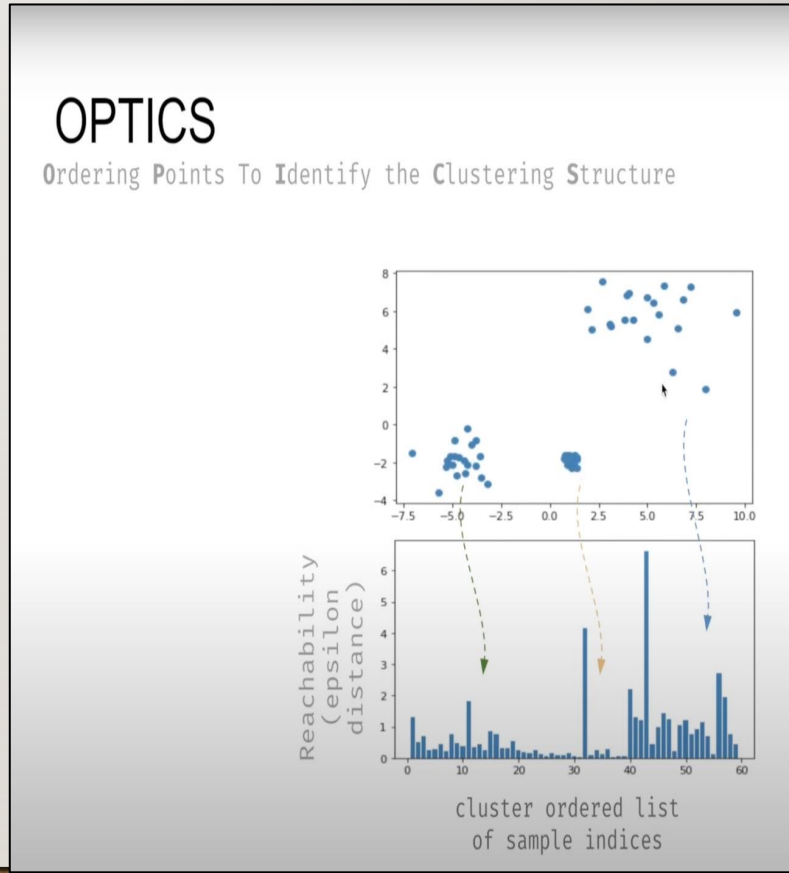
- Performance can be sensitive to the choice of hyperparameters.
- Poor performance on datasets with clusters of significantly different densities.
- Performance can be limited on high-dimensional data due to the "curse of dimensionality”.

OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure) is a density-based clustering algorithm, similar to DBSCAN (Density-Based Spatial Clustering of Applications with Noise), but it can extract clusters of varying densities and shapes. It is useful for identifying clusters of different densities in large, high-dimensional datasets.







முத்து வசுமதி

OPTICS- PROS & CONS

Pros

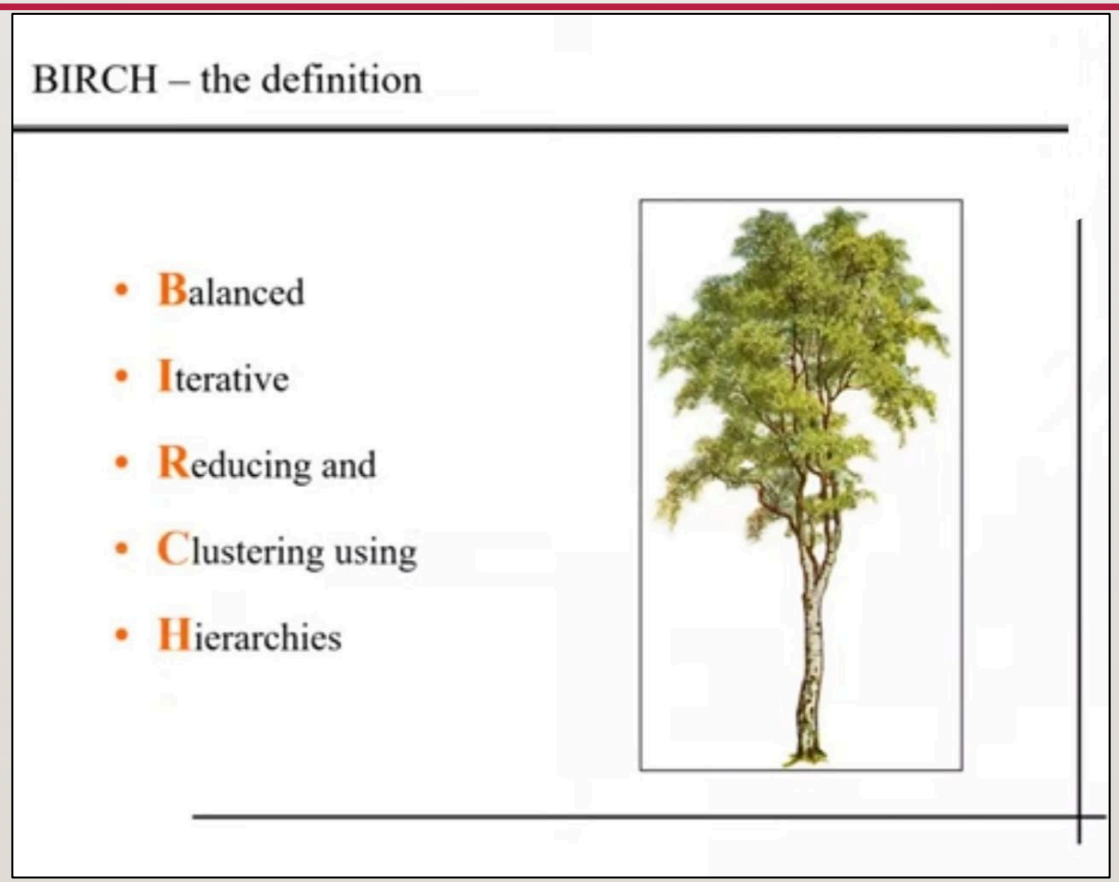
- OPTICS clustering doesn't require a predefined number of clusters in advance.
- Clusters can be of any shape, including non-spherical ones.
- Able to identify outliers(noise data).

Cons

- 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

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) is an algorithm used for clustering large datasets in machine learning and data mining. It was proposed by Tian Zhang, Tian Zhang, Raghu Ramakrishnan, and Miron Livny in 1996.

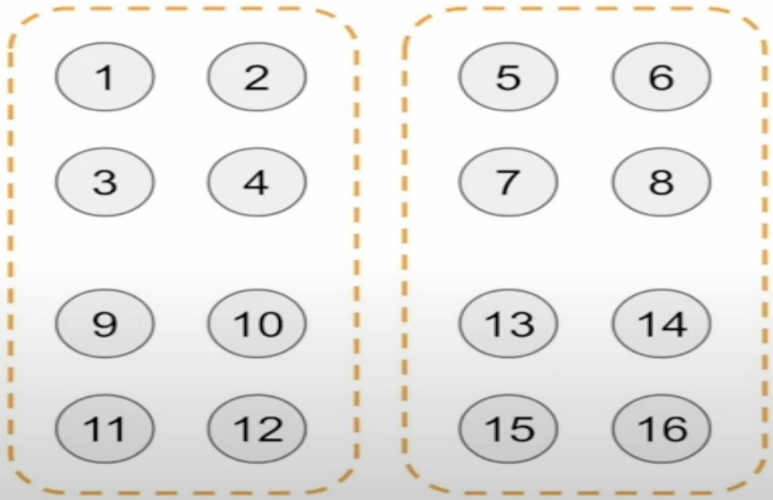


BIRCH-
HIERARCHICAL CLUSTERING

Definition :It is a type of clustering algorithm to group similar data points into clusters based on their pairwise similarities or dissimilarities.

Hierarchical clustering

How many clusters are there?



2

BIRCH

BIRCH concepts and terminology

BIRCH introduces two concepts :

- ✓ Clustering feature (CF)
- ✓ Clustering feature tree (CF tree)

These structures

- ➢ used to summarize cluster representations.
- ➢ helps the clustering method to achieve good speed & scalability in large databases.

BIRCH - CF

BIRCH concepts and terminology

Clustering Feature

Given N d-dimensional data points in a cluster,
X_i (i = 1, 2, 3, … , N)

CF vector of the cluster is defined as a triplet CF = (N,LS,SS):

- N - number of data points in the cluster
- LS - linear sum of the N data points

$$\sum_{i=1}^{N} X_i$$

- SS - square sum of the N data points

$$\sum_{i=1}^{N} X_i^2$$

Properties of Clustering Feature

- CF entry is a summary of statistics of the cluster
- A representation of the cluster
- A CF entry has sufficient information to calculate the centroid, radius, diameter and many other distance measures
- Additively theorem allows us to merge sub-clusters incrementally

Distance Measures

- Given a cluster with data points
  - → Centroid:

$$x_0 = \frac{\sum_{i=1}^{n} x_i}{n} = \frac{LS}{n},$$

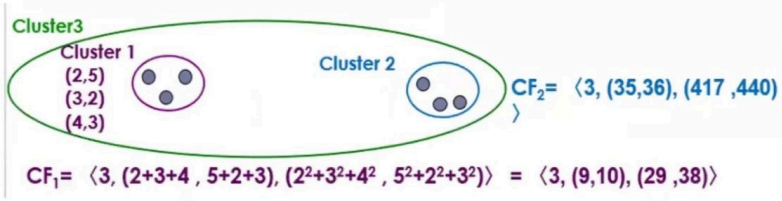
  - → Radius: average distance from any point of the cluster to its centroid

$$R = \sqrt{\frac{\sum_{i=1}^{n} (x_i - x_0)^2}{n}} = \sqrt{\frac{nSS - 2LS^2 + nLS}{n^2}},$$

  - → Diameter: average pairwise distance within a cluster.

$$D = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} (x_i - x_j)^2}{n(n-1)}} = \sqrt{\frac{2nSS - 2LS^2}{n(n-1)}}.$$

Clustering Feature



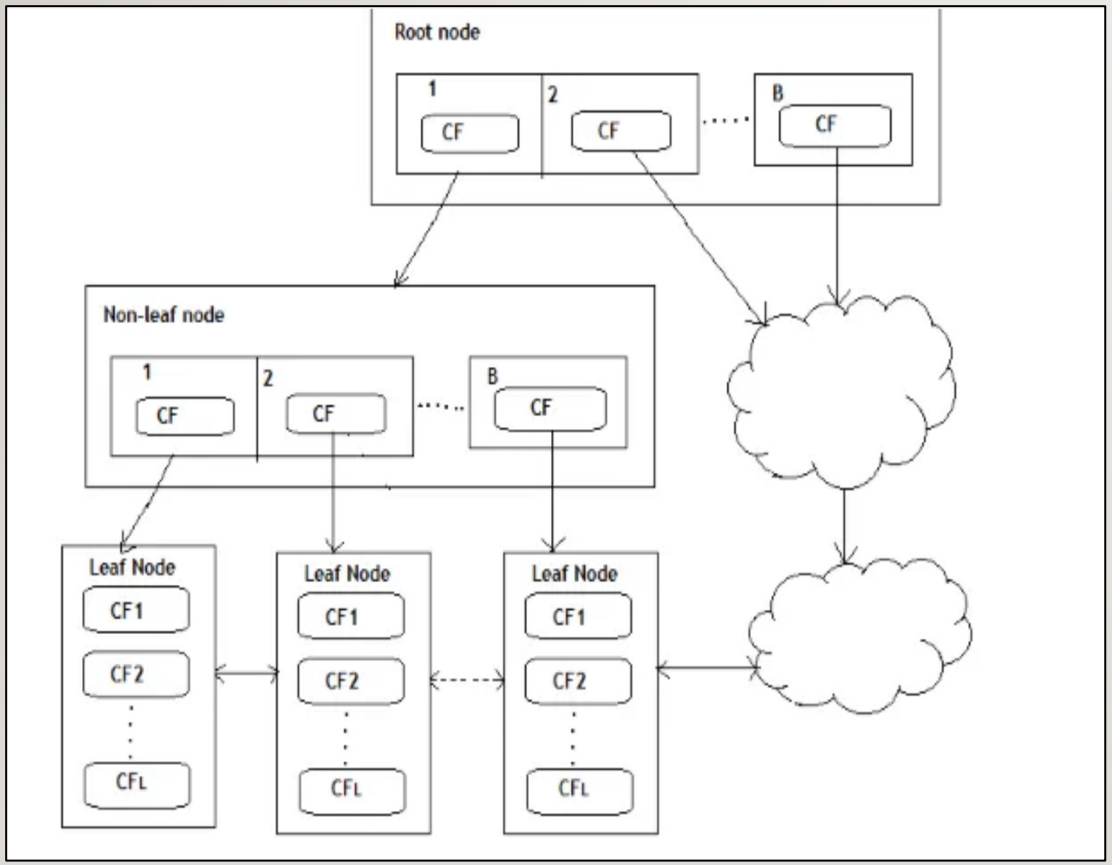
CF_2= ⟨3, (35,36), (417 ,440)⟩

CF_1= ⟨3, (2+3+4 , 5+2+3), ($2^2+3^2+4^2$, $5^2+2^2+3^2$)⟩ = ⟨3, (9,10), (29 ,38)⟩

CF3=CF1+CF2= ⟨3+3, (9+35, 10+36), (29+417 , 38+440)⟩ = ⟨6, (44,46), (446 ,478)⟩

முத்து வசுமதி

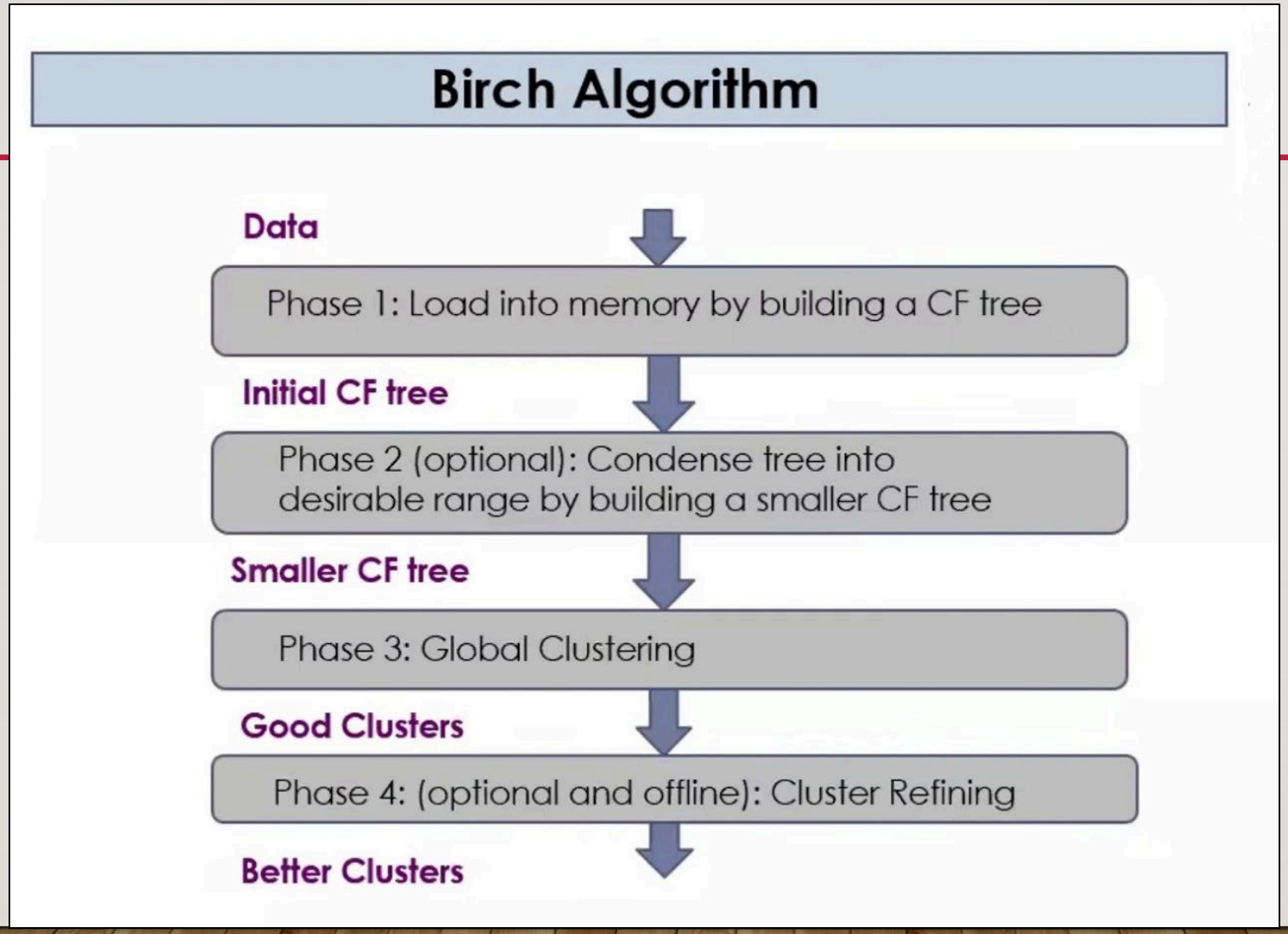
BIRCH-CLUSTERING FEATURE TREE (CF TREE)



A CF Tree structure is given as below:

- A CF tree is a height-balanced tree that stores the clustering features for a hierarchical clustering.
- By definition, a nonleaf node in a tree has descendants or "children."
- The nonleaf nodes store sums of the CFs of their children, and thus summarize clustering information about their children.
- A CF tree has two parameters: branching factor, B, and threshold, T.
- The branching factor specifies the maximum number of children per non leaf node.
- The threshold parameter specifies the maximum diameter of subclusters stored at the leaf nodes of the tree.
- These two parameters influence the size of the resulting tree.

BIRCH

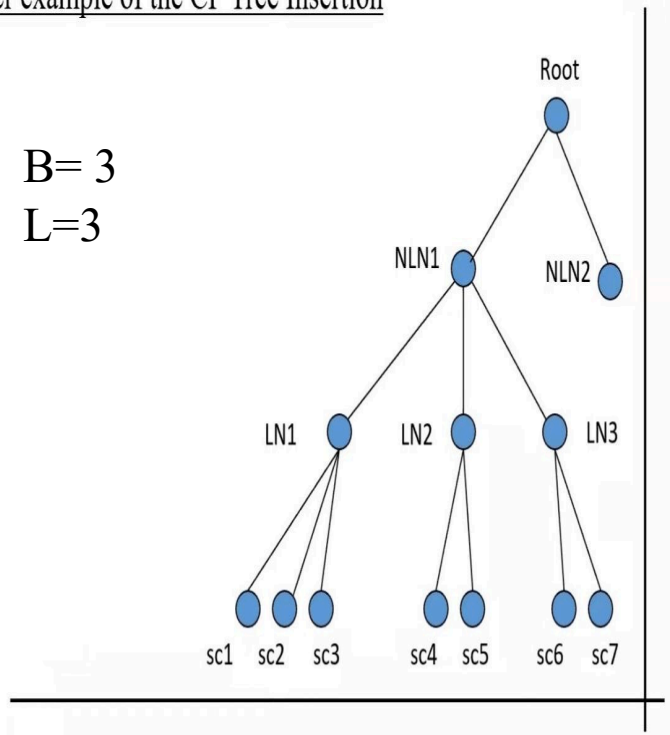

Birch Algorithm
Data
Phase 1: Load into memory by building a CF tree
Initial CF tree
Phase 2 (optional): Condense tree into desirable range by building a smaller CF tree
Smaller CF tree
Phase 3: Global Clustering
Good Clusters
Phase 4: (optional and offline): Cluster Refining
Better Clusters

BIRCH- ALGORITHM

BIRCH algorithm

- Another example of the CF Tree Insertion

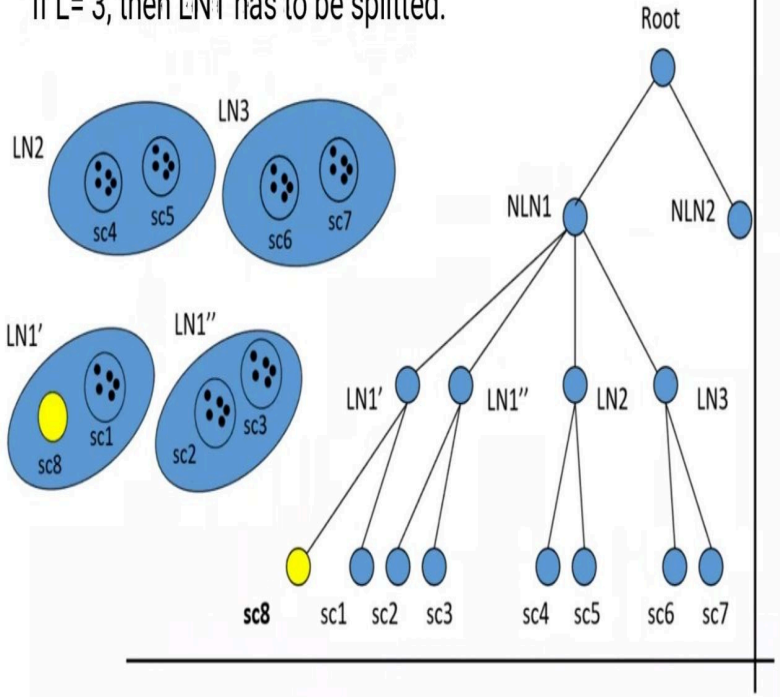
B= 3
L=3



BIRCH algorithm

- Another example of the CF Tree

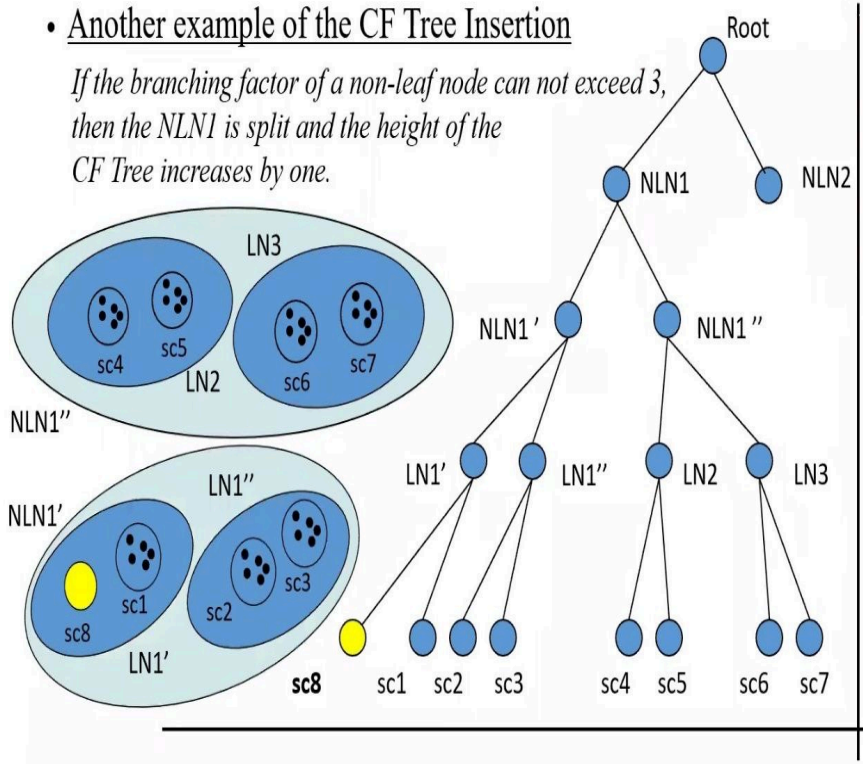
If L= 3, then LN1 has to be spiltted.



BIRCH algorithm

- Another example of the CF Tree Insertion

If the branching factor of a non-leaf node can not exceed 3, then the NLN1 is split and the height of the CF Tree increases by one.



BIRCH – PROS & CONS

Pros

- Birch performs faster than existing algorithms (CLARANS and KMEANS) on large datasets.
- Scans whole data only once.
- Handles outliers better.
- Superior to other algorithms in stability and scalability.

Cons

- Since each node in a CF tree can hold only a limited number of entries due to the size, natural clusters cannot be formed.
- Moreover, if the clusters are not spherical in shape, it doesn't perform well because it uses the notion of radius or diameter to control the boundary of a cluster.

நன்றி

முத்து வசுமதி