Clustering

Data Science with Python
CS677

Farshid Alizadeh-Shabdiz, PhD, MBA
Alizadeh@bu.edu
Fall 2021

K-means Initialization

Problem: K-means is sensitive to initial points, and initial points can result to a sub-optimum solution.

Example:

K-Mean Initialization Solutions

- Random selection of initial points. In this case run the algorithm multiple times and select the best answer

- Question – What is the best solution?

It is the model with the lowest Mean Squared distance between data points and their centroids.

This is called *“intertia”*

- A kmeans object has ”inertia_”

  *Kmeans_object.inertia_*

Initialization Solution – K-means++

- K-means++ is a suggestion to smart initialization of K-means.

The idea is as follows

If a center point is C_i and distance between a point and its centroid is $D(x_i, C_i)$

1. Choose the first random point
2. Choose point x_i as the next centroid with probability
$$D(x_i, C_i) \Big/ \sum_{j=1}^{M} D(x_j, C_j)$$
3. Go to step 2 until K centroids have been selected

K-means++

- Adds extra computation
- But it reduces number of iterations dramatically
- Most of the time find optimum or very close to optimum solution

Note that K-means function in python uses this method by default.

Note “init” parameter in kmeans object instantiation chooses initialization method.

How to Set K?

- Looking at the "Inertia" for a range of K values and decide the elbow point
- Silhouette score,
  - Silhouette coefficient is $(b-a)/\max(b,a)$
    - a is the mean distance to the other instances in the same cluster
    - b is the mean distance to the nearest cluster instances
  - The mean silhouette coefficient over all the instances.
- Silhouette score varies between -1 and 1
  - +1 means good separation between clusters
  - 0 means cluster boundaries are fuzzy
  - -1 means clusters are merged to the same space

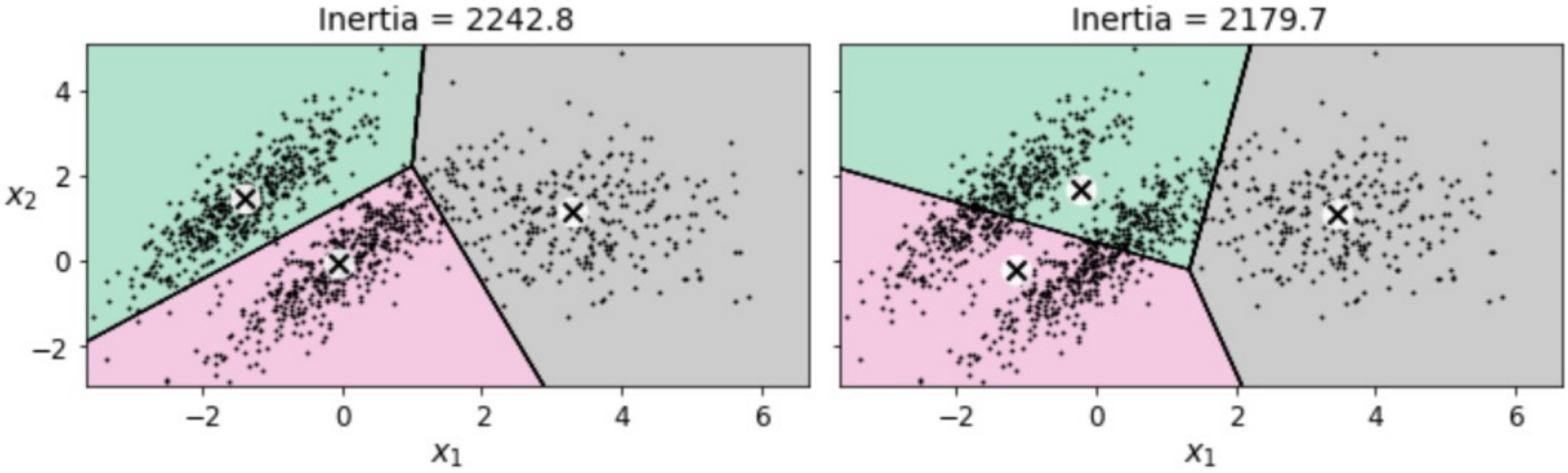
- Scikit learn function

  Silhouette_score(data, kmeans_obj.labels_)

K-means

- Fast
- Scalable
- But
  - Sometimes provide sub-optimum solutions
  - Need K!
  - K-means doesn't work well with clusters with different size and density
  - K-means doesn't behave well in non-spherical shapes
  - Scaling of variables of observations is suggested
  - Distance function is important

K-means example



Reference: Hands-on Machine learning with Sci_kit learn Keras & Tensorflow by A. Geron

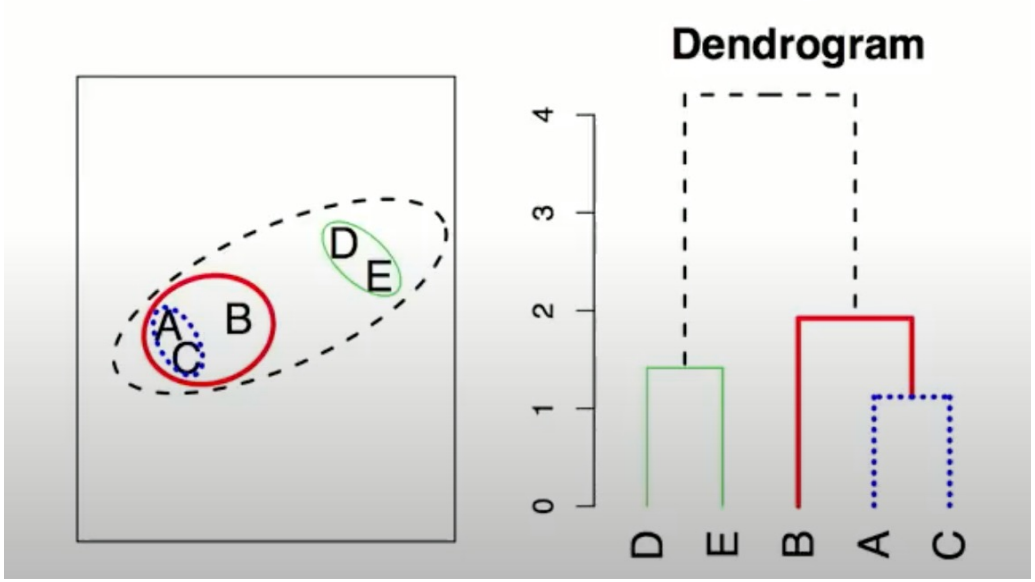
Hierarchical Clustering

Agglomerative and BIRCH

Hierarchical Clustering

- Attractive approach since no need to decide on K in advance.
- Bottom-up or agglomerative Hierarchical clustering

Example



Reference: Tibshirani and Hastie – Intro to statistical learning

BIRCH – Balanced Iterative Reducing and Clustering Using Hierarchies

BIRCH Main Purpose

- Create a tree in the memory, when the entire data set is big and cannot fit in memory
- The tree contains all the information that we need. Therefore, we don't need to go back to data again and again
- It is an incremental construct of the tree
- It scales linearly

BIRCH

- Incremental construct of CF (clustering features) tree, holding information for the rough hierarchical clustering & fine clustering later

  phase 1: Scan the data one by one and build CF tree in-memory. Rough clustering.

  Phase 2: Use any clustering algorithm to cluster the leaf nodes of the tree, merge clusters or remove outliers

Let us start with BIRCH Cluster Features(CF)

- Clustering features – a summary of data to build the CF-tree, and also the final clustering
  1. N: number of points
  2. LS: Linear sum of N points: $\sum_{i=1}^{N} x_i$
  3. SS: Sum of square of N points: $\sum_{i=1}^{N} \overrightarrow{x_i}^2$

Note: CFs are 0^{th}, 1^{st}, and 2nd moments of a cluster

BIRCH – Measures of a Cluster

- Centroid: middle of a cluster

$$C_k = \frac{\sum_{i=1}^{N} x_i}{N}$$

- Radius: average distance of members to centriod

$$\sqrt{\frac{\sum_{i=1}^{N}(x_i - C_k)^2}{N}}$$

- Diameter: Average pairwise distance

$$\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i - x_j)^2}{N(N-1)}}$$

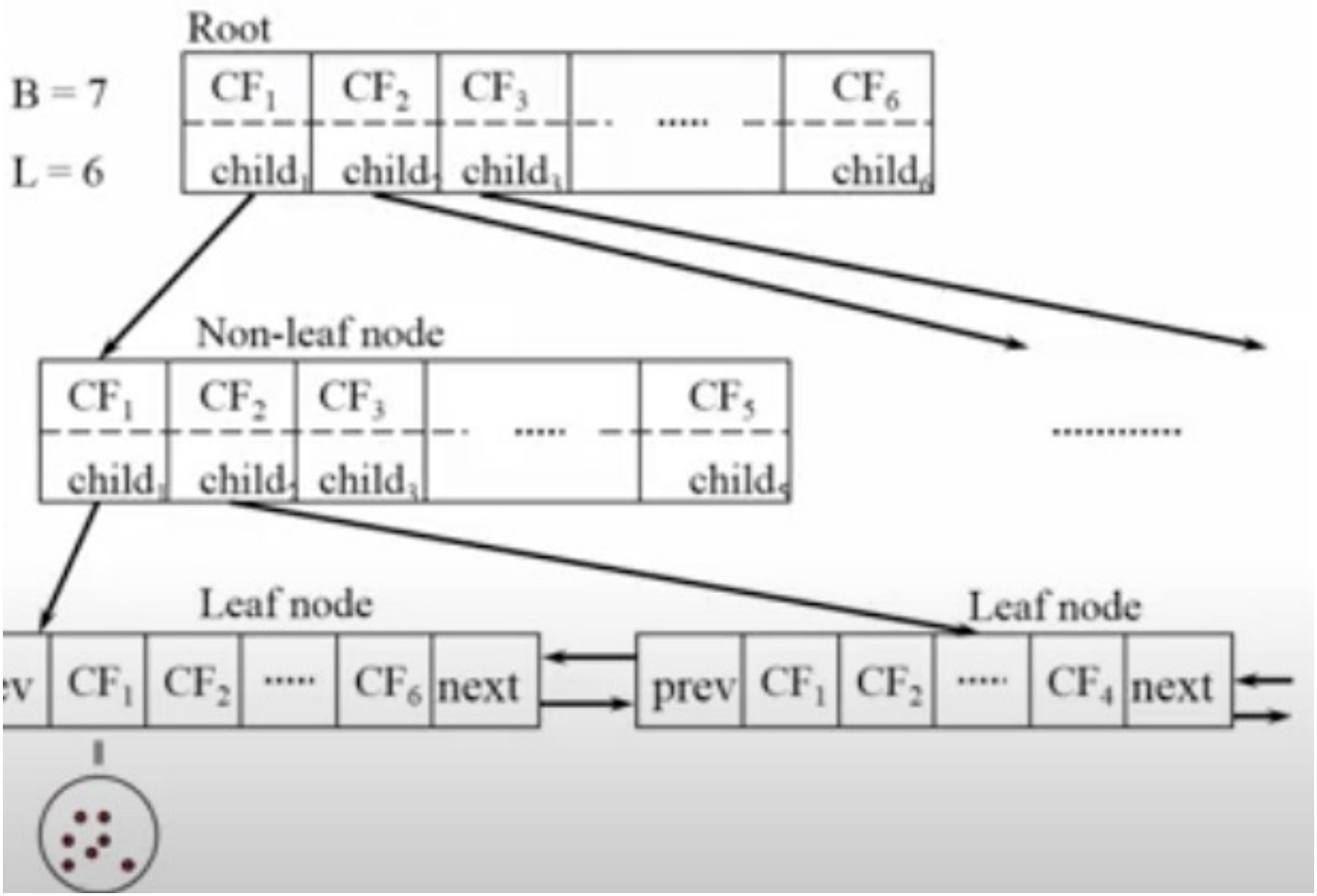
Clustering Computations Based on CF

- Centroid: C = LS/N
- Radius R =

$$\sqrt{\frac{\sum_{i=1}^{N}(x_i - C_k)^2}{N}} = \sqrt{\frac{N.C^2 + SS - 2C.LS}{N}} = \sqrt{\frac{SS}{N} + \left(\frac{LS}{N}\right)^2}$$

- Diameter D = $\sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i - x_j)^2}{N(N-1)}} = \sqrt{\frac{2N.SS - 2LS^2}{N(N-1)}}$
- CF merge = CF1 + CF2 = (N1+N2, LS1 + LS2, SS1+SS2)

Note LS is a vector and SS is a scalar



Ref: Prof Han course in University of Illinois Urbana Champaign

BIRCH Hyperparameters

How To build a CF tree

Note: CFs are 0^{th}, 1^{st}, and 2nd moments of a cluster

- BRICH Important parameters
  - T: Maximum diameter of a leaf node
  - B: max number of branches
  - L: Max length of leaf node
- CF (non leaf node) = sum of CF of nodes below

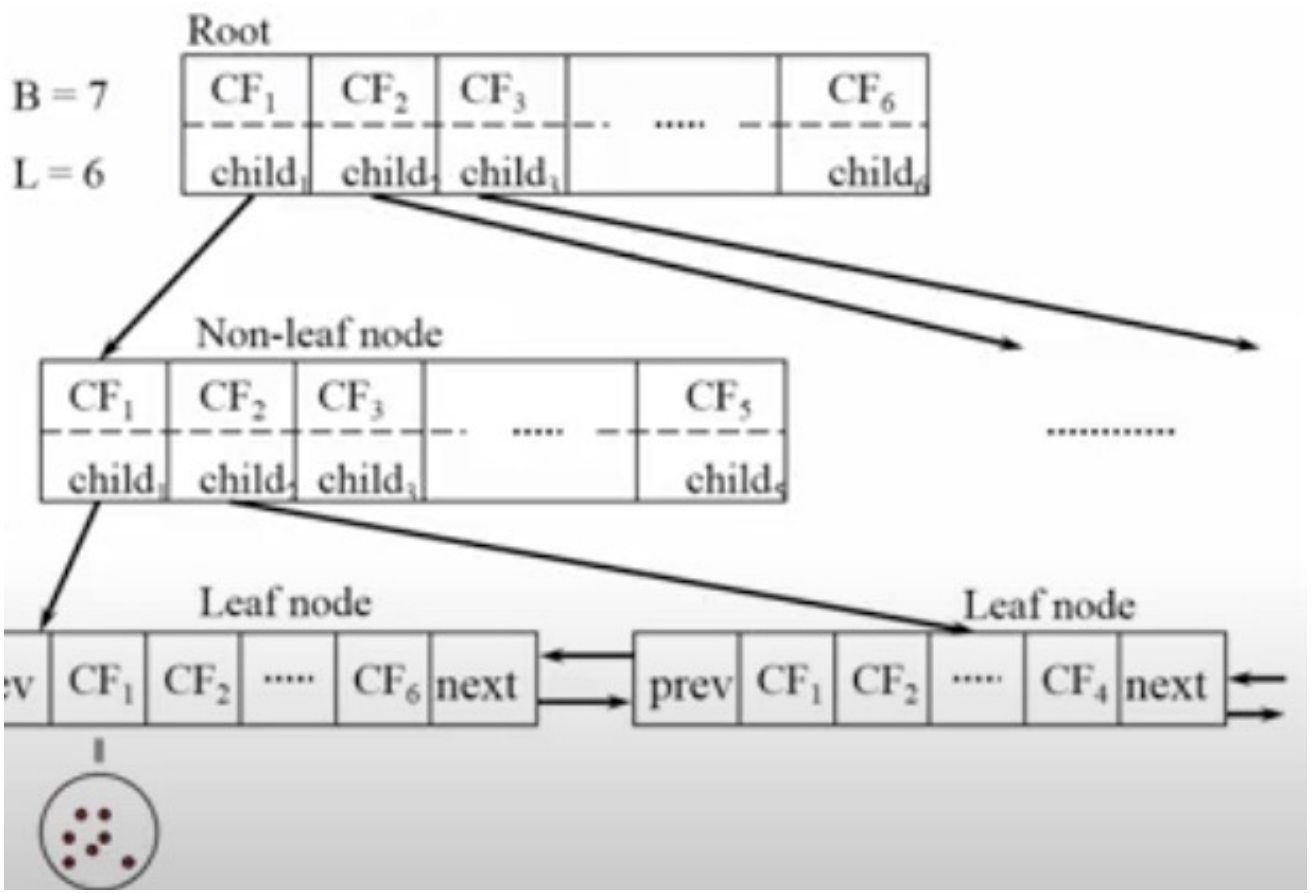
How to Build a Cluster

Assume T=L=2, and B=3.
Let us use Manhattan dis.

Data points

(3,4), (2,6), (4,5), (4,7), (8,4)

1. (3,4) -> CF1(1,(3,4),25)
2. (2,6)-> (D>T)=> CF2(1,(2,6),40)
3. (4,5)->(D<T)=>merge with into CF1 => CF1(2,(7,9),66)
4. (4,7)->(D>T)->CF3(1,(4,7),55)
5. (8,4)->(D>T) & (B>3)=> two non-leaf nodes. One CF2&CF3 and one CF1&CF4(1,(8,4),80)



BIRCH

- Add new points to the tree incrementally, by adding the point to the closest leaf
- Update CF values
- Check two criterias
  1. If (New Diameter > max_diameter) split the leaf and possibly parents
  2. If (# of leafs > max_children) split the parent

BIRCH Concerns

Pros:

- Efficient for clustering large data which doesn't fit in memory
- Different clustering algorithms can be used on the "tight" clusters on the leaf nodes
- Outlier removal which can be any algorithm or use leaf size as a criteria
- It grows linearly

Cons:

- Need numerical data
- Sensitive to insertion order of data points
  - Duplicates might end up in different clusters
  - Solution:
    - Randomized reading
    - Reading the data again with knowing centroids and make assignments based on distance to centroids
- Ceiling on number of leaf nodes is an artificial restriction
- Clusters are expected to be spherical because of D criteria

Probabilistic Hierarchical Clustering

Probabilistic Hierarchical Clustering

- Non-probabilistic hierarchical clustering /algorithmic clustering
  - Good distance measure is critical and hard to choose
  - Cannot handle missing attributes
  - Optimization goal is heuristic
  - Relies on local search
- Probabilistic approach
  - Generative model
  - Easy to generalize

Note: any model can be used as a base of the generative models

Gaussian Generative Model

- Probability of a point Xi generated by a Gaussian distribution $N(\mu, \sigma)$

$$P(x_i|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

Then for probability of a set of X is generated by the Gaussian model is product of probilities

Goal: Finding Gaussian distributions, which maximizes probability

Probabilistic Clustering Algorithm

Goal: Finding Gaussian distributions, which the best explains the existing data set

- Criteria is maximizing aggregate probabilities that the data set is generate by a Gaussian distribution

$$Max\ [P(X|\mu,\sigma)]$$

- And

$$P(X|\mu,\sigma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Therefore, if merge/split clusters if the result increase the probability of getting generated by a Gaussian model.

OPTICS

Optics: Ordering Points to Identify Clustering Structure

- Proposed by the same inventors of DBSCAN in 1999
- DBSCAN:
  - Sensitive to parameter selection (eps, minPoints)
  - One global eps doesn't work
- It is based on the following observation:

  high density clusters are contained by low density clusters
- Idea is processing higher density clusters first

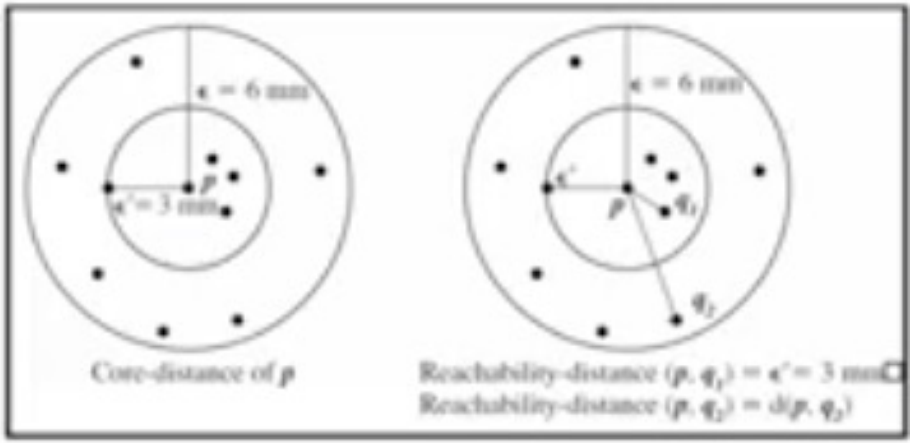
OPTICS

- Optics introduces two parameter
  - Core distance: is the smallest eps distance from a core point p to cover at least minPoints

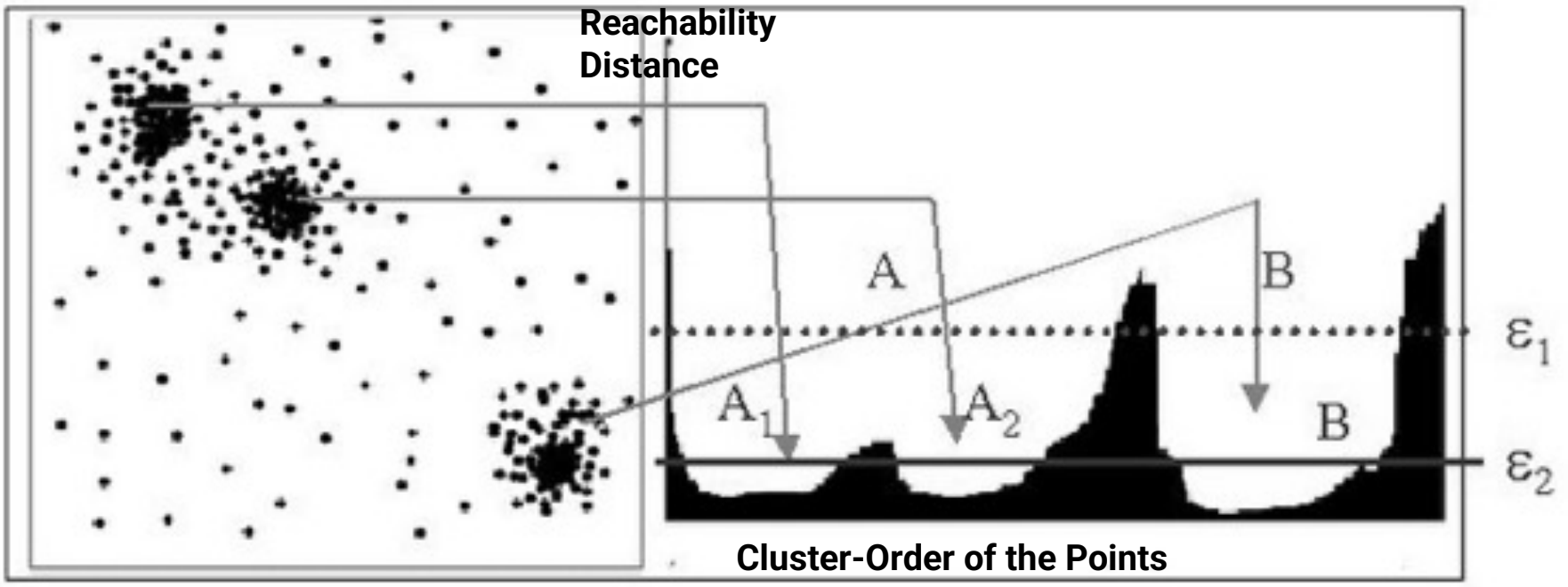
$$core\ dis(p) = \begin{cases} Min(eps)\ cover\ minPoints \\ Not\ defined\ if\ p\ isn't\ core \end{cases}$$

  - Reachability distance:

$$Max[core\ dis(q), dis(p,q)]$$



Ref: Prof Han course in University of Illinois Urbana Champaign



Ref: Effective Similarity Search on Voxelized CAD Object.by: P. Kroger, and Et. Al.

- The smaller reachability, the denser the cluster

OPTICS Algorithm

1. Set eps (which can be set to a high number)
2. Start with a random point, *p (*Set reachability of *p* to undefined).
3. Choose all neighbors within eps distance of *p.* If *p* is not a core object select next random point
4. If *p* is a core point, order points in ascending reachability order
5. Go to step 3 until all the data points are covered

- Initial eps and MinPoints identifies core points
- Core distance helps to choose high density points
- Ascending order helps to have a flat valley
- Reachability distance helps to have different eps

OPTICS

- Allows a range of eps to get explored
- Orders points to identify clusters
- Trying to produces an order of points based on density
  - In way - It is a hierarchical clustering
- It can be used for interactive and automatic analysis of clusters
- It can use clusters with different size, shapes, and density
- Can be used to detect outliers
- It does have a good visualization by-product
- Complexity O(N *log*N)