Data Stream Mining- Lecture 3

Clustering of Data Streams

Chandresh Kumar Maurya, Research Assistant Professor, Peter Kiss Teaching Assistant

Eötvös Loránd University, Budapest, Hungary

October 22, 2020

What is clustering?

Clustering is an unsupervised technique to group (clusters henceforth) the data such that the examples in the group satisfy two requirements:

- Examples in the same cluster are similar
- Examples in the different cluster are dissimilar

Many methods (not all!) build on the assumption :

- that instances are points in some space
- we can define some *distance function* d,
- "more similar" examples are those that are closer according to d

Clustering

centroid based methods: a set of centroid points in space defines the clustering
for some k number of clusters ,P set of instances
A clustering algorithm computes C set of centroids/cluster centers that minimizes an objective cost function

$$\text{cost}(C, P) = \frac{1}{|P|} \sum_{x \in P} d(x, C) \tag{1}$$

$$\text{for } d(x, C) = \min_{c \in C} d(x, c). \tag{2}$$

(minimize the sum of distances o points x to the closest centroid point c)

Clustering of data streams

Clustering of streaming data is non-trivial due to the following challenges:

1. Concept drifts
2. High data rate
3. Low memory footprints
4. Low processing time
5. Single (or 2-3 passes)
6. Algorithms needs to be robust
7. and so on.

Traditional algorithms needs to be adapted to account for the aforementioned issues.

Types of clustering

Should we cluster examples or features?

- Partition based clustering (e.g. K-means, k-medoids)
- Hierarchical clustering (e.g. Agglomerative vs Divisive)
- Density-based clustering (e.g. DBSCAN, OPTICS)
- (Grid-based clustering (e.g. CLIQUE and STING))
- (Modal-based clustering(e.g. COWEB, SOM))

Basic concepts

Requirements of data stream clustering:-

1. Need a compact representation of the data stream (Why?).
2. Fast and incremental processing of incoming examples.
3. Tracking cluster changes.
4. Clear and fast identification of outliers.

Cluster feature

Definition

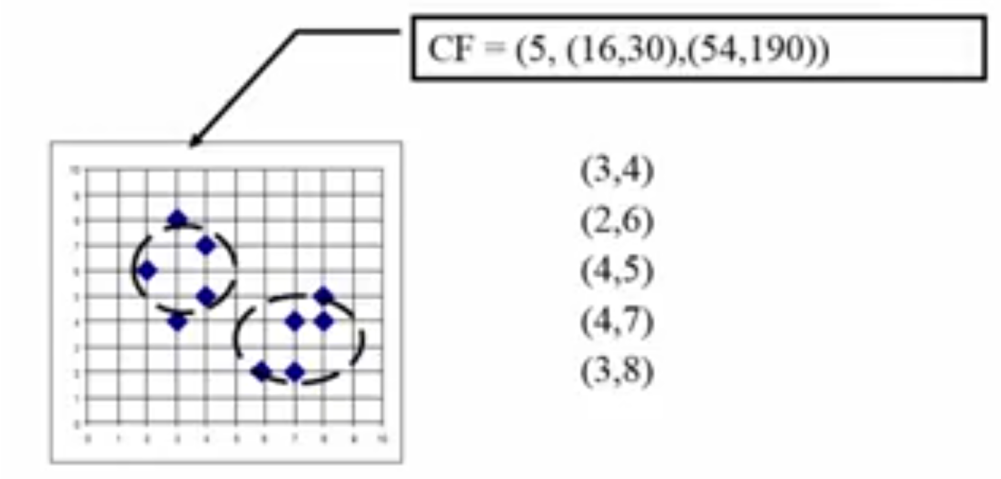
Cluster feature: A cluster feature is a triple (N, LS, SS) used to store the sufficient statistics of the data points.

where

- **N**–number of data points.
- **LS**–a vector to store linear sum of N points.
- **SS**–a sector to store sum of squares of the points.

CF easy to maintain:

- Incrementality
- Additivity
- Centroid
- Radius



Statistics of a CF

Centroid : middle point- the average of the points $c = \frac{\sum_{i=1}^{n} x_i}{n}$

Radius: average distance from the centroid $R = \sqrt{\frac{\sum_{i=1}^{n}(x_i - c)}{n}}$

Diameter: average pairwise distance of the members

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

The Leader Algorithm

- Input: user specified distance threshold - maximum distance of example and centroid $d(x, c)$
- At any step assigns the current example to the most similar cluster(if it is allowed)
- otherwise the example is added itself as a leader

Thus:

- no need for prior information on the number of clusters
- unstable - depends too much on the order of examples, and on the correct guess of the threshold

The Leader Algorithm

Algorithm 1: The Leader

Input : X: A sequence of Examples x_i
δ: Control distance parameter
Output : Centroid of k clusters
Initialization: Initialize the set of centroids $C = x_1$

```
for x_i ∈ X do
    Find the cluster c_r whose center is closest to x_i;
    if d(x_i, c_r) < δ then
        C = C ∪ x_i
    end
    else
        make a new leader with centroid x_i ;
    end
end
```

Single Pass $k-$Means Algorithm

1. the stream is processed in blocks
2. we use a buffer, where points of the dataset are kept in a compressed way
3. at the beginning of a round all the available space on the buffer is filled with points
4. based on the buffer find k centers such that the sum of distances of the points from their centroid is minimal
5. store the k centroid as CFs
6. in the next round fill again and the clusters will be initialized with the CFs found in previous round.

Single Pass $k-$Means Algorithm

Algorithm 11: Algorithm for Single Pass `k-Means` Clustering.

input : S: A Sequence of Examples
k: Number of desired Clusters.
output: Centroids of the k Clusters
begin
  Randomly initialize cluster means.;
  Each cluster has a discard set that keeps track of the sufficient statistics.;
  while *TRUE* **do**
    Fill the buffer with examples ;
    Execute iterations of k-means on points and discard set in the buffer, until convergence. ;
    `/* For this clustering, each discard set is treated like a regular point weighted with the number of points in the discard set. */`
    ;
    foreach *group* **do**
      update sufficient statistics of the discard set with the examples assigned to that group;
    Remove points from the buffer;

Figure: Single pass k-means

Hierarchical clustering

- Create clusters of various sizes
- two types: Agglomerative and divisive clustering
- An example is BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)
- Extension of BIRCH for data streams is CluStream [Aggarwal et al., 2003]

BIRCH- Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

- agglomerative clustering integrated with other flexible clustering methods
- low level micro clustering preserves inherent clustering structure
- high level macro clustering - (clustering CF-s) provide flexibility for integration with other methods
- using a tree representation to find the closes CF
- parameters: branching factor maximum number for children and maximum diameter for CF

concerns

- sensitive to order of data points
- fixed size of leaf nodes - clusters might not be so natural

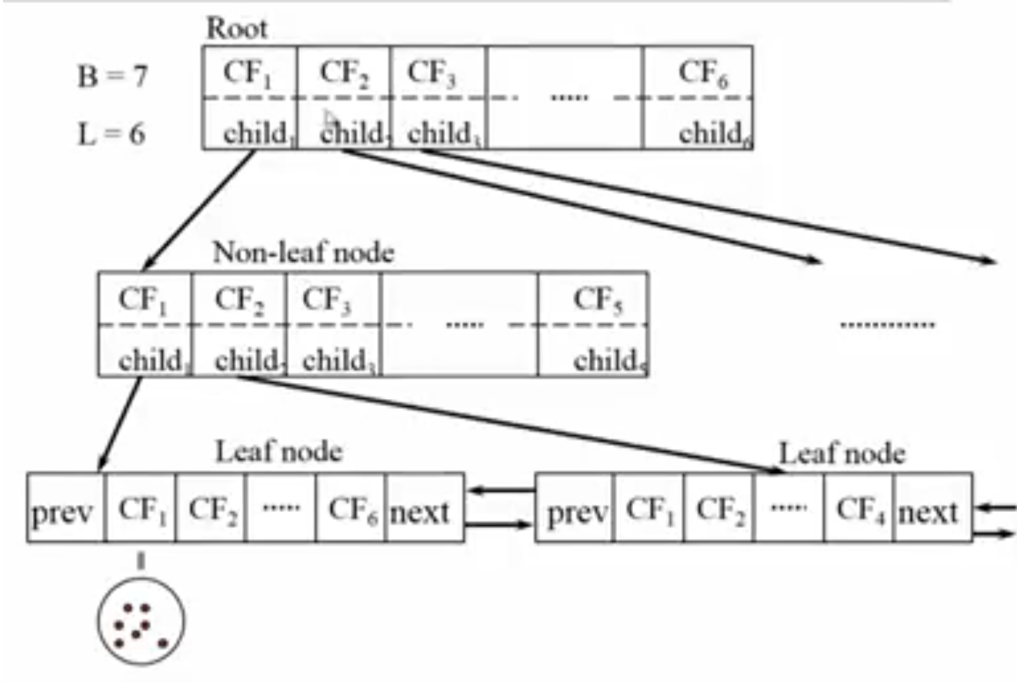
BIRCH - CF -tree

CF-treeTree

1. CF tree a height balanced tree that stores the clustering features
2. non leaf nodes store the sums of its children incremental insertion

Insertion:

1. for each point in the input find the closest leaf entry add point to leaf and update corresponding CF
2. if diameter > max_diameter split the leaf, and possibly parents



CluStream

adaptive extension of BIRCH
additional features :

- LT sum of timestamps
- ST sum of squares of timestamps

Key Ideas:

- It is a two-phase algorithm
- In the first phase, it stores summary statistics of the data points seen so far into **micro-clusters** (with a CF-tree).
- In the second-phase, which is an offline phase, actual clustering happens for example with k-means.

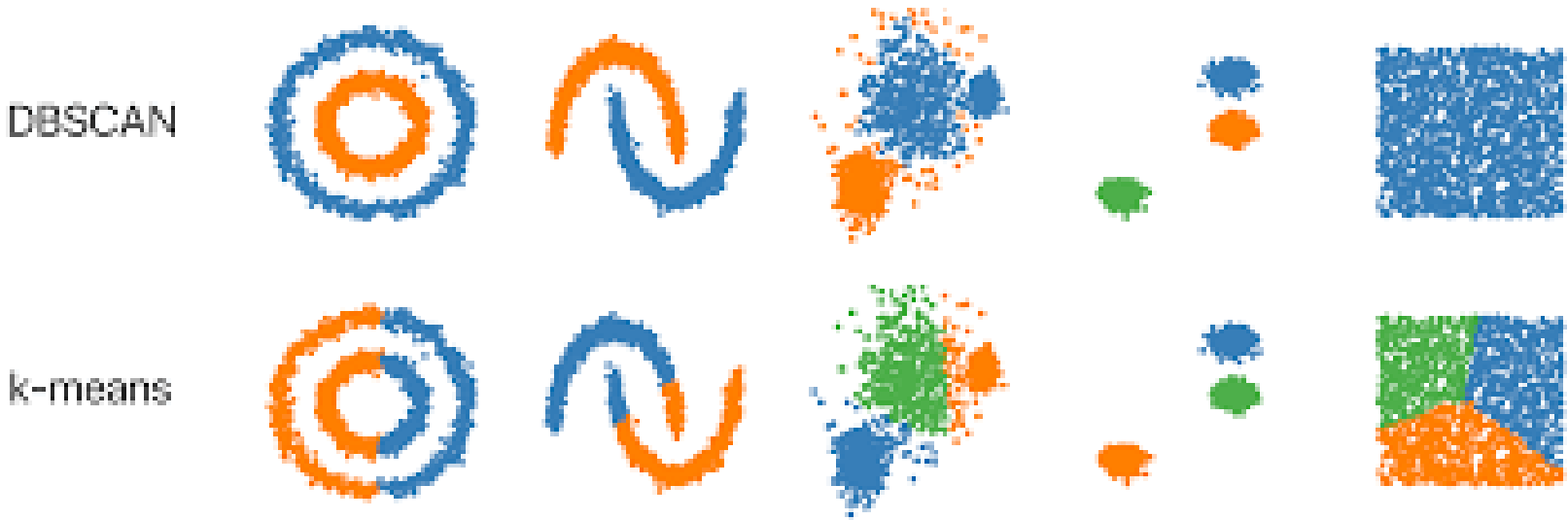
Online Cluster Maintenance

How to maintain clusters online?

- CluStream maintains q initial micro-clusters created by applying k-means algorithm on CF in the offline phase
- Whenever a new data points arrive, calculate distance of it from existing CFs, and if the distance is within the maximum boundary (RMS of deviations of the points) of the CF, absorb it.
- Otherwise create a new micro-cluster with a unique ID associated to it.
- Whenever, number of micro clusters exceeds more than q, either i) delete the oldest micro-cluster or ii) merge two cluster sharing some similarity.

Density based clustering

- clusters might be formed not only by distance (spherical)
- density of connections -



From: `https://github.com/NSHipster/DBSCAN`

Density based clustering

- ϵ-neighbourhood of x set of points a distance $\leq \epsilon$
- core point -whose ϵ- neighbourhood has a weight (fraction of datapoint)$\geq \mu$
- *density area* union of ϵ-neighbourhoods of the core points
- a point y is directly reachable from x if y is in the ϵ-neighbourhood of x
- a point y is reachable from x if there is a sequence $x_1, \ldots, x_n$, that $x_1 = x$ and $x_n = y$, and each x_{i+1} is directly reachable from x_i
- a core point c forms a cluster with all the points reacheable from it
- outliers: oints that are not reachable from core points

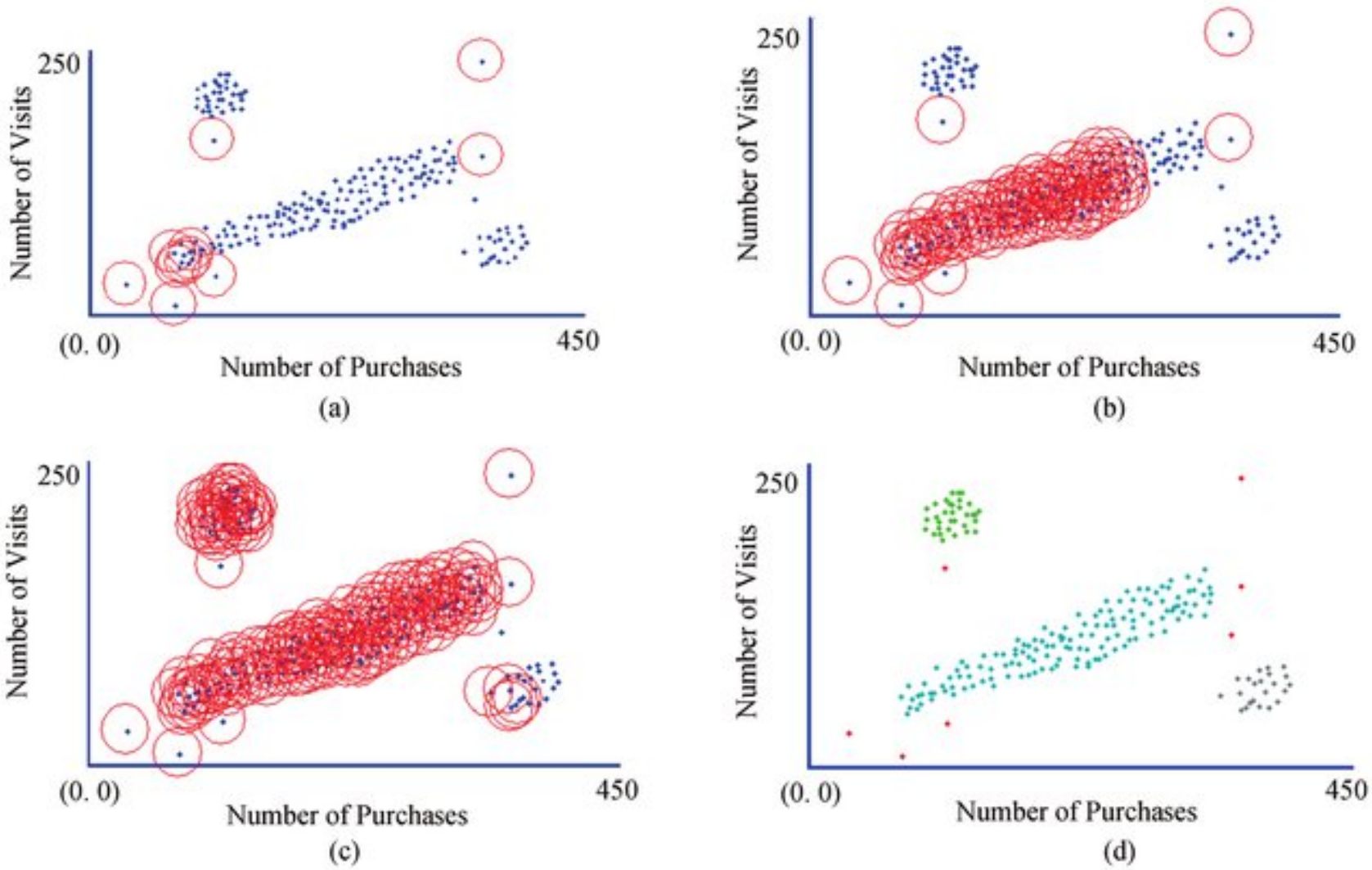
DBSCAN

A cluster is described by any core point
unique clustering given ϵ and μ

- visits the points p in an arbitrary order, and finds all directly ϵ-reachble points from it
- if p found to be μ- core point new cluster is formed and all points reachable will added to it, and marked visited
- otherwise temporarily marked as outlier
- outlier later might be found asto be reacheable from a core point

Strongly nonstreaming - multiple pass

DBscan - visulization:



https://www.researchgate.net/publication/258442676_On_Density-Based_Data_Streams_Clustering_Algorithms_

Den-Stream

- based on idea of dbscan
- uses microclusters - to compute online statistics quickly
- damped window model -weight of each data point decreases exponentially $f(t) = 2^{-\lambda t}, \lambda > 0$
- for a microcluster with center c and points $p_1, \ldots, p_n$, with the corresponding timestamps $T_1, \ldots, T_n$ at each timestep t the algorithm updates the following values:

$$\text{Weight} w = \sum_{i=1}^{n} f(t - T_i) \tag{3}$$

$$\text{Center} c = \sum_{i=1}^{n} f(t - T_i) p_i / w \tag{4}$$

$$\text{Radius} r = \sum_{i=1}^{n} d(p_i, c) / w \tag{5}$$

(for the radius in fact sum of the squares pf p_i is kept, from which r can be computed whenever it is needed)

Den-Stream

microclusters

- o(outlier)-microcluster when $w < \mu$
- p(potential)-microcluster when $w > \mu$

two phase clustering
online: when a new datapoint arrives:

1. try to merge with a p-mc
2. if not possible - try to merge with an o-mc $\rightarrow$ if its weight increases over μ promote it
3. if no merge - new microcluster

offline:

1. remove o-microclusters
2. batch DBScan over on mc-s

Den-stream

DEN-STREAM(*Stream*, λ, μ, β)

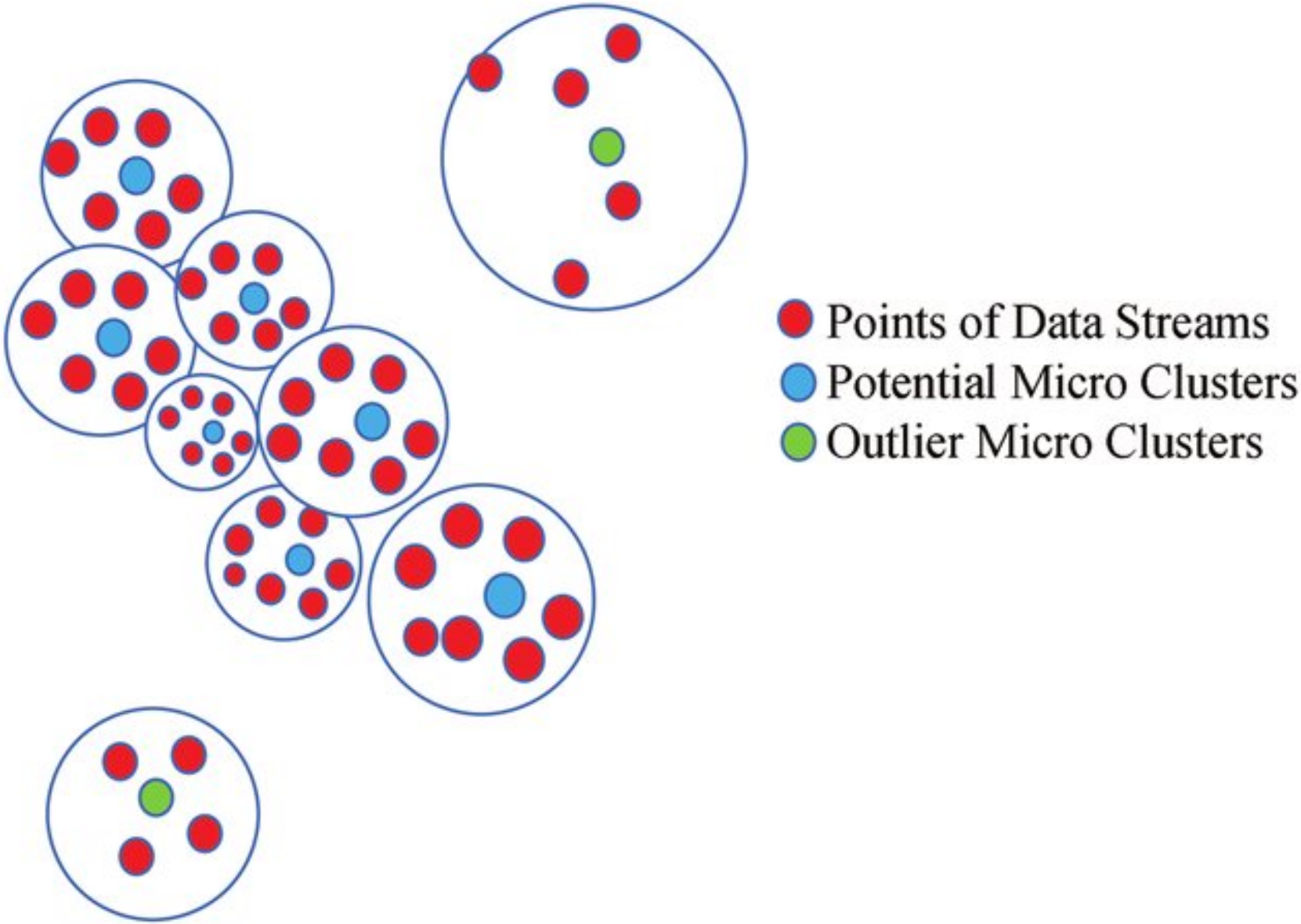
Input: a stream of points, decaying factor λ,
core weight threshold μ, tolerance factor β

▷ Online phase
$T_p \leftarrow \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu - 1}) \rceil$
for each new point that arrives
    do try to merge to a p-microcluster; if not possible,
        merge to nearest o-microcluster
            if o-microcluster weight $> \beta\mu$
                then convert the o-microcluster to p-microcluster
                else create a new o-microcluster

▷ Offline phase
if ($t \bmod T_p = 0$)
    then for each p-microcluster c_p
            do if $w_p < \beta\mu$
                then remove c_p
        for each o-microcluster c_o
            do if $w_o < (2^{-\lambda(t - t_o + T_p)} - 1)/(2^{-\lambda T_p} - 1)$
                then remove c_o
        apply DBSCAN using microclusters as points

Source:https://www.cms.waikato.ac.nz/~abifet/book/chapter_9.html

Den-stream



https://www.researchgate.net/publication/258442676_

Bibliography I

Aggarwal, C. C., Watson, T. J., Ctr, R., Han, J., Wang, J., and Yu, P. S. (2003).
A framework for clustering evolving data streams.
In *In VLDB*, pages 81–92.