Data Stream Clustering

Presented by : Sören Becker & Kumar Awanish

Agenda

1. Introduction & Basic Terminology
2. Challenges in online & offline Clustering
3. Different types of clustering
4. Simple Example: Doubling Algorithm
5. CluStream
6. DenStream
7. Stream clustering platforms
8. Summary

Introduction & Basic Terminology

Clustering:

- The process of organizing objects into groups whose members are similar in some way.
- A *cluster* is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

Possible Applications:

- Marketing
- Online Shopping

Introduction & Basic Terminology

Data stream clustering :

- It is defined as the clustering of data that arrive continuously such as telephone records, multimedia data, financial transactions etc.

Main Method:

Partitioning : K-Means ...

Hierarchical: BIRCH...

Density-Based : DBSCAN...

Challenges in Clustering

- Volume: Not possible to store all the data
- One-time access: Not possible to process the data using multiple passes
- Real-time analysis: Certain applications need real-time analysis of the data
- Cluster Validity

Challenges in Clustering

- Load- shedding in Data Streams
- High dimensional data stream
- Protecting privacy and confidentiality
- Temporal Locality: Data evolves over time, so model should be adaptive

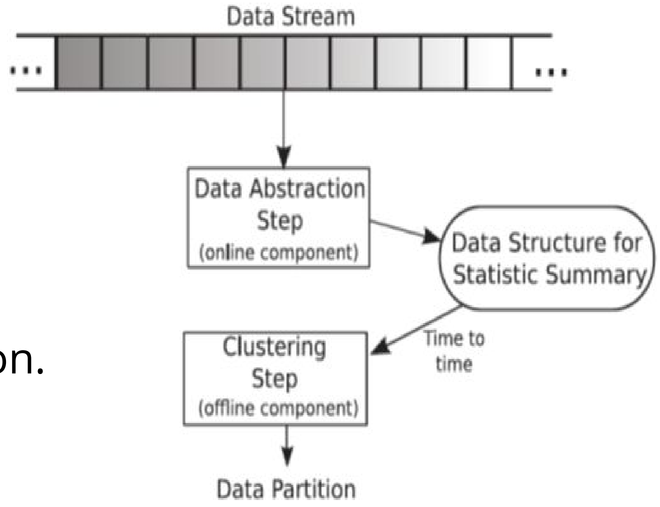
Online vs Offline Approaches

Online :

- Pulling of data when required, on demand.
- Summarize the data into memory-efficient data structures.
- Micro-clustering component

Offline:

- Push of data in large database.
- Use a clustering algorithm to find the data partition.
- Macro-clustering component



K-center problem

Problem settings:

Parameter *k* (number of clusters)

Set of points *S* (data)

Distance function *D*

Goal:

Find set $C = \{c_1, \ldots, c_k\}$ from *S* such that $\max_{x \in S} \min_{c_i \in C} D(c_i, x)$

is minimized.

Approximation guarantees

Definition:

A *p-approximation* algorithm is defined for a minimization problem to be an algorithm that gives an approximate solution which is at most *p* times larger than the true minimum solution.

Offline:

Farthest point algorithm: 2-approximation for k-center problem

Online:

Doubling algorithm: 8-approximation for k-center (streaming) problem

Doubling algorithm

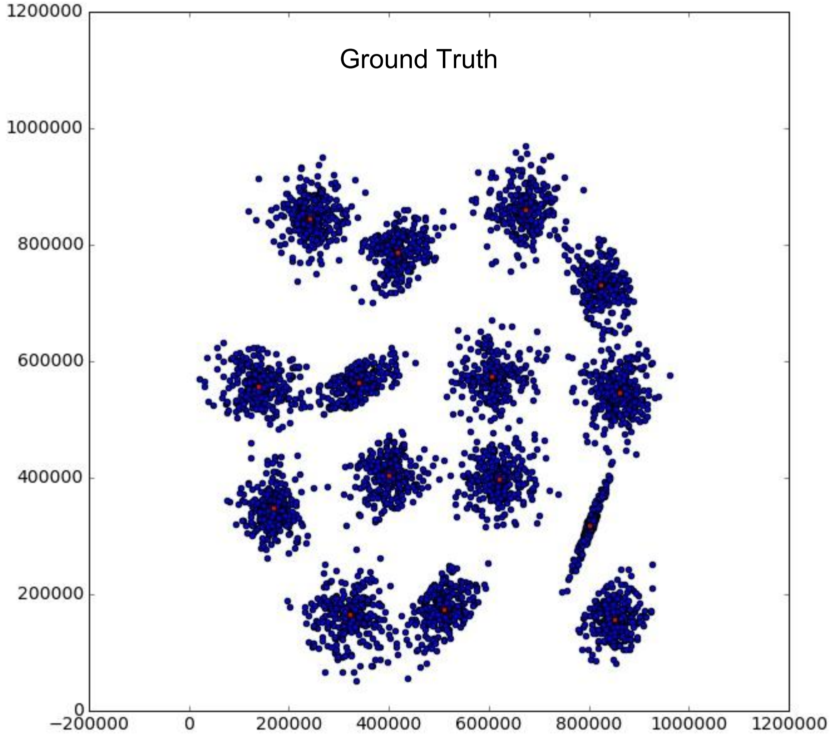
Input: sequence of points from a metric space, arriving one at a time.

Output: set of k centers.

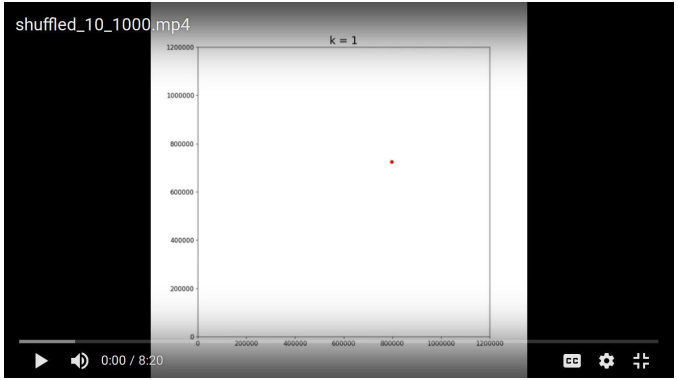
Init Choose first k points as centers and r $\leftarrow$ 0
for *every new point i* **do**
  Suppose the centers are $c_1, ..., c_l$
  if *i is within distance 4r from any center* **then**
    assign i to that cluster (choose arbitrarily if there are more than one)
  else
    if $l < k$ **then**
      A. make $c_{l+1} = i$, i.e., start a new cluster with i as its center
    else
      A. Let the smallest distance between any pair of points in the set $C = c_1, ..., c_k, i$ be t
      B. r $\leftarrow \frac{t}{2}$
      C. Pick a point from C and let this be a new center c'. Remove all points from C that are within distance $4r$ from this center c'. All the clusters (corresponding to the removed centers and possibly the singleton cluster i) are merged into a cluster with center c'.
      D. The above step is repeated until C is empty. These centers are carried over to handle the next point.
    end
  end
end
Result: Output the remaining cluster centers

Sudipto Guha & Nina Mishra: Clustering Data Streams, Springer 2016

Doubling algorithm: Demo



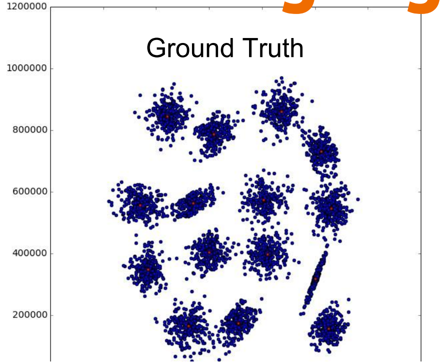
Synthetic 2-d dataset with 5000 points from 15 Gaussian clusters

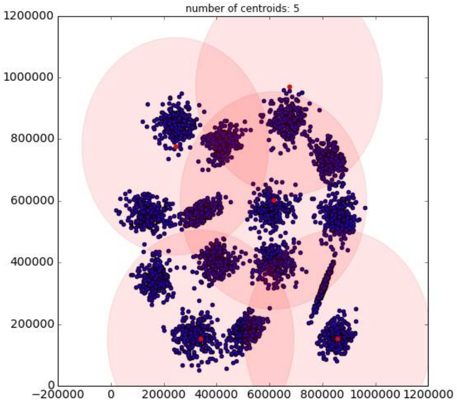


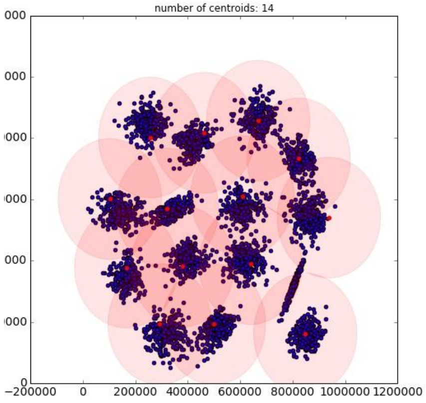
P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems", *Pattern Recognition*, 39 (5), 761-765, May 2006.

Doubling algorithm: Results

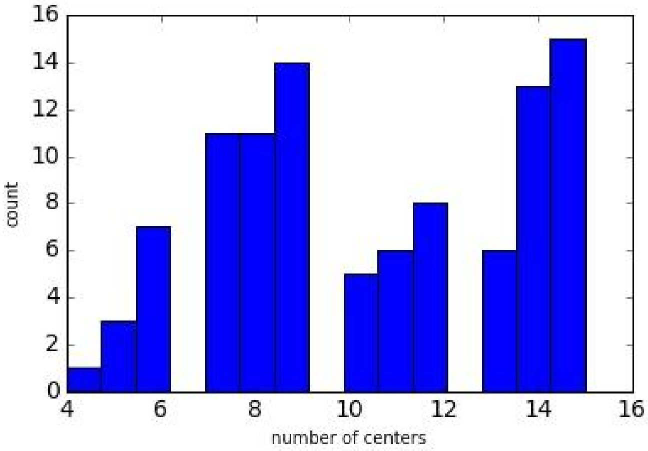
- High sensitivity to order of data input
- No information about the evolution of the data







Distribution after 100 runs



Type of Clustering

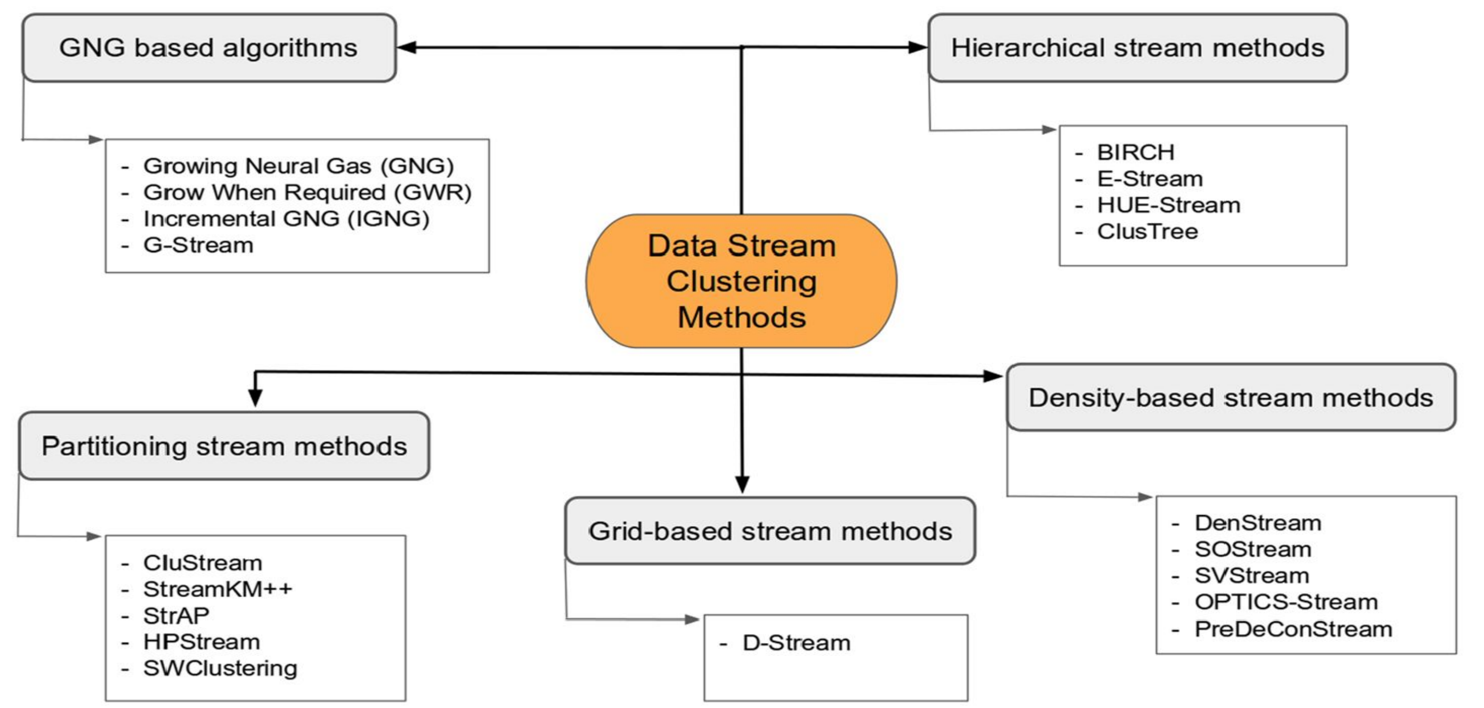


Fig. 1 Data stream clustering methods. The presented algorithms categorized according to the nature of their underlying clustering approach

CluStream:

- Cluster streaming data
- Cope with temporal variations in the data
- Allow time-horizon analysis

https://scholar.google.com/citations?view_op=view_citation&hl=en&user=x_wsduUAAAAJ&citation_for_view=x_wsduUAAAAJ:u5HHmVD_uO8C

CluStream

General Setting

- D-dimensional data chunks $X_1, \ldots, X_k$ arriving at time stamps $T_1, \ldots, T_k$
- For each chunk: $X_i = (x_i^1, \ldots, x_i^d)$

Core concepts:

- Microclusters
- Pyramidal Time Frame

CluStream: Micro-clusters

Micro-clusters are (2*d + 3) tuples of simple statistics.

For chunks $(X_i)_1, \dots, (X_i)_n$ with time stamps $(T_i)_1, \dots, (T_i)_n$, a tuple consists of:

- $CF2^x$: elementwise sum of squares over $(X_i)_1, \dots, (X_i)_n$ ([p x 1]-vector)
- $CF1^x$: elementwise sum over $(X_i)_1, \dots, (X_i)_n$ ([p x 1]-vector)
- N : number of data points (scalar)

CluStream: Micro-clusters

A Micro-cluster M_i is a (2*d + 3) tuple of simple statistics.

For chunks $(X_i)_1, \dots, (X_i)_n$ with time stamps $(T_i)_1, \dots, (T_i)_n$, M_i consists of:

- $CF2^x$: elementwise sum of squares over $(X_i)_1, \dots, (X_i)_n$ ([p x 1]-vector)
- $CF1^x$: elementwise sum over $(X_i)_1, \dots, (X_i)_n$ ([p x 1]-vector)
- N : number of data points (scalar)

- $CF2^t$: elementwise sum of squares over $(T_i)_1, \dots, (T_i)_n$ (scalar)
- $CF1^t$: elementwise sum over $(T_i)_1, \dots, (T_i)_n$ (scalar)

Temporal extension of BIRCH cluster features!

CluStream: Micro-clusters

Cluster Features contain information to easily obtain

Centroid:

$$\bar{x} = \frac{1}{N}\sum_i^N x_i$$

RMS deviation from centroid:

$$\sqrt{\frac{1}{N}\sum_i^N (x_i - \bar{x})^2} = \sqrt{\frac{1}{N}(\sum_i^N x_i^2 - 2\bar{x}x_i + \bar{x}^2)}$$

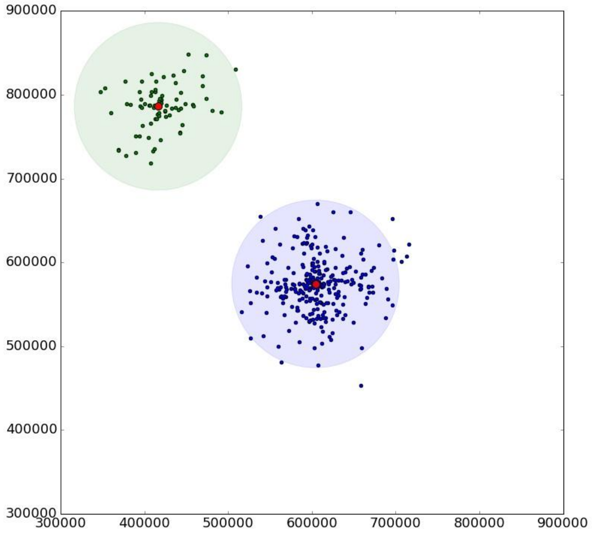
$$= \sqrt{\frac{1}{N}(\sum_i^N x_i^2 - 2\bar{x}\sum_i^N x_i + N\bar{x}^2)}$$

- $CF2^x$: elementwise sum of squares over $(X_i)_1, \ldots, (X_i)_n$ ([p x 1]-vector)
- $CF1^x$: elementwise sum over $(X_i)_1, \ldots, (X_i)_n$ ([p x 1]-vector)
- N : number of data points (scalar)

CluStream: Micro-clusters

Additivity property: merging clusters becomes easy

$$CF_{new} = CF_1 + CF_2$$



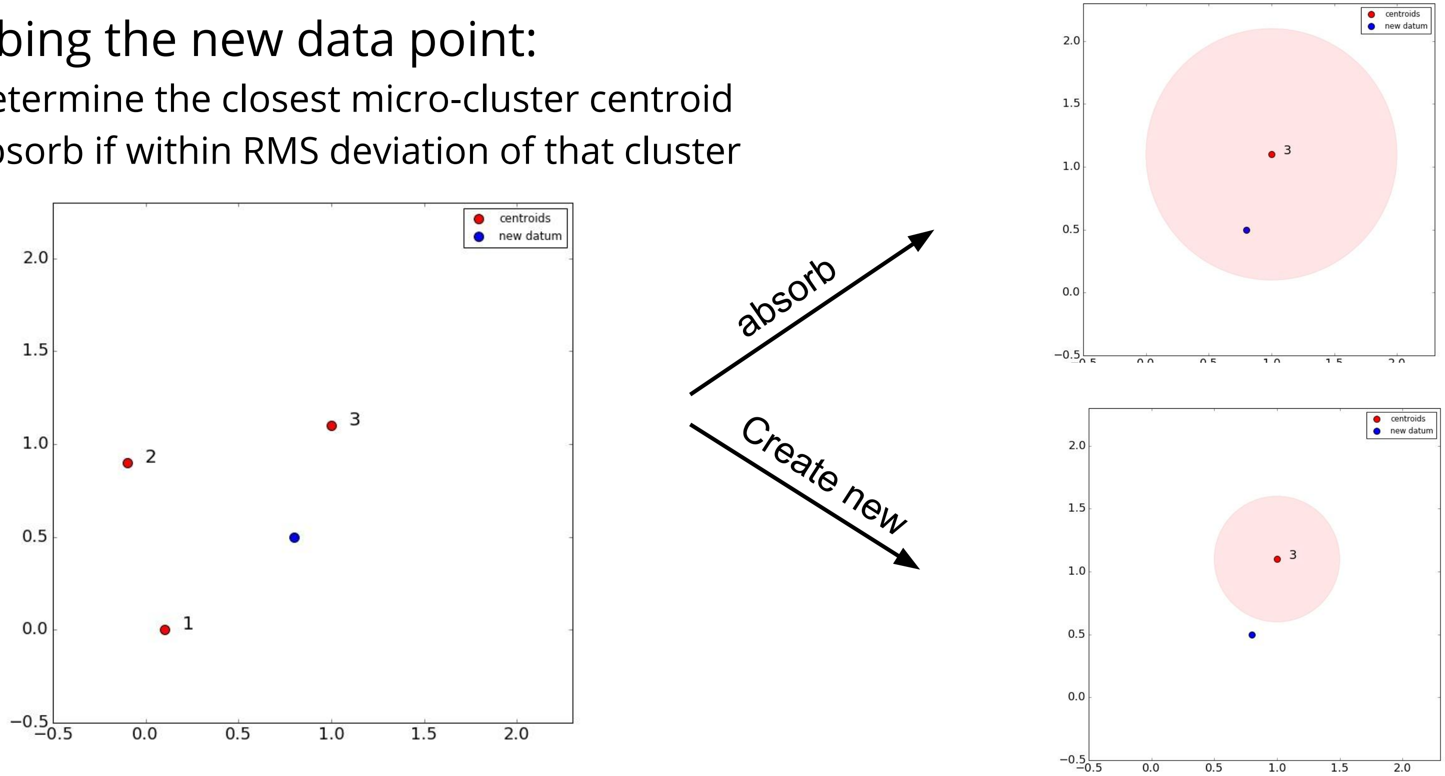
CluStream: Online clustering

1. Initialize:
    a. Store the first *initNumber* points on disk
    b. Use k-means to create *q* initial micro-clusters

2. Online updating:
    a. Absorb new data points into a micro-cluster *or*
    b. Create a new cluster for the data point

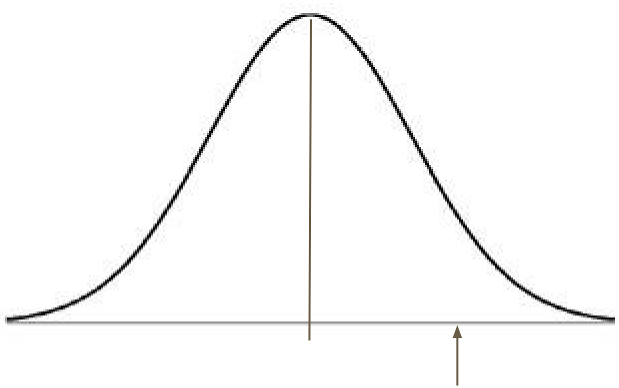
CluStream: Online clustering

a) Absorbing the new data point:
   i) Determine the closest micro-cluster centroid
   ii) Absorb if within RMS deviation of that cluster



CluStream: Online clustering

a) Creating a new cluster:
   i) Create a new cluster with the new data point as centroid
   ii) Reduce number of old clusters to make space for new cluster
      1) Delete an old cluster based on average and variance of timestamps:
      2) If no cluster can be deleted, merge the two closest cluster



Assume: Gaussian distribution of time stamps of a micro-cluster

Find time stamp of predefined percentile: delete cluster if above predefined threshold

CluStream: Pyramidal Time Frame

Snapshots: (depend on parameters *α* and *L)*

- Micro-clusters are stored (in secondary memory) at particular moments in time.
- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

$L = 1$

$\alpha = 2$

Time															1
2^0															x
2^1															
2^2															
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time														1	2
2^0														x	x
2^1															x
2^2															
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time													1	2	3
2^0													x	x	x
2^1														x	
2^2															
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

L = 1

Time												1	2	3	4
2^0													x	x	x
2^1													x		x
2^2															x
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time											1	2	3	4	5
2^0													x	x	x
2^1												x		x	
2^2														x	
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time										1	2	3	4	5	6
2^0													x	x	x
2^1											x		x		x
2^2													x		
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time									1	2	3	4	5	6	7
2^0													x	x	x
2^1										x		x		x	
2^2												x			
...															
2^∞															

CluStream: Pyramidal Time Frame

- For any positive integer i, store a snapshot of order i whenever the current timestamp is divisible by α^i.
- Store at most $\alpha^L + 1$ snapshots of order i.

Time								1	2	3	4	5	6	7	8
2^0													x	x	x
2^1											x		x		x
2^2											x				x
...															
2^∞															

CluStream: Pyramidal Time Frame

User: “Where were the centroids in the last 5 [time units]”?

$$\overbrace{10 \quad 11 \quad 12 \quad 13 \quad 14}^{5}$$

Time	...	8	9	10	11	12	13	14
2^0						x	x	x
2^1				x		x		x
2^2		x				x		
...								
2^∞								

- Find micro-clusters $(CF_{14})_{1..N}$ and $(CF_8)_{1..N}$
- For i = 1..N: $((CF_{14})_i$ - $(CF_8)_i$) approximates micro-cluster i of the desired period of 5 time units.
- Use an offline algorithm to cluster the micro-cluster snapshots of the desired period

CluStream: Quality vs. Space

Approximate any time horizon h with factor $\mathbf{(1 + 1/\alpha^{L-1})}$ with storage requirement of $\mathbf{(\alpha^{L} + 1)*log_{\alpha}(T)}$ snapshots.

Example: Stream running for 100 years, sending samples every second.

With $\alpha = 2$ and $L = 10$:

Quality: Approximate any horizon within 0.2%.

Space: Need to store 32343 snapshots.

CluStream: Scalability

Samples per second

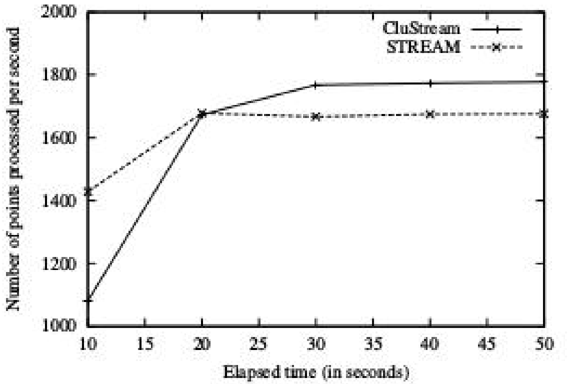


Figure 5: Stream Processing Rate (Charitable Donation dataset, stream_speed=2000)

Number of data dimensions

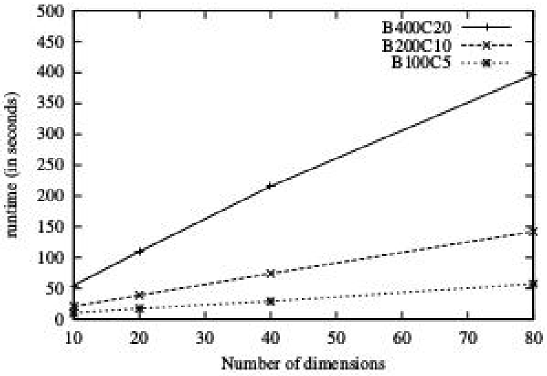


Figure 7: Scalability with Data Dimensionality (stream_speed=2000)

Number of clusters

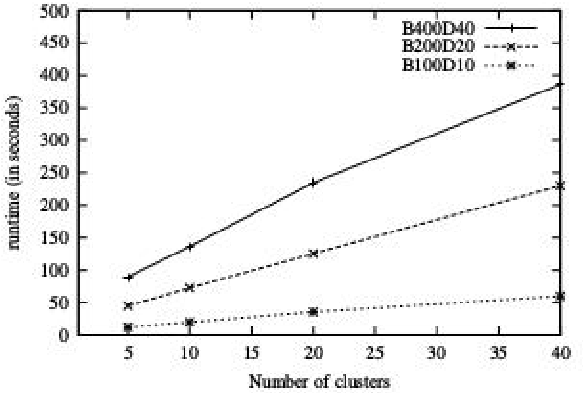
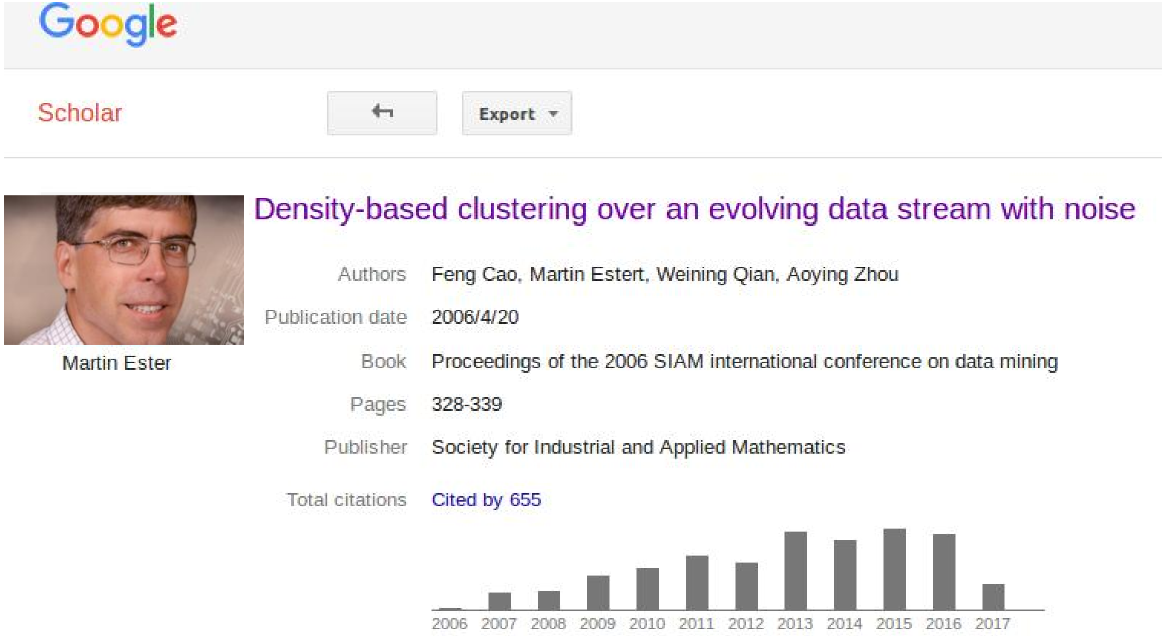
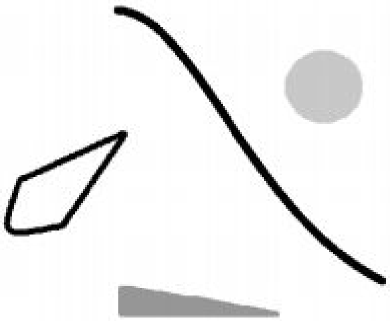


Figure 8: Scalability with Number of Clusters (stream_speed=2000)

DenStream- Motivation





- DenStream finds clusters of arbitrary shape.
- DenStream can handle noise.

https://scholar.google.de/citations?view_op=view_citation&hl=en&user=ZYwC_CQAAAAJ&citation_for_view=ZYwC_CQAAAAJ:zYLM7Y9cAGgC

DenStream- Terminology

Damped Window, weight of data objects decreases exponentially over time: $f(t) = 2^{-\alpha \cdot t}, \alpha > 0$

Micro-Clusters: $MC = (WLS, WSS, w, t_c)$, where

- $WLS = \sum_{i=1}^{n} f(t - T_i) \cdot p_i$ (Weighted Linear Sum)
- $WSS = \sum_{i=1}^{n} f(t - T_i) \cdot p_i^2$ (Weighted Squared Sum)
- w (Weight of MC)
- t_c (Creation Time of MC)

DenStream- Terminology

c = WLS / w (center of MC)

r= $\sqrt{\frac{\|WSS\|_2}{w} - \left(\frac{\|WLS\|_2}{w}\right)^2} \leq \varepsilon$; ε= Maximum radius

If w ≥ ***μ***, MC is a core -micro-cluster.

Potential core-micro-clusters (p-micro-clusters), with w ≥ β · μ, β is a parameter of the algorithm.

Outlier micro-clusters (o-micro-clusters), with w < β · μ

Micro-Clusters can be maintained incrementally:

- MC = $(2^{-\alpha.\delta t} \cdot WLS, 2^{-\alpha.\delta t} \cdot WSS, 2^{-\alpha.\delta t} \cdot w, tc)$
- If a point p is merged, MC = (WLS +p,WSS +p 2 ,w +1,tc).

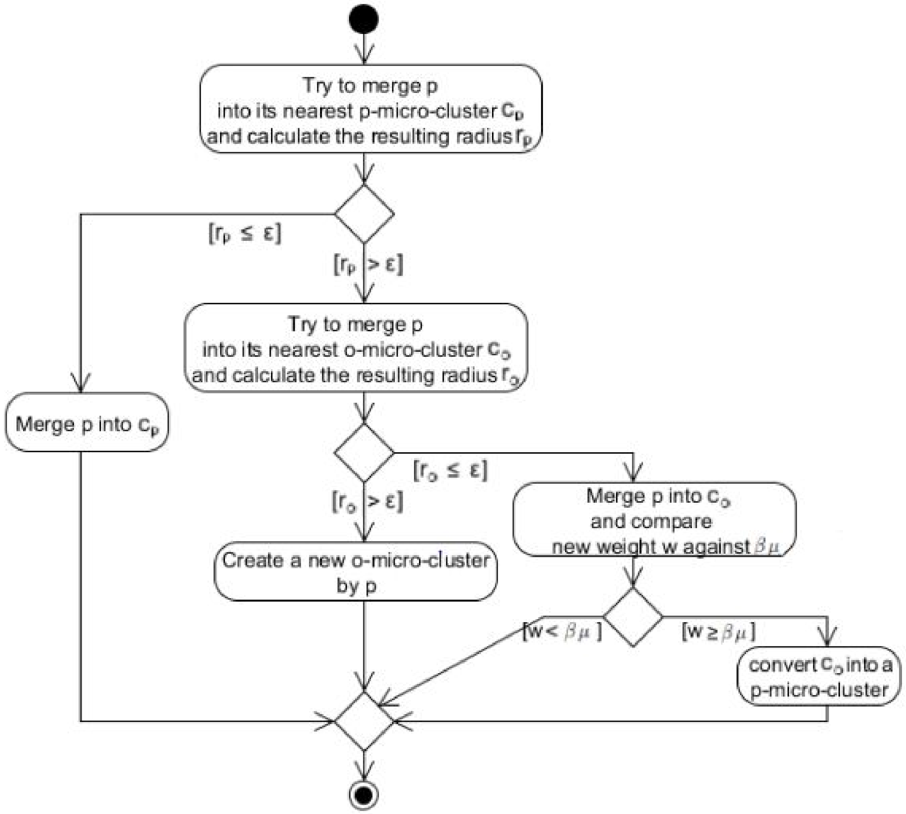
DenStream: Algorithm Overview

- Online and Offline Part
- Initialization with DBSCAN
- Maintains p-micro-clusters and o-micro-clusters during online-component

  New points are merged using a Merging Algorithm
- Pruning Strategy is performed periodically
- DBSCAN based offline component generates final clusters on demand using p-micro-clusters as virtual points

Merging Technique

Try to merge p into its nearest p-micro-cluster c_p;
if r_p (the new radius of c_p) $\leq \epsilon$ **then**
  Merge p into c_p;
else
  Try to merge p into its nearest o-micro-cluster c_o;
  if r_o (the new radius of c_o) $\leq \epsilon$ **then**
    Merge p into c_o;
    if w (the new weight of c_o) $> \beta \cdot \mu$ **then**
      Delete c_o and create a new p-micro-cluster for i
    end if
  else
    Create a new o-micro-cluster by p;
  end if
end if

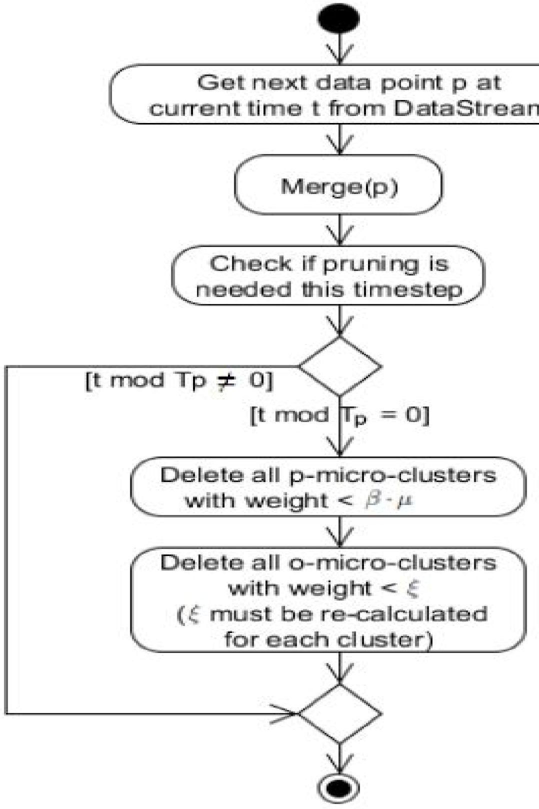


Pruning Strategy

- Pruning strategy is performed every Tp time steps
- $Tp = [1/ \alpha \cdot \log_2(\beta \cdot \mu / \beta (\mu - 1))]$
- All p-micro-clusters with weight $w < \beta \cdot \mu$ are pruned
- O-micro-clusters must be pruned too to release memory space
- If o-micro-clusters are pruned too early, they can't become p-micro-clusters

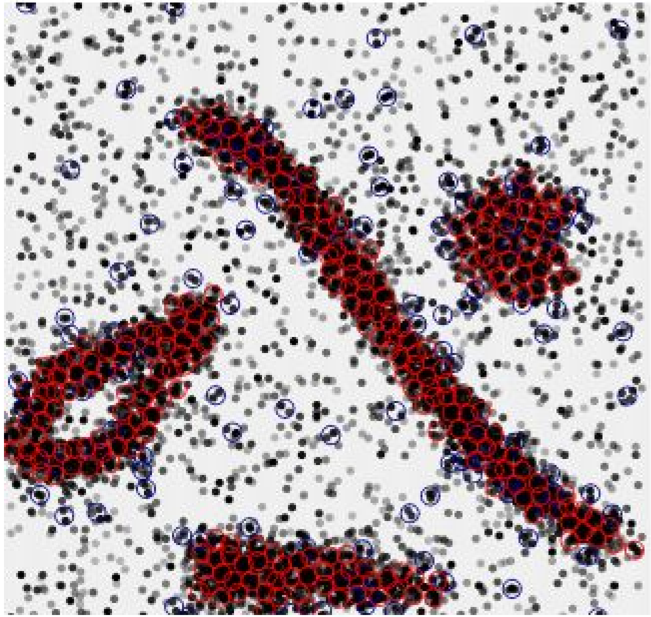
DenStream: The Algorithm

$T_p \leftarrow \lceil \frac{1}{\alpha} \cdot \log_2 \left(\frac{\beta \cdot \mu}{\beta\mu - 1} \right) \rceil$;
Get the next point p at current time t from $DataStream$;
Merging(p);
if $(t \mod T_p) = 0$ **then**
  for each p-micro-cluster c_p **do**
    if w_p (the weight of c_p) $< \beta \cdot \mu$ **then**
      Delete c_p;
    end if
  end for
  for each o-micro-cluster c_o **do**
    $\xi \leftarrow \frac{2^{-\alpha(t - t_0 + T_p)} - 1}{2^{-\alpha T_p} - 1}$;
    if w_o (the weight of c_o) $< \xi$ **then**
      Delete c_o;
    end if
  end for
end if
if a clustering request arrives **then**
  Generate clusters;
end if

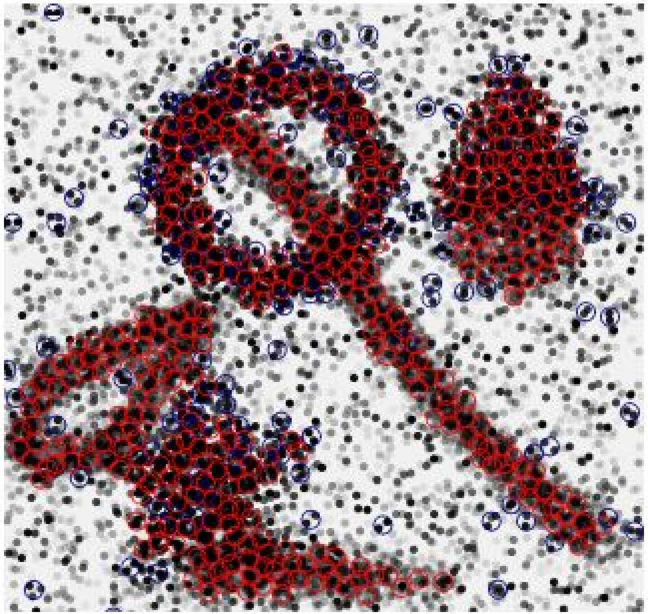


Micro-clusters for initial distribution

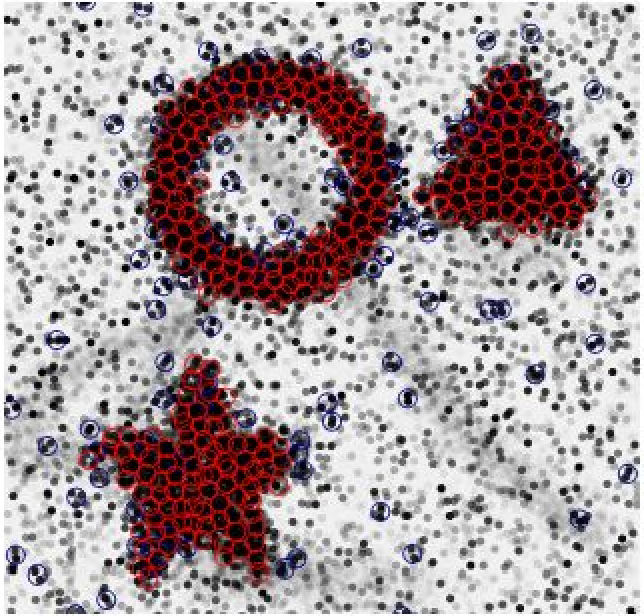
- Resulting micro clusters of the initial distribution.
- Micro clusters are represented by circles.
- Red circles are p-micro-clusters.
- Blue circles denote o-micro-clusters.



Micro-Cluster for Fading & Final Distribution



Micro-Clusters for Fading Distribution



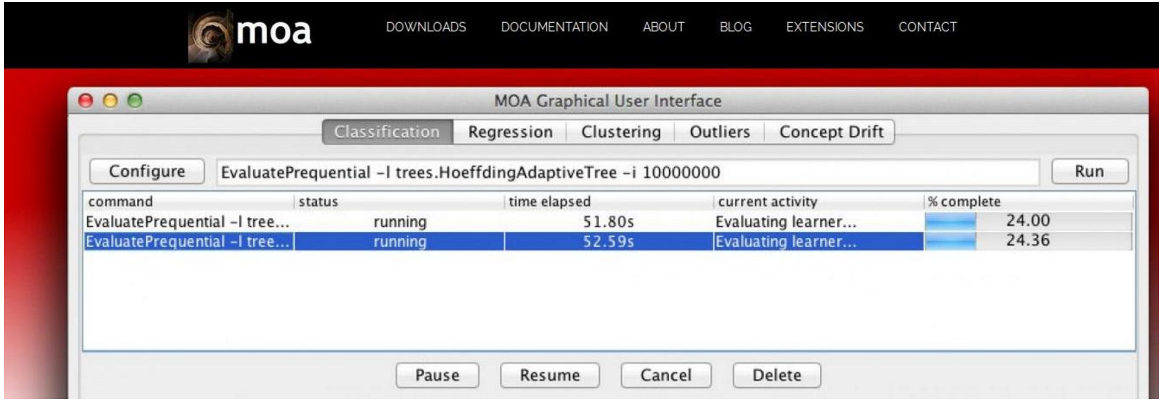
Final Distribution

Comparison of Clustering Algorithms

Table 1 Comparison between algorithms (WL: weighted links, 2 phases : online+offline)

Algorithms	Based on	Topology	WL	Phases	Remove	Merge	Split	Fade
SVStream	SVC, SVDD	✗	✗	online	✓	✓	✓	✓
StreamKM++	*k*-means++	✗	✗	2 phases	✓	✓	✓	✓
StrAP	AP	✗	✗	2 phases	✓	✗	✗	✓
SOStream	DBSCAN, SOM	✗	✗	online	✓	✓	✗	✓
OPTICS-Stream	OPTICS	✗	✗	2 phases	✓	✓	✗	✓
IGNG	NG	✓	✗	online	✗	✗	✗	✗
HCluStream	*k*-prototypes	✗	✗	2 phases	✓	✓	✓	✓
GWR	NG	✓	✗	online	✗	✗	✗	✗
G-Stream	NG	✓	✓	online	✓	✗	✗	✓
E-Stream	*k*-means	✗	✗	2 phases	✓	✓	✓	✓
D-Stream	-	✗	✗	2 phases	✓	✓	✓	✓
DenStream	DBSCAN	✗	✗	2 phases	✓	offline	✗	✓
ClusTree	*k*-means or DBSCAN	✗	✗	2 phases	✓	offline	✓	✓
CluStream	*k*-means	✗	✗	2 phases	✓	offline	✗	✗
AING	NG	✓	✗	online	✗	✓	✗	✗

Streaming Platform -MOA(Massive On-Line Analysis)



- Open Source project
- Framework for data stream mining
- Support Machine Learning algorithm
- Goal is to provide a benchmark suite for the stream mining community
- It contains several stream clustering methods including: StreamKM++, **CluStream**, ClusTree, **DenStream**, D-Stream

Summary

- Concepts of Data Streaming Clustering
- Different types of Data Streaming Clustering
- Challenges in clustering
- Different platform for clustering
- Algorithms: Doubling, CluStream, DenStream
- Comparison of different clustering algorithms

References

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, VLDB '03, pages 81–92. VLDB Endowment, 2003.

[2] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *J. Mach. Learn. Res.*, 11:1601–1604, August 2010.

[3] Feng Cao, Martin Estert, Weining Qian, and Aoying Zhou. *Density-Based Clustering over an Evolving Data Stream with Noise*, pages 328–339.

[4] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–765, 2006.

[5] Mohammed Ghesmoune, Mustapha Lebbah, and Hanene Azzag. State-of-the-art on clustering data streams. *Big Data Analytics*, 1(1):13, 2016.

[6] Sudipto Guha and Nina Mishra. *Clustering Data Streams*, pages 169–187. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.