Lecture: Machine Learning for Data Science

Winter semester 2021/22

Lectures 25 & 26: Velocity (stream clustering)

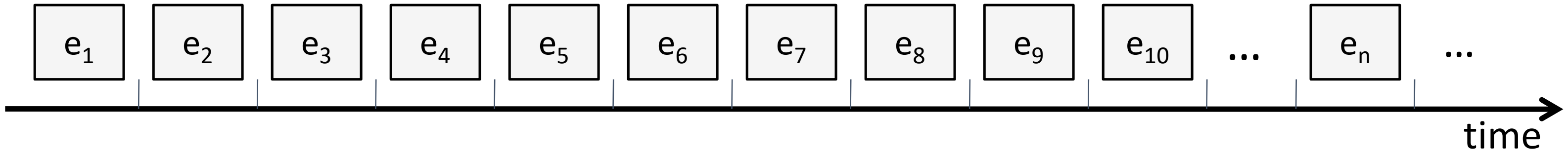
Prof. Dr. Eirini Ntoutsi

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

(recap from previous week) Data streams

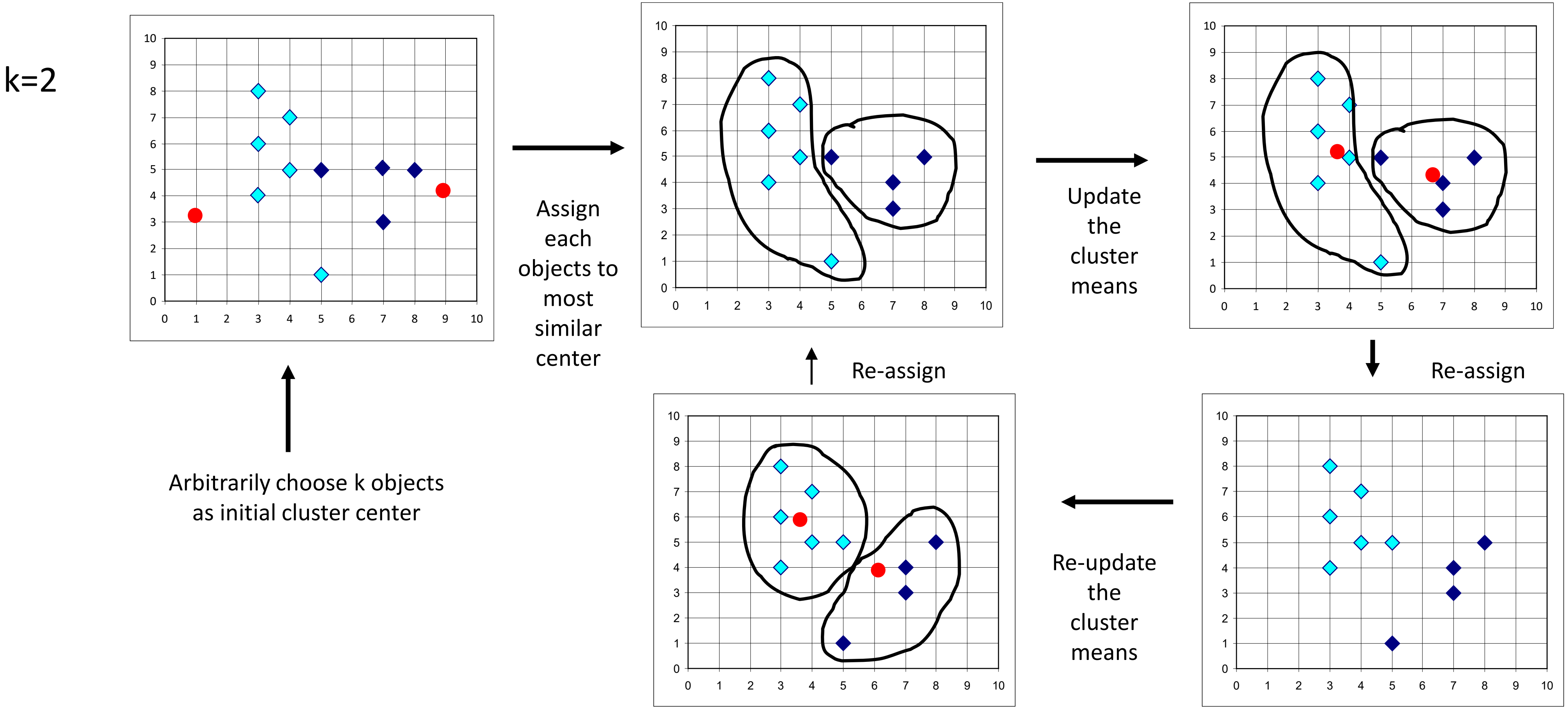
- "A data stream is a potentially unbounded, ordered sequence of data items, which arrive continuously at high-speeds" Springer Encyclopedia of Machine Leaning, 2017



- Key characteristics
  - Huge volumes of continuous data, possibly infinite: Random access is expensive or undesirable (due to e.g., privacy)
  - High arrival rate: response time matters
  - Non-stationary/ evolving data: Data evolve over time as new data arrive and old data become obsolete/irrelevant

(recap from previous week) Example: batch k-Means (see also lecture 10)

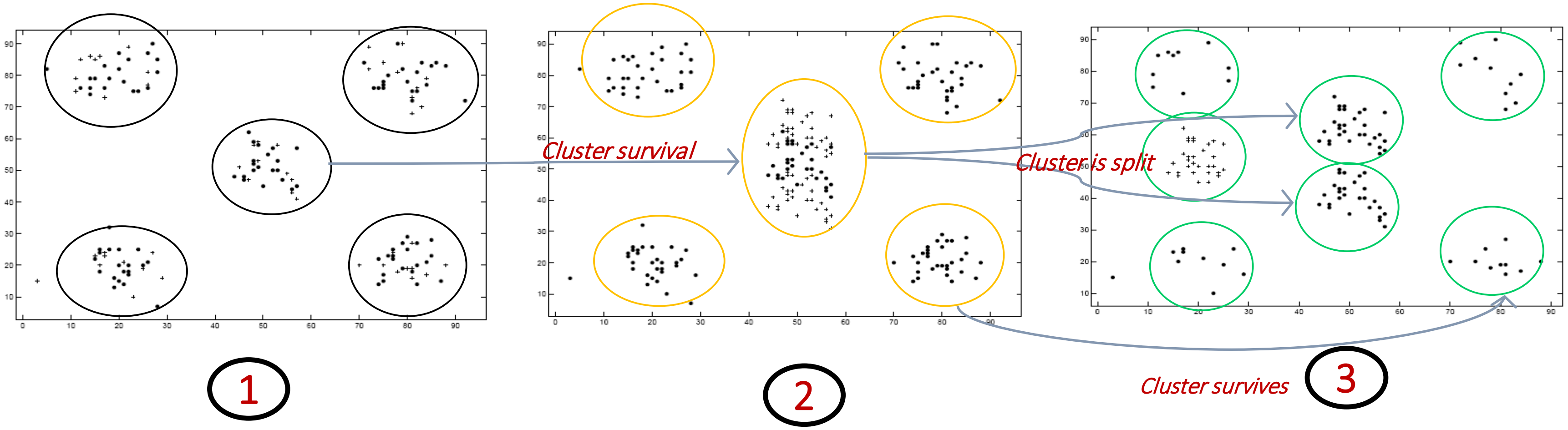
- The complete dataset is given as input to the algorithm
- The dataset is accessed multiple times during the iterations



(recap from previous week) Why the stationarity violation is problematic?

Unsupervised learning case

- As data evolve with time, the clustering is becoming invalid/obsolete
- External changes: the relationship of a cluster to the other clusters might change, e.g., cluster survival, split, merge, appearance, disappearance
- Internal changes: the description of a cluster might change both externally (i.e., cluster members) and internally (cluster properties)



Source: The MONIC framework, Spiliopoulou et al, KDD06

(recap from previous week) Requirements for stream learning

- Need for new learning algorithms that
  - have the ability to incorporate new data (incremental models)
  - deal with non-stationary data generation processes
    - Ability to discard obsolete data (or, obsolete (parts of the) model) (data ageing/ forgetting)
- subject to:
  - resource constraints (processing time, memory)
  - single scan of the data (one look, no random access)

Clustering data streams

- Clustering is one of the core learning tasks
  - Used as either a standalone tool or as a preprocessing tool
- The (batch) clustering problem:
  - Given a set of data instances, the goal is to group the data into groups of similar data (clusters)
  - The dataset is available from the beginning to the algorithm
  - The algorithm is allowed to iterate over the dataset
- The data stream clustering problem:
  - Maintain a good clustering over the (non-stationary) stream, subject to resource constraints
    - No random access to the data (you only have a look at the data instances when they arrive)
    - The processing time per instance should be low

Challenges & Requirements for data stream clustering

- Traditional clustering methods require access upon the whole dataset

→ work with summaries, rather than raw data

- The underlying population distribution might change:

→ the clustering structure needs to be maintained/adapted online

→ one clustering model might not be adequate to capture the evolution of the underlying population

- The role of outliers and clusters are often exchanged in a stream

→ timely and accurate identification of outliers is necessary

Outline

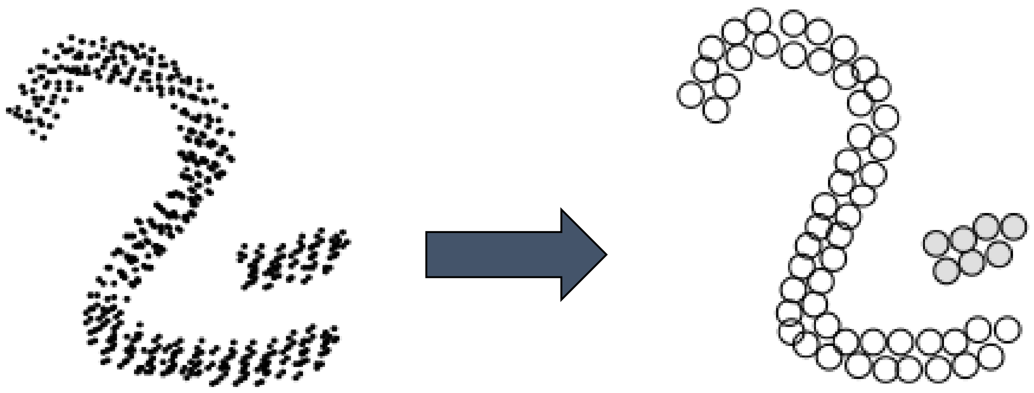
- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

Dealing with the efficiency requirements

- In the stream classification part, we mainly focused on the non-stationary aspect and we didn't directly address efficiency!
  - For the clustering part, many of the methods directly address efficiency (and of course aim at good quality clustering results)
- Efficiency challenge (not restricted to clustering): How can we learn from high volumes of data faster?
- Performance depends on
  - the volume of the data set (cardinality, dimensionality)
  - the scalability of the learning algorithms
  - …
- Solutions for speeding up learning
  - Use high-performance computing architectures
    - Parallel computing; Distributed computing; Cloud computing; …
  - Reduce the number of objects being processed
    - Summarization/Compression: "compress" the data using "higher-level" descriptors (summaries) (the most relevant for our discussion on stream clustering)
    - Sampling: select a subset of the data to work on
    - Keep quality data (lately known as data-centric AI)
      - Andrew Ng is one of the supporters of this idea, listen e.g., "A Chat with Andrew on MLOps: From Model-centric to Data-centric AI"
  - Develop more efficient methods ☺
  - ….

Summarization/Compression

- Summarization/Compression is one way to speed up learning
- Main idea:
  - Summarize/Compress the input data into a set of summaries. Original/ raw data are discarded.
  - Apply machine learning algorithms upon the summaries afterwards
- Why does summarization makes sense?
  - Summaries comprise lossy but still good representations of the original raw data
  - Having good summaries is of course critical!



Raw data

Summaries

Summarization/Compression

- Examples of summaries
  - Cluster feature vectors [Zhang et al 1996]
  - Data bubbles [Breuning et al 2001]
  - …
- Again, the quality of the summaries is of paramount importance for the success of the learning task

Cluster-feature (CF) vectors/ BIRCH algorithm

- BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [Zhang et al 1996]
  - BIRCH is the first approach for clustering large scale data
- BIRCH introduced the idea of cluster feature vector summaries
  - also known as microclusters in the stream clustering domain
- BIRCH organizes the CF summaries into a tree structure
  - CF tree: A multi-level compression of the data that tries to preserve the inherent clustering structure of the data

Cluster feature vector (CF) summaries or micro-clusters

Given N d-dimensional points in a cluster C, the cluster feature (CF) vector of C is defined as a triple:

$$CF = (N, \overrightarrow{LS}, SS)$$

where

- $N = |C|$ is the number of points in C
- $\overrightarrow{LS} = \sum_{i=1}^{N} \vec{X}_i$ is the linear sum of the N data points
- $SS = \sum_{i=1}^{N} \vec{X}_i^2 = \sum_{i=1}^{N} \langle X_i, X_i \rangle$ is the square sum of the N data points

LS stands for Linear Sum
SS stands for Square Sum

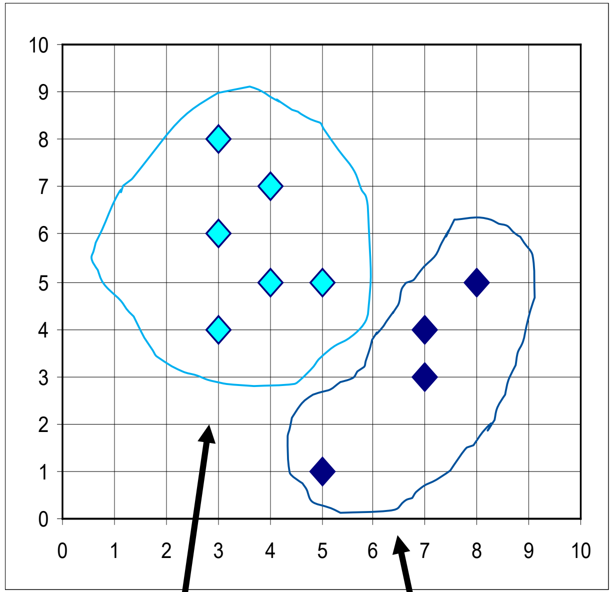
CF vector example

$$CF = (N, \overrightarrow{LS}, SS)$$

$$\overrightarrow{LS} = \sum_{i=1}^{N} \vec{X}_i$$

$$SS = \sum_{i=1}^{N} \vec{X}_i^2 = \sum_{i=1}^{N} \langle X_i, X_i \rangle$$

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



(5,1)
(7,3)
(7,4)
(8,5)

CF_1 = (6, (22,35),299)

CF_2 = (4, (27,13),238)

CF vector properties 1/4

$$CF = (N, \overrightarrow{LS}, SS)$$

$$\overrightarrow{LS} = \sum_{i=1}^{N} \vec{X}_i$$

$$SS = \sum_{i=1}^{N} \vec{X}_i^2 = \sum_{i=1}^{N} \langle X_i, X_i \rangle$$

- The CF vector is not only efficient, as it compresses the input dataset, but also accurate, as it is sufficient to compute several measures we need for clustering.

- the centroid of C:

$$\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X}_i}{N} \Longrightarrow \frac{\overrightarrow{LS}}{N}$$

- the radius of *C* (avg distance from cluster members to the centroid):

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X}_i - \vec{X0})^2}{N}\right)^{\frac{1}{2}} \Longrightarrow \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2}$$

Homework: Prove it!

- the diameter of C (avg pairwise distance within a cluster)

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X}_i - \vec{X}_j)^2}{N(N-1)}\right)^{\frac{1}{2}} \Longrightarrow$$

Homework: express it in terms of CF statistics!

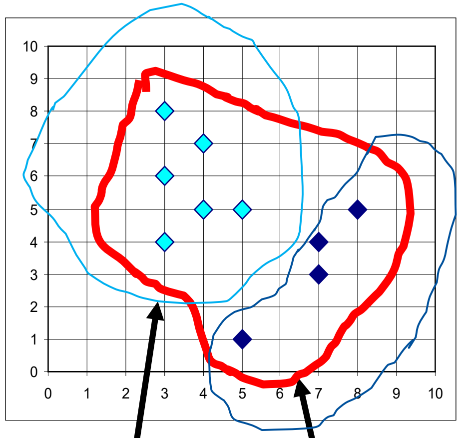
CF vector properties 2/4

- CF additivity property: Let two disjoint clusters C_1 und C_2. The CF vector of the cluster that is formed by merging the two disjoint clusters, is:

$$CF(C_1 \cup C_2) = CF\,(C_1) + CF\,(C_2) = (N_1 + N_2,\ LS_1 + LS_2,\ QS_1 + QS_2)$$

- *What is the CF of the marked (in red) cluster? How is it related to the CF1, CF2?*

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



(5,1)
(7,3)
(7,4)
(8,5)

$CF_1 = (6, (22,35),299)$

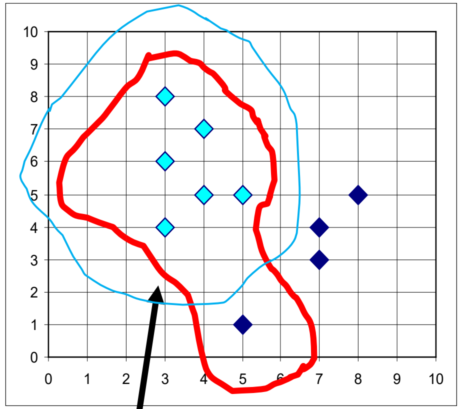
$CF_2 = (4, (27,13),238)$

CF vector properties 3/4

- CF incremental property: The updated CF of a cluster C_1 after the addition of a new point p, is:

$$CFT(C_1 \cup p) = CFT(C_1) + p$$

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



(5,1)
(7,3)
(7,4)
(8,5)

$CF_1 = (6, (22,35),299)$

- *What is the CFP of the marked (in red) set?*
- *How is it related to CF1?*

CF vector properties 4/4

- Cluster feature (CF)
  - A summary of the statistics of the points in a cluster *C*
  - Utilizes storage efficiently
  - Keeps sufficient statistics for clustering
  - Allows for easy merge of clusters, based on the additivity property
  - Allows for easy addition of new points, based on the incremental property

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

A (non-complete) taxonomy of stream clustering approaches
(& representative methods)

	Batch/Static clustering	Dynamic/Stream clustering
Partitioning methods	• k-Means • k-Medoids	• Leader • Simple single pass k-Means • STREAM k-Means [O'Callaghan et al 2002] • CluStream [Aggrawal et al 2003]
Density-based methods	• DBSCAN • OPTICS	• DenStream [Cao et al 2006] • incDBSCAN * • incOPTICS *
Grid-based methods	• STING	• DStream [Chen & Tu 2007]

(*) These methods require access to the raw data (this access might be limited though)

An example of an incremental clustering method - incDBSCAN

- Goal of incremental methods: To update the old clustering based on the new data (point *p*), without reclustering the data from scratch.
  - Access to raw data is possible but unnecessary access should be avoided
  - It refers not only to adding a new point, but also to the removal of existing points
- incDBSCAN[Ester et al, 1998] exploits the locality of information in density-based clustering and reorganizes the information only locally (as required)
- In our example:
  - Only the affected cluster is re-organized,

not everything is reclustered from scratch

  - Requires (limited) access to raw data

(to the affected highlighted cluster in our example)

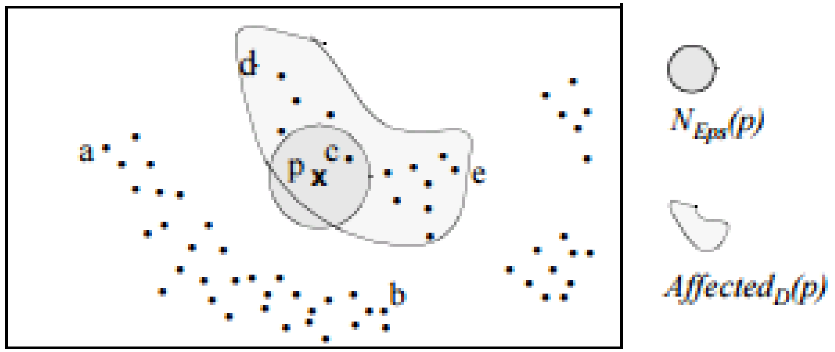


Figure 3: : Affected objects in a sample database

Incremental clustering methods vs stream clustering methods

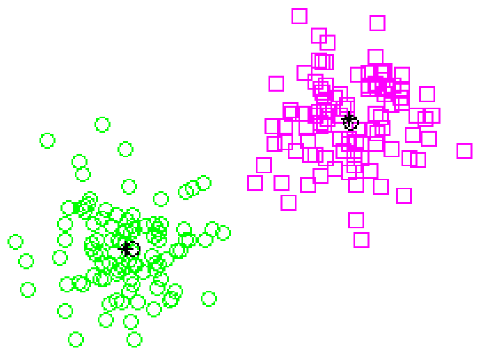
- Incremental methods require random access to the raw data to update the old clustering based on new instances.
  - They typically result in exact solutions
- Stream clustering methods do not assume random access to the data
  - The results are typically approximate
- Incremental methods might be appropriate for dynamic data arriving at a low rate
  - For potentially infinite streams, however they are not appropriate, and therefore new solutions are needed that can deal with the amount and complexity of the data

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

Partitioning methods

- Goal: Construct a partition of a set of objects into k clusters so that some clustering criterion is optimized
  - e.g. *k*-Means, *k*-Medoids



- Two types of methods:
  - Adaptive methods:
    - Leader (Spath 1980)
    - Simple single pass k-Means (Farnstrom et al, 2000)
    - STREAM k-Means [OCaEtAl02]
  - Online summarization - offline clustering methods:
    - CluStream [Aggrwal et al, 2003]

Adaptive methods: Leader (Spath 1980)

- The simplest single-pass partitioning algorithm
- Whenever a new instance p arrives from the stream
  - Find its closest cluster (leader), c_{clos}
  - Assign p to c_{clos} if their distance is below the threshold d_{thresh}
  - Otherwise, create a new cluster (leader) with p

A cluster is "defined" by its first point

- Discussion

\+ 1-pass and fast algorithm

\+ No prior information on the number of clusters

– The number of clusters is not controllable

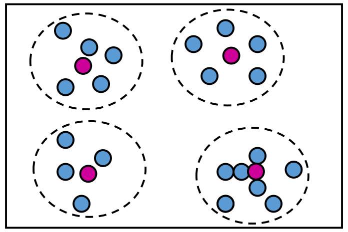
– It depends on the order of the examples

– It depends on a correct guess of d_{thresh}

Adaptive methods: STREAM k-Means [O'Callaghan et al 2002]

- An extension of k-Means for streams
  - The iterative process of static *k*-Means cannot be applied to streams
  - Idea: Use a buffer that fits in memory and apply *k*-Means locally in the buffer

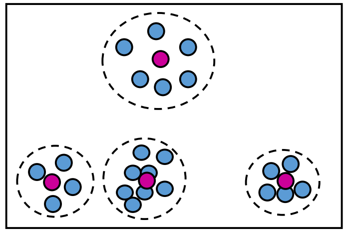
- Stream is processed in chunks $X_1, X_2 \ldots X_i \ldots$, each fitting in memory
  - For the current chunk X_i
    - Apply *k*-Means locally on X_i (retain only the *k* cluster centers from X_i)
    - X': the cluster centers seen thus far over the stream (# $i*k$ centers)
      - Each center is treated as a point, weighted with the number of points it compresses
    - Apply *k*-Means on X' to obtain the current clustering result



$\boldsymbol{X_1}$

…

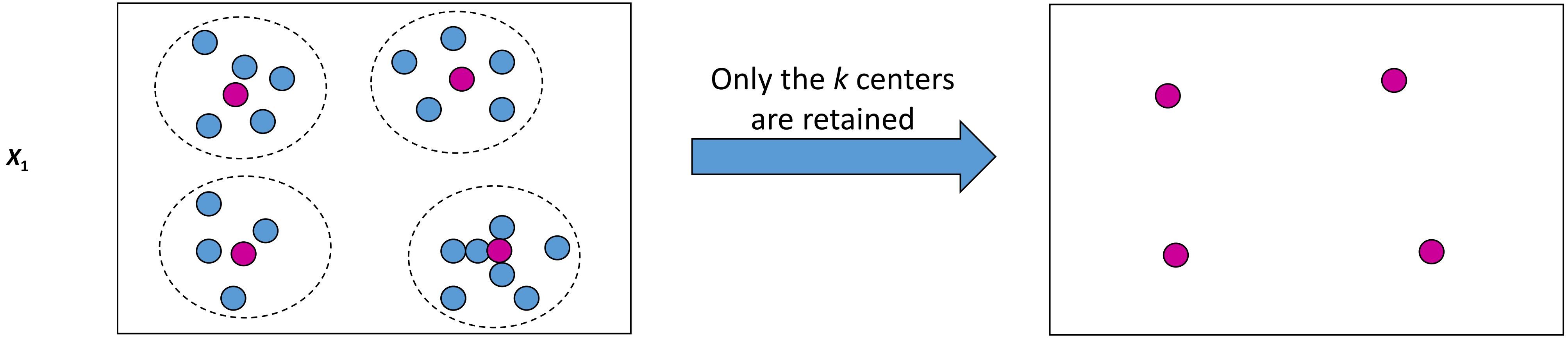
…



$\boldsymbol{X_i}$

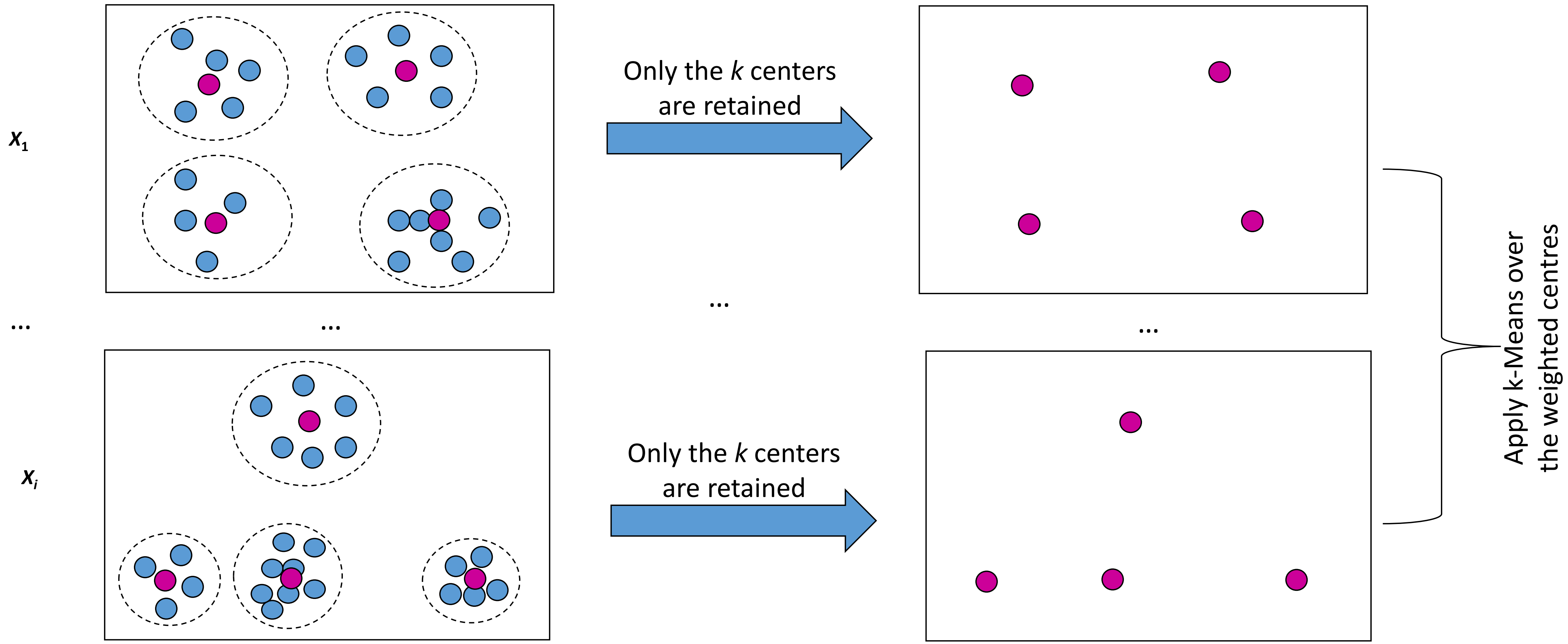
Adaptive methods: STREAM k-Means [O'Callaghan et al 2002]

- For each batch



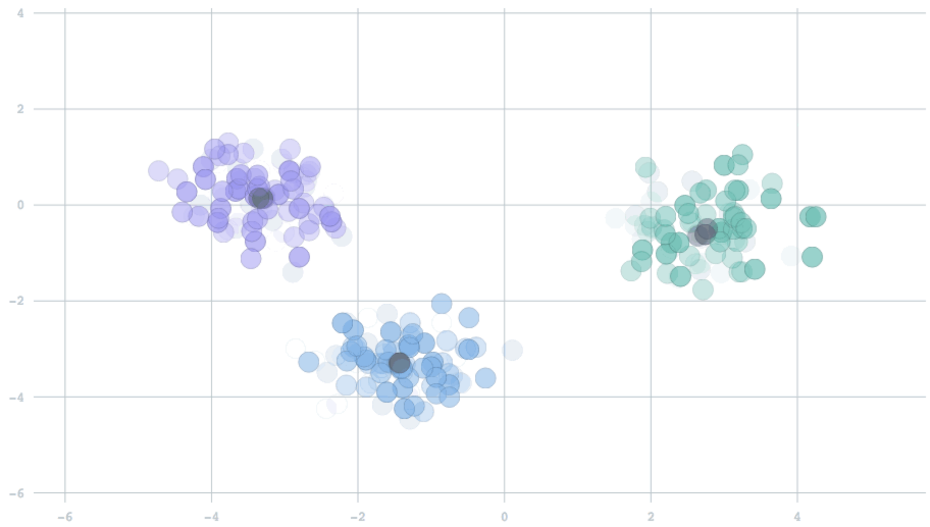
Adaptive methods: STREAM k-Means [O'Callaghan et al 2002]

- For each batch



Adaptive methods: STREAM k-Means [O'Callaghan et al 2002]

- An [example of Stream k-Means in SPARK]



- Discussion
  - \+ The number of clusters is controllable (*k* on each batch)
  - \+ Good results on each batch (via typical *k*-Means)
  - – The number of clusters is fixed over the stream (*k)*
  - – Only one clustering model is reported at each time point

Online-Offline methods: CluStream [Aggrawal et al, 2003]

- The stream clustering process is separated into two components:
  - an online micro-cluster component, that summarizes the stream locally as new data arrive over time
    - Micro-clusters are stored in disk at snapshots in time that follow a pyramidal time frame.
  - an offline macro-cluster component, that clusters these summaries into global clusters
    - Clustering is performed upon summaries instead of raw data

CluStream: the micro-cluster summary structure

- The microcluster summaries are extensions of the cluster feature vector (CF) summary of BIRCH
- The micro-cluster summary for a set of *d*-dimensional points (X_1, X_2, …, X_n) arriving at time points T_1, T_2, …, T_n is defined as:

$$CFT = (\underbrace{SS^x}_{\sum^{N}_{i=1} X_i^2}\ ,\ \underbrace{LS^x}_{\sum^{N}_{i=1} X_i} \boxed{,\ \underbrace{SS^t}_{\sum^{N}_{i=1} T_i^2},\ \underbrace{LS^t}_{\sum^{N}_{i=1} T_i},}\ n)$$

LS stands for Linear Sum
SS stands for Square Sum

CluStream: the micro-cluster summary structure

- The micro-cluster summary for a set of d-dimensional points (X_1, X_2, …, X_n) arriving at time points T_1, T_2, …, T_n is defined as:

CFT = (SS^x , LS^x , SS^t, LS^t, n)

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

LS stands for Linear Sum
SS stands for Square Sum

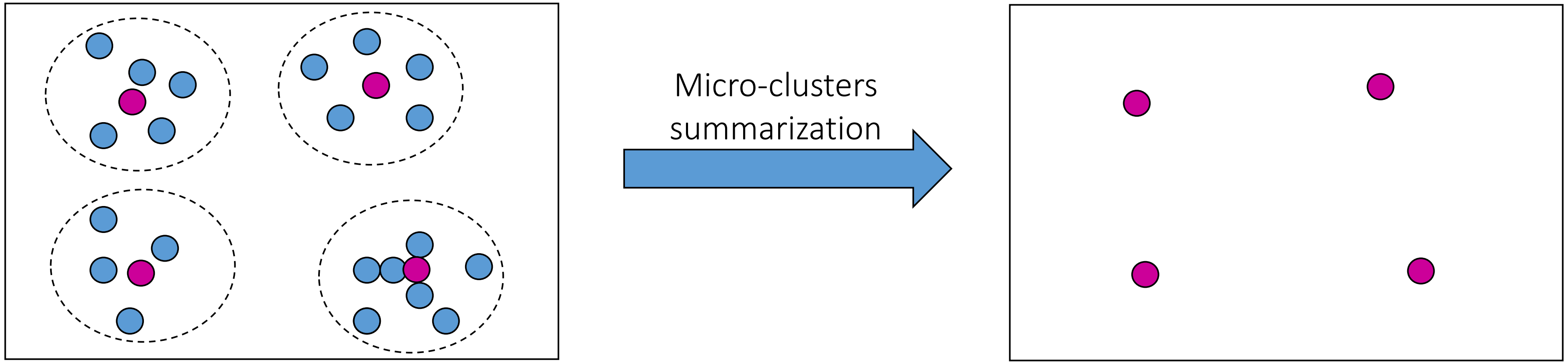
- Using the summaries, we can easily calculate basic measures to characterize a cluster:
  - Center: $$\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X_i}}{N} \Longrightarrow \frac{\vec{LS}}{N}$$
  - Radius: $$R = \left(\frac{\sum_{i=1}^{N} (\vec{X_i} - \vec{X0})^2}{N}\right)^{\frac{1}{2}} \Longrightarrow \sqrt{\frac{SS}{n} - \left(\frac{LS}{n}\right)^2}$$

….

- Similarly information on the cluster recency can be derived, e.g., avg cluster timestamp

CluStream: the micro-cluster summary structure

- In other words, we summarize the stream via micro-clusters
  - Each microcluster is represented through its CFT summary



CluStream: the micro-cluster summary structure

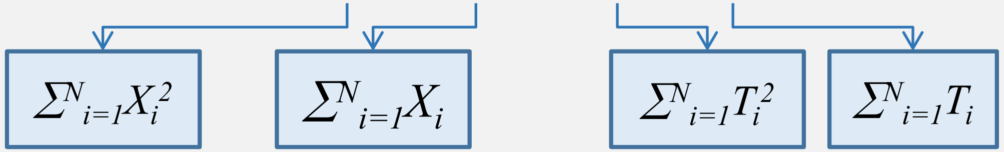
- Micro-clusters have very appealing properties for streams
  - Incrementality: $CFT(C_1 \cup p) = CFT(C_1) + p$

    ➔ we can easily add new points to a microcluster
  - Additivity: $CFT(C_1 \cup C_2) = CFT(C_1) + CFT(C_2)$

    ➔ we can easily merge two microclusters
  - Subtractivity: $CFT(C_1 - C_2) = CFT(C_1) - CFT(C_2)$, $C_1 \supseteq C_2$

    ➔ we can remove the effect of an old microcluster
- Recall the 2-directional learning in streams
  - Incrementality helps us to incorporate new information in the clustering model
  - Subtractivity helps us to remove outdated information from the clustering model

$CFT = (CF2^x, CF1^x, CF2^t, CF1^t, n)$



CluStream algorithm: overview

- Input:
  - The stream
  - *q:* #micro-clusters to be maintained over time (fixed)
  - *t*: radius factor

- 4 steps
  - Initialization: How we build the initial set of microclusters?
  - Online micro-cluster maintenance: How do we add new points from the stream?
  - Periodic storage: Decide when to store snapshots of micro-clusters on disk?
  - Offline macro-clustering: How to derive the final clusters?

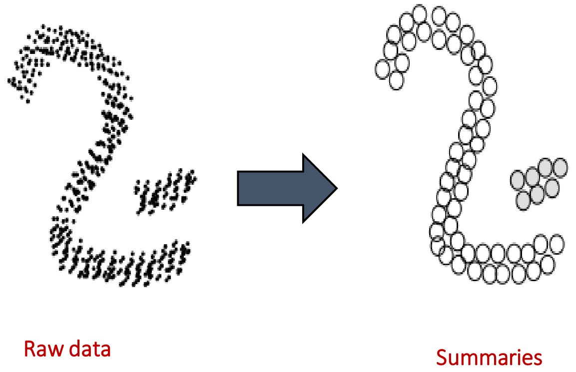
CluStream: Initialization step

- Initialization: How we build the initial set of microclusters?
  - Done using an offline process in the beginning of the stream
  - Wait for the first *InitNumber* points to arrive
  - Apply a standard *k*-Means algorithm with k=q to create q clusters
  - For each discovered cluster, assign it a unique ID and create its micro-cluster summary.

$$CFT = (CF2^x, CF1^x, CF2^t, CF1^t, n)$$

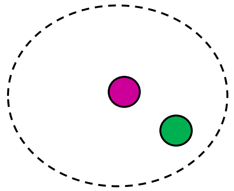
$$\Sigma^N_{i=1} X_i^2 \qquad \Sigma^N_{i=1} X_i \qquad \Sigma^N_{i=1} T_i^2 \qquad \Sigma^N_{i=1} T_i$$

- How should we set q?
  - much larger than the natural number of clusters
  - much smaller than the total number of points arrived

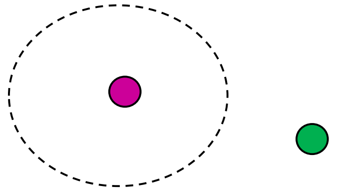


CluStream: Online step - How do we add new points from the stream?

- A fixed number of q micro-clusters is maintained over time
- Whenever a new point p arrives from the stream
  - Compute distance between p and each of the q maintained micro-cluster centroids
  - *clu* ← the closest micro-cluster to p
  - Find the max boundary of *clu*
    - It is defined as a factor of t of *clu* radius
  - If p falls within the maximum boundary of *clu*
    - p is absorbed by *clu*
    - Update *clu* statistics (incremental property of microclusters)
  - Else, create a new micro-cluster with p, assign it a new ID, initialize its statistics
    - To keep the total number of micro-clusters fixed (i.e., q):
      - Delete the most obsolete micro-cluster or
        - If it is safe based on its time statistics
      - Merge the two closest ones (Additivity property of microclusters)
        - When two micro-clusters are merged, a list of ids is created. This way, we can identify the component micro-clusters that comprise a micro-cluster.



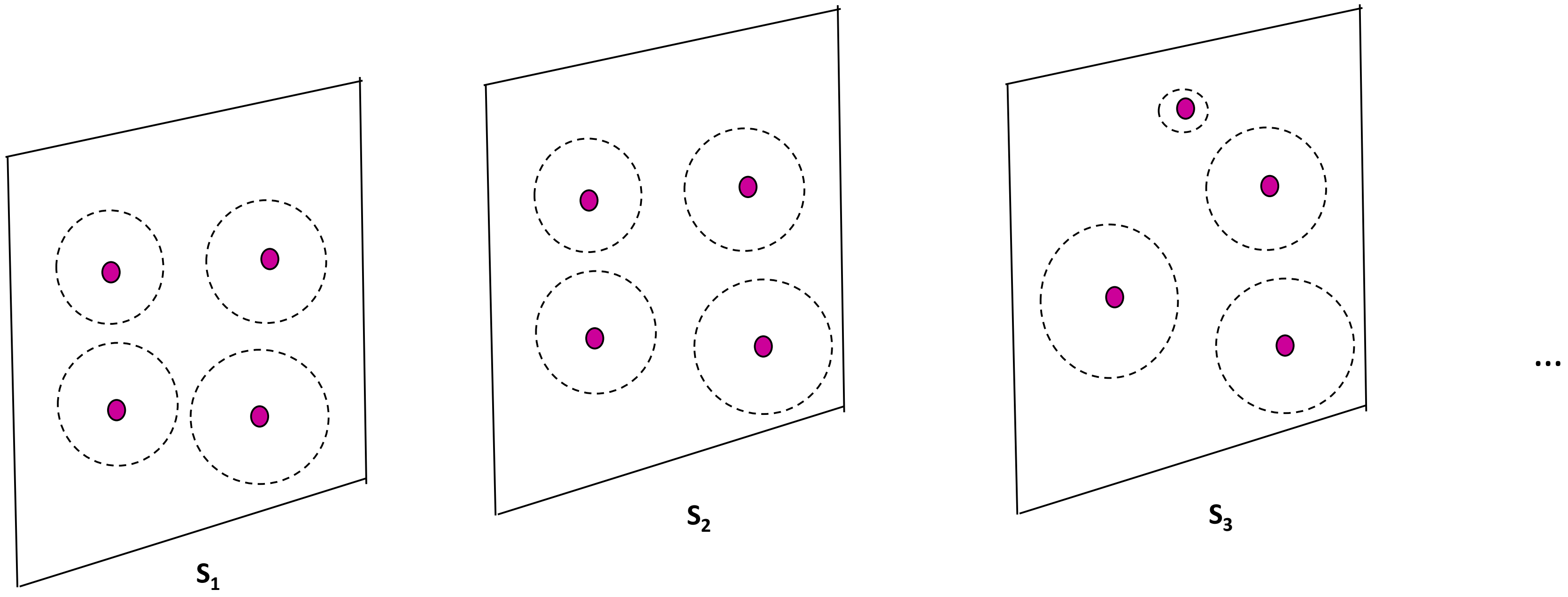
The green point can be absorbed by the summary.



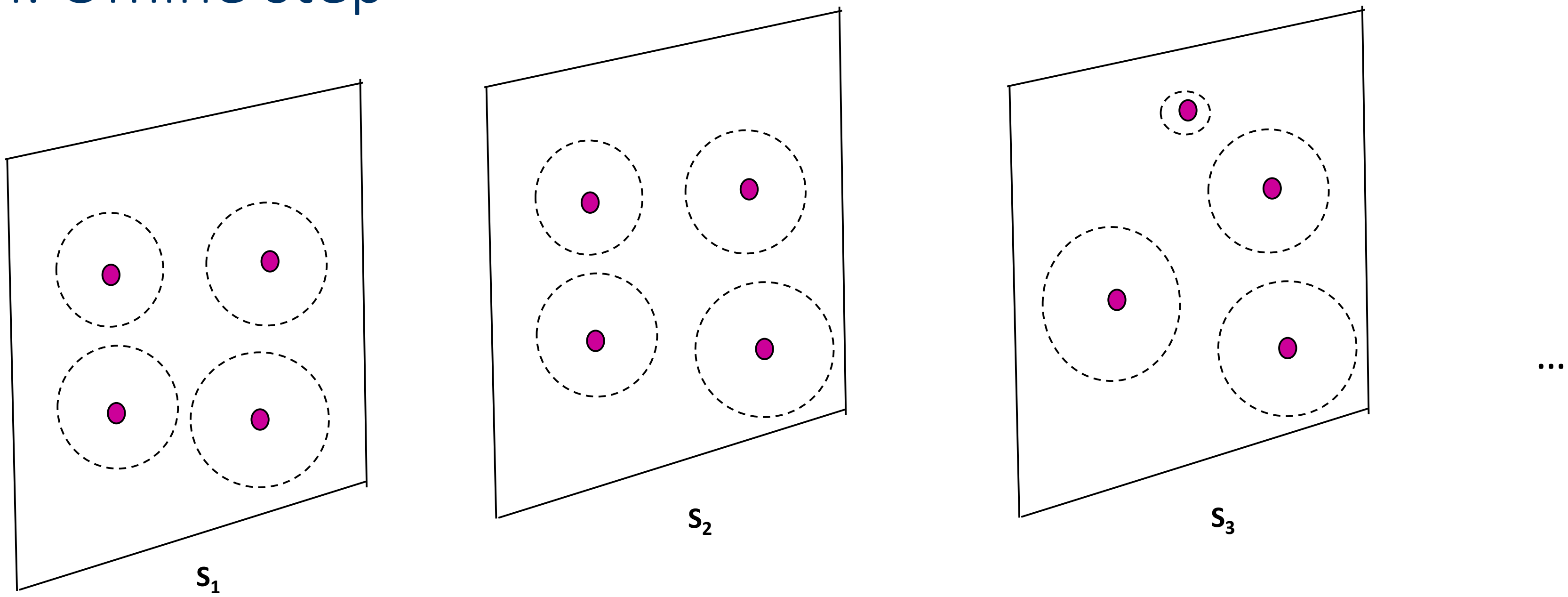
The green point cannot be absorbed by the summary.

CluStream: Periodic micro-cluster storage

- Micro-clusters are stored as snapshots in time following the pyramidal pattern framework



CluStream: Offline step

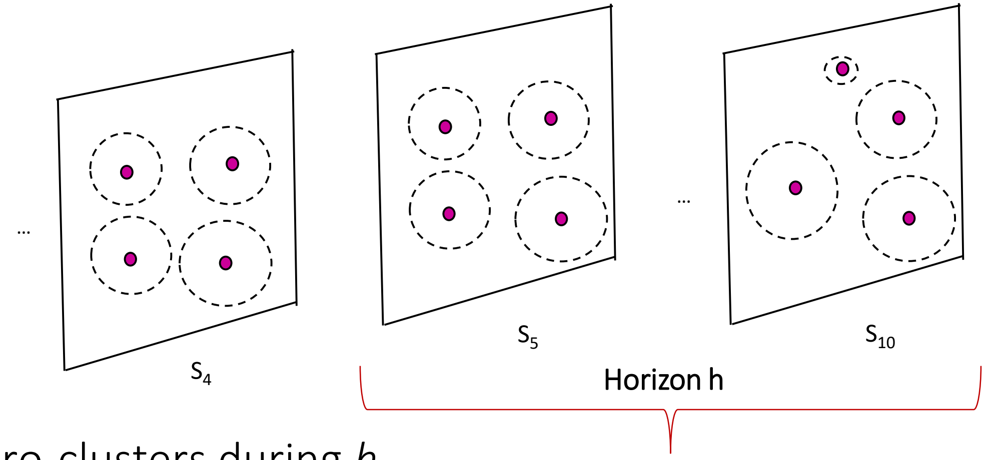


- The offline step is applied on demand. The user specifies the time horizon h for clustering, e.g., h=10, so from S_T-S_{T-10} where T is the current timepoint.
  - Different clusterings are possible, if the horizon of clustering changes→ allows the user can explore the history of the stream
- User input: time horizon h, # macro-clusters k to be detected, current time T
- Output: the clusters in (T-h, T)

2 steps

- Step 1: Find the active micro-clusters during h
- Step 2: Apply k-Means over the active micro-clusters in h to derive the k macro-clusters

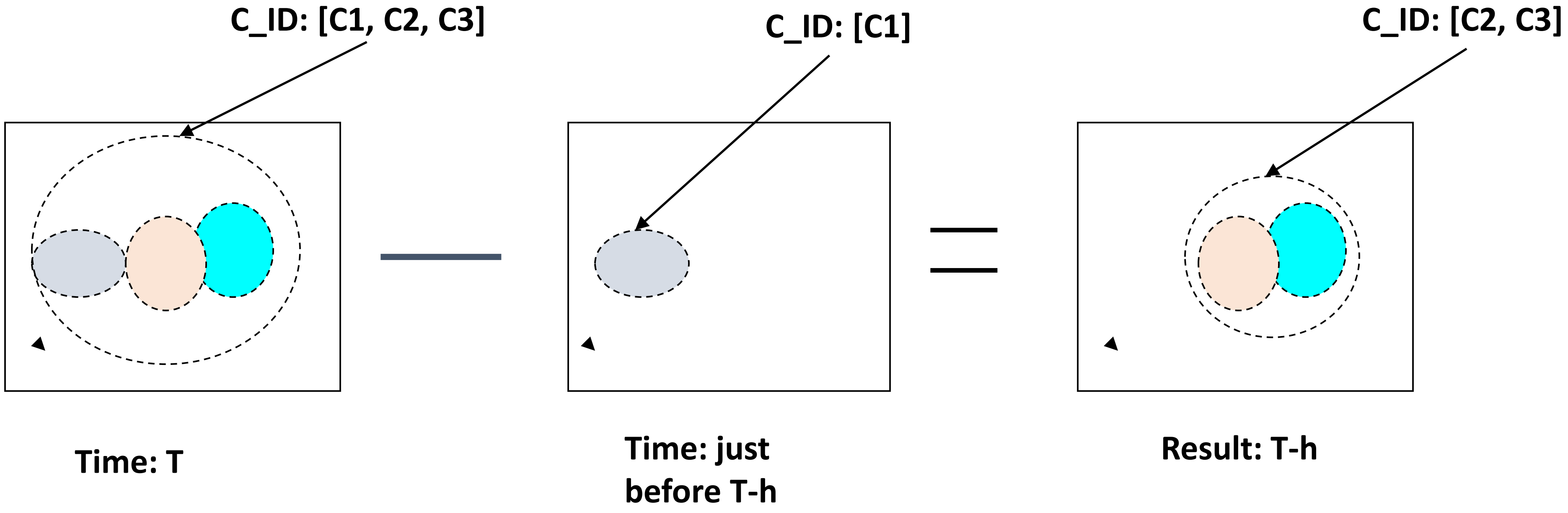
CluStream: Offline step – step 1



- **Step 1**: Find the active micro-clusters during horizon *h*:
- We exploit the subtractivity property to find the active micro-clusters during *h*
- Suppose current time is *T*. Let S(*T*) be the set of micro-clusters at *T*.
- Find the stored snapshot which occurs just before time *T-h.* Let S(*T*–h') be the set of micro-clusters.
- For each micro-cluster in the current set S(*T*), we find the list of its component micro-cluster ids. For each of the list of ids, find the corresponding micro-clusters in S(*T*–h').
- Subtract the CF vectors for the corresponding micro-clusters in S(*T*–h') (subtractivity property of microclusters)
- This ensures that the micro-clusters created before the user-specified horizon do not dominate the result of clustering process

CluStream: Offline step – step 1 - example

- **Example**: if we have a merged cluster with id list (C1,C2,C3) in S(T) and a cluster with ID C1 in S(T-h'), then we should remove C1 as it was created before the user horizon.
  - To this end, we can use the subtractivity property: CFT(C1,C2,C3)-CFT(C1) = CFT(C2,C3)
  - The result is the active IDs during h



CluStream: Offline step – step 2

- **Step 2:** Apply k-Means over the active micro-clusters in *h* to derive the *k* macro-clusters
  - Initialization: centers are not picked up randomly, rather sampled with probability proportional to the number of points in a given micro-cluster
  - Distance is the centroid distance
  - New centers are defined as the weighted centroids of the micro-clusters in that micro-cluster partition

CluStream algorithm: overview

Input: The stream,
 q: #micro-clusters to be maintained over time (fixed)
 t: radius factor

- Initialize: How we build the initial set of microclusters
  - Apply k-Means, with $k=q$, over *initPoints*. Built a summary for each cluster.
- Online micro-cluster maintenance: How do we add new points from the stream?
  - Does a new point fit into some existing summary?
    - If yes, just update the summary
    - If no, create a new summary. But because of fixed q, you have to reduce #microclusters by one
- Periodic storage of micro-clusters snapshots into disk
  - At different levels of granularity depending upon their recency
- Offline (on demand) macro-clustering
  - Input: A user defined time horizon h and number of macro-clusters k to be detected
  - Locate the valid micro-clusters during h
  - Apply k-Means upon these micro-clusters → k macro-clusters

CluStream: discussion

- \+ One pass over the raw data
- \+ Views the stream as a changing process over time, rather than clustering the whole stream
- \+ Provides flexibility to an analyst in a real-time and changing environment
- \+ Can characterize clusters over different time horizons in a changing environment

- – Fixed number of micro-clusters maintained over time
- – Sensitive to outliers/ noise
  - – We might delete a valid microcluster just because of an outlier point

Homework (could also be a potential example topic)

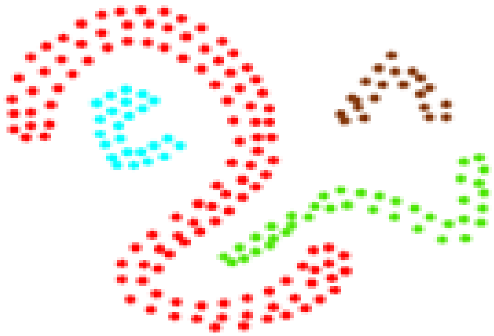
- **Data stream clustering: ageing/window model**
  - We discussed different forms of ageing in a stream environment: landmark windows, sliding windows, damped window models, …
  - Questions:
    - i) Please explain the ageing schema in the CluStream algorithm.
    - ii) Can it be categorized as sliding or landmark or damped window model?
  - Answer:
    - Landmark: Yes/No. Why?
    - Sliding: Yes/No. Why?
    - Damped: Yes/No. Why?

Outline

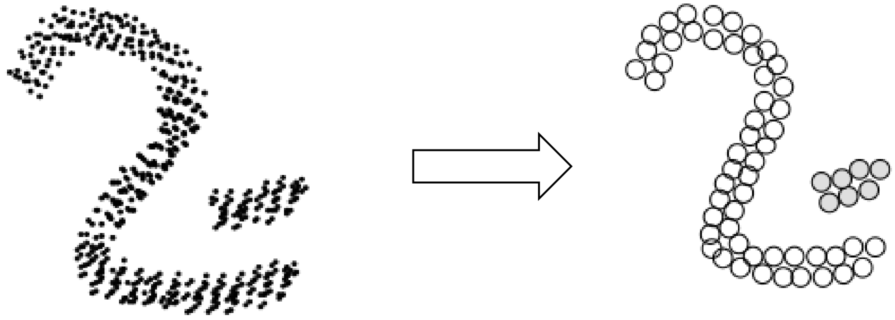
- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

Density-based methods

- Clusters as regions of high density surrounded by regions of low density (noise)
  - Density is measured locally, in the ε-neighborhood of each point
    - e.g. DBSCAN, OPTICS



- Very appealing for streams
  - No assumption on the number of clusters
  - Discovering clusters of arbitrary shapes
  - Ability to handle outliers and noise
- But, they miss a clustering model (or it is to complicated)
  - Clusters are represented by all their points!!!!
- Solution: Describe clusters as set of summaries
  - DenStream [Cao et al 2006]

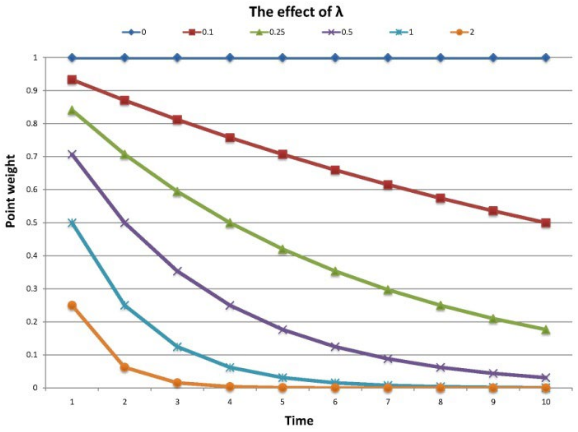


DenStream [Cao et al 2006]

- The online-offline rationale is followed:
  - Online summarization as new data arrive over time
    - They distinguish between different types of summaries: Core, potential core and outlier micro–clusters
  - Offline clustering over the summaries to derive the final clusters
    - A modified version of DBSCAN over the summaries
- Data are subject to ageing according to the exponential ageing function (damped window model) – recall previous lectures on streams

$$f(o,t) = e^{-\lambda(t-t_o)}$$

  - t-t_o is the time from point occurrence
  - λ (λ>0) is the decay rate which determines the importance of historical data
  - The higher the value of λ, the lower the importance of old data



DenStream: summarizing the stream

Note that the microcluster is now a temporal object

- The micro-cluster summary at time *t* for a set of d-dimensional points (p_1, p_2, …, p_n) arriving at time points T_1, T_2, …, T_n is:

$$MC = (CF^1,\ CF^2,\ w)$$

$$CF^1 = \sum_{i=1}^{n} f(t-T_i)p_i \qquad CF^2 = \sum_{i=1}^{n} f(t-T_i)p_i^2 \qquad w = \sum_{i=1}^{n} f(t-T_i)$$

- Easy computation of basic measures, e.g., (see also discussion on micro-cluster summaries):
  - Center: $c = \frac{CF^1}{w}$
  - Radius: $r = \sqrt{\frac{CF^2}{w} - \left(\frac{CF^1}{w}\right)^2}$

- A micro-cluster summary c_p can be maintained incrementally
  - If a new point p is added to c_p:

$$c_p = (CF^1+p,\ CF^2+p^2,\ w+1)$$

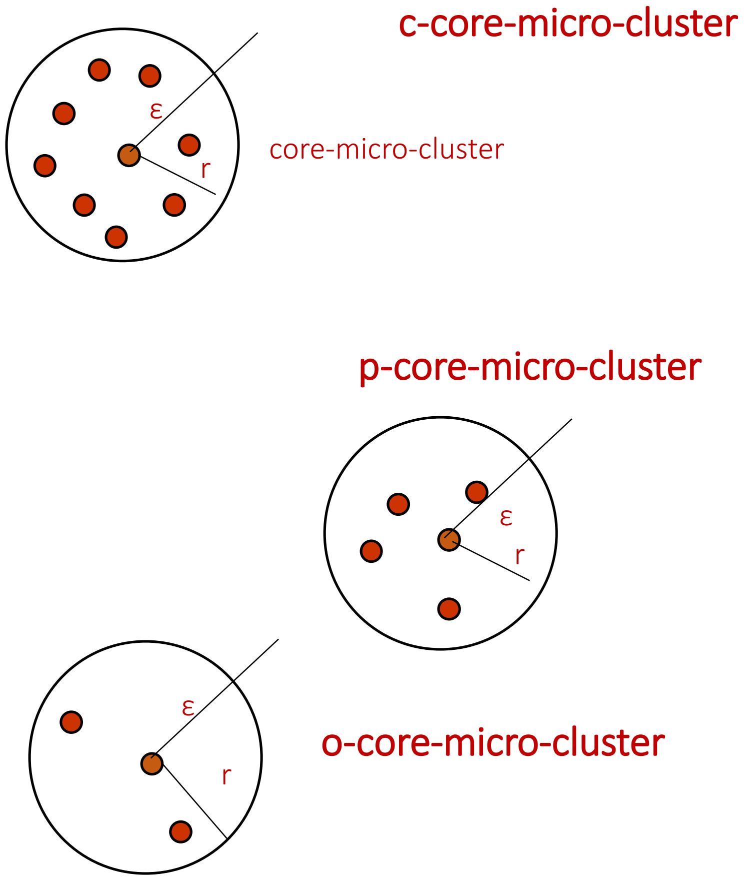
  - If no point is added to c_p for time interval δt, decay should be applied:

$$c_p = (2^{-\lambda\delta t}*CF^1,\ 2^{-\lambda\delta t}*CF^2, 2^{-\lambda\delta t}*w)$$

DenStream: core, potential core & outlier summaries

User-defined parameters:
μ: the density threshold for core-microclusters (similar to DBSCAN)
ε: the radius threshold (similar to DBSCAN)
β: β*μ the density threshold for potential-core mictoclusters

- Core (or dense) micro-clusters
  - $(w \geq \mu)$ & $(r \leq \varepsilon)$
- But, in an evolving stream, the role of clusters and outliers often interchange:
  - Should provide opportunity for the gradual growth of new clusters
  - Should promptly get rid of the outliers
- Potential core micro-clusters
  - $(w \geq \beta*\mu)$ & $(r \leq \varepsilon)$, $0 < \beta \leq 1$
- Outlier micro-clusters
  - $(w < \beta*\mu)$ & $(r \leq \varepsilon)$, $0 < \beta \leq 1$

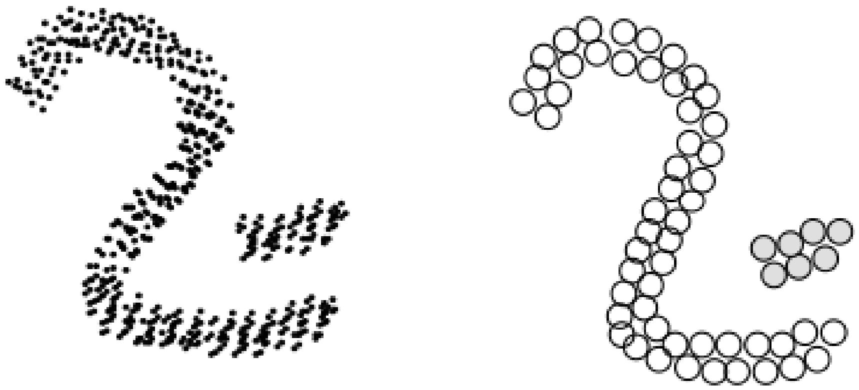


DenStream: the algorithm

- The algorithm consists of 4 main components:
  - Initialize: apply DBSCAN over initPoints → the core points are the p-micro-clusters
  - Online step: Maintain the micro-clusters as new points arrive from the stream
    - 2 lists of p-micro-clusters and o-micro-clusters are maintained over time
  - Periodic micro-cluster maintenance due to data ageing
  - Offline macro-clustering: upon user request, extract the final clusters

DenStream: online step

- 2 lists of p-micro-clusters and o-micro-clusters are maintained over time
- When a new point d arrives
  - Find its closest p-micro-cluster pclu
    - If the updated radius of pclu $\leq \varepsilon$, merge d to pclu
  - otherwise find its closest o-micro-cluster oclu
    - If the updated radius of oclu $\leq \varepsilon$, merge d to oclu
    - Check if oclu can be upgraded to a p-micro-cluster (if now $w \geq \beta * \mu$)
  - o.w., create a new o-micro-cluster with d (keep also the creation time t_o for the microcluster)



DenStream: periodic microcluster maintenance 1/2

- DenStream maintains 2 separate memories: i) for p-micro-clusters and ii) o-micro-clusters
- How to ensure that the memory is bounded?
  - General principle: delete outdated information
- For p-micro-clusters memory, the idea is to delete p-micro-clusters that turn into o-micro-clusters.
  - This means, checking the weight of each p-micro-cluster and delete those with weight < β*μ
- The question is how often should we check the weight (so, efficiency).
  - Recall that β*μ is the minimum weight of a p-micro-cluster.
  - Therefore the minimum time for a p-micro-cluster to fade into an o-micro-cluster is given by:

$$2^{-\lambda T_p} * \beta * \mu = \beta * \mu - 1 \quad \Rightarrow \quad T_p = \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu - 1}) \rceil,$$

- So, we need to check for deletion of p-micro-clusters every T_p time periods
- This checking strategy also ensures that the p-microclusters memory is bounded by *W/ β*μ* where the constant W is the overall weight of the datastream (proof omitted, see paper)

DenStream: periodic microcluster maintenance 2/2

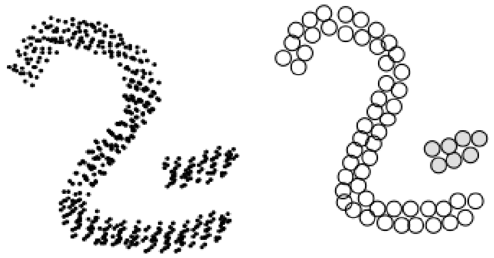
- DenStream maintains two separate memories: i) for p-micro-clusters and ii) o-micro-clusters
- How to ensure that the memory is bounded? General principle: delete outdated information
- For o-micro-clusters memory, the problem is that their number might continuously grow. We need to keep them as an o-micro-cluster might be the beginning of a p-micro-cluster. But we cannot keep them forever.
- So the idea is to delete those o-micro-clusters that are not promising anymore.
- How promising a microcluster is depends on its current weight *w* vs its expected weight ξ:

$$\xi(t_c, t_o) = \frac{2^{-\lambda(t_c - t_o + T_p)} - 1}{2^{-\lambda T_p} - 1}$$

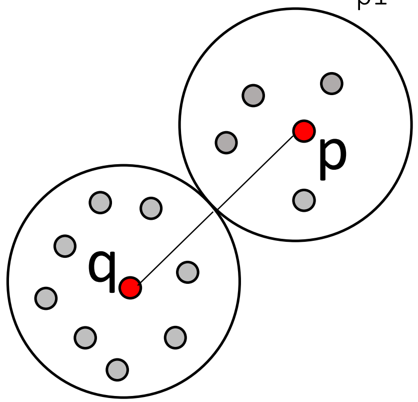
  - Intuitively, the longer a microcluster exists (larger $(t_c - t_o)$), the higher its weight is expected to be.
- If $w < \xi$, the o-micro-cluster might not grow into a p-micro-cluster and can be safely deleted.
- The authors prove that the number of o-micro-clusters is bounded (proof omitted, see paper)

DenStream: offline step

- Upon request, apply a variant of DBSCAN over the set of online maintained p-micro-clusters
  - Each p-micro-cluster c_p is treated as a virtual point located at the center of c_p with weight w.



- Core-micro-clusters (redefined)
- Directly density reachable (redefined)
  - c_p is directly density reachable from c_q if:
    - c_q is a c-micro-cluster and
    - $dist(c_p,c_q) \leq 2\varepsilon$ (i.e. they are tangent or intersecting)
- Density reachable (redefined)
  - A p-micro-cluster c_p is density reachable from a c-micro-cluster c_q if there is a chain of c-micro-clusters $c_{p1}=c_q$, cp_2, …, $c_{pn}=c_p$.
- Density connected (redefined)
- A cluster is a maximum set of density connected points



DenStream: overview

+ DenStream clusters large evolving data streams
+ Discover clusters of arbitrary shapes, following the density-based paradigm
+ No assumption on the number of clusters
+ Noise/ outlier handling

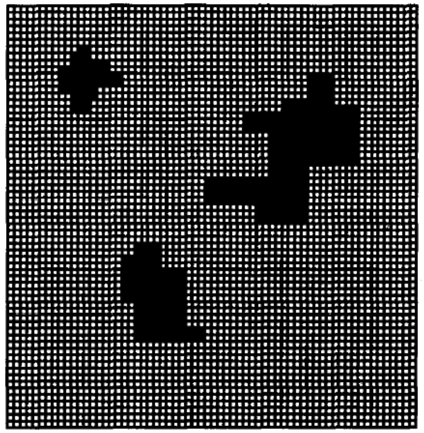
– The choice of the parameters ε, β, μ
– Constant parameters over time, what about clusters with different density

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

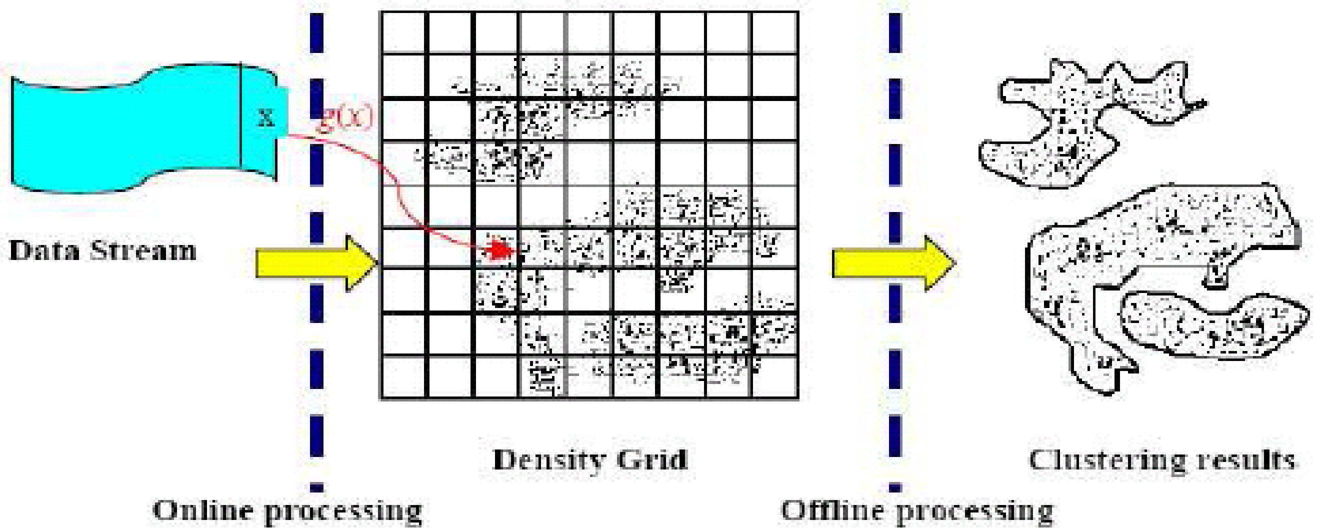
Grid based methods

- A grid structure is used to capture the density of the dataset.
  - A cluster is a set of connected dense cells
  - e.g. STING
- Appealing features for streams
  - No assumption on the number of clusters
  - Discovering clusters of arbitrary shapes
  - Ability to handle outliers
- In case of streams
  - The grid cells “constitute” the summary structure
  - Update the grid structure as the stream proceeds
  - DStream [Chen & Tu 2007]



DStream [Chen & Tu 2007]

- Resembles the online-offline rationale of CluStream/DenStream but there is no real offline part, rather a final clustering structure is maintained online.
  - Online mapping of the new data into the grid (so summarization)
  - Periodic final clustering maintenance



DStream: Summarizing the stream into the grid

Note that the cell-summary is a temporal object

- Data ageing (damped window model):
  - $D(x,t) = \lambda^{t-t_c}$, t_c is the arrival time for point x, t is the current timepoint
  - λ in (0,1) is the *decay factor*
- The density of a grid cell *g* at time *t*: $$D(g,t) = \sum_{x \in E(g,t)} D(x,t)$$
- The characteristic vector of a grid cell g is defined as:

$(t_g, t_m, D, \text{label}, \text{status})$

Last update time

last time g was removed from grid_list

Last density update

the class label of the grid

{sporadic, normal}

- The grid density can be updated incrementally

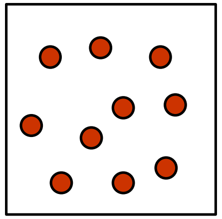
$$D(g,t_n) = \lambda^{t_n - t_l} D(g,t_l) + 1$$

t_n: the new record arrival time; t_l: the last record arrival

DStream: Dense, Sparse and Transitional grid cells

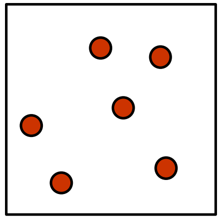
- The density of a grid is constantly changing over time.
- Dense grid cells

$$D(g,t) \geq \frac{C_m}{N(1-\lambda)} = D_m \qquad C_m > 1$$



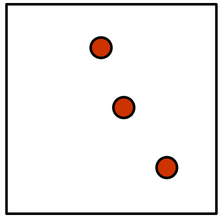
- Transitional grid cells

$$D(g,t) \leq \frac{C_l}{N(1-\lambda)} = D_l \qquad 0 < C_l < 1$$



- Sparse grid cells

$$\frac{C_l}{N(1-\lambda)} \leq D(g,t) \leq \frac{C_m}{N(1-\lambda)}$$



DStream: the algorithm

```
procedure D-Stream
    t_c = 0;
    initialize an empty hash table grid_list;
    while data stream is active do
        read record x = (x_1, x_2, ··· , x_d);
        determine the density grid g that contains x;
        if(g not in grid_list) insert g to grid_list;
        update the characteristic vector of g;
        if t_c == gap then
            call initial_clustering(grid_list);
        end if
        if t_c mod gap == 0 then
            detect and remove sporadic grids from grid_list;
            call adjust_clustering(grid_list);
        end if
        t_c = t_c + 1;
    end while
end_procedure
```

Grid update

Initialization

Clustering

DStream: overview

+ DStream clusters large evolving data stream
+ It can discover clusters of arbitrary shapes
+ No assumption on the number of clusters
+ Distinguishes noise and outliers
+ The grid provides a level of abstraction over the data

– The choice of the grid parameters
– Fixed grid parameters

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

Stream clustering evaluation 1/3

- Similar to what we discussed for batch clustering, but in streams we are interested also in the overtime performance monitoring of the clustering
- Two clustering quality categories (similarly to batch evaluation, see corresponding lectures)
  - Internal measures of similarity, e.g., SSE measuring the goodness of a clustering structure
  - External measures of similarity, e.g., entropy, measuring the extent to which cluster labels match externally supplied class labels.
- The difference is that those measures are evaluated over a future horizon

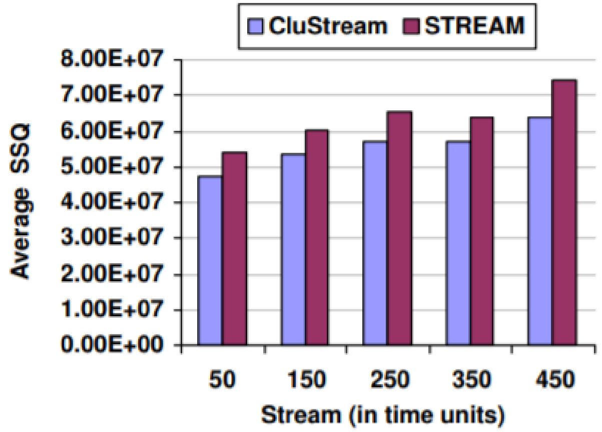


Figure 4: Quality comparison (Charitable Donation dataset, horizon=16, stream_speed=200)

Stream clustering evaluation 2/3

- Except for the quality, other important aspects
  - Time (how fast the points are processed)

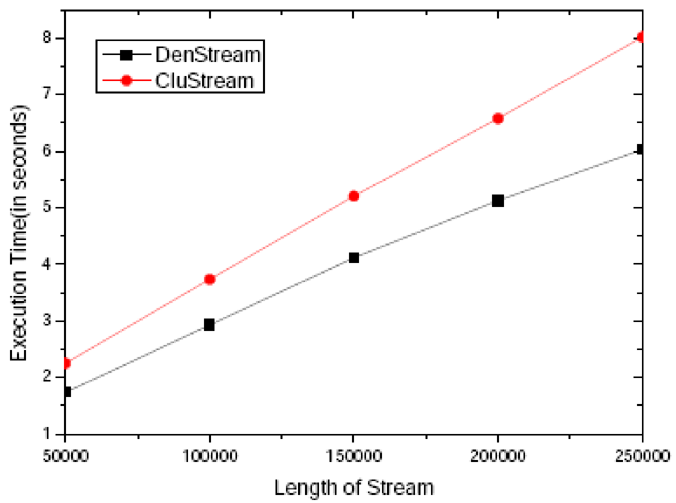


Figure 14: Execution time vs. length of stream(Network Intrusion data set)

Stream clustering evaluation 3/3

- Except for the quality, other important aspects
  - Memory
    - For example, for methods that not assume a fixed number of summaries, plotting this number over time is very informative about the underlying population distribution

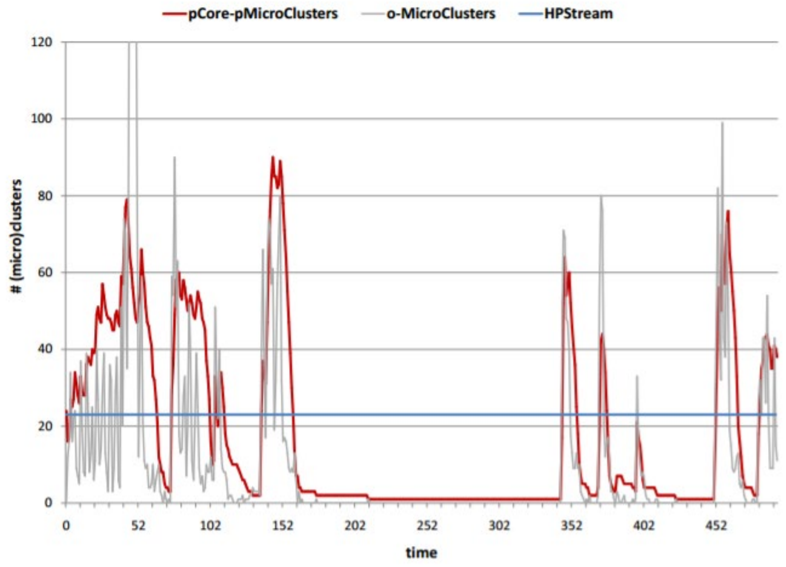


Figure 5: Number of (micro)clusters (Network Intrusion dataset, window size w=1000)

Outline

- Data stream clustering basics
- Summarization
- Overview of stream clustering methods
- Partitioning methods
- Density-based methods
- Grid-based methods
- Data stream clustering: evaluation aspects
- Things you should know from this lecture & reading material

Stream clustering overview

- A very important task given the availability of streams nowadays
- Stream clustering algorithm maintain a valid clustering of the evolving stream population over time
- Two generic approaches
  - Online maintenance of a final clustering model
  - Online summarization of the stream and offline clustering
    - Summaries!
- Different window models
- Handling outliers (or, potential future clusters) is very important
- Specialized approaches for text streams, high-dimensional streams.
- Evaluation is not straightforward

Hands on experience

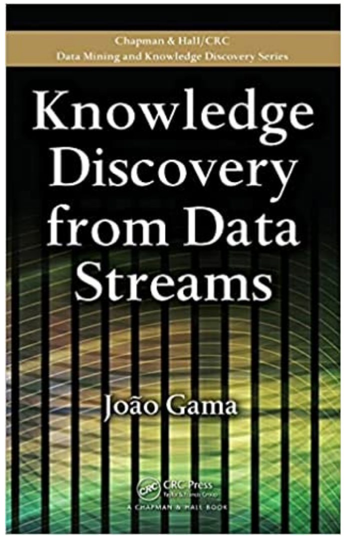
- Familiarize yourself with popular frameworks for stream learning
  - MOA: the most popular open source framework for data stream mining https://moa.cms.waikato.ac.nz/ (in Java)
    - "Machine Learning for Data Streams with Practical Examples in MOA" book
  - Scikit-multiflow "A machine learning package for streaming data in Python"
    - https://scikit-multiflow.github.io/
-

Related Open Source Software

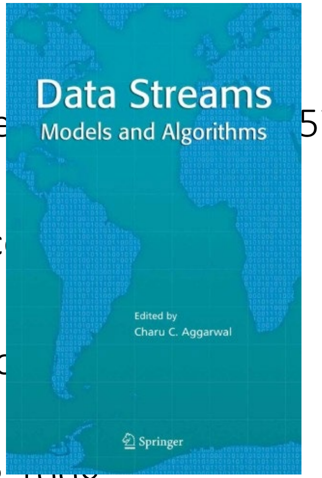
- RIVER, a new framework for stream mining in Python.
- streamDM for Spark Streaming, a new framework for Spark.
- Apache SAMOA , a new framework for distributed stream mining, can be easily used with Apache Flink, Apache Storm, S4, or Samza.
- streamDM C++ , a framework in C++ for data stream mining.
- ADAMS, a novel, flexible workflow engine, is the perfect tool for maintaining MOA real-world, complex knowledge workflows.
- The MEKA project provides an open source implementation of methods for multi-label classification and evaluation.

Source: https://moa.cms.waikato.ac.nz/

Reading material



- Book: Knowledge discovery from data streams, J. Gamma
- Book: Data streams – Models and Algorithms, C. Aggrawal
- [Zhang et al 1996] BIRCH: an efficient data clustering method for very large databases
- [Breuning et al 2001] Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering
- [Aggrawal et al 2003] A framework for clustering evolving data streams
- [Cao et al 2006] Density-Based Clustering over an Evolving Data Stream with Noise
- [Chen & Tu 2007] Density-Based Clustering for Real-Time Stream Data
- F. Farnstrom, J. Lewis, C. Elkan: Scalability for clustering algorithms revisited. ACM SIGKDD Explorations Ne 57, 2000.
- S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O' Callaghan: Clustering data streams: Theory and practic 15(3):515–528, 2003.
- [O'Callaghan et al 2002] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani: Streaming-Data Algo Quality Clustering. ICDE, 2002.
- [Ester et al 1998] Ester et al, Incremental Clustering for Mining in a Data Warehousing Environment, VLDB 1998.



Thank you

Questions/Feedback/Wishes?

Acknowledgements

- The slides are based on
  - *DM2 lecture@LUH(@Eirini Ntoutsi), KDD2/ SS16 lecture@LMU Munich (@Eirini Ntoutsi, Matthias Schubert, Arthur Zimek)*