

# Knowledge Discovery in Databases II

Summer Semester 2018

## Lecture 3: Data Streams

**Lectures : Prof. Dr. Peer Kröger, Yifeng Lu**

**Tutorials: Yifeng Lu**

Script © 2015, 2017 Eirini Ntoutsis, Matthias Schubert, Arthur Zimek, Peer Kröger, Yifeng Lu

[http://www.dbs.ifi.lmu.de/cms/studium\\_lehre/lehre\\_master/kdd218](http://www.dbs.ifi.lmu.de/cms/studium_lehre/lehre_master/kdd218)

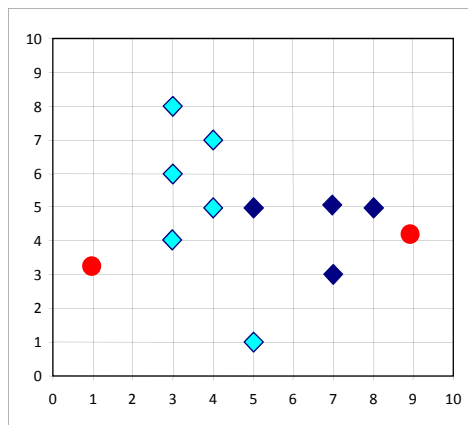
- Data streams usually are a very challenging source of data
- Analysis of data streams require to address several aspects such as
  - The hardware
  - The processing environment (like the operating system, the programming language and the programming schema, ...)
  - The algorithmic design
  - ...
- In this lecture, we focus on the algorithmic aspects that are necessary for processing data streams
- The lecture Big Data Management focuses on other aspects

- Motivation
- Data streams
- Data stream clustering
- Data stream classification

- Most of the DM algorithms focus on batch learning
- Batch learning:
  - The complete training/data set is available to the learning algorithm
  - Data instances can be accessed multiple times
  - e.g., for clustering: k-Means, DBSCAN
  - e.g., for classification: decision trees, Naïve Bayes
- Assumption:
  - Instances are generated by some stationary probability distribution

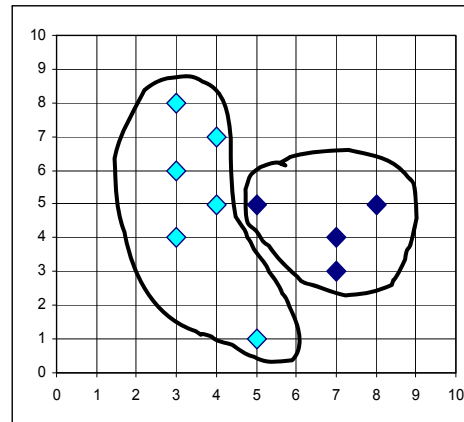
# Example: k-Means batch

- $k=2$

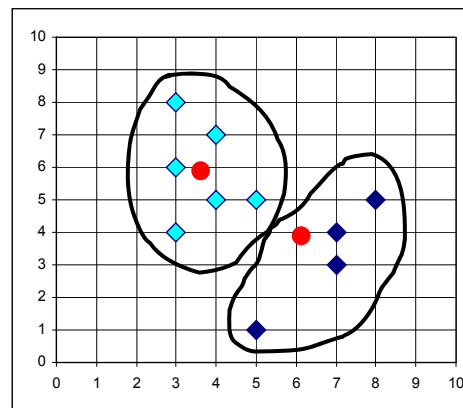


Arbitrarily choose  $k$  objects  
as initial cluster center

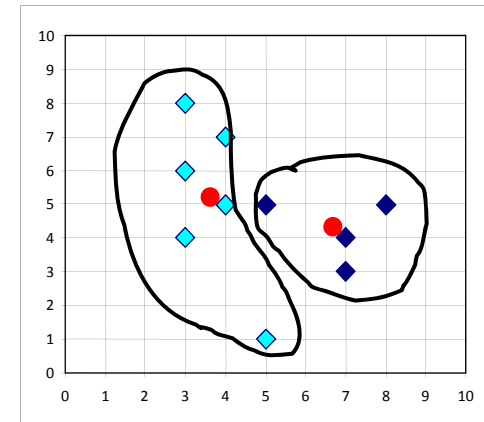
Assign  
each  
objects to  
most  
similar  
center



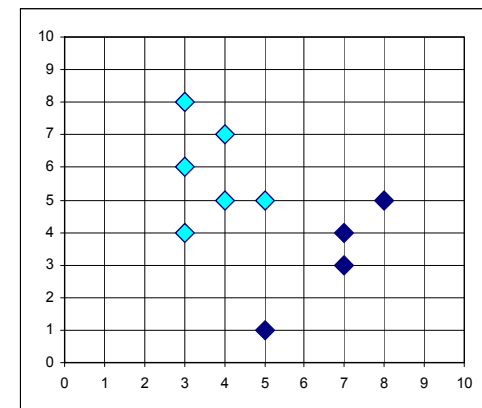
Re-assign



Update  
the  
cluster  
means

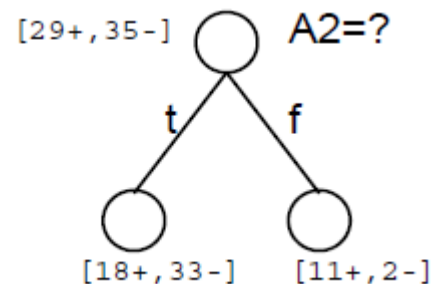
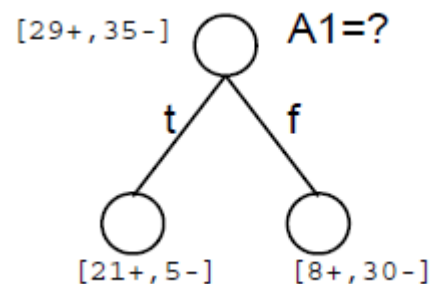


Re-assign



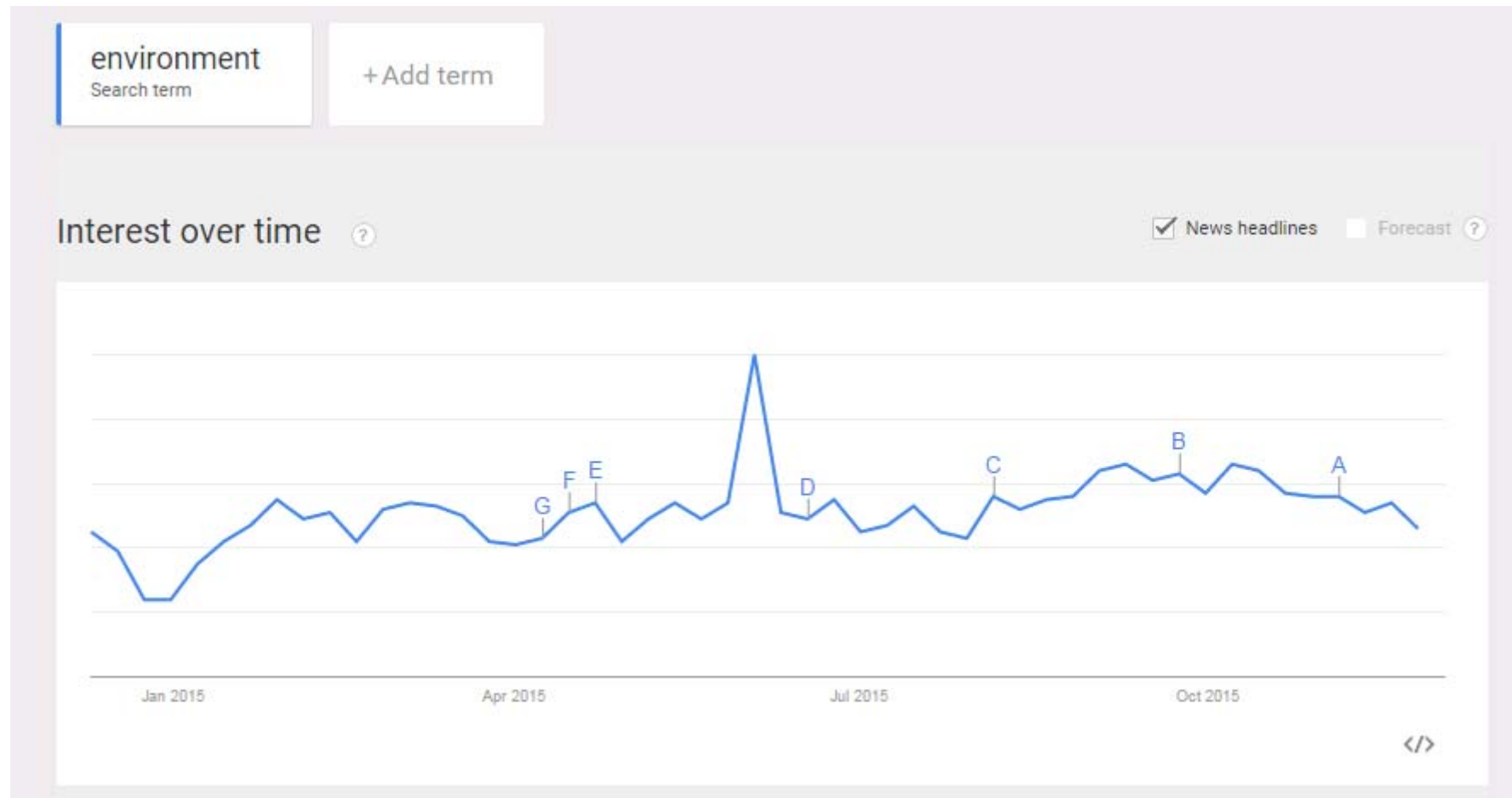
Re-update  
the  
cluster  
means

- Tree is constructed in a top-down recursive divide-and-conquer manner
- At start, *all* the training examples are at the root node.
- The best attribute is selected and used as the splitting attribute at the root
  - For each possible value of the test attribute, a descendant of the root node is created and the instances are mapped to the appropriate descendant node.
- *Repeat* the splitting attribute decision for each descendant node, so instances are partitioned recursively.
- Different attribute selection criteria: information gain, gini index,...



- But, most interesting applications nowadays come from dynamic environments where data are generated over time
  - e.g., customer transactions, call records, customer click data, social media interactions.
- Batch learning is not sufficient anymore as
  - Data is never ending. What is the training set?
  - Multiple access to the data is not possible or desirable
- And also, the data generation process is subject to changes over time
  - The patterns extracted upon such sort of data are also evolving
  - Algorithms should respond to change
    - Incorporate new data instances
    - Forget obsolete data instances

- Stream based on the “environment” keyword

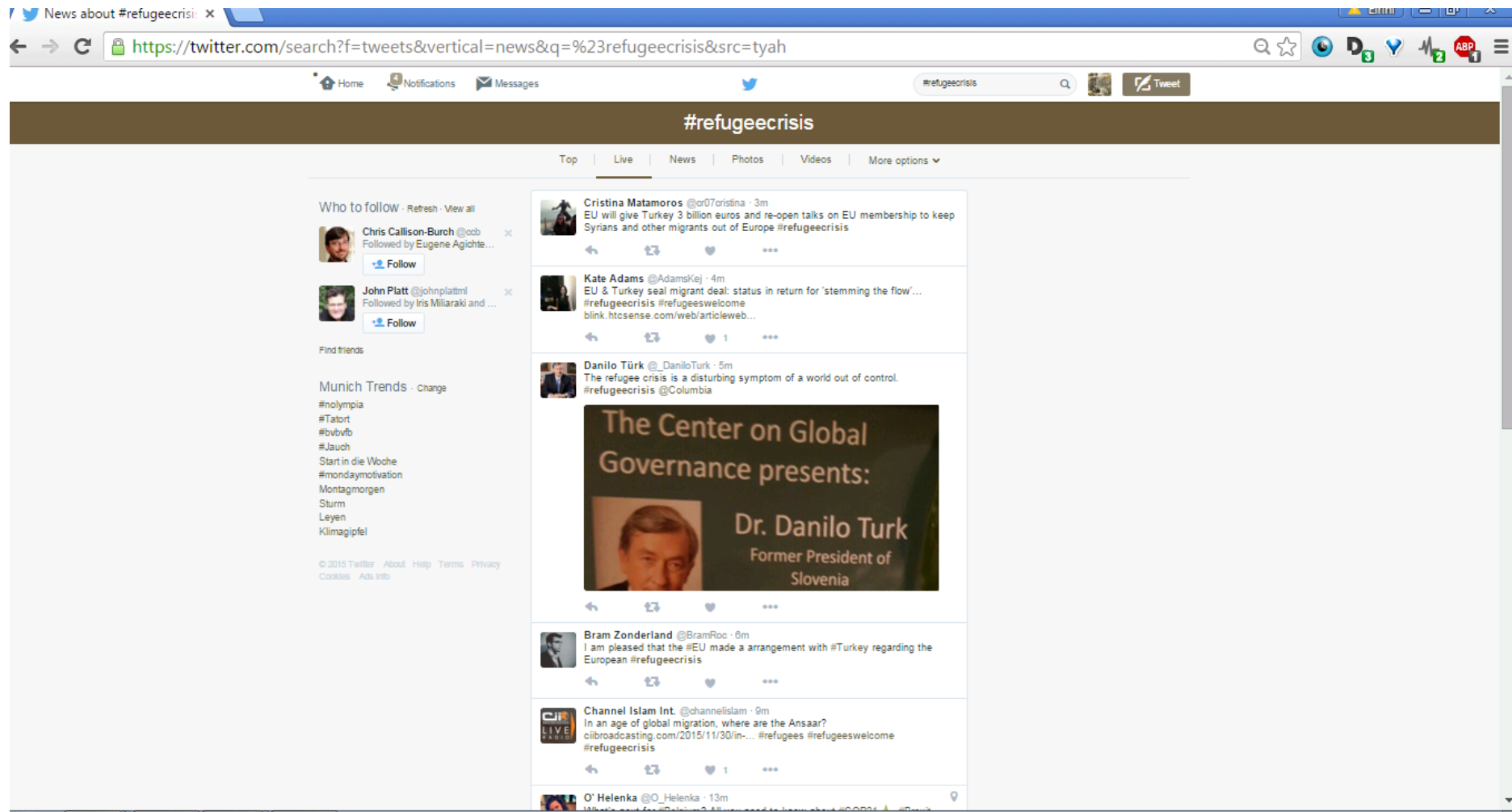


Source: <https://www.google.com/trends/>



## Example: Social streams (Twitter)

- Stream based on the “#refugeecrisis” hashtag



News about #refugeecrisis: x

https://twitter.com/search?f=tweets&vertical=news&q=%23refugeecrisis&src=tyah

#refugeecrisis

Top | Live | News | Photos | Videos | More options

Who to follow · Refresh · View all

Chris Callison-Burch @ccb  
Followed by Eugene Agichte...

John Platt @johnplattml  
Followed by Iris Miliaraki and ...

Find friends

Munich Trends · change

#nolympia  
#Tatort  
#bvbvb  
#Jauch  
Start in die Woche  
#mondaymotivation  
Montagmorgen  
Sturm  
Leyen  
Klimagipfel

© 2015 Twitter · About · Help · Terms · Privacy · Cookies · Ads info

Cristina Matamoros @cr07cristina · 3m  
EU will give Turkey 3 billion euros and re-open talks on EU membership to keep Syrians and other migrants out of Europe #refugeecrisis

Kate Adams @AdamsKej · 4m  
EU & Turkey seal migrant deal: status in return for 'stemming the flow'... #refugeecrisis #refugeeswelcome blink.htcsense.com/web/articleweb...

Danilo Türk @DaniloTurk · 5m  
The refugee crisis is a disturbing symptom of a world out of control. #refugeecrisis @Columbia

The Center on Global Governance presents:  
Dr. Danilo Turk  
Former President of Slovenia

Bram Zonderland @BramRoo · 8m  
I am pleased that the #EU made an arrangement with #Turkey regarding the European #refugeecrisis

Channel Islam Int. @channelislam · 9m  
In an age of global migration, where are the Ansaar? cibroadcasting.com/2015/11/30/in-... #refugees #refugeeswelcome #refugeecrisis

O' Helenka @O\_Helenka · 13m  
What's out for #refugees? #refugeeswelcome #refugeecrisis

## Example: Scientific experiments (CERN)



- Experiments at CERN are generating an entire petabyte (1PB=10<sup>6</sup> GB) of data every second as particles fired around the Large Hadron Collider (LHC) at velocities approaching the speed of light are smashed together
- “*We don’t store all the data* as that would be impractical. Instead, from the collisions we run, we only keep the few pieces that are of interest, the rare events that occur, which our filters spot and send on over the network,” he said.
- This still means CERN is storing 25PB of data every year – the same as 1,000 years' worth of DVD quality video – which can then be analyzed and interrogated by scientists looking for clues to the structure and make-up of the universe.

[Source: http://public.web.cern.ch/public/en/LHC/Computing-en.html](http://public.web.cern.ch/public/en/LHC/Computing-en.html)

[Source: http://www.v3.co.uk/v3-uk/news/2081263/cern-experiments-generating-petabyte](http://www.v3.co.uk/v3-uk/news/2081263/cern-experiments-generating-petabyte)

## Example: Network monitoring (The Network intrusion data stream)

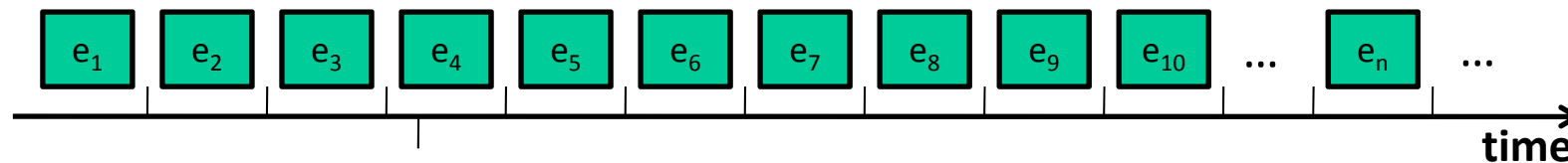
time	duration	protocol_type	service	flag	src_bytes	dst_bytes	...	class
t <sub>1</sub>	0	tcp	http	SF	181	5450	...	normal
t <sub>2</sub>	0	tcp	http	SF	239	486	...	normal
...	...	...	...	...	...	...	...	...
t <sub>7838</sub>	0	icmp	ecr_i	SF	1032	0	...	smurf
t <sub>7839</sub>	0	icmp	ecr_i	SF	1032	0	...	smurf
...	...	...	...	...	...	...	...	...
t <sub>70531</sub>	0	tcp	private	S0	0	0	...	neptune
t <sub>70532</sub>	0	tcp	private	S0	0	0	...	neptune
...	...	...	...	...	...	...	...	...
t <sub>492310</sub>	0	tcp	http	SF	244	7161	...	normal
t <sub>492311</sub>	0	tcp	http	SF	258	9517	...	normal
...	...	...	...	...	...	...	...	...

- The dataset consists of TCP connection records of LAN network traffic managed by Lincoln Labs.
- A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol.
- Connections are described in terms of 42 features like duration, protocol\_type, service, flag, src\_bytes, dst\_bytes etc,.
- Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. There are 4 main categories of attacks: DOS, R2L, U2R, PROBING and are further classified into attack types, like buffer-overflow, guess-passwd, neptune etc.
- Most of the connections in this dataset are normal, but occasionally there could be a burst of attacks at certain times.

More on this dataset: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

- Banks (credit card transactions, loan applications,...)
- Telecommunication (call records, sms, www usage,...)
- Health care systems (customer records in a hospital,...)
- Retail industry (transactions in a supermarket,...)
- WWW (content, interactions, TCP/IP traffic, customer click data,...)
- Science (experiment results,...)
- ...

*“Τα πάντα ρεῖ και οὐδὲν μένει”  
 (“Ta panta rhei kai ouden menei”)  
 “Everything flows, nothing stands still”  
 Heraclitus (535-475 BC)*



- Data evolve over time as new data arrive (and old data become obsolete/irrelevant).
- We can distinguish between:
  - Dynamic data arriving at a low rate (as e.g. in DWs)
    - incremental methods might work for such cases , e.g., incDBSCAN[EsterEtAl98]
  - Data streams: possible infinite sequence of elements arriving at a rapid rate
    - new methods are required to deal with the amount and complexity of these data

- Focus is on how to update the old clustering based on the new data (point  $p$ ), without re-clustering from scratch.
  - Requires (limited) access to raw data

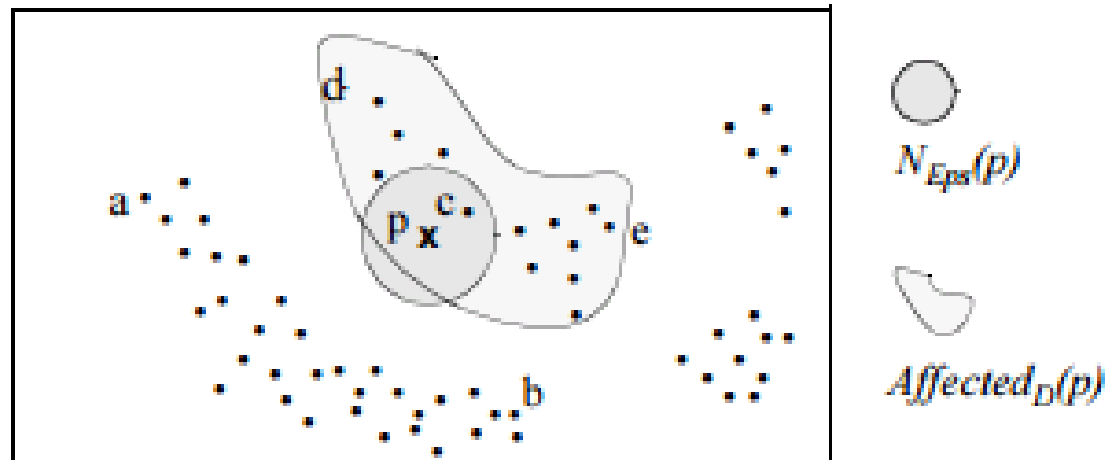


Figure 3: : Affected objects in a sample database

- Data Mining over stream data is more challenging than batch learning:
  - Huge amounts of data → only a small amount can be stored in memory
  - Arrival at a rapid rate → no much time for processing
  - The generative distribution of the stream might change over time rather than being stationary → adapt and report on changes
- Requirements for stream mining algorithms:
  - Use limited computational resources:
    - Bounded memory
    - Small processing time
  - No random access to the data
    - Only 1 look at the data (upon their arrival)

- Example of cluster evolution over time

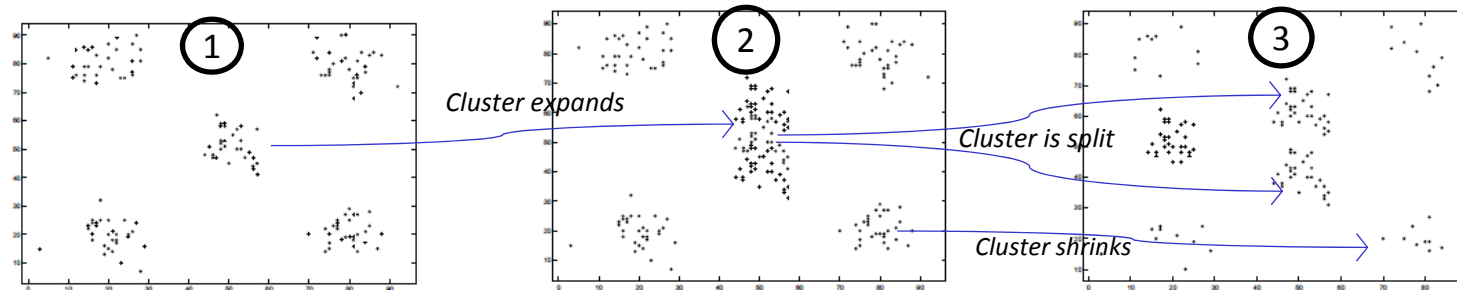


Figure: Data records at three consecutive time stamps, the clustering gradually changes  
(from: *MONIC - Modeling and Monitoring Cluster Transitions*, Spiliopoulou et al, KDD 2006)

- Example of decision boundary drift over time

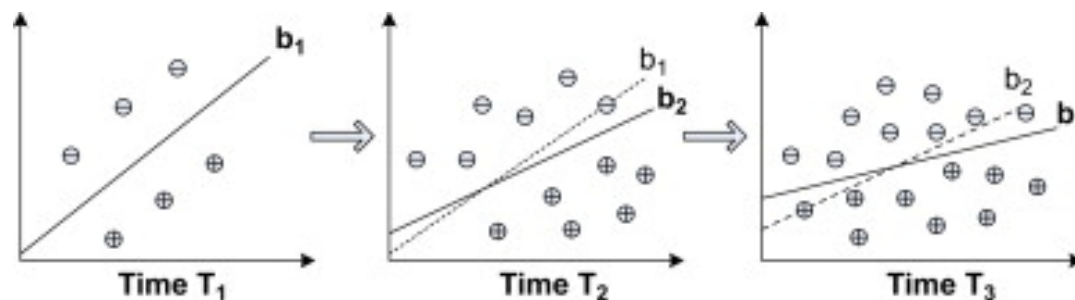


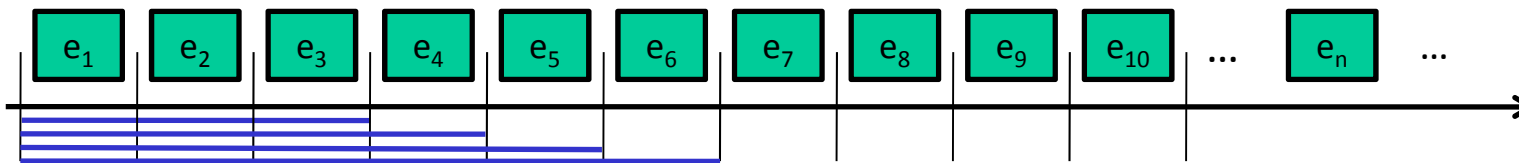
Fig. 1. An illustration of concept drifting in data streams. In the three consecutive time stamps  $T_1$ ,  $T_2$  and  $T_3$ , the classification boundary gradually drifts from  $b_1$  to  $b_2$  and finally to  $b_3$ .

(from: *A framework for application-driven classification of data streams*, Zhang et al, Journal Neurocomputing 2012)

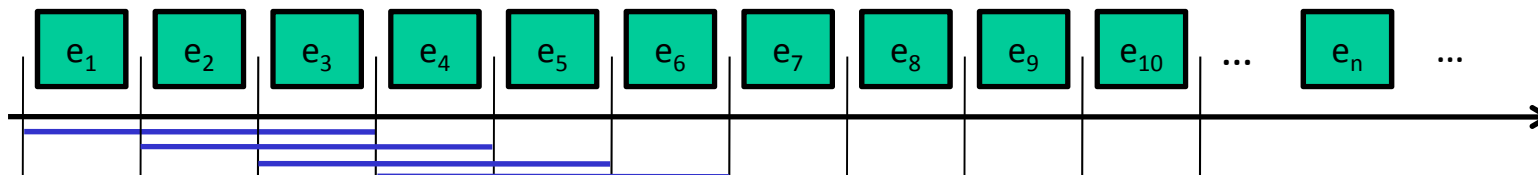


- Usually we are not interested in the whole history of the stream but only in the recent history
  - also a way to deal with non-stationary data generators.
- There are different *ageing/weighting mechanisms or window models* that reflect which part of the stream history is important for learning
  - Landmark window model
  - Sliding window model
  - Damped window model

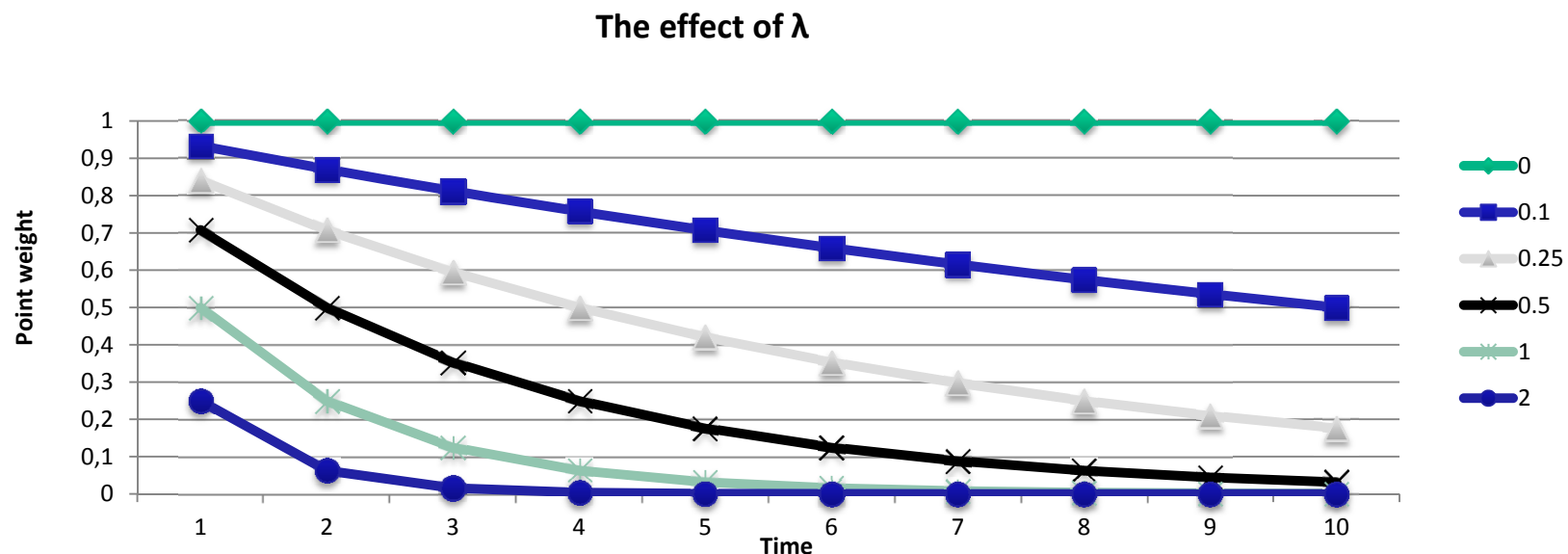
- Landmark window model:
  - Include all objects from a given landmark.
  - All points have a weight  $w=1$ .



- Sliding window model:
  - Remember only the  $n$  more recent entries, where  $n$  is the window size.
  - All points within the window have a weight  $w=1$ , for the rest:  $w=0$ .



- Damped window model:
  - Data are subject to ageing according to a fading function  $f(t)$ , i.e., each point is assigned a weight that decreases with time  $t$  via  $f(t)$ .
  - A widely used fading function in temporal applications is the exponential fading function:  $f(t)=2^{-\lambda t}$ ,  $\lambda>0$ .
    - The decay rate  $\lambda$  determines the importance of historical data
    - The higher the value of  $\lambda$ , the lower the importance of old data



- We focus on:
  - Data stream clustering
  - Data stream classification

- Clustering is one of the core tasks in DM
  - Used as either a standalone tool or as a preprocessing tool
- The (batch) clustering problem:
  - Given a set of measurements, observations, etc., the goal is to group the data into groups of similar data (clusters)
- The data stream clustering problem:
  - Maintain a continuously consistent good clustering of the sequence observed so far, using a small amount of memory and time.
- This implies:
  - Incremental clustering
  - Maintaining cluster structures that evolve over time
  - Working with summaries instead of raw data

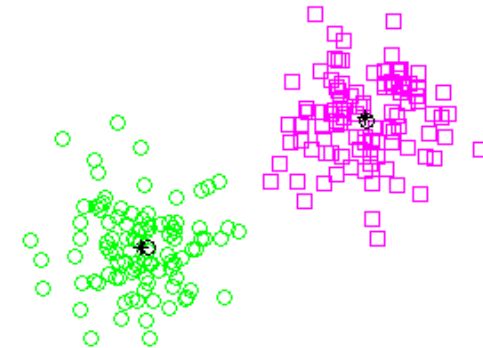
- Traditional clustering methods require access upon the whole dataset
  - Online maintenance of clustering
- The underlying population distribution might change: drifts/ shifts of concepts
  - One clustering model might not be adequate to capture the evolution
- The role of outliers and clusters are often exchanged in a stream
  - Clear and fast identification of outliers

# A taxonomy of stream clustering approaches (*& representative methods*)

	Batch/Static clustering	Dynamic/Stream clustering
Partitioning methods	<ul style="list-style-type: none"> <li>• k-Means</li> <li>• k-Medoids</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Leader</a></li> <li>• Simple single pass k-Means</li> <li>• <a href="#">STREAM k-Means [OCaEtAl02]</a></li> <li>• <a href="#">CluStream [AggEtAl03]</a></li> </ul>
Density-based methods	<ul style="list-style-type: none"> <li>• DBSCAN</li> <li>• OPTICS</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">DenStream [CaoEtAl06]</a></li> <li>• <a href="#">incDBSCAN *</a></li> <li>• <a href="#">incOPTICS *</a></li> </ul>
Grid-based methods	<ul style="list-style-type: none"> <li>• STING</li> </ul>	<ul style="list-style-type: none"> <li>• <a href="#">Dstream [CheTu07]</a></li> </ul>

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

- Goal: Construct a partition of a set of objects into  $k$  clusters
  - 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 [AggEtAl03]



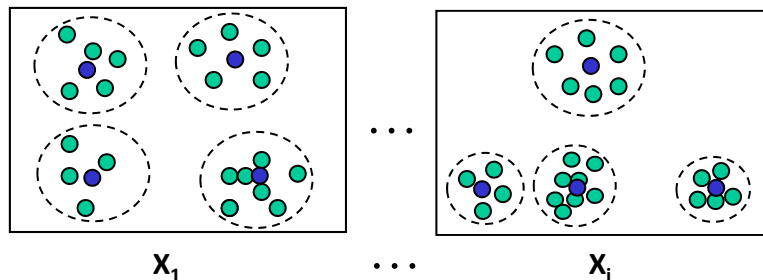


# Leader

(Spath 1980)

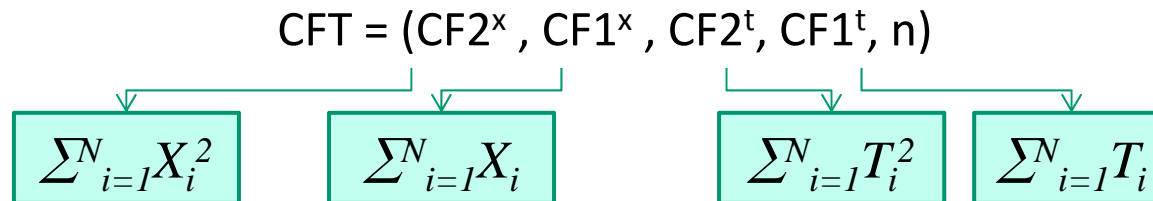
- The simplest single-pass partitioning algorithm
  - Whenever a new instance  $p$  arrives from the stream
    - Find its closest cluster (leader),  $c_{\text{clos}}$
    - Assign  $p$  to  $c_{\text{clos}}$  if their distance is below the threshold  $d_{\text{thresh}}$
    - Otherwise, create a new cluster (leader) with  $p$
- + 1-pass and fast algorithm
- + No prior information on the number of clusters
- Unstable algorithm
  - It depends on the order of the examples
  - It depends on a correct guess of  $d_{\text{thresh}}$

- An extension of k-Means for streams
  - The iterative process of static k-Means cannot be applied to streams
  - Use a buffer that fits in memory and apply k-Means locally in the buffer
  
- Stream is processed in chunks  $X_1, X_2, \dots$ , each fitting in memory
  - For each chunk  $X_i$ 
    - Apply  $k$ -Means locally on  $X_i$  (retain only the  $k$  cluster centers)
    - $X' \leftarrow i \cdot k$  weighted centers obtained from chunks  $X_1 \dots X_i$ 
      - Each center is treated as a point, weighted with the number of points it compresses
    - Apply  $k$ -Means on  $X'$  to obtain the  $k$  centers for the whole stream



- The stream clustering process is separated into:
  - 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

- Assume that the data stream consists of a set of multi-dimensional records  $X_1, \dots, X_n, \dots$ , arriving at  $T_1, \dots, T_n, \dots$ :  $X_i = (x_i^1, \dots, x_i^d)$
- The micro-cluster summary for a set of d-dimensional points  $(X_1, X_2, \dots, X_n)$  arriving at time points  $T_1, T_2, \dots, T_n$  is defined as:

$$\text{CFT} = (\text{CF2}^x, \text{CF1}^x, \text{CF2}^t, \text{CF1}^t, n)$$


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

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

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

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

- Easy calculation of basic measures to characterize a cluster:

- Center:  $\frac{\text{CF1}^x}{n}$
- Radius:  $\sqrt{\frac{\text{CF2}^x}{n} - \left(\frac{\text{CF1}^x}{n}\right)^2}$

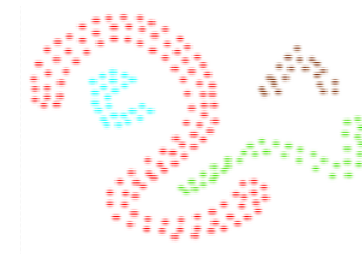
- Important properties of micro-clusters:

- Incrementality:  $\text{CFT}(C_1 \cup p) = \text{CFT}(C_1) + p$
- Additivity:  $\text{CFT}(C_1 \cup C_2) = \text{CFT}(C_1) + \text{CFT}(C_2)$
- Subtractivity:  $\text{CFT}(C_1 - C_2) = \text{CFT}(C_1) - \text{CFT}(C_2), \quad C_1 \supseteq C_2$

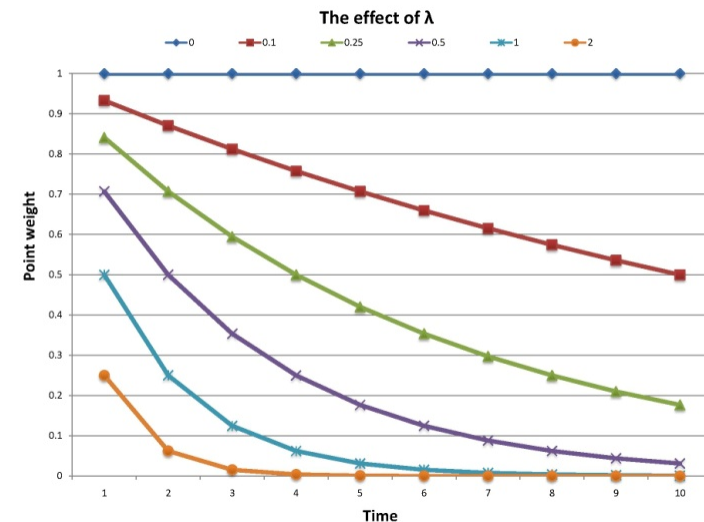
- A fixed number of  $q$  **micro-clusters** is maintained over time
- **Initialize**: apply  $q$ -Means over *initPoints* and build a summary for each cluster
- **Online** micro-cluster maintenance as a new point  $p$  arrives from the stream
  - Find the closest micro-cluster  $clu$  for the new point  $p$ 
    - If  $p$  is within the max-boundary of  $clu$ ,  $p$  is absorbed by  $clu$
    - otherwise, a new cluster is created with  $p$
  - The number of micro-clusters should not exceed  $q$ 
    - Delete most obsolete micro-cluster or merge the two closest ones
- **Periodic** storage of micro-clusters snapshots into disk
  - At different levels of granularity depending upon their recency
- **Offline** 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  $\rightarrow k$  macro-clusters

- + CluStream clusters large evolving data streams
- + Views the stream as a changing process over time, rather than clustering the whole stream at a time
- + Can characterize clusters over different time horizons in changing environment
- + Provides flexibility to an analyst in a real-time and changing environment
  
- Fixed number of micro-clusters maintained over time
- Sensitive to outliers/ noise

- Clusters as regions of high density surrounded by regions of low density (noise)
  - Density is measured locally, in the  $\epsilon$ -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 too complicated)
  - Clusters are represented by all their points
- Solution: Describe clusters as set of summaries
  - DenStream [CaoEtAl06]

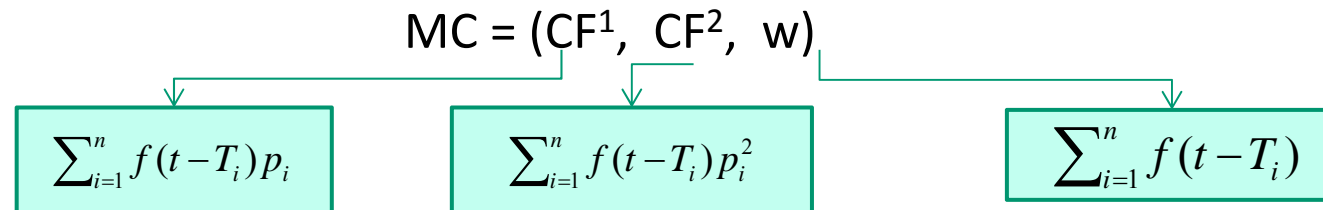


- The online-offline rationale is followed:
  - Online summarization as new data arrive over time
    - Core, potential core and outlier micro-clusters
  - Offline clustering over the summaries to derive the final clusters
    - A modified version of DBSCAN over the summaries
- Data are subject to ageing according to the exponential ageing function (damped window model)
  - $f(t)=2^{-\lambda t}$ ,  $\lambda>0$
  - $\lambda$  the decay rate
  - higher  $\lambda$ , less important the historical data comparing to more recent data





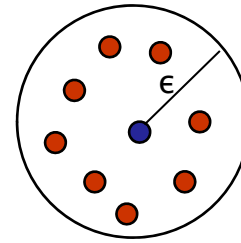
- The micro-cluster summary at time  $t$  for a set of  $d$ -dimensional points  $(p_1, p_2, \dots, p_n)$  arriving at time points  $T_1, T_2, \dots, T_n$  is:



- Easy computation of basic measures:
  - 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  $c_p$ :  $c_p = (CF^2+p^2, CF^1+p, w+1)$
  - If no point is added to  $c_p$  for time interval  $\delta t$ :
    - $c_p = (2^{-\lambda\delta t} * CF^2, 2^{-\lambda\delta t} * CF^1, 2^{-\lambda\delta t} * w)$

- Core (or dense) micro-clusters
  - $(w \geq \mu) \ \& \ (r \leq \epsilon)$

*$\epsilon$ : the radius threshold  
 $\mu$ : the weight threshold*

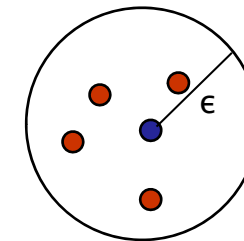


core-micro-cluster

**c-core-micro-cluster**

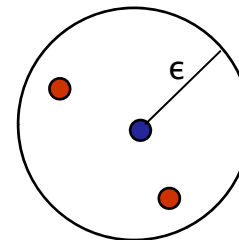
- But, in an evolving stream, the role of clusters and outliers often interchange:
  - Should provide opportunity for the gradual growth of new clusters
  - Should promptly get rid of the outliers

- Potential core micro-clusters
  - $(w \geq \beta * \mu) \ \& \ (r \leq \epsilon), \ 0 < \beta \leq 1$



**p-core-micro-cluster**

- Outlier micro-clusters
  - $(w < \beta * \mu) \ \& \ (r \leq \epsilon), \ 0 < \beta \leq 1$

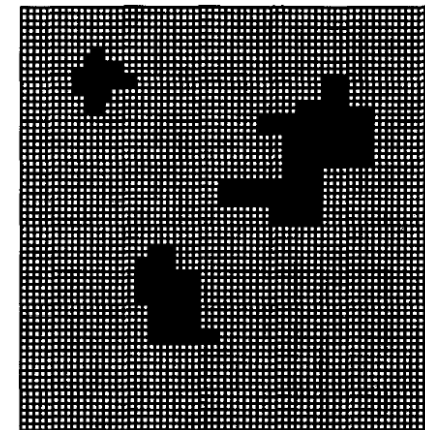


**o-core-micro-cluster**

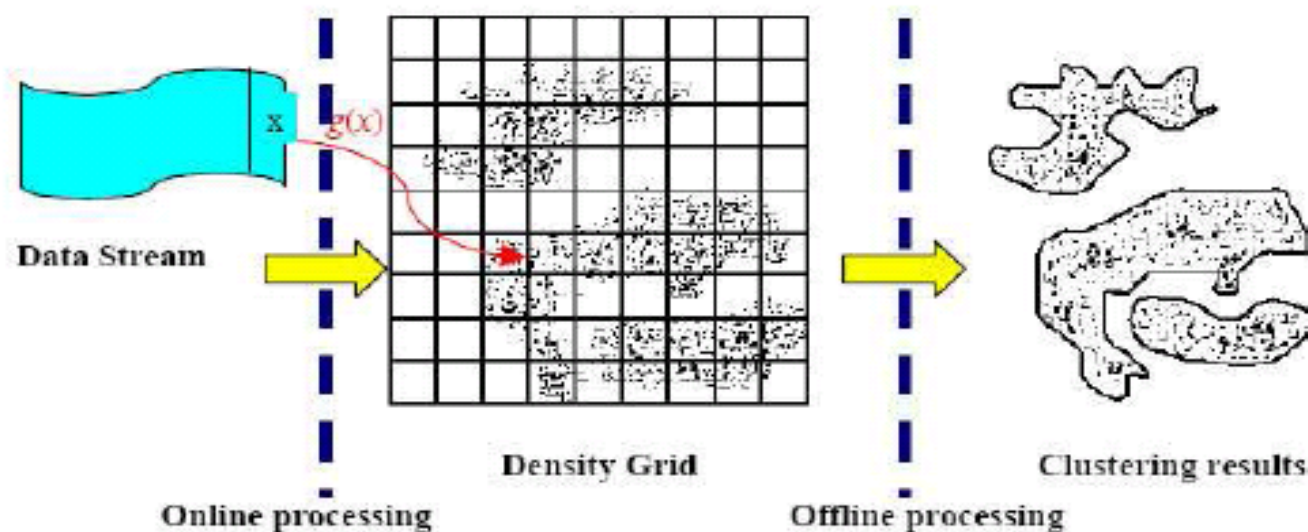
- Two lists of p-micro-clusters and o-micro-clusters are maintained over time
- **Initialize**: apply DBSCAN over *initPoints* → p-micro-clusters
- **Online** step as a new point  $p$  arrives from the stream
  - Try to assign it to its closest p-micro-cluster
  - If this is not possible, try to assign it to its closest o-micro-cluster
    - Check if this o-micro-cluster can be upgraded
  - If both are not possible, create a new o-micro-cluster with  $p$
- **Periodic** micro-cluster maintenance based on data ageing
- **Offline** macro-clustering
  - On demand, upon user request apply a modified version of DBSCAN over p-micro-clusters to derive the final clusters

- + DenStream clusters large evolving data stream
- + 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  $\epsilon$ ,  $\beta$ ,  $\mu$
- Constant parameters over time, what about clusters with different density

- 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
  - 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 [CheTu07]



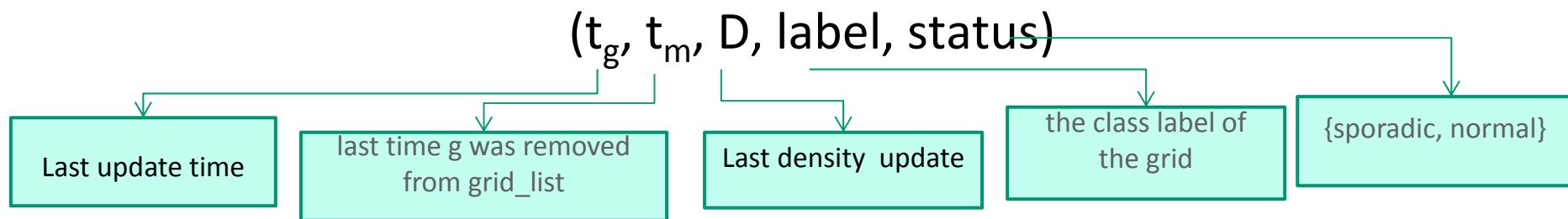
- The online-offline rationale is followed:
  - Online mapping of the new data into the grid
  - Offline computation of grid density and clustering of dense cells



- 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
  - $\lambda$  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:

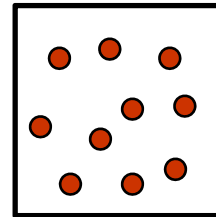


- The grid density can be updated incrementally

$$D(g, t_n) = \lambda^{t_n - t_l} D(g, t_l) + 1 \quad t_n: \text{the new record arrival time}; t_l: \text{the last record arrival}$$

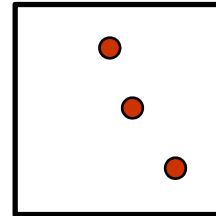
- The density of a grid is constantly changing over time.
- Dense grid cells

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



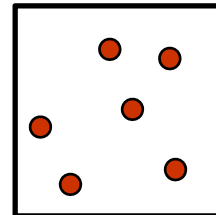
- Transitional grid cells

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



- Sparse grid cells

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



N: #cells in the grid



1. **procedure D-Stream**

2.  $t_c = 0;$

3. initialize an empty hash table *grid\_list*;

4. **while** data stream is active **do**

5.     read record  $x = (x_1, x_2, \dots, x_d);$

6.     determine the density grid  $g$  that contains  $x;$

7.     if ( $g$  not in *grid\_list*) insert  $g$  to *grid\_list*;

8.     update the characteristic vector of  $g;$

9.     **if**  $t_c == gap$  **then**

10.         call *initial\_clustering*(*grid\_list*);

11.     **end if**

12.     **if**  $t_c \bmod gap == 0$  **then**

13.         detect and remove sporadic grids from *grid\_list*;

14.         call *adjust\_clustering*(*grid\_list*);

15.     **end if**

16.      $t_c = t_c + 1;$

17.     **end while**

18. **end\_procedure**

Grid update

Initialization

Clustering

- + 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

- 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
- Evaluation is not straightforward
  - Internal measures of clustering quality (e.g., centroid's radius)
  - External measures of clustering quality (e.g., class labels)
- Specialized approaches for text streams, high-dimensional streams.

- C. Aggarwal, Data Streams: Models and Algorithms, Springer, 2007.
- [AggEtAl03] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A framework for clustering evolving data streams. VLDB, 2003.
- [EsterEtAl98] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, „Incremental Clustering for Mining in a Data Warehousing Environment”, VLDB 1998.
- J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall/CRC, 2010.
- [CaoEtAl06] F. Cao, M. Ester, W. Qian, A. Zhou: Density-Based Clustering over an Evolving Data Stream with Noise. SDM, 2006.
- [CheTu07] Y. Chen, L. Tu: Density-Based Clustering for Real-Time Stream Data. KDD, 2007.
- F. Farnstrom, J. Lewis, C. Elkan: Scalability for clustering algorithms revisited. ACM SIGKDD Explorations Newsletter 2(1):51-57, 2000.
- S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O' Callaghan: Clustering data streams: Theory and practice. IEEE TKDE 15(3):515–528, 2003.
- [OCaEtAl02] L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani: Streaming-Data Algorithms for High-Quality Clustering. ICDE, 2002.
- [www.utdallas.edu/~lkhan/Spring2008G/Charu03.ppt](http://www.utdallas.edu/~lkhan/Spring2008G/Charu03.ppt)