

Lecture notes

# Knowledge Discovery in Databases II

Winter Semester 2012/2013

## Lecture 4: Stream clustering

Lectures: PD Dr Matthias Schubert, Dr. Eirini Ntoutsis  
Tutorials: Erich Schubert

Notes © 2012 Eirini Ntoutsis

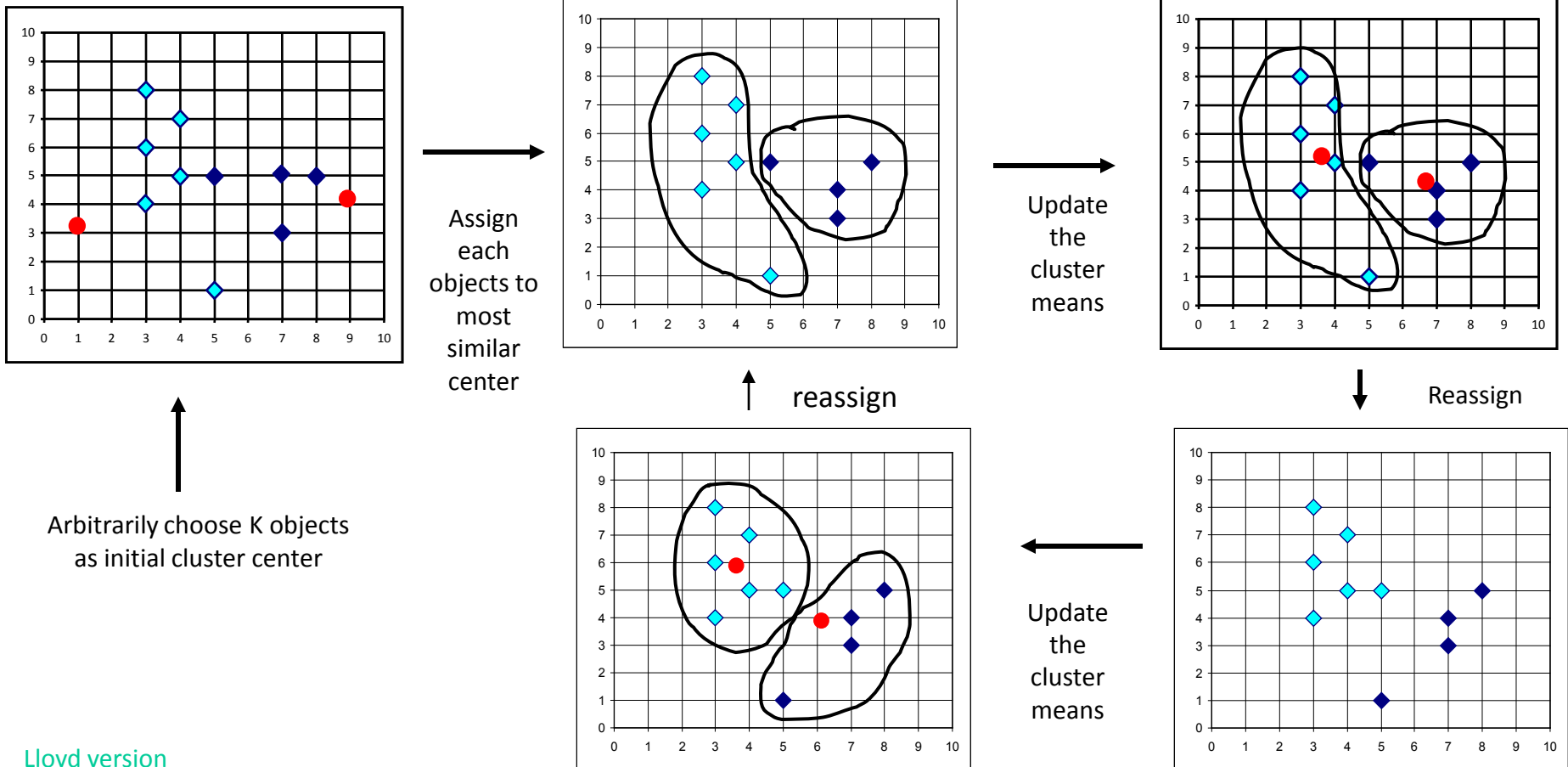
[http://www.dbs.ifi.lmu.de/cms/Knowledge\\_Discovery\\_in\\_Databases\\_II\\_\(KDD\\_II\)](http://www.dbs.ifi.lmu.de/cms/Knowledge_Discovery_in_Databases_II_(KDD_II))

- Motivation
- Data streams
- Data stream clustering
- Data stream clustering algorithms
- Things you should know
- Homework/tutorial

- So far: focus on batch learning using small datasets
- Batch learning:
  - Whole training data is available to the learning algorithm
  - Data instances are processed multiple times
  - e.g. k-Means clusterer (c.f., slide 4), ID3 classifier (c.f., slide 5)
- Assumption:
  - Instances are generated by some stationary probability distribution

# Motivation II: (batch) k-Means example

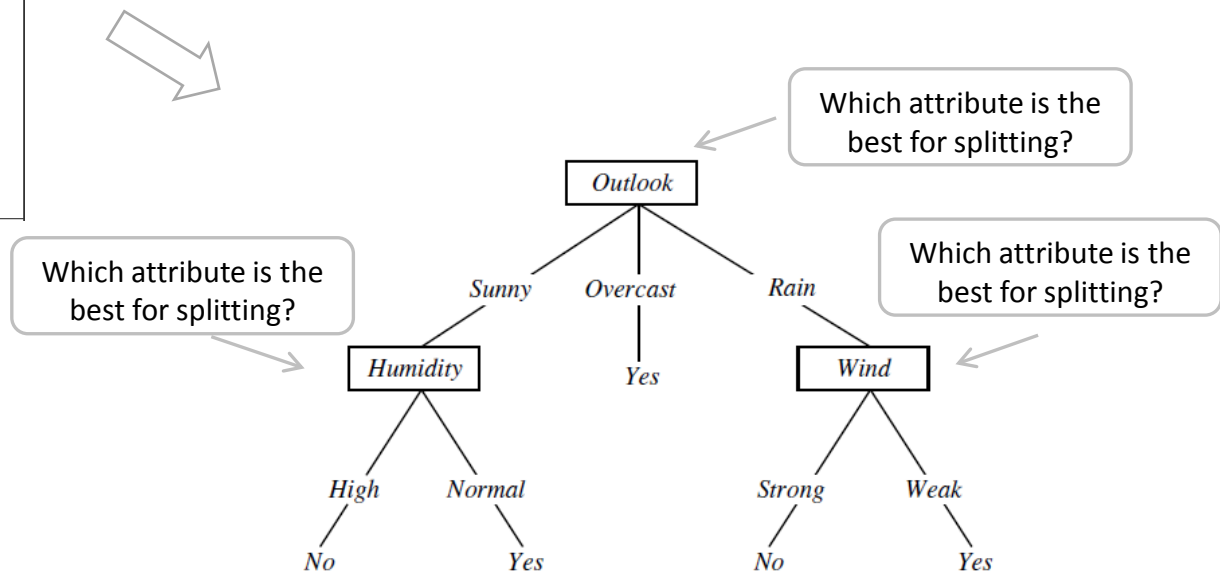
- $k=2$



Lloyd version

# Motivation III: (batch) ID3 example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



- But, most interesting applications come from dynamic environments where data are collected over time
  - e.g., customer transactions, call records, customer click data.
- Batch learning is not sufficient anymore ...
- Algorithms should be able to incorporate new data
  - There are algorithms that are incremental by nature, e.g. kNN classifiers, Naïve Bayes classifiers
  - But the majority of the algorithms need changes to make incremental induction, e.g., ID3, k-Means, DBSCAN.
- Algorithms should be able to deal with non-stationary data generation processes
  - Incorporate concept drift
  - Forget outdated data and adapt to the most recent state of the data

- Cluster evolution

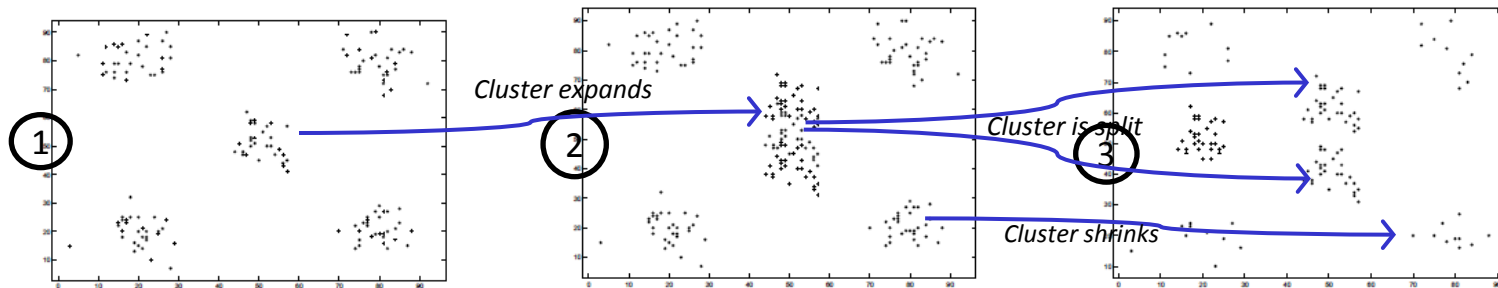


Figure: Data records at three consecutive time stamps, the clustering gradually changes  
(from: *MONIC - Modeling and Monitoring Cluster Transitions*, Spiliopoulou et al, KDD 2006)

- Concept drift

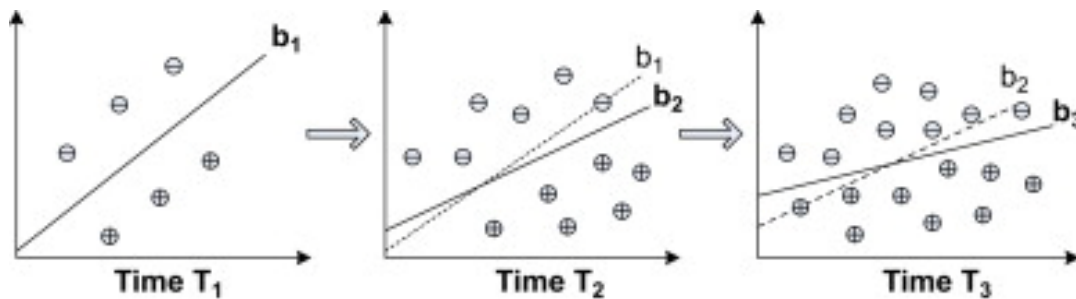


Fig. 1. An illustration of concept drifting in data streams. In the three consecutive time stamps  $T_1$ ,  $T_2$  and  $T_3$ , the classification boundary gradually drifts from  $b_1$  to  $b_2$  and finally to  $b_3$ .

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



# Example applications



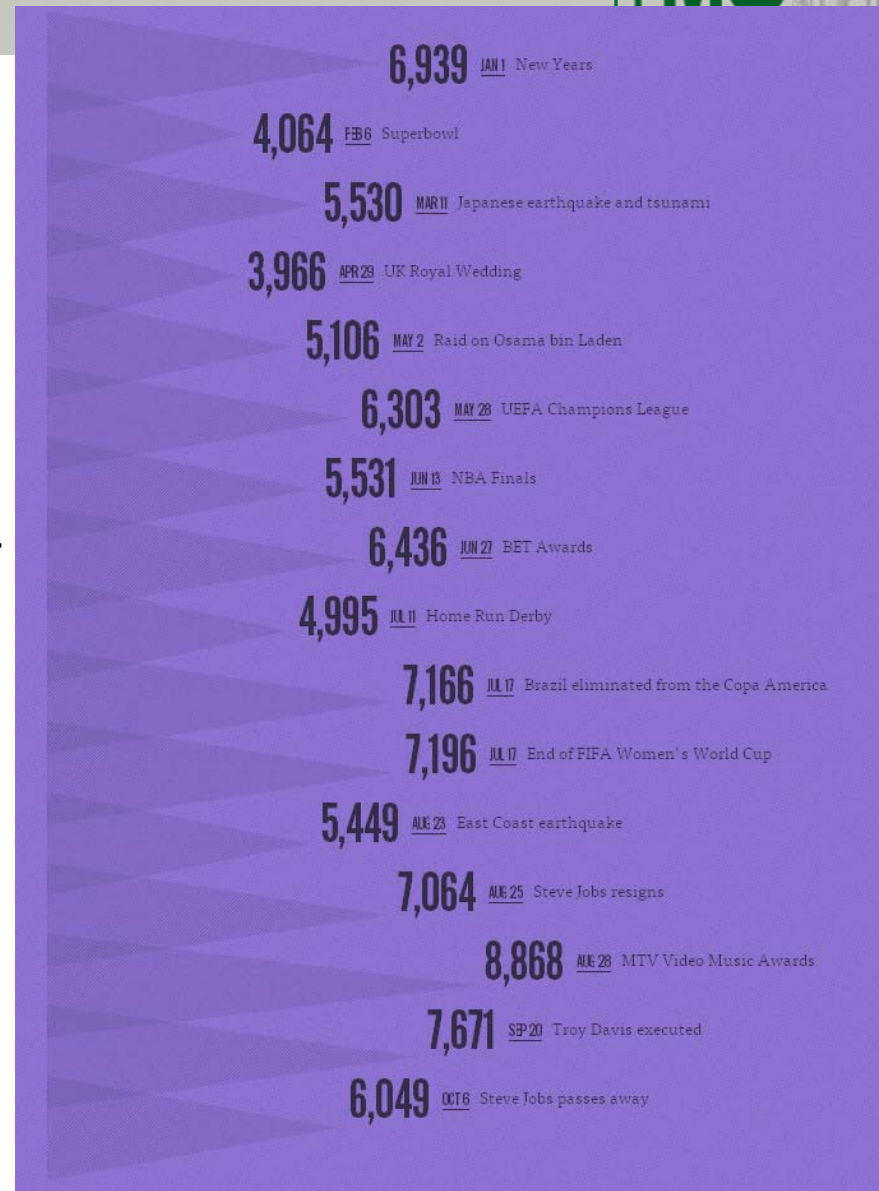
- Banks (credit card transactions, loan applications,...)
- Telecommunication (call records, sms, www usage,...)
- Health care systems (customer records in a hospital,...)
- Retail industry (transactions in a supermarket,...)
- WWW (content, interactions, TCP/IP traffic, customer click data,...)
- Science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



# Example application: Twitter



Tweets per second



# Example application: CERN



- Experiments at CERN are generating an entire petabyte ( $1\text{PB}=10^6\text{ GB}$ ) 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.

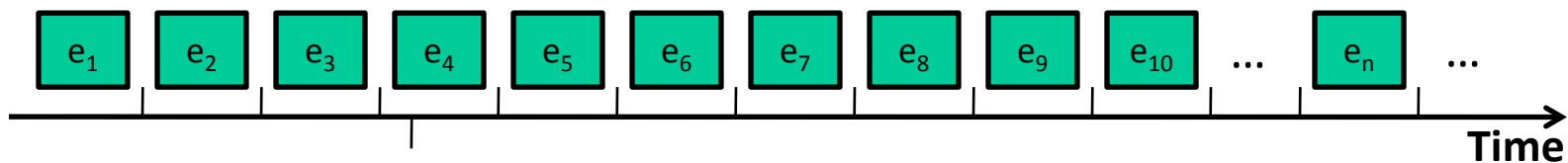
Source: <http://public.web.cern.ch/public/en/LHC/Computing-en.html>

Source: <http://www.v3.co.uk/v3-uk/news/2081263/cern-experiments-generating-petabyte>

- Motivation
- Data streams
- Data stream clustering
- Data stream clustering algorithms
- Things you should know
- Homework/tutorial

*“Τα πάντα ρεῖ καὶ 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
- We can distinguish between:
  - Dynamic data arriving at a low rate
    - Incremental methods might work for such cases
  - 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



# An example dataset: Network traffic data stream

time	duration	protocol_type	service	flag	src_bytes	dst_bytes	...	class
$t_1$	0	tcp	http	SF	181	5450	...	normal
$t_2$	0	tcp	http	SF	239	486	...	normal
...	...	...	...	...	...	...	...	...
$t_{7838}$	0	icmp	ecr_i	SF	1032	0	...	smurf
$t_{7839}$	0	icmp	ecr_i	SF	1032	0	...	smurf
...	...	...	...	...	...	...	...	...
$t_{70531}$	0	tcp	private	S0	0	0	...	neptune
$t_{70532}$	0	tcp	private	S0	0	0	...	neptune
...	...	...	...	...	...	...	...	...
$t_{492310}$	0	tcp	http	SF	244	7161	...	normal
$t_{492311}$	0	tcp	http	SF	258	9517	...	normal
...	...	...	...	...	...	...	...	...

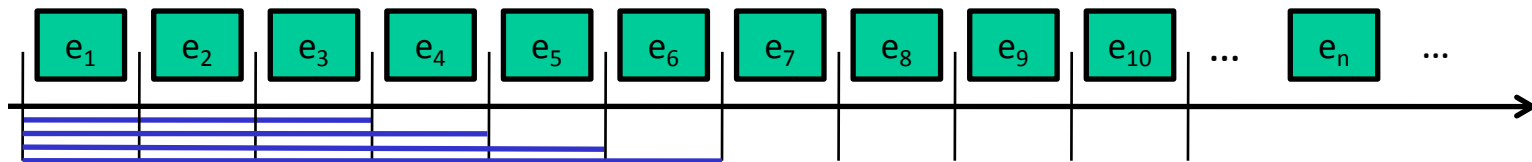
- The dataset consists of TCP connection records of LAN network traffic managed by Lincoln Labs.
- A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol.
- 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>

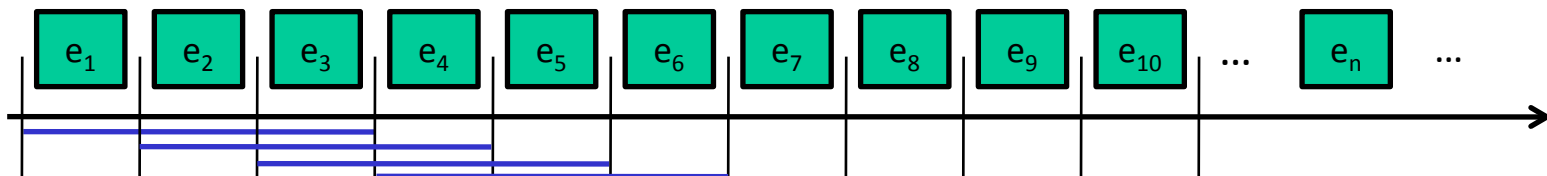
- Data Mining over stream data is even more challenging:
  - Huge amounts of data → only a small amount can be stored in memory
  - Arrival at a rapid rate → no much time for processing
  - The generative distribution of the stream might change over time rather than being stationary → adapt and report 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)

# Data ageing and window models I

- Usually we are not interested in the whole history of the stream but only in the recent history.
- There are different *ageing/weighting mechanisms or window models* that reflect which part of the stream history is important
  - Landmark window:
    - Nothing is forgotten.
    - All points have a weight  $w=1$ .

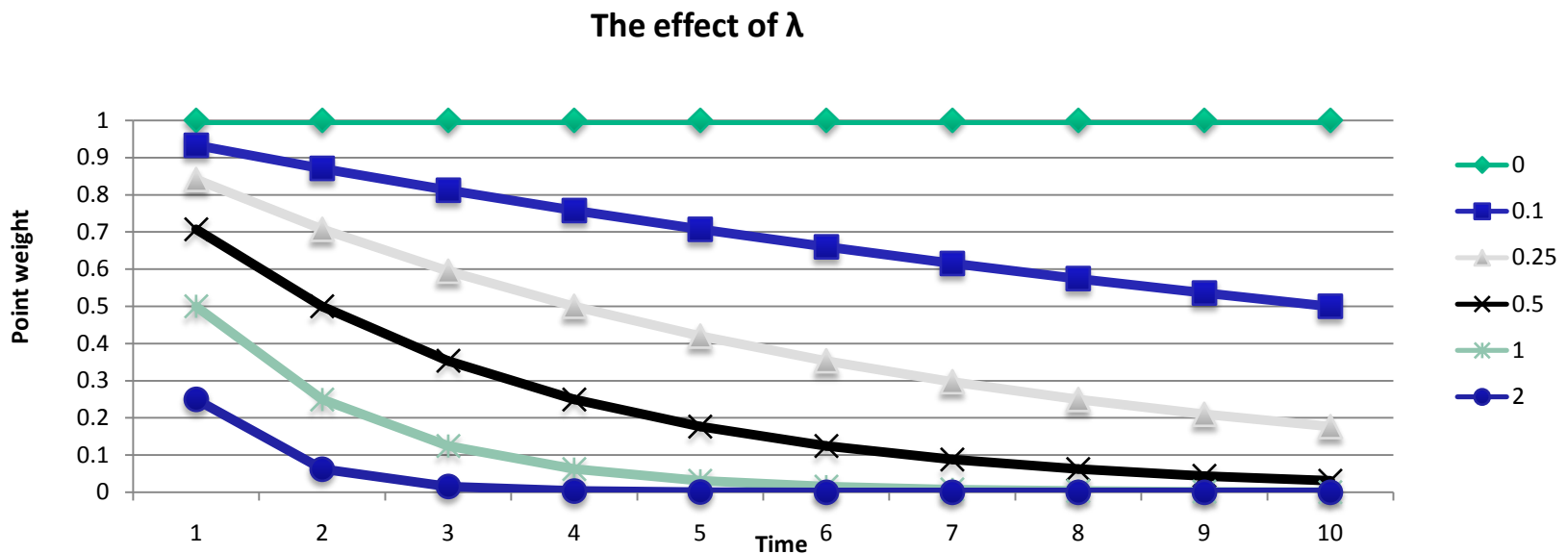


- Sliding window:
  - Remember only the  $n$  more recent entries.
  - All points within the window have a weight  $w=1$ , for the rest:  $w=0$ .



## – Damped window:

- 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





- Motivation
- Data streams
- Data stream clustering
- Data stream clustering algorithms
- Things you should know
- Homework/tutorial

- 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

# Challenges & Requirements for data stream clustering

- 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

# Basic concepts: maintaining simple statistics over a stream

- Mean of a variable  $X$ 
  - We only need to maintain in memory: the linear sum of the variable values  $\sum X_i$  and the number of observations  $n$

$$\mu = \frac{\sum_{i=1}^n X_i}{n}$$

- Stdev of a variable  $X$ 
  - We need to maintain 3 quantities: the linear sum  $\sum X_i$ , the square sum  $\sum (X_i)^2$ , the number of observations  $n$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n}} = \sqrt{\frac{\sum_{i=1}^n X_i^2}{n} - \left(\frac{\sum_{i=1}^n X_i}{n}\right)^2}$$

# Basic concepts:

## Cluster Feature (CF) vectors I

- Introduced in BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*), Zhang et al, SIGMOD'96
- A Cluster Feature or Microcluster is a compact representation of a set of points.
- It's a triple:  $CF = (N, LS, SS)$ 
  - N: # points
  - LS: a vector with the linear sum of the N points
  - SS: a vector with the square sum of the N points
- The CF can be maintained online, which comprises a very appealing property for data streams
  - Incrementality property
  - Additivity property
  - Subtractivity property

# Basic concepts:

## Cluster Feature (CF) vectors II

- Incrementality property

- If a new point  $x$  is added to CF, its statistics are updated as follows:

$$LS_A \leftarrow LS_A + x$$

$$SS_A \leftarrow SS_A + x^2$$

$$N_A \leftarrow N_A + 1$$

*It allows us to add a new point incrementally*

- Additivity property

- If  $A$ ,  $B$  are two CFs merging them is equal to the sum of their counterparts

$$LS_{A \cup B} \leftarrow LS_A + LS_B$$

$$SS_{A \cup B} \leftarrow SS_A + SS_B$$

$$N_{A \cup B} \leftarrow N_A + N_B$$

*It allows us to merge sub-clusters incrementally*

- Subtractivity property

- If  $A$ ,  $B$  are two CFs:  $A \supseteq B$ , splitting them is equal to the subtraction of their counterparts

$$LS_{A-B} \leftarrow LS_A - LS_B$$

$$SS_{A-B} \leftarrow SS_A - SS_B$$

$$N_{A-B} \leftarrow N_A - N_B$$

*It allows us to remove a sub-cluster incrementally*

# Basic concepts:

## Cluster Feature (CF) vectors III

- A CF entry  $CF = (N, LS, SS)$  has sufficient information to derive basic measures to characterize a cluster:
  - Centroid, defined as the gravity center of the cluster:

$$c = \frac{LS}{N}$$

- Radius, defined as the avg distance from member points to the centroid:

$$R = \sqrt{\frac{SS}{N} - \left(\frac{LS}{N}\right)^2}$$

- Motivation
- Data streams
- Data stream clustering
- Data stream clustering algorithms
- Things you should know
- Homework/tutorial

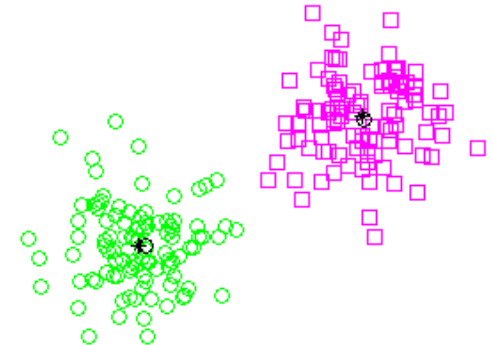


# A taxonomy of stream clustering approaches (*& representative methods*)

	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</a></li> <li>• <a href="#">CluStream</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</a></li> <li>• incDBSCAN *</li> <li>• incOPTICS *</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</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 (O'Callaghan et al, 2002)
  - Online summarization - offline clustering methods:
    - CluStream (Aggarwal et al, 2003)

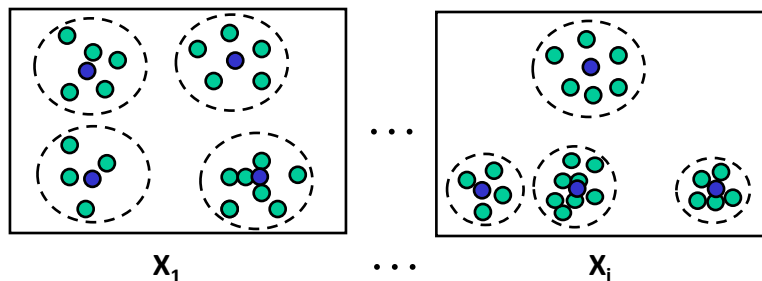


- The simplest single-pass partitioning algorithm
  - Whenever a new instance  $p$  arrives from the stream
    - Find its closest cluster (leader),  $c_{\text{clos}}$
    - Assign  $p$  to  $c_{\text{clos}}$  if their distance is below the threshold  $d_{\text{thresh}}$
    - Otherwise, create a new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

# 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
  - 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  $k$  centers)
    - $X' \leftarrow i * k$  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  $X'$  output the  $k$  centers

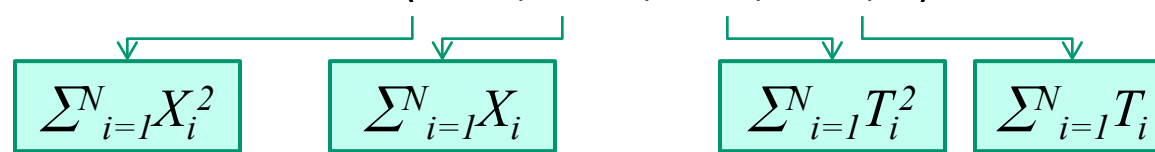


- The stream clustering process is separated into:
  - an online **micro-cluster** component, that summarizes the stream locally as new data arrive over time
    - Micro-clusters are stored in disk at snapshots in time that follow a pyramidal time frame.
  - an offline **macro-cluster** component, that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

# CluStream:

## the micro-cluster summary structure

- 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:  $\text{CFT}(C_1 - C_2) = \text{CFT}(C_1) - \text{CFT}(C_2)$ ,  $C_1 \supseteq C_2$

- A fixed number of  $q$  micro-clusters is maintained over time
- **Initialize**: apply  $q$ -Means over *initPoints*, built 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  $clu$
    - o.w., 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

- Initialization
  - Done using an offline process in the beginning
  - Wait for the first *InitNumber* points to arrive
  - Apply a standard  $k$ -Means algorithm to create  $q$  clusters
    - For each discovered cluster, assign it a unique ID and create its micro-cluster summary.
- Comments on the choice of  $q$ 
  - much larger than the natural number of clusters
  - much smaller than the total number of points arrived



- 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
  - $clu \leftarrow$  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 (incrementality property)
  - Else, create a **new** micro-cluster with  $p$ , assign it a new cluster ID, initialize its statistics
    - To keep the total number of micro-clusters fixed (i.e.,  $q$ ):
      - **Delete** the most obsolete micro-cluster or
        - If its safe (based on how far in the past, the micro-cluster received new points)
      - **Merge** the two closest ones (Additivity property)
        - 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.

- Micro-clusters snapshots are stored at particular moments
- If current time is  $t_c$  and user wishes to find clusters based on a history of length  $h$ 
  - Then we use the subtractive property of micro-clusters at snapshots  $t_c$  and  $t_c-h$
  - In order to find the macro-clusters in a history or time horizon of length  $h$
- How many snapshots should be stored?
  - It will be too expensive to store snapshots at every time stamp
  - They are stored in a pyramidal time frame
- It is an effective trade-off between the storage requirements and the ability to recall summary statistics from different time horizons.

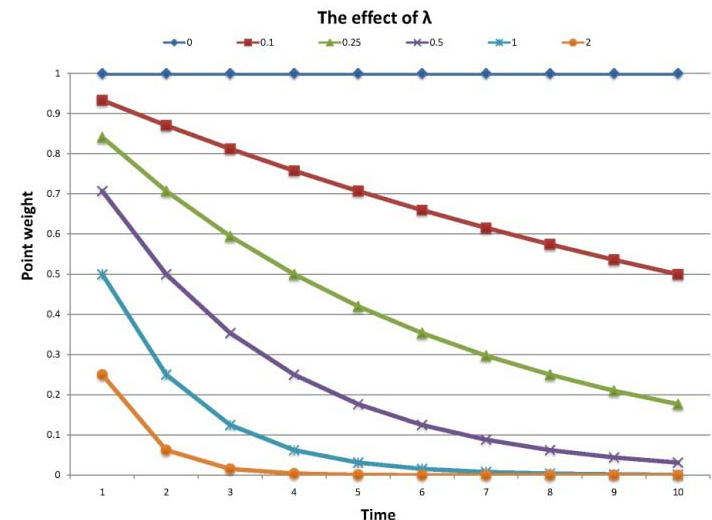
- The offline step is applied on demand upon the  $q$  maintained micro-clusters instead of the raw data
- User input: time horizon  $h$ , # macro-clusters  $k$  to be detected
- Find the active micro-clusters during  $h$ :
  - We exploit the subtractivity property to find the active micro-clusters during  $h$ :
    - Suppose current time is  $t_c$ . Let  $S(t_c)$  be the set of micro-clusters at  $t_c$ .
    - Find the stored snapshot which occurs just before time  $t_c-h$ . We can always find such a snapshot  $h'$ . Let  $S(t_c-h')$  be the set of micro-clusters.
    - For each micro-cluster in the current set  $S(t_c)$ , we find the list of ids. For each of the list of ids, find the corresponding micro-clusters in  $S(t_c-h')$ .
    - Subtract the CF vectors for the corresponding micro-clusters in  $S(t_c-h')$
    - This ensures that the micro-clusters created before the user-specified horizon do not dominate the result of clustering process
- Apply k-Means over the active micro-clusters in  $h$  to derive the  $k$  macro-clusters
  - Initialization: seeds 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 seed for a given partition is the weighted centroid of the micro-clusters in that partition

- + CluStream clusters large evolving data stream
- + 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

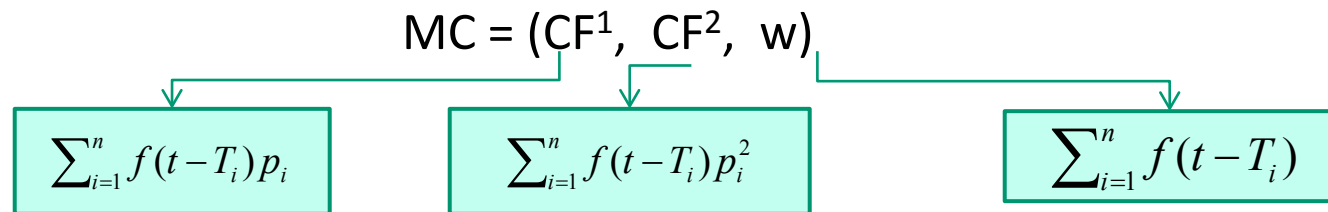
- Clusters as regions of high density surrounded by regions of low density (noise)
  - Density is measured locally, in the  $\varepsilon$ -neighborhood of each point
    - e.g. DBSCAN, OPTICS
- Very appealing for streams
  - No assumption on the number of clusters
  - Discovering clusters of arbitrary 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 (Cao et al, 2006)



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

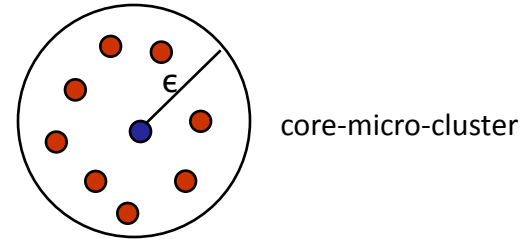
# DenStream:

## core, potential core & outlier summaries

- Core (or dense) micro-clusters

- $(w \geq \mu) \ \& \ (r \leq \epsilon)$

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

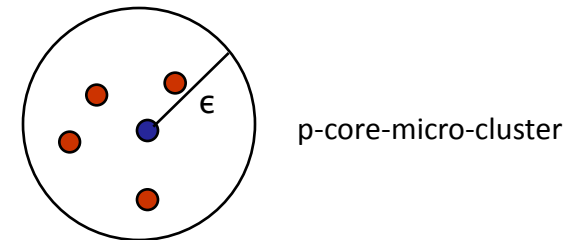


- But, in an evolving stream, the role of clusters and outliers often interchange:

- Should provide opportunity for the gradual growth of new clusters
- Should promptly get rid of the outliers

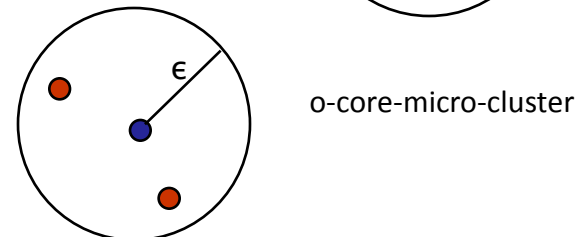
- Potential core micro-clusters

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



- Outlier micro-clusters

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





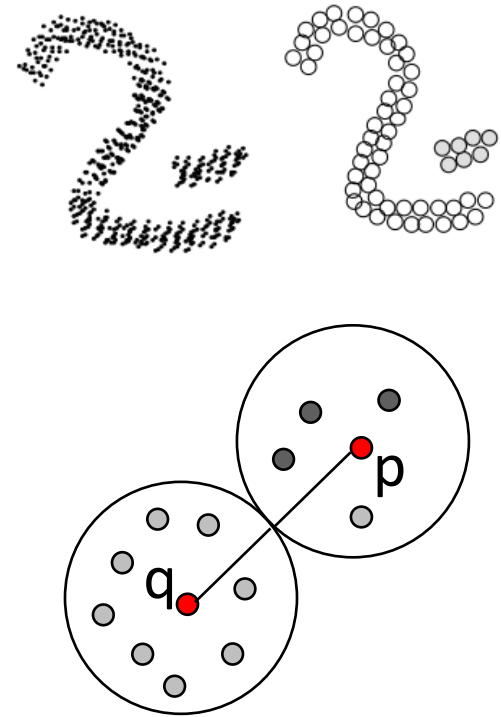
# DenStream: the algorithm

- 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

Two 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  $pclu \leq \epsilon$ , merge  $d$  to  $pclu$
  - o.w. find its closest o-micro-cluster  $oclu$ 
    - If the updated radius of  $oclu \leq \epsilon$ , 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  $d$
- Periodic p- and o-micro-clusters update due to ageing
  - Delete a p-micro-cluster when  $w < \beta * \mu$
  - Delete an o-micro-cluster when  $w < \xi$  (expected weight based on its creation time)
    - The longer an o-micro-cluster exists, the higher its weight is expected to be
  - The number of p- and o-micro-clusters is bounded

- 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
    - $\text{dist}(c_p, c_q) \leq 2\epsilon$  (i.e. they are tangent or intersecting)
- Density reachable (redefined)
  - A p-micro-cluster  $c_p$  is density reachable from a p-micro-cluster  $c_q$  if there is a chain of p-micro-clusters  $c_{p1}=c_q, c_{p2}, \dots, c_{pn}=c_p$ .
- Density connected (redefined)





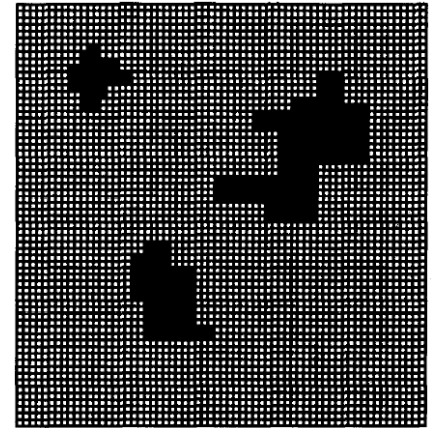
# DenStream

## overview

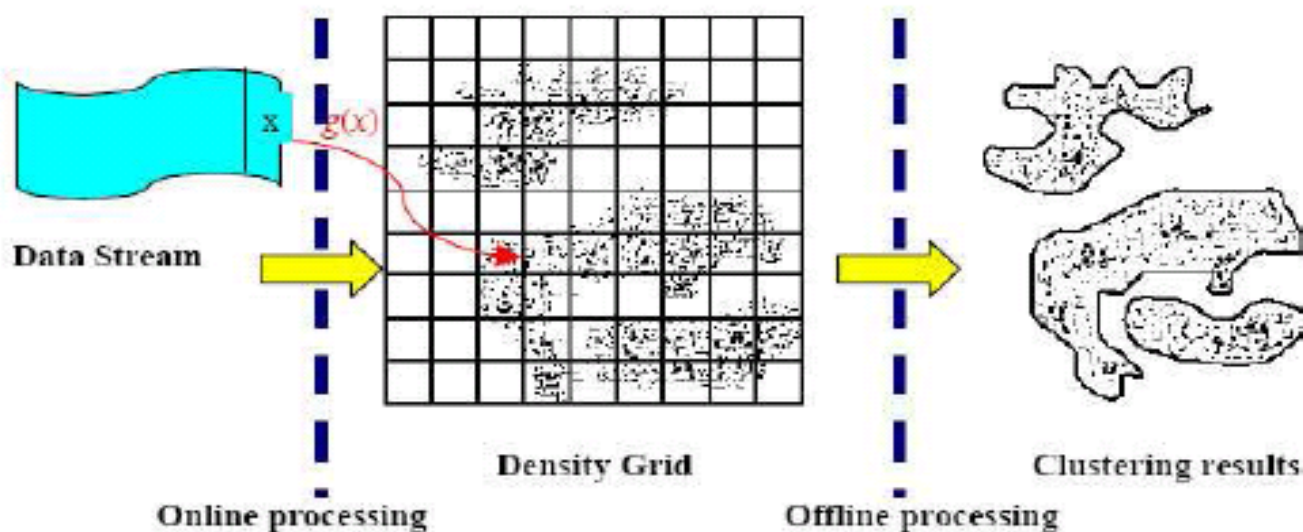


- + DenStream clusters large evolving data stream
- + Discover clusters of arbitrary shapes, following the density-based paradigm
- + No assumption on the number of clusters
- + Noise/ outlier handling
  
- The choice of the parameters  $\varepsilon$ ,  $\beta$ ,  $\mu$
- Constant parameters over time, 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 (Chen & Tu, 2007)



- 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



# Dstream:

## Summarizing the stream into the grid

- 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:

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

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

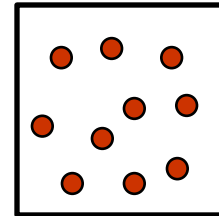
$$D(g, t_n) = \lambda^{t_n - t_l} D(g, t_l) + 1 \quad t_n: \text{the new record arrival time}; t_l: \text{the last record arrival}$$

# Dstream:

## Dense, Sparse and Transitional grid cells

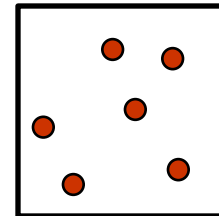
- The density of a grid is constantly changing over time.
- Dense grid cells

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



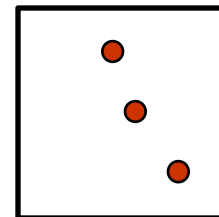
- Transitional grid cells

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



- 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



# DStream: the algorithm

## 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

- Motivation
- Data streams
- Data stream clustering
- Data stream clustering algorithms
- Things you should know
- Homework/tutorial



# Things you should know



- Stream challenges/ requirements for DM
- Stream clustering vs traditional clustering
- The notion and usefulness of summaries
- Stream clustering algorithms, general idea, pros, cons
  - STREAM k-Means
  - CluStream
  - DenStream
  - DStream

- J. Gama, Knowledge Discovery from Data Streams, Chapman and Hall/CRC, 2010.
- C. Aggarwal, Data Streams: Models and Algorithms, Springer, 2007.
- C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A framework for clustering evolving data streams. VLDB, 2003.
- F. Cao, M. Ester, W. Qian, A. Zhou: Density-Based Clustering over an Evolving Data Stream with Noise. SDM, 2006.
- 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.
- 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)

- Tutorial:
  - Thursday 29.11.2012 on stream clustering
- Homework:
  - Try some stream clustering algorithm in the MOA framework (Weka extension to streams): <http://moa.cms.waikato.ac.nz/>

