Table of Contents

1. Introduction to Data Streams

2. Clustering in Data Streams
   2.1 Adaptive Approaches
   2.2 Online - Offline Approaches
   2.3 Continuous Grid-based Approaches
   2.4 Change Detection

3. Classification in Data Streams
1. Introduction to Data Streams

2. Clustering in Data Streams

3. Classification in Data Streams
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.
Batch Learning

• Most of the DM algorithms focus on 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

• Implicit assumption: instances are generated by some stationary probability distribution; data is not volatile and so are patterns
Example: Batch Clustering

- $k$-means (here $k = 2$) needs full access to the data in each iteration

Arbitrarily choose $k$ objects as initial cluster center

Assign each objects to most similar center

Update the cluster means

Re-assign

Re-update the cluster means

Re-assign
Example: Batch Classification

- Decision Trees are constructed in a top-down recursive divide-and-conquer manner requiring full access to the data for each split
  - At start, all the training examples are at the root node
  - Select the best attribute for 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

![Decision Tree Diagram]

\[\begin{align*}
[29+, 35-] & \quad A1=? \\
\text{t} & \quad [21+, 5-] \\
\text{f} & \quad [8+, 30-] \\
\end{align*}\]

\[\begin{align*}
[29+, 35-] & \quad A2=? \\
\text{t} & \quad [18+, 33-] \\
\text{f} & \quad [11+, 2-] \\
\end{align*}\]
• Many 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)
Examples

- Twitter stream for hashtag "#refugeecrisis"

Source: https://www.twitter.com/
• Trend of the search for “environment”

Source: https://www.google.com/trends/
Examples

• Experiments at CERN are generating an entire petabyte (1PB=106 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 do not 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”

• 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://www.v3.co.uk/v3-uk/news/2081263/cern-experiments-generating-petabyte
Examples

- Network monitoring records e.g. TCP connection records of LAN network traffic
- 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
- Most of the connections are usually normal, but occasionally there could be a burst of attacks at certain times

What is a Data Stream?

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
  • 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
Incremental Methods

- Focus is on how to update the current pattern based on the newly arrived data, without re-computing the pattern from scratch.
- Requires (limited) access to raw data (i.e., only the data that is affected by the changes).
- Example: incremental DBSCAN (insertion of a new point $p$)

Figure 3: Affected objects in a sample database
Challenges for Streams

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

- Requirements for stream mining algorithms
  - Use limited computational resources (bounded memory, small amount of available processing time)
  - No random access to the data but rather only one look at the data (upon their arrival)
From Data Changes to Pattern Changes

Example: cluster evolution over time

Figure: Data records at three consecutive time stamps, the clustering gradually changes

Example: 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 $b_3$.
Data Ageing

• 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 for learning
  • Landmark window model
  • Sliding window model
  • Damped window model
Data Ageing Models

• Landmark (window) model
  • Include all objects from a given landmark
  • All points have an equal weight (usually $w = 1$)

• Sliding window model
  • Remember only the $n$ most recent entries, where $n$ is the window size
  • All points within the window have a weight $w = 1$, for the rest: $w = 0$
Data Ageing Models

• 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: $f(t) = 2^{-\lambda t}$, where $\lambda > 0$ is the decay rate that determines the importance of historical data (the higher the value of $\lambda$, the lower the importance of old data).

![The effect of $\lambda$](image)
Time Frames

• Task: maintain the history of the stream
  • Store snapshots at (regular) time intervals
  • Use finer granularity for recent data for a detailed representation
  • Use coarser granularity for older data to save space

• Tilted time frame (tilt time frame)
  • Example: align time axis with natural calendar time, e.g.:
    1 snapshot per minute for the 15 most recent minutes
    1 snapshot per quarter for the 4 most recent quarters
    1 snapshot per hour for the 24 most recent hours
    1 snapshot per day for the 30 most recent day
    1 snapshot per month for the 12 most recent months
    Total number of snapshots for one year: 85
    (compare to 60*24*30*12 = 518400 snapshots)
Pyramidal Time Frame Model

- Stores snapshots in levels of decreasing cardinality (pyramid)
- Size (number of snapshots) is controlled by two parameters $\alpha, \beta \in \mathbb{N}$

- For a new snapshot $A$ provided at time $t_{now}$
  - Store $A$ on the highest level $i$ with $t_{now} \mod \alpha^i = 0$
  - If a level contains more than $\alpha^\beta + 1$ snapshots, remove the oldest
- Maximal height $\lceil \log_\alpha (t_{now}) \rceil$
- Number of snapshots is smaller than $(\alpha^\beta + 1) \cdot \lceil \log_\alpha (t_{now}) \rceil$
- Example: 1 snapshot per second for 100 years using $\alpha = 2$ and $\beta = 1$ results in 96 snapshots
1. Introduction to Data Streams

2. Clustering in Data Streams

2.1 Adaptive Approaches

2.2 Online - Offline Approaches

2.3 Continuous Grid-based Approaches

2.4 Change Detection

3. Classification in Data Streams
Overview

• The (batch) clustering problem
  • Given a set of measurements, observations, etc., the goal is to group the data into groups of similar data objects (clusters)

• The data stream clustering problem
  • Continuously maintain a consistently good clustering of the sequence observed so far, using a small amount of memory and time

• This implies
  • Use incremental computations and techniques
  • Maintaining cluster structures that evolve over time
  • Working with summaries (of such cluster structures) instead of raw data
Challenges

• Traditional clustering methods require access upon the whole data set
• Rather, we need online maintenance of patterns that captures pattern drifts
• 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
• A clear and fast identification of outliers is often crucial for the success
<table>
<thead>
<tr>
<th>Cluster Model</th>
<th>Batch/static clustering</th>
<th>Dynamic/stream clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning methods</td>
<td>k-means, k-medoid</td>
<td>- Leader</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STREAM k-Means</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- CluStream</td>
</tr>
<tr>
<td>Density-based methods</td>
<td>DBSCAN, OPTICS</td>
<td>- DenStream</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- incDBSCAN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- incOPTICS</td>
</tr>
<tr>
<td>Grid-based methods</td>
<td>STRING</td>
<td>- Dstream</td>
</tr>
</tbody>
</table>
• **Goal:** Construct a partition of a set of objects into $k$ clusters
• **Two types of methods**
  • Adaptive methods such as Leader (Spath 1980), Simple single pass $k$-Means (Farnstrom et al., 2000), STREAM $k$-Means (OCaEtAl02)
  • Online summarization - offline clustering methods such as CluStream (AggEtAl03), DenStream (CaoEtAl06)
  • Continuous grid-based such as DStream (CheTu07)
1. Introduction to Data Streams

2. Clustering in Data Streams

2.1 Adaptive Approaches

2.2 Online - Offline Approaches

2.3 Continuous Grid-based Approaches

2.4 Change Detection

3. Classification in Data Streams
• 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 \( p \)

• Properties
  • 1-pass and fast algorithm
  • No prior information on the number of clusters required
  • Result depends on the order of the examples
  • Sensitive to a correct guess of \( d_{thresh} \) (which is fixed)
• Simple extension of batch $k$-Means to streams:
  • Use a buffer (chunk) that fits in memory and apply $k$-Means locally in the buffer

• STEAM $k$-Means:
  • Apply $k$-Means on chunk $X_i$
  • $X'$ denotes the set of $i \cdot k$ cluster centers from all chunks $X_1, \ldots, X_i$ each weighted by the number of points assigned to it
  • Output the $k$ centers obtained by clustering $X'$
Properties:

- **Pros:**
  - Single scan

- **Cons:**
  - Expensive (according to authors)
  - No aging
  - Cluster model inherent limitations (no noise handling, ...)
  - Fixed $k$ in all chunks
Kapitel 2: Clustering

1. Introduction to Data Streams

2. Clustering in Data Streams
   2.1 Adaptive Approaches
   2.2 Online - Offline Approaches
   2.3 Continuous Grid-based Approaches
   2.4 Change Detection

3. Classification in Data Streams
Basic Idea

• Online component
  • Maintain a larger number of small clusters (micro-cluster)
  • Reduce data, keep sufficient details
  • Separate clusters for noise (improved robustness)
  • Provide accurate and fine grained input for further steps

• Offline component
  • Generate actual clustering on user request using micro-cluster information
  • Exchangeable clustering method
  • Individual and changing parameterization possible
  • Only approximate clustering
Micro Clusters: Cluster Features

• Clustering Features\(^1\) for a set of points \(X\): \(CF_X = (N_X, LS_X, SS_X)\) with
  • \(N_X\) is the number of points, i.e., \(|X|\)
  • \(LS_X\) is the linear sum of all points in \(X\), i.e., \(\sum_{x_i \in X} x_i\)
  • \(SS_X\) is the squared sum of all points in \(X\), i.e., \(\sum_{x_i \in X} x_i^2\)

• From \(CF_X\) we can easily compute basic statistics of \(X\) such as
  • Mean (centroid) of \(X\)
  • Compactness measures such as radius, diameter, variance and std. deviation

• \(CFs\) are additive, i.e., given two (disjunctive) sets \(X\) and \(Y\) with their corresponding \(CF_X\) and \(CF_Y\), we can compute \(CF_{X \cup Y}\) as follows:

\[
CF_{X \cup Y} = CF_X + CF_Y = (N_X + N_Y, LS_X + LS_Y, SS_X + SS_Y)
\]

\(^1\)Zhang, Ramakrishnan, Linvy: BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proc. ACM SIGMOD 1996
Micro Clusters: Data Bubbles

• While CFs are good for partitioning based clustering, they do not capture density estimations necessary for e.g. OPTICS
• Data Bubbles\(^2\) for a set of points \(X\): \(B_X = (N_X, M_X, r_X)\) with
  • \(N_X\) is the number of points, i.e., \(|X|\)
  • \(M_X\) is the centroid of \(X\)
  • \(r_X\) is the radius of the ball centered at \(M\) capturing all points in \(X\)
• Data Bubbles can be computed from CFs
• Data Bubbles allow a good approximation of core/reachability distances for hierarchical clustering

CluStream — Basics

• One of the first algorithms for streams proposing an online/offline framework

• Uses cluster features to propose a $k$-Means like stream clustering method

• Cluster Features (see above) are extended by the information of the time slots $T$ when points in $X$ have arrived, i.e. $x_i$ has arrived at time $t_i$:

$$CFT_X = (N_X, LS_X, SS_X, LST_X, SST_X),$$

where

- $N$, $LS_X$, and $SS_X$ are defined as above (note that $LS_X$ and $SS_X$ are vectors)
- $LST_X$ is the linear sum of time slots of $X$, i.e., $\sum_{t_i \in T} t_i$
- $SST_X$ is the linear sum of time slots of $X$, i.e., $\sum_{t_i \in T} t_i^2$

• Again, important for the stream situation:

  • $CFT$s can be maintained incrementally, i.e. $CFT_{X \cup p} = CFT_X + p$
• General idea: a fixed number of $q$ micro-clusters (represented as CFTs) is maintained over time

• Initialize: apply $q$-Means over a buffer of $initP$ observations and build a summary for each cluster

• Both $q$ and $initP$ are input parameters

• Upon request, $k$-Means can be applied to a snapshot of the $q$ CFTs
CluStream — Online Phase

- Maintain $q$ micro-clusters while adding a new observation $x_i$ from the stream
  - Find closest micro-cluster $MC_j$ according to distance $\text{dist}(x_i, \mu_j)$
  - If $\text{dist}(x_i, \mu_j) < \alpha \cdot \sigma_j$ then add $x_i$ to $MC_j$
  - Else create a new micro-cluster containing only $x_i$ and delete a micro cluster by using one of the following actions:
    - Delete the least recent $MC$ if its relevance stamp $t_r < t_{now} - \tau$
    - Merge the two closest micro clusters

- $\alpha \cdot \sigma_j$ is called the maximal boundary of $MC_j$

- The relevance stamp $t_r$ of $MC_j$ approximates the average time stamp of the last $m$ objects

- It is computed as the time of arrival of the $m/(2 \cdot N)$-th percentile (i.e., $1 - m/2 \cdot N$ of the time stamps in $MC_j$)
CluStream — Offline Phase

- Snapshots of micro-clusters are stored in pyramidal time frame
- Given $k$ and a time horizon $h$
- Locate all valid micro-clusters within $h$
- Final clusters are gained using a modified $k$-Means
  - Micro-clusters over a certain time horizon are treated as pseudo-points
  - In the initialization: seeds are not picked randomly, but sampled with a probability proportional to $N$
  - Distances are calculated between centroids of the micro-clusters
  - New seeds are weighted by $N$
  - The $k$ clusters obtained from applying $k$-Means on the micro-clusters are called macro-clusters
CluStream — Properties

• Single scan, stream compression using micro-clusters
• 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
• Aging only for entire clusters
• Noise handling offline
• Not adaptive $q$ is fixed, requires $k < q$
• Many parameters
• Sensitive to outliers and noise (also model inherent)
DenStream — Motivation

- Density-base cluster model: clusters as regions of high density surrounded by regions of low density (noise)
- 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
- So we can again only hope to approximate an arbitrary shaped cluster by many small (circular) micro-clusters
• The DenStream algorithm uses time-weighted cluster features at time slot \( t \) given a time weighting function \( f \) for observations \( x_i \) arriving at time \( t_i < t \):

\[
CF^t_X = (N^t_X, LS^t_X, SS^t_X)
\]

where

- \( N^t_X = \sum_{x_i \in X} f(t - t_i) \)
- \( LS^t_X = \sum_{x_i \in X} f(t - t_i)x_i \)
- \( SS^t_X = \sum_{x_i \in X} f(t - t_i)x_i^2 \)

- Usually, \( f(t) = 2^{-\lambda t} \) models the damped window model (but other functions are possible)
• If a new observation $x_i$ is added, a micro-cluster summary $CF^t_X$ can be maintained incrementally (analogously as above)

• If no point is added to $CF^t_X$ for time interval $\Delta t$, then

$$CF^t_X = (2^{-\lambda \Delta t} \cdot N, 2^{-\lambda \Delta t} \cdot LS^t_X, 2^{-\lambda \Delta t} \cdot SS^t_X)$$

• The radius $r_X$ of a micro-cluster $X$ can be derived from the cluster feature $CF^t_X$ as follows

$$r_X = \sqrt{SS^t_X/N^t_X - (LS^t_X/N^t_X)^2}$$

• Analogously, the center $c_X$ of a micro-cluster can be computed from its $CF^t_X$
Given the density threshold $\mu$ (#points) and $\varepsilon$ (volume) and a weighting factor $\beta$ ($0 < \beta \leq 1$), DenStream maintains three different types of micro-clusters:

- **Core (or dense) micro-clusters (CMC) $X$** if $N^t_X \geq \mu$ and $r_X \leq \varepsilon$

- **Potential core micro-clusters (PCMC) $X$** if $N^t_X \geq \beta \cdot \mu$ and $r_X \leq \varepsilon$

  (provides the opportunity for transitions between new clusters and outliers)

- **Outlier micro-clusters (OMC) $X$** if $r_X \leq \varepsilon$ and $N^t_X < \beta \cdot \mu$

Note: all MC types always have a radius $\leq \varepsilon$
DenStream — Initialization

- Collect a set \( I \) of \( initP \) of initial points
- For any \( p \in I \):
  - Compute \( \epsilon \)-neighborhood \( N_\epsilon(p) \) of \( p \)
  - If \( |N_\epsilon(p)| \geq \mu \) (\( p \) is core), create a new CMC \( X = N_\epsilon(p) \) and remove \( X \) from \( I \)
- For all remaining \( p \in I \): create a new OMCs \( X = N_\epsilon(p) \) and remove \( X \) from \( I \)
Online micro-cluster maintenance (when a new observation $x_i$ arrives)

- Core micro-clusters are not considered
- Find closest potential core micro-cluster $X_p$
  - If $\text{dist}(x_i, c_{X_p}) \leq \varepsilon$
    - Add $x_i$ to $X_p$
    - Check if $X_p$ becomes a CMC
  - Else
    - Find closest outlier micro-cluster $X_o$
    - If $\text{dist}(x_i, c_{X_o}) \leq \varepsilon$, add $x_i$ to $X_o$ and check if $X_o$ becomes a PCMC
    - Else: create a new OMC $X_{x_i} = \{x_i\}$

- After a given number of $T$ time steps, check:
  - Delete all CMC $X$ with $N_X^t < \mu$
  - Delete all OMC that did not become CMC within the last $T$ time steps
DenStream — Offline Phase

- Upon user request, run DBSCAN on current CMCs and PCMCs
- Use centers and weights of the micro-clusters
DenStream — Discussion

- Single scan, stream compression using micro-clusters
- Noise/ outlier handling (model inherent)
- Flexible data aging model (for individual objects)
- Constant parameters over time, what about clusters with changing density?
## Online/Offline-Approaches: Summary

<table>
<thead>
<tr>
<th></th>
<th>CluStream</th>
<th>DenStream</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online</td>
<td>convex micro cluster</td>
<td></td>
</tr>
<tr>
<td>Offline</td>
<td>$k$-Means</td>
<td>DBSCAN</td>
</tr>
<tr>
<td>Aging</td>
<td>entire MCs</td>
<td>individual objects</td>
</tr>
</tbody>
</table>

- Cluster algorithm in offline phase exchangeable in principle
- Still “high” online costs (check all MCs)
- Many variants exist
1. Introduction to Data Streams

2. Clustering in Data Streams
   2.1 Adaptive Approaches
   2.2 Online - Offline Approaches
   2.3 Continuous Grid-based Approaches
   2.4 Change Detection

3. Classification in Data Streams
Basic Idea

- A grid structure is used to capture the density of the data set
- A cluster is a set of connected dense cells (see 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 are considered as micro-clusters, i.e., summary information on cells are maintained
  - Update these summaries on the grid structure as the stream proceeds
  - Sample method: DStream (CheTu07)
• DStream divides each dimension into $l$ partitions resulting in $l^d$ cells ($d$: data dimensionality)
• Populated grid cells are maintained in a hash list
• For a grid cell $C$, the following summary is stored:

$$CF_C = (t_{update}, t_{spor}, N_C, label_C, status_C)$$

where

• $t_{update}$ is the last update time
• $t_{spor}$ last time, $C$ has been removed
• $N_C = \sum_{x_i \in C} \lambda^{t-t_i} \cdot x_i$ (count using damped window window aging)
• $label$ is the cluster label
• $status \in \{sporadic, normal\}$
DStream — Overview

• DStream follows the online/offline paradigm
• Online mapping of the new data into the grid
• Offline computation of grid density and clustering of dense cells
• Three cell types are defined by parameters $\tau_{dense}$ and $\tau_{sparse}$:
  • Cell $C$ is dense if $N_C > \tau_{dense}$
  • Cell $C$ is sparse if $N_C < \tau_{sparse}$
  • Cell $C$ is transitional if $\tau_{sparse} < N_C < \tau_{dense}$

• Connected regions of dense or transitional cells form a cluster

• Changes of the status occur in the online component
  • Set status to normal, if $C$ changed from sparse to another type
  • Set status to sporadic, if for $C$ the number of insertions into $C$ is less than expected since the last update
Online grid cell maintenance (for new observation $o_i$):

- Determine the grid cell $C$ that $x_i$ falls into
- Add $C$ to the hash list if it is not already contained
- Update $CF_C$ w.r.t. $x_i$ and set status to normal if type changed from sparse
- Periodically after $T$ time steps
  - Delete all grid cells from the hash list that have been marked as *sporadic* and did not receive new points within the last $T$ time steps
  - Mark sparse grid cells as *sporadic* if requirements (see previous slide) are met
  - Adjust the clustering
• Single scan, stream compression using micro-clusters
• Noise/ outlier handling (model inherent)
• Aging model for entire cells
• Constant parameters over time, what about clusters with changing density?
• Curse of dimensionality (number of grid cells is $l^d$)
Grid-based Variant: Dynamic Splitting

- Initially generate coarse grid

- Online split cells with interval > $\lambda$:
  - Mean split in the dimension with maximal variance
  - Around mean segment in the dimension with minimal variance

- Delete outdated cells based on user defined threshold (initial grid cells are never deleted)

- No particular offline component
1. Introduction to Data Streams

2. Clustering in Data Streams
   2.1 Adaptive Approaches
   2.2 Online - Offline Approaches
   2.3 Continuous Grid-based Approaches
   2.4 Change Detection

3. Classification in Data Streams
Basic Idea

• Detect and differentiate different types of changes
  • Disappearance of concepts
  • Migration/changes/drift of concepts
  • Merging of existing concepts
  • Splitting of groups vs. newly emerging clusters

![Diagram showing concepts of drift, growth, split, novelty, and disappearance over time](image-url)
MONIC: Modeling and Monitoring Cluster Transitions

- MONIC\(^3\) does not assume a particular cluster model
- Cluster matching for different points in time
  - Let \( X \) be a cluster at \( t_X \) and \( Y \) a cluster from the set of clusters \( \zeta_Y \) at a later slot \( t_Y > t_X \) and let the overlap between \( X \) and \( Y \) be
    \[
    \text{overlap}(X, Y) = \frac{\sum_{o \in X \cap Y} \text{age}(t_Y, o)}{\sum_{x \in X} \text{age}(t_Y, x)}
    \]
  - \( Y \) is a match for \( X \), \( \text{match}_\tau(X, \zeta_Y) = Y \) subject to a threshold \( \tau \in [0.5, 1] \) if \( Y \) is the cluster with the maximum overlap of at least \( \tau \) where the overlap between two clusters \( X \) and \( Y \) is
  - If there is no cluster in the clustering \( \zeta_Y \) at \( t_Y \) with an overlap of at least \( \tau \), then \( \text{match}_\tau(X, \zeta_Y) = \emptyset \)
  - The matching is not unique: several old clusters can be matched with the same new cluster

\(^3\)Spiliopoulou et al.: MONIC - Modeling and Monitoring Cluster Transitions. Proc. KDD’06
MONIC: Cluster Transitions

Table 1: External transitions of a cluster

<table>
<thead>
<tr>
<th>Transition type</th>
<th>Subtype</th>
<th>Notation</th>
<th>Indicator</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Size transition</td>
<td>1a. the cluster shrinks</td>
<td>$X \setminus Y$</td>
<td>$\sum_{x \in X} \text{age}(x, t_i) &gt; \sum_{y \in Y} \text{age}(y, t_j) + \varepsilon$</td>
</tr>
<tr>
<td>2. Compactness transition</td>
<td>2a. the cluster becomes compacter</td>
<td>$X \leftarrow Y$</td>
<td>$\sigma(Y) &lt; \sigma(X) - \delta$</td>
</tr>
<tr>
<td>3. Location transition</td>
<td>Shift of center (I1) or distribution (I2)</td>
<td>$X \cdot \rightarrow Y$</td>
<td>I1. $</td>
</tr>
</tbody>
</table>

Table 2: Internal transitions of a cluster

<table>
<thead>
<tr>
<th>Transition type</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>No change</td>
<td>$X \rightarrow Y$</td>
</tr>
</tbody>
</table>

Summary: Stream Clustering

- 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
- Different window models
- Evaluation is not straightforward (existing measures mostly for static case)
- Specialized approaches for text streams, high-dimensional streams.
Selected Readings on Stream Clustering

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

2. Clustering in Data Streams

3. Classification in Data Streams