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

Overview

Cluster Model	Batch/static clustering	Dynamic/stream clustering
Partitioning methods	k-means, k-medoid	– Leader – STREAM k-Means – CluStream
Density-based methods	DBSCAN, OPTICS	– DenStream – incDBSCAN – incOPTICS
Grid-based methods	STRING	– Dstream

Overview

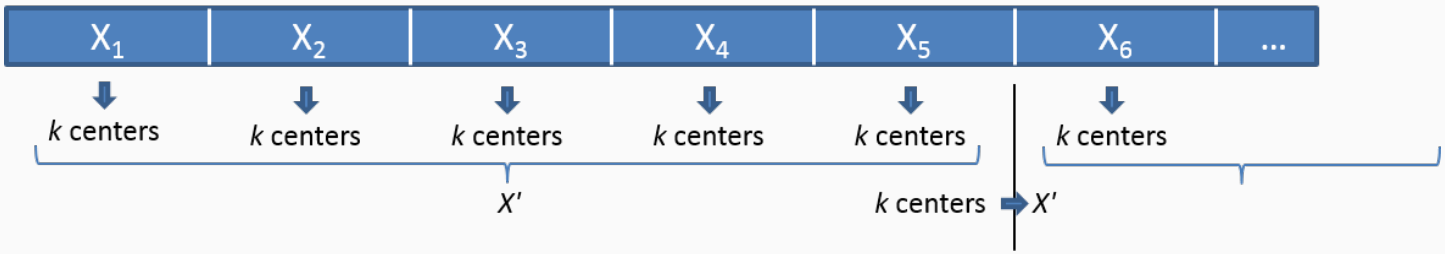
- Goal: Construct a partition of a set of objects into k clusters
- Two types of methods
  - Adaptive methods such as Leader (Spath 1980), Simple single pass k-Means (Farnstrom et al., 2000), STREAM k-Means (OCaEtAl02)
  - Online summarization - offline clustering methods such as CluStream (AggEtAl03), DenStream (CaoEtAl06)
  - Continous grid-based such as DStream (CheTu07)

Leader

- 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)

STEAM k-Means

- 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'



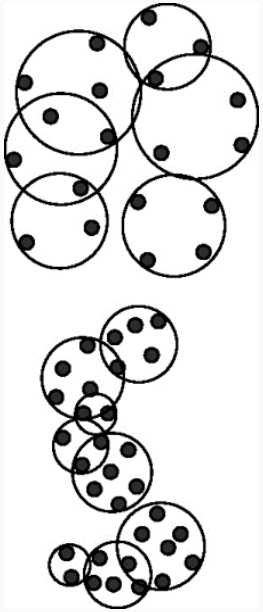
STEAM k-Means

Properties:

- Pros:
  - Single scan
- Cons:
  - Expensive (according to authors)
  - No aging
  - Cluster model inherent limitations (no noise handling, ...)
  - Fixed k in all chunks

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

[2] Breunig, Kriegel, Kröger, Sander: Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering. Proc. ACM SIGMOD 2001

CluStream — Basics

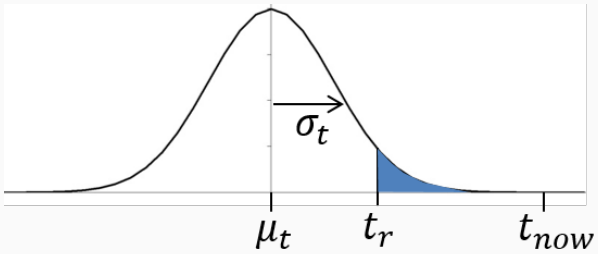
- One of the first algorithms for streams proposing an online/offline framework
- Uses cluster features to propose a k-Means like stream clustering method
- Cluster Features (see above) are extended by the information of the time slots T when points in X have arrived, i.e. x_i has arrived at time t_i:
  $CFT_X = (N_X, LS_X, SS_X, LST_X, SST_X)$, 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:
  - CFTs can be maintained incrementally, i.e. $CFT_{X \cup p} = CFT_X + p$

CluStream — Method at a Glance

- 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 $dist(x_i, \mu_j)$
  - If $dist(x_i, \mu_j) < \alpha \cdot \sigma_j$ 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

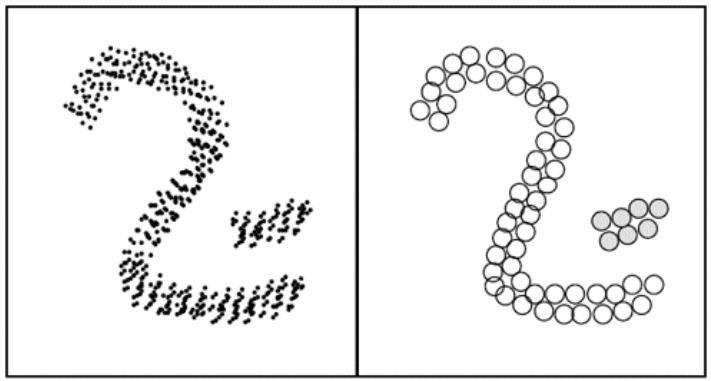


CluStream — Offline Phase

- 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

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



DenStream — Basics

- 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 $t_i < t$:

$$CF_X^t = (N_X^t, LS_X^t, SS_X^t)$$

where

  - $N_X^t = \sum_{x_i \in X} f(t - t_i)$
  - $LS_X^t = \sum_{x_i \in X} f(t - t_i) x_i$
  - $SS_X^t = \sum_{x_i \in X} f(t - t_i) x_i^2$

- Usually, $f(t) = 2^{-\lambda t}$ models the damped window model (but other functions are possible)

DenStream — Basics

- If a new observation x_i is added, a micro-cluster summary CF_X^t can be maintained incrementally (analogously as above)
- If no point is added to CF_X^t for time interval Δt, then $CF_X^t = (2^{-\lambda \Delta t} \cdot N, 2^{-\lambda \Delta t} \cdot LS_X^t, 2^{-\lambda \Delta t} \cdot SS_X^t)$
- The radius r_X of a micro-cluster X can be derived from the cluster feature CF_X^t as follows

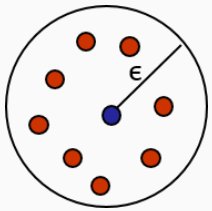
$$r_X = \sqrt{SS_X^t / N_X^t - (LS_X^t / N_X^t)^2}$$

- Analogously, the center c_X of a micro-cluster can be computed from its CF_X^t

DenStream — Basics

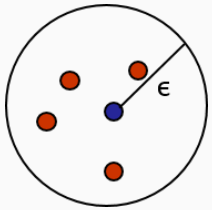
Given the density threshold μ (#points) and ε (volume) and a weighting factor β ($0 < \beta \leq 1$), DenStream maintains three different types of micro-clusters:

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

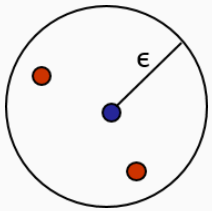


- Potential core micro-clusters (PCMC) X if $N_X^t \geq \beta \cdot \mu$ and $r_X \leq \varepsilon$

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



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



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 ε-neighborhood $N_\varepsilon(p)$ of p
  - If $|N_\varepsilon(p)| \geq \mu$ (p is core), create a new CMC $X = N_\varepsilon(p)$ and remove X from I
- For all remaining $p \in I$: create a new OMCs $X = N_\varepsilon(p)$ and remove X from I

DenStream — Online Phase

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 $dist(x_i, c_{X_p}) \leq \varepsilon$
  - Add x_i to X_p
  - Check if X_p becomes a CMC
- Else
  - Find closest outlier micro-cluster X_o
  - If $dist(x_i, c_{X_o}) \leq \varepsilon$, 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

	CluStream	DenStream
Online	convex micro cluster	
Offline	k-Means	DBSCAN
Aging	entire MCs	individual objects

- Cluster algorithm in offline phase exchangeable in principle
- Still “high” online costs (check all MCs)
- Many variants exist

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 — Basics

- 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:

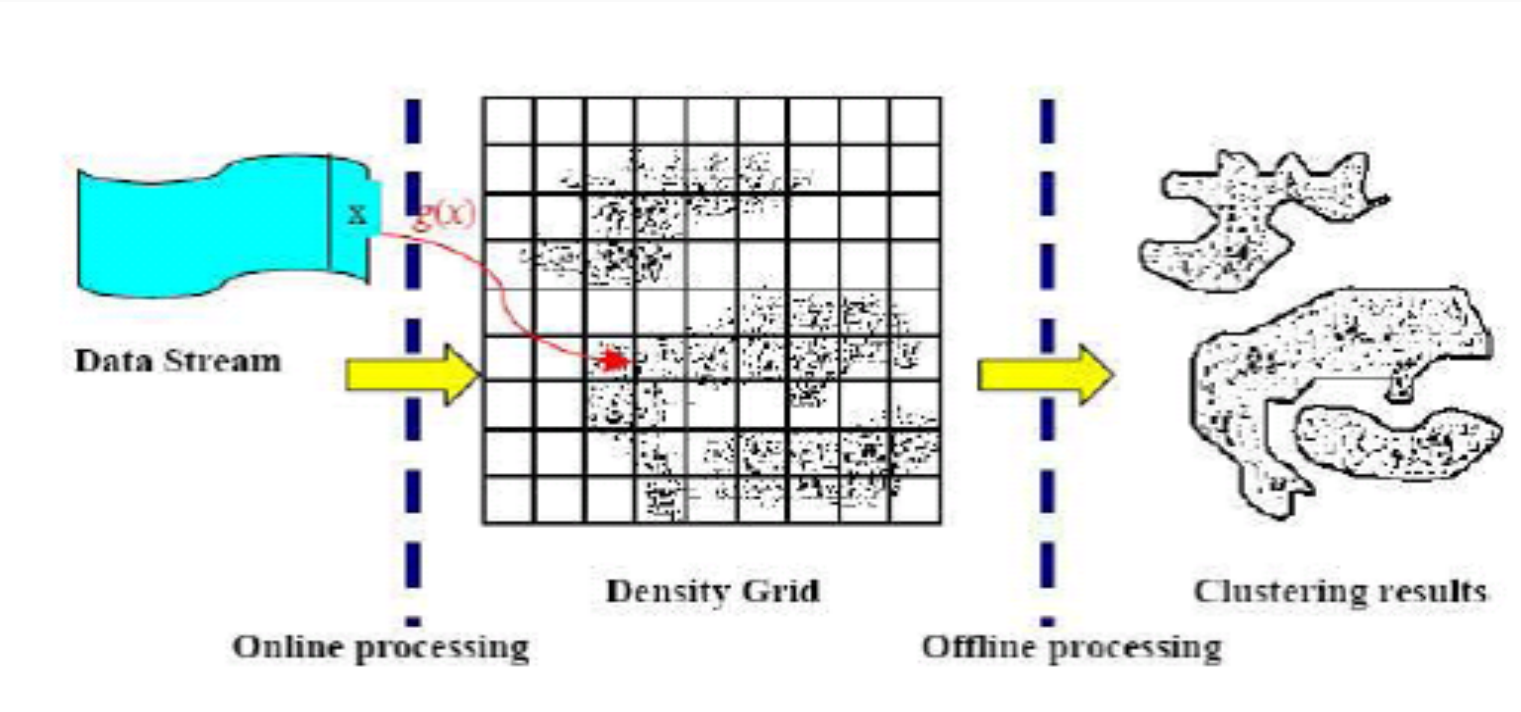
$$CF_C = (t_{update}, t_{spor}, N_C, label_C, status_C)$$

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 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



DStream — Summaries

- Three cell types are defined by parameters τ_{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

DStream — Online Phase

- 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

DenStream — Discussion

- 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 I^d)