Summarizing A 3 Way Relational Data Stream

Baptiste Csernel, 3rd year PhD Student
Fabrice Clérot, Supervisor FT R&D
Georges Hébrail, Supervisor ENST

Plan

- Problem Presentation
  - Context
  - Problematic
- Useful Tools
  - CluStream
  - Bloom Filters
- Method Presentation
  - Entity Summary
  - Relation Summary
  - Storage Management
- Work in Progress and Perspectives
Problem Presentation

- Motivation
- Context
- Problematic
- Goal

Motivations

- Data Stream processing is an ever growing preoccupation.
- For both DSMS and stream mining applications, summaries are a necessity.
- Most information is by nature, relational.
**Context**

- Data stream summaries generate a lot of interest.
- Static tables as well as data stream join evaluation are a popular subject as well.
- Single stream mining and single table mining are the norm.
- Relational stream mining is not a very active research area.

**Problematic**

Entity Stream $E_i$ of Elements $E_i$

- $E_i : (K_e, t, e_1, e_2, ..., e_p)_i$

Relation Stream $R_l$ of Elements $R_l$

- $R_l : (K_e, K_f, t, r_1, r_2, ..., r_d)_l$

Entity Stream $F_j$ of Elements $F_j$

- $F_j : (K_f, t, f_1, f_2, ..., f_q)_j$

**Additional Constraints:**

- All Streams are insert only.
- $R$ speed $<<<$ E and F speeds.
- All attributes are numerical.
- References are not broken.
Goal

- Summarizing three data streams sharing a relational link with one another.
- Building separate summaries for each entity stream, and for the relation stream.
- Summarizing the information contained in the relational links between the streams.

Useful Tools

- CluStream
  - Cluster Feature Vector (CFV)
  - SnapShot System
- Bloom Filters
Cluster Feature Vector (CFV)  

- **Structure** :
  \[(n, CF_1(t), CF_2(t), CF_1(a1), CF_2(a1), \ldots, CF_1(ad), CF_2(ad)).\]

- **With**
  - \(CF_1(ak) = \sum_{i=1}^{n} (ak_i)\)
  - \(CF_2(ak) = \sum_{i=1}^{n} (ak_i)^2\)

- **Remark**
  - Time has the same role as any other variable.

SnapShot System

- The state of the system is saved at regular time intervals

- The data structure is chosen in order to allow arithmetic operation between snapshots.

- The time at which snapshots are taken is chosen in accordance to the user’s needs.
Snapshot System:
Distribution example: $2^n$

<table>
<thead>
<tr>
<th>Order $o$</th>
<th>Snapshots</th>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>69 67 65</td>
<td>$2^1$</td>
</tr>
<tr>
<td>1</td>
<td>70 66 62</td>
<td>$2^2$</td>
</tr>
<tr>
<td>2</td>
<td>68 60 52</td>
<td>$2^3$</td>
</tr>
<tr>
<td>3</td>
<td>56 40 24</td>
<td>$2^4$</td>
</tr>
<tr>
<td>4</td>
<td>48 16</td>
<td>$2^5$</td>
</tr>
<tr>
<td>5</td>
<td>64 32</td>
<td>$2^6$</td>
</tr>
</tbody>
</table>

CluStream: Data Stream Clustering Algorithm (Aggarwal 2003)

- Algorithm based on three principles:
  - Dividing processing in two parts, an on-line part and an off-line part.
  - Creating and maintaining a large population of micro clusters.
  - Storing the state of those micro clusters with a snapshot system.
**CluStream (1/4) (on-line part)**

- **Initialization**
  - Off-line initialization of the micro clusters.

- **For each element**
  - Locate the closest micro cluster.
  - Admission test
    - If admitted, update CFV.
    - Otherwise, create a new micro cluster, and remove an outdated one.

**CluStream (2/4) (on-line part)**

- **Micro cluster removal**
  - Remove an old micro cluster.
    - (criteria based on the arrival date of the last elements)

  - If none is available, fuse the two closest micro cluster.
    - (Update the idlist of the absorbing micro cluster)
CluStream (3/4) (partie en ligne)

- **Storage**
  - Snapshot system with a distribution in $2^o$
  - Each snapshot contains
    - The CFV of each micro cluster.
    - The id list of each micro cluster.

CluStream (4/4) (off-line part)

- Use the snapshot to rebuild the stream part to be analyzed. (as a set of micro clusters)
- Apply a classic classification algorithm to the resulting set of micro clusters.
- The resulting clusters represent the final clustering of the stream.
Bloom Filters (Bloom 1970) (1/2)

- **Idea:**
  Can remember whether or not it has previously seen any number of elements.

- **Supports two operations:**
  - Learn a new element.
  - Test if an element has been previously learned or not.

**Structure:**
- A bloom filter is a simple binary word $B$ of $b$ bytes.
- At initialization, all the bytes are set to 0.

- **Learn a new element $E$:**
  - Hash $E$ to a $b$ bytes word $W_E$.
  - Set all the bytes at 1 in $W_E$ to 1 in $B$.

- **Test a new element $N$:**
  - Hash $N$ to a $b$ bytes word $W_N$.
  - If all the bytes at 1 in $W_N$ are at 1 in $B$, then, with high probability, $N$ was previously learned.
  - Otherwise, $N$ was never learned before.

- **Remark:**
  - Bloom filters are additive.
Method Presentation

- System Overview
- Entity Summary
- Relation Summary
- Storage System

System Overview

Entity Stream E
- Entity Summary Structure:
  - $N_e$ Micro Clusters
  - $N_e$ Bloom Filters

Relation Stream R
- Relation Summary Structure:
  - CFV Cross Table
  - $N_e \times N_f$ CFV Cross Table

Entity Stream F
- Entity Summary Structure:
  - $N_f$ Micro Clusters
  - $N_f$ Bloom Filters
**Entity Summary**

- Upon the arrival of each new element $E_i (K_e, t, e1, e2, \ldots, e_p)$:
  - Find the closest micro cluster.
  - Test for admission
    - If admitted:
      - Update the micro cluster CFV information.
      - Learn $K_e$ with the bloom filter attached to the micro cluster.
    - If not admitted:
      - Create a new micro cluster with $E_i$ as its seed.
      - Make room for it by fusing the two closest micro clusters.
        (this implies adding their two Bloom filters as well)

**Relation Summary**

- Upon the arrival of each new element $R_i (K_e, K_f, t, r1, r2, \ldots, r_d)$:
  - Check all the Bloom filters for $E$ to locate the one containing $K_e$. Mark its associated micro cluster $C_i$.
  - Check all the Bloom filters for $F$ to locate the one containing $K_f$. Mark its associated micro cluster $C_j$.
  - If the couple $(i,j)$ is unique, add the element $R_i$ to the CFV of indices $(i,j)$ in the CFV cross table if the couple.
Storage Management

- The storage system used is the same one as the one described in CluStream.
- All three streams are considered to share the same system clock.
- The information saved in each snapshot is:
  - For each entity:
    - The CFV and IdList of each micro cluster.
  - For the relation:
    - All the CFV matrix.

Work in Progress

- A Prototype of the algorithm already exists.
- Algorithm Testing:
  - Exploring suitable real datasets:
    - Telecommunication (services/usage/client)
    - Peer 2 Peer (documents/requests/users)
    - Airline Companies (flight/reservations/passengers)
  - Constructing an artificial dataset:
    - What kind of distribution should be used (Zipf?)
    - What kind of clusters, and what evolution for them.
  - Finding an appropriate evaluation criteria and evaluation scheme.
Conclusions and Perspectives

- This work is still in progress despite a working prototype.
- Perspectives include:
  - Extensive evaluation with real and artificial data.
  - Studying the summary querying mechanisms.
  - Extending the method to more complex data schemes (star first, then any relational type).
  - Adapting the method to deal with deletions in the streams processed.