An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System

Massimo Quadrana (UPC & Politecnico di Milano)  
Albert Bifet (Yahoo! Research) 
Ricard Gavaldà (UPC)

CCIA 2013, Vic, oct. 24th
Frequent Itemset Mining

The model
- Fix a set of possible items
- An itemset is a set of items
- A sequence of itemsets is a transaction database

The frequent itemset mining problem
Given a transaction database, find all the itemsets appearing (as a subset of) at least \( x\% \) of transactions

E.g. In a supermarket, bread, butter, and jam often bought together
\( x\% = \text{minimum support} \)
Formal Definition

Transaction database $\mathcal{D}$:

<table>
<thead>
<tr>
<th>trans. ID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abde</td>
</tr>
<tr>
<td>2</td>
<td>bce</td>
</tr>
<tr>
<td>3</td>
<td>abde</td>
</tr>
<tr>
<td>4</td>
<td>abce</td>
</tr>
<tr>
<td>5</td>
<td>abcde</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
</tr>
</tbody>
</table>

- Let $\mathcal{I}$ be the set of items and $\mathcal{T}$ be the set of transactions.
- A set $X = \{X_1, \ldots, X_n\}$, $X \subseteq \mathcal{I}$ is called an itemset.
- The fraction of transactions in $\mathcal{D}$ that contain $X$ is called its support.

$\text{support(ab)} = \frac{4}{6}$, $\text{support(bcd)} = \frac{2}{6}$
Examples of Application

- Market Basket Analysis: Placement in shelves, pricing policies
- Click-streams in web pages
- Credit card bank fraud detection
- Real-time failure detection in sensor networks
Data arrive as a stream of itemsets at high speed

Can’t store all of it, not even in secondary memory

Each itemset can be processed once

Needs to provide accurate answers at all times

Data distribution evolves over time: Concept drift

Mined itemsets must be created, revised, possibly dropped
Goal of this project

A robust, efficient algorithm for frequent itemset mining on streams

- Publicly available
- Usable for practical applications
- Reference for future research
Massive Online Analysis (MOA)

Open-source environment for stream mining
http://moa.cms.waikato.ac.nz/

- Closely related to WEKA, also by U. of Waikato, New Zealand
- Java for portability and extendability
- Command line, GUI, and API interfaces
- Several classification and clustering algorithms over data streams
- No frequent pattern mining capabilities
Frequent Closed Itemsets

Definition A frequent itemset $X$ is closed if it has no frequent superset with the same support.

For example, for $\minsupp = 3/6$,

<table>
<thead>
<tr>
<th>trans. ID</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>abde</td>
</tr>
<tr>
<td>2</td>
<td>bce</td>
</tr>
<tr>
<td>3</td>
<td>abde</td>
</tr>
<tr>
<td>4</td>
<td>abce</td>
</tr>
<tr>
<td>5</td>
<td>abcde</td>
</tr>
<tr>
<td>6</td>
<td>bcd</td>
</tr>
</tbody>
</table>

- $abde$ is a frequent closed itemset (support = 3)
- $abd$ is frequent, but not closed ($abde$ has the same support)
Closed itemsets are a *complete* and *non-redundant* representation

- *Compact* representation
- Reconstruct the support information of every itemset (also frequent)
- Less itemsets in output
- Save *memory* and *computations* in Frequent Itemset mining!!!
Algorithms considered

Restricted to frequent closed itemset stream miners

**Exact**
- MOMENT [Chi+ 06], NEWMOMENT [Li+ 09],
- CLOSTREAM [Yen+ 11]

High computational cost for exactness

**Approximate**
- IncMine [Cheng+ 08], CLAIM [Song+ 07]

Maybe more efficient at the expense of false positives and/or negatives
The IncMine Algorithm [Cheng, Ke, Ng 08]

Some features:
- Approximate algorithm, controlled by relaxation parameter
- Drops non-promising itemsets: may have false negatives
- Inverted FCI index to keep updated itemsets within window
- Requires a batch method for finding FCI in new batch
  → we chose CHARM [Zaki+ 02]
Accuracy

Precision and recall w.r.t. exact ECLAT [Zaki 00]
T40I10D100K dataset. Sliding window of size $10 \times$ and 500 trans./batch

Figure: Fixed $\minsup$. Variable relaxation rate

Figure: Variable $\minsup$. Fixed relaxation rate
Throughput

Average number of transactions processed per second
IncMine (Java) is compared with MOMENT (C++)

Figure: Fixed $\text{min supp}$. Variable relaxation rate

Figure: Variable $\text{min supp}$. Fixed relaxation rate
## Memory usage

- Average memory consumption of the JVM
- Garbage collector skews results (no comparison with MOMENT)
- Lower \textit{minsupp}, higher memory usage
- Larger window size, higher memory usage
- Static frequent closed itemset mining in batches is the most memory intensive task

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>Total Memory Usage (MB)</th>
<th>Data Structures Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>225.2</td>
<td>23.1</td>
</tr>
<tr>
<td>0.04</td>
<td>226.6</td>
<td>3.1</td>
</tr>
<tr>
<td>0.06</td>
<td>217.8</td>
<td>0.9</td>
</tr>
<tr>
<td>0.08</td>
<td>198.3</td>
<td>0.5</td>
</tr>
<tr>
<td>0.10</td>
<td>187.2</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Concept Drift

**Concept** Quantity we are going to mine (target variable)

**Drift** Change over time in unforeseen ways

Usually concept drifts are classified in:

- Sudden, or abrupt, drifts
- Gradual drifts

Drift detected monitoring:

- The total number of frequent itemsets (in *synthetic* data streams)
- The number of added/removed frequent itemsets (in *real* data streams)
Introducing Concept Drift

Given two concepts (streams), to introduce the drift we use a sigmoid probability function.

\[
 f(t) = \frac{1}{1 + e^{-s(t-t_0)}}
\]

Probability that a new instance of the stream belongs to the second concept.

- \(t_0\) is the point of change
- \(s = 4/L\), where \(L\) is the length of the change
T40I10kD1MP6 drifts to T50I10kD1MP6C05 dataset (Zaki’s IBM Datagen Software).

Reaction time grows linearly with window size.
Reaction to Gradual Drift

- **Fast reaction** with small windows
- **Stable response** with big windows
Analyzing MOVIELENS (I)

About 10 million ratings over 10681 movies by 71567 users

- Static data set for *movie rating* (from 29 Jan 1996 to 15 Aug 2007)
- Movies grouped by rating time (every 5 minutes)
- Transactions passed in ascending time to create a *stream*
- Stream of 620,000 transactions with average length 10.4

Results:

- Evolution of popular movies over time
- Unnoticed with static dataset analysis
Analyzing MOVIELENS (II)

<table>
<thead>
<tr>
<th>date</th>
<th>Frequent Itemsets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Minority Report (2002); Signs (2002).</td>
</tr>
</tbody>
</table>
Conclusions

- Perfect integration with MOA
- Good accuracies and performances compared with MOMENT
- Good throughput and reasonable memory consumption
- Good adaptivity to concept drift
- Usable in real contexts
Future Works

- Bypass memory consumption of frequent closed itemset batch mining
- **Self-adaption**: a general problem in Data Mining
- ADWIN [Bifet 07] to control window size
An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System

Massimo Quadrana (UPC & Politecnico di Milano)
Albert Bifet (Yahoo! Research)
Ricard Gavaldà (UPC)

CCIA 2013, Vic, oct. 24th