Mining Correlated High-Utility Itemsets using the Bond Measure

Philippe Fournier-Viger\textsuperscript{1}, Jerry Chun-Wei Lin\textsuperscript{2}, Quang-Huy Duong\textsuperscript{3}, Thu-Lan Dam\textsuperscript{3,4},

\textsuperscript{1} School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, China
\textsuperscript{2} School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China
\textsuperscript{3} Faculty of Information Technology, Ho Chi Minh city Industry and Trade College
\textsuperscript{4} Faculty of Information Technology, University of Science, Vietnam
philippe.fournier-viger@umoncton.ca, jerrylin@ieee.org, duytai@fit-hitu.edu.vn, lhbac@fit.hcmus.edu.vn
Topic of this talk

• The problem of discovering interesting and useful patterns in databases.
• In particular, extensions of the problem of high-utility itemset mining
What is a transaction database?

• Let be a set of items \{a, b, c, d, e, ...\} sold in a store.

• A *transaction* is a set of items bought by a customer.

• Example:

<table>
<thead>
<tr>
<th>Transaction</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_1</td>
<td>{a, b, c, d, e}</td>
</tr>
<tr>
<td>T_2</td>
<td>{a, b, e}</td>
</tr>
<tr>
<td>T_3</td>
<td>{c, d, e}</td>
</tr>
<tr>
<td>T_4</td>
<td>{a, b, d, e}</td>
</tr>
</tbody>
</table>

four transactions
Discovering Frequent Patterns

• The task of frequent pattern mining was proposed by Agrawal (1993).

• Input: a transaction database and a parameter $\text{minsup} \geq 1$.

• Output: the frequent itemsets (all sets of items appearing in at least $\text{minsup}$ transactions).
An example

**transaction database**

<table>
<thead>
<tr>
<th>Transaction</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>{a, b, c, d, e}</td>
</tr>
<tr>
<td>T₂</td>
<td>{a, b, e}</td>
</tr>
<tr>
<td>T₃</td>
<td>{c, d, e}</td>
</tr>
<tr>
<td>T₄</td>
<td>{a, b, d, e}</td>
</tr>
</tbody>
</table>

**minsup = 2**

**frequent itemsets**

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{e}</td>
<td>4</td>
</tr>
<tr>
<td>{d, e}</td>
<td>3</td>
</tr>
<tr>
<td>{b, d, e}</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
How to solve this problem?

The naïve approach:

– scan the database to count the frequency of each possible itemset.

eg.: \{a\}, \{a,b\}, \{a,c\}, \{a,d\}, \{a,e\}, \{a,b,c\}, \{a,b,d\} … \{b\}, \{b, c\}, … \{a,b,c,d,e\}

– If \(n\) items, then \(2^n - 1\) possible itemsets.

– Thus, inefficient.

• Several efficient algorithms:

  – Apriori, FPGrowth, H-Mine, LCM, etc.
The “Apriori” property

Property (anti-monotonicity).
Let be itemsets $X$ and $Y$. If $X \subseteq Y$, then the support of $Y$ is less than or equal to the support of $X$.

Example

<table>
<thead>
<tr>
<th>Transaction</th>
<th>items</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>{a, b, c, d, e}</td>
</tr>
<tr>
<td>$T_2$</td>
<td>{a, b, e}</td>
</tr>
<tr>
<td>$T_3$</td>
<td>{c, d, e}</td>
</tr>
<tr>
<td>$T_4$</td>
<td>{a, b, d, e}</td>
</tr>
</tbody>
</table>

The support of \{a,b\} is 3.
Thus, supersets of \{a,b\} have a support \leq 3.
Limitations of frequent patterns

• **Frequent pattern mining** has many applications.

• However, it has important **limitations**
  – many frequent patterns are not interesting,
  – quantities of items in transactions must be 0 or 1
  – all items are considered as equally important (having the same weight)
High Utility Itemset Mining

• A generalization of frequent pattern mining:
  – items can appear more than once in a transaction
    (e.g. a customer may buy 3 bottles of milk)
  – items have a unit profit
    (e.g. a bottle of milk generates 1 $ of profit)
  – the goal is to find patterns that generate a high profit

• Example:
  – \{caviar, wine\} is a pattern that generates a high profit, although it is rare
High-utility itemset mining

**Input**

- a transaction database
- a unit profit table

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$(a, 1), (b, 5), (c, 1), (d, 3), (e, 1), (f, 5)$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$(b, 4), (c, 3), (d, 3), (e, 1)$</td>
</tr>
<tr>
<td>$T_3$</td>
<td>$(a, 1), (c, 1), (d, 1)$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$(a, 2), (c, 6), (e, 2), (g, 5)$</td>
</tr>
<tr>
<td>$T_5$</td>
<td>$(b, 2), (c, 2), (e, 1), (g, 2)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
<th>$e$</th>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profit</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**minutil**: a minimum utility threshold set by the user (a positive integer)
High-utility itemset mining

**Input**

- a transaction database
- a unit profit table

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$(a, 1), (b, 5), (c, 1), (d, 3), (e, 1), (f, 5)$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$(b, 4), (c, 3), (d, 3), (e, 1)$</td>
</tr>
<tr>
<td>$T_3$</td>
<td>$(a, 1), (c, 1), (d, 1)$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$(a, 2), (c, 6), (e, 2), (g, 5)$</td>
</tr>
<tr>
<td>$T_5$</td>
<td>$(b, 2), (c, 2), (e, 1), (g, 2)$</td>
</tr>
</tbody>
</table>

**Output**

- All high-utility itemsets (itemsets having a utility $\geq \text{minutil}$)
- For example, if $\text{minutil} = 33\$, the high-utility itemsets are:

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Utility</th>
<th>Transactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>${b,d,e}$</td>
<td>36$</td>
<td>2 transactions</td>
</tr>
<tr>
<td>${b,c,d}$</td>
<td>34$</td>
<td>2 transactions</td>
</tr>
<tr>
<td>${b,c,d,e}$</td>
<td>40$</td>
<td>2 transactions</td>
</tr>
<tr>
<td>${b,c,e}$</td>
<td>37 $</td>
<td>3 transactions</td>
</tr>
</tbody>
</table>

$\text{minutil}$: a minimum utility threshold set by the user (a positive integer)
Utility calculation

The utility of the itemset \{b,d,e\} is calculated as follows:

\[
u(\{b,d,e\}) = (5 \times 2) + (3 \times 2) + (3 \times 1) + (4 \times 2) + (2 \times 3) + (1 \times 3) = 36 \$\]
How to solve this problem?

• **Several algorithms:**
  – Two-Phase (PAKDD 2005),
  – IHUP (TKDE, 2010),
  – UP-Growth (KDD 2011),
  – HUI-Miner (CIKM 2012),
  – FHM (ISMIS 2014)
  – EFIM (2015)
  – mHUIIMiner (2017)

• **Key idea:** calculate an upper-bound on the utility of itemsets (e.g. the **TWU**) that respects the Apriori property to be able to prune the search space.
Problem

High-utility itemset mining
  – is **useful** for discovering **profitable itemsets**.
  – but **may discover many itemsets that are weakly correlated**
  – e.g. **bread with caviar has a high profit**

We propose a new type of patterns:
How to detect if items are correlated?

Several approaches:

• using **statistical tests** to find productive itemsets (Webb et al., 2010)
• the **affinity** measure (Ahmed et al. 2011)
• the **bond** measure (Bouasker et al., 2015)
The *bond* of an itemset

- The **conjunctive support** of an itemset $X$ in a database is the number of transactions that contains $X$.

- The **disjunctive support** of an itemset $X$ in a database is the number of transactions that contains any items from $X$.

- The **bond** of an itemset $X$ is defined as:
  $$\text{bond}(X) = \frac{\text{conj\_sup}(X)}{\text{disj\_sup}(X)}.$$
Problem definition

• Discovering all correlated high utility itemsets, that is itemsets:
  – having a utility no less than a threshold \textit{minutil}
  – having a bond no less than a threshold \textit{minbond}
For example, if \( \text{minutil} = 30 \) and \( \text{minbond} = 0.5 \), correlated high utility itemsets are:

\[
\begin{align*}
\{b, d\} & \quad \text{bond} = \frac{2}{4} = 0.5 \\
\{b, e\} & \quad \text{bond} = \frac{3}{4} = 0.75 \\
\{b, c, e\} & \quad \text{bond} = \frac{3}{5} = 0.6
\end{align*}
\]
The FCHM algorithm

- An algorithm for mining correlated high utility-itemsets
- It performs a depth-first search.
- It prune the search space using the bond and utility measures.
- **Key challenge:** how to calculate the bond of an itemset
The FCHM algorithm (cont’d)

- The algorithm is inspired by the FHM algorithm for high-utility itemset mining.
- **FCHM** annotates each itemset with a structure called utility-list.
- Moreover each itemset \( X \) is annotated with a **conjunctive bitvector** that stores the union of all items in \( X \)
  
  e.g. the conj. bitvector of \{a,c\} is \( T_1, T_3, T_5, T_6 \) 101011
  
  the conj. bitvector of \{d\} is \( T_3, T_4, T_5 \) 001110
  
  the conj. bitvector of \{a,c,d\} is \( T_1, T_3, T_4, T_5, T_6 \) 101111

The conjunctive bitvector of an itemset can be obtained by performing the union of the bitvectors of some of its subsets.
Additional optimizations

Strategy 1. Directly Outputting Single items (DOS). A first optimization is based on the observation that the bond of single items is always equal to 1. Thus, HUIs containing a single item can be directly output without calculating their bond.

Strategy 2. Pruning Supersets of Non correlated itemsets (PSN). Because the bond measure is anti-monotonic (Property 5), if the bond of an itemset $P_{xy}$ is less than $minbond$, any extensions of $P_{xy}$ should not be explored. Thus, Line 8 of Algorithm 1 is modified so that if $bond(P_{xy}) < minutil$ for an itemset $P_{xy}$, the utility-list of $P_{xy}$ is not added to the set $ExtensionsOfP_x$.

Strategy 3. Pruning using the Bond Matrix (PBM). The third strategy is introduced to avoid constructing the utility-list of an itemset $P_{xy}$, and prune all its extensions. During the second database scan, a novel structure named Bond Matrix is created to store the bond of all itemsets containing two items from $I^*$. The design of this structure is similar to the EUCS. The Bond Matrix is formally defined as follows. The Bond Matrix is a set of triples of the form $(a, b, c) \in I^* \times I^* \times \mathbb{R}$. A triple $(a,b,c)$ indicates that $bond(\{a,b\}) = c$. Note that it only stores tuples of the form $(a, b, c)$ such that $c \neq 0$. For example, the Bond Matrix for the running example is shown in Fig. 2.

Then, Line 5 of Algorithm 1 is modified to add the condition that the utility-list of $P_{xy}$ should only be built if $\exists (x, y, c) \in BondMatrix$ such that $c \geq minbond$. It can be easily seen that this pruning condition preserve the correctness and completeness of FCHM, because any superset of an itemset $\{x, y\}$ such that $bond(\{x, y\}) < minbond$ is not a CHIs by Property 5.
Strategy 4. Abandoning Utility-List construction early (AUL). The fourth strategy introduced in FCHM is to stop constructing the utility-list of an itemset if some specific condition is met, indicating that the itemset may not be a CHI. The strategy is based on the following novel observation:

Property 6 (Required conjunctive support for an extension \( P_{xy} \)). An itemset \( P_{xy} \) is a correlated itemset only if its conjunctive support is no less than the value \( \text{lowerBound}(P_{xy}) = \lceil |\text{consup}(P_{xy})| \times \text{minBond} \rceil \) transactions.

This property is directly derived from the definition of the bond measure, and proof is therefore omitted. Now, consider the construction of the utility-list of an itemset \( P_{xy} \) by the Construct procedure (Algorithm 3). As mentioned, a first modification to this procedure is to create the disjunctive bit vector of \( P_{xy} \) by performing the OR operation with the bit vectors of \( P_x \) and \( P_y \) before Line 1. This allows obtaining the value \( \text{consup}(P_{xy}) = |bv(P_{xy})| \). A second modification is to create a variable \( \text{maxSupport} \) that is initialized to the conjunctive support of \( P_x \) \( (\text{consup}(P_x) = |bv(P_x)|) \), before Line 1. The third modification is to the following lines, where the utility-list of \( P_{xy} \) is constructed by checking if each tuple in the utility-lists of \( P_x \) appears in the utility-list of \( P_y \) (Line 3). For each tuple not appearing in \( P_y \), the variable \( \text{maxSupport} \) is decremented by 1. If \( \text{maxSupport} \) is smaller than \( \text{lowerBound}(P_{xy}) \), the construction of the utility-list of \( P_{xy} \) can be stopped because the conjunctive support of \( P_{xy} \) will not be higher than \( \text{lowerBound}(P_{xy}) \). Thus \( P_{xy} \) is not a CHI by Property 6, and its extensions can also be ignored by Property 5. As it will be shown in the experiment, this strategy is very effective, and decrease execution time and memory usage by stopping utility-list construction early. A similar pruning strategy based on the utility of itemset \( P_{xy} \) instead of the bond measure is also integrated in FCHM. This strategy is called the LA-prune strategy and was introduced in HUP-Miner [8], and it is thus not described here.
## Experimental Evaluation

### Datasets’ characteristics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>transaction count</th>
<th>distinct item count</th>
<th>average transaction length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retail</td>
<td>88,162</td>
<td>16,470</td>
<td>10.30</td>
</tr>
<tr>
<td>Kosarak</td>
<td>990,000</td>
<td>41,270</td>
<td>8.09</td>
</tr>
<tr>
<td>Foodmart</td>
<td>1,559</td>
<td>4,141</td>
<td>4.4</td>
</tr>
<tr>
<td>Mushroom</td>
<td>8,124</td>
<td>120</td>
<td>23</td>
</tr>
</tbody>
</table>

**Retail, Foodmart** are real-life transaction datasets from retail stores.

**Mushroom** is a dense dataset with long transactions

**Kosarak** is a large sparse dataset
Experimental Evaluation

- **FCHM** was run with five different minbond threshold values (0.1, 0.3, 0.5, 0.7 and 0.9).
- It was compared with the **FHM** algorithm for high utility itemset mining.
The FCHM algorithm can filter many non correlated patterns.
Execution times

The FCHM algorithm can be up to 100 times faster than FHM
Other results

Memory consumption was also compared, although detailed results are not shown as a figure due to space limitation. It was observed that FCHM can use up to 30 times less memory than FHM depending on how the minbond threshold is set. For example, on mushroom and minutil = 5,000,000, FHM, FCHM0.9, FCHM0.7, FCHM0.5, FCHM0.3, and FCHM0.1, respectively consume 666 MB, 424 MB, 136 MB, 59 MB, 26 MB and 20 MB.

Lastly, the efficiency of the proposed PBM and AUL strategies was also assessed. Detailed results are not shown due to space limitation. But these strategies were shown to prune a huge amount of candidates for most datasets, and it greatly increases when minbond is increased. For example, on mushroom and minutil = 5,000,000, FHM, FCHM0.9, FCHM0.7, FCHM0.5, FCHM0.3, and FCHM0.1, visit 1,282,377, 536,136, 5,254, 1,160, 883, and 848 candidates.
Conclusion

• Contributions:
  ➢ New type of pattern: correlated high-utility itemsets
  ➢ A novel algorithm, named FCHM with novel pruning strategies
  ➢ Experimental results:
    ➢ FCHM eliminate a large number of non correlated patterns.
    ➢ Can be much faster than FHM in many cases.

• Source code and datasets available as part of the SPMF data mining library (GPL 3).

Open source Java data mining software, 130 algorithms
http://www.phillippe-fournier-viger.com/spmf/
Thank you. Questions?

Open source Java data mining software, 133 algorithms

http://www.phillippe-fournier-viger.com/spmf/
Introduction

SPMF is an open-source data mining library written in Java, specialized in pattern mining.

It is distributed under the GPL v3 license.

It offers implementations of 120 data mining algorithms for:

- association rule mining,
- itemset mining,
- sequential pattern mining,
- sequential rule mining,
- sequence prediction,
- periodic pattern mining,
- high-utility pattern mining,
- clustering and classification

The source code of each algorithm can be easily integrated in other Java software.

Moreover, SPMF can be used as a standalone program with a simple user interface or from the command line.

SPMF is fast and lightweight (no dependencies to other libraries).

The current version is v0.99j and was released the 16th June 2016.

http://www.philippe-fournier-viger.com/spmf/
Running an algorithm

Choose an algorithm: CM-SPAM
Choose input file: snake_192_converted.txt
Set output file: test.txt
Choose minsup (%): 0.98 (e.g. 0.5 or 50%)
Min pattern length (optional): 4 (e.g. 1 items)
Max pattern length (optional): (e.g. 10 items)
Max gap (optional):
Required items (optional): (e.g. 1,2,3)
Show sequence ids? (optional): (default: false)
Open output file: using SPMF viewer

Algorithm is running...
CM-SPAM v0.97 - STATISTICS
Frequent sequences count: 447
Total time: ~135 ms
Max memory (MB): 39.53382110595703447
mnsup 157
Intersection count 2141

Discovered patterns

http://www.philippe-fournier-viger.com/spmf/
Time series visualization

Cluster visualization

http://www.philippe-fournier-viger.com/spmf/
CALL FOR CHAPTERS

High-Utility Pattern Mining: Theory, Algorithms and Applications

Editors: Philippe Fournier-Viger, Chun-Wei Lin, Roger Nkambou, Bay Vo

An edited book to be published by Springer in 2018

- Chapter proposal deadline: 1st October 2017
- Proposal acceptance date: 10th October 2017
- Full chapter submission deadline: 15th January 2017
- Planned publication date: 1st July 2018

Introduction
High utility pattern mining is a popular and emerging research area in the field of data mining, which consists of discovering patterns of high importance in databases. The importance of patterns can be expressed in terms of various criteria such as the utility, weight or importance of patterns. One of the major applications of utility mining is the analysis of customer transaction databases to discover sets of items that generate a high profit when purchased together. This book provides an introduction to this field, reviews state-of-the-art techniques, and discusses recent advances.

Book aims

- Presents an overview of the theory and core methods used in utility mining


References (cont’d)