Transaction data analysis and association rules

www.mimuw.edu.pl/~son/datamining

Nguyen Hung Son

This presentation was prepared on the basis of the following public materials:

1. Jiawei Han and Micheline Kamber, „Data mining, concept and techniques” [http://www.cs.sfu.ca]
2. Gregory Piatetsky-Shapiro, „kdnugget”, [http://www.kdnuggets.com/data_mining_course/]
Lecture plan

- Association rules
- Algorithm Apriori
- Algorithm Apriori-Tid
- FP-tree
What Is Association Mining?

- **Association rule mining:**
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.

- **Applications:**
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

- **Examples.**
  
  **Rule form:** “Body => Head [support, confidence]”.
  
  buys(x, “diapers”) => buys(x, “beers”) [0.5%, 60%]
  
  major(x, “CS”) \(\land\) takes(x, “DB”) => grade(x, “A”) [1%, 75%]
Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)

- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., 98% of people who purchase tires and auto accessories also get automotive services done

- Applications
  - * ⇒ Maintenance Agreement (What the store should do to boost Maintenance Agreement sales)
  - Home Electronics ⇒ * (What other products should the store stocks up?)
  - Attached mailing in direct marketing
  - Detecting “ping-pong”ing of patients, faulty “collisions”
Rule Measures: Support and Confidence

- Find all the rules $X \& Y \Rightarrow Z$ with minimum confidence and support
  - support, $s$, probability that a transaction contains $\{X \& Y \& Z\}$
  - confidence, $c$, conditional probability that a transaction having $\{X \& Y\}$ also contains $Z$

Let minimum support 50%, and minimum confidence 50%, we have
- $A \Rightarrow C$ (50%, 66.6%)
- $C \Rightarrow A$ (50%, 100%)
Association Rule Mining: A Road Map

- **Boolean vs. quantitative associations** (Based on the types of values handled)
  - buys(x, “SQLServer”) ^ buys(x, “DMBook”) => buys(x, “DBMiner”) [0.2%, 60%]
  - age(x, “30..39”) ^ income(x, “42..48K”) => buys(x, “PC”) [1%, 75%]

- **Single dimension vs. multiple dimensional associations** (see ex. above)

- **Single level vs. multiple-level analysis**
  - What brands of beers are associated with what brands of diapers?

- **Various extensions**
  - Correlation, causality analysis
    - Association does not necessarily imply correlation or causality
  - Maxpatterns and closed itemsets
  - Constraints enforced
    - E.g., small sales (sum < 100) trigger big buys (sum > 1,000)?
Lecture plan

- Association rules
- Algorithm Apriori
- Algorithm Apriori-Tid
- FP-tree
Mining Association Rules – An Example

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items Bought</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>A,B,C</td>
</tr>
<tr>
<td>1000</td>
<td>A,C</td>
</tr>
<tr>
<td>4000</td>
<td>A,D</td>
</tr>
<tr>
<td>5000</td>
<td>B,E,F</td>
</tr>
</tbody>
</table>

Frequent Itemset Support

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>{A}</td>
<td>75%</td>
</tr>
<tr>
<td>{B}</td>
<td>50%</td>
</tr>
<tr>
<td>{C}</td>
<td>50%</td>
</tr>
<tr>
<td>{A,C}</td>
<td>50%</td>
</tr>
</tbody>
</table>

Min. support 50%  
Min. confidence 50%

For rule $A \Rightarrow C$:

$\text{support} = \text{support} \left( \{A \, \underline{\Rightarrow} \, C \} \right) = 50\%$

$\text{confidence} = \frac{\text{support}(\{A \, \underline{\Rightarrow} \, C\})}{\text{support}(\{A\})} = 66.6\%$

The Apriori principle:

Any subset of a frequent itemset must be frequent
Possible number of rules

- Given $d$ unique items
- Total number of itemsets = $2^d$
- Total number of possible association rules:

$$R = \sum_{k=1}^{d-1} \left(\begin{array}{c} d \\ k \end{array}\right) \times \sum_{j=1}^{d-k} \left(\begin{array}{c} d-k \\ j \end{array}\right)$$

$$= 3^d - 2^{d+1} + 1$$

If $d=6$, $R = 602$ rules
How to Mine Association Rules?

- Two step approach:
  1. Generate all frequent itemsets (sets of items whose support > minsup)
  2. Generate high confidence association rules from each frequent itemset
     - Each rule is a binary partition of a frequent itemset

- Frequent itemset generation is more expensive operation.
  (There are $2^d$ possible itemsets)
Mining Frequent Itemsets: the Key Step

- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if \( \{AB\} \) is a frequent itemset, both \( \{A\} \) and \( \{B\} \) should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to \( k \) (\( k \)-itemset)
- Use the frequent itemsets to generate association rules.
Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

- Apriori principle holds due to the following property of the support measure:

  \[ \forall X, Y: (X \subseteq Y) \Rightarrow s(X) \geq s(Y) \]

  - Support of an itemset never exceeds the support of any of its subsets
  - This is known as the anti-monotone property of support
Key observation

If an itemset is infrequent, then all of its supersets must also be infrequent.

Found to be Infrequent

Pruned supersets
The Apriori Algorithm

- **Join Step**: $C_k$ is generated by joining $L_{k-1}$ with itself
- **Prune Step**: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- **Pseudo-code**:

  $C_k$: Candidate itemset of size k  
  $L_k$: frequent itemset of size k

  $L_1 = \{\text{frequent items}\};$
  
  \[
  \text{for } (k = 1; L_k \neq \emptyset; k++) \text{ do begin}
  \]
  
  $C_{k+1} = \text{candidates generated from } L_k;$
  
  \[
  \text{for each transaction } t \text{ in database do}
  \]
  
  increment the count of all candidates in $C_{k+1}$ that are contained in $t$
  
  $L_{k+1} = \text{candidates in } C_{k+1} \text{ with min\_support}$
  
  \[
  \text{end}
  \]
  
  return $\bigcup_k L_k;$
An idea of Apriori algorithm

\[ C_1 = I \]

\[ L_1 \]

\[ C_2 = \text{AprGen}(F_1) \]

\[ L_2 \]

\[ C_k = \text{AprGen}(F_{k-1}) \]

\[ L_k \]

\[ C_k \] – a set of candidates for k-frequent itemsets

\[ F_k \] – A set of k-frequent itemsets

Computing in memory
Apriori Algorithm — Example

Database D

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>1 3 4</td>
</tr>
<tr>
<td>200</td>
<td>2 3 5</td>
</tr>
<tr>
<td>300</td>
<td>1 2 3 5</td>
</tr>
<tr>
<td>400</td>
<td>2 5</td>
</tr>
</tbody>
</table>

Scan D

C1

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{4}</td>
<td>1</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

L1

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>2</td>
</tr>
<tr>
<td>{2}</td>
<td>3</td>
</tr>
<tr>
<td>{3}</td>
<td>3</td>
</tr>
<tr>
<td>{5}</td>
<td>3</td>
</tr>
</tbody>
</table>

Scan D

C2

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
<td>1</td>
</tr>
<tr>
<td>{1 3}</td>
<td>2</td>
</tr>
<tr>
<td>{1 5}</td>
<td>1</td>
</tr>
<tr>
<td>{2 3}</td>
<td>2</td>
</tr>
<tr>
<td>{2 5}</td>
<td>3</td>
</tr>
<tr>
<td>{3 5}</td>
<td>2</td>
</tr>
</tbody>
</table>

L2

<table>
<thead>
<tr>
<th>itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1 2}</td>
</tr>
<tr>
<td>{1 3}</td>
</tr>
<tr>
<td>{1 5}</td>
</tr>
<tr>
<td>{2 3}</td>
</tr>
<tr>
<td>{2 5}</td>
</tr>
<tr>
<td>{3 5}</td>
</tr>
</tbody>
</table>

Scan D

C3

<table>
<thead>
<tr>
<th>itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2 3 5}</td>
</tr>
</tbody>
</table>

L3

<table>
<thead>
<tr>
<th>itemset</th>
<th>sup.</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2 3 5}</td>
<td>2</td>
</tr>
</tbody>
</table>
How to Generate Candidates?

- Suppose the items in $L_{k-1}$ are listed in an order

- Step 1: self-joining $L_{k-1}$
  
  insert into $C_k$
  
  select $p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}$
  
  from $L_{k-1} p, L_{k-1} q$
  
  where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

- Step 2: pruning

  forall itemsets $c$ in $C_k$ do

  forall (k-1)-subsets $s$ of $c$ do

  if (s is not in $L_{k-1}$) then delete $c$ from $C_k$
Example of Generating Candidates

- \( L_3 = \{\text{abc, abd, acd, ace, bcd} \} \)
- Self-joining: \( L_3 * L_3 \)
  - \( \text{abcd} \) from \( \text{abc} \) and \( \text{abd} \)
  - \( \text{acde} \) from \( \text{acd} \) and \( \text{ace} \)
- Pruning:
  - \( \text{acde} \) is removed because \( \text{ade} \) is not in \( L_3 \)
- \( C_4 = \{\text{abcd}\} \)

- \( L_3 = \{\text{abc, abd, abe acd, ace, bcd} \} \)
- Self-joining: \( L_3 * L_3 \)
  - \( \text{abcd} \) from \( \text{abc} \) and \( \text{abd} \)
  - \( \text{abce} \)
  - \( \text{abde} \)
  - \( \text{acde} \) from \( \text{acd} \) and \( \text{ace} \)
Illustration of candidate generation

Items (1-itemsets)

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bread</td>
<td>4</td>
</tr>
<tr>
<td>Coke</td>
<td>2</td>
</tr>
<tr>
<td>Milk</td>
<td>4</td>
</tr>
<tr>
<td>Beer</td>
<td>3</td>
</tr>
<tr>
<td>Diaper</td>
<td>4</td>
</tr>
<tr>
<td>Eggs</td>
<td>1</td>
</tr>
</tbody>
</table>

Pairs (2-itemsets)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread, Milk}</td>
<td>3</td>
</tr>
<tr>
<td>{Bread, Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Bread, Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk, Beer}</td>
<td>2</td>
</tr>
<tr>
<td>{Milk, Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Beer, Diaper}</td>
<td>3</td>
</tr>
</tbody>
</table>

Minimum Support = 3

If every subset is considered,
\[ ^6C_1 + ^6C_2 + ^6C_3 = 41 \]
With support-based pruning,
\[ 6 + 6 + 2 = 14 \]

Triplets (3-itemsets)

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Bread, Milk, Diaper}</td>
<td>3</td>
</tr>
<tr>
<td>{Milk, Diaper, Beer }</td>
<td>2</td>
</tr>
</tbody>
</table>

Association rules 19
Rule generation

- Given a frequent itemset $L$, find all non-empty subsets $f \subseteq L$ such that $f \Rightarrow L - f$ satisfies the minimum confidence requirement.

- If $\{A,B,C,D\}$ is a frequent itemset, candidate rules:
  
  $ABC \Rightarrow D$, $ABD \Rightarrow C$, $ACD \Rightarrow B$, $BCD \Rightarrow A$, $A \Rightarrow BCD$, $B \Rightarrow ACD$, $C \Rightarrow ABD$, $D \Rightarrow ABC$
  
  $AB \Rightarrow CD$, $AC \Rightarrow BD$, $AD \Rightarrow BC$, $BC \Rightarrow AD$, $BD \Rightarrow AC$, $CD \Rightarrow AB$

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \Rightarrow \emptyset$ and $\emptyset \Rightarrow L$)
Rule generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an antimonotone property
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - \( L = \{A,B,C,D\} \):
    \[ c(ABC \Rightarrow D) \geq c(AB \Rightarrow CD) \geq c(A \Rightarrow BCD) \]
- Confidence is non-increasing as number of items in rule consequent increases
Lattice of rules

Association rules 22
Apriori for rule generation

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
  - $\text{join}(CD=>AB, BD=>AC)$ would produce the candidate rule $D => ABC$
  - Prune rule $D=>ABC$ if its subset $AD=>BC$ does not have high confidence
How to Count Supports of Candidates?

Why counting supports of candidates a problem?

- The total number of candidates can be very huge
- One transaction may contain many candidates

Method:

- Candidate itemsets are stored in a hash-tree
- *Leaf node* of hash-tree contains a list of itemsets and counts
- *Interior node* contains a hash table
- *Subset function*: finds all the candidates contained in a transaction
Hash tree

Hash Function

Candidate Hash Tree

\( h(a) = a \mod 3 \)
Insert a candidate to hash-tree

Candidate Hash Tree

Association rules 26
Apriori Candidate evaluation:
Finding candidates contained in transaction

TID 300

hash-tree of candidates

counter associated with each leaf node
Apriori Candidate evaluation:
Finding candidates contained in transaction

hash-tree of candidates

counter associated with each leaf node
Apriori Candidate evaluation
Finding candidates contained in transaction

hash-tree of candidates

counter associated with each leaf node
Apriori Candidate evaluation
Finding candidates contained in transaction

hash-tree of candidates
counter associated with each leaf node
Lecture plan

- Association rules
- Algorithm Apriori
- Algorithm Apriori-Tid
- FP-tree
Observations

- Apriori algorithm scans the whole database to determine supports of candidates

**Improvement:**
- Using new data structure called *counting_base* to store only those transactions which can support the actual list of candidates
- Algorithm AprioriTid
AprioriTid

**Input:** transaction data set $D$, $min\_sup$ – minimal support

**Output:** the set of all frequent itemset $F$

**Variables:** $CB_k$ - counting_base at $k^{th}$ iteration of the algorithm

1: $F_1 = \{\text{frequent 1-itemsets}\}$
2: $k = 2$;
3: **while** ($F_{k-1}$ is not empty) **do** {
4: $C_k = \text{Apriori\_generate} (F_{k-1})$;
5: $CB_k = \text{Counting\_base\_generate} (C_k, CB_{k-1})$
6: $Support\_count (C_k, CB_k)$;
7: $F_k = \{c \in C_k \mid \text{support}(c) \geq min\_support\}$;
8: **end while**
9: $F = \text{sum of all } F_k$;
AprioriTid: *Counting_base_generate*

**Step 1:**

\[\text{counting}_\text{base} = \{(r_i, S_i): r_i \text{ is the ID and } S_i \text{ is the itemset of the } i^{th} \text{ transaction}\}\]

**Step i:**

\[\text{counting}_\text{base} = \{(r, S_i): S_i \text{ is created as a joint of } S_{i-1} \text{ with } S_{i-1} \text{ as follows:} \]

\[\text{IF } \{u_1 u_2 \ldots u_{i-2} a\} \text{ and } \{u_1 u_2 \ldots u_{i-2} b\} \in S_{i-1} \text{ THEN} \]

\[\{u_1 u_2 \ldots u_{i-2} a b\} \in S_i \]

}
AprioriTid: Example

\[ D = \{(1,acd), (2, bce), (3,abce), (4,be)\}. \]

\[ \text{min}_\text{sup} = 0.5 \]

**Step 1**

\[ \text{counting\_base} = \{(1,\{a,c,d\}), (2,\{b,c,e\}), (3,\{a,b,c,e\}), (4,\{b, e\}) \} \]

\[ F_1 = \{a, b, c, e\} \]

\[ C_2 = \{ab, ac, ae, bc, be, ce\} \]

**Step 2**

\[ \text{counting\_base} = \{(1,\{ac\}), (2,\{bc,be,ce\}), (3,\{ab,ac,ae,bc,be,ce\}), (4,\{be\}) \} \]

\[ F_2 = \{ac, bc, be, ce\} \]

\[ C_3 = \{bce\} \]

**Step 3**

\[ \text{counting\_base} = \{(2,\{bce\}), (3,\{bce\})\} \]

\[ F_3 = \{bce\} \]
Is Apriori Fast Enough? — Performance Bottlenecks

- The core of the Apriori algorithm:
  - Use frequent \((k - 1)\)-itemsets to generate candidate frequent \(k\)-itemsets
  - Use database scan and pattern matching to collect counts for the candidate itemsets

- The bottleneck of Apriori: candidate generation
  - Huge candidate sets:
    - \(10^4\) frequent 1-itemset will generate \(10^7\) candidate 2-itemsets
    - To discover a frequent pattern of size 100, e.g., \(\{a_1, a_2, \ldots, a_{100}\}\), one needs to generate \(2^{100} \approx 10^{30}\) candidates.
  - Multiple scans of database:
    - Needs \((n + 1)\) scans, \(n\) is the length of the longest pattern
Algorithm AprioriHybrid

- AprioriTid replaces pass over data by pass over $TC_k$
  - effective when $TC_k$ becomes small compared to size of database

- AprioriTid beats Apriori
  - when $TC_k$ sets fit in memory
  - distribution of large itemsets has long tail

- Hybrid algorithm AprioriHybrid
  - use Apriori in initial passes
  - switch to AprioriTid when $TC_k$ expected to fit in memory
Algorithm AprioriHybrid

- Heuristic used for switching
  - estimate size of $TC_k$ from $C_k$
    - $\text{size}(TC_k) = \sum_{\text{candidates } c \in C_k} \text{support}(c) + \text{number of transactions}$
  - if $TC_k$ fits in memory and nr of candidates decreasing then switch to AprioriTid

- AprioriHybrid outperforms Apriori and AprioriTid in almost all cases
  - little worse if switch pass is last one
    - cost of switching without benefits
  - AprioriHybrid up to 30% better than Apriori, up to 60% better than AprioriTid
AprioriHybrid
Scale-up Experiment

<table>
<thead>
<tr>
<th>name</th>
<th>MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5.I2.D10M</td>
<td>239</td>
</tr>
<tr>
<td>T10.I4.D10M</td>
<td>439</td>
</tr>
<tr>
<td>T20.I6.D10M</td>
<td>838</td>
</tr>
</tbody>
</table>

Graph showing the relationship between relative time and the number of transactions (in millions) for different names (T5.I2, T10.I4, T20.I6).
Lecture plan

- Association rules
- Algorithm Apriori
- Algorithm Apriori-Tid
- FP-tree
Mining Frequent Patterns Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans

- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!
Construct FP-tree from a Transaction DB

<table>
<thead>
<tr>
<th>TID</th>
<th>Items bought</th>
<th>(ordered) frequent items</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>{f, a, c, d, g, i, m, p}</td>
<td>{f, c, a, m, p}</td>
</tr>
<tr>
<td>200</td>
<td>{a, b, c, f, l, m, o}</td>
<td>{f, c, a, b, m}</td>
</tr>
<tr>
<td>300</td>
<td>{b, f, h, j, o}</td>
<td>{f, b}</td>
</tr>
<tr>
<td>400</td>
<td>{b, c, k, s, p}</td>
<td>{c, b, p}</td>
</tr>
<tr>
<td>500</td>
<td>{a, f, c, e, l, p, m, n}</td>
<td>{f, c, a, m, p}</td>
</tr>
</tbody>
</table>

Steps:
1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree

\[ \text{min}\_\text{support} = 0.5 \]
Benefits of the FP-tree Structure

- **Completeness:**
  - never breaks a long pattern of any transaction
  - preserves complete information for frequent pattern mining

- **Compactness**
  - reduce irrelevant information—infrequent items are gone
  - frequency descending ordering: more frequent items are more likely to be shared
  - never be larger than the original database (if not count node-links and counts)
  - Example: For Connect-4 DB, compression ratio could be over 100
Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
  - Recursively grow frequent pattern path using the FP-tree

- Method
  - For each item, construct its conditional pattern-base, and then its conditional FP-tree
  - Repeat the process on each newly created conditional FP-tree
  - Until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)
Major Steps to Mine FP-tree

1) Construct conditional pattern base for each node in the FP-tree
2) Construct conditional FP-tree from each conditional pattern-base
3) Recursively mine conditional FP-trees and grow frequent patterns obtained so far
   • If the conditional FP-tree contains a single path, simply enumerate all the patterns
Step 1: From FP-tree to Conditional Pattern Base

- Starting at the frequent header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base

<table>
<thead>
<tr>
<th>Item</th>
<th>frequency</th>
<th>head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Conditional pattern bases

<table>
<thead>
<tr>
<th>item</th>
<th>cond. pattern base</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>f:3</td>
</tr>
<tr>
<td>a</td>
<td>fc:3</td>
</tr>
<tr>
<td>b</td>
<td>fca:1, f:1, c:1</td>
</tr>
<tr>
<td>m</td>
<td>fca:2, fcab:1</td>
</tr>
<tr>
<td>p</td>
<td>fcam:2, cb:1</td>
</tr>
</tbody>
</table>
Properties of FP-tree for Conditional Pattern Base Construction

- **Node-link property**
  - For any frequent item $a_i$, all the possible frequent patterns that contain $a_i$ can be obtained by following $a_i$'s node-links, starting from $a_i$'s head in the FP-tree header.

- **Prefix path property**
  - To calculate the frequent patterns for a node $a_i$ in a path $P$, only the prefix sub-path of $a_i$ in $P$ need to be accumulated, and its frequency count should carry the same count as node $a_i$. 
Step 2: Construct Conditional FP-tree

- For each pattern-base
  - Accumulate the count for each item in the base
  - Construct the FP-tree for the frequent items of the pattern base

---

**Header Table**

<table>
<thead>
<tr>
<th>Item</th>
<th>Frequency</th>
<th>Head</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

---

**m-conditional pattern base:**

- fca:2, fcab:1

**All frequent patterns concerning m**

- m,
- fm, cm, am,
- fcm, fam, cam,
- fcam

**m-conditional FP-tree**
## Mining Frequent Patterns by Creating Conditional Pattern-Bases

<table>
<thead>
<tr>
<th>Item</th>
<th>Conditional pattern-base</th>
<th>Conditional FP-tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>{(fcam:2), (cb:1)}</td>
<td>{(c:3)}</td>
</tr>
<tr>
<td>m</td>
<td>{(fca:2), (fcab:1)}</td>
<td>{(f:3, c:3, a:3)}</td>
</tr>
<tr>
<td>b</td>
<td>{(fca:1), (f:1), (c:1)}</td>
<td>Empty</td>
</tr>
<tr>
<td>a</td>
<td>{(fc:3)}</td>
<td>{(f:3, c:3)}</td>
</tr>
<tr>
<td>c</td>
<td>{(f:3)}</td>
<td>{(f:3)}</td>
</tr>
<tr>
<td>f</td>
<td>Empty</td>
<td>Empty</td>
</tr>
</tbody>
</table>
Step 3: Recursively mine the conditional FP-tree

Cond. pattern base of “am”: (fc:3)
f:3
| c:3
a:3

Cond. pattern base of “cm”: (f:3)
f:3

Cond. pattern base of “cam”: (f:3)
f:3

am-conditional FP-tree

cm-conditional FP-tree

cam-conditional FP-tree
Suppose an FP-tree T has a single path P.

The complete set of frequent patterns of T can be generated by enumeration of all the combinations of the sub-paths of P.

```
{}  
|   
| f:3  
| c:3  
| a:3  
```

All frequent patterns concerning m:

- m,
- fm, cm, am,
- fcm, fam, cam,
- fcam

*m-conditional FP-tree*
Principles of Frequent Pattern Growth

- Pattern growth property
  - Let $\alpha$ be a frequent itemset in DB, B be $\alpha$'s conditional pattern base, and $\beta$ be an itemset in B. Then $\alpha \cup \beta$ is a frequent itemset in DB iff $\beta$ is frequent in B.

- “abcdef” is a frequent pattern, if and only if
  - “abcde” is a frequent pattern, and
  - “f” is frequent in the set of transactions containing “abcde”
Why Is Frequent Pattern Growth Fast?

- Our performance study shows
  - FP-growth is an order of magnitude faster than Apriori, and is also faster than tree-projection

- Reasoning
  - No candidate generation, no candidate test
  - Use compact data structure
  - Eliminate repeated database scan
  - Basic operation is counting and FP-tree building
FP-growth vs. Apriori: Scalability With the Support Threshold

Data set T25I20D10K

Support threshold(%) vs. Run time(sec.)

- D1 FP-growth runtime
- D1 Apriori runtime

Association rules 54
FP-growth vs. Tree-Projection: Scalability with Support Threshold

Data set T25I20D100K
Some issues on association mining

- Interestingness measures
- Pattern visualization
- Multi-level association rules
- Discretization
- Mining association rules with constraints
Interestingness Measurements

- **Objective measures**
  Two popular measurements:
  - \textit{support}; and
  - \textit{confidence}

- **Subjective measures** (Silberschatz & Tuzhilin, KDD95)
  A rule (pattern) is interesting if
  - it is \textit{unexpected} (surprising to the user); and/or
  - \textit{actionable} (the user can do something with it)
Criticism to Support and Confidence

- Example 1: (Aggarwal & Yu, PODS98)
  - Among 5000 students
    - 3000 play basketball
    - 3750 eat cereal
    - 2000 both play basketball and eat cereal
  - \( \text{play basketball} \Rightarrow \text{eat cereal} \) [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.
  - \( \text{play basketball} \Rightarrow \text{not eat cereal} \) [20%, 33.3%] is far more accurate, although with lower support and confidence

<table>
<thead>
<tr>
<th></th>
<th>basketball</th>
<th>not basketball</th>
<th>sum(row)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cereal</td>
<td>2000</td>
<td>1750</td>
<td>3750</td>
</tr>
<tr>
<td>not cereal</td>
<td>1000</td>
<td>250</td>
<td>1250</td>
</tr>
<tr>
<td>sum(col.)</td>
<td>3000</td>
<td>2000</td>
<td>5000</td>
</tr>
</tbody>
</table>
Criticism to Support and Confidence (Cont.)

- Example 2:
  - X and Y: positively correlated,
  - X and Z, negatively related
  - support and confidence of X=>Z dominates

- We need a measure of dependent or correlated events

\[
corr_{A,B} = \frac{P(A \cup B)}{P(A)P(B)}
\]

<table>
<thead>
<tr>
<th>Rule</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>X=&gt;Y</td>
<td>25%</td>
<td>50%</td>
</tr>
<tr>
<td>X=&gt;Z</td>
<td>37.50%</td>
<td>75%</td>
</tr>
</tbody>
</table>

P(B | A) / P(B) is also called the lift of rule A => B
Other Interestingness Measures: Interest

- Interest (correlation, lift) \[ \frac{P(A \land B)}{P(A)P(B)} \]
  - taking both P(A) and P(B) in consideration
  - P(A \land B) = P(B) \times P(A), if A and B are independent events
  - A and B negatively correlated, if the value is less than 1; otherwise A and B positively correlated

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Support</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>X,Y</td>
<td>25%</td>
<td>2</td>
</tr>
<tr>
<td>X,Z</td>
<td>37.50%</td>
<td>0.9</td>
</tr>
<tr>
<td>Y,Z</td>
<td>12.50%</td>
<td>0.57</td>
</tr>
</tbody>
</table>
References

- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. ICDE'96, 106-114, New Orleans, LA.
References (2)

- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95, 420-431, Zurich, Switzerland.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00, 1-12, Dallas, TX, May 2000.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.
References (3)

- H. Lu, J. Han, and L. Feng. Stock movement and n-dimensional inter-transaction association rules. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'98), 12:1-12:7, Seattle, Washington.
- R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD'98, 13-24, Seattle, Washington.
References (4)

- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00, Dallas, TX, 11-20, May 2000.
- B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. ICDE'98, 412-421, Orlando, FL.
References (5)

- R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. SIGMOD’96, 1-12, Montreal, Canada.