Tutorial on Assignment 3 in Data Mining 2009

Frequent Itemset and Association Rule Mining

Gyozo Gidofalvi
Uppsala Database Laboratory
Announcements

- Updated material for assignment 3 on the lab course homepage.
- Posted sign-up sheets for labs and examinations for assignment 3 outside P1321.
- Please make sure you sign up for a slot.
  - Limited number of slot → Sign up early!
  - Tight assignment deadline? → Attend more labs!
Outline

- Association rules and Apriori-based frequent itemset mining
- Pattern growth by database projections
- Frequent itemset mining - elements of a DB-projection based implementation using Amos II
- The assignment
Association Rules and Apriori-Based Frequent Itemset Mining
Association Rules – The Basic Idea

- By examining transactions, or shop carts, we can find which items are commonly purchased together. This knowledge can be used in advertising or in goods placement in stores.

- Association rules have the general form:

\[ I_1 \rightarrow I_2 \]

where \( I_1 \) and \( I_2 \) are disjoint sets of items that for example can be purchased in a store.

- The rule should be read as:

Given that someone has bought the items in the set \( I_1 \) they are likely to also buy the items in the set \( I_2 \).
Frequent Itemsets

- **Transaction**: set of items purchased together by one customer
- **Transaction database** $D$: set of transactions
- **Itemset**: set of items
- **Support count** of an itemset $i$: number of transactions in $D$ that contain $i$, i.e., $t$ in $D$ s.t. $i$ is a subset of $t$.
- **Support** of an itemset $i$: support count of $i$ relative to $|D|$, i.e., the number of transactions in $D$

$$\text{Supp}_i = \frac{\text{Number of transactions containing } I}{\text{Total number of transactions}}$$

- An itemset $i$ is **frequent itemset** if $\text{supp}_i \geq \text{min\_supp}$. 
Finding the Frequent Itemsets

- The Brute Force Approach
  Just take all items and form all possible combinations of them and count away. Unfortunately, this will take some time...
  Given n items, how many possible itemsets are there?

- A better approach: The Apriori Algorithm
  Basic idea:
  
  An itemset can only be a frequent itemset if all its subsets are frequent itemsets
Example

- Assume that we have a transaction database with 100 transactions and we have the items \( a, b \) and \( c \). Assume that the minimum support is set to 0.60, which gives us a minimum support count of 60.

<table>
<thead>
<tr>
<th>Itemset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>{a}</td>
<td>65</td>
</tr>
<tr>
<td>{b}</td>
<td>50</td>
</tr>
<tr>
<td>{c}</td>
<td>80</td>
</tr>
</tbody>
</table>

- Since the support count of \{b\} is below the minimum support count no itemset containing \( b \) can be a frequent itemset. This means that when we look for itemsets of size 2 we do not have to look for any sets containing \( b \), which in turn leaves us with the only possible candidate \{a,c\}.
The Apriori Algorithm

A general outline of the algorithm is the following:

1. Count all itemsets of size $K$.
2. Prune the unsupported itemsets of size $K$.
3. Generate new candidates of size $K+1$.
4. Prune the new candidates based on supported subsets.
5. Repeat from 1 until no more candidates or frequent itemsets are found.
6. When all supported itemsets have been found, generate the rules.
Candidate Generation

- Assume that we have the frequent itemsets of size 2
  \{a,b\}, \{b,c\}, \{a,c\} and \{c,d\}

- From these sets we can form the candidates of size 3
  \{a,b,c\}, \{a,c,d\} and \{b,c,d\}

- But... Why not \{a,b,d\}? 
- Answer:
Candidate Pruning

- Again, assume that we have the frequent 2-itemsets
  \{a,b\}, \{b,c\}, \{a,c\} and \{c,d\}

- And the 3-candidates
  \{a,b,c\}, \{a,c,d\} and \{b,c,d\}

- Can all of these 3-candidates be frequent itemsets?
- Answer:
Find Out if the Itemset is Frequent

- When we have found our final candidates we can simply count all occurrences of the itemset in the transaction database and then remove those that have too low support count.
- If at least one of the candidates have enough support count we loop again until we can find no more candidates or frequent itemsets.
Rule Metrics

- When we have our frequent itemsets we want to form association rules from them. As we said earlier the association rule has the form

\[ I_1 \rightarrow I_2 \]

- The **support**, \( \text{Supp}_{tot} \), of the rule is the support of the itemset \( I_{tot} \) where

\[ I_{tot} = I_1 \cup I_2 \]

- The **confidence**, \( C \), of the rule is

\[ C = \frac{\text{Supp}_{tot}}{\text{Supp}_{I_1}} \]
Rule Metrics (cont.)

- How can we interpret the support and the confidence of a rule?
  - The support is how common the rule is in the transaction database.
  - The confidence is how often the left hand side of the rule is associated with the right hand side.

- In what situations can we have a rule with:
  - High support and low confidence?
  - Low support and high confidence?
  - High support and high confidence?
Pattern Growth by Database Projections
The Frequent Pattern Growth Approach to FIM

- The bottleneck of Apriori is candidate generation and testing. High cost of mining long itemsets!
- **Idea**: Instead of bottom up, in a top-down fashion extend frequent prefix by adding a single *locally* frequent item to it.
- **Question**: What does “*locally*” mean?
- **Answer**: To find the frequent itemsets that contain an item \( i \), the only transactions that need to be considered are transactions that contain \( i \).
- **Definition**: A frequent item \( i \)-related projected transaction table, denoted as \( PT_i \), contains all frequent items (larger than \( i \)) in the transactions that contain \( i \).
- Let’s look at an **example**!
### Example

**Transactions**

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 3, 4, 6, 7, 9, 13, 16</td>
</tr>
<tr>
<td>2</td>
<td>1, 2, 3, 6, 12, 13, 15</td>
</tr>
<tr>
<td>3</td>
<td>2, 6, 8, 10, 15</td>
</tr>
<tr>
<td>4</td>
<td>2, 3, 11, 16, 19</td>
</tr>
<tr>
<td>5</td>
<td>1, 3, 5, 6, 12, 13, 16</td>
</tr>
</tbody>
</table>

**Items co-occurring with item 1**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
</tr>
</tbody>
</table>

**Projected table on item 1**

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>

**Frequent items**

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>

**Relational format**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>

**Filtered transactions**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>3</td>
</tr>
</tbody>
</table>

**Items co-occurring with item 3 (and 1)**

<table>
<thead>
<tr>
<th>Item</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
</tr>
</tbody>
</table>

**Projected table on item 3 (and 1)**

<table>
<thead>
<tr>
<th>Tid</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>16</td>
</tr>
</tbody>
</table>

11/20/2009

Gyozo Gidofalvi
Discover all frequent itemsets by recursively filtering and projecting transactions in a depth-first-search manner until there are frequent items in the filtered/projected table.
Frequent Itemset Mining – Elements of a DB-Projection Based Implementation Using Amos II / AmosQL
Transactions

- Stored function (relation) to store transactions:
  ```
  create function transact(Integer tid)->Bag of Integer as stored;
  ```

- Population of the transaction table:
  ```
  add transact(1)=in({1,2,3});
  remove transact(1)=2;
  ```

- Selection of a transaction:
  ```
  transact(1);
  ```

- Transactions as a bag of tuples:
  ```
  create function transact_bt()->Bag of <Integer tid, Integer item>
  as select tid, item
  where transact(tid)=item;
  ```
Support Counting

- Function to calculate the support counts of items in bt:

```sql
create function itemsupps(Bag of <Integer, Integer> bt)
    ->Bag of <Integer/*item*/,Integer/*supp*/>
    as groupby ((select item,tid
                  from Integer item, Integer tid
                  where <tid,item> in bt),
                  #'count');
```
create function irpft(Bag of <Integer, Integer> bt,
    Integer minsupp,
    Integer pitem)
    ->Bag of <Integer, Integer>

    /* Calculates the pitem-related frequent item projection
     of the bag of transactions bt according to minsupp. */
    as select tid, item
    from Integer tid, Integer item, Integer s
    where <tid, item> in bt
        and <item, s> in freq_items(bt, minsupp)
        and tid in item_suppby(bt, pitem)
        and item > pitem;
create function FIS(Vector) -> Integer as stored;
add FIS({'beer'})=5;
add FIS({'other'})=2;
add FIS({'diapers'})=3;
add FIS({'beer','diapers'})=2;
add FIS({'beer','other'})=1;
add FIS({'beer','diapers','other'})=1;
add FIS({'diapers','other'})=1;

create function showFIS() -> Bag of <Vector, Integer>
  as select fis, supp
    from Vector fis, Integer supp
    where FIS(fis)=supp;

create function vunion(Vector v1, Vector v2) -> Vector
  as sort(select distinct I from object i
    where i in v1 or i in v2);

create function ARconf(vector prec, vector cons) -> real
  as FIS(vunion(prec,cons))/FIS(prec);

ARconf({'beer'},{'diapers'});
ARconf({'diapers'},{'beer'});
create function bproject(Bag b, Object p)->Bag
    /* Returns a bag of the elements of b that are greater than p. */
    as (select o from object o
        where o in b and o > p);

create function bsubset_tcf(Bag b, Vector rv)->Bag of <Bag, Vector>
    /* Generates all the children of a node in the subset-tree. */
    as select pb, concat(rv, {p})
        from Object p, Bag pb
        where p in b
            and pb = materialize(bproject(b,p));

create function bsubset_traverse(Bag b)->Bag of <Vector, Vector>
    /* Generates all the subsets of the elements of bag b
        by traversing the subset-tree defined by bsubset_tcf(). */
    as select vectorof(pb), rv
        from Bag pb, Vector rv
        where <pb, rv> in traverse(#'bsubset_tcf', b, {});
The Assignment
The Assignment

- Implement database projection based frequent itemset and association rule mining according to the provided skeleton (a3arm.osql) in Amos II.
- The program must run in a few minutes since we are going to run it during the examination. Too slow programs will be rejected.
- The algorithm is easy to get wrong and then you will get a super-exponential behavior that causes the execution time to blow up!
- There will be example runs on the lab course's home page. Your solution must be able to get these results before the examination.
Exercise

- Find the rules with support 0.5 and confidence 0.75 in the following database:

<table>
<thead>
<tr>
<th>TID</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a b c d}</td>
</tr>
<tr>
<td>2</td>
<td>{a c d}</td>
</tr>
<tr>
<td>3</td>
<td>{a b c}</td>
</tr>
<tr>
<td>4</td>
<td>{b c d}</td>
</tr>
<tr>
<td>5</td>
<td>{a b c}</td>
</tr>
<tr>
<td>6</td>
<td>{a b c}</td>
</tr>
<tr>
<td>7</td>
<td>{c d e}</td>
</tr>
<tr>
<td>8</td>
<td>{a c}</td>
</tr>
</tbody>
</table>
Reference and Further Reading

1. Efficient Frequent Pattern Mining in Relational Databases. X. Shang, K.-U. Sattler, and I. Geist. 5. Workshop des GI-Arbeitskreis Knowledge Discovery (AK KD) im Rahmen der LWA 2004.


5. Mining Frequent Patterns without Candidate Generation. J. Han, H. Pei, and Y. Yin. In Proceedings of the ACM-SIGMOD International Conference on Management of Data (SIGMOD'00), pp. 1-12, May 2000.