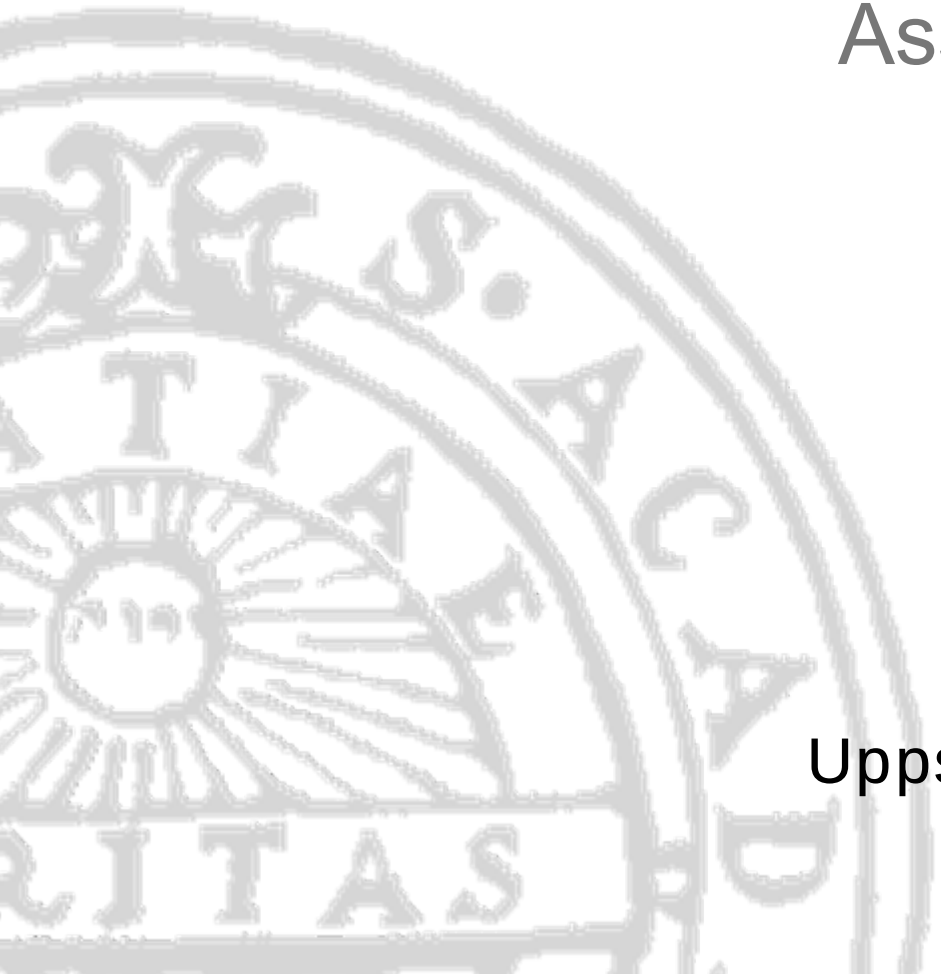
Tutorial on Assignment 3 in Data Mining 2009

Frequent Itemset and Association Rule Mining



Gyozo Gidofalvi
Uppsala Database Laboratory

UPPSALA UNIVERSITET

Announcements

- Updated material for assignment 3 on the lab course home page.
- 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!

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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.

UPPSALA UNIVERSITET

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*

$$Supp_I = \frac{Number\ of\ transactions\ containing\ I}{Total\ number\ of\ transactions}$$

- An itemset *i* is ***frequent itemset*** if $supp_i \geq$ min_supp.

UPPSALA UNIVERSITET

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*

UPPSALA UNIVERSITET

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.

Itemset	Count
{*a*}	65
{*b*}	50
{*c*}	80

- 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.**

UPPSALA UNIVERSITET

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:

UPPSALA UNIVERSITET

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:

UPPSALA UNIVERSITET

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.

UPPSALA UNIVERSITET

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***, $Supp_{tot}$, 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

$$C = \frac{Supp_{tot}}{Supp_{I1}}$$

UPPSALA UNIVERSITET

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

UPPSALA UNIVERSITET

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

TID	$Transaction$
1	$1, 3, 4, 6, 7, 9, 13, 16$
2	$1, 2, 3, 6, 12, 13, 15$
3	$2, 6, 8, 10, 15$
4	$2, 3, 11, 16, 19$
5	$1, 3, 5, 6, 12, 13, 16$

Transactions

Tid	$Item$
1	1
1	3
1	4
1	6
1	7
1	9
1	13
1	16
...	...

(a) T

Relational format

$Item$	$Count$
1	3
2	3
3	4
6	4
13	3
16	3

(b) F

Frequent items

Tid	$Item$
1	1
1	3
1	6
1	13
1	16
...	...

(c) TF

Filtered transactions

Tid	$Item$
1	3
1	6
1	13
1	16
2	2
2	3
2	6
2	13
5	3
5	6
5	13
5	16

(d) $TEMP$

Items co-occuring with item 1

$Item$	$Count$
3	3
6	3
13	3

(e) F

Frequent items

Tid	$Item$
1	3
1	6
1	13
2	3
2	6
2	13
5	3
5	6
5	13

(f) PT_1

Projected table on item 1

Tid	$Item$
1	6
1	13
2	6
2	13
5	6
5	13

(h) $TEMP$

Items co-occuring with item 3 (and 1)

$Item$	$Count$
6	3
13	3

(i) F

Frequent items

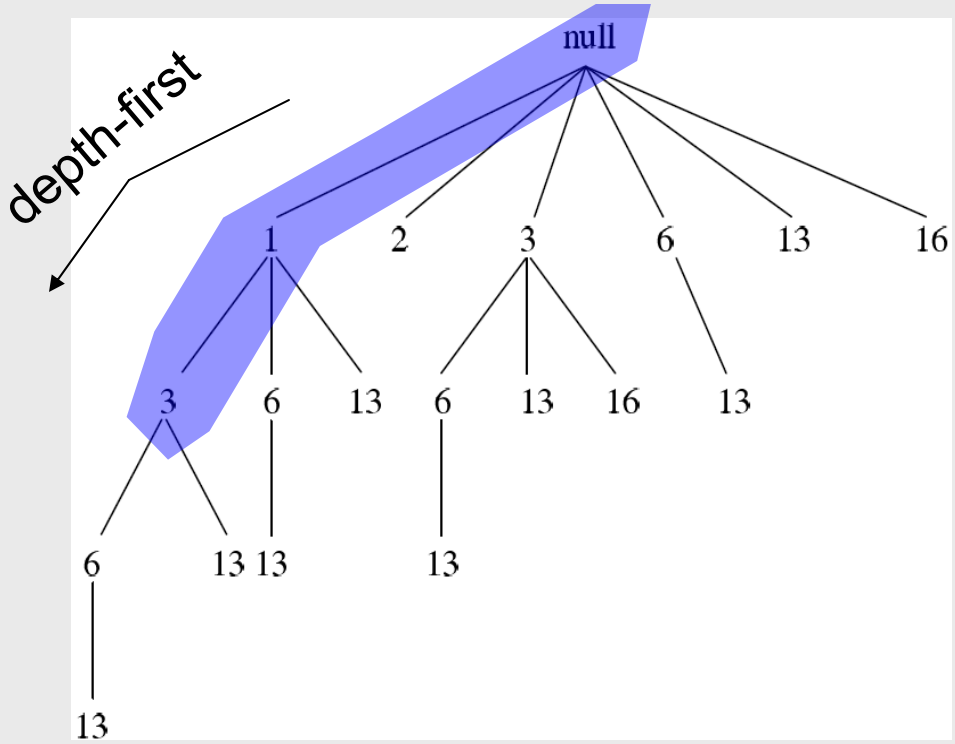
Tid	$Item$
1	6
1	13
2	6
2	13
5	6
5	13

(j) PT_1_3

Projected table on item 3 (and 1)

UPPSALA UNIVERSITET

Frequent itemset tree



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

UPPSALA UNIVERSITET

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;
```

UPPSALA UNIVERSITET

Support Counting

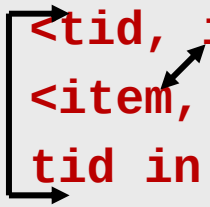
- Function to calculate the support counts of items in bt:

```
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');
```

UPPSALA UNIVERSITET

Item-Related Projection

```
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;
```



UPPSALA UNIVERSITET

Sample Association Rule DB

```
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'});
```

UPPSALA UNIVERSITET

Subset Generation

```
create function bproject(Bag b, Object p)->Bag
/* Returns a bag of the elements of b that are grater 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.

UPPSALA UNIVERSITET

Exercise

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

TID	*Transaction*
1	{*a b c d*}
2	{*a c d*}
3	{*a b c*}
4	{*b c d*}
5	{*a b c*}
6	{*a b c*}
7	{*c d e*}
8	{*a c*}

UPPSALA UNIVERSITET

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.
2. Mining Associations between Sets of Items in Large Databases. R. Agrawal, T. Imielinski, and A. Swami. In Proceedings of the ACM SIGMOD International Conference on the Management of Data, pp. 207-216, May 1993.
3. Fast Algorithms for Mining Association Rules. R. Agrawal and R. Srikant. In Proceedings of the 20th International Conference on Very Large Databases, pp. 487-499, September 1994.
4. An Effective Hash Based Algorithm for Mining Association Rules. J. S. Park, M.-S. Chen, and P. S. Yu. In Proceedings of the ACM SIGMOD International Conference on the Management of Data, pp. 175-186, May 1995.
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.