



## Mining Associations from Large Databases – Part 2

## Outline

- Algorithm
- Large example (largeExample.xls)
- Choosing rules
- Efficiency of Apriori
- Possible Improvements

## Apriori Algorithm

```

(1) L1 = find_frequent_1-itemsets(D);
(2) for (k = 2; Lk-1 ≠ ∅; k++) {
(3)   Ck = apriori_gen(Lk-1);
(4)   for each transaction t ∈ D { // scan D for counts
(5)     Ct = subset(Ck, t); // get the subsets of t that are candidates
(6)     for each candidate c ∈ Ct
(7)       c.count++;
(8)   }
(9)   Lk = { c ∈ Ck | c.count ≥ min_sup };
(10) }
(11) return L = ∪k Lk;

procedure apriori_gen(Lk-1:frequent (k-1)-itemsets)
(1) for each itemset l1 ∈ Lk-1
(2)   for each itemset l2 ∈ Lk-1
(3)     if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k-2] = l2[k-2]) ∧ (l1[k-1] < l2[k-1]) then {
(4)       c = l1 ⋈ l2; // join step: generate candidates
(5)       if has_infrequent_subset(c, Lk-1) then
(6)         delete c; // prune step: remove unfruitful candidate
(7)       else add c to Ck;
(8)     }
(9) return Ck;

procedure has_infrequent_subset(c: candidate k-itemset;
                               Lk-1: frequent (k-1)-itemsets); // use prior knowledge
(1) for each (k-1)-subset s of c
(2)   if s ∉ Lk-1 then
(3)     return TRUE;
(4) return FALSE;
  
```

## Important Details!

- How to generate candidates?
  - Step 1: self-joining  $L_{k-1}$
  - Step 2: pruning
- How to count supports of candidates?

## Generating candidates - Join

```

procedure apriori_gen(Lk-1:frequent (k-1)-itemsets)
(1) for each itemset l1 ∈ Lk-1
(2)   for each itemset l2 ∈ Lk-1
(3)     if (l1[1] = l2[1]) ∧ (l1[2] = l2[2]) ∧ ... ∧ (l1[k-2] = l2[k-2]) ∧ (l1[k-1] < l2[k-1]) then {
(4)       c = l1 ⋈ l2; // join step: generate candidates
(5)       if has_infrequent_subset(c, Lk-1) then
(6)         delete c; // prune step: remove unfruitful candidate
(7)       else add c to Ck;
(8)     }
(9) return Ck;
  
```

## Generating candidates - Join

- It is assumed that the elements (k-item sets) in  $L_{k-1}$  are sorted.
- When  $k=3$ , we want to generate 3-items set based on 2-items sets in  $L_2$ 
  - only join 2-items sets that have a common first element of the set.
  - {a,b} and {a,c} can be joined into {a,b,c}.
  - but {a,b} and {b,x} cannot be joined into {a,b,x}, why?
    - > All the elements in  $L_{k-1}$  are supposed to be sorted, hence we won't find {a,x} in  $L_{k-1}$ .
- When  $k=4$ , we want to generate 4-items set, we need to check for the first two elements.
  - Example: {a,b,c} can be joined with {a,b,x} into {a,b,c,x}.

## Generating Candidate - Pruning

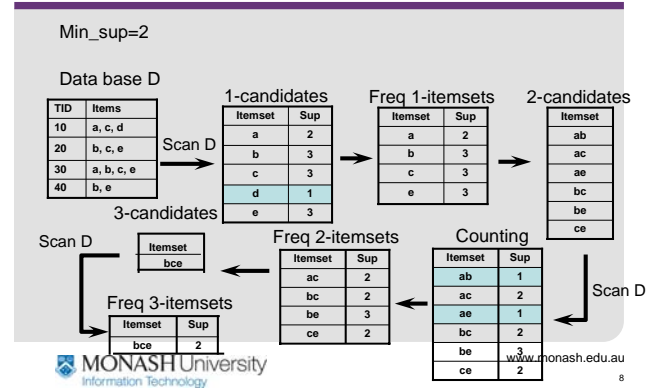
```

procedure has_infrequent_subset(c: candidate k-itemset;
    Lk-1: frequent (k-1)-itemsets); // use prior knowledge
(1) for each (k-1)-subset s of c
(2)   if s ∉ Lk-1 then
(3)     return TRUE;
(4) return FALSE;
    
```

An k-items set will be pruned if a subset of this k-items set does not exist in  $L_{k-1}$

A 3-items set will be pruned after the join operation if there is a subset of this 3-items set that is not in  $L_2$

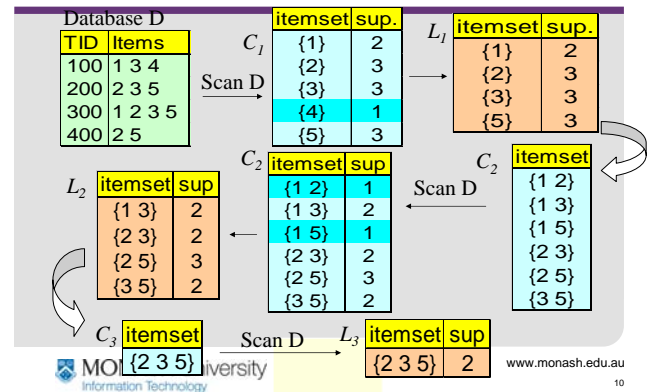
{a,b,c} will be pruned if either {a,b}, {a,c} or {b,c} is not in  $L_2$ .



## Generating Rules

- Most of association mining algorithms produce rules that have been filtered according to particular level of support and confidence.
- A low support threshold leads to large number of rules being generated.
  - Rules may need to be filtered further.
- Additional filter such as 'lift' may be used to prune the rules further.
- Some rules are redundant, hence can be removed.

## Algorithm Efficiency - Illustration



## Efficiency - Problems

- The size of candidate set, in particular,  $C_2$  can be very big for large number of items.
- Apriori requires  $n+1$  number of database scans for data set with  $n$  number of items in the largest items set.
- Rules generated can be redundant.
  - $A \rightarrow B, AC \rightarrow B$
- Assume a sparse data set
  - The number of items in each transaction is small in comparison to the total number of items.

## Efficiency - Possible Solutions

- Reduce the number of candidate item sets.
  - In particular 2-items set.
  - The size of candidate items set for 2 items is the biggest.
- Reduce the number of transactions.
  - For example removing all transactions that do not have at least two frequent items after  $L_1$  has been computed.
- Reduce the number of comparisons.
  - Use appropriate data structure so we don't need to compare every candidate against every transaction.
- Generate candidate sets efficiently.
  - It may be possible to compute  $C_k$  and from it to compute  $C_{k+1}$  without have to compute  $L_k$ .
  - Use a single pass to search both  $k$ -items sets and  $k+1$  items sets.

## Apriori-TID (1)

Database D		TID Database X <sub>1</sub>	
TID	Items	TID	Items
100	1 3 4	100	{{1},{3},{4}}
200	2 3 5	200	{{2},{3},{5}}
300	1 2 3 5	300	{{1},{2},{3},{5}}
400	2 5	400	{{2},{5}}

C <sub>1</sub>	itemset	sup.	L <sub>1</sub>	itemset	sup.
Scan X <sub>1</sub>	{1}	2	→	{1}	2
	{2}	3		{2}	3
	{3}	3		{3}	3
	{4}	1		{5}	3
	{5}	3			

## Apriori-TID (2)

C <sub>2</sub>	itemset	C <sub>2</sub>	itemset	sup.	L <sub>2</sub>	itemset	sup.
Scan X <sub>1</sub>	{1 2}	→	{1 2}	1	→	{1 3}	2
	{1 3}		{1 3}	2		{2 3}	2
	{1 5}		{1 5}	1		{2 5}	3
	{2 3}		{2 3}	2		{3 5}	2
	{2 5}		{2 5}	3			
	{3 5}		{3 5}	2			

TID	Items
100	{{1,3}}
200	{{2,3},{2,5},{3,5}}
300	{{1,3},{2,3},{2,5},{3,5}}
400	{{2,5}}

## Apriori-TID (3)

C <sub>3</sub>	itemset	L <sub>3</sub>	itemset	sup.
Scan X <sub>2</sub>	{2 3 5}	→	{2 3 5}	2

TID	Items
200	{{2,3,5}}
300	{{2,3,5}}

## Apriori-TID

- Reduce the size of transaction database.
- In most cases, the number of transactions in X<sub>n</sub> < D, except for X<sub>1</sub> which is always equal to D.
- Less total number of transactions to scan in order to generate L<sub>k</sub> from C<sub>k</sub>.

## Conclusion

- Apriori algorithm is a useful market basket analysis tool.
  - Strengths
    - > It produce clear understandable results
    - > Supports undirected data mining
    - > Works on variable length data
    - > Is simple to understand
  - Weaknesses
    - > Requires exponentially more computational effort as the problem size grows
    - > Rely on support as a filtering mechanism.
      - What support value is appropriate?
    - > It does not handle rare items well; simply considering the level of support will exclude these items