Association rule mining

- Proposed by Agrawal in 1993.
- It is an important data mining model studied extensively by the database and data mining community.
- Assume all data are categorical.
- Initially used for Market Basket Analysis to find how items purchased by customers are related

What Is Association Rule Mining?

- Frequent patterns: patterns (set of items, sequence, etc.) that occur frequently in a database
- Frequent pattern mining: finding regularities in data
  - What products were often purchased together?
  - What are the subsequent purchases after buying a car?
  - Can we automatically profile customers?

Why Essential?

- Foundation for many data mining tasks
  - Association rules, correlation, causality, sequential patterns, structural patterns, spatial and multimedia patterns, associative classification, cluster analysis…
  - Broad applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, web log (click stream) analysis.

Association Rule

- Let A={I_1,I_2,… I_m} be a set of items.
- Let T is a transaction database which contains a set of transaction where each transaction t is a set of items.
- So t is a subset of A

<u>Support</u>

- A transaction t is said to support an item li, if li is present in t.
- t is said to support a subset of items ,if t supports each item li in *x*
- An item set has a support s in T, denoted by s(*x*)$_T$, if s% of transactions in T support *x*.

Example

- Consider a set of 6 transactions of purchases of books

Say A={ANN,CC,D,TC,CG} and T= {t1,t2,t3,t4,t5,t6}

t1={ANN,CC,TC,CG}

t2={CC,D,CG}

t3={ANN,CC,TC,CG}

t4={ANN,CC,D,CG}

t5={ANN,CC,D,TC,CG}

t6={CC,D,TC}

Here t2 supports the item CC,D and CG.

The item D is supported by 4 out of 6 transactions in T

So the support of D is 66.6%

- For a given transaction database T, an association rule is an expression of the form , where x and y are subsets of A and holds with confidence τ, if τ% transactions in T that support x also support y
- The rule has support σ in the transaction set T if σ% of transaction in T support
- The left hand side is called antecedent and the right hand side is called consequent.

Example

t1={ANN,CC,TC,CG}

t2={CC,D,CG}

t3={ANN,CC,TC,CG}

t4={ANN,CC,D,CG}

t5={ANN,CC,TC,CG}

t6={CC,D,TC}

t7={TC}

What is the value of Confidence (τ) and support (σ) for CC D t1={ANN,CC,TC,CG}

t2={CC,D,CG}

t3={ANN,CC,TC,CG}

t4={ANN,CC,D,CG}

t5={ANN,CC,TC,CG}

t6={CC,D,TC}

t7={TC}

Total transaction T=7

CC present in 6 transitions

D present in 3 transactions

CC and D both present in 3 transactions

We know that Support measures how often both the item occur together as a percentage of total transaction

And Confidence measures how much a particular item is dependant on another.

So Support = 3/7 =42.8%

Confidence = 3/6=50%

Example

t1={ANN,CC,TC,CG}

t2={CC,D,CG}

t3={ANN,CC,TC,CG}

t4={ANN,CC,D,CG}

t5={ANN,CC,D,TC,CG}

t6={CC,D,TC}

Assume that σ=50% and τ=60%.

ANN CC holds.

The confidence of this rule is in fact 100%, because all the transactions that support ANN also support CC.

On the other hand , CC ANN also holds but its confidence is 66%

Example Contd..

- Let T consist of 50 transaction. 20 transactions of these contain diapers. 30 transactions contain beer. 10 transaction contain both diaper and beer.
- So support will be 2% (10/50)
- Confidence for the rule (diaper beer) will be 10/20 = 50%
- Confidence for the rule (beer diaper) will be 10/30 = 33.3%

we can say when people buy beer they also buy diapers 33.3% of the time

$$computer \Rightarrow antivirus\_software \ [support = 2\%, confidence = 60\%]$$

- Rule support and confidence are two measures of Association rule interestingness.
- They respectively reflect the usefulness and certainty of discovered rules.
- A support of 2% for Association Rule means that 2% of all the transactions under analysis show that computer and antivirus software are purchased together.
- A confidence of 60% means that 60% of the customers who purchased a computer also bought the software.
- Typically, association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold.
- Such thresholds can be set by users or domain experts.
- Rules that satisfy both a minimum support threshold (*min sup) and a minimum confidence* threshold (*min conf) are called strong.*

The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is also known, simply, as the frequency, support count, or count of the itemset

Support and Confidence for Support count

- Support count: The support count of an itemset *X*, denoted by *X.count*, in a data set *T* is the number of transactions in *T* that contain *X*. Assume *T* has *n* transactions.
- Then,

$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

Frequent Set

- Let T be the transaction database and σ be the user-specified minimum support.
- An itemset is said to be a frequent itemset in T with respect to σ, if

$s(X)_T \geq \sigma$

If we assume σ =50%, then {ANN,CC,TC}

is a frequent set as it is supported by at least 3

out of 6 transaction.

But {ANN, CC,D} is not a frequent itemset

t1={ANN,CC,TC,CG}
t2={CC,D,CG}
t3={ANN,CC,TC,CG}
t4={ANN,CC,D,CG}
t5={ANN,CC,D,TC,CG}
t6={CC,D,TC}

Problem Decomposition

In general, association rule mining can be viewed as a two-step process:

1. **Find all frequent itemsets:** By definition, each of the itemsets will occur at least as frequently as a predetermined minimum support count, *min sup.*
2. **Generate strong association rules from the frequent itemsets:** By definition, these rules must satisfy minimum support and minimum confidence.

Downward Closure property

- Any subset of a frequent set is a frequent set

Upward Closure Property

- Any superset of an infrequent set is an infrequent set

Maximal Frequent Set

- A frequent set is a maximal frequent set if it is a frequent set and no superset of this is a frequent set.

Border Set

- An itemset is a border set if it is not frequent set, but all its proper subsets are frequent sets.

EXAMPLE

- Consider the following transaction database
- Where total item set A={A_1,A_2,A_4,A_5,A_6,A_7,A_8,A_9}
- Total transaction T= {T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9,T_{10},T_{11},T_{12},T_{13},T_{14},T_{15}}
- Assume σ =20%. Since T contains 15 records, it means that an item set that is supported by at least three transaction is a frequent set.

	A1	A2	A3	A4	A5	A6	A7	A8	A9
T1	1	0	0	0	1	1	0	1	0
T2	0	1	0	1	0	0	0	1	0
T3	0	0	0	1	1	0	1	0	0
T4	0	1	1	0	0	0	0	0	0
T5	0	0	0	0	1	1	1	0	0
T6	0	1	1	1	0	0	0	0	0
T7	0	1	0	0	0	1	1	0	1
T8	0	0	0	0	1	0	0	0	0
T9	0	0	0	0	0	0	0	1	0
T10	0	0	1	0	1	0	1	0	0
T11	0	0	1	0	1	0	1	0	0
T12	0	0	0	0	1	1	0	1	0
T13	0	1	0	1	0	1	1	0	0
T14	1	0	1	0	1	0	1	0	0
T15	0	1	1	0	0	0	0	0	1

Table Sample Database

X	SUPPORT COUNT
{1}	2
{2}	6
{3}	6
{4}	4
{5}	8
{6}	5
{7}	7
{8}	4
{9}	2
{5, 6}	3
{5, 7}	5
{6, 7}	3
{5, 6, 7}	1

Table Frequent Count for Some Itemsets

- Here {1} is not frequent set with respect to σ
- {2},{3},{4},{5},{6},{7},{8},{5,6},{5,7},{6,7} are frequent set with respect to σ
- {5,6,7} is border set , because its proper subset {5,6} and {7} are frequent set.

X	SUPPORT COUNT
{1}	2
{2}	6
{3}	6
{4}	4
{5}	8
{6}	5
{7}	7
{8}	4
{9}	2
{5, 6}	3
{5, 7}	5
{6, 7}	3
{5, 6, 7}	1

Table Frequent Count for Some Itemsets

The Apriori Algorithm: Basics

The Apriori Algorithm is an influential algorithm for mining frequent itemsets for boolean association rules.

Key Concepts :

- Frequent Itemsets: The sets of item which has minimum support (denoted by L_i for i^{th}-Itemset).
- Apriori Property: Any subset of frequent itemset must be frequent.
- Join Operation: To find L_k , a set of candidate k-itemsets is generated by joining L_{k-1} with itself.

- 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.

The Apriori Algorithm : Pseudo code

- 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 = {frequent items};
for (k = 1; L_k !=∅; k++) do begin
    C_k+1 = candidates generated from L_k;
   for each transaction t in database do
          increment the count of all candidates in C_k+1
        that are contained in t
   L_k+1  = candidates in C_k+1 with min_support
   end
return ∪_k L_k;
```

The Apriori Algorithm: Example

TID	List of Items
T1	I1, I2, I5
T 2	I2, I4
T 3	I2, I3
T 4	I1, I2, I4
T 5	I1, I3
T 6	I2, I3
T 7	I1, I3
T 8	I1, I2 ,I3, I5
T 9	I1, I2, I3

- Consider a database, D , consisting of 9 transactions.
- Suppose min. support count required is 2 (i.e. min_sup = 2/9 = 22 %)
- Let minimum confidence required is 70%.
- We have to first find out the frequent itemset using Apriori algorithm.
- Then, Association rules will be generated using min. support & min. confidence.

Step 1: Generating 1-itemset Frequent Pattern

Scan D for count of each candidate

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

C_1

Compare candidate support count with minimum support count

Itemset	Sup.Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

L_1

- The set of frequent 1-itemsets, L_1 , consists of the candidate 1-itemsets satisfying minimum support.
- In the first iteration of the algorithm, each item is a member of the set of candidate.

Step 2: Generating 2-itemset Frequent Pattern

- To discover the set of frequent 2-itemsets, L_2 , the algorithm uses L_1 *Join* L_1 to generate a candidate set of 2-itemsets, C_2.
- Next, the transactions in D are scanned and the support count for each candidate itemset in C_2 is accumulated
- The set of frequent 2-itemsets, L_2 , is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

Step 2: Generating 2-itemset Frequent Pattern

Generate C_2 candidates from L_1

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

C_2

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

C_2

Compare candidate support count with minimum support count

Itemset	Sup Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

L_2

Step 3: Generating 3-itemset Frequent Pattern

- Based on the Apriori property that all subsets of a frequent itemset must also be frequent
- For example , lets take {I1, I2, I3}. The 2-item subsets of it are {I1, I2}, {I1, I3} & {I2, I3}. Since all 2-item subsets of {I1, I2, I3} are members of L_2, We will keep {I1, I2, I3} in C_3.
- Lets take another example of {I2, I3, I5} which shows how the pruning is performed. The 2-item subsets are {I2, I3}, {I2, I5} & {I3,I5}.
- BUT, {I3, I5} is not a member of L_2 and hence it is not frequent violating Apriori Property. Thus We will have to remove {I2, I3, I5} from C_3.
- Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after checking for all members of result of Join operation for Pruning.
- Now, the transactions in D are scanned in order to determine L_3, consisting of those candidates 3-itemsets in C_3 having minimum support.

Step 3: Generating 3-itemset Frequent Pattern

Scan D for count of each candidate

Itemset
{I1, I2, I3}
{I1, I2, I5}

C_3

Scan D for count of each candidate

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

C_3

Compare candidate support count with min support count

Itemset	Sup Count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

- The generation of the set of candidate 3-itemsets, C_3 , involves use of the Apriori Property.
- In order to find C_3, we compute L_2 *Join* L_2.
- C_3 = L2 *Join* L2 = {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.
- Now, Join step is complete and Prune step will be used to reduce the size of C_3. Prune step helps to avoid heavy computation due to large C_k.

Step 4: Generating 4-itemset Frequent Pattern

- The algorithm uses L_3 *Join* L_3 to generate a candidate set of 4-itemsets, C_4. Although the join results in {{I1, I2, I3, I5}}, this itemset is pruned since its subset {{I2, I3, I5}} is not frequent.
- Thus, $C_4 = \varphi$, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.
- What's Next ?

  These frequent itemsets will be used to generate strong association rules (where strong association rules satisfy both minimum support & minimum confidence).

Step 5: Generating Association Rules from Frequent Itemsets

- Procedure:
  - For each frequent itemset ***"l",*** generate all nonempty subsets of ***l.***
  - For every nonempty subset ***s*** of ***l***, output the rule **"s → (l-s)"** if **support_count(l) / support_count(s) >= min_conf** where min_conf is minimum confidence threshold.

- Back To Example:

  We had L = {{I1}, {I2}, {I3}, {I4}, {I5}, {I1,I2}, {I1,I3}, {I1,I5}, {I2,I3}, {I2,I4}, {I2,I5}, {I1,I2,I3}, {I1,I2,I5}}.
  - Lets take ***l*** = {I1,I2,I5}.
  - Its all nonempty subsets are {I1,I2}, {I1,I5}, {I2,I5}, {I1}, {I2}, {I5}.

Step 5: Generating Association Rules from Frequent Itemsets

- Let minimum confidence threshold is , say 70%.
- The resulting association rules are shown below, each listed with its confidence.
  - R1: I1 ^ I2 → I5
    - Confidence = sc{I1,I2,I5}/sc{I1,I2} = 2/4 = 50%
    - R1 is Rejected.
  - R2: I1 ^ I5 → I2
    - Confidence = sc{I1,I2,I5}/sc{I1,I5} = 2/2 = 100%
    - R2 is Selected.
  - R3: I2 ^ I5 → I1
    - Confidence = sc{I1,I2,I5}/sc{I2,I5} = 2/2 = 100%
    - R3 is Selected.

Step 5: Generating Association Rules from Frequent Itemsets

- R4: I1 → I2 ^ I5
  - Confidence = sc{I1,I2,I5}/sc{I1} = 2/6 = 33%
  - R4 is Rejected.
- R5: I2 → I1 ^ I5
  - Confidence = sc{I1,I2,I5}/{I2} = 2/7 = 29%
  - R5 is Rejected.
- R6: I5 → I1 ^ I2
  - Confidence = sc{I1,I2,I5}/ {I5} = 2/2 = 100%
  - R6 is Selected.

  In this way, We have found three strong association rules.

Partition Algorithm

- **The partition algorithm is based on the observation that the frequent sets are normally very few in number compared to the set of all itemsets.**
- **If we partition the set of transactions to smaller segments such that each segment can be accommodated in the main memory , then we can compute the set of frequent sets of each of these partitions.**
- **The partition algorithm executes in two phases.**
- **In the first phase, the partition algorithm logically divides the database into a number of non – overlapping partitions. The partitions are considered one at a time and all frequent itemsets for that partiton are generated.**
- **If there are n-partitons, phase-1 of the algorithm takes n iterations.**
- **At the end of the phase-1 , frequent itemsets are merged to generate a set of all potential frequent itemsets.**

- **In phase -2 the actual support for the itemsets are generated and the frequent itemsets are identified.**
- **The algorithm reads the entire database once du ring phase-1 and once phase-2**

Partition Algorithm

P = partition_database(T); n = Number of partitions

// Phase I

for $i = 1$ to n *do begin*

read_in_partition(T_i in P)

L^i = generate all frequent itemsets of T_i using *a priori* method in main memory.

end

// Merge Phase

for ($k = 2$; $L_k^i \neq \emptyset$, $i = 1, 2, \ldots, n$; k++) *do begin*

$$C_k^G = \bigcup_{i=1}^{n} L_i^k$$

end

// Phase II

for $i = 1$ to n *do begin*

read_in_partition(T_i in P)

for all candidates $c \in C^G$ compute $s(c)_{T_i}$

end

$L^G = \{c \in C^G \mid s(c)_{T_i} \geq \sigma\}$

Answer = L^G

EXAMPLE

- **Consider the following transaction database**
- **Where total item set A={A_1,A_2,A_4,A_5,A_6,A_7,A_8,A_9}**
- **Total transaction T= {T_1,T_2,T_3,T_4,T_5,T_6,T_7,T_8,T_9,T_{10},T_{11},T_{12},T_{13},T_{14},T_{15}}**
- **Assume σ =20%. Since T contains 15 records, it means that an item set that is supported by at least three transaction is a frequent set.**

	A1	A2	A3	A4	A5	A6	A7	A8	A9
T1	1	0	0	0	1	1	0	1	0
T2	0	1	0	1	0	0	0	1	0
T3	0	0	0	1	1	0	1	0	0
T4	0	1	1	0	0	0	0	0	0
T5	0	0	0	0	1	1	1	0	0
T6	0	1	1	1	0	0	0	0	0
T7	0	1	0	0	0	1	1	0	1
T8	0	0	0	0	1	0	0	0	0
T9	0	0	0	0	0	0	0	1	0
T10	0	0	1	0	1	0	1	0	0
T11	0	0	1	0	1	0	1	0	0
T12	0	0	0	0	1	1	0	1	0
T13	0	1	0	1	0	1	1	0	0
T14	1	0	1	0	1	0	1	0	0
T15	0	1	1	0	0	0	0	0	1

Table Sample Database

- **Let us partition T into three partition T1, T2, T3, each containing 5 transactions**
- **T1 contains transaction 1 to 5**
- **T2 contains transaction 6 to 10**
- **T3 contains transaction 11 to 15**
- **We fix the local support as equal to the given support, i.e 20%**
- **Any itemset that appears in just one of the transactions in any partition is a local frequent set in partition**

$L^1 := \{ \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{1, 5\}, \{1, 6\}, \{1, 8\}, \{2, 3\}, \{2, 4\}, \{2, 8\}, \{4, 5\}, \{4, 7\}, \{4, 8\}, \{5, 6\}, \{5, 8\}, \{5, 7\}, \{6, 7\}, \{6, 8\}, \{1,6, 8\}, \{1,5,6\}, \{1,5,8\}, \{2,4,8\}, \{4,5,7\}, \{5,6,8\}, \{5,6,7\}, \{1,5,6,8\} \}$

Similarly,

$L^2 := \{ \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{2,3\}, \{2,4\}, \{2,6\}, \{2,7\}, \{2,9\}, \{3,4\}, \{3, 5\}, \{3, 7\}, \{5, 7\}, \{6,7\}, \{6,9\}, \{7,9\}, \{2,3,4\}, \{2,6,7\}, \{2,6,9\}, \{2,7,9\}, \{3, 5, 7\}, \{2,6,7,9\} \}$

$L^3 := \{ \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{1,3\}, \{1,5\}, \{1,7\}, \{2,3\}, \{2,4\}, \{2,6\}, \{2,7\}, \{2,9\}, \{3,5\}, \{3,7\}, \{3,9\}, \{4,6\}, \{4,7\}, \{5,6\}, \{5,7\}, \{5,8\}, \{6,7\}, \{6,8\}, \{1,3,5\}, \{1,3,7\}, \{1,5,7\}, \{2,3,9\}, \{2,4,6\}, \{2,4,7\}, \{3,5,7\}, \{4,6,7\}, \{5,6,8\}, \{1, 3, 5, 7\}, \{2, 4, 6, 7\} \}$

In Phase II, we have the candidate set as

$C := L^1 \cup L^2 \cup L^3$

$C := \{ \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}, \{1,3\}, \{1, 5\}, \{1, 6\}, \{1,7\}, \{1, 8\}, \{2,3\}, \{2, 4\}, \{2,6\}, \{2,7\}, \{2, 8\}, \{2,9\}, \{3,4\}, \{3,5\}, \{3,7\}, \{3,9\}, \{4, 5\}, \{4,6\}, \{4,7\}, \{4, 8\}, \{5, 6\}, \{5,7\}, \{5, 8\}, \{5, 7\}, \{6, 7\}, \{6, 8\}, \{6,9\}, \{7,9\}, \{1,3,5\}, \{1,3,7\}, \{1,5,6\}, \{1,5,7\}, \{1,5,8\}, \{1,6,8\}, \{2,3,4\}, \{2,3,9\}, \{2,4,6\}, \{2,4,7\}, \{2,4,8\}, \{2,6,7\}, \{2,6,9\}, \{2,7,9\}, \{3,5,7\}, \{4,5,7\}, \{4,6,7\}, \{5,6,8\}, \{5,6,7\}, \{1,5,6,8\}, \{2,6,7,9\}, \{1, 3, 5, 7\}, \{2, 4, 6, 7\} \}$

- **Read the database once to compute the global support of the sets in c and get the final set of frequent sets**

Pincer-Search Algorithm

- **The pincer-search algorithm is based on the principle of bi-directional search, which takes the advantages of both bottom-up and top-down approach .**
- **In this algorithm, in each pass , in addition to counting the supports of the candidate in the bottom- up direction, it also counts the supports of the item sets of some itemsets using top-down approach. These are called Maximal Frequent Candidate Set(MFCS).**
- **This process helps in pruning the candidate sets early on in the algorithm. If we find a maximal frequent set in this process , then it is recoded in the MFCS.**

Pincer-Search Method

```
L_0 := ∅; k := 1; C_1 := {{i} | i ∈ I}; S_0 = ∅;
MFCS := {{1,2, …, n}}; MFS := ∅;
do until C_k = ∅ and S_{k-1} = ∅
        read database and count supports for C_k and MFCS;
        MFS := MFS ∪ {frequent itemsets in MFCS};
        S_k := {infrequent itemsets in C_k};
        call MFCS-gen algorithm if S_k ≠ ∅;
        call MFS-pruning procedure;
        generate candidates C_{k+1} from C_k; (similar to a priori's generate & prune)
        if any frequent itemset in C_k is removed in MFS-pruning procedure
            call the recovery procedure to recover candidates to C_{k+1};
        call MFCS prune procedure to prune candidates in C_{k+1};
        k := k+1;
return MFS
```

MFCS-gen

```
for all itemsets s ∈ S_k
        for all itemsets m ∈ MFCS
         if s is a subset of m
         MFCS := MFCS\{m};
         for all items e ∈ itemset s
          if m\{e} is not a subset of any itemset in MFCS
                MFCS := MFCS ∪ {m\{e}};
return MFCS
```

Recovery

```
for all itemsets l ∈ C_k
        for all itemsets m ∈ MFS
         if the first k–1 items in l are also in m
                /* suppose m.item_j = l.item_{k-1} */
                for i from j+1 to |m|
                 C_{k+1} := C_{k+1} ∪ {{l.item_1, l.item_2, …, l.item_k, m.item_i}}
```

MFS-Prune

for all itemsets c in C_k
if c is a subset of any itemset is the current MFS
delete c from C_k;

MFCS-Prune

for all itemsets c in C_{k+1}
if c is not a subset of any itemset in the current MFCS
delete c from C_{k+1};

STEP 1: $L_0 := \varnothing$; $k := 1$;

$C_1 := \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}\}$

MFCS := {1, 2, 3, 4, 5, 6, 7, 8, 9}

MFS := $\varnothing$;

PASS ONE: Database is read to count the support as follows

$\{1\} \rightarrow 2$, $\{2\} \rightarrow 6$, $\{3\} \rightarrow 6$, $\{4\} \rightarrow 4$, $\{5\} \rightarrow 8$, $\{6\} \rightarrow 5$, $\{7\} \rightarrow 7$, $\{8\} \rightarrow 4$, $\{9\} \rightarrow 2$

$\{1, 2, 3, 4, 5, 6, 7, 8, 9\} \rightarrow 0$.

So MFCS := {1, 2, 3, 4, 5, 6, 7, 8, 9} and MFS := $\varnothing$;

$L_1 := \{\{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}\}$

$S_1 := \{\{1\}, \{9\}\}$

At this stage we call the MFCS-gen to update MFCS.

For {1} in S_1 and for {1, 2, 3, 4, 5, 6, 7, 8, 9} in MFCS, we get the new element in MFCS as {2, 3, 4, 5, 6, 7, 8, 9}.

For {9} in S_1 and for {2, 3, 4, 5, 6, 7, 8, 9} in MFCS, we get the new element in MFCS as {2, 3, 4, 5, 6, 7, 8}.

We generate the candidate itemsets

$C_2 := \{$ {2,3}, {2,4}, {2,5}, {2,6}, {2,7}, {2,8}, {3,4}, {3,5}, {3,6}, {3,7}, {3,8}, {4,5}, {4,6}, {4,7}, {4,8}, {5,6}, {5,7}, {5,8}, {6,7}, {6,8}, {7,8} $\}$

PASS TWO: read the database to count the support of elements in C_2 and MFCS as given below:

$\{2,3\} \rightarrow 3$, $\{2,4\} \rightarrow 3$, $\{2,5\} \rightarrow 0$, $\{2,6\} \rightarrow 2$, $\{2,7\} \rightarrow 2$, $\{2,8\} \rightarrow 1$, $\{3,4\} \rightarrow 1$, $\{3,5\} \rightarrow 3$, $\{3,6\} \rightarrow 0$, $\{3,7\} \rightarrow 3$, $\{3,8\} \rightarrow 0$, $\{4,5\} \rightarrow 1$, $\{4,6\} \rightarrow 1$, $\{4,7\} \rightarrow 2$, $\{4,8\} \rightarrow 1$, $\{5,6\} \rightarrow 3$, $\{5,7\} \rightarrow 5$, $\{5,8\} \rightarrow 2$, $\{6,7\} \rightarrow 3$, $\{6,8\} \rightarrow 2$, $\{7,8\} \rightarrow 0$

$\{2, 3, 4, 5, 6, 7, 8\} \rightarrow 0$.

MFS:= $\varnothing$;

L_2 := { {2,3}, {2,4}, {3,5}, {3,7}, {5,6}, {5,7}, {6,7} }

S_2 := { {2,5}, {2,6}, {2, 7} {2,8}, {3,4}, {3,6}, {3,8}, {4,5}, {4,6}, {4,7}, {4,8}, {5,8}, {6,8}, {7,8} }

For {2,5} in S_2 and for {2, 3, 4, 5, 6, 7, 8} in MFCS, we get the new elements in MFCS as {3, 4, 5, 6, 7, 8} and {2, 3, 4, 6, 7, 8}

For {2,6} in S_2 and for {3, 4, 5, 6, 7, 8} in MFCS, since {2,6} is not contained in this element of MFCS and hence, no action.

For {2, 3, 4, 6, 7, 8} we get two new elements in MFCS in place of {2, 3, 4, 6, 7, 8} as {3, 4, 6, 7, 8} and {2, 3, 4, 7, 8}. Since {3, 4, 6, 7, 8} is already contained in an element of MFCS, it is excluded from MFCS.

So at this stage MFCS := {{ 3, 4, 5, 6, 7, 8}, {2, 3, 4, 7, 8}}.

For {2,7} in S_2, we get

MFCS := {{3, 4, 5, 6, 7, 8}, {2, 3, 4, 8}}.

For {2,8} in S_2, we get

MFCS := {{3, 4, 5, 6, 7, 8}, {2, 3, 4}}.

For {3,4} in S_2, we get

MFCS := {{3, 5, 6, 7, 8}, {4, 5, 6, 7, 8}, {2, 3}, {2, 4}}.

For {3,6} in S_2, we get

MFCS := {{3, 5, 7, 8}, {4, 5, 6, 7, 8}, {2, 3}, {2, 4}}.

For {3,8} in S_2, we get

MFCS := {{3, 5, 7}, {4, 5, 6, 7, 8}, {2, 3}, {2, 4}}.

For {4,5} in S_2, we get

MFCS := {{3, 5, 7}, {5, 6, 7, 8}, {4, 6, 7, 8}, {2, 3}, {2, 4}}.

For {4,5} in S_2, we get

MFCS := {{3, 5, 7}, {5, 6, 7, 8}, {4, 6, 7, 8}, {2, 3}, {2, 4}}.

For {4,6} in S_2, we get

MFCS := {{3, 5, 7}, {5, 6, 7, 8}, {4, 7, 8}, {2, 3}, {2, 4}}.

For {4,7} in S_2, we get

MFCS := {{3, 5, 7}, {5, 6, 7, 8}, {4, 8}, {2, 3}, {2, 4}}.

For {4,8} in S_2, we get

MFCS := {{3, 5, 7}, {5, 6, 7, 8}, {2, 3}, {2, 4}}.

For {5,8} in S_2, we get

MFCS := {{3, 5, 7}, {6, 7, 8}, {5, 6, 7}, {2, 3}, {2, 4}}.

For {6,8} in S_2, we get

MFCS := {{7, 8}, {3, 5, 7}, {5, 6, 7}, {2, 3}, {2, 4}}.

For {7,8} in S_2, we get

MFCS := {{8}, {3, 5, 7}, {5, 6, 7}, {2, 3}, {2, 4}}.

We generate the candidate sets as

C_3 := {{2, 3, 4}, {3, 5, 7}, {5, 6, 7}}

In the pruning stage the itemsets {2, 3, 4} are pruned from C_3 and hence,

C_3 := {{3, 5, 7}, {5, 6, 7}}

At this stage we make one more pass of the database to count the supports of {{3, 5, 7}, {5, 6, 7}}.