CHAPTER FOUR

MINING ASSOCIATION RULE FROM LARGE DATABASE

Outline

- Association Rule
- Frequent Pattern
- Association mining from frequent Pattern
- Issues to be considered?
- Classification of Frequent Pattern Mining
  Mining Frequent Itemsets: the Key Step
- Algorithm to find Frequent Itemsets
  - The Apriori Algorithm

Association Rule

- Association rule is a rule which is described in the form of X➔Y with interestingness measure of support and confidence where
  - X and Y are Simple or complex Statements
  - A simple Statement is to mean a statement formed from a single attribute say age, buy or sex and a value which is related by relational operator
  - Example:

Buy(X, "Computer")➔ Buy(X, "Printer")[Supp = 25%, conf=95%]

  - *Which is to mean a person X who buy a computer also buy a printer .*
  - *25% of the entire data shows a person buy a computer and printer (support). Out of the tuples that buy a computer, 95% of them also buy printer (confidence)*

Association Rule

- A complex statement is usually represented as conjunction of simple statements
  - Example:

  Buy(X, "Computer")$\wedge$ Buys(X, "printer")➔ Buy(X, "Scanner")[Supp = 50%, conf=90%]

    - *Which is to mean a person X who buy a computer and a printer also buy a scanner.*
    - *50% of the entire data shows a person buy a computer, a printer and scanner among the entire data set(support).*
    - *Out of all transactions with a person that buy computer and printer, 90% of them also buy printer (confidence)*
- In order to mine such association rule, we need to discuss deeply about frequent pattern and its extraction algorithm

Frequent Pattern

- Frequent pattern are patterns (such as item set, sub sequences, or sub structures) that appear in a data set frequently.

- An ***item set*** are two or more items that appear together in a transaction data set.
- An item set is said to be ***frequent item set*** if the item set appear frequently together in a transaction data set.
- For example a milk and bread may occur together frequently in a single transaction and hence are frequent item set.

- ***Subsequence*** refers to items that happen in transaction in a sequential order.
- For example, buying computer at time t_0 may be followed by buying a digital camera at time t_1, and buying memory card at time t_2.
- A sub sequence that appear most frequently is said to be ***frequent subsequence.***

Frequent Pattern

- A **sub structure** refers to different structural forms of the data set, such as ***sub-graphs***, ***sub-trees***, or ***sub-lattices***, which may be combined with item sets or subsequences.
- If a substructure occurs frequently, it is called a ***(frequent) structured pattern.***
- Finding such frequent patterns plays an essential role **in mining associations**, **correlations**, **classification**, **clustering**, and other data mining tasks as well.
- Thus, frequent pattern mining has become an important data mining task and a focused theme in data mining research.
- This chapter is dedicated to methods of ***frequent itemset mining.***

Frequent Pattern

- We look into the following questions:
  - How can we find frequent itemsets from large amounts of data, where the data are either transactional or relational?
  - How can we mine association rules in multilevel and multidimensional space?
  - Which association rules are the most interesting?
  - How can we help or guide the mining procedure to discover interesting associations or correlations?
  - How can we take advantage of user preferences or constraints to speed up the mining process?

Frequent Pattern

- Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets.

- With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining frequent itemset patterns from their databases.

- The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes such as:
  - market basket analysis, catalog design, cross-marketing, loss-leader analysis and customer shopping behavior analysis.

Association mining from frequent Pattern

- Rule form: "Body (X) -> Head (Y) [support, confidence]".
- Which is read as if body (X) then head (Y) will occur together in the transaction with the stated support and confidence
- Rule ***support*** and ***confidence*** are two measures of rule **interestingness**. They respectively reflect the **usefulness** and **certainty** of discovered rules.
- 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.

Association mining from frequent Pattern

- Additional analysis can be performed to uncover interesting statistical correlations between associated items.
- Let I ={I_1, I_2, …, I_m} *be a set of items.*
- *Let D, the task-relevant data set, be a set of database* transactions where each transaction *T is a set of items such that* $T \subseteq I$.
- *Each transaction* is associated with an identifier, called TID (Transaction ID).
- Let *A be a set of items.*
- *A transaction T is* said to contain *A if and only if* $A \subseteq T$.
- *An association rule is an implication of the form* $A \Rightarrow B$, *where* $A \subset I$, $B \subset I$, *and* $A \cap B = \emptyset$.

Association mining from frequent Pattern

- *The rule A* ➔ *B holds in the transaction set D* with **support** *s, where s is the percentage of transactions in D that contain $A \cup B$ (i.e., the union of itemsets A and B, or say, both A and B).*

- *This is taken to be the probability, $P(A \cup B)$ =*

$$\frac{\text{\# of transaction with itemset } A \cup B}{\text{\#of total transaction}}$$

- Support shows the probability that all the predicates in A and B fulfill together.
  - Count of tuples that has both A and B divided by total number of tuples in the working data set

Association mining from frequent Pattern

- The rule *A* ➔ *B has* ***confidence*** *c in the transaction set D, where c is the percentage of* transactions in *D containing A that also contain B.*
- *This is taken to be the conditional* probability, *P(B|A)=*

$$\frac{\text{\# of transaction with itemset } A \cup B}{\text{\#of transactionwith itemset } A}$$

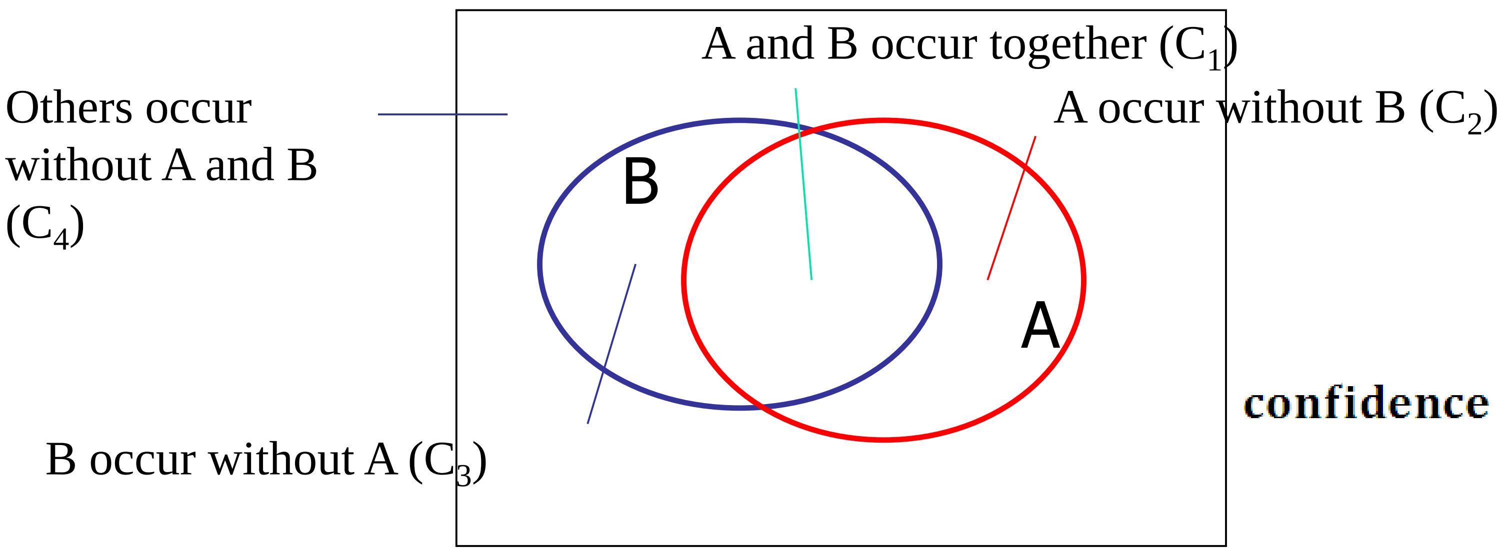
- Confidence measure how often predicates B fulfilled if predicate A get fulfilled.
  - Count of tuples that has both A and B together divided by total number of tuples that has A
- *That is*

$$support(A \rightarrow B) = P(A \cup B)$$
$$confidence(A \rightarrow B) = P(B|A)$$

Association mining from frequent Pattern

•Rules that satisfy both a minimum support threshold (*min sup) and a minimum confidence* threshold (*min conf) are called* ***strong***.

•*By convention, we write support and confidence* values so as to occur between 0% and 100%, rather than 0 to 1.0 which require to multiply by 100%.



$$\text{confidence} = \frac{C_1}{C_1 + C_2} 100\%$$

$$\text{Support} = \frac{C_1}{C_1 + C_2 + C_3 + C_4} 100\%$$

Association mining from frequent Pattern

- A set of items is referred to as an itemset.
- An itemset that contains *k items is a k-itemset.*
- *The set {computer, antivirus software} is a 2-itemset.*
- *The occurrence* frequency of an itemset is the number of transactions that contain the itemset.
- This is also known as the **frequency**, **support count**, or **count** of the itemset.

- **Note that** the itemset **support** defined before is sometimes referred to as ***relative support,*** whereas the occurrence frequency is called the ***absolute support***.
- If the relative support of an itemset *I satisfies a prespecified minimum support threshold (i.e., the absolute* support of *I satisfies the corresponding minimum support count threshold), then I is a* frequent itemset.

Association mining from frequent Pattern

- The set of frequent *k-itemsets is commonly denoted by* L_k.
- From the previous equation, we have

- $confidence(A \rightarrow B) = P(B \mid A)$
  $= support(A \cup B) / support(A)$ **(relative support)**
  $= support\_count(A \cup B) / support\_count(A)$ **(absolute support)**

- The above equation shows that the confidence of rule *A* $\rightarrow$ *B can be easily derived from the* support counts of *A and A* $\cup$ *B.*
- *That is, once the support counts of A, B, and A* $\cup$ *B are* found, it is straightforward to derive the corresponding association rules *A* $\rightarrow$ *B and B* $\rightarrow$ *A* and check whether they are strong.
- Thus, the problem of mining association rules can be reduced to that of mining frequent itemsets.

Association mining from frequent Pattern:

Support and Confidence example

- *Consider the following 4 transactions.*

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

- *The support for the various item set can be computed and the result shows:*

support(A)	3
support(A,C)	2
support(B)	2
support(C)	2
support(A,B,C)	1

support(B,E,F)	1
support(A,D)	1
support(A,B)	1
support(B,C)	1
support(B,E)	1

support(B,F)	1
support(E,F)	1
support(D)	1
support(E)	1
support(F)	1

Association mining from frequent Pattern: Support and Confidence example

- *The following are some of the association rules with support and confidence.*

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

support(A)	3
support(A,C)	2
support(B)	2
support(C)	2
support(A,B,C)	1

support(B,E,F)	1
support(A,D)	1
support(A,B)	1
support(B,C)	1
support(B,E)	1

support(B,F)	1
support(E,F)	1
support(D)	1
support(E)	1
support(F)	1

- $A \Rightarrow B$ (25%, 33.3%)
- $A \Rightarrow C$ (50%, 66.6%)
- $A \Rightarrow D$ (25%, 33.3%)
- $B \Rightarrow A$ (25%, 50%)
- $C \Rightarrow A$ (50%, 100%)
- $D \Rightarrow A$ (25%, 100%)
- $B \Rightarrow C$ (25%, 50%)
- $B \Rightarrow E$ (25%, 50%)
- $B \Rightarrow F$ (25%, 50%)
- $C \Rightarrow B$ (25%, 50%)
- $E \Rightarrow B$ (25%, 100%)
- $F \Rightarrow B$ (25%, 100%)

- $A \wedge B \Rightarrow C$ (25%, 100%)
- $A \wedge C \Rightarrow B$ (25%, 50%)
- $B \wedge C \Rightarrow A$ (25%, 100%)
- $A \Rightarrow C \wedge B$ (25%, 33.3%)
- $B \Rightarrow A \wedge C$ (25%, 50%)
- $C \Rightarrow A \wedge B$ (25%, 50%)
- $B \wedge E \Rightarrow F$ (25%, 100%)
- $B \wedge F \Rightarrow E$ (25%, 100%)
- $E \wedge F \Rightarrow B$ (25%, 100%)
- $B \Rightarrow E \wedge F$ (25%, 50%)
- $E \Rightarrow B \wedge F$ (25%, 100%)
- $F \Rightarrow B \wedge E$ (25%, 100%)

Association mining from frequent Pattern

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

1. **Find all frequent itemsets**
2. **Generate strong association rules from the frequent itemsets**

Find all frequent itemsets

- By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, *min sup*.
- *Let the minimum support count is 50% for the previous transaction which consists of 4 transactions.*
- *This enable generation of the following item set*

support(A)	3
support(A,C)	2
support(B)	2
support(C)	2

Association mining from frequent Pattern

Generate strong association rules from the frequent itemsets:

- At this step, we need to select association rules that must satisfy minimum support and minimum confidence.

support(A)	3
support(A,C)	2
support(B)	2
support(C)	2

- In the example considered above, only two associations are possible: A➔ C and C➔ A.
- Let the minimum confidence is 80%
- Hence the rule which full fill the condition is $\boldsymbol{C \Rightarrow A}$ **(50%, 100%)**
- Where as $\boldsymbol{A \Rightarrow C}$ **(50%, 66.6%)** doesn't fulfill the requirement of confidence and filtered out
- As the second step is much less costly than the first, the overall performance of mining association rules is determined by the first step

Issues to be considered?

- A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support (*min sup) threshold, especially when min sup is set low.*
- *This is because if an itemset is* frequent, each of its subsets is frequent as well.

- A long itemset will contain a combinatorial number of shorter, frequent sub-itemsets.
- For example, a frequent itemset I of length N items, the total number of item set which can be derived from I becomes

$$= \sum_{i=1}^{N} \binom{N}{i} = 2^n - 1$$

- *Where* $\binom{N}{i}$ *the number of frequent items which are subsets of I having i elements*

Issues to be considered?

- The stated frequent itemset number is a major issue from CPU requirement of our computer which demands appropriate algorithm.
- To overcome this difficulty, concepts of ***closed frequent itemset*** and ***maximal frequent itemset*** get introduced.
- An itemset X is ***closed itemset*** in a data set S if there exists no proper super-itemset Y such that Y has the same support count as X in S.
- The table to the left shows the ***closed itemset*** for the data set we have considered before
- An itemset X is a ***closed frequent itemset*** in set S if X is both ***closed*** and ***frequent*** in S.
- Lets assume support is 50% for our example above which has 4 transactions. The ***closed frequent itemset*** becomes

support(A)	3
support(A,C)	2
support(B)	2
support(A,B,C)	1
support(B,E,F)	1
support(A,D)	1

support(A)	3
support(A,C)	2
support(B)	2

Issues to be considered?

- An itemset X is a ***maximal frequent itemset*** (or ***max-itemset***) in set S if X is frequent, and there exists no super-itemset Y such that $X \subset Y$ and Y is frequent in S.
- The maximal frequent itemset of our sample data set becomes

support(A,C)	**2**
support(B)	**2**

- Summarizing the whole jargon

support(A)	3	support(B,E,F)	1	support(B,F)	1
support(A,C)	2	support(A,D)	1	support(E,F)	1
support(B)	2	support(A,B)	1	support(D)	1
support(C)	2	support(B,C)	1	support(E)	1
support(A,B,C)	1	support(B,E)	1	support(F)	1

Frequent itemset

support(A)	**3**
support(A,C)	**2**
support(B)	**2**
support(C)	**2**

Closed itemset

support(A)	**3**
support(A,C)	**2**
support(B)	**2**
support(A,B,C)	**1**
support(B,E,F)	**1**
support(A,D)	**1**

Closed frequent itemset

support(A)	**3**
support(A,C)	**2**
support(B)	**2**

Maximal frequent itemset

support(A,C)	**2**
support(B)	**2**

Issues to be considered?

- Let C be the set of ***closed frequent itemsets*** for a data set S satisfying a minimum support threshold, ***min_sup***.
- Let M be the set of ***maximal frequent itemsets*** for S satisfying ***min_sup***.
- Note: M $\subseteq$ C (all maximal frequent item set is member of closed frequent itemset)
- Consider the table bellow that shows sample closed and maximal frequent itemset

Closed frequent itemset			**Maximal frequent itemset**	
itemset	***count***		***itemset***	***count***
A	70%		A,B,C	51%
B	75%			
C	72%			
B,C	60%		A, C, D	55%
A, B	65%			
A,C,D	55%			
A,B,C	51%			

Issues to be considered?

- Suppose that we have the support count of each itemset in C and M.
- Notice that C and its count information can be used to derive the whole set of frequent itemsets and their support count.
  - Thus we say that C contains complete information regarding its corresponding frequent itemsets and their support count.
  - For example we know that support of D in the above table is 55%
- On the other hand, M registers only the support of the maximal itemsets.
  - It usually does not contain the complete support information regarding its corresponding frequent itemsets.
  - For example, it is not possible to know the support of A, B, C, D, A&B, etc other than saying they are frequent.

Classification of Frequent Pattern Mining

- Frequent pattern mining can be classified in various ways, based on different criteria, two of which are
  1. *Based on the levels of abstraction involved in the rule set:*
  2. *Based on the number of data dimensions involved in the rule:*

- ***Some other criterion may be***
  A. *Based on the completeness of patterns to be mined:*
  B. *Based on the types of values handled in the rule:*
  C. *Based on the kinds of rules to be mined*

Classification of Frequent Pattern Mining

- ***Based on the levels of abstraction involved in the rule set:***
  - Based on the level of abstraction, we can classify frequent pattern mining as **single level** and **multiple level** mining
  - **Multiple level** frequent pattern mining for association rule can find rules at differing levels of abstraction.
    - For example, suppose that a set of association rules mined includes the following rules where X is a variable representing a customer:

      buys(X, "computer")➔buys(X, "HP printer")

      buys(X, "desktop computer")➔buys(X, "HP printer")
    - In the above Rules, the items bought are referenced at different levels of abstraction (e.g., "computer" is a higher-level abstraction of "desktop computer").
  - If, instead, the rules within a given set do not reference items or attributes at different levels of abstraction, then the set contains single-level association rules.

Classification of Frequent Pattern Mining

- **Based on the number of data dimensions involved in the rule:**
  - Based on the number of data dimensions involved in the rule we can classify frequent pattern mining as **single dimensional** or **multidimensional**
  - If the items or attributes in an association rule reference only one dimension, then it is a single-dimensional association rule.

    buys(X, "computer") ➔buys(X, "antivirus software")

    buys(X, "computer")➔buys(X, "HP printer")

    buys(X, "laptop computer")➔buys(X, "HP printer")
  - The above rules are single-dimensional association rules because they each refer to only one dimension, **buys**.
  - If a rule references two or more dimensions, such as the dimensions age, income, and buys, then it is a multidimensional association rule.
  - The following rule is an example of a multidimensional rule:

    age(X, "30. . . 39") ^ income(X, "42K. . .48K")➔buys(X, "high resolution TV")

Mining Frequent Itemsets: the Key Step

- In order to mine association rule using frequent itemset from a database, we should perform the following basic steps

1. Find the *frequent itemsets*:
   - the sets of items that have minimum support
   - A subset of a frequent itemset is also frequent i.e., if $\{AB\}$ is a frequent itemset, both $\{A\}$ and $\{B\}$ are frequent
   - A number of algorithms are suggested to find the set of closed or maximal frequent items
2. Use the frequent itemsets to generate association rules that fulfill the confidence criteria.

Algorithm to find Frequent Itemsets

- There are a number of algorithms to find frequent itemset in mining association pattern from the data set
- Some of them are:
  1. The apriori algorithm
  2. Frequent pattern growth method
  3. Vertical data format method
- Several other algorithms have been proposed to mine association rules:
  - ☐ Sampling algorithms
  - ☐ Frequent-pattern tree algorithm
  - ☐ Partition algorithm

Algorithm to find Frequent Itemsets

1. The apriori algorithm:
   - It iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)

2. Frequent pattern growth method
   - Find frequent item set using divide and conquer method of frequent pattern tree

Algorithm to find Frequent Itemsets

3. Vertical data format method
   - Usually working data set is represented as a set of record where each record is identified by transaction id (TID) and associated itemsets.
   - This format is called **horizontal data format**
   - **Vertical data format** represent a record which is uniquely identified by an item name and having associated transaction ids for that item.
   - This approach uses this format of input data to discover all frequent pattern

- We will discuss in this chapter only the first approach ***(Apriori algorithm)***

The Apriori Algorithm

- Assume:
  - L_k be the set of all frequent k-itemsets which are ordered lexicographically (i.e. the i^{th} itemset in L_k is smaller than the j^{th} itemset iff $i < j$)
  - C_k be the set of k-itemset which is a super set of L_k.
  - l_i and l_j be the i^{th} and j^{th} k-itemset from a given L_k and each of their elements are also sorted lexicographically.

The Apriori Algorithm

- The Apriori algorithm will have the following steps
  - Initialization
  - Join Step
  - Prune Step
  - Generation

The Apriori Algorithm

- Initialization
  - Generate all the frequent itemset with cardinality of 1 (i.e. L_1) in which each elements are sorted lexicographically.
    - Let L_1 be $\{\{i_1\}, \{i_4\}, \{i_7\}, \{i_9\}, \{i_{11}\}\}$ (Note the ordering)

The Apriori Algorithm

- **Join Step**:
  - Generate the candidate k-itemsets by joining $L_{k-1}$with itself (i.e. $C_k = L_{k-1} \bowtie L_{k-1}$) using the following procedure:
    - Take any two element from L_{k-1} where each of them are similar in all their elements except the last
    - Form k-itemset set by union operation of the two (k-1)-itemset
    - Repeat the procedure for all possible such elements

The Apriori Algorithm

- Join Step:
  - Let's assume $L_2 = \{\{i_1,i_4\}, \{i_1,i_9\}, \{i_1,i_{11}\}, \{i_4,i_9\}, \{i_4,i_{11}\}, \{i_7,i_9\}, \{i_7,i_{11}\}\}$
  - The candidate 3-itemsets are $\{\{i_1,i_4,i_9\}, \{i_1,i_4,i_{11}\},$ **$\{i_1,i_9,i_{11}\}$, $\{i_4,i_9,i_{11}\}$, $\{i_7,i_9,i_{11}\}$** $\}$ (Note each elements are sorted and the elements of the elements are also sorted)
  - Note **$\{i_9,i_{11}\}$** is subset of the generated 3-itemset but not in L2.
  - As a result, some of the 2-itemset are not frequent and hence those 3-item set having **$\{i_9,i_{11}\}$** as its subset could not fulfill the requirement to be frequent itemset.
  - which has leads into immediate removal of the 3 candidate 3-itemsets in the next step

The Apriori Algorithm

- Prune Step:
  - generate C_k from the candidate k-itemset by pruning apriori those elements which has subsets that are not frequent
  - This can be best done by checking if an element in the k-itemset has Any (k-1)-itemset that is not frequent.
  - If such an element exist, it should be prunned as it is not frequent

The Apriori Algorithm

- Generation:
  - Generate L_k from C_k by eliminating elements which are not frequent
  - This can be best done by assigning count to each k-itemset in C_k by exploring the entire database transaction

The Apriori Algorithm

- Input:
  - *D, a database of transactions;*
  - *Min_sup, the minimum support count threshold.*
- Output:
  - *L, frequent itemsets in D.*

The Apriori Algorithm

- Method:

```
L1 = find frequent 1-itemsets(D); //initialize
for (k = 2; Lk-1 ≠ ∅; k++) {
        Ck = apriori_gen(Lk-1); //join and prune
        for each transaction t ∈ D {// scan D for counts
                Ct = subset(Ck, t);
                // get the subsets of t that are candidates
                for each candidate c ∈ Ct
                        c.count++;
        }
        Lk = { c ∈ Ck | c:count ≥ min_sup} //generate
}
return L = ∪k Lk;
```

The Apriori Algorithm

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
for each itemset l_1 ∈ L_{k-1}{
        for each itemset l_2 ∈ L_{k-1} {
          if (l_1[1] = l_2[1])∧(l_1[2] = l_2[2])∧ . . . ∧(l_1[k-2] = l_2[k-2])∧
             (l_1[k-1] < l_2[k-1]) then {
                c = l_1 ⋈ l_2; // join step: generate candidates
                if (not (has_infrequent_subset(c, L_{k-1})) then
                        add c to C_k;
                else  delete c; // prune step: remove unfruitful candidate
          }
        }
}
 return C_k;
```

The Apriori Algorithm

procedure ***has_infrequent_subset*** (*c: candidate k-itemset;* L_{k-1}*: frequent (k-1)-itemsets); // use prior knowledge*

1. for each *(k-1)-subset s of c*
2. if $s \not\in L_{k-1}$ *then*
3. return TRUE;
4. return FALSE;

The Apriori Algorithm — Example

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D →

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2