UNIT-II

Association Rule Mining: Mining Frequent Patterns–Associations and correlations – Mining Methods– Mining Various kinds of Association Rules– Correlation Analysis– Constraint based Association mining. Graph Pattern Mining, SPM.

I.Mining Frequent Patterns

- Association Rule Mining is to find out association rules or Frequent patterns or subsequences or correlation relationships among large set of data items that satisfy the predefined minimum support and confidence from a given database.

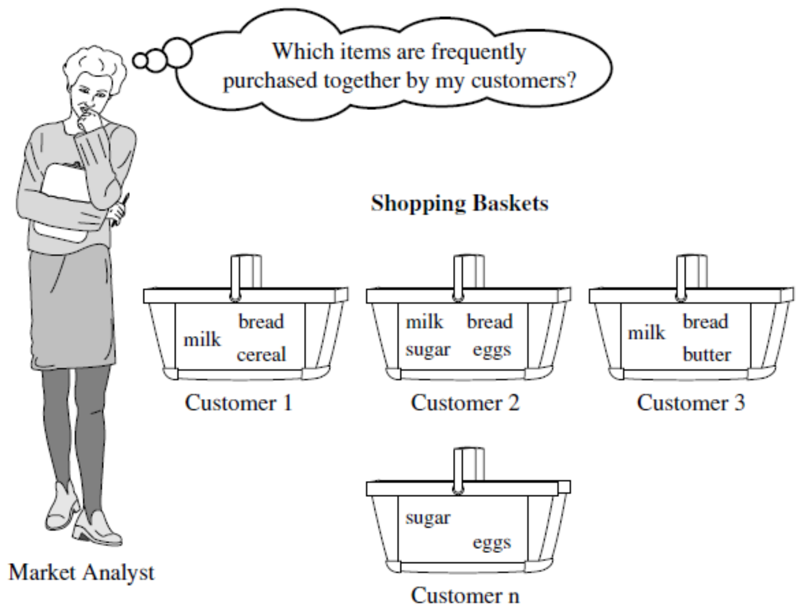
- **Frequent patterns** are patterns (such as itemsets, subsequences, or substructures) that appear in a data set frequently. For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a *frequent itemset*. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (*frequent*) *sequential pattern*. A *substructure* can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences.

What Is Association Mining?

- **Association rule mining:**
  - Finding frequent patterns, associations, correlations, or causal structures among sets of items or objects in transaction databases, relational databases, and other information repositories.
- **Applications:**
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

Market Basket Analysis:

Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets. A typical example of frequent itemset mining is market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their “shopping baskets”. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.



Frequent Itemsets, Closed Itemsets, and Association Rules

Let I ={*I*1, *I*2, ... , *Im*} be a set of items. Let *D*, the task-relevant data, be a set of database transactions where each transaction *T* is a set of items such that *T* _ I . Each transaction is associated with an identifier, called TID.

An association rule is an implication of the form $A \Rightarrow B$, where $A \subset$ I , $B \subset$ I , and $A \cap B$=f. The rule $A \Rightarrow B$ holds in the transaction set *D* with support *s*, where *s* is the percentage of transactions in *D* that contain $A \subset B$ (i.e., the *union* of sets *A* and *B*, or say, both *A* and *B*). The rule $A \Rightarrow B$ 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*). That is,

$$\begin{aligned} support(A \Rightarrow B) &= P(A \cup B) \\ confidence(A \Rightarrow B) &= P(B|A). \end{aligned}$$

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. The set of frequent *k*-itemsets is commonly denoted by *Lk*.

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} = \frac{support\_count(A \cup B)}{support\_count(A)}.$$

Support is used to eliminate uninteresting rules. Low support is likely to be uninteresting from a business perspective because it may not be profitable to promote items.

Confidence measures the reliability of the inference made by a rule. For rule A→B, the higher confidence , the more likely it is for **B** to be present in transactions that contain **A**. It is used to estimate conditional probability of **B** given **A.**

2. From Association Analysis to Correlation Analysis

A correlation measure can be used to augment the support-confidence framework for association rules. This leads to *correlation rules* of the form
A=>B [*support*, *confidence*, *correlation*]

- That is, a correlation rule is measured not only by its support and confidence but alsoby the correlation between itemsets*A* and *B*. There are many different correlation measuresfrom which to choose. In this section, we study various correlation measures todetermine which would be good for mining large data sets.

- Lift is a simple correlation measure that is given as follows. The occurrence of itemset *A* is independent of the occurrence of itemset*B* if $P(A \cup B)$ = *P*(*A*)*P*(*B*); otherwise, itemsets*A* and *B* are dependent and correlated as events. This definition can easily be extended to more than two itemsets.

The lift between the occurrence of *A* and *B* can bemeasured by computing

$$lift(A, B) = \frac{P(A \cup B)}{P(A)P(B)}.$$

- If the lift(A,B) is less than 1, then the occurrence of A is negativelycorrelated with the occurrence of B.
- If the resulting value is greater than 1, then A and B are positively correlated, meaning that the occurrence of one implies the occurrence of the other.
-

If the resulting value is equal to 1, then A and B are independent and there is no correlation between them.

3. Mining Methods

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

1. Find all frequent itemsets: By definition, each of these 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.

The Apriori Algorithm:

Apriori is a seminal algorithm proposed by R. Agrawal and R. Srikant in 1994 for mining frequent itemsets for Boolean association rules. Apriori employs an iterative approach known as a *level-wise* search, where *k*-itemsets are usedtoexplore (*k*+1)-itemsets. First, the setof frequent 1-itemsets is found by scanning the database to accumulate the count for each item, and collecting those items that satisfy minimum support. The resulting set is denoted *L*1.Next, *L*1 is used to find *L*2, the set of frequent 2-itemsets, which is used to find *L*3, and so on, until no more frequent *k*-itemsets can be found. The finding of each *Lk* requires one full scan of the database.

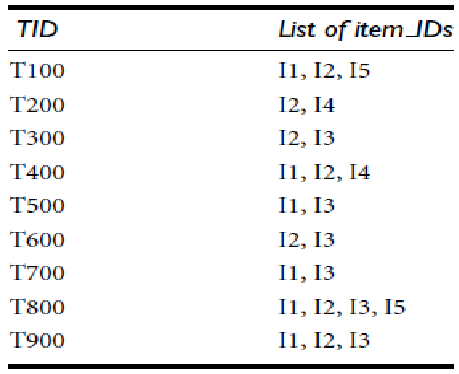
Apriori Property:

1. *It makes use of "Upward Closure property" (Any superset of infrequent itemset is also an infrequent set). It follows Bottom-up search, moving upward level-wise in the lattice.*
2. *It makes use of "downward closure property"(any subset of a frequent itemset is a frequent itemset).*
3. *If* Support of an itemset exceeds the support of its subsets, then it is known as the anti-monotone property of support.

Steps in candidate generation:

- Join Step: To find *Lk*, a set of candidate *k*-itemsets is generated by joining *Lk*-1 with itself.
- Prune Step: Any k-itemset that is not frequent cannot be a subset of a frequent (k+1)-itemset .

Table 5.1 Transactional data for an *AllElectronics* branch.

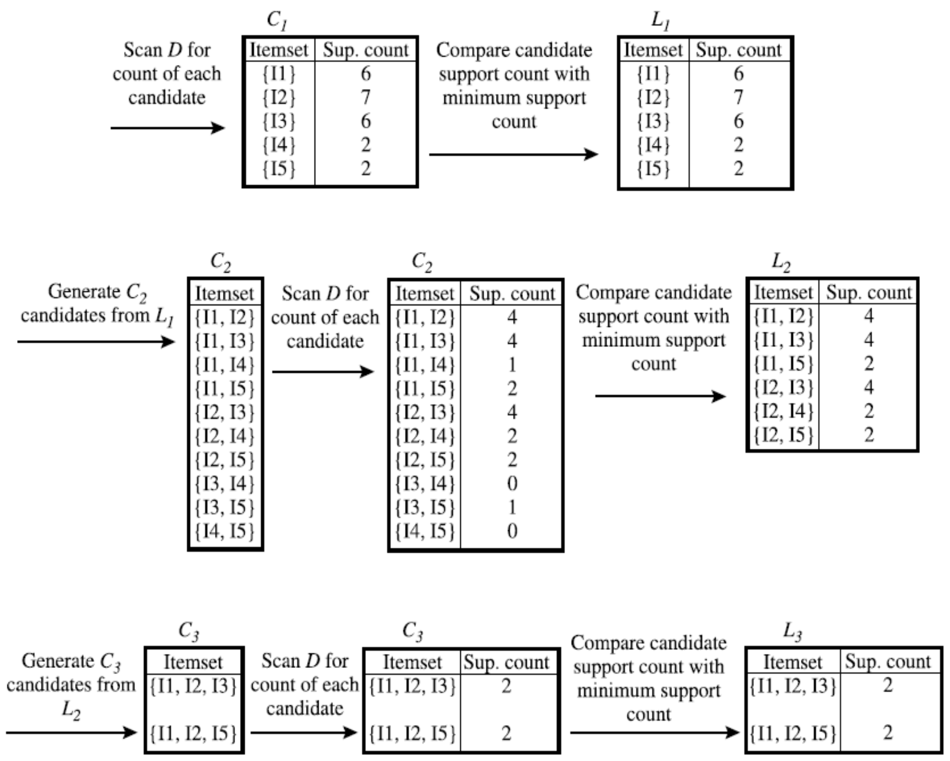


TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Based on the *AllElectronics* transaction database,*D*, of Table 5.1

Apriori algorithm steps for finding frequent itemsets in D:-

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, *C*1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.
2. Suppose that the minimum support count required is 2, that is, *min sup* = 2. The set of frequent 1-itemsets, *L*1, can then be determined. It consists of the candidate 1-itemsets satisfying minimum support.
3. To discover the set of frequent 2-itemsets, *L*2, the algorithm uses the join *L*1 ⊗ *L*1 to generate a candidate set of 2-itemsets, *C*2.
4. Next, the transactions in*D*are scanned and the support count of each candidate itemset in*C*2 is accumulated, as shown in the middle table of the second row in Figure 5.2.


Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_2 candidates from L_1
Scan D for count of each candidate
Compare candidate support count with minimum support count
Generate C_3 candidates from L_2
Scan D for count of each candidate
Compare candidate support count with minimum support count

C_1

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

L_1

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

C_2

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

C_2

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

L_2

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

C_3

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

C_3

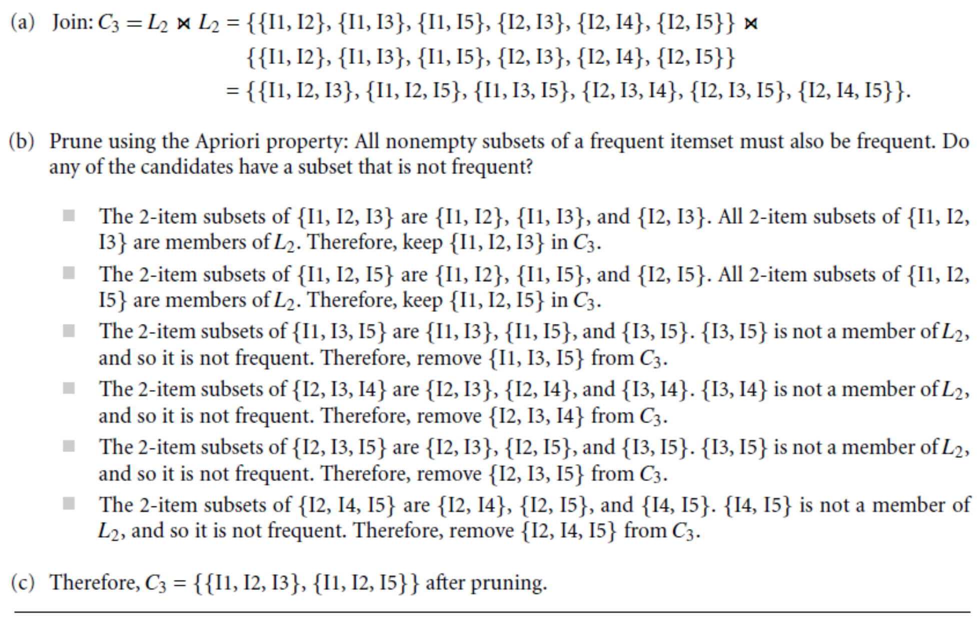
Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

L_3

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

Figure 5.2 Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2.

5. The set of frequent 2-itemsets, *L*2, is then determined, consisting of those candidate 2-itemsets in *C*2 having minimum support.
6. The generation of the set of candidate 3-itemsets,*C*3, is detailed in Figure 5.3.


(a) Join: $C_3 = L_2 \bowtie L_2$ = {{I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}} $\bowtie$
{{I1, I2}, {I1, I3}, {I1, I5}, {I2, I3}, {I2, I4}, {I2, I5}}
= {{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}.

(b) Prune using the Apriori property: All nonempty subsets of a frequent itemset must also be frequent. Do any of the candidates have a subset that is not frequent?

- The 2-item subsets of {I1, I2, I3} are {I1, I2}, {I1, I3}, and {I2, I3}. All 2-item subsets of {I1, I2, I3} are members of L_2. Therefore, keep {I1, I2, I3} in C_3.
- The 2-item subsets of {I1, I2, I5} are {I1, I2}, {I1, I5}, and {I2, I5}. All 2-item subsets of {I1, I2, I5} are members of L_2. Therefore, keep {I1, I2, I5} in C_3.
- The 2-item subsets of {I1, I3, I5} are {I1, I3}, {I1, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I1, I3, I5} from C_3.
- The 2-item subsets of {I2, I3, I4} are {I2, I3}, {I2, I4}, and {I3, I4}. {I3, I4} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I4} from C_3.
- The 2-item subsets of {I2, I3, I5} are {I2, I3}, {I2, I5}, and {I3, I5}. {I3, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I3, I5} from C_3.
- The 2-item subsets of {I2, I4, I5} are {I2, I4}, {I2, I5}, and {I4, I5}. {I4, I5} is not a member of L_2, and so it is not frequent. Therefore, remove {I2, I4, I5} from C_3.

(c) Therefore, C_3 = {{I1, I2, I3}, {I1, I2, I5}} after pruning.

Figure 5.3 Generation and pruning of candidate 3-itemsets, C_3, from L_2 using the Apriori property.

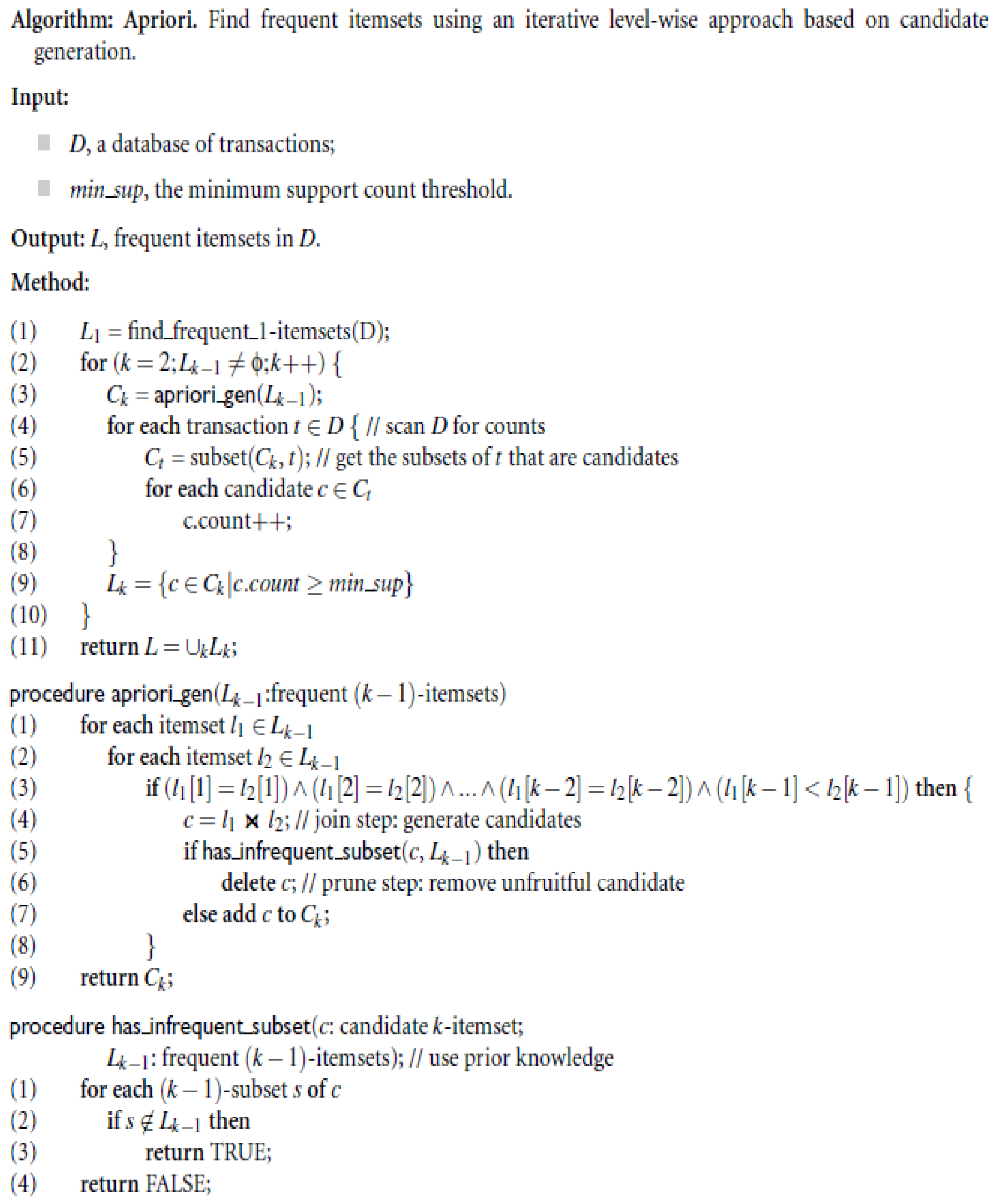
7. The transactions in *D* are scanned in order to determine *L*3, consisting of those candidate 3-itemsets in *C*3 having minimum support (Figure 5.2).

8. The algorithm uses *L*3 ⊗ *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 because its subset {I2, I3,I5} is not frequent. Thus, *C*4 = f, and the algorithm terminates, having found all of

the frequent itemsets.

pseudo-code for the Apriori algorithm:-



Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- min_sup, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
(1)   $L_1$ = find_frequent_1-itemsets(D);
(2)   for ($k = 2; L_{k-1} \neq \phi; k{+}{+}$) {
(3)       $C_k$ = apriori_gen($L_{k-1}$);
(4)       for each transaction $t \in D$ { // scan $D$ for counts
(5)           $C_t$ = subset($C_k$, $t$); // get the subsets of $t$ that are candidates
(6)           for each candidate $c \in C_t$
(7)               c.count++;
(8)       }
(9)       $L_k = \{c \in C_k | c.count \geq min\_sup\}$
(10)  }
(11)  return $L = \cup_k L_k$;

procedure apriori_gen($L_{k-1}$:frequent $(k-1)$-itemsets)
(1)   for each itemset $l_1 \in L_{k-1}$
(2)       for each itemset $l_2 \in L_{k-1}$
(3)           if $(l_1[1] = l_2[1]) \wedge (l_1[2] = l_2[2]) \wedge ... \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1])$ then {
(4)               $c = l_1 \bowtie l_2$; // join step: generate candidates
(5)               if has_infrequent_subset($c$, $L_{k-1}$) then
(6)                   delete $c$; // prune step: remove unfruitful candidate
(7)               else add $c$ to $C_k$;
(8)           }
(9)   return $C_k$;

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 $s \notin L_{k-1}$ then
(3)           return TRUE;
(4)   return FALSE;
```

How to Generate Candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

insert into $\boldsymbol{C_k}$
select $\boldsymbol{p.item_1, p.item_2, \ldots, p.item_{k-1}, q.item_{k-1}}$
from $\boldsymbol{L_{k-1}\ p, L_{k-1}\ q}$
where $\boldsymbol{p.item_1=q.item_1, \ldots, p.item_{k-2}=q.item_{k-2}, p.item_{k-1} < q.item_{k-1}}$

- Step 2: pruning

```
forall itemsets c in C_k do
    forall (k-1)-subsets s of c do
        if (s is not in L_{k-1}) then delete c from C_k
```

Example of Generating Candidates

- L_3={*abc, abd, acd, ace, bcd*}
- Self-joining: L_3*L_3
  - *abcd* from *abc* and *abd*
  - *acde* from *acd* and *ace*
- Pruning:
  - *acde* is removed because *ade* is not in L_3
- C_4={*abcd*}

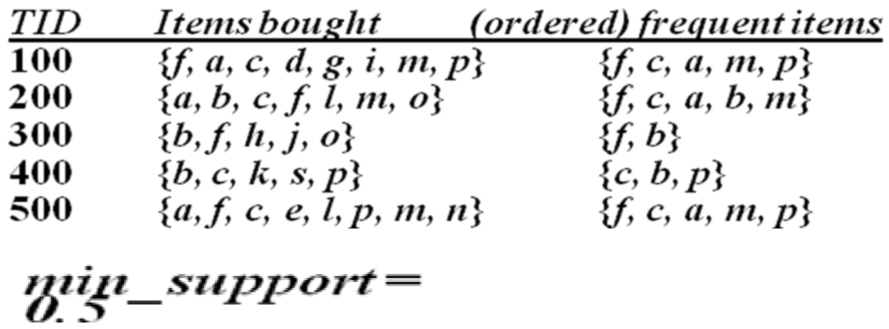
Methods to Improve Apriori's Efficiency

- Hash-based itemset counting: A *k*-itemset whose corresponding hashing bucket count is below the threshold cannot be frequent
- Transaction reduction: A transaction that does not contain any frequent k-itemset is useless in subsequent scans
- Partitioning: Any itemset that is potentially frequent in DB must be frequent in at least one of the partitions of DB
- Sampling: mining on a subset of given data, lower support threshold + a method to determine the completeness
- Dynamic itemset counting: add new candidate itemsets only when all of their subsets are estimated to be frequent

FP-Tree Growth Algorithm :(Mining Frequent Patterns) Without Candidate Generation

- Compress a large database into a compact, Frequent-Pattern tree (FP-tree) structure
  - highly condensed, but complete for frequent pattern mining
  - avoid costly database scans
- Develop an efficient, FP-tree-based frequent pattern mining method
  - A divide-and-conquer methodology: decompose mining tasks into smaller ones
  - Avoid candidate generation: sub-database test only!

Construct FP-tree from a Transaction DB

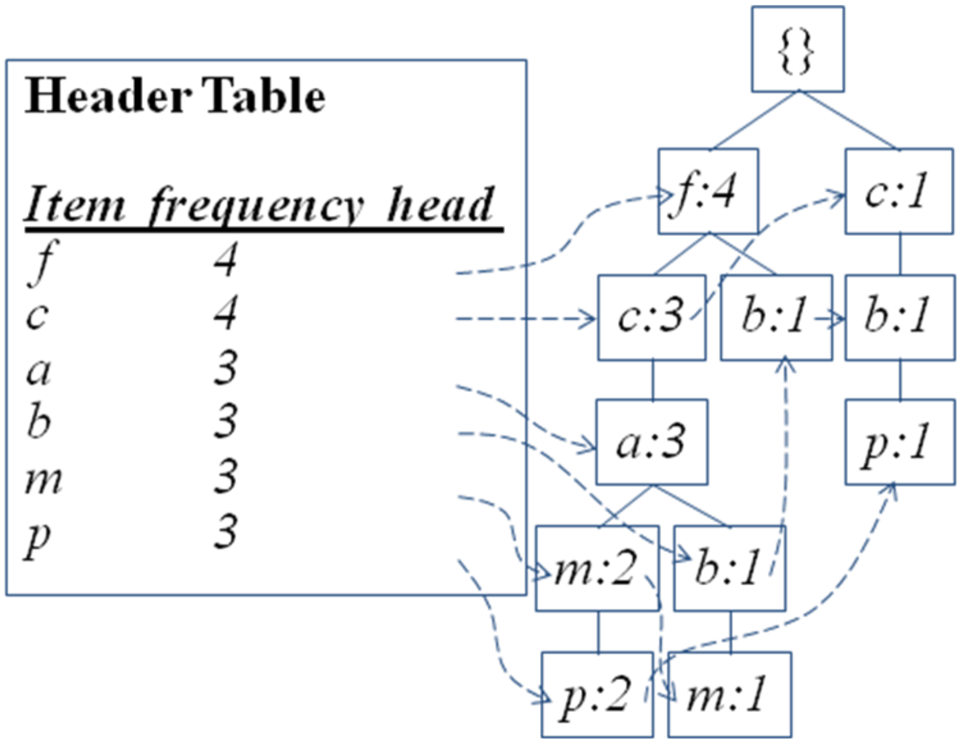


TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

min_support = 0.5

Steps:

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Order frequent items in frequency descending order
3. Scan DB again, construct FP-tree



Benefits of the FP-tree Structure

- Completeness:
    - never breaks a long pattern of any transaction
    - preserves complete information for frequent pattern mining
- Compactness
    - reduce irrelevant information—infrequent items are gone
    - frequency descending ordering: more frequent items are more likely to be shared
    - never be larger than the original database (if not count node-links and counts)
    - Example: For Connect-4 DB, compression ratio could be over 100

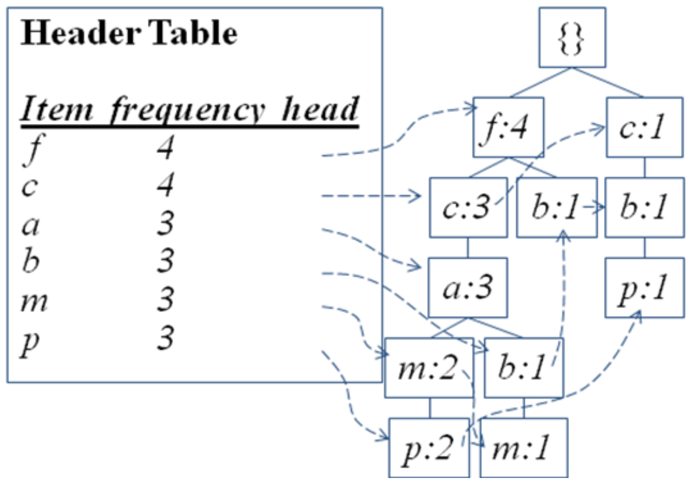
Mining Frequent Patterns Using FP-tree

- General idea (divide-and-conquer)
    - Recursively grow frequent pattern path using the FP-tree
- Method
    - For each item, construct its conditional pattern-base, and then its conditional FP-tree
    - Repeat the process on each newly created conditional FP-tree

- Until the resulting FP-tree is empty, or it contains only one path (single path will generate all the combinations of its sub-paths, each of which is a frequent pattern)

Step 1: From FP-tree to Conditional Pattern Base

- Starting at the frequent header table in the FP-tree
- Traverse the FP-tree by following the link of each frequent item
- Accumulate all of transformed prefix paths of that item to form a conditional pattern base



Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

Conditional **pattern bases**

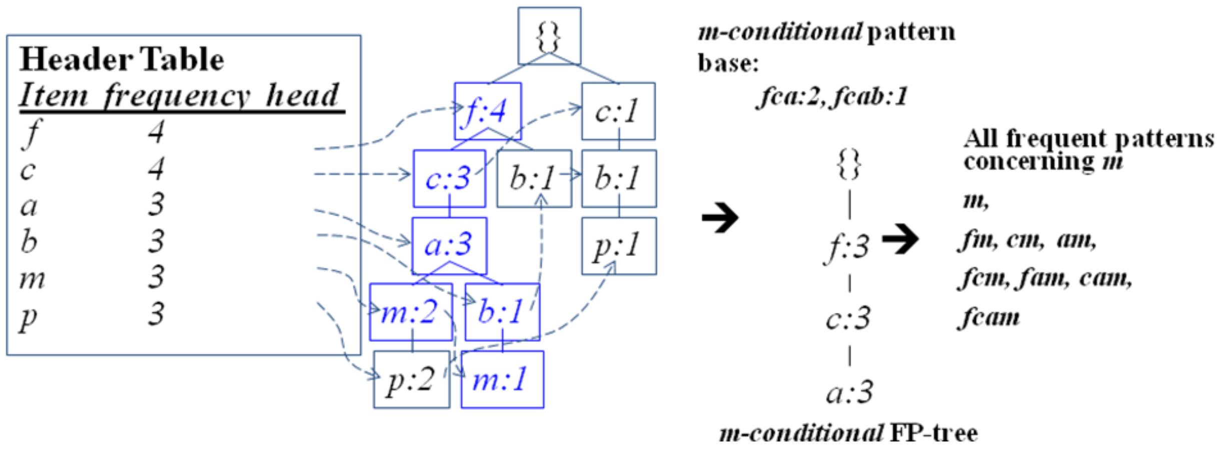
item	*cond. pattern base*
c	*f:3*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

Properties of FP-tree for Conditional Pattern Base Construction

- Node-link property
    - For any frequent item a_i, all the possible frequent patterns that contain a_i can be obtained by following a_i's node-links, starting from a_i's head in the FP-tree header
- Prefix path property
    - To calculate the frequent patterns for a node a_i in a path P, only the prefix sub-path of a_i in P need to be accumulated, and its frequency count should carry the same count as node a_i.

Step 2: Construct Conditional FP-tree

- For each pattern-base
    - Accumulate the count for each item in the base
    - Construct the FP-tree for the frequent items of the pattern base



Compact Representation of Frequent Itemset

- Usually, huge number of frequent itemsets produced from transactional dataset.
- It is useful to identify a **small representative set of itemsets** from which all **other frequent itemsets** can be derived.
- There are two such representations

1. **Maximal Frequent Itemsets:**

    ➢ An itemset is maximal frequent if none of its immediate supersets is frequent.

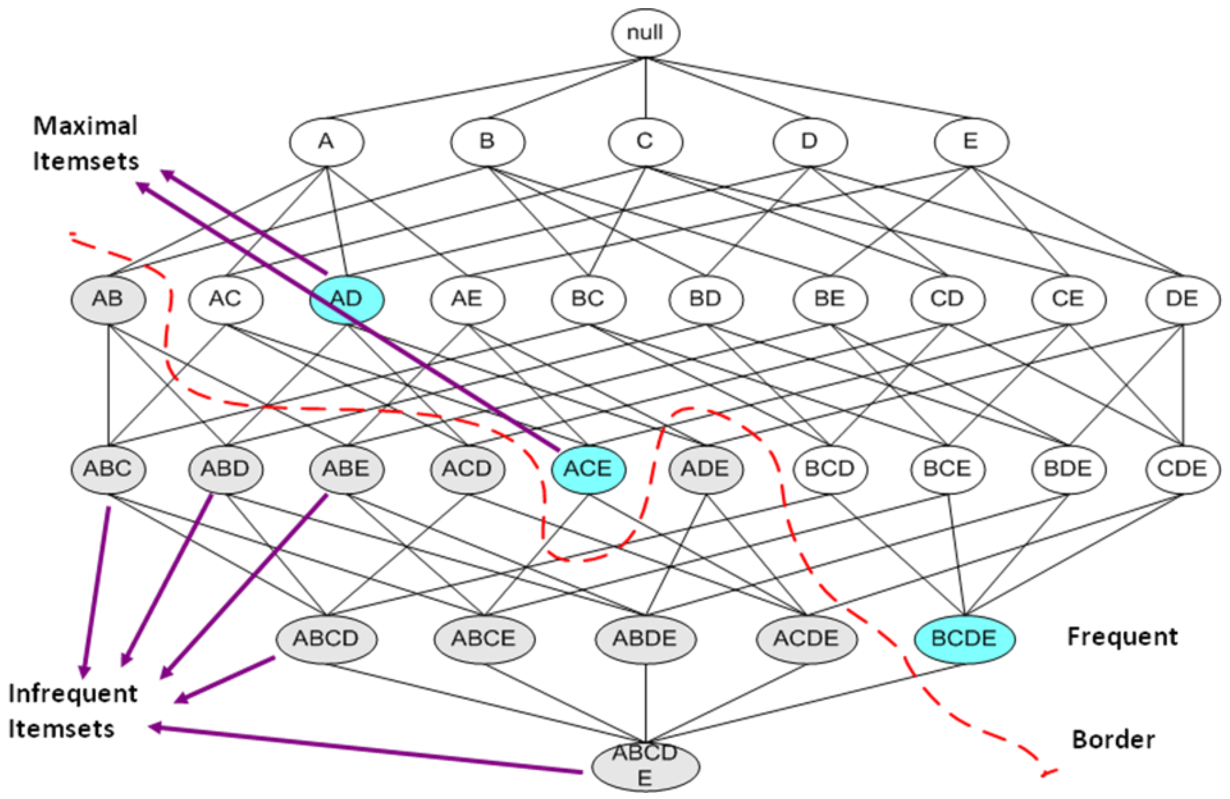
2. **Closed Frequent Itemsets:**

    ➢ An itemset is closed if none of it's immediate supersets has exactly the same support count .

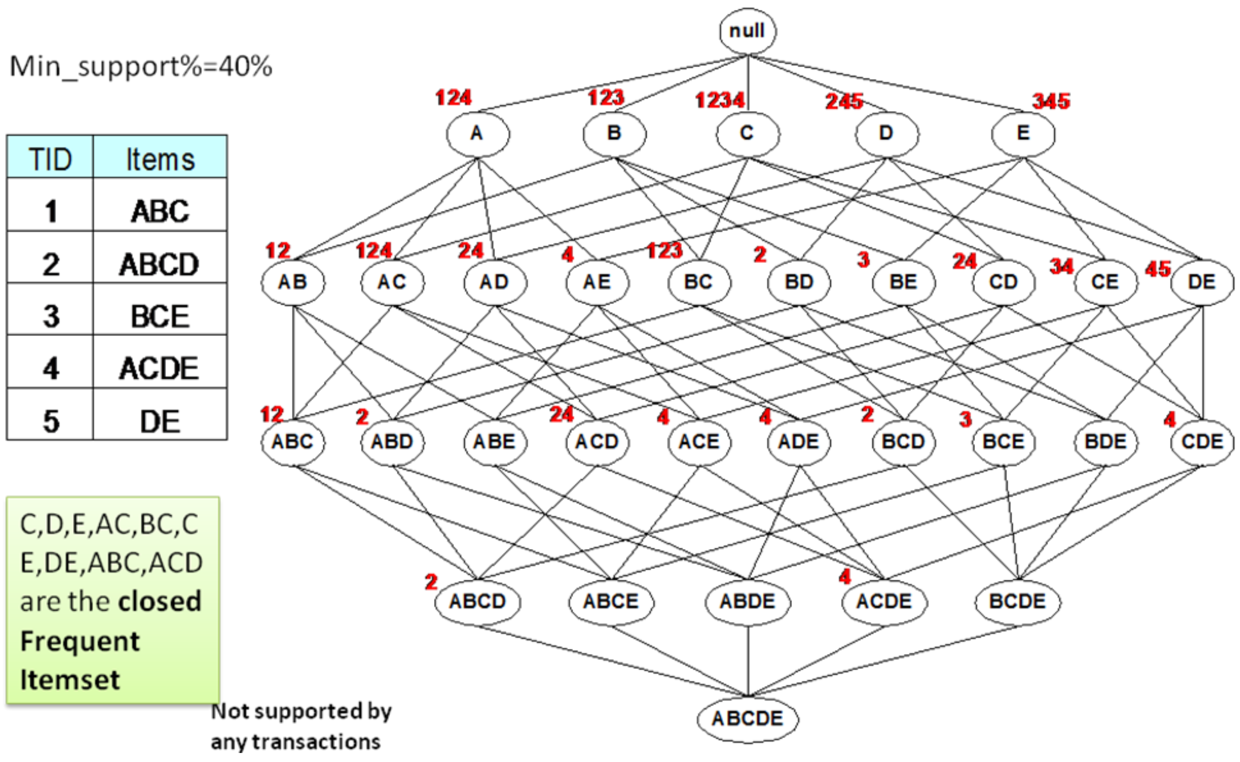
    ➢ Closed itemsets provide a minimal representation of itemsets without losing their support information.

Maximal Frequent Itemset

An itemset is maximal frequent if none of its immediate supersets is frequent.



Closed Frequent Itemsets



TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

C,D,E,AC,BC,CE,DE,ABC,ACDE are the **closed Frequent Itemset**

4.Mining Various Kinds of Association Rules

1. **Based on the completeness of patterns to be mined:**

- We can mine the complete set of frequent itemsets, the closed frequent itemsets, and themaximal frequent itemsets, given a minimum support threshold.
- We can also mine constrained frequent itemsets, approximate frequent itemsets,near-match frequent itemsets, top-k frequent itemsets and so on.

2. **Based on the levels of abstraction involved in the rule set:**

Some methods for associationrule mining can find rules at differing levels of abstraction.

For example, supposethat a set of association rules mined includes the following rules where X is a variablerepresenting a customer:

buys(X, –computer‖))=>buys(X, –HP printer‖) (1)

buys(*X*, –*laptop computer*‖)) =>*buys*(*X*, –*HP printer*‖) (2)

In rule (1) and (2), the items bought are referenced at different levels ofabstraction (e.g., ―*computer*‖ is a higher-level abstraction of ―*laptop computer*‖).

3. **Based on the number of data dimensions involved in the rule:**

- If the items or attributes in an association rule reference only one dimension, then it is asingle-dimensional association rule.
  buys(X, ―computer‖))=>buys(X, ―antivirus software‖)

- If a rule references two or more dimensions, such as the dimensions age, income, and buys, then it is amultidimensional association rule. The following rule is an exampleof a multidimensional rule:
  age(X, ―30,31…39‖) ^ income(X, ―42K,…48K‖))=>buys(X, ―high resolution TV‖)

4. **Based on the types of values handled in the rule:**

- If a rule involves associations between the presence or absence of items, it is a Booleanassociation rule.
- If a rule describes associations between quantitative items or attributes, then it is aquantitative association rule.

5. **Based on the kinds of rules to be mined:**

- Frequent pattern analysis can generate various kinds of rules and other interesting relationships.
- Association rule mining cangenerate a large number of rules, many of which are redundant or do not indicatea correlation relationship among itemsets.
- The discovered associations can be further analyzed to uncover statistical correlations,leading to correlation rules.

6. **Based on the kinds of patterns to be mined:**

- Many kinds of frequent patterns can be mined from different kinds of data sets.
- Sequential pattern mining searches for frequent subsequences in a sequence data set, where a sequence records an ordering of events.
- For example, with sequential pattern mining, we can study the order in which items are frequently purchased. For instance, customers may tend to first buy a PC, followed by a digitalcamera,and then a memory card.
- Structuredpatternminingsearches for frequent substructuresin a structured data

set. • Single items are the simplest form of structure.

- Each element of an itemsetmay contain a subsequence, a subtree, and so on.
- Therefore, structuredpattern mining can be considered as the most general formof frequent pattern mining.

2.1 Mining Multilevel Association Rules:

- For many applications, it is difficult to find strong associations among data items at lowor primitive levels of abstraction due to the sparsity of data at those levels.
- Strong associations discovered at high levels of abstraction may represent

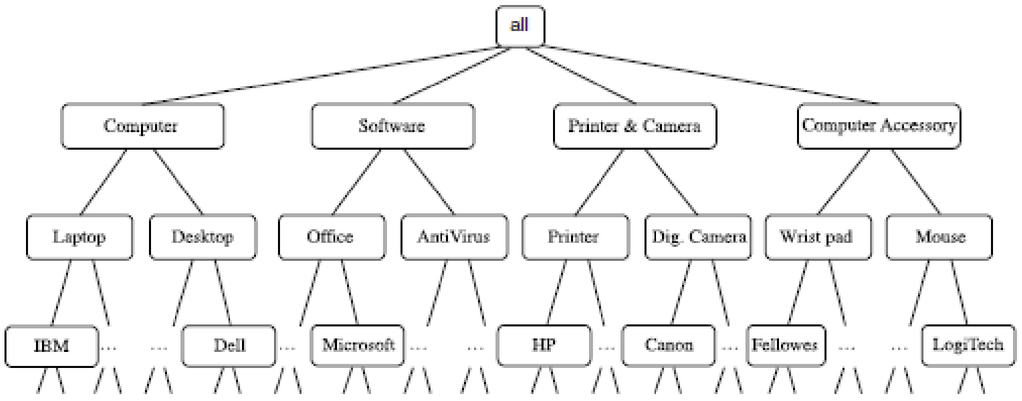
commonsenseknowledge.

- Therefore, data mining systems should provide capabilities for mining association rules at multiple levels of abstraction, with sufficient flexibility for easy traversal amongdifferentabstraction spaces.

- Association rules generated from mining data at multiple levels of abstraction arecalled multiple-level or multilevel association rules.
- Multilevel association rules can be mined efficiently using concept hierarchies under a support-confidence framework.
- In general, a top-down strategy is employed, where counts are accumulated for the calculation of frequent itemsets at each concept level, starting at the concept level 1 and working downward in the hierarchy toward the more specific concept levels,until no more frequent itemsets can be found.

A concepthierarchy defines a sequence of mappings froma set of low-level concepts to higherlevel,more general concepts. Data can be generalized by replacing low-level conceptswithin the data by their higher-level concepts, or ancestors, from a concept hierarchy.

TID	*Items Purchased*
T100	IBM-ThinkPad-T40/2373, HP-Photosmart-7660
T200	Microsoft-Office-Professional-2003, Microsoft-Plus!-Digital-Media
T300	Logitech-MX700-Cordless-Mouse, Fellowes-Wrist-Rest
T400	Dell-Dimension-XPS, Canon-PowerShot-S400
T500	IBM-ThinkPad-R40/P4M, Symantec-Norton-Antivirus-2003
...	...



A concept hierarchy for *AllElectronics* computer items.

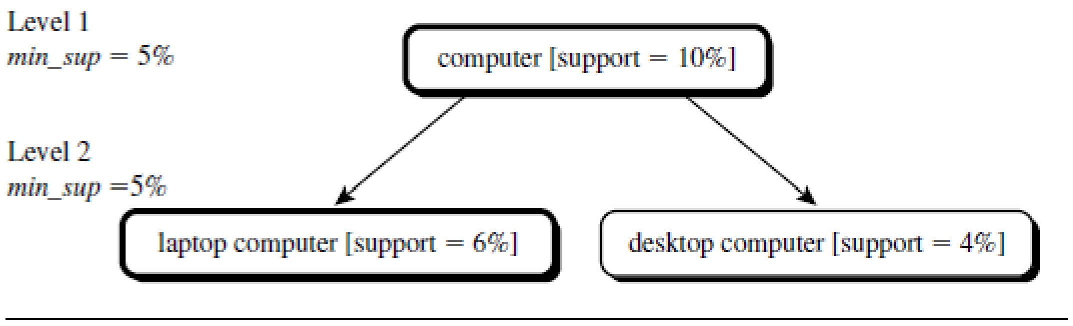
The concept hierarchy has five levels, respectively referred to as levels 0to 4, starting with level 0 at the root node for all.

- Here, Level 1 includes computer, software, printer&camera, and computer accessory.
- Level 2 includes laptop computer, desktop computer, office software, antivirus software
- Level 3 includes IBM desktop computer, . . . , Microsoft office software, and so on.
- Level 4 is the most specific abstraction level of this hierarchy.

2.1.1 Approaches ForMining Multilevel Association Rules:

1. UniformMinimum Support:

- The same minimum support threshold is used when mining at each level of abstraction.
- When a uniform minimum support threshold is used, the search procedure is simplified.
- The method is also simple in that users are required to specify only one minimum support threshold.
- The uniform support approach, however, has some difficulties. It is unlikely thatitems at lower levels of abstraction will occur as frequently as those at higher levelsof abstraction.
- If the minimum support threshold is set too high, it could miss somemeaningful associations occurring at low abstraction levels. If the threshold is set too low, it may generate many uninteresting associations occurring at high abstractionlevels.

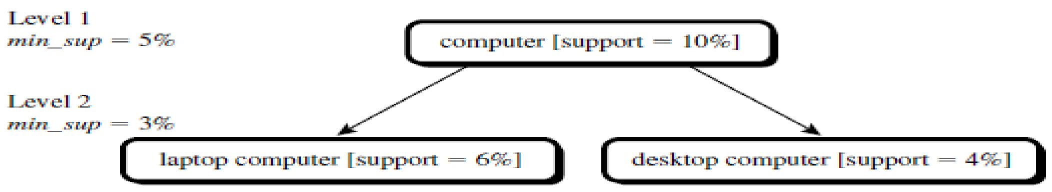


Multilevel mining with uniform support.

2. Reduced Minimum Support:

- Each level of abstraction has its own minimum support threshold.

- The deeper the level of abstraction, the smaller the corresponding threshold is.
- For example,the minimum support thresholds for levels 1 and 2 are 5% and 3%,respectively. In this way, –computer,‖ –laptop computer,‖ and –desktop computer‖ areall considered frequent.



3. Group-Based Minimum Support:

- Because users or experts often have insight as to which groups are more important than others, it is sometimes more desirable to set up user-specific, item, or group based minimal support thresholds when mining multilevel rules.
- For example, a user could set up the minimum support thresholds based on product price, or on items of interest, such as by setting particularly low support thresholds for laptop computersand flash drives in order to pay particular attention to the association patterns containing items in these categories.

2.2 Mining Multidimensional Association Rules from RelationalDatabases and Data Warehouses:

- Single dimensional or intradimensional association rule contains a single distinct predicate (e.g., buys)with multiple occurrences i.e., the predicate occurs more than once within the rule.

  buys(*X*, –*digital camera*‖)=>*buys*(*X*, –*HP printer*‖)

- Association rules that involve two or more dimensions or predicates can be referred toas multidimensional association rules.

age(X, “20…29”)^occupation(X, “student”)=>buys(X, “laptop”)

- Above Rule contains three predicates (age, occupation,and buys), each of which occurs only once in the rule. Hence, we say that it has norepeated predicates.
- Multidimensional association rules with no repeated predicates arecalled interdimensional association rules.
- We can also mine multidimensional associationrules with repeated predicates, which contain multiple occurrences of some predicates.These rules are called hybrid- dimensional association rules. An example of sucha rule is the following, where the predicate buys is repeated:
  age(X, –20…29‖)^buys(X, –laptop‖)=>buys(X, –HP printer‖)

2.3 Mining Quantitative Association Rules:

- Quantitative association rules are multidimensional association rules in which the numeric attributes are *dynamically* discretized during the mining process so as to satisfy some mining criteria, such as maximizing the confidence or compactness of the rules mined.
- In this section, we focus specifically on how to mine quantitative association rules having two quantitative attributes on the left-hand side of the rule and one categorical attribute on the right-hand side of the rule. That is

*Aquan*1 ^*Aquan*2 =>*Acat*

where*Aquan*1 and *Aquan*2 are tests on quantitative attribute interval

*Acat*tests a categorical attribute fromthe task-relevantdata.

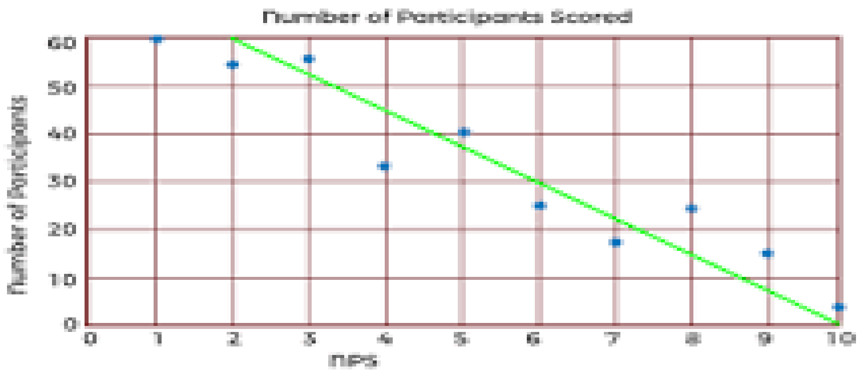
- Such rules have been referred to as two-dimensional quantitative association rules, because
they contain two quantitative dimensions.
- For instance, suppose you are curious about the association relationship between pairs of quantitative attributes, like customer age and income, and the type of television (such as *high-definition TV*, i.e., *HDTV*) that customers like to buy.

An example of such a 2-D quantitative association rule is

age(*X*, –30…39‖)^*income*(*X*, –42*K*…48*K*‖)=>*buys*(*X*, –*HDTV*‖)

5. CorrelationAnalysis:

Correlation analysis is a statistical method used to measure the strength of the linear relationship between two variables and compute their association. Correlation analysis calculates the level of change in one variable due to the change in the other.



Correlation analysis is a statistical method used to measure the strength of the linear relationship between two variables and compute their association. Correlation analysis calculates the level of change in one variable due to the change in the other. A high correlation points to a strong relationship between the two variables, while a low correlation means that the variables are weakly related.

Correlation is a bivariate analysis that measures the strength of association between two variables and the direction of the relationship. In terms of the strength of the relationship, the correlation coefficient's value varies between +1 and -1. A value of ± 1 indicates a perfect degree of association between the two variables.

As the correlation coefficient value goes towards 0, the relationship between the two variables will be weaker. The coefficient sign indicates the direction of the relationship; a + sign indicates a positive relationship, and a - sign indicates a negative relationship.

Types of Correlation Analysis in Data Mining

Usually, in statistics, we measure four types of correlations: Pearson correlation, Kendall

rank correlation, Spearman correlation, and the Point-Biserial correlation.

1. Pearson r correlation

Pearson r correlation is the most widely used correlation statistic to measure the degree of the relationship between linearly related variables. For example, in the stock market, if we want to measure how two stocks are related to each other, Pearson r correlation is used to measure the degree of relationship between the two. The point-biserial correlation is conducted with the Pearson correlation formula, except that one of the variables is dichotomous. The following formula is used to calculate the Pearson r correlation:

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\ \sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

r_{xy}= Pearson r correlation coefficient between x and y

n= number of observations

x_i = value of x (for ith observation)

y_i= value of y (for ith observation)

2. Kendall rank correlation

Kendall rank correlation is a non-parametric test that measures the strength of dependence between two variables. Considering two samples, a and b, where each sample size is n, we know that the total number of pairings with a b is n(n-1)/2. The following formula is used to calculate the value of Kendall rank correlation

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

Nc= number of concordant

Nd= Number of discordant

3. Spearman rank correlation

Spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. The Spearman rank correlation test does not carry any assumptions about the data distribution. It is the appropriate correlation analysis when the variables are measured on an at least ordinal scale.

This coefficient requires a table of data that displays the raw data, its ranks, and the difference between the two ranks. This squared difference between the two ranks will be shown on a scatter graph, which will indicate whether there is a positive, negative, or no correlation between the two variables. The constraint that this coefficient works under is $-1 \leq r \leq +1$, where a result of 0 would mean that there was no relation between the data whatsoever. The following formula is used to calculate the Spearman rank correlation:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)}$$

ρ= Spearman rank correlation

di= the difference between the ranks of corresponding variables

n= number of observations

When to Use These Methods

The two methods outlined above will be used according to whether there are parameters associated with the data gathered. The two terms to watch out for are:

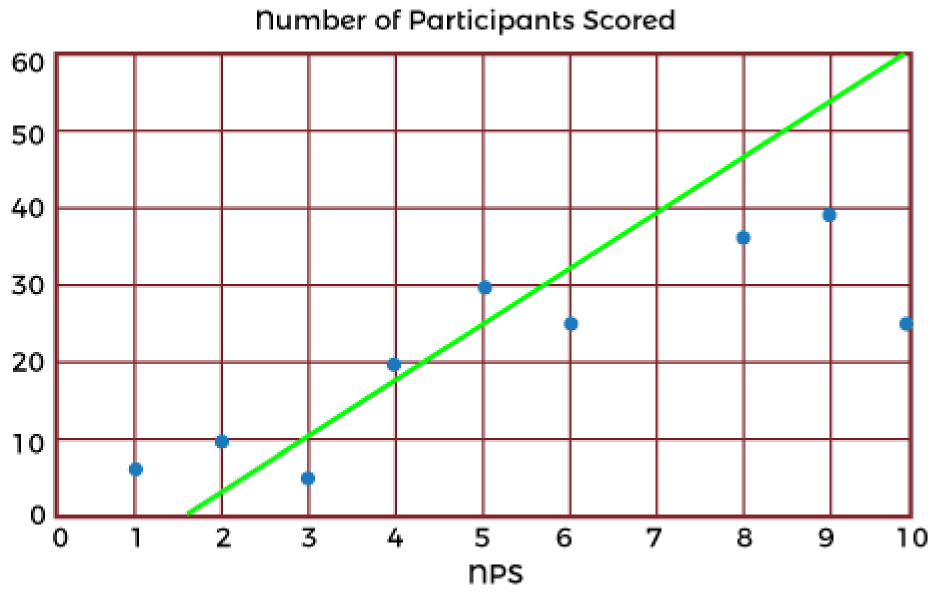
- **Parametric:***(Pearson's Coefficient)* The data must be handled with the parameters of populations or probability distributions. Typically used with quantitative data already set out within said parameters.
- **Non-parametric:***(Spearman's Rank)* Where no assumptions can be made about the probability distribution. Typically used with qualitative data, but can be used with quantitative data if Spearman's Rank proves inadequate.

In cases when both are applicable, statisticians recommend using the parametric methods such as Pearson's Coefficient because they tend to be more precise. But that doesn't mean discounting the non-parametric methods if there isn't enough data or a more specified accurate result is needed.

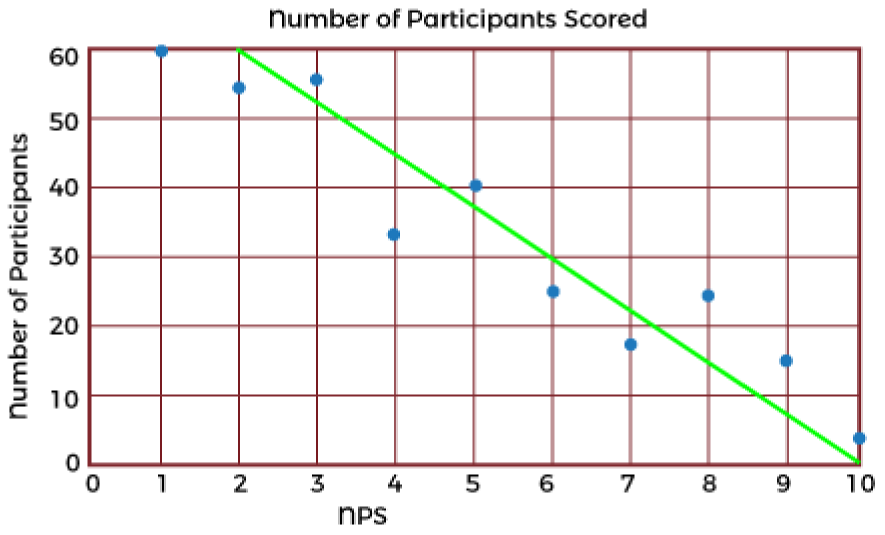
Interpreting Results

Typically, the best way to gain a generalized but more immediate interpretation of the results of a set of data is to visualize it on a scatter graph such as these:

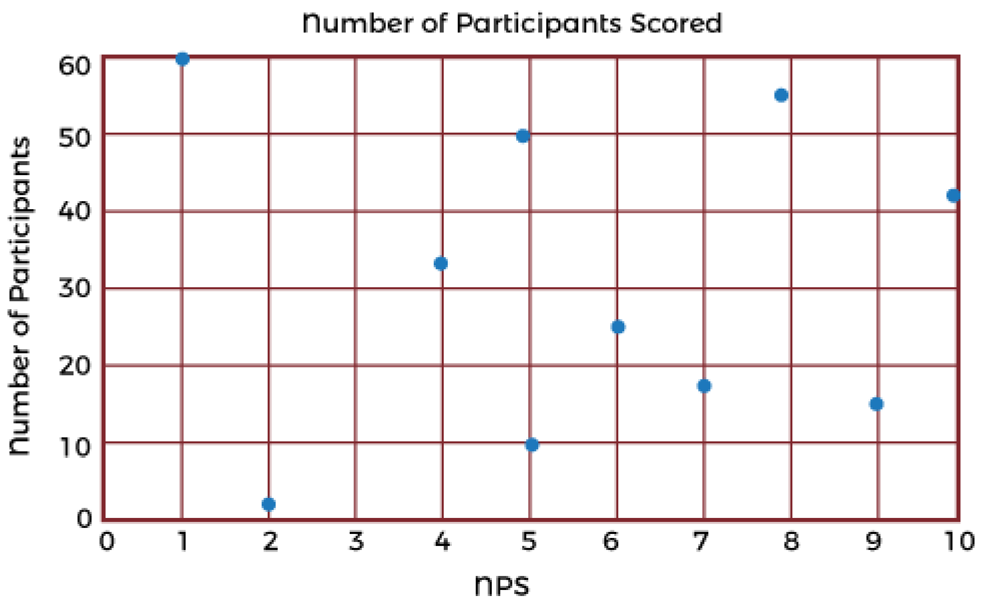
1. **Positive Correlation:** Any score from +0.5 to +1 indicates a very strong positive correlation, which means that they both increase simultaneously. This case follows the data points upwards to indicate the positive correlation. The line of best fit, or the trend line, places to best represent the graph's data.



2.Correlation: Any score from -0.5 to -1 indicates a strong negative correlation, which means that as one variable increases, the other decreases proportionally. The line of best fit can be seen here to indicate the negative correlation. In these cases, it will slope downwards from the point of origin.



3.**No Correlation:** Very simply, a score of 0 indicates no correlation, or relationship, between the two variables. This fact will stand true for all, no matter which formula is used. The more data inputted into the formula, the more accurate the result will be. The larger the sample size, the more accurate the result.



Outliers or anomalies must be accounted for in both correlation coefficients. Using a scatter graph is the easiest way of identifying any anomalies that may have occurred. Running the correlation analysis twice (with and without anomalies) is a great way to assess the strength of the influence of the anomalies on the analysis. Spearman's Rank coefficient may be used if anomalies are present instead of Pearson's Coefficient, as this formula is extremely robust against anomalies due to the ranking system used.

Benefits of Correlation Analysis

Here are the following benefits of correlation analysis, such as:

1. Reduce Time to Detection

In anomaly detection, working with many metrics and surfacing correlated anomalous metrics helps draw relationships that reduce time to detection (TTD) and support shortened time to remediation (TTR). As data-driven decision-making has become the norm, early and robust detection of anomalies is critical in every industry domain, as delayed detection adversely impacts customer experience and revenue.

2. Reduce Alert Fatigue

Another important benefit of correlation analysis in anomaly detection is reducing alert fatigue by filtering irrelevant anomalies (based on the correlation) and grouping correlated anomalies into a single alert. Alert storms and false positives are significant challenges organizations face - getting hundreds, even thousands of separate alerts from multiple systems when many of them stem from the same incident.

3. Reduce Costs

Correlation analysis helps significantly reduce the costs associated with the time spent investigating meaningless or duplicative alerts. In addition, the time saved can be spent on more strategic initiatives that add value to the organization.

6.Constraint based Association Mining

Data mining process may uncover thousands of rules from a given set of data, most of which end up being unrelated or uninteresting to the users. Often, users have a good sense of which

direction‖ of mining may lead to interesting patterns and the —form‖ of the patterns or rules they would like to find. Thus, a good heuristic is to have the users specify such intuition or expectations as *constraints* to confine the search space. This strategy is known as constraint-based mining. The constraints can include the following:

- **Knowledge type constraints:** These specify the type of knowledge to be mined, such as association or correlation.
- **Data constraints:** These specify the set of task-relevant data.
- **Dimension/level constraints:** These specify the desired dimensions (or attributes) of the data, or levels of the concept hierarchies, to be used in mining.
- **Interestingness constraints:** These specify thresholds on statistical measures of rule interestingness, such as support, confidence, and correlation.
- **Rule constraints:** These specify the form of rules to be mined. Such constraints may be expressed as metarules (rule templates), as the maximum or minimum number of predicates that can occur in the rule antecedent or consequent, or as relationships among attributes, attribute values, and/or aggregates.

Metarule-

Guided Mining of Association Rules *"How are metarules useful?"* Metarules allow users to specify the syntactic form of rules that they are interested in mining. The rule forms can be used as constraints to help improve the efficiency of the mining process. Metarules may be based on the analyst's experience, expectations, or intuition regarding the data or may be automatically generated based on the database schema.

Metarule-guided mining:- Suppose that as a market analyst for *AllElectronics*, you have access to the data describing customers (such as customer age, address, and credit rating) as well as the list of customer transactions. You are interested in finding associations between customer traits and the items that customers buy. However, rather than finding *all* of the association rules reflecting these relationships, you are particularly interested only in determining which pairs of customer traits SCE Department of Information Technology promote the sale of office software.A metarule can be used to specify this information describing the form of rules you are interested in finding. An example of such a metarule is

$$P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{"office software"}),$$

where $P1$ and $P2$ are predicate variables that are instantiated to attributes from the given database during the mining process, X is a variable representing a customer, and Y and W take on values of the attributes assigned to $P1$ and $P2$, respectively. Typically, a user will specify a list of attributes to be considered for instantiation with $P1$ and $P2$. Otherwise, a default set may be used.

2. Constraint Pushing: Mining Guided by Rule Constraints

Rule constraints specify expected set/subset relationships of the variables in the mined rules, constant initiation of variables, and aggregate functions. Users typically employ their knowledge of the application or data to specify rule constraints for the mining task. These rule constraints may be used together with, or as an alternative to, metarule-guided mining. In this section, we examine rule constraints as to how they can be used to make the mining process more efficient. Let's study an example where rule constraints are used to mine hybrid-dimensional association rules.

Our association mining query is to *"Find the sales of which cheap items (where the sum of the prices is less than* $100) *may promote the sales of which expensive items (where the minimum price is* $500) *of the same group for Chicago customers in 2004."* This can be expressed in the DMQL data mining query language as follows,

(1) mine associations as
(2) $lives\_in(C, \_, \text{"Chicago"}) \wedge sales^{+}(C, ?\{I\}, \{S\}) \Rightarrow sales^{+}(C, ?\{J\}, \{T\})$
(3) from sales
(4) where S.year = 2004 and T.year = 2004 and I.group = J.group
(5) group by C, I.group
(6) having sum(I.price) < 100 and min(J.price) ≥ 500
(7) with support threshold = 1%
(8) with confidence threshold = 50%

7. Graph Pattern Mining:

Data mining is the process of collecting and processing data from a heap of unprocessed data. When the patterns are established, various relationships between the datasets can be identified and they can be presented in a summarized format which helps in statistical analysis in various industries. Among the other data structures, the graph is widely used in modeling advanced structures and patterns. In data mining, the graph is used to find subgraph patterns for discrimination, classification, clustering of data, etc. The graph is used in network analysis. By linking the various nodes, graphs form network-like communications, web and computer networks, social networks, etc. In multi-relational data mining, graphs or networks is used because of the varied interconnected relationship between the datasets in a relational database.

Graph mining

Graph mining is a process in which the mining techniques are used in finding a pattern or relationship in the given real-world collection of graphs. By mining the graph, frequent substructures and relationships can be identified which helps in clustering the graph sets, finding a relationship between graph sets, or discriminating or characterizing graphs. Predicting these patterning trends can help in building models for the enhancement of any application that is used in real-time. To implement the process of graph mining, one must learn to mine frequent subgraphs.

Frequent Subgraph Mining

Let us consider a graph h with an edge set E(h) and a vertex set V(h). Let us consider the existence of subgraph isomorphism from h to h' in such a way that h is a subgraph of h'. A label function is a function that plots either the edges or vertices to a label. Let us consider a labeled graph dataset, Let us consider s(h) as the support which means the percentage of graphs in F where h is a subgraph. A frequent graph has support that will be no less than the minimum support threshold. Let us denote it as min_support.

Steps in finding frequent subgraphs:

There are two steps in finding frequent subgraphs.

- The first step is to create frequent substructure candidates.
- The second step is to find the support of each and every candidate. We must optimize and enhance the first step because the second step is an NP-completed set where the computational complexity is accurate and high.

8. Sequential Pattern Mining (SPM):

Sequential pattern mining is the mining of frequently appearing series events or subsequences as patterns. An instance of a sequential pattern is users who purchase a Canon digital camera are to purchase an HP color printer within a month.

For retail information, sequential patterns are beneficial for shelf placement and promotions. This industry, and telecommunications and different businesses, can also use sequential patterns for targeted marketing, user retention, and several tasks.

There are several areas in which sequential patterns can be used such as Web access pattern analysis, weather prediction, production processes, and web intrusion detection.

Given a set of sequences, where each sequence includes a file of events (or elements) and each event includes a group of items, and given a user-specified minimum provide threshold of min sup, sequential pattern mining discover all frequent subsequences, i.e., the subsequences whose occurrence frequency in the group of sequences is no less than min_sup.

Let $I = \{I_1, I_2, ..., I_p\}$ be the set of all items. An itemset is a nonempty set of items. A sequence is an ordered series of events. A sequence s is indicated $\{e_1, e_2, e_3 \ldots e_l\}$ where event e_1 appears before e_2, which appears before e_3, etc. Event e_j is also known as element of s.

In the case of user purchase information, an event defines a shopping trip in which a customer purchase items at a specific store. The event is an itemset, i.e., an unordered list of items that the customer purchased during the trip. The itemset (or event) is indicated $(x_1x_2\cdots x_q)$, where x_k is an item.

An item can appear just once in an event of a sequence, but can appear several times in different events of a sequence. The multiple instances of items in a sequence is known as the length of the sequence. A sequence with length l is known as l-sequence.

A sequence database, S, is a group of tuples, (SID, s), where SID is a sequence_ID and s is a sequence. For instance, S includes sequences for all users of the store. A tuple (SID, s) is include a sequence α, if α is a subsequence of s.

This phase of sequential pattern mining is an abstraction of user-shopping sequence analysis. Scalable techniques for sequential pattern mining on such records are as follows −

There are several sequential pattern mining applications cannot be covered by this phase. For instance, when analyzing Web clickstream series, gaps among clicks become essential if one required to predict what the next click can be.

In DNA sequence analysis, approximate patterns become helpful because DNA sequences can include (symbol) insertions, deletions, and mutations. Such diverse requirements can be considered as constraint relaxation or application.