References :

1. Y. Cai, N. Cercone, and J. Han. Attribute-oriented induction in relational databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, pages 213-228. AAAI/MIT Press, 1991.
2. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
3. C. Carter and H. Hamilton. Efficient attribute-oriented generalization for knowledge discovery from large databases. IEEE Trans. Knowledge and Data Engineering, 10:193-208, 1998.
4. W. Cleveland. Visualizing Data. Hobart Press, Summit NJ, 1993.
5. J. L. Devore. Probability and Statistics for Engineering and the Science, 4th ed. Duxbury Press, 1995.
6. T. G. Dietterich and R. S. Michalski. A comparative review of selected methods for learning from examples. In Michalski et al., editor, Machine Learning: An Artificial Intelligence Approach, Vol. 1, pages 41-82. Morgan Kaufmann, 1983.
7. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
8. J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. IEEE Trans. Knowledge and Data Engineering, 5:29-40, 1993.

Unit 5

MINING ASSOCIATION RULES IN LARGE DATABASES

INTRODUCTION

Many business enterprises accumulate large quantities of data continuously from their day-to-day operations. The discovery of interesting associations and relationships among huge amount of business transaction records can help in decision making processes. For this reason, they are becoming interested in mining association rules from their databases.

There are numerous applications, such as Basket data analysis, cross-marketing, inventory management, catalog design, loss-leader analysis, clustering, and classification, etc., come into this frame work. A typical example of association rule mining is **market basket transactions**. In this process, a huge amounts of customer purchase data are collected daily at the checkout counters of grocery stores. It is used to analyze customer buying habits by finding association between the different items that customer place in their "shopping baskets" as in Fig 5.1. Each row in Table 5.1 corresponds to a transaction, which contains unique identifier labeled TID and a set of items bought by a given customer.

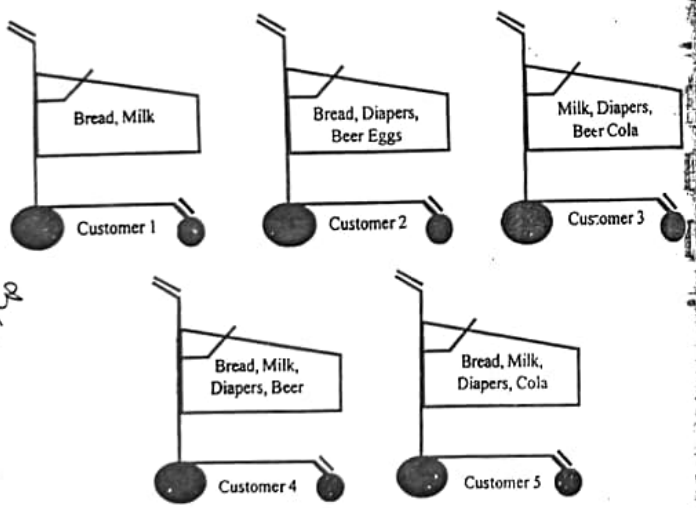

Fig 5.1: Market Basket Transactions

The discovery of such association can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. Associative rule mining searches for interesting relationships among items in a given data set.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Table 5.1: Market Basket Transactions

For instance, if customers are buying diapers or milk , how likely are they to also buy beer or bread respectively? Using this information, the retailer can increase their sales by marketing and plan their shelf space. For example, placing diapers, beer and milk, bread within close proximity may further encourage the sale of these items together within single visits to the store. The association rules of the above data are

{Diapers} → {Beer},
{Milk} → { Bread },

5.1 Association Mining

In this section, the basic definitions and concepts of the association rules are introduced.

5.1.1 What is an association rule?

- Association rule mining searches for interesting relationships among items in a given data set.
- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.
- Searching for interesting relationships between set of items or objects in transactional databases, relational databases, and other information repositories.

5.1.2 Basic Concepts

Definition 5.1: Let $I = \{i_1, i_2, ..., i_d\}$ be the set of all items and $T = \{t_1, t_2, ..., t_N\}$ be the set of all transactions. Each transaction t_i contains a subset of items chosen from I. In association analysis, a collection of zero or more items is termed an **itemset**. If an itemset contains k items, it is called a **k-itemsets**.

Example 5.1: For instance, {Beer, Diapers, Milk} is an example of a 3-itemset. The null (or empty) set is an itemset that does not contain any items.

Definition 5.2: An important property of an itemset is its **support count**, which refer to the number of transactions that contain a particular itemset. Mathematically, the support count, $\sigma(X)$, for an itemset X can be stated as follows:

$$\sigma(X) = |\{t_i | X \subseteq t_i, t_i \in T\}|,$$

Where the symbol $|.|$ denote the number of elements in the set.

Example 5.2: In the data set shown in Table 5.1, the support count for {Beer, Diapers, Milk} is equal to two because there are only two transactions(3 & 4) that contain all three items.

$\sigma(\{Beer, Diapers, Milk\}) = |\{3,4\}| \{Beer, Diapers, Milk\} \subseteq \{3,4\}, \{3,4\} \in \{1,2,3,4,5\}|$
$= 2.$

Definition 5.3: An association rule is an implication expression of the form $X \rightarrow Y$, where X and Y are disjoint itemsets, i.e., $X \cap Y = \phi$. The strength of an association rule can be measured in terms of its **support(s) and confidence(c). Support** determines how often a rule is applicable to a given data set, while **confidence** determines how frequently items in Y appears in transactions that contain X.

Support is often used to eliminate uninteresting rules. support also has a desirable property that can be exploited for the efficient discovery of association rules. Confidence, on the other hand, measures the reliability of the inference made by a rule.

The formal Definitions of these metrics are

$$\text{Support, } s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}.$$

$$\text{Confidence, } c((X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(x)}$$

Example 5.3: Consider the rule {Milk, Diapers}→ {Beer} from Table 5.1.
σ(Milk, Diapers, Beer) = 2
N = number of transactions = 5
σ(Milk, Diapers) = 3

$$\text{Support, } s(\{Milk, Diapers\} \rightarrow \{Beer\}) = \frac{\sigma(Milk, Diapers, Beer)}{N} = \frac{2}{5} = 0.4$$

$$\text{Confidence, } c(\{Milk, Diapers\} \rightarrow \{Beer\}) = \frac{\sigma(Milk, Diapers, Beer)}{\sigma(Milk, Diapers)} = \frac{2}{3} = 0.67$$

Definition 5.4: Given a set of transactions T, the association rule $X \rightarrow Y$(support, confidence), mining is to find all rules having

- support $\geq$ *minsup*
- confidence $\geq$ *minconf*,

where *minsup and minconf* are the corresponding support and confidence thresholds(User specified value).

The intuitive meaning of such a rule is that a transaction of the database which contains X tends to contain Y. Given a set of transactions, T, the problem of mining association rules is to discover all rules that have support and confidence greater than or equal to the user-specified minimum support and minimum confidence, respectively.

Example 5.4: Let minimum support is 0.5, and minimum confidence is 0.5, we have association rules
{Diapers} ⟶ {Beer}(0.60, 0.75),
{Milk} ⟶ { Bread }(0.60, 0.75).

Fig 5.2 gives the associations between diapers and beer; milk and bread

Definition 5.5: A **Frequent Itemset** is an itemset whose support is greater than or equal to a *minsup* threshold. Mathematically, let T be the transaction database and σ be the user-specified minimum support. An itemset $X \subseteq A$ is said to be a frequent itemset in T with respect to σ, if

$$s(X) \geq \sigma$$

Example5.5: Consider again the Market basket data of Fig 5.1 and *minsup* = 0.5. Among five transactions Diphers comes in 4 transactions. Therfore, s(Diphers) = 4/5 = 0.8 > *minsup* and so Diphers is a frequent itemset.

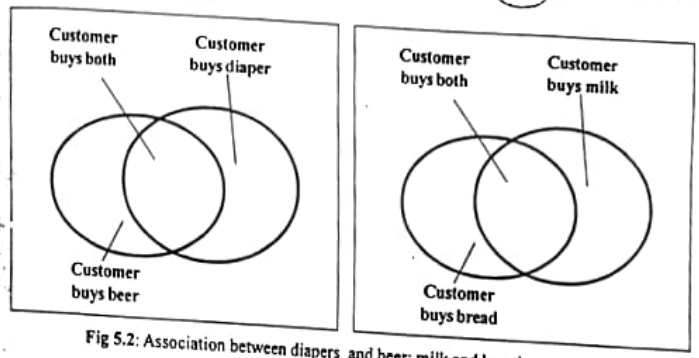

Fig 5.2: Association between diapers and beer; milk and bread.

You can also see that the set of frequent sets for a given T, with respect to a given σ, exhibits some interesting properties.

- **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.

Definition 5.6: Binary Representation of data can be represented in a binary format, where each row corresponds to a transaction and each column corresponds to an item. An item can be treated as a binary variable whose value is one if the item is present in a transaction and zero otherwise.

Example 5.6: Binary Representation of Market basket data of Fig5.1 shown in Table 5.2. This representation is perhaps a very simplistic view of real market basket data because it ignores certain important aspects of the data such as the quantity of items sold or the price paid to purchase them. This representation is more effectively used to find association rules.

TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Table 5.2: A binary 0/1 representation of market basket data.

Brute-force approach

- A brute-force approach for mining association rules is to list all possible association rules and compute the support and confidence for each rule.
- This approach prohibitively expensive because there are exponentially many rules that can be extracted from a data set.
- More specifically, the total number of possible rules extracted from a data set that contains m items is

$$R = 3^m - 2^{m+1} + 1.$$

- Even for the small data set shown in Table 5.1, this approach requires us to compute the support and confidence for $3^6 - 2^7 + 1 = 602$ rules.
- More than 80% of the rules are discarded after applying minsup = 0.5 and minconf = 0.5, thus making most of the computations become wasted.
- To avoid performing needless computations, it would be useful to prune the rules early without having to compute that support and confidence values.

Example 5.7:

{Milk,Diaper} ⟶ {Beer} (s=0.4, c=0.67)
{Milk,Beer} ⟶ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} ⟶ {Milk} (s=0.4, c=0.67)
{Beer} ⟶ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} ⟶ {Milk,Beer} (s=0.4, c=0.5)
{Milk} ⟶ {Diaper,Beer} (s=0.4, c=0.5)

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}.
- Rules originating from the same itemset have identical support but can have different confidence.
- If the itemset is infrequent, then all six candidate rules can be pruned immediately without computing their confidence values.

Therefore, the problem of mining association rules can be decomposed into two sub problems.

- **Frequent Itemset Generation:** Generate all itemsets whose support greater than or equal to *minsup*.
- **Rule Generation:** Generate high confidence rules from each frequent itemsets found in the previous step. These rules are called strong rules. The general idea is that if say, ABCD and AB are frequent itemsets, then we can determine if the rule AB → CD holds by checking the following inequality.

$\frac{s(\{A,B,C,D\})}{s(\{A,B\})} \geq minconf$, where s(X) is the support of X in T.

Many researchers have focused on the first sub problem, because the overall performance of mining association rules is determined by it.

5.1.3 Types of Association Rule Mining

Market Basket Analysis is one form of association rule mining. In fact there are many kinds of association rules, these rules can be classified in various ways based on the following criteria:

Based on types of values handled in the rule: It is categorized into Boolean association and Quantitative association rules.

- **Boolean Association rule:** This rule concerns association between the presence or absence of items.

  Eg: Computer → fin_mgmt_sw

- **Quantitative Association rule** - This rule describes associations between quantitative items or attributes. In these rules, quantitative values for items or attributes are partitioned into intervals. The attributes age and income have been discretized.

Eg:Age(X,"30....39")^income(X,"42K....48K") → buys(X, High.resol. TV)

Based on the dimension of data involved in the rule: These are Single dimensional Associative rules and Multi dimensional Associative rules.

- **Single dimensional Associative rule:** In single dimensional association rule, the items or attributes in an association rule reference only one dimension. The rule(5.1) can be written as

  Eg: buys(X,"computer") → buys(X,"fin_mgmt_sw")

- **Multi dimensional Associative rule:** The rule that refers two/more dimensions such as buys, time_of_transaction and customer_category etc., The Rule (5.2) is also treated as an example of multi dimensional Associative rule since it involves three dimensions: age, income and buys.

  Eg: Age (X,"30....39")^income(X,"42K....48K") → buys(X,High.resol. TV)

Based on the levels of abstractions involved in the rule set: Some methods for association rule mining can find rules at different levels of abstraction.

- **Single level Association rules:** The rules with in a given set do not reference items as attributes at different levels of abstraction.
- **Multilevel Associative rule:** The items bought are referenced at different levels of abstraction.

  Eg: Age(X,"30..39") → buys(X,"laptop")
  Age(X,"30..39") → buys(X,computer")

  In rules (5.5) and (5.6), the item computer is a higher level abstraction of laptop.

Based on various extensions to association mining: Association mining can be extended to correlation analysis, where the absence or presence of correlation items can be identified. It can also extended to mining max patterns(maximal frequent patterns) and closed itemsets.

5.2 Mining single dimensional Boolean Association Rule from Transactional DataBases

This section describes simplest form of association rules such as single dimensional, single-level and Boolean association rules. First it presents Apriori, a basic algorithm for finding frequent itemset, and procedure for generating strong association rules from frequent itemset.

A lattice structure can be used to enumerate the list of all possible itemsets. Fig5.3 shows an itemset lattice for A = {a, b, c, d, e}. In general, a data set that contains k items can potentially generate up to $2^k - 1$ frequent itemsets, excluding the null set. Because k can be very large in many practical applications, the search space of itemsets that need to be explored is exponentially large.

A brute-force approach for finding frequent itemsets is to determine the support count for every candidate itemset in the lattice structure. To do this, we need to compare each candidate against every transaction. If the candidate is contained in a transaction, its support count will be incremented. For example, the support for {Bread, Milk} is incremented three times because the itemset is contained in transactions 1, 4 and 5. Such an approach can be very expensive because it requires O(NMw) comparisons, where N is the number of transactions, $M = 2^k - 1$ is the number of candidate itemsets, and w is the maximum transaction width.

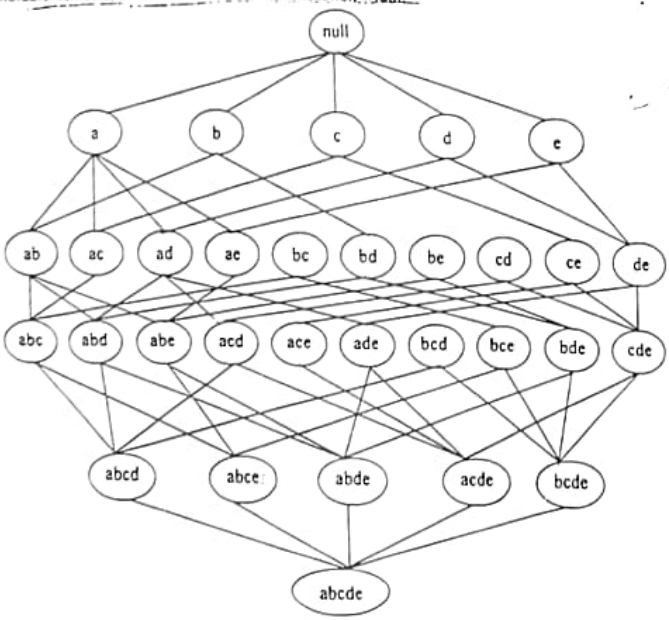

Fig 5.3: Latice of subset

There are several ways to reduce the computational complexity of frequent itemset generation.

- **Reduce the number of candidate itemsets(M)**: The Apriori algorithm, discussed next sub section, is an effective way to eliminate some of the candidate itemsets without counting their support values.
- **Reduce the number of comparisons**: Instead of matching each candidate itemset against every transaction, we can reduce the number of comparisions by using more advanced data structures, either to store the candidate itemsets or to compress the data set, discussed in later sections.

Example 5.8: Consider the following database. $A = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$. Assume *minsup* = 0.2. Since T contains 15 records, it means that an itemset that is supported by at least three transactions is a frequent set. We shall use this database for illustration of Apriori algorithm.

TID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
1	1	0	0	0	1	1	0	1	0
2	0	1	0	1	0	0	0	1	0
3	0	0	0	1	1	0	1	0	0
4	0	1	1	0	0	0	0	0	0
5	0	0	0	0	1	1	1	0	0
6	0	1	1	1	0	0	0	0	0
7	0	1	0	0	0	1	1	0	1
8	0	0	0	0	1	0	0	0	0
9	0	0	0	0	0	0	0	1	0
10	0	0	1	0	1	0	1	0	0
11	0	0	1	0	1	0	1	0	0
12	0	0	0	0	1	1	0	1	0
13	0	1	0	1	0	1	1	0	0
14	1	0	1	0	1	0	1	0	0
15	0	1	1	0	0	0	0	0	1

Table 5.3 Sample Database

5.2.1 The Apriori Algorithm

The apriori algorithm is also called the *level-wise* algorithm. It was proposed by Agrawal and Srikanth in 1994. It is the most popular algorithm to find all the frequent sets. It makes use of the downward closure property. As name suggests, the algorithm is a bottom-up search, moving downward level-wise in the lattice. However, the nicety of the method is that before reading the database at every level, it graciously prunes many of the sets which are unlikely to be frequent sets.

The first pass of the algorithm simply counts item occurrences to determine the frequent itemsets. A subsequent pass, say pass k, consists of two phases :

- First, the frequent itemsets L_{k-1} found in the $(k-1)^{th}$ pass are used to generate candidate itemsets C_k using the apriori candidate generation procedure described below.
- Next, the database is scanned and support of candidates in C_k is counted. For fast counting, we need to efficiently determine the candidate in C_k contained in a given transaction t.

The set of candidate itemsets is subjected in pruning process to ensure that all the subsets of the candidate sets are already known to be frequent itemsets. The candidate generation process and the pruning process are the most important parts of this algorithm. We shall describe these two processes separately below.

Candidate Generation

Given L_{k-1}, the set of all frequent (k-1)-itemsets, we want to generate a superset of the set of all frequent k-itemset. The intuition behind the apriori candidate-generation procedure is that if an itemset X has minimum support, so do all subsets of X.

Let us assume that the set of frequent 3-itemsets are {1, 2, 3}, {1, 2, 5}, {1, 3, 5}, {2, 3, 5}, {2, 3, 4}. Then, the 4-itemsets that are generated as candidate itemsets are the super sets of these 3-itemsets. In addition, all the 3-itemset subsets of any candidate 4-itemsets must be already known to be in L_3.

The candidate-generation method:

gen_candidate_itemsets with the given L_{k-1} as follows:

$C_k = \emptyset$
for all itemsets $l_1 \in L_{k-1}$ **do**
for all itemsets $l_2 \in L_{k-1}$ **do**
if $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[k-1] < l_2[k-1]$
then $c = l_1[1], l_1[2] \ldots l_1[k-1], l_2[k-1]$
$C_k = C_k \cup \{c\}$

Using this algorithm: $C_4 := \{\{1, 2, 3, 5\}, \{2, 3, 4, 5\}\}$ is obtained from $L_3 := \{\{1, 2, 3\}, \{1, 2, 5\}, \{1, 3, 5\}, \{2, 3, 5\}, \{2, 3, 4\}\}$. {1, 2, 3, 5} is generated by from {1, 2, 3} and {1, 2, 5}. Similarly, {2, 3, 4, 5} is generated from {2, 3, 4} and {2, 3, 5}. No other pair of 3-itemset satisfy the condition.
$l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \ldots \wedge l_1[k-1] < l_2[k-1]$

Pruning :

The pruning step eliminates the extensions of (k-1)-itemset which are not found to be frequent. For example, from C_4, the itemset {2, 3, 4, 5} is pruned, since all its 3-subsets are not in L_3 (clearly, {2, 4, 5} is not in L_3).

The pruning algorithm :

Prune(C_k)
for all $c \in C_k$
for all (k-1)-subsets d of c **do**
if $d \notin L_{k-1}$
then $C_k = C_k / \{c\}$

The Apriori frequent itemset discovery algorithm uses these two functions (candidate generation and pruning) at every iteration. It moves downward in the lattice starting from level 1 till level k, where no candidate set remains after pruning.

Apriori Algorithm

Initialize: k = 1, C_1 = all the 1-itemsets;
read the database to count the support of C_1 to determine L_1
$L1$ = {frequent 1-itemsets};
k = 2; // k represents the pass number //
while ($L_{k-1} \neq \emptyset$) **do**
begin
C_k = **gen_candidate_itemsets** with the given L_{k-1}
prune(C_k)
for all transactions $t \in T$ **do**
increment the count of all candidates in C_k that are contained in t;
L_k = All candidates in C_k with minimum support;
K = k+1;
end
Answer = $\cup_k L_k$;

Example 5.9: We illustrate the working of the algorithm with Example 5.2 discussed above.

k = 1
Read the database to count the support of 1-itemsets (Table 5..3). The frequent 1-itemsets and their support counts are given below.
$L1$ = { {2}→ 6, {3}→ 6, {4}→ 4, {5}→ 8, {6}→ 5, {7}→ 7, {8}→ 4}.

k = 2
In the candidate generation step, we get
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}}
The Pruning step does not change C_2
Read the database to count the support of elements in C_2 to get
L_2 = { {2, 3}→ 3, {2, 4}→ 3, {3, 5}→ 3, {3, 7}→ 3, {5, 6}→ 3, {5, 7}→ 5, {6, 7}→ 3}

k = 3
In the candidate generation step,
using {2, 3} and {2, 4}, we get {2, 3, 4}
using {3, 5} and {3, 7}, we get {3, 5, 7} and
similarly from {5, 6} and {5, 7}, we get {5, 6, 7}.
C_3 = { {2, 3, 4}, {3, 5, 7}, {5, 6, 7}}.

The Pruning step prunes {2, 3, 4} as not all subsets of size 2, i.e., {2, 3}, {2, 4}, {3, 4} are present in L_2. The other two itemsets are retained.
Thus the pruned C_3 is { {3, 5, 7}, {5, 6, 7}}.

Read the database to count the support of the itemsets in C_3 to get
$L_3 = \{ \{3, 5, 7\} \rightarrow 3 \}$.
$k = 4$
Since L_3 contains only one element, C_4 is empty and hence the algorithm stops, returning the set of frequent sets along with their respective support values as
$L = L1 \cup L_2 \cup L_3$
Therefore, $L = \{ \{2\} \rightarrow 6, \{3\} \rightarrow 6, \{4\} \rightarrow 4, \{5\} \rightarrow 8, \{6\} \rightarrow 5, \{7\} \rightarrow 7, \{8\} \rightarrow 4, \{2, 3\} \rightarrow 3, \{2, 4\} \rightarrow 3, \{3, 5\} \rightarrow 3, \{3, 7\} \rightarrow 3, \{5, 6\} \rightarrow 3, \{5, 7\} \rightarrow 5, \{6, 7\} \rightarrow 3, \{3, 5, 7\} \rightarrow 3 \}$.

Drawbacks Of Apriori algorithm

As we have seen, in many cases the Apriori candidate generation method reduces the size of candidate sets significantly and leads to good performance gain. However, it may suffer from two nontrivial costs.

- It may need to generate a huge number of candidate sets. For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets and accumulate and test their occurrence frequencies.
- It may need to repeatedly scan the database and check a large set of candidates by pattern matching.

5.2.2 Variations Of Apriori

As we have already stated that Apriori may be defined and derived in different manners. Here I am presenting two different variations of Apriori as AprioriAll and AprioriSome where we can consider the performance of both the problems of these algorithms.

AprioriAll Algorithm

The algorithm is given below:

- In each pass, we use the large sequences from the previous pass to generate the candidate sequences and then measure their support by making a pass over the database.
- At the end of the pass, the support of the candidates is used to determine the large sequences.
- In the first pass, the output of the 1-itemset phase is used to initialize the set of large 1-sequences.
- The candidates are stored in *hash-tree* to quickly find all candidates contained in a customer sequence.

AprioriAll()

L_1 = large 1-sequences; // Result of litemset phase
for (k = 2; L_{k-1} 0; k++) **do**
begin
C_k = New Candidates generated from L_{k-1} (see below)
for each customer-sequence *c* in the database **do**
Increment the count of all candidates in C_k that are contained in *c*.
L_k = Candidates in C_k with minimum support.
end
Answer = Maximal Sequences in L_k ;

Algorithm AprioriSome

This algorithm is described as follows:

- In the forward pass, we only count sequences of certain lengths. For example, we might count sequences of length 1, 2, 4 and 6 in the forward phase and count sequences of length 3 and 5 in the backward phase.
- The function next takes as parameter the length of sequences counted in the last pass and returns the length of sequences to be counted in the next pass.
- Thus, this function determines exactly which sequences are counted, and balances the tradeoff between the time wasted in counting non-maximal sequences versus counting extensions of small candidate sequences.
- One extreme is next (k) = k + 1 (k is the length for which candidates were counted last), when all non-maximal sequences are counted, but no extensions of small candidate sequences are counted.
- In this case, AprioriSome degenerates into AprioriAll.
- The other extreme is a function like next (k) = 100 * k, when almost no non-maximal large sequence is counted, but lots of extensions of small candidates are counted.

// **Forward Phase**
L_1 = large 1-sequences; // Result of litemset phase
$C_1 = L_1$; // so that we have a nice loop condition
last = 1; // we last counted C_{last}
for (k = 2; C_{k-1} =0 and L_{last} =0; k++) **do**
begin
if (L_{k-1} known) **then**
C_k = New candidates generated from L_{k-1} ;
else
C_k = New candidates generated from C_{k-1} ;
if (k == next(last)) **then begin**
for each customer-sequence *c* in the database **do**
Increment the count of all candidates in C_k that are contained in *c*.
L_k = Candidates in C_k with minimum support.
last = k;
end
end

// **Backward Phase**
for (k--; k >=1; k==) **do**
if (L_k not found in forward phase) **then begin**
Delete all sequences in C_k contained in some L_i, i>k;
foreach customer-sequence in D_T **do**
Increment the count of all candidates in C_k that are contained in *c*.
L_k = Candidates in C_k with minimum support.
end
else // L_k already known
Delete all sequences in L_k contained in some L_i, i > k.
Answer = L_k ;

AprioriSome Algorithm

Let hit_k denote the ratio of the number of large k-sequences to the number of candidate k-sequences (i.e., $|L_k| / |C_k|$). The *next* function we used in the experiments is given below. The intuition behind the heuristic is that as the percentage of candidates counted in the current pass which had minimum support increases, the time wasted by counting extensions of small candidates when we skip a length goes down.

```
function next(k: integer)
begin
if (hit_k < 0.666) return k + 1;
elsif (hit_k <0.75) return k + 2;
elsif (hit_k < 0.80) return k + 3;
elsif (hit_k < 0.85) return k + 4;
else return k + 5;
end
```

We use the gen_candidate_itemsets function given in section 5.2.1 to generate new candidate sequences. However, in the k^{th} pass, we may not have the large sequence set L_{k-1} available as we did not count the (k-1)-candidate sequences. In that case, we use the candidate set C_{k-1} to generate C_k. Correctness is maintained because $C_{k-1} \supseteq L_{k-1}$.

In the backward phase, we count sequences for the lengths we skipped over during the forward phase, after first deleting all sequences contained in some large sequence. These smaller sequences cannot be in the answer because we are only interested in maximal sequences. We also delete the large sequences found in the forward phase that are non-maximal.

5.2.3 Methods to Improve Apriori's Efficiency

Many other methods have been proposed to improve the efficiency of the original algorithm. Several of these techniques are given below:

- **Hash-based technique (hashing itemset counts):** A hash- based technique can be used to reduce the size of the candidate k-itemsets, C_k, for k >1.
- **Transaction reduction (reducing the number of transactions scanned in future iterations):** A transaction that does not contain any frequent k-itemsets cannot contain any frequent (k + 1) –itemset. Such a transaction can be marked or removed from further consideration since subsequent scans of the database for j-itemsets, where j>k, will not require it.
- **Partitioning (partitioning the date to find candidate itemsets):** A partitioning requires just two database scans to mine the frequent itemsets. It consists of two phase in which first phase, the algorithms subdivides the transaction into two nonoverlapping partitions and first scan is done and local frequent itemset is found, and in the second phase, the second scan is conducted in which the actual support of each candidate is assessed in order to determine the global frequent itemset.
- **Sampling (mining on a subset of the given data):** The basic idea of the sampling approach is to pick a random sample S of the given data D, and then search for frequent itemsets in S instead of D.

- **Dynamic itemset counting (adding candidate itemsets at different point during scan):** A dynamic itemset counting technique was proposed in which the database is partitioned into blocks marked by start points. The technique is dynamic in that it estimates the support of all of the itemsets that have been counted so far, adding new candidate itemsets if all of their subsets are estimated to be frequent. This algorithm requires fewer database scans.

5.2.4 Mining Frequent Item Sets Without Candidate Generation (FP-Tree Growth Algorithm)

The apriori gen_candidate_itemsets method reduces the size of candidates sets significantly and leads to good performance gain. However it may need to suffer from two non trivial costs.

- It is costly to handle large number of candidate sets. For instance, if there are 10^4 frequent 1-itemsets, then approximately, 10^7 candidate 2-itemsets are generated. Moreover, if there is a frequent set of size 100, then roughly 10^{30} candidate sets are generated in this process.
- It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching.

Can we generate a complete set of frequent itemsets with out candidate generation? Keeping this in mind, a new class of algorithms have recently been populated which avoids the generation of large number of candidate sets. We describe one such method, called the FP-Tree Growth algorithm as follows:

- The main idea of the algorithm is to maintain a Frequent Pattern Tree (FP-Tree) of the database.
- It is an extended prefix-tree structure, storing crucial and quantitative information about frequent sets.
- The tree nodes are frequent items and are arranged in such a way that more frequently occurring nodes will have better chances of sharing than the less frequently occurring ones.
- The method starts from frequent 1-itemsets as an initial suffix pattern and examines only its conditional pattern base (a subset of the database), which consists of the set of frequent items co-occurring with the suffix pattern.
- The algorithm constructs the conditional FP-Tree and performs mining on this tree.

The algorithm involves two phases :

- In Phase I, it constructs the FP-Tree with respect to a given *minsup*. The construction of this tree requires two passes over the whole database.
- In Phase II, the algorithm does not use the transaction database any more, but it uses the FP-tree. Interestingly, the FP-tree contains all the information about frequent itemsets with respect to the given σ.

Definition 5.7: A frequent pattern tree (or FP-tree) is tree structure consisting of an item-prefix-t and a frequent-item-header table.

- Item-prefix-tree:
  - It consists of a root node labeled null.
  - Each non-root node consists of three fields:
    - Item name,
    - support count, and
    - node link.
- frequent-item-header-table: It consists of two fields:
  - Item name
  - Head of node link which points to the first node in the FP-Tree carrying the ite name.

It may be noted that the FP-tree is dependent on the support threshold *minsup*. For differe values of *minsup*, the trees are different. There is another typical feature of the FP-tree; it depends the ordering of the items. The ordering that is followed in the original paper is the decreasing ord of the support counts. However, different ordering may offer different advantages. Thus, the hea table arranged in this order of the frequent items. Let us study the construction of a FP-tree from transaction database.

- We make one scan of the database T and compute L_1, the set of frequent 1-itemsets. F convenience, let us call this the set of frequent items. In other words, we refer to L_1 as of items rather than a class of singleton sets.
- Then we sort the items in L1 in the decreasing order of frequency counts.
- From this stage onwards, the algorithm ignores all the non-frequent items from the transacti and views any transaction as a list of frequent items in the decreasing order of frequen
- Without any ambiguity, we can refer to the transaction t as such a list. The first element the list corresponding to any given transaction, is the most frequent item among the ite supported by t.
- For a list t, we denote head_t as its first element and body_t as the remaining part of the l (the portion of the list t after removal of head_t). Thus, t is [head_t | body_t].
- The FP-tree construction grows the tree recursively using these concepts.

FP-tree construction Algorithm

```
create a root node., root of the FP-Tree and label it as null.
    do for every transaction t
        if t is not empty
        insert(t, root)
        link the new nodes to other node with similar labels links originating from header list.
    end do
return FP-Tree
insert(t, any_node)
    do while t is not empty
        if any_node has a child with label head_t
```



```
        then increment the link count between any_node and head_t by 1
        else create a new child node of any_node with label; head_t with link count 1.
        call insert(body_t, head_t)
    end do
```

A1	A2	A3	A4	A5	A6	A7	A8	A9
1	0	0	0	1	1	0	1	0
0	1	0	1	0	0	0	1	0
0	0	0	1	1	0	1	0	0
0	0	1	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0
0	1	1	1	0	0	0	0	0
0	1	0	0	0	1	1	0	1
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0
0	0	1	1	1	0	1	0	0
0	0	1	1	1	0	1	0	0
0	0	0	0	1	1	0	1	0
1	0	1	1	0	1	1	0	0
0	1	1	0	0	0	1	0	0

Table 5.4

Example 5.10: Let us consider the database given in Table 5.4.. The frequent items are 2, 3, 4, 5, 6, 7 and 8. If we sort them in the order of their frequency, then they appear in the order 5, 3, 4, 7, 2, 6 and if the transactions are written in terms of only frequent items, then the transactions are

5, 6, 8
4, 2, 8
5, 4, 7
3
5, 7, 6
3, 4, 2
7, 2, 6
5
8
5, 3, 4, 7
5, 3, 4, 7
5, 6, 8
3, 4, 7, 2, 6
5, 3, 4, 7
3, 2

The scan of the first transaction leads to the construction of the first branch of the tree (Fig 5.4). Notice that the branch is not ordered in the same way as the transaction appears in the database. The items are ordered according to the decreasing order of frequency of the frequent items. The complete table is given in the Fig 5.5.

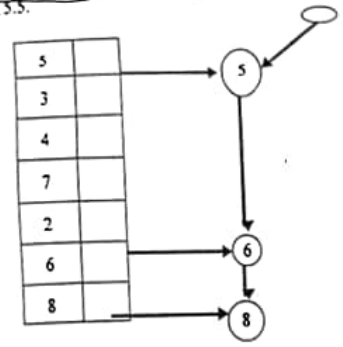

Fig 5.4: Illustration of insert(t, root) operation

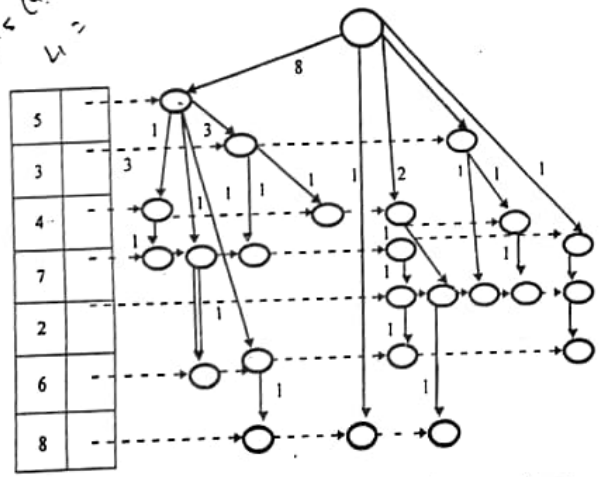

Fig 5.5: Complete FP-tree of the above example. Labels on edges represent frequency.

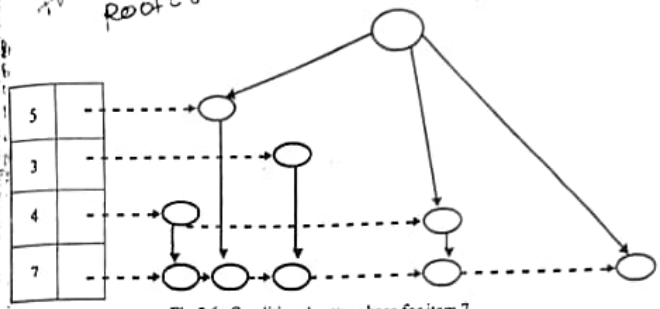

Fig 5.6: Conditional pattern base for item 7.

It is now easy to read the frequent itemsets from the FP-tree.

- The algorithm starts from the leaf node in bottom-up approach.
- Thus, after processing item 6, it processes item 7.
- Let us consider, for instance, the frequent item {6}.
- There are four paths to 6 from the root; these are {5, 7, 6}, {5, 6}, {4, 7, 2, 6} and {7, 2, 6}.
- All these paths have labels of 1.
- A label of a path is the smallest of the link counts of its links.
- Thus, each of these combinations appears just once.
- The paths {5, 7, 6}, {5, 6}, {4, 7, 2, 6} and {7, 2, 6} from the root to the nodes with label 6 are called the prefix sub paths of 6.
- The prefix sub path of a node a are the paths from the root to the nodes labeled a and the count of links along a path are adjusted by adjusting the frequency count of every link in such path, so that they are the same as the count of the link incident on a along the path. This is called a transformed prefix path.
- The transformed prefix paths of a node a form a truncated database of pattern which co-occur with a. This is called a conditional pattern base.
- Once the conditional pattern base is derived from the FP-tree, one can compute all the frequent patterns associated with it in the conditional pattern base.
- By creating a small conditional FP-tree for a, the process is recursively carried out starting from the leaf nodes.

The conditional pattern base for {6} is the following:

- For the prefix subpath {5, 7, 6}, we get {5, 6}→1, {7, 6}→1, {5, 7, 6}→ 1.
- For the prefix subpath {5, 6}, we get {5, 6}→1.
- For the path {4, 7, 2, 6}, we have {4, 6}, {7, 6}, {2, 6}, {4, 7, 6}, {4, 2, 6}, {7, 2, 6}, {4, 7, 2, 6} and for the path {7, 2, 6}, {7, 6}, {2, 6}, {7, 2, 6} all with label 1.
- Thus the only frequent pattern with prefix 6 is {7, 6}→ 3.

In Fig5.6, the conditional pattern base of 7 is illustrated. Please note that since the processing now is bottom-up the combination {6, 7} is already considered when the item 6 was considered.

5.3 Mining Multilevel Association Rules From Transaction Databases

In this section, we will discuss methods of mining multilevel association rules involves items at different levels of abstraction. Methods for checking for redundant multilevel rules are also discussed.

5.3.1 Multi level association rules

For many applications, it is difficult to find strong association among data iems at low primitive levels at abstraction due to the sparsity of data in multidimensional space. Strong associations discovered at high concept levels may represent common sense knowledge.

high support = too few rules

low support = too many rules, most uninteresting

Therefore, data mining system should provide capabilities to mine association rules at multiple levels of abstraction and traverse easily among different abstraction spaces. Let us examine the following example.

Example 5.11: Suppose we are given the task relevant set of transactions data in Table5.5 for sales at the computer department, showing the items purchased for each transaction TID.

- The concept hierarchy defines the sequence of mappings from a set of low-level concepts to higher level, more general concepts. The concept hierarchy of Fig 5.7 has four levels, referred to as levels 0, 1, 2, and 3.
- Data can be generalized by replacing low level concepts with in the data by their higher level concepts or ancestors, from a concept hierarchy.
- Concept hierarchy may be specified by users familiar with the data may exist implicitly in the data. Level 3 represents the most specific abstraction level of this hierarchy.
- The items in Table 5.5 are at the lowest level of the concept hierarchy of Fig 5.7. It is difficult to find interesting purchase patterns at such raw or primitive level data.

TID	ITEMS
T1	IBM desktop computer, sony b/w printer
T2	MS educational software,MS financial management software
T3	Longitude mouse, Ergoway wrist pad
T4	IBM desktop computer, MS financial management software
T5	IBM desktop computer

Table 5. 5: Task relevant data

For instance, if "IBM desktop computer" or "sony b/w printer" each occurs in a very small fraction of the transactions, then it may be difficult to find strong associations involving such items. Few people may buy such items together, making it unlikely that the itemset {IBM desktop computer,sony b/w printer} will satisfy minimum support. However, consider the generalization of

"sony b/w printer" to "b/w printer". One would expect that it is easier to find strong associations between "IBM desktop computer" and "b/w printer" than between "IBM desktop computer" and "sony b/w printer". Similarly, many people may purchase "computer" and "printer" together, rather than specifically purchasing "IBM desktop computer" and "sony b/w printer" together. In other words, itemsets containing generalized items, such as {IBM desktop computer, printer} and {computer, printer} are more likely to have minimum support than itemsets containing only primitive level data, such as {IBM desktop computer" or "sony b/w printer}.

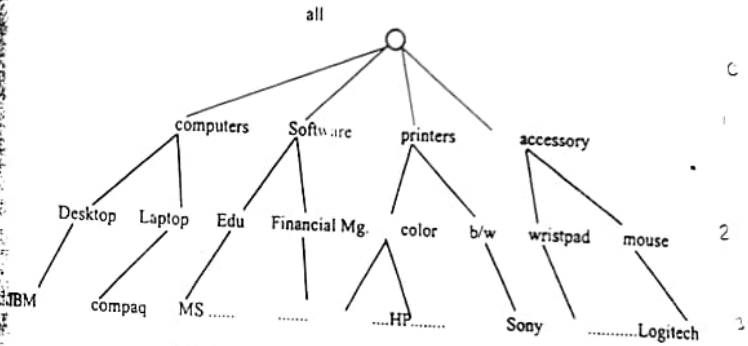

Fig 5.7: A concept hierarchy for computer items

- Hence it is easier to find interesting associations among items at multiple concepts level, rather than only among low level data.

Rules generated from association rule mining with concept hierarchy are called multiple level or multilevel association rules, since they consider more than one concept level.

5.3.2 Approaches To Mining Multilevel Association Rules

How can we mine multilevel association rules efficiently using concept hierarchies? Let's look at some approaches based on a support-confidence frame work.

Support and Confidence of Multilevel Association Rules

- Generalizing/specializing values of attributes affects support and confidence.
- from specialized to general: support of rules increases (new rules may become valid).
- from general to specialized: support of rules decreases (rules may become not valid, their support falls under the threshold).
- Confidence is not affected.

multilevel

1. Using uniform minimum support for all levels

- 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 and uses need to specify one minimum support threshold.
- An optimization technique can be adopted based on / knowledge that an ancestor is a superset of its descendents. The search avoids examines item sets containing any item whose ancestors do not have minimum support.

For example, in Fig 5.8, a minimum support threshold of 5% is used throughout(e.g., for mining from "computer" down to "laptop"). Both "computer" and "laptop" are founc to be frecuent, while "desktop computer" is not.

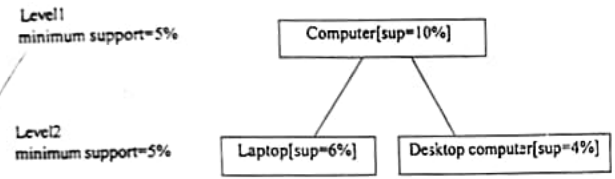

Fig5.8: Multilevel mining with uniform support threshold

Difficulties

- It is unlikely that items at lower levels of abstraction will occur as frequently as those at high levels of abstraction.
- If minimum support threshold is too high , miss several meaningful association.

2. Using reduced minimum support at lower levels

- Each level of abstraction has its own minimum support threshold.
- The lower the abstraction level smaller the correspond threshold.
- For mining multiple level association with reduced support, there are a number of alternative search strategies

For example, in Fig5.9, a minimum support threshold for levels 1 and 2 are 5% and 3%, respectively. In this way, "computer", "laptop" and "desktop computer" are all considered frequent.

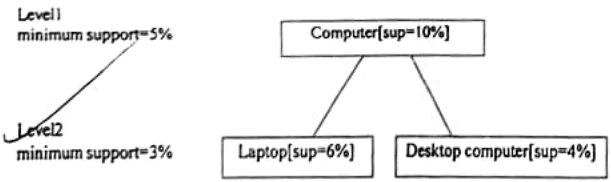

Fig 5.9: Multilevel mining with reduced support

There are 4 search strategies in reduced support. They are

- Level-by-level independent
- Level-cross filtering by k-itemset
- Level-cross filtering by single item
- Controlled level-cross filtering by single item

Level by level independent

- This is full breadth search , where no back ground knowledge of frequent item set is used for running.
- Each mode is examined, regardless of whether or not its pattern node is found to be frequent.

Level cross filtering by single item

- An item at the i^{th} level is examined if and only if its parent node at the $(i-1)^{th}$ level is frequent.
- Otherwise, its descendents are preserved from the search. otherwise, its descendents are preserved from the search.

For example, in Fig5.10, the descendent nodes of "computer"(ie.,"laptop" and "desktop computer") are not examined, since "computer" is not frequent.

Level cross filtering by k-itemset

- A k-item set at the i^{th} level is examined if & only if its corresponding parent k-item set at the $(i-1)^{th}$ level is frequent.

For example, in Fig 5.11, the 2-itemset {computer, printer} is frequent, therefore the nodes {laptop and B/W printer}, {laptop and color printer}, {desktop computer, B/W printer} are examined.

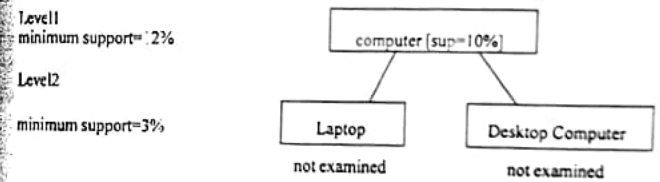

Fig5.10:Multilevel mining with reduced support using Level cross filtering by single item

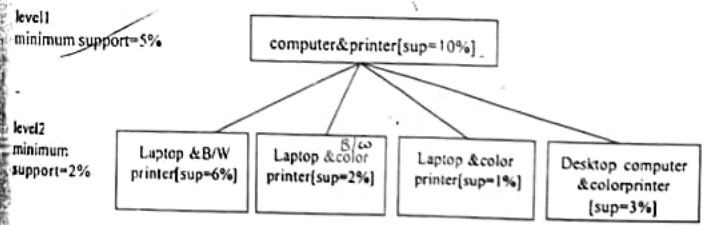

Fig 5.11: Mining with reduced support,using level-cross filtering by k itemset.Here K=2

Controlled level-cross filtering by single item

The level-cross filtering by single item strategy represents a compromise between the two extremes. However, this method may miss associations between low level items that are frequent based on a reduced minimum support, but whose ancestors do not satisfy minimum support.

A modified version of the level-cross filtering by single item strategy, known as the Controlled level-cross filtering by single item strategy, addresses the above concern as follows:

- A threshold, called the level passage threshold, can be set up for "passing down" relatively frequent items to lower levels.
- This method allows the children of items that do not satisfy the minimum support threshold to be examined if these items satisfy the level passage threshold.
- Each concept level can have its own level passage threshold.
- The level passage threshold for a given level is typically set to a value between the minimum support threshold of the next lower level and the minimum support threshold of the given level.
- Users may choose to lower the level passage threshold at high concept levels to allow the descendents of the subfrequent items at lower levels to be examined.

For example, in Fig 5.12, setting the level passage threshold of level 1 to 8% allows the nodes "laptop" and "desktop computer" at level 2 to be examined and found frequent, even though their parent node, "computer", is not frequent.

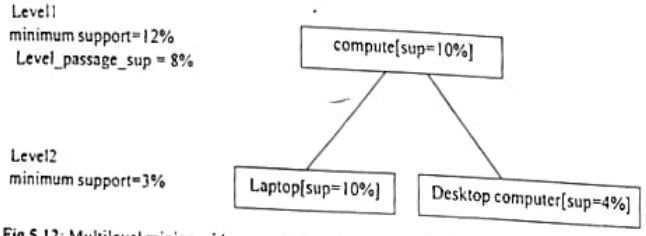

Fig 5.12: Multilevel mining with controlled level cross filtering by single item

5.3.3 Redundant Multiple Association Rules

Concept hierarchies are useful in data mining since they permit the discovery of knowledge at different levels of abstraction, such as multilevel association rules. However, when multilevel association rules are mined, some of the rules found will be redundant due to "ancestor" relationships between items. For example, consider the following rules where "desktop computer" is an ancestor of "IBM desktop computer" based on the concept hierarchy of Fig 5.7.

desktop computer → B/W printer [sup = 8%, conf = 70%]

IBM desktop computer → B/W printer [sup = 2%, conf = 72%]

In the above, the second rule does not provide new information. So, it is removed.

5.4 Mining Multidimensional Association Rules From Relational Databases and Data Warehouses

This section describes the methods for mining multidimensional association rules, that is, rules involving more than one dimension /predicate. These methods can be organized according to their treatment of quantitative attributes.

5.4.1 Multidimensional Association Rules

- **Single dimensional Association Rules:** These are Intra-dimension association rules. Association rules that imply a single predicate are called Single-dimensional Association Rules

  buys (X,"IBM desktop computer") → buys(X,"sony B/W printer")

  buys(X, "milk") → buys(X, "bread")

  where X is a variable representing customer who purchased item.

- **Multi dimensional Association Rules:** These are Inter-dimension association rules. Association rules that imply two or more dimensions / predicates are called Multi-dimensional Association Rules.

  - Inter-dimension association rules (*no repeated predicates*)

    age(X,"19-25") ∧ occupation(X,"student") → buys(X,"coke")

    age(X,"20..29")^occupation(X,"student") → buys(X,"laptop")

  - Hybrid-dimension association rules (*repeated predicates*)

    age(X,"19-25") ∧ buys(X, "popcorn") → buys(X, "coke")

    age(X,"20..29")^ buys(X,"laptop") → buys(X,"printer")

Database attributes can be categorical or quantitative.

- **Categorical Attributes:** Attributes have a finite number of possible values and no ordering among values are called Categorical Attributes (e.g. color of car).
- **Quantitative Attributes:** Attributes are numeric and have an implicit ordering among values are called Quantitative Attributes (e.g. age, income).

Techniques for mining Multidimensional Association Rules can be categorized according to three basic approaches regarding the treatment of quantitative attributes.

- In first approach, quantitative attributes are discretized using predefined concept hierarchies. This discretizations occurs prior to mining. For instance, a concept hierarachy for income may be used to replace the original numeric values of this attribute by ranges, such as "0...20K", "21K...30K", and so on. Here, discretization is static and predetermined. The discretized numeric attributes with their range values, can then be treated as categorized attributes. we refer to this as mining multidimensional association rules using static discretizations of quantitative attributes.

- In the second approach, quantitative attributes are discretized into"bins"based on the distribution of the data. These "bins" may be further combined during the mining process. the discretization process is dynamic and established so as to satisfy some mining criteria, such as maximizing the confidence of the rule mined. It is also referred as quantitative association rules, since it treats numeric attribute values as quantities rather than as predefined range or categories.
- In the third approach, quantitative attributes are discretized so as to capture the semantic meaning of such interval data. This dynamic discretization procedure considers the distance between data points. Hence, such quantitative association rules are also referred as distance-based association rules.

5.4.2 Mining Multidimensional Association Rules Using Static Discretization of Quantitative Attributes

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges.
- In relational database, finding all frequent k-predicate sets will require k or $k+1$ table scans.
- Data cube is well suited for mining.
- The cells of an n-dimensional cuboid correspond to the predicate sets.
- Mining from data cubes can be much faster.

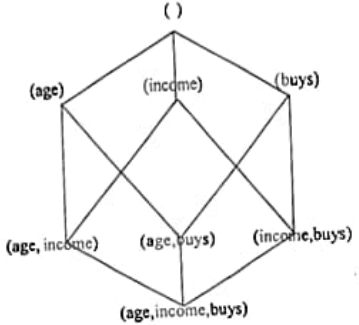

Fig 5.13: Data cube

A data cube consists of a lattice of cuboids that are multi dimensional data structures. These structures can hold the given task relevant data, as well as aggregate, group by information. Fig 5.13 shows the lattice of cuboids defining a data cube for the dimensions age, income, and buys.

5.4.3 Mining Quantitative Association Rules

- Numeric attributes are dynamically discretized during the mining process so as to satisfy some mining criteria, such as maximizing the confidence/compactness of the rules mined.

- 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 rule.

$$A_{quan1} \wedge A_{quan2} \rightarrow A_{cat}$$

where $A_{quan1} \wedge A_{quan2}$ tests on quantitative attributes ranges (where the ranges are dynamically determined), and A_{cat} tests a categorical attribute from the task relevant data. Such rules are referred to as two-dimensional quantitative association rules, since they contain two quantitative dimensions. For example, age(X,"30..39") ∧ income(X,"42k...48K")→ buys(X,"high resolution TV").

- An approach called (ARCS) Association Rule Clustering System which borrows idea from image processing, which maps pairs of quantitative attribute onto a 2-D grid for tuples satisfying a given categorical attribute condition. The grid is then searched for clusters of points, from which the association rules are generated.

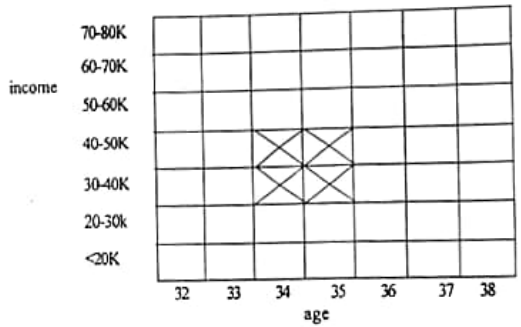

Fig 5.14:A 2-D grid for tuples representing customers who purchase high-resolution TVs.

ARCS (Association Rule Clustering System) :

How does ARCS work? The following steps are involved in ARCS (Fig 5.15):

- **Binning:** Quantitative attributes can have a very wide range(2-D grid) of values defining their domain. To keep grids down to a manageable size, partition the ranges of quantitative attributes into intervals. These intervals are dynamic in that they may later be further combined during the mining process. The partitioning process is referred to as binning. Three common binning strategies are
  - Equiwidth binning: wher the interval size of each bin is the same.
  - Equidepth binning: where each bin has approximately the same number of tuples assigned to it.
  - Homogeneity based binning: where bin size is determined so that the tuples in each bin are uniformly distributed.
- **Find frequent predicate sets:** Once the 2-D array containing the count distribution for each category is set up, this can be scanned in order to find the frequent predicate sets(those

satisfying minimum support) that also satisfy minimum confidence. Strong association rules can then be generated from these predicate sets, using a rule generation algorithm like that described in section5.2.2.

- **Clustering:** The strong association rules obtained in the previous step are then mapped to a 2-D grid Fig 5.14 shows a 2-D grid for 2-D quantitative association rules predicting the condition buys(X,"high resolution TV") on the rule right hand side, given the quantitative attributes age and income. The four Xs correspond to the rules

  age(X,34) ∧ income(X,"31K...40K")→ buys(X,"high resolution TV")

  age(X,35) ∧ income(X,"31K...40K")→ buys(X,"high resolution TV")

  age(X,34) ∧ income(X,"41K...50K")→ buys(X,"high resolution TV")

  age(X,35) ∧ income(X,"41K....50K")→ buys(X,"high resolution TV").

"Can we find a simpler rule to replace the above four rules?" Notice that these rules are quite "close" to one another, forming a rule cluster on the grid. Indeed, the four rules can be combined or "clustered" together to form the following simpler rule, which subsumes and replaces the above four rules:

age(X,"34...35") ∧ income(X,"31K...50K")→ buys(X,"high resolution TV")

- **Optimizer**

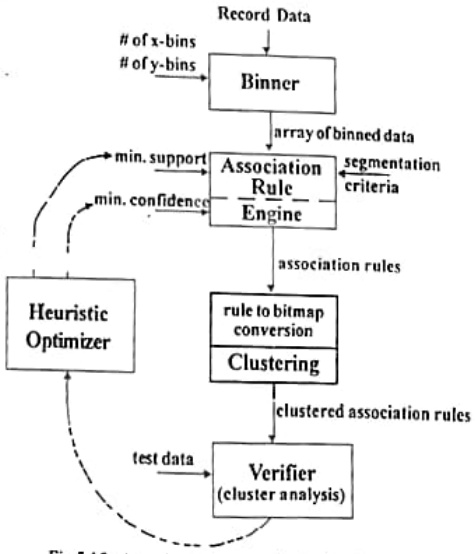

Fig 5.15: Association Rule Clustering System

Limitations of ARCS

- Only quantitative attributes on LHS of rules.
- Only 2 attributes on LHS (2D limitation).

5.4.4 Mining Distance Based Association Rules

The previous section described quantitative association rules where quantitative attributes are descretized by binning methods and the resulting intervals are then combined. Such an approach, however, may not capture the semantics of interval data since they do not consider the relative distance between data points or between intervals.

- In Table 5.6, price partitioned according to equiwidth and equidepth binning versus a distance based partitioning.
- The distance-based partitioning seems the most infinitive since it groups values that are close together within the same interval.
- Equidepth partitioning groups distant values together.
- Equiwidth may split values that are close together and create intervals for which there are no data.
- Clearly a distance-based partitioning that considers the density or no of points in an interval, as well as the "closeness" of points in an interval, helps produce a more meaningful discretization.
- Intervals for each quantitative attributes can be established by clustering the values for the attribute.

A disadvantage of association rule is that they do not allow for approximations of attribute values. Consider the following association rule:

item_type (X,"electronic") ∧ manufacturers of (X,"foreign") → price(X,$200)

Note that the support and confidence do not consider the closeness of values for a given attribute. This motivates the mining of distance based association rule, which capture the semantics of data while allowing for approximation of in data values.

Price($)	Equiwidth(w$10)	Equidepth(dp2)	Distance -based
7	[0,10]	[7,20]	[7,7]
20	[11,20]	[22,50]	[20,22]
22	[21,30]	[51,53]	[50,53]
50	[31,40]		
51	[41,50]		
53	[51,60]		

Table 5.6: binning methods do not capture the semantics of interval data

A two phase algorithm can be used to mine distance based association rules.

- The first phase employs cluster up to find the intervals or clusters, adopting to the amount of available memory.
- The second phase obtains distance based association rules by searching for groups of clusters that occur frequent together.

5.5 From Association Mining to Correlation Analysis

Most association rule mining algorithms employ a support, confidence framework. In spite of using minimum support and confidence thresholds to help exclude the exploration of uninteresting rules, many rules that are not interesting to the user may still be produced. In this section, we look at how even strong association rules can be uninteresting and misleading, and then discuss additional measures based on statistical independence and correlation analysis.

5.5.1 Interestingness Measurements

Whether a rule is interesting or not can be judged either subjectively or objectively. Ultimately, only the user can judge if a given rule is interesting or not, and this judgment, being subjective, may differ from one user to another. However, objective interestingness measures, based on the statistics "behind" the data, can be used as one step towards the goal of weeding out uninteresting rules from presentation to the user.

- Objective measures: Two popular measurements are support; and confidence.
- Subjective measures: A rule (pattern) is interesting if
  - it is unexpected (surprising to the user); and/or
  - actionable (the user can do something with it)

5.5.2 Criticism to Support and Confidence

Association rules mined using a support-confidence framework are useful for many applications. However, the support-confidence framework can be misleading in that it may identify a rule $A \rightarrow B$ as interesting, even the occurrence of A does not imply the occurrence of B. The following example illustrates this criticism.

Example 5.12: In Table5.7, among 5000 students
- 3000 play basketball
- 3750 eat cereal
- 2000 both play basket ball and eat cereal

play basketball $\rightarrow$ eat cereal [40%, 66.7%] is misleading because the overall percentage of students eating cereal is 75% which is higher than 66.7%.

play basketball $\rightarrow$ not eat cereal [20%, 33.3%] is far more accurate, although with lower support and confidence.

	basketball	not basketball	sum(row)
cereal	2000	1750	3750
not cereal	1000	250	1250
sum(column)	3000	2000	5000

Table 5.7

5.5.3 Correlation analysis

In this section, we consider an alternative framework for finding interesting relationships between data itemsets based on correlation.

- 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 correlation between the occurrence of A and B can be measured by computing

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

- If the resulting value of $corr_{A,B}$ is less than 1, then the occurrence of A is negatively correlated with the occurrence of B.
- If the resulting value is greater than 1, then A and B are positively correlated, meaning 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.
- If $corr_{A,B}$ is equal to P(B|A)/P(B), then it is also called the lift of rule $A \rightarrow B$.

A	1	1	1	1	0	0	0	0
B	1	1	0	0	0	0	0	0
C	0	1	1	1	1	1	1	1

Rule	Support	Confidence
$A \rightarrow B$	25%	50%
$A \rightarrow C$	37,5%	75%

Table5.8

Example 5.13: In Table5.8
- A and B: positively correlated,
- A and C, negatively related
- support and confidence of $A \rightarrow C$ dominates

An advantage of correlation is that it is upward closed. This means that if a set S of items is correlated, then every superset of S is also correlated.

5.6 Constraint Based Association Mining

For a given set of task relevant data, the data mining process may uncover thousands of rules many of which are uninteresting to the user. In constraint based mining, mining is performed under the guidance of various kinds of constraints provided by the user. These constraints include the following:

- **Knowledge type constraints:** These specify the type of knowledge to be mined such as classification, association, etc.
- **Data constraints:** These specify set of task relevant data. For example, SQL-like queries such as "Find product pairs sold together in Vancouver in Dec.'98"
- **Dimension/ level constraints:** These specify the dimension of the data as level of concept hierarchies to be used. In relevance to region, price, brand, customer category etc.,.
- **Interestingness constraints:** These specify thresholds on statistical measures of rule interestingness, such as support and confidence. Strong rules (minsup ≥ 3%, minconf ≥ 60%).
- **Rule constraints:** These specify the form of rules to be mined. Such constraints maybe expressed mete rules(rule templates), as the maximum or minimum number of predicates that can occur in the rule antecedent and consequent or as relationships among attributes, attributes values ,and/or aggregates. Small sales (price < $10) triggers big sales (sum > $200).

The above constraint can be specified using a high level declaration data mining query languages. The constraint based mining allows users to specify the rule to be mined according to their intension, thereby making the mining process more efficient. In addition, a sophisticated mining query optimizes can be used to exploit the constraints specified by the user, thereby making the mining process more efficient. Constraint based mining encourages interactive explanatory mining and analysis. Two kinds of rule constraints are there.

5.6.1 MetaRule Guided Mining (Rule Form Constraints)

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 automatically generated based on the database schema.

Example 5.14: One may be interested in finding associations(for computers database) between customer traits and the items that customers buy. However, rather than finding all of the association rules reflecting these relationships, one may particularly be interested only in determining which pairs of customer traits promote the sale of educational software. A metarule can be used to specify this information describing the form of interesting rules. An example of such a metarule is

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

where P_1 and P_2 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 P_1 and P_2, respectively.

In general, a meta rule forms a hypothesis regarding the relationships that the user is interested in probing or confirming. The data mining system search for rules that match the given metarule. For instance, the above metarule matches the rule

$$\text{age}(X, \text{"30,..39"}) \wedge \text{income}(X, \text{"41K...60K"}) \rightarrow \text{buys}(X, \text{"educational software"})$$

5.6.2 Rule constraints Guided Mining (constraint-based query optimization)

Rule constraints specifying set/subset relationships, constant initiation of variables, and aggregate functions can be specified by the user. These 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 us study an example where rule constraints are used to mine hybrid-dimension association rules.

Example 5.15: Suppose that Computer database has a sales multidimensional database with the following interrelated relations:

- sales(customer _name, item_name, transaction_id)
- lives (customer _name, region, city)
- item(item_name, category, price)
- transaction(transaction_id, day, month, year)

where lives, item, and transaction are three dimension tables, linked to the fact table sales via three keys, customer_name, item_name, and transaction_id, respectively.

Our association mining query is to "Find the sales of what cheap items (where the sum of the prices is less than $100) that may promote the sales of what expensive items (where the minimum price is $500) in the same category for Vancouver customers in 1999". This can be expressed as

$$\text{sum(price)} < 100 \wedge \text{min(price)} > 500 \wedge \text{city} = \text{"Vancouver"} \wedge \text{year} = 1999.$$

Rule constraints can be classified into the following five categories with respect to frequent itemset mining:

- Antimonotone
- Monotone
- SIlccinct,
- Convertible, and
- Inconvertible.

Summary

Many business enterprises accumulate large quantities of data continuously from their day-to-day operations. The discovery of interesting associations and relationships among huge amount of business transaction records can help in decision making processes. There are numerous applications, such as Basket data analysis, cross-marketing, inventory management, catalog design, loss-leader analysis, clustering, and classification, etc., come into this frame work.

Association rule mining searches for interesting relationships among items in a given data set. Association rules can be classified in various ways, based on the types of values handled in the rule, the dimension of data involved in the rule, the levels of abstractions involved in the rule set, and various extension to association mining.

The Apriori algorithm is an efficient association rule mining algorithm that explores the level-wise mining Apriori property. As we have seen, in many cases the Apriori candidate generation

method reduces the size of candidate sets significantly and leads to good performance gain. However, it may suffer from two nontrivial costs. Many variations of the Apriori algorithm have been proposed that focus on improving the efficiency of the original algorithm. Several of these variations are Hash-based technique, Transaction reduction, Partitioning, Sampling, and Dynamic itemset counting.

A new class of algorithms have recently been populated which avoids the generation of large number of candidate sets. We describe one such method, called the FP-Tree Growth algorithm.

Multilevel association rules can be mined using several strategies, based on how minimum support thresholds are defined at each level of abstraction. Techniques for mining multidimensional association rules can be categorized according to their treatment of quantitative attributes. Not all strong association rules are interesting. Correlation rules can be mined for items that are statistically correlated. Constraint-based rule mining allow users to focus the search for rules by providing metarules (i.e., pattern templates) and additional mining constraints.

Exercises

1. Define a frequent set. Define an association rule.
2. Define the importance of discovering association rules.
3. Discuss the concepts of frequent sets, support and confidence.
4. Describe the principle of pruning in level-wise algorithms .The set of all frequent sets can be generated even by ignoring the pruning step.Comment.
5. Describe the candidate-generating method of level-wise algorithms. What is the importance?
6. Define a fp-tree .Discuss the method of computing a fp-tree.
7. A database has four transactions. Let min_sup=60% and min_conf=80%.

TID	Date	Items_bought
T100	10/15/99	{K,A,D,B}
T200	10/15/99	{D,A,C,E,B}
T300	10/19/99	{C,A,B,E}
T400	10/22/99	{B,A,D}

(a) Find all frequent itemsets using Apriori and FP-growth, respectively. Compare the efficiency of the two mining processes.

(b) List all of the strong association rules (with supports and confidence c) matching the following metarule, where X is a variable representing customers, and item; denotes variables representing items (e.g., "A", "B", etc.):

$$\forall x \in \text{transaction}, \text{buys}(X,\text{item}_1) \wedge \text{buys}(X,\text{item}_2) \Rightarrow \text{buys}(X,\text{item}_3) \quad [s,c]$$

8. A database has four transactions. Let minsup=60% and minconf=80%.

CID	TID	Items_bought(in the form of brand-item_category)
01	T100	{King's-Crab, Sunset-Milk, Dairyland-Cheese, Best_Bread
02	T200	{Best-Cheese, Dairyland_Milk, Goldenfarm-Apple, Tasty-Pie, Wondeer-Bread}
01	T300	{WestCoast-Apple, Dairyland-Milk, Wonder-Bread, Tasty-Pie}
03	T400	{Wonder-Bread, Sunset-Milk, Dairyland-Cheese}

(a) At the granularity of item_category (e.g., item_i could be "Milk"), for the following rule template,

$$\forall X \in \text{transaction}, \text{buys}(X,\text{item}_1) \wedge \text{buys}(X,\text{item}_2) \Rightarrow \text{buys}(X,\text{item}_3) \quad [s,c]$$

list the frequent k-itemset for the largest k, and all of the strong association rules (with their support s and confidence c) containing the frequent k-itemset for the largest k. At the granularity of brand-item_category (e.g., item_i could be "Sunset-Milk"), for the following rule template,

$$\forall X \in \text{customer}, \text{buys}(X,\text{item}_1) \wedge \text{buys}(X,\text{item}_2) \Rightarrow \text{buys}(X,\text{item}_3)$$

list the frequent k-itemset for the largest k.

9. Propose a method for mining hybrid-dimension association rules (multidimentional association rules with repeating predicates).
10. Give a short example to show that items in a strong association rule may actually be negatively correlated.
11. The price of each item in a store is nonnegative. The store manager is only interested in rules of the form: "one free item may trigger $200 total purchases in the same transaction". State how to mine such rules efficiently.
12. What is the problem of association rule with item-constraints? Propose a method for this.

Multiple Choice Questions

1. Association Rule Mining finds
   a. Interesting Association & Correlation among data item.
   b. Classifies data items.
   c. Groups data items.
   d. None of the above
2. Example of Association Rule Mining is
   a. Market Based Analysis.
   b. Classification Tree.
   c. Income Analysis.
   d. None of the above.

3. Measures of Association Rules are
   a. t-Weight and d-Weight.
   b. Entropy.
   c. Support.
   d. None of the above.

4. Single Dimensional Rule refers to the
   a. Levels of Abstraction involved in the rule.
   b. Based on Attribute Type.
   c. Based on Dimension Involved in the rule.
   d. None of the above.

5. Max Pattern refers to
   a. Frequency Closed Item Set.
   b. Frequent Item Set.
   c. Maximal Frequent pattern.
   d. None of the above.

6. Aprori Property belongs to a special category of proportion called
   a. Monotone.
   b. Anti-Monotone.
   c. Both a & b.
   d. None of the above.

7. Aprori contains two steps as
   a. Search & Join.
   b. Search & Prune.
   c. Join & Prune.
   d. None of the above.

8. Strong Association Rule Supports
   a. Only Minimum Support.
   b. Maximum Support.
   c. Both Minimum Support & Confidence.
   d. None of the above.

9. To reduce the difficulties in Aprori one of the following variation can be used
   a. ID3.
   b. Hash-Based Technique.
   c. Decline.
   d. None of the above.

10. Adding candidate Itemsets at different points during a scan is called
    a. Sampling.
    b. Dynamic Item set Count.
    c. Partitioning.
    d. None of the above.

11. Frequent pattern growth adopts the following strategy
    a. Level-Wise Search.
    b. Divide & Conquer.
    c. Both a & b
    d. None of the above.

12. FP Tree reduces
    a. Memory Consumption.
    b. CPU Consumption.
    c. Search Cost.
    d. None of the above.

13. Multilevel association rules are mined with the help of
    a. Meta patters.
    b. Concept hierarchy.
    c. Meta Rule.
    d. None of the above.

14. One among the mining multilevel association rule is
    a. Using minimum support.
    b. Using level support.
    c. Using reduced minimum support at lower levels.
    d. None of the above.

15. Distance based partitioning considers
    a. Density and closeness of points.
    b. Distance among points.
    c. Interval among data.
    d. All of the above.

16. Random walk algorithm refers to
    a. Stepping down the lattice.
    b. Walk through itemset space.
    c. Both a & b.
    d. None of the above.

17. Constraint based association mining depends on the following constraints
    a. Data constraints
    b. Interestingness constraint
    c. Both a and b
    d. None of the above

18. A constraint where we can enumerate all and only those sets that are guaranteed to satisfy the constraint is called as
    a. Inconvertible constraints
    b. Anti-monotone
    c. succinct
    d. None of the above.

Answers

	1. a	2. a	3. c	4. d	5. c	6. b	7. c	8. c	9. b	10. b
11. b	12. c	13. b	14. c	15. d	16. b	17. c	18. c			

References

- R. Agarwal, C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent itemsets. In Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining), 2000.
- R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93, 207-216, Washington, D.C.
- R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94 487-499, Santiago, Chile.
- R. Agrawal and R. Srikant. Mining sequential patterns. ICDE'95, 3-14, Taipei, Taiwan.
- R. J. Bayardo. Efficiently mining long patterns from databases. SIGMOD'98, 85-93, Seattle, Washington.
- S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. SIGMOD'97, 265-276, Tucson, Arizona.
- S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket analysis. SIGMOD'97, 255-264, Tucson, Arizona, May 1997.
- D.W. Cheung, J. Han, V. Ng, and C.Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. ICDE'96, 106-114, New Orleans, LA.
- G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. ICDE'00, 512-521, San Diego, CA, Feb. 2000.
- Y. Fu and J. Han. Meta-rule-guided mining of association rules in relational databases. KDOOD'95, 39-46, Singapore, Dec. 1995.
- T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. SIGMOD'96, 13-23, Montreal, Canada.
- E.-H. Han, G. Karypis, and V. Kumar. Scalable parallel data mining for association rules. SIGMOD'97, 277-288, Tucson, Arizona.
- J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. ICDE'99, Sydney, Australia.
- J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. VLDB'95, 420-431, Zurich, Switzerland.

- J. Han and M. Kamber. Data Mining.Concepts and techniques, Academic Press, 2001.
- J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00, 1-12, Dallas, TX, May 2000.
- T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of ACM, 39:58-64, 1996.
- M. Kamber, J. Han, and J. Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. KDD'97, 207-210, Newport Beach, California.
- M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A.I. Verkamo. Finding interesting rules from large sets of discovered association rules. CIKM'94, 401-408, Gaithersburg, Maryland.
- B. Lent, A. Swami, and J. Widom. Clustering association rules. ICDE'97, 220-231, Birmingham, England.
- H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. KDD'94, 181-192, Seattle, WA, July 1994.
- H. Mannila, H Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1:259-289, 1997.
- R. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. SIGMOD'98, 13-24, Seattle, Washington.
- N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99, 398-416, Jerusalem, Israel, Jan. 1999.
- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA, May 1995.
- J. Pei, J. Han, and R. Mao. CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. DMKD'00, Dallas, TX, 11-20, May 2000.
- T. Pang-Ning, M. Steinbach, and Vipin Kumar, Introduction to Data Mining, Pearson Education, 2007.
- J.S. Park, M.S. Chen, and P.S. Yu. An effective hash-based algorithm for mining association rules. SIGMOD'95, 175-186, San Jose, CA.
- A.K. Pujari, Data Mining Techniques, Universities Press, 2001.
- S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. VLDB'98, 368-379, New York, NY..
- S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. SIGMOD'98, 343-354, Seattle, WA.