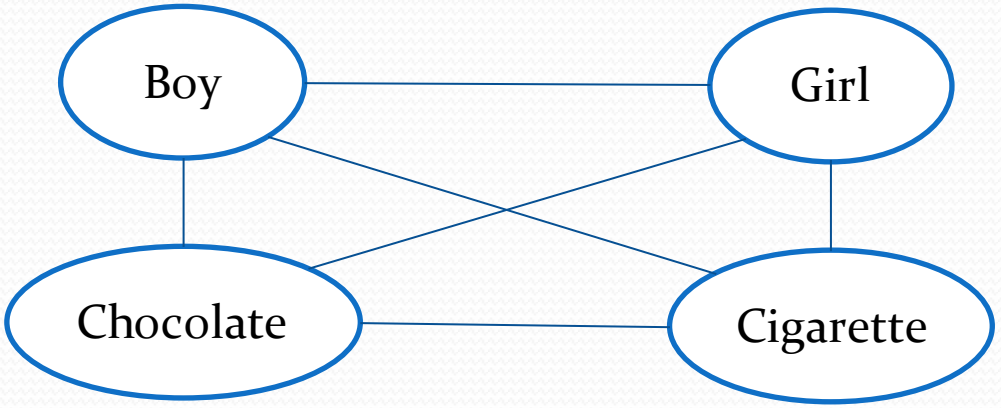
Market Analysis

- Introduction to Association Rule Mining
- Association rule vs. Classification rule
- Some basic definitions
- Generation of Association rules
- Evaluation of Association rule set

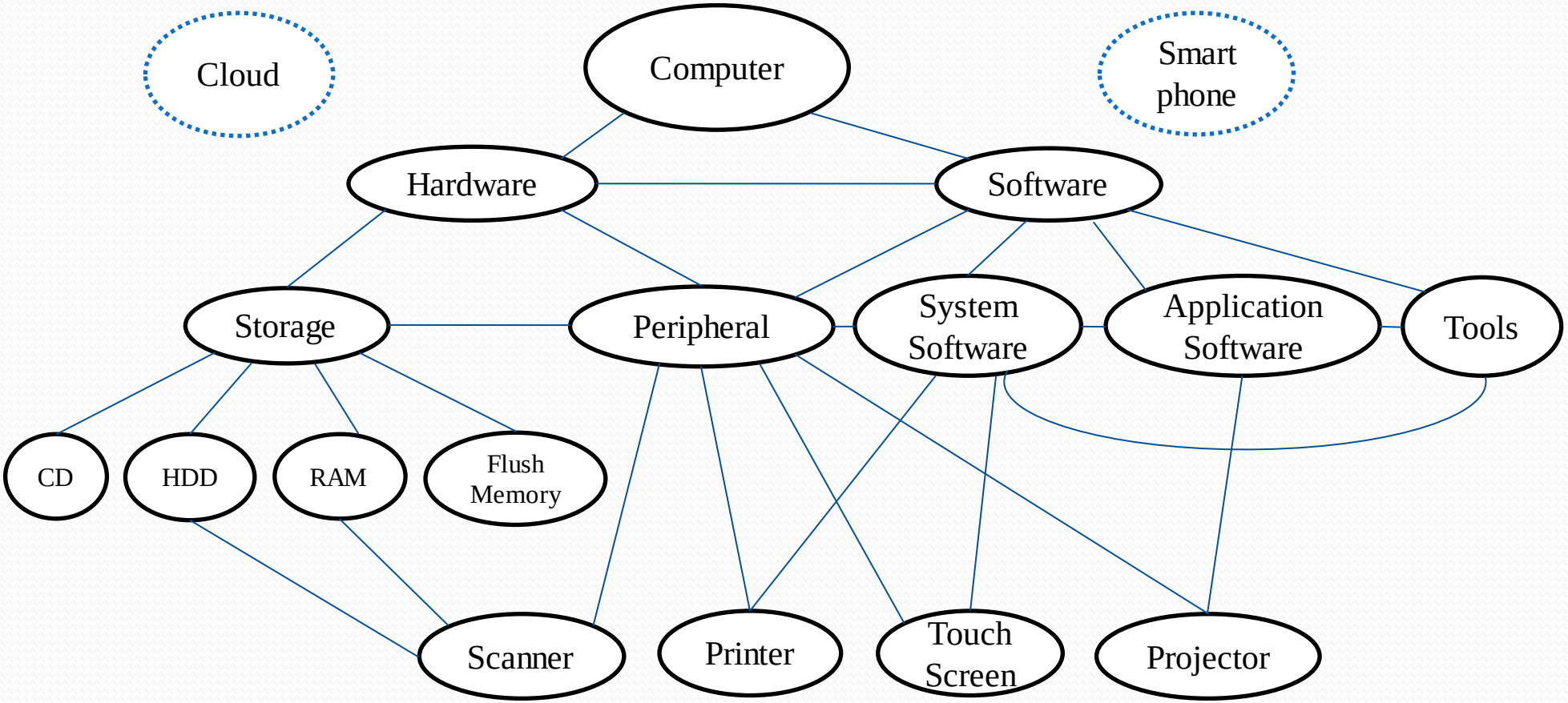
Introduction

The word "association" is very common in our every sphere of life. The literal meaning of association is a spatial or temporal relation between things, attributes, occurrences etc. In fact, the word "relation" is synonymous to the word "association". To understand the concept of association, let us look at Fig. 19.1 (a). Here, we see four things namely "boy", "girl", "chocolate" and "cigarette". Several associations from one (or more) element(s) to other(s) can be interpreted. For example, $girl \xrightarrow{likes} chocolate$ (interpreting girl prefers chocolate) or $\{boy, girl\} \xrightarrow{shops} \{cigarette, chocolate\}$ (interpreting in college canteen, there is a large sales of cigarette and chocolate, if there are both boys and girls students) etc. This is an example showing only few simple things are associated through simple relations. Now, let us turn our focus to a relatively large collection of things as shown in Fig. 19.1 (b).



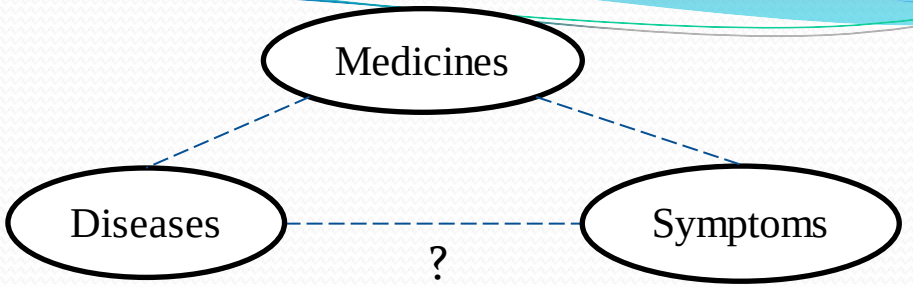
(a) Association among simple things

Introduction



(b) Association among moderately large collection of things

Introduction



(c) Complex association among diseases, symptoms and medicines

Considering only computer and only few elements thereafter, we see how elaborate associationship exists among them. The things even compounded if we consider "smartphone" and/or cloud (of cloud services) into them. If this is not large enough, then consider the scenario of our health world, where three main things namely disease, symptom and medicine are involved. Indeed, there are a large and very complex associations among different constituents in these three things! May be that a medicine specialist can find them. Now, we can formally define the term association which is as follows.

Definition 19.1:*Association:* An association indicates a logical dependency between various things.

These are the three examples pointed out here, and list of such example is, in fact, enormous. One thing is evident from these examples that finding association is not so trivial even if entities are simple or complex in terms of their types or numbers. It, therefore, raises an issue that given a set of things, how to discover association among them. We may note that if S_A and S_C are any two set of elements and $S_A \overset{?}{-} S_C$ denotes association between S_A and S_C, then finding all ? is a hard problem because $|S_A| \geq 1$, $|S_C| \geq 1$ and number of $?_s$ increases exponentially as $|S_A| + |S_C|$ increases. This is one of the fundamental problem in data mining and called Association Rule Mining (ARM). We can define the association rule mining problem as follows.

Introduction (Contd..)

Definition 19.2: ARM

Association rule mining is to derive all logical dependencies among different attributes given a set of entities.

The association rule mining problem was first fundamental by Agarwal, Imielinski and Swami [R. Agarwal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases", Proc. Of Inlt. Conf. on Management of Data, 1993] and is often referred to as the Market-Basket Analysis (MBA) problem. Let us discuss the MBA problem in the following.

Market-Basket Analysis: The concept of MBA stem out from the analysis of supermarket. Here, given a set of transactions of items the task is to find relationships between the presences of various items. In other words, the problem is to analyze customer's buying habits by finding associations between the different items that customers place in their shopping baskets. Hence, it is called Market-Basket. The discovery of such association rules can help the super market owner to develop marketing strategies by gaining insight into matters like "which items are most frequently purchased by customers". It also helps in inventory management, sale promotion strategies, supply-chain management, etc.

Introduction (Contd..)

Example 19.1: Market-Basket Analysis

Suppose, a set of customer purchase data are collected from a grocery stores. Table 19.1 shows a small sample. From this data, the store owner is interested to learn about the purchasing behavior of their customers. That is, which items are occurring more frequent. For example following two are more frequent.

- $Bread - milk$
- $Diaper - beer$

Table 19.1 A sample data

Basket	Items
1	bread, milk, diaper, cola
2	bread, diaper, beer, eeg
3	milk, diaper, beer, cola
4	bread, milk, tea
5	bread, milk, diaper, beer
6	milk, tea, sugar, diaper

Introduction (Contd..)

We can say that two rules $\{bread\} \rightarrow \{milk\}$ and $\{beer\} \rightarrow \{diaper\}$, or more precisely, two association rules suggesting a strong relationships between the sales of bread and milk, and beer, diaper exist (this means, customer, who purchases bread (or beer) also likely to purchase milk (or diaper).

The process of analyzing customer buying habits by finding association between different items can be extended to many application domains like medical diagnosis, web mining, text mining, bioinformatics, scientific data analysis, insurance schemas, etc. In general, an association rule can be expressed as

$$\{S_L\} \rightarrow \{S_R\}$$

Where S_L and S_R denotes a set of items (non-empty). However, there is no universally accepted notation for expression association rules and we may merely follow the popular convention. Further, note that in our convention an arrow is used to specify whether the relation is bi-directional. Sometimes, the relationship may be unidirectional, that is, $\{S_R\} \rightarrow \{S_L\}$ are both equivalent.

Association rule vs. Classification rule

Classification rule of the form $X \rightarrow Y$ and association rule of the same form $X \rightarrow Y$ look alike but they have different implications.

1. First, classification rules are concerned with predicting an entity to which it belongs. So, the right-hand part of the rule is always a single attribute called class label. Whereas $X \rightarrow Y$ representing an association rule, Y may be consists of one or more element(s).

2. Both the rules imply logical dependence of Y on X. In case of classification rule, X is a conjunctions of attributes and relations with their values. In other words, the left-hand part in classification rule potentially includes test on the value of any attribute or combination of attribute. On the other hand, $X(or\ Y)$ in association rule includes a set of values rather than tests.

3. Classification rule IF $X \rightarrow$ THEN Y is either fires (that is, satisfies) or not; whereas, in case of association rule $X \rightarrow Y$ it is a matter of degree of strength how X is related to Y.

4. In the case of classification rules, we are generally interested in the quality of a rule set as a whole. It is all the rules working in combination that determine the effectiveness of a classifier, not any individual rule or rules.
   In the case of association rules, the emphasis is on the quality of each individual rule, instead of the whole set of rules.

Some basic definitions and terminologies

Before going to discuss our actual topic, first we should go through some definitions and terminologies, which will be frequently referred to. We define each term followed by relevant example(s) to understand the term better.

Following notations will be frequently referred to in our definitions.

Table 19.2: Some notations

Notation	Description
D	$Database\ of\ transactions$
t_i	$Transaction\ (ith\ )\ in D$
X, Y	$Itemsets$
$X \rightarrow Y$	$Association\ rule$
s	$Support$
α	$Confidence$
I	$Set\ of\ large\ itemsets$
i	$Large\ itemset\ in\ I$
C	$Set\ of\ candidate\ itemsets$

Some basic definitions and terminologies (Contd..)

Definition 19.3: Transactions and itemsets

Suppose, D is a database comprising n transactions that is, D = $\{t_1, t_2, .., t_n\}$ where each transaction denotes a record (for example, in the context of market-basket, it is a set of items shopped). I denotes the set of all possible items. For an example, Table 19.3 shows a database of transaction. In this table, a, b, c, d *and* e denote items. Here, $I = \{a, b, c, d, e\}$ comprising set of all items. Any one transaction, say $\{a, b, c\}$ is called an itemset.

Table 19.3: Database of transactions

Transaction Id	Transaction (item set)
1	$\{a, b, c\}$
2	$\{a, b, c, d, e\}$
3	$\{b\}$
4	$\{c, d, e\}$
5	$\{c, d\}$
6	$\{b, c, d\}$
7	$\{c, d, e\}$
8	$\{c, e\}$

Note: $\{a, b, c\}$ and $\{c, b, a\}$ are the same itemset. For convenience, all items in an itemset are presented in an order (say alphabetical order). An itemset is a non-null set. Thus, the maximum number of transactions, that is possible, with m items in I, is $2^m - 1$ (it is all subsets from m elements).

Some basic definitions and terminologies (Contd..)

Definition 19.4: Association rule

Given a set of items, I = $\{i_1, i_2, .., i_m\}$ and a database of transactions D = $\{t_1, t_2, .., t_n\}$ where t_i= $\{i_{11}, i_{12}, .., i_{ik}\}$ and $i_{ij} \in I$, an association rule is an implication of the form $X \rightarrow Y$, where $X, Y \subset I$ are itemsets, and $X \cap Y = \Phi$ (a null-set).

Note:

- X *and* Y are disjoint and cardinality of $X \cup Y$, that is, $|X \cup Y|$ should be at least 2.
- Usually, a rule is stated as $r_i: X \rightarrow Y$. Here X is called body and Y is head. Also, X *and* Y is called antecedent and consequent.
- Inorder to improve, the interpretability there are many optional parameters are included. Thus, $r_i: X \rightarrow Y$ [support, confidence] [start Time, end Time] [validity Period] etc. We shall discuss later about these parameters.
- The head part is usually a single attribute. If there are more than one attribute in head, then it can be splitted into further association rules. For example

$$r_i: \{a, b\} \rightarrow \{c, d\} \Rightarrow \begin{cases} \{a, b\} \rightarrow \{c\} \; and \\ \{a, b\} \rightarrow \{d\} \; etc. \end{cases}$$

Some basic definitions and terminologies (Contd..)

Definition 19.5:Support Count and Support

A transaction t_i is said to contain an itemset$X$, if X is a subset of t_i. Support count refers to the number of transactions that contain a particular itemset. Mathematically, the support count of an itemset$X$ is denoted as $\sigma(X)$ and is defined as (Equation 19.1).

$$\sigma(X) = |\{t_i | X \in t_i, t_i \in D\}| \quad \ldots\ldots(19.1)$$

Where the symbol $|X|$ denotes the number of elements in a set X.

Example: With reference to Table 19.3, the support count of the set $\{c, d\}$ is $\sigma(\{c, d\}) = 5$

Support is the ratio (or percentage) of the number of itemsets satisfying both body and head to the total number of transactions. Support of a rule $r_i: X \rightarrow Y$ is denoted as $s(r_i)$ and mathematically defined as

$$s(r_i) = \frac{\sigma(X \cup Y)}{|D|} \quad \ldots\ldots(19.2)$$

Example: From Table 19.3:

$$s(\{c, d\} \rightarrow \{e\}) = \frac{3}{8} = 0.375$$
$$s(\{a, b, c\} \rightarrow \{d, e\}) = \frac{1}{8} = 0.125$$

Note:

- Support of $X \rightarrow Y$also can be taken to be the probability $\mathrm{P}(X \cup Y)$. That is
$$s(X \rightarrow Y) = \mathrm{P}(X \cup Y)$$
Further, note that notation $\mathrm{P}(X \cup Y)$ indicates the probability that a transaction contains the union of itemsets$X$ *and* Y. This should not be confused with $\mathrm{P}(X \; or \; Y)$, which indicates the probability that a transaction contains either X *or* Y.
- Support implies a measurement of strength of strength of a rule. $s(r_i) = 0$ implies "no-match", whereas $s(r_i) = 1$ implies "all matches". In other words, a rule that has very low support may occur simply by chance, and hence association rule is insignificant. Whereas a rule with high value of s is significant.

Some basic definitions and terminologies (Contd..)

Definition 19.6:Confidence

Confidence of a rule $r_i: X \rightarrow Y$ in a database D is represented by $\alpha(r_i)$ and defined as the ratio (or percentage) of the transactions in D containing X that also contain Y to the support count of X. More precisely,

$$\alpha(X \rightarrow Y) = \frac{\sigma(\mathrm{X} \cup Y)}{\sigma(X)} \ldots (19.3)$$

Note:

- Confidence of a rule also can be expressed as follows

$$\alpha(X \rightarrow Y) = \frac{\sigma(\mathrm{X} \cup Y)}{\sigma(X)}$$

$$= \frac{\frac{\sigma(\mathrm{X} \cup Y)}{|D|}}{\frac{\sigma(\mathrm{X})}{|D|}}$$

$$= \frac{\mathrm{P}(\mathrm{X} \cup Y)}{\mathrm{P}(\mathrm{X})}$$

$$= \mathrm{P}(\mathrm{X}|Y)$$

  So, alternatively, confidence of a rule $X \rightarrow Y$is the conditional probability that Y occurs given that X occurs.
- Also, $\mathrm{P}(\mathrm{X} \cup Y)$ and $\mathrm{P}(\mathrm{X})$ called the frequencies of occurrences of itemsets$\{X, Y\}$ and $\{X\}$, respectively.
- Confidence measures the reliability of the inference made by a rule. For a given rule $X \rightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transaction that contains X.

Note: Support (s) is also called "relative support" and confidence (α) is called "absolute support" or "reliability".

Some basic definitions and terminologies (Contd..)

Definition 19.7:Completeness

Another measure of a rule strength is called completeness. It is the ratio of the matching right-hand side (i.e., head) that correctly matched by the rule. Thus, for any rule $r_i: X \rightarrow Y$, the completeness c is defined as

$$c(r) = \frac{\sigma(X \rightarrow Y)}{\sigma(Y)}$$

Example: For the rule $r: \{c, d\} \rightarrow \{e\}$ (See Table 19.3), we have

$$\sigma(\{c, d\} \rightarrow \{e\}) = 3 \; and$$
$$\sigma(\{e\}) = 4$$
$$\text{Thus, } c(r) = \frac{3}{4} = 0.75$$

Definition 19.8:minsup and minconf

It is customary to reject any rule for which the support is below a minimum threshold (μ). This minimum threshold of support is called minsup and denoted as μ.

Typically the value of $\mu = 0.01$ $(i.e., 1\%)$. Also, if the confidence of a rule is below a minimum threshold (τ), it is customary to reject the rule. This minimum threshold of confidence is called minconf and denoted as τ.

Typically, the value of $\tau = 0.8$ $(i.e., 80\%)$.

Example: For a database of 2500 transactions, for five association rule chosen at random, decide the rules, which is/are rejectable.

Some basic definitions and terminologies (Contd..)

Example: For a database of 2500 transactions, for five association rule chosen at random, decide the rules, which is/are rejectable.

Table 19.4: Characteristics of few association rules from D^*

Rule Id $X \rightarrow Y$	$\sigma(X \rightarrow Y)$	$\sigma(X)$	$\sigma(Y)$
r_1	700	720	800
r_2	140	150	650
r_3	1000	1000	2000
r_4	200	400	250
r_5	295	300	700

$|D|^* = 2500$ $(Number\ of\ transactions\ in\ D)$

[Left as an exercise]

Some basic definitions and terminologies (Contd..)

Definition 19.9: Frequent Itemset

Let μ be the user specified minsup. An itemset i in D is said to be a frequent itemset in D with respect to μ if

$$s(i) \geq \mu$$

Example: An itemset with k elements in it is called $k - itemset$. Decide whether 2-itemset $\{a, e\}$ in Table 19.3 is a frequent itemset with respect to $\mu = 0.1$

Here, $s(\{a, e\}) = \frac{1}{8} = 0.125$.

Hence, $\{a, e\}$ is a frequent itemset.

Note: A frequent $k - itemset$ is also called large itemset and denoted as L_k.

Some basic definitions and terminologies (Contd..)

Definition 19.10: Itemset Lattice

Suppose, I denotes a set of items in a given data set of transactions. If the cardinality of I is n, then it is possible have $2^n - 1$itemsets. A lattice structure can be used to enumerate the list of all possible itemsets. Figure 19.2 show a an itemset lattice for $I = \{a, b, c, d\}$. From such itemset lattice we can easily list all $k - itemset$ $(k = 1,2,..,n)$. For example, $3 - itemsets$ in $I = \{a, b, c, d\}$ are (a, b, c), (a, b, d), (a, c, d) and (b, c, d).

This itemset lattice is useful to generate all frequent itemsets given a database of transactions with items I.

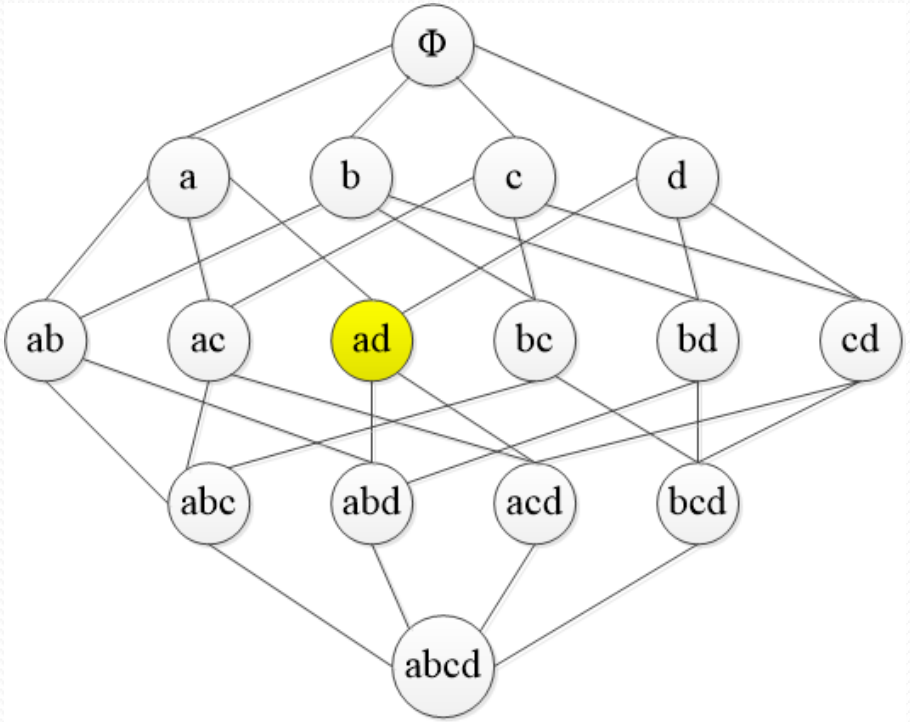


Figure 19.2: Itemset lattice with I={a,b,c,d}

Some basic definitions and terminologies (Contd..)

Definition 19.11: Closed frequent itemsets and maximal frequent itemsets

An itemset X is closed in a data set D if there exist no proper super-itemset Y such that Y has the same support count as X in D.

Proper super-itemset: Y is a proper super-itemset of X if X is a proper sub-itemset of Y, that is, if $X \subset Y$. In other words, every item of X is contained in Y but there is at least one item of Y, that is not in X.

An itemset X is a closed frequent itemset in set D, if X is both closed and frequent in D.

Example: Consider the itemset lattice in Fig. 19.2. Assume that $\{a, d\}$ is a frequent itemset, that is, $\sigma\{a, d\} \geq \mu$ for some minsupμ. The proper super-itemset of $\{a, d\}$ is $\{a, c, d\}$, $\{a, b, d\}$ and $\{a, b, c, d\}$. Also, assume that none of them has support count more than or equal to $\sigma(\{a, d\})$. Hence, $\{a, d\}$ is the closed frequent itemset. Note that here $\{a, c, d\}, \{a, b, d\}$ and $\{a, b, c, d\}$ are not necessarily be frequent.

An itemset X is a maximal frequent itemset in set D if X is frequent, and there exist no super-itemset Y such that $X \subset Y$ and Y is frequent in D.

Example: This is continuing with $\{a, d\}$ in itemset lattice shown in Fig. 19.2, suppose $\sigma\{a, d\} \geq \mu$ for some μ. If all $\{a, b, d\}$, $\{a, c, d\}$ and $\{a, b, c, d\}$ have their support counts less than μ, then $\{a, d\}$ is the maximal frequent itemset.

Note that if X is a maximal frequent itemset then it is also closed frequent, but the reverse is not true.

Some basic definitions and terminologies (Contd..)

Definition 19.12: Monotonicity property

Let I be a set of items and $J = 2^I$ be the power set of I.

Upward Closed: A measure f is upward closed (that is monotone) if

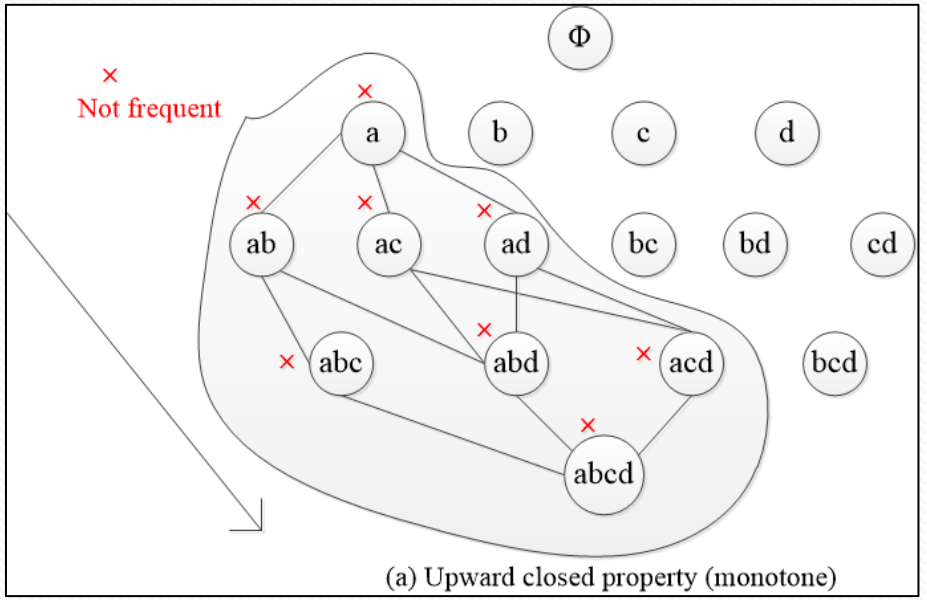
$$\forall X, Y \in J: (X \subseteq Y) \longrightarrow f(X) \leq f(Y)$$

This monotone property implies that if X is a subset of Y, then $f(X)$ must not exceed $f(Y)$. In other words, if X is not a frequent set, then so also Y. See Fig. 19.3(a).

Downward Closed: A measure f is downward closed (that is anti-monotone) if

$$\forall X, Y \in J: (X \subseteq Y) \longrightarrow f(Y) \leq f(X)$$

This anti-monotone property implies that if X is a subset of Y, then $f(Y)$ must not exceed $f(X)$. In other words, if Y is a frequent itemset, then so also X. See Fig. 19.3 (b).



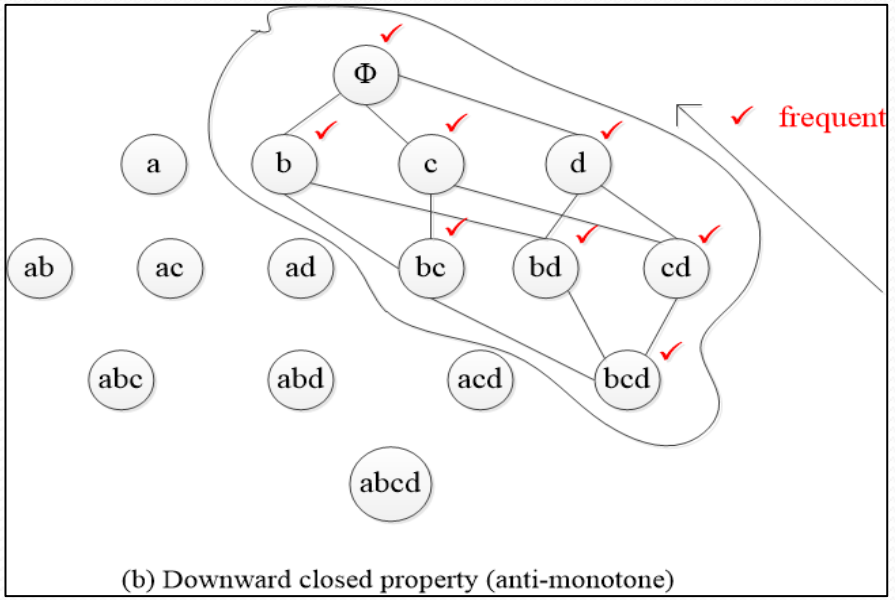


Fig. 19.3: Monotonicity properties

Some basic definitions and terminologies (Contd..)

Definition 19.13: Activity Indicators

To improve the readability and comprehensiveness many (intermediate) forms are followed. This is starting with the input data set to maintaining the association rules. Few such practices are stated here.
Consider a data set recorded in a medical centre regarding the symptoms of patients.

Table 19.5: Records of Patients

Patient	Symptom(s)
1	$Fever, Throat\ pain, Cold$
2	$Fever, Headache$
3	$Cold, Dizziness, Fever$
4	$Pneumonia, Cold, Dizziness$
5	$Diarrhea, Pneumonia$

Binary representation of data

It is customary to represent a data set in a binary form shown in Table 19.6

Table: 19.6 Binary representation of data in Table 16.5

Patient	Cold	Diarrhea	Dizziness	Fever	Headache	Pneumonia	Throat Pain
1	1	0	0	1	0	0	1
2	0	0	0	1	1	0	0
3	1	0	1	1	0	0	0
4	1	0	1	0	0	1	0
5	0	1		0	0	1	0

Binary representation of data

In the binary representation, each row corresponds to a transaction (i.e., record) 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.

Note that items are presented in an order (such as alphabetical order by their names) and this simplistic representation provides many operations to be performed faster.

Co-occurrence matrix

Another matrix representation, showing the occurrence of each item with respect to all others is called co-occurrence matrix. For example, the co-occurrence matrix for the data set in Table 19.5 is shown in Table 19.7.

Table 19.7: Co-occurrence Matrix of data in Table 19.5

	Cold	Diarrhea	Dizziness	Fever	Headache	Pneumonia	Throat Pain
Cold	3	0	2	2	0	1	1
Diarrhea	0	1	0	0	0	1	0
Dizziness	2	0	2	1	0	1	0
Fever	2	0	1	3	1	0	0
Headache	0	0	0	1	1	0	0
Pneumonia	1	1	1	0	0	2	0
Throat Pain	1	0	0	1	0	0	1

Note that the co-occurrence matrix is symmetric.

Association Rule Mining

Now, we are in a position to discuss the core problem of this topic: how to discover association rules, given a dataset of transactions. The discovery of association rules is the most well-studied problem in data mining. In fact, there are many types of frequent itemsets, association rules and correlation relationships. As a consequence, the association rule mining problem can be classified into the following different criteria:

1. Mining based on completeness of itemsets.
2. Mining based on level of abstractions in the rule set.
3. Mining based on dimensionality in rules.
4. Mining based on category of data in rules.
5. Mining based on type of itemsets in rules.
6. Mining based on type of rules.

We briefly mention the above criteria as follows:

1. Completeness criterion: According to this criterion, we are to mine a set of itemsets (called $complete-$, $closed-$, or maximal-frequent itemsets) satisfying some threshold value (s) such as minsup, minconf, etc. Such a rule mining is called complete association rule mining.
2. Level of abstraction criterion: Sometimes, we need to mine association rules at different level of abstractions. For example, in the following rules

$$\{Computer, Windows\} \rightarrow \{HP\ Printer\} \dots (A)$$

$$\{Laptop, Linux\} \rightarrow \{HP\ Printer\} \dots (B)$$

Here, assuming "computer" is a higher-level of abstraction of "laptop", the rule (A) is mined differently than the rule (B). Such a rule mining is called multilevel association rule mining.

Association Rule Mining (Contd..)

3. Dimensionality criterion: If an association rule is limited to only one dimensional attribute, then the rule is called single-dimensional association rule. On the other hand, if a rule references two-or more dimensions, then it is a multi-dimensional association rule. Suppose, Buy, Age, Income denotes three attributes. Then rule (C) in the following is a single-dimensional association rule whereas rule (D) is the multidimensional association rule.

$$Buy\{Computer, Windows\} \rightarrow Buy\{HP\ Printer\}..(C)$$

$$Age\{25 \ldots 40\} \wedge Income\{30K \ldots 50K\} \rightarrow Buy\{Laptop, Linux\}..(D)$$

4. Data category criterion: Often a rule describes associations regarding presence of items. Such an association rule is called Boolean association rule. For example, rule (A), (B) and (C) all are Boolean association rules. In contrast, the rule (D) describes an association between numerical attributes and hence is called quantitative association rule.
5. Type of pattern criterion: Usually, we mine set of items and associations between them. This is called frequent itemsets mining. In some other situations, we are to find a pattern such as subsequences from a sequence of data. Such a mining is called sequence pattern mining.
6. Type of rule criterion: In general, the rule we are to discover among itemsets are the association rule. Other than the association rule, some other rules such as “correlation rule”, “fuzzy rule”, “relational rule”, etc. can be applied.

   Different criteria are to meet different needs of applications. But basic problem of all types remains same. Without any loss of generality, we assume single-level, single-dimensional, Boolean, association rule mining in our discussions.

Association Rule Mining (Contd..)

3. Dimensionality criterion: If an association rule is limited to only one dimensional attribute, then the rule is called single-dimensional association rule. On the other hand, if a rule references two-or more dimensions, then it is a multi-dimensional association rule. Suppose, Buy, Age, Income denotes three attributes. Then rule (C) in the following is a single-dimensional association rule whereas rule (D) is the multidimensional association rule.
   Buy{Computer, Windows}→Buy{HP Printer}..(C)
   Age{25…40} ∧ Income{30K…50K} →Buy{Laptop, Linux}..(D)
4. Data category criterion: Often a rule describes associations regarding presence of items. Such an association rule is called Boolean association rule. For example, rule (A), (B) and (C) all are Boolean association rules. In contrast, the rule (D) describes an association between numerical attributes and hence is called quantitative association rule.
5. Type of pattern criterion: Usually, we mine set of items and associations between them. This is called frequent itemsets mining. In some other situations, we are to find a pattern such as subsequences from a sequence of data. Such a mining is called sequence pattern mining.
6. Type of rule criterion: In general, the rule we are to discover among itemsets are the association rule. Other than the association rule, some other rules such as “correlation rule”, “fuzzy rule”, “relational rule”, etc. can be applied.

   Different criteria are to meet different needs of applications. But basic problem of all types remains same. Without any loss of generality, we assume single-level, single-dimensional, Boolean, association rule mining in our discussions.

Problem specification and solution strategy

For the type of problem, that is, given a set of transactions D, we are to discover all the rule r_i such that $s(r_i) \geq minconf$, where $s(r)$ and $\alpha(r)$ denote support and confidence of rule r. Solution to such a problem can be obtained at two different steps:

1. Generating frequent itemsets, that is, given a set of items I, we are to find all the itemsets that satisfy the **minsup** threshold. These itemsets are called frequent itemsets.
2. Generating association rules, that is, having a frequent itemsets, we are to extract rules those satisfy the constraint minimal confidence, that is, **minconf**.

Out of the above mentioned two tasks, the first task is computationally very expensive and the second task is fairly straight forward to implement. Let us first examine the naïve approach to accomplish the task of frequent itemset generation.

Naïve approach to frequent itemsets generation

We may note that the cardinality of an itemset (corresponding to a rule $L \rightarrow R$) is at least two. Thus, first we are to i) generate all possible itemsets of cardinality two or more and then ii) check which of them are supported. The first task, is therefore, finding all possible subsets of I (i.e., the power set 2^I), where I is the set of all items, which has cardinality. There are 2^m subsets; of these, m have a single item and one has no item (the empty set). Thus, the number of itemsets with cardinality at least two is $2^m - m - 1$.

For a practical application, this can be very large. For example, if $m = 20$, then $2^{20} - 20 - 1 = 1{,}048{,}555$. If m takes more realistic but still relatively small value of 100, then the total number of itemsets is $2^{100} - 100 - 1 \approx 10^{30}$!

Thus, generating all the possible itemsets and then checking against the transactions in the database to establish which ones are supported is clearly unrealistic or impossible in practice (with brute-force approach).

Once, the candidate itemset is generated, the next task is to count the occurences of each itemset. To do this using naïve approach, we need to compare each candidate against every transaction (also see Fig. 19.4). Such an approach is also expensive because it requires $O(n \times p \times w)$ where $p = |C|$, the number of candidate itemsets $(= 2^m - m - 1 \quad if \quad |I| = m), n =$ number of transactions in D and w is the maximum transaction width $(\leq m)$.

We may note that we can save computations if we ignore the candidate itemsets, which are not relevant to the input data set and further reduce the second stage computation by eliminating the candidate itemstes, which are not prospective. The first breakthrough algorithm in this context in Apriori algorithm, which is our next topic of discussion.

Naïve approach to frequent itemsets generation (Contd..)

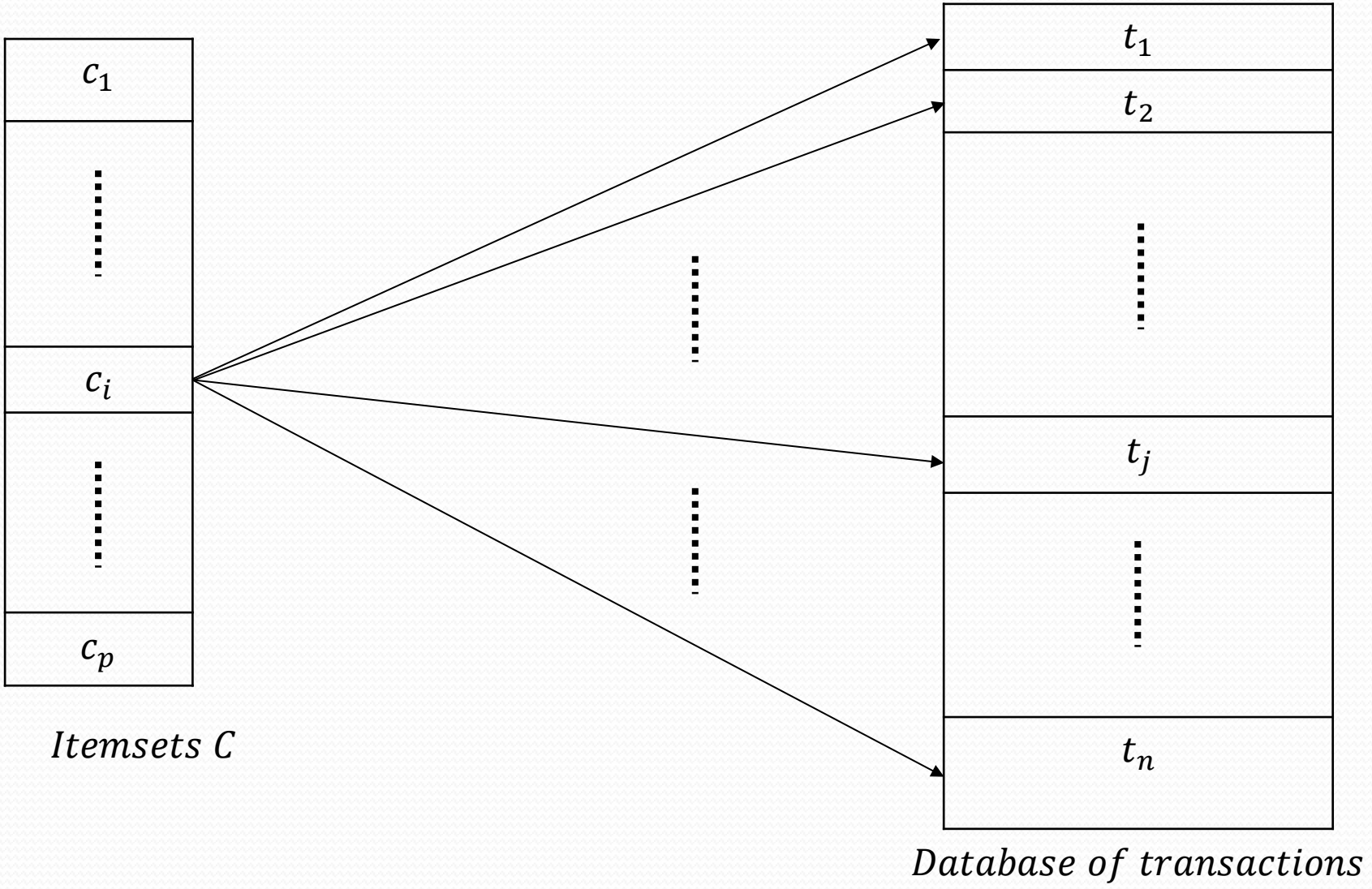


Figure 19.4: Counting support Count

Apriori algorithm to frequent itemsets generation

The Apriori algorithm is known to be a breakthrough algorithm to solve the problem of generating frequent itemsets in a realistic time. The algorithm proposed by R. Agarwal and R. Srikant in 1994. This algorithm is called "apriori" because it uses prior knowledge of frequent itemset property. The algorithm is also called as level-wise search algorithm, because it follows a top-down search, moving downward level-wise in the itemset lattice. To improve the efficiency of the level-wise generation of frequent itemset i.e., to reduce the search space), it relies on an important property of frequent itemsets is called apriori property which is stated below.

Theorem 19.1 Apriori Property

If an itemset is frequent, then al of its subsets (non-empty) must also be frequent.

This property is also called downward closed property of itemsets. To illustrate the apriori property, let us consider the itemset lattice shown in Fig. 19.5. Suppose, $\{c, d, e\}$ is a frequent itemset. Clearly, $\{c, d\}, \{c, e\}, \{d, e\}, \{c\}, \{d\}$ *and* $\{e\}$ are also frequent. This is because any transaction that contains $\{c, d, e\}$ must also contain its all the above mentioned subsets.

Apriori algorithm to frequent itemsets generation (Contd..)

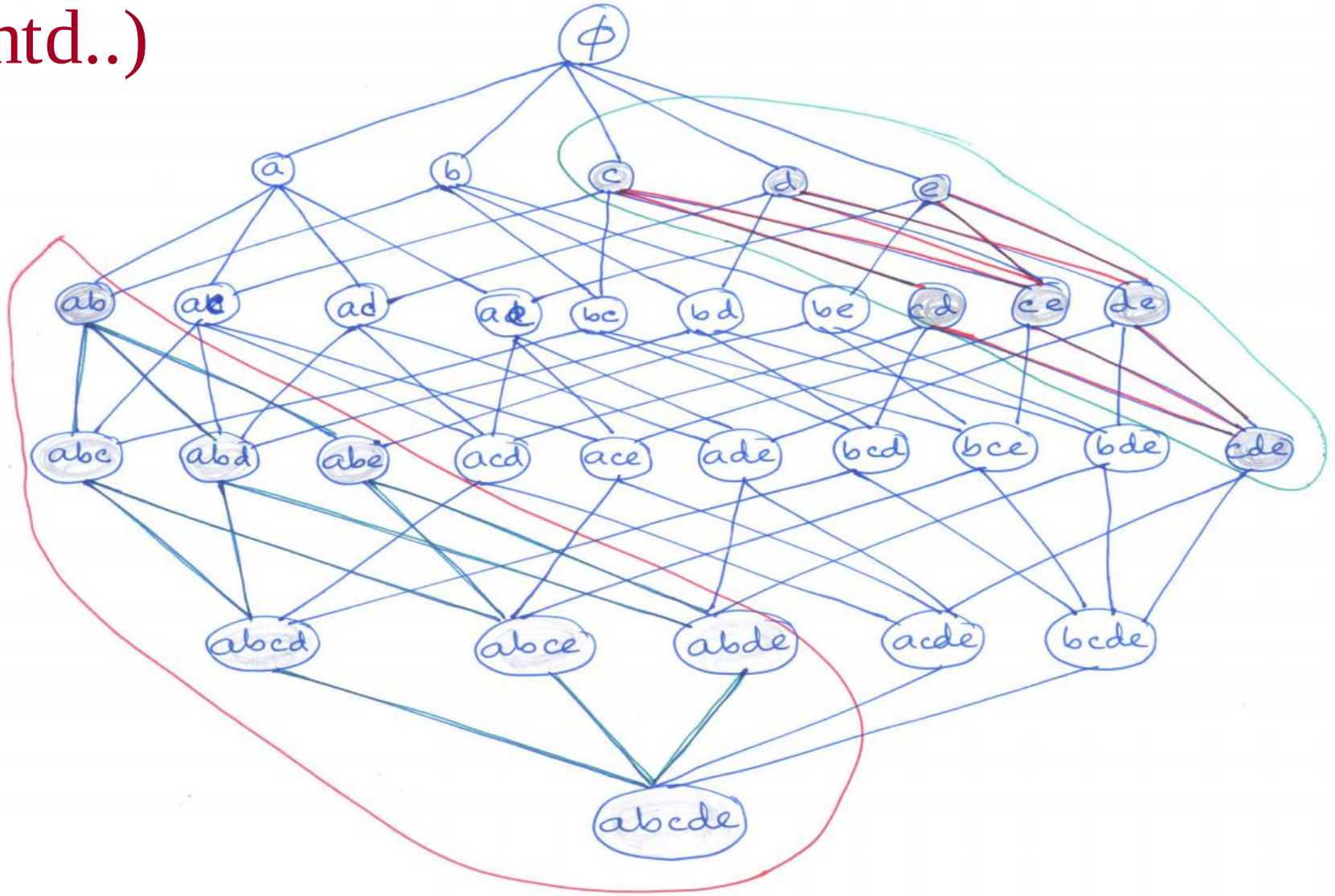


Figure 19.5: Illustration of apriory property

Apriori algorithm to frequent itemsets generation (Contd..)

The apriori property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will also fail the same test. It is called anti-monotone because the property is monotonic in the context of failing test. For example, if $\{a, b\}$ is infrequent, then all of its supersets must be frequent too (see Fig. 19.5). From this, we can write the following.

Corolary 19.1: Let us denote L_k be the set containing all the frequent $k - itemsets$.
If $L_k = \Phi$ (the empty set), then $L_{k+1}, L_{k+2}, \ldots etc.$ must also be empty.

The apriori algorithm takes the advantages of the above two results. It generates the frequent itemsets in ascending order of their cardinality. That is, it starts with all those with one element (L_1) first, then all those with two elements (L_2), then all those with three elements (L_3) and so on. In other words, at a $k - th$ stage, the set L_k of frequent $k - itemsets$ is generated from the previous L_{k-1} set. At any stage, if L_k is Φ, the empty set, we know that $L_{k+1}, L_{k+2}, etc.$ must also be empty. Thus, itemsets of cardinality $k + 1$ or higher do not need to be generated and hence tested as they are certain to turn out not to be frequent.
In addition to this, apriori algorithm follows a method of going from each set L_{k-1} to the next L_k in turn. This is done with two steps:

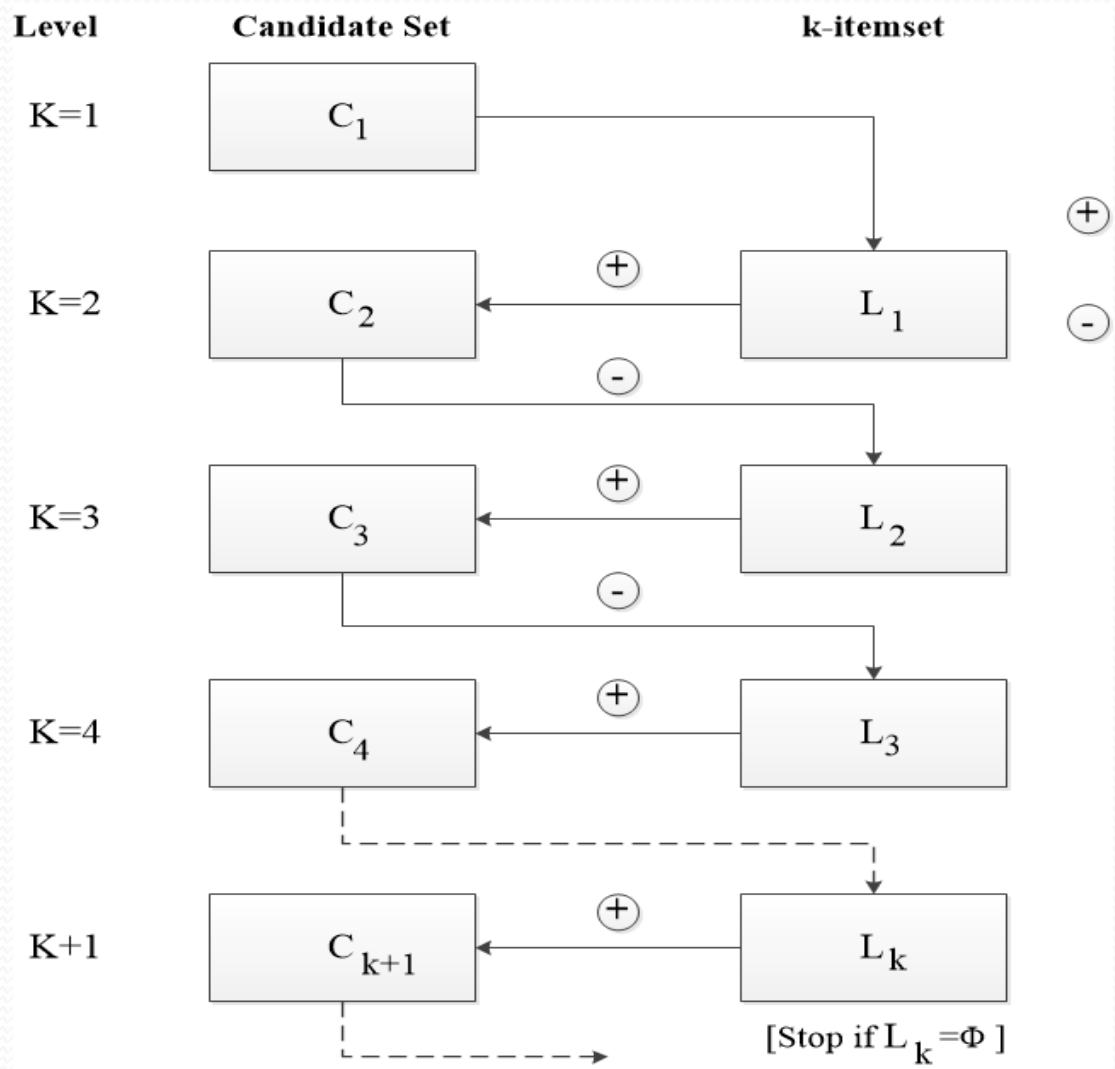
1. Generate a candidate set (containing $k - itemsets$) from L_{k-1};
2. Prune C_k to eliminate those k-itemsets, which are not frequent. Overall working of Apriori algorithm is illustrated in Fig. 19.6.

Apriori algorithm to frequent itemsets generation (Contd..)

Finally, when $L_k = \Phi$, the algorithm terminates generating all frequent itemsets as $L_1 \cup L_2 \cup \ldots \cup L_{k-1}$.
The Step 1 is popularly called Apriori-gen procedure and Step 2 is the Prune procedure.

Apriori algorithm is precisely stated as Algorithm 19.1 in the following. Note that the algorithm aim to generate all frequent $k - itemsets$ $(k = 1,2, \ldots k)$ for some k and suitable for extracting association rules as single-level, single-dimensional Boolean association rules.

Apriori algorithm to frequent itemsets generation (Contd..)



- ⊕ Create (k+1)-itemsets from k-itemsets in L_k $(k = 1,2, \ldots.)$ $C_{k+1} = Apriori - gen(L_k)$
- ⊖ Prune all infrequent itemsets $Prune\ (C_k)$

Fig. 19.6: Working scenarios of Apriori algorithm

Apriori algorithm to frequent itemsets generation (Contd..)

Algorithm 19.1: Aprori Frequent Itemset Generation

Input

$D = \{t_1, t_2, \ldots, t_n\}$ //a data set of n transactions

$I = \{i_1, i_2, \ldots, i_m\}$ //set of items in D

$s = minsup\ threshold$

$\alpha = minconf\ threshold$

Output

$L = All\ frequent\ itemsets.$

Apriori algorithm to frequent itemsets generation (Contd..)

Algorithm 19.1: Aprori Frequent Itemset Generation

Steps

//Initialization

1. $C_1 =$ All the $1-itemsets$ from D
2. K=1 //Level-1 search at top-level
3. Scan the data set D to count the support of each $1-itemset$ $l \in C_1$ and store each l with their support count in L_1. That is $L_1 = frequent\ 1-itemset$

// Top-down level-wise search

4. k=2 //Begin level-2 search
5. while ($L_{k-1} \neq \Phi$) do
6. $C_k = Apriory - gen\ (L_{k-1})$ //Create k-itemsets from L_{k-1}
7. Prune (C_k) //Delete the k-itemsets if they violates apriori property. Scan data set D to update the support counts of all k-itemsets in C_k
8. For all $t_i \in D$ do
9. Increment the support count of all k-itemsets in C_k that are contained in t_i.
10. That is $L_k = All\ frequent\ k-itemsets\ from\ C_k$
11. $k = k + 1$ //Go to next level-search
12. Result $L = \cup_k L_k$

Apriori algorithm to frequent itemsets generation (Contd..)

Algorithm 19.1: Aprori Frequent Itemset Generation

Steps

//Initialization

1. $C_1 =$ All the $1-itemsets$ from D
2. K=1 //Level-1 search at top-level
3. Scan the data set D to count the support of each $1-itemset$ $l \in C_1$ and store each l with their support count in L_1. That is $L_1 = frequent\ 1-itemset$

// Top-down level-wise search

4. k=2 //Begin level-2 search
5. while ($L_{k-1} \neq \Phi$) do
6. $C_k = Apriory - gen\ (L_{k-1})$ //Create k-itemsets from L_{k-1}
7. Prune (C_k) //Delete the k-itemsets if they violates apriori property. Scan data set D to update the support counts of all k-itemsets in C_k
8. For all $t_i \in D$ do
9. Increment the support count of all k-itemsets in C_k that are contained in t_i.
10. That is $L_k = All\ frequent\ k-itemsets\ from\ C_k$
11. $k = k + 1$ //Go to next level-search
12. Result $L = \cup_k L_k$

Apriori algorithm to frequent itemsets generation (Contd..)

Procedure Apriori –gen

Input: L_{k-1}, the set of frequent $(k-1)$ itemsets

Output: C_k, the set of k-itemsets from joins of $(k-1)-itemsets$ in L_{k-1}

Remark: If $|L_{k-1}| = 1$, the algorithm is not invocable.

Step:

1. $C_k = \Phi$ // Initially the candidate set is empty
2. If $L_{k-1} = 1$ Return //No candidate set generation is possible
3. For all $l_i \in L_{k-1}$ do
4. | For $l_j \in L_{k-1}$ do
5. | | If $(l_{i1} = l_{j1}) \wedge (l_{i2} = l_{j2}) \wedge \cdots \wedge (l_{ik-1} \neq l_{jk-1})$ then
6. | | | $c = \{l_{i1}, l_{i2}, \ldots\ldots, l_{ik-1}, l_{jk-1}\}$ //join an item to produce a new $k-itemset$
7. | | | $C_k = C_k \cup c$ //Add c to C_k
8. Return C_k.

Apriori algorithm to frequent itemsets generation (Contd..)

Procedure Prune

Input: C_k, the $k-itemset$

Output: C_k, the frequent $k-itemset$

Step

1. For all $c \in C_k$ do
2. SS=Compute all $(k-1)$ subsets of c
3. If all $(d \in SS) \notin L_{k-1}$
4. $C_k = C_k - c$ //Remove c from C_k
5. Return C_k

Apriori algorithm to frequent itemsets generation (Contd..)

Illustration of Apriori algorithm

To illustrate the Apriori algorithm, let us consider the binary representation of a data set of transactions shown in Table 19.8. For brevity, 9 items are denoted as 1,2,3,...,8,9.Type equation here.

Table 19.8: Dataset of transactions

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
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	1	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	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	1	0
0	1	0	1	0	1	1	0	0
1	0	1	0	1	0	1	0	0
0	1	1	0	0	0	0	0	1

This table contains 15 transactions and support counts for different transactions are shown in Table 19.9.

Apriori algorithm to frequent itemsets generation (Contd..)

Illustration of Apriori algorithm

Table 19.9: Support counts for some itemsets in Table 19.8

Itemset	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}	{9}	{5,6}	{5,7}	{6,7}	{5,6,7}
Support Count	2	6	6	4	8	5	7	4	2	3	5	3	1

We trace the Apriori algorithm with reference to the dataset shown in Table 19.8. Assume that minsup $threshold = 20\%$ (that is, an itemset which is supported by at least three transactions is a frequent itemset).

Initialization

$$C_1 = \text{The candidate itemsts with all } 1-\text{itemsets}$$
$$= \{\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\}\}$$

$K = 1$ $\quad //Level-1\ (top)\ search$

We scan the dataset (Table 19.8) and copy the frequent 1-itemsets into the list L_1 (here, those itemsets in C_1 are having support count three or more are copied into list L_1).

L_1: Frequent 1-itemsets from C_1

Frequent 1-Itemset	{2}	{3}	{4}	{5}	{6}	{7}	{8}
Support Count	6	6	4	8	5	7	4

Apriori algorithm to frequent itemsets generation (Contd..)

$\underline{K = 2 \qquad //Level - 2 \ search}$

We follow Apriori-gen (L_1) algorithm to compute the set of candidates C_2 containing $2 - itemsets$.

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

$Prune(C_2)$: This operation does not delete any element in C_2

Next to create L_2, the list of frequent 2-itemsets from C_2. To do this, for each 2-itemset in C_2, we scan the dataset D and include the if its support count is more than minsup (=20% in our case). The L_2 obtained following the above is shown below.

$L_2: Frequent \ 2 - itemsets \ from \ C_2$

Frequent 2-Itemset	{2,3}	{2,4}	{3,5}	{3,7}	{5,6}	{5,7}	{6,7}
Support Count	3	3	3	3	3	5	3

Apriori algorithm to frequent itemsets generation (Contd..)

$\underline{K = 3 \quad //Level - 3\ search}$

We repeat Level-3 search as $L_2 \neq \Phi$.

The Apriori-gen (L_2) produces C_3, shown below

$C_3 = \{\{3,5,7\},\{5,6,7\}\}$

Next we scan entire data set D to update the support count of the 3-itemsets in C_3 which produces the list L_3 as below.

$L_3: Frequent\ 3 - itemsets\ from\ C_3$

Frequent 3-Itemset	{3,5,7}
Support Count	3

$\underline{K = 4 \quad //Level - 4\ search}$

Since $L_3 \neq \Phi$, we are to repeat the level-wise search.

Since, L_3 contains only one element, so Apriori-gen (L_3) is not invocable. That is $C_4 \neq \Phi$. As C_4 is empty set, so also L_4.

$\underline{K = 5 \quad //Level - 5\ search}$

Here $L_4 \neq \Phi$, so search comes to an end.

Frequent itemsets

The Apriori algorithm produces frequent itemsets as L=$L_1 \cup L_2 \cup L_3 \cup L_4$

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

Analysis of Apriori algorithm

Apriori algorithm is a significant step forward, and is quite comparable to its naïve approach counterpart. The effectiveness of the Apriori algorithm can be compared by the number of candidate itemsets generated throughout the process as the total time is influenced by the total number of candidate item sets generated. Let us do a calculation with reference to the example, that we have just discussed. For that dataset, frequent item sets are up to maximum 3-item sets. Hence, using naïve approach, the total number of candidate item sets would be with 9 items is:

$${}^{9}C_1 + {}^{9}C_2 + {}^{9}C_3 = 9+36+84 = 129$$

On the contrary, with Apriori algorithm, we have to generate candidate item sets at 3-different levels of searching, whose total number is:

$$9+20+3 = 32$$

Note that in real-life application number of items is very large and Apriori algorithm shows even better performance with very large number of items. An exact mathematical analysis on time and storage complexity of Apriori algorithm is not possible. This is because, it depends on many factors in addition to the pattern of occurrences of the items in transactions. The deciding factors are:

1) Number of transactions in input data set (n).
2) The average transaction width (w).
3) The threshold value of minsup (s).
4) The number of items (dimensionality, m).

Analysis of Apriori algorithm

Scope of improving Apriori algorithm:

Since, Agrawal and Srikant's paper was published a great deal of research effort has been devoted to finding more efficient ways of generating frequent item sets. These generally involve reducing the number of passes (iterations) through all the transactions in the dataset, reducing the number of frequent item sets in C_k, more efficient counting of the number of transactions matched by each of the item set in C_k, or some combination of these.

Generating association rules:

•The next step after the generation of frequent item sets is to extract association rules from the frequent item sets. We shall discuss the problem and possible deals to handle the problem.

•**Problem:** For a K-item set, we are to deduce a rule of the form L → R, where L is the set of items in the body of the rule and R the set of items in the head of the rule. That is $|L \cup R| = K$. In other words, from a frequent *K*-item set, we can generate a large number of candidate rules, and then check the value of *confidence* for each one to judge whether a rule to be included or ignored given a threshold value of *minconf*.

•The above problem thus turns out to generate head of the rules in turn; each one must have at least one and at most (K-1) items. Having a fix on head of a rule, all the unused items in the dataset must then be on the body of the rule.

Analysis of Apriori algorithm

The number of ways selecting i items from the K items in a frequent item set is ${}^{K}c_i = \frac{K!}{(K-i)!i!}$. Therefore, the total number of possible right hand sides R and thus the total number of possible rules that can be constructed from a k-item set is:

${}^{K}c_1 + {}^{K}c_2 + \ldots + {}^{K}c_i + {}^{K}c_{k-1}$.

It can be shown that

$$\sum_{i=1}^{k} K_{C_i} = 2^k - 2 \qquad \ldots(19.5)$$

- Assuming, that k is reasonably small, say 10, this number is manageable. For $k = 10$, there are $2^{10} - 2 = 1022$ possible rules (from one k –item set!). However, in real-life applications k may be as small as 20, when number of rules is 1,048,574.
- Now, if this is for one item set of size k, there are set of many other item sets of size 2 to a large value say k.
- If n_x be the number of x- item sets, then the total number association rules possible for a given set of frequent item sets are:

$$\sum_{i=2}^{k} n_x = 2^x - 2 \qquad \ldots(19.6)$$

Analysis of Apriori algorithm

This is indeed an extremely large number (if not infinite) for moderate value of k and $\sum_{i=2}^{k} n_x = |L_k|$ and impractical to solve using brute-force approach.

Solution: Fortunately there is a way out to reduce the number of candidate rules considerably using the monotonicity property of confidence estimate. This property is stated as Theorem 19.2.

Theorem 19.2 Confidence Inequality

Suppose, $i = A \cup B \cup C$ is an item set of size $|i| \geq 2$ and A, B, C are mutually exclusive sets of item sets. If a rule $A \cup B \rightarrow C$ does not satisfy the confidence threshold *minconf*, then the rule $A \rightarrow B \cup C$ must not satisfy the confidence threshold *minconf* as well.

Proof: The theorem can be proved as follows:

The confidence of the rules $A \rightarrow B \cup C$ and $A \cup B \rightarrow C$ can be written as:

$$\alpha(A \rightarrow B \cup C) = \frac{s(i)}{s(A)} \quad \text{---(i)}$$

$$\alpha(A \cup B \rightarrow C) = \frac{s(i)}{s(A \cup B)} \quad \text{---(ii)}$$

It is obvious that $s(A) \geq s(A \cup B)$. This is because the proportion of transactions in the database matched by an item set A must be at least as large as the proportion matched by a larger item set $A \cup B$.

Analysis of Apriori algorithm

Hence, it follows that $\sigma(A \rightarrow B \cup C) \leq \sigma(A \cup B \rightarrow C)$.

Alternatively, this theorem can be interpreted as transferring members of a frequent item set from antecedent of a rule to a consequent can not increase the value of the rule confidence. In other words, if $A \cup B \rightarrow C$ is low confidence rule, then $A \rightarrow B \cup C$ is also low-confidence rule.

This is further illustrated in Fig. 19.7. Figure 19.7 shows an item set lattice structure for the association rule generated from an item set $i = \{a, b, c, d\}$.

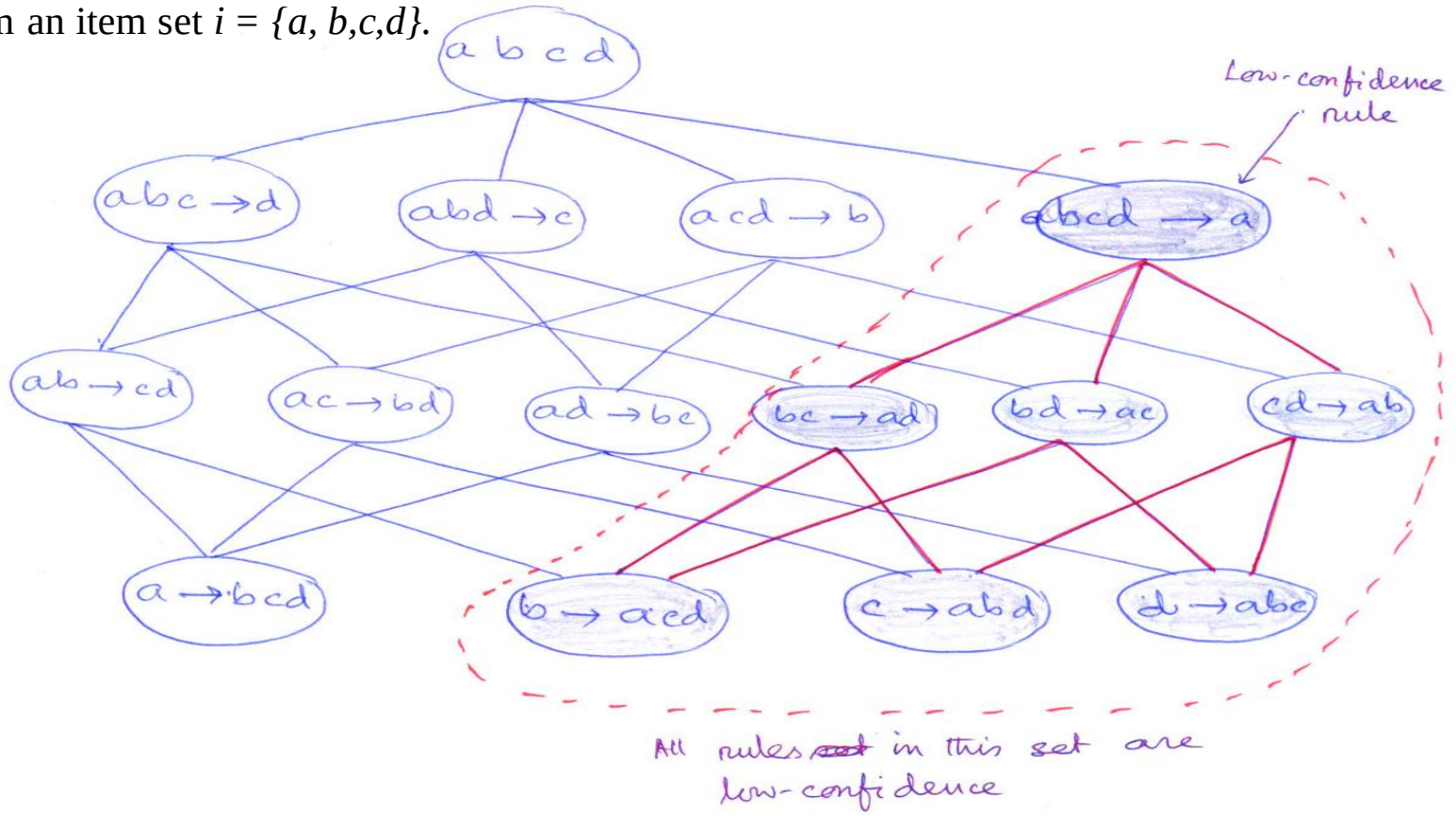


Fig. 19.7: Illustration of Theorem 19.2

Analysis of Apriori algorithm

If any node in the lattice show low-confidence, then according to Theorem 19.2, the entire sub graph spanned by the rules are also of low-confidence values. For example, if $bcd \rightarrow a$ is a low confidence rule, then all rules containing a in the consequent, that is, $bc \rightarrow ad$, $bd \rightarrow ac$, $cd \rightarrow ab$, $b \rightarrow acd$, $c \rightarrow abd$ and $d \rightarrow abc$ are also low-confidence rules.

Theorem 19.2 is utilized to reduce the number of candidate rules from a given frequent k-item set ($k \geq 2$). Initially, all the high confidence rule that have only one item in the rule consequent are extracted.

These rules are then used to generate candidate rules by merging one of them to other. For example, if $A \rightarrow C$ and $B \rightarrow D$ are two rules, then they can be merged to a new candidate rule $A \cap B \rightarrow C \cup D$.
Example: $\{a,b,d\} \rightarrow c$
$\{a,c,d\} \rightarrow b$

•All these new candidate rules are then passed to the next level of rule extraction.

•Based on the idea mentioned above, the Apriori algorithm uses a level-wise approach for generating association rules, given a set of frequent item sets. The *i-th* level is corresponding to i number of items ($i = 1,2,\ldots,k-1$) that belong to the rule consequent and $k \geq 2$.

Analysis of Apriori algorithm

A pseudocode for the rule generation according to Apriori algorithm is precisely stated in Algorithm 19.2.

Algorithm 19.2: Apriori Association Rule Generation

Input: $L=L_2 \cup L_3 \cup \ldots \cup L_k$, set of frequent item sets. Minimum confidence threshold *minconf*

Output: AR, set of association rules.

Step:

//Initialization: Get high-confidence rule with one-item in rule-head.

1. AR = ∅ // The list of rules is initially empty
2. For each $i \in L$ do
3. Get all rules in the form R= $\{l \rightarrow r \mid r \in i, |r| = 1, l = i - r, \sigma (l \rightarrow r) \geq$ misconf $\}$
4. AR = AR ∪ R // Add this rule to the rule of association
   // This completes the extraction of all association rule with only one item in the head.
   //Level-wise generation of association rules.
5. For each i_k, the k-item set rules ($k \geq 2$) in AR do
6. $H_k = \{i \mid i \in i_k\}$
7. R= Apriori-Rules (i_k, H_k)
8. AR = AR ∪ R // Add new extracted rules
9. Return AR
10. Stop.

Analysis of Apriori algorithm

We now state the Apriori-Rules (...) procedure follows:

Procedure Apriori-Rules

Input: H_m, all item sets $i_k = \{l_b \cup r_m\}$

$k = btm$ corresponding to rule $l_b \rightarrow r_m$ i.e., rule with consequent size m

Output: R_{m+1}, return rule with consequent size m+1

Step:

1. H_{m+1} = Apriori_Gen (H_m) // Generate all candidate rule i.e., $\begin{cases} a \rightarrow b \\ a \rightarrow c \end{cases}$ k=2 $a \rightarrow bc$; $\begin{cases} ab \rightarrow c \\ ab \rightarrow d \end{cases}$ k=3 $ab \rightarrow cd$; ... etc.
2. For each $i = \{l_b \cup r_{m+1}, b + m + 1 = |i|\} \in H_{m+1}$ do // Check the confidence of each candidate rule.
3. conf = $\frac{\sigma(i)}{\sigma(r_{m+1})}$
4. If conf $\geq$ *minconf* then
5. R = $\{l_b \rightarrow r_{m+1}\}$ //Add the rule
6. Return R
7. Stop

Analysis of Apriori algorithm

Illustration: Students are advised to extract all association rules for the frequent item set generated for data set in Table 19.8. Assume the threshold of *minconf* = 50%

[Work out to be added here]

Also, compare the performance of Apriori-rules with Brute-Force approach.

Any question?

You may post your question(s) at the “Discussion Forum” maintained in the course Web page!