6

Association Analysis: Basic Concepts and Algorithms

Many business enterprises accumulate large quantities of data from their day-to-day operations. For example, huge amounts of customer purchase data are collected daily at the checkout counters of grocery stores. Table 6.1 illustrates an example of such data, commonly known as **market basket transactions**. Each row in this table corresponds to a transaction, which contains a unique identifier labeled TID and a set of items bought by a given customer. Retailers are interested in analyzing the data to learn about the purchasing behavior of their customers. Such valuable information can be used to support a variety of business-related applications such as marketing promotions, inventory management, and customer relationship management.

This chapter presents a methodology known as **association analysis**, which is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of **associa-**

Table 6.1. An example of market basket transactions.

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}

tion rules or sets of frequent items. For example, the following rule can be extracted from the data set shown in Table 6.1:

$$\{\texttt{Diapers}\} \longrightarrow \{\texttt{Beer}\}.$$

The rule suggests that a strong relationship exists between the sale of diapers and beer because many customers who buy diapers also buy beer. Retailers can use this type of rules to help them identify new opportunities for cross-selling their products to the customers.

Besides market basket data, association analysis is also applicable to other application domains such as bioinformatics, medical diagnosis, Web mining, and scientific data analysis. In the analysis of Earth science data, for example, the association patterns may reveal interesting connections among the ocean, land, and atmospheric processes. Such information may help Earth scientists develop a better understanding of how the different elements of the Earth system interact with each other. Even though the techniques presented here are generally applicable to a wider variety of data sets, for illustrative purposes, our discussion will focus mainly on market basket data.

There are two key issues that need to be addressed when applying association analysis to market basket data. First, discovering patterns from a large transaction data set can be computationally expensive. Second, some of the discovered patterns are potentially spurious because they may happen simply by chance. The remainder of this chapter is organized around these two issues. The first part of the chapter is devoted to explaining the basic concepts of association analysis and the algorithms used to efficiently mine such patterns. The second part of the chapter deals with the issue of evaluating the discovered patterns in order to prevent the generation of spurious results.

6.1 Problem Definition

This section reviews the basic terminology used in association analysis and presents a formal description of the task.

Binary Representation Market basket data can be represented in a binary format as shown in Table 6.2, 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. Because the presence of an item in a transaction is often considered more important than its absence, an item is an **asymmetric** binary variable.

Table 6.2. A binary $0/1$ representation of market basket data.

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

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. Methods for handling such non-binary data will be explained in Chapter 7.

Itemset and Support Count   Let $I = \{i_1, i_2, \ldots, i_d\}$ be the set of all items in a market basket data and $T = \{t_1, t_2, \ldots, t_N\}$ 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-itemset. 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.

The transaction width is defined as the number of items present in a transaction. A transaction t_j is said to contain an itemset X if X is a subset of t_j. For example, the second transaction shown in Table 6.2 contains the itemset {Bread, Diapers} but not {Bread, Milk}. An important property of an itemset is its support count, which refers 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:

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

where the symbol $|\cdot|$ denote the number of elements in a set. In the data set shown in Table 6.2, the support count for {Beer, Diapers, Milk} is equal to two because there are only two transactions that contain all three items.

Association Rule   An association rule is an implication expression of the form $X \longrightarrow Y$, where X and Y are disjoint itemsets, i.e., $X \cap Y = \emptyset$. The strength of an association rule can be measured in terms of its **support** and **confidence**. Support determines how often a rule is applicable to a given

data set, while confidence determines how frequently items in Y appear in transactions that contain X. The formal definitions of these metrics are

$$\text{Support, } s(X \longrightarrow Y) \;=\; \frac{\sigma(X \cup Y)}{N}; \tag{6.1}$$

$$\text{Confidence, } c(X \longrightarrow Y) \;=\; \frac{\sigma(X \cup Y)}{\sigma(X)}. \tag{6.2}$$

Example 6.1. Consider the rule {Milk, Diapers} $\longrightarrow$ {Beer}. Since the support count for {Milk, Diapers, Beer} is 2 and the total number of transactions is 5, the rule's support is $2/5 = 0.4$. The rule's confidence is obtained by dividing the support count for {Milk, Diapers, Beer} by the support count for {Milk, Diapers}. Since there are 3 transactions that contain milk and diapers, the confidence for this rule is $2/3 = 0.67$. ∎

Why Use Support and Confidence?   Support is an important measure because a rule that has very low support may occur simply by chance. A low support rule is also likely to be uninteresting from a business perspective because it may not be profitable to promote items that customers seldom buy together (with the exception of the situation described in Section 6.8). For these reasons, support is often used to eliminate uninteresting rules. As will be shown in Section 6.2.1, 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. For a given rule $X \longrightarrow Y$, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Confidence also provides an estimate of the conditional probability of Y given X.

Association analysis results should be interpreted with caution. The inference made by an association rule does not necessarily imply causality. Instead, it suggests a strong co-occurrence relationship between items in the antecedent and consequent of the rule. Causality, on the other hand, requires knowledge about the causal and effect attributes in the data and typically involves relationships occurring over time (e.g., ozone depletion leads to global warming).

Formulation of Association Rule Mining Problem   The association rule mining problem can be formally stated as follows:

Definition 6.1 (Association Rule Discovery).   Given a set of transactions T, find all the rules having support $\geq minsup$ and confidence $\geq minconf$, where $minsup$ and $minconf$ are the corresponding support and confidence thresholds.

A brute-force approach for mining association rules is to compute the support and confidence for every possible rule. This approach is 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 d items is

$$R = 3^d - 2^{d+1} + 1.\tag{6.3}$$

The proof for this equation is left as an exercise to the readers (see Exercise 5 on page 405). Even for the small data set shown in Table 6.1, 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 $minsup = 20\%$ and $minconf = 50\%$, 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 their support and confidence values.

An initial step toward improving the performance of association rule mining algorithms is to decouple the support and confidence requirements. From Equation 6.2, notice that the support of a rule $X \longrightarrow Y$ depends only on the support of its corresponding itemset, $X \cup Y$. For example, the following rules have identical support because they involve items from the same itemset, {Beer, Diapers, Milk}:

{Beer, Diapers} $\longrightarrow$ {Milk},   {Beer, Milk} $\longrightarrow$ {Diapers},
{Diapers, Milk} $\longrightarrow$ {Beer},   {Beer} $\longrightarrow$ {Diapers, Milk},
{Milk} $\longrightarrow$ {Beer,Diapers},   {Diapers} $\longrightarrow$ {Beer,Milk}.

If the itemset is infrequent, then all six candidate rules can be pruned immediately without our having to compute their confidence values.

Therefore, a common strategy adopted by many association rule mining algorithms is to decompose the problem into two major subtasks:

1. **Frequent Itemset Generation**, whose objective is to find all the itemsets that satisfy the $minsup$ threshold. These itemsets are called frequent itemsets.

2. **Rule Generation**, whose objective is to extract all the high-confidence rules from the frequent itemsets found in the previous step. These rules are called strong rules.

The computational requirements for frequent itemset generation are generally more expensive than those of rule generation. Efficient techniques for generating frequent itemsets and association rules are discussed in Sections 6.2 and 6.3, respectively.
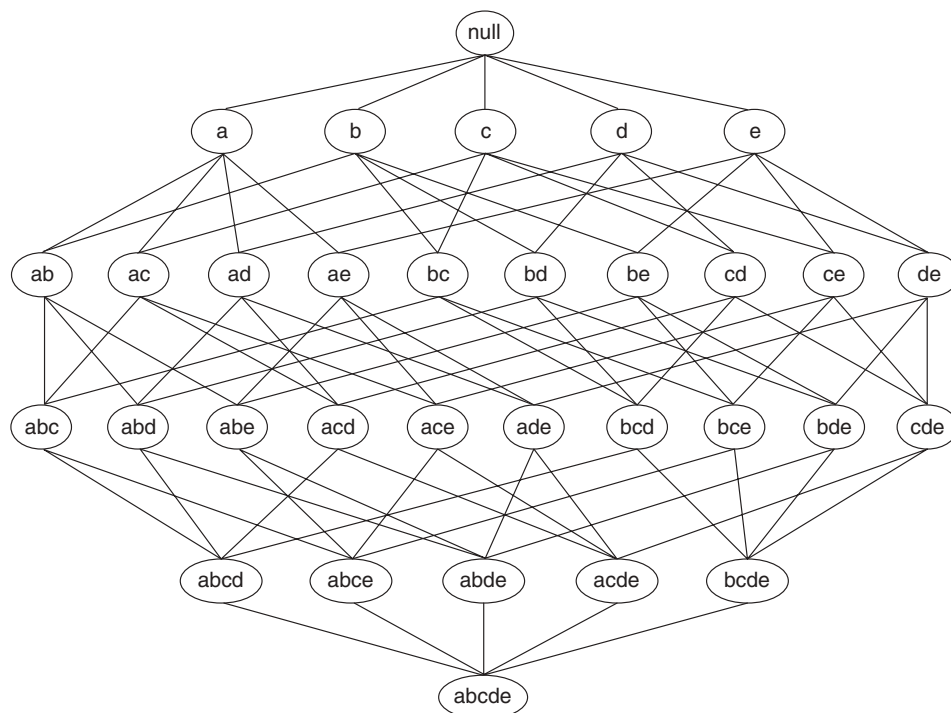
Figure 6.1. An itemset lattice.

6.2 Frequent Itemset Generation

A lattice structure can be used to enumerate the list of all possible itemsets. Figure 6.1 shows an itemset lattice for $I = \{a, b, c, d, e\}$. 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, an operation that is shown in Figure 6.2. 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.
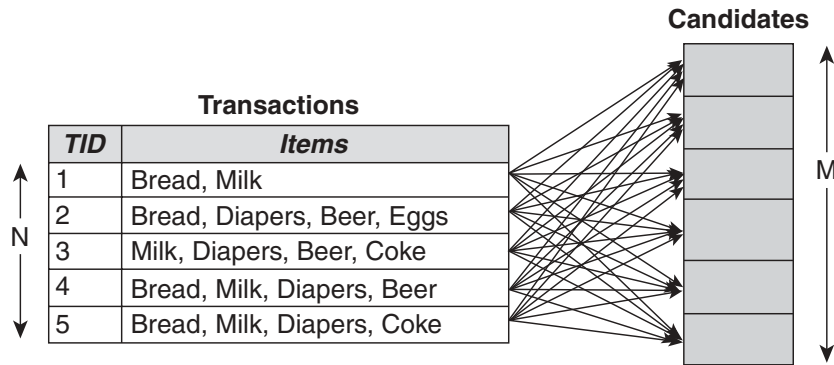
Candidates

Figure 6.2. Counting the support of candidate itemsets.

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

1. **Reduce the number of candidate itemsets** (M). The *Apriori* principle, described in the next section, is an effective way to eliminate some of the candidate itemsets without counting their support values.

2. **Reduce the number of comparisons.** Instead of matching each candidate itemset against every transaction, we can reduce the number of comparisons by using more advanced data structures, either to store the candidate itemsets or to compress the data set. We will discuss these strategies in Sections 6.2.4 and 6.6.

6.2.1 The *Apriori* Principle

This section describes how the support measure helps to reduce the number of candidate itemsets explored during frequent itemset generation. The use of support for pruning candidate itemsets is guided by the following principle.

Theorem 6.1 (*Apriori* Principle). *If an itemset is frequent, then all of its subsets must also be frequent.*

To illustrate the idea behind the *Apriori* principle, consider the itemset lattice shown in Figure 6.3. Suppose $\{c, d, e\}$ is a frequent itemset. Clearly, any transaction that contains $\{c, d, e\}$ must also contain its subsets, $\{c, d\}$, $\{c, e\}$, $\{d, e\}$, $\{c\}$, $\{d\}$, and $\{e\}$. As a result, if $\{c, d, e\}$ is frequent, then all subsets of $\{c, d, e\}$ (i.e., the shaded itemsets in this figure) must also be frequent.
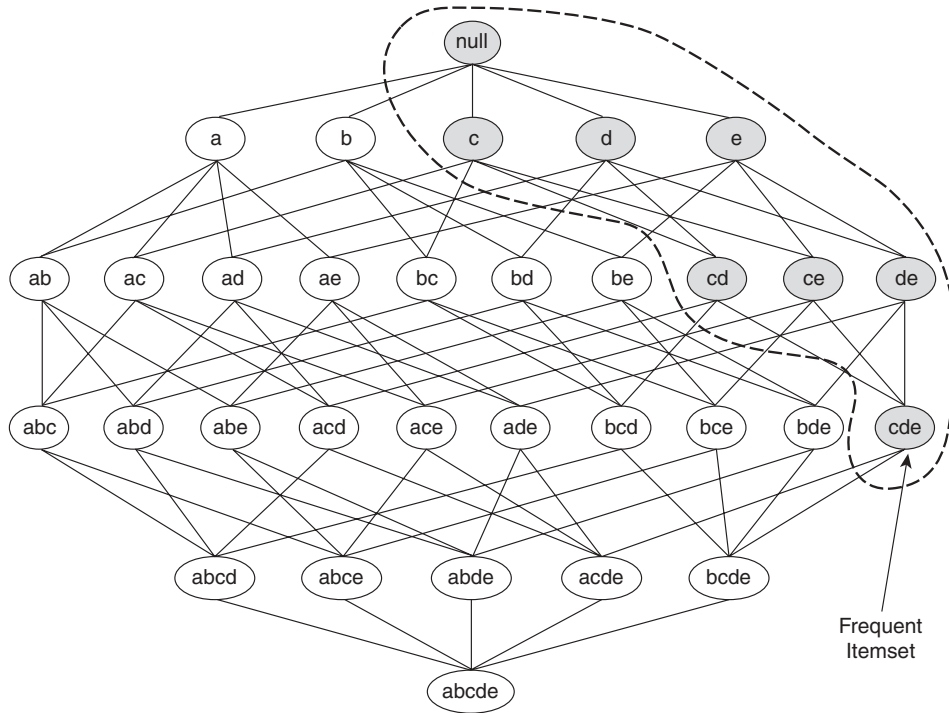
Figure 6.3.   An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

Conversely, if an itemset such as $\{a, b\}$ is infrequent, then all of its supersets must be infrequent too. As illustrated in Figure 6.4, the entire subgraph containing the supersets of $\{a, b\}$ can be pruned immediately once $\{a, b\}$ is found to be infrequent. This strategy of trimming the exponential search space based on the support measure is known as **support-based pruning**. Such a pruning strategy is made possible by a key property of the support measure, namely, that the support for an itemset never exceeds the support for its subsets. This property is also known as the **anti-monotone** property of the support measure.

Definition 6.2 (Monotonicity Property).   Let I be a set of items, and $J = 2^I$ be the power set of I. A measure f is monotone (or upward closed) if

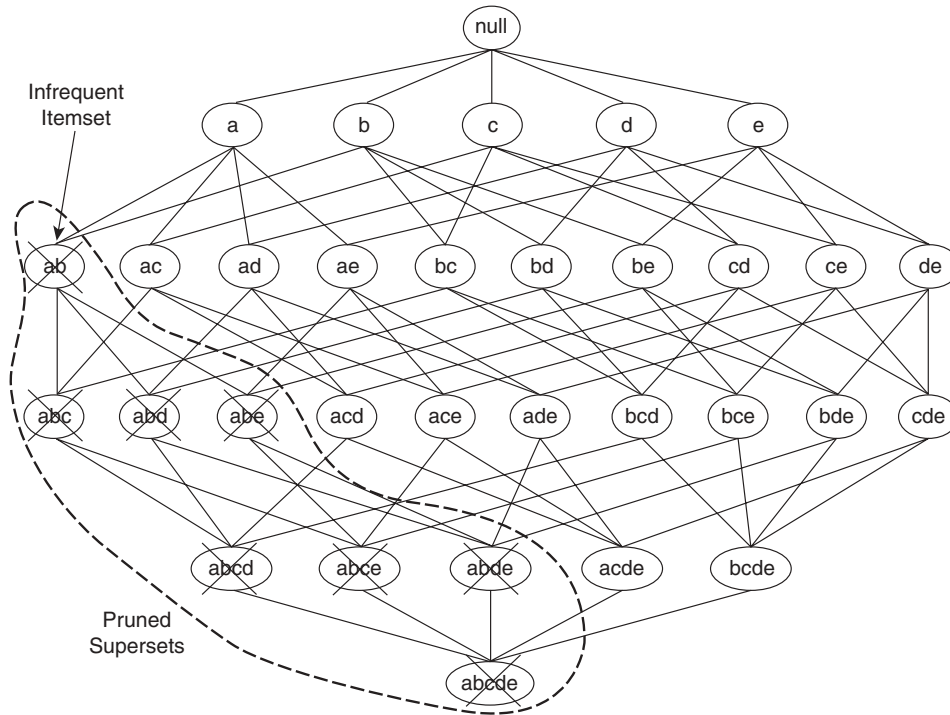$$\forall X, Y \in J : \ (X \subseteq Y) \longrightarrow f(X) \leq f(Y),$$

Figure 6.4. An illustration of support-based pruning. If $\{a, b\}$ is infrequent, then all supersets of $\{a, b\}$ are infrequent.

which means that if X is a subset of Y, then $f(X)$ must not exceed $f(Y)$. On the other hand, f is anti-monotone (or downward closed) if

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

which means that if X is a subset of Y, then $f(Y)$ must not exceed $f(X)$.

Any measure that possesses an anti-monotone property can be incorporated directly into the mining algorithm to effectively prune the exponential search space of candidate itemsets, as will be shown in the next section.

6.2.2 Frequent Itemset Generation in the *Apriori* Algorithm

Apriori is the first association rule mining algorithm that pioneered the use of support-based pruning to systematically control the exponential growth of candidate itemsets. Figure 6.5 provides a high-level illustration of the frequent itemset generation part of the *Apriori* algorithm for the transactions shown in
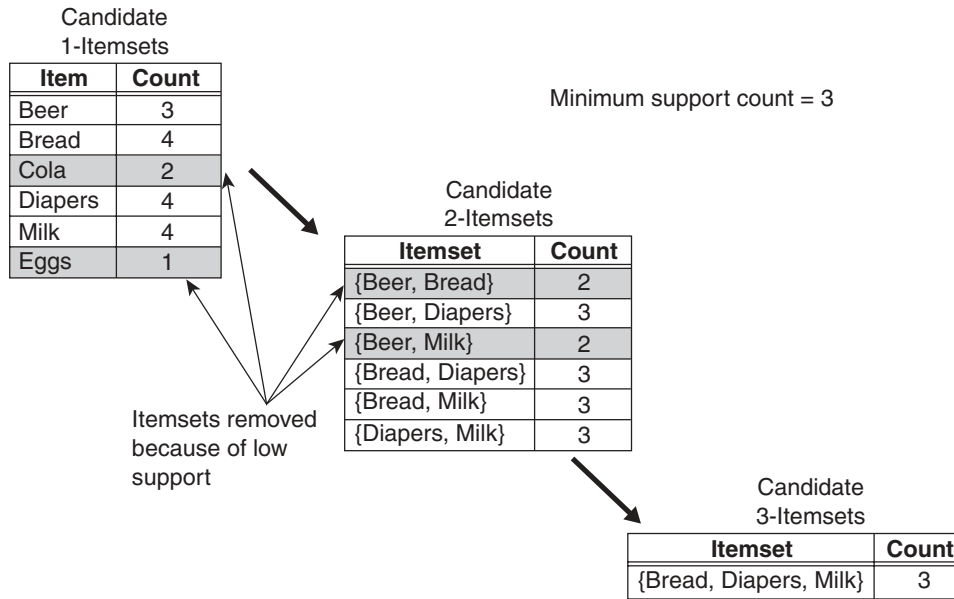
Candidate
1-Itemsets

Item	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

Minimum support count = 3

Candidate
2-Itemsets

Itemset	Count
{Beer, Bread}	2
{Beer, Diapers}	3
{Beer, Milk}	2
{Bread, Diapers}	3
{Bread, Milk}	3
{Diapers, Milk}	3

Itemsets removed
because of low
support

Candidate
3-Itemsets

Itemset	Count
{Bread, Diapers, Milk}	3

Figure 6.5. Illustration of frequent itemset generation using the *Apriori* algorithm.

Table 6.1. We assume that the support threshold is 60%, which is equivalent to a minimum support count equal to 3.

Initially, every item is considered as a candidate 1-itemset. After counting their supports, the candidate itemsets {Cola} and {Eggs} are discarded because they appear in fewer than three transactions. In the next iteration, candidate 2-itemsets are generated using only the frequent 1-itemsets because the *Apriori* principle ensures that all supersets of the infrequent 1-itemsets must be infrequent. Because there are only four frequent 1-itemsets, the number of candidate 2-itemsets generated by the algorithm is $\binom{4}{2} = 6$. Two of these six candidates, {Beer, Bread} and {Beer, Milk}, are subsequently found to be infrequent after computing their support values. The remaining four candidates are frequent, and thus will be used to generate candidate 3-itemsets. Without support-based pruning, there are $\binom{6}{3} = 20$ candidate 3-itemsets that can be formed using the six items given in this example. With the *Apriori* principle, we only need to keep candidate 3-itemsets whose subsets are frequent. The only candidate that has this property is {Bread, Diapers, Milk}.

The effectiveness of the *Apriori* pruning strategy can be shown by counting the number of candidate itemsets generated. A brute-force strategy of

enumerating all itemsets (up to size 3) as candidates will produce

$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} = 6 + 15 + 20 = 41$$

candidates. With the *Apriori* principle, this number decreases to

$$\binom{6}{1} + \binom{4}{2} + 1 = 6 + 6 + 1 = 13$$

candidates, which represents a 68% reduction in the number of candidate itemsets even in this simple example.

The pseudocode for the frequent itemset generation part of the *Apriori* algorithm is shown in Algorithm 6.1. Let C_k denote the set of candidate k-itemsets and F_k denote the set of frequent k-itemsets:

- The algorithm initially makes a single pass over the data set to determine the support of each item. Upon completion of this step, the set of all frequent 1-itemsets, F_1, will be known (steps 1 and 2).

- Next, the algorithm will iteratively generate new candidate k-itemsets using the frequent $(k-1)$-itemsets found in the previous iteration (step 5). Candidate generation is implemented using a function called apriori-gen, which is described in Section 6.2.3.

Algorithm 6.1 Frequent itemset generation of the *Apriori* algorithm.

1: $k = 1$.
2: $F_k = \{\, i \mid i \in I \wedge \sigma(\{i\}) \geq N \times minsup\}$.   {Find all frequent 1-itemsets}
3: **repeat**
4:    $k = k + 1$.
5:    $C_k = $ apriori-gen(F_{k-1}).   {Generate candidate itemsets}
6:    **for** each transaction $t \in T$ **do**
7:       $C_t = $ subset(C_k, t).   {Identify all candidates that belong to t}
8:       **for** each candidate itemset $c \in C_t$ **do**
9:          $\sigma(c) = \sigma(c) + 1$.   {Increment support count}
10:       **end for**
11:    **end for**
12:    $F_k = \{\, c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$.   {Extract the frequent k-itemsets}
13: **until** $F_k = \emptyset$
14: Result $= \bigcup F_k$.

```
subset(Ck,t) returns all itemsets of Ck
that are contained in transaction t

See Section 6.2.4
```

- To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 6–10). The subset function is used to determine all the candidate itemsets in C_k that are contained in each transaction t. The implementation of this function is described in Section 6.2.4.

- After counting their supports, the algorithm eliminates all candidate itemsets whose support counts are less than *minsup* (step 12).

- The algorithm terminates when there are no new frequent itemsets generated, i.e., $F_k = \emptyset$ (step 13).

The frequent itemset generation part of the *Apriori* algorithm has two important characteristics. First, it is a **level-wise** algorithm; i.e., it traverses the itemset lattice one level at a time, from frequent 1-itemsets to the maximum size of frequent itemsets. Second, it employs a **generate-and-test** strategy for finding frequent itemsets. At each iteration, new candidate itemsets are generated from the frequent itemsets found in the previous iteration. The support for each candidate is then counted and tested against the *minsup* threshold. The total number of iterations needed by the algorithm is $k_{\max} + 1$, where $k_{\max}$ is the maximum size of the frequent itemsets.

6.2.3 Candidate Generation and Pruning

The apriori-gen function shown in Step 5 of Algorithm 6.1 generates candidate itemsets by performing the following two operations:

1. **Candidate Generation.** This operation generates new candidate k-itemsets based on the frequent $(k-1)$-itemsets found in the previous iteration.

`apriori-gen does not access the database`

2. **Candidate Pruning.** This operation eliminates some of the candidate k-itemsets using the support-based pruning strategy.

To illustrate the candidate pruning operation, consider a candidate k-itemset, $X = \{i_1, i_2, \ldots, i_k\}$. The algorithm must determine whether all of its proper subsets, $X - \{i_j\}$ ($\forall j = 1, 2, \ldots, k$), are frequent. If one of them is infrequent, then X is immediately pruned. This approach can effectively reduce the number of candidate itemsets considered during support counting. The complexity of this operation is $O(k)$ for each candidate k-itemset. However, as will be shown later, we do not have to examine all k subsets of a given candidate itemset. If m of the k subsets were used to generate a candidate, we only need to check the remaining $k - m$ subsets during candidate pruning.

In principle, there are many ways to generate candidate itemsets. The following is a list of requirements for an effective candidate generation procedure:

1. It should avoid generating too many unnecessary candidates. A candidate itemset is unnecessary if at least one of its subsets is infrequent. Such a candidate is guaranteed to be infrequent according to the anti-monotone property of support.

2. It must ensure that the candidate set is complete, i.e., no frequent itemsets are left out by the candidate generation procedure. To ensure completeness, the set of candidate itemsets must subsume the set of all frequent itemsets, i.e., $\forall k : F_k \subseteq C_k$.

3. It should not generate the same candidate itemset more than once. For example, the candidate itemset $\{a, b, c, d\}$ can be generated in many ways—by merging $\{a, b, c\}$ with $\{d\}$, $\{b, d\}$ with $\{a, c\}$, $\{c\}$ with $\{a, b, d\}$, etc. Generation of duplicate candidates leads to wasted computations and thus should be avoided for efficiency reasons.

Next, we will briefly describe several candidate generation procedures, including the one used by the apriori-gen function.

Brute-Force Method The brute-force method considers every k-itemset as a potential candidate and then applies the candidate pruning step to remove any unnecessary candidates (see Figure 6.6). The number of candidate itemsets generated at level k is equal to $\binom{d}{k}$, where d is the total number of items. Although candidate generation is rather trivial, candidate pruning becomes extremely expensive because a large number of itemsets must be examined. Given that the amount of computations needed for each candidate is $O(k)$, the overall complexity of this method is $O\left(\sum_{k=1}^{d} k \times \binom{d}{k}\right) = O(d \cdot 2^{d-1})$.

$\mathbf{F}_{k-1} \times \mathbf{F}_1$ Method An alternative method for candidate generation is to extend each frequent $(k-1)$-itemset with other frequent items. Figure 6.7 illustrates how a frequent 2-itemset such as {Beer, Diapers} can be augmented with a frequent item such as Bread to produce a candidate 3-itemset {Beer, Diapers, Bread}. This method will produce $O(|F_{k-1}| \times |F_1|)$ candidate k-itemsets, where $|F_j|$ is the number of frequent j-itemsets. The overall complexity of this step is $O(\sum_k k|F_{k-1}||F_1|)$.

The procedure is complete because every frequent k-itemset is composed of a frequent $(k-1)$-itemset and a frequent 1-itemset. Therefore, all frequent k-itemsets are part of the candidate k-itemsets generated by this procedure.
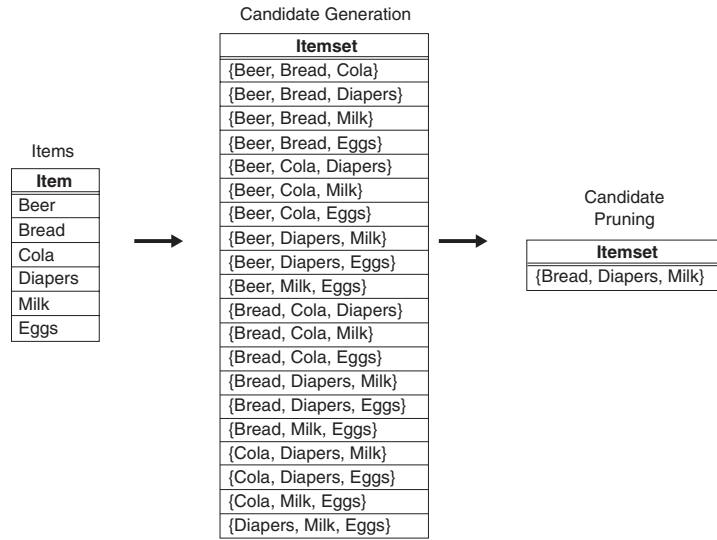
Candidate Generation

Itemset
{Beer, Bread, Cola}
{Beer, Bread, Diapers}
{Beer, Bread, Milk}
{Beer, Bread, Eggs}
{Beer, Cola, Diapers}
{Beer, Cola, Milk}
{Beer, Cola, Eggs}
{Beer, Diapers, Milk}
{Beer, Diapers, Eggs}
{Beer, Milk, Eggs}
{Bread, Cola, Diapers}
{Bread, Cola, Milk}
{Bread, Cola, Eggs}
{Bread, Diapers, Milk}
{Bread, Diapers, Eggs}
{Bread, Milk, Eggs}
{Cola, Diapers, Milk}
{Cola, Diapers, Eggs}
{Cola, Milk, Eggs}
{Diapers, Milk, Eggs}

Items

Item
Beer
Bread
Cola
Diapers
Milk
Eggs

Candidate
Pruning

Itemset
{Bread, Diapers, Milk}

Figure 6.6. A brute-force method for generating candidate 3-itemsets.

Frequent
2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Frequent
1-itemset

Item
Beer
Bread
Diapers
Milk

Candidate Generation

Itemset
{Beer, Diapers, Bread}
{Beer, Diapers, Milk}
{Bread, Diapers, Milk}
{Bread, Milk, Beer}

Candidate
Pruning

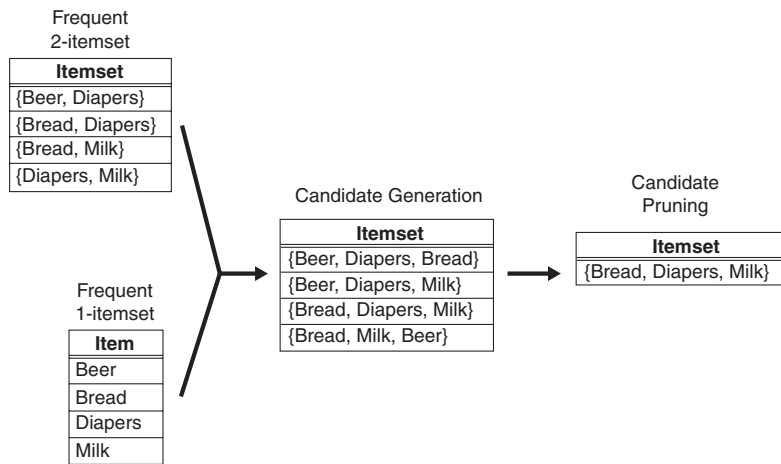| Itemset |
|---|
| {Bread, Diapers, Milk} |

Figure 6.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

This approach, however, does not prevent the same candidate itemset from being generated more than once. For instance, {Bread, Diapers, Milk} can be generated by merging {Bread, Diapers} with {Milk}, {Bread, Milk} with {Diapers}, or {Diapers, Milk} with {Bread}. One way to avoid generating

duplicate candidates is by ensuring that the items in each frequent itemset are kept sorted in their lexicographic order. Each frequent $(k-1)$-itemset X is then extended with frequent items that are lexicographically larger than the items in X. For example, the itemset {Bread, Diapers} can be augmented with {Milk} since Milk is lexicographically larger than Bread and Diapers. However, we should not augment {Diapers, Milk} with {Bread} nor {Bread, Milk} with {Diapers} because they violate the lexicographic ordering condition.

While this procedure is a substantial improvement over the brute-force method, it can still produce a large number of unnecessary candidates. For example, the candidate itemset obtained by merging {Beer, Diapers} with {Milk} is unnecessary because one of its subsets, {Beer, Milk}, is infrequent. There are several heuristics available to reduce the number of unnecessary candidates. For example, note that, for every candidate k-itemset that survives the pruning step, every item in the candidate must be contained in at least $k-1$ of the frequent $(k-1)$-itemsets. Otherwise, the candidate is guaranteed to be infrequent. For example, {Beer, Diapers, Milk} is a viable candidate 3-itemset only if every item in the candidate, including Beer, is contained in at least two frequent 2-itemsets. Since there is only one frequent 2-itemset containing Beer, all candidate itemsets involving Beer must be infrequent.

$\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ **Method** The candidate generation procedure in the apriori-gen function merges a pair of frequent $(k-1)$-itemsets only if their first $k-2$ items are identical. Let $A = \{a_1, a_2, \ldots, a_{k-1}\}$ and $B = \{b_1, b_2, \ldots, b_{k-1}\}$ be a pair of frequent $(k-1)$-itemsets. A and B are merged if they satisfy the following conditions:

$$a_i = b_i \text{ (for } i = 1, 2, \ldots, k-2) \text{ and } a_{k-1} \neq b_{k-1}.$$

In Figure 6.8, the frequent itemsets {Bread, Diapers} and {Bread, Milk} are merged to form a candidate 3-itemset {Bread, Diapers, Milk}. The algorithm does not have to merge {Beer, Diapers} with {Diapers, Milk} because the first item in both itemsets is different. Indeed, if {Beer, Diapers, Milk} is a viable candidate, it would have been obtained by merging {Beer, Diapers} with {Beer, Milk} instead. This example illustrates both the completeness of the candidate generation procedure and the advantages of using lexicographic ordering to prevent duplicate candidates. However, because each candidate is obtained by merging a pair of frequent $(k-1)$-itemsets, an additional candidate pruning step is needed to ensure that the remaining $k-2$ subsets of the candidate are frequent.
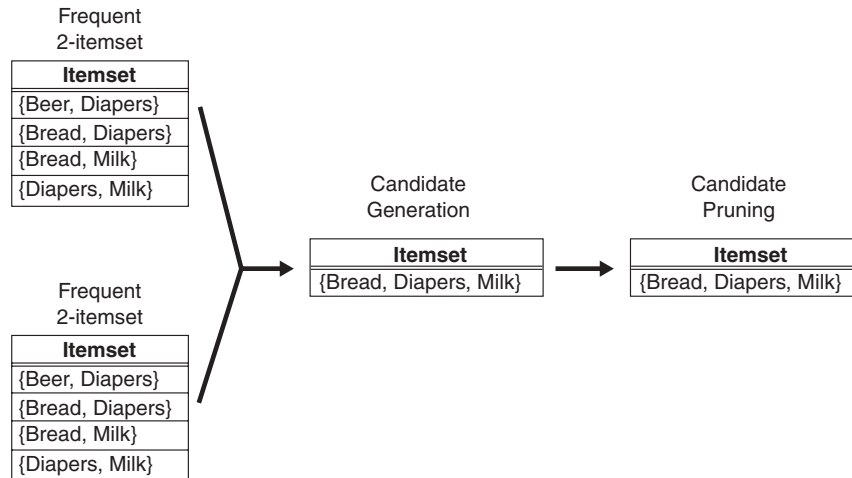
Frequent
2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Candidate
Generation

Itemset
{Bread, Diapers, Milk}

Candidate
Pruning

Itemset
{Bread, Diapers, Milk}

Frequent
2-itemset

Itemset
{Beer, Diapers}
{Bread, Diapers}
{Bread, Milk}
{Diapers, Milk}

Figure 6.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

6.2.4   Support Counting

Support counting is the process of determining the frequency of occurrence for every candidate itemset that survives the candidate pruning step of the apriori-gen function. Support counting is implemented in steps 6 through 11 of Algorithm 6.1. One approach for doing this is to compare each transaction against every candidate itemset (see Figure 6.2) and to update the support counts of candidates contained in the transaction. This approach is computationally expensive, especially when the numbers of transactions and candidate itemsets are large.

An alternative approach is to enumerate the itemsets contained in each transaction and use them to update the support counts of their respective candidate itemsets. To illustrate, consider a transaction t that contains five items, $\{1, 2, 3, 5, 6\}$. There are $\binom{5}{3} = 10$ itemsets of size 3 contained in this transaction. Some of the itemsets may correspond to the candidate 3-itemsets under investigation, in which case, their support counts are incremented. Other subsets of t that do not correspond to any candidates can be ignored.

Figure 6.9 shows a systematic way for enumerating the 3-itemsets contained in t. Assuming that each itemset keeps its items in increasing lexicographic order, an itemset can be enumerated by specifying the smallest item first, followed by the larger items. For instance, given $t = \{1, 2, 3, 5, 6\}$, all the 3-itemsets contained in t must begin with item 1, 2, or 3. It is not possible to construct a 3-itemset that begins with items 5 or 6 because there are only two
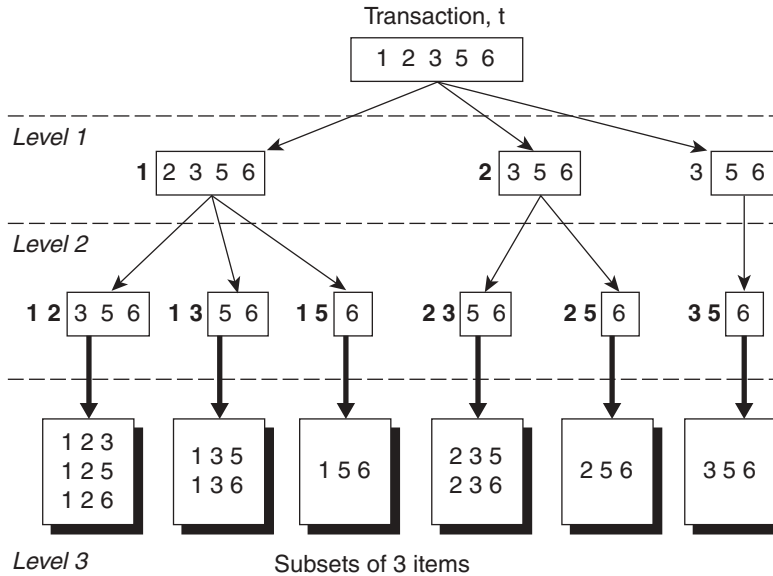
Transaction, t

1 2 3 5 6

Level 1

1 2 3 5 6   **2** 3 5 6   3 5 6

Level 2

1 2 3 5 6   **1 3** 5 6   **1 5** 6   **2 3** 5 6   **2 5** 6   **3 5** 6

1 2 3
1 2 5   1 3 5   1 5 6   2 3 5   2 5 6   3 5 6
1 2 6   1 3 6   2 3 6

Level 3   Subsets of 3 items

Figure 6.9. Enumerating subsets of three items from a transaction t.

items in t whose labels are greater than or equal to 5. The number of ways to specify the first item of a 3-itemset contained in t is illustrated by the Level 1 prefix structures depicted in Figure 6.9. For instance, 1 $\boxed{2\ 3\ 5\ 6}$ represents a 3-itemset that begins with item 1, followed by two more items chosen from the set $\{2, 3, 5, 6\}$.

After fixing the first item, the prefix structures at Level 2 represent the number of ways to select the second item. For example, 1 2 $\boxed{3\ 5\ 6}$ corresponds to itemsets that begin with prefix (1 2) and are followed by items 3, 5, or 6. Finally, the prefix structures at Level 3 represent the complete set of 3-itemsets contained in t. For example, the 3-itemsets that begin with prefix $\{1\ 2\}$ are $\{1, 2, 3\}$, $\{1, 2, 5\}$, and $\{1, 2, 6\}$, while those that begin with prefix $\{2\ 3\}$ are $\{2, 3, 5\}$ and $\{2, 3, 6\}$.

The prefix structures shown in Figure 6.9 demonstrate how itemsets contained in a transaction can be systematically enumerated, i.e., by specifying their items one by one, from the leftmost item to the rightmost item. We still have to determine whether each enumerated 3-itemset corresponds to an existing candidate itemset. If it matches one of the candidates, then the support count of the corresponding candidate is incremented. In the next section, we illustrate how this matching operation can be performed efficiently using a hash tree structure.
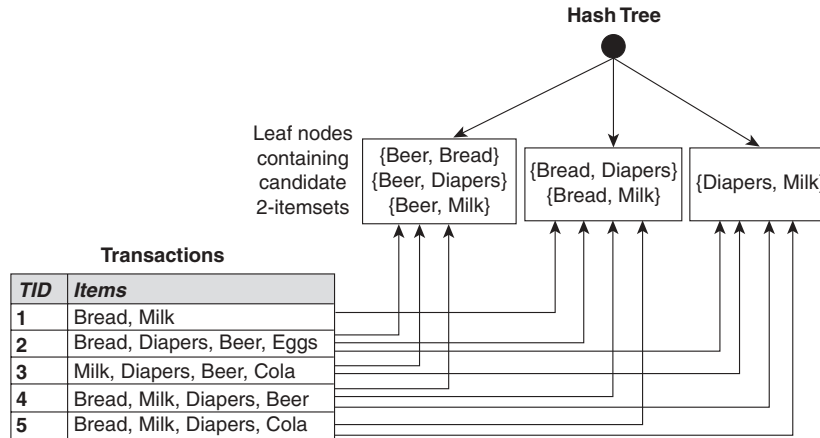
Hash Tree

Leaf nodes containing candidate 2-itemsets

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

{Bread, Diapers}
{Bread, Milk}

{Diapers, Milk}

Transactions

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

Figure 6.10. Counting the support of itemsets using hash structure.

Support Counting Using a Hash Tree

In the *Apriori* algorithm, candidate itemsets are partitioned into different buckets and stored in a hash tree. During support counting, itemsets contained in each transaction are also hashed into their appropriate buckets. That way, instead of comparing each itemset in the transaction with every candidate itemset, it is matched only against candidate itemsets that belong to the same bucket, as shown in Figure 6.10.

Figure 6.11 shows an example of a hash tree structure. Each internal node of the tree uses the following hash function, $h(p) = p \bmod 3$, to determine which branch of the current node should be followed next. For example, items 1, 4, and 7 are hashed to the same branch (i.e., the leftmost branch) because they have the same remainder after dividing the number by 3. All candidate itemsets are stored at the leaf nodes of the hash tree. The hash tree shown in Figure 6.11 contains 15 candidate 3-itemsets, distributed across 9 leaf nodes.

Consider a transaction, $t = \{1, 2, 3, 5, 6\}$. To update the support counts of the candidate itemsets, the hash tree must be traversed in such a way that all the leaf nodes containing candidate 3-itemsets belonging to t must be visited at least once. Recall that the 3-itemsets contained in t must begin with items 1, 2, or 3, as indicated by the Level 1 prefix structures shown in Figure 6.9. Therefore, at the root node of the hash tree, the items 1, 2, and 3 of the transaction are hashed separately. Item 1 is hashed to the left child of the root node, item 2 is hashed to the middle child, and item 3 is hashed to the right child. At the next level of the tree, the transaction is hashed on the second
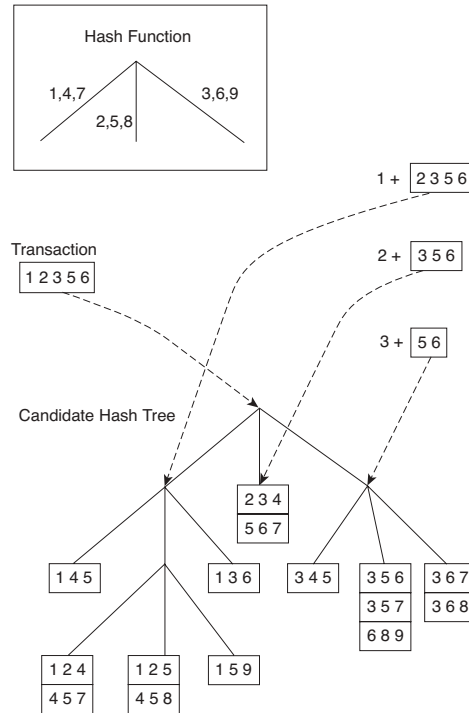
Figure 6.11. Hashing a transaction at the root node of a hash tree.

item listed in the Level 2 structures shown in Figure 6.9. For example, after hashing on item 1 at the root node, items 2, 3, and 5 of the transaction are hashed. Items 2 and 5 are hashed to the middle child, while item 3 is hashed to the right child, as shown in Figure 6.12. This process continues until the leaf nodes of the hash tree are reached. The candidate itemsets stored at the visited leaf nodes are compared against the transaction. If a candidate is a subset of the transaction, its support count is incremented. In this example, 5 out of the 9 leaf nodes are visited and 9 out of the 15 itemsets are compared against the transaction.

6.2.5 Computational Complexity

The computational complexity of the *Apriori* algorithm can be affected by the following factors.

Support Threshold Lowering the support threshold often results in more itemsets being declared as frequent. This has an adverse effect on the com-
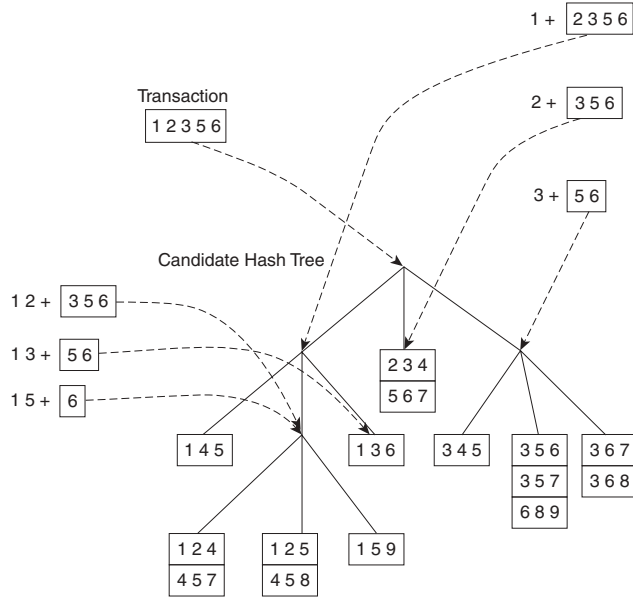
Figure 6.12. Subset operation on the leftmost subtree of the root of a candidate hash tree.

putational complexity of the algorithm because more candidate itemsets must be generated and counted, as shown in Figure 6.13. The maximum size of frequent itemsets also tends to increase with lower support thresholds. As the maximum size of the frequent itemsets increases, the algorithm will need to make more passes over the data set.

Number of Items (Dimensionality)   As the number of items increases, more space will be needed to store the support counts of items. If the number of frequent items also grows with the dimensionality of the data, the computation and I/O costs will increase because of the larger number of candidate itemsets generated by the algorithm.

**Number of Transactions**   Since the *Apriori* algorithm makes repeated passes over the data set, its run time increases with a larger number of transactions.

Average Transaction Width   For dense data sets, the average transaction width can be very large. This affects the complexity of the *Apriori* algorithm in two ways. First, the maximum size of frequent itemsets tends to increase as the
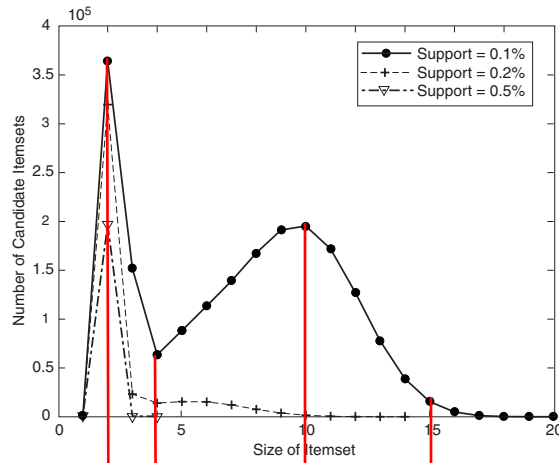
(a) Number of candidate itemsets.
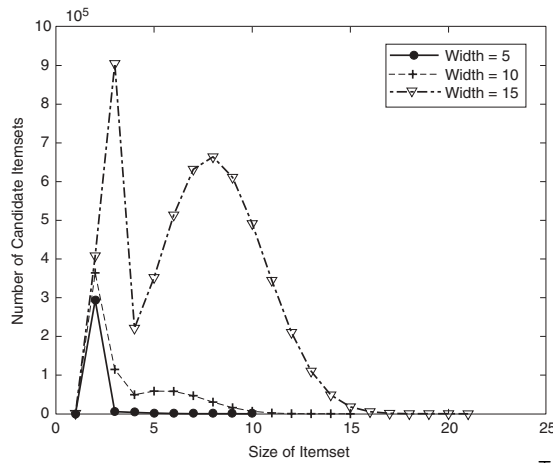
Dicuss this behavior.
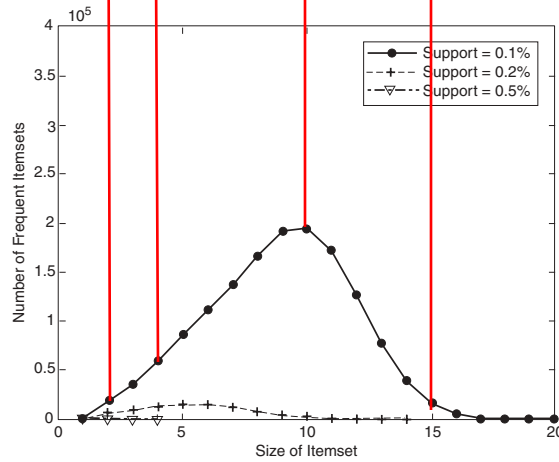
(b) Number of frequent itemsets.

Figure 6.13. Effect of support threshold on the number of candidate and frequent itemsets.

average transaction width increases. As a result, more candidate itemsets must be examined during candidate generation and support counting, as illustrated in Figure 6.14. Second, as the transaction width increases, more itemsets

(a) Number of candidate itemsets.

```
If the average tranasaction
width = 10, frequent itemsets
of size >> 10 are unlikely.
```
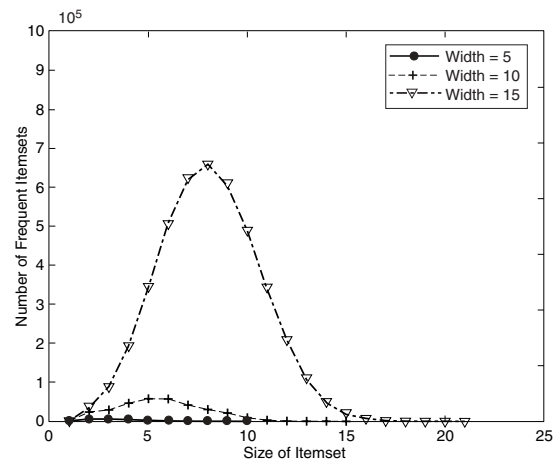

(b) Number of Frequent Itemsets.

Figure 6.14. Effect of average transaction width on the number of candidate and frequent itemsets.

are contained in the transaction. This will increase the number of hash tree traversals performed during support counting.

A detailed analysis of the time complexity for the *Apriori* algorithm is presented next.

Generation of frequent 1-itemsets For each transaction, we need to update the support count for every item present in the transaction. Assuming that w is the average transaction width, this operation requires $O(Nw)$ time, where N is the total number of transactions.

Candidate generation To generate candidate k-itemsets, pairs of frequent $(k - 1)$-itemsets are merged to determine whether they have at least $k - 2$ items in common. Each merging operation requires at most $k - 2$ equality comparisons. In the best-case scenario, every merging step produces a viable candidate k-itemset. In the worst-case scenario, the algorithm must merge every pair of frequent $(k-1)$-itemsets found in the previous iteration. Therefore, the overall cost of merging frequent itemsets is

$$\sum_{k=2}^{w}(k-2)|C_k| < \text{ Cost of merging } < \sum_{k=2}^{w}(k-2)|F_{k-1}|^2.$$

A hash tree is also constructed during candidate generation to store the candidate itemsets. Because the maximum depth of the tree is k, the cost for populating the hash tree with candidate itemsets is $O\left(\sum_{k=2}^{w} k|C_k|\right)$. During candidate pruning, we need to verify that the $k - 2$ subsets of every candidate k-itemset are frequent. Since the cost for looking up a candidate in a hash tree is $O(k)$, the candidate pruning step requires $O\left(\sum_{k=2}^{w} k(k-2)|C_k|\right)$ time.

Support counting Each transaction of length $|t|$ produces $\binom{|t|}{k}$ itemsets of size k. This is also the effective number of hash tree traversals performed for each transaction. The cost for support counting is $O\left(N \sum_{k} \binom{w}{k}\alpha_k\right)$, where w is the maximum transaction width and α_k is the cost for updating the support count of a candidate k-itemset in the hash tree.

N is the database size.

6.3 Rule Generation

This section describes how to extract association rules efficiently from a given frequent itemset. Each frequent k-itemset, Y, can produce up to $2^k - 2$ association rules, ignoring rules that have empty antecedents or consequents ($\emptyset \longrightarrow Y$ or $Y \longrightarrow \emptyset$). An association rule can be extracted by partitioning the itemset Y into two non-empty subsets, X and $Y - X$, such that $X \longrightarrow Y - X$ satisfies the confidence threshold. Note that all such rules must have already met the support threshold because they are generated from a frequent itemset.

Example 6.2. Let $X = \{1, 2, 3\}$ be a frequent itemset. There are six candidate association rules that can be generated from X: $\{1, 2\} \longrightarrow \{3\}$, $\{1, 3\} \longrightarrow \{2\}$, $\{2, 3\} \longrightarrow \{1\}$, $\{1\} \longrightarrow \{2, 3\}$, $\{2\} \longrightarrow \{1, 3\}$, and $\{3\} \longrightarrow \{1, 2\}$. As each of their support is identical to the support for X, the rules must satisfy the support threshold. ∎

Computing the confidence of an association rule does not require additional scans of the transaction data set. Consider the rule $\{1, 2\} \longrightarrow \{3\}$, which is generated from the frequent itemset $X = \{1, 2, 3\}$. The confidence for this rule is $\sigma(\{1, 2, 3\})/\sigma(\{1, 2\})$. Because $\{1, 2, 3\}$ is frequent, the anti-monotone property of support ensures that $\{1, 2\}$ must be frequent, too. Since the support counts for both itemsets were already found during frequent itemset generation, there is no need to read the entire data set again.

6.3.1 Confidence-Based Pruning

Unlike the support measure, confidence does not have any monotone property. For example, the confidence for $X \longrightarrow Y$ can be larger, smaller, or equal to the confidence for another rule $\tilde{X} \longrightarrow \tilde{Y}$, where $\tilde{X} \subseteq X$ and $\tilde{Y} \subseteq Y$ (see Exercise 3 on page 405). Nevertheless, if we compare rules generated from the same frequent itemset Y, the following theorem holds for the confidence measure.

Theorem 6.2. *If a rule $X \longrightarrow Y - X$ does not satisfy the confidence threshold, then any rule $X' \longrightarrow Y - X'$, where X' is a subset of X, must not satisfy the confidence threshold as well.*

To prove this theorem, consider the following two rules: $X' \longrightarrow Y - X'$ and $X \longrightarrow Y - X$, where $X' \subset X$. The confidence of the rules are $\sigma(Y)/\sigma(X')$ and $\sigma(Y)/\sigma(X)$, respectively. Since X' is a subset of X, $\sigma(X') \geq \sigma(X)$. Therefore, the former rule cannot have a higher confidence than the latter rule.

6.3.2 Rule Generation in *Apriori* Algorithm

The *Apriori* algorithm uses a level-wise approach for generating association rules, where each level corresponds to the number of items that belong to the rule consequent. Initially, all the high-confidence rules that have only one item in the rule consequent are extracted. These rules are then used to generate new candidate rules. For example, if $\{acd\} \longrightarrow \{b\}$ and $\{abd\} \longrightarrow \{c\}$ are high-confidence rules, then the candidate rule $\{ad\} \longrightarrow \{bc\}$ is generated by merging the consequents of both rules. Figure 6.15 shows a lattice structure for the association rules generated from the frequent itemset $\{a, b, c, d\}$. If any node in the lattice has low confidence, then according to Theorem 6.2, the
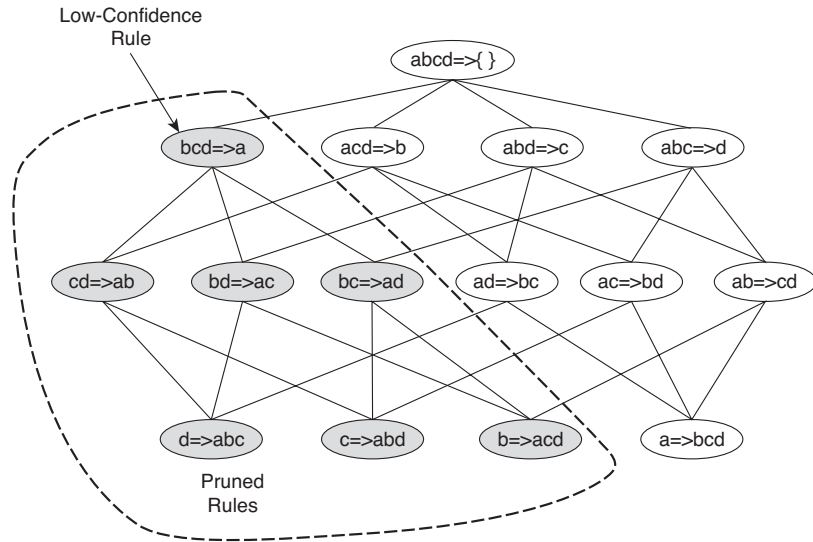
Figure 6.15. Pruning of association rules using the confidence measure.

entire subgraph spanned by the node can be pruned immediately. Suppose the confidence for $\{bcd\} \longrightarrow \{a\}$ is low. All the rules containing item a in its consequent, including $\{cd\} \longrightarrow \{ab\}$, $\{bd\} \longrightarrow \{ac\}$, $\{bc\} \longrightarrow \{ad\}$, and $\{d\} \longrightarrow \{abc\}$ can be discarded.

A pseudocode for the rule generation step is shown in Algorithms 6.2 and 6.3. Note the similarity between the `ap-genrules` procedure given in Algorithm 6.3 and the frequent itemset generation procedure given in Algorithm 6.1. The only difference is that, in rule generation, we do not have to make additional passes over the data set to compute the confidence of the candidate rules. Instead, we determine the confidence of each rule by using the support counts computed during frequent itemset generation.

Algorithm 6.2 Rule generation of the *Apriori* algorithm.

1: **for** each frequent k-itemset f_k, $k \geq 2$ **do**
2:    $H_1 = \{i \mid i \in f_k\}$ {1-item consequents of the rule.}
3:    **call** `ap-genrules`$(f_k, H_1.)$
4: **end for**

---

fk denotes a frequent k-itemset.

Algorithm 6.3 Procedure ap-genrules(f_k, H_m).

1: $k = |f_k|$ {size of frequent itemset.}
2: $m = |H_m|$ {size of rule consequent.}
3: **if** $k > m + 1$ **then**
4:    $H_{m+1} = $ apriori-gen(H_m).
5:    **for** each $h_{m+1} \in H_{m+1}$ **do**
6:       $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$.
7:       **if** $conf \geq minconf$ **then**
8:          **output** the rule $(f_k - h_{m+1}) \longrightarrow h_{m+1}$.
9:       **else**
10:          **delete** h_{m+1} from H_{m+1}.
11:       **end if**
12:    **end for**
13:    **call** ap-genrules(f_k, H_{m+1}.)
14: **end if**

(annotation: Hm is a set of m-itemsets)

(annotation: fk is a frequent k-itemset)

(annotation: Mistake: m = size of elements in Hm)

(annotation: For example, A->BCD is frequent only if each of the following is frequent: A->BC A->BD A->CD)

...ssional Voting Records

...lts of applying association analysis to the ...nited States House of Representatives. The ...gressional Voting Records Database, which ...earning data repository. Each transaction ...ty affiliation for a representative along with his or her voting record on 16 key issues. There are 435 transactions and 34 items in the data set. The set of items are listed in Table 6.3.

The *Apriori* algorithm is then applied to the data set with $minsup = 30\%$ and $minconf = 90\%$. Some of the high-confidence rules extracted by the algorithm are shown in Table 6.4. The first two rules suggest that most of the members who voted yes for aid to El Salvador and no for budget resolution and MX missile are Republicans; while those who voted no for aid to El Salvador and yes for budget resolution and MX missile are Democrats. These high-confidence rules show the key issues that divide members from both political parties. If $minconf$ is reduced, we may find rules that contain issues that cut across the party lines. For example, with $minconf = 40\%$, the rules suggest that corporation cutbacks is an issue that receives almost equal number of votes from both parties—52.3% of the members who voted no are Republicans, while the remaining 47.7% of them who voted no are Democrats.

Table 6.3. List of binary attributes from the 1984 United States Congressional Voting Records. Source: The UCI machine learning repository.

1. Republican	18. aid to Nicaragua = no
2. Democrat	19. MX-missile = yes
3. handicapped-infants = yes	20. MX-missile = no
4. handicapped-infants = no	21. immigration = yes
5. water project cost sharing = yes	22. immigration = no
6. water project cost sharing = no	23. synfuel corporation cutback = yes
7. budget-resolution = yes	24. synfuel corporation cutback = no
8. budget-resolution = no	25. education spending = yes
9. physician fee freeze = yes	26. education spending = no
10. physician fee freeze = no	27. right-to-sue = yes
11. aid to El Salvador = yes	28. right-to-sue = no
12. aid to El Salvador = no	29. crime = yes
13. religious groups in schools = yes	30. crime = no
14. religious groups in schools = no	31. duty-free-exports = yes
15. anti-satellite test ban = yes	32. duty-free-exports = no
16. anti-satellite test ban = no	33. export administration act = yes
17. aid to Nicaragua = yes	34. export administration act = no

Table 6.4. Association rules extracted from the 1984 United States Congressional Voting Records.

Association Rule	Confidence
{budget resolution = no, MX-missile=no, aid to El Salvador = yes } $\longrightarrow$ {Republican}	91.0%
{budget resolution = yes, MX-missile=yes, aid to El Salvador = no } $\longrightarrow$ {Democrat}	97.5%
{crime = yes, right-to-sue = yes, physician fee freeze = yes} $\longrightarrow$ {Republican}	93.5%
{crime = no, right-to-sue = no, physician fee freeze = no} $\longrightarrow$ {Democrat}	100%

6.4 Compact Representation of Frequent Itemsets

In practice, the number of frequent itemsets produced from a transaction data set can be very large. It is useful to identify a small representative set of itemsets from which all other frequent itemsets can be derived. Two such representations are presented in this section in the form of maximal and closed frequent itemsets.
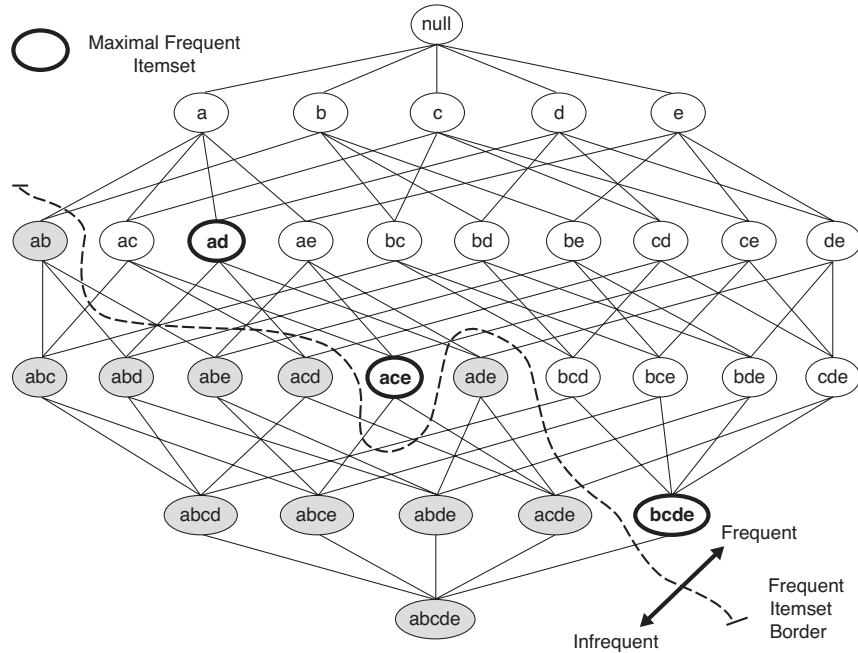
Figure 6.16. Maximal frequent itemset.

6.4.1   Maximal Frequent Itemsets

Definition 6.3 (Maximal Frequent Itemset).   A maximal frequent itemset is defined as a frequent itemset for which none of its immediate supersets are frequent.

To illustrate this concept, consider the itemset lattice shown in Figure 6.16. The itemsets in the lattice are divided into two groups: those that are frequent and those that are infrequent. A frequent itemset border, which is represented by a dashed line, is also illustrated in the diagram. Every itemset located above the border is frequent, while those located below the border (the shaded nodes) are infrequent. Among the itemsets residing near the border, $\{a, d\}$, $\{a, c, e\}$, and $\{b, c, d, e\}$ are considered to be maximal frequent itemsets because their immediate supersets are infrequent. An itemset such as $\{a, d\}$ is maximal frequent because all of its immediate supersets, $\{a, b, d\}$, $\{a, c, d\}$, and $\{a, d, e\}$, are infrequent. In contrast, $\{a, c\}$ is non-maximal because one of its immediate supersets, $\{a, c, e\}$, is frequent.

Maximal frequent itemsets effectively provide a compact representation of frequent itemsets. In other words, they form the smallest set of itemsets from

which all frequent itemsets can be derived. For example, the frequent itemsets shown in Figure 6.16 can be divided into two groups:

- Frequent itemsets that begin with item a and that may contain items c, d, or e. This group includes itemsets such as $\{a\}$, $\{a,c\}$, $\{a,d\}$, $\{a,e\}$, and $\{a,c,e\}$.

- Frequent itemsets that begin with items b, c, d, or e. This group includes itemsets such as $\{b\}$, $\{b,c\}$, $\{c,d\}$,$\{b,c,d,e\}$, etc.

Frequent itemsets that belong in the first group are subsets of either $\{a,c,e\}$ or $\{a,d\}$, while those that belong in the second group are subsets of $\{b, c, d, e\}$. Hence, the maximal frequent itemsets $\{a,c,e\}$, $\{a,d\}$, and $\{b,c,d,e\}$ provide a compact representation of the frequent itemsets shown in Figure 6.16.

Maximal frequent itemsets provide a valuable representation for data sets that can produce very long, frequent itemsets, as there are exponentially many frequent itemsets in such data. Nevertheless, this approach is practical only if an efficient algorithm exists to explicitly find the maximal frequent itemsets without having to enumerate all their subsets. We briefly describe one such approach in Section 6.5.

Despite providing a compact representation, maximal frequent itemsets do not contain the support information of their subsets. For example, the support of the maximal frequent itemsets $\{a,c,e\}$, $\{a,d\}$, and $\{b,c,d,e\}$ do not provide any hint about the support of their subsets. An additional pass over the data set is therefore needed to determine the support counts of the non-maximal frequent itemsets. In some cases, it might be desirable to have a minimal representation of frequent itemsets that preserves the support information. We illustrate such a representation in the next section.

6.4.2 Closed Frequent Itemsets

Closed itemsets provide a minimal representation of itemsets without losing their support information. A formal definition of a closed itemset is presented below.

Definition 6.4 (Closed Itemset). An itemset X is closed if none of its immediate supersets has exactly the same support count as X.

Put another way, X is not closed if at least one of its immediate supersets has the same support count as X. Examples of closed itemsets are shown in Figure 6.17. To better illustrate the support count of each itemset, we have associated each node (itemset) in the lattice with a list of its corresponding

```
Notice: A is non-closed iff for some item c not in A,
the rule A->c has confidence = 100%.
Indeed, in that case support(A)=support(Ac).
```
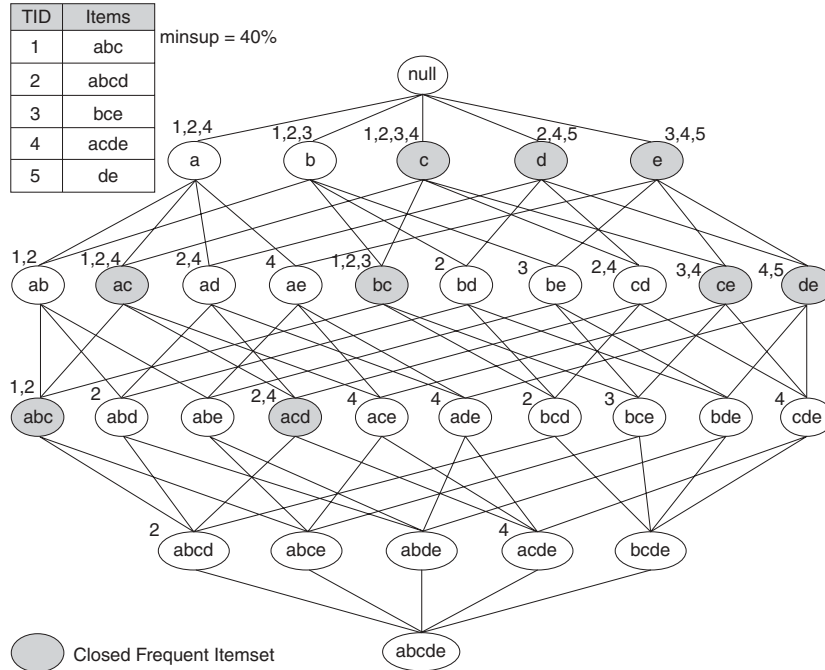
Figure 6.17. An example of the closed frequent itemsets (with minimum support count equal to 40%).

transaction IDs. For example, since the node $\{b, c\}$ is associated with transaction IDs 1, 2, and 3, its support count is equal to three. From the transactions given in this diagram, notice that every transaction that contains b also contains c. Consequently, the support for $\{b\}$ is identical to $\{b, c\}$ and $\{b\}$ should not be considered a closed itemset. Similarly, since c occurs in every transaction that contains both a and d, the itemset $\{a, d\}$ is not closed. On the other hand, $\{b, c\}$ is a closed itemset because it does not have the same support count as any of its supersets.

Definition 6.5 (Closed Frequent Itemset). An itemset is a closed frequent itemset if it is closed and its support is greater than or equal to *minsup*.

In the previous example, assuming that the support threshold is 40%, {b,c} is a closed frequent itemset because its support is 60%. The rest of the closed frequent itemsets are indicated by the shaded nodes.

Algorithms are available to explicitly extract closed frequent itemsets from a given data set. Interested readers may refer to the bibliographic notes at the end of this chapter for further discussions of these algorithms. We can use the closed frequent itemsets to determine the support counts for the non-closed

---

Algorithm 6.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets
2: Let $k_{\max}$ denote the maximum size of closed frequent itemsets
3: $F_{k_{\max}} = \{f | f \in C, \ |f| = k_{\max}\}$ {Find all frequent itemsets of size $k_{\max}$.}
4: **for** $k = k_{\max} - 1$ downto 1 **do**
5:    $F_k = \{f | f \subset F_{k+1}, \ |f| = k\}$ {Find all frequent itemsets of size k.}
6:    **for** each $f \in F_k$ **do**
7:       **if** $f \notin C$ **then**
8:          $f.support = \max\{f'.support | f' \in F_{k+1}, \ f \subset f'\}$
9:       **end if**
10:    **end for**
11: **end for**

---

frequent itemsets. For example, consider the frequent itemset $\{a, d\}$ shown in Figure 6.17. Because the itemset is not closed, its support count must be identical to one of its immediate supersets. The key is to determine which superset (among $\{a, b, d\}$, $\{a, c, d\}$, or $\{a, d, e\}$) has exactly the same support count as $\{a, d\}$. The *Apriori* principle states that any transaction that contains the superset of $\{a, d\}$ must also contain $\{a, d\}$. However, any transaction that contains $\{a, d\}$ does not have to contain the supersets of $\{a, d\}$. For this reason, the support for $\{a, d\}$ must be equal to the largest support among its supersets. Since $\{a, c, d\}$ has a larger support than both $\{a, b, d\}$ and $\{a, d, e\}$, the support for $\{a, d\}$ must be identical to the support for $\{a, c, d\}$. Using this methodology, an algorithm can be developed to compute the support for the non-closed frequent itemsets. The pseudocode for this algorithm is shown in Algorithm 6.4. The algorithm proceeds in a specific-to-general fashion, i.e., from the largest to the smallest frequent itemsets. This is because, in order to find the support for a non-closed frequent itemset, the support for all of its supersets must be known.

```
support(ad)
>= support(abd)
>= support(acd)
>= support(ade)
```

To illustrate the advantage of using closed frequent itemsets, consider the data set shown in Table 6.5, which contains ten transactions and fifteen items. The items can be divided into three groups: (1) Group A, which contains items a_1 through a_5; (2) Group B, which contains items b_1 through b_5; and (3) Group C, which contains items c_1 through c_5. Note that items within each group are perfectly associated with each other and they do not appear with items from another group. Assuming the support threshold is 20%, the total number of frequent itemsets is $3 \times (2^5 - 1) = 93$. However, there are only three closed frequent itemsets in the data: ($\{a_1, a_2, a_3, a_4, a_5\}$, $\{b_1, b_2, b_3, b_4, b_5\}$, and $\{c_1, c_2, c_3, c_4, c_5\}$). It is often sufficient to present only the closed frequent itemsets to the analysts instead of the entire set of frequent itemsets.

There are only three closed itemsets.

Table 6.5. A transaction data set for mining closed itemsets.

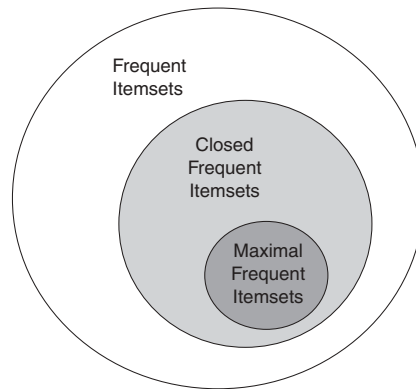| TID	a_1	a_2	a_3	a_4	a_5	b_1	b_2	b_3	b_4	b_5	c_1	c_2	c_3	c_4	c_5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 6.18. Relationships among frequent, maximal frequent, and closed frequent itemsets.

Closed frequent itemsets are useful for removing some of the redundant association rules. An association rule $X \longrightarrow Y$ is redundant if there exists another rule $X' \longrightarrow Y'$, where X is a subset of X' and Y is a subset of Y', such that the support and confidence for both rules are identical. In the example shown in Figure 6.17, $\{b\}$ is not a closed frequent itemset while $\{b, c\}$ is closed. The association rule $\{b\} \longrightarrow \{d, e\}$ is therefore redundant because it has the same support and confidence as $\{b, c\} \longrightarrow \{d, e\}$. Such redundant rules are not generated if closed frequent itemsets are used for rule generation.

Finally, note that all maximal frequent itemsets are closed because none of the maximal frequent itemsets can have the same support count as their immediate supersets. The relationships among frequent, maximal frequent, and closed frequent itemsets are shown in Figure 6.18.

6.5 Alternative Methods for Generating Frequent Itemsets

Apriori is one of the earliest algorithms to have successfully addressed the combinatorial explosion of frequent itemset generation. It achieves this by applying the *Apriori* principle to prune the exponential search space. Despite its significant performance improvement, the algorithm still incurs considerable I/O overhead since it requires making several passes over the transaction data set. In addition, as noted in Section 6.2.5, the performance of the *Apriori* algorithm may degrade significantly for dense data sets because of the increasing width of transactions. Several alternative methods have been developed to overcome these limitations and improve upon the efficiency of the *Apriori* algorithm. The following is a high-level description of these methods.

Traversal of Itemset Lattice   A search for frequent itemsets can be conceptually viewed as a traversal on the itemset lattice shown in Figure 6.1. The search strategy employed by an algorithm dictates how the lattice structure is traversed during the frequent itemset generation process. Some search strategies are better than others, depending on the configuration of frequent itemsets in the lattice. An overview of these strategies is presented next.

- **General-to-Specific versus Specific-to-General:**   The *Apriori* algorithm uses a general-to-specific search strategy, where pairs of frequent $(k-1)$-itemsets are merged to obtain candidate k-itemsets. This general-to-specific search strategy is effective, provided the maximum length of a frequent itemset is not too long. The configuration of frequent itemsets that works best with this strategy is shown in Figure 6.19(a), where the darker nodes represent infrequent itemsets. Alternatively, a specific-to-general search strategy looks for more specific frequent itemsets first, before finding the more general frequent itemsets. This strategy is useful to discover maximal frequent itemsets in dense transactions, where the frequent itemset border is located near the bottom of the lattice, as shown in Figure 6.19(b). The *Apriori* principle can be applied to prune all subsets of maximal frequent itemsets. Specifically, if a candidate k-itemset is maximal frequent, we do not have to examine any of its subsets of size $k-1$. However, if the candidate k-itemset is infrequent, we need to check all of its $k-1$ subsets in the next iteration. Another approach is to combine both general-to-specific and specific-to-general search strategies. This bidirectional approach requires more space to
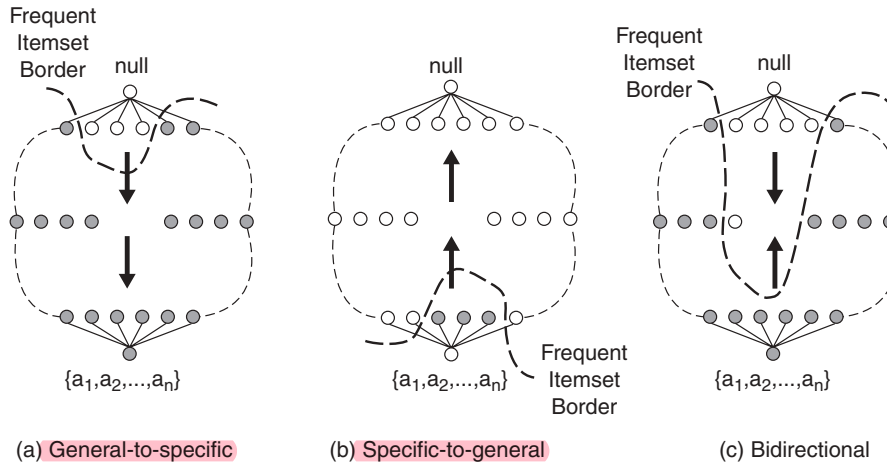
Figure 6.19. General-to-specific, specific-to-general, and bidirectional search.

store the candidate itemsets, but it can help to rapidly identify the frequent itemset border, given the configuration shown in Figure 6.19(c).

- **Equivalence Classes:** Another way to envision the traversal is to first partition the lattice into disjoint groups of nodes (or equivalence classes). A frequent itemset generation algorithm searches for frequent itemsets within a particular equivalence class first before moving to another equivalence class. As an example, the level-wise strategy used in the *Apriori* algorithm can be considered to be partitioning the lattice on the basis of itemset sizes; i.e., the algorithm discovers all frequent 1-itemsets first before proceeding to larger-sized itemsets. Equivalence classes can also be defined according to the prefix or suffix labels of an itemset. In this case, two itemsets belong to the same equivalence class if they share a common prefix or suffix of length k. In the prefix-based approach, the algorithm can search for frequent itemsets starting with the prefix a before looking for those starting with prefixes b, c, and so on. Both prefix-based and suffix-based equivalence classes can be demonstrated using the tree-like structure shown in Figure 6.20.

- **Breadth-First versus Depth-First:** The *Apriori* algorithm traverses the lattice in a breadth-first manner, as shown in Figure 6.21(a). It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated. The itemset

(a) Prefix tree.

(b) Suffix tree.
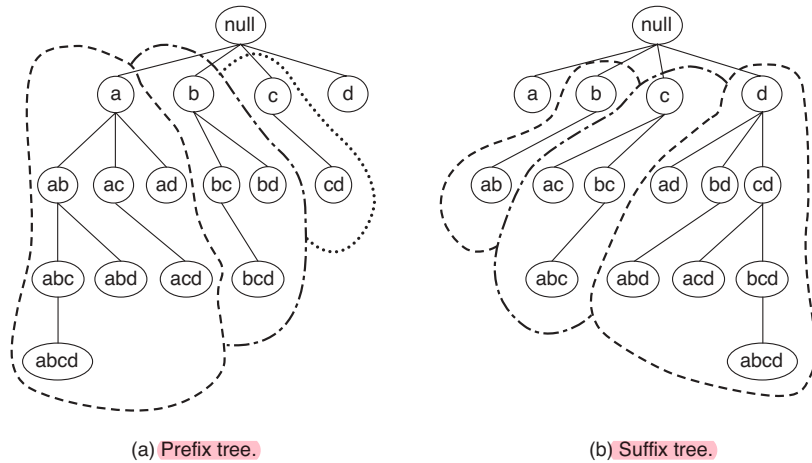
Figure 6.20. Equivalence classes based on the prefix and suffix labels of itemsets.

(a) Breadth first
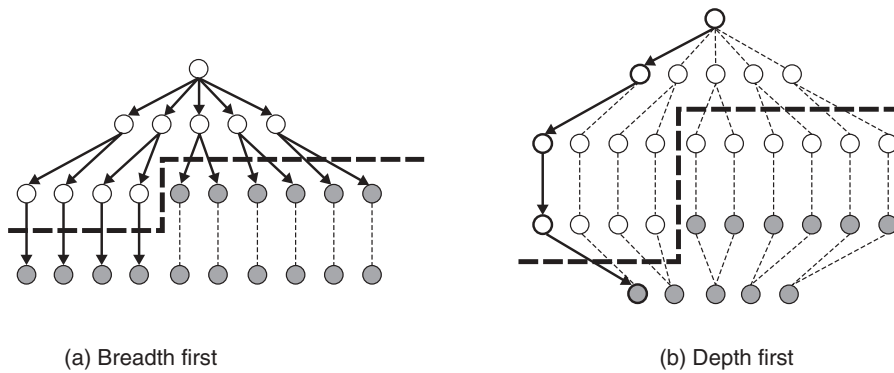
(b) Depth first

Figure 6.21. Breadth-first and depth-first traversals.

lattice can also be traversed in a depth-first manner, as shown in Figures 6.21(b) and 6.22. The algorithm can start from, say, node a in Figure 6.22, and count its support to determine whether it is frequent. If so, the algorithm progressively expands the next level of nodes, i.e., ab, abc, and so on, until an infrequent node is reached, say, $abcd$. It then backtracks to another branch, say, $abce$, and continues the search from there.

The depth-first approach is often used by algorithms designed to find maximal frequent itemsets. This approach allows the frequent itemset border to be detected more quickly than using a breadth-first approach. Once a maximal frequent itemset is found, substantial pruning can be
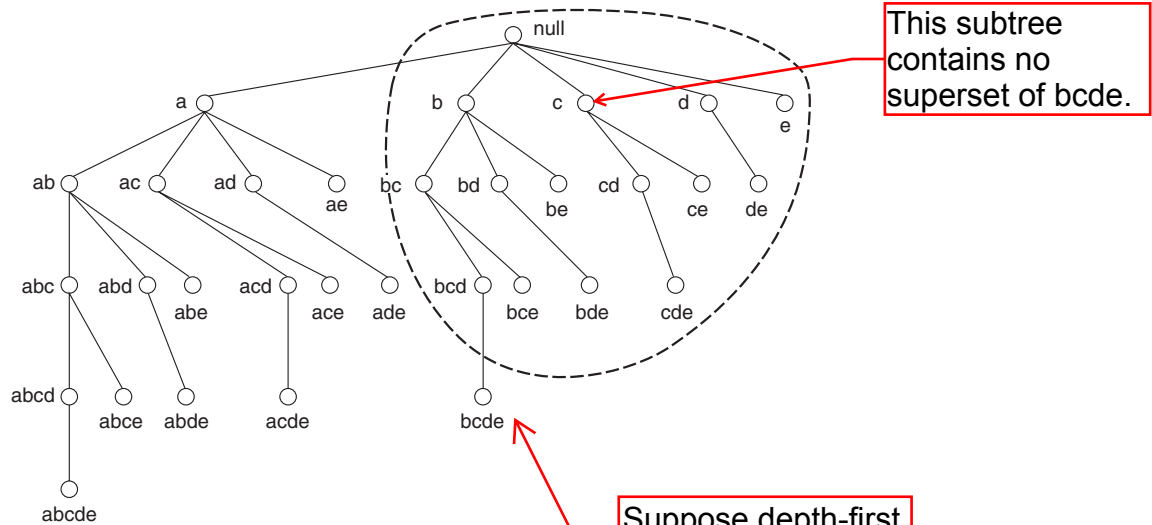
Figure 6.22. Generating candidate itemsets using the depth-first approach.

performed on its subsets. For example, if the node *bcde* shown in Figure 6.22 is maximal frequent, then the algorithm does not have to visit the subtrees rooted at *bd*, *be*, *c*, *d*, and *e* because they will not contain any maximal frequent itemsets. However, if *abc* is maximal frequent, only the nodes such as *ac* and *bc* are not maximal frequent (but the subtrees of *ac* and *bc* may still contain maximal frequent itemsets). The depth-first approach also allows a different kind of pruning based on the support of itemsets. For example, suppose the support for $\{a, b, c\}$ is identical to the support for $\{a, b\}$. The subtrees rooted at *abd* and *abe* can be skipped because they are guaranteed not to have any maximal frequent itemsets. The proof of this is left as an exercise to the readers.

Representation of Transaction Data Set   There are many ways to represent a transaction data set. The choice of representation can affect the I/O costs incurred when computing the support of candidate itemsets. Figure 6.23 shows two different ways of representing market basket transactions. The representation on the left is called a **horizontal** data layout, which is adopted by many association rule mining algorithms, including *Apriori*. Another possibility is to store the list of transaction identifiers (TID-list) associated with each item. Such a representation is known as the **vertical** data layout. The support for each candidate itemset is obtained by intersecting the TID-lists of its subset items. The length of the TID-lists shrinks as we progress to larger

Horizontal Data Layout

TID	Items
1	a,b,e
2	b,c,d
3	c,e
4	a,c,d
5	a,b,c,d
6	a,e
7	a,b
8	a,b,c
9	a,c,d
10	b

Vertical Data Layout

a	b	c	d	e
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

Figure 6.23. Horizontal and vertical data format.

sized itemsets. However, one problem with this approach is that the initial set of TID-lists may be too large to fit into main memory, thus requiring more sophisticated techniques to compress the TID-lists. We describe another effective approach to represent the data in the next section.

6.6 FP-Growth Algorithm

See the note eclat.pdf

This section presents an alternative algorithm called **FP-growth** that takes a radically different approach to discovering frequent itemsets. The algorithm does not subscribe to the generate-and-test paradigm of *Apriori*. Instead, it encodes the data set using a compact data structure called an **FP-tree** and extracts frequent itemsets directly from this structure. The details of this approach are presented next.

6.6.1 FP-Tree Representation

An FP-tree is a compressed representation of the input data. It is constructed by reading the data set one transaction at a time and mapping each transaction onto a path in the FP-tree. As different transactions can have several items in common, their paths may overlap. The more the paths overlap with one another, the more compression we can achieve using the FP-tree structure. If the size of the FP-tree is small enough to fit into main memory, this will allow us to extract frequent itemsets directly from the structure in memory instead of making repeated passes over the data stored on disk.
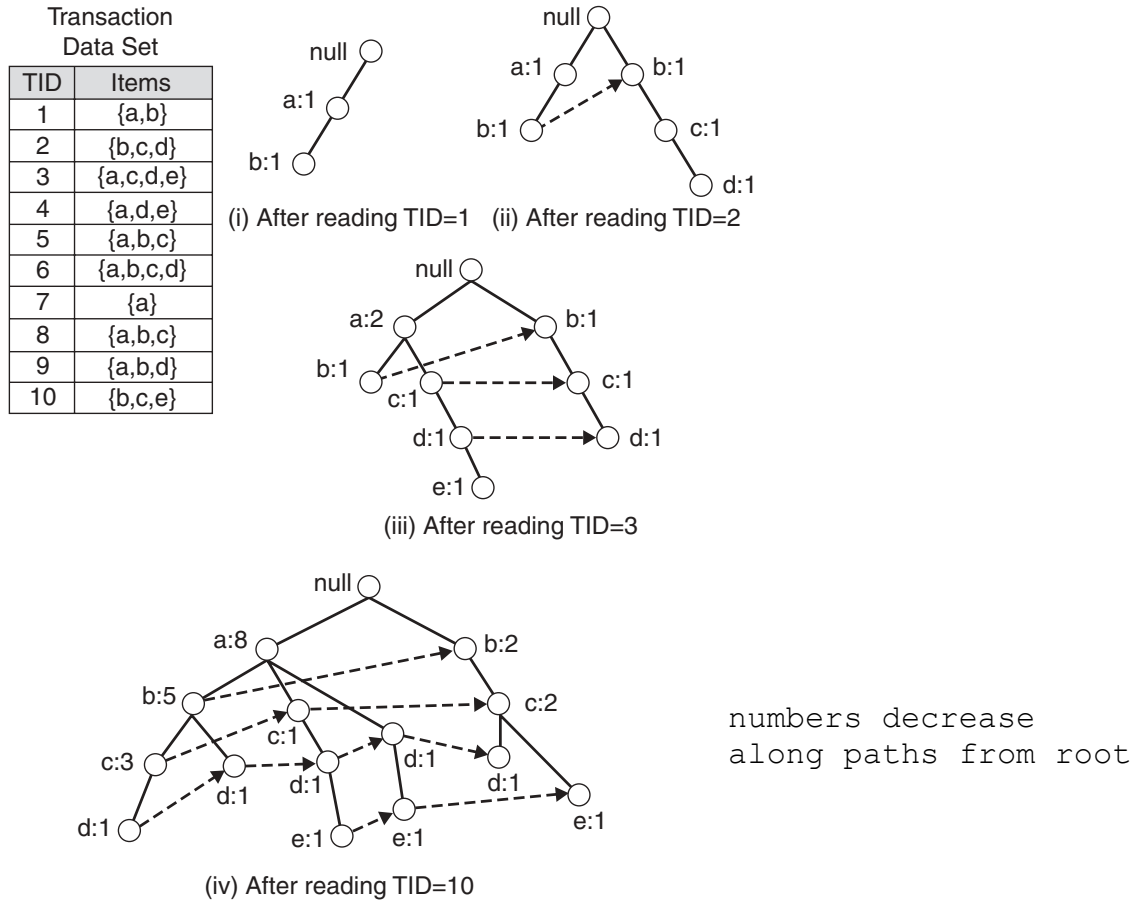
Figure 6.24. Construction of an FP-tree.

Figure 6.24 shows a data set that contains ten transactions and five items. The structures of the FP-tree after reading the first three transactions are also depicted in the diagram. Each node in the tree contains the label of an item along with a counter that shows the number of transactions mapped onto the given path. Initially, the FP-tree contains only the root node represented by the **null** symbol. The FP-tree is subsequently extended in the following way:

1. The data set is scanned once to determine the support count of each item. Infrequent items are discarded, while the frequent items are sorted in decreasing support counts. For the data set shown in Figure 6.24, a is the most frequent item, followed by b, c, d, and e.

   Thus, items are listed in decreasing support count.

2. The algorithm makes a second pass over the data to construct the FP-tree. After reading the first transaction, $\{a, b\}$, the nodes labeled as a and b are created. A path is then formed from $\mathtt{null} \rightarrow a \rightarrow b$ to encode the transaction. Every node along the path has a frequency count of 1.

3. After reading the second transaction, $\{b,c,d\}$, a new set of nodes is created for items b, c, and d. A path is then formed to represent the transaction by connecting the nodes $\mathtt{null} \rightarrow b \rightarrow c \rightarrow d$. Every node along this path also has a frequency count equal to one. Although the first two transactions have an item in common, which is b, their paths are disjoint because the transactions do not share a common prefix.

4. The third transaction, $\{a,c,d,e\}$, shares a common prefix item (which is a) with the first transaction. As a result, the path for the third transaction, $\mathtt{null} \rightarrow a \rightarrow c \rightarrow d \rightarrow e$, overlaps with the path for the first transaction, $\mathtt{null} \rightarrow a \rightarrow b$. Because of their overlapping path, the frequency count for node a is incremented to two, while the frequency counts for the newly created nodes, c, d, and e, are equal to one.

5. This process continues until every transaction has been mapped onto one of the paths given in the FP-tree. The resulting FP-tree after reading all the transactions is shown at the bottom of Figure 6.24.

The size of an FP-tree is typically smaller than the size of the uncompressed data because many transactions in market basket data often share a few items in common. In the best-case scenario, where all the transactions have the same set of items, the FP-tree contains only a single branch of nodes. The worst-case scenario happens when every transaction has a unique set of items. As none of the transactions have any items in common, the size of the FP-tree is effectively the same as the size of the original data. However, the physical storage requirement for the FP-tree is higher because it requires additional space to store pointers between nodes and counters for each item.

The size of an FP-tree also depends on how the items are ordered. If the ordering scheme in the preceding example is reversed, i.e., from lowest to highest support item, the resulting FP-tree is shown in Figure 6.25. The tree appears to be denser because the branching factor at the root node has increased from 2 to 5 and the number of nodes containing the high support items such as a and b has increased from 3 to 12. Nevertheless, ordering by decreasing support counts does not always lead to the smallest tree. For example, suppose we augment the data set given in Figure 6.24 with 100 transactions that contain $\{e\}$, 80 transactions that contain $\{d\}$, 60 transactions
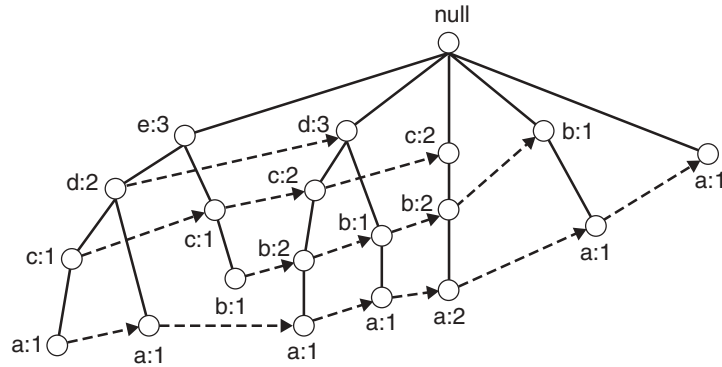
Figure 6.25. An FP-tree representation for the data set shown in Figure 6.24 with a different item ordering scheme.

that contain $\{c\}$, and 40 transactions that contain $\{b\}$. Item e is now most frequent, followed by d, c, b, and a. With the augmented transactions, ordering by decreasing support counts will result in an FP-tree similar to Figure 6.25, while a scheme based on increasing support counts produces a smaller FP-tree similar to Figure 6.24(iv).

An FP-tree also contains a list of pointers connecting between nodes that have the same items. These pointers, represented as dashed lines in Figures 6.24 and 6.25, help to facilitate the rapid access of individual items in the tree. We explain how to use the FP-tree and its corresponding pointers for frequent itemset generation in the next section.

6.6.2   Frequent Itemset Generation in FP-Growth Algorithm

FP-growth is an algorithm that generates frequent itemsets from an FP-tree by exploring the tree in a bottom-up fashion. Given the example tree shown in Figure 6.24, the algorithm looks for frequent itemsets ending in e first, followed by d, c, b, and finally, a. This bottom-up strategy for finding frequent itemsets ending with a particular item is equivalent to the suffix-based approach described in Section 6.5. Since every transaction is mapped onto a path in the FP-tree, we can derive the frequent itemsets ending with a particular item, say, e, by examining only the paths containing node e. These paths can be accessed rapidly using the pointers associated with node e. The extracted paths are shown in Figure 6.26(a). The details on how to process the paths to obtain frequent itemsets will be explained later.

See Fig. 6.20b

(a) Paths containing node e

(b) Paths containing node d

(c) Paths containing node c

(d) Paths containing node b

(e) Paths containing node a
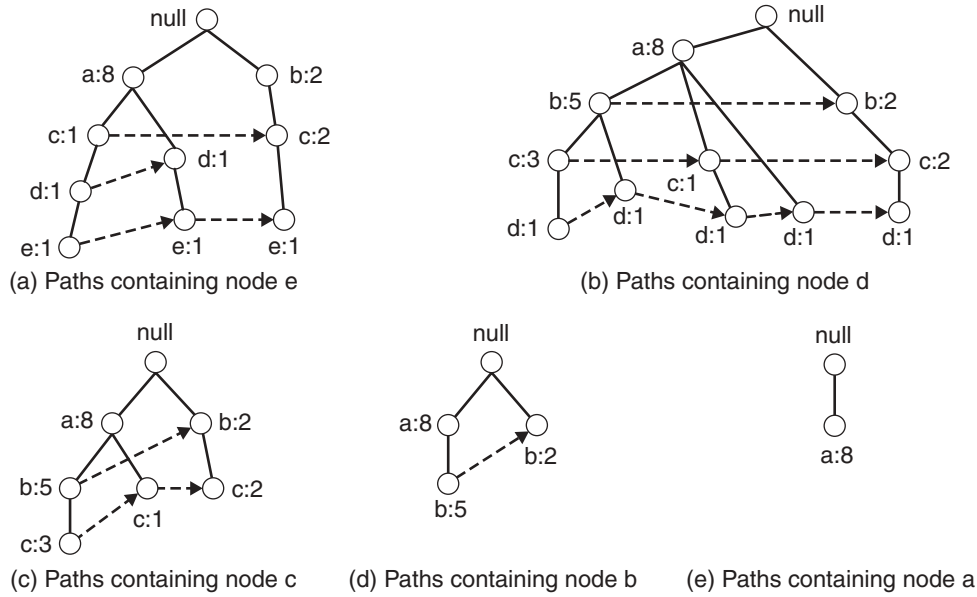
Figure 6.26. Decomposing the frequent itemset generation problem into multiple subproblems, where each subproblem involves finding frequent itemsets ending in e, d, c, b, and a.

Table 6.6. The list of frequent itemsets ordered by their corresponding suffixes.

Suffix	Frequent Itemsets
e	{e}, {d,e}, {a,d,e}, {c,e},{a,e}
d	{d}, {c,d}, {b,c,d}, {a,c,d}, {b,d}, {a,b,d}, {a,d}
c	{c}, {b,c}, {a,b,c}, {a,c}
b	{b}, {a,b}
a	{a}

After finding the frequent itemsets ending in e, the algorithm proceeds to look for frequent itemsets ending in d by processing the paths associated with node d. The corresponding paths are shown in Figure 6.26(b). This process continues until all the paths associated with nodes c, b, and finally a, are processed. The paths for these items are shown in Figures 6.26(c), (d), and (e), while their corresponding frequent itemsets are summarized in Table 6.6.

FP-growth finds all the frequent itemsets ending with a particular suffix by employing a divide-and-conquer strategy to split the problem into smaller subproblems. For example, suppose we are interested in finding all frequent
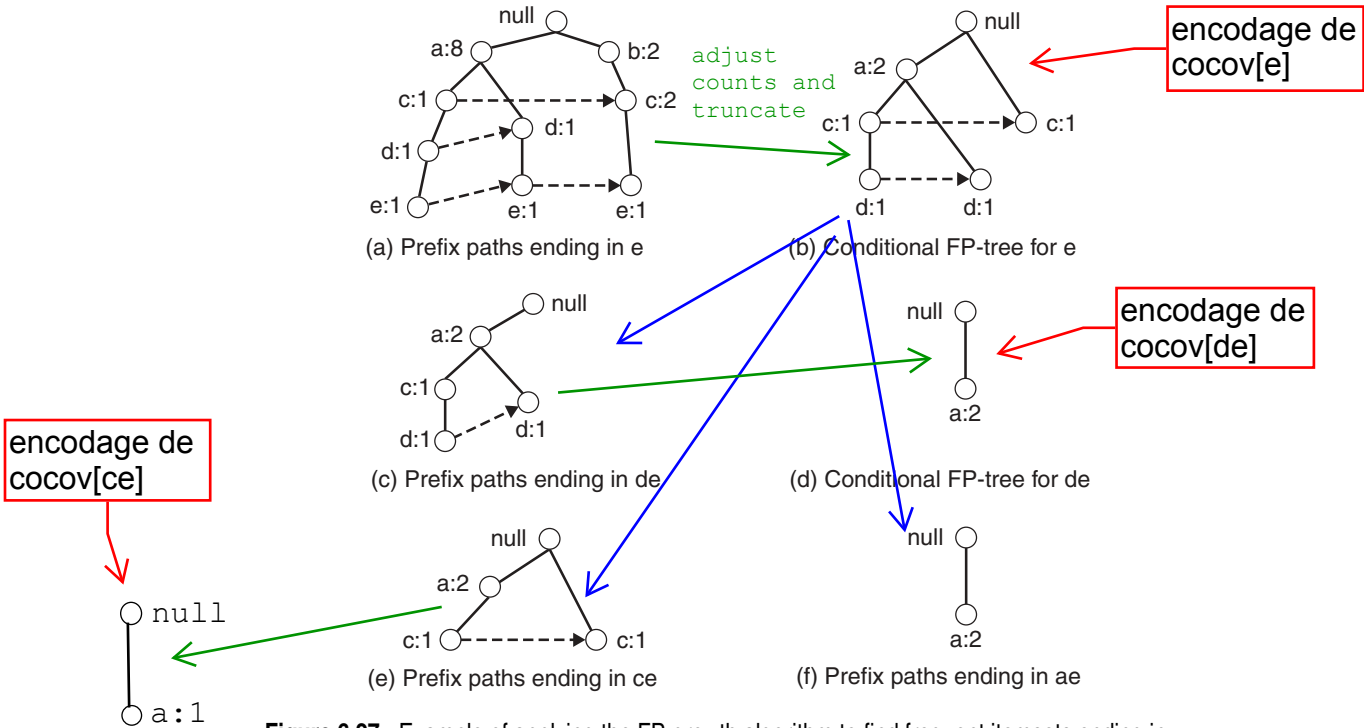
Figure 6.27. Example of applying the FP-growth algorithm to find frequent itemsets ending in e.

itemsets ending in e. To do this, we must first check whether the itemset $\{e\}$ itself is frequent. If it is frequent, we consider the subproblem of finding frequent itemsets ending in de, followed by ce, be, and ae. In turn, each of these subproblems are further decomposed into smaller subproblems. By merging the solutions obtained from the subproblems, all the frequent itemsets ending in e can be found. This divide-and-conquer approach is the key strategy employed by the FP-growth algorithm.

For a more concrete example on how to solve the subproblems, consider the task of finding frequent itemsets ending with e.

1. The first step is to gather all the paths containing node e. These initial paths are called **prefix paths** and are shown in Figure 6.27(a).

2. From the prefix paths shown in Figure 6.27(a), the support count for e is obtained by adding the support counts associated with node e. Assuming that the minimum support count is 2, $\{e\}$ is declared a frequent itemset because its support count is 3.

3. Because $\{e\}$ is frequent, the algorithm has to solve the subproblems of finding frequent itemsets ending in de, ce, be, and ae. Before solving these subproblems, it must first convert the prefix paths into a **conditional FP-tree**, which is structurally similar to an FP-tree, except it is used to find frequent itemsets ending with a particular suffix. A conditional FP-tree is obtained in the following way:

   (a) First, the support counts along the prefix paths must be updated because some of the counts include transactions that do not contain item e. For example, the rightmost path shown in Figure 6.27(a), `null` $\longrightarrow$ `b:2` $\longrightarrow$ `c:2` $\longrightarrow$ `e:1`, includes a transaction $\{b, c\}$ that does not contain item e. The counts along the prefix path must therefore be adjusted to 1 to reflect the actual number of transactions containing $\{b, c, e\}$.

   (b) The prefix paths are truncated by removing the nodes for e. These nodes can be removed because the support counts along the prefix paths have been updated to reflect only transactions that contain e and the subproblems of finding frequent itemsets ending in de, ce, be, and ae no longer need information about node e.

   (c) After updating the support counts along the prefix paths, some of the items may no longer be frequent. For example, the node b appears only once and has a support count equal to 1, which means that there is only one transaction that contains both b and e. Item b can be safely ignored from subsequent analysis because all itemsets ending in be must be infrequent.

   The conditional FP-tree for e is shown in Figure 6.27(b). The tree looks different than the original prefix paths because the frequency counts have been updated and the nodes b and e have been eliminated.

4. FP-growth uses the conditional FP-tree for e to solve the subproblems of finding frequent itemsets ending in de, ce, and ae. To find the frequent itemsets ending in de, the prefix paths for d are gathered from the conditional FP-tree for e (Figure 6.27(c)). By adding the frequency counts associated with node d, we obtain the support count for $\{d, e\}$. Since the support count is equal to 2, $\{d, e\}$ is declared a frequent itemset. Next, the algorithm constructs the conditional FP-tree for de using the approach described in step 3. After updating the support counts and removing the infrequent item c, the conditional FP-tree for de is shown in Figure 6.27(d). Since the conditional FP-tree contains only one item,

a, whose support is equal to *minsup*, the algorithm extracts the frequent itemset $\{a, d, e\}$ and moves on to the next subproblem, which is to generate frequent itemsets ending in *ce*. After processing the prefix paths for *c*, only $\{c, e\}$ is found to be frequent. The algorithm proceeds to solve the next subprogram and found $\{a, e\}$ to be the only frequent itemset remaining.

This example illustrates the divide-and-conquer approach used in the FP-growth algorithm. At each recursive step, a conditional FP-tree is constructed by updating the frequency counts along the prefix paths and removing all infrequent items. Because the subproblems are disjoint, FP-growth will not generate any duplicate itemsets. In addition, the counts associated with the nodes allow the algorithm to perform support counting while generating the common suffix itemsets.

FP-growth is an interesting algorithm because it illustrates how a compact representation of the transaction data set helps to efficiently generate frequent itemsets. In addition, for certain transaction data sets, FP-growth outperforms the standard *Apriori* algorithm by several orders of magnitude. The run-time performance of FP-growth depends on the **compaction factor** of the data set. If the resulting conditional FP-trees are very bushy (in the worst case, a full prefix tree), then the performance of the algorithm degrades significantly because it has to generate a large number of subproblems and merge the results returned by each subproblem.

6.7 Evaluation of Association Patterns

Association analysis algorithms have the potential to generate a large number of patterns. For example, although the data set shown in Table 6.1 contains only six items, it can produce up to hundreds of association rules at certain support and confidence thresholds. As the size and dimensionality of real commercial databases can be very large, we could easily end up with thousands or even millions of patterns, many of which might not be interesting. Sifting through the patterns to identify the most interesting ones is not a trivial task because "one person's trash might be another person's treasure." It is therefore important to establish a set of well-accepted criteria for evaluating the quality of association patterns.

The first set of criteria can be established through statistical arguments. Patterns that involve a set of mutually independent items or cover very few transactions are considered uninteresting because they may capture spurious relationships in the data. Such patterns can be eliminated by applying an

objective interestingness measure that uses statistics derived from data to determine whether a pattern is interesting. Examples of objective interestingness measures include support, confidence, and correlation.

The second set of criteria can be established through subjective arguments. A pattern is considered subjectively uninteresting unless it reveals unexpected information about the data or provides useful knowledge that can lead to profitable actions. For example, the rule $\{Butter\} \longrightarrow \{Bread\}$ may not be interesting, despite having high support and confidence values, because the relationship represented by the rule may seem rather obvious. On the other hand, the rule $\{Diapers\} \longrightarrow \{Beer\}$ is interesting because the relationship is quite unexpected and may suggest a new cross-selling opportunity for retailers. Incorporating subjective knowledge into pattern evaluation is a difficult task because it requires a considerable amount of prior information from the domain experts.

The following are some of the approaches for incorporating subjective knowledge into the pattern discovery task.

Visualization   This approach requires a user-friendly environment to keep the human user in the loop. It also allows the domain experts to interact with the data mining system by interpreting and verifying the discovered patterns.

Template-based approach   This approach allows the users to constrain the type of patterns extracted by the mining algorithm. Instead of reporting all the extracted rules, only rules that satisfy a user-specified template are returned to the users.

Subjective interestingness measure   A subjective measure can be defined based on domain information such as concept hierarchy (to be discussed in Section 7.3) or profit margin of items. The measure can then be used to filter patterns that are obvious and non-actionable.

Readers interested in subjective interestingness measures may refer to resources listed in the bibliography at the end of this chapter.

6.7.1   Objective Measures of Interestingness

An objective measure is a data-driven approach for evaluating the quality of association patterns. It is domain-independent and requires minimal input from the users, other than to specify a threshold for filtering low-quality patterns. An objective measure is usually computed based on the frequency

Table 6.7. A 2-way contingency table for variables A and B.

	B	$\overline{B}$	
A	f_{11}	f_{10}	f_{1+}
$\overline{A}$	f_{01}	f_{00}	f_{0+}
	f_{+1}	f_{+0}	N

counts tabulated in a **contingency table**. Table 6.7 shows an example of a contingency table for a pair of binary variables, A and B. We use the notation $\overline{A}$ $(\overline{B})$ to indicate that A (B) is absent from a transaction. Each entry f_{ij} in this 2×2 table denotes a frequency count. For example, f_{11} is the number of times A and B appear together in the same transaction, while f_{01} is the number of transactions that contain B but not A. The row sum f_{1+} represents the support count for A, while the column sum f_{+1} represents the support count for B. Finally, even though our discussion focuses mainly on asymmetric binary variables, note that contingency tables are also applicable to other attribute types such as symmetric binary, nominal, and ordinal variables.

Limitations of the Support-Confidence Framework Existing association rule mining formulation relies on the support and confidence measures to eliminate uninteresting patterns. The drawback of support was previously described in Section 6.8, in which many potentially interesting patterns involving low support items might be eliminated by the support threshold. The drawback of confidence is more subtle and is best demonstrated with the following example.

Example 6.3. Suppose we are interested in analyzing the relationship between people who drink tea and coffee. We may gather information about the beverage preferences among a group of people and summarize their responses into a table such as the one shown in Table 6.8.

Table 6.8. Beverage preferences among a group of 1000 people.

	$Coffee$	$\overline{Coffee}$	
Tea	150	50	200
$\overline{Tea}$	650	150	800
	800	200	1000

The information given in this table can be used to evaluate the association rule $\{Tea\} \longrightarrow \{Coffee\}$. At first glance, it may appear that people who drink tea also tend to drink coffee because the rule's support (15%) and confidence (75%) values are reasonably high. This argument would have been acceptable except that the fraction of people who drink coffee, regardless of whether they drink tea, is 80%, while the fraction of tea drinkers who drink coffee is only 75%. Thus knowing that a person is a tea drinker actually decreases her probability of being a coffee drinker from 80% to 75%! The rule $\{Tea\} \longrightarrow \{Coffee\}$ is therefore misleading despite its high confidence value. ∎

The pitfall of confidence can be traced to the fact that the measure ignores the support of the itemset in the rule consequent. Indeed, if the support of coffee drinkers is taken into account, we would not be surprised to find that many of the people who drink tea also drink coffee. What is more surprising is that the fraction of tea drinkers who drink coffee is actually less than the overall fraction of people who drink coffee, which points to an inverse relationship between tea drinkers and coffee drinkers.

Because of the limitations in the support-confidence framework, various objective measures have been used to evaluate the quality of association patterns. Below, we provide a brief description of these measures and explain some of their strengths and limitations.

Interest Factor   The tea-coffee example shows that high-confidence rules can sometimes be misleading because the confidence measure ignores the support of the itemset appearing in the rule consequent. One way to address this problem is by applying a metric known as **lift**:

$$Lift = \frac{c(A \longrightarrow B)}{s(B)},\qquad\qquad (6.4)$$

which computes the ratio between the rule's confidence and the support of the itemset in the rule consequent. For binary variables, lift is equivalent to another objective measure called **interest factor**, which is defined as follows:

$$I(A, B) = \frac{s(A, B)}{s(A) \times s(B)} = \frac{N f_{11}}{f_{1+} f_{+1}}.\qquad\qquad (6.5)$$

Interest factor compares the frequency of a pattern against a baseline frequency computed under the statistical independence assumption. The baseline frequency for a pair of mutually independent variables is

$$\frac{f_{11}}{N} = \frac{f_{1+}}{N} \times \frac{f_{+1}}{N}, \quad \text{or equivalently,} \quad f_{11} = \frac{f_{1+} f_{+1}}{N}.\qquad (6.6)$$

Table 6.9. Contingency tables for the word pairs ({p,q} and {r,s}.

	p	$\bar{p}$	
q	880	50	930
$\bar{q}$	50	20	70
	930	70	1000

	r	$\bar{r}$	
s	20	50	70
$\bar{s}$	50	880	930
	70	930	1000

This equation follows from the standard approach of using simple fractions as estimates for probabilities. The fraction f_{11}/N is an estimate for the joint probability $P(A, B)$, while f_{1+}/N and f_{+1}/N are the estimates for $P(A)$ and $P(B)$, respectively. If A and B are statistically independent, then $P(A, B) = P(A) \times P(B)$, thus leading to the formula shown in Equation 6.6. Using Equations 6.5 and 6.6, we can interpret the measure as follows:

$$I(A, B) \begin{cases} = 1, & \text{if } A \text{ and } B \text{ are independent;} \\ > 1, & \text{if } A \text{ and } B \text{ are positively correlated;} \\ < 1, & \text{if } A \text{ and } B \text{ are negatively correlated.} \end{cases} \quad (6.7)$$

For the tea-coffee example shown in Table 6.8, $I = \frac{0.15}{0.2 \times 0.8} = 0.9375$, thus suggesting a slight negative correlation between tea drinkers and coffee drinkers.

Limitations of Interest Factor We illustrate the limitation of interest factor with an example from the text mining domain. In the text domain, it is reasonable to assume that the association between a pair of words depends on the number of documents that contain both words. For example, because of their stronger association, we expect the words `data` and `mining` to appear together more frequently than the words `compiler` and `mining` in a collection of computer science articles.

Table 6.9 shows the frequency of occurrences between two pairs of words, {p, q} and {r, s}. Using the formula given in Equation 6.5, the interest factor for {p, q} is 1.02 and for {r, s} is 4.08. These results are somewhat troubling for the following reasons. Although p and q appear together in 88% of the documents, their interest factor is close to 1, which is the value when p and q are statistically independent. On the other hand, the interest factor for {r, s} is higher than {p, q} even though r and s seldom appear together in the same document. Confidence is perhaps the better choice in this situation because it considers the association between p and q (94.6%) to be much stronger than that between r and s (28.6%).

Correlation Analysis   Correlation analysis is a statistical-based technique for analyzing relationships between a pair of variables. For continuous variables, correlation is defined using Pearson's correlation coefficient (see Equation 2.10 on page 77). For binary variables, correlation can be measured using the ϕ-coefficient, which is defined as

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}. \tag{6.8}$$

The value of correlation ranges from -1 (perfect negative correlation) to $+1$ (perfect positive correlation). If the variables are statistically independent, then $\phi = 0$. For example, the correlation between the tea and coffee drinkers given in Table 6.8 is -0.0625.

Limitations of Correlation Analysis   The drawback of using correlation can be seen from the word association example given in Table 6.9. Although the words p and q appear together more often than r and s, their ϕ-coefficients are identical, i.e., $\phi(p, q) = \phi(r, s) = 0.232$. This is because the ϕ-coefficient gives equal importance to both co-presence and co-absence of items in a transaction. It is therefore more suitable for analyzing symmetric binary variables. Another limitation of this measure is that it does not remain invariant when there are proportional changes to the sample size. This issue will be discussed in greater detail when we describe the properties of objective measures on page 377.

IS Measure IS is an alternative measure that has been proposed for handling asymmetric binary variables. The measure is defined as follows:

$$IS(A, B) = \sqrt{I(A, B) \times s(A, B)} = \frac{s(A, B)}{\sqrt{s(A)s(B)}}. \tag{6.9}$$

Note that IS is large when the interest factor and support of the pattern are large. For example, the value of IS for the word pairs $\{p, q\}$ and $\{r, s\}$ shown in Table 6.9 are 0.946 and 0.286, respectively. Contrary to the results given by interest factor and the ϕ-coefficient, the IS measure suggests that the association between $\{p, q\}$ is stronger than $\{r, s\}$, which agrees with what we expect from word associations in documents.

It is possible to show that IS is mathematically equivalent to the cosine measure for binary variables (see Equation 2.7 on page 75). In this regard, we

Table 6.10. Example of a contingency table for items p and q.

	q	$\overline{q}$	
p	800	100	900
$\overline{p}$	100	0	100
	900	100	1000

consider **A** and **B** as a pair of bit vectors, $\mathbf{A} \bullet \mathbf{B} = s(A, B)$ the dot product between the vectors, and $|\mathbf{A}| = \sqrt{s(A)}$ the magnitude of vector **A**. Therefore:

$$IS(A, B) = \frac{s(A, B)}{\sqrt{s(A) \times s(B)}} = \frac{\mathbf{A} \bullet \mathbf{B}}{|\mathbf{A}| \times |\mathbf{B}|} = cosine(\mathbf{A}, \mathbf{B}). \qquad (6.10)$$

The IS measure can also be expressed as the geometric mean between the confidence of association rules extracted from a pair of binary variables:

$$IS(A, B) = \sqrt{\frac{s(A, B)}{s(A)} \times \frac{s(A, B)}{s(B)}} = \sqrt{c(A \rightarrow B) \times c(B \rightarrow A)}. \qquad (6.11)$$

Because the geometric mean between any two numbers is always closer to the smaller number, the IS value of an itemset $\{p, q\}$ is low whenever one of its rules, $p \longrightarrow q$ or $q \longrightarrow p$, has low confidence.

Limitations of IS Measure The IS value for a pair of independent itemsets, A and B, is

$$IS_{\text{indep}}(A, B) = \frac{s(A, B)}{\sqrt{s(A) \times s(B)}} = \frac{s(A) \times s(B)}{\sqrt{s(A) \times s(B)}} = \sqrt{s(A) \times s(B)}.$$

Since the value depends on $s(A)$ and $s(B)$, IS shares a similar problem as the confidence measure—that the value of the measure can be quite large, even for uncorrelated and negatively correlated patterns. For example, despite the large IS value between items p and q given in Table 6.10 (0.889), it is still less than the expected value when the items are statistically independent ($IS_{\text{indep}} = 0.9$).

Alternative Objective Interestingness Measures

Besides the measures we have described so far, there are other alternative measures proposed for analyzing relationships between pairs of binary variables. These measures can be divided into two categories, **symmetric** and **asymmetric** measures. A measure M is symmetric if $M(A \longrightarrow B) = M(B \longrightarrow A)$. For example, interest factor is a symmetric measure because its value is identical for the rules $A \longrightarrow B$ and $B \longrightarrow A$. In contrast, confidence is an asymmetric measure since the confidence for $A \longrightarrow B$ and $B \longrightarrow A$ may not be the same. Symmetric measures are generally used for evaluating itemsets, while asymmetric measures are more suitable for analyzing association rules. Tables 6.11 and 6.12 provide the definitions for some of these measures in terms of the frequency counts of a 2×2 contingency table.

Consistency among Objective Measures

Given the wide variety of measures available, it is reasonable to question whether the measures can produce similar ordering results when applied to a set of association patterns. If the measures are consistent, then we can choose any one of them as our evaluation metric. Otherwise, it is important to understand what their differences are in order to determine which measure is more suitable for analyzing certain types of patterns.

Table 6.11. Examples of symmetric objective measures for the itemset $\{A, B\}$.

Measure (Symbol)	Definition
Correlation (ϕ)	$\dfrac{Nf_{11} - f_{1+}f_{+1}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}$
Odds ratio (α)	$\left(f_{11}f_{00}\right)/\left(f_{10}f_{01}\right)$
Kappa (κ)	$\dfrac{Nf_{11} + Nf_{00} - f_{1+}f_{+1} - f_{0+}f_{+0}}{N^2 - f_{1+}f_{+1} - f_{0+}f_{+0}}$
Interest (I)	$\left(Nf_{11}\right)/\left(f_{1+}f_{+1}\right)$
Cosine (IS)	$\left(f_{11}\right)/\left(\sqrt{f_{1+}f_{+1}}\right)$
Piatetsky-Shapiro (PS)	$\dfrac{f_{11}}{N} - \dfrac{f_{1+}f_{+1}}{N^2}$
Collective strength (S)	$\dfrac{f_{11} + f_{00}}{f_{1+}f_{+1} + f_{0+}f_{+0}} \times \dfrac{N - f_{1+}f_{+1} - f_{0+}f_{+0}}{N - f_{11} - f_{00}}$
Jaccard (ζ)	$f_{11}/\left(f_{1+} + f_{+1} - f_{11}\right)$
All-confidence (h)	$\min\left[\dfrac{f_{11}}{f_{1+}}, \dfrac{f_{11}}{f_{+1}}\right]$

Table 6.12. Examples of asymmetric objective measures for the rule $A \longrightarrow B$.

Measure (Symbol)	Definition
Goodman-Kruskal (λ)	$\left(\sum_j \max_k f_{jk} - max_k f_{+k}\right)/\left(N - \max_k f_{+k}\right)$
Mutual Information (M)	$\left(\sum_i \sum_j \frac{f_{ij}}{N} \log \frac{N f_{ij}}{f_{i+} f_{+j}}\right)/\left(-\sum_i \frac{f_{i+}}{N} \log \frac{f_{i+}}{N}\right)$
J-Measure (J)	$\frac{f_{11}}{N} \log \frac{N f_{11}}{f_{1+} f_{+1}} + \frac{f_{10}}{N} \log \frac{N f_{10}}{f_{1+} f_{+0}}$
Gini index (G)	$\frac{f_{1+}}{N} \times \left(\frac{f_{11}}{f_{1+}}\right)^2 + \left(\frac{f_{10}}{f_{1+}}\right)^2] - \left(\frac{f_{+1}}{N}\right)^2$
	$+ \frac{f_{0+}}{N} \times [\left(\frac{f_{01}}{f_{0+}}\right)^2 + \left(\frac{f_{00}}{f_{0+}}\right)^2] - \left(\frac{f_{+0}}{N}\right)^2$
Laplace (L)	$\left(f_{11} + 1\right)/\left(f_{1+} + 2\right)$
Conviction (V)	$\left(f_{1+} f_{+0}\right)/\left(N f_{10}\right)$
Certainty factor (F)	$\left(\frac{f_{11}}{f_{1+}} - \frac{f_{+1}}{N}\right)/\left(1 - \frac{f_{+1}}{N}\right)$
Added Value (AV)	$\frac{f_{11}}{f_{1+}} - \frac{f_{+1}}{N}$

Table 6.13. Example of contingency tables.

Example	f_{11}	f_{10}	f_{01}	f_{00}
E_1	8123	83	424	1370
E_2	8330	2	622	1046
E_3	3954	3080	5	2961
E_4	2886	1363	1320	4431
E_5	1500	2000	500	6000
E_6	4000	2000	1000	3000
E_7	9481	298	127	94
E_8	4000	2000	2000	2000
E_9	7450	2483	4	63
E_{10}	61	2483	4	7452

Suppose the symmetric and asymmetric measures are applied to rank the ten contingency tables shown in Table 6.13. These contingency tables are chosen to illustrate the differences among the existing measures. The ordering produced by these measures are shown in Tables 6.14 and 6.15, respectively (with 1 as the most interesting and 10 as the least interesting table). Although some of the measures appear to be consistent with each other, there are certain measures that produce quite different ordering results. For example, the rankings given by the ϕ-coefficient agree with those provided by κ and collective strength, but are somewhat different than the rankings produced by interest

Table 6.14. Rankings of contingency tables using the symmetric measures given in Table 6.11.

	ϕ	α	κ	I	IS	PS	S	ζ	h
E_1	1	3	1	6	2	2	1	2	2
E_2	2	1	2	7	3	5	2	3	3
E_3	3	2	4	4	5	1	3	6	8
E_4	4	8	3	3	7	3	4	7	5
E_5	5	7	6	2	9	6	6	9	9
E_6	6	9	5	5	6	4	5	5	7
E_7	7	6	7	9	1	8	7	1	1
E_8	8	10	8	8	8	7	8	8	7
E_9	9	4	9	10	4	9	9	4	4
E_{10}	10	5	10	1	10	10	10	10	10

Table 6.15. Rankings of contingency tables using the asymmetric measures given in Table 6.12.

	λ	M	J	G	L	V	F	AV
E_1	1	1	1	1	4	2	2	5
E_2	2	2	2	3	5	1	1	6
E_3	5	3	5	2	2	6	6	4
E_4	4	6	3	4	9	3	3	1
E_5	9	7	4	6	8	5	5	2
E_6	3	8	6	5	7	4	4	3
E_7	7	5	9	8	3	7	7	9
E_8	8	9	7	7	10	8	8	7
E_9	6	4	10	9	1	9	9	10
E_{10}	10	10	8	10	6	10	10	8

factor and odds ratio. Furthermore, a contingency table such as E_{10} is ranked lowest according to the ϕ-coefficient, but highest according to interest factor.

Properties of Objective Measures

The results shown in Table 6.14 suggest that a significant number of the measures provide conflicting information about the quality of a pattern. To understand their differences, we need to examine the properties of these measures.

Inversion Property Consider the bit vectors shown in Figure 6.28. The 0/1 bit in each column vector indicates whether a transaction (row) contains a particular item (column). For example, the vector **A** indicates that item a

A	B		C	D		E	F
1	0		0	1		0	0
0	0		1	1		1	0
0	0		1	1		1	0
0	0		1	1		1	0
0	1		1	0		1	1
0	0		1	1		1	0
0	0		1	1		1	0
0	0		1	1		1	0
0	0		1	1		1	0
1	0		0	1		0	0
(a)			(b)			(c)	

Figure 6.28. Effect of the inversion operation. The vectors C and E are inversions of vector A, while the vector D is an inversion of vectors B and F.

belongs to the first and last transactions, whereas the vector **B** indicates that item b is contained only in the fifth transaction. The vectors **C** and **E** are in fact related to the vector **A**—their bits have been inverted from 0's (absence) to 1's (presence), and vice versa. Similarly, **D** is related to vectors **B** and **F** by inverting their bits. The process of flipping a bit vector is called **inversion**. If a measure is invariant under the inversion operation, then its value for the vector pair $(\mathbf{C}, \mathbf{D})$ should be identical to its value for $(\mathbf{A}, \mathbf{B})$. The inversion property of a measure can be tested as follows.

Definition 6.6 (Inversion Property). An objective measure M is invariant under the inversion operation if its value remains the same when exchanging the frequency counts f_{11} with f_{00} and f_{10} with f_{01}.

Among the measures that remain invariant under this operation include the ϕ-coefficient, odds ratio, κ, and collective strength. These measures may not be suitable for analyzing asymmetric binary data. For example, the ϕ-coefficient between **C** and **D** is identical to the ϕ-coefficient between **A** and **B**, even though items c and d appear together more frequently than a and b. Furthermore, the ϕ-coefficient between **C** and **D** is less than that between **E** and **F** even though items e and f appear together only once! We had previously raised this issue when discussing the limitations of the ϕ-coefficient on page 375. For asymmetric binary data, measures that do not remain invariant under the inversion operation are preferred. Some of the non-invariant measures include interest factor, IS, PS, and the Jaccard coefficient.

Null Addition Property Suppose we are interested in analyzing the relationship between a pair of words, such as `data` and `mining`, in a set of documents. If a collection of articles about ice fishing is added to the data set, should the association between `data` and `mining` be affected? This process of adding unrelated data (in this case, documents) to a given data set is known as the **null addition** operation.

Definition 6.7 (Null Addition Property). An objective measure M is invariant under the null addition operation if it is not affected by increasing f_{00}, while all other frequencies in the contingency table stay the same.

For applications such as document analysis or market basket analysis, the measure is expected to remain invariant under the null addition operation. Otherwise, the relationship between words may disappear simply by adding enough documents that do not contain both words! Examples of measures that satisfy this property include cosine (IS) and Jaccard (ξ) measures, while those that violate this property include interest factor, PS, odds ratio, and the ϕ-coefficient.

Scaling Property Table 6.16 shows the contingency tables for gender and the grades achieved by students enrolled in a particular course in 1993 and 2004. The data in these tables showed that the number of male students has doubled since 1993, while the number of female students has increased by a factor of 3. However, the male students in 2004 are not performing any better than those in 1993 because the ratio of male students who achieve a high grade to those who achieve a low grade is still the same, i.e., 3:4. Similarly, the female students in 2004 are performing no better than those in 1993. The association between grade and gender is expected to remain unchanged despite changes in the sampling distribution.

Table 6.16. The grade-gender example.

	Male	Female	
High	30	20	50
Low	40	10	50
	70	30	100

(a) Sample data from 1993.

	Male	Female	
High	60	60	120
Low	80	30	110
	140	90	230

(b) Sample data from 2004.

Table 6.17. Properties of symmetric measures.

Symbol	Measure	Inversion	Null Addition	Scaling
ϕ	ϕ-coefficient	Yes	No	No
α	odds ratio	Yes	No	Yes
κ	Cohen's	Yes	No	No
I	Interest	No	No	No
IS	Cosine	No	Yes	No
PS	Piatetsky-Shapiro's	Yes	No	No
S	Collective strength	Yes	No	No
ζ	Jaccard	No	Yes	No
h	All-confidence	No	No	No
s	Support	No	No	No

Definition 6.8 (Scaling Invariance Property). An objective measure M is invariant under the row/column scaling operation if $M(T) = M(T')$, where T is a contingency table with frequency counts $[f_{11}; \ f_{10}; \ f_{01}; \ f_{00}]$, T' is a contingency table with scaled frequency counts $[k_1 k_3 f_{11}; \ k_2 k_3 f_{10}; \ k_1 k_4 f_{01}; \ k_2 k_4 f_{00}]$, and k_1, k_2, k_3, k_4 are positive constants.

From Table 6.17, notice that only the odds ratio (α) is invariant under the row and column scaling operations. All other measures such as the ϕ-coefficient, κ, IS, interest factor, and collective strength (S) change their values when the rows and columns of the contingency table are rescaled. Although we do not discuss the properties of asymmetric measures (such as confidence, J-measure, Gini index, and conviction), it is clear that such measures do not preserve their values under inversion and row/column scaling operations, but are invariant under the null addition operation.

6.7.2   Measures beyond Pairs of Binary Variables

The measures shown in Tables 6.11 and 6.12 are defined for pairs of binary variables (e.g., 2-itemsets or association rules). However, many of them, such as support and all-confidence, are also applicable to larger-sized itemsets. Other measures, such as interest factor, IS, PS, and Jaccard coefficient, can be extended to more than two variables using the frequency tables tabulated in a multidimensional contingency table. An example of a three-dimensional contingency table for a, b, and c is shown in Table 6.18. Each entry f_{ijk} in this table represents the number of transactions that contain a particular combination of items a, b, and c. For example, f_{101} is the number of transactions that contain a and c, but not b. On the other hand, a marginal frequency

Table 6.18. Example of a three-dimensional contingency table.

c	b	$\bar{b}$			$\bar{c}$	b	$\bar{b}$	
a	f_{111}	f_{101}	f_{1+1}		a	f_{110}	f_{100}	f_{1+0}
$\bar{a}$	f_{011}	f_{001}	f_{0+1}		$\bar{a}$	f_{010}	f_{000}	f_{0+0}
	f_{+11}	f_{+01}	f_{++1}			f_{+10}	f_{+00}	f_{++0}

such as f_{1+1} is the number of transactions that contain a and c, irrespective of whether b is present in the transaction.

Given a k-itemset $\{i_1, i_2, \ldots, i_k\}$, the condition for statistical independence can be stated as follows:

$$f_{i_1 i_2 \ldots i_k} = \frac{f_{i_1+\ldots+} \times f_{+i_2\ldots+} \times \ldots \times f_{++\ldots i_k}}{N^{k-1}}. \tag{6.12}$$

With this definition, we can extend objective measures such as interest factor and PS, which are based on deviations from statistical independence, to more than two variables:

$$I = \frac{N^{k-1} \times f_{i_1 i_2 \ldots i_k}}{f_{i_1+\ldots+} \times f_{+i_2\ldots+} \times \ldots \times f_{++\ldots i_k}}$$

$$PS = \frac{f_{i_1 i_2 \ldots i_k}}{N} - \frac{f_{i_1+\ldots+} \times f_{+i_2\ldots+} \times \ldots \times f_{++\ldots i_k}}{N^k}$$

Another approach is to define the objective measure as the maximum, minimum, or average value for the associations between pairs of items in a pattern. For example, given a k-itemset $X = \{i_1, i_2, \ldots, i_k\}$, we may define the ϕ-coefficient for X as the average ϕ-coefficient between every pair of items (i_p, i_q) in X. However, because the measure considers only pairwise associations, it may not capture all the underlying relationships within a pattern.

Analysis of multidimensional contingency tables is more complicated because of the presence of partial associations in the data. For example, some associations may appear or disappear when conditioned upon the value of certain variables. This problem is known as **Simpson's paradox** and is described in the next section. More sophisticated statistical techniques are available to analyze such relationships, e.g., loglinear models, but these techniques are beyond the scope of this book.

Table 6.19. A two-way contingency table between the sale of high-definition television and exercise machine.

Buy HDTV	Buy Exercise Machine Yes	Buy Exercise Machine No	
Yes	99	81	180
No	54	66	120
	153	147	300

Table 6.20. Example of a three-way contingency table.

Customer Group	Buy HDTV	Buy Exercise Machine Yes	Buy Exercise Machine No	Total
College Students	Yes	1	9	10
	No	4	30	34
Working Adult	Yes	98	72	170
	No	50	36	86

6.7.3 Simpson's Paradox

It is important to exercise caution when interpreting the association between variables because the observed relationship may be influenced by the presence of other confounding factors, i.e., hidden variables that are not included in the analysis. In some cases, the hidden variables may cause the observed relationship between a pair of variables to disappear or reverse its direction, a phenomenon that is known as Simpson's paradox. We illustrate the nature of this paradox with the following example.

Consider the relationship between the sale of high-definition television (HDTV) and exercise machine, as shown in Table 6.19. The rule {HDTV=Yes} $\longrightarrow$ {Exercise machine=Yes} has a confidence of $99/180 = 55\%$ and the rule {HDTV=No} $\longrightarrow$ {Exercise machine=Yes} has a confidence of $54/120 = 45\%$. Together, these rules suggest that customers who buy high-definition televisions are more likely to buy exercise machines than those who do not buy high-definition televisions.

However, a deeper analysis reveals that the sales of these items depend on whether the customer is a college student or a working adult. Table 6.20 summarizes the relationship between the sale of HDTVs and exercise machines among college students and working adults. Notice that the support counts given in the table for college students and working adults sum up to the frequencies shown in Table 6.19. Furthermore, there are more working adults

than college students who buy these items. For college students:

$$c(\{\text{HDTV=Yes}\} \longrightarrow \{\text{Exercise machine=Yes}\}) = 1/10 = 10\%,$$
$$c(\{\text{HDTV=No}\} \longrightarrow \{\text{Exercise machine=Yes}\}) = 4/34 = 11.8\%,$$

while for working adults:

$$c(\{\text{HDTV=Yes}\} \longrightarrow \{\text{Exercise machine=Yes}\}) = 98/170 = 57.7\%,$$
$$c(\{\text{HDTV=No}\} \longrightarrow \{\text{Exercise machine=Yes}\}) = 50/86 = 58.1\%.$$

The rules suggest that, for each group, customers who do not buy high-definition televisions are more likely to buy exercise machines, which contradict the previous conclusion when data from the two customer groups are pooled together. Even if alternative measures such as correlation, odds ratio, or interest are applied, we still find that the sale of HDTV and exercise machine is positively correlated in the combined data but is negatively correlated in the stratified data (see Exercise 20 on page 414). The reversal in the direction of association is known as Simpson's paradox.

The paradox can be explained in the following way. Notice that most customers who buy HDTVs are working adults. Working adults are also the largest group of customers who buy exercise machines. Because nearly 85% of the customers are working adults, the observed relationship between HDTV and exercise machine turns out to be stronger in the combined data than what it would have been if the data is stratified. This can also be illustrated mathematically as follows. Suppose

$$a/b < c/d \text{ and } p/q < r/s,$$

where a/b and p/q may represent the confidence of the rule $A \longrightarrow B$ in two different strata, while c/d and r/s may represent the confidence of the rule $\overline{A} \longrightarrow B$ in the two strata. When the data is pooled together, the confidence values of the rules in the combined data are $(a+p)/(b+q)$ and $(c+r)/(d+s)$, respectively. Simpson's paradox occurs when

$$\frac{a+p}{b+q} > \frac{c+r}{d+s},$$

thus leading to the wrong conclusion about the relationship between the variables. The lesson here is that proper stratification is needed to avoid generating spurious patterns resulting from Simpson's paradox. For example, market
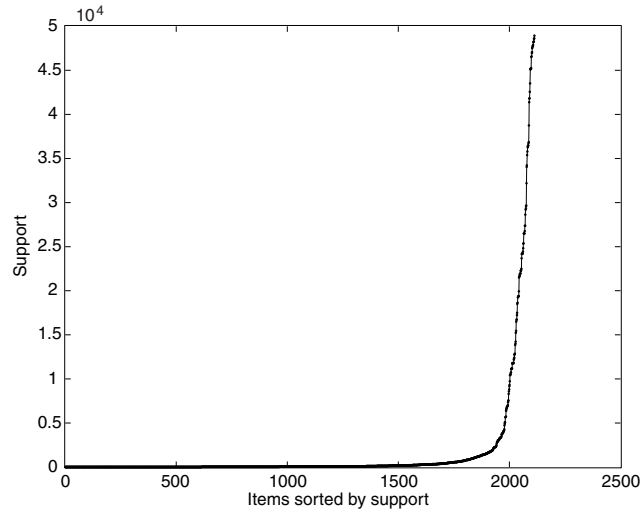
Figure 6.29. Support distribution of items in the census data set.

basket data from a major supermarket chain should be stratified according to store locations, while medical records from various patients should be stratified according to confounding factors such as age and gender.

6.8 Effect of Skewed Support Distribution

The performances of many association analysis algorithms are influenced by properties of their input data. For example, the computational complexity of the *Apriori* algorithm depends on properties such as the number of items in the data and average transaction width. This section examines another important property that has significant influence on the performance of association analysis algorithms as well as the quality of extracted patterns. More specifically, we focus on data sets with skewed support distributions, where most of the items have relatively low to moderate frequencies, but a small number of them have very high frequencies.

An example of a real data set that exhibits such a distribution is shown in Figure 6.29. The data, taken from the PUMS (Public Use Microdata Sample) census data, contains 49,046 records and 2113 asymmetric binary variables. We shall treat the asymmetric binary variables as items and records as transactions in the remainder of this section. While more than 80% of the items have support less than 1%, a handful of them have support greater than 90%.

Table 6.21. Grouping the items in the census data set based on their support values.

Group	G_1	G_2	G_3
Support	$< 1\%$	$1\% - 90\%$	$> 90\%$
Number of Items	1735	358	20

To illustrate the effect of skewed support distribution on frequent itemset mining, we divide the items into three groups, G_1, G_2, and G_3, according to their support levels. The number of items that belong to each group is shown in Table 6.21.

Choosing the right support threshold for mining this data set can be quite tricky. If we set the threshold too high (e.g., 20%), then we may miss many interesting patterns involving the low support items from G_1. In market basket analysis, such low support items may correspond to expensive products (such as jewelry) that are seldom bought by customers, but whose patterns are still interesting to retailers. Conversely, when the threshold is set too low, it becomes difficult to find the association patterns due to the following reasons. First, the computational and memory requirements of existing association analysis algorithms increase considerably with low support thresholds. Second, the number of extracted patterns also increases substantially with low support thresholds. Third, we may extract many spurious patterns that relate a high-frequency item such as milk to a low-frequency item such as caviar. Such patterns, which are called **cross-support** patterns, are likely to be spurious because their correlations tend to be weak. For example, at a support threshold equal to 0.05%, there are 18,847 frequent pairs involving items from G_1 and G_3. Out of these, 93% of them are cross-support patterns; i.e., the patterns contain items from both G_1 and G_3. The maximum correlation obtained from the cross-support patterns is 0.029, which is much lower than the maximum correlation obtained from frequent patterns involving items from the same group (which is as high as 1.0). Similar statement can be made about many other interestingness measures discussed in the previous section. This example shows that a large number of weakly correlated cross-support patterns can be generated when the support threshold is sufficiently low. Before presenting a methodology for eliminating such patterns, we formally define the concept of cross-support patterns.

Definition 6.9 (Cross-Support Pattern). A cross-support pattern is an itemset $X = \{i_1, i_2, \ldots, i_k\}$ whose support ratio

$$r(X) = \frac{\min\left[s(i_1), s(i_2), \ldots, s(i_k)\right]}{\max\left[s(i_1), s(i_2), \ldots, s(i_k)\right]}, \tag{6.13}$$

is less than a user-specified threshold h_c.

Example 6.4. Suppose the support for milk is 70%, while the support for sugar is 10% and caviar is 0.04%. Given $h_c = 0.01$, the frequent itemset {milk, sugar, caviar} is a cross-support pattern because its support ratio is

$$r = \frac{\min\left[0.7, 0.1, 0.0004\right]}{\max\left[0.7, 0.1, 0.0004\right]} = \frac{0.0004}{0.7} = 0.00058 < 0.01.$$

∎

Existing measures such as support and confidence may not be sufficient to eliminate cross-support patterns, as illustrated by the data set shown in Figure 6.30. Assuming that $h_c = 0.3$, the itemsets $\{p, q\}$, $\{p, r\}$, and $\{p, q, r\}$ are cross-support patterns because their support ratios, which are equal to 0.2, are less than the threshold h_c. Although we can apply a high support threshold, say, 20%, to eliminate the cross-support patterns, this may come at the expense of discarding other interesting patterns such as the strongly correlated itemset, $\{q, r\}$ that has support equal to 16.7%.

Confidence pruning also does not help because the confidence of the rules extracted from cross-support patterns can be very high. For example, the confidence for $\{q\} \longrightarrow \{p\}$ is 80% even though $\{p, q\}$ is a cross-support pattern. The fact that the cross-support pattern can produce a high-confidence rule should not come as a surprise because one of its items (p) appears very frequently in the data. Therefore, p is expected to appear in many of the transactions that contain q. Meanwhile, the rule $\{q\} \longrightarrow \{r\}$ also has high confidence even though $\{q, r\}$ is not a cross-support pattern. This example demonstrates the difficulty of using the confidence measure to distinguish between rules extracted from cross-support and non-cross-support patterns.

Returning to the previous example, notice that the rule $\{p\} \longrightarrow \{q\}$ has very low confidence because most of the transactions that contain p do not contain q. In contrast, the rule $\{r\} \longrightarrow \{q\}$, which is derived from the pattern $\{q, r\}$, has very high confidence. This observation suggests that cross-support patterns can be detected by examining the lowest confidence rule that can be extracted from a given itemset. The proof of this statement can be understood as follows.

p	q	r
0	1	1
1	1	1
1	1	1
1	1	1
1	1	1
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Figure 6.30. A transaction data set containing three items, p, q, and r, where p is a high support item and q and r are low support items.

1. Recall the following anti-monotone property of confidence:

$$conf(\{i_1 i_2\} \longrightarrow \{i_3, i_4, \ldots, i_k\}) \leq conf(\{i_1 i_2 i_3\} \longrightarrow \{i_4, i_5, \ldots, i_k\}).$$

   This property suggests that confidence never increases as we shift more items from the left- to the right-hand side of an association rule. Because of this property, the lowest confidence rule extracted from a frequent itemset contains only one item on its left-hand side. We denote the set of all rules with only one item on its left-hand side as R_1.

2. Given a frequent itemset $\{i_1, i_2, \ldots, i_k\}$, the rule

$$\{i_j\} \longrightarrow \{i_1, i_2, \ldots, i_{j-1}, i_{j+1}, \ldots, i_k\}$$

   has the lowest confidence in R_1 if $s(i_j) = \max\left[s(i_1), s(i_2), \ldots, s(i_k)\right]$. This follows directly from the definition of confidence as the ratio between the rule's support and the support of the rule antecedent.

3. Summarizing the previous points, the lowest confidence attainable from a frequent itemset $\{i_1, i_2, \ldots, i_k\}$ is

$$\frac{s(\{i_1, i_2, \ldots, i_k\})}{\max\big[s(i_1), s(i_2), \ldots, s(i_k)\big]}.$$

This expression is also known as the **h-confidence** or **all-confidence** measure. Because of the anti-monotone property of support, the numerator of the h-confidence measure is bounded by the minimum support of any item that appears in the frequent itemset. In other words, the h-confidence of an itemset $X = \{i_1, i_2, \ldots, i_k\}$ must not exceed the following expression:

$$\text{h-confidence}(X) \leq \frac{\min\big[s(i_1), s(i_2), \ldots, s(i_k)\big]}{\max\big[s(i_1), s(i_2), \ldots, s(i_k)\big]}.$$

Note the equivalence between the upper bound of h-confidence and the support ratio (r) given in Equation 6.13. Because the support ratio for a cross-support pattern is always less than h_c, the h-confidence of the pattern is also guaranteed to be less than h_c.

Therefore, cross-support patterns can be eliminated by ensuring that the h-confidence values for the patterns exceed h_c. As a final note, it is worth mentioning that the advantages of using h-confidence go beyond eliminating cross-support patterns. The measure is also anti-monotone, i.e.,

$$\text{h-confidence}(\{i_1, i_2, \ldots, i_k\}) \geq \text{h-confidence}(\{i_1, i_2, \ldots, i_{k+1}\}),$$

and thus can be incorporated directly into the mining algorithm. Furthermore, h-confidence ensures that the items contained in an itemset are strongly associated with each other. For example, suppose the h-confidence of an itemset X is 80%. If one of the items in X is present in a transaction, there is at least an 80% chance that the rest of the items in X also belong to the same transaction. Such strongly associated patterns are called **hyperclique patterns**.

6.9 Bibliographic Notes

The association rule mining task was first introduced by Agrawal et al. in [228, 229] to discover interesting relationships among items in market basket

transactions. Since its inception, extensive studies have been conducted to address the various conceptual, implementation, and application issues pertaining to the association analysis task. A summary of the various research activities in this area is shown in Figure 6.31.

Conceptual Issues

Research in conceptual issues is focused primarily on (1) developing a framework to describe the theoretical underpinnings of association analysis, (2) extending the formulation to handle new types of patterns, and (3) extending the formulation to incorporate attribute types beyond asymmetric binary data.

Following the pioneering work by Agrawal et al., there has been a vast amount of research on developing a theory for the association analysis problem. In [254], Gunopoulos et al. showed a relation between the problem of finding maximal frequent itemsets and the hypergraph transversal problem. An upper bound on the complexity of association analysis task was also derived. Zaki et al. [334, 336] and Pasquier et al. [294] have applied formal concept analysis to study the frequent itemset generation problem. The work by Zaki et al. have subsequently led them to introduce the notion of closed frequent itemsets [336]. Friedman et al. have studied the association analysis problem in the context of **bump hunting** in multidimensional space [252]. More specifically, they consider frequent itemset generation as the task of finding high probability density regions in multidimensional space.

Over the years, new types of patterns have been defined, such as profile association rules [225], cyclic association rules [290], fuzzy association rules [273], exception rules [316], negative association rules [238, 304], weighted association rules [240, 300], dependence rules [308], peculiar rules[340], inter-transaction association rules [250, 323], and partial classification rules [231, 285]. Other types of patterns include closed itemsets [294, 336], maximal itemsets [234], hyperclique patterns [330], support envelopes [314], emerging patterns [246], and contrast sets [233]. Association analysis has also been successfully applied to sequential [230, 312], spatial [266], and graph-based [268, 274, 293, 331, 335] data. The concept of cross-support pattern was first introduced by Hui et al. in [330]. An efficient algorithm (called Hyperclique Miner) that automatically eliminates cross-support patterns was also proposed by the authors.

Substantial research has been conducted to extend the original association rule formulation to nominal [311], ordinal [281], interval [284], and ratio [253, 255, 311, 325, 339] attributes. One of the key issues is how to define the support measure for these attributes. A methodology was proposed by Steinbach et

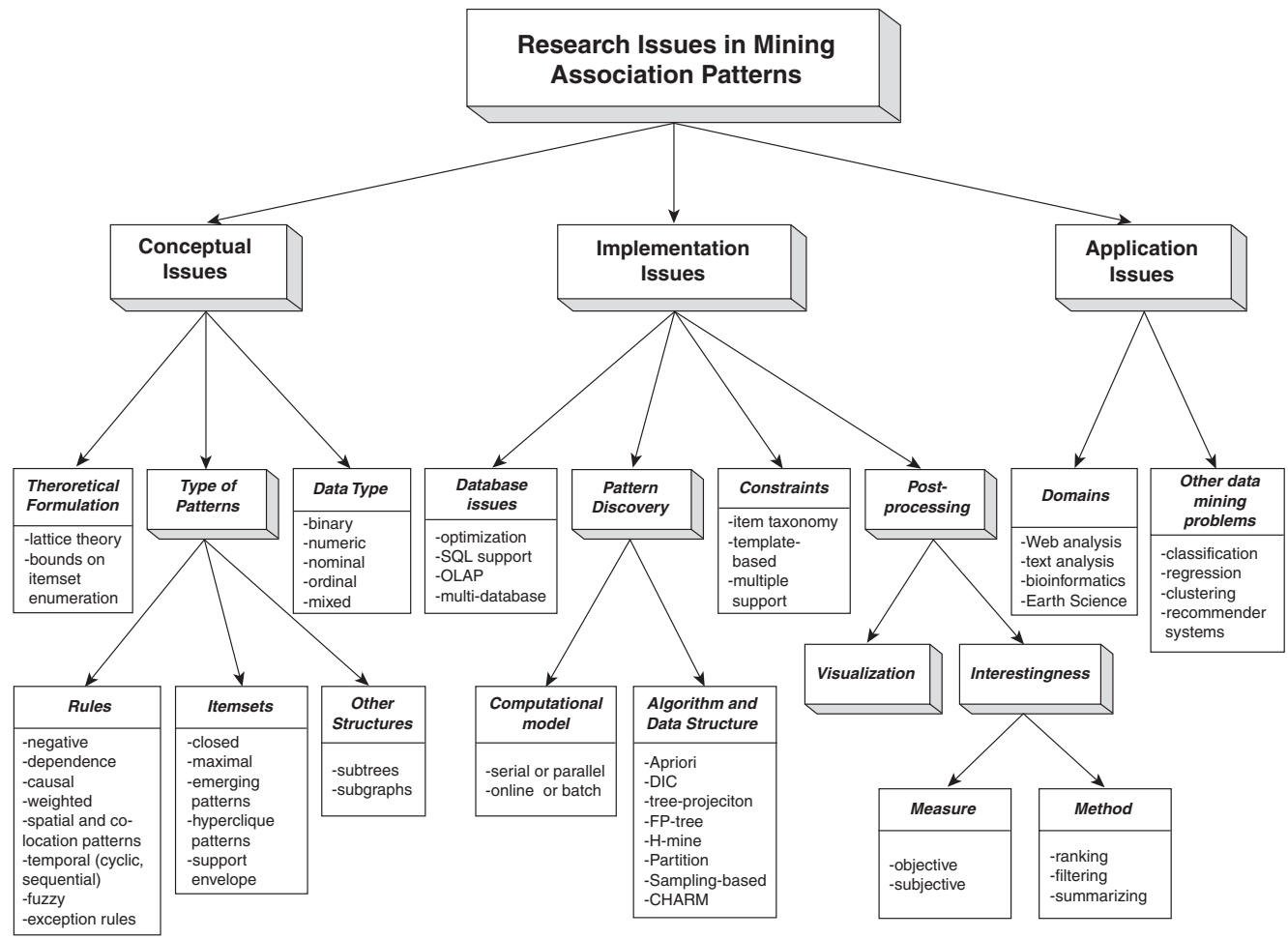al. [315] to extend the traditional notion of support to more general patterns and attribute types.

Figure 6.31. A summary of the various research activities in association analysis.

Implementation Issues

Research activities in this area revolve around (1) integrating the mining capability into existing database technology, (2) developing efficient and scalable mining algorithms, (3) handling user-specified or domain-specific constraints, and (4) post-processing the extracted patterns.

There are several advantages to integrating association analysis into existing database technology. First, it can make use of the indexing and query processing capabilities of the database system. Second, it can also exploit the DBMS support for scalability, check-pointing, and parallelization [301]. The SETM algorithm developed by Houtsma et al. [265] was one of the earliest algorithms to support association rule discovery via SQL queries. Since then, numerous methods have been developed to provide capabilities for mining association rules in database systems. For example, the DMQL [258] and M-SQL [267] query languages extend the basic SQL with new operators for mining association rules. The Mine Rule operator [283] is an expressive SQL operator that can handle both clustered attributes and item hierarchies. Tsur et al. [322] developed a generate-and-test approach called **query flocks** for mining association rules. A distributed OLAP-based infrastructure was developed by Chen et al. [241] for mining multilevel association rules.

Dunkel and Soparkar [248] investigated the time and storage complexity of the *Apriori* algorithm. The FP-growth algorithm was developed by Han et al. in [259]. Other algorithms for mining frequent itemsets include the DHP (dynamic hashing and pruning) algorithm proposed by Park et al. [292] and the Partition algorithm developed by Savasere et al [303]. A sampling-based frequent itemset generation algorithm was proposed by Toivonen [320]. The algorithm requires only a single pass over the data, but it can produce more candidate itemsets than necessary. The Dynamic Itemset Counting (DIC) algorithm [239] makes only 1.5 passes over the data and generates less candidate itemsets than the sampling-based algorithm. Other notable algorithms include the tree-projection algorithm [223] and H-Mine [295]. Survey articles on frequent itemset generation algorithms can be found in [226, 262]. A repository of data sets and algorithms is available at the Frequent Itemset Mining Implementations (FIMI) repository (http://fimi.cs.helsinki.fi). Parallel algorithms for mining association patterns have been developed by various authors [224, 256, 287, 306, 337]. A survey of such algorithms can be found in [333]. Online and incremental versions of association rule mining algorithms had also been proposed by Hidber [260] and Cheung et al. [242].

Srikant et al. [313] have considered the problem of mining association rules in the presence of boolean constraints such as the following:

(Cookies ∧ Milk) ∨ (descendents(Cookies) ∧ ¬ancestors(Wheat Bread))

Given such a constraint, the algorithm looks for rules that contain both cookies and milk, or rules that contain the descendent items of cookies but not ancestor items of wheat bread. Singh et al. [310] and Ng et al. [288] had also developed alternative techniques for constrained-based association rule mining. Constraints can also be imposed on the support for different itemsets. This problem was investigated by Wang et al. [324], Liu et al. in [279], and Seno et al. [305].

One potential problem with association analysis is the large number of patterns that can be generated by current algorithms. To overcome this problem, methods to rank, summarize, and filter patterns have been developed. Toivonen et al. [321] proposed the idea of eliminating redundant rules using **structural rule covers** and to group the remaining rules using clustering. Liu et al. [280] applied the statistical chi-square test to prune spurious patterns and summarized the remaining patterns using a subset of the patterns called **direction setting rules**. The use of objective measures to filter patterns has been investigated by many authors, including Brin et al. [238], Bayardo and Agrawal [235], Aggarwal and Yu [227], and DuMouchel and Pregibon[247]. The properties for many of these measures were analyzed by Piatetsky-Shapiro [297], Kamber and Singhal [270], Hilderman and Hamilton [261], and Tan et al. [318]. The grade-gender example used to highlight the importance of the row and column scaling invariance property was heavily influenced by the discussion given in [286] by Mosteller. Meanwhile, the tea-coffee example illustrating the limitation of confidence was motivated by an example given in [238] by Brin et al. Because of the limitation of confidence, Brin et al. [238] had proposed the idea of using interest factor as a measure of interestingness. The all-confidence measure was proposed by Omiecinski [289]. Xiong et al. [330] introduced the cross-support property and showed that the all-confidence measure can be used to eliminate cross-support patterns. A key difficulty in using alternative objective measures besides support is their lack of a monotonicity property, which makes it difficult to incorporate the measures directly into the mining algorithms. Xiong et al. [328] have proposed an efficient method for mining correlations by introducing an upper bound function to the ϕ-coefficient. Although the measure is non-monotone, it has an upper bound expression that can be exploited for the efficient mining of strongly correlated itempairs.

Fabris and Freitas [249] have proposed a method for discovering interesting associations by detecting the occurrences of Simpson's paradox [309]. Megiddo and Srikant [282] described an approach for validating the extracted

patterns using hypothesis testing methods. A resampling-based technique was also developed to avoid generating spurious patterns because of the multiple comparison problem. Bolton et al. [237] have applied the Benjamini-Hochberg [236] and Bonferroni correction methods to adjust the p-values of discovered patterns in market basket data. Alternative methods for handling the multiple comparison problem were suggested by Webb [326] and Zhang et al. [338].

Application of subjective measures to association analysis has been investigated by many authors. Silberschatz and Tuzhilin [307] presented two principles in which a rule can be considered interesting from a subjective point of view. The concept of unexpected condition rules was introduced by Liu et al. in [277]. Cooley et al. [243] analyzed the idea of combining soft belief sets using the Dempster-Shafer theory and applied this approach to identify contradictory and novel association patterns in Web data. Alternative approaches include using Bayesian networks [269] and neighborhood-based information [245] to identify subjectively interesting patterns.

Visualization also helps the user to quickly grasp the underlying structure of the discovered patterns. Many commercial data mining tools display the complete set of rules (which satisfy both support and confidence threshold criteria) as a two-dimensional plot, with each axis corresponding to the antecedent or consequent itemsets of the rule. Hofmann et al. [263] proposed using Mosaic plots and Double Decker plots to visualize association rules. This approach can visualize not only a particular rule, but also the overall contingency table between itemsets in the antecedent and consequent parts of the rule. Nevertheless, this technique assumes that the rule consequent consists of only a single attribute.

Application Issues

Association analysis has been applied to a variety of application domains such as Web mining [296, 317], document analysis [264], telecommunication alarm diagnosis [271], network intrusion detection [232, 244, 275], and bioinformatics [302, 327]. Applications of association and correlation pattern analysis to Earth Science studies have been investigated in [298, 299, 319].

Association patterns have also been applied to other learning problems such as classification [276, 278], regression [291], and clustering [257, 329, 332]. A comparison between classification and association rule mining was made by Freitas in his position paper [251]. The use of association patterns for clustering has been studied by many authors including Han et al.[257], Kosters et al. [272], Yang et al. [332] and Xiong et al. [329].

Bibliography

[223] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. A Tree Projection Algorithm for Generation of Frequent Itemsets. *Journal of Parallel and Distributed Computing (Special Issue on High Performance Data Mining)*, 61(3):350–371, 2001.

[224] R. C. Agarwal and J. C. Shafer. Parallel Mining of Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, March 1998.

[225] C. C. Aggarwal, Z. Sun, and P. S. Yu. Online Generation of Profile Association Rules. In *Proc. of the 4th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 129–133, New York, NY, August 1996.

[226] C. C. Aggarwal and P. S. Yu. Mining Large Itemsets for Association Rules. *Data Engineering Bulletin*, 21(1):23–31, March 1998.

[227] C. C. Aggarwal and P. S. Yu. Mining Associations with the Collective Strength Approach. *IEEE Trans. on Knowledge and Data Engineering*, 13(6):863–873, January/February 2001.

[228] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5:914–925, 1993.

[229] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 207–216, Washington, DC, 1993.

[230] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. of Intl. Conf. on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995.

[231] K. Ali, S. Manganaris, and R. Srikant. Partial Classification using Association Rules. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 115–118, Newport Beach, CA, August 1997.

[232] D. Barbará, J. Couto, S. Jajodia, and N. Wu. ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *SIGMOD Record*, 30(4):15–24, 2001.

[233] S. D. Bay and M. Pazzani. Detecting Group Differences: Mining Contrast Sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.

[234] R. Bayardo. Efficiently Mining Long Patterns from Databases. In *Proc. of 1998 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 85–93, Seattle, WA, June 1998.

[235] R. Bayardo and R. Agrawal. Mining the Most Interesting Rules. In *Proc. of the 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 145–153, San Diego, CA, August 1999.

[236] Y. Benjamini and Y. Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal Royal Statistical Society B*, 57 (1):289–300, 1995.

[237] R. J. Bolton, D. J. Hand, and N. M. Adams. Determining Hit Rate in Pattern Search. In *Proc. of the ESF Exploratory Workshop on Pattern Detection and Discovery in Data Mining*, pages 36–48, London, UK, September 2002.

[238] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 265–276, Tucson, AZ, 1997.

[239] S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for market basket data. In *Proc. of 1997 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 255–264, Tucson, AZ, June 1997.

[240] C. H. Cai, A. Fu, C. H. Cheng, and W. W. Kwong. Mining Association Rules with Weighted Items. In *Proc. of IEEE Intl. Database Engineering and Applications Symp.*, pages 68–77, Cardiff, Wales, 1998.

[241] Q. Chen, U. Dayal, and M. Hsu. A Distributed OLAP infrastructure for E-Commerce. In *Proc. of the 4th IFCIS Intl. Conf. on Cooperative Information Systems*, pages 209–220, Edinburgh, Scotland, 1999.

[242] D. C. Cheung, S. D. Lee, and B. Kao. A General Incremental Technique for Maintaining Discovered Association Rules. In *Proc. of the 5th Intl. Conf. on Database Systems for Advanced Applications*, pages 185–194, Melbourne, Australia, 1997.

[243] R. Cooley, P. N. Tan, and J. Srivastava. Discovery of Interesting Usage Patterns from Web Data. In M. Spiliopoulou and B. Masand, editors, *Advances in Web Usage Analysis and User Profiling*, volume 1836, pages 163–182. Lecture Notes in Computer Science, 2000.

[244] P. Dokas, L. Ertöz, V. Kumar, A. Lazarevic, J. Srivastava, and P. N. Tan. Data Mining for Network Intrusion Detection. In *Proc. NSF Workshop on Next Generation Data Mining*, Baltimore, MD, 2002.

[245] G. Dong and J. Li. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. In *Proc. of the 2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 72–86, Melbourne, Australia, April 1998.

[246] G. Dong and J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences. In *Proc. of the 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 43–52, San Diego, CA, August 1999.

[247] W. DuMouchel and D. Pregibon. Empirical Bayes Screening for Multi-Item Associations. In *Proc. of the 7th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 67–76, San Francisco, CA, August 2001.

[248] B. Dunkel and N. Soparkar. Data Organization and Access for Efficient Data Mining. In *Proc. of the 15th Intl. Conf. on Data Engineering*, pages 522–529, Sydney, Australia, March 1999.

[249] C. C. Fabris and A. A. Freitas. Discovering surprising patterns by detecting occurrences of Simpson's paradox. In *Proc. of the 19th SGES Intl. Conf. on Knowledge-Based Systems and Applied Artificial Intelligence)*, pages 148–160, Cambridge, UK, December 1999.

[250] L. Feng, H. J. Lu, J. X. Yu, and J. Han. Mining inter-transaction associations with templates. In *Proc. of the 8th Intl. Conf. on Information and Knowledge Management*, pages 225–233, Kansas City, Missouri, Nov 1999.

[251] A. A. Freitas. Understanding the crucial differences between classification and discovery of association rules—a position paper. *SIGKDD Explorations*, 2(1):65–69, 2000.

[252] J. H. Friedman and N. I. Fisher. Bump hunting in high-dimensional data. *Statistics and Computing*, 9(2):123–143, April 1999.

[253] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining Optimized Association Rules for Numeric Attributes. In *Proc. of the 15th Symp. on Principles of Database Systems*, pages 182–191, Montreal, Canada, June 1996.

[254] D. Gunopulos, R. Khardon, H. Mannila, and H. Toivonen. Data Mining, Hypergraph Transversals, and Machine Learning. In *Proc. of the 16th Symp. on Principles of Database Systems*, pages 209–216, Tucson, AZ, May 1997.

[255] E.-H. Han, G. Karypis, and V. Kumar. Min-Apriori: An Algorithm for Finding Association Rules in Data with Continuous Attributes. http://www.cs.umn.edu/~han, 1997.

[256] E.-H. Han, G. Karypis, and V. Kumar. Scalable Parallel Data Mining for Association Rules. In *Proc. of 1997 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 277–288, Tucson, AZ, May 1997.

[257] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher. Clustering Based on Association Rule Hypergraphs. In *Proc. of the 1997 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Tucson, AZ, 1997.

[258] J. Han, Y. Fu, K. Koperski, W. Wang, and O. R. Zaïane. DMQL: A data mining query language for relational databases. In *Proc. of the 1996 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Montreal, Canada, June 1996.

[259] J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. In *Proc. ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*, pages 1–12, Dallas, TX, May 2000.

[260] C. Hidber. Online Association Rule Mining. In *Proc. of 1999 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 145–156, Philadelphia, PA, 1999.

[261] R. J. Hilderman and H. J. Hamilton. *Knowledge Discovery and Measures of Interest.* Kluwer Academic Publishers, 2001.

[262] J. Hipp, U. Guntzer, and G. Nakhaeizadeh. Algorithms for Association Rule Mining— A General Survey. *SigKDD Explorations*, 2(1):58–64, June 2000.

[263] H. Hofmann, A. P. J. M. Siebes, and A. F. X. Wilhelm. Visualizing Association Rules with Interactive Mosaic Plots. In *Proc. of the 6th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 227–235, Boston, MA, August 2000.

[264] J. D. Holt and S. M. Chung. Efficient Mining of Association Rules in Text Databases. In *Proc. of the 8th Intl. Conf. on Information and Knowledge Management*, pages 234–242, Kansas City, Missouri, 1999.

[265] M. Houtsma and A. Swami. Set-oriented Mining for Association Rules in Relational Databases. In *Proc. of the 11th Intl. Conf. on Data Engineering*, pages 25–33, Taipei, Taiwan, 1995.

[266] Y. Huang, S. Shekhar, and H. Xiong. Discovering Co-location Patterns from Spatial Datasets: A General Approach. *IEEE Trans. on Knowledge and Data Engineering*, 16 (12):1472–1485, December 2004.

[267] T. Imielinski, A. Virmani, and A. Abdulghani. DataMine: Application Programming Interface and Query Language for Database Mining. In *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 256–262, Portland, Oregon, 1996.

[268] A. Inokuchi, T. Washio, and H. Motoda. An Apriori-based Algorithm for Mining Frequent Substructures from Graph Data. In *Proc. of the 4th European Conf. of Principles and Practice of Knowledge Discovery in Databases*, pages 13–23, Lyon, France, 2000.

[269] S. Jaroszewicz and D. Simovici. Interestingness of Frequent Itemsets Using Bayesian Networks as Background Knowledge. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 178–186, Seattle, WA, August 2004.

[270] M. Kamber and R. Shinghal. Evaluating the Interestingness of Characteristic Rules. In *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 263–266, Portland, Oregon, 1996.

[271] M. Klemettinen. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases.* PhD thesis, University of Helsinki, 1999.

[272] W. A. Kosters, E. Marchiori, and A. Oerlemans. Mining Clusters with Association Rules. In *The 3rd Symp. on Intelligent Data Analysis (IDA99)*, pages 39–50, Amsterdam, August 1999.

[273] C. M. Kuok, A. Fu, and M. H. Wong. Mining Fuzzy Association Rules in Databases. *ACM SIGMOD Record*, 27(1):41–46, March 1998.

[274] M. Kuramochi and G. Karypis. Frequent Subgraph Discovery. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, pages 313–320, San Jose, CA, November 2001.

[275] W. Lee, S. J. Stolfo, and K. W. Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, 14(6):533–567, 2000.

[276] W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-association Rules. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, pages 369–376, San Jose, CA, 2001.

[277] B. Liu, W. Hsu, and S. Chen. Using General Impressions to Analyze Discovered Classification Rules. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 31–36, Newport Beach, CA, August 1997.

[278] B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule Mining. In *Proc. of the 4th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 80–86, New York, NY, August 1998.

[279] B. Liu, W. Hsu, and Y. Ma. Mining association rules with multiple minimum supports. In *Proc. of the 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 125–134, San Diego, CA, August 1999.

[280] B. Liu, W. Hsu, and Y. Ma. Pruning and Summarizing the Discovered Associations. In *Proc. of the 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 125–134, San Diego, CA, August 1999.

[281] A. Marcus, J. I. Maletic, and K.-I. Lin. Ordinal association rules for error identification in data sets. In *Proc. of the 10th Intl. Conf. on Information and Knowledge Management*, pages 589–591, Atlanta, GA, October 2001.

[282] N. Megiddo and R. Srikant. Discovering Predictive Association Rules. In *Proc. of the 4th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 274–278, New York, August 1998.

[283] R. Meo, G. Psaila, and S. Ceri. A New SQL-like Operator for Mining Association Rules. In *Proc. of the 22nd VLDB Conf.*, pages 122–133, Bombay, India, 1996.

[284] R. J. Miller and Y. Yang. Association Rules over Interval Data. In *Proc. of 1997 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 452–461, Tucson, AZ, May 1997.

[285] Y. Morimoto, T. Fukuda, H. Matsuzawa, T. Tokuyama, and K. Yoda. Algorithms for mining association rules for binary segmentations of huge categorical databases. In *Proc. of the 24th VLDB Conf.*, pages 380–391, New York, August 1998.

[286] F. Mosteller. Association and Estimation in Contingency Tables. *Journal of the American Statistical Association*, 63:1–28, 1968.

[287] A. Mueller. Fast sequential and parallel algorithms for association rule mining: A comparison. Technical Report CS-TR-3515, University of Maryland, August 1995.

[288] R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory Mining and Pruning Optimizations of Constrained Association Rules. In *Proc. of 1998 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 13–24, Seattle, WA, June 1998.

[289] E. Omiecinski. Alternative Interest Measures for Mining Associations in Databases. *IEEE Trans. on Knowledge and Data Engineering*, 15(1):57–69, January/February 2003.

[290] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *Proc. of the 14th Intl. Conf. on Data Eng.*, pages 412–421, Orlando, FL, February 1998.

[291] A. Ozgur, P. N. Tan, and V. Kumar. RBA: An Integrated Framework for Regression based on Association Rules. In *Proc. of the SIAM Intl. Conf. on Data Mining*, pages 210–221, Orlando, FL, April 2004.

[292] J. S. Park, M.-S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD Record*, 25(2):175–186, 1995.

[293] S. Parthasarathy and M. Coatney. Efficient Discovery of Common Substructures in Macromolecules. In *Proc. of the 2002 IEEE Intl. Conf. on Data Mining*, pages 362–369, Maebashi City, Japan, December 2002.

[294] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. of the 7th Intl. Conf. on Database Theory (ICDT'99)*, pages 398–416, Jerusalem, Israel, January 1999.

[295] J. Pei, J. Han, H. J. Lu, S. Nishio, and S. Tang. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, pages 441–448, San Jose, CA, November 2001.

[296] J. Pei, J. Han, B. Mortazavi-Asl, and H. Zhu. Mining Access Patterns Efficiently from Web Logs. In *Proc. of the 4th Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 396–407, Kyoto, Japan, April 2000.

[297] G. Piatetsky-Shapiro. Discovery, Analysis and Presentation of Strong Rules. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. MIT Press, Cambridge, MA, 1991.

[298] C. Potter, S. Klooster, M. Steinbach, P. N. Tan, V. Kumar, S. Shekhar, and C. Carvalho. Understanding Global Teleconnections of Climate to Regional Model Estimates of Amazon Ecosystem Carbon Fluxes. *Global Change Biology*, 10(5):693–703, 2004.

[299] C. Potter, S. Klooster, M. Steinbach, P. N. Tan, V. Kumar, S. Shekhar, R. Myneni, and R. Nemani. Global Teleconnections of Ocean Climate to Terrestrial Carbon Flux. *J. Geophysical Research*, 108(D17), 2003.

[300] G. D. Ramkumar, S. Ranka, and S. Tsur. Weighted Association Rules: Model and Algorithm. http://www.cs.ucla.edu/~czdemo/tsur/, 1997.

[301] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating Mining with Relational Database Systems: Alternatives and Implications. In *Proc. of 1998 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 343–354, Seattle, WA, 1998.

[302] K. Satou, G. Shibayama, T. Ono, Y. Yamamura, E. Furuichi, S. Kuhara, and T. Takagi. Finding Association Rules on Heterogeneous Genome Data. In *Proc. of the Pacific Symp. on Biocomputing*, pages 397–408, Hawaii, January 1997.

[303] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In *Proc. of the 21st Int. Conf. on Very Large Databases (VLDB'95)*, pages 432–444, Zurich, Switzerland, September 1995.

[304] A. Savasere, E. Omiecinski, and S. Navathe. Mining for Strong Negative Associations in a Large Database of Customer Transactions. In *Proc. of the 14th Intl. Conf. on Data Engineering*, pages 494–502, Orlando, Florida, February 1998.

[305] M. Seno and G. Karypis. LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint. In *Proc. of the 2001 IEEE Intl. Conf. on Data Mining*, pages 505–512, San Jose, CA, November 2001.

[306] T. Shintani and M. Kitsuregawa. Hash based parallel algorithms for mining association rules. In *Proc of the 4th Intl. Conf. on Parallel and Distributed Info. Systems*, pages 19–30, Miami Beach, FL, December 1996.

[307] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. on Knowledge and Data Engineering*, 8(6):970–974, 1996.

[308] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, 1998.

[309] E.-H. Simpson. The Interpretation of Interaction in Contingency Tables. *Journal of the Royal Statistical Society*, B(13):238–241, 1951.

[310] L. Singh, B. Chen, R. Haight, and P. Scheuermann. An Algorithm for Constrained Association Rule Mining in Semi-structured Data. In *Proc. of the 3rd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 148–158, Beijing, China, April 1999.

[311] R. Srikant and R. Agrawal. Mining Quantitative Association Rules in Large Relational Tables. In *Proc. of 1996 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 1–12, Montreal, Canada, 1996.

[312] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proc. of the 5th Intl Conf. on Extending Database Technology (EDBT'96)*, pages 18–32, Avignon, France, 1996.

[313] R. Srikant, Q. Vu, and R. Agrawal. Mining Association Rules with Item Constraints. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 67–73, Newport Beach, CA, August 1997.

[314] M. Steinbach, P. N. Tan, and V. Kumar. Support Envelopes: A Technique for Exploring the Structure of Association Patterns. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 296–305, Seattle, WA, August 2004.

[315] M. Steinbach, P. N. Tan, H. Xiong, and V. Kumar. Extending the Notion of Support. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 689–694, Seattle, WA, August 2004.

[316] E. Suzuki. Autonomous Discovery of Reliable Exception Rules. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 259–262, Newport Beach, CA, August 1997.

[317] P. N. Tan and V. Kumar. Mining Association Patterns in Web Usage Data. In *Proc. of the Intl. Conf. on Advances in Infrastructure for e-Business, e-Education, e-Science and e-Medicine on the Internet*, L'Aquila, Italy, January 2002.

[318] P. N. Tan, V. Kumar, and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns. In *Proc. of the 8th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 32–41, Edmonton, Canada, July 2002.

[319] P. N. Tan, M. Steinbach, V. Kumar, S. Klooster, C. Potter, and A. Torregrosa. Finding Spatio-Temporal Patterns in Earth Science Data. In *KDD 2001 Workshop on Temporal Data Mining*, San Francisco, CA, 2001.

[320] H. Toivonen. Sampling Large Databases for Association Rules. In *Proc. of the 22nd VLDB Conf.*, pages 134–145, Bombay, India, 1996.

[321] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and Grouping Discovered Association Rules. In *ECML-95 Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, pages 47 – 52, Heraklion, Greece, April 1995.

[322] S. Tsur, J. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, and A. Rosenthal. Query Flocks: A Generalization of Association Rule Mining. In *Proc. of 1998 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 1–12, Seattle, WA, June 1998.

[323] A. Tung, H. J. Lu, J. Han, and L. Feng. Breaking the Barrier of Transactions: Mining Inter-Transaction Association Rules. In *Proc. of the 5th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 297–301, San Diego, CA, August 1999.

[324] K. Wang, Y. He, and J. Han. Mining Frequent Itemsets Using Support Constraints. In *Proc. of the 26th VLDB Conf.*, pages 43–52, Cairo, Egypt, September 2000.

[325] K. Wang, S. H. Tay, and B. Liu. Interestingness-Based Interval Merger for Numeric Association Rules. In *Proc. of the 4th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 121–128, New York, NY, August 1998.

[326] G. I. Webb. Preliminary investigations into statistically valid exploratory rule discovery. In *Proc. of the Australasian Data Mining Workshop (AusDM03)*, Canberra, Australia, December 2003.

[327] H. Xiong, X. He, C. Ding, Y. Zhang, V. Kumar, and S. R. Holbrook. Identification of Functional Modules in Protein Complexes via Hyperclique Pattern Discovery. In *Proc. of the Pacific Symposium on Biocomputing, (PSB 2005)*, Maui, January 2005.

[328] H. Xiong, S. Shekhar, P. N. Tan, and V. Kumar. Exploiting a Support-based Upper Bound of Pearson's Correlation Coefficient for Efficiently Identifying Strongly Correlated Pairs. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 334–343, Seattle, WA, August 2004.

[329] H. Xiong, M. Steinbach, P. N. Tan, and V. Kumar. HICAP: Hierarchial Clustering with Pattern Preservation. In *Proc. of the SIAM Intl. Conf. on Data Mining*, pages 279–290, Orlando, FL, April 2004.

[330] H. Xiong, P. N. Tan, and V. Kumar. Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution. In *Proc. of the 2003 IEEE Intl. Conf. on Data Mining*, pages 387–394, Melbourne, FL, 2003.

[331] X. Yan and J. Han. gSpan: Graph-based Substructure Pattern Mining. In *Proc. of the 2002 IEEE Intl. Conf. on Data Mining*, pages 721–724, Maebashi City, Japan, December 2002.

[332] C. Yang, U. M. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proc. of the 7th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 194–203, San Francisco, CA, August 2001.

[333] M. J. Zaki. Parallel and Distributed Association Mining: A Survey. *IEEE Concurrency, special issue on Parallel Mechanisms for Data Mining*, 7(4):14–25, December 1999.

[334] M. J. Zaki. Generating Non-Redundant Association Rules. In *Proc. of the 6th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 34–43, Boston, MA, August 2000.

[335] M. J. Zaki. Efficiently mining frequent trees in a forest. In *Proc. of the 8th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 71–80, Edmonton, Canada, July 2002.

[336] M. J. Zaki and M. Orihara. Theoretical foundations of association rules. In *Proc. of the 1998 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Seattle, WA, June 1998.

[337] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *Proc. of the 3rd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 283–286, Newport Beach, CA, August 1997.

[338] H. Zhang, B. Padmanabhan, and A. Tuzhilin. On the Discovery of Significant Statistical Quantitative Rules. In *Proc. of the 10th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 374–383, Seattle, WA, August 2004.

[339] Z. Zhang, Y. Lu, and B. Zhang. An Effective Partioning-Combining Algorithm for Discovering Quantitative Association Rules. In *Proc. of the 1st Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, Singapore, 1997.

[340] N. Zhong, Y. Y. Yao, and S. Ohsuga. Peculiarity Oriented Multi-database Mining. In *Proc. of the 3rd European Conf. of Principles and Practice of Knowledge Discovery in Databases*, pages 136–146, Prague, Czech Republic, 1999.

6.10   Exercises

1. For each of the following questions, provide an example of an association rule from the market basket domain that satisfies the following conditions. Also, describe whether such rules are subjectively interesting.

   (a) A rule that has high support and high confidence.
   (b) A rule that has reasonably high support but low confidence.
   (c) A rule that has low support and low confidence.
   (d) A rule that has low support and high confidence.

2. Consider the data set shown in Table 6.22.

Table 6.22. Example of market basket transactions.

Customer ID	Transaction ID	Items Bought
1	0001	$\{a, d, e\}$
1	0024	$\{a, b, c, e\}$
2	0012	$\{a, b, d, e\}$
2	0031	$\{a, c, d, e\}$
3	0015	$\{b, c, e\}$
3	0022	$\{b, d, e\}$
4	0029	$\{c, d\}$
4	0040	$\{a, b, c\}$
5	0033	$\{a, d, e\}$
5	0038	$\{a, b, e\}$

   (a) Compute the support for itemsets $\{e\}$, $\{b, d\}$, and $\{b, d, e\}$ by treating each transaction ID as a market basket.

   (b) Use the results in part (a) to compute the confidence for the association rules $\{b, d\} \longrightarrow \{e\}$ and $\{e\} \longrightarrow \{b, d\}$. Is confidence a symmetric measure?

   (c) Repeat part (a) by treating each customer ID as a market basket. Each item should be treated as a binary variable (1 if an item appears in at least one transaction bought by the customer, and 0 otherwise.)

   (d) Use the results in part (c) to compute the confidence for the association rules $\{b, d\} \longrightarrow \{e\}$ and $\{e\} \longrightarrow \{b, d\}$.

   (e) Suppose s_1 and c_1 are the support and confidence values of an association rule r when treating each transaction ID as a market basket. Also, let s_2 and c_2 be the support and confidence values of r when treating each customer ID as a market basket. Discuss whether there are any relationships between s_1 and s_2 or c_1 and c_2.

3. (a) What is the confidence for the rules $\emptyset \longrightarrow A$ and $A \longrightarrow \emptyset$?

   (b) Let c_1, c_2, and c_3 be the confidence values of the rules $\{p\} \longrightarrow \{q\}$, $\{p\} \longrightarrow \{q, r\}$, and $\{p, r\} \longrightarrow \{q\}$, respectively. If we assume that c_1, c_2, and c_3 have different values, what are the possible relationships that may exist among c_1, c_2, and c_3? Which rule has the lowest confidence?

   (c) Repeat the analysis in part (b) assuming that the rules have identical support. Which rule has the highest confidence?

   (d) Transitivity: Suppose the confidence of the rules $A \longrightarrow B$ and $B \longrightarrow C$ are larger than some threshold, $minconf$. Is it possible that $A \longrightarrow C$ has a confidence less than $minconf$?

4. For each of the following measures, determine whether it is monotone, anti-monotone, or non-monotone (i.e., neither monotone nor anti-monotone).

   Example: Support, $s = \frac{\sigma(X)}{|T|}$ is anti-monotone because $s(X) \geq s(Y)$ whenever $X \subset Y$.

   (a) A characteristic rule is a rule of the form $\{p\} \longrightarrow \{q_1, q_2, \ldots, q_n\}$, where the rule antecedent contains only a single item. An itemset of size k can produce up to k characteristic rules. Let ζ be the minimum confidence of all characteristic rules generated from a given itemset:

   $$\zeta(\{p_1, p_2, \ldots, p_k\}) \quad = \quad \min \left[c(\{p_1\} \longrightarrow \{p_2, p_3, \ldots, p_k\}), \ldots \right.$$
   $$\left. c(\{p_k\} \longrightarrow \{p_1, p_3 \ldots, p_{k-1}\}) \right]$$

   Is ζ monotone, anti-monotone, or non-monotone?

   (b) A discriminant rule is a rule of the form $\{p_1, p_2, \ldots, p_n\} \longrightarrow \{q\}$, where the rule consequent contains only a single item. An itemset of size k can produce up to k discriminant rules. Let η be the minimum confidence of all discriminant rules generated from a given itemset:

   $$\eta(\{p_1, p_2, \ldots, p_k\}) \quad = \quad \min \left[c(\{p_2, p_3, \ldots, p_k\} \longrightarrow \{p_1\}), \ldots \right.$$
   $$\left. c(\{p_1, p_2, \ldots p_{k-1}\} \longrightarrow \{p_k\}) \right]$$

   Is η monotone, anti-monotone, or non-monotone?

   (c) Repeat the analysis in parts (a) and (b) by replacing the min function with a max function.

5. Prove Equation 6.3. (Hint: First, count the number of ways to create an itemset that forms the left hand side of the rule. Next, for each size k itemset selected for the left-hand side, count the number of ways to choose the remaining $d - k$ items to form the right-hand side of the rule.)

Table 6.23. Market basket transactions.

Transaction ID	Items Bought
1	{Milk, Beer, Diapers}
2	{Bread, Butter, Milk}
3	{Milk, Diapers, Cookies}
4	{Bread, Butter, Cookies}
5	{Beer, Cookies, Diapers}
6	{Milk, Diapers, Bread, Butter}
7	{Bread, Butter, Diapers}
8	{Beer, Diapers}
9	{Milk, Diapers, Bread, Butter}
10	{Beer, Cookies}

6. Consider the market basket transactions shown in Table 6.23.

   (a) What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?

   (b) What is the maximum size of frequent itemsets that can be extracted (assuming $minsup > 0$)?

   (c) Write an expression for the maximum number of size-3 itemsets that can be derived from this data set.

   (d) Find an itemset (of size 2 or larger) that has the largest support.

   (e) Find a pair of items, a and b, such that the rules $\{a\} \longrightarrow \{b\}$ and $\{b\} \longrightarrow \{a\}$ have the same confidence.

7. Consider the following set of frequent 3-itemsets:

   $$\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{3, 4, 5\}.$$

   Assume that there are only five items in the data set.

   (a) List all candidate 4-itemsets obtained by a candidate generation procedure using the $F_{k-1} \times F_1$ merging strategy.

   (b) List all candidate 4-itemsets obtained by the candidate generation procedure in *Apriori*.

   (c) List all candidate 4-itemsets that survive the candidate pruning step of the *Apriori* algorithm.

8. The *Apriori* algorithm uses a generate-and-count strategy for deriving frequent itemsets. Candidate itemsets of size $k + 1$ are created by joining a pair of frequent itemsets of size k (this is known as the candidate generation step). A candidate is discarded if any one of its subsets is found to be infrequent during the candidate pruning step. Suppose the *Apriori* algorithm is applied to the

Table 6.24. Example of market basket transactions.

Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$

data set shown in Table 6.24 with $minsup = 30\%$, i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

(a) Draw an itemset lattice representing the data set given in Table 6.24. Label each node in the lattice with the following letter(s):

- **N**: If the itemset is not considered to be a candidate itemset by the *Apriori* algorithm. There are two reasons for an itemset not to be considered as a candidate itemset: (1) it is not generated at all during the candidate generation step, or (2) it is generated during the candidate generation step but is subsequently removed during the candidate pruning step because one of its subsets is found to be infrequent.
- **F**: If the candidate itemset is found to be frequent by the *Apriori* algorithm.
- **I**: If the candidate itemset is found to be infrequent after support counting.

(b) What is the percentage of frequent itemsets (with respect to all itemsets in the lattice)?

(c) What is the pruning ratio of the *Apriori* algorithm on this data set? (Pruning ratio is defined as the percentage of itemsets not considered to be a candidate because (1) they are not generated during candidate generation or (2) they are pruned during the candidate pruning step.)

(d) What is the false alarm rate (i.e, percentage of candidate itemsets that are found to be infrequent after performing support counting)?

9. The *Apriori* algorithm uses a hash tree data structure to efficiently count the support of candidate itemsets. Consider the hash tree for candidate 3-itemsets shown in Figure 6.32.
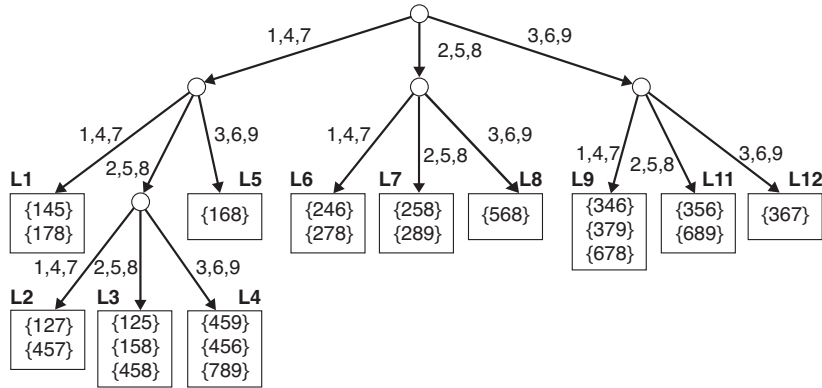
Figure 6.32. An example of a hash tree structure.

(a) Given a transaction that contains items $\{1, 3, 4, 5, 8\}$, which of the hash tree leaf nodes will be visited when finding the candidates of the transaction?

(b) Use the visited leaf nodes in part (b) to determine the candidate itemsets that are contained in the transaction $\{1, 3, 4, 5, 8\}$.

10. Consider the following set of candidate 3-itemsets:

$$\{1, 2, 3\}, \{1, 2, 6\}, \{1, 3, 4\}, \{2, 3, 4\}, \{2, 4, 5\}, \{3, 4, 6\}, \{4, 5, 6\}$$

(a) Construct a hash tree for the above candidate 3-itemsets. Assume the tree uses a hash function where all odd-numbered items are hashed to the left child of a node, while the even-numbered items are hashed to the right child. A candidate k-itemset is inserted into the tree by hashing on each successive item in the candidate and then following the appropriate branch of the tree according to the hash value. Once a leaf node is reached, the candidate is inserted based on one of the following conditions:

Condition 1: If the depth of the leaf node is equal to k (the root is assumed to be at depth 0), then the candidate is inserted regardless of the number of itemsets already stored at the node.

Condition 2: If the depth of the leaf node is less than k, then the candidate can be inserted as long as the number of itemsets stored at the node is less than $maxsize$. Assume $maxsize = 2$ for this question.

Condition 3: If the depth of the leaf node is less than k and the number of itemsets stored at the node is equal to $maxsize$, then the leaf node is converted into an internal node. New leaf nodes are created as children of the old leaf node. Candidate itemsets previously stored
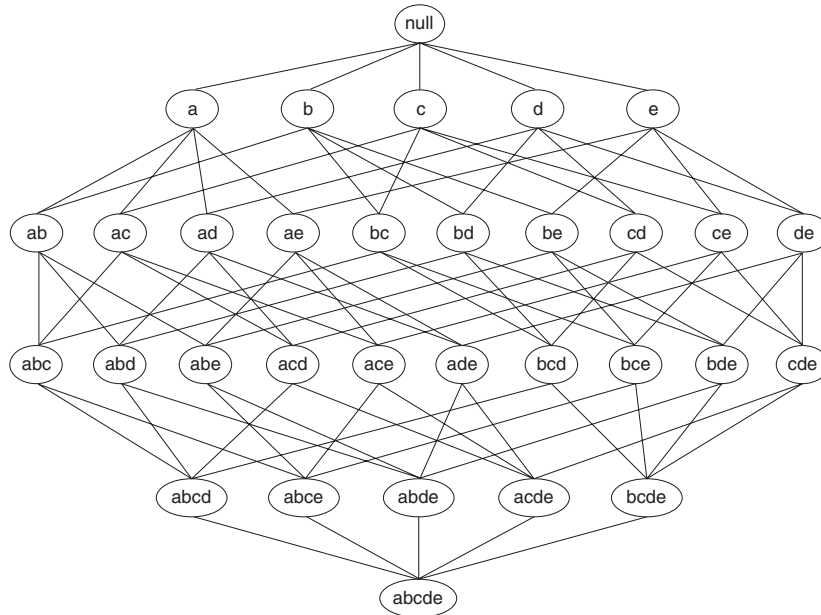
Figure 6.33. An itemset lattice

  in the old leaf node are distributed to the children based on their hash values. The new candidate is also hashed to its appropriate leaf node.

 (b) How many leaf nodes are there in the candidate hash tree? How many internal nodes are there?

 (c) Consider a transaction that contains the following items: $\{1, 2, 3, 5, 6\}$. Using the hash tree constructed in part (a), which leaf nodes will be checked against the transaction? What are the candidate 3-itemsets contained in the transaction?

11. Given the lattice structure shown in Figure 6.33 and the transactions given in Table 6.24, label each node with the following letter(s):

 • M if the node is a maximal frequent itemset,

 • C if it is a closed frequent itemset,

 • N if it is frequent but neither maximal nor closed, and

 • I if it is infrequent.

Assume that the support threshold is equal to 30%.

12. The original association rule mining formulation uses the support and confidence measures to prune uninteresting rules.

(a) Draw a contingency table for each of the following rules using the transactions shown in Table 6.25.

Table 6.25. Example of market basket transactions.

Transaction ID	Items Bought
1	$\{a, b, d, e\}$
2	$\{b, c, d\}$
3	$\{a, b, d, e\}$
4	$\{a, c, d, e\}$
5	$\{b, c, d, e\}$
6	$\{b, d, e\}$
7	$\{c, d\}$
8	$\{a, b, c\}$
9	$\{a, d, e\}$
10	$\{b, d\}$

Rules: $\{b\} \longrightarrow \{c\}$, $\{a\} \longrightarrow \{d\}$, $\{b\} \longrightarrow \{d\}$, $\{e\} \longrightarrow \{c\}$, $\{c\} \longrightarrow \{a\}$.

(b) Use the contingency tables in part (a) to compute and rank the rules in decreasing order according to the following measures.

   i. Support.

   ii. Confidence.

   iii. Interest$(X \longrightarrow Y) = \frac{P(X,Y)}{P(X)} P(Y)$.

   iv. IS$(X \longrightarrow Y) = \frac{P(X,Y)}{\sqrt{P(X)P(Y)}}$.

   v. Klosgen$(X \longrightarrow Y) = \sqrt{P(X,Y)} \times (P(Y|X) - P(Y))$, where $P(Y|X) = \frac{P(X,Y)}{P(X)}$.

   vi. Odds ratio$(X \longrightarrow Y) = \frac{P(X,Y)P(\overline{X},\overline{Y})}{P(X,\overline{Y})P(\overline{X},Y)}$.

13. Given the rankings you had obtained in Exercise 12, compute the correlation between the rankings of confidence and the other five measures. Which measure is most highly correlated with confidence? Which measure is least correlated with confidence?

14. Answer the following questions using the data sets shown in Figure 6.34. Note that each data set contains 1000 items and 10,000 transactions. Dark cells indicate the presence of items and white cells indicate the absence of items. We will apply the *Apriori* algorithm to extract frequent itemsets with *minsup* = 10% (i.e., itemsets must be contained in at least 1000 transactions)?

   (a) Which data set(s) will produce the most number of frequent itemsets?

(b) Which data set(s) will produce the fewest number of frequent itemsets?

(c) Which data set(s) will produce the longest frequent itemset?

(d) Which data set(s) will produce frequent itemsets with highest maximum support?

(e) Which data set(s) will produce frequent itemsets containing items with wide-varying support levels (i.e., items with mixed support, ranging from less than 20% to more than 70%).

15. (a) Prove that the ϕ coefficient is equal to 1 if and only if $f_{11} = f_{1+} = f_{+1}$.

(b) Show that if A and B are independent, then $P(A, B) \times P(A, \overline{B}) = P(A, \overline{B}) \times P(\overline{A}, B)$.

(c) Show that Yule's Q and Y coefficients

$$Q = \left[\frac{f_{11}f_{00} - f_{10}f_{01}}{f_{11}f_{00} + f_{10}f_{01}} \right]$$

$$Y = \left[\frac{\sqrt{f_{11}f_{00}} - \sqrt{f_{10}f_{01}}}{\sqrt{f_{11}f_{00}} + \sqrt{f_{10}f_{01}}} \right]$$

are normalized versions of the odds ratio.

(d) Write a simplified expression for the value of each measure shown in Tables 6.11 and 6.12 when the variables are statistically independent.

16. Consider the interestingness measure, $M = \frac{P(B|A) - P(B)}{1 - P(B)}$, for an association rule $A \longrightarrow B$.

(a) What is the range of this measure? When does the measure attain its maximum and minimum values?

(b) How does M behave when $P(A, B)$ is increased while $P(A)$ and $P(B)$ remain unchanged?

(c) How does M behave when $P(A)$ is increased while $P(A, B)$ and $P(B)$ remain unchanged?

(d) How does M behave when $P(B)$ is increased while $P(A, B)$ and $P(A)$ remain unchanged?

(e) Is the measure symmetric under variable permutation?

(f) What is the value of the measure when A and B are statistically independent?

(g) Is the measure null-invariant?

(h) Does the measure remain invariant under row or column scaling operations?

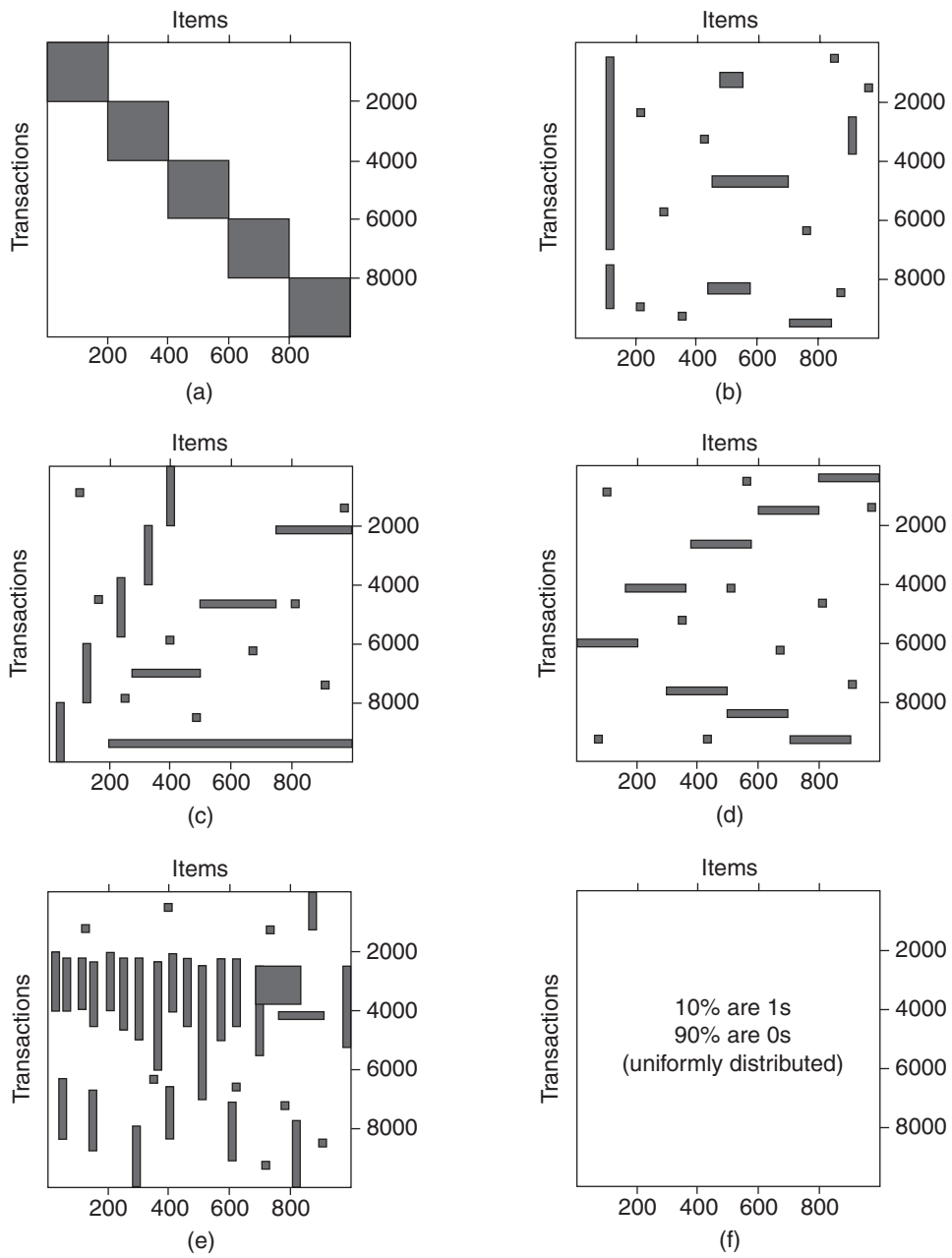(i) How does the measure behave under the inversion operation?

Figure 6.34. Figures for Exercise 14.

17. Suppose we have market basket data consisting of 100 transactions and 20 items. If the support for item a is 25%, the support for item b is 90% and the support for itemset $\{a, b\}$ is 20%. Let the support and confidence thresholds be 10% and 60%, respectively.

    (a) Compute the confidence of the association rule $\{a\} \rightarrow \{b\}$. Is the rule interesting according to the confidence measure?

    (b) Compute the interest measure for the association pattern $\{a, b\}$. Describe the nature of the relationship between item a and item b in terms of the interest measure.

    (c) What conclusions can you draw from the results of parts (a) and (b)?

    (d) Prove that if the confidence of the rule $\{a\} \longrightarrow \{b\}$ is less than the support of $\{b\}$, then:

    i. $c(\{\overline{a}\} \longrightarrow \{b\}) > c(\{\overline{a}\} \longrightarrow \{b\})$,
    ii. $c(\{\overline{a}\} \longrightarrow \{b\}) > s(\{b\})$,

    where $c(\cdot)$ denote the rule confidence and $s(\cdot)$ denote the support of an itemset.

18. Table 6.26 shows a $2 \times 2 \times 2$ contingency table for the binary variables A and B at different values of the control variable C.

Table 6.26. A Contingency Table.

			A	
			1	0
C = 0	B	1	0	15
		0	15	30
C = 1	B	1	5	0
		0	0	15

    (a) Compute the ϕ coefficient for A and B when $C = 0$, $C = 1$, and $C = 0$ or 1. Note that $\phi(\{A, B\}) = \frac{P(A,B) - P(A)P(B)}{\sqrt{P(A)P(B)(1 - P(A))(1 - P(B))}}$.

    (b) What conclusions can you draw from the above result?

19. Consider the contingency tables shown in Table 6.27.

    (a) For table I, compute support, the interest measure, and the ϕ correlation coefficient for the association pattern $\{A, B\}$. Also, compute the confidence of rules $A \rightarrow B$ and $B \rightarrow A$.

Table 6.27. Contingency tables for Exercise 19.

	B	$\overline{B}$
A	9	1
$\overline{A}$	1	89

	B	$\overline{B}$
A	89	1
$\overline{A}$	1	9

(a) Table I.   (b) Table II.

(b) For table II, compute support, the interest measure, and the ϕ correlation coefficient for the association pattern $\{A, B\}$. Also, compute the confidence of rules $A \rightarrow B$ and $B \rightarrow A$.

(c) What conclusions can you draw from the results of (a) and (b)?

20. Consider the relationship between customers who buy high-definition televisions and exercise machines as shown in Tables 6.19 and 6.20.

(a) Compute the odds ratios for both tables.

(b) Compute the ϕ-coefficient for both tables.

(c) Compute the interest factor for both tables.

For each of the measures given above, describe how the direction of association changes when data is pooled together instead of being stratified.