5

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 5.1 gives 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 sets of items present in many transactions, which are known as **frequent itemsets**,

Table 5.1. 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}

or **association rules**, that represent relationships between two itemsets. For example, the following rule can be extracted from the data set shown in Table 5.1:

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

The rule suggests a relationship between the sale of diapers and beer because many customers who buy diapers also buy beer. Retailers can use these types 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 data from other application domains such as bioinformatics, medical diagnosis, web mining, and scientific data analysis. In the analysis of Earth science data, for example, 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 may be spurious (happen simply by chance) and even for non-spurious patterns, some are more interesting than others. 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 help prevent the generation of spurious results and to rank the patterns in terms of some interestingness measure.

5.1 Preliminaries

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 5.2, 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

Table 5.2. 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

more important than its absence, an item is an **asymmetric** binary variable. This representation is a simplistic view of real market basket data because it ignores 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 6.

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.

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 5.2 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 $|\cdot|$ denotes the number of elements in a set. In the data set shown in Table 5.2, the support count for {`Beer`, `Diapers`, `Milk`} is equal to two because there are only two transactions that contain all three items.

Often, the property of interest is the support, which is fraction of transactions in which an itemset occurs:

$$s(X) = \sigma(X)/N.$$

An itemset X is called frequent if $s(X)$ is greater than some user-defined threshold, *minsup*.

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

$$\text{Support, } s(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{N}; \tag{5.1}$$

$$\text{Confidence, } c(X \longrightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}. \tag{5.2}$$

Example 5.1. Consider the rule {`Milk`, `Diapers`} $\longrightarrow$ {`Beer`}. Because 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 $2/3 = 0.67$. ■

what is a good thresh,in

Why Use Support and Confidence? Support is an important measure because a rule that has very low support might occur simply by chance. Also, from a business perspective a low support rule is unlikely to be interesting because it might not be profitable to promote items that customers seldom buy together (with the exception of the situation described in Section 5.8). For these reasons, we are interested in finding rules whose support is greater than some user-defined threshold. As will be shown in Section 5.2.1, 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 can sometimes suggest a strong co-occurrence relationship between items in the antecedent and consequent of the rule. Causality, on the other hand, requires knowledge about which attributes in the data capture cause and effect, and typically involves relationships occurring over time (e.g., greenhouse gas emissions lead to global warming). See Section 5.7.1 for additional discussion.

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

Definition 5.1 (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, assuming that neither the left nor the right-hand side of the rule is an empty set, the total number of possible rules, R, extracted from a data set that contains d items is

$$R = 3^d - 2^{d+1} + 1. \tag{5.3}$$

The proof for this equation is left as an exercise to the readers (see Exercise 5 on page 440). Even for the small data set shown in Table 5.1, this approach requires us to compute the support and confidence for $3^6 - 2^7 + 1 = 602$ rules. More than 80% of the rules are discarded after applying $minsup = 20\%$ and $minconf = 50\%$, thus wasting most of the computations. 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 5.1, notice that the support of a rule $X \longrightarrow Y$ is the same as 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}:

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

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.

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 5.2 and 5.3, respectively.

5.2 Frequent Itemset Generation

A lattice structure can be used to enumerate the list of all possible itemsets. Figure 5.1 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 5.2. 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. **Transaction width** is the number of items present in a transaction.

There are three main approaches for reducing 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 5.2.4 and 5.6, respectively.

3. **Reduce the number of transactions (N).** As the size of candidate itemsets increases, fewer transactions will be supported by the itemsets.

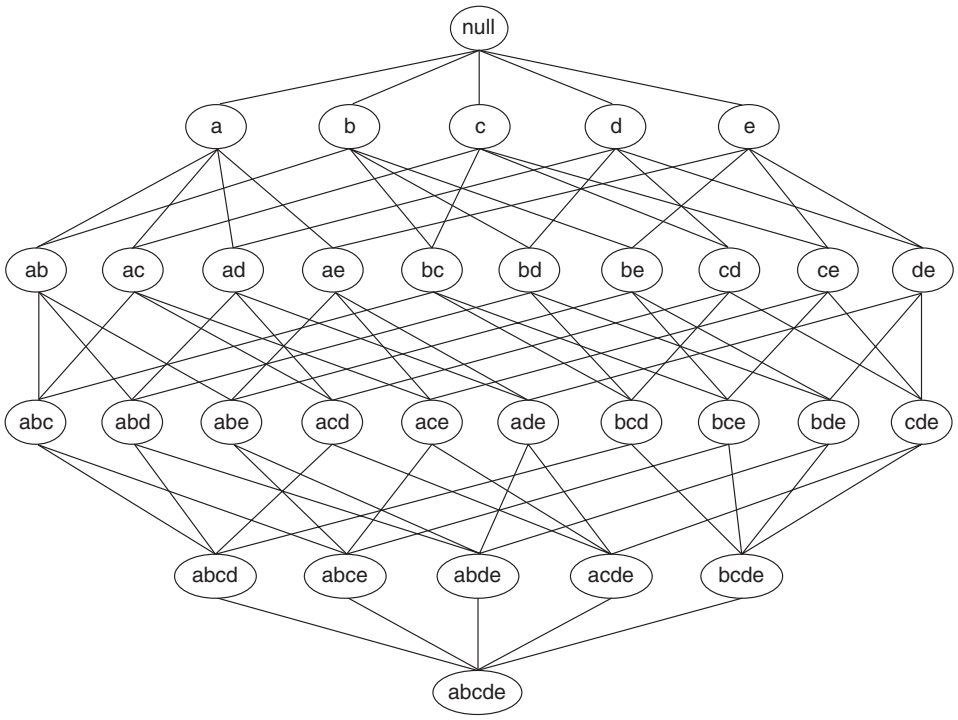


Figure 5.1. An itemset lattice.

For instance, since the width of the first transaction in Table 5.1 is 2, it would be advantageous to remove this transaction before searching for frequent itemsets of size 3 and larger. Algorithms that employ such a strategy are discussed in the Bibliographic Notes.

5.2.1 The *Apriori* Principle

This section describes how the support measure can be used 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 5.1 (*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 5.3. Suppose $\{c, d, e\}$ is a frequent itemset. Clearly, any transaction that contains $\{c, d, e\}$ must also contain its subsets, $\{c, d\}$, $\{c, e\}$,

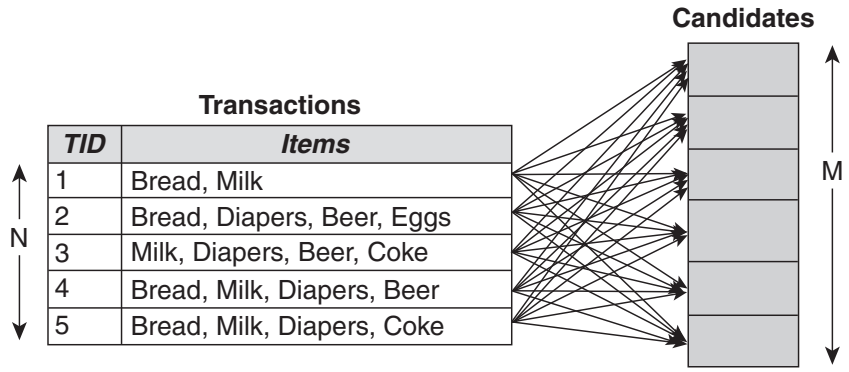


Figure 5.2. Counting the support of candidate itemsets.

$\{d, e\}$, $\{c\}$, $\{d\}$, 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.

Conversely, if an itemset such as $\{a, b\}$ is infrequent, then all of its supersets must be infrequent too. As illustrated in Figure 5.4, 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 5.2 (Anti-monotone Property)**.** A measure f possesses the anti-monotone property if for every itemset X that is a proper subset of itemset Y, i.e. $X \subset Y$, we have $f(Y) \leq f(X)$.

More generally, a large number of measures–see Section 5.7.1–can be applied to itemsets to evaluate various properties of itemsets. As will be shown in the next section, any measure that has the anti-monotone property can be incorporated directly into an itemset mining algorithm to effectively prune the exponential search space of candidate itemsets.

5.2.2 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 5.5 provides a high-level illustration of the frequent itemset generation part of the *Apriori* algorithm for the transactions shown in

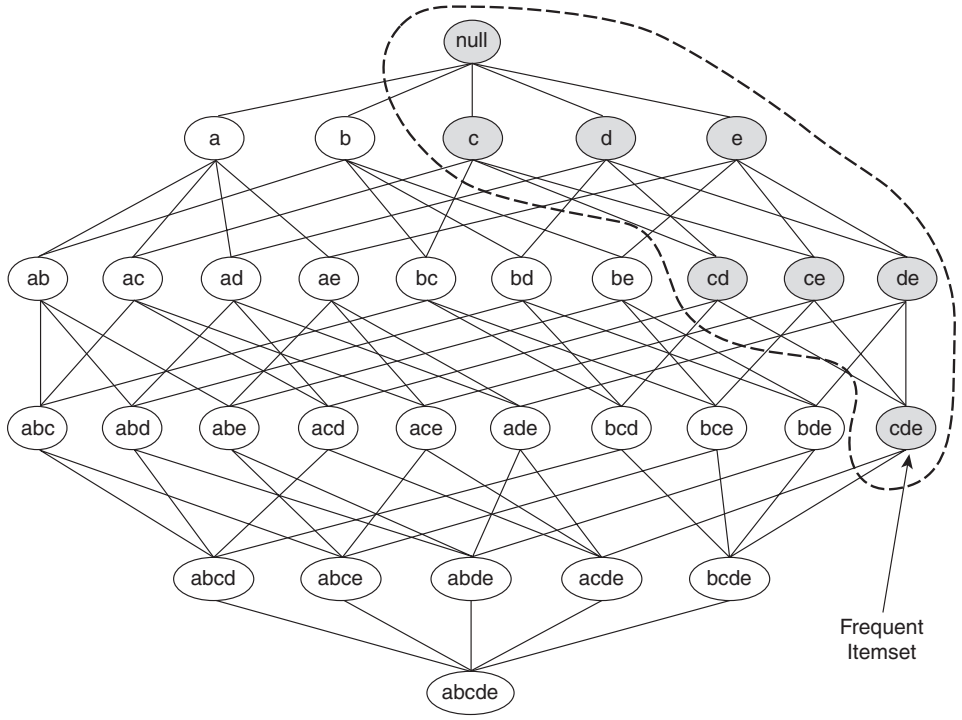


Figure 5.3. An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

Table 5.1. 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`,

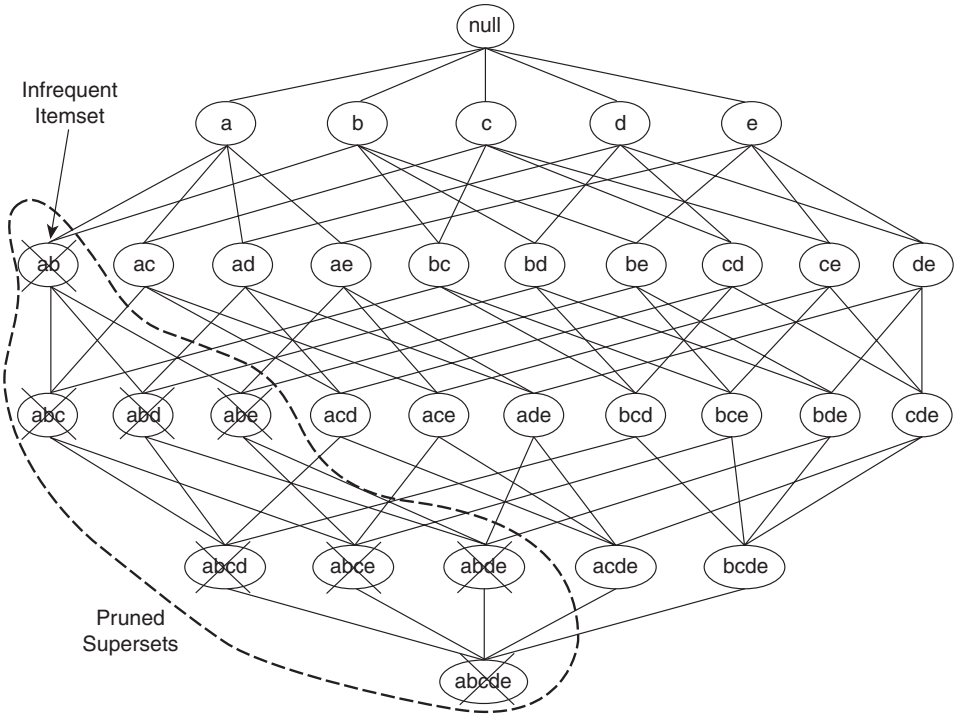


Figure 5.4. An illustration of support-based pruning. If $\{a, b\}$ is infrequent, then all supersets of $\{a, b\}$ are infrequent.

`Diapers`, `Milk`}. However, even though the subsets of {`Bread`, `Diapers`, `Milk`} are frequent, the itemset itself is not.

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.

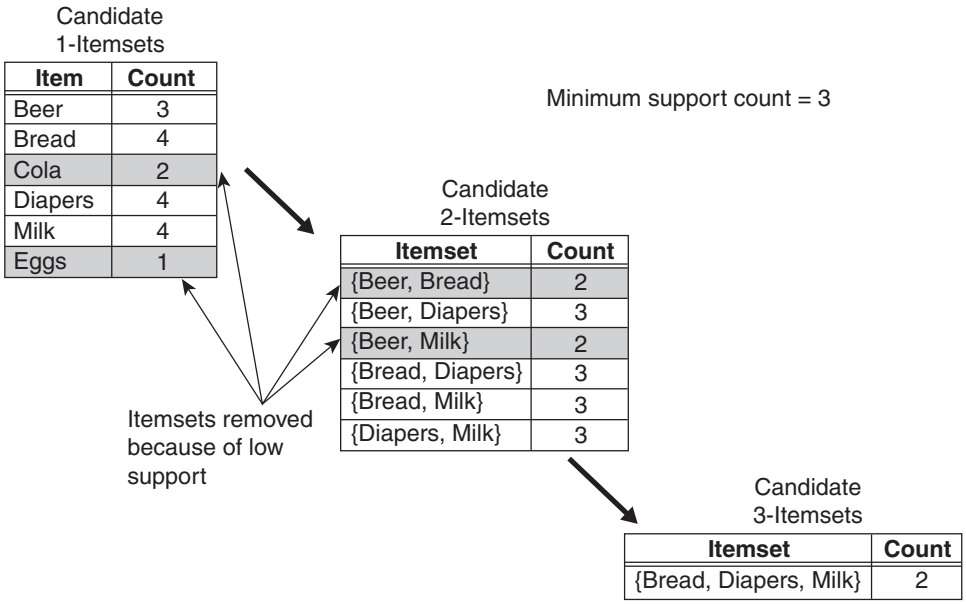


Figure 5.5. Illustration of frequent itemset generation using the *Apriori* algorithm.

The pseudocode for the frequent itemset generation part of the *Apriori* algorithm is shown in Algorithm 5.1. 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 and prune unnecessary candidates that are guaranteed to be infrequent given the frequent $(k-1)$-itemsets found in the previous iteration (steps 5 and 6). Candidate generation and pruning is implemented using the functions candidate-gen and candidate-prune, which are described in Section 5.2.3.

- To count the support of the candidates, the algorithm needs to make an additional pass over the data set (steps 7–12). 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 5.2.4.

- After counting their supports, the algorithm eliminates all candidate itemsets whose support counts are less than $N \times minsup$ (step 13).
- The algorithm terminates when there are no new frequent itemsets generated, i.e., $F_k = \emptyset$ (step 14).

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 (level), 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.

5.2.3 Candidate Generation and Pruning

The candidate-gen and candidate-prune functions shown in Steps 5 and 6 of Algorithm 5.1 generate candidate itemsets and prunes unnecessary ones by performing the following two operations, respectively:

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

Algorithm 5.1 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 = candidate-gen(F_{k-1}). {Generate candidate itemsets.}
6: C_k = candidate-prune(C_k, F_{k-1}). {Prune candidate itemsets.}
7: **for** each transaction $t \in T$ **do**
8: C_t = subset(C_k, t). {Identify all candidates that belong to t.}
9: **for** each candidate itemset $c \in C_t$ **do**
10: $\sigma(c) = \sigma(c) + 1$. {Increment support count.}
11: **end for**
12: **end for**
13: $F_k = \{ c \mid c \in C_k \wedge \sigma(c) \geq N \times minsup\}$. {Extract the frequent k-itemsets.}
14: **until** $F_k = \emptyset$
15: Result = $\bigcup F_k$.

2. **Candidate Pruning.** This operation eliminates some of the candidate k-itemsets using support-based pruning, i.e. by removing k-itemsets whose subsets are known to be infrequent in previous iterations. Note that this pruning is done without computing the actual support of these k-itemsets (which could have required comparing them against each transaction).

Candidate Generation

In principle, there are many ways to generate candidate itemsets. An effective candidate generation procedure must be complete and non-redundant. A candidate generation procedure is said to be *complete* if it does not omit any frequent itemsets. To ensure completeness, the set of candidate itemsets must subsume the set of all frequent itemsets, i.e., $\forall k : F_k \subseteq C_k$. A candidate generation procedure is *non-redundant* if it does 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. Also, an effective candidate generation procedure should avoid generating too many unnecessary candidates. A candidate itemset is unnecessary if at least one of its subsets is infrequent, and thus, eliminated in the candidate pruning step.

Next, we will briefly describe several candidate generation procedures, including the one used by the candidate-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 whose subsets are infrequent (see Figure 5.6). 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.

$\mathbf{F}_{k-1} \times \mathbf{F}_1$ Method An alternative method for candidate generation is to extend each frequent $(k-1)$-itemset with frequent items that are not part of the $(k-1)$-itemset. Figure 5.7 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`}.

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

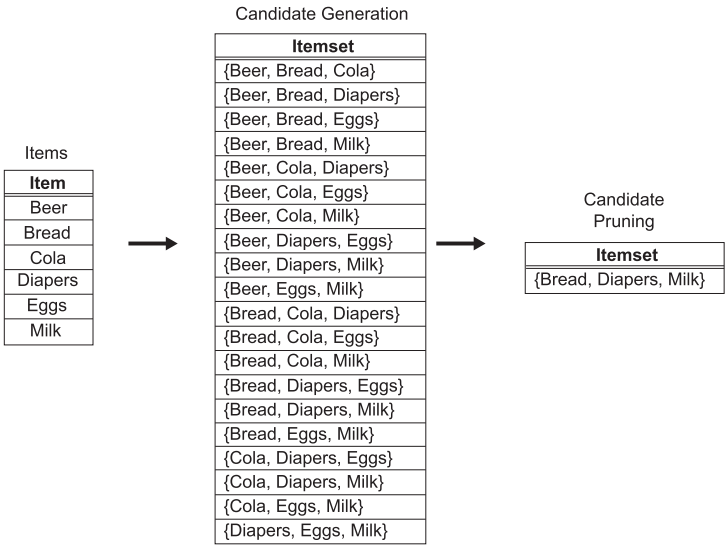


Figure 5.6. A brute-force method for generating candidate 3-itemsets.

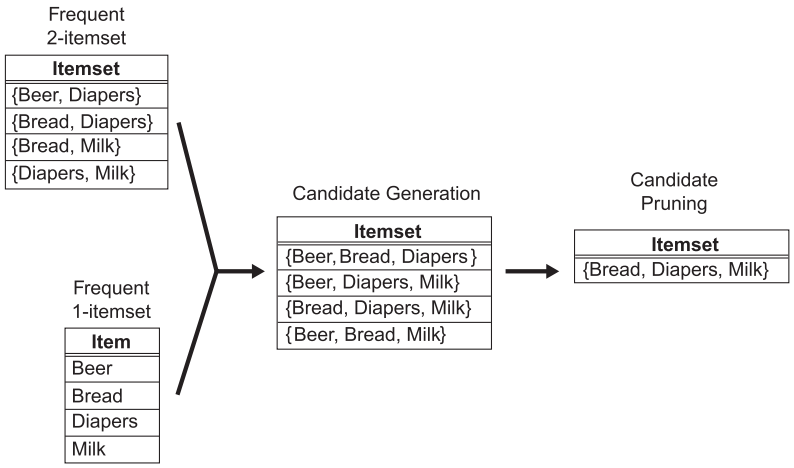


Figure 5.7. Generating and pruning candidate k-itemsets by merging a frequent $(k-1)$-itemset with a frequent item. Note that some of the candidates are unnecessary because their subsets are infrequent.

k-itemsets are part of the candidate k-itemsets generated by this procedure. Figure 5.7 shows that the $\mathbf{F}_{k-1} \times \mathbf{F}_1$ candidate generation method only produces four candidate 3-itemsets, instead of the $\binom{6}{3} = 20$ itemsets produced by the brute-force method. The $\mathbf{F}_{k-1} \times \mathbf{F}_1$ method generates lower number

of candidates because every candidate is guaranteed to contain at least one frequent $(k-1)$-itemset. While this procedure is a substantial improvement over the brute-force method, it can still produce a large number of unnecessary candidates, as the remaining subsets of a candidate itemset can still be infrequent.

Note that the approach discussed above 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. For example, itemsets such as {`Bread`, `Diapers`}, {`Bread`, `Diapers`, `Milk`}, and {`Diapers`, `Milk`} follow lexicographic order as the items within every itemset are arranged alphabetically. 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} because `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. Every candidate k-itemset is thus generated exactly once, by merging the lexicographically largest item with the remaining $k-1$ items in the itemset. If the $\mathbf{F}_{k-1} \times \mathbf{F}_1$ method is used in conjunction with lexicographic ordering, then only two candidate 3-itemsets will be produced in the example illustrated in Figure 5.7. {`Beer`, `Bread`, `Diapers`} and {`Beer`, `Bread`, `Milk`} will not be generated because {`Beer`, `Bread`} is not a frequent 2-itemset.

$\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ Method This candidate generation procedure, which is used in the candidate-gen function of the *Apriori* algorithm, merges a pair of frequent $(k-1)$-itemsets only if their first $k-2$ items, arranged in lexicographic order, 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, arranged lexicographically. A and B are merged if they satisfy the following conditions:

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

Note that in this case, $a_{k-1} \neq b_{k-1}$ because A and B are two distinct itemsets. The candidate k-itemset generated by merging A and B consists of the first $k-2$ common items followed by a_{k-1} and b_{k-1} in lexicographic order. This candidate generation procedure is complete, because for every lexicographically ordered frequent k-itemset, there exists two lexicographically

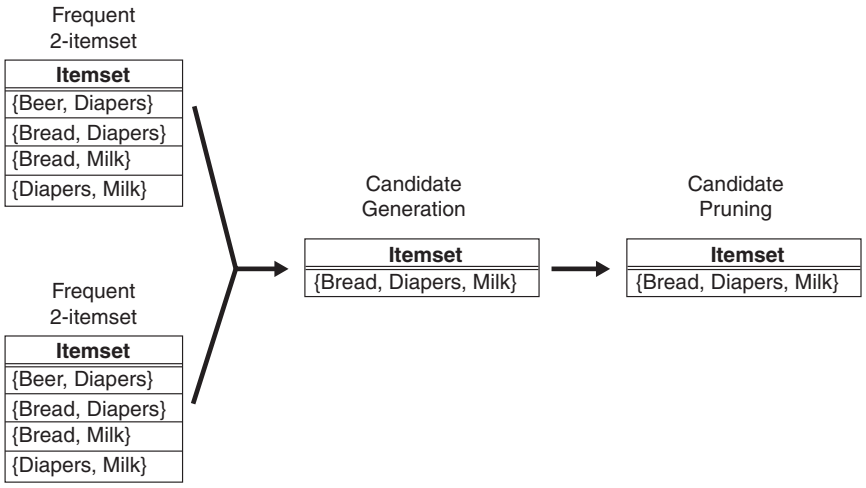


Figure 5.8. Generating and pruning candidate k-itemsets by merging pairs of frequent $(k-1)$-itemsets.

ordered frequent $(k-1)$-itemsets that have identical items in the first $k-2$ positions.

In Figure 5.8, 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. Also, if we order the frequent $(k-1)$-itemsets according to their lexicographic rank, itemsets with identical first $k-2$ items would take consecutive ranks. As a result, the $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ candidate generation method would consider merging a frequent itemset only with ones that occupy the next few ranks in the sorted list, thus saving some computations.

Figure 5.8 shows that the $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ candidate generation procedure results in only one candidate 3-itemset. This is a considerable reduction from the four candidate 3-itemsets generated by the $\mathbf{F}_{k-1} \times \mathbf{F}_1$ method. This is because the $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ method ensures that every candidate k-itemset contains at least two frequent $(k-1)$-itemsets, thus greatly reducing the number of candidates that are generated in this step.

Note that there can be multiple ways of merging two frequent $(k-1)$-itemsets in the $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ procedure, one of which is merging if their first

$k-2$ items are identical. An alternate approach could be to merge two frequent $(k-1)$-itemsets A and B if the last $k-2$ items of A are identical to the first $k-2$ itemsets of B. For example, {`Bread`, `Diapers`} and {`Diapers`, `Milk`} could be merged using this approach to generate the candidate 3-itemset {`Bread`, `Diapers`, `Milk`}. As we will see later, this alternate $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ procedure is useful in generating sequential patterns, which will be discussed in Chapter 6.

Candidate Pruning

To illustrate the candidate pruning operation for a candidate k-itemset, $X = \{i_1, i_2, \ldots, i_k\}$, consider its k proper subsets, $X - \{i_j\}$ $(\forall j = 1, 2, \ldots, k)$. If any of them are infrequent, then X is immediately pruned by using the *Apriori* principle. Note that we don't need to explicitly ensure that all subsets of X of size less than $k-1$ are frequent (see Exercise 7). This approach greatly reduces the number of candidate itemsets considered during support counting. For the brute-force candidate generation method, candidate pruning requires checking only k subsets of size $k-1$ for each candidate k-itemset. However, since the $\mathbf{F}_{k-1} \times \mathbf{F}_1$ candidate generation strategy ensures that at least one of the $(k-1)$-size subsets of every candidate k-itemset is frequent, we only need to check for the remaining $k-1$ subsets. Likewise, the $\mathbf{F}_{k-1} \times \mathbf{F}_{k-1}$ strategy requires examining only $k-2$ subsets of every candidate k-itemset, since two of its $(k-1)$-size subsets are already known to be frequent in the candidate generation step.

5.2.4 Support Counting

Support counting is the process of determining the frequency of occurrence for every candidate itemset that survives the candidate pruning step. Support counting is implemented in steps 6 through 11 of Algorithm 5.1. A brute-force approach for doing this is to compare each transaction against every candidate itemset (see Figure 5.2) and to update the support counts of candidates contained in a 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.

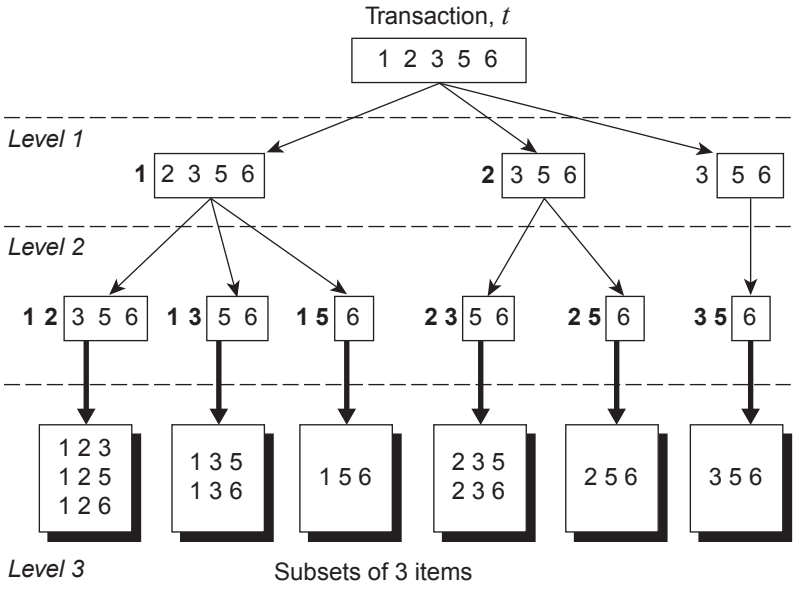


Figure 5.9. Enumerating subsets of three items from a transaction t.

Figure 5.9 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 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 tree structure depicted in Figure 5.9. 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 tree structure at Level 2 represents the number of ways to select the second item. For example, 1 2 $\boxed{3\ 5\ 6}$ corresponds to itemsets that begin with the prefix (1 2) and are followed by the items 3, 5, or 6. Finally, the prefix tree structure at Level 3 represents 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 tree structure shown in Figure 5.9 demonstrates how itemsets contained in a transaction can be systematically enumerated, i.e., by specifying

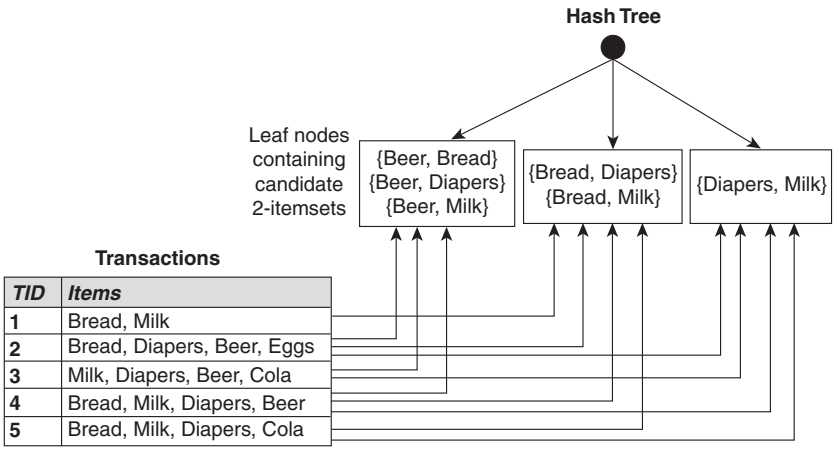


Figure 5.10. Counting the support of itemsets using hash structure.

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.

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 5.10.

Figure 5.11 shows an example of a hash tree structure. Each internal node of the tree uses the following hash function, $h(p) = (p - 1) \bmod 3$, where mode refers to the modulo (remainder) operator, 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 5.11 contains 15 candidate 3-itemsets, distributed across 9 leaf nodes.

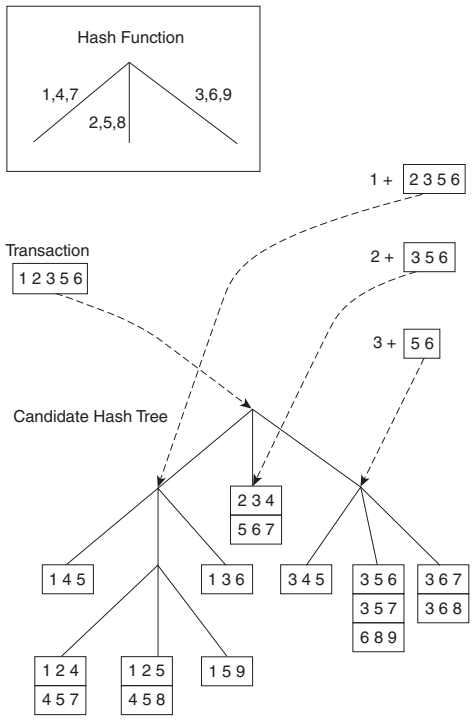


Figure 5.11. Hashing a transaction at the root node of a hash tree.

Consider the 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 tree structure shown in Figure 5.9. 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 item listed in the Level 2 tree structure shown in Figure 5.9. For example, after hashing on item 1 at the root node, items 2, 3, and 5 of the transaction are hashed. Based on the hash function, items 2 and 5 are hashed to the middle child, while item 3 is hashed to the right child, as shown in Figure 5.12. 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. Note that not all the leaf nodes are visited

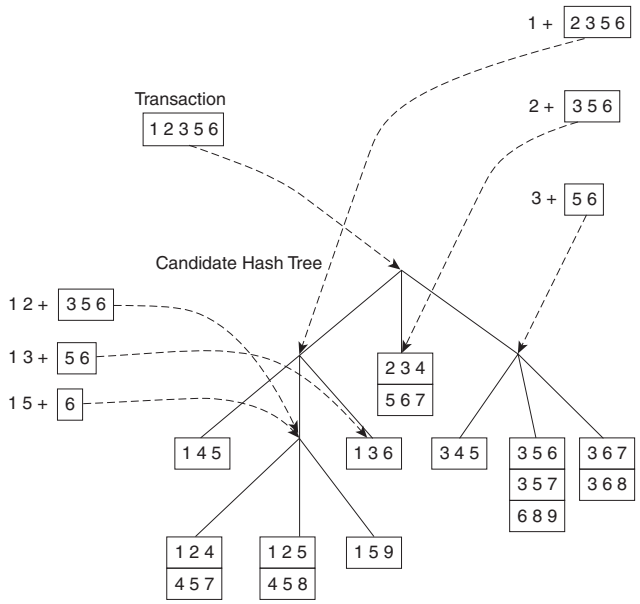


Figure 5.12. Subset operation on the leftmost subtree of the root of a candidate hash tree.

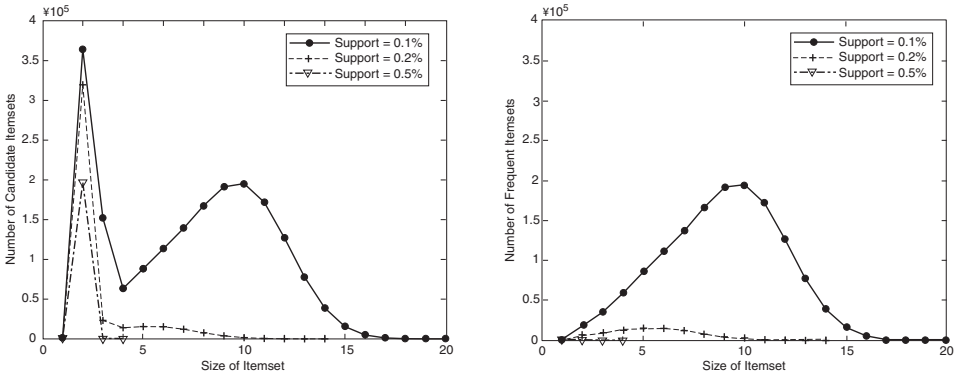
while traversing the hash tree, which helps in reducing the computational cost. In this example, 5 out of the 9 leaf nodes are visited and 9 out of the 15 itemsets are compared against the transaction.

5.2.5 Computational Complexity

The computational complexity of the *Apriori* algorithm, which includes both its runtime and storage, can be affected by the following factors.

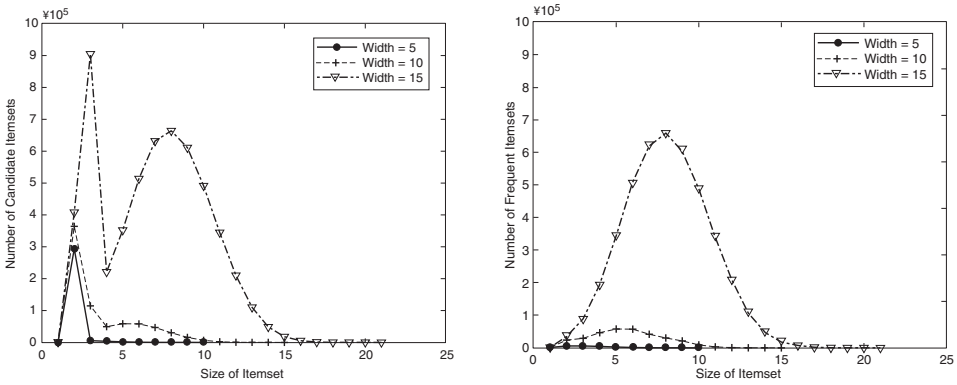
Support Threshold Lowering the support threshold often results in more itemsets being declared as frequent. This has an adverse effect on the computational complexity of the algorithm because more candidate itemsets must be generated and counted at every level, as shown in Figure 5.13. The maximum size of frequent itemsets also tends to increase with lower support thresholds. This increases the total number of iterations to be performed by the *Apriori* algorithm, further increasing the computational cost.

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 runtime and



(a) Number of candidate itemsets. (b) Number of frequent itemsets.

Figure 5.13. Effect of support threshold on the number of candidate and frequent itemsets obtained from a benchmark data set.



(a) Number of candidate itemsets. (b) Number of Frequent Itemsets.

Figure 5.14. Effect of average transaction width on the number of candidate and frequent itemsets obtained from a synthetic data set.

storage requirements will increase because of the larger number of candidate itemsets generated by the algorithm.

Number of Transactions Because the *Apriori* algorithm makes repeated passes over the transaction 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 average transaction width increases. As a result, more candidate itemsets must be examined during candidate generation and support counting, as illustrated in Figure 5.14. Second, as the transaction width increases, more 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. Every merging step can produce at most one viable candidate k-itemset, while in the worst-case, the algorithm must try to merge every pair of frequent $(k-1)$-itemsets found in the previous iteration. Therefore, the overall cost of merging frequent itemsets is

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

where w is the maximum transaction width. 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 $O\big(\sum_{k=2}^{w} k|C_k|\big)$. 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 $O\big(\sum_{k=2}^{w} k(k-2)|C_k|\big)$ time.

Support counting Each transaction of width $|t|$ 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.

5.3 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 5.2. Let $X = \{a, b, c\}$ be a frequent itemset. There are six candidate association rules that can be generated from X: $\{a, b\} \longrightarrow \{c\}$, $\{a, c\} \longrightarrow \{b\}$, $\{b, c\} \longrightarrow \{a\}$, $\{a\} \longrightarrow \{b, c\}$, $\{b\} \longrightarrow \{a, c\}$, and $\{c\} \longrightarrow \{a, b\}$. As each of their support is identical to the support for X, all the rules 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.

5.3.1 Confidence-Based Pruning

Confidence does not show the anti-monotone property in the same way as the support measure. 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 439). Nevertheless, if we compare rules generated from the same frequent itemset Y, the following theorem holds for the confidence measure.

Theorem 5.2. *Let Y be an itemset and X is a subset of Y. If a rule $X \longrightarrow Y - X$ does not satisfy the confidence threshold, then any rule $\tilde{X} \longrightarrow Y - \tilde{X}$, where $\tilde{X}$ is a subset of X, must not satisfy the confidence threshold as well.*

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

5.3.2 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 5.15 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 5.2, the 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 5.2 and 5.3. Note the similarity between the `ap-genrules` procedure given in Algorithm 5.3 and the frequent itemset generation procedure given in Algorithm 5.1. 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 5.2 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**

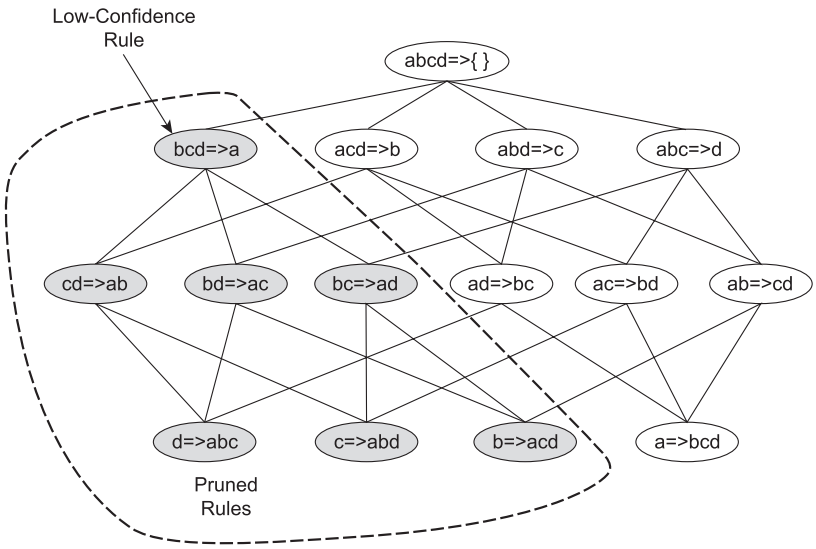


Figure 5.15. Pruning of association rules using the confidence measure.

Algorithm 5.3 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} = candidate-gen(H_m).
5: H_{m+1} = candidate-prune(H_{m+1},H_m).
6: **for** each $h_{m+1} \in H_{m+1}$ **do**
7: $conf = \sigma(f_k)/\sigma(f_k - h_{m+1})$.
8: **if** $conf \geq minconf$ **then**
9: **output** the rule $(f_k - h_{m+1}) \longrightarrow h_{m+1}$.
10: **else**
11: **delete** h_{m+1} from H_{m+1}.
12: **end if**
13: **end for**
14: **call** ap-genrules(f_k, H_{m+1}.)
15: **end if**

5.3.3 An Example: Congressional Voting Records

This section demonstrates the results of applying association analysis to the voting records of members of the United States House of Representatives. The data is obtained from the 1984 Congressional Voting Records Database, which is available at the UCI machine learning data repository. Each transaction

contains information about the party 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 5.3.

Table 5.3. 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 5.4. 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%

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 5.4. 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.

5.4 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 frequent 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.

5.4.1 Maximal Frequent Itemsets

Definition 5.3 (Maximal Frequent Itemset)**.** A frequent itemset is maximal if none of its immediate supersets are frequent.

To illustrate this concept, consider the itemset lattice shown in Figure 5.16. 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 maximal frequent itemsets because all of their immediate supersets are infrequent. For example, the itemset $\{a, d\}$ 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, every frequent itemset in Figure 5.16 is a subset of one of the three maximal frequent itemsets, $\{a, d\}$, $\{a, c, e\}$, and $\{b, c, d, e\}$. If an itemset is not a proper subset of any of the maximal frequent itemsets, then it is either infrequent (e.g., $\{a, d, e\}$) or maximal frequent itself (e.g., $\{b, c, d, e\}$). Hence, the maximal frequent itemsets $\{a, c, e\}$, $\{a, d\}$, and $\{b, c, d, e\}$ provide a compact representation of

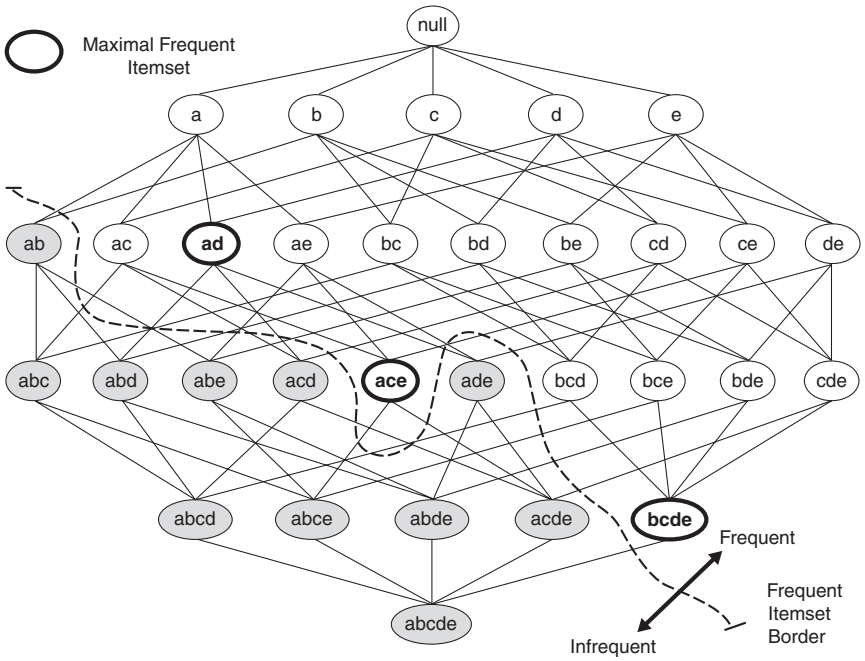


Figure 5.16. Maximal frequent itemset.

the frequent itemsets shown in Figure 5.16. Enumerating all the subsets of maximal frequent itemsets generates the complete list of all frequent itemsets.

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. We briefly describe one such approach in Section 5.5.

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 information about the support of their subsets except that it meets the support threshold. 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 is desirable to have a minimal representation of itemsets that preserves the support information. We describe such a representation in the next section.

5.4.2 Closed Itemsets

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

Definition 5.4 (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 5.17. To better illustrate the support count of each itemset, we have associated each node (itemset) in the lattice with a list of its corresponding 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 the support for $\{b\}$ is identical to $\{b, c\}$. This is because every transaction that contains b also contains c. Hence, $\{b\}$ is not a closed itemset. Similarly, since c occurs in every transaction that contains both a and d, the itemset $\{a, d\}$ is not closed as it has the same support as its superset $\{a, c, d\}$. On the other hand, $\{b, c\}$ is a closed itemset because it does not have the same support count as any of its supersets.

An interesting property of closed itemsets is that if we know their support counts, we can derive the support count of every other itemset in the itemset lattice without making additional passes over the data set. For example, consider the 2-itemset $\{b, e\}$ in Figure 5.17. Since $\{b, e\}$ is not closed, its support must be equal to the support of one of its immediate supersets, $\{a, b, e\}$, $\{b, c, e\}$, and $\{b, d, e\}$. Further, none of the supersets of $\{b, e\}$ can have a support greater than the support of $\{b, e\}$, due to the anti-monotone nature of the support measure. Hence, the support of $\{b, e\}$ can be computed by examining the support counts of all of its immediate supersets of size three and taking their maximum value. If an immediate superset is closed (e.g., $\{b, c, e\}$), we would know its support count. Otherwise, we can recursively compute its support by examining the supports of its immediate supersets of size four. In general, the support count of any non-closed $(k-1)$-itemset can be determined as long as we know the support counts of all k-itemsets. Hence, one can devise an iterative algorithm that computes the support counts of itemsets at level $k-1$ using the support counts of itemsets at level k, starting from the level $k_{\max}$, where $k_{\max}$ is the size of the largest closed itemset.

Even though closed itemsets provide a compact representation of the support counts of all itemsets, they can still be exponentially large in number. Moreover, for most practical applications, we only need to determine the

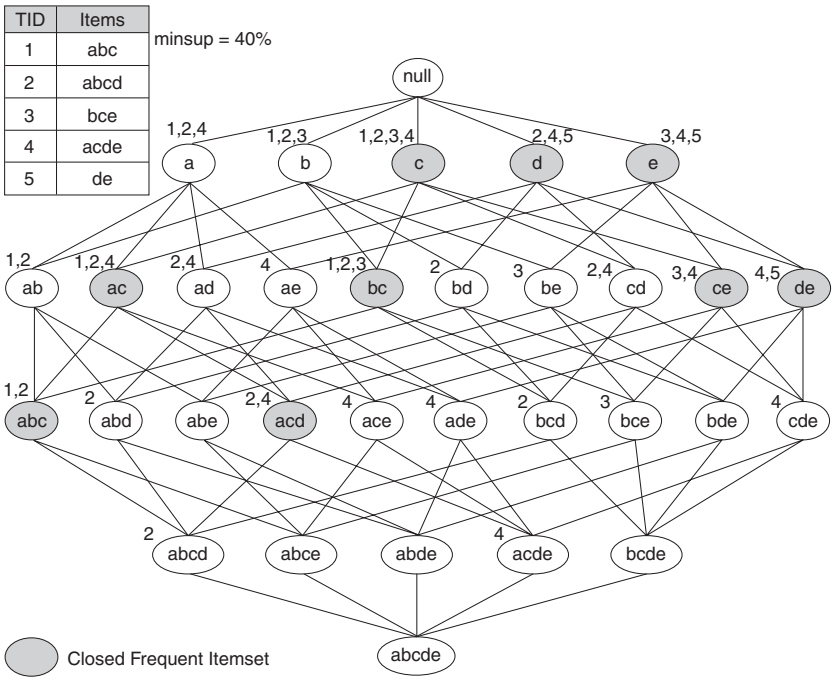


Figure 5.17. An example of the closed frequent itemsets (with minimum support equal to 40%).

support count of all frequent itemsets. In this regard, closed frequent itemsets provide a compact representation of the support counts of all frequent itemsets, which can be defined as follows.

Definition 5.5 (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%. In Figure 5.17, 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 closed frequent itemsets to determine the support counts for all non-closed frequent itemsets. For example, consider the frequent itemset $\{a, d\}$ shown in Figure 5.17. Because this itemset is not closed, its support count must be equal to the maximum support count of its immediate supersets, $\{a, b, d\}$, $\{a, c, d\}$, and $\{a, d, e\}$. Also, since $\{a, d\}$ is frequent, we only need to consider the support of its frequent supersets. In general, the support count of every

Algorithm 5.4 Support counting using closed frequent itemsets.

1: Let C denote the set of closed frequent itemsets and F denote the set of all 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** $k = k_{\max} - 1$ down to 1 **do**
5: $F_k = \{f | f \in F,\ |f| = k\}$ {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**

non-closed frequent k-itemset can be obtained by considering the support of all its frequent supersets of size $k + 1$. For example, since the only frequent superset of $\{a, d\}$ is $\{a, c, d\}$, its support is equal to the support of $\{a, c, d\}$, which is 2. Using this methodology, an algorithm can be developed to compute the support for every frequent itemset. The pseudocode for this algorithm is shown in Algorithm 5.4. 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. Note that the set of all frequent itemsets can be easily computed by taking the union of all subsets of frequent closed itemsets.

To illustrate the advantage of using closed frequent itemsets, consider the data set shown in Table 5.5, 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. Assuming that the support threshold is 20%, itemsets involving items from the same group are frequent, but itemsets involving items from different groups are infrequent. The total number of frequent itemsets is thus $3 \times (2^5 - 1) = 93$. However, there are only four closed frequent itemsets in the data: ($\{a_3, a_4\}$, $\{a_1, a_2, a_3, a_4, a_5\}$, $\{b_1, b_2, b_3, b_4, b_5\}$, 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.

Table 5.5. 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	**1**	**1**	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**

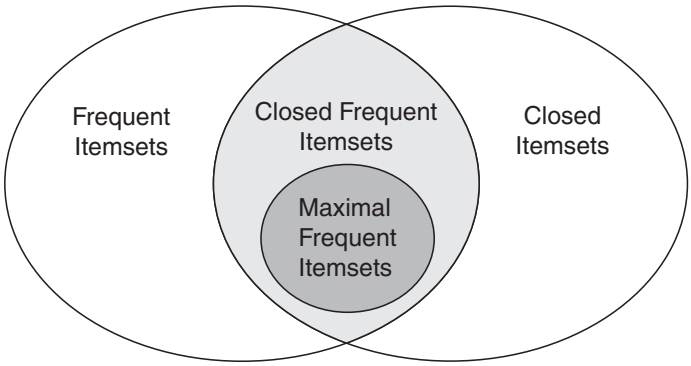


Figure 5.18. Relationships among frequent, closed, closed frequent, and maximal frequent itemsets.

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, closed, closed frequent, and maximal frequent itemsets are shown in Figure 5.18.

5.5 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 5.2.5, 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 5.1. 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 5.19(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 5.19(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 store the candidate itemsets, but it can help to rapidly identify the frequent itemset border, given the configuration shown in Figure 5.19(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

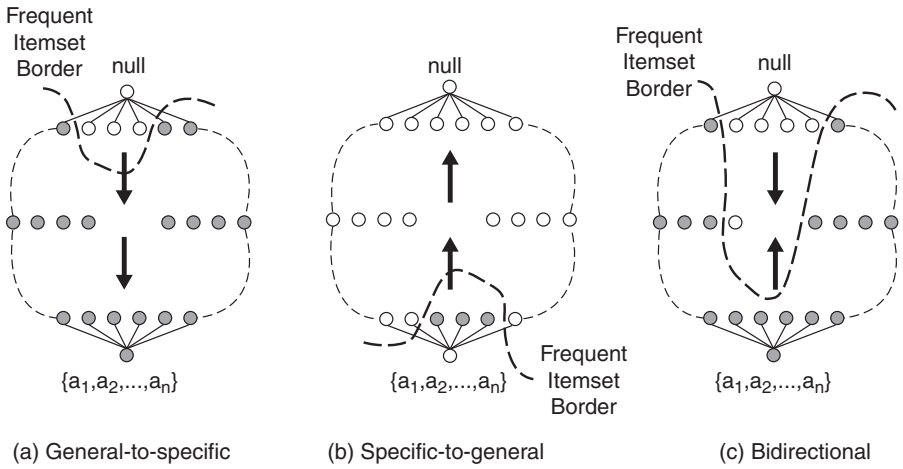
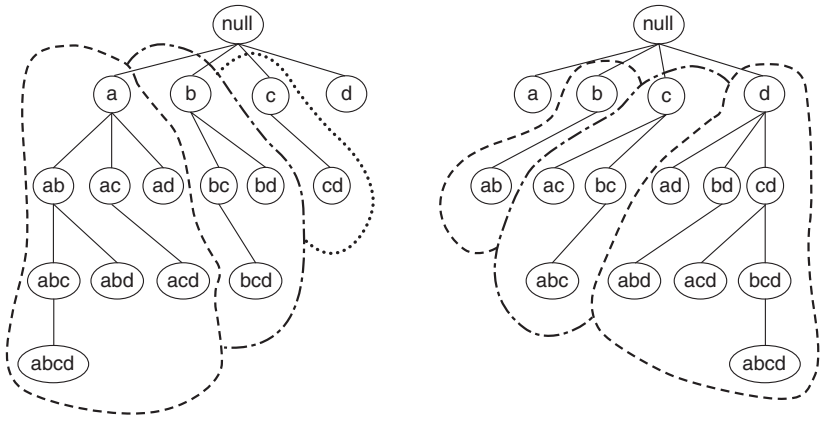


Figure 5.19. General-to-specific, specific-to-general, and bidirectional search.

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 5.20.

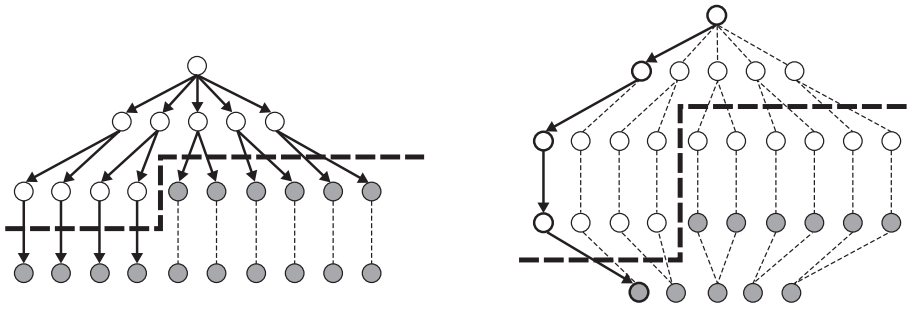
- **Breadth-First versus Depth-First:** The *Apriori* algorithm traverses the lattice in a breadth-first manner, as shown in Figure 5.21(a). It first discovers all the frequent 1-itemsets, followed by the frequent 2-itemsets, and so on, until no new frequent itemsets are generated. The itemset lattice can also be traversed in a depth-first manner, as shown in Figures 5.21(b) and 5.22. The algorithm can start from, say, node a in Figure 5.22, 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.



(a) Prefix tree. (b) Suffix tree.

Figure 5.20. Equivalence classes based on the prefix and suffix labels of itemsets.



(a) Breadth first (b) Depth first

Figure 5.21. Breadth-first and depth-first traversals.

Once a maximal frequent itemset is found, substantial pruning can be performed on its subsets. For example, if the node *bcde* shown in Figure 5.22 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

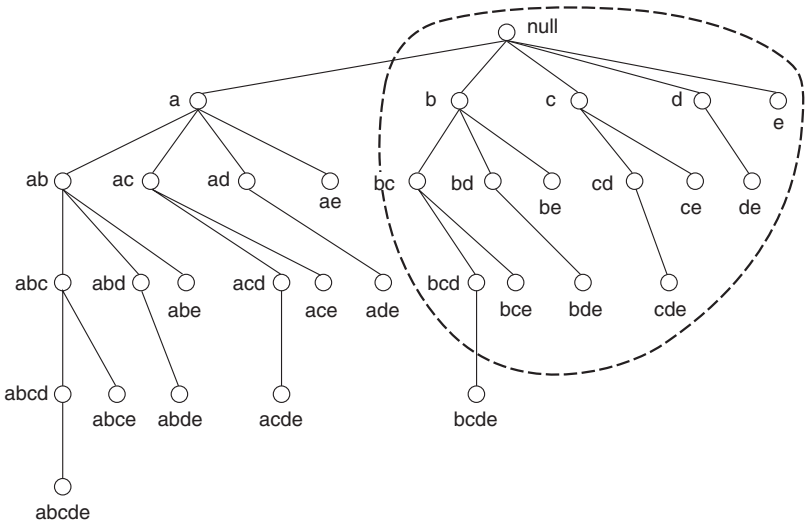


Figure 5.22. Generating candidate itemsets using the depth-first approach.

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 5.23 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 sized itemsets. However, one problem with this approach is that the initial set of TID-lists might 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.

5.6 FP-Growth Algorithm*

This section presents an alternative algorithm called **FP-growth** that takes a radically different approach to discovering frequent itemsets. The algorithm

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 5.23. Horizontal and vertical data format.

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.

5.6.1 FP-Tree Representation

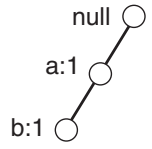
An FP-tree is a compressed representation of the input data. It is constructed by reading the data set one transaction at a time and mapping each transaction onto a path in the FP-tree. As different transactions can have several items in common, their paths might overlap. The more the paths overlap with 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 5.24 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:

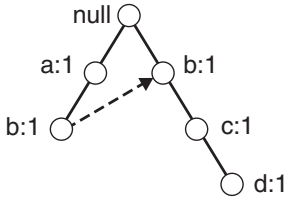
1. The data set is scanned once to determine the support count of each item. Infrequent items are discarded, while the frequent items are sorted in decreasing support counts inside every transaction of the data set. For

Transaction Data Set

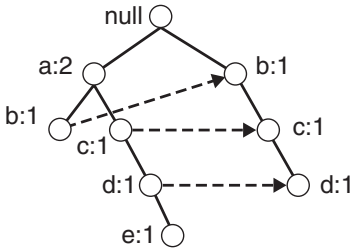
TID	Items
1	{a,b}
2	{b,c,d}
3	{a,c,d,e}
4	{a,d,e}
5	{a,b,c}
6	{a,b,c,d}
7	{a}
8	{a,b,c}
9	{a,b,d}
10	{b,c,e}



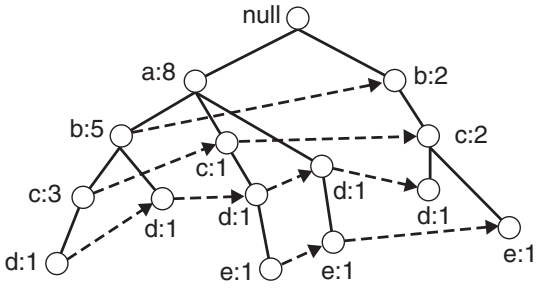
(i) After reading TID=1



(ii) After reading TID=2



(iii) After reading TID=3



(iv) After reading TID=10

Figure 5.24. Construction of an FP-tree.

the data set shown in Figure 5.24, a is the most frequent item, followed by b, c, d, and e.

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 `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 `null` $\rightarrow b \rightarrow c \rightarrow d$. Every node along this path also has a frequency count equal to one. Although the

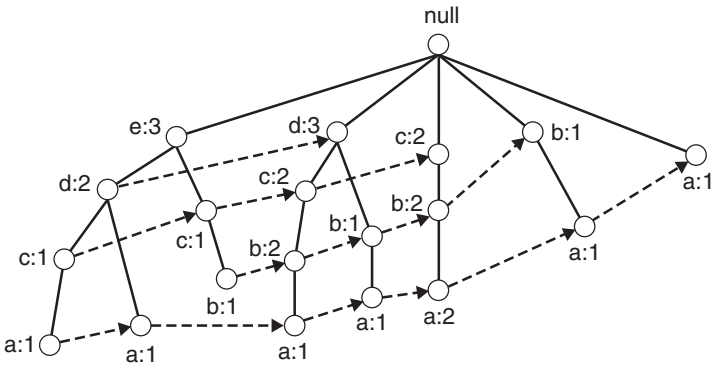


Figure 5.25. An FP-tree representation for the data set shown in Figure 5.24 with a different item ordering scheme.

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, `null` $\rightarrow a \rightarrow c \rightarrow d \rightarrow e$, overlaps with the path for the first transaction, `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 5.24.

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. The notion of ordering items in decreasing order of support counts relies on the possibility that the high support items occur more frequently across all paths and hence must be used as most commonly occurring prefixes. For example,

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 5.25. 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, especially when the high support items do not occur frequently together with the other items. For example, suppose we augment the data set given in Figure 5.24 with 100 transactions that contain $\{e\}$, 80 transactions that contain $\{d\}$, 60 transactions 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 5.25, while a scheme based on increasing support counts produces a smaller FP-tree similar to Figure 5.24(iv).

An FP-tree also contains a list of pointers connecting nodes that have the same items. These pointers, represented as dashed lines in Figures 5.24 and 5.25, 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.

5.6.2 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 5.24, 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 5.5. 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 5.26 (a). Similar paths for itemsets ending in d, c, b, and a are shown in Figures 5.26 (b), (c), (d), and (e), respectively.

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 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

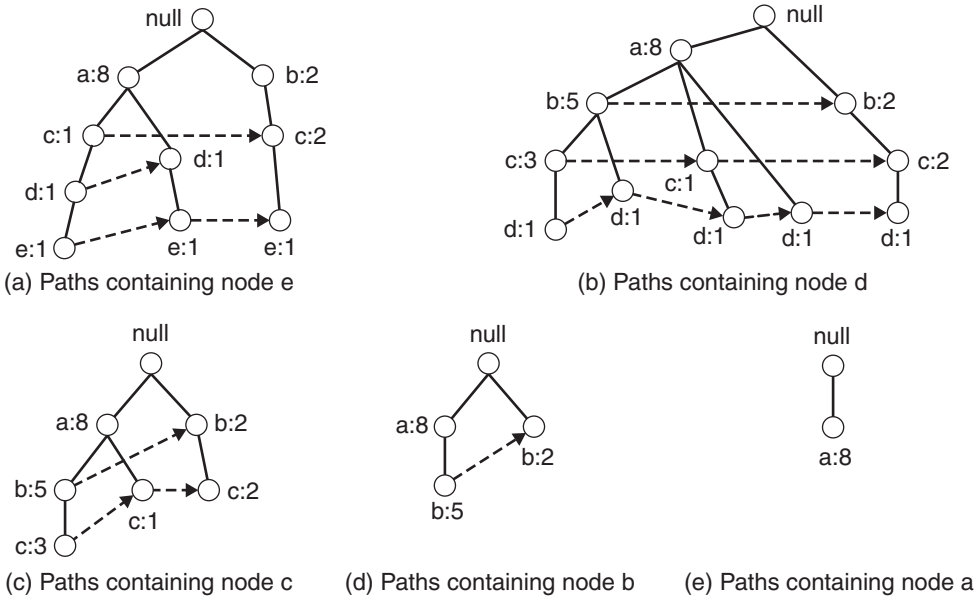


Figure 5.26. Decomposing the frequent itemset generation problem into multiple subproblems, where each subproblem involves finding frequent itemsets ending in e, d, c, b, and a.

the solutions obtained from the subproblems, all the frequent itemsets ending in e can be found. Finally, the set of all frequent itemsets can be generated by merging the solutions to the subproblems of finding frequent itemsets ending in e, d, c, b, and a. 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 5.27(a).

2. From the prefix paths shown in Figure 5.27(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

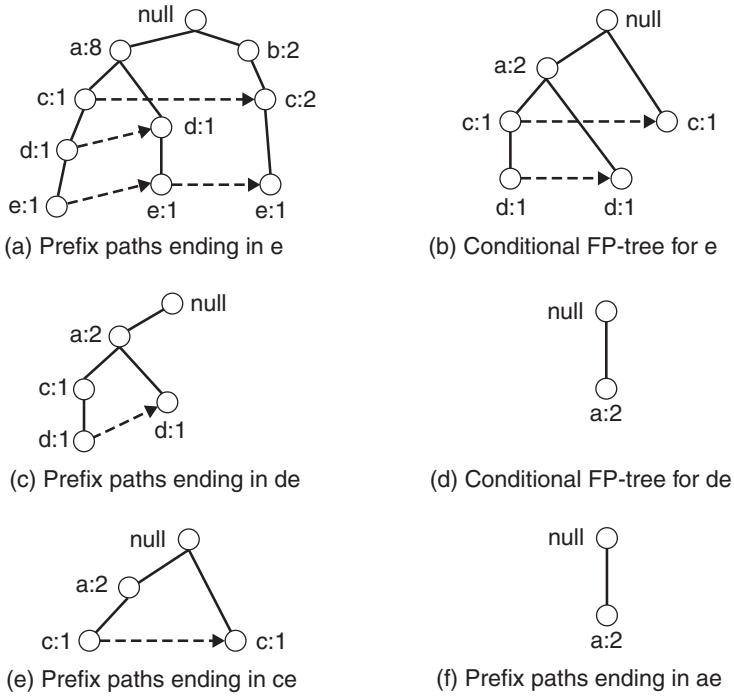


Figure 5.27. Example of applying the FP-growth algorithm to find frequent itemsets ending in e.

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 5.27(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 5.27(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 5.27(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 5.27(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, $\{c, e\}$ is found to be frequent. However, the conditional FP-tree for ce will have no frequent items and thus will be eliminated. The algorithm proceeds to solve the next subproblem and finds $\{a, e\}$ 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.

5.7 Evaluation of Association Patterns

Although the *Apriori* principle significantly reduces the exponential search space of candidate itemsets, association analysis algorithms still have the potential to generate a large number of patterns. For example, although the data set shown in Table 5.1 contains only six items, it can produce hundreds of association rules at particular support and confidence thresholds. As the size and dimensionality of real commercial databases can be very large, we can easily end up with thousands or even millions of patterns, many of which might not be interesting. Identifying the most interesting patterns from the multitude of all possible 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 a data-driven approach to define **objective interestingness measures**. These measures can be used to rank patterns—itemsets or rules—and thus provide a straightforward way of dealing with the enormous number of patterns that are found in a data set. Some of the measures can also provide statistical information, e.g., itemsets that involve a set of unrelated items or cover very few transactions are considered uninteresting because they may capture spurious relationships in the data and should be eliminated. 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 might 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 domain experts. Readers interested in subjective interestingness measures may refer to resources listed in the bibliography at the end of this chapter.

How can we capture humanc

Table 5.6. 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

5.7.1 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 only that the user specifies a threshold for filtering low-quality patterns. An objective measure is usually computed based on the frequency counts tabulated in a **contingency table**. Table 5.6 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 uses mainly on asymmetric binary variables, note that contingency tables are also applicable to other attribute types such as symmetric binary, nominal, and ordinal variables.

worth it to extend measures,o

Limitations of the Support-Confidence Framework The classical association rule mining formulation relies on the support and confidence measures to eliminate uninteresting patterns. The drawback of support, which is described more fully in Section 5.8, is that 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 5.3. 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 contingency table such as the one shown in Table 5.7.

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

Table 5.7. Beverage preferences among a group of 1000 people.

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

150/200 for confidence1

(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.

Table 5.8. Information about people who drink tea and people who use honey in their beverage.

	$Honey$	$\overline{Honey}$	
Tea	100	100	200
$\overline{Tea}$	20	780	800
	120	880	1000

sup = 100/1000conf = 100/20

Now consider a similar problem where we are interested in analyzing the relationship between people who drink tea and people who use honey in their beverage. Table 5.8 summarizes the information gathered over the same group of people about their preferences for drinking tea and using honey. If we evaluate the association rule $\{Tea\} \longrightarrow \{Honey\}$ using this information, we will find that the confidence value of this rule is merely 50%, which might be easily rejected using a reasonable threshold on the confidence value, say 70%. One thus might consider that the preference of a person for drinking tea has no influence on her preference for using honey. However, the fraction of people who use honey, regardless of whether they drink tea, is only 12%. Hence, knowing that a person drinks tea significantly increases her probability of using honey from 12% to 50%. Further, the fraction of people who do not drink tea but use honey is only 2.5%! This suggests that there is definitely some information in the preference of a person of using honey given that she

for real data, we were l

drinks tea. The rule $\{Tea\} \longrightarrow \{Honey\}$ may therefore be falsely rejected if confidence is used as the evaluation measure.

■

Note that if we take the support of coffee drinkers into account, we would not be surprised to find that many of the people who drink tea also drink coffee, since the overall number of coffee drinkers is quite large by itself. 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. Similarly, if we account for the fact that the support of using honey is inherently small, it is easy to understand that the fraction of tea drinkers who use honey will naturally be small. Instead, what is important to measure is the change in the fraction of honey users, given the information that they drink tea.

The limitations of the confidence measure are well-known and can be understood from a statistical perspective as follows. The support of a variable measures the probability of its occurrence, while the support $s(A, B)$ of a pair of a variables A and B measures the probability of the two variables occurring together. Hence, the joint probability $P(A, B)$ can be written as

$$P(A, B) = s(A, B) = \frac{f_{11}}{N}.$$

If we assume A and B are statistically independent, i.e. there is no relationship between the occurrences of A and B, then $P(A, B) = P(A) \times P(B)$. Hence, under the assumption of statistical independence between A and B, the support $s_{\text{indep}}(A, B)$ of A and B can be written as

$$s_{\text{indep}}(A, B) = s(A) \times s(B) \quad \text{or} \quad \text{equivalently,} \quad s_{\text{indep}}(A, B) = \frac{f_{1+}}{N} \times \frac{f_{+1}}{N}. \tag{5.4}$$

If the support between two variables, $s(A, B)$ is equal to $s_{\text{indep}}(A, B)$, then A and B can be considered to be unrelated to each other. However, if $s(A, B)$ is widely different from $s_{\text{indep}}(A, B)$, then A and B are most likely dependent. Hence, any deviation of $s(A, B)$ from $s(A) \times s(B)$ can be seen as an indication of a statistical relationship between A and B. Since the confidence measure only considers the deviance of $s(A, B)$ from $s(A)$ and not from $s(A) \times s(B)$, it fails to account for the support of the consequent, namely $s(B)$. This results in the detection of spurious patterns (e.g., $\{Tea\} \longrightarrow \{Coffee\}$) and the rejection of truly interesting patterns (e.g., $\{Tea\} \longrightarrow \{Honey\}$), as illustrated in the previous example.

Various objective measures have been used to capture the deviance of $s(A, B)$ from $s_{\text{indep}}(A, B)$, that are not susceptible to the limitations of the confidence measure. Below, we provide a brief description of some of these measures and discuss some of their properties.

Interest Factor The interest factor, which is also called as the "lift," can be defined as follows:

$$I(A, B) = \frac{s(A, B)}{s(A) \times s(B)} = \frac{N f_{11}}{f_{1+} f_{+1}}. \tag{5.5}$$

Notice that $s(A) \times s(B) = s_{\text{indep}}(A, B)$. Hence, the interest factor measures the ratio of the support of a pattern $s(A, B)$ against its baseline support $s_{\text{indep}}(A, B)$ computed under the statistical independence assumption. Using Equations 5.5 and 5.4, we can interpret the measure as follows:

$$I(A, B) \begin{cases} = 1, & \text{if } A \text{ and } B \text{ are independent;} \\ > 1, & \text{if } A \text{ and } B \text{ are positively related;} \\ < 1, & \text{if } A \text{ and } B \text{ are negatively related.} \end{cases} \tag{5.6}$$

For the tea-coffee example shown in Table 5.7, $I = \frac{0.15}{0.2 \times 0.8} = 0.9375$, thus suggesting a slight negative relationship between tea drinkers and coffee drinkers. Also, for the tea-honey example shown in Table 5.8, $I = \frac{0.1}{0.12 \times 0.2} = 4.1667$, suggesting a strong positive relationship between people who drink tea and people who use honey in their beverage. We can thus see that the interest factor is able to detect meaningful patterns in the tea-coffee and tea-honey examples. Indeed, the interest factor has a number of statistical advantages over the confidence measure that make it a suitable measure for analyzing statistical independence between variables.

again, conf ignores s(B) in

Piatesky-Shapiro (PS) Measure Instead of computing the ratio between $s(A, B)$ and $s_{\text{indep}}(A, B) = s(A) \times s(B)$, the PS measure considers the difference between $s(A, B)$ and $s(A) \times s(B)$ as follows.

$$PS = s(A, B) - s(A) \times s(B) = \frac{f_{11}}{N} - \frac{f_{1+} f_{+1}}{N^2} \tag{5.7}$$

The PS value is 0 when A and B are mutually independent of each other. Otherwise, $PS > 0$ when there is a positive relationship between the two variables, and $PS < 0$ when there is a negative relationship.

Correlation Analysis Correlation analysis is one of the most popular techniques 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 83). For binary variables, correlation can be measured using the ϕ-coefficient, which is defined as

$$\phi = \frac{f_{11}f_{00} - f_{01}f_{10}}{\sqrt{f_{1+}f_{+1}f_{0+}f_{+0}}}. \tag{5.8}$$

If we rearrange the terms in 5.8, we can show that the ϕ-coefficient can be rewritten in terms of the support measures of A, B, and $\{A, B\}$ as follows:

$$\phi = \frac{s(A,B) - s(A) \times s(B)}{\sqrt{s(A) \times (1 - s(A)) \times s(B) \times (1 - s(B))}}. \tag{5.9}$$

Note that the numerator in the above equation is identical to the PS measure. Hence, the ϕ-coefficient can be understood as a normalized version of the PS measure, where that the value of the ϕ-coefficient ranges from -1 to $+1$. From a statistical viewpoint, the correlation captures the normalized difference between $s(A, B)$ and $s_{\text{indep}}(A, B)$. A correlation value of 0 means no relationship, while a value of $+1$ suggests a perfect positive relationship and a value of -1 suggests a perfect negative relationship. The correlation measure has a statistical meaning and hence is widely used to evaluate the strength of statistical independence among variables. For instance, the correlation between tea and coffee drinkers in Table 5.7 is -0.0625 which is slightly less than 0. On the other hand, the correlation between people who drink tea and people who use honey in Table 5.8 is 0.5847, suggesting a positive relationship.

IS Measure IS is an alternative measure for capturing the relationship between $s(A, B)$ and $s(A) \times s(B)$. The IS measure is defined as follows:

$$IS(A,B) = \sqrt{I(A,B) \times s(A,B)} = \frac{s(A,B)}{\sqrt{s(A)s(B)}} = \frac{f_{11}}{\sqrt{f_{1+}f_{+1}}}. \tag{5.10}$$

Although the definition of IS looks quite similar to the interest factor, they share some interesting differences. Since IS is the geometric mean between the interest factor and the support of a pattern, IS is large when both the interest factor and support are large. Hence, if the interest factor of two patterns are identical, the IS has a preference of selecting the pattern with higher support. It is also possible to show that IS is mathematically equivalent to the cosine

measure for binary variables (see Equation 2.6 on page 81). The value of IS thus varies from 0 to 1, where an IS value of 0 corresponds to no co-occurrence of the two variables, while an IS value of 1 denotes perfect relationship, since they occur in exactly the same transactions. For the tea-coffee example shown in Table 5.7, the value of IS is equal to 0.375, while the value of IS for the tea-honey example in Table 5.8 is 0.6455. The IS measure thus gives a higher value for the $\{Tea\} \longrightarrow \{Honey\}$ rule than the $\{Tea\} \longrightarrow \{Coffee\}$ rule, which is consistent with our understanding of the two rules.

Alternative Objective Interestingness Measures

Note that all of the measures defined in the previous section use different techniques to capture the deviance between $s(A, B)$ and $s_{\text{indep}}(A, B) = s(A) \times s(B)$. Some measures use the ratio between $s(A, B)$ and $s_{\text{indep}}(A, B)$, e.g., the interest factor and IS, while some other measures consider the difference between the two, e.g., the PS and the ϕ-coefficient. Some measures are bounded in a particular range, e.g., the IS and the ϕ-coefficient, while others are unbounded and do not have a defined maximum or minimum value, e.g., the Interest Factor. Because of such differences, these measures behave differently when applied to different types of patterns. Indeed, the measures defined above are not exhaustive and there exist many alternative measures for capturing different properties of relationships between pairs of binary variables. Table 5.9

Table 5.9. Examples of objective measures for the itemset $\{A, B\}$.

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

Table 5.10. 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

provides 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.

Suppose the measures defined in Table 5.9 are applied to rank the ten contingency tables shown in Table 5.10. These contingency tables are chosen to illustrate the differences among the existing measures. The ordering produced by these measures is shown in Table 5.11 (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, others produce quite different ordering results. For example, the rankings given by the ϕ-coefficient agrees mostly with those provided by κ and collective strength, but are quite different than the rankings produced by interest factor. Furthermore, a contingency table such as E_{10} is ranked lowest according to the ϕ-coefficient, but highest according to interest factor.

Table 5.11. Rankings of contingency tables using the measures given in Table 5.9.

	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

Properties of Objective Measures

The results shown in Table 5.11 suggest that the measures greatly differ from each other and can provide conflicting information about the quality of a pattern. In fact, no measure is universally best for all applications. In the following, we describe some properties of the measures that play an important role in determining if they are suited for a certain application.

Inversion Property Consider the binary vectors shown in Figure 5.28. The 0/1 value in each column vector indicates whether a transaction (row) contains a particular item (column). For example, the vector A indicates that the item appears in the first and last transactions, whereas the vector B indicates that the item is contained only in the fifth transaction. The vectors $\overline{A}$ and $\overline{B}$ are the inverted versions of A and B, i.e., their 1 values have been changed to 0 values (absence to presence) and vice versa. Applying this transformation to a binary vector is called **inversion**. If a measure is invariant under the inversion operation, then its value for the vector pair $\{\overline{A}, \overline{B}\}$ should be identical to its value for $\{A, B\}$. The inversion property of a measure can be tested as follows.

Definition 5.6 (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}.

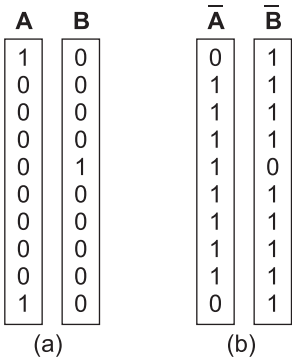


Figure 5.28. Effect of the inversion operation. The vectors $\overline{A}$ and $\overline{E}$ are inversions of vectors A and B, respectively.

Measures that are invariant to the inversion property include the correlation (ϕ-coefficient), odds ratio, κ, and collective strength. These measures are especially useful in scenarios where the presence (1's) of a variable is as important as its absence (0's). For example, if we compare two sets of answers to a series of true/false questions where 0's (true) and 1's (false) are equally meaningful, we should use a measure that gives equal importance to occurrences of 0–0's and 1–1's. For the vectors shown in Figure 5.28, the ϕ-coefficient is equal to -0.1667 regardless of whether we consider the pair $\{A, B\}$ or pair $\{\overline{A}, \overline{B}\}$. Similarly, the odds ratio for both pairs of vectors is equal to a constant value of 0. Note that even though the ϕ-coefficient and the odds ratio are invariant to inversion, they can still show different results, as will be shown later.

Measures that do not remain invariant under the inversion operation include the interest factor and the IS measure. For example, the IS value for the pair $\{\overline{A}, \overline{B}\}$ in Figure 5.28 is 0.825, which reflects the fact that the 1's in $\overline{A}$ and $\overline{B}$ occur frequently together. However, the IS value of its inverted pair $\{A, B\}$ is equal to 0, since A and B do not have any co-occurrence of 1's. For asymmetric binary variables, e.g., the occurrence of words in documents, this is indeed the desired behavior. A desired similarity measure between asymmetric variables should not be invariant to inversion, since for these variables, it is more meaningful to capture relationships based on the presence of a variable rather than its absence. On the other hand, if we are dealing with symmetric binary variables where the relationships between 0's and 1's are equally meaningful, care should be taken to ensure that the chosen measure is invariant to inversion.

Table 5.12. Contingency tables for the pairs $\{p,q\}$ and $\{r,s\}$.

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

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

Although the values of the interest factor and IS change with the inversion operation, they can still be inconsistent with each other. To illustrate this, consider Table 5.12, which shows the contingency tables for two pairs of variables, $\{p, q\}$ and $\{r, s\}$. Note that r and s are inverted transformations of p and q, respectively, where the roles of 0's and 1's have just been reversed. The interest factor for $\{p, q\}$ is 1.02 and for $\{r, s\}$ is 4.08, which means that the interest factor finds the inverted pair $\{r, s\}$ more related than the original pair $\{p, q\}$. On the contrary, the IS value decreases upon inversion from 0.9346 for $\{p, q\}$ to 0.286 for $\{r, s\}$, suggesting quite an opposite trend to that of the interest factor. Even though these measures conflict with each other for this example, they may be the right choice of measure in different applications.

Scaling Property Table 5.13 shows two contingency tables for gender and the grades achieved by students enrolled in a particular course. These tables can be used to study the relationship between gender and performance in the course. The second contingency table has data from the same population but has twice as many males and three times as many females. The actual number of males or females can depend upon the samples available for study, but the relationship between gender and grade should not change just because of differences in sample sizes. Similarly, if the number of students with high and low grades are changed in a new study, a measure of association between gender and grades should remain unchanged. Hence, we need a measure that is invariant to scaling of rows or columns. The process of multiplying a row or column of a contingency table by a constant value is called a row or column scaling operation. A measure that is invariant to scaling does not change its value after any row or column scaling operation.

Definition 5.7 (Scaling Invariance Property)**.** Let T be a contingency table with frequency counts $[f_{11};\ f_{10};\ f_{01};\ f_{00}]$. Let T' be the transformed a contingency table with scaled frequency counts $[k_1k_3f_{11};\ k_2k_3f_{10};\ k_1k_4f_{01};\ k_2k_4f_{00}]$, where k_1, k_2, k_3, k_4 are positive constants used to scale the two rows and the

Table 5.13. The grade-gender example.

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

(a) Sample data of size 100.

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

(b) Sample data of size 230.

two columns of T. An objective measure M is invariant under the row/column scaling operation if $M(T) = M(T')$.

Note that the use of the term 'scaling' here should not be confused with the scaling operation for continuous variables introduced in Chapter 2 on page 23, where all the values of a variable were being multiplied by a constant factor, instead of scaling a row or column of a contingency table.

Scaling of rows and columns in contingency tables occurs in multiple ways in different applications. For example, if we are measuring the effect of a particular medical procedure on two sets of subjects, *healthy* and *diseased*, the ratio of healthy and diseased subjects can widely vary across different studies involving different groups of participants. Further, the fraction of healthy and diseased subjects chosen for a controlled study can be quite different from the true fraction observed in the complete population. These differences can result in a row or column scaling in the contingency tables for different populations of subjects. In general, the frequencies of items in a contingency table closely depends on the sample of transactions used to generate the table. Any change in the sampling procedure may affect a row or column scaling transformation. A measure that is expected to be invariant to differences in the sampling procedure must not change with row or column scaling.

Of all the measures introduced in Table 5.9, only the odds ratio (α) is invariant to row and column scaling operations. For example, the value of odds ratio for both the tables in Table 5.13 is equal to 0.375. 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. Indeed, the odds ratio is a preferred choice of measure in the medical domain, where it is important to find relationships that do not change with differences in the population sample chosen for a study.

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 5.8 (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, we would like to use a measure that remains invariant under the null addition operation. Otherwise, the relationship between words can be made to change 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.

To demonstrate the effect of null addition, consider the two contingency tables T_1 and T_2 shown in Table 5.14. Table T_2 has been obtained from T_1 by adding 1000 extra transactions with both A and B absent. This operation only affects the f_{00} entry of Table T_2, which has increased from 100 to 1100, whereas all the other frequencies in the table (f_{11}, f_{10}, and f_{01}) remain the same. Since IS is invariant to null addition, it gives a constant value of 0.875 to both the tables. However, the addition of 1000 extra transactions with occurrences of 0–0's changes the value of interest factor from 0.972 for T_1 (denoting a slightly negative correlation) to 1.944 for T_2 (positive correlation). Similarly, the value of odds ratio increases from 7 for T_1 to 77 for T_2. Hence, when the interest factor or odds ratio are used as the association measure, the relationships between variables changes by the addition of null transactions where both the variables are absent. In contrast, the IS measure is invariant to null addition, since it considers two variables to be related only if they frequently occur together. Indeed, the IS measure (cosine measure) is widely used to measure similarity among documents, which is expected to depend only on the joint occurrences (1's) of words in documents, but not their absences (0's).

Table 5.15 provides a summary of properties for the measures defined in Table 5.9. Even though this list of properties is not exhaustive, it can serve as a useful guide for selecting the right choice of measure for an application. Ideally, if we know the specific requirements of a certain application, we can ensure that the selected measure shows properties that adhere to those requirements. For example, if we are dealing with asymmetric variables, we would prefer to

Table 5.14. An example demonstrating the effect of null addition.

	B	$\overline{B}$	
A	700	100	800
$\overline{A}$	100	100	200
	800	200	1000

(a) Table T_1.

	B	$\overline{B}$	
A	700	100	800
$\overline{A}$	10	1100	1200
	800	1200	2000

(b) Table T_2.

use a measure that is not invariant to null addition or inversion. On the other hand, if we require the measure to remain invariant to changes in the sample size, we would like to use a measure that does not change with scaling.

Asymmetric Interestingness Measures

Note that in the discussion so far, we have only considered measures that do not change their value when the order of the variables are reversed. More specifically, if M is a measure and A and B are two variables, then $M(A, B)$ is equal to $M(B, A)$ if the order of the variables does not matter. Such measures are called **symmetric**. On the other hand, measures that depend on the order of variables ($M(A, B) \neq M(B, A)$) are called **asymmetric** measures. For example, the 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. Note that the use of the term 'asymmetric' to describe a particular type of measure of relationship—one in which the order of the variables is important—should not be confused with the use of 'asymmetric' to describe a binary variable for which only 1's are important. Asymmetric measures are more suitable for analyzing association rules, since the items in a rule do have a specific order. Even though we only considered symmetric measures to discuss the different properties of association measures, the above discussion is also relevant for the asymmetric measures. See Bibliographic Notes for more information about different kinds of asymmetric measures and their properties.

5.7.2 Measures beyond Pairs of Binary Variables

The measures shown in Table 5.9 are defined for pairs of binary variables (e.g., 2-itemsets or association rules). However, many of them, such as support and

Table 5.15. 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	Yes	No
s	Support	No	No	No

Table 5.16. Example of a three-dimensional contingency table.

c	b	$\overline{b}$	
a	f_{111}	f_{101}	f_{1+1}
$\overline{a}$	f_{011}	f_{001}	f_{0+1}
	f_{+11}	f_{+01}	f_{++1}

$\overline{c}$	b	$\overline{b}$	
a	f_{110}	f_{100}	f_{1+0}
$\overline{a}$	f_{010}	f_{000}	f_{0+0}
	f_{+10}	f_{+00}	f_{++0}

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 5.16. 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 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:

$$f_{i_1 i_2 \ldots i_k} = \frac{f_{i_1+\ldots+} \times f_{+i_2\ldots+} \times \ldots \times f_{++\ldots i_k}}{N^{k-1}}. \tag{5.11}$$

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. Also, care should be taken in using such alternate measures for more than two variables, since they may not always show the anti-monotone property in the same way as the support measure, making them unsuitable for mining patterns using the *Apriori* principle.

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 Section 5.7.3. More sophisticated statistical techniques are available to analyze such relationships, e.g., loglinear models, but these techniques are beyond the scope of this book.

5.7.3 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 televisions (HDTV) and exercise machines, as shown in Table 5.17. 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.

Table 5.17. A two-way contingency table between the sale of high-definition television and exercise machine.

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

Table 5.18. Example of a three-way contingency table.

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

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 5.18 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 5.17. Furthermore, there are more working adults than college students who buy these items. For college students:

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

while for working adults:

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

The rules suggest that, for each group, customers who do not buy high-definition televisions are more likely to buy exercise machines, which contradicts 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 related in the combined data but is negatively related in

the stratified data (see Exercise 21 on page 449). The reversal in the direction of association is known as Simpson's paradox.

The paradox can be explained in the following way. First, notice that most customers who buy HDTVs are working adults. This is reflected in the high confidence of the rule {HDTV=Yes} $\longrightarrow$ {Working Adult} (170/180 = 94.4%). Second, the high confidence of the rule {Exercise machine=Yes} $\longrightarrow$ {Working Adult} (148/153 = 96.7%) suggests that most customers who buy exercise machines are also working adults. Since working adults form the largest fraction of customers for both HDTVs and exercise machines, they both look related and the rule {HDTV=Yes} $\longrightarrow$ {Exercise machine=Yes} turns out to be stronger in the combined data than what it would have been if the data is stratified. Hence, customer group acts as a *hidden* variable that affects both the fraction of customers who buy HDTVs and those who buy exercise machines. If we factor out the effect of the hidden variable by stratifying the data, we see that the relationship between buying HDTVs and buying exercise machines is not direct, but shows up as an indirect consequence of the effect of the hidden variable.

The Simpson's paradox can also be illustrated mathematically as follows. Suppose

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

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 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.

5.8 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

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
1	0	0
1	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Figure 5.29. 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.

the *Apriori* algorithm depends on properties such as the number of items in the data, the average transaction width, and the support threshold used. 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.

Figure 5.29 shows an illustrative example of a data set that has a skewed support distribution of its items. While p has a high support of 83.3% in the data, q and r are low-support items with a support of 16.7%. Despite their low support, q and r always occur together in the limited number of transactions that they appear and hence are strongly related. A pattern mining algorithm therefore should report $\{q, r\}$ as interesting.

However, note that choosing the right support threshold for mining itemsets such as $\{q, r\}$ can be quite tricky. If we set the threshold too high (e.g., 20%), then we may miss many interesting patterns involving low-support items such as $\{q, r\}$. Conversely, setting the support threshold too low can be

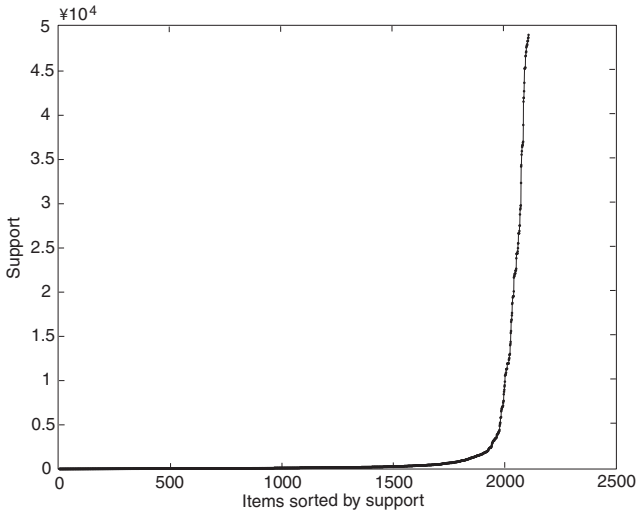


Figure 5.30. Support distribution of items in the census data set.

detrimental to the pattern mining process for 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, which makes their analysis and interpretation difficult. In particular, we may extract many spurious patterns that relate a high-frequency item such as p to a low-frequency item such as q. Such patterns, which are called **cross-support** patterns, are likely to be spurious because the association between p and q is largely influenced by the frequent occurrence of p instead of the joint occurrence of p and q together. Because the support of $\{p, q\}$ is quite close to the support of $\{q, r\}$, we may easily select $\{p, q\}$ if we set the support threshold low enough to include $\{q, r\}$.

An example of a real data set that exhibits a skewed support distribution is shown in Figure 5.30. 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. While more than 80% of the items have support less than 1%, a handful of them have support greater than 90%. To understand 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, as shown in Table 5.19. We can see that more than 82% of items belong to G_1 and have

Table 5.19. 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

a support less than 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. Patterns involving such low-support items, though meaningful, can easily be rejected by a frequent pattern mining algorithm with a high support threshold. On the other hand, setting a low support threshold may result in the extraction of spurious patterns that relate a high-frequency item in G_3 to a low-frequency item in G_1. 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.

This example shows that a large number of weakly related cross-support patterns can be generated when the support threshold is sufficiently low. Note that finding interesting patterns in data sets with skewed support distributions is not just a challenge for the support measure, but similar statements can be made about many other objective measures discussed in the previous sections. Before presenting a methodology for finding interesting patterns and pruning spurious ones, we formally define the concept of cross-support patterns.

Definition 5.9 (Cross-support Pattern)**.** Let us define the support ratio, $r(X)$, of an itemset $X = \{i_1, i_2, \ldots, i_k\}$ as

$$r(X) = \frac{\min\left[s(i_1), s(i_2), \ldots, s(i_k)\right]}{\max\left[s(i_1), s(i_2), \ldots, s(i_k)\right]}, \tag{5.12}$$

Given a user-specified threshold h_c, an itemset X is a cross-support pattern if $r(X) < h_c$.

Example 5.4. 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. For example, if we assume $h_c = 0.3$ for the data set presented in Figure 5.29, 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. However, their supports are comparable to that of $\{q, r\}$, making it difficult to eliminate cross-support patterns without loosing interesting ones using a support-based pruning strategy. 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 patterns and interesting patterns involving strongly connected but low-support items.

Even though the rule $\{q\} \longrightarrow \{p\}$ has very high confidence, 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 $\{q, r\}$, 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. An approach for finding the rule with the lowest confidence given an itemset can be described as follows.

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 $s(i_j) = \max\big[s(i_1), s(i_2), \ldots, s(i_k)\big]$. This follows directly from the definition of confidence as the ratio between the rule's support and the support of the rule antecedent. Hence, the confidence of a rule will be lowest when the support of the antecedent is highest.

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 that the upper bound of h-confidence in the above equation is exactly same as support ratio (r) given in Equation 5.12. 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, 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 involving low-support items are called **hyperclique patterns**.

Definition 5.10 (Hyperclique Pattern)**.** An itemset X is a hyperclique pattern if h-confidence$(X) > h_c$, where h_c is a user-specified threshold.

5.9 Bibliographic Notes

The association rule mining task was first introduced by Agrawal et al. [324, 325] to discover interesting relationships among items in market basket transactions. Since its inception, extensive research has been conducted to address the various issues in association rule mining, from its fundamental concepts to its implementation and applications. Figure 5.31 shows a taxonomy of the various research directions in this area, which is generally known as *association analysis*. As much of the research focuses on finding patterns that appear significantly often in the data, the area is also known as *frequent pattern mining*. A detailed review on some of the research topics in this area can be found in [362] and in [319].

Conceptual Issues

Research on the conceptual issues of association analysis has focused on developing a theoretical formulation of association analysis and extending the formulation to new types of patterns and going beyond asymmetric binary attributes.

Following the pioneering work by Agrawal et al. [324, 325], there has been a vast amount of research on developing a theoretical formulation for the association analysis problem. In [357], Gunopoulos et al. showed the connection between finding maximal frequent itemsets and the hypergraph transversal problem. An upper bound on the complexity of the association analysis task was also derived. Zaki et al. [454, 456] and Pasquier et al. [407] have applied formal concept analysis to study the frequent itemset generation problem. More importantly, such research has led to the development of a class of patterns known as closed frequent itemsets [456]. Friedman et al. [355] have studied the association analysis problem in the context of **bump hunting** in multidimensional space. Specifically, they consider frequent itemset generation as the task of finding high density regions in multidimensional space. Formalizing association analysis in a statistical learning framework is another active research direction [414, 435, 444] as it can help address issues related to identifying statistically significant patterns and dealing with uncertain data [320, 333, 343].

Over the years, the association rule mining formulation has been expanded to encompass other rule-based patterns, such as, profile association rules [321], cyclic association rules [403], fuzzy association rules [379], exception rules [431], negative association rules [336, 418], weighted association rules [338, 413], dependence rules [422], peculiar rules[462], inter-transaction association

rules [353, 440], and partial classification rules [327, 397]. Additionally, the concept of frequent itemset has been extended to other types of patterns including closed itemsets [407, 456], maximal itemsets [330], hyperclique patterns [449], support envelopes [428], emerging patterns [347], contrast sets [329], high-utility itemsets [340, 390], approximate or error-tolerant itemsets [358, 389, 451], and discriminative patterns [352, 401, 430]. Association analysis techniques have also been successfully applied to sequential [326, 426], spatial [371], and graph-based [374, 380, 406, 450, 455] data.

Substantial research has been conducted to extend the original association rule formulation to nominal [425], ordinal [392], interval [395], and ratio [356, 359, 425, 443, 461] attributes. One of the key issues is how to define the support measure for these attributes. A methodology was proposed by Steinbach et al. [429] to extend the traditional notion of support to more general patterns and attribute types.

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 [415]. The SETM algorithm developed by Houtsma et al. [370] 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 [363] and M-SQL [373] query languages extend the basic SQL with new operators for mining association rules. The Mine Rule operator [394] is an expressive SQL operator that can handle both clustered attributes and item hierarchies. Tsur et al. [439] developed a generate-and-test approach called **query flocks** for mining association rules. A distributed OLAP-based infrastructure was developed by Chen et al. [341] for mining multilevel association rules.

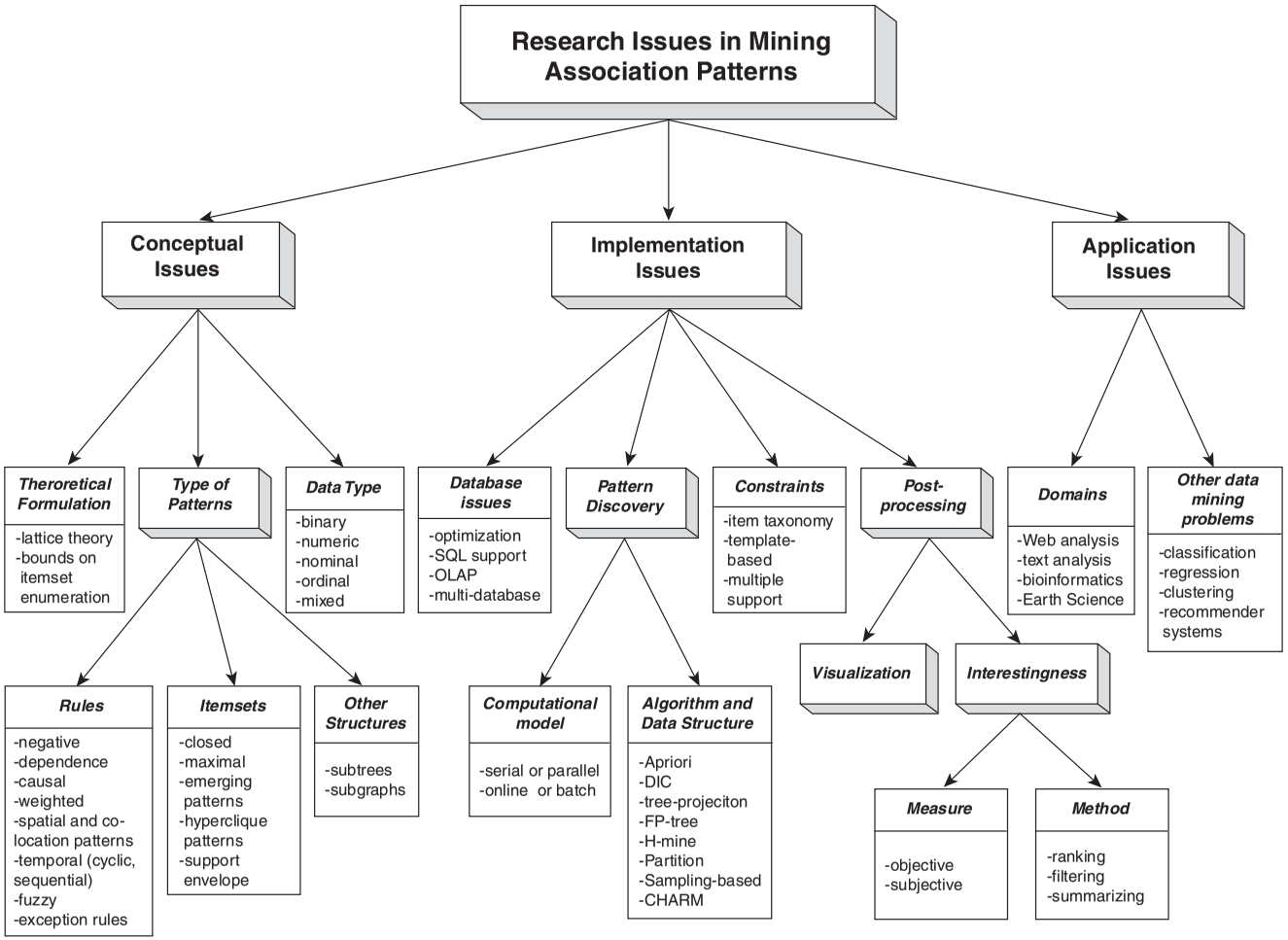


Figure 5.31. An overview of the various research directions in association analysis.

Despite its popularity, the *Apriori* algorithm is computationally expensive because it requires making multiple passes over the transaction database. Its runtime and storage complexities were investigated by Dunkel and Soparkar [349]. The FP-growth algorithm was developed by Han et al. in [364]. Other algorithms for mining frequent itemsets include the DHP (dynamic hashing and pruning) algorithm proposed by Park et al. [405] and the Partition algorithm developed by Savasere et al [417]. A sampling-based frequent itemset generation algorithm was proposed by Toivonen [436]. 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 [337] 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 [317] and H-Mine [408]. Survey articles on frequent itemset generation algorithms can be found in [322, 367]. A repository of benchmark data sets and software implementation of association rule mining algorithms is available at the Frequent Itemset Mining Implementations (FIMI) repository (http://fimi.cs.helsinki.fi).

Parallel algorithms have been developed to scale up association rule mining for handling big data [318, 360, 399, 420, 457]. A survey of such algorithms can be found in [453]. Online and incremental association rule mining algorithms have also been proposed by Hidber [365] and Cheung et al. [342]. More recently, new algorithms have been developed to speed up frequent itemset mining by exploiting the processing power of GPUs [459] and the MapReduce/Hadoop distributed computing framework [382, 384, 396]. For example, an implementation of frequent itemset mining for the Hadoop framework is available in the Apache Mahout software[1].

Srikant et al. [427] have considered the problem of mining association rules in the presence of Boolean constraints such as the following:

$$(\text{Cookies} \wedge \text{Milk}) \vee (\text{descendants}(\text{Cookies}) \wedge \neg\text{ancestors}(\text{Wheat Bread}))$$

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. [424] and Ng et al. [400] 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. [442], Liu et al. in [387], and Seno et al. [419]. In addition, constraints arising from privacy concerns of mining sensitive data have led to the development of privacy-preserving frequent pattern mining techniques [334, 350, 441, 458].

[1] http://mahout.apache.org

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. [437] proposed the idea of eliminating redundant rules using **structural rule covers** and grouping the remaining rules using clustering. Liu et al. [388] 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. [336], Bayardo and Agrawal [331], Aggarwal and Yu [323], and DuMouchel and Pregibon[348]. The properties for many of these measures were analyzed by Piatetsky-Shapiro [410], Kamber and Singhal [376], Hilderman and Hamilton [366], and Tan et al. [433]. 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 [398] by Mosteller. Meanwhile, the tea-coffee example illustrating the limitation of confidence was motivated by an example given in [336] by Brin et al. Because of the limitation of confidence, Brin et al. [336] had proposed the idea of using interest factor as a measure of interestingness. The all-confidence measure was proposed by Omiecinski [402]. Xiong et al. [449] 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. [447] 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 item pairs.

Fabris and Freitas [351] have proposed a method for discovering interesting associations by detecting the occurrences of Simpson's paradox [423]. Megiddo and Srikant [393] 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. [335] have applied the Benjamini-Hochberg [332] 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 [445], Zhang et al. [460], and Llinares-Lopez et al. [391].

Application of subjective measures to association analysis has been investigated by many authors. Silberschatz and Tuzhilin [421] 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 [385]. Cooley et al. [344] 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 [375] and neighborhood-based information [346] 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. [368] 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 [409, 432], document analysis [369], telecommunication alarm diagnosis [377], network intrusion detection [328, 345, 381], and bioinformatics [416, 446]. Applications of association and correlation pattern analysis to Earth Science studies have been investigated in [411, 412, 434]. Trajectory pattern mining [339, 372, 438] is another application of spatio-temporal association analysis to identify frequently traversed paths of moving objects.

Association patterns have also been applied to other learning problems such as classification [383, 386], regression [404], and clustering [361, 448, 452]. A comparison between classification and association rule mining was made by Freitas in his position paper [354]. The use of association patterns for clustering has been studied by many authors including Han et al.[361], Kosters et al. [378], Yang et al. [452] and Xiong et al. [448].

Bibliography

[317] 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.

[318] 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.

[319] C. Aggarwal and J. Han. *Frequent Pattern Mining*. Springer, 2014.

[320] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38, Paris, France, 2009.

[321] 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.

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

[323] 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.

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

[325] 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.

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

[327] 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.

[328] 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.

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

[330] 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.

[331] 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.

[332] 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.

[333] T. Bernecker, H. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128, Paris, France, 2009.

[334] R. Bhaskar, S. Laxman, A. D. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 503–512, Washington, DC, 2010.

[335] 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.

[336] 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.

[337] 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.

[338] 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.

[339] H. Cao, N. Mamoulis, and D. W. Cheung. Mining Frequent Spatio-Temporal Sequential Patterns. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 82–89, Houston, TX, 2005.

[340] R. Chan, Q. Yang, and Y. Shen. Mining High Utility Itemsets. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 19–26, Melbourne, FL, 2003.

[341] 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.

[342] 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.

[343] C. K. Chui, B. Kao, and E. Hung. Mining Frequent Itemsets from Uncertain Data. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 47–58, Nanjing, China, 2007.

[344] 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.

[345] 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.

[346] 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.

[347] 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.

[348] 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.

[349] 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.

[350] A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217–228, Edmonton, Canada, 2002.

[351] 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.

[352] G. Fang, G. Pandey, W. Wang, M. Gupta, M. Steinbach, and V. Kumar. Mining Low-Support Discriminative Patterns from Dense and High-Dimensional Data. *IEEE Trans. Knowl. Data Eng.*, 24(2):279–294, 2012.

[353] 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.

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

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

[356] 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.

[357] 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.

[358] R. Gupta, G. Fang, B. Field, M. Steinbach, and V. Kumar. Quantitative evaluation of approximate frequent pattern mining algorithms. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 301–309, Las Vegas, NV, 2008.

[359] E. Han, G. Karypis, and V. Kumar. Min-apriori: An algorithm for finding association rules in data with continuous attributes. *Department of Computer Science and Engineering, University of Minnesota, Tech. Rep*, 1997.

[360] 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.

[361] 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.

[362] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[363] 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.

[364] 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.

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

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

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

[368] 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.

[369] 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.

[370] 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.

[371] 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.

[372] S. Hwang, Y. Liu, J. Chiu, and E. Lim. Mining Mobile Group Patterns: A Trajectory-Based Approach. In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 713–718, Hanoi, Vietnam, 2005.

[373] 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.

[374] 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.

[375] 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.

[376] 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.

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

[378] 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.

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

[380] 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.

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

[382] N. Li, L. Zeng, Q. He, and Z. Shi. Parallel Implementation of Apriori Algorithm Based on MapReduce. In *Proceedings of the 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 236–241, Kyoto, Japan, 2012.

[383] 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.

[384] M. Lin, P. Lee, and S. Hsueh. Apriori-based frequent itemset mining algorithms on MapReduce. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, pages 26–30, Kuala Lumpur, Malaysia, 2012.

[385] 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.

[386] 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.

[387] 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.

[388] 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.

[389] J. Liu, S. Paulsen, W. Wang, A. B. Nobel, and J. Prins. Mining Approximate Frequent Itemsets from Noisy Data. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 721–724, Houston, TX, 2005.

[390] Y. Liu, W.-K. Liao, and A. Choudhary. A two-phase algorithm for fast discovery of high utility itemsets. In *Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 689–695, Hanoi, Vietnam, 2005.

[391] F. Llinares-López, M. Sugiyama, L. Papaxanthos, and K. M. Borgwardt. Fast and Memory-Efficient Significant Pattern Mining via Permutation Testing. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 725–734, Sydney, Australia, 2015.

[392] 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.

[393] 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.

[394] 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.

[395] 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.

[396] S. Moens, E. Aksehirli, and B. Goethals. Frequent Itemset Mining for Big Data. In *Proceedings of the 2013 IEEE International Conference on Big Data*, pages 111–118, Santa Clara, CA, 2013.

[397] 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.

[398] F. Mosteller. Association and Estimation in Contingency Tables. *JASA*, 63:1–28, 1968.

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

[400] 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.

[401] P. K. Novak, N. Lavrač, and G. I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *Journal of Machine Learning Research*, 10(Feb):377–403, 2009.

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

[403] 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.

[404] 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.

[405] 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.

[406] 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.

[407] 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.

[408] 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.

[409] 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.

[410] 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.

[411] 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.

[412] 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. *Journal of Geophysical Research*, 108(D17), 2003.

[413] G. D. Ramkumar, S. Ranka, and S. Tsur. Weighted association rules: Model and algorithm. In *Proc. ACM SIGKDD*, 1998.

[414] M. Riondato and F. Vandin. Finding the True Frequent Itemsets. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 497–505, Philadelphia, PA, 2014.

[415] 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.

[416] 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.

[417] 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.

[418] 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.

[419] 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.

[420] 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.

[421] 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.

[422] 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.

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

[424] 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.

[425] 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.

[426] 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.

[427] 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.

[428] 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.

[429] 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.

[430] M. Steinbach, H. Yu, G. Fang, and V. Kumar. Using constraints to generate and explore higher order discriminative patterns. *Advances in Knowledge Discovery and Data Mining*, pages 338–350, 2011.

[431] 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.

[432] 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.

[433] 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.

[434] 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.

[435] N. Tatti. Probably the best itemsets. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 293–302, Washington, DC, 2010.

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

[437] 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.

[438] I. Tsoukatos and D. Gunopulos. Efficient mining of spatiotemporal patterns. In *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, pages 425–442, 2001.

[439] 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.

[440] 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.

[441] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644, Edmonton, Canada, 2002.

[442] 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.

[443] 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.

[444] L. Wang, R. Cheng, S. D. Lee, and D. W. Cheung. Accelerating probabilistic frequent itemset mining: a model-based approach. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 429–438, 2010.

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

[446] 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.

[447] 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.

[448] 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.

[449] 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.

[450] 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.

[451] C. Yang, U. M. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–203, , San Francisco, CA, 2001.

[452] 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.

[453] 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.

[454] 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.

[455] 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.

[456] 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.

[457] 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.

[458] C. Zeng, J. F. Naughton, and J. Cai. On differentially private frequent itemset mining. *Proceedings of the VLDB Endowment*, 6(1):25–36, 2012.

[459] F. Zhang, Y. Zhang, and J. Bakos. GPApriori: GPU-Accelerated Frequent Itemset Mining. In *Proceedings of the 2011 IEEE International Conference on Cluster Computing*, pages 590–594, Austin, TX, 2011.

[460] 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.

[461] 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.

[462] 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.

5.10 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 5.20.

Table 5.20. 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:

$$\begin{aligned} \zeta(\{p_1, p_2, \ldots, p_k\}) &= \min \big[\, c\big(\{p_1\} \longrightarrow \{p_2, p_3, \ldots, p_k\}\big), \ldots \\ &\qquad c\big(\{p_k\} \longrightarrow \{p_1, p_2 \ldots, p_{k-1}\}\big) \,\big] \end{aligned}$$

   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:

$$\begin{aligned} \eta(\{p_1, p_2, \ldots, p_k\}) &= \min \big[\, c\big(\{p_2, p_3, \ldots, p_k\} \longrightarrow \{p_1\}\big), \ldots \\ &\qquad c\big(\{p_1, p_2, \ldots p_{k-1}\} \longrightarrow \{p_k\}\big) \,\big] \end{aligned}$$

   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 5.3. (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.) Assume that neither of the itemsets of a rule are empty.

6. Consider the market basket transactions shown in Table 5.21.

   (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 $minsup > 0$)?

Table 5.21. 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}

(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. Show that if a candidate k-itemset X has a subset of size less than $k-1$ that is infrequent, then at least one of the $(k-1)$-size subsets of X is necessarily infrequent.

8. 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.

9. 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 data set shown in Table 5.22 with $minsup = 30\%$, i.e., any itemset occurring in less than 3 transactions is considered to be infrequent.

Table 5.22. 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	$\{b, d\}$

(a) Draw an itemset lattice representing the data set given in Table 5.22. 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)?

10. 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 5.32.

(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?

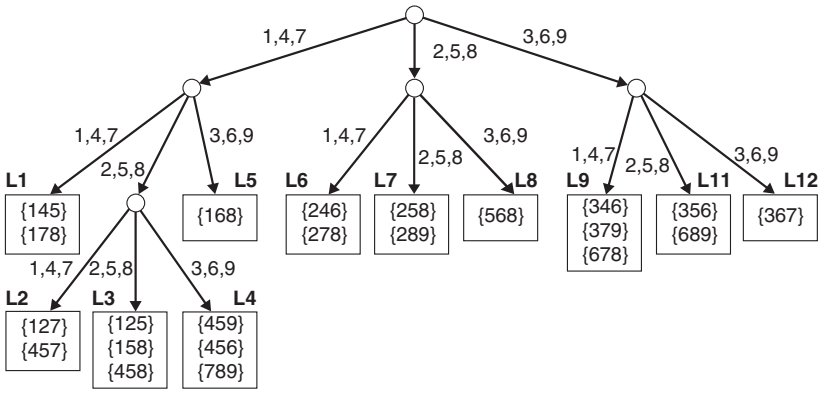


Figure 5.32. An example of a hash tree structure.

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

11. 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 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.

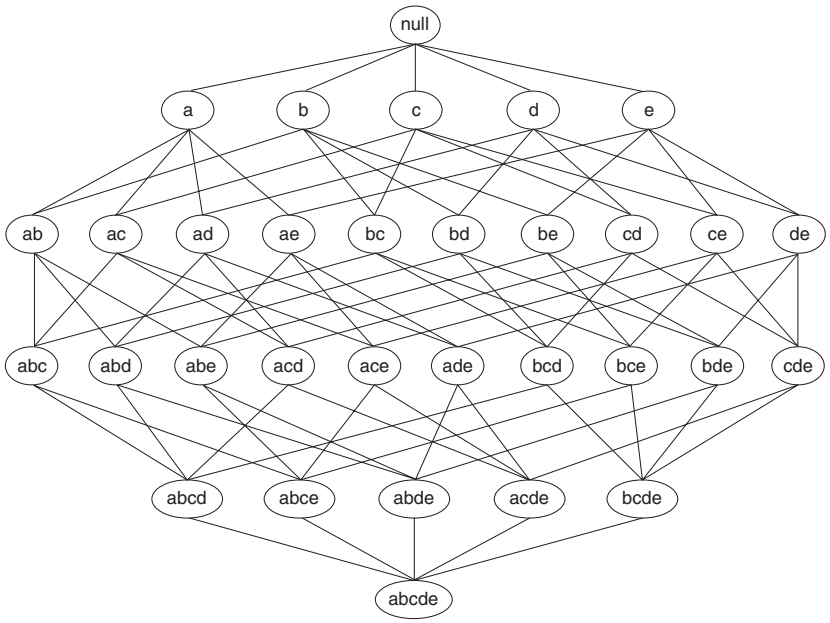


Figure 5.33. An itemset lattice

(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?

12. Given the lattice structure shown in Figure 5.33 and the transactions given in Table 5.22, 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%.

13. 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 5.23.

Table 5.23. 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$(X \longrightarrow Y) = \sqrt{P(X,Y)} \times \max(P(Y|X) - P(Y), P(X|Y) - P(X))$, where $P(Y|X) = \frac{P(X,Y)}{P(X)}$.

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

14. Given the rankings you had obtained in Exercise 13, 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?

15. Answer the following questions using the data sets shown in Figure 5.34. 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%)?

16. (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 $P(A,B) \times P(\overline{A},\overline{B}) = P(A,\overline{B}) \times P(\overline{A},B)$.

(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 Table 5.9 when the variables are statistically independent.

17. 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?

18. Suppose we have market basket data consisting of 100 transactions and 20 items. Assume 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.

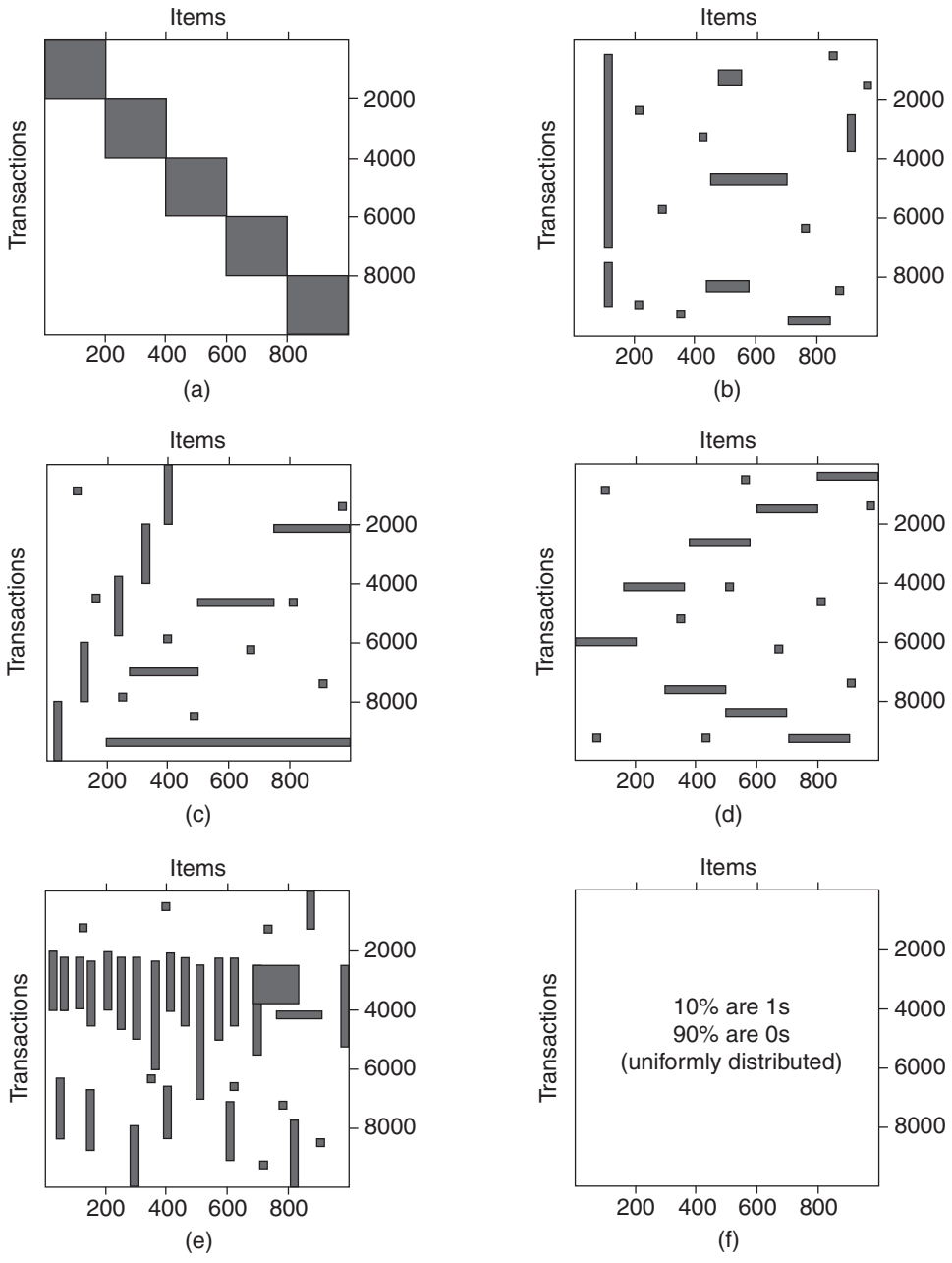


Figure 5.34. Figures for Exercise 15.

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

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

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

19. Table 5.24 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 5.24. 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?

20. Consider the contingency tables shown in Table 5.25.

(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 $B \rightarrow A$.

(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 $B \rightarrow A$.

Table 5.25. Contingency tables for Exercise 20.

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

(a) Table I.

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

(b) Table II.

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

21. Consider the relationship between customers who buy high-definition televisions and exercise machines as shown in Tables 5.17 and 5.18.

    (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.