Chapter 2
Association Rules Mining in Inventory Database

Association rules mining is an important topic in data mining which is the discovery of association relationships or correlations among a set of items. It can help in many business decision-making processes, such as catalog design, cross-marketing, cross-selling and inventory control. This chapter reviews some of the essential concepts related to association rules mining, which will be then applied to the real inventory control system in Chapter 12. Some related research into development of mining association rules are also introduced.

This chapter is organized as follows. In Section 2.1 we begin with explaining briefly the background of association rules mining. In Section 2.2, we outline certain necessary basic concepts of association rules. In Section 2.3, we introduce the Apriori algorithm, which can search frequent itemsets in large databases. Section 2.4 introduces some research into development of mining association rules in an inventory database. Finally, we summarize this chapter in Section 2.5.

2.1 Introduction

Data mining is a process of discovering valuable information from large amounts of data stored in databases. This valuable information can be in the form of patterns, associations, changes, anomalies and significant structures (Zhang and Zhang 2002). That is, data mining attempts to extract potentially useful knowledge from data. Therefore data mining has been treated popularly as a synonym for knowledge discovery in databases (KDD). The emergence of data mining and knowledge discovery in databases as a new technology has occurred because of the fast development and wide application of information and database technologies.

One of the important areas in data mining is association rules mining. Since its introduction in 1993 (Agrawal *et al.* 1993), the area of association rules mining has received a great deal of attention. Association rules mining finds interesting association or correlation relationships among a large set of data items. With massive

amounts of data continuously being collected and stored, many industries are becoming interested in mining association rules from their database. The discovery of interesting association relationships among huge amounts of business transaction records can help in many business decision-making processes, such as catalog design, cross-marketing, cross-selling and inventory control. How can we find association rules from large amounts of data, either transactional or relational? Which association rules are the most interesting? How can we help or guide the mining procedure to discover interesting associations? In this chapter we will explore each of these questions.

A typical example of association rules mining is market basket analysis. For instance, if customers are buying milk, how likely are they to also buy bread on the same trip to the supermarket? Such information can lead to increased sales by helping retailers to selectively market and plan their shelf space, for example, placing milk and bread within single visits to the store. This process analyzes customer buying habits by associations between the different items that customers place in their shopping baskets. The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. The results of market basket analysis may be used to plan marketing or advertising strategies, as well as store layout or inventory control. In one strategy, items that are frequently purchased together can be placed in close proximity in order to further encourage the sale of such items together. If customers who purchase milk also tend to buy bread at the same time, then placing bread close to milk may help to increase the sale of both of these items. In an alternative strategy, placing bread and milk at opposite ends of the store may entice customers who purchase such items to pick up other items along the way (Han and Micheline 2001). Market basket analysis can also help retailers to plan which items to put on sale at reduced prices. If customers tend to purchase coffee and bread together, then having a sale on coffee may encourage the sale of coffee as well as bread.

If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of the item. Each basket can then be represented be a Boolean vector of values assigned to these variables. The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together. These patterns can be represented in the form of association rules. For example, the information that customers who purchase milk also tend to buy bread at the same time is represented in association rule as following form:

$$buys(X, "milk") \Rightarrow buys(X, "bread")$$

where X is a variable representing customers who purchased such items in a transaction database. There are a lot of items can be represented by the form above however most of them are not interested. Typically, association rules are considered interesting if they satisfy several measures that will be described below.

2.2 Basic Concepts of Association Rule

Agrawal *et al.* (1993) first developed a framework to measure the association relationship among a set of items. The association rule mining can be defined formally as follows.

$I = \{i_1, i_2, \cdots i_m\}$ is a set of items. For example, goods such as milk, bread and coffee for purchase in a store are items. $D = \{t_1, t_2, \cdots t_n\}$ is a set of transactions, called a *transaction database*, where each transaction t has an identifier *tid* and a set of items *t-itemset*, *i.e.*, $t = (tid, t\text{-}itemset)$. For example, a customer's shopping cart going through a checkout is a transaction. X is an *itemset* if it is a subset of I. For example, a set of items for sale at a store is an itemset.

Two measurements have been defined as *support* and *confidence* as below. An itemset X in a transaction database D has a *support*, denoted as $sp(X)$. This is the ratio of transactions in D containing X,

$$sp(X) = \frac{|X(t)|}{|D|}$$

where $X(t) = \{t \text{ in } D | t \text{ contains } X\}$.

An itemset X in a transaction database D is called frequent if its support is equal to, or greater than, the threshold minimal support (*min_sp*) given by users. Therefore support can be recognized as frequencies of the occurring patterns.

Two itemsets X and Y in a transaction database D have a confidence, denoted as $cf(X \Rightarrow Y)$. This is the ratio of transactions in D containing X that also contain Y.

$$cf(X \Rightarrow Y) = \frac{|(X \cup Y)(t)|}{|X(t)|} = \frac{sp(X \cup Y)}{sp(X)} .$$

Because the confidence is represented as a conditional probability of both X an Y, having been purchased under the condition if X had been purchased, then the confidence can be recognized as the strength of the implication of the form $X \Rightarrow Y$.

An *association rule* is the implication of the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$. Each association rule has two quality measurements, *support* and *confidence*, defined as: (1) the support of a rule $X \Rightarrow Y$ is $sp(X \cup Y)$ and (2) the confidence of a rule $X \Rightarrow Y$ is $cf(X \Rightarrow Y)$.

Rules that satisfy both a minimum support threshold (*min_sp*) and a minimum confidence threshold (*min_cf*), which are defined by users, are called strong or valid. Mining association rules can be broken down into the following two subproblems:

1. Generating all itemsets that have support greater than, or equal to, user specified minimum support. That is, generating all frequent itemsets.
2. Generating all rules that have minimum confidence in the following simple way: for every frequent itemset X, and any $B \subset X$, let $A = X - B$. If the confidence of a rule $A \Rightarrow B$ is greater than, or equal to, the minimum confidence (*min_cf*), then it can be extracted as a valid rule.

Table 2.1 A transaction database

tid	Items
$t001$	I_2, I_3, I_5
$t002$	I_1, I_2, I_3, I_5
	I_2, I_3, I_5
$t003$	I_1, I_2, I_3, I_5
$t004$	I_2, I_5

To demonstrate the support-confidence framework, we use a database example from the supermarket shown in Table 2.1.

In Table 2.1, let the item universe be $I = \{I_1, I_2, I_3, I_4, I_5\}$ and transaction universe be $tid = \{t001, t002, t003, t004\}$. Therfore, *tid* identifies uniquely a transaction in which several *items* have been purchased. Association rule $X \Rightarrow Y$ has a support $sp(X \cup Y)$ equal to the frequencies of X and Y items purchased together in transaction database D, and a confidence $cf(X \Rightarrow Y)$ equal to the ratio of how many times X items were purchased with Y items. There are many association rules that can be constructed by combining various items. For example, using only 10 items can make about 57,000 association rules (numbers of association rules can be calculated by $\sum_{k=2}^{m} C_m^k \cdot (2^k - 2)$, where selecting k items from m items produces association rules). However, not all of the association rules are interesting in practice; only parts of the association rules are valid. Under the consideration of the support-confidence framework, the association rules which have larger frequencies (their supports are larger than *min_sp*) and stronger relationships (their confidences are larger than *min_cf*) can be recognized as being valid. For example, let *min_sp* $= 50\%$ (to be frequent, an itemset must occur in at least two transactions in the above transaction database), and *min_cf* $= 60\%$ (to be a high-confidence or valid rule, at least 60% of the time you find the antecedent of the rule in the transactions; you must also find the consequence of the rule there). We can generate all frequent itemsets in Table 2.1 as follows. By scanning the database D, item $\{I_1\}$ occurs in the two transactions, $t001$ and $t003$. Its frequency is 2, and its support $sp\{I_1\} = 50\%$ is equal to *min_sp* $= 50\%$. Therefore $\{I_1\}$ is a frequent item. Similarly, we can find that $\{I_2\}, \{I_3\}$ and $\{I_5\}$ are frequent items but $\{I_4\}$ is not a frequent item (because $sp\{I_1\} = 25\%$ is smaller than *min_sp*).

Now consider two-item sets in Table 2.1, where 8 two-item sets exist (because the combination of 2 items from all items (5 items) is 10, there should be 10 two-item sets, however 2 two-item sets do not appear in the transaction database). For example, $\{I_1, I_2\}$ occurs in the one transaction ($t003$). Its frequency is 1, and its support, $sp\{I_1 \cup I_2\} = 25\%$, which is less than *min_sp* $= 50\%$. Therefore $\{I_1, I_2\}$ is not a frequent itemset. On the other hand, itemset $\{I_1, I_3\}$ occurs in the two transactions ($t001$ and $t003$), its frequency is 2, and its support, $sp\{I_1 \cup I_3\} = 50\%$, which is equal to *minsupp* $= 50\%$. Therefore $\{I_1, I_3\}$ is a frequent itemset. Similarly, we can find that $\{I_2, I_3\}$, $\{I_2, I_5\}$ and $\{I_3, I_5\}$ are frequent itemsets but $\{I_1, I_4\}$, $\{I_1, I_5\}$, $\{I_2, I_5\}$ and $\{I_3, I_4\}$ are not frequent itemsets.

We also need to consider three-item sets in Table 2.1, where 5 three-item sets exist (because the combination of 3 items from all items (5 items) is 10, there should be 10 three-item sets, however five two-item sets do not appear in the transaction database). For example, $\{I_1, I_2, I_3\}$ occurs in the one transaction ($t003$). Its frequency is 1, and its support, $sp\{I_1 \cup I_2 \cup I_3\} = 25\%$, which is less than *min_sp* $= 50\%$. Therefore $\{I_1, I_2, I_3\}$ is not a frequent itemset. On the other hand, itemset $\{I_2, I_3, I_5\}$ occurs in the two transactions ($t002$ and $t003$), its frequency is 2, and its support, $sp\{I_2 \cup I_3 \cup I_5\} = 50\%$, which is equal to *minsupp* $= 50\%$. Therefore $\{I_2, I_3, I_5\}$ is a frequent itemset. In the same way, we can find that all three-item sets are not frequent except itemset $\{I_2, I_3, I_5\}$.

Similarly, four-items and five-items set also should be considered in Table 2.1. Only one four-item set $\{I_1, I_2, I_3, I_5\}$ exists but it is not frequent, and there are no five-item sets.

According to the above definition, $\{I_1\}$, $\{I_2\}$, $\{I_3\}$, $\{I_5\}$, $\{I_1, I_3\}$, $\{I_2, I_3\}$, $\{I_2, I_5\}$, $\{I_3, I_5\}$ and $\{I_2, I_3, I_5\}$ in Table 2.1 are frequent itemsets.

When the frequent itemsets are determined, generating association rules from the frequent itemsets is easy. For example, consider frequent three-item set $\{I_2, I_3, I_5\}$, because

$$cf\{I_2 \cup I_3 \Rightarrow I_5\} = \frac{sp(I_2 \cup I_3 \cup I_5)}{sp(I_2 \cup I_3)} = \frac{2}{2} = 100\% \ .$$

Since this is greater than *min_cf* $= 60\%$, $I_2 \cup I_3 \Rightarrow I_5$ can be extracted as a valid rule. In the same way, $I_2 \cup I_5 \Rightarrow I_3$ and $I_3 \cup I_5 \Rightarrow I_2$ are also valid association rules but the relationships among items are different (because $cf\{I_2 \cup I_5 \Rightarrow I_3\} = 66.7\%$ and $cf\{I_3 \cup I_5 \Rightarrow I_2\} = 100\%$). Also because

$$cf\{I_2 \Rightarrow I_3 \cup I_5\} = \frac{sp(I_2 \cup I_3 \cup I_5)}{sp(I_2)} = \frac{2}{3} = 66.7\% \ .$$

This is greater than *min_cf* $= 60\%$, and so $I_2 \Rightarrow I_3 \cup I_5$ can be extracted as a valid rule. In the same way, $I_2 \Rightarrow I_5 \cup I_3$ and $I_3 \Rightarrow I_5 \cup I_2$ are also valid association rules.

Similarly, we can find that $I_1 \Rightarrow I_3$ and $I_3 \Rightarrow I_1$ are valid from the frequent itemset $\{I_1, I_3\}$. Considering the association rules exist over two-item itemsets, generating all over two-item frequent itemsets, we can obtain all of association rules.

It is not clear which itemsets are frequent, and then which and how many association rules are valid if we do not investigate all of frequent itemsets and association rules. As mentioned above, there are too many candidate itemsets to search all of the itemsets. Therefore mining all valid frequent itemsets automatically from a transaction database is very important. In fact, it is a main research topic in data mining studies. Agrawal *et al.* (1993) have built an Apriori algorithm for mining association rules from databases. This algorithm has since become a common model for mining association rules. In this chapter, we just introduce the Apriori algorithm to show how to reduce the search space of frequent itemsets. In fact, there is a lot of research on mining association rules. Detailed overviews of mining association rule algorithms can be found in Zhang and Zhang (2002), Zhang *et al.* (2004) and Han and Micheline (2001).

2.3 Mining Association Rules

Identifying frequent itemsets is one of the most important issues faced by the knowledge discovery and data mining community. There have been a number of excellent algorithms developed for extracting frequent itemsets in very large databases. Apriori is a very famous and widely used algorithm for mining frequent itemsets (Agrawal *et al.* 1993). For efficiency, many variations of this approach have been constructed. To match the algorithms already developed, we first present the Apriori algorithm, and then present selectively several related algorithms of mining special association rules.

2.3.1 The Apriori Algorithm: Searching Frequent Itemsets

Apriori is an influential algorithm for mining frequent itemsets. The algorithm employs an iterative approach known as a level-wise search, where k-itemsets are used to explore $(k+1)$-itemsets. To improve the efficiency of the level-wise generation of frequent itemsets, an important property called Apriori property is used to reduce the search: all non-empty subsets of a frequent itemset must also be frequent. This property belongs to a special category of properties called antimonotone, in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called antimonotone because the property is monotonic in the context of failing a test. That is, there are itemsets I and J where $I \subset J$ then $sp(I) \geq sp(J)$. That means if itemset I is not frequent then (for example in Table 2.1, $\{I_4\}$ is not a frequent item) itemset J, which included I, is also not frequent (for example in Table 2.1 $\{I_1, I_4\}$ and $\{I_3, I_4\}$ are not frequent itemsets).

By using the Apriori property, first the set of frequent one-item sets are found. This set is denoted L_1. L_1 can be used to find L_2, the set of frequent two-item sets in the following way. A set of candidate one-item sets is generated by joining L_1 with itself, which is denoted C_1. That is, C_1 is a superset of L_1, its members may or may not be frequent but all of the frequent one-item sets are included in C_1. A scan of the database to determine the count of each candidate in C_1 would result in the determination of L_1. Any one-item set that is not frequent cannot be a subset of a frequent two-item set. It is true for any k-itemsets. Hence, if any $(k-1)$-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. Then L_2 can be used to find L_3, and so on, until no more frequent k-itemsets can be found. Therefore, we need to scan fully the database k times when searching for frequent k-itemsets.

The following algorithm in Figure 2.1 is used to generate all frequent itemsets in a given database D. This is the Apriori algorithm.

In the Apriori algorithm shown in Figure 2.1, step 1 finds the frequent one-item sets, L_1. Then L_{k-1} is used to generate candidates C_k in order to find L_k in steps 2 through 5. The *apriori_gen* procedure generates the candidates and then uses the apriori property to eliminate those having a subset that is not frequent, which is shown in detail in Figure 2.2.

Algorithm Apriori (D, *min_sp*)
Input: D, *min_sp*
Output: Answer
Begin
1) $L_1 = \{\text{large one-item sets}\}$;
2) For $(k = 2; L_{k-1} \neq \theta; \text{k} + +)$ do {
3) C_k = *apriori_gen* (L_{k-1}); // New candidates
4) $L_k = \{c \in C_k \mid c.\text{support} \geq min\_sp\}$
5) }
6) Answer $= \cup L_k$;
End

Figure 2.1 Main program of the Apriori algorithm

Function *apriori_gen*(L_{k-1})
Input: L_{k-1}
Output: C_k
Begin
For (each pair $p, q \in L_{k-1}$) do {
If
$p.item_1 = q.item_1 \wedge \cdots \wedge p.item_{k-1} = q.item_{k-1} \wedge p.item_{k-1} < q.item_{k-1}$
Then
$c = p \cup q.item_{k-1}$
$C_k = C_k \cup \{c\}$
}
For (each $p \in C_k$) do {
For (each k-subsets s of c) do {
If $c \notin L_{k-1}$ Then
Delete c
}
}
return C_k
End

Figure 2.2 Algorithm of function *apriori_gen*(L_{k-1})

In Figure 2.2, the *apriori_gen* function performs two kinds of actions. The first action is the join component in which L_{k-1} is joined with L_{k-1} to generate potential candidates. The second action employs the Apriori property to remove candidates that have a subset that is not frequent. An example to show the action of *apriori_gen* can be illustrated as follows with Table 2.1. Let $L_1 = \{I_1\}, \{I_2\}, \{I_3\}, \{I_5\}$, which consists of all frequent one-item sets. In order to join with L_1 to generate a candidate set of two-item sets, each pair of $\{I_1\}, \{I_2\}, \{I_3\}, \{I_5\}$ are combined as $\{I_1, I_2\}, \{I_1, I_3\}, \{I_1, I_5\}, \{I_2, I_3\}, \{I_2, I_5\}$ and $\{I_3, I_5\}$ as C_2. Consider the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that those six candidates may remain in C_2. Then the set of frequent two-item sets, L_2, is determined, consisting of $\{I_1, I_3\}, \{I_2, I_3\}, \{I_2, I_5\}, \{I_3, I_5\}$, which have support greater than or equal to *min_sp*. Similarly, the set of candidate three-item sets, C_2, can also be generated with L_2. Each pair of $\{I_1, I_3\}, \{I_2, I_3\}$,

$\{I_2, I_5\}, \{I_3, I_5\}$ are combined as $\{I_1, I_2, I_3\}, \{I_1, I_2, I_5\}, \{I_1, I_3, I_5\}$ and $\{I_2, I_3, I_5\}$ as C_3. However, we can find that $\{I_1, I_2\}$ and $\{I_1, I_5\}$ are not frequent two-item sets. Therefore we remove them from C_3, to save the effort of unnecessarily obtaining their counts during the subsequent scan of the database to determine L_3 because C_3 only includes $\{I_2, I_3, I_5\}$. Note that when given a candidate k-itemset, we only need to check if its $(k-1)$-subsets are frequent since the Apriori algorithm uses a level-wise search strategy. This is because there is only one itemset in L_3 so that the *apriori_gen* function is completed.

2.3.2 Generating Association Rules from Frequent Itemsets

Once the frequent itemsets have been searched, it is straightforward to generate association rules. Notice that $sp(I) \geq sp(J)$ where itemsets I are the non-empty subsets of J (*i.e.*, $I \subset J$); if J is a frequent itemset, then for every subset I the association rule $I \Rightarrow J - I$ can be established if $cf\{I \Rightarrow J - I\} \geq \min\_cf$. According to this property, the following algorithm in Figure 2.3 can be used to generate all of association rules.

Generating association rule
Input: Frequent itemsets
Output: Association rules
Begin
1. For each frequent itemsets l_k $(k \geq 2)$ do {
2. $H_1 := \{h \in l_k\} | cf(l_k - \{h\} \Rightarrow \{h\}) \geq \min\_cf\}$
3. Call **Ap-Genrule**(l_k, H_1);
4. }

5. **Procedure Ap-Genrule**(l_k, H_m) {
6. If $k > m + 1$){
7. $H_{m+1} = apriori\_gen(H_m)$;
8. For each $h_{m+1} \in H_{m+1}$ {
9. $cf = sp((l_k)/\, sp(l_k - h_{m+1})$
10. If $cf \geq \min\_cf$ then
11. *Output* $(l_k - h_{m+1}) \Rightarrow h_{m+1}$;
12. Else
13. $H_{m+1} := H_{m+1} - \{h_{m+1}\}$;
14. }
15. **Ap-Genrule**(l_k, H_{m+1});
16. }
17. }
End

Figure 2.3 Algorithm to generate association rules

In Figure 2.3, H_m is the consequent set of m which has larger confidence than *min_cf*. First in step 2, the consequent sets of one item should be made. For example, by using the data from Table 2.1, $\{I_2, I_3\}$ is a two-item frequent itemset. Suppose

$min\_cf = 60\%$, then

$$cf(I_2 \Rightarrow I_3\} = 66.7\% > \min\_cf$$
$$cf(I_3 \Rightarrow I_2\} = 66.7\% > \min\_cf\ .$$

For generating larger H_{m+1}, we can use the function of the *apriori_gen* (H_m) in step 7, and calculate the confidence of each. If their confidence is larger than *min_cf* then output them as association rules; else withdraw them from H_{m+1} (in steps 8–14). We can also use an example to illustrate the algorithm. Because $\{I_2, I_3\}$ is a frequent itemset then *apriori_gen* $(\{I_2, I_3\})$ makes several three-item sets, but only $\{I_2, I_3, I_5\}$ is frequent. Calculate the confidence of the subset of itemset $\{I_2, I_3, I_5\}$ to give

$$cf(I_2 \cup I_3 \Rightarrow I_5\} = 100\%$$
$$cf(I_2 \cup I_5 \Rightarrow I_3\} = 66.7\%$$
$$cf(I_3 \cup I_5 \Rightarrow I_2\} = 100\%$$
$$cf(I_2 \Rightarrow I_3 \cup I_5\} = 66.7\%$$
$$cf(I_3 \Rightarrow I_2 \cup I_5\} = 66.7\%$$
$$cf(I_5 \Rightarrow I_2 \cup I_3\} = 66.7\%\ .$$

Because the confidence of the six subsets of $\{I_2, I_3, I_5\}$ is larger than *min_cf*, they then become association rules. Moreover, there are no frequent itemsets of four items in Table 2.1 so that the procedure of generating association rules is completed.

2.4 Related Studies on Mining Association Rules in Inventory Database

Since the Apriori is a very basic algorithm for mining frequent itemsets in large databases, many variations of the Apriori have been proposed that focus on improving the efficiency of the original algorithm and on the developing the algorithm for other research fields. There are many excellent publications that summarize this topic, for example, Zhang and Zhang (2002), Zhang *et al.* (2004) and Han and Micheline (2001). In this section, we discuss two typical approaches of mining association rules established in current literature: mining multidimensional association rules from relational databases (Han and Micheline 2001, Fukuda *et al.* 1996a, 1996b, 1999, 2001), and mining association rules with a time-window (Xiao *et al.* 2009), which are considered to be very important in inventory management.

2.4.1 Mining Multidimensional Association Rules from Relational Databases

We have studied association rules that imply a single predicate, for example, the predicate buys. Hence we can refer to the association rules with a form of $A \Rightarrow B$

as a one-dimensional association rule because it contains a single distinct predicate (for example buys) with multiple occurrences. Such association rules can commonly be mined from transactional data.

However, rather than using a transactional database, sales and related information are stored in a relational database. Such a store of data is multidimensional, by definition. For instance, in addition to keeping track of the items purchased in sales transactions, a relational database may record other attributes associated with the items, such as the quantity purchased or the price, or the branch location of the sales. Additional relational information regarding the customers who purchased the items, such as customer age, occupation, credit rating, income, and address, may also be stored. Considering each database attribute as a predicate, it can therefore be interesting to mine association rules containing multiple predicates. Association rules that involve two or more dimensions or predicates can be referred to as multidimensional association rules, such as

$$age(X, \text{"20...29"}) \wedge occupation(X, \text{"}student\text{"}) \Rightarrow buys(X, \text{"}laptop\text{"})$$

where X is a variable representing customers who purchased items in relational databases. There are three predicates (age, occupation, and buys) in the above association rules. Note that these predicates can be categorical (nominal attributes) or quantitative (numeric attributes). Here, we just consider a simple case in which quantitative attributes are separated using predefined concept hierarchies, namely optimized association rules, which is proposed by Fukuda *et al.* (1996b, 1999, 2001). This rule has a simple form

$$A \in [v_1, v_2] \Rightarrow B$$

which states that customers who buy item A in the range between v_1 and v_2 are likely to buy item B. If the resulting task-relevant data are stored in a relational table, then the Apriori algorithm requires just a slight modification so as to find all frequent ranges rather than frequent itemsets. Also if an instance of the range is given, the confidence of this rule can be calculated easily. In practice, however, we want to find a range that yields a confident rule. Such a range is called a *confident* range. Unfortunately, a confident range is not always unique, and we may find a confident range that contains only a very small number of items. If the confident range has a maximum support then the association rule is called an *optimized support* rule. This range captures the largest cluster of customers that are likely to buy item B with a probability no less than the given *min_cf*. Instead of the optimized support rule it is also interesting to find the frequent range that has a maximum confidence. Such association rules are called *optimized confidence* rules. This range clarifies a cluster of more than, for instance, 10% of customers that buy item B with the highest confidence.

Tables 2.2 and 2.3 show the examples of the optimized support rule and optimized confidence rule. Suppose $min\_sp = 10\%$ and $min\_cf = 50\%$. We may have many instances of ranges that yield confident and ample rules, shown in Tables 2.2 and 2.3.

Table 2.2 Examples of confidence rules

Range	[1000, 10 000]	[5000, 5500]	[500, 7000]
Support	20%	2%	15%
Confidence	50%	55%	52%

Among those ranges in Table 2.2, range [1000, 10 000] is a candidate range for an optimized support rule.

Table 2.3 Examples of ample rules

Range	[1000, 5000]	[2000, 4000]	[3000, 8000]
Support	13%	10%	11%
Confidence	65%	50%	52%

Among those ranges in Table 2.3, range [1000, 5000] is a candidate range for an optimized confidence rule. It can be considered that although [1000, 5000] is a superset of [2000, 4000], the confidence of the rule of the former range is greater than that of the latter range, but observation will confirm that corresponding situations could really occur.

Fukuda *et al.* (1996b, 1999, 2001) proposed two asymptotically optimal algorithms to find optimized support rules and optimized confidence rules, on the assumption that data are sorted with respect to the numeric attributes. Sorting the database, however, could create a serious problem if the database is much larger than the main memory, because sorting data for each numeric attribute would take an enormous amount of time. To handle giant databases that cannot fit in the main memory, they presented other algorithms for approximating optimized rules, by using randomized algorithms. The essence of those algorithms is that they generated thousands of almost equidepth buckets (*i.e.*, buckets are made by dividing the value of the attribute into a sequence of disjointed ranges, and the size of any bucket is the same), and then combines some of those buckets to create approximately optimized ranges. In order to obtain such almost equidepth buckets, a sample of data is first created that fits into the main memory, thus ensuring the efficiency with which the sample is sorted.

2.4.2 Mining Association Rules with Time-window

Most of the early studies on mining association rules focus on the quantitative property of transaction. The time property of transaction, however, is another important feature that has also attracted many studies in recent years. Transaction time is believed to be valuable for discovering a customer's purchasing patterns over time,

e.g., finding periodic association rules. Periodic association rules were first studied by Ramaswamy and Siberschatz (1998) to discover the association rules that repeat in every cycle of a fixed time span. Li *et al.* (2003) presented a level-wise Apriori-based algorithm, named Temporal-Apriori, to discover the calendar-based periodic association rules, where the periodicity is in the form of a calendar, *e.g.*, day, week or month. Lee and Jiang (2008) relaxed the restrictive crisp periodicity of the periodic association rule to a fuzzy one, and developed an algorithm for mining fuzzy periodic association rules.

The periodic association rule is based on the assumption that people always do the same purchasing activities after regular time intervals so that some association rules might not hold on the whole database but on a regularly partitioned database. However, the key drawback of the mining periodic association rule is that the periodicities have to be user-specified, *e.g.*, days, weeks or months, which limit the ability of algorithms on discovering a customer's arbitrary time-cycled activities with unknown interval. Moreover, a customer's purchasing patterns over time are more complex than just periodically repeated behaviors. Many association rules might appear occasionally or repeat asynchronously (Huang and Chang 2005) or with changing periodicities. Finding all of these association rules which appear in different time segments, and then using them to discover all potential patterns of a customer is challenging work.

In practice, there are many candidates of association rules that do not satisfy the thresholds of *min_sp* and *min_cf* over the whole database, but satisfy them in segments of the database partitioned by specified time-windows, which are also called *part-time* association rules. These *part-time* association rules reflect the customer's purchasing pattern at different time phases, which is useful information for timely market management because market managers need to know the behaviors of customers on different time phases. However, these rules cannot be discovered by all of the existing algorithms, including those algorithms for mining periodic association rule. To discover *part-time* association rules, we first need to introduce the notion of the association rule with time-windows (ARTW), which is formally described as follows.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. $D = \{t_1, t_2, \cdots t_n\}$ is a set of transactions, where each transaction t has an identifier *tid* and a set of items *t-itemset*, *i.e.*, $t =$ (*tid, t-itemset*). A time stamp t_t indicating the time when the transaction occurred, and a set of items t_c such that $t_c \in I$. Let T be the total time span of the real-time database such that $t_t \in T$ for all transactions. Let (t_1, t_2) be a time-window such that $t_1, t_2 \in T$. Let $|D^{(t_1,t_2)}|$ be the number of transactions that occur in (t_1, t_2), where $|D(X)^{(t_1,t_2)}|$ is the number of transactions containing itemset X occurring in (t_1, t_2), and $|(t_1, t_2)|$ is the width of the time-window. An ARTW is an implication of the form $X \Rightarrow^{(t_1,t_2)} Y$, where $X \subseteq I, Y \subseteq I, X \cap Y = \phi$, and $(t_1, t_2) \in T$. This rule has support $s\%$ in time-window (t_1, t_2) if and only if

$$\frac{|D(X)^{(t_1,t_2)}|}{|D^{(t_1,t_2)}|} \geq s\% \ ,$$

and has conference $c\%$ in time-window (t_1, t_2) if and only if

$$\frac{|D(X \cup Y)^{(t_1,t_2)}|}{|D(X)^{(t_1,t_2)}|} \geq c\% \ .$$

As usual, two thresholds, *i.e.*, the *min_sp* and the *min_cf*, are required to determine whether an ARTW holds or not. Besides, another threshold, namely, the minimum time-window, noted as *min_win*, is also required in determining an ARTW. Generally, an association rule with too narrow of a time-window is often meaningless to market management because it does not reflect a stable purchase pattern of customer. Therefore, we use the following criteria to determine whether an ARTW: $X \Rightarrow^{(t_1,t_2)} Y$ holds or not:

1. Its support $s\%$ is greater than or equal to the predefined *min_sp*.
2. Its conference $c\%$ is greater than or equal to the predefined *min_cf*.
3. The width of time-window $|(t_1, t_2)|$ is greater than or equal to the predefined *min_win*.

In addition, to avoid the algorithm tramping in the situation of *one transaction is frequent*, the *min_win* is required to be much greater than $1/min\_sp$.

An ARTW might have many pairs of disjointed time-windows with different widths and different intervals. For example, the association rule may hold in the time-window of (t_1, t_2), (t_3, t_4), ... and so on. Therefore, the periodic association rules are just a subset of the ARTW that are with fixed time-window width and fixed length of interval. Especially, when the *min_win* is equal to the total time span of the whole database, the found ARTW are exactly the traditional notion of association rules that hold on the whole database.

An important index of ARTW, *i.e.*, the *tc%* named time-coverage rate, is employed to represent the strength of ARTW on time length, which is defined as follows:

$$tc\% = \frac{|\,(t_1, t_2)_{A \Rightarrow B}\,|}{|T|} \times 100\% \ ,$$

where $|(t_1, t_2)|_{A \Rightarrow B}$ is the length of the time-window of ARTW $A \Rightarrow^{(t_1,t_2)} B$, and $|T|$ is the total time span of the database. Therefore, an ARTW holds if and only if its time-coverage rate is equal to or greater than the $min\_win/|T|$. Since an association rule may be associated with many different and disjointed time-windows, the same association rule with different time-windows may be recognized as different ARTW by the algorithm. So we merge those ARTW with identical itemset A and itemset B into one. Therefore, an ARTW always indicates an association rule $A \Rightarrow^{\{(t_1,t_2)\}} B$ that holds on a set of disjointed time-windows. The time-coverage of ARTW is consequently changed to:

$$tc\% = \frac{\sum |\,(t_1, t_2)_{A \Rightarrow B}\,|}{|T|} \times 100\% \ .$$

Especially, when $tc\% = 100\%$, the ARTW is degraded to a traditional *full-time* association rule. If we relax the time-coverage rate from 100% to lower values, like 80% or 50%, more association rules are expected to be discovered.

In order to discover all ARTW from a real-time database, as usual, two subproblems are involved. The first is to find all of the frequent itemsets with time-window (FITW), which accordingly indicates the itemsets that are frequent only in a specified time-window. The second is to find out, from the set of FITW discovered in step one, all of ARTW in the database. Since the solution to the second subproblem is straightforward, our research efforts mainly focus on the first subproblem to search all FITW from a real-time database with respect to the specified thresholds of *min_sp*, *min_cf* and *min_win*, efficiently and completely, which can be described as follows.

For a pair of time-windows $(t_1, t_2) \in T$, let $|D^{(t_1,t_2)}|$ be the number of transactions occurring in (t_1, t_2), let $|D(X)^{(t_1,t_2)}|$ be the number of transactions occurring in (t_1, t_2) while containing itemset X, and let $|(t_1, t_2)|$ be the width of time-window. The support of itemsets X in time-window (t_1, t_2) can be defined as

$$\text{support}\left(X^{(t_1,t_2)}\right) = \frac{|D(X)^{(t_1,t_2)}|}{|D^{(t_1,t_2)}|} \times 100\% \ .$$

Therefore, an itemset X is said to be frequent if and only if its support is greater than or equal to the predefined *min_sp* and the width of its time-window is greater than or equal to the predefined *min_win*, *i.e.*, $|(t_1, t_2)| > min\_win$. It is also noticeable that an itemset X may be frequent in many pairs of disjointed time-windows. For this case, all of them will be noted as independent FITW, such as $X^{(t_1,t_2)}$, $X^{(t_3,t_4)}$, ..., and so on.

The traditional way to discover frequent itemsets is to use the knowledge that all subsets of a frequent itemsets are also frequent. Analogously, if an itemset X in time-window (t_1, t_2) has a support $s\%$, then all of its subsets will have supports equal to or greater than $s\%$ in the same time-window (t_1, t_2), which is straightforward. Therefore, we develop the similar knowledge that all subsets of an FITW are also FITW. This insight will help us in developing an Apriori-based algorithm to quickly discover all FITW from a large database.

2.5 Summary

Mining association rules from huge amounts of data is useful in business. Market basket analysis studies the buying habits of customers by searching for sets of items that are frequently purchased together. Association rule mining consists of first finding frequent itemsets. Agrawal *et al.* (1993) have proposed a support-confidence framework for discovering association rules. This framework is now widely accepted as a measure of mining association rules. Han and Micheline (2001), Zhang and Zhang (2002), Zhang *et al.* (2004), and Fukuda *et al.* (2001) summarize the topics on various technologies of mining association rules. Because mining association rules in inventory databases focus on the discovery of association relationships among large items data which have some special characteristics of inventory management, we have summarized and discussed briefly the topics related to mining

association rules in inventory databases in this chapter. The key points of this chapter are:

1. basic concepts for dealing with association rules;
2. support-confidence framework concerning association rule mining; and
3. discussion of two topics related to inventory data mining: multidimensional association rules and association rules with time-windows.

Acknowledgements The work of mining association rules with a time-window is research conducted by Dr. Yiyong Xiao, Dr. Renqian Zhang and Professor Ikou Kaku. Their contribution is appreciated.

References

Agrawal R, Imilienski T, Swami A (1993) Mining association rules between sets of items in large datasets. In: Proceedings of SIGMOD, pp 207–216

Fukuda T, Morimoto Y, Morishita S, Tokuyama T (1996a) Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization. In: Proceedings of the ACM SIGMOD Conference on Management of Data, pp 13–23

Fukuda T, Morimoto Y, Morishita S, Tokuyama T (1996b) Mining optimized association rules for numeric attributes. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp 182–191

Fukuda T, Morimoto Y, Morishita S, Tokuyama T (1999) Mining optimized association rules for numeric attributes. J Comput Syst Sci 58(1):1–15

Fukuda T, Morimoto Y, Tokuyama T (2001) Data mining in data science series. Kyouritu, Tokyo

Han J, Micheline K (2001) Data mining: concepts and techniques. Morgan Kaufmann, San Fransisco, CA, pp 226–269

Huang KY, Chang CH (2005) SMCA: a general model for mining asynchronous periodic patterns in temporal databases. IEEE Trans Know Data Eng 17(6):774–785

Lee SJ, Jiang YJ (2008) Mining fuzzy periodic association rules. Data Know Eng 65:442–462

Li Y, Ning P, Wang XS, Jajodia S (2003) Discovering calendar-based temporal association rules. Data Know Eng 44(2):193–218

Ramaswamy BS, Siberschatz A (1998) Cyclic association rules. In: Proceedings of the 14th International Conference on Data Engineering, pp 412–421

Xiao YY, Zhang RQ, Kaku I (2009) A framework of mining association rules with time-window on real-time transaction database. In: Proceedings of the 2nd International Conference on Supply Chain Management, pp 412–421

Zhang C, Zhang S (2002) Association rule mining: models and algorithms, chaps 1–2. Springer, Berlin Heidelberg New York, pp 1–46

Zhang C, Zhang S, Wu X (2004) Knowledge discovery in multiple databases, chap 2. Springer, Berlin Heidelberg New York, pp 27–62

http://www.springer.com/978-1-84996-337-4

Data Mining
Concepts, Methods and Applications in Management and Engineering Design
Yin, Y.; Kaku, I.; Tang, J.; Zhu, J.
2011, XIV, 312 p., Hardcover
ISBN: 978-1-84996-337-4