CHAPTER 2

Association Rules

2.1 INTRODUCTION

Many large retail organizations are interested in instituting information-driven marketing processes, managed by database technology, that enable them to develop and implement customized marketing programs and strategies. These organizations collect and store massive amounts of sales data, also referred to as *basket data*. Each record in the basket database consists of a transaction date and a list of items in the transaction. This data can be used for decision making through the process of synthesis and analysis of the records in the basket database. In this section, we will consider the problem of discovering association rules in a large basket database.

2.2 MINING OF ASSOCIATION RULES IN MARKET BASKET DATA

An association rule in a basket database takes the form $X \Rightarrow Y$, where X and Y each are a set of some items appearing in transactions. An example of such a rule might be that a customer purchasing bread and milk will also get cheese with 90% likelihood. Mining for association rules in the basket database is used to find all such rules. Generally, these rules are valuable for cross-marketing, attached mailing applications, add-on sales, catalog design, store layout, and customer segmentation based on purchasing patterns.

The formal definition of association-rule mining can be stated as follows:

Let $Z = \{i_1, i_2, \ldots, i_n\}$ be a set of items, and let D be a set of transactions, where $T \in D$ and $T \subseteq Z$. A unique identifier called a TID is assigned to each

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

transaction. An association is an implication of the form $X \Rightarrow Y$, where $X, Y \subseteq Z$, and $X \cap Y = \varnothing$. Two factors affect the significance of association rules: support and confidence. We say that the rule $X \Rightarrow Y$ has support s in the transaction set D if s% of the transactions in D contain $X \cup Y$. On the other hand, we say that the rule $X \Rightarrow Y$ holds in the transaction set D with confidence c if c% of the transactions in D that contain X also contain Y.

Given a set of transactions D, the problem of mining association rules is to generate all association rules that have support and confidence greater than a minimum support (called *minsup*) and a minimum confidence (called *minconf*), respectively. Both *minsup* and *minconf* are normally specified by users. The set of transactions D could be represented as a flat data file, as a relational table, or as the result of a relational expression. The association rules that we consider are probabilistic in nature and are distinct from functional dependencies. A functional dependency $X \Rightarrow A$ implies an additional functional dependency $X + Y \Rightarrow A$ through redundancy. However, the presence of an association rule $X \Rightarrow A$ does not necessarily imply that the rule $X + Y \Rightarrow A$ also holds in the database because the latter may not satisfy the *minsup*. Similarly, the presence of two association rules $X \Rightarrow Y$ and $Y \Rightarrow A$ does not necessarily mean that the association $X \Rightarrow A$ holds because the latter may not satisfy the *minconf*.

■ 2.2.1 Apriori Algorithm

The problem of discovering all association rules can be broken down into two parts as follows:

1. Find all sets of items that have support values greater than the minimum support. These itemsets are called large itemsets. All others are called small itemsets.
2. Use the large itemsets to generate the desired rules. A simple algorithm for this task is as follows. Find all non-empty subsets of every large itemset L. For every such subset A, generate a rule of the form $A \Rightarrow (L - A)$ if the ratio of support(L) to support(A) is at least *minconf*. All subsets of L must be considered to generate rules with multiple consequents.

Algorithms for discovering large itemsets make multiple passes over the data. In the first pass, large itemsets of size one are generated. These itemsets are individual items with support at least *minsup*. In each subsequent pass, we use

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

the large itemsets found in the previous pass. This seed set of large itemsets is used to generate new, potentially large itemsets called candidate itemsets. The support for these candidate itemsets is counted during the passes over the data. At the end of each pass, we determine which candidate itemsets are actually large. These large itemsets are used in the next pass. This process continues until no new large itemsets are found.

The Apriori algorithm generates candidate itemsets using the large itemsets found in the previous pass without considering the transactions in the database. The rationale behind this principle is that any subset of a large itemset must be large. Therefore, the candidate itemsets of size K can be generated by joining two large itemsets of size $(K - 1)$ and deleting those that are not large. A description of the Apriori algorithm is given in Figure 2.1. In this algorithm, we assume that the items in each transaction and the items within each itemset are kept in lexicographic order. A count field is associated with each itemset to store the support for the itemset. The count field is initialized to zero when the itemset is first created.

In Figure 2.1, two set notations, L_k and C_k, are used. L_k is the set of large k-itemsets (i.e., those itemsets of size k with *minsup*). C_k is the set of candidate k-itemsets (i.e., potentially large itemsets of size k). Each member of both sets has two fields associated with it: itemset and support count. The first pass of the algorithm counts the occurrences of each individual item to determine the large 1-itemsets. Each subsequent pass k consists of two parts.

FIGURE 2.1 Apriori algorithm

```
L_1 = {large 1-itemsets};
for (k=2; L_{k-1} ≠ Ø; k++) do
    begin
     C_k = apriori-gen(L_{k-1}); // new candidates
     forall transactions t∈D do
        begin
         C_t=subset(C_k, t);
         forall candidates c∈C_t do
            c.count++;
        end
     L_k={c∈C_k| c.count ≥ minsup}
    end
answer = ∪_k L_k;
```

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

In the first part, L_{k-1}, the large itemsets generated in the previous pass are used to generate C_k, the large candidate k-itemsets. The apriori-gen( ) function is used for this purpose. In the second part, each itemset in C_k is counted in all transactions in the database. Only those itemsets with support that is at least *minsup* are collected to generate L_k.

■ 2.2.2 Apriori-gen( ) Function

The apriori-gen( ) function takes an argument L_{k-1} to generate C_k, the set of potentially large k-itemsets. In other words, C_k is a superset of the set of all large k-itemsets. The function consists of two steps. First, join operations are performed to join L_{k-1} with L_{k-1}. In the next step, we perform pruning to delete all itemsets $c \in C_k$ such that some $(k - 1)$-subset of c is not in L_{k-1}. The algorithm is as follows:

```
(join step)
insert into $C_k$
select p.$item_1$, p.$item_2$, ..., p.$item_{k-1}$, q.$item_{k-1}$
from $L_{k-1}$ p, $L_{k-1}$ q
where p.$item_1$ = q.$item_1$, ..., p.$item_{k-2}$ = q.$item_{k-2}$, p.$item_{k-1}$
< q.$item_{k-1}$

(prune step)
forall itemsets c∈$C_k$ do
    forall (k-1)-subsets s of c do
          if ( s ∉ $L_{k-1}$) then delete c from $C_k$;
```

For example, let L_3 be {{1 2 3} {1 2 4} {1 3 4} {1 3 5} {2 3 4}}. After the join step, C_4 will be {{1 2 3 4} {1 3 4 5}}. After the prune step, the itemset {1 3 4 5} will be deleted from C_4 because {3 4 5}, {1 4 5}, and {1 3 4} are not in L_3. Therefore, the final value of C_4 will be {1 2 3 4}.

■ 2.2.3 Apriori Example

Consider the database in Figure 2.2 and assume that the *minsup* is 3. Calling apriori-gen( ) with L_1 produces C_2, which contains six candidate itemsets. Four out of these six itemsets are chosen as large itemsets to be included in L_2. The apriori-gen( ) is called again with L_2 to generate C_3, which contains only one candidate itemset {1, 3, 4}. The candidate {1, 3, 4} in C_3 turns out

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.2 Apriori example

TID	Items
1	1, 3, 4
2	1, 2, 3, 4
3	2, 4
4	1, 2, 4
5	1, 3, 4, 5

Database

Itemset	Support
{1}	4
{2}	3
{3}	3
{4}	5

L_1

Itemset	Support
{1, 2}	2
{1, 3}	3
{1, 4}	4
{2, 3}	1
{2, 4}	3
{3, 4}	3

C_2

Itemset	Support
{1, 3}	3
{1, 4}	4
{2, 4}	3
{3, 4}	3

L_2

Itemset	Support
{1, 3, 4}	3

C_3

Itemset	Support
{1, 3, 4}	3

L_3

to be large and is the only member of L_3. When we generate C_4 with L_3, it turns out to be empty, and we terminate.

2.2.4 AprioriTid Algorithm

The AprioriTid algorithm is very similar to the Apriori algorithm in that both algorithms use the apriori-gen( ) function. In the Apriori algorithm, the database D is used to count support on every pass when C_k is generated. In AprioriTid, however, the database D is not used after the first pass. Rather, the set C'_k is used for counting support. Each member of the set C'_k is of the form <TID, $\{X_k\}$>, where each X_k is a potentially large k-itemset present in the transaction associated with TID. For $k = 1$, C'_1 is the database D with each item i replaced by the itemset $\{i\}$. For $k > 1$, C'_k is generated by the algorithm shown in Figure 2.3. The member of C'_k corresponding to a transaction t is $< t.\text{TID}, \{c \in C_k \mid c \text{ is contained in } t\} >$. If transaction t does not contain any candidate k-itemset, C'_k will not have any entry for this transaction. Thus, the number of entries in C'_k may be smaller than the number of transactions in the database. Especially for large values of k, each entry in C'_k may be smaller than the corresponding transaction because very few candidates may be contained in the transaction. For small values of k, however, each entry in C'_k may be larger than the corresponding transaction because the entry includes all candidate k-itemsets contained in the transaction.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.3 AprioriTid algorithm

```
$L_1$ = {large 1-itemsets};
$C'_1$ = database D
for (k=2; $L_{k-1} \neq \emptyset$; k++) do
    begin
     $C_k$=apriori-gen($L_{k-1}$); // new candidates
     $C'_k = \emptyset$
      forall entries t ∈ $C'_{k-1}$ do
        // determine candidate itemsets in $C_k$ contained
        // in the transaction with t.TID
          begin
           $C_t$={c∈$C_k$ | (c−c[k]) ∈ t.set-of-itemsets ∧
                      (c−c[k-1]) ∈ t.set-of-itemsets};
           forall candidates c ∈ $C_t$ do
             c.count++;
           if ($C_t \neq \emptyset$) then $C'_k$ += <t.TID, $C_t$>;
          end
        $L_k$={c∈$C_k$ | c.count ≥ minsup}
    end
answer = $\cup_k L_k$;
```

Applying AprioriTid to the example in Figure 2.2 will construct C'_1, C'_2, and C'_3 as shown in Figure 2.4. The actual sequence of the generated tables resulting from the application of AprioriTid is $C'_1 \Rightarrow L_1 \Rightarrow C_2$ (itemsets) $\Rightarrow C'_2 \Rightarrow C_2$ (support) $\Rightarrow L_2 \Rightarrow C_3$ (itemsets) $\Rightarrow C'_3 \Rightarrow C_3$ (support) $\Rightarrow L_3$. Notice that there is no entry in C'_3 for the transactions

FIGURE 2.4 C'_k (k = 1, 2, 3) from AprioriTid algorithm

TID	Set-of-Itemsets
1	{{1} {3} {4}}
2	{{1} {2} {3} {4}}
3	{{2} {4}}
4	{{1} {2} {4}}
5	{{1} {3} {4} {5}}

C'_1

TID	Set-of-Itemsets
1	{{1 3}{1 4} {3 4}}
2	{{1 2}{1 3} {1 4} {2 3}{2 4} {3 4}}
3	{{2 4}}
4	{{1 2}{1 4}}
5	{{1 3}{1 4} {3 4}}

C'_2

TID	Set-of-Itemsets
1	{{1 3 4}}
2	{{1 3 4}}
5	{{1 3 4}}

C'_3

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

with TIDs 3 and 4 because they do not contain any itemset in C_3. The candidate {1, 3, 4} in C_3 turns out to be large and is the only member in L_3. We terminate the algorithm since C_4 is empty after applying Apriori-gen( ) with L_3.

2.3 ATTRIBUTE-ORIENTED RULE GENERALIZATION

The Apriori algorithm discussed in Section 2.2 deals mainly with market basket data in which each record consists of a set of transaction items. The size of each record varies because each record contains a variable-length list of items purchased in each transaction. This method may not be appropriate for knowledge discovery in relational databases where each record is represented in terms of a fixed number of attributes. Furthermore, the records in a relational database are not a variable-length itemset, but instead are instances of a relation consisting of the same number of attributes. To extract generalized data from actual data in relational databases, a machine-learning technique integrated with database operations should be adopted.

In this section, we present an attribute-oriented induction method for knowledge discovery in relational databases. This approach is used to generate different types of knowledge, including characteristic rules, discrimination rules, and data evolution regularities. A characteristic rule is an assertion that characterizes a concept that is satisfied by all or the majority of cases in a target class under consideration. For example, the symptoms of a disease can be described by a characteristic rule. On the other hand, a discrimination rule is an assertion that discriminates a concept in the target class from other concepts in the contrasting classes. For example, to distinguish cardiovascular diseases from other kinds of disorders, a discrimination rule would be used to summarize their distinctive features. The data evolution regularity of the characteristic rules summarizes the characteristics of the changed data, whereas the discrimination rules summarize the features that discriminate the instances of target data from the contrasting ones. If a generated rule is associated with a quantitative measurement, it is called a quantitative rule.

The key to this approach is *tree ascension for attribute generalization*, which applies set-oriented database operations. This approach can be extended to knowledge discovery in other kinds of databases, such as nested relational databases, deductive databases, and databases containing noisy data and irregularities.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Furthermore, the generated rules can be used for intelligent query processing, cooperative query answering, and semantic query optimization.

2.3.1 Concept Hierarchies

Since a database may consist of multiple relations, each with a varying number of attributes, containing a huge amount of data, not all the data in the database may be relevant to a specific learning task. Rule induction in databases relies mainly on generalization. To derive a characteristic rule by generalization, the task-relevant data are collected into a class called the *target* class, whereas to derive a discrimination rule, the data are collected into two classes, the *target* class and the *contrasting* class.

Concept hierarchies are used to represent appropriate background knowledge and to control the generalization process. A concept taxonomy represents different levels of concepts according to a general-to-specific ordering. The values of each attribute in the database correspond to the most specific concepts, and the null description, denoted as "ANY," corresponds to the most general concepts in the hierarchy. The generated rules are generalized according to the concept hierarchies and should be given in a simple and explicit form, which is desirable to most users. Figure 2.5 shows a concept hierarchy table for a typical university database.

In Figure 2.5, $A \subset B$ indicates that B is a generalization of A. From this concept hierarchy table, we can generate a concept tree for status as shown in Figure 2.6. The concept hierarchies can be specified either explicitly by knowledge engineers and/or domain experts or implicitly by database schema. For example, a hierarchical relationship such as "city $\subset$ province $\subset$ country" can be attached to a relation containing "city," "province," and "country" attributes. From the relation, the information that "Dallas is a city of Texas which in turn is a state of the U.S.A." can be obtained.

Some concept hierarchies can be discovered semi-automatically or automatically through the careful analysis of the attribute domains. For numerical domains, a discrete concept hierarchy can be automatically constructed based on database statistics. For example, one may discover that a grade point average falls between 0 and 4, and therefore classify the GPA domain into the following four categories: {0.0–1.99}, {2.0–2.99}, {3.0–3.49}, and {3.5–4.0}. For domains with discrete values, a statistical method can also be used. For example, the country-of-origin for most students is scattered among

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.5 A concept hierarchy table for a university database

```
{biology, chemistry, computing, ..., physics} ⊂ science
{civil, electrical, ..., mechanical} ⊂ engineering
{engineering, science} ⊂ ANY-major
{freshman, sophomore, junior, senior} ⊂ undergraduate
{master's, Ph.D.} ⊂ graduate
{undergraduate, graduate} ⊂ ANY-status
{Austin, Dallas, ..., Houston} ⊂ Texas
{Jacksonville, Miami, ..., Tampa} ⊂ Florida
{Texas, Florida} ⊂ U.S.A.
{Pusan, Seoul, ..., Taegu} ⊂ Korea
{Beijing, Nanjing, ..., Shanghai} ⊂ China
{China, Japan, Korea, ..., Switzerland} ⊂ foreign
{foreign, U.S.A.} ⊂ ANY-place
{0.0–1.99} ⊂ poor
{2.0–2.99} ⊂ average
{3.0–3.49} ⊂ good
{3.5–4.00} ⊂ excellent
{poor, average, good, excellent} ⊂ ANY-grade
```

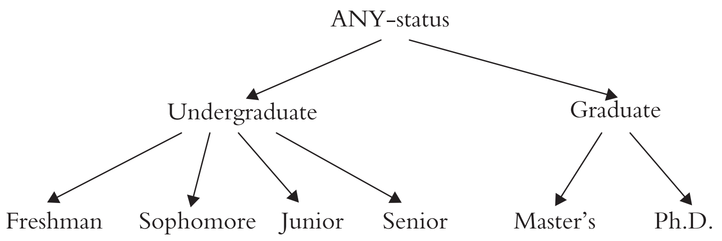

FIGURE 2.6 A concept tree for status

foreign countries and concentrated on the United States, so we may be able to categorize the concept of the attribute into "domestic" and "foreign." The concept hierarchy for an attribute can be modified based on database statistics, and it can be discovered and refined based on its relationship with other attributes. Furthermore, different hierarchies may be constructed from the same set of attributes, according to different viewpoints or preferences.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

From a logical viewpoint, each tuple in a relation is a logical expression in conjunctive terms. Similarly, a data relation may be characterized by a large set of disjunctions of such conjunctive terms. Users may specify the preferred generalization threshold, that is, the maximum number of disjuncts of the resulting expression. For example, if the threshold value is 5, the final generalized rule will consist of at most five disjuncts. The generalization threshold controls the complexity of the rule. A large threshold value may lead to a complex rule with many disjuncts, whereas a small threshold value leads to a simple rule with few disjuncts that may result in an overly general rule and loss of valuable information.

■ 2.3.2 Basic Strategies for Attribute-Oriented Induction

Seven basic strategies are suggested to perform attribute-oriented induction in relational databases. These strategies are applied to an initial relation that results from preprocessing the original databases. Pre-processing involves identifying the learning task, consulting the concept hierarchy to extract the set of primitive attribute sets in the relation, and forming the initial relation on which the induction can be performed. Consider the student relation in Figure 2.7 which contains information on 100 graduate students and 100 undergraduate students. The relation consists of five attributes: *id_no, status, major, origin*, and *GPA*.

FIGURE 2.7 The relation *student* in a university database

Id_no	Status	Major	Origin	GPA
1	Master's	Civil	Houston	3.5
2	Junior	Biology	Dallas	3.7
3	Junior	Electrical	Jacksonville	2.6
4	Master's	Physics	Miami	3.9
5	Ph.D.	Biology	Beijing	3.3
6	Sophomore	Chemistry	Austin	2.7
7	Senior	Computing	Tampa	3.5
8	Ph.D.	Biology	Shanghai	3.4
…	…	…	…	…
197	Sophomore	Electrical	Fort Worth	3.0
198	Ph.D.	Computing	Tampa	3.8
199	Master's	Biology	Nanjing	3.2
200	Freshman	Mechanical	Plano	3.9

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.8 The initial relation for induction

Id_no	Major	Origin	GPA	Vote
1	Civil	Houston	3.5	1
4	Physics	Miami	3.9	1
5	Biology	Beijing	3.3	1
8	Biology	Shanghai	3.4	1
…	…	…	…	…
198	Computing	Tampa	3.8	1
199	Biology	Nanjing	3.2	1

Assume that the learning task is to generate characteristic rules for graduate students with respect to *id_no, major, origin*, and *GPA* using the default hierarchy in Figure 2.6 and the default threshold value of 3. To begin the preprocessing we consult the concept hierarchy to retrieve the corresponding primitive attribute set from the *student* relation. The retrieved primitive attribute set is {Master's, Ph.D.}. Then the data about the graduate students are projected upon *id_no, major, origin*, and *GPA* to retrieve the initial data relation. The initial relation for the induction obtained as the result of preprocessing is shown in Figure 2.8. Notice that a special attribute called *vote* is attached to each tuple in the relation with the initial value set to 1. Once this initial relation is obtained, we then apply the following seven basic strategies in sequence.

Strategy 1: Generalization on the Smallest Decomposable Components

To avoid overgeneralization, the least commitment principle (commitment to the minimally generalized concept) is enforced to ensure that the smallest possible opportunity for generalization is considered. Hence, we perform the generalization on the smallest decomposable components (or on attributes) rather than on larger sets of decomposable components or attributes.

Strategy 2: Attribute Removal

If there is a large set of distinct values of an attribute and there is no higher-level concept associated with the attribute, the attribute cannot be generalized to a higher-level concept, and thus we eliminate the attribute from the generalization. For example, from the initial relation in Figure 2.8, we find after examining the task-relevant attributes in sequence that there is

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

no higher-level concept on the attribute *id_no*. Thus, we remove the *id_no* attribute from the generalization because a graduate student cannot be characterized by the attribute *id_no*.

Strategy 3: Concept Tree Ascension

The substitution of an attribute with a higher-level concept makes the tuple cover more cases than the original one, and thus generalizes the tuple. If there is a higher-level concept for an attribute in the concept tree, the substitution of the attribute with the higher-level concept should be performed by ascending the concept tree one level at a time to enforce the minimal generalization. This follows the least commitment principle and reduces the chances of overgeneralization. At this point, we add the special attribute *vote* to incorporate the quantitative information into the generalization process.

Strategy 4: Vote Propagation

The *vote* information in the generalized tuple indicates the total number of tuples generalized in the initial relation. Therefore, to keep the correct vote, when merging multiple tuples together, the vote counts should be accumulated in the process. After eliminating the attribute *id_no* and generalizing the relation in terms of the remaining three attributes, we get the generalized relation in Figure 2.9.

Strategy 5: Threshold Control on Each Attribute

The number of distinct tuples in the generalized relation must be less than or equal to the threshold value. If the number is larger than the threshold value, it means that one attribute contains more distinct values than the threshold, and thus further generalization must be performed.

FIGURE 2.9 A generalized relation

Major	Origin	GPA	Vote
Engineering	Texas	Excellent	35
Science	Florida	Excellent	10
Science	Texas	Excellent	30
Science	Korea	Good	10
Science	China	Good	15

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

Strategy 6: Threshold Control on Generalized Relations

If the total number of tuples in the generalized relation of the target is greater than the generalization threshold value, further generalization on the relation should be performed. The size of the generalized relation is further reduced by generalizing the relation on selected attributes and merging the identical tuples together. The generalization process continues until the total number of distinct tuples in the relation is less than or equal to the threshold value. As criteria for selecting attributes for further generalization, we may consider the following:

- Preference of a larger reduction ratio on the number of tuples or the number of distinct attribute values
- Simplicity of the final generalized rules
- Explicit selection and control of the attribute by users or experts

The third criterion above is based on the rationale that different rules can be discovered by following different paths, which may lead to multiple generalized relations for further examination. Experts and users may interactively filter out trivial and redundant rules and keep useful and interesting rules. Since the relation in Figure 2.9 contains five tuples, which is greater than the generalization threshold of 3, further generalization is performed. In this case the attribute *origin* is chosen since it has four distinct values. Ascending one level up in the concept hierarchy tree results in the relation shown in Figure 2.10. This final generalized relation meets the generalization threshold requirement since the number of tuples in the relation is 3. A characteristic rule can be retrieved by converting this relation to a simple logical formula, as follows, which justifies the need for strategy 7.

Strategy 7: Rule Transformation

Each tuple in the final generalized relation is transformed into conjunctive normal form, and these are combined into a formula in disjunctive

FIGURE 2.10 Further generalization of the relation

Major	Origin	GPA	Vote
Engineering	U.S.A.	Excellent	35
Science	U.S.A.	Excellent	40
Science	Foreign	Good	25

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.11 Simplification of the final generalized relation

Major	Origin	GPA	Vote
ANY	U.S.A.	Excellent	75
Science	Foreign	Good	25

normal form. The final relation in Figure 2.10 can be simplified as shown in Figure 2.11.

We can convert the final generalized relation to the following characteristic rule, which characterizes all of the data in the target class (i.e., the graduate student class).

Characteristic rule:
A graduate student is either a U.S. student with an excellent GPA with 75% probability, or a foreign student with a good GPA with 25% probability.

This rule above corresponds to the following quantitative rule:

$$\forall(x)\ \text{graduate}(x) \rightarrow (\text{place-of-origin}(x) \in \text{U.S.A.} \wedge \text{GPA}(x) \in \text{excellent})\ [75\%] \vee$$
$$(\text{place-of-origin}(x) \in \text{foreign} \wedge \text{GPA}(x) \in \text{good})\ [25\%]$$

2.3.3 Basic Attribute-Oriented Induction Algorithm

Based on the strategies discussed in the previous section, we now summarize the attribute-oriented induction algorithm below:

Algorithm: Basic attribute-oriented induction in relational databases
Input: A relational database, the learning task, the preferred concept hierarchy (optional), and the preferred form to express the learning result (such as a generalization threshold).
Output: A characteristic rule generated from the database.
Method: The basic attribute-oriented induction method consists of the following four steps:

Step 1: Collect the task-relevant data.
Step 2: Perform basic attribute-oriented induction as follows:

```
  begin {basic attribute-oriented induction}
  for each attribute A_i (1 ≤ i ≤n, n = the number of
attributes) in the generalization relation GR do {
```

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

```
while (# of distinct values in $A_i$) > threshold do {
   if there is no higher-level concept for $A_i$ in the
concept hierarchy table
   then remove $A_i$
   else
               (1) substitute the values of $A_i$ with its
                   corresponding minimally generalized
                   concept;
               (2) merge the identical tuples;
   }

while (# of tuples in GR) > threshold do {
         (1) selectively generalize attributes;
         (2) merge the identical tuples;
   }
}
end {the end of basic attribute-oriented induction}
```

Step 3: Simplify the generalized relation.
Step 4: Transform the final relation into a logical rule. □

The basic attribute-oriented induction algorithm extracts a characteristic rule of the form *learning_class* (*x*) → *condition* (*x*) from the initial relation. This rule covers all of the positive examples in the database and hence forms a necessary condition for the learning (generalized) concept. However, it may not cover data in other classes that also meet the specified condition.

■ 2.3.4 Generation of Discrimination Rules through Attribute-Oriented Induction

Discrimination rules distinguish concepts of one class (the target class) from those of another class (the contrasting class). Therefore, common concepts that exist in both classes should be detected and removed from the description of the discrimination rules. A discrimination rule can be obtained by simultaneously generalizing both the target class and the contrasting class and removing the conditions that exist in both classes from the final generalized rules. Consider a discrimination rule that distinguishes *graduate* classes from *undergraduate* classes in the *student* relation in Figure 2.7. Let's assume that the generalization process (through attribute removal and concept tree

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.12 A generalized relation

Class	Major	Origin	GPA	Vote	Mark
Undergraduate	Science	Florida	Excellent	15	
	Engineering	Florida	Average	20	
	Science	Texas	Average	60	
	Science	*Texas*	*Excellent*	*35*	*m1*
	Engineering	Texas	Average	50	
	Engineering	Alabama	Excellent	20	
Graduate	Engineering	Texas	Excellent	35	
	Science	Alabama	Excellent	10	
	Science	*Texas*	*Excellent*	*30*	*m1*
	Science	Korea	Good	10	
	Science	China	Good	15	

ascension on both classes) results in the generalized relation in Figure 2.12. Note that the tuples marked with *m1* in both classes are overlapping tuples. They indicate that *Texas*-born, *science* major students with *excellent* grades may be either *graduate* or *undergraduate* students. These overlapping rules must be eliminated in the generalization process to generate an effective discrimination rule. Thus, we need an additional strategy to handle the overlapping tuples when generating discrimination rules.

Strategy 8: Handling Overlapping Tuples

The tuples that are shared by both the target and the contrasting classes are called *overlapping* tuples. The overlapping tuples must be marked and eliminated from the final discrimination rules.

Assuming a threshold value of 3, the relation in Figure 2.12 is not the final discrimination rule because both classes have more than three generalized tuples. Hence, we continue with further generalization on the attribute *origin*. The resulting relation is shown in Figure 2.13. From the relation we notice that the overlapping marks are inherited because the generalized concepts overlap in both classes. The total number of unmarked tuples in both the target and contrasting classes is less than the specified threshold value of 3, so we stop the process.

The unmarked tuple in the target (graduate) class yields the following qualitative discrimination rule, which excludes all overlapping disjuncts. The rule states that if a student is born in a foreign country, has a good GPA, and

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.13 A final generalization relation

Class	Major	Origin	GPA	Vote	Mark
Undergraduate	Science	U.S.A.	Excellent	50	m1
	Engineering	U.S.A.	Average	70	
	Science	U.S.A.	Average	60	
	Engineering	U.S.A.	Excellent	20	m2
Graduate	Engineering	U.S.A.	Excellent	35	m2
	Science	U.S.A.	Excellent	40	m1
	Science	Foreign	Good	25	

is a science major, (s)he is a graduate student. A qualitative discrimination rule provides a sufficient condition, but not a necessary condition, for a tuple (an object) to be in the target class, since the rule may not cover all the positive examples of the target class in the database. In other words, the tuples that meet the condition are in the target class, but some tuples in the target class may not satisfy the condition. Thus, the rule is represented in the form *learning-class*(x) ← *condition*(x) as follows.

$$\forall(x)\ graduate(x) \leftarrow major(x) \in science \wedge origin\ (x) \in foreign \wedge GPA(x) \in good$$

In many cases, it is necessary to associate each disjunct in the generalized relation with a quantitative measurement (called *d-weight*) to derive a quantitative rule from the final generalized relation. The *d-weight* for a concept q is defined as the ratio of the number of original tuples in the target class covered by q to the total number of tuples in both the target and the contrasting classes covered by q. The formal definition of the *d-weight* of the concept q in a class C_i is given as follows. Note that the *d-weight* values are in the range [0–1].

$$d\text{-}weight = votes\ (q \in C_i)/\textstyle\sum_{i=[1..k]} votes(q \in C_i)$$

Using this *d-weight* formula, we can derive the following quantitative discrimination rule for graduate students in the database. The *d-weight* for the first tuple is 35/(35 + 20) = 63.64% and is 40/(40 + 50) = 44.44% for the second tuple. Note that any unmarked tuples have 100% *d-weight* value.

$$\forall(x)\ graduate(x) \leftarrow$$

$$(major(x) \in science \wedge origin\ (x) \in foreign \wedge GPA(x) \in good)\ [100\%] \vee$$

$$(major(x) \in engineering \wedge origin\ (x) \in U.S.A. \wedge GPA(x) \in excellent)\ [63.64\%] \vee$$

$$(major(x) \in science \wedge origin\ (x) \in U.S.A. \wedge GPA(x) \in excellent)\ [44.44\%]$$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

The quantitative rule above presents the quantitative measurements of the graduate students' properties in the target class (graduate class) with that of the contrasting class (undergraduate class). The 100% *d-weight* value illustrates that the generalized tuple is in the target class only. All other *d-weight* values show the possibility of the generalized tuple in the target class.

2.4 ASSOCIATION RULES IN HYPERTEXT DATABASES

A hypertext system is a database in which each page is connected to other pages through a set of page links that allows nonsequential access to relevant information. Navigation in a hypertext system is highly dependent upon each individual user's familiarity with the system, preferences, domain-knowledge level, and the specific piece of information that is being searched. Internet businesses such as online bookstores and online general merchandise stores rely heavily on hypertext databases. The success of such websites depends on the quality of the database services since on the Web it only takes a mouse click for a customer to move to a competitor's site. Therefore, it is crucial to analyze users' behavioral patterns when browsing web pages and to understand their preferences. This data can help design the best hypertext system to achieve the highest profit and cost savings.

Mining for access patterns in a Web-like environment may be viewed as a generalization of association rules in the context of flat transactions. Server log files contain information that characterizes user access to the server, including user identification, the sequence of the requested pages, and the date and time of access. In this section we formalize the concept of composite association rules in a hypertext system in the context of a directed graph by generalizing the concept of association rules with confidence and support. From the server log data, user navigation sessions can be reconstructed to build a weighted directed graph that summarizes it. The graph represents the user view of the hypertext system and delineates the domain for mining user patterns. Each node in the graph denotes a page visited by the user, and an arc in the graph denotes a link traversed by the user. Initially, weight on each arc is set to zero, and its value is incremented by one every time the user traverses the arc. Therefore, the weight on each arc represents the frequency of the user's traversing along that particular link.

This model does not represent a user's individual site visits, but rather, it summarizes a collection of sessions. Therefore, the generated weighted

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

graph is intended to represent the navigation patterns of a single user or a group of users with similar interests. In the following section, we present a formal model of a simple hypertext system in terms of a weighted directed graph.

■ 2.4.1 Formal Model

The formal model for a hypertext system is given as a weighted directed graph $G = (N, E)$, where N is a set of nodes and E is a set of arcs, each connecting a pair of nodes. A node represents a page, while an arc (A, B) represents a link between a pair of pages. In the arc, A is called the source node, and B is the target node. For simplicity, we can assume that there is at most one link between any two nodes. The graph is weighted, and the weight on each arc represents the number of times the user traverses the link. To discuss the problem of mining association rules in a hypertext system, we start with the following assumptions:

1. The hypertext system contains only elementary pages that represent a single concept or idea.
2. Every node connected by a link refers to a page as a whole, not to a subset of the page or its contents.
3. The hypertext system does not contain any loop in which the source node coincides with the target node.
4. There is at most one link between any two nodes.

Initial interaction of the user with the hypertext system starts with the selection of a link on a page among all those available. User preferences among the available links in the pages of the system can be represented as a trail, which is an alternating sequence of nodes and arcs. A trail is $(v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ such that $v_i \in N$, $1 \leq i \leq n$, and every $e_i = (v_i, v_{i+1}) \in E$, $1 \leq i \leq n - 1$ is distinct. For simplicity, a trail $(v_1, e_1, v_2, e_2, \ldots, e_{n-1}, v_n)$ will be referred to as $(v_1, v_2, \ldots, v_n)$ since the graph does not contain two links with the same source and target node by assumption.

We now define a composite association rule in the hypertext system as follows. A composite association rule is a statement such as "When a user browses the hypertext system, he is likely to traverse a trail $A_1, A_2, \ldots, A_n$ with a certain probability." It can be expressed as $[(A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_3) \wedge \ldots \wedge (A_{n-1} \rightarrow A_n)]$, where $A_i \in N$ $(1 \leq i \leq n)$ and $\{(A_i, A_{i+1}) \mid i = 1, 2, \ldots, n - 1\} \in E$. The

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

above expression means that when a user traverses a hypertext system, he will follow the trail $(A_1, A_2, \ldots, A_n)$ with a certain confidence. The validity of the rules is supported by two factors called *confidence* and *support* as defined below.

Confidence

Let a rule $r = [(A_1 \to A_2) \wedge (A_2 \to A_3) \wedge \cdots \wedge (A_{n-1} \to A_n)]$. Then the *confidence* of the rule r, written as C_r, is defined as the product of the confi–dences of all the corresponding single rules, that is $C_r = \prod_{i=1}^{n-1} C(A_i \to A_{i+1})$, where $C(A_i \to A_{i+1})$ is

$$\frac{|(A_i, A_{i+1})|}{\sum_{\{x|(A_i,\, x)\in E\}} |(A_i, x)|}.$$

In the last equation, $|(A_i, A_{i+1})|$ the number of times the link $(\mathrm{A}_i, \mathrm{A}_{i+1})$ is traver–sed and $\sum_{\{x|(A_i,x)\in E\}} |(A_i, x)|$ is the number of times the user traversed any link having A_i as the source node.

Support

The composite association rule $r = [(A_1 \to A_2) \wedge (A_2 \to A_3) \wedge \cdots \wedge (A_{n-1} \to A_n)]$ holds with *support* S_r, where S_r represents the average number of times the links of the rule were traversed over the average number of times all of the links in the graph were traversed., $S_r = \mathrm{X}/\mathrm{Y}$where

$$X = \frac{\sum_{i=1}^{n-1} |(A_i, A_{i+1})|}{n-1}, \qquad Y = \frac{\sum_{\{i|(x_i,x_i+1)\in E\}} |(x_i, x_{i+1})|}{|E|},$$

and $|E|$ is the total number of links in E. In the equation, $|(A_i, A_{i+1})|$ is the number of times the link (A_i, A_{i+1}) is traversed. The values of this *support* tend to distribute around the value of 1, that is, the rules with a *support* value greater than 1 consist of the links that were traversed more than average. Furthermore, we define the *monotonic support*, S_m, of a composite association rule as the minimum *support* value among all the single rules that constitute the composite rule.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

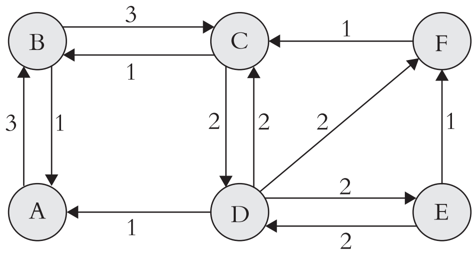

FIGURE 2.14 A weighted directed graph

Now, let's consider the weighted directed graph in Figure 2.14. Let the rule *r* be $(B \to C \to D)$. Then

$$C_{(B\to C\to D)} = \frac{3}{3+1} \times \frac{2}{2+1} = 0.5 \quad \text{and} \quad S_{(B\to C\to D)} = \frac{\frac{3+2}{2}}{\frac{21}{12}} = 1.43.$$

With an appropriate *support* value, we can capture rules that are globally frequent, but the rules may contain some links that are below the support threshold. From the trail $B \xrightarrow{3} C \xrightarrow{2} D \xrightarrow{2} E \xrightarrow{1} F$ in the example in Figure 2.14, we get

$$S_r = \frac{8/4}{1.75} = 1.14$$

Note in this case that the minimum support value among all the single rules constituting the composite rule is 0.57 since $S_{B\to C} = 3/1.75 = 1.71$, $S_{C\to D} = 2/1.75 = 1.14$, $S_{D\to E} = 2/1.75 = 1.14$, and $S_{E\to F} = 1/1.75 = 0.57$. If we assume a support threshold value of 1, this trail is a rule only with the nonmonotonic definition.

2.4.2 Algorithms for Generating Composite Association Rules

In this section, two algorithms for mining composite association rules in hypertext databases are presented. In these algorithms, we assume the non-monotonic definition of support in which the support value cannot be used as a pruning criterion. The algorithms will generate the set of Candidate Rules (CR), that is, the set of all trails with *confidence* above the threshold. From this CR set, only the trails with *support* above the threshold value are generated and included in the composite rules set. We now define some useful concepts needed to describe the algorithms that follow.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

> *A* neighbor *link of a given trail $t = (v_1, v_2, \ldots, v_n)$ is a link that can be appended to the trail, t, yielding a trail that satisfies the properties of trails given in Section 2.4.1. A* backward neighbor *is a link (x, v_1), where $x \in N$, $x \neq v_n$, $(x, v_1) \in E$, and $(x, v_1) \notin t$. A forward neighbor is a link (v_n, x), where $x \in N$, $(v_n, x) \in E$, and $(v_n, x) \notin t$.*

The first algorithm presented is a modified Depth-First Search (DFS) algorithm, a special case of a directed-graph DFS. For each link in the graph, a DFS tree is constructed. Each branch in the tree corresponds to a candidate rule, and therefore each DFS tree generates all possible candidate rules that have the root link as the first link. Furthermore, each branch exploration will identify a new trail, which is its own independent list of visited links. The branch exploration will stop when it finds a node whose links are already marked as visited or when the *confidence* value of the corresponding trail drops below the threshold value *C*. In the algorithm, given a trail *t*, we use *forw_neigh*(*t*) to refer to its next nonvisited forward neighbor and *last_link*(*t*) to refer to the last link in *t*. Given a trail *t* and a link *a*, *t* + *a* denotes the concatenation of *t* and *a*.

```
Modified DFS(G, C) Algorithm

begin
    for each {e∈E, $C_e \geq C$}
          EXPLORE(e, $C_e$)
    end for
end

EXPLORE(trail, $C_{trail}$) Algorithm
begin
    CR: = CR ∪ trail; //CR is the candidate rule set and
initially empty
    for each {f: = forw_neigh(trail)}
          if ($C_f \times C_{trail} \geq C$) then
          EXPLORE(trail + f, $C_f \times C_{trail}$)
    end for
end
```

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

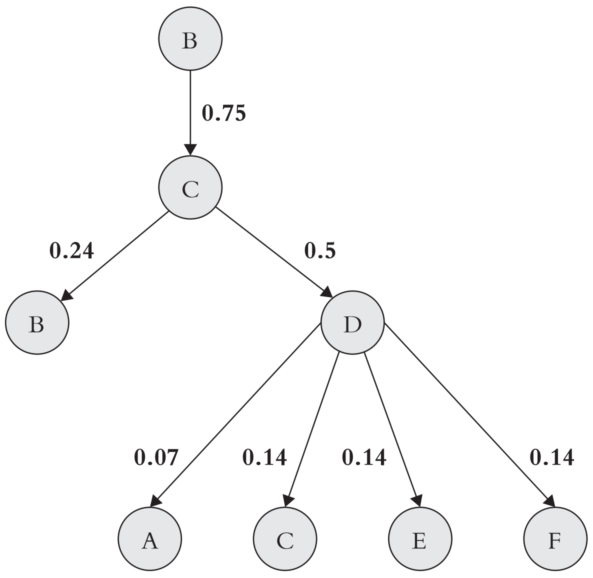

FIGURE 2.15
A DFS tree built from the B → C link

The modified DFS (MDFS) algorithm above considers every link in the graph G and expands it in the same manner that the DFS method does. Let's consider the link $B \to C$ among the 12 distinct links. Assuming the support threshold $S = 1.0$ and the confidence threshold $C = 0.3$, the MDFS algorithm generates a DFS tree as shown in Figure 2.15.

Each branch of the tree corresponds to a candidate rule. From the tree we observe the following:

$$\begin{aligned}
&C_{B\to C} = 0.75\\
&C_{B\to C\to B} = 0.24 \text{ (stop)}\\
&C_{B\to C\to D} = 0.5\\
&C_{B\to C\to D\to A} = 0.07 \text{ (stop)}\\
&C_{B\to C\to D\to C} = 0.14 \text{ (stop)}\\
&C_{B\to C\to D\to \mathrm{E}} = 0.14 \text{ (stop)}\\
&C_{B\to C\to D\to \mathrm{F}} = 0.14 \text{ (stop)}
\end{aligned}$$

Among the seven trails, only two trails meet the confidence threshold of 0.3. All other trails are dropped from further exploration. Furthermore, both rules meet the support threshold of 1 since $S_{B\to C} = \frac{3}{1.75} = 1.71$ and $S_{B\to C\to D} = \frac{2.5}{1.75} = 1.43$. Therefore, EXPLORE($B \to C$, 0.3) generates two

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

trails $\{B \to C,\ B \to C \to D\}$. The process can be repeated with all other links of the graph in Figure 2.14 to generate all trails meeting the support and confidence thresholds.

The other algorithm for mining composite association rules in hypertext databases is called an Incremental Step Algorithm (ISA). In this algorithm, we take advantage of the property that states that every subtrail of a trail with confidence above the threshold itself has confidence above the threshold. This could mean that a trail with n links can be obtained by combining the trails with $n-1$ links. The algorithm starts with the trails with one link, CR_1, that have confidence above the threshold C. It recursively builds the trails with n links, CR_n, from the set of the trails with $n-1$ links, CR_{n-1}. The algorithm is given as follows:

```
ISA(G, C) Algorithm

begin
    for each {e ∈ E, $C_e \geq C$}
          $CR_1$ := $CR_1 \cup$ e;
    end for
    i=1;
    repeat
     for each r ∈ $CR_i$;
          for each {x ∈ $CR_i$; x ≠ r ∧ (x[j] = r[j+1], 1 ≤ j
≤ i)}
          if $C_r \times C_{last\_link(x)} \geq C$ then
                $CR_{i+1}$ = $CR_{i+1} \cup$ (r + last_link(x) );
          end for
     end for
     i++;
    until ($CR_{i+1}$ = ∅)
end
```

From Figure 2.14, we find that $CR_1 = \{B \to C, A \to B, C \to D, C \to B, F \to C, E \to F, E \to D\}$, $CR_2 = \{B \to C \to D, A \to B \to C, F \to C \to D, F \to C \to B, E \to F \to C\}$, $CR_3 = \{A \to B \to C \to D\}$, and $CR_4 = \varnothing$. From these sets, we can check the support values for each rule to choose only the rules with support above the threshold. For example, since $S_{A \to B \to C \to D} = 1.52$, the rule $A \to B \to C \to D$ could be considered a meaningful rule.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.5 QUANTITATIVE ASSOCIATION RULES

Data mining in a market basket database is used to find all association rules that satisfy user-specified minimum support and minimum confidence constraints. This problem can be conceptually viewed as finding all associations among the "1" values in a relational table in which all the attribute values are Boolean. The value "1" for the value of an attribute for a given record means that the item corresponding to the attribute is present in the transaction corresponding to the record. The value "0" means exactly the opposite. This problem, often referred to as the "Boolean association rules" problem, is less general than the mining of association rules in relational tables that include richer data types for the attributes. These attributes can be either quantitative, such as age and income, or categorical, such as gender, marital status, and zip code. Boolean attributes can be considered a special case of categorical attributes.

In this chapter, the problem of mining association rules over quantitative and categorical attributes in a relational table is described. This rule-mining problem is referred to as the "quantitative association rules" problem. For illustration purposes, Figure 2.16 shows a "student" table with three nonkey attributes. "Age" and "Courses" are quantitative attributes, whereas "Gender" is a categorical attribute. A quantitative association rule present in the table is <Age: 20..29> ∧ <Gender: M> ⇒ <Courses: 1..2>.

2.5.1 Mapping of Quantitative Association Rules

There are many algorithms for finding Boolean association rules, and a couple of them were introduced in Section 2.2. These algorithms can be used to mine quantitative association rules if a relational table with quantitative or categorical attributes can be mapped to a table with Boolean

FIGURE 2.16 Example of quantitative association rules

Record#	Age	Gender	Courses
1	21	M	1
2	24	F	2
3	25	F	3
4	29	M	2
5	30	F	3
6	32	M	2
7	34	M	2
8	39	F	3

Sample Rules	Support	Confidence
<Age: 20..29> ∧ <Gender: M> ⇒ <Courses: 1..2>	25%	100%
<Courses: 3..4> ⇒ <Gender: F>	37.5%	100%

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.17 Mapping to Boolean association rules problem

Record#	Age: 20..29	Age: 30..39	Gender: F	Gender: M	Courses: 1	Courses: 2	Courses: 3
1	1	0	0	1	1	0	0
2	1	0	1	0	0	1	0
3	1	0	1	0	0	0	1
4	1	0	0	1	0	1	0
5	0	1	1	0	0	0	1
6	0	1	0	1	0	1	0
7	0	1	0	1	0	1	0
8	0	1	1	0	0	0	1

values. This mapping will be straightforward if all attributes in the table are categorical or quantitative with only a few possible values. Instead of having only one field for the attribute, as many fields as there are attribute values are created. The value of the field corresponding to <attribute_m, value_n> will be 1 if the attribute_m has value_n in the original record and 0 otherwise. If the size of the domain for the attribute values is large, one simple approach is to partition the values into intervals and map each <attribute, interval> pair to a Boolean attribute.

Figure 2.17 shows the mapping of the three nonkey attributes of the student table shown in Figure 2.16. The categorical attribute *gender* has two Boolean attributes <Gender: F> and <Gender: M>. The *courses* attribute is not partitioned into intervals since the number of values for the attribute is small. Instead, each value is mapped into a separate Boolean field. Finally, the third attribute *age* is partitioned into two intervals <Age: 20..29> and <Age: 30..39>.

Two problems are noted with this approach when applied to quantitative attributes. First, if the number of partitioned intervals for a quantitative attribute is large, the support for any single interval can be low. Hence, some rules involving this attribute may not be found due to not satisfying the minimum support. The other problem is that whenever the values are partitioned into intervals, some information may be lost. This information loss increases as the interval sizes become larger. For example, in Figure 2.17, the rule <Courses: 3> $\Rightarrow$ <Gender: F> has 100% confidence. But when the *courses* attributes are partitioned into two intervals such that 2 courses and 3 courses are placed into the same interval, then the closest rule is <Courses: 2..3> $\Rightarrow$ <Gender: F>, which has only 44.4% confidence.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

These two problems create the following conflicting situations: (1) large intervals may generate rules that do not satisfy minimum confidence, and (2) small intervals may generate rules that do not satisfy minimum support. These situations can be resolved by considering all possible continuous ranges over the values of the quantitative attribute or over the partitioned intervals since the latter problem (*minSup* problem) disappears by combining adjacent intervals or values. Although the *minConf* problem is still present, the information loss is reduced by increasing the number of intervals that avoid the *minSup* problem. This implies that the number of intervals must be increased while simultaneously combining adjacent intervals. This causes the following two problems:

1. Time complexity of $O(n^2)$: When there are n values or intervals, there are on average $n(n + 1)/2$ ranges. Hence, the number of items per record will sharply increase, which in turn increases execution time.
2. Many rules problem: The number of rules will increase: if a value or an interval has minimum support, any range containing this value or interval will also have the minimum support. Many of the generated rules will not be of interest.

A solution exists to realize faster execution time and fewer intervals to mitigate execution time. This is achieved by combining adjacent values or intervals. The process of combining intervals stops when their combined support exceeds a user-specified maximum support value, with the exception that any single interval or value is still considered, even though its support value exceeds the maximum support. On the other hand, information loss can be reduced by creating more intervals, thus mitigating the *minConf* problem. Hence, a special measure is needed to decide whether or not to partition a quantitative attribute and how many partitions there should be if it is partitioned. In the following sections, an approach to these problems is discussed.

■ 2.5.2 Problem Decomposition

The problem of mining quantitative association rules is to find all quantitative association rules from a given set of records that have support and confidence greater than the user-specified minimum support and confidence. To address the problem of quantitative association rule mining, some terminology is introduced first as follows:

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$A = \{a_1, a_2, \ldots, a_m\}$, is a set of attributes.
P is a set of positive integers.
A_V denotes the set $\{<x, v> \in A \times P\}$.
A_R denotes the set $\{<x, l, u> \in A \times P \times P \mid l \leq u$ if x is quantitative; $l = u$ if x is categorical$\}$.
For any $X \subseteq A_R$ attributes(X) is the set $\{x \mid <x, l, u> \in X\}$.

From the above definitions, it should be noted that either a pair $<x, v> \in A_V$ or a triple $<x, l, u> \in A_R$ represents an item. $\hat{X}$ is a generalization of X (i.e., X is a specialization of $\hat{X}$) if attributes(X) = attributes$(\hat{X})$ and $\forall x \in$ attributes$(X)\{<x, l_1, u_1> \in X \wedge <x, l_2, u_2> \in \hat{X} \Rightarrow l_2 \leq l_1 \leq u_1 \leq u_2\}$. For the uniform treatment of categorical and quantitative attributes, the categorical attribute values are mapped into a set of consecutive integers. If the quantitative attributes are not partitioned into intervals, the values are mapped to consecutive integers with the order of the values preserved. If they are partitioned into intervals, then the intervals are mapped to consecutive integers with the order of values preserved. Now the problem of mining quantitative association rules can be solved with the following five steps:

1. Determine the number of partitions for each quantitative attribute.
2. Perform mapping for categorical or quantitative attributes.
3. Find the support for each value of categorical and quantitative attributes.
4. Generate association rules by using the frequent itemsets.
5. Determine the interesting rules to keep in the output.

For example, consider the *student* table shown in Figure 2.18, which assumes 25% minimum support and 50% minimum confidence. The table contains one categorical attribute, *gender*, and two quantitative attributes, *age* and *courses*. The *age* attribute is partitioned into four intervals. After mapping to consecutive integers using tables (b) and (c), the table looks as shown in table (d). Table (e) shows sample frequent itemsets, and (f) shows a couple of sample rules.

■ 2.5.3 Partitioning of Quantitative Attributes

In this section, we address when quantitative attributes should be partitioned and how many partitions should be created. When rules are generated

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.18 Example of problem decomposition

(a) The *student* table

Record#	Age	Gender	Courses
1	21	M	1
2	24	F	2
3	25	F	3
4	29	M	2
5	30	F	3
6	32	M	2
7	34	M	2
8	39	F	3

(b) Partition & mapping for *age*

Interval	Mapping integer
20..24	1
25..29	2
30..34	3
35..39	4

(c) Mapping for *gender*

Value	Integer
F	1
M	2

(d) After mapping attributes

Record#	Age	Gender	Courses
1	1	2	1
2	1	1	2
3	2	1	3
4	2	2	2
5	3	1	3
6	3	2	2
7	3	2	2
8	4	1	3

(e) Sample frequent itemsets

Item sets	Support
{<Age: 20..29>}	4
{<Age: 30..39>}	4
{<Gender: F>}	4
{<Gender: M>}	4
{<Courses: 1..2>}	5
{<Courses: 3..4>}	3
{<Age: 30..39>, <Gender: F>}	2

(f) Some sample rules

Rules	Support	Confidence
<Age: 30..39> ∧ <Gender: F> ⇒ <Courses: 3..4>	25%	100%
<Gender: F> ⇒ <Courses: 3..4>	37.5%	75%

considering all ranges over the partitions of quantitative attributes instead of raw values, information loss will occur. Since partitioning will cause loss of information, a special measure called *partial completeness* is used to control the amount of information lost by partitioning. Partial completeness measures how far the generated rules over the partitions are from the rules generated

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.19 Example of a 1.5-complete set

Itemset Number	Itemsets	Support
1	<Age: 20..29>	10%
2*	<Age: 30..40>	12%
3*	<Age: 20..40>	18%
4	<Courses: 0..1>	10%
5*	<Courses: 0..3>	12%
6	<Age: 20..29> and <Courses: 0..1>	8%
7*	<Age: 20..40> and <Courses: 0..3>	12%

over the raw values of the quantitative attributes. The definition of partial completeness is given below assuming that $K \geq 1$, that C is the set of all frequent itemsets in D, and that $P \subseteq C$.

P is K-complete with respect to C if

1. $X \in$ P and $X' \subseteq X \Rightarrow X' \in$ P, and $\forall$ X $\in$ C, $\exists \hat{X} \in$ P such that
2. $\hat{X}$ is a generalization of X and support($\hat{X}$) $\leq K \times$ support(X), and
3. $\forall Y \subseteq X$, $\exists \hat{Y} \subseteq \hat{X}$ such that $\hat{Y}$ is a generalization of Y and support($\hat{Y}$) $\leq K \times$ support(Y)

For example, if the table shown in Figure 2.19 is assumed to contain the frequent itemsets C, then the itemsets 2, 3, 5, and 7 would form a 1.5-complete set, since for any subset X, either 2, 3, 5, or 7 is a generalization with support at most 1.5 times the support of X. However, the itemsets 3, 5, and 7 do not form a 1.5-complete set, since the itemset 3 is the only generalization of the itemset 1 among the set (3, 5, and 7) and 18% is more than 1.5 times 10%.

Given the partial completeness level desired by the user and the minimum support, the number of partitions required can be calculated by the following formula, assuming equi-depth partitioning.

Number of intervals = $(2 \times n)/[m \times (K - 1)]$, *where n is the number of quantitative attributes, m is the user-provided minimum support, and K is the partial completeness level. From the formula, it can be noted that the support of each base interval must be at most* $[m \times (K - 1)]/(2 \times n)$ *for the frequent itemset to be K-complete.*

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.6 MINING OF COMPACT RULES

As the size of the database increases, it gets more difficult to analyze larger amounts of data by finding all possible semantic relationships among them. Hence, the rule-mining focus can be on deriving rules with full confidence. As described in the previous sections, these rules are derived based on the dependencies between a target attribute and other condition attributes. A problem with this plain rule-mining approach arises when many rules are derived or many condition attributes are involved in the antecedent of each rule. Hence, for the efficient use of the rules, it is important to be able to derive rules that are compact.

In this section, a new concept, called a Semantic Association Relationship (SAR), is introduced, which will facilitate the extraction of compact association rules in a database. Through the use of the SAR, the number of attributes involved in the antecedent of each rule and the number of derived rules can be reduced, thus simplifying the complexity of the rules and the dependencies between the attributes.

2.6.1 Semantic Association Relationships

Diminishing the number of distinct values in each attribute makes data easier to characterize in terms of rules. Domain knowledge in the form of the Domain Concept Hierarchy (DCH) is used for this purpose. The DCH is a hierarchy structure that provides a general-to-specific structure of the domain knowledge relevant to an attribute. It is a multilevel abstraction of domain knowledge in which the most specific values (i.e., the terminal nodes in the hierarchy) are domain elements of an attribute, and the remaining values in the hierarchy are domain concepts. This hierarchy is generally specified by domain experts.

The DCH can be either an IS-A hierarchy or a taxonomy for a non-numerical attribute. "A department manager is an employee" is an example of an IS-A hierarchy through which an instance of an object can be specified, whereas "there are many cities in a state" is an example of a taxonomy. The elements of a numerical attribute can be divided into groups, each of which is represented by a concept.

SARs are associations between domain elements and concepts in attributes. They are another important factor that contributes to the reduction of distinct values in attributes. For example, suppose that from a relation

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

called *employment*, a rule "if occupation is medical doctor, then salary is high" is derived. In this rule, "medical doctor" is a domain element of attribute *occupation* and "high" is the concept specified in the DCH for the attribute *salary*. Hence, a semantic association from "Occupation is medical doctor" to "Salary is high" can be derived.

2.6.2 Generalization Algorithm

Generalization is necessary when there are a large number of distinct values for each attribute in a relation. When each attribute contains a large number of distinct values, the derived rules from the relation may not be useful because the rules may be complex or the rules may characterize few tuples. To derive compact rules from reduced relations, the learning process should go through a generalization process that ascends from specific values to higher-level concepts by climbing the DCH so that the number of distinct values is reduced. The generalization process can be performed on each attribute in an initial relation that is formed by joining a target attribute with the condition attributes from a base relation.

The generalization process for an attribute *A* can be summarized in three steps as follows:

Step 1: Let *R* be the ratio of the number of distinct values of attribute *A* to the total number of tuples in the initial relation *R*. If *R* is greater than the generalization threshold *GH*, go to the next step. Otherwise, the generalization on attribute *A* terminates.

Step 2: If there are no more higher-level concepts for the values of attribute *A* in the DCH, remove attribute *A* from *R* since there is a large set of distinct values that cannot be generalized. Otherwise, replace all values of attribute *A* with a higher-level concept by ascending one level of the DCH.

Step 3: Repeat steps 1 and 2 until *R* is less than or equal to *GH*.

The above process can be applied to all attributes in an initial relation to generalize the entire relation. When going through the generalization steps above, a set of tuples may be generalized by the same tuple. For example, the *gautomobile* relation shown in Figure 2.20 may be the result of the generalization process performed on a relation *automobile.*

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.20 Relation *gautomobile*

Id	Car	MPG	Weight	Drv-Ratio	Horsepower	Displace	Cyl
1	Buick Estate	Low	High	Medium	High	High	8
2	Ford Country	Low	High	Medium	Medium	High	8
3	Chevy Malibu	Low	Medium	Low	Medium	High	8
4	Chrysler LeBaron	Low	Medium	Low	High	High	8
5	Chevrolet	High	Low	High	Low	Low	4
6	Toyota Corona	Medium	Low	High	Low	Low	4
7	Datsun 510	Medium	Low	High	Low	Low	4
8	Dodge Omni	High	Low	Medium	Low	Low	4
9	Audi 5000	Medium	Low	High	Low	Low	5
10	Volvo 240 GL	Low	Medium	High	High	High	6
11	Saab 99 GLE	Medium	Low	High	Medium	Low	4
12	Peugeot 694 SL	Low	Medium	High	Medium	High	6
13	Buick Century	Medium	Medium	High	Low	Medium	6
14	Mercury Zephyr	Medium	Medium	High	Low	Medium	6
15	Dodge Aspen	Low	Medium	Medium	Low	Medium	6
16	AMC Concord D/L	Low	Medium	Medium	Medium	High	6
17	Chevy Caprice	Low	Medium	Low	High	High	8
18	Ford LTD	Low	Medium	Low	High	High	8
19	Mercury Grand	Low	Medium	Low	High	High	8
20	Dodge St Regis	Low	Medium	Low	High	Medium	8

2.6.3 Learning Process

The learning process first begins with the recording of the tuple numbers for the same attribute value of a single attribute and for each combination of attribute values of two different attributes by scanning the generalized relation once. For example, the single-attribute subtables for the two attributes *MPG* (miles per gallon) and *weight* can be created as shown in Figure 2.21.

The single-attribute subtables can be expanded further with more attributes. For example, consider the three attributes *MPG*, *weight*, and *horsepower*. Two-attribute subtables involving these three attributes can be created as shown in Figure 2.22.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.21 Single-attribute subtables for *MPG* and *weight*

MPG	
High	5, 8
Medium	6, 7, 9, 11, 13, 14
Low	1, 2, 3, 4, 10, 12, 15, 16, 17, 18, 19, 20

Weight	
High	1, 2
Medium	3, 4, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20
Low	

An event can be defined to be an attribute–value pair. For example, an event $X = (MPG, low)$ describes an event named X, namely, the attribute *MPG* has "low" as its value. A frequency function F maps one or more of these events into the frequency of these events appearing in the generalized relation. For example, $F(X) = 12$. On the other hand, a probability function P maps one or more of these events into the probability of these events occurring in the generalized relation. Hence, $P(X) = F(X)/N$, where N is the total number of tuples in the generalized relation. If we let $Y = (weight, high)$, $P(X)$ and $P(X, Y)$ are equal to 12/20 and 2/20, respectively.

Semantic Association Degrees (SADs) are degrees of semantic associations between domain elements or concepts, which are generally specified by database designers. Suppose that there are two events C and T. A degree D associated with "if C then T" is a SAD from C to T in which C is a condition event and T is a target event. D can be represented as $P(T \mid C) = P(C, T)/P(C) = F(C, T)/F(C)$. Furthermore, D can be extended with more condition events: $C_1, C_2, C_3, \ldots, C_k$. Hence, $P(T \mid C_1, C_2, C_3, \ldots, C_k) = F(C_1, C_2, C_3, \ldots, C_k, T)/F(C_1, C_2, C_3, \ldots, C_k)$. $F(C_1, C_2, C_3, \ldots, C_k)$ can be obtained as $Card\{TS(E_1, E_2) \cap TS(E_2, E_3) \cap \cdots \cap TS(E_{k-1}, E_k)\}$,where *Card* denotes the cardinality of a set and $TS(A, B)$ is a tuple set in which the two events A and B both appear simultaneously in the generalized relation.

For example, let $C_1 = (weight, low)$, $C_2 = (MPG, medium)$, and $C_3 = (horsepower, low)$ be condition events and $T = (drv\_ratio, high)$ be a target event. Then $P(T \mid C_1, C_2, C_3) = F(C_1, C_2, C_3, T) / F(C_1, C_2, C_3) = Card\{TS(C_1, C_2) \cap TS(C_2, C_3) \cap TS(C_3, T)\}/Card\{TS(C_1, C_2) \cap TS(C_2, C_3)\} = Card\{\{6, 7, 9, 11\} \cap \{6, 7, 9, 13, 14\} \cap \{5, 6, 7, 9, 13, 14\}\}/Card\{\{6, 7, 9, 11\} \cap \{6, 7, 9, 13, 14\}\} = 1$. This semantic association can be used to generate a rule "if (*weight* = *low*) and (*MPG* = *medium*) and (*horsepower* = *low*), then (*drv_ratio* = *high*)." This rule is supported by the tuple set {6, 7, 9}.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.22 Two attribute subtables for *MPG*, *weight*, and *horsepower*

MPG \ Weight	High	Medium	Low
High	—	—	5, 8
Medium	—	13, 14	6, 7, 9, 11
Low	1, 2	3, 4, 10, 12, 15, 16, 17, 18, 19, 20	—

MPG \ Horsepower	High	Medium	Low
High	—	—	5, 8
Medium	—	11	6, 7, 9, 13, 14
Low	1, 4, 10, 17, 18, 19, 20	2, 3, 12, 16	15

Weight \ Horsepower	High	Medium	Low
High	1	2	—
Medium	4, 10, 17, 18, 19, 20	3, 12, 16	13, 14, 15
Low	—	11	5, 6, 7, 8, 9

2.6.4 Learning Algorithm

In this section, the steps to derive rules using SADs are described with an example. First, it is assumed that there are k condition attributes (CA_1, CA_2, …, CA_k) and a Target Attribute (TA) in the generalized relation. The steps to derive the rules are as follows:

Step 1 (First iteration):
Compute the SAD from each condition event to the target event.
Step 2 (Iterations 2 through *k*):
For each subsequent iteration i ($2 \le i \le k$), compute the SAD from each combination of $j - i + 1$ condition events $CE = \{(CA_i, ca_i), (CA_{i+1}, ca_{i+1}), \ldots, (CA_j, ca_j)\}$ to each target event with the following strategies:
Strategy 1:
If the SAD from CE to a target event (TA, ta) is 1, derive the following rule R_1 with full confidence (100%): "If (CA_i, ca_i) and (CA_{i+1}, ca_{i+1}) and …

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

(CA_j, ca_j), then (TA, ta)." Furthermore, do not combine this *CE* with any other events since the sets of tuples corresponding to these combined *CE*s must be subsets of the set of tuples corresponding to the rule R_1. It is also noteworthy that the SADs from the *CE* to other target events must be 0.

Strategy 2:

If the SAD from *CE* to a target event (TA, ta) is 0, do not compute SADs from any combination containing the *CE* to the target event (TA, ta) in later iterations. TS_R is a tuple set supporting the rule R. Assume that $R_1, R_2, \ldots, R_{t-1}$ are previously derived rules.

Strategy 3:

When a new rule R_t is derived, if $TS_{R_1} \cup TS_{R_2} \cup \cdots \cup TS_{R_t}$ contains all tuples in the generalized relation, terminate this algorithm. On the other hand, if all tuples in which the target event (TA, ta) appears in the generalized relation are contained in $TS_{R_1} \cup TS_{R_2} \cup \cdots \cup TS_{R_t}$, do not compute the SAD from any condition event combination to the target event (TA, ta).

Strategy 4:

If TS_{R_i} $(1 \le i \le t-1)$ is a proper subset of TS_{R_t}, then discard R_i.

Strategy 5:

If TS_{R_t} is a subset of $TS_{R_1} \cup TS_{R_2} \cup \cdots \cup TS_{R_{(t-1)}}$, then discard R_t.

For example, consider the generalized relation *gautomobile* shown in Figure 2.20. The relationships between the target attribute *drv_ratio* and the three condition attributes *weight, horsepower*, and *MPG* are used to derive compact rules about the attribute *drv_ratio* from the relation. In the first iteration, the SADs from each condition event to the target event are obtained as shown in Figure 2.23. In the figure, the SAD from (*MPG, medium*) to the target event is 1. Hence, the rule "IF (*MPG* = *medium*) THEN (*drv_ratio* = *high*)" is derived and recorded as rule 1. This rule covers tuples 6, 7, 9, 11, 13, and 14. In the subsequent iteration, the event (*MPG* = *medium*) is removed from forming any other combination of condition events.

Continuing the algorithm in the next iteration to obtain SADs from a combination of two condition events to the target event yields the table in Figure 2.24. From the table, five rules with full confidence are derived and recorded as rules 2, 3, 4, 5, and 6 as follows:

Rule 2: IF (*MPG* = *low*) and (*weight* = *high*) THEN (*drv_ratio* = *medium*)
Rule 3: IF (*MPG* = *low*) and (*horsepower* = *low*) THEN (*drv_ratio* = *medium*)

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.23 SADs from *MPG*, *weight*, and *horsepower* to *drv_ratio*

MPG \ Drv_ratio	High	Medium	Low
High	1/2	1/2	0
Medium	1	0	0
Low	1/6	1/4	7/12

Weight \ Drv_ratio	High	Medium	Low
High	0	1/2	1/2
Medium	1/3	1/6	1/2
Low	5/6	1/6	0

Horsepower \ Drv_ratio	High	Medium	Low
High	1/7	1/7	5/7
Medium	2/5	1/5	2/5
Low	1/2	1/2	0

Rule 4: IF (*weight* = *high*) and (*horsepower* = *medium*) THEN (*drv_ratio* = *medium*)
Rule 5: IF (*weight* = *high*) and (*horsepower* = *high*) THEN (*drv_ratio* = *medium*)
Rule 6: IF (*weight* = *low*) and (*horsepower* = *medium*) THEN (*drv_ratio* = *high*)

Note that rule 2 is supported by tuples 1 and 2, rule 3 is supported by tuple 15, rule 4 by tuple 2, rule 5 by tuple 1, and rule 6 by tuple 11. In Figure 2.24, the × marks in the table indicate that the SADs need not be computed because the condition event combinations corresponding to them include the condition events corresponding to the value 0 in the table in Figure 2.23.

Continuing the algorithm with an additional condition event, the table in Figure 2.25 is obtained. In the table, no rules with full confidence are found. All the rules that can be generated from the table are ones with partial confidence. Hence, the algorithm to generate compact rules with full confidence terminates at this point.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

■ **FIGURE 2.24** SADs from two condition events to *drv_ratio*

MPG	Weight	Drv_ratio		
		High	Medium	Low
High	Low	1/2	1/2	×
High	Medium	0	0	×
High	High	×	0	×
Low	Low	0	0	0
Low	Medium	1/5	1/5	3/5
Low	High	×	1	0

MPG	Horsepower	Drv_ratio		
		High	Medium	Low
High	Low	1/2	1/2	×
High	Medium	0	0	×
High	High	0	0	×
Low	Low	0	1	×
Low	Medium	1/4	1/2	¼
Low	High	1/7	1/7	5/7

Weight	Horsepower	Drv_ratio		
		High	Medium	Low
High	Low	×	0	×
High	Medium	×	1	0
High	High	×	1	0
Medium	Low	2/3	1/3	×
Medium	Medium	1/3	1/3	1/3
Medium	High	1/6	0	5/6
Low	Low	4/5	1/5	×
Low	Medium	1	0	0
Low	High	0	0	0

■ **FIGURE 2.25** SADs with three condition events

MPG	Weight	Horsepower	Drv_ratio		
			High	Medium	Low
High	Low	Low	1/2	1/2	0
Low	Medium	Medium	1/3	1/3	1/3
Low	Medium	High	1/6	0	5/6

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.7 MINING OF TIME-CONSTRAINED ASSOCIATION RULES

Some databases, such as website traffic logs, contain sequential and timing information. For example, the rules generated from web log databases need to indicate sets of pages visited together in a certain order. This information can be used to forecast the next set of pages a visitor might frequent. In the generation of association rules considered so far, timing and sequential data has not been taken into account. To indicate timing and sequential constraints in association rules, the association rule generation algorithms presented in the previous sections need to be extended.

2.7.1 Time-Constrained Association Rules

In this section, we extend the association rule framework by introducing time constraints. For this purpose, traditional databases need to be augmented with a time stamp, which indicates sequence and timing of events. Suppose that we have a relation as shown in Figure 2.26. It should be noted that only the timing within a transaction is considered important, and hence the first item in every transaction is always timestamped at 0.

The support and confidence used in the previous sections are extended as follows, where $I = \{i_1, i_2, i_3, \ldots, i_n\}$ is a universe of all possible descriptive items. Assume that $X, Y \subseteq I$, and $X \cap Y = \varnothing$.

$$\mathit{forward\_sup}(X \Rightarrow Y) = \frac{\text{number of transactions containing } X \cup Y \text{, where } \mathit{time}(X) \leq \mathit{time}(Y)}{\text{total number of transactions}}$$

FIGURE 2.26 A relation with a time stamp

Transaction Numbers	Items	Time Stamp
T_1	B A	0 4
T_2	A B G D	0 2 5 9
T_3	C B A F G	0 3 5 12 45
T_4	A B C D	0 5 9 19
T_5	B F G	0 7 36
T_6	A B G D	0 6 12 28
T_7	F G A B	0 4 7 8

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

$$\mathit{forward\_conf}(X \Rightarrow Y) = \frac{\mathit{forward\_sup}(X \Rightarrow Y)}{\mathit{sup}(X)}$$

From the example relation, $\mathit{forward\_sup}(A \Rightarrow B) = 4/7 = 57\%$ and $\mathit{forward\_conf}(A \Rightarrow B) = 4/6 = 66.7\%$. Notice that the first and third transactions are not counted because A is preceded by B. Furthermore, $\mathit{forward\_conf}(F \Rightarrow G) = 100\%$, which means F is always visited in combination with G, whereas $\mathit{forward\_conf}(D \Rightarrow G) = 0\%$, from which it is deduced that D is never visited before G. Combining the above two measures gives more meaning to the association rules and gives us better insight into the data. Similarly, it is possible to define backward support and confidence measures as follows:

$$\mathit{backward\_sup}(X \Rightarrow Y) = \frac{\text{number of transactions containing } X \cup Y \text{, where } \mathit{time}(X) \geq \mathit{time}(Y)}{\text{total number of transactions}}$$

$$\mathit{backward\_conf}(X \Rightarrow Y) = \frac{\mathit{backward\_sup}(X \Rightarrow Y)}{\mathit{sup}(X)}$$

Therefore, both forward and backward measures of support and confidence can be generalized as the following definitions:

$$\mathit{time\_sup}(t_1, t_2)(X \Rightarrow Y) = \frac{\text{number of transactions containing } X \cup Y \text{, where } t_1 \leq \mathit{time}(Y) - \mathit{time}(X) \leq t_2}{\text{total number of transactions}}$$

$$\mathit{time\_conf}(t_1, t_2)(X \Rightarrow Y) = \frac{\mathit{time\_sup}(t_1, t_2)(X \Rightarrow Y)}{\mathit{sup}(X)}$$

In the definitions above, t_1 and t_2 are integers that are used to define a timing window and $t_1 \leq t_2$. From these generalized definitions, the following can be observed with $t_i \leq t_j$:

1. $\mathit{time\_sup}(t_j, t_k)(X \Rightarrow Y) \leq \mathit{time\_sup}(t_i, t_k)(X \Rightarrow Y) \leq \mathit{sup}(X \Rightarrow Y)$
2. $\mathit{time\_sup}(t_k, t_i)(X \Rightarrow Y) \leq \mathit{time\_sup}(t_k, t_j)(X \Rightarrow Y) \leq \mathit{sup}(X \Rightarrow Y)$
3. $\mathit{time\_conf}(t_j, t_k)(X \Rightarrow Y) \leq \mathit{time\_conf}(t_i, t_k)(X \Rightarrow Y) \leq \mathit{conf}(X \Rightarrow Y)$
4. $\mathit{time\_conf}(t_k, t_i)(X \Rightarrow Y) \leq \mathit{time\_conf}(t_k, t_j)(X \Rightarrow Y) \leq \mathit{conf}(X \Rightarrow Y)$

From the definition of *time_sup* and *time_conf* above, it can be observed that for $t_1 = -\infty$ and $t_2 = +\infty$, $\mathit{time\_sup}(t_1, t_2)(X \Rightarrow Y)$ and $\mathit{time\_conf}(t_1, t_2)(X \Rightarrow Y)$

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

converge to normal $sup(X \Rightarrow Y)$ and $conf(X \Rightarrow Y)$. Furthermore, for $t_1 = 0$ and $t_2 = +\infty$ they converge to *forward_sup* and *forward_conf* and for $t_1 = -\infty$ and $t_2 = 0$ they converge to *backward_sup* and *backward_conf*.

2.7.2 Properties of Time Constraints

From the definitions of *time_sup* and *time_conf*, two special measures called *support_ratio* and *confidence_ratio* can be defined as follows:

$$support\_ratio\ (t_1, t_2)(X \Rightarrow Y) = \frac{time\_sup(t_1, t_2)(X \Rightarrow Y)}{sup(X \Rightarrow Y)}$$

$$confidence\_ratio\ (t_1, t_2)(X \Rightarrow Y) = \frac{time\_conf\,(t_1, t_2)(X \Rightarrow Y)}{conf\ (\mathrm{X} \Rightarrow \mathrm{Y})}$$

It should be noted that both ratios are within the range of 0 to 1 because *time_sup* and *time_conf* are always smaller than normal *sup* and *conf*. Given the fact that both X and Y appear in the transaction, both ratios express the conditional probability of X and Y within the time window defined by t_1 and t_2.

In a real-life situation, an item may appear more than once. For example, in web navigation a visitor may access web pages more than once. In that case, the time difference between two items is not unique. Consider the case in Figure 2.27 in which some items appear more than once.

To calculate the *time_sups* of two items that appear multiple times, there are several measures that can be used. For example, to calculate $time\_sup(t_1, t_2)$ $(A \Rightarrow B)$, a transaction can be counted if

1. $t_1 \leq time(B) - time(A) \leq t_2$ is valid for all occurrences of A and B;
2. $t_1 \leq time(B) - time(A) \leq t_2$ is valid for at least one occurrence of A and B;
3. $t_1 \leq average[time(B) - time(A)] \leq t_2$ is valid; and
4. $t_1 \leq time(\text{first occurrence of } B) - time(\text{first occurrence of } A) \leq t_2$ is valid.

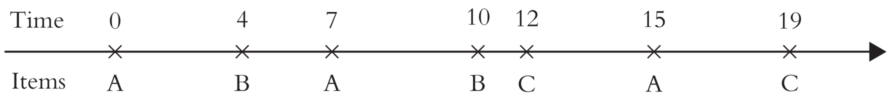

FIGURE 2.27 Transaction with multiple occurrences of some items

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.7.3 Potential Applications

The main goal of web mining is to extract useful knowledge from web-server databases containing data about the behavior of customers. The potential application of the discovery of time-constrained association rules is the mining of a website visitor's navigational patterns. On the data collected about a visitor's activities, a slightly modified version of the Apriori algorithm can be used with values for t_1 and t_2 to define a time window.

In market basket analysis, our first application of association rules, there was no ordering of items since all of the items were placed in a basket and were paid for. It was actually impossible to detect the order in which customers picked the items from the shelves. However, random checks can be performed to gain worthwhile information to improve sales. Marketers can see how customers wander through the store to determine a customer's pattern of moving from shelf to a shelf. However, it's not so easy to see why (say) eggs and wine are frequently sold together. It may be that the wine was the primary reason for shopping and eggs were picked up while passing by. The use of time-constrained association rule mining can provide useful tips for answering some of these questions.

2.8 CHAPTER SUMMARY

Association-rule mining in databases was discussed in this chapter. The problem of discovering all association rules in market basket data was described in the first section. The Apriori algorithm was used for association-rule mining in market basket data. In the next section, the attribute-oriented rule induction method was presented. A concept hierarchy was used to mine characteristic and discrimination rules in a database. The third section dealt with association rules in hypertext databases. In this section, hypertext systems were modeled as weighted graphs. Finding a set of web pages connected by a link was the focus of the discussion. The mining of association rules over quantitative and categorical attributes was described in the next section. Next, the problem of mining compact rules was discussed. In this section, a new concept called the semantic association relationship was defined and was used to facilitate the extraction of compact association rules. The chapter concluded with a discussion on mining time-constrained association rules, an extension of the association-rule framework, which was illustrated by introducing time constraints.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.9 EXERCISES

1. Use the database table in Figure 2.28, which contains ten market transactions. Use the Apriori algorithm to complete the association rule generation process. Experiment with different values of support and confidence to observe how they control the number of association rules generated.
2. Consider the weighted directed graph shown in Figure 2.29. Generate all possible composite association rules meeting the support values of 1.0, 1.5, and 2.0.
3. The database table shown in Figure 2.30 contains a dataset with the following five attributes: *color* (yellow, purple), *size* (large, small), *act* (stretch, dip),

FIGURE 2.28 Table for Apriori example

TID	Items
1	1, 3, 4, 5, 6, 7, 9, 10
2	2, 4, 5, 7, 8, 9, 10
3	1, 2, 3, 4, 10
4	2, 4, 5, 6, 7, 10
5	1, 3, 4, 5, 7, 8
6	3, 5, 7, 8, 9, 10
7	1, 3, 4, 8, 9, 10
8	2, 5, 6, 7, 8, 9
9	1, 3, 4, 5, 6, 7, 10
10	1, 2, 3, 4, 5, 7, 8, 9, 10

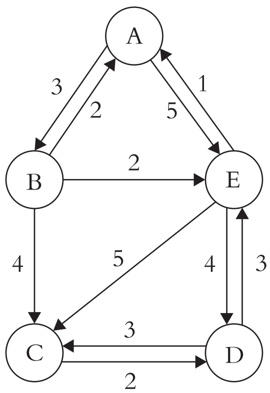

FIGURE 2.29 A weighted directed graph

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.30
Balloon table

Color	Size	Act	Age	Inflated
Yellow	Small	Stretch	Adult	T
Yellow	Small	Stretch	Child	T
Yellow	Small	Dip	Adult	T
Yellow	Small	Dip	Child	T
Yellow	Large	Stretch	Adult	T
Yellow	Large	Stretch	Child	F
Yellow	Large	Dip	Adult	F
Yellow	Large	Dip	Child	F
Purple	Small	Stretch	Adult	T
Purple	Small	Stretch	Child	F
Purple	Small	Dip	Adult	F
Purple	Small	Dip	Child	F
Purple	Large	Stretch	Adult	T
Purple	Large	Stretch	Child	F
Purple	Large	Dip	Adult	F
Purple	Large	Dip	Child	F

age (adult, child), and *inflated* (T, F). Follow the method used in Section 2.6 for generating compact rules to derive all rules with full confidence.

4. The major-salary table in Figure 2.31 shows 30 instances of IT professionals in terms of four attributes: *age*, *education*, *major* (computer science, electrical engineering, management information science, decision science), and *salary*. Construct conceptual hierarchies for *age*, *edu*, *major*, and *salary*. Generate interesting learning tasks for deriving discrimination and characteristic rules from the table. Use the AO induction method to derive either discrimination or characteristic rules based on the learning tasks generated.
5. Figure 2.32 shows a contact lenses table that contains information about contact lens prescriptions (hard lenses, soft lenses, and no contact lenses). From the table, derive quantitative association rules by mapping tables to Boolean association rules.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

FIGURE 2.31 Major–salary table

ID	Age	Edu	Major	Salary	ID	Age	Edu	Major	Salary
1	35	Ph.D	EE	70K	16	34	B.S	MIS	58K
2	45	B.S	EE	60K	17	36	Ph.D	EE	64K
3	55	B.S	CS	65K	18	34	Ph.D	CS	70K
4	28	M.S	EE	45K	19	56	M.S	CS	72K
5	32	B.S	DS	46K	20	52	B.S	EE	65K
6	31	B.S	EE	44K	21	37	M.S	CS	62K
7	42	B.S	EE	55K	22	42	Ph.D	MIS	70K
8	29	M.S	DS	41K	23	42	M.S	CS	69K
9	35	Ph.D	CS	72K	24	38	Ph.D	MIS	73K
10	39	Ph.D	CS	75K	25	45	M.S	CS	65K
11	41	Ph.D	CS	75K	26	49	M.S	CS	67K
12	32	B.S	DS	43K	27	36	Ph.D	MIS	69K
13	54	M.S	CS	68K	28	28	M.S	EE	46K
14	37	M.S	EE	56K	29	36	B.S	EE	46K
15	35	Ph.D	DS	67K	30	32	Ph.D	CS	77K

FIGURE 2.32 Contact lenses table

ID	Age	Spectacle	Astigmatic	Tear Production	Contact lens
1	21	Myope	No	Reduced	None
2	24	Myope	No	Normal	Soft
3	20	Myope	Yes	Reduced	None
4	26	Myope	Yes	Normal	Hard
5	27	Hypermetrope	No	Reduced	None
6	22	Hypermetrope	No	Normal	Soft
7	28	Hypermetrope	Yes	Reduced	None
8	27	Hypermetrope	Yes	Normal	Hard
9	38	Myope	No	Reduced	None
10	32	Myope	No	Normal	Soft
11	36	Myope	Yes	Reduced	None
12	37	Myope	Yes	Normal	Hard
13	33	Hypermetrope	No	Reduced	None
14	32	Hypermetrope	No	Normal	Soft
15	39	Hypermetrope	Yes	Reduced	None
16	34	Hypermetrope	Yes	Normal	None
17	52	Myope	No	Reduced	None
18	51	Myope	No	Normal	None
19	50	Myope	Yes	Reduced	None
20	54	Myope	Yes	Normal	Hard
21	52	Hypermetrope	No	Reduced	None
22	55	Hypermetrope	No	Normal	Soft
23	58	Hypermetrope	Yes	Reduced	None
24	54	Hypermetrope	Yes	Normal	None

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

2.10 SELECTED BIBLIOGRAPHIC NOTES

The Apriori and AprioriTid algorithms described in Section 2.2 are from [Agrawal 1994]. General survey and comparisons of various association-rule mining algorithms are given in [Hipp 2000]. [Agrawal 1994] and [Manilla 1994a] give new and improved methods for discovering association rules. Generalized association-rule mining is discussed in [Srikant 1995], whereas [Denwattana 2001] gives a parameterized algorithm for association-rule mining.Temporal constraints in association-rule generation are from [Li 2003], [Lee 2001], and [Ting 2003]. Detailed algorithms and strategies for mining attribute-oriented associations described in Section 2.3 are directly from [Han 1992] and [Hwang 1995]. [Brin 1997] discusses the problem of generalizing association rules to correlations, whereas the mining of regression rules and trees is shown in [Sher 1998].

Special hashing techniques are used to improve the efficiency of association-rule mining in [Ozel 2001] and [Park 1997]. Generating navigation patterns of users on hypertext-based web pages is an important application area of association-rule mining. Section 2.4 addresses this issue. The method of rule generation proposed is from [Borges 1998]. Preserving data privacy in mining of association rules is covered in [Evfimievski 2002] and [Rizvi 2002]. Special constraints other than minimum support and confidence are used in [Bayardo 1999] and [Yen 2001].

Special techniques for mining of special patterns from data are proposed in [Monge 1996], [Pei 2001], [Michail 2000], [Nahm 1986], and [Nahm 2002]. For induction used as the primary tool for knowledge extraction, see [Moshkovich 2002], [Wu 1999], and [Stefanowski 1994]. Relating association rules with weights is discussed in [Cai 1998], whereas [Pôssas 2000] and [Hong 1999] address the issue of generating association rules from quantitative data. The method for generating quantitative association rules presented in Section 2.5 is from [Srikant 1996]. Memory-adaptive association-rule mining is discussed in [Nanopoulos 2004]. For predictive association-rule discovery, see [Megiddo 1998].

For efficient algorithms for association-rule mining in large databases, see [Savasere 1995], [Agrawal 1993], and [Han 1999]. [Cheung 1996] and [Dehaspe 1995] discuss the mining of association rules in distributed databases and multiple relations. For parallel mining of association rules, see [Agrawal 1996]. The algorithm presented in Section 2.6 for generating compact rules

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

is proposed in [Yen 1995]. The chapter concluded with a discussion of time-constrained association-rule mining. The material presented in this section is proposed in [Huysmans 2004]. [Liu 1998] discusses the integration of classification with association-rule mining. The application of association rules for protein-protein interaction networks is proposed in [Besemann 2004], whereas its application to remotely sensed data is discussed in [Dong 2000]. Concepts of decision rules and decision trees are used in [Quinlan 1987] and [Shan 1993] to generate rules from databases. Association-rule mining is combined with formal concept analysis in [Deogun 1998]. For approximate rule mining, see [Nayak 2001].

2.11 CHAPTER BIBLIOGRAPHY

[Agrawal 1993] R. Agrawal, T. Imielinski, and A. Swami: "Mining Association Rules Between Sets of Items in Large Databases," *ACM SIGMOD*, pp. 207–216, Washington, 1993.

[Agrawal 1994] R. Agrawal and R. Srikant: "Fast Algorithms for Mining Association Rules," *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.

[Agrawal 1996] R. Agrawal and J. C. Shafer: "Parallel Mining of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 962–969, 1996.

[Agrawal 2001] C. Agrawal and P. Yu: "A New Approach to Online Generation of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 4, pp. 527–540, 2001.

[Bayardo 1999] R. Bayardo, R. Agrawal, and D. Cunopulos: "Constraint-Based Rule Mining in Large, Dense Databases," *Proceedings of the 15th International Conference on Data Engineering*, 1999.

[Besemann 2004] C. Besemann, A. Denton, and A. Yekkirala: "Differential Association Rule Mining for the Study of Protein-Protein Interaction Networks," *Proceedings of the 4th Workshop on Data Mining in Bioinformatics (BIOKDD)*, Seattle, WA, pp. 1–9, 2004.

[Borges 1998] J. Borges and M. Levene: "Mining Association Rules in Hypertext Databases," *International Conference on KDD*, pp. 149–153, 1998.

[Brin 1997] S. Brin, R. Motwani, and C. Silverstein: "Beyond Market Baskets—Generalizing Association Rules to Correlations," *SIGMOD*, pp. 265–276, 1997.

[Cai 1998] C. H. Cai, A. Fu, C. H. Cheng, and W. W. Kwong: "Mining Association Rules with Weighted Items," *Proceedings of IEEE International Database Engineering and Applications Symposium (IDEAS)*, United Kingdom, pp. 68–77, 1998.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Cheung 1996] D. W. Cheung, V. T. Ng, A. W. Fu, and Y. Fu: "Efficient Mining of Association Rules in Distributed Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 6, pp. 911–922, 1996.

[Dehaspe 1995] L. Dehaspe and L. De Raedt: "Mining Association Rules in Multiple Relations," *Proceedings of 7th International Workshop on ILP*, Vol. 1297 of LNCS, pp. 125–132, 1995.

[Denwattana 2001] N. Denwattana and J. R. Getta: "A parameterized algorithm for mining association rules," *Proceedings of the 12th Australasian Database Conference*, pp. 45–51, Queensland, Australia, 2001.

[Deogun 1998] J. Deogun, V. Raghavan, and H. Sever: "Association Mining and Formal Concept Analysis," *Proceedings of the 6th International Workshop on Rough Sets, Data Mining and Granular Computing*, Vol. 1, pp. 335–338, 1998.

[Dong 2000] J. Dong, W. Perrizo, Q. Ding, and J. Zhou: "The Application of Association Rule Mining to Remotely Sensed Data," *Proceedings of the ACM Symposium on Applied Computing*, pp. 340–45, Vol. 1, 2000.

[Evfimievski 2002] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke: "Privacy Preserving Mining of Association Rules," *Proceedings of 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[Han 1992] J. Han, Y. Cai, and N. Cercone: "Knowledge Discovery in Databases: An Attribute-Oriented Approach," *Proceedings of the 18th VLDB Conference,* Vancouver*, British Columbia, Canada*, 1992.

[Han 1999] J. Han, Y. Fu: "Mining Multiple-Level Association Rules in Large Databases," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 5, pp. 1–8, 1999.

[Hipp 2000] J. Hipp, U. Guntzer, and G. Nakaeizadeh: "Algorithms for Association Rule Mining—A General Survey and Comparison," *ACM SIGKDD Explorations*, Vol. 2, No. 1, pp. 58–64, 2000.

[Hong 1999] T. P. Hong, C. S. Kuo, S. C. Chi: "Mining Association Rules from Quantitative Data," *Intelligent Data Analysis*, Vol. 3, pp. 363–376, 1999.

[Huysmans 2004] J. Huysmans, B. Baesens, C. Mues, and J. Vanthienen: "Web Usage Mining with Time Constrained Association Rules," *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS)*, Porto, Portugal, pp. 343–348, 2004.

[Hwang 1995] H. Hwang and W. Fu: "Efficient Algorithms for Attribute-Oriented Induction," *International Conference on KDD*, 1995.

[Lee 2001] C. H. Lee, C. R. Lin, and M. S. Chen: "On Mining General Temporal Association Rules in a Publication Database," *ICDM*, pp. 337–344, 2001.

[Li 2003] Y. Li, P. Ning, X. S. Wang, and S. Jajodia: "Discovering Calendar-based Temporal Association Rules," *Data and Knowledge Engineering*, Vol. 44, No. 2, pp. 193–218, 2003.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Liu 1998] B. Liu, W. Hsu, Y. Ma: "Integrating Classification and Association Rule Mining," *Proceedings of International Conference on Knowledge Discovery in Databases*, pp. 80–86, 1998.

[Mannila 1994a] H. Mannila, H. Toivonen, A. Verkamo: "Efficient Algorithms for Discovering Association Rules," *Proceedings of AAAI Workshop: Knowledge Discovery in Databases*, pp. 181–192, 1994.

[Mannila 1994b] H. Mannila, H. Toivonen, and A. Verkamo: "Improved Methods for Finding Association Rules," *Proceedings of AAAI Workshop: Knowledge Discovery in Databases*, 1994.

[Megiddo 1998] N. Megiddo and R. Srikant: "Discovering Predictive Association Rules," *Proceedings of the 4th International Conference on Knowledge Discovery in Databases and Data Mining*, pp. 274–278, New York, 1998.

[Mehta 1996] M. Mehta, R. Agrawal, and J. Rissanen: "SLIQ: A Fast Scalable Classifier for Data Mining," *EDBT*, pp. 18–32, 1996

[Michail 2000] A. Michail: "Data Mining Library Reuse Patterns Using Generalized Association Rules," *International Conference on Software Engineering*, pp. 167–176, 2000.

[Monge 1996] A. E. Monge and C. P. Elkan: "The Field Matching Problem: Algorithms and Applications," *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 267–270, 1996.

[Moshkovich 2002] H. M. Moshkovich, A. Mechitov, and D. L. Olson: "Rule Induction in Data Mining: Effect of Ordinal Scales," *Expert Systems with Applications*, Vol. 22, pp. 303–311, 2002.

[Nahm 1986] U. Y. Nahm and R. J. Mooney: "Mining Soft-Matching Rules from Textual Data," *IJCAI*, pp. 979–986, 2001.

[Nahm 2002] U. Y. Nahm and R. J. Mooney: "Mining Soft-Matching Association Rules," *Proceedings of the CIKM*, pp. 681–683, McLean, VA, 2002.

[Nanopoulos 2004] A. Nanopoulos and Y. Manolopoulos: "Memory-adaptive Association Rules Mining," *Information Systems*, Vol. 29, No. 5, pp. 365–384, 2004.

[Nayak 2001] J. R. Nayak and D. J. Cook: "Approximate Association Rule Mining," *FLAIRS Conference* (http://ranger.uta.edu/~cook/pubs/flairsj01.pdf), pp. 259–263, 2001.

[Ozel 2001] S. A. Ozel and H. A. Guvenir: "An Algorithm for Mining Association Rules Using Perfect Hashing and Database Pruning," *Proceedings of the 10th Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN)*, A. Acan, I. Aybay, and M. Salamah (Eds.), Gazimagusa, T.R.N.C. pp. 257–264, 2001.

[Park 1997] J. S. Park, M. S. Chen, and P. S. Yu: "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, pp. 813–825, Vol. 9, No. 5, 1997.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.

[Pei 2001] J. Pei, A. K. H. Tung, and J. Han: "Fault-Tolerant Frequent Pattern Mining: Problems and Challenges," *Proceedings of the 2001 ACM-SIGMOD International Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD)*, Santa Barbara, CA, 2001.

[Pôssas 2000] B. Pôssas, W. Meira Jr., M. Carvalho, and R. Resende: "Using Quantitative Information for Efficient Association Rule Generation," *SIGMOD Record*, Vol. 29, No. 4, pp. 19–25, 2000.

[Quinlan 1987] J. R. Quinlan: "Generating Production Rules From Decision Trees," *Proceeding of the IJCAIS*, pp. 304–307, 1987.

[Rizvi 2002] S. Rizvi and J. Haritsa: "Maintaining Data Privacy in Association Rule Mining," *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[Savasere 1995] A. Savasere, E. Omiecinski, and S. Navathe: "An Efficient Algorithm for Mining Association Rules in Large Databases," *Proceedings of the 2nd VLDB Conference*, Zurich, Switzerland, pp. 432–444, 1995.

[Shan 1993] N. Shan and W. Ziarko: "An Incremental Learning Algorithm for Constructing Decision Rules," *Proceedings of the International Workshop on Rough Sets and Knowledge Discovery*, Banff, Canada, pp. 326–334, 1993.

[Sher 1998] B.Y. Sher, S. C. Shao, and W. S. Hsieh: "Mining Regression Rules and Regression Trees," *PAKDD*, pp. 271–282, 1998.

[Srikant 1995] R. Srikant and R. Agrawal: "Fast Algorithms for Mining Association Rules," *Proceeding of the 21st VLDB Conference*, Zurich, Switzerland, 1995.

[Srikant 1996] R. Srikant and R. Agrawal: "Mining Quantitative Association Rules in Large Relational Tables," *SIGMOD*, pp. 1–12, 1996.

[Stefanowski 1994] J. Stefanowski and D. Vanderpooten: "A General Two-Stage Approach to Inducing Rules from Examples," *Rough Sets, Fuzzy Sets and Knowledge Discovery,* pp. 317–325, 1994.

[Wu 1999] X. Wu and D. Urpani: "Induction by Attribute Elimination," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, pp. 805–812, 1999.

[Yen 1995] S. Yen and A. Chen: "An Efficient Algorithm for Deriving Compact Rules from Databases," *Proceedings of the 4th International Conference on Database Systems for Advanced Applications* (*DASFAA*), pp. 364–371, Singapore, 1995.

[Yen 2001] S. Yen and A. Chen: "A Graph-Based Approach for Discovering Various Types of Association Rules," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 13, No. 5, pp. 839–845, 2001.

[Ying 2003] L. Yingjiu, P. Ning, X. S. Wang, and S. Jajodia: "Discovering Calendar-based Temporal Association Rules," *Data and Knowledge Engineering*, Vol. 44, No. 2, pp. 193–218, 2003.

© Jones & Bartlett Learning, LLC. NOT FOR SALE OR DISTRIBUTION.