10

Unsupervised Learning: Association Rules

In this Chapter, we cover the second key technique of unsupervised learning, namely, association rules. The first technique, clustering, was covered in Chapter 9. We discuss both the algorithms and data from which association rules are generated.

1. Introduction

Association rules mining is another key **unsupervised** data mining method, after clustering, that finds interesting **associations** (relationships, dependencies) in large sets of data **items**. The items are stored in the form of **transactions** that can be generated by an external process, or extracted from relational databases or data warehouses. Due to good **scalability** characteristics of the association rules algorithms and the ever-growing size of the accumulated data, association rules are an essential data mining tool for extracting knowledge from data. The discovery of interesting associations provides a source of information often used by businesses for decision making. Some application areas of association rules are market-basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, data preprocessing, genomics, etc. Interesting examples are personalization and recommendation systems for browsing web pages (such as Amazon's recommendations of related/associated books) and the analysis of genomic data.

Market-basket analysis, one of the most intuitive applications of association rules, strives to analyze customer buying patterns by finding associations between items that customers put into their baskets. For instance, one can discover that customers buy milk and bread together, and even that some particular brands of milk are more often bought with certain brands of bread, e.g., multigrain bread and soy milk. These and other more interesting (and previously unknown) rules can be used to maximize profits by helping to design successful marketing campaigns, and by customizing store layout. In the case of the milk and bread example, the retailer may not offer discounts for both at the same time, but just for one; the milk can be put at the opposite end of the store with respect to bread, to increase customer traffic so that customers may possibly buy more products. A number of interesting associations can be found in the market basket data, as illustrated in Figure 10.1.

This Chapter provides background and explains which data are suitable for association rule mining, what kinds of association rules can be generated, how to generate association rules, and which rules are the most interesting. The explanations are supported by easy-to-understand examples. Upon finishing this Chapter, the reader will know how to generate and interpret association rules.

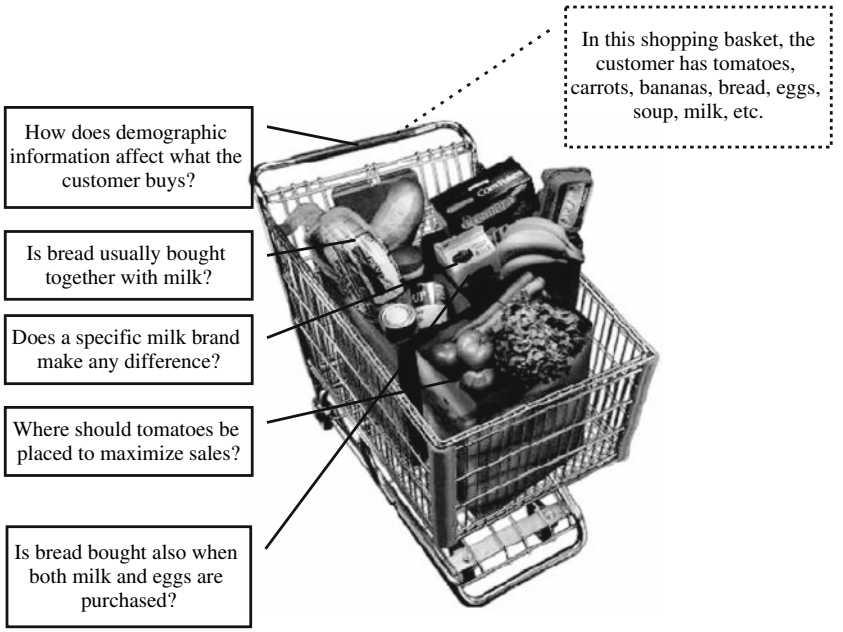

Figure 10.1. Application of association rules in market-basket analysis.

2. Association Rules and Transactional Data

Continuing our example of market-basket analysis, we represent each product in a store as a Boolean variable, which represents whether an item is present or absent. Each customer's basket is represented as a Boolean vector, denoting which items are purchased. The vectors are analyzed to find which products are frequently bought together (by different customers), i.e., **associated** with each other. These cooccurrences are represented in the form of **association rules**:

$$\text{LHS} \Rightarrow \text{RHS} \ [\text{support, confidence}]$$

where the left-hand side (LHS) implies the right-hand side (RHS), with a given value of support and confidence.

Support and **confidence** are used to measure the quality of a given rule, in terms of its usefulness (strength) and certainty. Support tells how many examples (transactions) from a data set that was used to generate the rule include items from both LHS and RHS. Confidence expresses how many examples (transactions) that include items from LHS also include items from RHS. Measured values are most often expressed as percentages. An association rule is considered interesting if it satisfies minimum values of confidence and support, which are to be specified by the user (domain expert). The following examples are used to illustrate the concepts.

Example: An association rule that describes customers who buy milk and bread.

$$\text{buys (x, milk)} \Rightarrow \text{buys (x, bread)} \ [25\%, 60.0\%]$$

The rule shows that customers who buy milk also buy bread. The direction of the association, from left to right, shows that buying milk "triggers" buying bread. These items are bought together in 25% of store purchases (transactions), and 60% of the baskets that include milk also include bread.

Example: An association rule describing graduate students might read as follows:

major(x, Computer Engineering) AND takes_course(x, Advanced Data Analysis and Decision Making) $\Rightarrow$ level(x, PhD) [1%, 75%]

The rule has two items in the LHS and one item in the RHS. Association rules can include multiple items in their LHS. The rule in this example states that students who major in Computer Engineering and who take Advanced Data Analysis and Decision Making course are at the Ph.D. level with 1% support and 75% confidence. Again, the support shows that 1% of all students in the database satisfy this association. At the same time, among those who major in Computer Engineering and who take Advanced Data Analysis and Decision Making courses, 75% are the Ph.D. students.

Association rules are derived when data describe events (items in a transaction) that occur at the same time or in close proximity. The two main types of association rules are **single-dimensional** and **multidimensional**. The former refers to one dimension, such as *buy* in the first example, while the latter refers to more than one dimension, as in the second example where we have three dimensions: *major*, *takes_course*, and *level*. Both of these types of association rules could also be categorized as **Boolean** or **quantitative**. The former concerns the presence or absence of an item, while the latter considers quantitative values, which are partitioned into item intervals. An example of a multidimensional quantitative rule follows.

Example: An association rule that describes the finding that young adult customers who earn below 20K buy bread.

age(x, "(18, 25)") $\wedge$ income(x, "<20K") $\Rightarrow$ buy(x, bread) [0.5%, 50.0%]

The quantitative items such as *age* and *income* are discretized (see Chapter 8).

Association rules can be also categorized as **single-level** and **multilevel**. The former operate on a single level of abstraction, while the latter are based on items that can be expressed at different levels in a **hierarchy** (see Chapter 6). The example below shows a multilevel association rule (which can be contrasted with the single-level rule given in the first example).

Example: A multilevel association rule that describes customers buying skim milk and large white bread.

buys (x, skim_milk) $\Rightarrow$ buys (x, large_white_bread) [2.5%, 60.0%]

In the above rule, the *milk* and *bread* items are subdivided into different kinds that constitute a hierarchy. For instance, *bread* can be divided into *white* and *wheat*, and each of these two types can be subdivided into *small*, *medium*, and *large*.

In what follows, we describe data and methods that can be used to generate single-dimensional (single-level) Boolean association rules. Next, we discuss how to extend these basic algorithms to address multidimensional, multilevel, and quantitative rules.

2.1. Transactional Data

Input data for an association-rule mining algorithm are provided in the **transactional form**. Each record (example) should consist of a transaction ID and information about all items (in a consistent format) that constitute the transaction, as shown in Figure 10.2.

An example of transactional data that concern the market-basket example for a grocery store is shown in Table 10.1.

Two example association rules that can be found by visual analysis of the above table are

$$\text{Beer} \Rightarrow \text{Eggs}$$

$$\text{Apples} \Rightarrow \text{Celery}$$

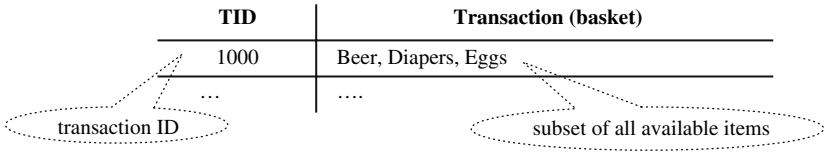

Figure 10.2. Example transaction.

Table 10.1. Example of transactional data..

TID	Transaction (basket)
1000	Apples, Celery, Diapers
2000	Beer, Celery, Eggs
3000	Apples, Beer, Celery, Eggs
4000	Beer, Eggs

The transactional data can be obtained directly from a transactional database (which could be stored by a grocery or a retail store), or alternatively, these data can be obtained by a simple transformation of relational data. A relational table describing patients of a heart clinic (see Chapter 3) can be transformed to the transactional form, as shown in Figure 10.3.

Given a suitable transactional database, in which each transaction is a list of items (say, items being purchased by a customer in a single visit to a store), we aim at finding association rules that relate the presence of one set of items with another set of items. In the following, we formally define basic concepts used to describe an association mining algorithm.

2.2. Basic Concepts

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a **set of items** and D be the **set of transactions** (transactional data set) where each **transaction** $T \subseteq I$ is associated with an **identifier** TID and m is the number of items. Let A and B be two sets of items. A transaction T is said to contain A if and only if $A \subseteq T$. An **association rule** is an implication in the form $A \Rightarrow B$ where $A \subset I$, $B \subset I$, and $A \cap B = \emptyset$

The interestingness of an association rules describes how significant the rule is with respect to D. Two measures are used to quantify the interestingness of a rule:

patient (relational)

patient ID	name	age	sex	chest pain type	defect type	diagnosis
P1	Konrad Black	31	male	1	normal	absent
P2	Konrad Black	26	female	4	fixed	present
P3	Anna White	56	female	2	normal	absent
…	…	…	…	…	…	…

patient (transactional)

TID	Transaction (basket)
1000	name=Konrad Black, age=31, sex=male, chest_pain_type=1, defect type=normal, diagnosis=absent
2000	name=Konrad Black, age=26, sex=female, chest_pain_type=4, defect type=fixed, diagnosis=present
3000	name=Anna White, age=56, sex=female, chest_pain_type=2, defect type=normal, diagnosis=absent
…	…

Figure 10.3. Transformation from relational to transactional data.

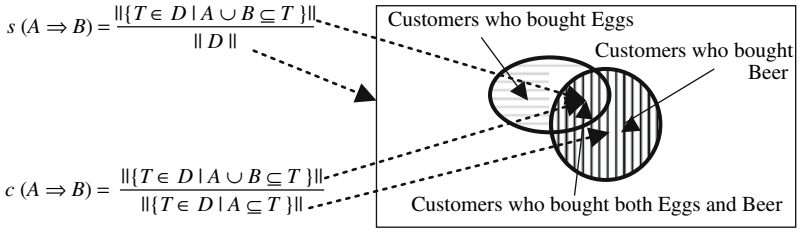

Figure 10.4. Computing support and confidence for the association rule Beer ⇒ Eggs.

– **Support**, which indicates the frequency (probability) of the entire rule with respect to D. It is defined as ratio of the number of transactions containing A and B to the total number of transactions (the probability of both A and B cooccurring in D):

$$\text{support}(A \Rightarrow B) = P(A \cup B) = \frac{||\{T \in D | A \cup B \subseteq T\}||}{||D||}$$

– **Confidence**, which indicates the strength of implication in the rule. It is defined as ratio of the number of transactions containing A and B to the number of transactions containing A (conditional probability of B given A):

$$\text{confidence}(A \Rightarrow B) = P(B|A) = \frac{||\{T \in D | A \cup B \subseteq T\}||}{||\{T \in D | A \subseteq T\}||}$$

Figure 10.4 illustrates these concepts using the rule Beer ⇒ Eggs as an example.

Rules that satisfy both the minimum support threshold and the minimum confidence threshold are called **strong association rules**, as explained in Figure 10.5. In Figure 10.6, we give another example to better explain the introduced concepts and definitions.

A set of items is referred to as an **itemset** (in data mining, the term itemset is used instead of item set). An itemset that contains k items is referred to as a **k-itemset**. For instance, {Beer, Eggs} is a 2-itemset. The **support count** (also known as frequency, occurrence frequency, or count) of

For the given transactions, find all rules such that LHS = $\{A, B\}$, RHS = $\{C\}$, with minimum support = 50% and minimum confidence = 50%.

TID	Transactions
1000	A, B, C
2000	A, C
3000	A, D
4000	B, E, F

Support is the probability that a transaction contains $\{A, B, C\}$, and *confidence* is the conditional probability that a transaction containing $\{A, B\}$ also contains C.
Rule $A \wedge B \Rightarrow C$ [support 25%, confidence 100%] does not satisfy the minimum confidence. Two (shorter) strong association rules are generated as:
$A \Rightarrow C$ [support 50%, confidence 66.6%]
$C \Rightarrow A$ [support 50%, confidence 100%]

Figure 10.5. Strong association rules.

For the given transactions

TID	Transactions
1000	Apples, Celery, Diapers
2000	Beer, Celery, Eggs
3000	Apples, Beer, Celery, Eggs
4000	Beer, Eggs

find I, T for TID = 2000, *support* (Beer $\Rightarrow$ Eggs), and *confidence* (Beer $\Rightarrow$ Eggs)
I = {Apples, Beer, Celery, Diapers, Eggs}
T = {Beer, Celery, Eggs}
support (Beer $\Rightarrow$ Eggs) = 75%
confidence (Beer $\Rightarrow$ Eggs) = 100%

Figure 10.6. Example transactional data and related association rule concepts.

an itemset is the number of transactions in D that contain the itemset. A **frequent itemset** is an itemset that satisfies a minimum support level, i.e., the support count of this frequent itemset is greater than or equal to the product of the minimum support and the total number of transactions in D. The number of transactions required for an itemset to satisfy minimum support is called the **minimum support count**. The set of frequent k-itemsets is commonly denoted as L_k.

In simple terms, the generation of association rules boils down to generation of frequent itemsets. Note, that this generation may be difficult if, for a given transactional data set, the number of items, m, is large.

Since we are interested only in strong association rules, i.e., those that satisfy minimum support and minimum confidence (user-defined parameters), we generate these only from frequent itemsets. Given an itemset, say, {Beer, Eggs}, we can generate four association rules:

$\Rightarrow$ Beer, Eggs
Beer $\Rightarrow$ Eggs
Eggs $\Rightarrow$ Beer
Eggs, Beer $\Rightarrow$

The examples below provide more insight into the process of generation of association rules from frequent itemsets. More details are given in Sec. 3.3.

Example: Given the transactions from Figure 10.6 and minimum support of 50%, the frequent itemset {Beer, Eggs, Celery} can be generated. Requiring minimum confidence of 60%, this itemset can be transformed into the following association rule:

Beer $\wedge$ Eggs $\Rightarrow$ Celery [50%, 66%]

The rule can be read as
IF customer buys Beer and Eggs THEN the probability of buying Celery is 66%.

Example: Given the following two itemsets:

{Beer, Eggs} with support 75%
{Beer, Celery, Eggs} with support 50%

we can generate the following rule:
IF customer buys Beer and Eggs THEN the probability of buying Celery is 66%.

3. Mining Single Dimensional, Single-Level Boolean Association Rules

The following four steps are used to generate single-dimensional association rules:

1. Prepare input data in the transactional format.
2. Choose items of interest, i.e., itemsets.
3. Compute support counts to evaluate whether selected itemsets are frequent, i.e., whether they satisfy minimum support.
4. Given the frequent itemsets, generate strong association rules that satisfy the minimum confidence by computing the corresponding conditional probabilities (counts).

Since the frequent itemsets used to generate the association rules satisfy the minimum support, the generated rules also satisfy the minimum support.

The computational performance (scalability) of an association-rule mining algorithm is determined by the second and third steps above. The remaining steps are much less computationally expensive. Therefore, the following discussion concentrates on these two steps.

3.1. The Naïve Algorithm

The simplest way to compute frequent itemsets is to consider all possible itemsets, compute their support, and check whether they are higher than the minimum support threshold. A naïve algorithm for generation of frequent itemsets (Steps 2 and 3 above) is shown in Figure 10.7.

Given that 2^m itemsets must be searched and n transactions must be scanned each time, this algorithm requires O $(2^m n)$ tests. This number grows exponentially with the number of items, and thus for larger problems the computations would take an unacceptably long time. Since 2^m is causing the problem, we need to find a way to reduce the number of tests. The itemsets that we can safely assume will not produce frequent itemsets do not need to be tested. This reasoning resulted in the development of the Apriori algorithm that is discussed next.

3.2. The Apriori Algorithm

The Apriori algorithm uses prior knowledge about an important property of frequent itemsets—hence its name. The **Apriori property** of an itemset says that all nonempty subsets of a frequent itemset must also be frequent. In other words, if a given itemset is not frequent (if it does not satisfy the minimum support threshold), then any superset of this itemset will also be not frequent, because it cannot occur more frequently than the original itemset. A proof follows:

Given n transactions, suppose A is a subset of i transactions (itemset).
If $A' \subset A$, then A' is a subset of $i' \leq i$ transactions.
Thus, if $i/n <$ minimum support, then i'/n is also $<$ minimum support.

```
n = |D|
for each subset s of |
  {
    counter = 0;
    for each transaction T in D
      {
        if s is a subset of T;
          counter = counter + 1; }
    if minimum support ≤ counter / n
        add s to frequent itemsets; }
```

Figure 10.7. Naïve algorithm for generation of frequent itemsets.

The simplest superset of an itemset is the itemset with one more added item. The Apriori property is an **antimonotone property**, i.e., if a set cannot satisfy a property, all of its supersets will also fail the same test.

The Apriori property is used to reduce the number of itemsets that must be searched to find frequent itemsets. The association-rule mining algorithm, the **Apriori algorithm**, performs the iterative search through itemsets, starting with 1-itemsets, through 2-itemsets, 3-itemsets, etc. In general, it finds and processes k-itemsets based on the exploration of (k–1)-itemsets. Using the Apriori property, the Apriori algorithm

– first finds all 1-itemsets
– next, finds among them a set of frequent 1-itemsets, L_1
– next extends L_1 to generate 2-itemsets
– next finds among these 2-itemsets a set of frequent 2-itemsets, L_2
– and repeats the process to obtain L_3, L_4, etc.

Based on the Apriori property, in each iteration, k-itemsets that do not satisfy the minimum support are removed and only the remaining k-itemsets are used to generate itemsets for the next, $k+1$, iteration. This process substantially reduces the number of itemsets that must be checked if they are frequent. The algorithm is shown in Figure 10.8.

The only unknown in implementing the Apriori algorithm is how to perform generation of C_k, which is a set of k-itemsets based on L_{k-1}. These k-itemsets are checked against the minimum support to derive L_k. The C_k is generated in two steps:

1. For each frequent itemset FI from L_{k-1}, find each item i that does not belong to FI, but belongs to some other frequent $(k-1)$-itemset in L_{k-1}. Add i to FI to create a k-itemset. Remove duplicate k-itemsets after all additions for all $(k-1)$-itemsets are finished.

   Example: Generation of frequent 2-itemsets from frequent 1-itemsets.
   Frequent 1-itemsets include $\{A\}$, $\{B\}$, $\{C\}$.
   The 2-itemsets generated based on Step 1 are $\{A, B\}$, $\{A, C\}$, $\{B, A\}$, $\{B, C\}$, $\{C, A\}$, and $\{C, B\}$. After elimination of duplicates, the following 2-itemsets are left: $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$.

2. If frequent (k–1)-itemsets from L_{k-1} have (k–2)-items in common, then create a k-itemset by adding the two different items to (k–2) common items.

   Example: Generation of frequent 4-itemsets from overlapping frequent 3-itemsets.
   For two 3-itemsets $\{A, B, C\}$ and $\{A, B, D\}$ the resulting 4-itemset is $\{A, B, C, D\}$.

```
n = |D|
L1 = {frequent 1-itemsets}
for (k = 2; Lk−1 is not empty; k++)
  {
  Ck is generated as k-itemset candidates from Lk−1;
  for each transaction T in D
    {
    Ct = subset(Ck, T);            // k-itemsets that are subsets of T
    for each k-itemset c in Ct
    c.count++; }
  Lk = {c in Ck such that c.count ≥ minimum support; }}
Result: the frequent itemsets are the union of all Lk
```

Figure 10.8. The Apriori algorithm for the generation of frequent itemsets.

3.3. Generating Association Rules from Frequent Itemsets

The last step of the four that are used to generate single-dimensional association rules is to generate association rules from frequent itemsets. The association-rule mining algorithm requires the generation of strong rules, i.e., those that satisfy both minimum confidence and minimum support. The minimum support level is guaranteed by using frequent itemsets, and thus we need only to (1) generate the rules and (2) prune those rules that do not satisfy the minimum confidence.

The confidence can be defined based on the corresponding support values as follows:

$$\text{confidence}(A \Rightarrow B) = \text{P}(B|A) = support\_count(A \cup B)/support\_count(A)$$

where $support\_count(A \cup B)$ is the number of transactions in D containing the itemset $A \cup B$, and $support\_count(A)$ is the number of transactions in D containing the itemset A.

Based on this formula, each frequent itemset FI is used to generate association rules in two steps:

1. Generate all nonempty subsets of items, Y, of FI
2. For each Y, output the rules "$\text{Y} \Rightarrow (\text{FI} - \text{Y})$" if support_count($FI$) / support_count($Y$) $\geq$ minimum confidence threshold.

To demonstrate the Apriori algorithm in action, we generate association rules from the transactional data given in Figure 10.1, as is shown in Figure 10.9.

3.4. Improving Efficiency of the Apriori Algorithm

The Apriori algorithm was further modified to improve its efficiency (computational complexity). Below we briefly explain the most important improvements.

- **Hashing** is used to reduce the size of the candidate k-itemsets, i.e., itemsets generated from frequent itemsets from iteration k–1, C_k, for $k > 1$. For instance, when scanning D to generate L_1 from the candidate 1-itemsets in C_1, we can at the same time generate all 2-itemsets for each transaction, hash (map) them into different buckets of the hash table structure, and increase the corresponding bucket counts. A 2-itemset whose corresponding bucket count is below the support threshold cannot be frequent, and thus we can remove it from the candidate set C_2. In this way, we reduce the number of candidate 2-itemsets that must be examined to obtain L_2.
- **Transaction removal** removes transactions that do not contain frequent itemsets. In general, if a transaction does not contain any frequent k-itemsets, it cannot contain any frequent $(k+1)$ itemsets, and thus it can be removed from the computation of any frequent t-itemsets, where $t > k$.
- **Data set partitioning** generates frequent itemsets based on the discovery of frequent itemsets in subsets (partition) of D. The method has two steps:

  1. Division of the transactions in D into s nonoverlapping subsets and the mining frequent itemsets in each subset. Given a minimum support threshold (*minimum_support*) for D, then the minimum itemset support for a subset equals *minimum_support*number_transactions_in_this_subset*. Based on this support count, all frequent itemsets (for all k) in each subset, referred to as **local frequent itemsets**, are found. A special data structure, which for each itemset records the TID of the transactions that contains the items in this itemset, is used to find all local frequent k-itemsets, for all $k = 1, 2, 3, \ldots$, in just one scan of D. The frequent local itemsets may or may not be frequent in D, but any itemset that is potentially frequent in D must be frequent in at least one subset. Therefore, local frequent itemsets from all subsets become candidate itemsets for D. The collection of all local frequent itemsets is referred to as **global candidate itemsets** with respect to D.

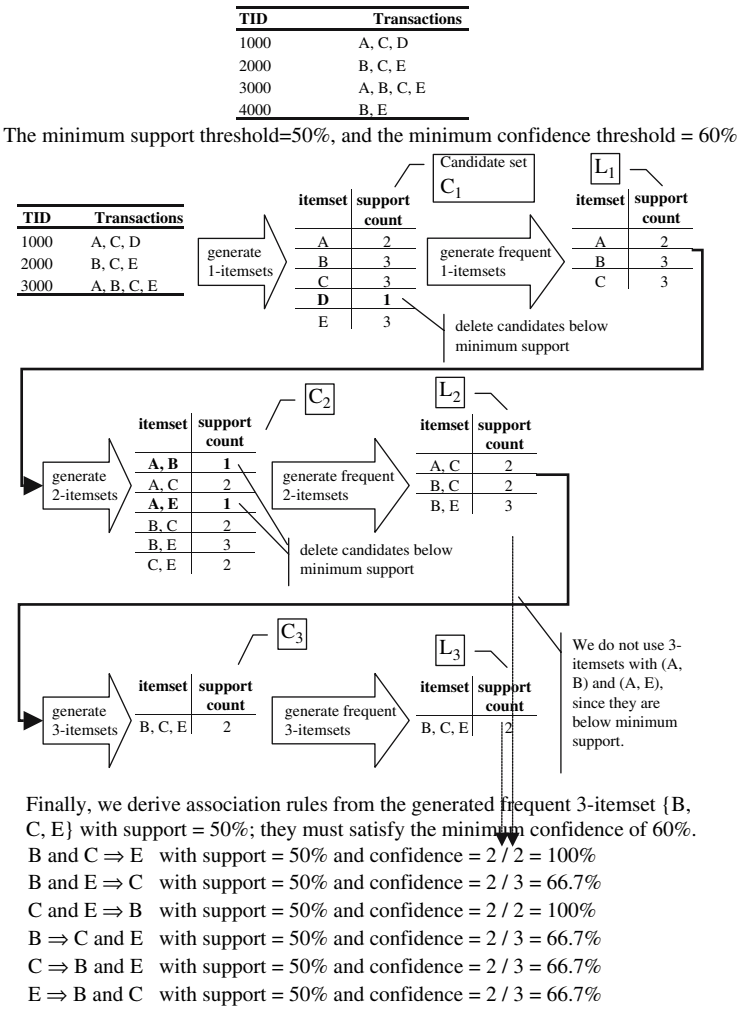

Figure 10.9. Example generation of association rules using the Apriori algorithm.

2. Computation of frequent itemsets for *D* based on the global candidate itemsets. One scan of D is performed to find out which of the global candidate itemsets satisfy the support threshold.

The size and number of subsets is usually set so that each of the subsets can fit into the main computer memory. Figure 10.10 illustrates the data set partitioning procedure.

– **Sampling** generates association rules based on a sampled subset of transactions in *D*. In this case, a randomly selected subset *S* of *D* is used to search for the frequent itemsets. The generation of frequent itemsets from *S* is more efficient (faster), but some of the rules that would have been generated from *D* may be missing, and some rules generated from *S* may not be present in *D*, i.e., the "accuracy" of the rules may be lower. Usually the size of *S* is selected so that the transactions can fit into the main memory, and thus only one scan of the data is required (no paging). To reduce the possibility that we will miss some of the frequent itemsets from *D* when generating frequent itemsets from *S*, we may use a lower support threshold for *S*

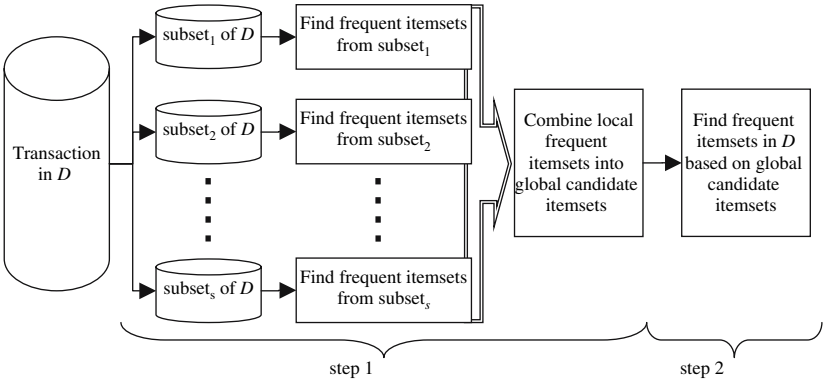

Figure 10.10. Generation of frequent itemsets using data set partitioning.

as compared with the support threshold for D. This approach is especially valuable when the association rules are computed on a very frequent basis.

– **Mining frequent itemsets without generation of candidate itemsets**. One of the main limiting aspects of the Apriori algorithm is that it can still generate very large number of candidate itemsets. For instance, for 10,000 1-itemsets, the Apriori algorithm generates approximately 10,000,000 candidate 2-itemsets and has to compute and store their occurrence frequencies. When a long frequent pattern is generated, say with 100 items, the Apriori algorithm generates as many as 2^{100} candidate itemsets. The other limiting aspect is that the Apriori algorithm may need to repeatedly scan the data set D to check frequencies of a large set of candidate itemsets—a process that is especially transparent when mining long itemsets, i.e., $n+1$ scans is required where n is the length of the longest itemset.

  To address these issues, a **divide-and-conquer** method, which decomposes the overall problem into a set of smaller tasks, is used. The method, referred to as **frequent-pattern growth** (FP-growth), compresses the set of frequent (individual) items from D into a **frequent pattern tree** (FP-tree). The tree preserves complete information about D with respect to the frequent pattern mining, is never larger than D, and reduces the irrelevant information, i.e., infrequent items are removed. The FP-tree, instead of D, is scanned to find frequent itemsets. Next, this compacted frequent-item-based data set is divided into a set of conditional data sets, each associated with one frequent item, and each of these conditional data sets is mined separately. A more detailed discussion of the FP-growth algorithms is beyond the scope of this textbook.

  Several studies that have considered the performance of frequent itemset generation indicate that the FP-growth algorithm is efficient and scalable for mining both short and long frequent itemsets, and is about an order of magnitude faster than the Apriori algorithm. The FP-growth algorithm is faster than another method known as the **tree-projection** algorithm, which recursively projects D into a tree of projected databases. Intuitively, this improvement is due to avoiding generation of candidate itemsets, using compact data structure and elimination of repeated scans of D. Instead of generation of candidate itemsets, the FP-tree is built and simple counts are computed.

3.5. Finding Interesting Association Rules

As suggested by the example given in Figure 10.9, and depending on the minimum support and confidence values, the user may generate a large number of rules to analyze and assess. The question then is how one can filter out the rules that are potentially the most interesting. First, whenever a rule is interesting (or not) it can be evaluated either objectively or subjectively.

The ultimate subjective user's evaluations cannot be quantified or anticipated; they are different for different users. That is why objective **interestingness measures**, based on the statistical information present in D, were developed in order to remove uninteresting rules before presenting them to the user.

The **subjective** evaluation of association rules often boils down to checking whether a given rule is unexpected (i.e., surprises the user) and actionable (i.e., the user can do something useful based on the rule). More specifically, the rules are categorized as

– **useful**, when they provide high-quality, actionable information, e.g., diapers $\Rightarrow$ beers
– **trivial**, when they are valid and supported by data, but useless since they confirm well-known facts, e.g., milk $\Rightarrow$ bread
– **inexplicable**, when they concern valid and new facts but cannot be utilized, e.g., grocery_store $\Rightarrow$ milk_is_sold_as_often_as_bread

In most cases, the confidence and support values associated with each rule are used as an **objective** measure to select the most interesting rules. Rules that have confidence and support values higher than other rules are preferred. Although this simple approach works in many cases, we will show that sometimes rules that have high confidence and support may be uninteresting and even misleading. Therefore, an additional quality measure is used. This measure is not used to generate the rules (support and confidence are sufficient for this purpose) but is helpful for selecting interesting rules.

Let us assume that a transactional data set concerning a grocery store contains milk and bread as the frequent items. The data show that on a given day 2,000 transactions were recorded and among these, in 1,200 transactions the customers bought milk, in 1,650 transactions the customers bought bread, and in 900 transactions the customers bough both milk and bread. Given a minimum support threshold of 40% and a minimum confidence threshold of 70%, the "milk $\Rightarrow$ bread [45%, 75%]" rule would be generated. On the other hand, due to low support and confidence values, the "milk $\Rightarrow$ not bread [15%, 25%]" rule would not be generated. At the same time, the latter rule is by far more "accurate," while the first may be misleading. This phenomenon is explained using the **contingency matrix** shown in Table 10.2, which provides corresponding support counts for all (four) combinations of the two Boolean items.

The probability of buying bread is 82.5%, while the confidence of milk $\Rightarrow$ bread is lower and equals 75%. In other words, bread and milk are negatively associated, i.e., buying one results in a decrease in buying the other. Obviously, using this rule would not be a wise decision. The confidence value is only an estimate of the conditional probability of itemset B (bread) given A (milk).

The alternative approach to evaluating the interestingness of association rules is to use measures based on correlation. For an $A \Rightarrow B$ rule, the itemset A is **independent** of the occurrence of the itemset B if $\mathrm{P}(A \cup B) = \mathrm{P}(A)\mathrm{P}(B)$. Otherwise, itemsets A and B are **dependent** and correlated as events. The **correlation measure** (also referred to as lift and interest), which is defined between

Table 10.2. Support count values for the milk $\Rightarrow$ bread example.

	milk	**not milk**	**total**
bread	900	750	1650
not bread	300	50	350
total	1200	800	2000

itemsets A and B (but can be easily extended to more than two itemsets), is defined as

$$\text{correlation}(A, B) = \frac{P(A \cup B)}{P(A)P(B)}$$

If the correlation value is less than 1, then the occurrence of A is **negatively correlated** (inhibits) the occurrence of B. If the value is greater than 1, then A and B are **positively correlated**, which means that the occurrence of one implies (promotes) the occurrence of the other. Finally, if the correlation equals 1, then A and B are **independent**, i.e., there is no correlation between these itemsets.

Based on the contingency matrix shown in Table 10.2, the correlation value for milk $\Rightarrow$ bread equals $0.45/(0.6^*0.825) = 0.45/0.495 = 0.91$. This result shows that milk and bread are negatively correlated, and thus the corresponding association rule should not be used. On the other hand, for the milk $\Rightarrow$ not bread rule, the correlation equals $0.15/(0.6^*0.175) = 0.15/0.105 = 1.43$. In this case, there is a relatively strong positive correlation between these two itemsets. This example demonstrates that the correlation cannot be successfully captured using support and confidence values.

4. Mining Other Types of Association Rules

The simplest form of an association rules is a Boolean, single-level, single-dimensional rule, which we described in Sec. 10.3 above. In what follows, we describe methods for mining multilevel, multidimensional, and quantitative association rules.

4.1. Multilevel Association Rules

In applications where items form a hierarchy, it may be difficult to find strong association rules at the low level of abstraction due to sparsity of data in the multidimensional space. Strong association rules usually can be found at the higher level of a hierarchy, but they often represent already known, commonsense knowledge. For instance, the milk $\Rightarrow$ bread rule is likely to have strong support, but it is trivial. At the same time, the rule skim_milk $\Rightarrow$ large_white_bread may be useful, but it may have weak support. The corresponding concept hierarchy is shown in Figure 10.11.

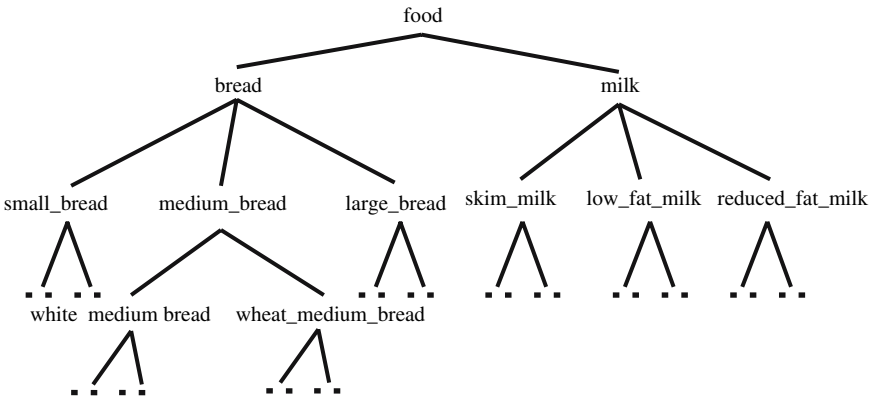

Figure 10.11. Example of a concept hierarchy concerning food.

An association mining algorithm should be able to generate and traverse between the rules at different levels of abstraction. **Multilevel association rules** are generated by performing a top-down, iterative deepening search. In simple terms, we first find strong rules at the high level(s) in the hierarchy, and then search for lower-level "weaker" rules. For instance, we first generate the milk $\Rightarrow$ bread rule and then concentrate on finding rules that concern breads of different sizes and milks with different fat content.

There are two main families of methods for multilevel association-rule mining:

– Methods based on **uniform support**, where the same minimum support threshold is used to generate rules at all levels of abstraction. In this case, the search for rules is simplified, since we can safely assume that itemsets containing item(s) whose ancestors (in the hierarchy) do not satisfy the minimum support are also not frequent. At the same time, it is very unlikely that items at the lower level of abstraction occur as frequently as those at the higher levels. Consequently, if the minimum support threshold is too high, the algorithm will miss potentially useful associations at lower levels of abstraction. On the other hand, if the threshold is too low, a very large number of potentially uninteresting associations would be generated at the higher levels.
– Methods based on **reduced support**, an approach that addresses the drawbacks of uniform support. In this case, each level of abstraction is furnished with its own minimum support threshold. The lower the abstraction level, the smaller the corresponding threshold. Figure 10.12 illustrates the difference between the uniform and reduced support methods.

If we assume a minimum support threshold value of 10% and use the uniform support method, then *milk* and *reduced_fat_milk* items are frequent and the *skim_milk* and *low_fat_milk* items are removed. In the case of the reduced support method, if the minimum support for the lower abstraction level is lowered by half, *milk*, *reduced_fat_milk*, and *low_fat_milk* items would all be considered frequent. There are three approaches to search for multilevel associations using the reduced support method:

– The **level-by-level independent** method, in which a breadth-first search is performed, i.e., each node in the hierarchy is examined, regardless of whether or not its parent node is found to be frequent.
– The **level-cross-filtering by single item** method, in which an item at a given level in the hierarchy is examined only if its parent at the preceding level is frequent. In this way, a more specific association is generated from a general one. For example, in Figure 10.12, if the minimum support threshold were set to 25%, then *reduced_fat_milk*, *low_fat_milk*, and *skim_milk* would not be considered.
– The **level-cross-filtering by *k*-itemset** method, in which a *k*-itemset at a given level is examined only if its parent *k*-itemset at the preceding level is frequent.

Once the multilevel association rules have been generated, some of them may be redundant due to the ancestor relationship between their items. For example, the following two rules are redundant:

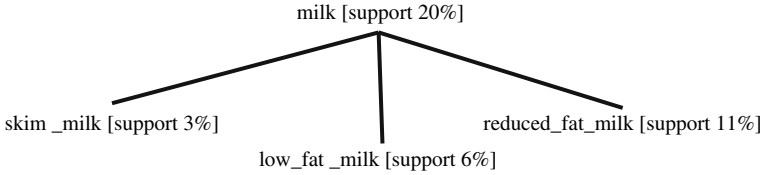

Figure 10.12. Multilevel association-rule mining using uniform and reduced support methods.

milk ⇒ large_bread [10%, 80%]
skim_milk ⇒ large_bread [2%, 75%]

The latter rule doe not provide useful information in the context of the former rule. The former rule is called **ancestor** of the latter rule, since it can be obtained by replacing an item in the latter rule by using their ancestor in the concept hierarchy. A rule is considered **redundant** if its support is similar to its "expected" value, based on the ancestor of the rule. Continuing our example with the two rules shown above, since the former rule has support of 10% and about one quarter of the sold *milk* is *skim_milk*, the expected support for the *skim_milk* is about 2.5%. The actual support for the latter rule is 2%, which means that the rule is redundant. As a consequence, only the former rule, which is more general, is kept.

4.2. Multidimensional and Quantitative Association Rules

The Boolean, single-dimensional association rules consider only one predicate and only the presence/absence of an item. In contrast, **multidimensional association rules** consider multiple predicates, and **quantitative association rules** consider items (attributes) that may assume a range of values instead of just two values.

The motivation for multidimensional association rules stems from data warehouses, which store multidimensional data. In this case, items correspond to features that describe different dimensions, and thus are associated with different predicates. An example in Sec. 2 showed a multidimensional rule that uses three predicates: *major*, *takes_course*, and *level*. Such a rule is referred to as an **interdimension association rule**, since none of the predicates is repeated. In contrast, the **hybrid-dimension association rule** incorporates some of the predicates multiple times (see the following example).

Example: An association rule that describes graduate students.

major(x, Computer Engineering) AND takes_course(x, Advanced Data Analysis and Decision Making) AND takes_course(x, Data Mining) ⇒ level(x, PhD) [0.9%, 90%]

Some of the data attributes are **discrete** (categorical) and **continuous**, in contrast to Boolean attributes, which are used in generic association mining. All attribute types can be categorized as **nominal**, in which case there is no ordering between their values, and **ordinal**, in which case some ordering exists. An example discrete ordinal attribute is *age*, while an example discrete nominal attribute is *major*. The continuous attributes must be preprocessed before being used for association mining. The preprocessing boils down to discretization, which divides the entire range of the attribute values into subintervals and associates a discrete value with each of the intervals. For instance, *age* can be divided into subintervals [0, 9], [9, 18], [18, 30], [30, 65], [65, 120], and these intervals can be associated with the following discrete values: *child*, *teenager*, *young_adult*, *adult*, *senior*. The discretization can be performed manually, based on an associated attribute hierarchy and/or an expert's knowledge. In the case of continuous ordinal attributes, discretization can also be performed automatically (see Chapter 8). A **quantitative association rule** uses discrete and/or discretized continuous items (attributes); see the example shown in Sec. 2.

The multidimensional and quantitative association rules can be generated using the same algorithms as the Boolean, single-dimensional rules. These include the Apriori algorithm and its modifications, such as hashing, partitioning, sampling, etc. The main difference is in how the input transactional data, is prepared. Namely, the continuous attributes must be discretized and the algorithm must search through all relevant attributes (attribute-value pairs; see Sec. 2.1) together, instead of searching just one attribute (predicate).

5. Summary and Bibliographical Notes

In this Chapter, we introduced **association rules** and **association rule mining** algorithms. The most important topics discussed in this Chapter are the following.

- Association rule mining is one of the two key **unsupervised** data mining methods that finds interesting **associations** (correlation) relationships within a large set of **items**.
- The items are stored using **transactions**.
- The association rules can be categorized as **single-dimensional** and **multidimensional**, **Boolean** and **quantitative**, and **single-level** and **multilevel**.
- The interestingness of association rules is measured using **support**, **confidence**, and **correlation**.
- The association rules are generated from **frequent itemsets**, which in turn are generated from transactions.
- The most popular algorithm for the generation of association rules is the **Apriori algorithm**.
- Several modifications to this algorithm have been proposed to improve its efficiency. These include **hashing**, **transaction removal**, **data set partitioning**, **sampling**, and **mining frequent itemsets without generation of candidate itemsets**.
- **Multilevel association rules** are generated using **uniform support**-based and **reduced support**-based methods.

A number of introductory-level textbooks provide material concerning **association rules** and **association rule mining**, such as [10, 12, 19]. Association rule mining was first proposed by Agrawal, Imielinski, and Swami [2], while the **Apriori algorithm** was introduced in [3, 4, 14]. The subsequent improvements to the Apriori algorithm, described in this Chapter, include **hashing** [15], **removal of transactions** [3, 8, 15], **partitioning** [16], **sampling** [20], and **mining itemsets without candidate generation** [9].

Multilevel association rule mining was introduced in [8, 17]. Mining for **quantitative association rules** includes approaches based on rule clustering [13], x-monotone regions [7], and partial completeness [18]. **Multidimensional association rules** were studied in [11]. Finally, the **interestingness** of the association rules was reported in [1, 5, 6].

References

1. Aggrawa, C., and Yu, P. 1999. A new framework for itemset generation. *Proceedings of the ACM Symposium on Principles of Database Systems*, Seattle, USA, 18–24
2. Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data*, Washington, USA, 207–216
3. Agrawal, R., and Srikant, R. 1994. Fast algorithm for mining association rules. *Proceedings of the 1994 International Conference on Very Large Databases*, Santiago, Chile, 487–499
4. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. 1996. Fast discovery of association rules. In Fayyad, U., Piatesky-Shapiro, G., Smyth, P., and Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*, AAAI Press/The MIT Press, Menlo Park, CA, 307–328
5. Brin, S., Motwani, R., and Silverstein, C. 1997. Beyond market basket: generalizing association rules to correlations. *Proceedings of the 1997 ACM-SIGMOD International Conference on Management of Data*, Tuscon, USA, 265–276
6. Chen, M., Han, J., and Yu, P. 1996. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, 8:866–883
7. Fukuda, T., Morimoto, Y., Morishita, S., and Tokuyama, T. 1996. Data mining using two-dimensional optimized association rules: scheme, algorithms and visualization. *Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data*, Montreal, Canada, 13–23
8. Han, J., and Fu, Y. 1995. Discovery of multiple-level association rules from large databases. *Proceedings of the 1995 International Conference on Very Large Databases*, Zurich, Swizerland, 420–431

9. Han, J., Pei, J., and Yin, Y. 2000. Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data*, Dallas, USA, 1–12
10. Han, K., and Kamber, M. 2001. *Data Mining: Concepts and Techniques*, Morgan Kaufmann, San Fransico, USA
11. Kamber, M., Han, J., and Chiang, J. 1997. Metarule-guided mining of multi-dimensional association rules using data cubes. *Proceedings of the 1997 International Conference on Knowledge Discovery and Data Mining*, Newport Beach, CA, USA, 207–210
12. Kantardzic, M. 2002. *Data Mining: Concepts, Models, Methods, and Algorithms*, Wiley-IEEE Press, Piscataway, NJ, USA
13. Lent, B., Swami, A., and Widom, J. 1997. Clustering association rules. *Proceedings of the 1997 International Conference on Data Engineering*, Birmingham, England, 220–231
14. Mannila, H., Toivonen, H., and Verkamo, A. 1994. Efficient algorithms for discovering association rules. *Proceedings of the AAAI'94 Workshop on Knowledge Discovery in Databases*, Seattle, Washington, USA, 181–192
15. Park, J., Chen, M., and Yu, P. 1995. An effective hash-based algorithm for mining association rules. *Proceedings of the 1995 ACM-SIGMOD International Conference on Management of Data*, San Jose, CA, USA, 175–186
16. Savasere, A., Omiecinski, E., and Navathe, S. 1995. An efficient algorithm for mining association rules in large databases. *Proceeding of the 1995 International Conference on Very Large Databases*, Zurich, Switzerland, 432–443
17. Srikant, R., and Agrawal, R. 1995. Mining generalized association rules. *Proceedings of the 1995 International Conference on Very Large Databases*, Zurich, Switzerland, 407–419
18. Srikant, R., and Agrawal, R. 1996. Mining quantitative association rules in large relational tables. *Proceedings of the 1996 ACM-SIGMOD International Conference on Management of Data*, Montreal, Canada, 1–12
19. Tan, P-N., Steinbach, M., and Kumar, V. 2005. *Introduction to Data Mining*, Pearson Addison Wesley
20. Toivonen, H. 1996. Sampling large databases for association rules. *Proceedings of the 1996 Conference on Very Large Databases*, Bombay, India, 134–145

6. Exercises

1. Generate frequent itemsets for the following transactional data. Assume that the minimum support threshold equals 40%.

TID	Transaction (basket)
1000	C, S, B, M
2000	E, R, D, B
3000	R, K, D, M
4000	C, R, B, D, M
5000	K, B, D, M
6000	R, B, D, M, S
7000	K, B, D, M

2. For the following transactional data, generate frequent itemsets and strong association rules. Assume a minimum support threshold equal to 33.3% and a minimum confidence threshold equal to 60%. Sort the resulting strong rules by their confidence and determine their corresponding support and confidence values.
3. The following transactional data include four transactions. Assuming that the minimum support threshold equals 60% and the minimum confidence threshold equals 80% find all frequent

TID	Transaction (basket)
1000	A, C, D
2000	A, D
3000	B, C, D
4000	B, C
5000	B, C
6000	B, C, D

TID	Transaction (basket)
1000	A, B, D, F
2000	A, B, C, D, E
3000	A, B, C, E
4000	A, B, D

itemsets. Generate all strong association rules of the form "x and $y \Rightarrow y$" where x, y, and z are the corresponding items, i.e., A, B, C, D, E, and F.

4. The number of generated association rules depends on the number of generated frequent itemsets. Assuming that, for a given transactional data set, the number of items, m, is large, estimate how many frequent itemsets can be generated from m items. Consider the case when all candidate itemsets satisfy the minimum support level and when, on average, half of the candidate itemsets satisfy the minimum support level.
5. Use WEKA software (available for free at http://www.cs.waikato.ac.nz/ml/weka/) to generate association rules. Run WEKA with the nominal data set *weather.nominal*, which is included with the software. You can initially use the default values of confidence and support, but you should also try several other combinations. Analyze and interpret the discovered rules.
6. Describe the FP-growth algorithm for mining association algorithms. Use transactions from Exercise 3 to demonstrate how it works. Briefly contrast this algorithm with a tree-projection algorithm.
7. Based on the contingency matrix below, a minimum support threshold equal to 30%, and a minimum confidence threshold equal to 60%, verify whether the "Beer $\Rightarrow$ Diapers" association rule is strong. Describe what kind of correlation (independent, positively correlated, negatively correlated) exists between *beer* and *diapers*.

	Beer	Not beer	Total
Diapers	200	50	250
Not diapers	100	150	250
Total	300	200	500

8. Come up with a simple example (different than the example used in the text) demonstrating that items that constitute a strong association rule can be negatively correlated.
9. Research the topic of *incremental association rules*. Create a bibliography of related literature and write a short paper that discusses the background, motivation, and related algorithms.