CHAPTER 6

Association Rules

6.1 INTRODUCTION

The purchasing of one product when another product is purchased represents an association rule. Association rules are frequently used by retail stores to assist in marketing, advertising, floor placement, and inventory control. Although they have direct applicability to retail businesses, they have been used for other purposes as well, including predicting faults in telecommunication networks. Association rules are used to show the relationships between data items. These uncovered relationships are not inherent in the data, as with functional dependencies, and they do not represent any sort of causality or correlation. Instead, association rules detect common usage of items. Example 6.1 illustrates this.

EXAMPLE 6.1

A grocery store chain keeps a record of weekly transactions where each transaction represents the items bought during one cash register transaction. The executives of the chain receive a summarized report of the transactions indicating what types of items have sold at what quantity. In addition, they periodically request information about what items are commonly purchased together. They find that 100% of the time that PeanutButter is purchased, so is Bread. In addition, 33.3% of the time PeanutButter is purchased, Jelly is also purchased. However, PeanutButter exists in only about 50% of the overall transactions.

A database in which an association rule is to be found is viewed as a set of tuples, where each tuple contains a set of items. For example, a tuple could be {PeanutButter, Bread, Jelly}, which consists of the three items: peanut butter, bread, and jelly. Keeping grocery story cash register transactions in mind, each item represents an item purchased, while each tuple is the list of items purchased at one time. In the simplest cases, we are not interested in quantity or cost, so these may be removed from the records before processing. Table 6.1 is used throughout this chapter to illustrate different algorithms. Here there are five transactions and five items: {Beer, Bread, Jelly, Milk, PeanutButter}. Throughout this chapter we list items in alphabetical order within a transaction. Although this is not required, algorithms often assume that this sorting is done in a preprocessing step.

The *support* of an item (or set of items) is the percentage of transactions in which that item (or items) occurs. Table 6.2 shows the support for all subsets of items from our total set. As seen, there is an exponential growth in the sets of items. In this case we could have 31 sets of items from the original set of five items (ignoring the empty set). This explosive growth in potential sets of items is an issue that most association

TABLE 6.1: Sample Data to Illustrate Association Rules

Transaction	Items
t_1	Bread, Jelly, PeanutButter
t_2	Bread, PeanutButter
t_3	Bread, Milk, PeanutButter
t_4	Beer, Bread
t_5	Beer, Milk

TABLE 6.2: Support of All Sets of Items Found in Table 6.1

Set	Support	Set	Support
Beer	40	Beer, Bread, Milk	0
Bread	80	Beer, Bread, PeanutButter	0
Jelly	20	Beer, Jelly, Milk	0
Milk	40	Beer, Jelly, PeanutButter	0
PeanutButter	60	Beer, Milk, PeanutButter	0
Beer, Bread	20	Bread, Jelly, Milk	0
Beer, Jelly	0	Bread, Jelly, PeanutButter	20
Beer, Milk	20	Bread, Milk, PeanutButter	20
Beer, PeanutButter	0	Jelly, Milk, PeanutButter	0
Bread, Jelly	20	Beer, Bread, Jelly, Milk	0
Bread, Milk	20	Beer, Bread, Jelly, PeanutButter	0
Bread, PeanutButter	60	Beer, Bread, Milk, PeanutButter	0
Jelly, Milk	0	Beer, Jelly, Milk, PeanutButter	0
Jelly, PeanutButter	20	Bread, Jelly, Milk, PeanutButter	0
Milk, PeanutButter	20	Beer, Bread, Jelly, Milk, PeanutButter	0
Beer, Bread, Jelly	0		

rule algorithms must contend with, as the conventional approach to generating association rules is in actuality counting the occurrence of sets of items in the transaction database.

Note that we are dealing with categorical data. Given a target domain, the underlying set of items usually is known, so that an encoding of the transactions could be performed before processing. As we will see, however, association rules can be applied to data domains other than categorical.

DEFINITION 6.1. Given a set of *items* $I = \{I_1, I_2, \ldots, I_m\}$ and a database of transactions $D = \{t_1, t_2, \ldots, t_n\}$ where $t_i = \{I_{i1}, I_{i2}, \ldots, I_{ik}\}$ and $I_{ij} \in I$, an **association rule** is an implication of the form $X \Rightarrow Y$ where $X, Y \subset I$ are sets of items called *itemsets* and $X \cap Y = \emptyset$.

DEFINITION 6.2. The **support (s)** for an association rule $X \Rightarrow Y$ is the percentage of transactions in the database that contain $X \cup Y$.

DEFINITION 6.3. The **confidence or strength** (α) for an association rule $X \Rightarrow Y$ is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X.

A formal definition, from [AIS93], is found in Definition 6.1. We generally are not interested in all implications but only those that are important. Here importance usually is measured by two features called *support* and *confidence* as defined in Definitions 6.2 and 6.3. Table 6.3 shows the support and confidence for several association rules, including those from Example 6.1.

The selection of association rules is based on these two values as described in the definition of the association rule problem in Definition 6.4. Confidence measures the strength of the rule, whereas support measures how often it should occur in the database. Typically, large confidence values and a smaller support are used. For example, look at Bread $\Rightarrow$ PeanutButter in Table 6.3. With $\alpha = 75\%$, this indicates that this rule holds 75% of the time that it could. That is, 3/4 times that Bread occurs, so does PeanutButter. This is a stronger rule than Jelly $\Rightarrow$ Milk because there are no times Milk is purchased when Jelly is bought. An advertising executive probably would not want to base an advertising campaign on the fact that when a person buys Jelly he also buys Milk. Lower values for support may be allowed as support indicates the percentage of time the rule occurs throughout the database. For example, with Jelly $\Rightarrow$ PeanutButter, the confidence is 100% but the support is only 20%. It may be the case that this association

TABLE 6.3: Support and Confidence for Some Association Rules

$X \Rightarrow Y$	s	α
Bread $\Rightarrow$ PeanutButter	60%	75%
PeanutButter $\Rightarrow$ Bread	60%	100%
Beer $\Rightarrow$ Bread	20%	50%
PeanutButter $\Rightarrow$ Jelly	20%	33.3%
Jelly $\Rightarrow$ PeanutButter	20%	100%
Jelly $\Rightarrow$ Milk	0%	0%

rule exists only in 20% of the transactions, but when the *antecedent* Jelly occurs, the *consequent* always occurs. Here an advertising strategy targeted to people who purchase Jelly would be appropriate.

The discussion so far has centered around the use of association rules in the *market basket* area. Example 6.2 illustrates a use for association rules in another domain: telecommunications. This example, although quite simplified from the similar real-world problem, illustrates the importance of association rules in other domains and the fact that support need not always be high.

EXAMPLE 6.2

A telephone company must ensure that a high percentage of all phone calls are made within a certain period of time. Since each phone call must be routed through many switches, it is imperative that each switch work correctly. The failure of any switch could result in a call not being completed or being completed in an unacceptably long period of time. In this environment, a potential data mining problem would be to predict a failure of a node. Then, when the node is predicted to fail, measures can be taken by the phone company to route all calls around the node and replace the switch. To this end, the company keeps a history of calls through a switch. Each call history indicates the success or failure of the switch, associated timing, and error indication. The history contains results of the last and prior traffic through the switch. A transaction of the type ⟨success, failure⟩ indicates that the most recent call could not be handled successfully, while the call before that was handled fine. Another transaction ⟨ERR1, failure⟩ indicates that the previous call was handled but an error occurred, ERR1. This error could be something like excessive time. The data mining problem can then be stated as finding association rules of the type $X \Rightarrow$ Failure. If these types of rules occur with a high confidence, we could predict failure and immediately take the node off-line. Even though the support might be low because the X condition does not frequently occur, most often when it occurs, the node fails with the next traffic.

DEFINITION 6.4. Given a set of *items* $I = \{I_1, I_2, \ldots, I_m\}$ and a database of transactions $D = \{t_1, t_2, \ldots, t_n\}$ where $t_i = \{I_{i1}, I_{i2}, \ldots, I_{ik}\}$ and $I_{ij} \in I$, the **association rule problem** is to identify all association rules $X \Rightarrow Y$ with a minimum support and confidence. These values (s, α) are given as input to the problem.

The efficiency of association rule algorithms usually is discussed with respect to the number of scans of the database that are required and the maximum number of itemsets that must be counted.

6.2 LARGE ITEMSETS

The most common approach to finding association rules is to break up the problem into two parts:

1. Find large itemsets as defined in Definition 6.5.
2. Generate rules from frequent itemsets.

An *itemset* is any subset of the set of all items, I.

Definition 6.5. A **large (frequent) itemset** is an itemset whose number of occurrences is above a threshold, s. We use the notation L to indicate the complete set of large itemsets and l to indicate a specific large itemset.

Once the large itemsets have been found, we know that any interesting association rule, $X \Rightarrow Y$, must have $X \cup Y$ in this set of frequent itemsets. Note that the subset of any large itemset is also large. Because of the large number of notations used in association rule algorithms, we summarize them in Table 6.4. When a specific term has a subscript, this indicates the size of the set being considered. For example, l_k is a large itemset of size k. Some algorithms divide the set of transactions into partitions. In this case, we use p to indicate the number of partitions and a superscript to indicate which partition. For example, D^i is the i^{th} partition of D.

Finding large itemsets generally is quite easy but very costly. The naive approach would be to count all itemsets that appear in any transaction. Given a set of items of size m, there are 2^m subsets. Since we are not interested in the empty set, the potential number of large itemsets is then $2^m - 1$. Because of the explosive growth of this number, the challenge of solving the association rule problem is often viewed as how to efficiently determine all large itemsets. (When $m = 5$, there are potentially 31 itemsets. When $m = 30$, this becomes 1073741823.) Most association rule algorithms are based on smart ways to reduce the number of itemsets to be counted. These potentially large itemsets are called *candidates*, and the set of all counted (potentially large) itemsets is the *candidate itemset* (c). One performance measure used for association rule algorithms is the size of C. Another problem to be solved by association rule algorithms is what data structure is to be used during the counting process. As we will see, several have been proposed. A trie or hash tree are common.

When all large itemsets are found, generating the association rules is straightforward. Algorithm 6.1, which is modified from [AS94], outlines this technique. In this algorithm we use a function *support*, which returns the support for the input itemset.

TABLE 6.4: Association Rule Notation

Term	Description
D	Database of transactions
t_i	Transaction in D
s	Support
α	Confidence
X, Y	Itemsets
$X \Rightarrow Y$	Association rule
L	Set of large itemsets
l	Large itemset in L
C	Set of candidate itemsets
p	Number of partitions

ALGORITHM 6.1

```
Input:
   $D$        //Database of transactions
   $I$        //Items
   $L$        //Large itemsets
   $s$        //Support
   $\alpha$        //Confidence
Output:
   $R$        //Association Rules satisfying $s$ and $\alpha$
ARGen algorithm:
   $R = \emptyset$;
   for each $l \in L$ do
      for each $x \subset l$ such that $x \neq \emptyset$ do
         if $\frac{\text{support}(l)}{\text{support}(x)} \geq \alpha$ then
            $R = R \cup \{x \Rightarrow (l - x)\}$;
```

To illustrate this algorithm, again refer to the data in Table 6.1 with associated supports shown in Table 6.2. Suppose that the input support and confidence are $s = 30\%$ and $\alpha = 50\%$, respectively. Using this value of s, we obtain the following set of large itemsets:

$$L = \{\{\text{Beer}\}, \{\text{Bread}\}, \{\text{Milk}\}, \{\text{PeanutButter}\}\{\text{Bread, PeanutButter}\}\}.$$

We now look at what association rules are generated from the last large itemset. Here $l = \{\text{Bread, PeanutButter}\}$. There are two nonempty subsets of l: {Bread} and {PeanutButter}. With the first one we see:

$$\frac{\text{support}(\{\text{Bread, PeanutButter}\})}{\text{support}(\{\text{Bread}\})} = \frac{60}{80} = 0.75$$

This means that the confidence of the association rule Bread $\Rightarrow$ PeanutButter is 75%, just as is seen in Table 6.3. Since this is above α, it is a valid association rule and is added to R. Likewise with the second large itemset

$$\frac{\text{support}(\{\text{Bread, PeanutButter}\})}{\text{support}(\{\text{PeanutButter}\})} = \frac{60}{60} = 1$$

This means that the confidence of the association rule PeanutButter $\Rightarrow$ Bread is 100%, and this is a valid association rule.

All of the algorithms discussed in subsequent sections look primarily at ways to efficiently discover large itemsets.

6.3 BASIC ALGORITHMS

6.3.1 Apriori Algorithm

The Apriori algorithm is the most well known association rule algorithm and is used in most commercial products. It uses the following property, which we call the *large itemset property*:

Any subset of a large itemset must be large.