4.5 MINING ASSOCIATION RULES

Association rules are like classification rules. You could find them in the same way, by executing a divide-and-conquer rule-induction procedure for each possible expression that could occur on the right side of the rule. However, not only might any attribute occur on the right side with any possible value, but a single association rule often predicts the value of more than one attribute. To find such rules, you would have to execute the rule-induction procedure once for every possible *combination* of attributes, with every possible combination of values, on the right side. That would result in an enormous number of association rules, which would then have to be pruned down on the basis of their *coverage* (the number of instances that they predict correctly) and their *accuracy* (the same number expressed as a proportion of the number of instances to which the rule applies). This approach is quite infeasible. (Note that, as we mentioned in Section 3.4, what we are calling *coverage* is often called *support* and what we are calling *accuracy* is often called *confidence*.)

Instead, we capitalize on the fact that we are only interested in association rules with high coverage. We ignore, for the moment, the distinction between the left and right sides of a rule and seek combinations of attribute–value pairs that have a prespecified minimum coverage. These are called *item sets*: An attribute–value pair is an *item*. The terminology derives from market basket analysis, in which the items are articles in your shopping cart and the supermarket manager is looking for associations among these purchases.

Item Sets

The first column of Table 4.10 shows the individual items for the weather data in Table 1.2 (page 10), with the number of times each item appears in the dataset given at the right. These are the one-item sets. The next step is to generate the two-item sets by making pairs of the one-item sets. Of course, there is no point in generating a set containing two different values of the same attribute (such as *outlook* = *sunny* and *outlook* = *overcast*) because that cannot occur in any actual instance.

Assume that we seek association rules with minimum coverage 2; thus, we discard any item sets that cover fewer than two instances. This leaves 47 two-item sets, some of which are shown in the second column along with the number of times they appear. The next step is to generate the three-item sets, of which 39 have a coverage of 2 or greater. There are six four-item sets, and no five-item sets—for this data, a five-item set with coverage 2 or greater could only correspond to a repeated instance. The first rows of the table, for example, show that there are five days when *outlook* = *sunny*, two of which have *temperature* = *hot*, and, in fact, on both of those days *humidity* = *high* and *play* = *no* as well.

Table 4.10 Item Sets for Weather Data with Coverage 2 or Greater

	One-Item Sets		Two-Item Sets		Three-Item Sets		Four-Item Sets	
1	outlook = sunny	5	outlook = sunny temperature = mild	2	outlook = sunny temperature = hot humidity = high	2	outlook = sunny temperature = hot humidity = high play = no	2
2	outlook = overcast	4	outlook = sunny temperature = hot	2	outlook = sunny temperature = hot play = no	2	outlook = sunny humidity = high windy = false play = no	2
3	outlook = rainy	5	outlook = sunny humidity = normal	2	outlook = sunny humidity = normal play = yes	2	outlook = overcast temperature = hot windy = false play = yes	2
4	temperature = cool	4	outlook = sunny humidity = high	3	outlook = sunny humidity = high windy = false	2	outlook = rainy temperature = mild windy = false play = yes	2
5	temperature = mild	6	outlook = sunny windy = true	2	outlook = sunny humidity = high play = no	3	outlook = rainy humidity = normal windy = false play = yes	2
6	temperature = hot	4	outlook = sunny windy = false	3	outlook = sunny windy = false play = no	2	temperature = cool humidity = normal windy = false play = yes	2
7	humidity = normal	7	outlook = sunny play = yes	2	outlook = overcast temperature = hot windy = false	2		
8	humidity = high	7	outlook = sunny play = no	3	outlook = overcast temperature = hot play = yes	2		

Continued

Table 4.10 Item Sets for Weather Data with Coverage or Greater (*Continued*)

	One-Item Sets		Two-Item Sets		Three-Item Sets		Four-Item Sets	
9	windy = true	6	outlook = overcast temperature = hot	2	outlook = overcast humidity = normal play = yes	2		
10	windy = false	8	outlook = overcast humidity = normal	2	outlook = overcast humidity = high play = yes	2		
11	play = yes	9	outlook = overcast humidity = high	2	outlook = overcast windy = true play = yes	2		
12	play = no	5	outlook = overcast windy = true	2	outlook = overcast windy = false play = yes	2		
13			outlook = overcast windy = false	2	outlook = rainy temperature = cool humidity = normal	2		
…			…		…			
38			humidity = normal windy = false	4	humidity = normal windy = false play = yes	4		
39			humidity = normal play = yes	6	humidity = high windy = false play = no	2		
40			humidity = high windy = true	3				
…			…					
47			windy = false play = no	2				

Association Rules

Shortly we will explain how to generate these item sets efficiently. But first let us finish the story. Once all item sets with the required coverage have been generated, the next step is to turn each into a rule, or a set of rules, with at least the specified minimum accuracy. Some item sets will produce more than one rule; others will produce none. For example, there is one three-item set with a coverage of 4 (row 38 of Table 4.10):

```
humidity = normal, windy = false, play = yes
```

This set leads to seven potential rules:

```
If humidity = normal and windy = false then play = yes     4/4
If humidity = normal and play = yes then windy = false     4/6
If windy = false and play = yes then humidity = normal     4/6
If humidity = normal then windy = false and play = yes     4/7
If windy = false then humidity = normal and play = yes     4/8
If play = yes then humidity = normal and windy = false     4/9
If – then humidity = normal and windy = false and play = yes    4/14
```

The figures at the right in this list show the number of instances for which all three conditions are true—that is, the coverage—divided by the number of instances for which the conditions in the antecedent are true. Interpreted as a fraction, they represent the proportion of instances on which the rule is correct—that is, its accuracy. Assuming that the minimum specified accuracy is 100%, only the first of these rules will make it into the final rule set. The denominators of the fractions are readily obtained by looking up the antecedent expression in Table 4.10 (although some are not shown in the table). The final rule above has no conditions in the antecedent, and its denominator is the total number of instances in the dataset.

Table 4.11 shows the final rule set for the weather data, with minimum coverage 2 and minimum accuracy 100%, sorted by coverage. There are 58 rules, 3 with coverage 4, 5 with coverage 3, and 50 with coverage 2. Only 7 have two conditions in the consequent, and none has more than two. The first rule comes from the item set described previously. Sometimes several rules arise from the same item set. For example, rules 9, 10, and 11 all arise from the four-item set in row 6 of Table 4.10:

```
temperature = cool, humidity = normal, windy = false, play = yes
```

which has coverage 2. Three subsets of this item set also have coverage 2:

```
temperature = cool, windy = false
temperature = cool, humidity = normal, windy = false
temperature = cool, windy = false, play = yes
```

and these lead to rules 9, 10, and 11, all of which are 100% accurate (on the training data).

Table 4.11 Association Rules for Weather Data

	Association Rule	Coverage	Accuracy
1	humidity = normal windy = false ⇒ play = yes	4	100%
2	temperature = cool ⇒ humidity = normal	4	100%
3	outlook = overcast ⇒ play = yes	4	100%
4	temperature = cool play = yes ⇒ humidity = normal	3	100%
5	outlook = rainy windy = false ⇒ play = yes	3	100%
6	outlook = rainy play = yes ⇒ windy = false	3	100%
7	outlook = sunny humidity = high ⇒ play = no	3	100%
8	outlook = sunny play = no ⇒ humidity = high	3	100%
9	temperature = cool windy = false ⇒ humidity = normal play = yes	2	100%
10	temperature = cool humidity = normal windy = false ⇒ play = yes	2	100%
11	temperature = cool windy = false play = yes ⇒ humidity = normal	2	100%
12	outlook = rainy humidity = normal windy = false ⇒ play = yes	2	100%
13	outlook = rainy humidity = normal play = yes ⇒ windy = false	2	100%
14	outlook = rainy temperature = mild windy = false ⇒ play = yes	2	100%
15	outlook = rainy temperature = mild play = yes ⇒ windy = false	2	100%
16	temperature = mild windy = false play = yes ⇒ outlook = rainy	2	100%
17	outlook = overcast temperature = hot ⇒ windy = false play = yes	2	100%
18	outlook = overcast windy = false ⇒ temperature = hot play = yes	2	100%

Table 4.11 *Continued*

	Association Rule	Coverage	Accuracy
19	temperature = hot play = yes ⇒ outlook = overcast windy = false	2	100%
20	outlook = overcast temperature = hot windy = false ⇒ play = yes	2	100%
21	outlook = overcast temperature = hot play = yes ⇒ windy = false	2	100%
22	outlook = overcast windy = false play = yes ⇒ temperature = hot	2	100%
23	temperature = hot windy = false play = yes ⇒ outlook = overcast	2	100%
24	windy = false play = no ⇒ outlook = sunny humidity = high	2	100%
25	outlook = sunny humidity = high windy = false ⇒ play = no	2	100%
26	outlook = sunny windy = false play = no ⇒ humidity = high	2	100%
27	humidity = high windy = false play = no ⇒ outlook = sunny	2	100%
28	outlook = sunny temperature = hot ⇒ humidity = high play = no	2	100%
29	temperature = hot play = no ⇒ outlook = sunny humidity = high	2	100%
30	outlook = sunny temperature = hot humidity = high ⇒ play = no	2	100%
31	outlook = sunny temperature = hot play = no ⇒ humidity = high	2	100%
…	…	…	…
58	outlook = sunny temperature = hot ⇒ humidity = high	2	100%

Generating Rules Efficiently

We now consider in more detail an algorithm for producing association rules with specified minimum coverage and accuracy. There are two stages: generating item sets with the specified minimum coverage, and from each item set determining the rules that have the specified minimum accuracy.

The first stage proceeds by generating all one-item sets with the given minimum coverage (the first column of Table 4.10) and then using this to generate the two-item sets (second column), three-item sets (third column), and so on. Each operation involves a pass through the dataset to count the items in each set, and after the pass the surviving item sets are stored in a hash table—a standard data structure that allows elements stored in it to be found very quickly. From the one-item sets, candidate two-item sets are generated, and then a pass is made through the dataset, counting the coverage of each two-item set; at the end the candidate sets with less than minimum coverage are removed from the table. The candidate two-item sets are simply all of the one-item sets taken in pairs, because a two-item set cannot have the minimum coverage unless both its constituent one-item sets have the minimum coverage, too. This applies in general: A three-item set can only have the minimum coverage if all three of its two-item subsets have minimum coverage as well, and similarly for four-item sets.

An example will help to explain how candidate item sets are generated. Suppose there are five three-item sets—(A B C), (A B D), (A C D), (A C E), and (B C D)—where, for example, A is a feature such as *outlook* = *sunny*. The union of the first two, (A B C D), is a candidate four-item set because its other three-item subsets (A C D) and (B C D) have greater than minimum coverage. If the three-item sets are sorted into lexical order, as they are in this list, then we need only consider pairs with the same first two members. For example, we do not consider (A C D) and (B C D) because (A B C D) can also be generated from (A B C) and (A B D), and if these two are not candidate three-item sets, then (A B C D) cannot be a candidate four-item set. This leaves the pairs (A B C) and (A B D), which we have already explained, and (A C D) and (A C E). This second pair leads to the set (A C D E) whose three-item subsets do not all have the minimum coverage, so it is discarded. The hash table assists with this check: We simply remove each item from the set in turn and check that the remaining three-item set is indeed present in the hash table. Thus, in this example there is only one candidate four-item set, (A B C D). Whether or not it actually has minimum coverage can only be determined by checking the instances in the dataset.

The second stage of the procedure takes each item set and generates rules from it, checking that they have the specified minimum accuracy. If only rules with a single test on the right side were sought, it would be simply a matter of considering each condition in turn as the consequent of the rule, deleting it from the item set, and dividing the coverage of the entire item set by the coverage of the resulting subset—obtained from the hash table—to yield the accuracy of the corresponding rule. Given that we are also interested in association rules with multiple tests in the

consequent, it looks like we have to evaluate the effect of placing each *subset* of the item set on the right side, leaving the remainder of the set as the antecedent.

This brute-force method will be excessively computation intensive unless item sets are small, because the number of possible subsets grows exponentially with the size of the item set. However, there is a better way. We observed when describing association rules in Section 3.4 that if the double-consequent rule

```
If windy = false and play = no
   then outlook = sunny and humidity = high
```

holds with a given minimum coverage and accuracy, then both single-consequent rules formed from the same item set must also hold:

```
If humidity = high and windy = false and play = no
   then outlook = sunny
If outlook = sunny and windy = false and play = no
   then humidity = high
```

Conversely, if one or other of the single-consequent rules does not hold, there is no point in considering the double-consequent one. This gives a way of building up from single-consequent rules to candidate double-consequent ones, from double-consequent rules to candidate triple-consequent ones, and so on. Of course, each candidate rule must be checked against the hash table to see if it really does have more than the specified minimum accuracy. But this generally involves checking far fewer rules than the brute-force method. It is interesting that this way of building up candidate $(n + 1)$-consequent rules from actual n-consequent ones is really just the same as building up candidate $(n + 1)$-item sets from actual n-item sets, described earlier.

Discussion

Association rules are often sought for very large datasets, and efficient algorithms are highly valued. The method we have described makes one pass through the dataset for each different size of item set. Sometimes the dataset is too large to read in to main memory and must be kept on disk; then it may be worth reducing the number of passes by checking item sets of two consecutive sizes at the same time. For example, once sets with two items have been generated, all sets of three items could be generated from them before going through the instance set to count the actual number of items in the sets. More three-item sets than necessary would be considered, but the number of passes through the entire dataset would be reduced.

In practice, the amount of computation needed to generate association rules depends critically on the minimum coverage specified. The accuracy has less influence because it does not affect the number of passes that must be made through the dataset. In many situations we would like to obtain a certain number of rules—say 50—with the greatest possible coverage at a prespecified minimum accuracy level. One way to do this is to begin by specifying the coverage to be rather high and to

then successively reduce it, reexecuting the entire rule-finding algorithm for each of the coverage values and repeating until the desired number of rules has been generated.

The tabular input format that we use throughout this book, and in particular the standard ARFF format based on it, is very inefficient for many association-rule problems. Association rules are often used in situations where attributes are binary—either present or absent—and most of the attribute values associated with a given instance are absent. This is a case for the sparse data representation described in Section 2.4; the same algorithm for finding association rules applies.