Chapter 10

Association Analysis

Many years ago, a number of new Internet businesses were created to sell books on-line. Over time, they collected information about the books that each of their customers were buying. Using association analysis, they were able to identify groups of books that customers with similar interests seem to have been buying. Using this information, they were able to develop recommendation systems that informed their customers that other customers who purchased some book of interest also purchased other related books. The customer would often find such recommendations quite useful.

Association analysis identifies relationships or correlations between observations and/or between variables in our datasets. These relationships are then expressed as a collection of so-called association rules. The approach has been particularly successful in mining very large transactional databases, like shopping baskets and on-line customer purchases. Association analysis is one of the core techniques of data mining.

For the on-line bookselling example, historic data is used to identify, for example, that customers who purchased two particular books also tended to purchase another particular book. The historic data might indicate that the first two books are purchased by only 0.5% of all customers. But 70% of these then also purchase the third book. This is an *interesting* group of customers. As a business, we will take advantage of this observation by targeting advertising of the third book to those customers who have purchased both of the other books.

The usual type of input data for association analysis consists of transactional data, such as the items in a shopping basket at a supermarket, books and videos purchased by a single client, or medical treatments and tests received by patients. We are interested in the items whose co-occurrence within a transaction is of interest.

This short chapter introduces the basic concepts of association analysis and how to perform it in both Rattle and R.

10.1 Knowledge Representation

A representation of association rules is required to identify relationships between items within transactions. Suppose each transaction is thought of as a basket of items (which we might represent as $\{A, B, C, D, E, F\}$). The aim is to identify collections of items that appear together in multiple baskets (e.g., perhaps the items $\{A, C, F\}$ appear together in quite a few shopping baskets). From these so called *itemsets* (i.e., sets of items) we identify rules like $A, F \Rightarrow C$ that tell us that when A and F appear in a transaction (e.g., a shopping basket) then typically so does C.

A collection of association rules then represents a model as the outcome of association analysis. The general format of an association rule is

$$\mathcal{A} \rightarrow \mathcal{C}.$$

Both $\mathcal{A}$ (the left hand side or antecedent) and $\mathcal{C}$ (the right side or consequent) are sets of items. Generally we think of items as being particular books, for example, or particular grocery items. Examples might be:

$$milk \rightarrow bread,$$

$$beer \ \& \ nuts \rightarrow potato\ crisps,$$

$$Shrek1 \rightarrow Shrek2 \ \& \ Shrek3.$$

The concept of an item can be generalised to a specific variable/value combination as the item. The concept of association analysis can then be applied to many different datasets. Using our *weather* dataset, for example, this representation will lead to association rules like

$$WindDir3pm = NNW \rightarrow RainToday = No.$$

10.2 Search Heuristic

The basis of an association analysis algorithm is the generation of frequent itemsets. A frequent itemset is a set of items that occur together frequently enough to be considered as a candidate for generating association rules.

The obvious approaches to identifying itemsets that appear frequently enough in the data are quite expensive computationally, even with moderately sized datasets. The apriori algorithm takes advantage of the simple observation that all subsets of a frequent itemset must also be frequent. That is, if $\{milk, bread, cheese\}$ is a frequent itemset then so must each of the smaller itemsets, $\{milk, bread\}$, $\{milk, cheese\}$, $\{bread, cheese\}$, $\{milk\}$, $\{bread\}$, and $\{cheese\}$. This observation allows the algorithm to consider a significantly reduced search space by starting with frequent individual items. This first step eliminates very rare items. We then combine the remaining single items into itemsets containing just two items and retain only those that are frequent enough and similarly for itemsets containing three items and so on.

The concept of *frequent enough* is a parameter of the algorithm used to control the number of association rules discovered. This is called the *support* and specifies how frequently the items must appear in the whole dataset before they can be considered as a candidate association rule. For example, the user may choose to consider only sets of items that occur in at least 5% of all transactions.

The second phase of the algorithm considers each of the frequent itemsets and for each generates all possible combinations of association rules. Thus, for an itemset containing three items $\{milk, bread, cheese\}$, the following are among the possible association rules that will be considered:

$$bread \ \& \ milk \rightarrow cheese,$$

$$milk \rightarrow bread \ \& \ cheese,$$

$$cheese \ \& \ milk \rightarrow bread,$$

and so on.

The actual association rules that we retain are those that meet a criterion called *confidence*. The confidence calculates the proportion of transactions containing $\mathcal{A}$ that also contain $\mathcal{C}$. The confidence specifies a minimal probability for the association rule. For example, the user may choose to generate only rules that are true at least 90% of the time (that is, when $\mathcal{A}$ appears in the basket, $\mathcal{C}$ also appears in the same basket at least 90% of the time).

The apriori algorithm is a breadth-first or generate-and-test type of search algorithm. Only after exploring all of the possibilities of associations containing k items does it then consider those containing $k+1$ items. For each k, all candidates are tested to determine whether they have enough support.

In summary, the algorithm uses a simple two-phase generate-and-merge process. In phase 1, we generate frequent itemsets of size k (iterating from 1 until we have no frequent k-itemsets) and then combine them to generate candidate frequent itemsets of size $k+1$. In phase 2, we build candidate association rules.

10.3 Measures

The two primary measures used in association analysis are the **support** and the **confidence**. The minimum support is expressed as a percentage of the total number of transactions in the dataset. Informally, it is simply how often the items appear together from amongst all of the transactions. Formally, we define *support* for a collection of items $\mathcal{I}$ as the proportion of all transactions in which all items in $\mathcal{I}$ appear and express the support for an association rule as

$$support(\mathcal{A} \rightarrow \mathcal{C}) = P(\mathcal{A} \cup \mathcal{C}).$$

Typically, we use small values for the support, since overall the items that appear together “frequently” enough that are of interest generally won’t be the obvious ones that regularly appear together.

The minimum confidence is also expressed as the proportion of the total number of transactions in the dataset. Informally, it is a measure of how often the items $\mathcal{C}$ appear whenever the items $\mathcal{A}$ appear in a transaction. Formally, it is a conditional probability:

$$confidence(\mathcal{A} \rightarrow \mathcal{C}) = P(\mathcal{C}|\mathcal{A}) = P(\mathcal{A} \cup \mathcal{C})/P(\mathcal{A}).$$

It can also be expressed in terms of the *support*:

$$confidence(\mathcal{A} \rightarrow \mathcal{C}) = support(\mathcal{A} \rightarrow \mathcal{C})/support(\mathcal{A}).$$

Typically, this measure will have larger values since we are looking for the association rules that are quite strong, so that if we find the items in $\mathcal{A}$ in a transaction then there is quite a good chance of also finding $\mathcal{C}$ in the transaction.

There are a collection of other measures that are used with association rule analysis. One that is used in R and hence Rattle is the **lift**. The lift is the increased likelihood of $\mathcal{C}$ being in a transaction if $\mathcal{A}$ is included in the transaction. It is calculated as

$$lift(\mathcal{A} \rightarrow \mathcal{C}) = confidence(\mathcal{A} \rightarrow \mathcal{C})/support(\mathcal{C}).$$

Another measure is the **leverage**, which captures the fact that a higher frequency of $\mathcal{A}$ and $\mathcal{C}$ with a lower lift may be interesting:

$$leverage(\mathcal{A} \rightarrow \mathcal{C}) = support(\mathcal{A} \rightarrow \mathcal{C}) - support(\mathcal{A}) * support(\mathcal{C}).$$

10.4 Tutorial Example

Two types of association rules were identified above, corresponding to the type of data made available. The simplest case, known as market basket analysis, is when we have a transaction dataset that records just a transaction identifier. The identifier might identify a single shopping basket containing multiple items from shopping or a particular customer or patient and their associated purchases or medical treatments over time. A simple example of a market basket dataset might record the purchases of DVDs by customers (three customers in this case):

```
ID,Item
1,Sixth Sense
1,LOTR1
1,Harry Potter1
1,Green Mile
1,LOTR2
2,Gladiator
2,Patriot
2,Braveheart
3,LOTR1
3,LOTR2
```

The resulting model will then be a collection of association rules that might include

$$LOTR1 \rightarrow LOTR2.$$

The second form of association rule uses a dataset that we are more familiar with. This approach treats each observation as a transaction and the variables as the items in the "shopping basket." Considering the *weather* dataset, we might obtain models that include rules of the form

$$Humidity3pm = High \ \& \ Pressure3pm = Low \rightarrow RainToday = Yes.$$

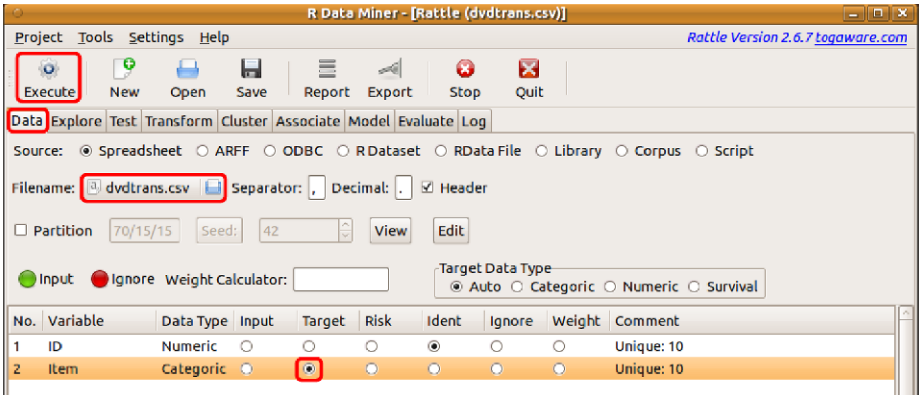
Both forms are supported in Rattle and R.

Building a Model Using Rattle

Rattle builds association rule models through the Associate tab. The two types of association rules are supported and the appropriate type is chosen using the Baskets check button. If the button is checked then Rattle will use the Ident and Target variables for the analysis, performing a market basket analysis. If the button is not checked, then Rattle will use the Input variables for a rules analysis.

For a basket analysis, the data is thought of as representing shopping baskets (or any other type of collection of items, such as a basket of medical tests, a basket of medicines prescribed to a patient, a basket of stocks held by an investor, and so on). Each basket has a unique identifier, and the variable specified as an Ident variable on the Data tab is taken as the identifier of a shopping basket. The contents of the basket are then the items contained in the column of data identified as the Target variable. For market basket analysis, these are the only two variables used.

To illustrate market basket analysis with Rattle, we can use a very simple and trivial dataset consisting of the DVD movies purchased by customers. The data is available as a CSV file (named `dvdtrans.csv`) from the Rattle package. The simplest way to load this dataset into Rattle is to first load the default sample *weather* dataset from the `weather.csv` file into Rattle. We do this by clicking the Execute button on starting Rattle. Then click the Filename button (which will now be showing `weather.csv`) to list the contents of Rattle's sample CSV folder. Choose `dvdtrans.csv` and click Open and then Execute. The `ID` variable will automatically be chosen as the Ident, but we will need to change the role of `Item` to be Target, as in Figure 10.1.


R Data Miner - [Rattle (dvdtrans.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
View Edit
Input Ignore

Source: [✓ Spreadsheet] ARFF ODBC R Dataset RData File Library Corpus Script

Filename: dvdtrans.csv Separator: , Decimal: . [✓ Header]

Partition 70/15/15 Seed: 42

Weight Calculator:

Target Data Type: [✓ Auto] Categoric Numeric Survival

No.	Variable	Data Type	Input	Target	Risk	Ident	Ignore	Weight	Comment
1	ID	Numeric	○	○	○	●	○	○	Unique: 10
2	Item	Categoric	○	●	○	○	○	○	Unique: 10

Figure 10.1: Choose the `dvdtrans.csv` file and load it into Rattle with a click of the Execute button. Then set the role for `Item` to be Target and click Execute for the new role to be noted.

On the Associate tab, ensure that the Baskets button is checked. Click the Execute button to build a model that will consist of a collection of association rules. Figure 10.2 shows the resulting text view, which we now review.

The first few lines of the text view list the number of association rules that make up the model. In our example, there are 127 rules:

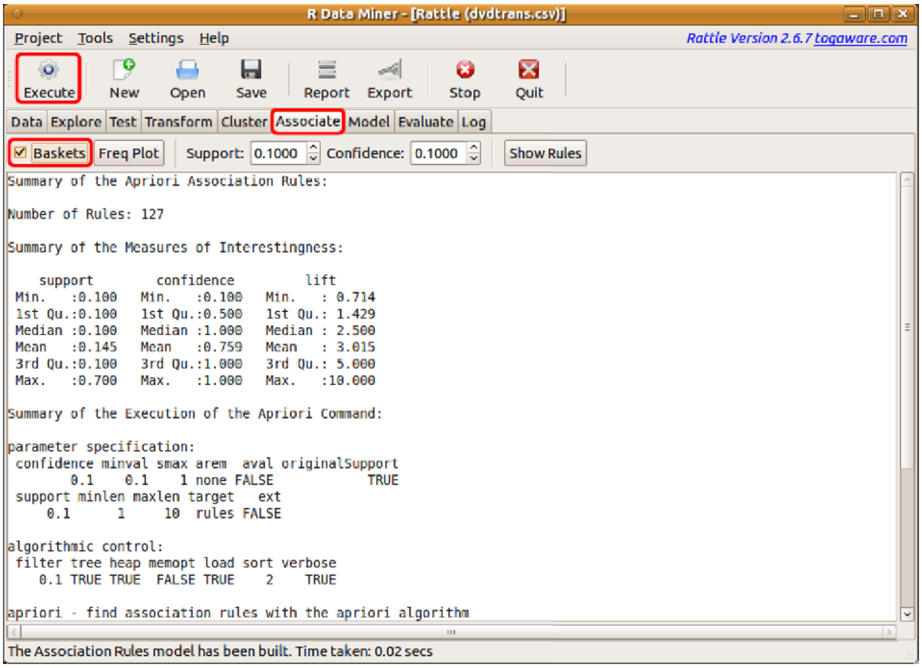
```
Summary of the Apriori Association Rules:

Number of Rules: 127
```

The next code block reports on the distribution of the three measures as found for the 127 rules of the model:

```
Summary of the Measures of Interestingness:

    support          confidence          lift
 Min.   :0.100    Min.   :0.100    Min.   : 0.714
 1st Qu.:0.100    1st Qu.:0.500    1st Qu.: 1.429
 Median :0.100    Median :1.000    Median : 2.500
 Mean   :0.145    Mean   :0.759    Mean   : 3.015
 3rd Qu.:0.100    3rd Qu.:1.000    3rd Qu.: 5.000
 Max.   :0.700    Max.   :1.000    Max.   :10.000
```


R Data Miner - [Rattle (dvdtrans.csv)]
Project Tools Settings Help
Rattle Version 2.6.7 togaware.com
Execute
New
Open
Save
Report
Export
Stop
Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
Baskets
Freq Plot
Show Rules

Support: 0.1000 Confidence: 0.1000

```
Summary of the Apriori Association Rules:

Number of Rules: 127

Summary of the Measures of Interestingness:

    support          confidence          lift
 Min.    :0.100   Min.    :0.100   Min.    : 0.714
 1st Qu.:0.100   1st Qu.:0.500   1st Qu.: 1.429
 Median :0.100   Median :1.000   Median : 2.500
 Mean    :0.145   Mean    :0.759   Mean    : 3.015
 3rd Qu.:0.100   3rd Qu.:1.000   3rd Qu.: 5.000
 Max.    :0.700   Max.    :1.000   Max.    :10.000

Summary of the Execution of the Apriori Command:

parameter specification:
 confidence minval smax arem  aval originalSupport
        0.1    0.1    1 none FALSE            TRUE
 support minlen maxlen target   ext
     0.1      1     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
```

The Association Rules model has been built. Time taken: 0.02 secs

Figure 10.2: Building an association rules model.

The 127 association rules met the criteria of having a minimum support of 0.1 and a minimum confidence of 0.1. Across the rules the support ranges from 0.1 up to 0.4. Confidence ranges from 0.1 up to 1.0 and lift from 0.83 up to 10.0.

This section is followed by a summary of the process of building the model. It begins with a review of the options supplied or the default values for the various parameters. We can see `confidence=` and `support=` listed:

```
Summary of the Execution of the Apriori Command:

parameter specification:
 confidence minval smax arem  aval originalSupport
        0.1    0.1    1 none FALSE            TRUE
 support minlen maxlen target   ext
     0.1      1     10  rules FALSE
```

These options are tunable through the Rattle interface. Others can be tuned directly through R. A set of parameters that control how the algorithm itself operates is then displayed:

```
algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE
```

The final section includes detailed information about the algorithm and the model that has been built:

```
apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09) (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[10 item(s), 10 trans] done [0.00s].
sorting and recoding items ... [10 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 done [0.00s].
writing ... [127 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

The Show Rules button will show all of the association rules for the model in the text view window, sorted by the level of confidence in the rule. The top five rules will be:

```
  lhs                  rhs               supp conf  lift
1 {Harry Potter2} => {Harry Potter1}  0.1    1 5.000
2 {Braveheart}    => {Patriot}        0.1    1 1.667
3 {Braveheart}    => {Gladiator}      0.1    1 1.429
4 {LOTR}          => {Green Mile}     0.1    1 5.000
5 {LOTR}          => {Sixth Sense}    0.1    1 1.667
```

These rules have only a single item on each side of the arrow, and all have a support of 0.1 and a confidence of 1. We can see that for either of the first two movies there is quite a large lift obtained.

Building a Model Using R

Arules (Hahsler et al., 2011) provides `apriori()` for R. The package provides an interface to the widely used, and freely available, apriori software from Christian Borgelt. This software was, for example, commercially licensed for use in the Clementine[1] data mining package and is a well-developed and respected implementation.

When loading a dataset to process with `apriori()`, it needs to be converted into a transaction data structure. Consider a dataset with two columns, one being the identifier of the "basket" and the other being an item contained in the basket, as is the case for the `dvdtrans.csv` data. We can load that data into R:

```
> library(arules)
> library(rattle)
> dvdtrans <- read.csv(system.file("csv", "dvdtrans.csv",
                                   package="rattle"))
> dvdDS <- new.env()
> dvdDS$data <- as(split(dvdtrans$Item, dvdtrans$ID),
                   "transactions")
> dvdDS$data

transactions in sparse format with
 10 transactions (rows) and
 10 items (columns)
```

We can then build the model using this transformed dataset:

```
> dvdAPRIORI <- new.env(parent=dvdDS)
> evalq({
    model <- apriori(data, parameter=list(support=0.2,
                                          confidence=0.1))
  }, dvdAPRIORI)
```

The rules can be extracted and ordered by confidence using `inspect()`. In the following code block we also use `[1:5]` to limit the display to just the first five association rules. We notice that the first two are symmetric, which is expected since everyone who purchases one of these movies always also purchases the other.

[1]Clementine became an SPSS product and was then purchased by IBM to become IBM SPSS Modeler.

```
> inspect(sort(dvdAPRIORI$model, by="confidence")[1:5])

  lhs                rhs             support confidence  lift
1 {LOTR1}         => {LOTR2}             0.2          1 5.000
2 {LOTR2}         => {LOTR1}             0.2          1 5.000
3 {Green Mile}    => {Sixth Sense}       0.2          1 1.667
4 {Patriot}       => {Gladiator}         0.6          1 1.429
5 {Patriot,
   Sixth Sense} => {Gladiator}           0.4          1 1.429
```

10.5 Command Summary

This chapter has referenced the following R packages, commands, functions, and datasets:

`apriori()`	function	Build an association rule model.
arules	package	Support for association rules.
`inspect()`	function	Display results of model building.
weather	dataset	Sample dataset from **rattle**.