Association Analysis: Apriori

Examples

The R function `apriori` from the `arules` package provides the apriori functionality using Borgelt's excellent implementation (See Chapter 42). We use the `arules` package here to illustrate the discovery of apriori rules.

Survey Data: Data Preparation

For this example we will use the survey dataset (see See Section 14.3.4). This dataset is a reasonable size and has some common real world issues. The vignette for `arules`, by the authors of the package (Hahsler et al., 2005), also use a similar dataset, available within the package through `data(Survey)`. We borrow some of their data transformations here.

We first review the dataset: there are 32,561 entities and 15 variables.

```r
> load("survey.RData")
> dim(survey)
[1] 32561 15
> summary(survey)

  Age             Workclass            fnlwgt
          Min. :17.00  Private :22696  Min. :  12285
  1st Qu.:28.00  Self-emp-not-inc: 2541  1st Qu.: 117827
  Median :37.00  Local-gov : 2093  Median : 178356
  Mean   :38.58  State-gov : 1298  Mean   : 189778
  3rd Qu.:48.00  Self-emp-inc : 1116  3rd Qu.: 237051
  Max.   :90.00  (Other) :  981  Max.   :1484705
  NA's            : 1836

  Education       Education.Num              Marital.Status
            HS-grad :10501  Min. : 1.00  Divorced : 4443
  Some-college: 7291  1st Qu.: 9.00  Married-AF-spouse : 23
  Bachelors : 5355  Median :10.00  Married-civ-spouse :14976
  Masters : 1723  Mean  :10.08  Married-spouse-absent: 418
  Assoc-voc : 1382  3rd Qu.:12.00  Never-married :10683
  11th : 1175  Max. :16.00  Separated : 1025
  (Other) : 5134  Widowed :   993

  Occupation          Relationship                Amer-Indian-Eskimo: 311
            Prof-specialty : 4140  Husband :13193  Asian-Pac-Islander: 1039
            Exec-managerial: 4066  Other-relative: 981  Other : 271
            Adm-clerical : 3770  Own-child : 5068
The first 5 rows of the dataset give some idea of the type of data:

```r
> survey[1:5,]

Age       Workclass   fnlwgt Education Education.Num Marital.Status
1    39         State-gov  77516   Bachelors            13     Never-married
2    50   Self-emp-not-inc  83311   Bachelors            13  Married-civ-spouse
3    38          Private 215646   HS-grad             9       Divorced
4    53          Private 234721       11th             7  Married-civ-spouse
5    28          Private 338409   Bachelors            13  Married-civ-spouse

Occupation Relationship     Race     Sex  Capital.Gain Capital.Loss
1        Adm-clerical   Not-in-family  White   Male            2174            0
2  Exec-managerial            Husband  White   Male            0            0
3 Handlers-cleaners Not-in-family  White   Male            0            0
4 Handlers-cleaners            Husband  Black   Male            0            0
5        Prof-specialty            Wife Black Female            0            0

Hours.Per.Week Native.Country Salary.Group
1             40        United-States <=50K
2             13        United-States <=50K
3             40        United-States <=50K
4             40        United-States <=50K
5             40          Cuba   <=50K
```

The dataset contains a mixture of categorical and numeric variables while the apriori algorithm works just with categorical variables (or factors). We note that the variable `fnlwgt` is a calculated value and not of interest to us.
so we can remove it from the dataset. The variable Education.Num is redundant since is it simply a numeric mapping of Education. We can remove these from the data frame simply by assigning NULL to them:

```r
> survey$fnlwgt <- NULL
> survey$Education.Num <- NULL
```

This still leaves Age, Capital.Gain, Capital.Loss, and Hours.Per.Week. Following Hahsler et al. (2005), we will partition Age and Hours.Per.Week into fours segments each:

```r
> survey$Age <- ordered(cut(survey$Age, c(15, 25, 45, 65, 100)),
    labels = c("Young", "Middle-aged", "Senior", "Old"))

> survey$Hours.Per.Week <- ordered(cut(survey$Hours.Per.Week,
    c(0, 25, 40, 60, 168)),
    labels = c("Part-time", "Full-time", "Over-time", "Workaholic"))
```

Again following Hahsler et al. (2005) we map Capital.Gain and Capital.Loss to None, and Low and High according to the median:

```r
> survey$Capital.Gain <- ordered(cut(survey$Capital.Gain,
    c(-Inf, 0, median(survey$Capital.Gain[survey$Capital.Gain >0]), 1e+06)),
    labels = c("None", "Low", "High"))

> survey$Capital.Loss <- ordered(cut(survey$Capital.Loss,
    c(-Inf, 0, median(survey$Capital.Loss[survey$Capital.Loss >0]), 1e+06)),
    labels = c("None", "Low", "High"))
```

That is pretty much it in terms of preparing the data for apriori:

```r
> survey[1:5,]
   Age       Workclass Education     Marital.Status        Occupation
1 Middle-aged State-gov Bachelors Never-married      Adm-clerical
2 Senior Self-emp-not-inc Bachelors Married-civ-spouse Exec-managerial
3 Middle-aged Private HS-grad Divorced Handlers-cleaners
4 Senior Private 11th Married-civ-spouse Handlers-cleaners
5 Middle-aged Private Bachelors Married-civ-spouse  Prof-specialty
   Relationship Race    Sex Capital.Gain Capital.Loss Hours.Per.Week
1       Not-in-family White   Male          Low         None      Full-time
2            Husband White   Male           None         None      Part-time
3       Not-in-family White   Male           None         None      Full-time
4            Husband Black   Male           None         None      Full-time
5                Wife Black Female None           None         None      Full-time
```
The apriori function will coerce the data into the transactions data type, and this can also be done prior to calling apriori using the as function to view the data as a transaction dataset:

```r
> library(arules)
> survey.transactions <- as(survey, "transactions")
> survey.transactions
transactions in sparse format with
  32561 transactions (rows) and
  115 items (columns)
```

This illustrates how the transactions data type represents variables in a binary form, one binary variable for each level of each categorical variable. There are 115 distinct levels (values for the categorical variables) across all 13 of the categorical variables.

The summary function provides more details:

```r
> summary(survey.transactions)
transactions as itemMatrix in sparse format with
  32561 rows (elements/itemsets/transactions) and
  115 columns (items)

most frequent items:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capital.Loss</td>
<td>None</td>
<td>31042</td>
</tr>
<tr>
<td>Capital.Gain</td>
<td>None</td>
<td>29849</td>
</tr>
<tr>
<td>Native.Country</td>
<td>United-States</td>
<td>29170</td>
</tr>
<tr>
<td>Race</td>
<td>White</td>
<td>27816</td>
</tr>
<tr>
<td>Salary.Group</td>
<td>&lt;=50K</td>
<td>24720</td>
</tr>
</tbody>
</table>

includes extended item information - examples:

<table>
<thead>
<tr>
<th>Labels</th>
<th>Variables</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Age</td>
<td>Young</td>
</tr>
<tr>
<td>2</td>
<td>Age</td>
<td>Middle-aged</td>
</tr>
</tbody>
</table>
The summary begins with a description of the dataset sizes. This is followed by a list of the most frequent items occurring in the dataset. A Capital.Loss of None is the single most frequent item, occurring 31,042 times (i.e., pretty much no transaction has any capital loss recorded). The length distribution of the transactions is then given, indicating that some transactions have NA's for some of the variables. Looking at the summary of the original dataset you'll see that the variables Workclass, Occupation, and Native.Country have NA's, and so the distribution ranges from 10 to 13 items in a transaction.

The final piece of information in the summary output indicates the mapping that has been used to map the categorical variables to the binary variables, so that Age = Young is one binary variable, and Age = Middle-aged is another.

Now it is time to find all association rules using apriori. After a little experimenting we have chosen a support of 0.05 and a confidence of 0.95. This gives us 4,236 rules.

```r
> survey.rules <- apriori(survey.transactions,
                          parameter = list(support=0.05, confidence=0.95))
```

```
parameter specification:
  confidence minval smax arem   aval originalSupport support minlen maxlen target
data: 0.95 0.1 1 none FALSE TRUE 0.05 1 5 rules
ext 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
version 4.21 (2004.05.09)  (c) 1996-2004 Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[115 item(s), 32561 transaction(s)] done [0.07s].
  sorting and recoding items ... [36 item(s)] done [0.01s].
  creating transaction tree ... done [0.08s].
  checking subsets of size 1 2 3 4 5 done [0.23s].
  writing ... [4236 rule(s)] done [0.00s].
  creating S4 object ... done [0.04s].
```

```
> survey.rules
set of 4236 rules
```

```
> summary(survey.rules)
set of 4236 rules

rule length distribution (lhs + rhs):
```
summary of quality measures:

<table>
<thead>
<tr>
<th></th>
<th>support</th>
<th>confidence</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0.05003</td>
<td>0.9500</td>
<td>0.9965</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0.06469</td>
<td>0.9617</td>
<td>1.0186</td>
</tr>
<tr>
<td>Median</td>
<td>0.08435</td>
<td>0.9715</td>
<td>1.0505</td>
</tr>
<tr>
<td>Mean</td>
<td>0.11418</td>
<td>0.9745</td>
<td>1.2701</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0.13267</td>
<td>0.9883</td>
<td>1.3098</td>
</tr>
<tr>
<td>Max.</td>
<td>0.95335</td>
<td>1.0000</td>
<td>2.9725</td>
</tr>
</tbody>
</table>

We can inspect the first 5 rules (slightly edited to suit publication):

```r
> inspect(survey.rules[1:5])

lhs                                 rhs                  support   conf lift
1 {}                               => {Capital.Loss = None}  0.953  0.953 1.00
2 {Occupation = Machine-op-inspct} => {Workclass = Private}  0.058  0.955 1.37
3 {Occupation = Machine-op-inspct} => {Capital.Loss = None}  0.059  0.966 1.01
4 {Race = Black}                   => {Capital.Loss = None}  0.093  0.967 1.01
5 {Occupation = Other-service}     => {Salary.Group = <=50K} 0.097  0.958 1.26
```

Or we can list the first 5 rules which have a lift greater that 2.5

```r
> subset(survey.rules, subset=lift>2.5)

set of 40 rules

> inspect(subset(survey.rules, subset=lift>2.5)[1:5])

lhs                              rhs                         support conf lift
1 {Age = Young,                  
   Hours.Per.Week = Part-time} => {Marital.Status = Never-married} 0.06 0.95 2.9
2 {Age = Young,                  
   Relationship = Own-child}   => {Marital.Status = Never-married} 0.10 0.97 2.9
3 {Age = Young,                  
   Hours.Per.Week = Part-time,  
   Salary.Group = <=50K}       => {Marital.Status = Never-married} 0.06 0.96 2.9
4 {Age = Young,                  
   Hours.Per.Week = Part-time,  
   Native.Country = United-States}=>{Marital.Status=Never-married} 0.05 0.95 2.9
5 {Age = Young,                  
   Capital.Gain = None,        
   Hours.Per.Week = Part-time} => {Marital.Status = Never-married} 0.05 0.96 2.9
```

Here we build quite a few more rules and then view the rule with highest lift:
```r
> survey.rules <- apriori(survey.transactions,
                      parameter = list(support = 0.05, confidence = 0.8))

parameter specification:
  confidence minval smax arem aval originalSupport support minlen maxlen target
  0.8    0.1    1 none FALSE            TRUE    0.05      1      5  rules
  ext
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
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[115 item(s), 32561 transaction(s)] done [0.09s].
sorting and recoding items ... [36 item(s)] done [0.02s].
creating transaction tree ... done [0.10s].
checking subsets of size 1 2 3 4 5 done [0.35s].
writing ... [13344 rule(s)] done [0.00s].
creating S4 object  ... done [0.08s].

> inspect(SORT(subset(survey.rules, subset=rhs %in% "Salary.Group"),
              by="lift")[1:3])

<table>
<thead>
<tr>
<th>lhs</th>
<th>rhs</th>
<th>support</th>
<th>conf</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>{Occupation = Exec-managerial, Relationship = Husband, Capital.Gain = High}</td>
<td>{Salary.Group = &gt;50K}</td>
<td>0.007</td>
<td>1</td>
<td>4.15</td>
</tr>
<tr>
<td>{Age = Middle-aged, Occupation = Exec-managerial, Capital.Gain = High}</td>
<td>{Salary.Group = &gt;50K}</td>
<td>0.005</td>
<td>1</td>
<td>4.15</td>
</tr>
<tr>
<td>{Age = Middle-aged, Education = Bachelors, Capital.Gain = High}</td>
<td>{Salary.Group = &gt;50K}</td>
<td>0.006</td>
<td>1</td>
<td>4.15</td>
</tr>
</tbody>
</table>
```
Video Marketing: Transactions from File

A simple example from e-commerce is that of an on-line retailer of DVDs, maintaining a database of all purchases made by each customer. (They will also, of course, have web log data about what the customers browsed.) The retailer might be interested to know what DVDs appear regularly together and to then use this information to make recommendations to other customers.

The input data consists of \texttt{``transactions''} like the following, which record on each line the purchase history of a customer, with each purchase separated by a comma (i.e., CSV format as discussed in See Section \texttt{14.3.4}):

\begin{verbatim}
Sixth Sense,LOTR1,Harry Potter1,Green Mile,LOTR2
Gladiator,Patriot,Braveheart
LOTR1,LOTR2
Gladiator,Patriot,Sixth Sense
Gladiator,Patriot,Sixth Sense
Gladiator,Patriot,Sixth Sense
Harry Potter1,Harry Potter2
Gladiator,Patriot
Gladiator,Patriot,Sixth Sense
Sixth Sense,LOTR,Galadior,Green Mile
\end{verbatim}

This data might be stored in the file \texttt{DVD.csv} which can be directly loaded into \texttt{R} using the \texttt{read.transactions} function of the \texttt{arules} package:

\begin{verbatim}
> library(arules)
> dvd.transactions <- read.transactions("DVD.csv", sep="",")
> dvd.transactions

transactions in sparse format with
10 transactions (rows) and
11 items (columns)
\end{verbatim}

This tells us that there are, in total, 11 items that appear in the basket. The \texttt{read.transactions} function can also read data from a file with transaction ID and a single item per line (using the \texttt{format="single"} option).

For example, if the data consists of:

\begin{verbatim}
1,Sixth Sense
1,LOTR1
1,Harry Potter1
1,Green Mile
1,LOTR2
2,Gladiator
2,Patriot
2,Braveheart
3,LOTR1
\end{verbatim}
we read the data with:

```r
> dvd.transactions <- read.transactions("DVD.csv", format="single",
sep=",", cols=c(1,2))
> dvd.transactions
transactions in sparse format with
 10 transactions (rows) and
 11 items (columns)
A summary of the dataset is obtained in the usual way:

> summary(dvd.transactions)
transactions as itemMatrix in sparse format with
 10 rows (elements/itemsets/transactions) and
 11 columns (items)
most frequent items:

<table>
<thead>
<tr>
<th>Item</th>
<th>Gladiator</th>
<th>Patriot</th>
<th>Sixth Sense</th>
<th>Green Mile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gladiator</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Harry Potter1</td>
<td>2</td>
<td>(Other)</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

element (itemset/transaction) length distribution:
min. 1st Qu. median Mean 3rd Qu. max.
2.00 2.25 3.00 3.00 3.00 5.00

includes extended transaction information - examples:

transactionIDs
1    1
2    2
3    3

The dataset is identified as a sparse matrix consisting of 10 rows (transactions in this case) and 11 columns or items. In fact, this corresponds to the total number of distinct items in the dataset, which internally are represented as a binary matrix, one column for each item. A distribution across the most frequent items (Gladiator appears in 6 "baskets") is followed by a distribution over the length of each transaction (one transaction has 5 items in the "basket"). The final extended transaction information can be ignored in this simple example, but is explained for the more complex example that follows.

Association rules can now be built from the dataset:

```r
> dvd.apriori <- apriori(dvd.transactions)
```

parameter specification:

- confidence minval smax arem aval originalSupport support minlen
- 0.8 0.1 1 none FALSE TRUE 0.1 1
  maxlen target ext
  5 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
version 4.21 (2004.05.09) (c) 1996-2004 Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 10 transaction(s)] done [0.00s].
sorting and recoding items ... [7 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [7 rule(s)] done [0.00s].
creating S4 object ... done [0.01s].

The output here begins with a summary of the parameters chosen for the algorithm. The default values of confidence (0.8) and support (0.1) are noted, in addition to the minimum and maximum number of items in an itemset (minlen=1 and maxlen=5). The default target is rules, but you could instead target itemsets or hyperedges. These can be set in the call to apriori with the parameter argument which takes a list of keyword arguments.
We view the actual results of the modelling with the *inspect* function:

```r
> inspect(dvd.apriori)

<table>
<thead>
<tr>
<th>lhs</th>
<th>rhs</th>
<th>support</th>
<th>confidence</th>
<th>lift</th>
</tr>
</thead>
<tbody>
<tr>
<td>{LOTR1}</td>
<td>=&gt; {LOTR2}</td>
<td>0.2</td>
<td>1.0</td>
<td>5.000000</td>
</tr>
<tr>
<td>{LOTR2}</td>
<td>=&gt; {LOTR1}</td>
<td>0.2</td>
<td>1.0</td>
<td>5.000000</td>
</tr>
<tr>
<td>{Green Mile}</td>
<td>=&gt; {Sixth Sense}</td>
<td>0.2</td>
<td>1.0</td>
<td>1.666667</td>
</tr>
<tr>
<td>{Gladiator}</td>
<td>=&gt; {Patriot}</td>
<td>0.6</td>
<td>1.0</td>
<td>1.666667</td>
</tr>
<tr>
<td>{Patriot}</td>
<td>=&gt; {Gladiator}</td>
<td>0.6</td>
<td>1.0</td>
<td>1.666667</td>
</tr>
<tr>
<td>{Sixth Sense, Gladiator}</td>
<td>=&gt; {Patriot}</td>
<td>0.4</td>
<td>1.0</td>
<td>1.666667</td>
</tr>
<tr>
<td>{Sixth Sense, Patriot}</td>
<td>=&gt; {Gladiator}</td>
<td>0.4</td>
<td>1.0</td>
<td>1.666667</td>
</tr>
</tbody>
</table>
```

The rules are listed in order of decreasing lift.

We can change the parameters to get other association rules. For example we might reduce the support and deliver many more rules (81 rules):

```r
> dvd.apriori <- apriori(dvd.transactions, par=list(supp=0.01))
```

Or else we might maintain support but reduce confidence (20 rules):

```r
> dvd.apriori <- apriori(dvd.transactions, par=list(conf=0.1))
```
Other Examples

Health data is another example where association analysis can be effectively employed. Suppose a patient is obtaining a series of pathology and diagnostic imaging tests as part of an investigation to determine the cause of some symptoms. The "shopping basket" here is the collection of tests performed. Are there items in the basket that don’t belong together? Or are there some patients who don’t seem to be getting the appropriate selection of tests? The Australian Health Insurance Commission discovered an unexpected correlation between two pathology tests performed by pathology laboratories and paid for by insurance (Viveros et al., 1999). It turned out that only one of the tests was actually necessary, yet regularly both were being performed. The insurance organisation was able to reduce over-payment by disallowing payment for both tests, resulting in a saving of some half a million dollars per year.

In a very different application, IBM's Advance Scout was developed to identify different strategies employed by basketball players in the US NBA. Discoveries include the observation that Scottie Pippen's favorite move on the left block is a right-handed hook to the middle. And when guard Ron Harper penetrates the lane, he shoots the ball 83% of the time. Also it was noticed that 17% of Michael Jordan's offence comes on isolation plays, during which he tends to take two or three dribbles before pulling up for a jumper (Bhandari et al., 1997).

There are many more examples of unexpected associations having been discovered between items and, importantly, found to be particularly useful for improving business (and other) processes.