IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Association Rules

Lab: Association Rules

- Generation of Frequent Itemsets
- Generation of Association Rules
- Quality of Association Rules
- Vizualization

Please send comment to (version 1.0 – February 4, 2020):

Romain Billot *romain.billot@telecom-bretagne.eu*
Philippe Lenca *philippe.lenca@telecom-bretagne.eu*

1 Data presentation

Association rules aim at analysing large collections of transactions, in which each transaction is composed of one or more items. The goal is then to see what items are frequently bought together and to discover a list of rules that describe the purchasing behaviour and hence relationships between the items. These relationships depend both on the business context and the nature of the algorithm being used for the discovery. Association rules can be implemented for the purpose of market basket analysis, enabling, for instance, cross-merchandising between products and high-margin or high-ticket items. Moreover, recommender systems can also use association rules to discover related products or identify customers who have similar interests. For example, association rules may suggest that those customers who have bought product A have also bought product B, or those customers who have bought products A, B, C, are more similar to this customer.

–The Grocery dataset– The Groceries dataset has been collected from 30 days of real-world transactions of a grocery store. It contains 9835 transactions and the items are aggregated into 169 categories. The installation of the R *arules* package is required. The class of the dataset is *transactions*, as defined by the *arules* package. The *transactions* class contains three slots:

Slot (not variable)	Meaning
`transactionInfo`	Data frame with vectors of the same length as the number of transactions
`itemInfo`	Data frame to store item labels
`data`	Binary incidence matrix that indicates which item labels appear in every transaction

Table 1: Structure of the Groceries dataset from class *transactions*

Question 1

Load, understand and describe carefully the dataset into R. Use the *data*, *class*, *summary* and the @ character to access the slots listed previously.

Solution 1

The summary shows that the most frequent items in the dataset include items such as whole milk, other vegetables, rolls/buns, soda and yogurt. The *transactionInfo* is not being used. With *Groceries@itemInfo*, we display all 169 grocery labels as well as their categories. Each grocery label is mapped to two levels of categories, *level1* and *level2*, where *level1* is a superset of *level2*. For example, grocery label *sausage* belongs to the *sausage* category in *level2*, and it is part of the *meat and sausage* category in *level1*. Each row in the output shows a transaction that includes one or more products, and each transaction corresponds to everything in a customer's shopping cart. For instance, in the first transaction, a customer has purchased whole milk and cereals.

R code:

```
library("arules")
library("arulesViz")
data("Groceries")
class(Groceries)
Groceries@itemInfo
apply(Groceries@data[,1:10],2, function (r) paste(Groceries@itemInfo[r,"labels"],collapse=",
    "))
```

2 Generation of Frequent Itemset

In this section, we are going to illustrate, step by step, the implementation of the *Apriori* algorithm to create frequent itemsets.

Question 2

We first assume that the minimum threshold is set to 0.02 based on management discretion. Hence an itemset should appear at least 198 times to be considered as a frequent itemset. The first iteration of the algorithm computes the support of each product in the dataset and retains those products that satisfy the minimum support. Use the *apriori* algorithm and set the $minlen = 1$ and $maxlen = 1$ in the parameter list in order to show 1 itemsets only.

Solution 2

The summary of the itemsets shows that the support of 1-itemsets ranges from 0.02105 to 0.25552. Since the maximum support of the 1-itemsets in the dataset is only 0.25552, it is advised to set the minimum support threshold not too close to that number in order to discover interesting rules.

R code:

```
itemsets <- apriori(Groceries,parameter=list(minlen=1,maxlen=1,support=0.02,target="frequent
    itemsets"))
summary(itemsets)
```

Question 3

Use the *inspect* function to display the top 10 frequent 1-itemsets sorted by their support.

Solution 3

Of all the transaction records, The 59 1-itemsets such as "whole milk", "other vegetables" and so on, satisfy the minimum support. They are called frequent 1-itemsets.

R code:

```
inspect(head(sort(itemsets,by="support"),10))
```

Question 4

In this question, the goal is to compute the support of each candidate 2-itemset and retain those that satisfy the minimum support. Use the *apriori* function again. Do the same step by step with 3-itemsets and 4-itemsets. Conclude.

Solution 4

To answer this question, one must have to change only $minlen = 2$ and $maxlen = 2$ in the parameters of the algorithm. We can notice that `whole milk` appears 6 times in the top 10 2-itemsets ranked by support. Next, the list of frequent 2-itemsets is joined onto itself to form candidate 3-itemsets. Only two frequent itemsets are identified by the algorithm. In the next iteration, there is only one candidate 4-itemset and its support is below 0.02. No frequent 4-itemsets have been found, the algorithm converges. After simulating the Apriori algorithm at each iteration, we have seen that it runs out of support when $k = 4$. Therefore, the frequent itemsets contain 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

R code:

```
itemsets <- apriori(Groceries,parameter=list(minlen=2,maxlen=2,support=0.02,target="frequent
    itemsets"))
summary(itemsets)
inspect(head(sort(itemsets,by="support"),10))
itemsets <- apriori(Groceries,parameter=list(minlen=3,maxlen=3,support=0.02,target="frequent
    itemsets"))
inspect(sort(itemsets,by="support"))
itemsets <- apriori(Groceries,parameter=list(minlen=4,maxlen=4,support=0.02,target="frequent
    itemsets"))
```

Question 5

Now run the algorithm without specifying the `maxlen` parameter. The algorithm continues each iteration until it runs out of support or until k reaches the default value of `maxlen` set to 10. What can you conclude ? Try to run the algorithm with other minimum support thresholds.

Solution 5

We retrieve the previous results. the frequent itemsets contain 59 frequent 1-itemsets, 61 frequent 2-itemsets, and 2 frequent 3-itemsets.

R code:

```
itemsets<- apriori(Groceries,parameter=list(minlen=1,support=0.02,target="frequent itemsets"))
```

3 Rule generation and visualization

Question 6

The *apriori* function can also be used to generate rules. Use it by assuming here that the minimum support threshold is 0.001, the minimum confidence threshold 0.6. How many rules are created? What is the impact of both thresholds on the number of generated rules?

Solution 6

The function creates 2918 rules from all transactions in the groceries dataset that satisfy both the minimum support and the minimum confidence. The summary of the rules shows the number of rules and ranges of the support, confidence and lift.

R code:

```
rules <- apriori(Groceries,parameter=list(support=0.001,confidence=0.6,target="rules"))
summary(rules)
```

Question 7

Display a scatterplot of the 2918 rules. Access to the quality of the rules (with @) to display also a scatterplot matrix that will compare the support, confidence and lift of the 2918 rules

Solution 7

Entering plot(rules) displays a scatterplot of the rules where the horizontal axis is the support, the vertical axis is the confidence, and the shading is the lift. We can see that the highest lift occurs at a low support and a low confidence. The scatterplot matrix of rule@quality shows that lift is proportional to confidence and illustrates several linear groupings. As we know, $Lift = \frac{Confidence}{Support}$ and when the support remains the same, lift is proportional to confidence, and the slope of the linear trend is the reciprocal of the support.

R code:

```
plot(rules)
plot(rules@quality)
```

Question 8

Re-use a previous function to show the top ten rules sortest by the lift. Which rule has the highest lift ? Then, create an object that contains the rules whose confidence is above 0.9. How many rules does this object contain?

Solution 8

The rule with the highest lift says that the combination of instant food products with soda leads to the consumption of hamburger meat, which seems to be logical. Then, 127 rules have a confidence greater than 0.9.

R code:

```
inspect(head(sort(rules,by="lift"),10))
confidentRules<-rules[quality(rules)$confidence>0.9]
confidentRules
```

Question 9

We denote as "antecedent" the left-hand side of a rule and as "consequent" the right-hand side of a rule. Tune the parameters of the plot function in order to create a matrix-based visualization of the consequents versus the antecedents. As a color legend, put a color matrix indicating the lift and the confidence to which each square in the main matrix corresponds.

Solution 9

Together with the plot command, the R console displays a distinct list of the antecedents and consequents.

R code:

```
plot(confidentRules,method="matrix",measure=c("lift","confidence"),control=list(reorder=TRUE))
```

Question 10

Prepare a graph visualization of the top five rules with the highest lift.

Solution 10

In the graph, the arrow always points from an antecedent item to a consequent item. For example an arrow connect ham and processed cheese to white bread. The legend on the top right suggests that the size of the circle indicates the support, ranging from 0.001 to 0.002. The colour (shade) represents the lift ranging from 11.279 to 18.996. Hence we retrieve the rule with the highest lift.

R code:

```
highliftRules <- head(sort(rules,by="lift"),5)
plot(highliftRules,method="graph",control=list(type="items"))
```