

Association Rule Mining

Exploiting Patterns and Relationships in Data

When we go grocery shopping, we often have a standard list of things to buy. Each shopper has a distinctive list, depending on one's needs and preferences. A housewife might buy healthy ingredients for a family dinner, while a bachelor might buy beer and chips. Understanding these buying patterns can help to increase sales in several ways. If there is a pair of items, X and Y, that are frequently bought together:

- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items.
- Advertisements on X could be targeted at buyers who purchase Y.
- X and Y could be combined into a new product, such as having Y in flavors of X.























While we may know that certain items are frequently bought together, the question is - how do we uncover these associations?

Besides increasing sales profits, association rules can also be used in other fields. In medical diagnosis for instance, understanding which symptoms tend to co-morbid can help to improve patient care and medicine prescription.

Definitions

Association rules analysis is a technique to uncover how items are associated to each other. There are three common ways to measure association.

Measure 1: Support. This says how *popular* an itemset is, as measured by the proportion of transactions in which an itemset appears. In Table 1 below, the support of {apple} is 4 out of 8, or 50%. Itemsets can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

Transaction 1	   
Transaction 2	  
Transaction 3	 
Transaction 4	 
Transaction 5	   
Transaction 6	  
Transaction 7	 
Transaction 8	 

This get translated mathematically into a matrix of zeros and ones:

<u>Transaction</u>	<u>Apple</u>	<u>Beer</u>	<u>Rice</u>	<u>Pear</u>	<u>Turkey leg</u>	<u>Baby bottle</u>
1	1	1	1	0	1	0
2	1	1	1	0	0	0
3	1	1	0	0	0	0
4	1	0	0	0	0	0
5	0	1	1	0	1	1
6	0	1	1	0	0	1
7	0	1	0	0	0	1
8	0	0	0	1	0	1

Table 1. Example Transactions

If you discover that sales of items beyond a certain proportion tend to have a significant impact on your profits, you might consider using that proportion as your support threshold. You may then identify itemsets with support values above this threshold as significant itemsets.

$$\text{Support} \{\text{🍎}\} = \frac{4}{8}$$

Measure 2: Confidence. This says how *likely* item Y is purchased when item X is purchased, expressed as {X -> Y}. This is measured by the proportion of transactions with item X, in which item Y also appears. In Table 1, the confidence of {apple -> beer} is 3 out of 4, or 75%.

$$\text{Confidence} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎, 🍺}\}}{\text{Support} \{\text{🍎}\}}$$

One drawback of the confidence measure is that it might misrepresent the importance of an association. This is because it only accounts for how popular apples are, but not beers. If beers are also very popular in general, there will be a higher chance that a transaction containing apples will also contain beers, thus inflating the confidence measure. To account for the base popularity of both constituent items, we use a third measure called lift.

Measure 3: Lift. This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. In Table 1, the lift of {apple -> beer} is 1, which implies no association between items. A lift value greater than 1 means that item Y is likely to be bought if item X is bought, while a value less than 1 means that item Y is unlikely to be bought if item X is bought.

$$\text{Lift} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎, 🍺}\}}{\text{Support} \{\text{🍎}\} \times \text{Support} \{\text{🍺}\}}$$

Example:

Below we refer to a dataset on grocery transactions from the arules R library. It contains transactions at a grocery outlet over 30 days. The graph shows associations between selected items.

Larger circles imply higher support, while red circles imply higher lift:

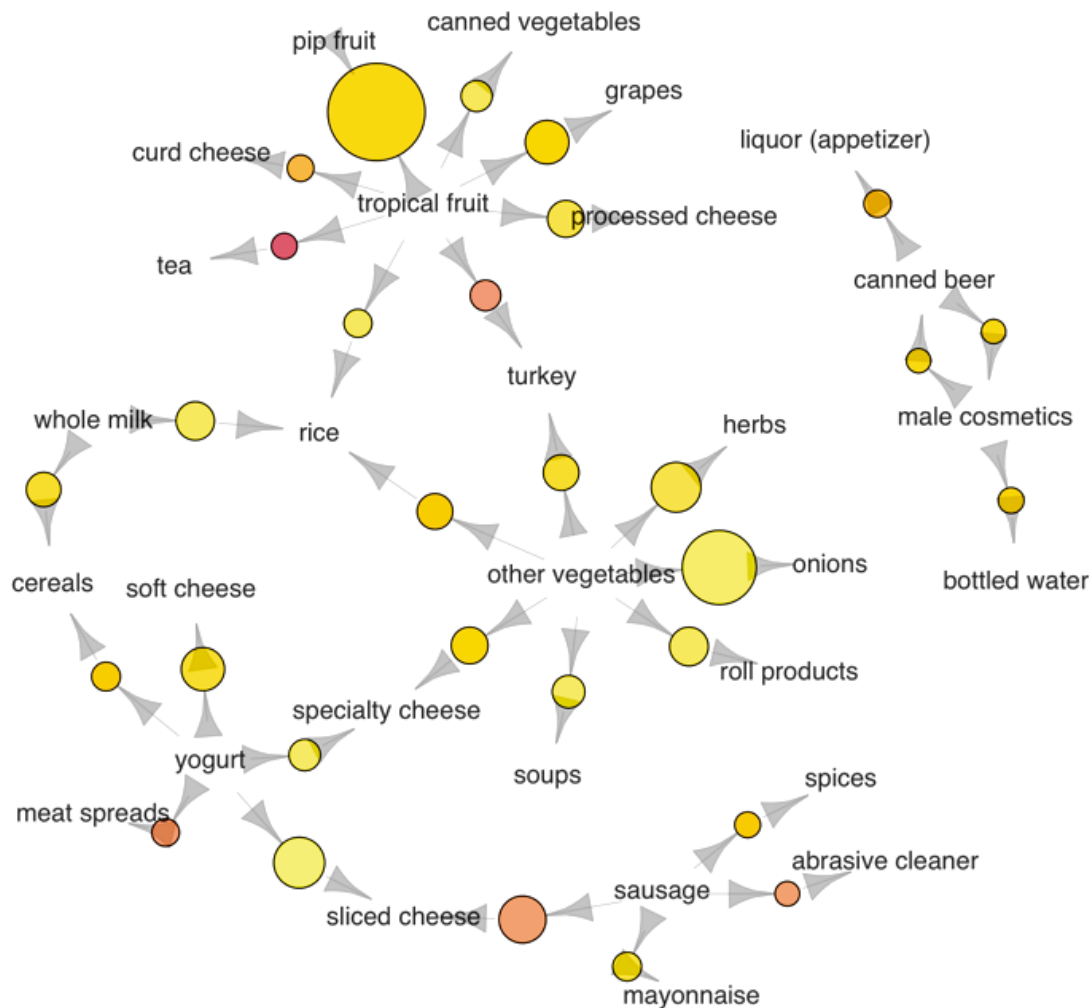


Figure 1. Associations between selected items visualized using the arulesViz R library.

Several purchase patterns can be observed.

- The most popular transaction was of pip and tropical fruits
- Another popular transaction was of onions and other vegetables
- If someone buys meat spreads, he is likely to have bought yogurt as well
- Relatively many people buy sausage along with sliced cheese
- If someone buys tea, he is likely to have bought fruit as well, possibly inspiring the production of fruit-flavored tea

One major drawback of the confidence measure is that it tends to misrepresent the importance of an association. To demonstrate this, we go back to the main dataset to pick 3 association rules containing beer:

Transaction	Support	Confidence	Lift
Canned Beer → Soda	1%	20%	1.0
Canned Beer → Berries	0.1%	1%	0.3
Canned Beer → Male Cosmetics	0.1%	1%	2.6

Table 2. Association measures for beer-related rules

The {beer → soda} rule has the highest confidence at 20%. But both beer and soda occur frequently across all transactions (see Table 3), therefore their association could just be a fluke. We can confirm this by looking at the lift value of {beer → soda}, which is 1. This implies no association between beer and soda.

Transaction	Support
Canned Beer	10%
Soda	20%
Berries	3%
Male Cosmetics	0.5%

Table 3. Support of individual items

The {beer → male cosmetics} rule has a low confidence, due to few purchases of male cosmetics in general. However, whenever someone *does* buy male cosmetics, he is very likely to buy beer as well, as can be determined from the high lift value of 2.6.

The converse is true for {beer → berries}. With a lift value below 1, we may conclude that if someone buys berries, he would likely be averse to buying beer.

Calculation of the popularity of a single itemset, like {beer, soda} is straightforward. However, a business owner would not typically ask about individual itemsets. Rather, the owner would be more interested in having a complete list of **popular** itemsets. To get this list, we need to calculate the support values for every possible combination of items and then shortlist those itemsets that meet some minimum support threshold.

In a store with just 10 items, the total number of possible configurations to examine would be

$2^{10} - 1 = 1023$. This number increases exponentially in a store with hundreds of items.

Is there a way to reduce the number of item configurations to consider?

Apriori Algorithm

The *apriori* principle can be used to reduce the number of itemsets we need to examine. The *apriori* principle states that if an itemset is infrequent, then all its subsets must also be infrequent. This means that if {beer} was found to be infrequent, we can expect {beer, pizza} to be equally or even more infrequent. So in consolidating the list of popular itemsets, we would then not need to consider {beer, pizza} nor any other itemset configuration that contains beer.

Finding itemsets with high support

Using the *apriori* principle, the number of itemsets that have to be examined can be pruned, and the list of popular itemsets can be obtained in these steps:

Step 0. Start with itemsets containing just a single item, such as {apple} or {pear}.

Step 1. Determine the support for itemsets. Keep the itemsets that meet your minimum support threshold, and remove itemsets that do not.

Step 2. Using the itemsets you have kept from Step 1, generate all the possible itemset configurations of size 2. Keep those that meet your minimum support threshold and remove itemsets that do not.

Step 3. Repeat Steps 1 & 2, increasing the size of the itemsets by one each time, until there are no more new itemsets.

This iterative process is illustrated in the animated GIF below:

[apriori](#)

Figure 2, Reducing candidate itemsets using the Apriori Algorithm

In the animation, {apple} was determined to have low support, so it was removed and all other itemset configurations that contain apple are therefore not considered. This reduced the number of itemsets to consider by more than half.

Note that the support threshold that is picked in Step 1 could be based on analysis or on past experience. For example, if you discover that sales of items beyond a certain threshold proportion tend to have a significant impact on profits, you might consider using that proportion as the support threshold.

Finding item rules with high confidence or lift

The *apriori* algorithm can be used to identify itemsets with high support. The same principle can also be used to identify item associations with **high confidence or high lift**.

Finding rules with high confidence or high lift is less computationally taxing once high-support itemsets have been identified, because confidence and lift values are calculated using support values. Recall

$$\text{Confidence} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\}}$$

and

$$\text{Lift} \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support} \{\text{🍎}, \text{🍺}\}}{\text{Support} \{\text{🍎}\} \times \text{Support} \{\text{🍺}\}}$$

For example, suppose we want to find high-confidence rules. If the rule

$$\{\text{beer, chips} \rightarrow \text{apple}\}$$

has low confidence, all other containing the same items and with apple on the right hand side would also have low confidence. Specifically, the rules

$$\begin{aligned} &\{\text{beer} \rightarrow \text{apple, chips}\} \\ &\{\text{chips} \rightarrow \text{apple, beer}\} \end{aligned}$$

would have low confidence as well. Therefore, lower level candidate item rules can be pruned using the *apriori* algorithm, so that fewer candidate rules need to be examined.

Limitations

- **Computationally Expensive.** Even though the *apriori* algorithm reduces the number of candidate itemsets to consider, this number could still be huge when store inventories are large or when the support threshold is low. An alternative solution would be to reduce the number of comparisons by using advanced data structures, such as hash tables, to sort candidate itemsets more efficiently.
- **Spurious Associations.** Analysis of large inventories would involve more itemset configurations.. Here the support threshold might have to be lowered to detect certain associations. However, lowering the support threshold might also increase the number of spurious associations detected. To ensure that identified associations are generalizable, they could first be distilled from a training dataset, before having their support and confidence assessed in a separate test dataset.