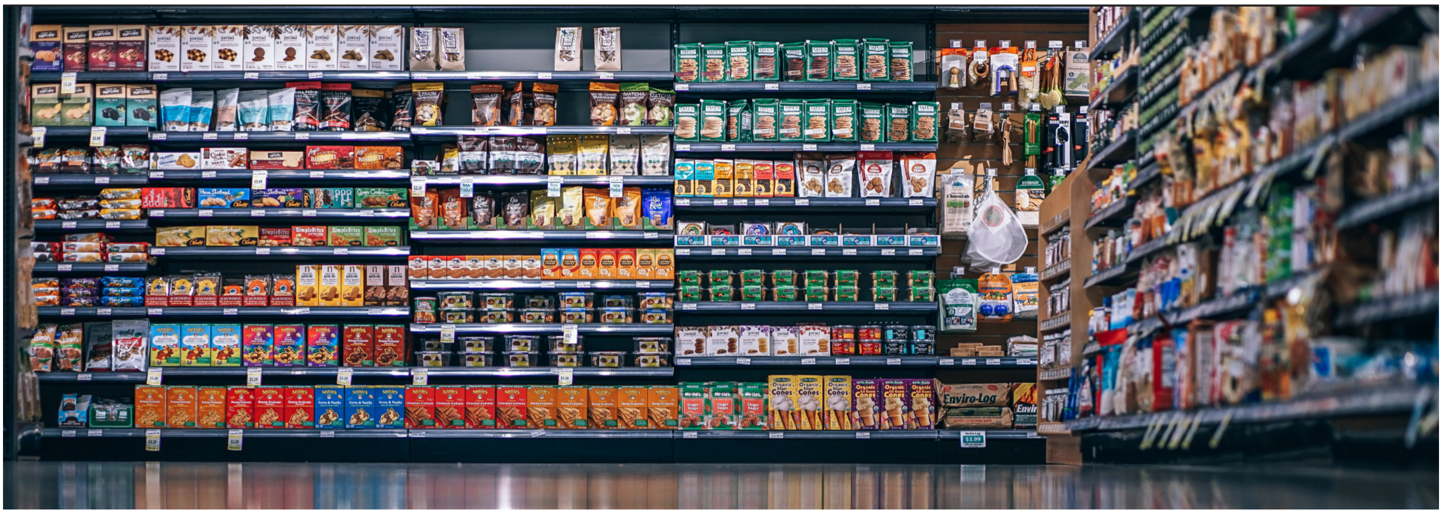
Complete guide to Association Rules (1/2)

Algorithms that help you shop faster and smarter

Anisha Garg
Sep 3, 2018 · 7 min read



Looking back at the multitude of concepts that have been introduced to me in the statistics boot camp, there is a lot to write and share. I choose to start with Association Rules because of two reasons. First, this was one of the concepts which I enjoyed learning the most and second, there are a limited resources available online to get a good grasp.

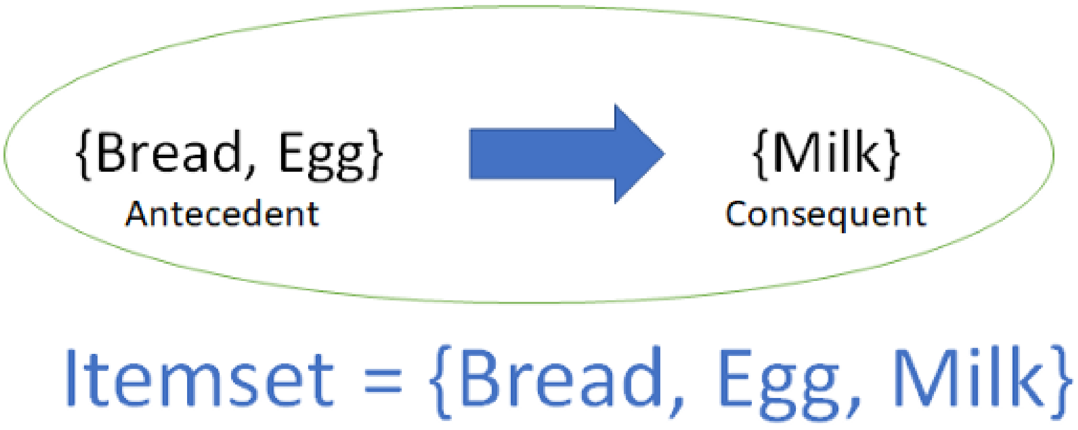
In Part 1 of the blog, I will be introducing some key terms and metrics aimed at giving a sense of what "association" in a rule means and some ways to quantify the strength of this association. Part 2 will be focused on discussing the mining of these rules from a list of thousands of items using *Apriori Algorithm*.

Association Rules is one of the very important concepts of machine learning being used in market basket analysis. In a store, all vegetables are placed in the same aisle, all dairy items are placed together and cosmetics form another set of such groups. Investing time and resources on deliberate product placements like this not only reduces a customer's shopping time, but also reminds the customer of what relevant items (s)he might be interested in buying, thus helping stores cross-sell in the process. Association rules help uncover all such relationships between items from huge databases. One important thing to note is-

> *Rules do not extract an individual's preference, rather find relationships between set of elements of every distinct transaction. This is what makes them different from collaborative filtering.*

To elaborate on this idea — Rules do not tie back a users' different transactions over time to identify relationships. List of items with unique transaction IDs (from all users) are studied as one group. *This is helpful in placement of products on aisles.* On the other hand, collaborative filtering ties back all transactions corresponding to a user ID to identify similarity between users' preferences. *This is helpful in recommending items on e-commerce websites, recommending songs on spotify, etc.*

Lets now see what an association rule exactly looks like. It consists of an antecedent and a consequent, both of which are a list of items. Note that implication here is co-occurrence and not causality. For a given rule, ***itemset*** is the list of all the items in the antecedent and the consequent.



Various metrics are in place to help us understand the strength of association between these two. Let us go through them all.

1. Support

This measure gives an idea of how frequent an *itemset* is in all the transactions. Consider *itemset1* = {bread} and *itemset2* = {shampoo}. There will be far more transactions containing bread than those containing shampoo. So as you rightly guessed, *itemset1* will generally have a higher support than *itemset2*. Now consider *itemset1* = {bread, butter} and *itemset2* = {bread, shampoo}. Many transactions will have both bread and butter on the cart but bread and shampoo? Not so much. So in this case, *itemset1* will generally have a higher support than *itemset2*. Mathematically, support is the fraction of the total number of transactions in which the itemset occurs.

$$Support(\{X\} \to \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions}$$

Value of support helps us identify the rules worth considering for further analysis. For example, one might want to consider only the itemsets which occur at least 50 times out of a total of 10,000 transactions i.e. support = 0.005. If an *itemset* happens to have a very low support, we do not have enough information on the relationship between its items and hence no conclusions can be drawn from such a rule.

2. Confidence

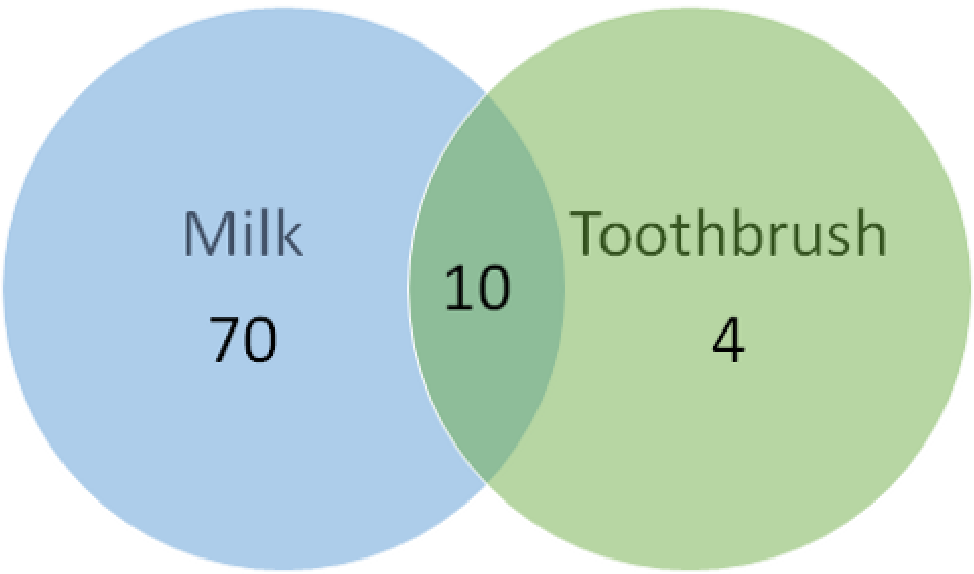
This measure defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents. That is to answer the question — of all the transactions containing say, {Captain Crunch}, how many also had {Milk} on them? We can say by common knowledge that {Captain Crunch} → {Milk} should be a high confidence rule. Technically, confidence is the conditional probability of occurrence of consequent given the antecedent.

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Transactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X}$$

Let us consider few more examples before moving ahead. What do you think would be the confidence for {Butter} → {Bread}? That is, what fraction of transactions having butter also had bread? Very high i.e. a value close to 1? That's right. What about {Yogurt} → {Milk}? High again. {Toothbrush} → {Milk}? Not so sure? Confidence for this rule will also be high since {Milk} is such a frequent itemset and would be present in every other transaction.

> *It does not matter what you have in the antecedent for such a frequent consequent. The confidence for an association rule having a very frequent consequent will always be high.*

I will introduce some numbers here to clarify this further.



Total transactions = 100. 10 of them have both milk and toothbrush, 70 have milk but no toothbrush and 4 have toothbrush but no milk.

Consider the numbers from figure on the left. Confidence for {Toothbrush} → {Milk} will be 10/(10+4) = 0.7

Looks like a high confidence value. But we know intuitively that these two products have a weak association and there is something misleading about this high confidence value. *Lift* is introduced to overcome this challenge.

> Considering just the value of confidence limits our capability to make any business inference.

3. Lift

Lift controls for the *support* (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}. *Lift* is a very literal term given to this measure. Think of it as the *lift* that {X} provides to our confidence for having {Y} on the cart. To rephrase, *lift* is the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}. Mathematically,

$$Lift(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ containing\ both\ X\ and\ Y)/(Transactions\ containing\ X)}{Fraction\ of\ transactions\ containing\ Y}$$

In cases where {X} actually leads to {Y} on the cart, value of lift will be greater than 1. Let us understand this with an example which will be continuation of the {Toothbrush} → {Milk} rule.

Probability of having milk on the cart with the knowledge that toothbrush is present (i.e. *confidence*) : 10/(10+4) = 0.7

Now to put this number in perspective, consider the probability of having milk on the cart without any knowledge about toothbrush: 80/100 = 0.8

These numbers show that having toothbrush on the cart actually reduces the probability of having milk on the cart to 0.7 from 0.8! This will be a lift of 0.7/0.8 = 0.87. Now that's more like the real picture. A value of lift less than 1 shows that having toothbrush on the cart does not increase the chances of occurrence of milk on the cart in spite of the rule showing a high confidence value. A value of lift greater than 1 vouches for high association between {Y} and {X}. More the value of lift, greater are the chances of preference to buy {Y} if the customer has already bought {X}. *Lift* is the measure that will help store managers to decide product placements on aisle.

Association Rule Mining

Now that we understand how to quantify the importance of association of products within an itemset, the next step is to generate rules from the entire list of items and identify the most important ones. This is not as simple as it might sound. Supermarkets will have thousands of different products in store. After some simple calculations, it can be shown that just 10 products will lead to 57000 rules!! And this number increases exponentially with the increase in number of items. Finding lift values for each of these will get computationally very very expensive. How to deal with this problem? How to come up with a set of most important association rules to be considered? *Apriori algorithm* comes to our rescue for this.

Read more about Apriori algorithm and find answers to all the unanswered questions here in Part 2.

Please let me know of your thoughts/questions on this blog in the comments.

Thanks to Alvira Swalin and Vaibhav Bhalekar.

Machine Learning Data Science Data Analysis Business Rules you to the iOS App store

you to the Google Play store

Complete guide to Association Rules (2/2)

Algorithms that help you shop faster and smarter

Anisha Garg
Sep 17, 2018 · 8 min read

In this blog, I will discuss the algorithms that enable efficient extraction of association rules from a list of transactions. Part 1 of this blog covers the terminology and concepts that form the foundation of association rule mining. Motivation behind this whole concept and meaning of some basic terms is explained there. Here are very brief definitions of some terms from the previous part.

1. Association Rule: Ex. {X → Y} is a representation of finding Y on the basket which has X on it
2. Itemset: Ex. {X,Y} is a representation of the list of all items which form the association rule
3. Support: Fraction of transactions containing the itemset
4. Confidence: Probability of occurrence of {Y} given {X} is present
5. Lift: Ratio of *confidence* to baseline probability of occurrence of {Y}

. . .

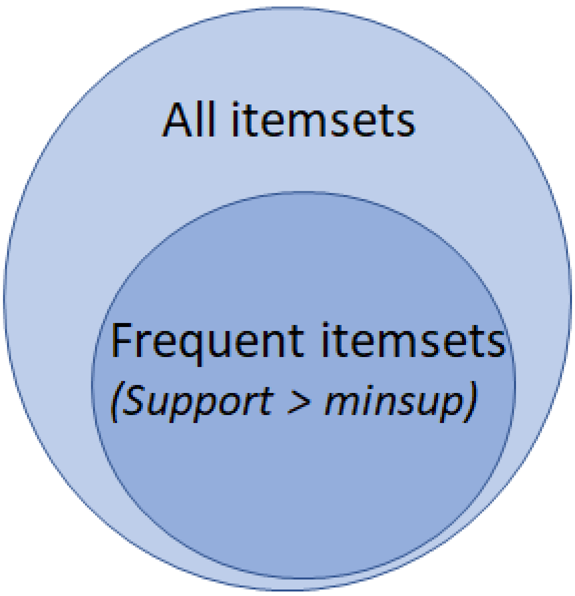
Now that we are familiar with these terms, lets proceed ahead with extracting the rules from a list of transactions. The very first step in the process is to get a list of all the items occurring at least once in our set of transactions.

> *The challenge is the mining of important rules from a massive number of association rules that can be derived from a list of items.*

Remember, rule-generation is a two step process. First is to generate an itemset like {Bread, Egg, Milk} and second is to generate a rule from each itemset like {Bread → Egg, Milk}, {Bread, Egg → Milk} etc. Both the steps are discussed below.

1. Generating itemsets from a list of items

First step in generation of association rules is to get all the *frequent* itemsets on which binary partitions can be performed to get the antecedent and the consequent. For example, if there are 6 items {Bread, Butter, Egg, Milk, Notebook, Toothbrush} on all the transactions combined, itemsets will look like {Bread}, {Butter}, {Bread, Notebook}, {Milk, Toothbrush}, {Milk, Egg, Vegetables} etc. Size of an itemset can vary from one to the total number of items that we have. Now, we seek only *frequent* itemsets from this and not all so as to put a check on the number of total itemsets generated.



Frequent itemsets are the ones which occur at least a minimum number of times in the transactions. Technically, these are the itemsets for which *support* value (fraction of transactions containing the itemset) is above a minimum threshold — *minsup*.

So, {Bread, Notebook} might not be a frequent itemset if it occurs only 2 times out of 100 transactions and (2/100) = 0.02 falls below the value of minsup.

A brute force approach to find frequent itemsets is to form all possible itemsets and check the support value of each of these. ***Apriori principle*** helps in making this search efficient. It states that

> All subsets of a frequent itemset must also be frequent.

This is equivalent to saying that number of transactions containing items {Bread, Egg} is greater than or equal to number of transactions containing {Bread, Egg, Vegetables}. If the latter occurs in 30 transactions, former is occurring in all 30 of them and possibly will occur in even some more transactions. So if support value of {Bread, Egg, Vegetables} i.e. (30/100) = 0.3 is above minsup, then we can be assured that support value of {Bread, Egg} i.e. (>30/100) = >0.3 is above minsup too. This is called the **anti-monotone property of support** where if we drop out an item from an itemset, support value of new itemset generated will either be the same or will increase.

> Apriori principle is a result of anti-monotone property of support.

Apriori principle allows us to prune all the supersets of an itemset which does not satisfy the minimum threshold condition for support. For example, if {Milk, Notebook} does not satisfy our threshold of minsup, an itemset with any item added to this will never cross the threshold too.

The methodology that results is called the ***apriori algorithm.*** Steps involved are:

Ref: https://annalyzin.files.wordpress.com/2016/04/association-rules-apriori-tutorial-explanation.gif

Generate all frequent itemsets (support ≥ minsup) having only one item. Next, generate itemsets of length 2 as all possible combinations of above itemsets. Then, prune the ones for which support value fell below minsup.

Now generate itemsets of length 3 as all possible combinations of length 2 itemsets (that remained after pruning) and perform the same check on support value.

We keep increasing the length of itemsets by one like this and check for the threshold at each step.

As can be seen from the graphic, pruning of infrequent itemsets reduces the number of itemsets to be considered by more than half! This proportion of reduction in computational power becomes more and more significant as the number of items increases.

This proportion also depends on the minimum support threshold (minsup) that we pick up which is completely subjective to the problem at hand and can be based on past experience.

2. Generating all possible rules from the frequent itemsets

Once the frequent itemsets are generated, identifying rules out of them is comparatively less taxing. Rules are formed by binary partition of each itemset. If {Bread,Egg,Milk,Butter} is the frequent itemset, candidate rules will look like:

(Egg, Milk, Butter → Bread), (Bread, Milk, Butter → Egg), (Bread, Egg → Milk, Butter), (Egg, Milk → Bread, Butter), (Butter→ Bread, Egg, Milk)

From a list of all possible candidate rules, we aim to identify rules that fall above a minimum confidence level *(minconf)*. Just like the anti-monotone property of support, *confidence of rules* ***generated from the same itemset*** *also follows an anti-monotone property*. It is anti-monotone with respect to the number of elements in consequent.

This means that confidence of (A,B,C→ D) ≥ (B,C → A,D) ≥ (C → A,B,D). To remind, confidence for {X → Y} = support of {X,Y}/support of {X}

As we know that support of all the rules generated from same itemset remains the same and difference occurs only in the denominator calculation of confidence. As number of items in X decrease, support{X} increases (as follows from the anti-monotone property of support) and hence the confidence value decreases.

An intuitive explanation for the above will be as follows. Consider F1 and F2:

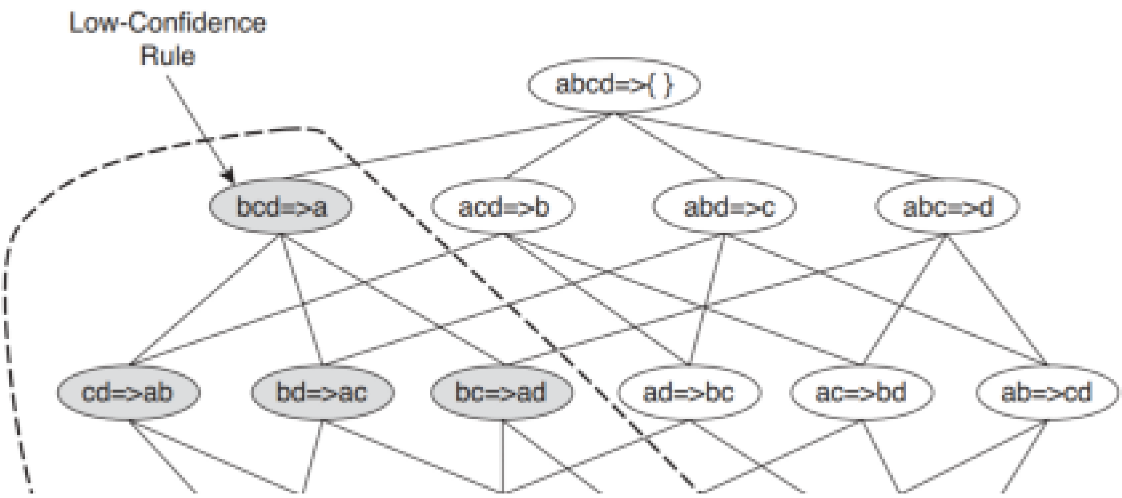
F1 = fraction of transactions having (butter) also having (egg, milk, bread)

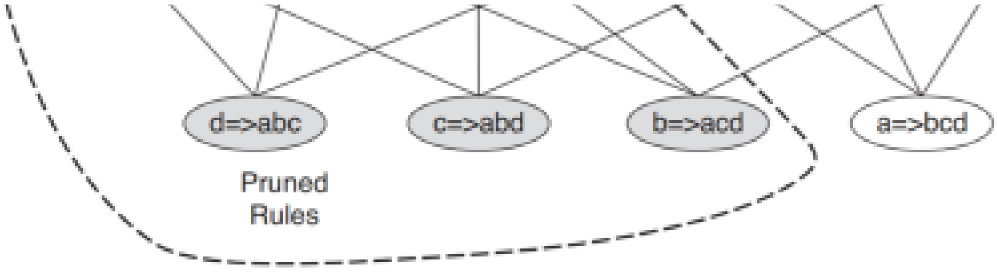
There will be many transactions having butter and all three of egg, milk and bread will be able to find place only in a small number of those.

F2 = fraction of transactions having (milk, butter, bread) also having (egg)

There will only be a handful of transactions having all three of milk, butter and bread (as compared to having just butter) and there will be high chances of having egg on those.

So it will be observed that F1 < F2. Using this property of confidence, pruning is done in a similar way as was done while looking for frequent itemsets. It is illustrated in the figure below.





Ref: https://www-users.cs.umn.edu/~kumar001/dmbook/ch6.pdf

We start with a frequent itemset {a,b,c,d} and start forming rules with just one consequent. Remove the rules failing to satisfy the minconf condition. Now, start forming rules using a combination of consequents from the remaining ones. Keep repeating until only one item is left on antecedent. This process has to be done for all frequent itemsets.

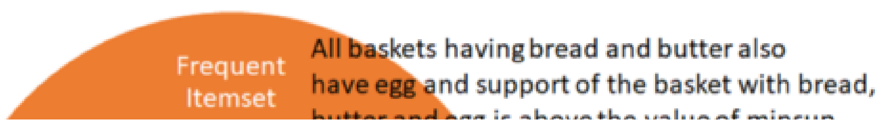
Here again, minimum confidence threshold that we pick up is completely subjective to the problem at hand.

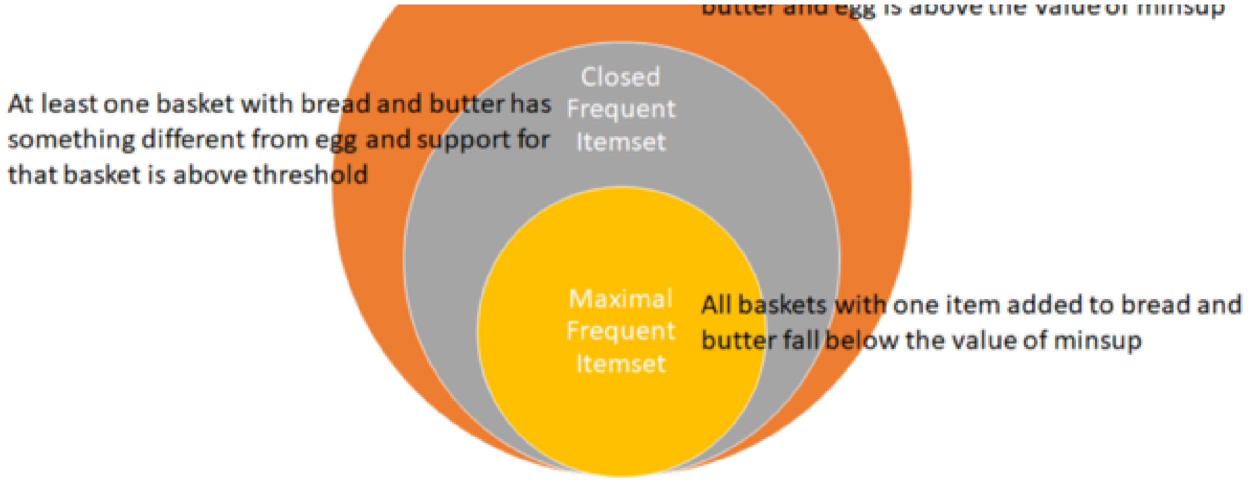
. . .

With these two steps, we have identified a set of association rules which satisfy both the minimum support and minimum confidence condition. The number of such rules obtained will vary with the values of *minsup* and *minconf*. Now, this subset of rules thus generated can be searched for highest values of lift to make business decisions. There are some nicely built libraries in R to fetch association rules from transactions by putting in minsup and minconf as parameters. They also provide capabilities to visualize the same. This article here discusses the whole process step-by-step with the code.

Before concluding, I would introduce two more terms, maximal frequent itemset and closed frequent itemset which are used as a compact representation of all the frequent itemsets.

Maximal frequent itemset: *It is a frequent itemset for which none of the immediate supersets are frequent*. This is like a frequent itemset X to which no item y can be added such that {X,y} still remains above minsup threshold.





Different cases for itemset {Bread, Butter}

Closed frequent itemset: *It is a frequent itemset for which there exists no superset which has the same support as the itemset*. Consider an itemset X. If ALL occurrences of X are accompanied by occurrence of Y, then X is NOT a closed set. *(Refer the figure below for example)*

> *Maximal frequent itemsets are valuable because they are the most compact form of representation of frequent itemsets.*

All the frequent itemsets can be derived as the subsets of maximal frequent itemsets. However, information on support of the subsets is lost. If this value is required, closed frequent itemset is another way to represent all the frequent itemsets.

> *Closed itemsets help in removing some redundant itemsets while not losing information about the support values.*

Calculation of support values of non-closed itemsets from closed itemsets is another algorithm, details of which are out of scope of this article.

. . .

I have tried to cover all the important terms and concepts related to mining of association rules through this blog going into details wherever necessary. Following are one line summaries for few terms introduced in this process —

1. Association rule mining: (a) Itemset generation, (b) Rule generation

2. Apriori principle: All subsets of a frequent itemset must also be frequent

3. Apriori algorithm: Pruning to efficiently get all the frequent itemsets
4. Maximal frequent itemset: none of the immediate supersets are frequent
5. Closed frequent itemset: none of the immediate supersets have the same value of support

I hope you enjoyed reading this and have more clarity in thoughts than before. If you are interested in reading more details, refer this resource. Let me know of your thoughts/questions through the comments.

Do read the introduction in first blog here if you haven't already!

Data Science Machine Learning Concept Analytics Analysis