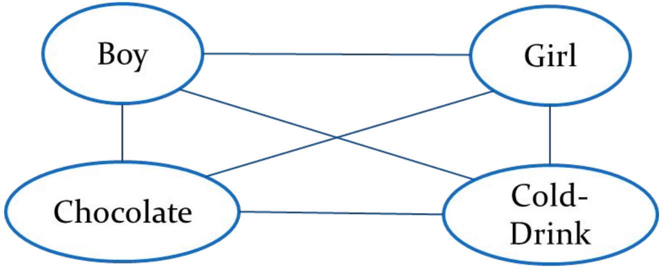
Association Rules Learning

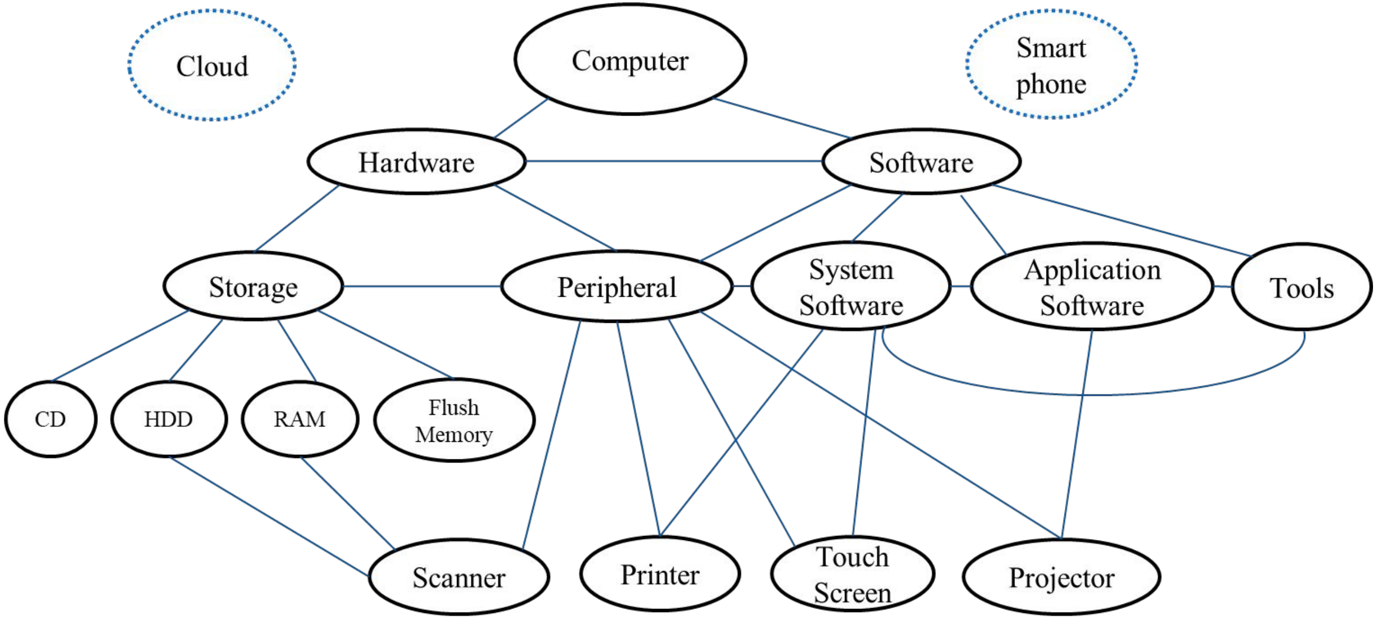
Prof. (Dr.) Honey Sharma

Need and Application of Association Rules Learning

The word “association” is very common in our every sphere of life. The literal meaning of association is a spatial or temporal relation between things, attributes, occurrences etc. In fact, the word “relation” is synonymous to the word “association”.

To understand the concept of association, let us look at below figure. Here, we see four things namely “boy”, “girl”, “chocolate” and “cold-drink”. Several associations from one (or more) element(s) to other(s) can be interpreted. For example, girl likes chocolate (interpreting girl prefers chocolate) or {boy, girl} shops {cold-drink, chocolate} (interpreting in college canteen, there is a large sales of cold-drinks and chocolates) etc.





Association among moderately large collection of things

Association rule mining is to **derive all logical dependencies among different attributes given a set of entities.**

Let us discuss the MBA problem in the following:

Market-Basket Analysis: The concept of the analysis of supermarket. Here, given a set of transactions of items the task is to find relationships between the presences of various items. In other words, the problem is to analyze customers buying habits by finding associations between the different items that customers place in their shopping baskets. Hence, it is called Market-Basket.

The discovery of such association rules can help the supermarket owner to develop marketing strategies by gaining insight into matters like “which items are most frequently purchased by customers”. It also helps in inventory management, sale promotion strategies, supply-chain management, etc.

Suppose, a set of customer purchase data are collected from a grocery stores. Table shows a small sample. From this data, the store owner is interested to learn about the purchasing behavior of their customers. That is, which items are occurring more frequent.

For example following two are more frequent.

- Bread-milk
- Diaper-chips

Basket	**Items**
1	bread, milk, diaper, cola
2	bread, diaper, chips, egg
3	milk, diaper, chips, cola
4	bread, milk, tea
5	bread, milk, diaper, chips
6	milk, tea, sugar, diaper

We can say that two rules {bread, milk} and {chips, diaper}, or more precisely, two association rules suggesting a strong relationships between the sales of bread and milk, and chips, diaper exist.
The process of analyzing customer buying habits by finding association between different items can be extended to many application domains like medical diagnosis, web mining, text mining, bioinformatics, scientific data analysis, insurance schemas, etc.

One important thing to note is-

Association Rules do not extract an individual’s preference, rather find relationships between set of elements of every distinct transaction.

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.

Applications of Association Rule Learning

It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:

- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the Catalog Design and Loss-leader Analysis and many more other applications.

Advantages and Drawbacks of the Association Rule

Advantages

1. This method is most descriptive, and can become predictive once we fix the consequent.
2. It is valid without any theoretical limit on the nature of the analyzed discrete data.
3. It offers a wide range of possibility; for example, we can add temporal, personal data, etc.
4. The resulting rules are very easy to understand.

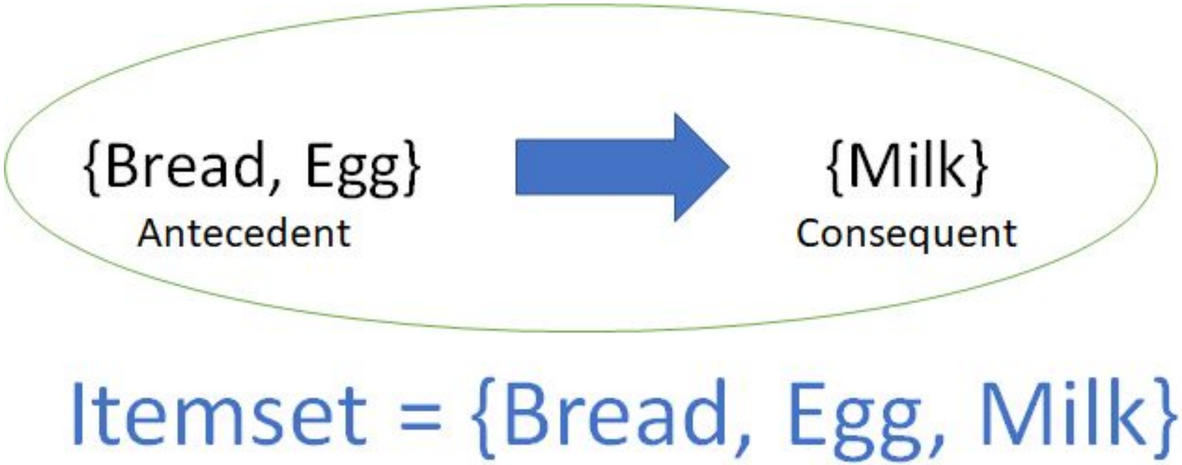
Drawbacks

1. The processing of huge amount of data (number of transactions or/and items) is very time-consuming.
2. It is not suitable for processing continuous variables.
3. This method only provides local rules, which do not give a global vision of a phenomenon, with emphasis on the preponderant factors and the interactions between factors.

Basic concepts of Association Rule Mining

Association rule is a rule-based machine learning method, used to deploy pattern recognition in order to identify relationships between different, yet related items. It has been used since the 1990s in the retail industry to help analyze products that are simultaneously purchased by customers. This can help store managers in finding better strategies for product placements, product discounts, stock management, etc. Of course, other industries can benefit from this technique.

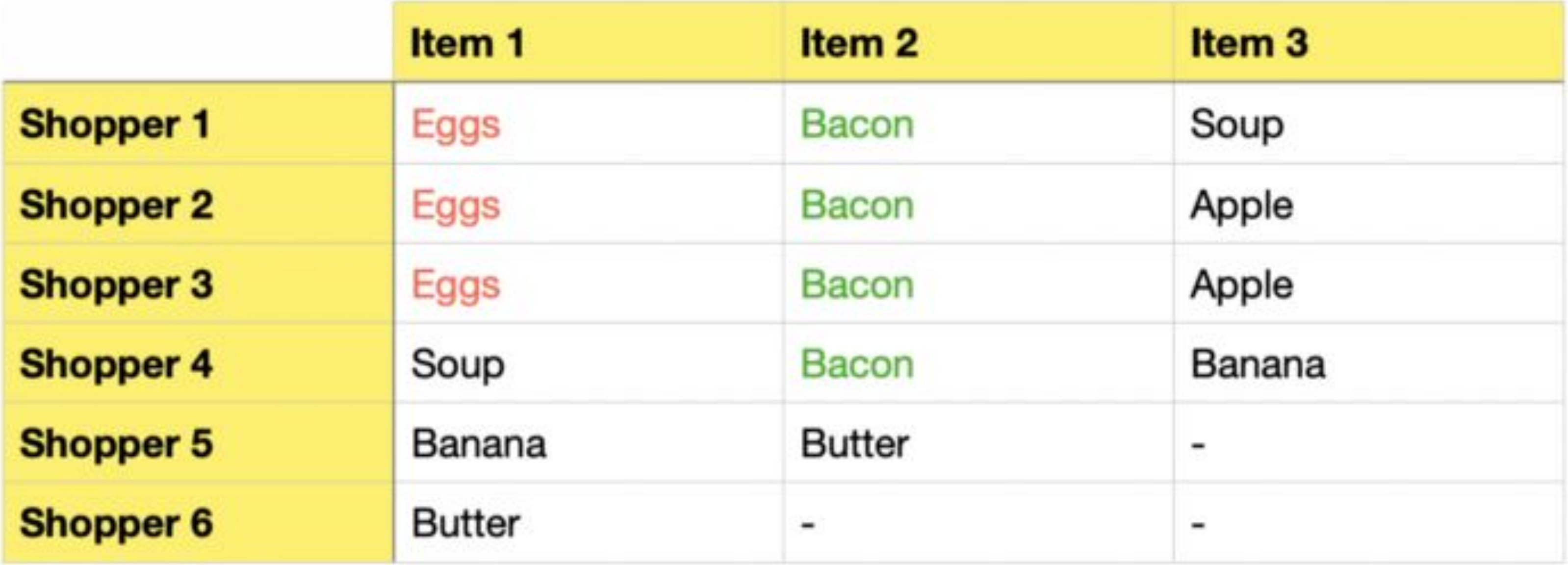
It consists of an antecedent and a consequent, both of which are a list of items. Note that implication here is co-occurrence and not causality. For a given rule, itemset is the list of all the items in the antecedent and the consequent.



{Bread, Egg}
Antecedent

{Milk}
Consequent

Itemset = {Bread, Egg, Milk}



	Item 1	Item 2	Item 3
Shopper 1	Eggs	Bacon	Soup
Shopper 2	Eggs	Bacon	Apple
Shopper 3	Eggs	Bacon	Apple
Shopper 4	Soup	Bacon	Banana
Shopper 5	Banana	Butter	-
Shopper 6	Butter	-	-

Support

This measure gives an idea of how frequent an itemset is in all the transactions. It represents the popularity of the items, also defined by its proportion of the total items sold.

Mathematically, support is the fraction of the total number of transactions in which the itemset occurs.

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

Support(Eggs) = 3/6 = 1/2 = 0.5
Support(Bacon) = 4/6 = 2/3 = 0.667
Support(Apple) = 2/6 = 1/3 = 0.333
...
Support(Eggs & Bacon) = 3/6 = 0.5

Here we can set our first constraint by telling the algorithm the minimum support level we want to explore, which is useful when working with large datasets. **We typically want to focus computing resources to search for associations between frequently bought items while discounting the infrequent ones.**

For the sake of our example, let’s set minimum support to 0.5, which leaves us to work with Eggs and Bacon for the rest of this example.

Important: while Support(Eggs) and Support(Bacon) individually satisfy our minimum support constraint, it is crucial to understand that we also need the combination of them (Eggs&Bacon) to pass this constraint. Otherwise, we would not have a single item pairing to progress forward to create association rules.

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?

Technically, confidence is the conditional probability of occurrence of consequent given the antecedent. Please beware of the notation. The above two equations are equivalent, although the notations are in different order: (A→B) is the same as (B|A).

$$Confidence(\{X\} \rightarrow \{Y\}) = \frac{Support(\{X\}\rightarrow\{Y\})}{Support\{X\}}$$

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

Confidence(Eggs→Bacon) = N(Eggs & Bacon) / N(Eggs) = (3) / (3) = 1

Confidence(Bacon→Eggs) = N(Eggs & Bacon) / N(Bacon) = (3) / (4) = 3/4 = 0.75

The above tells us that whenever eggs are bought, bacon is also bought 100% of the time. Also, whenever bacon is bought, eggs are bought 75% of the time.

Lift

Lift controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}. 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}→{Y}) = Probability({X} & {Y}) / (Support({X}) * Support({Y}))

Lift({X}→{Y}) = Confidence({X} & {Y}) / Support({Y})

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

Lift(Eggs→Bacon) = Confidence(Eggs→Bacon) / Support(Bacon) = 1 / (2/3) = 1.5

Lift(Bacon→Eggs) = Confidence(Bacon→Eggs) / Support(Eggs) = (3/4) / (1/2) = 1.5

First think about the probability of eggs being bought: P(Eggs)=Support(Eggs)=0.5.

Now think about the probability of eggs being bought whenever bacon was bought: P(Eggs|Bacon)=Confidence(Bacon->Eggs)=0.75.

Now, lift is simply a measure that tells us whether the probability of buying eggs increases or decreases given the purchase of bacon. Since the probability of buying eggs in such a scenario goes up from 0.5 to 0.75, we see a positive lift of 1.5 times (0.75/0.5=1.5). This means you are 1.5 times (i.e., 50%) more likely to buy eggs if you have already put bacon into your basket.

- If Lift score < 1, it means that if X is purchased, it is unlikely that Y will be purchased.
- If Lift score > 1, it means that X is highly associated with Y. In other words, if X is purchased, it is likely that Y will be purchased.
- If Lift score = 1, it means that there is no association between X and Y.

Conviction

Conviction is another way of measuring association, It compares the probability that X appears without Y if they were independent with the actual frequency of the appearance of X without Y. Let’s take a look at the general formula first:

Conviction({X}→{Y}) = (1 - Support({Y})) / (1 - Confidence({X}→{Y}))

Conviction(Eggs→Bacon) = (1 - Sup(Bacon) / (1 - Conf(Eggs→Bacon))

= (1 - 2/3) / (1 - 1) = (1/3) / 0 = infinity

Conviction(Bacon→Eggs) = (1 - Sup(Eggs) / (1 - Conf(Bacon→Eggs))

= (1 - 1/2) / (1 - 3/4) = (1/2) / (1/4) = 2

As you can see, we had a division by 0 when calculating conviction for (Eggs→Bacon) and this is because we do not have a single instance of eggs being bought without bacon (confidence=100%).

In general, high confidence for X→Y with low support for item Y would yield a high conviction.

In contrast to lift, conviction is a directed measure. Hence, while lift is the same for both (Eggs→Bacon) and (Bacon→Eggs), conviction is different between the two, with Conv(Eggs→Bacon) being much higher. Thus, you can use conviction to evaluate the directional relationship between your items.

Finally, similar to lift, conviction=1 means that items are not associated, while conviction>1 indicates the relationship between the items (the higher the value, the stronger the relationship).

Naïve approach to frequent itemsets generation

We may note that the cardinality of an itemset (corresponding to a rule X→Y) is at least two.

Thus, first we are to

1. generate all possible itemsets of cardinality two or more
2. check which of them are supported.

The first task, is therefore, finding all possible subsets. Thus, the number of itemsets with cardinality at least two is 2^m-m-1.

For a practical application, this can be very large. For example, if m=20, then 2^20-20-1=1,048,555. If m takes more realistic but still relatively small value of 100, then the total number of itemsets is 2^100-100-1≈10^30 !

Thus, generating all the possible itemsets and then checking against the transactions in the database to establish which ones are supported is clearly unrealistic or impossible in practice.

Once, the candidate itemset is generated, the next task is to count the occurences of each itemset. To do this using naïve approach, we need to compare each candidate against every transaction .

We may note that we can save computations if we ignore the candidate itemsets, which are not relevant to the input data set and further reduce the second stage computation by eliminating the candidate itemstes, which are not prospective.

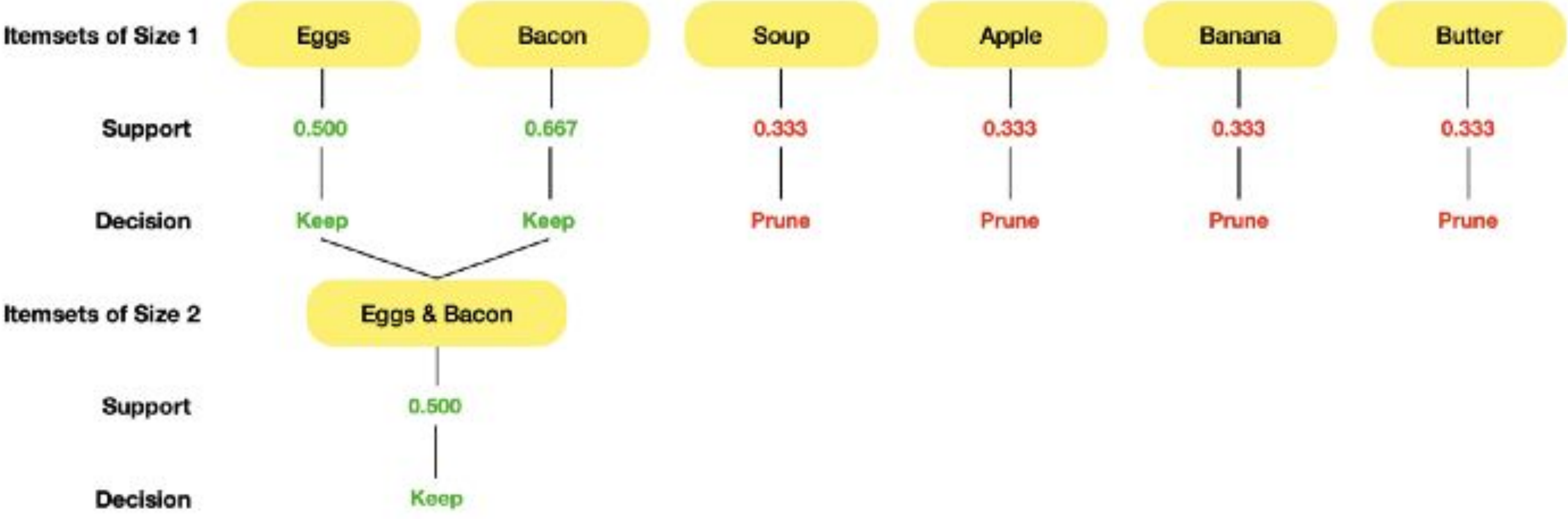
The first breakthrough algorithm in this context in Apriori algorithm, which is our next topic of discussion.

Apriori algorithm

As stated earlier, Apriori is part of the association rule learning algorithms, which sit under the unsupervised branch of Machine Learning. Apriori does not require us to provide a target variable for the model. Instead, the algorithm identifies relationships between data points subject to our specified constraints.

Apriori is a pretty straightforward algorithm that performs the following sequence of calculations:

1. Calculate support for itemsets of size 1.
2. Apply the minimum support threshold and prune itemsets that do not meet the threshold.
3. Move on to itemsets of size 2 and repeat steps one and two.
4. Continue the same process until no additional itemsets satisfying the minimum threshold can be found.



Itemsets of Size 1

Eggs — Support 0.500 — Decision Keep

Bacon — Support 0.667 — Decision Keep

Soup — Support 0.333 — Decision Prune

Apple — Support 0.333 — Decision Prune

Banana — Support 0.333 — Decision Prune

Butter — Support 0.333 — Decision Prune

Itemsets of Size 2

Eggs & Bacon — Support 0.500 — Decision Keep

Itemsets of Size 3 No itemset of size 3 can be formed since only eggs and bacon remain not pruned.

Note, minimum support threshold in this illustration is 0.5

As you can see, most itemsets in this example got pruned since they did not meet the minimum support threshold of 0.5.

It is important to realize that by setting a lower minimum support threshold we would produce many more itemsets of size 2. To be precise, with a minimum support threshold of 0.3, none of the itemsets of size 1 would get pruned giving us a total of 15 itemsets of size 2 (5+4+3+2+1=15).

The apriori property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will also fail the same test. For example, if {a,b} is infrequent, then all of its supersets must be infrequent too . From this, we can write the following.

Let us denote L_k be the set containing all the frequent k-itemsets.

If L_k=Φ (the empty set), then L_(k+1),L_(k+2),…etc. must also be empty.

It generates the frequent itemsets in ascending order of their cardinality. That is, it starts with all those with one element (L_1) first, then all those with two elements (L_2), then all those with three elements (L_3) and so on. In other words, at a k-th stage, the set L_k of frequent k-itemsets is generated from the previous L_(k-1) set. At any stage, if L_k is Φ, the empty set, we know that L_(k+1),L_(k+2), etc. must also be empty. Thus, itemsets of cardinality k+1 or higher do not need to be generated and hence tested as they are certain to turn out not to be frequent.

Apriori algorithm follows a method of going from each set L_(k-1) to the next L_k in turn. This is done with two steps:

1.Generate a candidate set (containing k-itemsets) from L_(k-1);

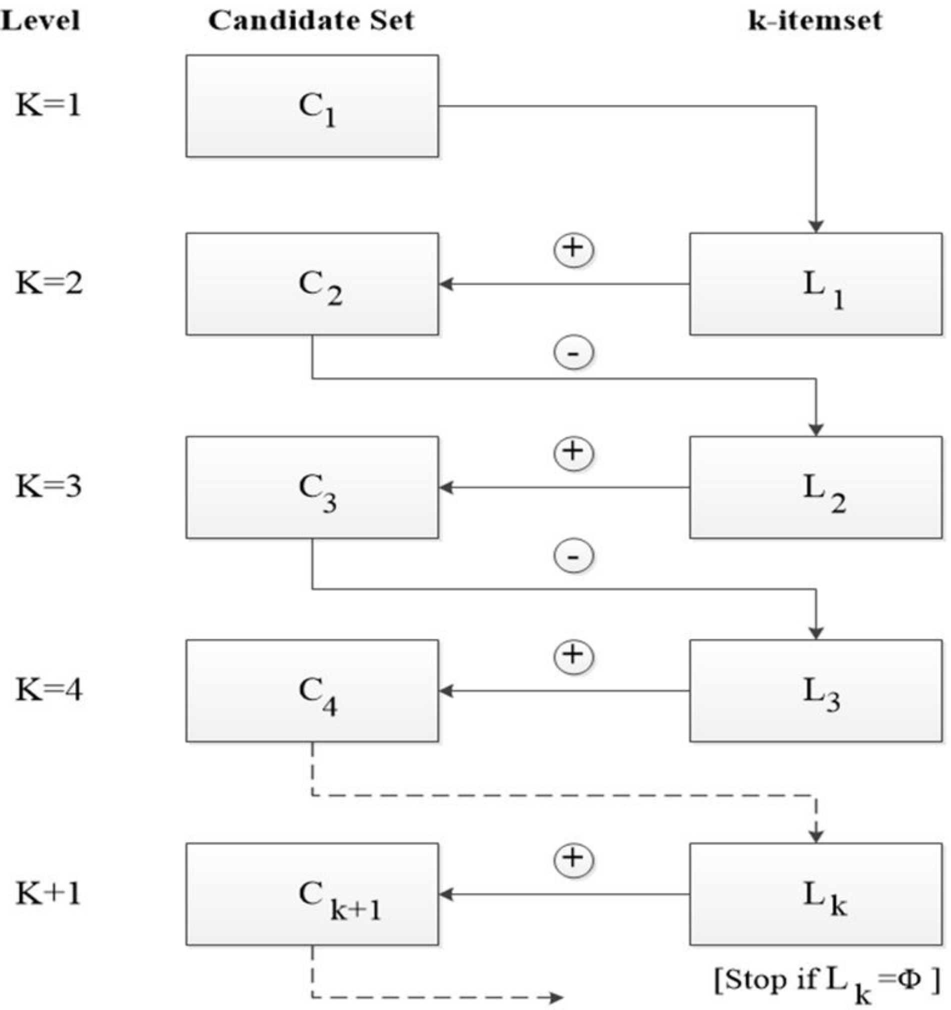
2.Prune C_k to eliminate those k-itemsets, which are not frequent.

Finally, when L_k=Φ, the algorithm terminates generating all frequent itemsets as L_1∪L_2∪…∪L_(k-1).

The Step 1 is popularly called Apriori-gen procedure and Step 2 is the Prune procedure.

L_k : the set of all large k-itemsets in the Database.

C_k : a set of candidate large k-itemsets. In the algorithm we will look at, it generates this set, which contains all the k-itemsets that might be large, and then eventually generates the set above.



⊕ Create (k+1)-itemsets from k-itemsets in L_k $(k = 1,2, \dots.)$ $C_{k+1} = Apriori - gen(L_k)$

⊖ Prune all infrequent itemsets $Prune\ (C_k)$

ID	a	b	c	d	e	f	g
1	1	1		1			1
2			1	1	1		
3		1	1			1	
4		1				1	
5			1		1		1
6						1	
7	1		1	1			
8						1	
9			1		1		
10		1					1

ID	a	b	c	d	e	f	g
11			1		1		1
12	1						
13			1			1	
14	1		1	1		1	
15							
16				1			
17	1		1			1	
18	1	1	1	1			
19	1	1	1	1			1
20					1		

We will assume this is our transaction database D and we will assume support 20% i.e. minimum 4 frequency.

a	b	c	d	e	f	g
7	6	11	7	5	7	5

First we find all the 1-itemsets. I.e., in this case, all the 1-itemsets that are contained by at least 4 records in the Database.

In this example, that's all of them. So, L1 = {{a}, {b}, {c}, {d}, {e}, {f}, {g}}

Now we set k = 2 and run apriori-gen to generate C2

The join step when k=2 just gives us the set of all alphabetically ordered pairs from L1, and we cannot prune any away, so we have

C2 = {{a, b}, {a, c}, {a, d}, {a, e}, {a, f}, {a, g}, {b, c}, {b, d}, {b, e}, {b, f}, {b, g}, {c, d}, {c, e}, {c, f}, {c, g}, {d, e}, {d, f}, {d, g}, {e, f}, {e, g}, {f, g}}

{a, b}	{a, c}	{a, d}	{a, e}	{a, f}	{a, g}	{b, c}	{b, d}	{b, e}	{b, f}	{b, g}	{c, d}	{c, e}	{c, f}	{c, g}	{d, e}	{d, f}	{d, g}	{e, f}	{e, g}	{f, g}
3	5	5	0	2	2	3	3	0	2	3	5	4	4	3	1	1	2	0	2	0

So, L2 = {{a, c}, {a, d}, {c, d}, {c, e}, {c, f}}

We now set k = 3 and run apriori-gen on L2 and get {{a,c,d}, {a,c,e}, {a,c,f}, {c,d,e}, {c,d,f}, {d,e,f}}

The join step finds the following pairs that meet the required pattern: {a, c}:{a, d} {c, d}:{c, e} {c, d}:{c, f} {c, e}:{c, f}

This leads to the candidates 3-itemsets: {a, c, d}, {c, d, e}, {c, d, f}, {c, e, f}

We prune {c, d, e} since {d, e} is not in L2

We prune {c, d, f} since {d, f} is not in L2

We prune {c, e, f} since {e, f} is not in L2

We are left with C3 = {a, c, d}

How many records contain {a, c, d}. The count is 4, so L3 = {a, c, d}

We now set k = 4, but when we run apriori-gen on L3 we get the empty set, and hence eventually we find L4 = {}

This means we now finish, and return the set of all of the non-empty Ls:

Result = {{a}, {b}, {c}, {d}, {e}, {f}, {g}, {a, c}, {a, d}, {c, d}, {c, e}, {c, f}, {a, c, d}}

Each itemset is intrinsically interesting, and may be of business value.

Generate Association Rules: confidence of the following associations are

{a, c}->{d}=4/5* 100=80%

{a, d}->{c}=4/5* 100=80%

{c, d}->{a}=4/5* 100=80%

{d}->{a, c}=4/7* 100=57%

{c}->{a, d}=4/11* 100=36%

{a}->{c, d}=4/7* 100=57%

If our threshold confidence limit is 75% then above three will be accepted

Let's find the lift for the 03 cases (Lift({X}→{Y}) = Confidence({X} & {Y}) / Support({Y}))

{a, c}->{d}=(4/5) /(7/20)=2.28

{a, d}->{c}=(4/5) /(11/20)=1.45

{c, d}->{a}=(4/5) /(7/20)=2.28

In all the cases lift is greater than 01.