

Association Rule Mining

rules \Rightarrow association of items

{ bread } \rightarrow { butter }

milk \rightarrow toast





What are association rules?

- 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

↳ two parts

basket = transaction

↳ set of purchased products

Applications



- Finding the set of items that has significant impact on business
- Collection information from numerous transactions
- Generating rules from count in transactions ✚ ✚ ✚

↳ market basket analysis



Apriori

a prior knowledge of
data (products)

Overview

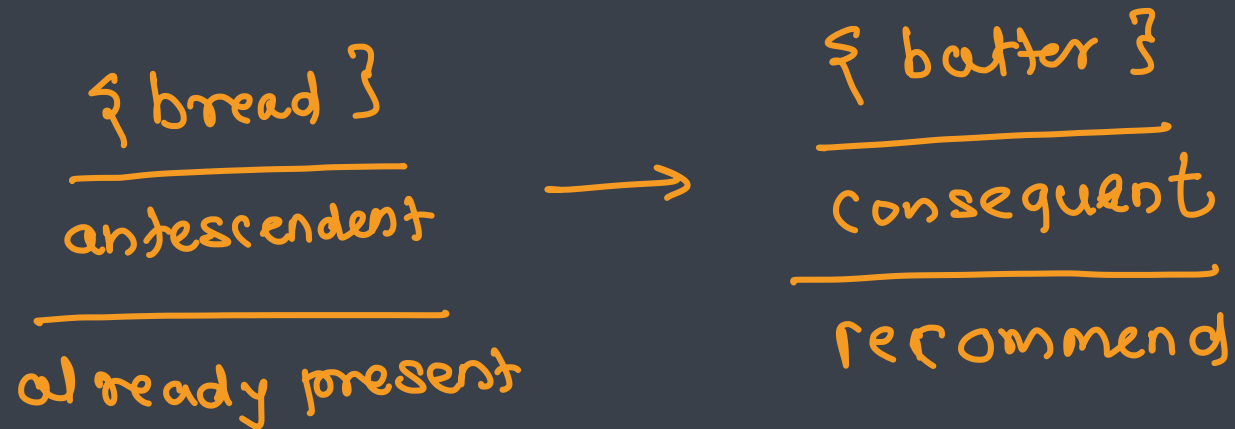


- Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule
- Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties
- We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets
- To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called *Apriori property* which helps by reducing the search space
- Apriori Property: All non-empty subset of frequent itemset must be frequent
- The key concept of Apriori algorithm is its anti-monotonicity of support measure

itemset = set of items
{ milk, bread, butter }

Terminology - Itemset

- It is a representation of the list of all items which form the association rule
- E.g.
 - Itemset = {Bread, Egg, Milk}





Terminology - Support

- This measure gives an idea of how frequent an *itemset* is in all the transactions
- E.g.
 - *itemset1* = {bread} and *itemset2* = {shampoo}
 - There will be far more transactions containing bread than those containing shampoo
 - So *itemset1* will generally have a higher support than *itemset2*
- E.g.
 - *itemset1* = {bread, butter} and *itemset2* = {bread, shampoo}
 - Many transactions will have both bread and butter on the cart but bread and shampoo are 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

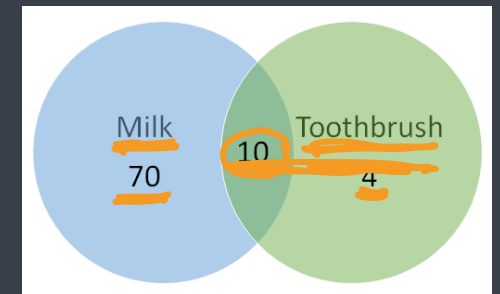
$$\text{Support}(\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both } X \text{ and } Y}{\text{Total number of transactions}}$$

Terminology - Confidence

- This measure defines the likeliness of occurrence of **consequent** on the cart given that the cart already has the **antecedents**
- Technically, confidence is the conditional probability of occurrence of consequent given the antecedent

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

- E.g.
 - Confidence for {Toothbrush} → {Milk} will be 10 / (10 + 4) = 0.7



Terminology - 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

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

Summary



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

Example

- Given the transactions generate rules using Apriori algorithm.
- Consider support = 50% and confidence = 75% ✓

Transaction Id	Items Purchased
1	Bread, Cheese, Egg, Juice
2	Bread, Cheese, Juice
3	Bread, Milk, Yogurt
4	Bread, Juice, Milk
5	Cheese, Juice, Milk

items	count	support
① Bread	4	$4/5 = 0.8$ ✓
② cheese	3	$3/5 = 0.6$ ✓
Egg	1	$1/5 = 0.2$ ✗
③ Juice	4	$4/5 = 0.8$ ✓
④ milk	3	$3/5 = 0.6$ ✓
Yogurt	1	$1/5 = 0.2$ ✗

itemset	count	support	confidence
bread \rightarrow cheese	2	$2/5 = 0.4$	
bread \rightarrow Juice	3	$3/5 = 0.6$ ✓	$\rightarrow 3/4 = 0.75$ ✓
bread \rightarrow milk	2	$2/5 = 0.4$	
cheese \rightarrow bread	2	$2/5 = 0.4$	
cheese \rightarrow Juice	3	$3/5 = 0.6$ ✓	$\rightarrow 3/3 = 1.0$ ✓
cheese \rightarrow milk	1	$1/5 = 0.2$	
Juice \rightarrow bread	3	$3/5 = 0.6$ ✓	$\rightarrow 3/4 = 0.75$ ✓
Juice \rightarrow cheese	3	$3/5 = 0.6$ ✓	$\rightarrow 3/4 = 0.75$ ✓
Juice \rightarrow milk	2	$2/5 = 0.4$	
milk \rightarrow bread	2	$2/5 = 0.4$	
milk \rightarrow cheese	1	$1/5 = 0.2$	
milk \rightarrow Juice	2	$2/5 = 0.4$	



Disadvantages

- It may need to generate a huge number of candidate sets
- It may need to repeatedly scan the database and check a large setoff candidates

Perform Apriori in R



```
library(arules)
```

```
transactions = read.transactions('Market_Basket_Optimisation.csv', rm.duplicates = TRUE, sep = ',')
```

```
itemFrequencyPlot(transactions, topN=10)
```

```
rules = apriori(transactions, parameter = list(confidence = 0.4, support = 0.04))
```

```
summary(rules)
```

```
inspect(rules)
```



FP-Growth

Overview



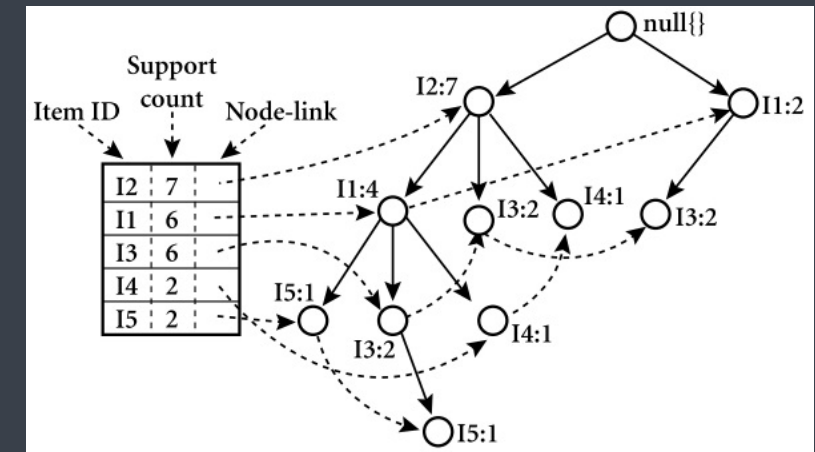
- Mining frequent itemsets without candidate generation
- The FP-Growth Algorithm, proposed by Han
- It is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree)
- In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm
- It has better performance than other methods

Steps



- Find frequent item sets without candidate generation
- Compress the database representing items into a frequent-pattern tree or FP-tree which retains the itemset association information
- Divide the compressed database into a set of conditional database, each associated with one frequent item or pattern fragment
- Mine each database separately

- The frequent-pattern tree (FP-tree) is a compact structure that stores quantitative information about frequent patterns in a database
- It contains
 - One root labelled as null with a set of item-prefix subtrees as children and frequent-item-header table
 - Each node in the item-prefix subtree consists of three fields
 - Item-name: registers which item is represented by the node
 - Count: the number of transactions represented by the portion of the path reaching the node;
 - Node-link: links to the next node in the FP-tree carrying the same item-name, or null if there is none.
- Each entry in the frequent-item-header table consists of two fields:
 - Item-name: as the same to the node;
 - Head of node-link: a pointer to the first node in the FP-tree carrying the item-name.



Example



- Generate FP tree for following data set

Id	Items
1	E, A, D, B
2	D, A, C, E, B
3	C, A, B, E
4	B, A, D
5	D
6	D, B
7	A, D, E
8	B, C