

Report: Apriori algorithm

Big Data Course

Author: Daniel Ciucur

Abstract

For the algorithms part I was interested in the problem of **market basket analysis**. Based on the purchasing behavior of previous customers, it attempts to identify associations between products. Later those associations can be used in many ways: store layout, catalog design, creating promotional offers and so on. I studied a scientific paper called *Fast algorithms for mining association rules* and I decided to implement the algorithm proposed there: Apriori.

Introduction

Progress in bar-code technology has made it possible for retail organizations to collect and store massive amounts of sales data, referred to as basket data. A record in such data typically consists of the transaction date and the items bought in the transaction. Successful organizations view such databases as important pieces of the marketing infrastructure. They are interested in instituting information-driven marketing processes, managed by database technology, that enables marketers to develop and implement customized marketing programs and strategies. An example of such rule might be that 98 percent of customers that purchase tires and auto accessories also get automotive services done. Finding all such rules is valuable for cross-marketing and attached mailing applications. Other applications include catalog design, add-on sales, store layout and customer segmentation based on buying patterns. The database involved in there applications are very large. It is imperative, therefore, to have fast algorithms for this task.

Nowadays in the era of Big Data using a sequential version of algorithms might be too slow given the huge amount of data we have. Therefore, I believe that there is a need for both fast algorithms (and useful data structures) and Big Data technologies such as Hadoop Map Reduce in order to successfully process such data and extract information from it.

Algorithms

Apriori algorithm is based in the Apriori principle which states:

“If an itemset is frequent, then all of its subsets must also be frequent. Conversely if an itemset is infrequent then all of its supersets must be infrequent too.”

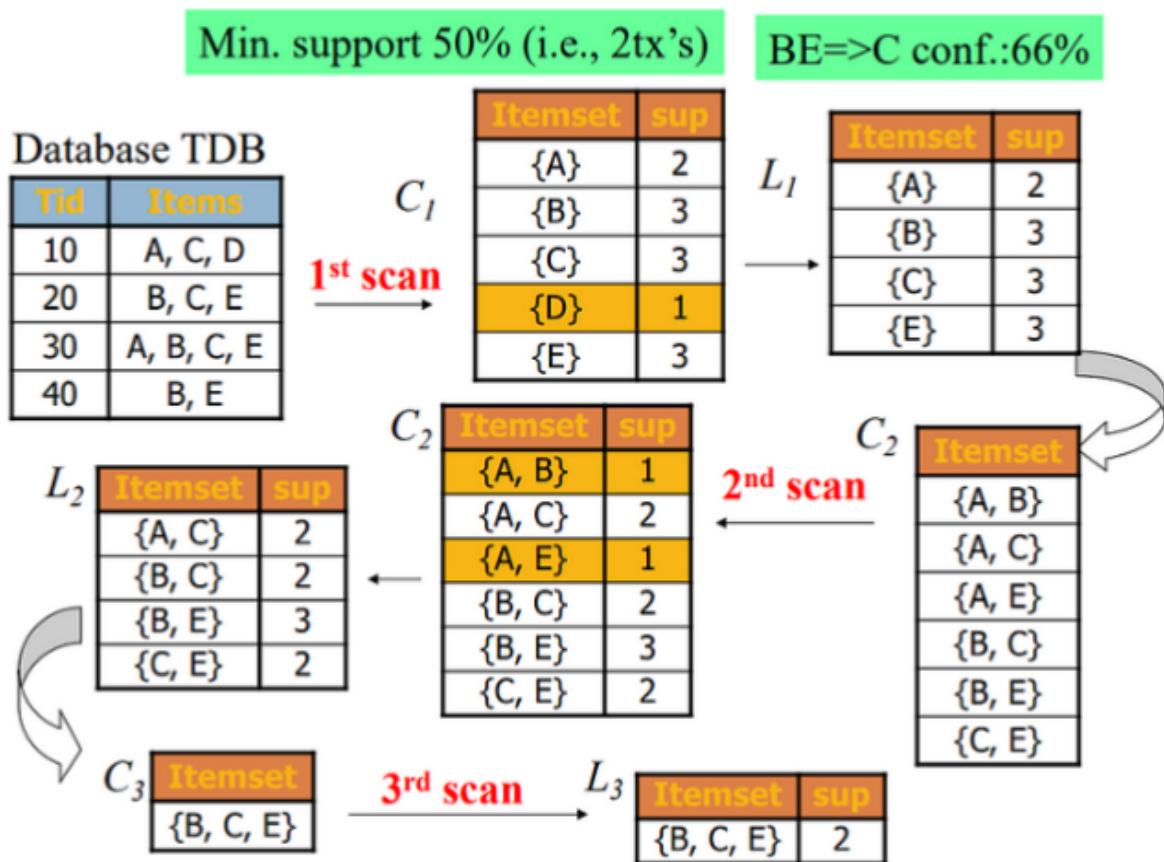
```
1)  $L_1 = \{\text{large 1-itemsets}\};$ 
2) for (  $k = 2; L_{k-1} \neq \emptyset; k++$  ) do begin
3)    $C_k = \text{apriori-gen}(L_{k-1});$  // New candidates
4)   forall transactions  $t \in \mathcal{D}$  do begin
5)      $C_t = \text{subset}(C_k, t);$  // Candidates contained in  $t$ 
6)     forall candidates  $c \in C_t$  do
7)        $c.\text{count}++;$ 
8)   end
9)    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$ 
10) end
11)  $\text{Answer} = \bigcup_k L_k;$ 
```

Figure: Apriori pseudo-code

Apriori works on transaction data and it's provided with a support value. It will first generate the 1-itemsets which are frequent. An itemset is frequent if the support value of that itemset is greater or equal than the minimum support threshold.

$$\text{Support}(x) = \frac{\text{Number of transactions containig X}}{\text{Total number of transactions}}$$

Next it will generate new candidates by joining the previous k-itemsets. Afterwards those candidate sets are pruned. That is according to apriori principle if any of their subsets is infrequent then the candidate can be pruned/removed. Next for the candidates that remain after pruning, we go through the transactions in order to calculate their support and we keep the itemsets whose support is greater or equal than the support threshold. The process goes on until we cannot generate new candidates.



In the figure we have a visual representation of the algorithm. Note: Confidence is not required by apriori, that is used after the frequent itemsets were generated in order to create the association rules.

Input: transaction file/database












Output: frequent itemsets

The output tells us that the items in a frequent itemset tend to always be together in transactions. That can be very beneficial for markets as they can use the information to make better discounts or offers, or even to arrange the aisles better in order to make more profits.

Implementation

For the implementation I wanted to try to use Docker-compose to be able to create the Hadoop Stack on my machine. I found this repository: [big-data-europe/docker-hadoop: Apache Hadoop](https://github.com/big-data-europe/docker-hadoop)

[docker image \(github.com\)](#) which has a Docker-compose file exactly for that purpose. With some minor adjustments, I managed to set up that environment.

<input type="checkbox"/>	▼  docker-hadoop	Running (5/5)	36 seconds ago	■	⋮	🗑
<input type="checkbox"/>	 historyserver d7d8e93ad8c2 	bde2020/hadoop-historyserve Running	37 seconds ago	■	⋮	🗑
<input type="checkbox"/>	 datanode cbef3371ac45 	bde2020/hadoop-datanode:2. Running	37 seconds ago	■	⋮	🗑
<input type="checkbox"/>	 nodemanager daa4ce5b4582 	bde2020/hadoop-nodemana Running	37 seconds ago	■	⋮	🗑
<input type="checkbox"/>	 namenode da3f5e2eb0cf 	bde2020/hadoop-namenode:2 Running	36 seconds ago	■	⋮	🗑
<input type="checkbox"/>	 resourcemanager 8ac8bb47ab96 	bde2020/hadoop-resourcema Running	36 seconds ago	■	⋮	🗑

Data set

The dataset I used is a dataset which contains transactions (exactly what's needed for Apriori to work on). Each line of the dataset represents a transaction. In total there are 7501 lines/transactions. Items in a transaction are separated by comma. Each transaction has a different number of items.

Example:

burgers,meatballs,eggs

chutney

turkey,avocado

mineral water,milk,energy bar,whole wheat rice,green tea

low fat yogurt

whole wheat pasta,french fries

soup,light cream,shallot

frozen vegetables,spaghetti,green tea

french fries

Experiments

I made 2 experiments with the algorithm. First was the sequential implementation which I made in python. For that approach I used a smaller dataset but I went further with the algorithm and I also generated the association rules.

Some example outputs from that look as follows:

{'milk'} -> {'bread'} <confidence: 0.8>

{'bournvita'} -> {'bread'} <confidence: 0.7499999999999999>

{'jam'} -> {'bread'} <confidence: 1.0>

{'maggi'} -> {'bread'} <confidence: 0.6>

{'coffee', 'biscuit'} -> {'apple'} <confidence: 1.0>

{'apple', 'biscuit'} -> {'coffee'} <confidence: 1.0>

{'coffee', 'apple'} -> {'biscuit'} <confidence: 0.6666666666666667>

{'cornflakes', 'maggi'} -> {'tea'} <confidence: 1.0>

{'tea', 'coffee'} -> {'cornflakes'} <confidence: 1.0>

{'apple'} -> {'cornflakes', 'coffee'} <confidence: 0.6666666666666667>

The second experiment was using Hadoop Map Reduce.

Results

The algorithm gave the desired results (as seen above). However after researching some other papers many people proposed improved versions of Apriori. Most of them use some advanced data structures in order to reduce the time wasted on going through transactions to count support for each itemset.

Conclusion

As a conclusion I think I managed to gain some insights into the topics of market basket analysis. I also managed to get familiar with one of the algorithms that helps with this problem,

namely the Apriori algorithm, first presented by Rakesh Agrawal and Ramakrishnan Srikan in the paper “**Fast Algorithms for Mining Association Rules**”. Besides that I learned how to use Hadoop, what steps are involved in setting up a Hadoop cluster and how to tackle issues along the way.