A Python Implementation Of Hyperclique Pattern Mining

Vineet Joshi - MT19020

Original Paper Link: https://www-users.cs.umn.edu/~kumar001/papers/clique_dmkd.pdf

Introduction:

Through this paper, the authors introduced the concept of *Hyperclique Patterns* and how they are different and more effective than the vanilla frequent itemsets.
Frequent Itemsets Mining share two problems on it's own.

- If the minimum support count is too high, we are going to miss important frequent patterns.
- If the minimum support count is too low, then we are going to get groups of items (itemsets) which are of no use together.

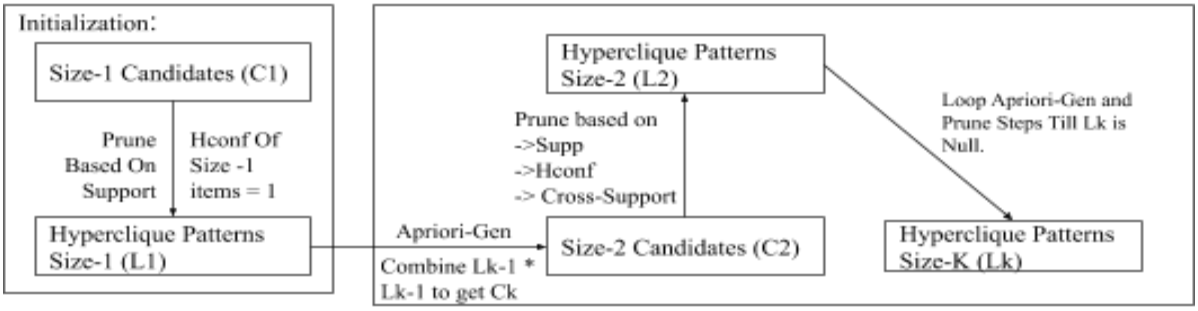
The authors tackle these problems by introducing a new measure called ***hconfidence***.
hconfidence shows the following properties which makes it a highly useful parameter for pattern mining.

- **Anti-Monotone Property:** If an itemset have an hconfidence value less than a threshold, then all it's supersets will have hconfidence values less than threshold.
- **Cross-Support Property:** A cross-support pattern with respect to a threshold t is guaranteed to have an hconfidence value greater than t.
- An itemset with hconfidence greater than minimum threshold will always have a level cosine similarity between items, that will be useful to us.

A **hyperclique pattern** is one which is frequent above a set threshold but also shows a global pairwise similarity between items.Thus using both *support* and *hconfidence* as mining parameters.

Hyperclique Miner Algorithm:

The authors proposed the following algorithms which was able to mine the hyperclique patterns from the datasets.



HyperClique Mining Algorithm

The Hyperclique mining algorithm builds on top of apriori algorithm with only major changes done in the pruning step. The pruning step in apriori considers only the support as its goal is to find only the frequent patterns, whereas the pruning step in Hyperclique Pattern also considers the hconfidence measure for the anti monotone and cross-support property.

NOTE: All the formulae used for this algorithm are directly taken from the paper and hence not mentioned here.

Dataset:

In order to experiment on the implemented algorithm, we used several trivial transaction datasets. These helped us to check the correctness of the code and also helped in figuring out the entire flow of the code provided.

The major experiments were performed on the PUMSB dataset. The authors originally also used the same dataset

Link: http://fimi.uantwerpen.be/data/

Details about the dataset are given below.

Format: .dat file.
Multiple rows containing space separated values. Each row represents a transaction, whereas each space separated value represents an item.

Number Of Transactions: **49046**
Number Of Unique Items: **2113**

Sample Output:

After setting the minimum support threshold to 0.95 and minimum hconfidence threshold to 0.95 as well, our program gives the following output.

min_supp: 0.95 min_hconf: 0.95
HCP Of Size: 1
[170]
.
[7062]
[7092]

HCP Of Size: 2
[170, 180]
.
[170, 4428]

[4432, 170]

HCP Of Size: 3
[170, 180, 4426]
.
.
[7062, 4940, 4438]
[7062, 7092, 4438]

HCP Of Size: 4
[180, 4438, 170, 4428]
.
.
[4434, 4438, 7062, 184]
[4438, 7062, 4426, 4428]

HCP Of Size: 5
[170, 4434, 180, 4438, 184]
.
.
[4434, 180, 4438, 7062, 184]
[4428, 180, 7062, 4438, 184]

HCP Of Size: 6
[]

The output can either be the extracted patterns, as shown above or the comparative table of execution time and patterns found in for various pairs of support and hconfidence pairs.
The submitted code outputs the table as default, by just commenting few lines it can output the Hyperclique Patterns as well.
The lines that needs to be commented for the same are mentioned in the code itself.

Results :

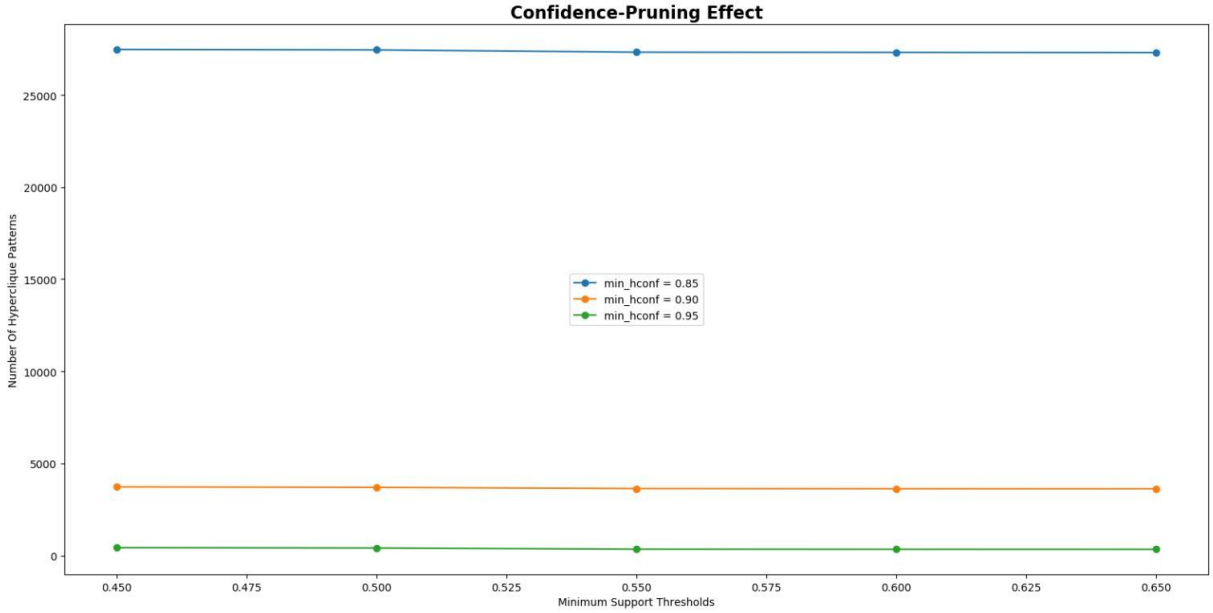
In this report we are trying mainly to study two major things:

- Pruning effect of hconfidence + support as compared to only support.
- Effect of hconfidence and support on execution time of the algorithm.
- Effect of data size on the execution time for the algorithm.

Let's see these graphs one by one.

Graph 1: The Confidence Pruning Effect.

The following graph shows us how pruning happens for various pairs of minimum support and minimum hconfidence.



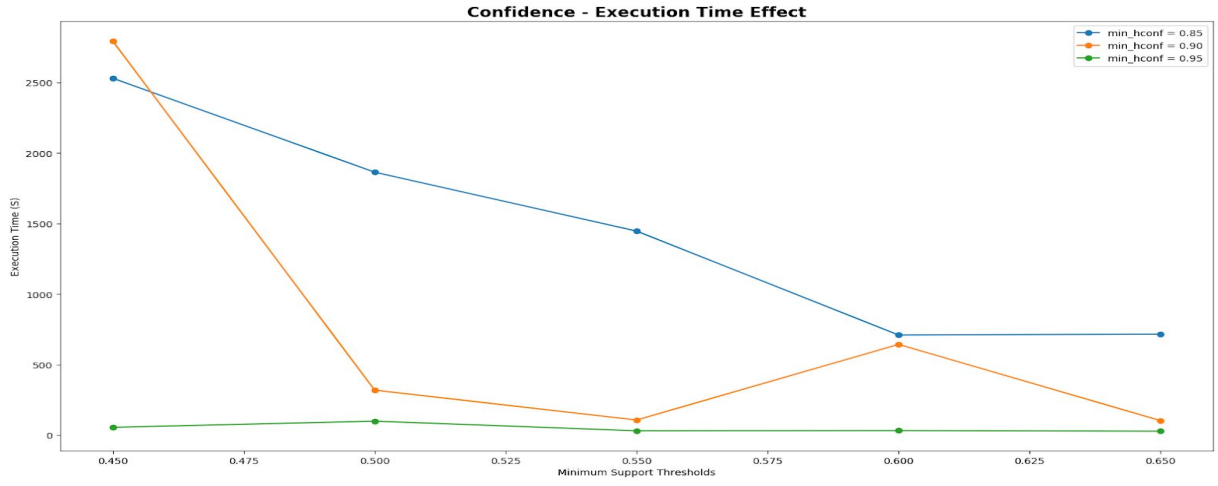
Inferences:

If we keep the value of hconfidence threshold constant, that is, study only one line graph from above 3, it can be clearly seen that for a real life dataset like pumsb, support has very less pruning effect. Even if we keep on increasing the support threshold, the total itemsets generated are changing with only a very small amount.

On the other hand, for any constant support value, changing the hconfidence value to even a small amount drastically reduced the itemsets generated.

Thus using both these measures ensures not only frequent itemset generation but also that the generated itemsets have some relevance together.

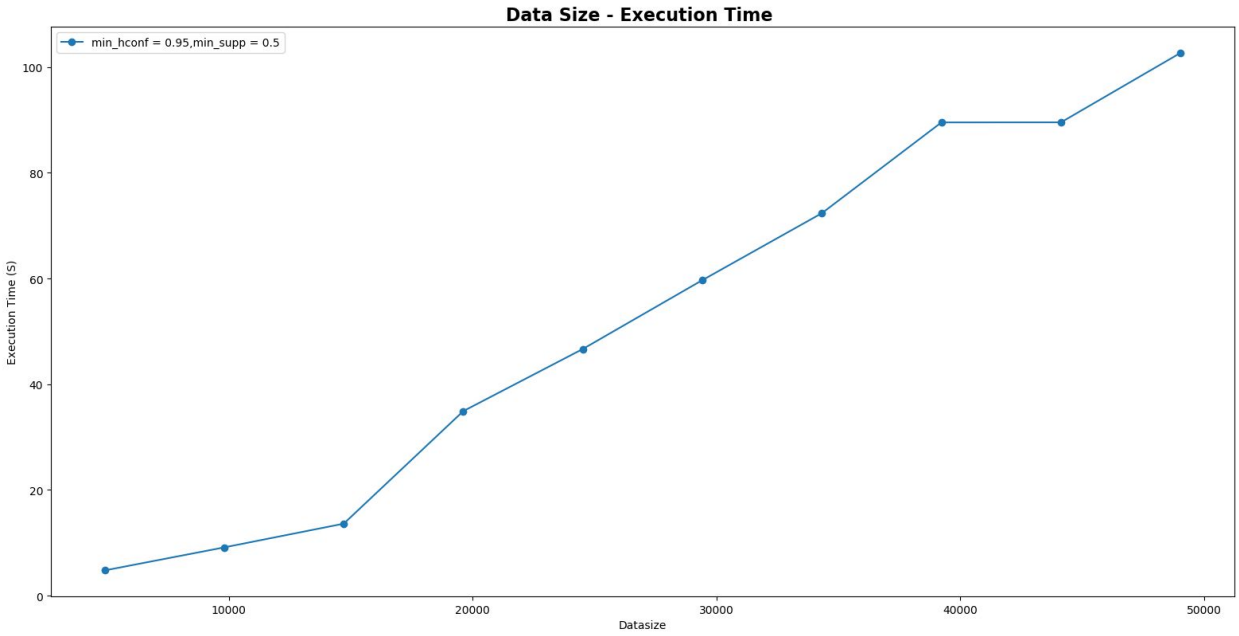
Graph 2: Effect Of Support & Confidence On Execution Time



Inferences:

The above graph shows the execution time of Hyperclique Miner Algorithm on the PUMSB dataset. As can be seen, with the increase in the hconfidence threshold the execution time reduces significantly.

Graph 3: Effect Of Data Size On Execution Time



Quite naturally, since the algorithm scans the database multiple times in order to get the support of each itemset, therefore it's time complexity is dependent on the data size.

This simple fact can be observed from the graph above. As the data size increases, the execution time increases as well.

CONCLUSION:

With this project, we were able to study the Apriori Algorithm by R.Agrawal and R.Srikant, as well as add over it to get Hyperclique Patterns, rather than mere frequent patterns.

We also studied the hyperparameters associated with Hyperclique Miner Algorithm (Hconfidence and Support) and study their effect on factors such execution time and their pruning effect.

At last we saw how our algorithm is dependent on the data size and the effect various data sizes have on the execution time of the algorithm.