# COMS 4995: ParPriori Proposal

Claire Liu (cl3944), Evan Li (el3078)

November 29, 2022

We plan to parallelize the apriori algorithm for association rule mining. Association rule mining is commonly used in the commerce space, where shoppers purchase a certain set of items in each transaction, to determine which sets of items are frequently purchased together. Association rule mining has typically been used to reveal surprising trends hidden in data.

A more general problem statement can be found in the paper Fast Algorithms for Mining Association Rules by Agrawal and Srikant.

> The following is a formal statement of the problem [4]: Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, called items. Let $\mathcal{D}$ be a set of transactions, where each transaction $T$ is a set of items such that $T \subseteq \mathcal{I}$. Associated with each transaction is a unique identifier, called its *TID*. We say that a transaction $T$ *contains* $X$, a set of some items in $\mathcal{I}$, if $X \subseteq T$. An *association rule* is an implication of the form $X \Longrightarrow Y$, where $X \subset \mathcal{I}$, $Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. The rule $X \Longrightarrow Y$ holds in the transaction set $\mathcal{D}$ with *confidence* $c$ if $c$% of transactions in $\mathcal{D}$ that contain $X$ also contain $Y$. The rule $X \Longrightarrow Y$ has *support* $s$ in the transaction set $\mathcal{D}$ if $s$% of transactions in $\mathcal{D}$ contain $X \cup Y$. Our rules are somewhat more general than in [4] in that we allow a consequent to have more than one item.

Figure 1: Section 1

Association rule mining relies on two specified thresholds: minimum support and minimum confidence. Another threshold commonly used, but not mentioned in the paper, is Lift. The Lift between two item sets A and B is defined as: Lift$(A \to B)$ = Confidence$(A \to B)$/Support(B). Lift$(A \to B)$ is meant to measure the increase in ratio of the sale of set B when set A is sold.

The apriori algorithm uses a bottom up approach to first find all large 1-item sets that satisfy a minimum support. Then, via a candidate generation algorithm, the large 2-item sets, 3-item sets, and so on are formed. As we form candidate sets, we also apply a pruning algorithm that removes all candidates that don't meet our minimum support. Once, we've finished generating frequent itemsets, we can form our strong association rules. After generating the association rules, we keep the ones that fulfill our minimum confidence.

Here is the pseudo-code for the algorithm from Agrawal and Srikant (Section 2.1):

```
L_1 = {large 1-itemsets};
for ( k = 2; L_{k-1} ≠ ∅; k++ ) do begin
    C_k = apriori-gen(L_{k-1}); // New candidates
    forall transactions t ∈ D do begin
        C_t = subset(C_k, t); // Candidates contained in t
        forall candidates c ∈ C_t do
            c.count++;
    end
    L_k = {c ∈ C_k | c.count ≥ minsup}
end
Answer = ⋃_k L_k;
```

Figure 1: Algorithm Apriori

Figure 2: Section 2.1

One major limitation of the Apriori algorithm is that it is slow. Some points improvement from parallelization include the candidate generation algorithm, which can grow to an exponential complexity, and the process of filtering items that don't meet our minimum support.

We plan to implement a Haskell script that takes a .txt file that contains a list of transactions, a min support value, and a min confidence value. The script will then print a list of association rules that satisfy our constraints. We will measure our script's performance against datasets of varying sizes. Some examples of datasets we plan to run our algorithm on:

Leading causes of death (find associations between cause of death and race, gender, etc.):
https://dev.socrata.com/foundry/data.cityofnewyork.us/jb7j-dtam

Recipes (find ingredients that are commonly used together):
https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions
http://pic2recipe.csail.mit.edu/ (1 mi+ recipes)