Name: Isha Aggarwal
Date: 11.09.2021

Data Mining- Frequent Itemsets Mining

Abstract: This assignment is about implementing the FP Growth algorithm- it is one of the popular algorithms used to mine frequent itemsets from a given transactional database. An itemset is said to be frequent if its count is greater than or equal to the minimum support count.

Algorithm Pseudocode:

Frequent itemset list : stores pairs- itemsets along with support count

```
FPGrowth(Database , conditional items set)
{
        Create Header table using Database
        Create Reduced( and Ordered) Database using Header Table , Database (according to the min support
        count)
        Create FPTree using Reduced Database and Header Table //this also modifies the header table.

        If(FP tree is single path)
        {
                Enumerate Subsets with support counts(= min(set of count of each item node )) and take union of
                each subset with conditional items set

                Add all these to frequent itemsets list
        }
        Else
        {
                For row = last row to first row in header table
                {
                        conditional items set for the row = Union of conditional item set and singleton set having
                        the item corresponding to the row

                        Create conditional pattern base for the row using FPtree structure

                        Add the conditional items set to the  frequent itemsets list with support count equal to
                        count stored in header table corresponding to the row

                        FPGrowth(Conditional Database for the row, Conditional item sets for the row ) //recursive
                        call

                }
        }


}
```

Note: This is a recursive algorithm, in the beginning the database = given transactional database and conditional items set is empty. Recursion stops when a single path FPTree (that is only 1 leaf node) is found.

Some notes about the implementation:

- Implemented this recursive algorithm in C++
- Choice of memory layout/data structures to store given database, header table, reduced database, FP tree and other intermediate structures(used to perform some modular task/helper functions) etc. greatly affects time and space complexity
- The current implementation stores the FP tree in a way that each node has a side node link to adjacent node having the same item, and a link to its parent node. It does not have any link to its child nodes.
- Tested the current implementation with 6-7 test cases, all seem to work.
- The current implementation does not work with a large database especially when the minimum support count is low due to main memory limits.

Future Improvements:

- Space complexity:

- When the database is too large it will not fit in the main memory, also we create many structures like: reduced database, FPtree etc.. and since the algorithm is recursive, everything keeps accumulating on the main memory as the recursion progresses. This is a severe drawback of the current implementation, it will not work for larger datasets or medium datasets where the minimum support count is small.

- Time complexity**:**

- Finding ways to decrease the time complexity using a more appropriate choice of data structures and techniques to perform small modular tasks. Example: storing links to children as well in the FP tree in addition to the links to parent and side node, might provide some benefits.

References:

Class notes and book: Data Mining Concepts and Techniques, Jiawei Han