Modification in Apriori algorithm

We have modified the Apriori algorithm to implement the AprioriTID algorithm. The difference between the two algorithms is in the step of counting the support of the candidate frequent item sets. While the Apriori algorithm makes a pass over the dataset for each candidate, AprioriTID algorithm does not use the database for counting the support of candidates after the 1st pass. In AprioriTID algorithm we use a vector containing potentially frequent k-item sets with id = TID. If c is a candidate k-itemset such that the k-1 itemset obtained by removing the last entry and the k-1 itemset obtained by removing the second to last entry both belong to the potential k-1 frequent vector with id t, then c is a k frequent itemset.

Code for counting support in Apriori algorithm:

```
Ck_plus1_sup = defaultdict(int)
    for item in T1:
      for item_1 in pruned_Ck_plus1:
        if (set(item_1).issubset(set(item))):
          Ck_plus1_sup[tuple(item_1)] += 1
```

Code for counting support in AprioriTID algorithm:

```
for t in C[k]:
      C_t = []
      for item in pruned_Ck_plus1:
        if (sorted(list(item[:-1])) in t) and (sorted(list(item[:-2])+[item[-1]]) in t):
          C_t.append(sorted(item))
          C_t_sup[tuple(sorted(item))] += 1
      if (C_t != []):
        Ck_plus1_prime.append(C_t)
```

Therefore, we only make a single pass over the dataset in case of AprioriTID algorithm, resulting in good reduction in the overall space complexity.

One disadvantage of the AprioriTID algorithm is that for small values of k, the potential k frequent vector can be larger than the database itself.

CONCLUSION

Time taken by Apriori algorithm = 90.23s

Time taken by AprioriTID algorithm = 131.45s

Modification in FPtree algorithm

We have modified the FPtree algorithm to implement the projected database FPtree algorithm. The difference between the two algorithm is that in the case of FPtree algorithm we are using the complete database to construct the FPtree of our algorithm but in the case of projected database FPtree algorithm we are partitioning the database into several projected databases and then for each projected database we construct and mining the corresponding FPtree. It is because disk size of a computer is fixed and for large data it is unrealistic that FPtree algorithm will work in such a case.

x-Projected database: x-Projected database contains frequent items such that items with frequency less than x are not included and item x is also not included.

Thus, it leads to much better space optimization.

Code for projected database:

```
DBS=[]
RTable=sorted(Reference_Table.items(),key=lambda p: p[1][0], reverse=True)
for i in range(len(RTable)):
    element1 =RTable[i][0]
    PDB={}
    y=DataBase.copy()
    y.clear()
    y=DataBase.copy()
    for Set,freq in y.items():
        if element1 in Set:
            PDB[Set]=freq
            del(DataBase[Set])
    DBS.append(PDB)
```

CONCLUSION

Time taken by FPtree algorithm: 1.59s

Time taken by projected database FPtree algorithm:7.89s