Assignment 2: Apriori Rule Generation

TDT4300

Jonas Myrlund

February 28, 2014

1 Implementation

a) BruteForce

The BruteForce approach generates every possible permutation of the exhaustive item set, then prunes every subset that is deemed infrequent. This leads to an intractable amount of computation, as well as memory usage.

The candidate set generation works in the following fashion:

1. Grab all item sets of length 1.
2. Add the full union of the previous level's candidate sets and the item sets of length 1.
3. Repeat step 2 for each successive level until no new candidate sets are added.

Then the candidates are successively pruned.

b) FKMinus1F1Apriori

The FKMinus1F1Apriori approach is quite a bit better than BruteForce. For each level, it generates candidates from the previous level unioned with the most frequent level 1 item sets. Although it bases the new levels' candidate sets only on frequent components, it still has the weakness of generating quite a lot of infrequent candidate sets.

c) FkMinus1FKMinus1

It starts off with the F_1 candidates, and recursively joins the previous level onto itself for each candidate level, until no new frequent candidate sets can be found.

Method	Small dataset	Large dataset
BruteForce	63 candidates 54 pruned	N/A
FKMinus1F1Apriori	35 candidates 26 pruned	3800 candidates 3768 pruned
FkMinus1FKMinus1	16 candidates 7 pruned	305 candidates 273 pruned

Table 1: Candidate generation count and prune count for each algorithm.

2 Performance

The various complexities of the algorithms are shown side-by-side in table 1.

a) BruteForce

For the small dataset of $k = 6$ elements, BruteForce generates the expected $M = 2^d \approx 63$ candidate sets, not including the empty set. It then prunes 54 candidates to come up with the final 9 frequent item sets used for rule generation.

With a space-time complexity of $O(NMw)$, where $M = 2^d$, the space-time complexity renders the BruteForce method completely unusable on the large dataset.

b) FKMinus1F1Apriori

On the small dataset, FKMinus1F1Apriori generates 35 candidate sets and ends up pruning 26 of them to come up with the final 9 frequent item sets.

c) FkMinus1FKMinus1

The FkMinus1FKMinus1 approach is the clearly best performer.

The FkMinus1FKMinus1 generates very few candidates compared to the FKMinus1F1Apriori, and runs noticably faster on both the small and large dataset.