Efficient Techniques for

the VIPER Assn. Rule Mining Algorithm

Pradeep Shenoy
95005026

Under the guidance of
Dr. S. Sudarshan

Introduction

- Association rule mining: find related columns
- Reduced to finding *large* items/itemsets
- We use a vertical approach
  - itemsets represented by bitvectors
  - computation is by intersecting vectors

VIPER: Overview

- Level-wise algorithm (level = itemset length)
- Smart merge with multi-level counting
- Pruning strategy, and better candidate generation
- Deferred Writes

Generator Pruning

- Input: set of candidates of length $i + 1$
- Each candidate is the union of any two subsets of size i
  - e.g., $\{ABCD\} = \{ABC\} \cup \{ABD\} = \{ABD\} \cup \{ACD\}$, etc.
- These subsets are called generators for that candidate
- Several choices for subsets – as above
- Goal: make individual choices that help global sharing

Pruning: Greedy Algorithm

- For each i-large itemset, count number of candidates it is a subset of.

Until all candidates covered, do the foll.

- Select i-itemset with maximum count
- Mark all candidates it is a subset of
- Any candidate marked twice is *covered*
- Update counts – decrement all i-subsets of covered candidates

Smart Merge

- Input: list L of i-length items, list of pairs of L to be intersected
- Preferable: Amortize scans
  - e.g., {ABD} and {ABE} both to be generated using {AB}
- Approach: Keep an nxn array, run a simultaneous merge (similar to a n-way merge)
- Regenerate each tuple, count all pairs of i-itemsets

Smart Merge: More Details

- All pairs counted, hence longer candidates, of level $i+1$ through $2i$ also counted
- Implementation:
  - Tuple is actually merge of $i/2$-snakes
  - $i/2$-snake pairs collapsed to i-snakes
  - All pairs counted
- All i-snakes written to disk

Candidate Generation

- What it is: Use the anti-monotonicity principle
- Candidate Generation used before. Our contribution: an efficient algorithm
- Problems with old: repeated work, costly data structure
  - To check {ABCDE}. {ABCDF}, {ABCDG}, we need to examine {BCDE}, {BCDF}, {BCDG}
  - Apriori will initiate traversal of hashtree from the root

ClusterGen

- Group i-large itemsets by $i-1$-prefix
- Combinations inside a prefix group are sufficient
- Maintain all ith elements in an extension list
- Choose one element to *extend* the prefix
- Generate all $i-1$ subsets of extended prefix
- Intersect the extension lists of each subset

Results

- Experiment datasets: flatfiles, synthetic data from generator
- VIPER beats Apriori (horizontal approach) clearly, sometimes by order of magnitude
- Another vertical algorithm (Zaki et. al.), do better at lower supports
  - MaxEclat/MaxClique are in-memory algorithms
  - At lower supports there is more writing-to-disk
- We need to remove this memory-assumption for fair comparison

Future Work

- Study the MaxEclat and MaxClique algorithms – their memory dependence
- Introduce partitioning schemes that will partition generators
- Explore applications of these techniques in other settings