Algorithms for Association Rule Mining – A General Survey and Comparison

Jochen Hipp, Ulrich Guentzer and Gholamreza Nakhaeizadeh

Presented by: Aamir Shahzad (02030001) & M. Usman Ilyas (02030019)
Presentation Outline

- Terminology We Use
- Problem Statement
- Formal Problem Description (More Terminology)
- Basic Principles
- Search Space Traversal
- Search Algorithms
- Comparison
- Conclusions
Terminology (1)

- A **Database** consist of **Transactions**
- Transactions consist of **Itemsets** \((X, Y, Z \ldots)\)
- \(X \Rightarrow Y\) is an **Association Rule**
- Association rules have probabilities assigned to them called **Confidence Levels**
- The fraction of transactions to which an association rule applies is called **Support**
Problem Statement (1)

- # of sets of items > # of items
- # of (possible) association rules >> # of items
“How can we separate the useful association rules from the useless ones?”

… That’s what this paper is about
Formal Problem Description (1)

- Set of items $I = \{x_1, x_2, \ldots, x_n\}$
- Itemset $X \subseteq I$ with $k = |X|$
- Database $D$ is a multiset of subsets of $I$
- $T \in D$ is called a transaction
- $T$ “supports” $X \subseteq I$ if $X \subseteq T$
Formal Problem Description (2)

- Association Rule \( X \Rightarrow Y \)
  where \( X, Y \in I \) and \( X \cap Y = 0 \)

- Support: “The fraction of transactions containing \( X \) is the support of \( X \)”
  \[
  \text{supp}(X) = \frac{|\{T \in D \mid X \subseteq T\}|}{|D|}
  \]
Confidence: “The fraction of transactions supporting X for which the association rule defined on X (X \rightarrow Y) holds true”

\[ \text{conf}(X \rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} \]

The # of association rules grows exponentially with \(|I|\)
Basic Principles (1)

Filter out all association rules ...

1. That are Applicable to itemsets of supp less than a min support threshold
2. Have a confidence below a min confidence threshold
Basic Principles (2)

- If $\text{supp}(X) < \text{minsupp}$ => drop it!

  Else => add $X$ to frequent itemset $F$

- If $\text{conf}(X \Rightarrow Y) < \text{minconf}$ => drop it!

minsupp = min support threshold
minconf = min confidence threshold
Search Space Traversal (1)

- Identify all itemsets that satisfy minsupp
  - Looking at all possible subsets is doomed to failure!

- We make use of the “Downward Closure Property” of itemset support which states ...

  “All subsets of a frequent itemset must also be frequent”

Example: Consider \( I = \{1, 2, 3, 4\} \)

\[ \Rightarrow \] Lattice structure
Search Space Traversal (2)

- Itemsets above border – Frequent
- Itemsets below border – Infrequent
- Reduce search space by ignoring infrequent itemsets
- Create a tree,
  - Group itemsets into **Classes** on the basis of **Prefix**

=> Tree
Search Space Traversal (3)

- “If a parent class E’ of class E does not contain at least two frequent itemsets, then E must also not contain any frequent itemsets”

You can look at an infrequent itemset as a pollutant
Search Space Traversal (4)

- Determine itemset support
  - Counting – count occurrences in $D$ (brute force!)
  - Set Intersection – maintain a list of Transaction IDs (TID) for each itemset called tidlist

  Example: The tidlist of $X \Rightarrow X$.tidlist

- If $C = X \cup Y$
  $$\Rightarrow C$.tidlist = $X$.tidlist $\cap Y$.tidlist
  $$\Rightarrow \text{supp}(C) = |C$.tidlist|
Search Algorithms

- BFS and Counting Occurrences
- BFS and TID-List Intersections
- DFS and Counting Occurrences
- DFS and TID-List Intersections
BFS

DFS

Counting

Intersecting

Apriori

AprioriTID

DIC

Partition

FP-growth

Eclat
BFS Algorithms

- BFS and Counting Occurrences
  - Apriori Algorithm
  - AprioriTID
  - DIC

- BFS and TID-List Intersections
  - Partition Algorithm
DFS Algorithms

- DFS and Counting Occurrences
  - FP-Growth

- DFS and TID-List Intersections
  - Eclat Algorithm
Comparison

- The basic choice affecting algorithm runtime is the selection between counting occurrences and intersecting tidlists.

- Results are based on C++ simulations of, Eclat, Partition (TID-List) and Apriori (Counting) algorithms when applied to “market-basket” like data (down sampled version of database).
Avg Size of Transactions = 10
Avg Size of Frequent Itemsets = 2
Size of Database = 100K
Avg Size of Transactions = 10
Avg Size of Frequent Itemsets = 4
Size of Database = 100K
Avg Size of Transactions = 20
Avg Size of Frequent Itemsets = 2
Size of Database = 100K
Avg Size of Transactions = 20
Avg Size of Frequent Itemsets = 4
Size of Database = 100K
Avg Size of Transactions = 20
Avg Size of Frequent Itemsets = 6
Size of Database = 100K
Conclusion

- All four algorithms that were compared showed similar runtime behavior.

- No one algorithms is better than any other, in spite of fundamental differences in the algos themselves.

- The advantages and disadvantages of various approaches are balanced out when applied to real data.
References


Thank You!

Q & A