IJNRD.ORG ISSN : 2456-4184
INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD) | IJNRD.ORG
An International Open Access, Peer-reviewed, Refereed Journal

Different Types of Algorithms used in Rule Mining

Shivansh Dwivedi

Abstract This research paper contrasts on different types of rule mining techniques and shows their algorithms and uses in mining frequent itemset or patterns in a large data. In this paper we will analyse and discuss about algorithms like Apriori algorithm, FP-Growth (Frequent Pattern Growth), Eclat (Equivalence Class Transformation), H-Mine an SPADE (Sequential Pattern Discovery using Equivalence classes).

Introduction

Phenomenon of uniting data into information is known as Data mining. Rule Mining (or Association Rule Mining) is one of the domain of Data Mining. There are numerous techniques for rule mining that aims to discover interesting patterns or relationships in large datasets. It is commonly used in various domains, such as market basket analysis, customer behaviour analysis, recommendation systems, and more. The process of rule mining involves extracting association rules from transactional data. Association rules express relationships between different items or itemset based on their co-occurrence patterns in the data. These rules are typically in the form of "if-then" statements, where an antecedent (left-hand side) itemset implies a consequent (right-hand side) itemset.

For example, consider a market basket analysis scenario with a transactional dataset from a grocery store. An association rule could be:

{Bread, Milk} => {Eggs}

This rule states that if a customer buys Bread and Milk together, then they are likely to buy Eggs as well. The rule has an associated measure called support, which indicates the proportion of transactions that contain the itemset on both

sides, and confidence, which represents the conditional probability of the consequent given the antecedent.

Different Types of Algorithms used in Rule Mining

These are few examples of rule mining algorithms. Each algorithm has its own strengths, weaknesses, and variations, and the choice of algorithm depends on the characteristics of the data, computational efficiency requirements, and specific goals of the rule mining task. They are as follows:

1. Apriori Algorithm

The Apriori algorithm is one of the earliest and most well-known algorithms for association rule mining. It generates frequent itemset by iteratively scanning the data and pruning infrequent itemset based on the Apriori property. It then generates rules from the frequent itemset based on support and confidence thresholds.

Here's a step-by-step overview of the Apriori algorithm:

1. **Initialization:** Scan the transactional dataset to identify the support count of each individual item (1-itemset).
2. **Generation of frequent itemset:** Iteratively generate candidate itemset of length k (k-itemset) based on the frequent itemset of length k-1. The candidate generation process involves joining frequent itemset and pruning infrequent itemset based on the Apriori property, which states that any subset of an infrequent itemset must also be infrequent.
3. **Scanning and counting:** Scan the transactional dataset to determine the support count of each candidate itemset. This involves counting the occurrences of candidate itemset in the transactions.
4. **Pruning:** Discard candidate itemset that do not meet the minimum support threshold specified by the user.
5. Repeat Steps 2-4 until no more frequent itemset can be generated.

The output of the Apriori algorithm is a set of frequent itemset, which are itemset that occur in the data above a specified minimum support threshold. These frequent itemset capture the co-occurrence patterns of items in the dataset.

From the frequent itemset, association rules can be generated by considering different itemset combinations. These rules typically have an antecedent (left-hand side) and a consequent (right-hand side). The selection of interesting rules

is based on measures such as support, confidence, lift, and others, which quantify the interestingness and strength of the association between the antecedent and consequent.

The Apriori algorithm has been widely used for market basket analysis, recommendation systems, and various other applications that involve discovering associations or patterns in large datasets.

2. FP-Growth (Frequent Pattern Growth)

FP-Growth (Frequent Pattern Growth) is an efficient algorithm for mining frequent itemsets, which are sets of items that frequently occur together in a transactional dataset. It was introduced by Han, Pei, and Yin in 2000 as an improvement over the Apriori algorithm.

The FP-Growth algorithm utilizes a divide-and-conquer approach and a compact data structure called an FP-tree to represent the frequent patterns in the dataset. Here's an overview of the FP-Growth algorithm:

1. **Construction of the FP-tree:** Scan the transactional dataset to construct an FP-tree, which is a compact representation of the frequent itemset. The FP-tree is built in a bottom-up manner, where each transaction is inserted into the tree based on the order and frequency of the items.
2. **Header table construction:** Create a header table that stores the support count and pointers to the occurrences of each unique item in the FP-tree. This table is used to efficiently navigate and mine the frequent itemset.
3. **Mining frequent itemset:** Starting from the least frequent item in the header table, perform a recursive process called FP-Growth to mine the frequent itemset. This process involves building conditional FP-trees for each item and recursively mining frequent itemset from these conditional trees.
4. **Generation of association rules:** Once the frequent itemset are discovered, association rules can be generated by considering different itemset combinations. The rules are evaluated based on measures like support, confidence, and lift to determine their interestingness and strength.

The key advantage of the FP-Growth algorithm is its ability to avoid the costly generation of candidate itemset, which is a bottleneck in the Apriori algorithm. By utilizing the FP-tree structure and recursive mining, FP-Growth significantly reduces the number of passes over the dataset and improves efficiency.

FP-Growth has been widely adopted for association rule mining tasks, especially in scenarios with large transactional datasets. It offers better performance

compared to the Apriori algorithm and is particularly effective when the dataset contains a large number of infrequent itemset or when there are memory constraints.

3. Eclat (Equivalence Class Transformation)

Eclat (Equivalence Class Transformation) is another algorithm used for mining frequent itemset in a transactional dataset. It was introduced by Zaki in 1997 as an alternative to the Apriori algorithm, with the goal of improving efficiency by using a vertical data format representation.

The Eclat algorithm employs a depth-first search strategy to discover frequent itemset by intersecting the transaction lists of items. Here's an overview of the Eclat algorithm:

1. **Vertical data format representation:** Transform the transactional dataset into a vertical data format, where each column represents a distinct item and each row contains the transaction identifiers where the item appears. This representation allows for efficient itemset intersections.
2. **Initialization:** Create a prefix tree (also known as a prefix lattice or a prefix tree structure) to store the intersections of itemset. Each node in the tree represents an item and contains pointers to the transactions in which the item appears.
3. **Depth-first search:** Perform a depth-first search traversal of the prefix tree to discover frequent itemset. Starting with each item, recursively traverse the tree to find intersections with other items. At each level, record the support count of the generated itemset.
4. **Pruning and generation of frequent itemset:** Prune infrequent itemset based on a minimum support threshold specified by the user. Generate frequent itemset by concatenating the items encountered during the depth-first search traversal.
5. **Generation of association rules:** Similar to other association rule mining algorithms, association rules can be generated from the frequent itemset by considering different itemset combinations. The rules are evaluated using measures like support, confidence, and lift.

Eclat is known for its simplicity and ability to handle large datasets efficiently. By exploiting the vertical data format, it avoids generating candidate itemset and reduces memory requirements compared to the Apriori algorithm. However, it may be less efficient when the number of items is large, as the depth-first search process can become computationally expensive.

Eclat has been widely used in association rule mining tasks, particularly in scenarios where the dataset contains a large number of transactions but a relatively small number of distinct items.

4. H-Mine

H-Mine is a hybrid algorithm that combines the Apriori algorithm and the FP-Growth algorithm for mining frequent itemset in transactional datasets. It was proposed by Zhang, Zhang, and Li in 2003 as a way to leverage the strengths of both algorithms and improve mining efficiency.

The H-Mine algorithm consists of the following steps:

1. **Initial candidate generation:** Generate the 1-itemsets by scanning the transactional dataset and counting the occurrences of each item.
2. **Apriori-based candidate generation**: Use the Apriori algorithm to generate candidate itemset of length k (k-itemset) based on the frequent itemset of length k-1. The Apriori algorithm employs a join and prune strategy to eliminate infrequent itemset.
3. **FP-Growth-based support counting:** Build an FP-tree from the transactional dataset and perform a conditional FP-Growth process to count the support of each candidate itemset. This step involves traversing the FP-tree to identify the occurrences of each candidate itemset and updating their support counts.
4. **Pruning:** Eliminate infrequent itemset based on the minimum support threshold specified by the user.
5. Repeat Steps 2-4 until no more frequent itemset can be generated.

By combining the Apriori algorithm and the FP-Growth algorithm, H-Mine aims to exploit the benefits of both approaches. The Apriori-based candidate generation reduces the number of unnecessary candidate itemset, while the FP-Growth-based support counting utilizes the compact FP-tree structure to improve efficiency.

H-Mine is particularly effective when the dataset contains a moderate number of frequent itemset but a large number of transactions. It leverages the strengths of the Apriori algorithm's pruning strategy and the FP-Growth algorithm's efficient support counting process to achieve faster mining performance.

It's worth noting that H-Mine is just one of the many algorithms developed for mining frequent itemset, and its performance may vary depending on the characteristics of the dataset and the specific implementation details.

5. SPADE (Sequential Pattern Discovery using Equivalence classes)

SPADE (Sequential Pattern Discovery using Equivalence classes) is an algorithm used for mining sequential patterns in sequential or time-stamped transactional datasets. It was introduced by Zaki in 2001 as an efficient approach to discovering frequent sequential patterns.

The SPADE algorithm employs a divide-and-conquer strategy, where it recursively partitions the dataset into smaller subproblems and merges the results to obtain the final set of sequential patterns. Here's an overview of the SPADE algorithm:

1. Equivalence class construction: Transform the sequential dataset into an equivalence class representation. Each equivalence class groups together similar transactions based on their common prefixes. This step reduces redundancy and improves efficiency.
2. Initial prefix tree construction: Build an initial prefix tree from the equivalence classes. The prefix tree represents the sequential patterns in a compact and efficient manner. Each node in the tree represents an item, and the paths from the root to the leaf nodes represent sequential patterns.
3. Projection and mining: Perform a recursive process called projection to mine sequential patterns. The projection step involves dividing the dataset into smaller subproblems by projecting the equivalence classes onto different items. For each projected subproblem, build a corresponding prefix tree and mine sequential patterns from it.
4. Merging and pattern growth: Merge the results of the projected subproblems and perform pattern growth to obtain the complete set of sequential patterns. Pattern growth involves extending the discovered patterns by considering the remaining items in the projected subproblems.
5. Pruning and support counting: Prune infrequent sequential patterns based on a user-specified minimum support threshold. Count the support of the remaining patterns by traversing the equivalence classes and updating the support counts.

SPADE is particularly effective in mining sequential patterns from large datasets, as it reduces redundancy through equivalence class construction and uses a prefix tree structure for efficient pattern representation. It avoids generating candidate patterns, which is a common bottleneck in other sequential pattern mining algorithms.

Sequential patterns discovered by SPADE can provide valuable insights into temporal or sequential dependencies in the data. Applications of SPADE include web clickstream analysis, DNA sequence analysis, customer behaviour analysis, and more.

It's worth noting that there have been subsequent improvements and variations of SPADE, such as SPADE+ and GSP (Generalized Sequential Pattern), which address specific limitations or introduce additional features to enhance the algorithm's performance and flexibility.