Data Mining- Practice 8 Association Patten Mining

Rajesh Kalakoti[§], Sven Nomm[*]
[*] Taltech, Estonia, 12616
[§] Email: rajesh.kalakoti@outlook.com

***Abstract*—Association Pattern Mining is a data mining technique used to discover interesting relationships, patterns, associations, or correlations among sets of items in large databases. These relationships are typically hidden within the data and can provide valuable insights into consumer behavior, market basket analysis, and various other domains.**

I. Introduction

Let U be the d-dimensional universe of elements (goods offered by the supermarket) and T is the set of transactions $T_1, \dots, T_n$. They said that transaction T_i is drawn on the universe of items U. T_i may be represented by d-dimensional binary record. Itemset is the set of items. k-itemset is the itemset containing exactly k-items.

Frequent Pattern Mining is a critical technique in data mining and has several models and algorithms designed to discover patterns within datasets.

A. The Frequent Pattern Mining Model

1) Definition: **Support:** The support of an itemset I is defined as the fraction of the transactions in the database $T = \{T_1, \dots, T_n\}$ that contain I as the subset. The support of the itemset I is defined by $sup(I)$. Not to be confused with supremum.

2) Definition: **Frequent Itemset Mining:** Given a set of transactions $T = \{T_1, \dots, T_n\}$ where each transaction T_i is drawn on the universe of elements U, determine all itemsets I that occur as a subset of at least a predefined fraction $minsup$ of the transactions in T. The predefined fraction $minsup$ is referred to as minimal support.

```
data <- data.frame(
  V1 = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12),
  V2 = c("Milk", "Milk", "Bread", "Milk", "Bread", "Milk", "Milk", "Milk", "Bread", "Milk", "
  V3 = c("Egg", "Butter", "Butter", "Bread", "Butter", "Bread", "Cookies", "Bread", "Butter",
  V4 = c("Bread", "Egg", "Ketchup", "Butter", "Cookies", "Butter", NA, "Butter", "Egg", "Brea
  V5 = c("Butter", "Ketchup", NA, NA, NA, "Cookies", NA, NA, "Cookies", NA, NA, "Ketchup")
)
```

```
 print(xtable(
   data,
   caption='Example of the iris dataset',
   label='tbl:iris.xtable',
   align=c(rep('r', 5), 'l')))
```

TABLE I: Example of the iris dataset

V1	V2	V3	V4	V5
1.00	Milk	Egg	Bread	Butter
2.00	Milk	Butter	Egg	Ketchup
3.00	Bread	Butter	Ketchup	
4.00	Milk	Bread	Butter	
5.00	Bread	Butter	Cookies	
6.00	Milk	Bread	Butter	Cookies
7.00	Milk	Cookies		
8.00	Milk	Bread	Butter	
9.00	Bread	Butter	Egg	Cookies
10.00	Milk	Butter	Bread	
11.00	Milk	Bread	Butter	
12.00	Milk	Bread	Cookies	Ketchup

3) Definition: **Frequent Itemset Mining (Set-wise):** Given a set of sets $T = \{T_1, \dots, T_n\}$, where each transaction T_i is drawn on the universe of elements U, determine all sets I that occur as a subset of at least a predefined fraction $minsup$ of the sets in T.

Support Monotonicity Property: The support of every subset J of I is at least equal to the support of itemset I: $sup(J) \geq sup(I)$ for all $J \subset I$.

Downward Closure Property: Every subset of the frequent itemset is also frequent.

4) Definition: **Maximal Frequent Itemsets:** A frequent itemset is maximal at a given minimum support level $minsup$, if it is frequent and no superset of it is frequent.

B. Association Rule Generation Framework

Informal definition

If the presence of item set X in certain transaction(s) leads (implies) the presence of the set of items Y in the same transaction(s), then we talk about the rule $X \Rightarrow Y$.

1) Definition: **Confidence:** Let X and Y be two sets of items. The confidence of the rule $X \Rightarrow Y$ is the conditional probability of $X \cup Y$ occurring in a transaction, given that the transaction contains X.

$$\text{conf}(X \Rightarrow Y) = \frac{\text{sup}(X \cup Y)}{\text{sup}(X)}$$

2) Definition: **Association Rule:** Let X and Y be two sets of items. Then, the rule $X \Rightarrow Y$ is said to be an association rule at a minimum support of $minsup$ and minimum confidence $minconf$ if it satisfies the following conditions:

1. $\text{sup}(X \cup Y) \geq minsup$
2. $\text{conf}(X \Rightarrow Y) \geq minconf$

C. Frequent Itemset Mining Algorithms

There are several algorithms used for frequent itemset mining:

1) 1. Brute Force Algorithms: Brute force algorithms exhaustively search through all possible itemsets to find frequent ones. This method can be computationally intensive and is generally not efficient for large datasets.

2) 2. The Apriori Algorithm: The Apriori algorithm is a classic method for frequent itemset mining. It uses an iterative approach and prior knowledge of smaller frequent itemsets to find larger ones. The key idea is that if an itemset is frequent, all of its subsets must also be frequent. The Apriori algorithm uses this property to reduce the search space and improve efficiency.

3) 3. Enumeration-Tree Algorithms: Enumeration-tree algorithms utilize tree structures to enumerate and discover frequent itemsets efficiently. One common approach is using Recursive Suffix-Based Pattern Growth Methods. These methods involve recursively building a tree-like structure where each node represents an item and its frequency. By exploring the tree, these algorithms efficiently identify frequent itemsets.

These algorithms form the backbone of frequent itemset mining, each with its advantages and use cases

Today, we will see how aprori algorithm works.

D. Apriori Algorithm

The **Apriori algorithm** is a classic algorithm used for frequent itemset mining and association rule learning in data mining and machine learning. It discovers interesting relationships hidden in large datasets. Here's the algorithm description:

```r
library(arules)
```

```
## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
# Convert the data frame to transactions format
transactions <- as(data, "transactions")
```

```
## Warning: Column(s) 1, 2, 3, 4, 5 not logical or factor. Applying default
## discretization (see '? discretizeDF').
```

```r
# Apply Apriori algorithm
frequent_itemsets <- apriori(transactions, parame
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSup
##         0.5    0.1    1 none FALSE
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3
##
## set item appearances ...[0 item(s)] done [0.00
## set transactions ...[17 item(s), 12 transactic
## sorting and recoding items ... [7 item(s)] don
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [9 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Print frequent itemsets
inspect(frequent_itemsets)
```

```
##     lhs                         rhs           suppo
## [1] {}                       => {V2=Milk}   0.750
## [2] {V4=Butter}              => {V3=Bread}  0.333
## [3] {V3=Bread}               => {V4=Butter} 0.333
## [4] {V4=Butter}              => {V2=Milk}   0.333
## [5] {V3=Bread}               => {V2=Milk}   0.416
## [6] {V2=Milk}                => {V3=Bread}  0.416
## [7] {V3=Bread, V4=Butter} => {V2=Milk}   0.333
## [8] {V2=Milk, V4=Butter}  => {V3=Bread}  0.333
## [9] {V2=Milk, V3=Bread}   => {V4=Butter} 0.333
##     lift     count
## [1] 1.000000 9
## [2] 2.400000 4
## [3] 2.400000 4
## [4] 1.333333 4
## [5] 1.333333 5
## [6] 1.333333 5
## [7] 1.333333 4
## [8] 2.400000 4
## [9] 2.400000 4
```