Quantitative Association Rules

Pietro Sala

Data Mining 24/25 - Exercises 1 and 2

QAR: definitions

We consider a quantitative dataset D on the schema $A_1, \dots, A_n, C$, where each A_i is in $\mathbb{R} \setminus \mathbb{N}$ and C is a positive natural number. In this context, we define two key concepts: itemsets and the satisfaction relation.

We define the set of all open intervals over natural numbers as follows:

$$\mathbb{IN} = \{(b, e) \mid b, e \in \mathbb{N}, b < e\}$$

For two intervals $(b_1, e_1), (b_2, e_2) \in \mathbb{IN}$, we say that (b_1, e_1) is contained in (b_2, e_2), denoted as $(b_1, e_1) \subseteq (b_2, e_2)$, if and only if:

$$(b_1, e_1) \subseteq (b_2, e_2) \iff b_2 \leq b_1 \leq e_1 \leq e_2$$

An itemset I over the dataset D is a function that maps each attribute to an interval over the natural numbers. Formally, we express this as:

$$I : \{A_1, \dots, A_n\} \to \mathbb{IN}$$

where $\{A_1, \dots, A_n\}$ is the set of attributes in D but C.

Over the itemsets, we define a containment relation $\sqsubseteq$ as follows. For two itemsets I and I', we say that I is contained in I' (denoted as $I \sqsubseteq I'$) if and only if for each attribute A_i, the interval $I(A_i)$ is contained into interval in $I'(A_i)$. Formally:

$$\begin{gathered} I \sqsubseteq I' \\ \Updownarrow \\ \forall A_i \in \{A_1, \dots, A_n\} : I(A_i) \subseteq I'(A_i) \end{gathered}$$

For a point $p \in \mathbb{R}$ and an interval $(b, e) \in \mathbb{IN}$, we say that p belongs to (b, e), denoted as $p \in (b, e)$, if and only if:

$$p \in (b, e) \iff b < p < e$$

Given an itemset I and a tuple t in D, we define a satisfaction relation to determine whether t satisfies I. We say that t satisfies I (denoted as $t \models I$) if and only if, for each attribute A_i, the value of $t[A_i]$ falls within the interval $I(A_i)$. Formally:

$$\begin{gathered} t \models I \\ \Updownarrow \\ \forall A_i \in \{A_1, \dots, A_n\} : t[A_i] \in I[A_i] \end{gathered}$$

where $t[A_i]$ denotes the value of attribute A_i in tuple t. This definition effectively checks each attribute A_i in the tuple, looking if $I(A_i)$ contains the value $t[A_i]$. If this holds for all attributes, then t is considered to satisfy I. It is important to note that while $t[A_i]$ is a single real number, $I(A_i)$ is an interval.

Now, let us consider a specific instance $\boldsymbol{d}$ of the dataset D. For each attribute A_i, we denote by max_i the maximum value of that attribute in $\boldsymbol{d}$. Formally, we can express this as:

$$max_i = \max\{\lceil t[A_i] \rceil \mid t \in \boldsymbol{d}\}$$

where $t[A_i]$ represents the value of attribute A_i in tuple t. This definition of max_i provides us with the upper bound of values for each attribute within the given instance $\boldsymbol{d}$ of our dataset.

Given an itemset I, a dataset instance $\boldsymbol{d}$, and a real number $\varepsilon \in [0, 1]$, we say that I is ε-supported in $\boldsymbol{d}$ if and only if the sum of C values for tuples in $\boldsymbol{d}$ that satisfy I, divided by the total sum of C values in $\boldsymbol{d}$, is greater than ε. Formally:

$$\begin{gathered} I \text{ is } \varepsilon\text{-supported in } \boldsymbol{d} \\ \Updownarrow \\ \frac{\sum_{t \in \boldsymbol{d} : t \models I} t[C]}{\sum_{t \in \boldsymbol{d}} t[C]} \geq \varepsilon \end{gathered}$$

This concept of ε-support provides a threshold for considering an itemset as sufficiently represented in a dataset instance. It allows us to focus on itemsets that occur frequently enough to be of interest,

Algorithm 1 Apriori Algorithm for Quantitative Itemsets

Require: Dataset $\boldsymbol{d}$, support threshold ε

Ensure: Relation $\mathcal{R}(itemset, support)$ of all ε-supported itemsets and their support values

Initialize empty relation $\mathcal{R}(itemset, support)$ with key $itemset$
$SW_0 \leftarrow \{I_0\}$ ▷ Set of supported witnesses of level 0
$k \leftarrow 1$
while $SW_{k-1} \neq \emptyset$ **do**
    $W_k \leftarrow \{I : \forall I'(I \sqsubseteq I' \wedge \Delta(I') = \Delta(I) - 1 \implies I' \in SW_{k-1})\}$
    $SW_k \leftarrow \{I : I \in W_k \wedge I \text{ is } \varepsilon\text{-supported in } \boldsymbol{d}\}$
    for all $I \in SW_k$ **do**
        Insert $(I, support(I))$ into $\mathcal{R}$
    end for
    $k \leftarrow k + 1$
end while
return $\mathcal{R}$

Algorithm 2 Randomic Apriori Algorithm for Quantitative Itemsets

Require: Dataset $\boldsymbol{d}$, support threshold ε

Ensure: Relation $\mathcal{R}(itemset, support)$ of ε-supported itemsets that contains the supported frontier and their support values

Initialize empty relation $\mathcal{R}(I, support)$ with key I
$LP \leftarrow \{I_0\}$ ▷ Local Pending Itemsets
$LS \leftarrow LNS \leftarrow \{\}$ ▷ Local Supported/Not Supported Itemsets $\mapsto$ support
while $LP \neq \emptyset$ **do**
    $I \leftarrow$ a random element from LP
    if $\{I' : I' \sqsubseteq I, I' \in Dom(LS)\} \cup \{I' : I \sqsubseteq I', I' \in Dom(LNS)\} = \emptyset$ **then**
        $s \leftarrow$ support of I in $\boldsymbol{d}$
        if $s \geq \epsilon$ **then**
            insert (I, s) in $\mathcal{R}$
            $LS \leftarrow LS \cup \{I \mapsto s\}$
            $LP \leftarrow LP \cup \{I' : I' \sqsubseteq_{+1} I\}$
        else
            $LNS \leftarrow LNS \cup \{I \mapsto s\}$
        end if
    end if
    remove I from LP
end while
return $\mathcal{R}$

Algorithm 3 Randomic Distributed Apriori Algorithm for Quantitative Itemsets

Require: Dataset $\boldsymbol{d}$, support threshold ε

Ensure: Relation $\mathcal{R}(\mathit{itemset}, \mathit{support})$ of ε-supported itemsets that contains the supported frontier and their support values

Initialize empty relation $\mathcal{R}(I, \mathit{support})$ with key I
$GP \leftarrow \{I_0\}$ ▷ Global Pending Itemsets shared among the workers
$GS \leftarrow GNS \leftarrow \{\}$ ▷ Global Supported/Not Supported Itemsets Itemsets $\mapsto$ support
parallel for N Workers **do** ▷ start N workers
    $LS \leftarrow LNS \leftarrow \{\}$ ▷ Local Supported/Not Supported Itemsets
    $I \leftarrow$ a random element from GP
    while I exists **do**
        if $\{I' : I' \sqsubseteq I, I' \in \mathit{Dom}(LS)\} \cup \{I' : I \sqsubseteq I', I' \in \mathit{Dom}(LNS)\} = \emptyset$ **then**
▷ check if I can be rejected locally
            $\mathit{global\_reject} \leftarrow \bot$
            for I' s.t. $I' \sqsubseteq_{+1} I$ **do** ▷ check if I can be globally just using successors
                if $I' \in GS$ **then**
                    $\mathit{global\_reject} \leftarrow \top$
                    $LS \leftarrow LS \cup \{I'\}$
                    break
                end if
            end for
            if $\neg \mathit{global\_reject}$ **then**
                for I' s.t. $I \sqsubseteq_{+1} I'$ **do** ▷ check if I can be globally just using predecessors
                    if $I' \in GNS$ **then**
                        $\mathit{global\_reject} \leftarrow \top$
                        $LNS \leftarrow LNS \cup \{I'\}$
                        break
                    end if
                end for
            end if
            if $\neg \mathit{global\_reject}$ **then** ▷ cannot reject I then we will check it against $\boldsymbol{d}$
                $s \leftarrow$ support of I in $\boldsymbol{d}$
                if $s \geq \epsilon$ **then**
                    $GS \leftarrow GS \cup \{I \mapsto s\}$
                    $GP \leftarrow GP \cup \{I' : I' \sqsubseteq_{+1} I\}$
                    $LS \leftarrow LS \cup \{I\}$
                else
                    $GNS \leftarrow GNS \cup \{I \mapsto s\}$
                    $LNS \leftarrow LNS \cup \{I\}$
                end if
            end if
        end if
        remove I from GP
        $I \leftarrow$ a random element from GP
    end while
end parallel for
$\mathcal{R} \leftarrow \{(I, s) : I \in \mathit{Dom}(GS) \wedge GS(I) = s\}$
return $\mathcal{R}$

with the threshold ε determining the minimum required level of support.
Given two intervals $[b, e]$ and $[b', e']$ such that $[b, e] \sqsubseteq [b', e']$, we define their shrink difference, denoted by $\delta([b, e], [b', e'])$, as:

$$\delta([b, e], [b', e']) = (b - b') + (e' - e)$$

This shrink difference quantifies the total amount by which the larger interval $[b', e']$ needs to be "shrunk" from both ends to obtain the smaller interval $[b, e]$. It provides a measure of how much the intervals differ in size and position.

We define the bottom itemset, denoted by I_0, as the itemset that maps each attribute A_i to the interval $[0, max_i]$, formally, $I_0(A_i) = [0, max_i]$, for each $1 \leq i \leq n$.
Clearly, for any itemset I defined over the same dataset instance, we have $I \sqsubseteq I_0$. This property allows us to define the shrinking of an itemset I, denoted by $\Delta(I)$, as the sum of the shrink differences between the intervals in I and the corresponding intervals in I_0 for all attributes. Formally:

$$\Delta(I) = \sum_{i=1}^{n} \delta(I(A_i), [0, max_i])$$

where $\delta([b, e], [0, max_i])$ is the shrink difference as defined earlier. This shrinking $\Delta(I)$ quantifies how much more specific the itemset I is compared to the bottom itemset I_0, considering all attributes and all intervals in I.

Lemma 1. *Let I be an itemset that is ε-supported in a dataset instance $\boldsymbol{d}$. Then, all itemsets I' such that $I \sqsubseteq I'$ and $\Delta(I') = \Delta(I) - 1$ are also ε-supported in $\boldsymbol{d}$.*

In the following we will use the notation $\sqsubseteq_{+1}$ for denoting the relation which holds on all and only the pairs of I, I' such that $I \sqsubseteq I'$ and $\Delta(I') = \Delta(I) - 1$.

We now present the Apriori in standard, randomic, and distributed fashion (Algorithm 1, Algorithm 2, and Algorithm 3) algorithm adapted for quantitative itemsets, which efficiently finds all ε-supported itemsets in a given dataset.
Where:

- $\boldsymbol{d}$ is the input dataset
- ε is the support threshold
- I_0 is the bottom itemset as defined earlier
- SW_k is the set of ε-supported witnesses at level k
- W_k is the set of candidate itemsets at level k
- $\Delta(I)$ is the shrinking of itemset I as defined earlier
- $support(I)$ is calculated as $\frac{\sum_{t \in \boldsymbol{d}: t \models I} t[C]}{\sum_{t \in \boldsymbol{d}} t[C]}$

Exercise 1

Implementation and Analysis of Randomic Distributed Apriori Algorithm for Quantitative Itemsets. This exercise will guide you through the process of implementing, testing, and applying the Apriori algorithm for quantitative itemsets. You will also perform post-processing on the results to extract and rank association rules.

Assignment - Algorithm Implementation and Testing

- Implement the Apriori, Randomic Apriori, and Randomic Distributed Apriori algorithms for quantitative itemsets in your programming language of choice.
- Create a set of test cases to verify the correctness of your implementation.
- Ensure your implementation can handle various input sizes and support thresholds.

Exercise 2

Association Rule Extraction and Analysis

Extract all the association rules with confidence greater than or equal to 0.8 involving just itemsets in the frontier. This means that for each required rule $I(A_i^1), ..., I(A_i^m) \rightarrow I(A_j^1), ..., I(A_j^n)$, we have that $I(A_i^1), ..., I(A_i^m), I(A_j^1), ..., I(A_j^n)$ is an element of the frontier.

For each rule:

- Measure the p-value
- Calculate the lift
- Visualize the rules in a re-translated fashion to gain insights on the real values of the attributes

Exercise 3

Rule Filtering and Shapley Value Analysis

From the results of Exercise 2, keep only the rules $I(A_i^1), ..., I(A_i^m) \rightarrow I(A_j^1), ..., I(A_j^n)$ that have:

- p-value < 0.05
- lift > 1.5

Let us call these "final rules".

Define the following sets:

$$\begin{aligned}
Ant = \{(i, [b, e]) &: \text{there exists a final rule } I(A_i^1), ..., I(A_i^m) \rightarrow I(A_j^1), ..., I(A_j^n) \\
&\text{and } i^k = i \text{ such that } I(A_i^k) = [b, e]\} \\
Cons = \{(j, [b, e]) &: \text{there exists a final rule } I(A_i^1), ..., I(A_i^m) \rightarrow I(A_j^1), ..., I(A_j^n) \\
&\text{and } j^k = j \text{ such that } I(A_j^k) = [b, e]\}
\end{aligned}$$

Note that Ant and $Cons$ may not be disjoint.

Given a set $Ant' \subseteq Ant$, we may have multiple (even disjoint or partially overlapping) sets that refer to the same interval on the same attribute. For that reason, we define the following set that resolves such conflicts:

$$Cl(Ant') = \{(i, [b, e]) : (i, [b, e]) \in Ant' \text{ and } Sup((i, [b, e])) = \max_{(i,[b',e']) \in Ant'} Sup((i, [b', e']))\}$$

If by chance there are two sets with the maximum support on the same attribute, pick one according to some criteria (e.g., lexicographical order on $[b, e]$).

For each $(j, [b, e]) \in Cons$, let Ant_j be the subset of Ant that excludes j:

$$Ant_j = \{(i, [b, e]) \in Ant : i \neq j\} \tag{1}$$

The CPO (coalition payoff function) for $(j, [b, e])$ is computed on all the possible subsets of Ant_j (so intervals on j would never be picked), and for each $Ant'_j \subseteq Ant_j$ it is defined as the J-Measure of the rule $Cl(Ant'_j) \rightarrow (j, [b, e])$.

Compute the Shapley values (approximated) of each element of Ant_j relative to each $(j, [b, e]) \in Cons$.