Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 933 – 955
© TÜBİTAK
doi:10.3906/elk-1905-88

Multitask-based association rule mining

Pelin YILDIRIM TAŞER[1], **Kökten Ulaş BİRANT**[2], **Derya BİRANT**[2,*]
[1]Department of Computer Engineering, Faculty of Engineering and Architecture, İzmir Bakırçay University, İzmir, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey

Received: 14.05.2019 • **Accepted/Published Online:** 28.01.2020 • **Final Version:** 28.03.2020

Abstract: Recently, there has been a growing interest in association rule mining (ARM) in various fields. However, standard ARM algorithms fail to discover rules for multitask problems as they do not consider task-oriented investigation and, therefore, they ignore the correlation among the tasks. Considering this situation, this paper proposes a novel algorithm, named multitask association rule miner (MTARM), that tends to jointly discover rules by considering multiple tasks. This paper also introduces two novel concepts: single-task rule and multiple-task rule. In the first phase of the proposed approach, highly frequent local rules (single-task rules) are explored for each task separately and then these local rules are combined to produce the global result (multitask rules) using a majority voting mechanism. Experiments were conducted on four different real-world multitask learning datasets. The experimental results indicated that the proposed MTARM approach discovers more information than that of traditional ARM algorithms by jointly considering the relationships among multiple tasks.

Key words: Association rule mining, multitask learning, data mining, the frequent pattern (FP)-Growth algorithm

1. Introduction

Association rule mining (ARM) has been extensively used to extract hidden and interesting rules from a large collection of data [1]. It has been an active research area owing to the challenges it presents and to its wide applications in various fields such as market basket analysis [2], recommendation systems [3], anomaly detection [4], bioinformatics [5], and text mining [6]. The ARM process involves finding patterns whose support and confidence values are at least the user-defined thresholds. For example, an association rule "milk $\rightarrow$ eggs (support = 0.2, confidence = 0.8)" for market basket analysis may indicate that 20% of all transactions contain both milk and eggs, and that 80% of all transactions that contain milk also contain eggs.

Current ARM studies discover rules from the entire dataset; they do not consider task-based investigation and they ignore intertask relations. However, real-world datasets sometimes include situations where there are multiple tasks available and these tasks are related to each other. From a general perspective, it would seem that jointly discovering association rules from these related tasks would help us uncover common knowledge and improve generalization performance. In fact, this perspective is supported by empirical evidence provided by recent developments in multitask learning [7].

Unlike the traditional (single-task-based) ARM, the novel approach proposed in this paper is "multitask-based ARM," which can discover more knowledge by jointly analyzing all tasks and by considering the relations between these tasks. The underlying assumption of our approach is that the rules of all tasks, or at least a

*Correspondence: derya@cs.deu.edu.tr

subset of them, are familiar to one mutual rule set with a small difference. The proposed approach can discover common association rules by responding to different tasks and by applying an algorithm to consider all the tasks collectively. Accordingly, the rule $X \rightarrow Y$ can have high support and confidence in a task. A rule may not be frequent in the entire dataset; however, it may be frequent in several specific tasks. Therefore, finding task-based frequent rules in an effective way is important.

This paper proposes a novel algorithm, named multitask association rule miner (MTARM), that tends to discover rules by jointly considering multiple tasks. The proposed algorithm mines datasets that include task information. The MTARM approach consists of two phases. In the first phase, the algorithm discovers local frequent rules (*single-task rules*) from the data of each task separately, and in the second phase, it combines these local rules to produce the global result (*multitask rules*). The discovered multitask rules can be used to deploy information systems that support the execution of their associated task or to improve decision-making in applications.

The main contributions and novelty of this study are threefold: (i) this is the first study that applies ARM to multitask problems; (ii) it proposes the MTARM approach, which attempts to find intertask rules based solely on the task identities and the observed data for each task; and, finally (iii) it also introduces two novel concepts: *single-task rule* and *multitask rule*. This paper also provides a brief survey of ARM as it is an active research area of data mining. In addition, it presents the results of experimental studies conducted on four different real-world datasets to demonstrate the capability of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 summarizes the related works on the subject. In Section 3, we first briefly introduce ARM and then explain the multitask-oriented ARM concept with its definitions. This section also describes our novel algorithm, namely MTARM, and discusses its advantages. Section 4 presents the experimental studies and discussions about the obtained results. Finally, concluding remarks and future directions are presented in Section 5.

2. Related works

ARM is one of the most important data mining techniques. It was introduced by Agrawal and Srikant in 1994 [8]. ARM finds the interesting relationships among a large set of data items. In previous studies, various algorithms, including Apriori [8], frequent pattern (FP)-growth [9], and equivalence class clustering and bottom-up lattice traversal (Eclat) [10], have been proposed to efficiently discover association rules in various applications. Available ARM algorithms can be classified into two general categories: breadth-first and depth-first search-based ARM algorithms. In breadth-first search-based ARM algorithms such as Apriori, k-itemsets are used to generate $(k{+}1)$-itemsets. On the other hand, depth-first search-based ARM algorithms such as FP-growth and Eclat are more efficient methods as they apply subset checking at any depth.

The ARM problem has been addressed by some researchers, and different types of association rules have been proposed such as multilevel association rules [11], multidimensional association rules [12], multirelational association rules [13], multiclass association rules [14], multiobjective association rules [15], multimode association rules [16], multigranule association rules [17], multimodal semantic association rules [18], multilevel fuzzy association rules [19], and multiagent association rules [20]. In contrast to these present types, the new type proposed in this paper is a multitask association rule. A multitask rule presents all the knowledge between the tasks, whereas a traditional (single-task) rule only gives the information about an independent task.

Multitask learning (MTL) is an active area of research in machine learning and is concerned with simultaneously learning multiple related tasks from a common dataset [21]. Inspired by this idea, many methods have been developed to determine multiple tasks simultaneously rather than separately [22]. All these techniques have the tendency to jointly determine multiple tasks within a common environment, which can complicate the learning process, but can enhance the performance of single-task learning models. MTL has been proven to increase the learning capability of each individual task, as reported by many studies via both extensive experiments and theoretical analysis [7, 22]. Multitask classification [23] and multitask clustering [24, 25] are two well-known types of multitask learning that have been recently presented in the literature. On the other hand, multitask-oriented ARM has not been studied until now. To the best of our knowledge, this is the first study that proposes multitask-oriented ARM.

As mining association rules are computationally time-consuming and the datasets to be mined are often very large, parallel/distributed ARM (P/D ARM) algorithms have recently been developed [26–28]. These algorithms can be grouped into three categories: count distribution, data distribution, and candidate distribution methods [27, 29, 30]. In count distribution, the dataset is partitioned equally among the nodes of the parallel system; each node computes the local support for every candidate k-item set; and at the end of each iteration, global support is generated by exchanging and summing up the local supports. In data distribution, the set of candidate item sets is partitioned among the nodes and each node counts mutually exclusive candidates. In candidate distribution, the algorithm partitions both the data and the candidates in such a way that each processor proceeds independently with a load balancing strategy. A comparison of these methods has been presented in several papers [29, 31].

Our proposed method (MTARM) differs from the previous P/D ARM methods in many respects (Table 1). First, P/D ARM algorithms discover the same rules as those of the traditional ARM algorithms, but in a shorter time by using multiple nodes; however, MTARM finds more rules than those of the standard ARM algorithms by jointly considering the semantic relationships among tasks. Second, MTARM supports interparallelism (multiple models are concurrently built, each model is assigned to a different processor, and then local models are merged at the end), whereas P/D ARM supports intraparallelism (one model is built by executing multiple processors in parallel). Third, MTARM does not require any message exchange between nodes, in contrast to P/D ARM. Hence, the overall system performance can be enhanced by reducing the communication cost. Fourth, synchronization between nodes is required only at the end, instead of after each pass. Fifth, P/D ARM algorithms aggregate local results by performing a mathematical sum operation to obtain the global result at each step, whereas MTARM combines local results by using a majority voting mechanism at the end. Sixth, in P/D ARM, the dataset can be partitioned among n nodes in any way (i.e. equally); however, in MTARM, the dataset should be semantically divided into parts according to the task information: node1 is assigned to task1 data, node2 is assigned to task2 data, etc.

Similar to our proposed approach, the split-and-merge (SaM) algorithm found in the literature applies a split-and-merge paradigm to mine frequent rules [32]. This approach uses a data structure that is processed using a general depth-first/divide-and-conquer scheme. Although the SaM algorithm resembles the proposed MTARM algorithm in terms of the split-and-merge methodology, the latter algorithm divides the entire dataset according to tasks, finds local rules individually, and then aggregates them by using a majority voting mechanism to obtain globally frequent patterns. Furthermore, the MTARM method differs from the SaM algorithm in many other respects such as in terms of data preprocessing, merge operation, execution time, time complexity, and storage resource. First, while MTARM does not require any data preprocessing step, the SaM algorithm requires some preprocessing of the dataset, which includes reordering the items according to their frequency

Table 1. Comparison of ARM, multitask-ARM, and parallel/distributed ARM.

Method		Type of parallelism	Number of messages exchanged	Types of messages exchanged	Synchronization required	Dataset layout	Architecture	Algorithms
ARM		none	none	none	no	whole	single node	Apriori, Eclat, Fp-Growth
P/D ARM	Count Dist.	intra-model	$n(n\text{-}1)$ (in each step)	local counts (in each step)	yes (after each step)	horizontal (#nodes)	multiple nodes shared-nothing	PEAR, PDM, NPA, FDM, FPM, CCPD
	Data Dist.	intra-model	$n(n\text{-}1)$ (in each step)	local frequent itemsets (in each step)	yes (after each step)	horizontal (#nodes)	multiple nodes shared-nothing	SPA, IDD, PCCD
	Cand. Dist.	intra-model	$n(n\text{-}1)$ (in initial step)	local frequent itemsets for initial step and dataset is repartitioned	no	horizontal (#nodes)	multiple nodes shared-nothing	HPA, HPA-ELD, ParEclat, ParMaxEclat, ParClique, ParMaxClique, APM, PPAR
Multitask ARM (proposed approach)		inter- model	none	none	yes (at the end)	horizontal (#tasks)	multiple nodes shared-nothing	MTARM

in the dataset [32]. Second, while the MTARM merges local rules by using a majority voting mechanism, the SaM algorithm essentially uses a single phase of the *mergesort* sorting algorithm [32]. Third, the main lack of the SaM algorithm is that this algorithm generally performs well on dense data but shows certain weaknesses on sparse data [33]. The reason for this situation is that the merge operation of the SaM algorithm shows higher performance when the transactions that will be merged have similar lengths. However, the performance of the MTARM approach (i.e. MT-Apriori) is not especially dependent on the dense or sparse data since it uses a different merge operation. Furthermore, our proposed method and the SaM algorithm have different time complexities. The time complexity of the SaM algorithm would deteriorate from $O(nlogn)$ to $O(n^2)$ [33], where n is the number of transactions, because of the merge operation, while the execution time of our proposed approach is dependent on the runtime of the preferred base ARM algorithm as mentioned in Section 3.4. Finally, the SaM algorithm uses a different data structure that is convenient to execute on external storage or a (relational) database system if the data to mine cannot be loaded into the main memory [33].

The main contribution of the MTARM approach proposed in this paper is that it not only finds multitask frequent rules throughout the dataset but also identifies single-task frequent rules in particular tasks. In a multitask model, several particular tasks may contain different (or special) association rules. Given several related tasks, our algorithm can discover rules for each task separately and also consider the relations between these tasks.

3. Materials and methods

3.1. Association rule mining

Let $I = \{i_1, i_2, ..., i_k\}$ be a set of distinct items. Let D be a dataset that contains a set of records $\{R_1, R_2, ..., R_n\}$, where each record R is a set of items with a unique identifier such that $R \subseteq I$ and where n

is the number of records. A record R contains X and/or Y, and a set of some items in I, if $X \subseteq R$ and/or $Y \subseteq R$. The most commonly used measures to indicate the strength of an association rule $X \rightarrow Y$ are support (S) and confidence (C). Support is the probability of records that contain both X and Y item sets (X U Y), whereas confidence is the probability of transactions that contain X and also contain Y. The equations are as follows:

$$S = Support(X \rightarrow Y) = \frac{|X \cup Y|}{|D|}, \tag{1}$$

$$C = Confidence(X \rightarrow Y) = \frac{|X \cup Y|}{|X|}, \tag{2}$$

where $|D|$ is the number of records in the dataset D, $|X|$ is defined as $|\{R|X \subseteq R \,\&\, R \in D\}|$, and $|X \cup Y|$ is the number of occurrences of the related item sets. The ARM process consists of two main steps: (i) find all frequent item sets, which must be at least as frequently supported as the minimum support (*MinSup*) count, and (ii) generate strong rules from the discovered frequent item sets, where each rule in the generated rule set ($\mathfrak{R}$) must satisfy the minimum confidence (*MinConf*) and the *number of rules* is the size of the rule set ($\mathfrak{R}$), i.e. the count of rules that are frequent and strong as in the following:

$$S = Support(X \rightarrow Y) \geq MinSup \ \ and \ \ C = Confidence(X \rightarrow Y) \geq MinConf. \tag{3}$$

3.2. Multitask-oriented ARM

A *task* is addressed to the discovery of a subgoal as an outcome target by using only one input source in general. Therefore, the term "multiple tasks" refers to the discovery of multioutput targets simultaneously by utilizing task relatedness using a single input source. According to the definition of "multitask," we can define multitask-oriented ARM as follows: multitask-based ARM is a process of generating globally strong association rules from large volumes of task-oriented data. Its aim is to find general (global) rules across tasks. The most significant and difficult aspect of multitask-based ARM is how to determine the common knowledge between tasks by protecting the independence of each task.

In real-world applications, the dataset can include information about correlated multiple tasks. This type of data requires the tasks to have some similarities. However, different tasks lead to different rules, which is what this paper focuses on. It is preferable to discover common association rules by responding to different tasks and performing an algorithm to consider all the tasks simultaneously. Motivated by this idea, this study considered multitask problems to find more meaningful rules in an efficient way to improve the effectiveness of knowledge discovery systems.

This paper proposes a novel multitask-based ARM approach that improves generalization by using the domain information contained in the data records of related tasks. It does this by learning tasks in parallel while using a shared representation. The basic assumption of multitask-based ARM is that all the tasks in mining are related to each other, and, therefore, the common rules between various tasks could show more successful mining performance if all these tasks are considered simultaneously, by comparing them independently. Therefore, the mining of one task can lead to the enhancement of the performance of the mining of other tasks. In this way, it is possible to mine all the tasks together, using the associated information between various tasks to improve the mining of each task.

Our approach, which aims to jointly find association rules for multiple tasks, can be a better method by utilizing beneficial information obtained from related tasks to identify intertask relations. A key concept in our framework is the idea of a multivariate "semantic descriptor" for tasks and domains.

This paper proposes two novel concepts: single-task rule and multitask rule.

Definition 1 (Single-task rule) *A single-task rule is an implication of* $X \overset{t}{\to} Y$ *discovered from the data partition* D_t *of a particular task* t*, where* $X \subseteq I$ *and* $Y \subseteq I$ *are frequent item sets, and* $X \cap Y = \oslash$ *and its support and confidence values are equal to or greater than the minimal support (MinSup) and minimum confidence (MinConf) thresholds, which are given by a user or an expert. Here,* X *is called the antecedent, whereas* Y *is called the consequent of the single-task rule. The support and confidence values of a single-task rule are calculated as follows:*

$$S(Single - Task\,Rule) = Support(X \overset{t}{\to} Y) = \frac{|X \cup Y|_t}{|D_T|}, \tag{4}$$

$$C(Single - Task\,Rule) = Confidence(X \overset{t}{\to} Y) = \frac{|X \cup Y|_t}{|X|_t}, \tag{5}$$

where T *is a set of tasks such as* $T = \{t_1, t_2, ..., t_m\}$*.*

Definition 2 (Multitask rule) *Given* m *tasks* $\{T_i\}_{i=1}^{m}$*, where all the tasks are related but not identical, a multitask rule, which is denoted by* $X \overset{T}{\to} Y$*, is a frequent single-task rule that appears in more than half of the tasks. Suppose all subsets of an item set* X *such as* $x_1, x_2, ..., x_k$ *are locally frequent; then item set* X *is a candidate for the multitask frequent item set and subsequently refers to all participating tasks that need to sort the support of item set* X *if it appears in the local datasets of any tasks. For each single-task rule discovered in at least* $m/2$ *data partitions, a multitask rule is derived with the global support, which is the minimum support and confidence values collected on the task partitions where the pattern is found to be frequent. The support and confidence values of a multitask rule are calculated as follows:*

$$\left.\begin{array}{r} S\left(Multi - task\ Rule\right) = Support\left(X \overset{T}{\to} Y\right) = \ \min_{t \in T}\left(Support\left(X \overset{t}{\to} Y\right)\right) \\ C\left(Multi - task\ Rule\right) = Confidence\left(X \overset{T}{\to} Y\right) = \ \min_{t \in T}\left(Confidence\left(X \overset{t}{\to} Y\right)\right) \end{array}\right\} if\ \left|X \overset{t}{\to} Y\right| \geq m/2. \tag{6}$$

On the basis of these definitions, we can say that there are two elementary factors for multitask-oriented ARM.

The first factor is the definition of a task. Many real-world learning problems can be divided into a number of interrelated subtasks. The conventional ARM strategy considers each mining problem as a single unit and does not incorporate information associated with the tasks that are closely related to each other. In contrast, a multitask problem could share information across the tasks. This limitation has been addressed in the multitask-based ARM paradigm, where the rules are discovered by using several associated tasks. Owing to the task factor in ARM, a rule may not occur frequently in the entire dataset (meaning that it is not a global pattern); however, it may appear frequently over specific tasks (meaning that it is a task-based pattern). Thus, the multitask-based ARM paradigm improves the covering capacity of the association rules.

The second factor is the task relatedness. In multitask-based ARM, we need to consider that all the tasks are associated with each other and share a common set of rules. The task relationship is based on the understanding of how different tasks are related, which will be encoded into a set of association rules. The way to learn the relationship is to find hidden rules between these tasks.

Figure 1 shows a graphical representation of the proposed multitask-based ARM approach. First, data partitioning is performed by horizontally splitting the set of records into m partitions according to their task information, such that the combination of the partitions is the whole dataset. After that, the m sets of single-task rules are obtained from m tasks' data and then merged by majority voting to generate the set of multitask rules. The multitask rules obtained following the majority voting procedure approximate the original rules that can be possibly mined on the entire dataset. The check that the same single-task rule occurs more than $m/2$ times in different task partitions is based on an equivalence test between two rules. Single-task rules occurring in less than $m/2$ partitions are filtered out.

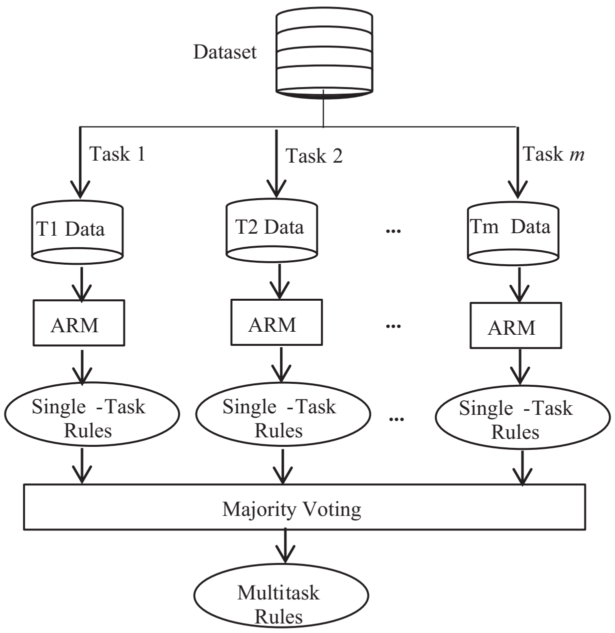


Figure 1. Illustration of the proposed multitask-based ARM approach.

The multitask-based ARM generates a set of global rules for various tasks associated with a problem under consideration by using the knowledge discovery between tasks. It is important to emphasize that the proposed approach finds both task-dependent (single-task) and task-related (multitask) rules available in the data. In our approach, multitask-based association rules are automatically generated without the need for any domain knowledge.

3.3. Example illustrating the multitask-based ARM approach

This section illustrates the proposed multitask-based ARM approach by using an example. The multitask-based ARM approach focuses on different objective tasks, as we will demonstrate in the example here. Whereas traditional ARM and P/D ARM are performed on data records without any task information (left part of Table 2), MTARM is applied on task-related data (right part of Table 2). Here, there are 15 records with their unique identifiers and 4 distinct items $I = \{A, B, C, E\}$. Suppose there are three tasks t_1, t_2, and t_3 such as

$T = \{t_1, t_2, t_3\}$, and each of them has its own dataset D_1, D_2, and D_3, respectively, where $D = D_1 \cup D_2 \cup D_3$. The minimum support is defined as 0.4 or 40% in percentage or 6 out of 15 records, whereas the minimum confidence is specified as 0.65.

Table 2. Example dataset for ARM and P/D ARM (left), and data samples for MTARM (right).

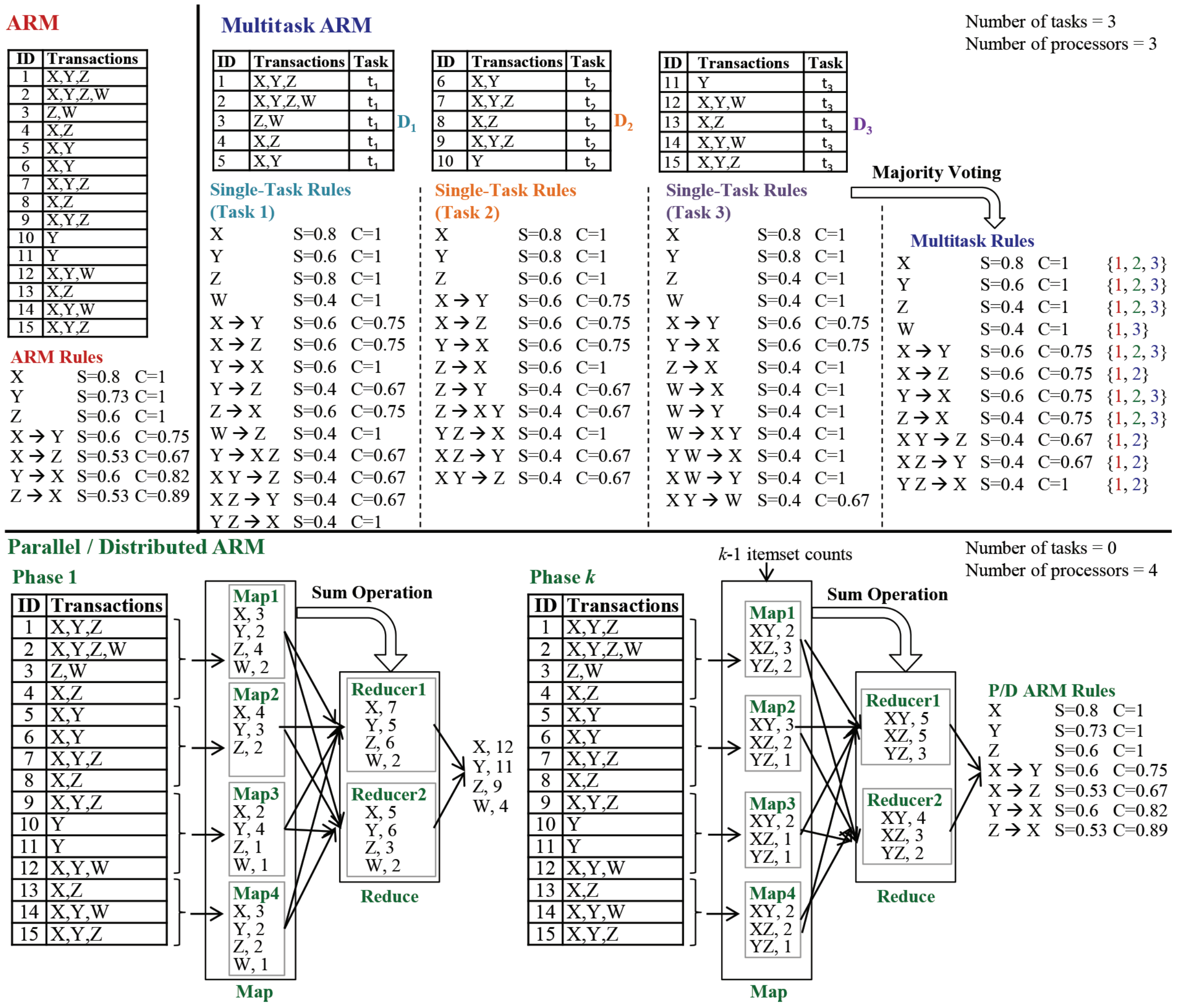
ID	Transactions
1	X,Y,Z
2	X,Y,Z,W
3	Z,W
4	X,Z
5	X,Y
6	X,Y
7	X,Y,Z
8	X,Z
9	X,Y,Z
10	Y
11	Y
12	X,Y,W
13	X,Z
14	X,Y,W
15	X,Y,Z

ID	Transactions	Task
1	X,Y,Z	t1
2	X,Y,Z,W	t1
3	Z,W	t1
4	X,Z	t1
5	X,Y	t1
6	X,Y	t2
7	X,Y,Z	t2
8	X,Z	t2
9	X,Y,Z	t2
10	Y	t2
11	Y	t3
12	X,Y,W	t3
13	X,Z	t3
14	X,Y,W	t3
15	X,Y,Z	t3

Figure 2 shows an example to illustrate the differences between ARM, MT-ARM, and P/D ARM rules discovered from the dataset given in Table 2 with a minimum support of 0.4 and a minimum confidence of 0.65.

The ARM part in Figure 2 shows the rules obtained by a traditional ARM algorithm. The algorithm uses only data transactions without considering any task information. If a rule contains k items, it can be called a k-item rule. For instance, $\{X \rightarrow Y\}$ is an example of a two-item rule. The frequent one-item rules mined from the data were $(\{X\}, \{Y\}, \{Z\})$, whereas the frequent two-item rules were $(\{X \rightarrow Y\}, \{X \rightarrow Z\}, \{Y \rightarrow X\}, \{Z \rightarrow X\})$. Totally, seven frequent rules were discovered here. For example, the rule "$X \rightarrow Y \;\; S = 0.6 \;\; C = 0.75$," where S represents support and C stands for confidence, specifies the coexistence of items "X" and "Y" in the transactions such that 60% of all transactions contain both X and Y and that 75% of all transactions that contain X also contain Y. Totally, the number of rules that were discovered in this part was only seven.

The P/D ARM part in Figure 2 shows the rules obtained by executing a count distribution-based parallel ARM algorithm on multiple nodes (i.e. four processors). The algorithm discovers the same rules as those of the traditional ARM algorithms, but in a shorter time by using multiple nodes. A MapReduce programming model can be used for processing, which consists of two main parts, namely map and reduce. The mapper provides parallelization to a system by computing the frequency of each item set in the subdataset and generates local key/value pairs. The reducer sums up the local outputs of the mappers and generates the global counts of the item sets at each iteration. When k-frequent item sets are found at the kth phase, $(k-1)$-frequent item sets are provided to the mappers by message passing.



ID	Transactions
1	X,Y,Z
2	X,Y,Z,W
3	Z,W
4	X,Z
5	X,Y
6	X,Y
7	X,Y,Z
8	X,Z
9	X,Y,Z
10	Y
11	Y
12	X,Y,W
13	X,Z
14	X,Y,W
15	X,Y,Z

ARM Rules

X	S=0.8	C=1
Y	S=0.73	C=1
Z	S=0.6	C=1
X → Y	S=0.6	C=0.75
X → Z	S=0.53	C=0.67
Y → X	S=0.6	C=0.82
Z → X	S=0.53	C=0.89

ID	Transactions	Task
1	X,Y,Z	t_1
2	X,Y,Z,W	t_1
3	Z,W	t_1
4	X,Z	t_1
5	X,Y	t_1

ID	Transactions	Task
6	X,Y	t_2
7	X,Y,Z	t_2
8	X,Z	t_2
9	X,Y,Z	t_2
10	Y	t_2

ID	Transactions	Task
11	Y	t_3
12	X,Y,W	t_3
13	X,Z	t_3
14	X,Y,W	t_3
15	X,Y,Z	t_3

Single-Task Rules (Task 1)

X	S=0.8	C=1
Y	S=0.6	C=1
Z	S=0.8	C=1
W	S=0.4	C=1
X → Y	S=0.6	C=0.75
X → Z	S=0.6	C=0.75
Y → X	S=0.6	C=1
Y → Z	S=0.4	C=0.67
Z → X	S=0.6	C=0.75
W → Z	S=0.4	C=1
Y → X Z	S=0.4	C=0.67
X Y → Z	S=0.4	C=0.67
X Z → Y	S=0.4	C=0.67
Y Z → X	S=0.4	C=1

Single-Task Rules (Task 2)

X	S=0.8	C=1
Y	S=0.8	C=1
Z	S=0.6	C=1
X → Y	S=0.6	C=0.75
X → Z	S=0.6	C=0.75
Y → X	S=0.6	C=0.75
Z → X	S=0.6	C=1
Z → Y	S=0.4	C=0.67
Z → X Y	S=0.4	C=0.67
Y Z → X	S=0.4	C=1
X Z → Y	S=0.4	C=0.67
X Y → Z	S=0.4	C=0.67

Single-Task Rules (Task 3)

X	S=0.8	C=1
Y	S=0.8	C=1
Z	S=0.4	C=1
W	S=0.4	C=1
X → Y	S=0.6	C=0.75
Y → X	S=0.6	C=0.75
Z → X	S=0.4	C=1
W → X	S=0.4	C=1
W → Y	S=0.4	C=1
W → X Y	S=0.4	C=1
Y W → X	S=0.4	C=1
X W → Y	S=0.4	C=1
X Y → W	S=0.4	C=0.67

Multitask Rules

X	S=0.8	C=1	{1, 2, 3}
Y	S=0.6	C=1	{1, 2, 3}
Z	S=0.4	C=1	{1, 2, 3}
W	S=0.4	C=1	{1, 3}
X → Y	S=0.6	C=0.75	{1, 2, 3}
X → Z	S=0.6	C=0.75	{1, 2}
Y → X	S=0.6	C=0.75	{1, 2, 3}
Z → X	S=0.4	C=0.75	{1, 2, 3}
X Y → Z	S=0.4	C=0.67	{1, 2}
X Z → Y	S=0.4	C=0.67	{1, 2}
Y Z → X	S=0.4	C=1	{1, 2}

P/D ARM Rules

X	S=0.8	C=1
Y	S=0.73	C=1
Z	S=0.6	C=1
X → Y	S=0.6	C=0.75
X → Z	S=0.53	C=0.67
Y → X	S=0.6	C=0.82
Z → X	S=0.53	C=0.89

Figure 2. Example illustrating the proposed multitask-based ARM approach.

The "single-task rules" parts in the MTARM part of Figure 2 show the frequent rules obtained for each task. Because the problem can be divided into T different tasks, our algorithm initially considers T tasks as independent tasks and generates T sets of single-task rules, one for each task. In this phase, the algorithm uses the local data of a particular task and finds the rules for each task individually. In this example, the first five records belong to the first task (t_1), and, therefore, single-task rules for t_1 will be generated from this part of the data. Consider the single-task rule "$XY \rightarrow Z\ \ S = 0.4\ \ C = 0.67$" for task 1. Because the support count for $\{X, Y, Z\}$ is 2 and the total number of transactions is 5, the support value of the rule is 2 / 5 = 0.4. The confidence value of the rule is obtained by dividing the support count for $\{X, Y, Z\}$ by the support count for $\{X, Y\}$. Because there are three transactions that contain X and Y, the confidence for this rule is 2 / 3 = 0.67.

The "multi-task rules" part in the MTARM part of Figure 2 shows the frequent rules obtained by incorporating the useful information from all tasks along with its own. During the process of aggregating all local rules, all the participating tasks are considered. Let us discover the rules that appear two or more times in the various partitions. We obtain single-task rules with a count of at least 2. In other words, single-task rules occurring in only one partition are filtered out. An example of a multitask rule is "$YZ \rightarrow X\ \ S = 0.4\ \ C = 1$ $\{1, 2\}$," which describes the relationships between three items, namely X, Y, and Z. In this rule, $\{1, 2\}$ means that this pattern is found in the first and second partitions (task 1 and task 2, respectively). In the same rule, $S = 0.4$ and $C = 1$ indicate the global support and confidence, respectively, which are obtained by selecting the minimum values computed on the partitions. For instance, the global support value of the rule "$Z \rightarrow X$" is 0.4 because its support values in the partitions are {0.6, 0.6, and 0.4}, and the minimum value is 0.4. Totally, 11 frequent multitask rules were discovered here, whereas the standard ARM algorithm found only seven rules. Therefore, the knowledge discovery from all the tasks helps to increase the covering capacity of association rules. Thus, the multitask-based ARM approach has been successfully applied on real-world multitask problems.

3.4. MTARM: Multitask association rule miner

This paper proposes a novel algorithm, named multitask association rule miner (MTARM), that tends to discover rules by jointly finding shared information across different related tasks. Our algorithm consists of iterations on two subsequent local and global steps: local frequency computation and majority voting-based aggregation. The algorithm first divides the entire dataset into data subsets according to the task information and then calculates the frequencies in the data of each task separately; afterwards, it aggregates local information and obtains globally frequent patterns in an effective way. Processing local rules to discover global ones requires a way of combining distinct sets of patterns into a single set and finding global support and confidence values.

The pseudocode of the MTARM approach, which has three main steps, is given in Algorithm 1. First, it scans the dataset D to horizontally divide it into m sets according to m different tasks, such that $D_1, D_2, ..., D_m$. Therefore, each local dataset contains a set of task-related patterns. Second, the algorithm generates single-task rules (STR_i) for each task individually, whose support and confidence values are greater than the *MinSup* and *MinConf* thresholds, respectively, by running a standard ARM algorithm at the ith data partition (D_i). The next phase is mainly an aggregation phase, in which the algorithm combines local frequencies using a majority voting mechanism and generates global frequencies. Finally, multitask rules are obtained by intersecting the single-task rules in different task partitions. A local frequent rule becomes globally frequent if it occurs in more than half of the partitions. At the same time, the local rules are scanned to determine the global support and confidence values by finding the minimum values, respectively.

The main aim is to find the global rules that should be locally frequent in at least $m/2$ data parts, where m is the number of tasks. In the case where m is set to 1, this guarantees that the local solution is equal to the global solution. However, a merge step with $m > 1$ may generate several false positives, i.e. rules that are locally frequent but are globally infrequent. Hence, the value of m is an important parameter in determining globally frequent rules. The combine stage also tries to approximate the support and confidence values for the global rules by selecting the minimum support and confidence values of the local rules.

The time complexity of the proposed MTARM approach, in the worst case, is related to the number of tasks, number of items, number of records, average length of the records, and time taken to merge single-task rules. It is given by $O(m.T(n/m) + s^2)$, where m is the number of tasks, n is the number of records, T represents the time required for the execution of a standard ARM algorithm on n/m records, and s is the number of single-task rules, and, therefore, s^2 represents the time needed for the merge operation. The overall execution time of the algorithm is dependent on the runtime of the preferred ARM algorithm (i.e. Apriori, FP-growth, and Eclat). For example, the time complexity of the Apriori algorithm is $O(n.q.w)$, where n is the

Algorithm 1 MTARM: Multitask association rule.

```
Inputs:
 $D$: the dataset that contains a set of records
 $n$: the number of records in the dataset $D$
 $T$: a set of tasks such that $T = \{t_1, t_2, ..., t_m\}$
 $m$: the number of tasks
 $MinSup$: minimum support value
 $MinConf$: minimum confidence value
Output:
 $MTR$: A list of multitask rules
Begin:
//Horizontally partition $D$ into $m$ sets
for $i = 1$ to $n$ do
  for $j = 1$ to $m$ do
    if ($R_i$ belongs to $T_j$) then
      $D_j$.Add($R_i$)
    end if
  end for
end for
//Discover single-task rules, $STR$
for $i = 1$ to $m$ do
  $STR_i$ = ARM_Algorithm($D_i$, $MinSup$, $MinConf$)
  $STR$ = $STR$ U $STR_i$
end for
//Find multitask rules, $MTR$
for $i = 1$ to $STR.Length$ do
  $count = 0$
  $rule$=$STR_i$.Sort()
  for $j = 1$ to $STR_j.Length$ do
    if (rule==$STR_j$.Sort()) then
      $count++$
    end if
  end for
  if ($count > m/2$) then
    $MTR$.Add($STR_i$)
    $MTR$.Support = Min($STR_i$.Support)
    $MTR$.Confidence = Min($STR_i$.Confidence)
  end if
end for
End Algorithm
```

number of records, $q = 2^d - 1$ is the total item set count, d is the unique item count, and w is the maximum record length [34]. On the other hand, the time complexity of the FP-growth algorithm is $O(n.d^2)$, where n is the number of records and d is the number of unique items [35].

3.5. Advantages of MTARM

This paper introduces the concepts of single- and multitask rules and demonstrates the efficient implementation of the MTARM approach to discover patterns in the datasets. MTARM discovers global rules by combining the local rules obtained for each task. It has many advantages, including suitability for multitask problems, convenience for parallel processing, easy implementation, scalability, acceptable overall performance, and high privacy protection.

• **Task-based model:** Standard ARM algorithms fail to discover rules for multitask problems. They do not consider task-oriented information when finding association rules, and they ignore the correlation among the tasks. However, in some contexts, task-based patterns may be more meaningful than all the other ones. For example, two items may be seldom associated with the entire dataset; however, they can frequently exist together in some task-based data. Our algorithm jointly finds association rules under the concept of multitask learning. Therefore, it is particularly helpful when the tasks share important common information.

Recently, multitask learning has attracted attention because of the increased need for a simple and convenient interface in multiple domains. For example, let us take a school dataset [36–38] that consists of the examination records of 15,362 students from 139 secondary schools in London. The traditional ARM algorithm finds association rules from the whole dataset to analyze the general performance of all students together. However, MTARM discovers both local performances of students belonging to a specific school (single-task rules) and global effectiveness of schools (multitask rules), if each school corresponds to a task and, thus, there are 139 different tasks in total. The MTARM applies a majority voting mechanism to jointly consider the relationships among multiple tasks; therefore, it can be able to detect any intertask rules between domains and find more meaningful rules.

• **Parallel processing:** One of the main advantages of MTARM is that it may be executed in parallel. Because the dataset is filtered according to tasks, multitask rule computation may be parallelized and distributed on m nodes of a grid platform, i.e. one node for each task. In this way, MTARM can generate m parallel executions at the same time and retrieve single-task rules that are frequent in the data partitions. Therefore, significant gains may be possible by using a large number of processors. In parallel execution, data with m tasks are not divided into n parts ($m \neq n$); instead, they are filtered according to the tasks and, therefore, m data partitions should always be generated for m tasks. In this way, data are semantically divided into interrelated instances. Each task is identified and associated with a different role, subject, milestone, or subgoal; hence, all tasks can be executed independently. In the literature, there are several parallel ARM studies that have been conducted to solve the problem of discovering association rules from a large amount of data [26–30]. In one of these studies [29], three parallel algorithms, which assume a shared-nothing architecture using the Apriori algorithm, were proposed and executed on a parallelized system to evaluate their performances. In another study [28], the authors proposed a multiple local frequent pattern tree algorithm that was developed for parallelizing the FP-tree algorithm. In their paper, they stated that their proposed approach overcomes the lack of Apriori-based parallel ARM algorithms. Mueller [26] compared parallel ARM algorithms (PEAR and PPAR) depending on the different numbers of processors. The present studies in the literature show that the parallel implementation of ARM is approximately linearly proportional to the number of processors, and, therefore, it provides speedups for extremely large datasets compared with traditional ARM studies. Therefore, it is possible to provide similar performance for MTARM with its parallel version.

• **Easy implementation:** Because of the efficiency of multitask learning, various single-task approaches have been extended to multitask approaches, including k-nearest neighbors and naive Bayes methods. However, to consider the intertask relations, standard algorithms generally require modifications (nontrivial changes) in the training phase. In contrast, the key feature of our approach is that there is no need to make any modification to the underlying ARM algorithm. The main idea of the MTARM approach is simply to transform a multitask problem into a series of single-task problems. Therefore, MTARM has advantages in terms of implementation.

• **Scalability:** A new task can be easily inserted into the system without any modification. The results obtained from other tasks remain the same.

• **High overall performance:** MTARM only needs to scan small partitions of the dataset to generate all the single-task rules, and, therefore, it can perform operations with an adequate performance. This allows for good flexibility when generating task-based rules while avoiding the need for large amounts of data for modeling.

• **Privacy:** Discovery of hidden rules may often disclose some sensitive information. MTARM discovers local rules for each task individually like a distributed manner. Because it stores partial information at different partitions, data privacy is provided by the method itself.

4. Experimental study

The proposed MTARM method is a general approach that aims to jointly discover rules by considering multiple tasks. Therefore, this approach is implemented by selecting a base algorithm. In the experiments of this study, three well-known ARM algorithms (Apriori, FP-Growth, and Eclat) were chosen as base miners of MTARM separately, referred to as MT-Apriori, MT-FP-Growth, and MT-Eclat. Unlike the traditional versions of them, these algorithms can discover more knowledge by jointly analyzing all tasks and by considering the relations between the tasks.

This section presents the experimental results of the proposed algorithm (MTARM) on four benchmark datasets, namely the School, Solar Flare, Entree Chicago recommendation, and Coil 2000 datasets. These datasets are publicly available and have been previously used in multitask problems [36–41].

4.1. Dataset description

In this study, experiments were performed on various benchmark multitask learning datasets, namely the School Data, Solar Flare, Entree Chicago Recommendation, and Coil 2000 datasets.

School Dataset[1]**:** This dataset has been extensively utilized for appraising many multitask learning techniques [36–38]. The data came from the Inner London Education Authority (ILEA). The dataset consists of the examination records of 15,362 students from 139 secondary schools during the years 1985, 1986, and 1987. It has been used to study the effectiveness of schools and predict the performances of students belonging to a specific school. Each record consists of the task information, eight features, and the corresponding examination scores. The features are the year of examination, three student-specific features (gender, verbal reasoning (VR) band, and ethnic group), and four school-specific features (percentage of students eligible for free school meals, percentage of students in the VR band, school gender, and school denomination). Here, each school corresponds to a task; thus, there are 139 different tasks in total.

Solar Flare (SF) Dataset[2]**:** This dataset is publicly available and has also been used in the context of multitarget learning [39, 40]. It is concerned with how often solar flares occur in a 24-h period on the basis of the observed characteristics of the Sun. This dataset has nine feature variables describing the active regions on the Sun, three potential types of solar flare (common, moderate, and severe), and six task codes (B, C, D, E, F, and H).

[1]Goldstein H (1991). School Dataset [online]. Website `http://www.bristol.ac.uk/cmm/learning/support/datasets/` [accessed 17 February 2020].

[2]UCI (1989). Solar Flare Dataset [online]. Website `http://archive.ics.uci.edu/ml/datasets/solar+flare` [accessed 17 February 2020].

Entree Chicago Recommendation (ECR) Dataset[3]: This dataset, which is publicly available for noncommercial use, contains a record of user interactions with the Entree Chicago restaurant recommender system on the Web over a 2.5-year period. The system recommends restaurants to users according to restaurant features such as cuisine, price, style, glamor, and atmosphere. The dataset consists of eight parts (tasks), each of which contains restaurant features in a region (e.g., Atlanta, Los Angeles, and New Orleans) of the USA. The combined dataset consists of 4160 transactions, each of which is a subset of 256 feature codes, where each feature code represents a feature of the restaurant, e.g., “Mexican,” “romantic,” “good decor,” “excellent service,” “dancing,” and “parking.”

Insurance Company (Coil 2000) Dataset[4]: This dataset has been previously used in multitask learning [41]. It contains information on customers of an insurance company. It consists of 5822 customer records, and each record consists of 86 attributes, containing sociodemographic data (attributes 1–43) and product ownership (attributes 44–86), including the information of whether or not the customers have a caravan insurance policy. Here, we followed the experimental setup used in [41], and out of the 86 variables, we used 6 categorical features, leaving the remaining 80 features as the joint dataset. Our target variables consisted of attributes 1, 4, 5, 6, 44, and 86, which respectively indicate the customer subtypes, the customer age ranges, the customer main type, a discretized percentage of Roman Catholics in that area, the contribution from a third-party insurance, and a binary value that indicates whether the customer has a caravan insurance policy or not. The tasks have a different number of output labels, although they share the same input data.

Before the ARM process, the datasets were generally passed through a data preprocessing step. The data preprocessing step is one of the most essential steps of data mining, which makes the data suitable (ready) for the input demands of ARM algorithms. It mainly consists of five substeps: data cleaning, data integration, data transformation, data reduction, and data discretization. Data cleaning is an essential step to improve data quality by performing various operations such as filling missing values, determining outliers, and fixing inconsistent data. Data integration is the process of combining data residing at different sources and representing them in a single format. In this study, the datasets obtained from different sources were used separately. Data transformation is the process of transforming the data into the format that is required by the ARM algorithm; the data undergo normalization, generalization, and aggregation operations. Data reduction is the simplification of the complexity of the data by removing irrelevant attributes and focusing on meaningful ones. In this study, the numerical attributes in the Coil 2000 dataset were eliminated in the data reduction step, as described in [41]. Data discretization is the process whereby continuous data are split into discrete intervals, where each interval is mapped into a partition label. ARM algorithms require categorical data, as continuous values cannot be considered in finding relations. Therefore, ARM algorithms cannot directly deal with continuous values. For this reason, in our study, the continuous values in the datasets were converted into discrete values using a discretization method. The attributes that had continuous values were discretized into three bins using an equal frequency technique; therefore, the number of instances in each bin was approximately equal.

The basic characteristics of the datasets used in this study are given in Table 3. These are the number of instances, attributes, and tasks, as well as the domains they belong to. We used four datasets from different domains to demonstrate that our algorithm automatically and efficiently generates all rules in an efficient manner without prior domain knowledge.

[3]UCI (2000). ECR Dataset [online]. Website https://archive.ics.uci.edu/ml/datasets/Entree+Chicago+Recommendation+Data [accessed 17 February 2020].

[4]UCI (2000). Coil Data [online]. Website https://archive.ics.uci.edu/ml/datasets/Insurance+Company+Benchmark+(COIL+2000) [accessed 17 February 2020].

Table 3. Basic characteristics of the datasets.

Dataset	No. of instances	No. of attributes	No. of tasks	Domain
School	15,362	10	139	Education
Solar Flare (SF)	1066	13	6	Astronomy
ECR	4160	irregular	8	Food
Coil 2000	5822	6	6	Insurance

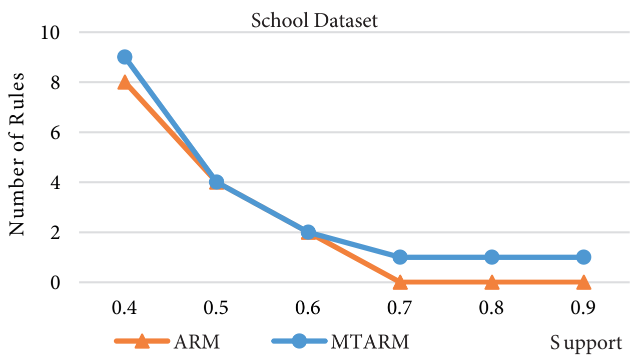
4.2. Experimental results

This section describes in detail the experiments performed with the proposed MTARM approach. We implemented the MTARM approach in the C# programming language by using the libraries provided by Borgelt [42]. We ran the algorithm with six parameters: $-s$ and $-c$ to set the minimum support and confidence, respectively; $-S$ to set maximum support to 100; $-tr$ to find the association rules; $-o$ to use the original definition of the support of a rule (if-part and then-part); and $-v$ to acquire a preferred output format. As recommended in [42], we produced association rules with a single item in the consequent (also called the "then-part"). We tested the algorithm on four benchmark multitask learning datasets described in the previous section. Experiments were performed on a desktop computer with an Intel Core i7 Duo 2.85-GHz processor and 8 GB of memory. In each experiment, the algorithms were executed 10 times and the average values are reported.

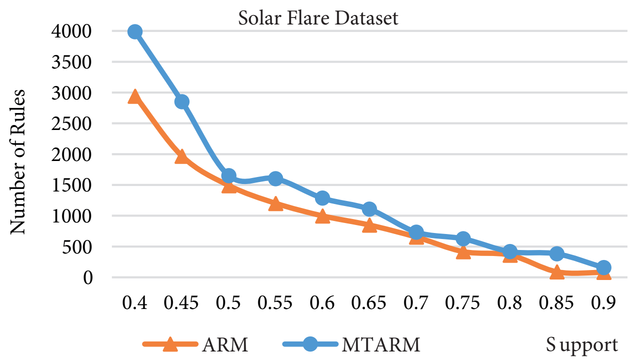
Using the proposed MTARM approach, we conducted six experiments to investigate the following: (i) the relation between the number of frequent rules and the minimum support settings, (ii) the distribution of k-item multitask rules, (iii) the comparison between single-task and multitask rules, (iv) the relation between execution time and number of tasks, (v) the relation between execution time and minimum support settings, and (vi) the relation between the execution times of the MT-Eclat and ARM algorithms (Eclat and SaM). All standard ARM algorithms (Apriori, FP-growth, and Eclat) discovered the same rules from the utilized dataset, but in different ways and at different execution times. Therefore, in the first four experiments, the Apriori algorithm was only used as a base miner in both the traditional ARM and the proposed MTARM approach. In the fifth experiment, all of them were compared in terms of running time, and in the last experiment, the Eclat and SaM algorithms were preferred for the comparison of execution times.

In the first experiment, we compared our proposed MTARM with the standard ARM using the Apriori algorithm. The relation between the number of frequent multitask rules and the support settings was determined by varying the minimum support thresholds with the same minimum confidence value. Figure 3 shows the comparative results obtained by the multitask and standard (single-task) ARM algorithms for the datasets at various minimum support (*MinSup*) values. The results clearly indicate that, for a given *MinSup*, MTARM always discovered more frequent rules than those of the conventional ARM algorithm for all datasets. For example, when *MinSup* was 0.25, MTARM discovered 28 frequent rules, whereas the standard ARM found 13 rules for the ECR dataset. In particular, MTARM outperformed the standard ARM by a factor of 2 to 3 when *MinSup* was less than 0.25 for the same dataset. In this respect, the single-task and multitask algorithms had comparable effects on the number of rules. The experiments herein revealed that it is possible to find more potential association rules when the task concept is considered. If the records related to multiple tasks are considered, then some missing rules can be detected.

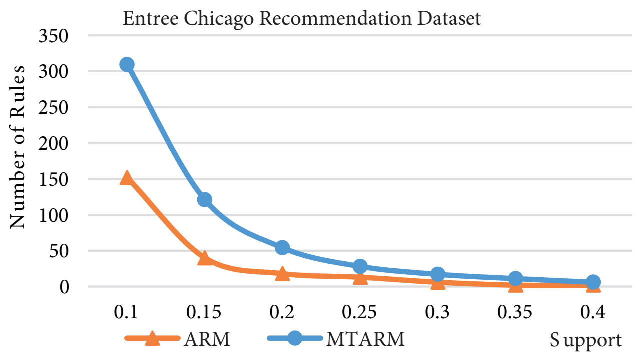
As shown in Figure 3, the number of patterns decreases virtually exponentially when the minimum support value increases. When the minimum support threshold decreases, the single-task approach can discover



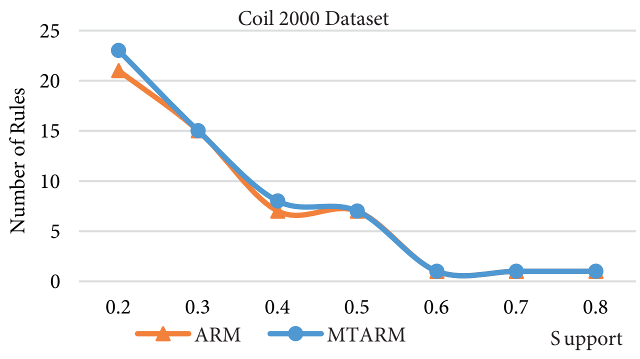
(a) Comparison between the algorithms on School Dataset



(b) Comparison between the algorithms on Solar Flare Dataset



(c) Comparison between the algorithms on ECR Dataset



(d) Comparison between the algorithms on Coil 2000 Dataset

Figure 3. Comparison between MTARM and standard ARM algorithms in terms of the number of frequent rules discovered.

enough knowledge from the data and, therefore, the multitask approach can obtain more rules. Thus, we can say that we can obtain a large amount of frequent rules by executing the ARM algorithm with small support values. For this reason, higher *MinSup* thresholds can be set to mine valuable frequent rules.

The MTARM approach performed better in finding patterns, especially when *MinSup* was low. This effect is significant as most frequent rules were found with small *MinSup* and *MinConf* values. The main reason for this result is that MTARM scans small partitions of the dataset to generate all the single-task rules, and, therefore, it can discover task-oriented rules with an adequate performance. In addition, MTARM can discover association rules that are frequent in some tasks but not throughout the entire dataset.

In the second experiment, the proposed MTARM approach with the Apriori algorithm was run on the Solar Flare dataset with support values ranging between 0.4 and 0.9 in increments of 0.05. Frequent multitask rules were obtained by the items whose support and confidence values were equal to or greater than the threshold level. However, the other items that had lower support and confidence values than the threshold level were discarded. Table 4 presents the numbers of obtained patterns separately, varying from one-item to seven-item rules. The acquired results show that the number of rules obtained from the dataset was high when the minimum support threshold was determined to be low, i.e. the number of four-item rules was 1180 when $MinSup = 0.4$. Nevertheless, the algorithm generated a comprehensive number of rules, i.e. the number of four-item rules was 16 when $MinSup = 0.9$. In this way, the collection of frequent rules can be compressed in a more manageable size.

Table 4. Number of k-item multitask rules with different support thresholds.

Support	Number of k-item multitask rules						
	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
0.4	152	525	1040	1180	774	273	40
0.45	140	432	776	810	498	168	24
0.5	88	249	436	460	294	105	16
0.55	79	231	420	455	294	105	16
0.6	79	228	392	370	180	35	0
0.65	77	210	340	305	144	28	0
0.7	64	159	232	185	78	14	0
0.75	62	147	204	150	54	7	0
0.8	58	120	140	80	18	0	0
0.85	58	117	128	65	12	0	0
0.9	36	57	48	16	0	0	0

Figure 4 plots the number of multitask rules according to their lengths, ranging from one to seven, for the Solar Flare dataset, with $MinSup = 0.5$ and $MinConf = 0.5$. The figure shows the experimental results for different rule lengths: one-item, two-item, three-item rule, etc. The four-item rule was found to have the greatest number of patterns. It can be observed that the numbers of the frequent two-item and five-item multitask rules are close to each other. As shown in Figure 4, the MTARM approach generally constructed a form quite similar to the bell curve. However, the skewness and kurtosis of the curve can change from dataset to dataset.

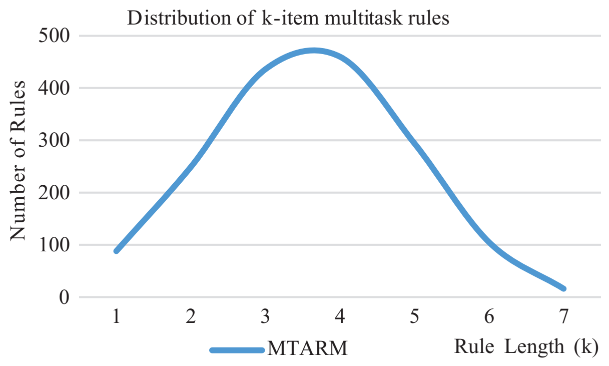


Figure 4. Distribution of the number of multitask rules according to their lengths.

In the third experiment, we compared the number of single-task and multitask rules. The MTARM approach with the Apriori algorithm was run on the Coil 2000 dataset with a support value ranging between 0.1 and 0.8 in increments of 0.1. Table 5 presents the changes in the number of frequent rules per task. According to the results, we can say that MTARM is particularly helpful when the tasks share significant commonalities.

In the fourth experiment, the proposed MTARM approach with the Apriori algorithm was executed on the dataset with varying numbers of tasks. Figures 5 and 6 show the efficiency of the algorithm when we again set the minimum support and confidence thresholds to 0.5. As expected, the execution times increased along with the number of tasks. For example, the School dataset contains 15,362 records, each of which belongs to one

Table 5. Number of frequent rules per task for the Coil 2000 dataset.

Support	Number of frequent rules per task						
	Task1	Task2	Task3	Task4	Task5	Task6	Multitask
0.1	174	76	92	90	106	285	61
0.2	89	26	27	29	29	73	23
0.3	48	15	17	15	16	50	15
0.4	26	8	7	7	5	21	8
0.5	23	7	5	7	4	12	7
0.6	4	1	2	4	1	7	1
0.7	1	1	1	1	1	5	1
0.8	1	1	1	1	1	4	1

of the 139 tasks, and, therefore, each task has about 110 records. According to Figure 5, a subset of the School dataset that includes only 40 tasks (4400 records) was executed in 4 s, whereas another subset that includes 100 tasks (11,000 records) was executed in 9 s. If we increase the number of tasks, the dataset will contain more records and, therefore, the execution time will also increase. The number of instances in the data of each task is also important because it greatly affects the processing time. The empirical results showed that the proposed MTARM approach requires a computation time that grows linearly with the number of tasks. Therefore, the computation time will still be reasonable if we consider a large number of tasks.

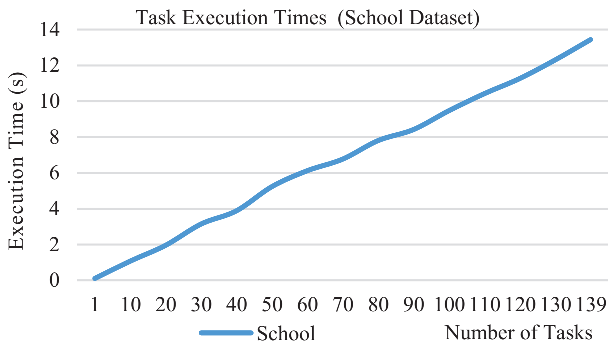


Figure 5. Execution times along with the number of tasks for the School dataset.

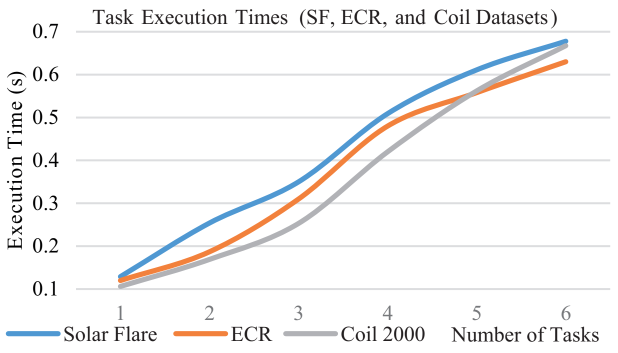
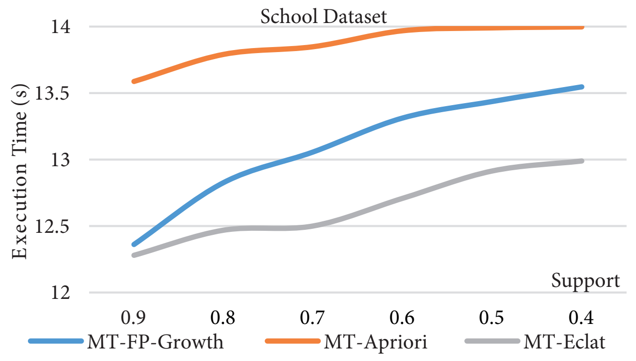


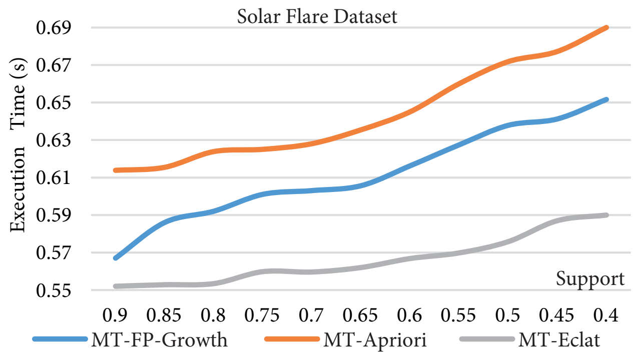
Figure 6. Execution times along with the number of tasks for the Solar Flare, ECR, and Coil 2000 datasets.

In the fifth experiment, the relation between execution time and support settings was determined by varying the thresholds. In the first four experiments, we ran the MTARM approach with the FP-growth algorithm [9] as a basic rule generator on the datasets horizontally partitioned according to the tasks; the latter algorithm was selected because of its popularity and efficiency. However, in the last experiment, we used alternative ARM algorithms such as Apriori [8] or Eclat [10] to compare with each other. Figure 7 presents the results obtained for the datasets with varying minimum support values. Four main points emerge from this figure.

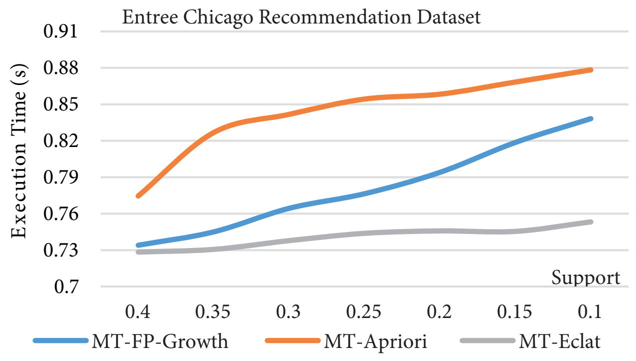
First, in the experiments, the running time increased as the minimum support value decreased. Lower thresholds yielded more multitask rules. Therefore, the minimum support and confidence values were the main factors that affected the execution time.



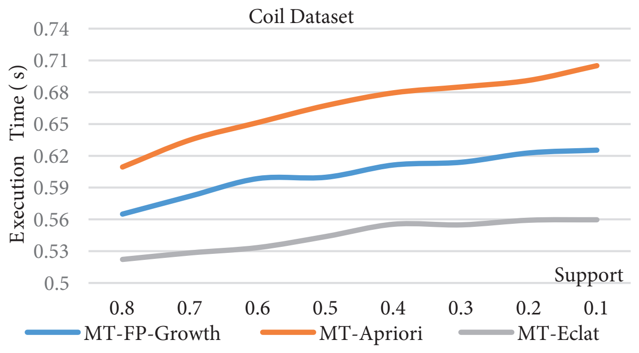
(a) Comparison between the algorithms on School Dataset



(b) Comparison between the algorithms on Solar Flare Dataset



(c) Comparison between the algorithms on ECR Data set



(d) Comparison between the algorithms on Coil 2000 Data set

Figure 7. Comparison between the multitask-based ARM algorithms in terms of computational costs.

Another important factor that affected the execution time was the total number of records in the dataset. For example, because the School dataset is larger than the other datasets, the time required to find multitask rules from it is also higher.

The number of tasks also affected the running time of the multitask-oriented ARM algorithm, such that the execution times increased along with the number of tasks. At each different number of tasks, the dataset was filtered to include only records related to these tasks. For example, the School dataset was first filtered with the T1 task (110 records), then filtered with the T1 and T2 tasks (220 records), and then filtered according to these three tasks (330 records), etc. Thus, as the number of tasks increases, the subdataset size also increases. Therefore, we can say that the multitask-oriented ARM algorithm runs faster with a small number of tasks.

Finally, the ARM algorithm selected to find frequent rules also affected the execution time. Figure 3 compares the multitask-based FP-growth, Apriori, and Eclat algorithms. It is obvious that MT-Eclat slightly outperforms the other two methods. The MT-FP-growth algorithm has a comparable speed with that of MT-Eclat on all the datasets and is much faster than the MT-Apriori algorithm. As long as the minimum support threshold decreases, the running time of all multiple task approaches increases slightly. Nevertheless, the execution time of MT-Apriori increases more greatly.

In the last experiment, parallel versions of the MTARM approach with the Eclat (MT-Eclat) and ARM algorithms (Eclat and SaM) were executed on the Coil 2000, Solar Flare, and ECR datasets. The corresponding execution times are listed in Table 6. The results indicated that, when the number of tasks increased, multitask

Table 6. Comparison between the parallel versions of the algorithms in terms of execution times.

	Execution times of the parallel versions of the algorithms (seconds)											
	Coil 2000			Solar Flare			School			ECR		
S	MT-Eclat	Eclat	SaM	MT-Eclat	Eclat	SaM	MT-Eclat	Eclat	SaM	MT-Eclat	Eclat	SaM
0.4	0.365	0.179	0.223	0.378	0.247	0.258	4.879	0.274	0.287	0.431	0.227	0.235
0.5	0.359	0.170	0.217	0.367	0.238	0.245	4.723	0.264	0.272	0.425	0.215	0.220
0.6	0.349	0.162	0.189	0.360	0.221	0.231	4.685	0.256	0.267	0.419	0.186	0.203
0.7	0.341	0.179	0.186	0.355	0.200	0.223	4.660	0.236	0.258	0.396	0.183	0.188
0.8	0.335	0.177	0.174	0.340	0.197	0.215	4.554	0.223	0.247	0.385	0.173	0.185
0.9	0.327	0.162	0.171	0.331	0.181	0.193	4.330	0.198	0.211	0.376	0.169	0.177

mining led to a more time-consuming process. Although the execution times were slightly lower in the parallel versions of the traditional algorithms, task-related data should be processed by considering the task information in terms of semantic analysis.

Table 7 shows some examples of the multitask association rules discovered by the MTARM approach from the School, Solar Flare, ECR, and Coil 2000 datasets. The number of items in the rule (the length of the rule) is given in the first column of the table. For example, the rule { vrBandOfStudent = 3 → schoolDenomination = England } is a frequent two-item multitask rule discovered from the School dataset. The results express the relationships among the features in the datasets. For example, the rule { examScore = high gender = female → ethnic = ESWI } indicates that 10.17% of all students were female; were born in England, Scotland, Wales, or Ireland (ESWI); and had high exam scores.

Table 7. Examples of the multitask association rules generated by MTARM.

Length	Multitask rules	S (%)	C (%)	Dataset (%)
2-item	{ vrBandOfStudent = 3 → schoolDenomination = maintained}	37.5	100	School
3-item	{ examScore = high gender = female → ethnic = ESWI }	10.17	32.43	School
2-item	{ Cflare = 0 → activity = reduced }	60.25	82.76	Solar Flare
3-item	{ Mflare = 0 previousFlareActivity = 1 → Xflare = 0 }	76.84	97.37	Solar Flare
4-item	{ areaLarge ≤ 5 historicallyComplex = 1 previousFlareActivity = 1 → Cflare = 0 }	62.56	86.84	Solar Flare
5-item	{ area = small areaLarge ≤ 5 Mflare = 0 previousFlareActivity = 1 → Xflare = 0 }	67.44	96.67	Solar Flare
6-item	{ areaLarg ≤ 5 complex = no Xflare = 0 previousFlareActivity = 1 spotDistribution = I → area = small }	40	81.48	Solar Flare
7-item	{ area = small complex = no areaLarge ≤ 5 evolution = nogrowth Mflare = 0 Xflare = 0 →previousFlareActivity = 1 }	40	91.3	Solar Flare
2-item	{ 075 → 250 }	30.92	61.97	ECR
3-item	{ 250 192 → 191 }	24.89	93.16	ECR
4-item	{ 053 075 250 → 205}	11.64	89.47	ECR
5-item	{ 075 205 191 250 → 192}	9.59	79.25	ECR
2-item	{ romanCatholic = 0 → caravan = 0 }	49.17	93.53	Coil 2000
3-item	{ contribution = 2 romanCatholic = 0 → caravan = 0 }	14.87	88.89	Coil 2000
4-item	{ customerMainType = 8 customerSubType = 33 romanCatholic = 0 → caravan = 0 }	8.95	91.43	Coil 2000

5. Conclusion and future works

Standard ARM algorithms discover rules from an entire dataset, they do not perform task-based searches, and they ignore intertask relations. This paper proposes a novel algorithm, named multitask association rule miner (MTARM), that tends to discover rules by jointly considering multiple tasks. It also introduces two novel concepts: single-task rule and multitask rule. The proposed MTARM approach consists of two phases. In the first phase, the algorithm discovers highly frequent local rules (single-task rules) for each task individually, and in the second phase, these local rules are combined to produce the global result (multitask rules).

Experiments were conducted on several real-world multitask learning datasets. Experimental results revealed that our novel algorithm discovers more information than that of traditional ARM algorithms by considering the relationships among multiple tasks available. MTARM can discover association rules that are frequent in some tasks but not throughout the entire dataset. Therefore, it can be effectively used in real-world applications for knowledge discovery when the task concept is considered.

Discovering association rules on a huge amount of data is a long and time-consuming process because it gives a huge number of different rules. This process also requires a significant storage capacity. To handle this challenge, we can implement closed and maximal versions of MTARM in future research to present concise representations of all association rules. Moreover, mining multitask-based sequential patterns can also be one of the future works. Therefore, multitask-based patterns can be discovered by considering the sequential ordering between items.

References

[1] Ai D, Pan H, Li X, Gao Y, He D. Association rule mining algorithms on high-dimensional datasets. Artificial Life and Robotics 2018; 23 (3): 420-427. doi: 10.1007/s10015-018-0437-y

[2] Moodley R, Chiclana F, Caraffini F, Carter J. Application of uninorms to market basket analysis. International Journal of Intelligent Systems 2019; 34 (1): 39-49. doi: 10.1002/int.22039

[3] Gao T, Cheng B, Chen J, Xue H, Duan L et al. Interest-aware service association rule creation for service recommendation and linking mode recommendation in user-generated service. IEEE Access 2018; 6: 57721-57737. doi: 10.1109/access.2018.2873708

[4] Hela S, Amel B, Badran R. Early anomaly detection in smart home: a causal association rule-based approach. Artificial Intelligence in Medicine 2018; 91: 57-71. doi: 10.1016/j.artmed.2018.06.001

[5] Agapito G, Guzzi P, Cannataro M. Parallel extraction of association rules from genomics data. Applied Mathematics and Computation 2019; 350: 434-446. doi: 10.1016/j.amc.2017.09.026

[6] Santos F, Domingues M, Sundermann C, Carvalho V, Moura M et al. Latent association rule cluster based model to extract topics for classification and recommendation applications. Expert Systems with Applications 2018; 112: 34-60. doi: 10.1016/j.eswa.2018.06.021

[7] Naik A, Rangwala H. Multi-task learning. In: Naik A (editor). Large Scale Hierarchical Classification: State of the Art. Cham, Switzerland: Springer International Publishing, 2019, pp. 75-88.

[8] Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases VLDB'94; San Francisco, CA, USA; 1994. pp. 487-499.

[9] Han J, Pei J, Yin Y, Mining frequent patterns without candidate generation. In: Proceedings of the 2000 International Conference on Management of Data ACM SIGMOD; Dallas, TX, USA; 2000. pp. 1-12.

[10] Zaki M. Scalable algorithms for association mining. IEEE Transactions on Knowledge and Data Engineering 2000; 12 (3): 372-390. doi: 10.1109/69.846291

[11] Kao W, Lo C, Li K, Yang H, Yan J. Research of mining multi-level association rule models. In: Hung J, Yen N, Li KC (editors). Frontier Computing, Lecture Notes in Electrical Engineering. Singapore: Springer, 2016, pp. 279-288.

[12] Alsukhni E, Al Eroud A, Saifan A. A hybrid pre-post constraint-based framework for discovering multi-dimensional association rules using ontologies. International Journal of Information Technology and Web Engineering 2019; 14 (1): 112-131. doi: 10.4018/ijitwe.2019010106

[13] Tran MD, d'Amato C, Nguyen BT, Tettamanzi AGB. Comparing rule evaluation metrics for the evolutionary discovery of multi-relational association rules in the semantic web. In: Castelli M, Sekanina L, Zhang M, Cagnoni S, Garcia-Sanchez P (editors). Genetic Programming, Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2018, pp. 289-305.

[14] Kamaruddin S, Yusof Y, Husni H, Al Refai M. Text classification using modified multi class association rule. Jurnal Teknologi 2016; 78 (8-2): 163-170. doi: 10.11113/jt.v78.9553

[15] Song A, Ding X, Chen J, Li M, Cao W et al. Multi-objective association rule mining with binary bat algorithm. Intelligent Data Analysis 2016; 20 (1): 105-128. doi: 10.3233/ida-150796

[16] Yang L, Li Z, Luo G. MH-ARM: A multi-mode and high-value association rule mining technique for healthcare data analysis. In: 2016 International Conference on Computational Science and Computational Intelligence (CSCI); Las Vegas, NV, USA; 2016. pp. 122-127.

[17] Wang D, Xie Q, Huang D, Yuan H, Lu H et al. Multi-granule association rule mining based on quantitative concept lattice. In: Lei J, Wang FL, Li M, Luo Y (editors). Network Computing and Information Security, Communications in Computer and Information Science. Berlin, Germany: Springer, 2012, pp. 268-274.

[18] He R, Xiong N, Yang L, Park J. Using multi-modal semantic association rules to fuse keywords and visual features automatically for web image retrieval. Information Fusion 2011; 12 (3): 223-230. doi: 10.1016/j.inffus.2010.02.001

[19] Liu P, Li X, Feng Y. An algorithm of multi-level fuzzy association rules mining with multiple minimum supports in network faults diagnosis. In: 2013 Ninth International Conference on Natural Computation (ICNC); Shenyang, China; 2013. pp. 884-888.

[20] Atteya WA, Dahal K, Hossain MA. Multi-agent association rules mining in distributed databases. In: Gaspar-Cunha A, Takahashi R, Schaefer G, Costa L (editors). Soft Computing in Industrial Applications, Advances in Intelligent and Soft Computing. Berlin, Germany: Springer, 2011, pp. 305-314.

[21] Zhang Y, Yang Q. An overview of multi-task learning. National Science Review 2017; 5 (1): 30-43. doi: 10.1093/nsr/nwx105

[22] Thung K, Wee C. A brief review on multi-task learning. Multimedia Tools and Applications 2018; 77 (22): 29705-29725. doi: 10.1007/s11042-018-6463-x

[23] Zhang C, Zhao P, Hao S, Soh YC, Lee BS. Distributed multi-task classification: a decentralized online learning approach. Machine Learning 2017; 107 (4): 727-747. doi: 10.1007/s10994-017-5676-y

[24] Zhang X, Zhang X, Liu H, Liu X. Multi-task clustering through instances transfer. Neurocomputing 2017; 251: 145-155. doi: 10.1016/j.neucom.2017.04.029

[25] Zhang Z, Zhou J. Multi-task clustering via domain adaptation. Pattern Recognition 2012; 45 (1): 465-473. doi: 10.1016/j.patcog.2011.05.011

[26] Mueller A. Fast Sequential and Parallel Algorithms for Association Rule Mining: A Comparison. Technical Report. College Park, MD, USA: University of Maryland, 1995.

[27] Zaki MJ. Parallel and distributed association mining: a survey. IEEE Concurrency 1999; 7 (4): 14-25. doi: 10.1109/4434.806975

[28] Zaiane OR, El-Hajj M, Lu P. Fast parallel association rule mining without candidacy generation. In: Proceedings of IEEE International Conference on Data Mining; San Jose, CA, USA; 2001. pp. 665-668.

[29] Agrawal R, Shafer JC. Parallel mining of association rules. IEEE Transactions on Knowledge and Data Engineering 1996; 8 (6): 962-969. doi: 10.1109/69.553164

[30] Zaki MJ. Large-scale parallel data mining. In: Zaki MJ, Ho CT (editors). Lecture Notes in Computer Science. San Jose, CA, USA: Springer, 2000, pp. 1-23.

[31] Sakhapara AM, Bharathi HN. Comparative study of Apriori algorithms for parallel mining of frequent itemsets. International Journal of Computer Applications 2014; 90 (8): 21-24.

[32] Borgelt C. Frequent item set mining. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2012; 2 (6): 437-456. doi: 10.1002/widm.1074

[33] Borgelt C, Wang X. (Approximate) Frequent item set mining made simple with a split and merge algorithm. In: Laurent A, Lesot M (editors). Scalable Fuzzy Algorithms for Data Management and Analysis: Methods and Design. Hershey, PA, USA: IGI Global, 2009, pp. 254-272.

[34] Olmezogullari E, Ari I. Online association rule mining over fast data. In: 2013 IEEE International Congress on Big Data; Santa Clara, CA, USA; 2013. pp. 2379-7703.

[35] Kadappa V, Nagesh S. Local support-based partition algorithm for frequent pattern mining. Pattern Analysis and Applications 2019; 22 (3): 1137-1147. doi: 10.1007/s10044-018-0752-x

[36] Xue W. Weighted feature-task-aware regularization learner for multitask learning. Pattern Analysis and Applications 2020; 23 (1): 253-263. doi: 10.1007/s10044-019-00781-8

[37] Kong Y, Shao M, Li K, Fu Y. Probabilistic low-rank multitask learning. IEEE Transactions on Neural Networks 2018; 29 (3): 670-680. doi: 10.1109/tnnls.2016.2641160

[38] Liu C, Zheng C, Qian S, Wu S, Wong H. Encoding sparse and competitive structures among tasks in multi-task learning. Pattern Recognition 2019; 88: 689-701. doi: 10.1016/j.patcog.2018.12.018

[39] Levatic J, Kocev D, Ceci M, Dzeroski S. Semi-supervised trees for multi-target regression. Information Sciences 2018; 450: 109-127. doi: 10.1016/j.ins.2018.03.033

[40] Borchani H, Varando G, Bielza C, Larranaga P. A survey on multi-output regression. WIRES Data Mining and Knowledge Discovery 2015; 5 (5): 216-233. doi: 10.1002/widm.1157

[41] Parameswaran S, Weinberger K. Large margin multi-task metric learning. In: Proceedings of Advances in Neural Information Processing Systems NIPS'10; Vancouver, British Columbia, Canada; 2010. pp. 1867-1875.

[42] Borgelt C. Simple algorithms for frequent item set mining. In: Koronacki J, Ras ZW, Wirezchon ST, Kacprzyk J (editors). Advances in Machine Learning II, Studies in Computational Intelligence. Berlin, Germany: Springer, 2010, pp. 351-369.