Swarm and Evolutionary Computation 29 (2016) 1–12

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

A hierarchical heterogeneous ant colony optimization based approach for efficient action rule mining

N.K. Sreeja [a,*], A. Sankar [b]

[a] Department of Computer Applications, Sri Krishna College of Technology, Coimbatore 641042, India
[b] Department of Computer Applications, PSG College of Technology, Coimbatore 641004, India

ARTICLE INFO

Article history:
Received 26 March 2015
Received in revised form
11 January 2016
Accepted 29 February 2016
Available online 5 March 2016

Keywords:
Action rule mining
Hierarchical heterogeneous ant colony optimization
Reclassification

ABSTRACT

Most data mining algorithms aim at discovering customer models and classification of customer profiles. Application of these data mining techniques to industrial problems such as customer relationship management helps in classification of customers with respect to their status. The mined information does not suggest any action that would result in reclassification of customer profile. Such actions would be useful to maximize the objective function, for instance, the net profit or minimizing the cost. These actions provide hints to a business user regarding the attributes that have to be changed to reclassify the customers from an undesirable class (e.g. disloyal) to the desired class (e.g. loyal). This paper proposes a novel algorithm called Hierarchical Heterogeneous Ant Colony Optimization based Action Rule Mining (HHACOARM) algorithm to generate action rules. The algorithm has been developed considering the resource constraints. The algorithm has ant agents at different levels in the hierarchy to identify the flexible attributes whose values need to be changed to mine action rules. The advantage of HHACOARM algorithm is that it generates optimal number of minimal cost action rules. HHACOARM algorithm does not generate invalid rules. Also, the computational complexity of HHACOARM algorithm is less compared to the existing action rule mining methods.

1. Introduction

Data mining focuses on discovering distributional knowledge from the underlying data. Data mining models such as Bayesian classifiers, decision trees, Support Vector Machines and association rules have been applied to various applications such as Customer Relationship Management (CRM) [21]. Existing data mining algorithms aim at discovery of customer models, producing distribution information on customer profiles. Applying such techniques to industrial problems such as CRM facilitates the process of classification of customers as loyal and disloyal. However, the necessity of reclassification of a disloyal customer to a loyal one is the need of the hour in such industrial applications. The existing data mining models do not focus on reclassification of data. To improve customer relationship, the enterprise must know what actions to take to reclassify the customers from an undesired status (such as disloyal) to a desired one (such as loyal customers) [21].

To extract actionable knowledge, the resource constraints must be considered. As the enterprises are increasingly constrained by cost cutting, the enterprise imposes serious limitations on the number of customer segments it can take, or in the number of actions that can be generated. Therefore, the cost as well as the benefit of actions to the enterprise has to be considered during the decision making process. Even though a large number of actions may be generated for each customer, the decision maker of the enterprise decides which of these rules are feasible, as actions do not depend only on a particular customers' situation, but also on other customers who would be benefited from the same action. The actions generated by the decision maker are termed action rules.

An action rule is defined as a rule extracted from a decision table (dataset) that describes a possible transition of objects from one state to another with respect to a distinguished attribute called a decision attribute [13]. The notion of an action rule was proposed in [8] and extended in [13]. Initially, action rule mining was based on comparing profiles of two groups of targeted objects those that are desirable and those that are undesirable [13].

Action rules may be discovered using a rule-based or object-based approach. Rule-based actionable patterns are built on the foundations of pre-existing rules whereas object-based approach extracts actionable patterns directly from a decision table [13,18,8]. It is assumed that the attributes in a decision table are either stable or flexible. Stable attributes are attributes whose

* Corresponding author.
E-mail addresses: sreeja.n.krishnan@gmail.com (N.K. Sreeja), dras@mca.psgtech.ac.in (A. Sankar).

http://dx.doi.org/10.1016/j.swevo.2016.02.004
2210-6502/

values cannot be changed. Examples of stable attributes include gender, date of birth etc. Attributes whose values can be changed are termed as flexible attributes. Examples of flexible attributes include interest rate offered by a bank, sales percentage etc [6]. An action rule provides suggestions to a business user regarding the flexible attributes that have to be changed to reclassify the objects from an undesirable class to a desirable class. Examples of action rules include improving the annual income of a person from low to high, improving from a high-investment, low-return business to a low-investment, high-return business etc.

In this paper, a Hierarchical Heterogeneous Ant Colony Optimization based Action Rule Mining (HHACOARM) algorithm is proposed for discovering action rules from a decision table. According to this algorithm, the ant agents at different levels identify the flexible attributes whose values need to be changed to mine action rules. The advantage of the approach in comparison with the existing algorithms for action rule mining is that it generates optimal number of action rules with minimal cost. The computational complexity of HHACOARM algorithm is much less compared to the existing algorithms.

The support and confidence of the action rule are not considered in HHACOARM algorithm, since these interestingness measures suffer from certain drawbacks as stated below. Selection of the threshold for support and confidence is a tricky problem. The drawback of support is the rare item problem which occurs when the support threshold is too high [1]. Items that occur very infrequently in the dataset are pruned although they would still produce interesting and potentially valuable rules [4]. In such a case, the action rules generated may not reclassify the undesirable objects to a desirable one. Selection of a low support threshold would generate large number of action rules, some of which may not prove useful for reclassification. The confidence measure ignores the support of the objects in the rule consequent [11]. Consequents with higher support will automatically produce higher confidence values even if there does not exist an association between the objects. Therefore, the support and confidence measures affect the quality of the action rules generated.

The paper is organized as follows. Section 2 gives an overview of the related work. Section 3 explains Action rules in an information system. Section 4 describes the cost of the action rules. Section 5 describes Hierarchical Heterogeneous Ant Colony Optimization based Action Rule Mining. Section 6 gives a case study. Section 7 discusses the experimental results. Section 8 compares HHACOARM algorithm with the existing algorithms for action rule mining. Finally, Section 9 presents the conclusion.

2. Related work

The notion of action rules was initially proposed by Liu et al. [8] and the work was extended by Ras & Wieczorkowska [13]. The work proposed in [8] and [13] used a rough set based classification process to discover action rules. According to this approach, each action rule is constructed from two rules, extracted earlier, defining different profitability classes.

Ling et al. [9] and Yang et al. [21] proposed techniques on post-processing decision trees to maximize expected net profit. They also incorporated the cost of each action in maximizing expected net profit. However, the drawback is that their methods are tightly coupled with decision trees and do not provide rules explicitly, which is hard to use in practice [6].

Yang and Chen [20] proposed an approach that uses 'role models' for generating advice and plans. These role models are typical cases that form a case base and can be used for customer advice generation. In this approach, a nearest neighbor algorithm is used to find a cost effective and highly probable plan to switch a customer to the most desirable role model. However, the computational cost incurred in generating actions is very high. Furthermore, it is stated in [6], that the actions generated according to only its nearest neighbor sometimes will miss better lower cost action rules.

Ras and Wieczorkowska [13] proposed action rules constructed from certain pairs of association rules. However, interventions introduced by Greco [5] and the concept of information changes proposed by Skowron and Synak [14] are conceptually very similar to action rules. The standard data mining methods such as Learning from Examples based on Rough Sets (LERS) [7,8] or Extracting Rules from Incomplete Decision system (ERID) [5] is used for incomplete datasets which extract only the shortest rules. However, the drawback of this approach is that some of the meaningful action patterns are easily missed.

He et al. [6] have stated that the previous works for action rule mining does not contain a formal definition on the problem of action rule mining. Hence, some meaningful action rules could be missed in these classification based techniques and thus existing algorithms cannot specify when and how the correct and complete underlying actions rules are discovered. Thus, Mining Action Rules From Scratch (MARFS) algorithm was proposed [6]. The MARFS algorithm generates all action rules and the correctness of these action rules is proved in [6]. However, MARFS algorithm generates large number of action rules. Also, MARFS algorithm is not fully optimized and algorithm with lower computational complexity can be developed [6].

Djellel Eddine et al. [2] have proposed a method for Frequent Association Action Rule Mining (FAARM) using a Frequent Pattern (FP) tree. The algorithm constructs a FP tree from an action table. Frequent patterns are generated from the FP tree from which association action rules are generated. Although, it is stated in [2] that FAARM algorithm achieves high performance, the computational complexity increases with the height of the tree.

Peng et al [12] have proposed utility based approach of action rule mining called Mining Valuable Action Rules (MVAR). MVAR algorithm generates action rules in two phases. In the first phase, all frequent action sets are generated from which candidate action rules with frequent action sets are generated. The generation of frequent action sets is similar to the procedure of generating frequent itemsets in Apriori algorithm for association rule mining. The second phase generates valuable action rules based on the candidate rules generated from the previous phase. However, the computational complexity of MVAR algorithm is higher.

3. Action rules in an information system

An information system can be represented as a pair $S=(O, A)$ where O is a non-empty, finite set of objects, and A is a non-empty, finite set of attributes or features of the objects.

Decision tables are a special case of information system. In a decision table, the attributes are partitioned in to a set of conditions such as stable and flexible. The decision table also contains the decision attribute (d). Therefore, a decision table can be represented as an information system as given in Eq. (1)

$$S=(O, A_s \cup A_f \cup \{d\}) \quad \text{where} \quad d \notin (A_s \cup A_f) \text{ and } A_s \neq A_f \tag{1}$$

where A_s and A_f are a set of stable and flexible attributes respectively. Rules are extracted from a decision table given preference to flexible attributes [6].

Consider a decision table with eight objects as shown in Table 1. Each object in the table has three attributes 'a', 'b', 'c' and a decision attribute 'd'. Among these, the attributes 'a' and 'c' are assumed to be stable; 'b' is assumed to be a flexible attribute. The decision attribute 'd' can take values 'L' or 'H', where H denotes