High-Utility Action Rules Mining

Lukáš Sýkora[1], Tomáš Kliegr[1] and Kateřina Hrudková[1]

[1]*Department of Information and Knowledge Engineering, Faculty of Informatics and Statistics, Prague University of Economics and Business*

Abstract

High-utility action rule mining is a data mining task that aims to generate rules that provide the user with information on which actions might be the most profitable. The actions correspond to proposed changes in attribute values that may move some instances originally predicted to belong to the undesirable class to the desirable class. Through a list of conditions on attribute values, an action rule delimits a set of instances to which the proposed action applies to. If this change is implemented, the action rule implies that the classification of a part of these instances will change. The profit of an action rule is calculated based on a user-set utility table specifying the costs of actions and the benefit of moving an instance to the desirable class. In this paper, we report on a new extension of the ActionRules package written in Python that implements high-utility action rule mining.

Keywords

Action Rules, Explainable Machine Learning, High-Utility Mining, Rule Learning, High-Utility Action Rules Mining

1. Introduction

In this article, we present a method combining two existing paradigms – action rules mining and high-utility itemset mining. Action rule mining outputs a specific form of classification rules. Classification rules are learnt from labelled training data to make predictions of the user-designated target class attribute based on values of the other attributes for a given instance. Action rules additionally indicate which changes to attribute values should be made for the classification to "improve". However, the original approach to action rule mining [1] does not take into account either the costs of the recommended actions or the resulting benefits. Drawing inspiration from high-utility itemset mining [2], profit of action rules can be calculated from a table with user-set utility values. Combining the aspects of action rules with the elements of high-utility itemset mining makes it possible to generate only rules that exceed a user-defined minimum utility threshold.

Our review has not found any public software implementation of high-utility action rule mining. In this paper, we thus introduce an extension to the ActionRules package [3] that allows to filter the discovered association rules according to the calculated profit. The extension is accompanied by a Jupyter Notebook, demonstrating a use case of high-utility action rules mining. The package is available under an open license (MIT).

RuleML+RR'22: 16th International Rule Challenge and 6th Doctoral Consortium, September 26–28, 2022, Virtual
lukas.sykora@vse.cz (L. Sýkora); tomas.kliegr@vse.cz (T. Kliegr); hruk03@vse.cz (K. Hrudková)
0000-0002-6700-5610 (L. Sýkora); 0000-0002-7261-0380 (T. Kliegr)

The structure of the work is the following. Section 2 provides an introduction to action rule mining. Section 3 provides a brief account of related work on high-utility action rule mining. Section 4 describes the proposed approach to computing utility of action rules. Section 5 provides a demonstration on a specific example. Section 6 describes the implementation. Conclusions summarise the contributions, point at the publicly available implementation and outline future work.

2. Brief Introduction to Action Rule Mining

Algorithmically, there are two principal approaches to action rules mining [4]: a rule-based approach and an object-based approach.

The *rule-based approach* (also referred to as the *loosely coupled framework* [5]) is divided into two independent phases. In the first phase, class association rules are mined by any suitable association rule mining algorithm (for example, Apriori [6] modified for Classification Association Rules [7]). The second phase generates action rules from candidate pairs of class association rules meeting user-defined quality settings.

The *object-based approach* (also referred to as the *tightly coupled framework* [5]) generates action rules without the pre-mining of class association rules. The algorithms include MARFS1 (Apriori-like algorithm), LERS (Learning from Examples based on Rough Sets) or ERID (Algorithm for Extracting Rules from Incomplete Decision System) [4].

In our work, we adopt the rule-based approach for which the two phases are described in the following.

2.1. Phase 1: Mining of Class Association Rules

In the following, we will informally define common concepts used in association rule mining. Class association rules are mined from a transaction database D containing N transactions. Each transaction t contains a set of items. In the input for action rule mining is typically a data table, which needs to be transformed into a transaction-format for the purpose of association rule mining. This transformation needs to preserve the connection between the original attribute and the generated items. We will therefore say that an item is *derived* from an attribute if it was generated from it.

Example illustrating transactions and items
The demonstration dataset in Table 1 (Section 6) contains rows corresponding to employees, which are described by an id and attributes (columns) $'Department'$, $'Salary'$ and $'Attrition'$. For example, the employee in the first row can be represented as a transaction $t_1 = \{Department : Sales, Salary : Medium, Attrition : False\}$. The first item in the transaction is derived from attribute $'Department'$, the second item from $'Salary'$ and third item from $'Attrition'$.

A formal representation of the type of the classification rule considered in our work is:

$$r_i : \phi \Rightarrow \psi, \tag{1}$$

where the antecedent ϕ is a set containing at least one item and the consequent ψ contains one item. A set of items is also referred to as an *itemset*.

As an input to the mining algorithm, the user needs to define the set of attributes A in the input data table that can serve as predictors. Only items in D derived from these attributes can be used for the antecedent ϕ. The user also designates the target attribute Y. Only items in D derived from Y can be used for the consequent ψ. Note that for utility action rule mining as defined in our work, Y is required to be a binary attribute.

Additionally, the user needs to define thresholds for support and confidence metrics. Only rules meeting these minimum values are returned.

The absolute support of a rule corresponds to the number of transactions in D that contains all items from the antecedent and the consequent of the given rule:

$$sup_a(\phi \Rightarrow \psi) = |t \in D : \psi \subset t \wedge \phi \subset t|. \tag{2}$$

The relative support normalizes this number by the total number of transactions:

$$sup(\phi \Rightarrow \psi) = \frac{|t \in D : \psi \subset t \wedge \phi \subset t|}{N}. \tag{3}$$

The confidence of a rule corresponds to the ratio between the number of transactions that contain all items from both the antecedent and the consequent to the number of transactions that contain all items from the antecedent:

$$conf(\phi \Rightarrow \psi) = \frac{sup_a(\phi \Rightarrow \psi)}{|t \in D : \phi \subset t|} \tag{4}$$

2.2. Phase 2: Generation of Action Rules

In the second step, action rules are formed from pairs of class association rules. For this, the user needs to additionally specify the set of flexible attributes $F \subset A$, the set of stable attributes $S \subset A$. The user also needs to designate which value $y_d \in Y$ of the binary target attribute Y is desirable. The other value is considered as undesirable.

This setting is used to preprocess candidate class association rules mined in phase 1. Consider a classification rule $r_i : \phi_i \implies \psi_i$ with $q = |\phi_i|$ items in the antecedent. The set ϕ_i is partitioned into a subset of items derived from flexible attributes in F, which we denote as $\{\alpha_i^{(1)}, \dots, \alpha_i^{(k)}\}$, and a set of items defined over stable attributes in S which we denote as $\{\omega_i^{(k+1)}, \dots, \omega_i^{(q)}\}$.

A pair of class association rules $r_1 : \phi_1 \rightarrow \psi_1$ and $r_2 : \phi_2 \rightarrow \psi_2$ need to fulfil several properties to form an action rule:

1. The rules need to have different consequents, where $\psi_1 \neq \psi_2 \wedge \psi_2 = y_d$.
2. The sets of stable and flexible attributes from which items in both rules are derived must be the same.[1]

Note that some eligible combinations can be skipped for optimization reasons, but still a complete set of action rules is returned (cf. [3] for details).

[1]This condition can be relaxed in several ways, cf. the ActionRulesDiscovery package documentation for details.

An action rule $r_{1\to 2}$ formed from r_1 and r_2 can be represented as:

$$r_{1\to 2} : [\alpha_1^{(1)} \to \alpha_2^{(1)} \wedge \ldots \wedge \alpha_1^{(k)} \to \alpha_2^{(k)}] \wedge [\omega_1^{(k+1)} \wedge \ldots \wedge \omega_1^{(q)}] \Rightarrow [\psi_1 \to \psi_2], \quad (5)$$

where k is the number of recommended actions. An action $\alpha_1^{(i)} \to \alpha_2^{(i)}$ denotes that the rule recommends to replace the item $\alpha_1^{(i)}$ in r_1 with $\alpha_2^{(i)}$ in r_2. In other words, actions correspond to changes in values of flexible attribute.

3. Related work

Inspiration for high-utility action rule mining can be drawn from research on high-utility itemset (pattern) mining [2], which is a generalization of frequent itemset mining allowing items in the database to be annotated with numerical values denoting their utility. The goal of high-utility mining is to discover all high-utility itemset [8]. An itemset I is a high-utility itemset if its utility $u(I)$ is no less than the minimum utility threshold *minutil* specified by the user ($u(I) \geq$ *minutil*) [2].

Using costs in action rule mining appeared in parallel with the work of Liu et al, 2005 [8] on high utility itemset mining. The cost of an action was introduced by Tzacheva and Ras, 2005 [9], who propose a cost formula to calculate the feasibility of actions. However, this research does not yet work with the notion of utility. A formula for computing utility of action rules is proposed in Tzacheva et al., 2016 [10]. This proposal does not work with an external utility table as the value of the utility is computed from the training data.

Su et al., 2012 [11] is the closest research to ours as their approach to computing utility of action rules is based on an externally set utility table defined for each flexible attribute and for the target attribute. For example, for a binary attribute f, this table separately defines the change in utility when f is changed from 0 to 1 and from 1 to 0. The utility can be either positive or negative, where a positive utility (benefit) is used for change from undesired to desired value of the target attribute and a negative utility (cost) may be used to express the cost incurred from an action on a flexible attribute.

With our approach, we are extending the pioneering work of Su et al., 2012 [11] in several ways. The formula used in Su et al. for computing the utility of an action rule is optimistic. Assuming binary target variable, their approach does not consider that the original rule (r_1 in our notation) assigns a probability denoted by $1 - conf(r_1)$ to the desired class even without performing the action proposed by the rule. We also simplify the utility table by associating the utility with values (items) rather than with a transition from one item to another. Also, the implementation for [11] is not publicly available.

4. Proposed Action Rule Utility

For mining high utility action rules according to the proposed approach, the user needs to provide the *item utility* function *ul* assigning items derived from the set of flexible attributes F and the target attribute Y with utility values. High utility action rule mining takes on the input the set of discovered action rules (cf. Phase 2 in Section 2), computes their utility and outputs only those with utility above the user-set threshold.

To compute the utility of an action rule $u(r_{1\to 2})$ we subtract the total item utility of all items in r_1 from the total item utility in r_2 taking into account the confidence of both rules:

$$u(r_{1\to 2}) = u(r_2) - u(r_1) = ((ul(\phi_2) - ul(\phi_1)) * (conf(r_2) - (1 - conf(r_1))) + \sum_{n=1}^{k} (ul(\alpha_2^{(n)}) - ul(\alpha_1^{(n)})), \tag{6}$$

where $ul(\phi_2)$ is the utility of the desired class, $ul(\phi_1)$ the utility of the undesired class, $ul(\alpha_1^{(n)})$ and $ul(\alpha_2^{(n)})$ are the utilities of items derived from flexible attributes before and after the recommended action. The expression $ul(\phi_2) - ul(\phi_1)$ computes the difference in the utility between the desired class and the undesired class. This value is multiplied by $conf(r_2) - (1 - conf(r_1))$, which is the change in confidence of the desired class according to r_2 compared to its confidence according to r_1. Note that $1 - conf(r_1)$ corresponds to the confidence of the desired class according to r_1 because we restricted the target attribute to be binary. The final part of the formula $\sum_{n=1}^{k}(ul(\alpha_2^{(n)}) - ul(\alpha_1^{(n)}))$ expresses the total change in utility resulting from the recommended actions.

5. Demonstration: Employee Attrition

To illustrate the application of high-utility action rules mining, consider an example employee database depicted in Table 1. The aim is to find actions that increase the probability that employees do not leave the company.

Table 1
Database D for Employee Attrition

TID	Department	Salary	Attrition
1	Sales	Medium	False
2	R&D	Medium	False
3	R&D	Medium	True
4	R&D	Medium	True
5	Sales	Low	False
6	R&D	High	False
7	R&D	High	False
8	R&D	High	True

The user specifies the set of predictors $A = \{'Department', 'Salary'\}$, the target attribute $Y = 'Attrition'$, the minimum support threshold to 25% and the minimum confidence to 60%. These settings are sufficient for mining of class association rules. For action rule mining, the user specifies the set of stable attributes $S = \{'Department'\}$ and the the set of flexible attributes $F = \{'Salary'\}$. The intuition is that the company probably cannot move employees between

Table 2
External Utility Table

Item	Utility
Salary: Low	-300
Salary: Medium	-500
Salary: High	-1000
Attrition: False	700
Attrition: True	0

Sales and R&D departments, while it can change their salary. Table 2 shows the user-specified utilities for the individual items derived from the flexible attribute and the target attribute. The user also sets the minimum utility of the discovered action rules to $min\_profit = -300$.

With this setting two class association rules are discovered from the data in Table 1:

$$r_1 = [(Department : R\&D \wedge Salary : Medium) \Rightarrow Attrition : True], \quad with\ support\ 25\%\ and\ confidence\ 66.6\%. \tag{7}$$

$$r_2 = [(Department : R\&D \wedge Salary : High) \Rightarrow Attrition : False], \quad with\ support\ 25\%\ and\ confidence\ 66.6\%. \tag{8}$$

Based on these two classification rules, the following action rule can be generated:

$$r = [(Department : R\&D) \wedge (Salary : Medium \rightarrow High)] \Rightarrow [Attr. : T \rightarrow F], \tag{9}$$

The discovered action rule does not contain information on whether this action is profitable on its own. Intuition may suggest that applying the rule pays off. The unit cost of the salary increase is 500 units (utility -500), while the unit benefit (revenue) is 700 units (utility +700). However, the entire benefit cannot be included as a result of the application of the action rule. The reason is that the confidence of r_1 indicates that some employees would not leave even with the lower salary, and the confidence of r_2 indicates that some would not stay even after a salary increase. The latter group needs to be subtracted from the increase in utility.

Since the target is binary, the utility of the action rule can be calculated using Eq. 6:

$$u(r_{1\rightarrow 2}) = ((700 - 0) * (0.666 - (1 - 0.666)) + (-1000 - (-500)) \approxeq -266.6 \tag{10}$$

This result suggests that the rule is not profitable. In spite of this, it is returned because $u(r_{1\rightarrow 2}) > min\_profit$.

6. High-Utility Action Rule Mining Implementation

Figure 1 depicts the workflow of the extended ActionRules package for high-utility action rule mining.

The blue part represents the original ActionRules workflow. The green part is the extension that enables the high-utility action rule generation.

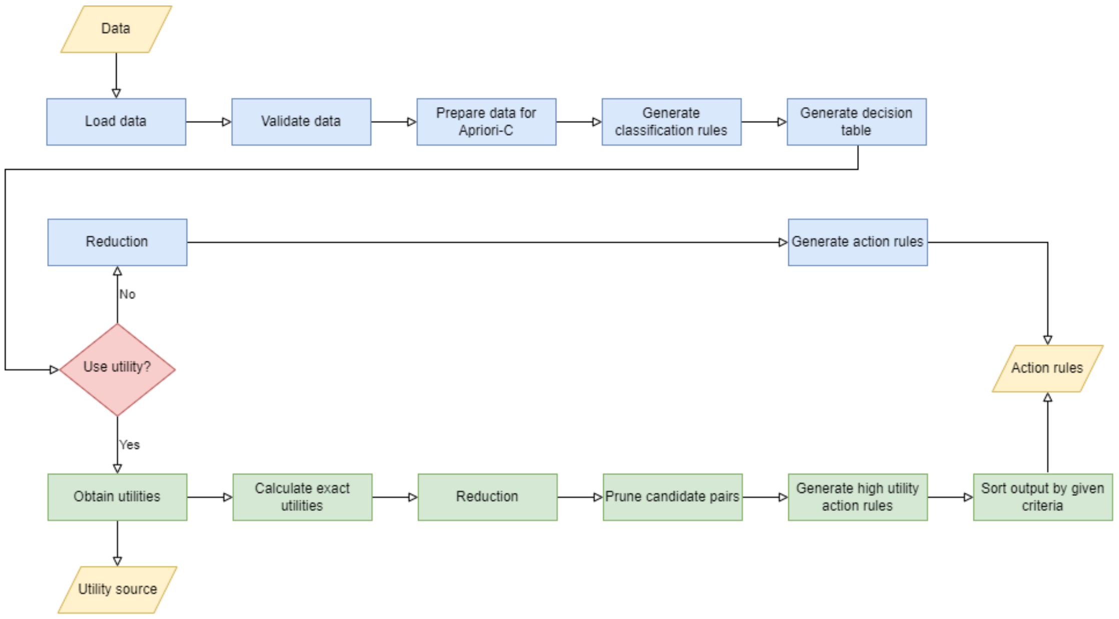

Figure 1: Flow diagram of extended ActionRules package (according to ISO 5807 notation).

The ActionRules package for action rules mining uses the PyFIM[2] library as a default option for association rule mining. This library returns an exhaustive list of class association rules used as input for action rule generation.

The ActionRules package uses similar general machine learning interfaces as Scikit-Learn [12], a popular data mining library.

6.1. Instantiate Model Object

In Code listing 1, the packages are imported , and the model is instantiated.

Code 1. *Instantiate model object*

```
import pandas as pd
from actionrules.actionRulesDiscovery import ActionRulesDiscovery

actionRDiscovery = ActionRulesDiscovery()
```

[2]http://www.borgelt.net/pyfim.html

6.2. Fit Model to Training Data

The model is fit to the training data. In our example, we will use the same data as used for illustration in Section 5. All features are nominal as required by the ActionRules package. The training data are provided in a comma separated values (csv) file shown in Listing 1.

Listing 1: attrition.csv

```
TID,Department,Salary,Attrition
1,Sales,Medium,False
2,R&D,Medium,False
3,R&D,Medium,True
4,R&D,Medium,True
5,Sales,Low,False
6,R&D,High,False
7,R&D,High,False
8,R&D,High,True
```

The utility values are loaded from a two-dimensional table exemplified in Listing 2. The first column is the index column, which must be in the form of *'attribute_attributeValue'* (for example, *'Salary_low'*). The second column contains the utility value as a float. The *'attributeValue'* part of the index must always be written in the lower case since the package code during the process of discovering action rules converts all these values to lower case. Following the approach adopted in [13], costs have a negative sign.

Listing 2: utility.csv

```
Item,Utility
Salary_low,-300
Salary_medium,-500
Salary_high,-1000
Attrition_false,700
Attrition_true,0
```

In Code listing 2, there is one stable attribute *'Department'*, there is one flexible attribute *'Salary'*, and the target attribute is set to *'Attrition'*. The minimum confidence for class association rule mining is set to 60% and the minimum support to 25%. These thresholds are used only for generating classification rules (phase 1). For phase 2, the listing defines the value of minimum utility ('min_profit=-300'). The desired class for *'Attrition'* is *'False'*.

Code 2. *Fit model to training data*

```
actionRDiscovery.read_csv("data/attrition.csv")
utility = pd.read_csv("data/utility.csv", index_col="Item")
utility. columns = [1]
actionRDiscovery.fit(stable_attributes = ["Department"],
               flexible_attributes = ["Salary"],
               consequent = "Attrition",
               conf=60,
               supp=25,
               desired_classes = ["False"],
               utility_source=utility,
               min_profit=-300)
```

6.3. Show Action Rules

Code listing 3 shows the last step of the action rule mining workflow. It returns all mined action rules with the utility.

Code 3. *Show Action Rules*

```
actionRDiscovery.get_pretty_action_rules()
```

The list of generated action rules is in Listing 3. In this case, this is just one rule.

Listing 3: Python Output

```
["If attribute 'Department' is 'R&D', attribute 'Salary' value
'medium' is changed to 'high', then 'Attrition' value 'True'
is changed to 'False'. Profit
of the action is -266.66666666666674."]
```

7. Conclusion

In this paper, we described a new extension of the Python ActionRules package for high-utility action rules mining. A new formula for high-utility action rules mining was introduced. The ActionRules package along with the described extension can be be found at: https://github.com/lukassykora/actionrules.

High-utility action rules require additional user information in the form of a utility table, which may not always be available or complete. Future work might explore approaches that would ameliorate this requirement. One possibility is to draw inspiration from the Learning Classifier Systems family of rule learning [14], where more complex internal utility definitions were developed and used as fitness functions to evaluate candidate rules.

Acknowledgments

Supported by Prague University of Economics and Business by grant IGA 40/2021.

References

[1] Z. W. Ras, A. Wieczorkowska, Action-rules: How to increase profit of a company, in: European Conference on Principles of Data Mining and Knowledge Discovery, Springer, 2000, pp. 587–592.

[2] P. Fournier-Viger, J. C.-W. Lin, R. Nkambou, B. Vo, V. S. Tseng, High-utility pattern mining, Cham: Springer (2019).

[3] L. Sỳkora, T. Kliegr, Action rules: Counterfactual explanations in python, in: RuleML Challenge, 2020.

[4] A. Dardzinska, Action rules mining, volume 468, Springer, 2012.

[5] Z. W. Raś, L.-S. Tsay, A. Dardzińska, Tree-based algorithms for action rules discovery, in: Mining Complex Data, Springer, 2009, pp. 153–163.

[6] R. Agrawal, R. Srikant, et al., Fast algorithms for mining association rules, in: Proc. 20th int. conf. very large data bases, VLDB, volume 1215, 1994, pp. 487–499.

[7] V. Jovanoski, N. Lavrač, Classification rule learning with apriori-c, in: Portuguese Conference on Artificial Intelligence, Springer, 2001, pp. 44–51.

[8] Y. Liu, W.-k. Liao, A. Choudhary, A fast high utility itemsets mining algorithm, in: Proceedings of the 1st international workshop on utility-based data mining, 2005, pp. 90–99.

[9] A. A. Tzacheva, Z. W. Raś, Action rules mining, International Journal of Intelligent Systems 20 (2005) 719–736.

[10] A. A. Tzacheva, C. C. Sankar, S. Ramachandran, R. A. Shankar, Support confidence and utility of action rules triggered by meta-actions, in: 2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA), IEEE, 2016, pp. 113–120.

[11] P. Su, D. Li, K. Su, An expected utility-based approach for mining action rules, in: Proceedings of the ACM SIGKDD Workshop on Intelligence and Security Informatics, 2012, pp. 1–4.

[12] An introduction to machine learning with scikit-learn, scikit-learn (2022). URL: https://scikit-learn.org/stable/tutorial/basic/tutorial.html.

[13] K. Singh, H. K. Shakya, A. Singh, B. Biswas, Mining of high-utility itemsets with negative utility, Expert Systems 35 (2018) e12296.

[14] R. J. Urbanowicz, W. N. Browne, Introduction to learning classifier systems, Springer, 2017.