Covering Approach to Action Rule Learning

Paweł Matyszok, Marek Sikora, and Łukasz Wróbel(✉)

Institute of Informatics, Silesian University of Technology,
ul. Akademicka 16, 44-100 Gliwice, Poland
{pawel.matyszok,marek.sikora,lukasz.wrobel}@polsl.pl

Abstract. Action rules specify recommendations which should be followed in order to transfer objects to the desired decision class. This paper presents a proposal of a novel method for induction of action rules directly from a dataset. The proposed algorithm follows the so-called covering schema and employs a pruning procedure, thus being able to produce comprehensible rule sets. An experimental study shows that the proposed method is able to discover strong actions of superior accuracy.

1 Introduction

Action rules are one of the ways to use rule-based representations to search for recommendations which indicate how the change of attribute values can cause the change in the assignment of examples to the given decision class. There have been several proposals of algorithms for action rule induction. Neither of the proposed algorithms employs one of the most popular and efficient approaches to the rule induction, which is the covering (known also as separate-and-conquer) strategy [4,13]. The paper features a proposal of an algorithm for action rule induction by means of the covering strategy. The algorithm generates action rules which are enough to cover an analysed set of examples, thus significantly limiting the number of generated rules.

The rule growing and pruning phases are guided by rule quality measures [6,12]. The possibility to use different quality measures allows generating more accurate or general rules, depending on the users' needs [17,23]. A unique feature of the algorithm is the possibility to generate action rules on the basis of numerical attributes with no necessity of their previous discretization.

The paper is organized as follows: Sect. 2 features related work on the action rule induction, Sect. 3 describes the proposed algorithm. Section 4 presents the results of the algorithm on datasets from the UCI repository. Section 5 gives the conclusions and presents the possible future work.

2 Related Work

The first works in the field of action rule induction focused on generating action rules on the basis of the existing classification rules generated by means of algorithms based on the rough set theory [8,14,16]. The next proposed methods

S. Kozielski et al. (Eds.): BDAS 2018, CCIS 928, pp. 182–193, 2018.
https://doi.org/10.1007/978-3-319-99987-6_14

included also algorithms for direct induction of action rules from data, like association action rule induction and the ARED algorithm [3,11,15]. The ARED method uses the concept of Pawlak's information system [14] in which certain relationships between granules, defined by indiscernibility relation of objects, are identified. Some of these relationships are used to define the action rule. All aforementioned approaches are based on the assumption that all possible rules are generated which meet the minimum support and confidence constraints. In [9] the authors proposed an algorithm for induction of action rules from temporal decision tables. In this approach the rules are generated on the basis of a dataset describing the behaviour/states of the object over the given period of time (e.g. two-year observation of a patient), thus containing all available information about the objects during the examined period (e.g. follow-up appointments, hospital stays).

Action rules reflect recommendations on the changes of attribute values but they do not indicate which operations cause the changes (e.g. recommendation "change blood sugar level from 95 to 80" does not show what kind of medicine should be taken to apply this recommendation). In this case the usability of action rules is understood as the analysis and identification of operations that must be undertaken to change the values of the attributes occurring in the premises of action rules. Such operations are called meta-actions. The analysis of dependencies between action rules and meta-actions was presented in [20,22]. Recently, Almardini et al. [1] have presented procedure paths as a sequence of procedures that a given patient undertakes to reach the desired treatment. In the majority of the quoted papers, the authors illustrate their algorithms on medical data (Florida State Inpatient Databases, HEPAR Dataset) and the described experiments contain mainly the analysis of a few rules from the generated ones. However, there is no research on the comparisons between the proposed algorithms in terms of criteria such as the number of rules, average number of conditions in inducted rules, average quality of rules, etc.

The action rule induction is part of a more general issue of the usability and actionability of the data exploration results. This issue has been raised more and more often [10]. In the case of rule-based representations, the issue of actions generation on the basis of classification rules, with no necessity to generate action rules, was proposed in [21]. In [7,18,24] the authors discussed the methods of rule assessment from the point of view of their usability and actionability. In [24] actionability is defined as the compatibility of a classification rule structure with specific requirements of the user, concerning the structure of conditions contained in the rule premise. The remaining papers [7,18] concentrate on the assessment of the strength of classification rules from the point of view of their potential to change the examples assignment between decision classes.

3 Covering Approach to Action Rule Induction

Basic Notions

An action rule may be considered a composition of two decision rules. To define the action rule, basic notions are presented in this section.

Let us consider that a finite set of examples T_r is given. Each example in this set can be described with a set of attributes $A \cup \{d\}$, where $a : T_r \to V_a$ for each $a \in A \cup \{d\}$. The set V_a is called *range* of attribute a. The elements of A are called conditional attributes, and the variable d is known as a decision attribute – its value is considered as an assignment of an example to a decision class. Conditional attributes can be numeric or symbolic type. Numeric attributes values are real numbers, while symbolic attributes values have to be discrete. The decision attribute is always symbolic. The conditional expression of the following form is called decision rule:

$$w_1 \ \wedge \ w_2 \ \wedge \ldots \wedge \ w_k \text{ THEN } d = v$$

The part to the left of the word *THEN* is called premise and the part to the right side of this word is called conclusion. The conclusion of a decision rule is understood as an assignment of an example fulfilling the premise of this rule to the concept bound with the value of the decision attribute d. The premise of the decision rule is a conjunction of elementary conditions w_i. The form of the elementary condition used in the presented approach is $w_i \equiv a_i \in Z_i$, where a_i is the name of the conditional attribute and Z_i is the interval in this attribute range V_{a_i}. For symbolic attributes the elementary condition is simple $a_i = v_j$, where $v_j \in V_{a_i}$.

Let us consider two decision rules r_1 and r_2, such conclusions of those rules are mutually exclusive (i.e. $v_1 \neq v_2$):

$$\textbf{r1: } w_{1_1} \wedge \ w_{1_2} \wedge \ldots \wedge \ w_{1_k} \text{ THEN } d = v_1$$
$$\textbf{r2: } w_{2_1} \wedge \ w_{2_2} \wedge \ldots \wedge \ w_{2_k} \text{ THEN } d = v_2$$

The following composition of such rules is called *action rule*:

$$w_{1_1} \to w_{2_1} \wedge \ w_{1_2} \to \ w_{2_2} \wedge \ldots \wedge \ w_{1_k} \to w_{2_k} \text{ THEN } d = v_1 \to v_2$$

This representation, with no loss of generality, may be rewritten in a simpler form:

$$(a_1, v_{a_{1_1}} \to v_{a_{1_2}}) \ \wedge \ (a_2, v_{a_{2_1}} \to v_{a_{2_2}}) \ \wedge \ldots \wedge \ (a_k, v_{a_{k_1}} \to v_{a_{k_2}})$$
$$\text{THEN } (d = v_1 \to v_2)$$

An action rule describes possible transition of examples between classes in the given example set T_r based on the change in the attribute values of examples in this set. The class on the left side of the action rule conclusion (v_1) will be called *source class* or *source concept*, while the one on the right side (v_2) – *target class* or *target concept*.

The elementary condition of the form $(a_i, v_{a_{i_1}} \to v_{a_{i_2}})$ is called *action* and should be interpreted as a request to change the value of the attribute a_i of examples fulfilling the condition $a_i \in v_{a_{i_1}}$, so that those examples could meet the condition $a_i \in v_{a_{i_2}}$ after the request (or action) is completed. The premise of the action rule is a conjunction of actions. The conclusion of the action rule is yet another action, which is interpreted as a change in the assignment of

examples fulfilling the left side of each action from the original concept v_1 to a new concept v_2 after all actions in the premise are performed (meaning that the examples fulfill now the right side of each action in the premise).

In the context of the action rules attributes can be treated as flexible or stable. A flexible (or mutable) attribute is a kind of attribute whose value can be changed, e.g. interest rate of a loan. Stable attributes are features of an example than cannot be changed, e.g. date of birth. An action rule induction algorithm must not create actions based on stable attributes.

Action Rule Induction

In this section a covering action rule induction algorithm is presented. The algorithm utilizes a well known separate-and-conquer framework [4] and classification rule quality measures [23] to induce action rules. The algorithm of rule induction is divided into two phases: growing and pruning (Algorithm 1). The action rule is added to the resulting rule set, and all source class examples covered by left side of newly created action rules are removed from the training set (line 11). The examples of the target class are never removed from the training set, because we want to preserve as much information about the target class in the training set as possible. Source class examples covered by the induced rule are, however, removed from the training set to follow the sequential covering approach. Rules are induced, until all examples of the source class in the training set T_r are covered by left sides of the induced action rules.

Growth of Action Rule (Algorithm 2). First, the conclusion of an action rule is stated and then actions are added to the action rule. The conclusion of a newly created action rule contains two classes: the one on the left side of the concluding action (source class), and the one on the right side of this action (target class). When the action rule is growing, two classification rules are built concurrently: one with a source class in conclusion, and one with a target class. First, the feature space of yet uncovered examples is searched for the best elementary condition which separates the examples of the source class (line 6). The proposed elementary conditions are built differently for categorical and numerical attributes. When a categorical attribute is processed, an elementary condition is induced for each distinct value of this attribute spotted in the training set. When numerical attributes are processed, an elementary condition is created for each cut-point between neighbouring values of that attribute in the training set. The best condition is selected using the rule quality measure, which is one of algorithm parameters, in the following way: the built classification rule is temporarily extended with the proposed elementary condition, and the quality of the resulting rule is assessed according to the selected measure. The elementary condition, which yields the best quality in this procedure, is selected to be finally added to the rule for the source class. The attribute used in the elementary condition added to this rule is recorded, and the feature space search is conducted for the best elementary condition separating target class examples, but only with regard to the recorded attribute (line 8). The search is performed in the same

Algorithm 1. Induction of action rule set

Input: $D(A, C)$—training set (set of attributes A and decision class C), q—classification rule quality measure, *mincov*—minimal coverage, s—source class, t—target class

Output: R—action rules set

```
1: function InduceActionRules(D, q, mincov, s, t)
2:     R ← ∅
3:     S ← GetExamples(s)
4:     D_t ← D \ S
5:     while D \ D_t ≠ ∅ do
6:         D_s ← S ∩ D
7:         r ← SpecializeActionRule(D, D_s, s, q, t, mincov)
8:         r ← PruneActionRule(r, D, q)
9:         R ← R ∪ {r}
10:        D_s ← Coverage(r, D)
11:        D ← D \ D_s
12:    end while
13:    return R
14: end function
```

manner as for elementary conditions for the source class, except that only the values of attributes belonging to the examples of the target class are used as the feature space. The conjunction of elementary conditions generated for the source and target class is added as an action to the premise of the built action rule.

The intention of this procedure is to find actions which will have high impact on transferring source class examples to the target class. The above described algorithm is conducted iteratively, until no more actions can be generated.

The action generation procedure can be stopped when the rule extended with the elementary condition generated for the source class does not cover the minimal number of examples in the training set T_r – this value is another parameter of the algorithm called *mincov*. After the rule is fully grown, the procedure of pruning begins.

Pruning of Action Rule (Algorithm 3). The action rule pruning process is performed according to the hill climbing strategy. In short, the actions in the premise of the action rule are being iteratively trimmed or removed, as long as the quality of the rule does not decrease. In this process the action rule is represented as two classification rules: left, responsible for source class examples and right, which selects target class examples.

The actions (elementary conditions) in the action rule premise are temporarily pruned. To prune the actions means to get rid of the right side of the action: $(a_i, v_{a_{i1}} \rightarrow v_{a_{i2}})$ becomes $(a_i, v_{a_{i1}})$. This means that the selected elementary condition is removed from the right rule only. If the quality of this rule was not decreased when compared with original rule quality, the temporarily pruned action would be marked as candidate to final pruning (line 14). Now the respective elementary condition from the left rule is also temporarily removed from

Algorithm 2. Specialization of action rule

Input: $D(A, C)$—training set: set of attributes A and decision class C, D_s—set of yet uncovered source class examples, D_t—set of all target class examples, q—classification rules quality measure function, *minvoc*—minimal coverage, s—source class, t—target class

Output: r—action rule

1: **function** SpecializeActionRule($D, D_s, q, s, t, mincov$)
2: $r \leftarrow (\emptyset \rightarrow (s \rightarrow t))$
3: ▷ Empty classification rules targeting source and target class
4: $r_s \leftarrow (\emptyset \rightarrow C = s)$, $r_t \leftarrow (\emptyset \rightarrow C = t)$
5: **repeat**
6: $w_{best_s} \leftarrow$ GetBestElementaryCondition($r_s, D, q, mincov$)
7: $atr \leftarrow$GetAttribute(w_{best_s})
8: w_{best_t} $\leftarrow$GetBestElementaryConditionForAttribute($atr, r_t, D_t, q, mincov$)
9: $r_s \leftarrow r_s \wedge w_{best_s}$, $r_t \leftarrow r_t \wedge w_{best_t}$
10: ▷ Extend action rule with action built from best conditions
11: $r \leftarrow r \wedge$MakeAction(w_{best_s}, w_{best_t})
12: **until** ($w_{best_s} = \emptyset$)
13: **return** r
14: **end function**

that rule, and the quality is recorded. When this value is not less the quality recorded on not modified left rule, the whole action is marked as candidate to removal. This process is performed on each action in the premise. The best candidate to prune or remove is selected, and the rule is modified according to this selection (lines 23 through 29). The whole process is repeated until no more candidates to prune or remove can be distinguished.

The rule pruned using the above described procedure is added to the overall rule set. Examples which are covered by the left rule are removed from the training set T_r. If there are more examples of the source class in the training set, the process of growing and pruning a new rule starts again.

4 Experiments

Experiments of the proposed method were conducted on several widely known datasets from the UCI repository. The program was parameterized in such a way that the minimum coverage of a rule extended by the proposed elementary condition during the rule specialization is equal to 5. During specialization and pruning of the rules the *Correlation* quality measure was used. The *Correlation* measure is defined as $(pN - Pn)/\sqrt{PN(p+n)(P-p+N-n)}$ and it evaluates the correlation coefficient between the predicted and the target labels. As empirical [5,17] studies show, it maintains good balance between accuracy and comprehensibility of rules generated in covering schemas. Let us recall the meaning of the variables used in the *Correlation* measure formula, with regard to the given classification rule r:

Algorithm 3. Pruning of action rule

Input: $D(A, C)$—training set (set of attributes A and decision class C), r—action rule to prune, q—classification rule quality measure
Output: r'—pruned action rule

function PruneActionRule(D, r, q)
  $r' \leftarrow r$
  ▷ Disassemble action rule to classification rules
  $r_s \leftarrow$GetLeftRule(r') , $r_t \leftarrow$GetRightRule(r')
  repeat
    $w_{removal} \leftarrow w_{prune} \leftarrow a \leftarrow \emptyset$
    $q_{removal} \leftarrow q(r_s, D)$, $q_{prune} \leftarrow q(r_t, D)$
    for $w \in r'$ **do**
      $w_p \leftarrow$GetRightSideOfAction(w)
      $w_r \leftarrow$GetLeftSideOfAction(w)
      ▷ Try to prune action
      $q_r \leftarrow q(r_t \setminus w_p, D)$
      if $q_r \geq q_{prune}$ **then**
        $w_{prune} \leftarrow w_p$, $q_{prune} \leftarrow q_r$
        $a \leftarrow w$
      end if
      ▷ Try to get rid of whole action
      $q_r \leftarrow q(r_s \setminus w_r, D)$
      if $q_r \geq q_{removal}$ **then**
        $w_{removal} \leftarrow w_r$, $q_{removal} \leftarrow q_r$
      end if
    end for
    if $w_{prune} \neq \emptyset$ **then**
      if $w_{removal} \neq \emptyset$ **then**
        $r' \leftarrow r' \setminus a$
      else
        $r' \leftarrow (r' \setminus a) \wedge$ PruneAction(a)
      end if
    end if
  until $((w_{removal} = \emptyset \wedge w_{prune} = \emptyset) \vee (|r'| = 1))$
  return r'
end function

- P stands for the number of examples in the dataset of the class pointed by the conclusion of the rule r,
- N is substituted by the number of examples that are not covered by the conclusion of r,
- p is the number of true positives, that is, the number of examples covered by both the premise and the conclusion of the rule r,
- n is the number of false positives, that is, the number of examples covered by the premise, but not covered by the conclusion of the rule r.

It is worth noticing that the quality measure is used to build or prune temporal classification rules during action rule induction, as described in Sect. 3.

Sequential covering of the training dataset is stopped when only 0.5% of the initial number of examples are uncovered. The rule growing phase is stopped when no new conditions can be induced (i.e. not meeting the minimum coverage criterion), or when the count of elementary conditions in the rule exceeds 0.9 × *number of attributes in dataset*.

The statistics of action rules generated on the examined datasets are shown in Table 1. The table presents the summary of induced rule sets before and after the pruning – the total number of rules, the average number of conditions in a rule and the average number of actions per rule. As it can be observed, the pruning procedure is able to significantly reduce the number of rules as well as the number of conditions occurring in rules. Additionally, the number of actions is reduced in each rule set obtained on tested datasets. This means that the pruning process allows better generalization of suggested actions – fewer actions will be needed to achieve the same effect regarding the class change. For comparison purposes, the results obtained with the custom ARED algorithm implementation are also presented. To make possible the generation of rules for datasets which contain numerical attributes (i.e. *Diabetes-c, Labor, Wine and Prnn-synth*), the discretization of those dataset was conducted using the entropy criterion. For all datasets *minimum_support* was set to the value 5 and the $minimum\_confidence$ parameter was equal to 0.9. For both ARED and the proposed method all attributes of all tested datasets were marked as flexible.

Table 1. The characteristics of unpruned and pruned action rule sets: the number of rules (#rules), the average number of elementary conditions per rule ($\overline{\text{conds}}$), the average number of actions per rule ($\overline{\text{actions}}$).

Dataset	Unpruned rule set			Pruned rule set			ARED rule set		
	#rules	$\overline{\text{conds}}$	$\overline{\text{actions}}$	#rules	$\overline{\text{conds}}$	$\overline{\text{actions}}$	#rules	$\overline{\text{conds}}$	$\overline{\text{actions}}$
Car	85	3.95	3.1	25	2.9	1.3	13930	3.79	3.79
Diabetes-c	7	3.71	3.71	7	3.57	2.0	22	1.04	1.04
Labor	2	3.5	3.5	2	3.0	2.0	15	1.0	1.0
Monk1	11	2.5	2.1	7	1.7	1.1	33	1.32	1.32
Prnn-synth	4	1.5	1.5	3	1.33	1.0	7	1.0	1.0
Titanic	6	3.0	1.83	2	2.0	0.5	8	1.0	1.0
Wine	1	12.0	12.0	1	2.0	1.0	5	1.0	1.0

An Illustrative Example. For three datasets from Table 1 a brief study of the rules induced was conducted too.

– *Car* – this dataset contains information about different car models. All 6 attributes are nominal: number of doors, purchase cost, maintenance cost, number of passangers, luggage boot size, and safety. A class attribute represents the concept of attractiveness of a given car for the buyer. For demonstration purpose this dataset has been reduced to contain only examples representing the classes *unacc* and *acc*, which are represented by

(respectively) 1210 and 384 examples. In the original set there are also representatives of the classes *good* and *v-good*.
- *Monk1 (First Monk's Problem)* – synthetic dataset with 6 nominal attributes $attr1 \ldots attr6$. Examples can belong to one of two classes. The examples belonging to the positive class are characterized by having the property $attr1 = attr2 \vee attr5 = 1$. During experiments a training set was used, which contains 62 examples of each class.
- *Wine* – consists of 13 numerical attributes which characterize different cultivars of red wine grown in a region of Italy. Among the attributes, one can find the color intensity or alcohol content. This dataset has also been reduced to contain only two classes, represented by 59 (class 1) and 71 (class 2) examples.

Two of the rules generated on the *Car* dataset show very clearly that knowledge they represent is close to intuition:

car-r1: (*persons*, 4) ∧ (*safety*, low→high) ∧ (*lug_boot*, big)
THEN (*class*, unacc→acc) $[p_l = 64, n_l = 0, p_r = 108, n_r = 36]$
car-r8: (*persons*, 2→4) ∧ (*safety*, high→med) ∧ (*lug_boot*, big)
THEN (*class*, unacc→acc) $[p_l = 64, n_l = 0, p_r = 40, n_r = 12]$

Apart from each rule, the number of true positives (p) and false positives (n) covered by the left (p_l, n_l) and the right-side (p_r, n_r) of the action rule are given.

The rule *car-r1* suggests that when designing a city car for four passengers with a big trunk, attention should be kept on security features of that car, as this influences the car perception by a potential buyer who may accept that the car is not spacious (only four passengers) but wants it be safe. On the other hand, the rule *car-r8* shows that the car may also be acceptable with a lower degree of safety, as long as it maintains a large luggage boot and can contain at least four passengers. Both rules show intuitive knowledge, as people tend to take care about safety when driving a car, and cars for two passengers are considered unpractical.

To present the ability to generate more general rules, a rule induced on the *Wine* dataset, before and after pruning, is shown.

wine-r1: (*Nonflavanoid_phenols*, $< 0.085, \infty) \rightarrow < 0.145, \infty)$) ∧
($Color\_intensity$, $(3.375, \infty) \rightarrow < 1.7, \infty)$) ∧
($Proanthocyanins$, $< 1.095, \infty) \rightarrow (0.52, \infty)$) ∧
(Ash, $< 1.36, \infty) \rightarrow < 1.095, \infty)$) ∧
($Total\_phenols$, $< 1.65, \infty) \rightarrow < 1.21, \infty)$) ∧
($Proline$, $< 755.0, \infty) \rightarrow (-\infty, 492.5)$) ∧
($Alcohol$, $< 6.935, \infty) \rightarrow < 5.78, \infty)$) ∧
($Magnesium$, $< 47.0, \infty) \rightarrow < 40.5, \infty)$) ∧
($Flavanoids$, $< 1.965, \infty) \rightarrow < 1.125, \infty)$)
THEN (*class*, 1→2) $[p_l = 57, n_l = 0, p_r = 35, n_r = 0]$
wine-r1-pruned: ($Color\_intensity$, $(3.375, \infty)$) ∧
($Proline$, $< 755.0, \infty) \rightarrow (-\infty, 492.5)$)
THEN (*class*, 1→2) $[p_l = 57, n_l = 0, p_r = 35, n_r = 0]$

The pruned rule does not lower the quality, but it is much shorter and more comprehensible for the user of the method. Due to the pruning procedure, many actions which do not have much influence on the change of examples assignment to a class are removed.

The proposed method performs well in the task of discovering hidden rules about data. This will be shown by means of a rule induced on the *Monk1* set. Let us recall that in this dataset the rule for assignment to class 1 is: $attr1 = attr2 \vee attr5 = 1$. The described method is unable to discover the exact condition $attr1 = attr2$, however it is able to generate rules following this rule, for example:

monk-r2: (*attr2*, 2→3) ∧ (*attr1*, 1→3)
THEN (*class*, 0→1) $[p_l = 15, n_l = 2, p_r = 17, n_r = 0]$
monk-r3: (*attr2*, 3) ∧ (*attr1*, 1→3)
THEN (*class*, 0→1) $[p_l = 16, n_l = 3, p_r = 17, n_r = 0]$

These two rules suggest actions that will have the effect of examples meeting the condition $attr1 = attr2$.

The proposed algorithm is also able to discover the other rule of the *Monk1* problem, which is $attr5 = 1$:

monk-r1: r1: (*attr5*, 4→1) ∧ (*attr1*, 1)
THEN (*class*, 0→1) $[p_l = 13, n_l = 1, p_r = 29, n_r = 0]$
monk-r4: (*attr5*, 4→1)
THEN (*class*, 0→1) $[p_l = 23, n_l = 11, p_r = 29, n_r = 0]$
monk-r7: (*attr5*, 3→1)
THEN (*class*, 0→1) $[p_l = 19, n_l = 11, p_r = 29, n_r = 0]$

Other rules induced from this dataset have the following forms:

monk-r5: (*attr6*, 2) ∧ (*attr5*, 2→1) ∧ (*attr2*, 1)
THEN (*class*, 0→1) $[p_l = 8.0, n_l = 1.0, p_r = 29.0, n_r = 0.0]$
monk-r6: (*attr4*, 2→1)
THEN (*class*, 0→1) $[p_l = 24.0, n_l = 15.0, p_r = 26.0, n_r = 16.0]$

Rule *monk-r5* recognizes the $attr5 = 1$ principle, while putting more constraints on examples to which it is applicable. Rule *monk-r6*, which is following neither of the *Monk1* rules, is an effect of using Correlation during rule induction – this measure prefers rules of greater coverage, though lower accuracy. Choosing a different measure, for example Precision, may result in producing a greater number of more accurate rules. In general, the algorithm provides recommendation compliant with (an unknown) class definition – even rules *monk-r5* and *monk-r6* are able to transfer most of the covered examples to the target class.

5 Conclusions

In this paper an idea of a new algorithm for the induction of action rules directly from data was presented. An important feature of the method is the ability

to generate rules from datasets consisting of both numerical and categorical attributes. The presented approach was tested on several datasets from the UCI repository. As an experimental study shows, the rules generated with our method on those datasets are more compact than rules generated by the ARED method.

It was also shown that rules induced by the proposed method are consistent with intuition. Particularly on the *Monk1* set the method was able to discover hidden rules of the assignment of examples to the positive class.

Future work will focus on the development of measures and methods for assessing the quality of inducted action rules and sets. It is planned to adapt the measures dedicated to the quality evaluation of classification rules. Our work will focus on measures which assess the coverage and precision of a rule at the same time. In the case of large datasets or the necessity to reduce the number of rules, the methods of example selection, as well as rules filtering will be used [2,19]. Employing the one-vs-all binarization schema will allow to use the method for multi-class problems. A very important next step is the preparation of datasets where successive decision tables will reflect successive moments of time (control points). The tables will contain information about the application of certain actions to specific examples. A part of the datasets will be generated synthetically (e.g. the inverted pendulum problem), another part will describe a real-life problem (PersonALL project – see acknowledgment).

Acknowledgement. This work was partially supported by Polish National Centre for Research and Development (NCBiR) within the programme Prevention and Treatment of Civilization Diseases – STRATEGMED III, grant number STRATEGMED3/304586/5/NCBR/2017 (PersonALL).

A part of the work was carried out within the statutory research project of the Institute of Informatics, BK-213/RAU2/2018.

References

1. Almardini, M., et al.: Reduction of readmissions to hospitals based on actionable knowledge discovery and personalization. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kostrzewa, D. (eds.) BDAS 2015-2016. CCIS, vol. 613, pp. 39–55. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34099-9_3
2. Blachnik, M.: Instance selection for classifier performance estimation in meta learning. Entropy **19**(11), 583 (2017). https://doi.org/10.3390/e19110583
3. Dardzinska, A.: Action Rules Mining, vol. 468. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35650-6
4. Fürnkranz, J.: Separate-and-conquer rule learning. Artif. Intell. Rev. **13**(1), 3–54 (1999)
5. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-75197-7. http://www.springer.com/978-3-540-75196-0
6. Geng, L., Hamilton, H.J.: Interestingness measures for data mining: a survey. ACM Comput. Surv. **38**(3), 9 (2006)
7. Greco, S., Matarazzo, B., Pappalardo, N., Słowinski, R.: Measuring expected effects of interventions based on decision rules. J. Exp. Theor. Artif. Intell. **17**(1–2), 103–118 (2005)

8. Grzymala-Busse, J.W., Ziarko, W.: Data mining based on rough sets. Data Min.: Oppor. Chall. **2**, 142–173 (2003)
9. Hajja, A., Raś, Z.W., Wieczorkowska, A.A.: Hierarchical object-driven action rules. J. Intell. Inf. Syst. **42**(2), 207–232 (2014)
10. He, Z., Xu, X., Deng, S.: Data mining for actionable knowledge: a survey. arXiv preprint cs/0501079 (2005)
11. Im, S., Raś, Z.W.: Action rule extraction from a decision table: ARED. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 160–168. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68123-6_18
12. Janssen, F., Fürnkranz, J.: On the quest for optimal rule learning heuristics. Mach. Learn. **78**(3), 343–379 (2010). https://doi.org/10.1007/s10994-009-5162-2
13. Kaufman, K.A., Michalski, R.S.: Learning in an inconsistent world: rule selection in STAR/AQ18. Technical report, Machine Learning and Inference Laboratory (1999)
14. Pawlak, Z.: Information systems theoretical foundations. Inf. syst. **6**(3), 205–218 (1981)
15. Raś, Z.W., Dardzinska, A., Tsay, L.S., Wasyluk, H.: Association action rules. In: IEEE International Conference on 2008 Data Mining Workshops, ICDMW 2008, pp. 283–290. IEEE (2008)
16. Raś, Z.W., Tzacheva, A.A., Tsay, L.S., Giirdal, O.: Mining for interesting action rules. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 187–193. IEEE (2005)
17. Sikora, M., Wróbel, Ł.: Data-driven adaptive selection of rule quality measures for improving rule induction and filtration algorithms. Int. J. Gen. Syst. **42**(6), 594–613 (2013). https://doi.org/10.1080/03081079.2013.798901
18. Słowiński, R., Greco, S.: Measuring attractiveness of rules from the viewpoint of knowledge representation, prediction and efficiency of intervention. In: Szczepaniak, P.S., Kacprzyk, J., Niewiadomski, A. (eds.) AWIC 2005. LNCS (LNAI), vol. 3528, pp. 11–22. Springer, Heidelberg (2005). https://doi.org/10.1007/11495772_3
19. Stańczyk, U., Zielosko, B.: On combining discretisation parameters and attribute ranking for selection of decision rules. In: Polkowski, L. (ed.) IJCRS 2017. LNCS (LNAI), vol. 10313, pp. 329–349. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60837-2_28
20. Touati, H., Raś, Z.W., Studnicki, J., Wieczorkowska, A.A.: Mining surgical meta-actions effects with variable diagnoses' number. In: Andreasen, T., Christiansen, H., Cubero, J.-C., Raś, Z.W. (eds.) ISMIS 2014. LNCS (LNAI), vol. 8502, pp. 254–263. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08326-1_26
21. Trépos, R., Salleb-Aouissi, A., Cordier, M.O., Masson, V., Gascuel-Odoux, C.: Building actions from classification rules. Knowl. Inf. Syst. **34**(2), 267–298 (2013)
22. Wang, K., Jiang, Y., Tuzhilin, A.: Mining actionable patterns by role models. In: Proceedings of the 22nd International Conference on 2006 Data Engineering, ICDE 2006, p. 16. IEEE (2006)
23. Wróbel, Ł., Sikora, M., Michalak, M.: Rule quality measures settings in classification, regression and survival rule induction-an empirical approach. Fundam. Inform. **149**(4), 419–449 (2016)
24. Zhu, H.M., Huang, W.D., Zheng, H.S.: Method for discovering actionable rule. In: Fourth International Conference on 2007 Fuzzy Systems and Knowledge Discovery, FSKD 2007, vol. 1, pp. 397–401. IEEE (2007)