EXTENDED ACTION RULE DISCOVERY BASED ON SINGLE CLASSIFICATION RULES AND REDUCTS

Zbigniew W. Ras
University of North Carolina
Department of Computer Science
Charlotte, N.C. 28223, USA
Phone: 704-687-8574; Fax: 704-687-3516
ras@uncc.edu

and

Elzbieta M. Wyrzykowska
Department of Computer Science
University of Information Technology and Management
ul. Newelska 6, 01-447 Warsaw, Poland
Phone: 022-837-0396
ewyrzyko@wit.edu.pl

EXTENDED ACTION RULE DISCOVERY BASED ON SINGLE CLASSIFICATION RULES AND REDUCTS

Abstract:
Action rules can be seen as logical terms describing knowledge about possible actions associated with objects which is hidden in a decision system. Classical strategy for discovering them from a database requires prior extraction of classification rules which next are evaluated pair by pair with a goal to build a strategy of action based on condition features in order to get a desired effect on a decision feature. An actionable strategy is represented as a term $r = [(\omega)\wedge(\alpha\rightarrow\beta)]\Rightarrow[\phi\rightarrow\psi]$*, where* ω*,* α*,* β*,* ϕ*, and* ψ *are descriptions of events. The term r states that when the fixed condition* ω *is satisfied and the changeable behavior* $(\alpha\rightarrow\beta)$ *occurs in objects represented as tuples from a database so does the expectation* $(\phi\rightarrow\psi)$*. With each object a number of actionable strategies can be associated and each one of them may lead to different expectations and the same to different reclassifications of objects. This chapter will focus on a new strategy of construction of action rules directly from single classification rules instead of pairs of classification rules. This way we do not only gain on the simplicity of the method of action rules construction but also on its time complexity. The paper will present a modified tree-based strategy for constructing action rules followed by a new simplified strategy of constructing them. Finally, these two strategies will be compared.*

Keywords – data, database, knowledge base, action research, information richness, relevance of information, value of information, meta IS, features, information presentation, action-oriented framework, activity description charts, decision tables, decision trees, information analysis techniques, complexity, cost, information attributes, information evaluation, knowledge discovery, knowledge management, decision support systems, knowledge-based systems.

Introduction

There are two aspects of interestingness of rules that have been studied in data mining literature, objective and subjective measures (Liu, 1997), (Adomavicius & Tuzhilin, 1997), (Silberschatz & Tuzhilin, 1995, 1996). Objective measures are data-driven and domain-independent. Generally, they evaluate the rules based on their quality and similarity between them. Subjective measures, including unexpectedness, novelty and actionability, are user-driven and domain-dependent.

The notion of an action rule, constructed from certain pairs of association rules, has been proposed in (Ras & Wieczorkowska, 2000). Its different definition was given earlier in (Geffner & Wainer, 1998). Also, interventions introduced in (Greco, 2006) are conceptually very similar to action rules. Action rules have been investigated further in (Tsay & Ras, 2005, 2006), (Tzacheva & Ras, 2005), (Ras & Dardzinska, 2006). To give an example justifying the need of action rules, let us assume that a number of customers have closed their accounts at one of the banks. We construct, possibly the simplest, description of that group of people and next search for a new description, similar to the one we have, with a goal to identify a new group of customers from which no-one left that bank. If these descriptions have a form of rules, then they can be seen as actionable rules. Now, by comparing these two descriptions, we may find the cause why these accounts have been closed and formulate an action which if undertaken by the

bank, may prevent other customers from closing their accounts. For example, an action rule may say that by inviting people from a certain group of customers for a glass of wine by the bank, it is almost guaranteed that these customers will not close their accounts and they do not move to another bank. Sending invitations by regular mail to all these customers or inviting them personally by giving them a call are examples of an action associated with that action rule.

In (Tzacheva & Ras, 2004) the notion of a cost and feasibility of an action rule was introduced. The cost is a subjective measure and feasibility is an objective measure. Usually, a number of action rules or chains of action rules can be applied to re-classify a certain set of objects. The cost associated with changes of values within one attribute is usually different than the cost associated with changes of values within another attribute. The strategy for replacing the initially extracted action rule by a composition of new action rules, dynamically built and leading to the same reclassification goal, was proposed in (Tzacheva & Ras, 2004). This composition of rules uniquely defines a new action rule. Objects supporting the new action rule also support the initial action rule but the cost of reclassifying them is lower or even much lower for the new rule. In (Ras & Dardzinska, 2006) authors propose a new simplified strategy for constructing action rules. In this chapter, we present an algebraic extension of this method and show the close correspondence between the rules generated by Tree-Based Strategy (Tsay & Ras, 2005) and rules constructed by this newest method.

Background

In the paper by (Ras & Wieczorkowska, 2000), the notion of an action rule was introduced. The main idea was to generate, from a database, special type of rules which basically form a hint to users showing a way to re-classify objects with respect to some distinguished attribute (called a decision attribute). Values of some of attributes, used to describe objects stored in a database, can be changed and this change can be influenced and controlled by user. However, some of these changes (for instance "profit") can not be done directly to a decision attribute. In such a case, definitions of this decision attribute in terms of other attributes (called classification attributes) have to be learned. These new definitions are used to construct action rules showing what changes in values of some attributes, for a given class of objects, are needed to re-classify objects the way users want. But, users may still be either unable or unwilling to proceed with actions leading to such changes. In all such cases, we may search for definitions of values of any classification attribute listed in an action rule. By replacing a value of such attribute by its definition extracted either locally or at remote sites (if system is distributed), we construct new action rules which might be of more interest to users than the initial rule (Tzacheva & Ras, 2004).

We start with a definition of an information system given in (Pawlak, 1991).
By an information system we mean a pair $S = (U, A)$, where:

1. U is a nonempty, finite set of objects (object identifiers),
2. A is a nonempty, finite set of attributes i.e. $a: U \rightarrow V_a$ for $a \in A$, where V_a is called the domain of a.

Information systems can be seen as decision tables. In any decision table together with the set of attributes a partition of that set into conditions and decisions is given. Additionally, we assume that the set of conditions is partitioned into stable and flexible (Ras & Wieczorkowska, 2000).

Attribute $a \in A$ is called stable for the set U if its values assigned to objects from U can not be changed in time. Otherwise, it is called flexible. "Place of birth" is an example of a stable attribute. "Interest rate" on any customer account is an example of a flexible attribute. For simplicity reason, we consider decision tables with only one decision. We adopt the following definition of a decision table:

By a decision table we mean an information system $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$, where $d \notin A_{St} \cup A_{Fl}$ is a distinguished attribute called the decision. The elements of A_{St} are called stable conditions, whereas the elements of $A_{Fl} \cup \{d\}$ are called flexible. Our goal is to change values of attributes in A_{Fl} for some objects in U so the values of the attribute d for these objects may change as well. Certain relationships between attributes from $A_{St} \cup A_{Fl}$ and the attribute d will have to be discovered first.

By Dom(r) we mean all attributes listed in the IF part of a rule r extracted from S. For example, if $r = [(a1,3)*(a2,4) \rightarrow (d,3)]$ is a rule, then Dom(r) = {a1,a2}. By d(r) we denote the decision value of rule r. In our example d(r) = 3.

If r1, r2 are rules and $B \subseteq A_{Fl} \cup A_{St}$ is a set of attributes, then r1/B = r2/B means that the conditional parts of rules r1, r2 restricted to attributes B are the same.

For example if $r1 = [(a1,3) \rightarrow (d,3)]$, then r1/{a1} = r/{a1}.

Assume also that $(a, v \rightarrow w)$ denotes the fact that the value of attribute a has been changed from v to w. Similarly, the term $(a, v \rightarrow w)(x)$ means that a(x)=v has been changed to a(x)=w. Saying another words, the property (a,v) of an object x has been changed to property (a,w). Assume now that rules r1, r2 have been extracted from S and $r1/[Dom(r1) \cap Dom(r2) \cap A_{St}] = r2/[Dom(r1) \cap Dom(r2) \cap A_{St}]$, d(r1)=k1, d(r2)=k2. Also, assume that (b1, b2,…, bp) is a list of all attributes in $Dom(r1) \cap Dom(r2) \cap A_{Fl}$ on which r1, r2 differ and r1(b1)= v1, r1(b2)= v2,…, r1(bp)= vp, r2(b1)= w1, r2(b2)= w2,…, r2(bp)= wp.

By (r1,r2)-action rule on $x \in U$ we mean a statement r:
$[r2/A_{St} \wedge (b1, v1 \rightarrow w1) \wedge (b2, v2 \rightarrow w2) \wedge \ldots \wedge (bp, vp \rightarrow wp)](x) \Rightarrow [(d, k1 \rightarrow k2)](x)$.

Object $x \in U$ supports action rule r, if x supports r1, $(\forall a \in Dom(r2) \cap A_{St})$ [a(x) = r2(a)], $(\forall i \leq p)$[bi(x)=vi], and d(x)=k1. The set of all objects in U supporting r is denoted by $U^{<r>}$.

a (St)	b (Fl)	c (St)	e (Fl)	g (St)	h (Fl)	d (Decision)
a1	b1	c1	e1			d1
a1	b2			g2	h2	d2

Table 1

To define an extended action rule (Ras & Tsay, 2003), let us assume that two classification rules are considered. We present them in Table 1 to better clarify the process of constructing extended action rules. Here, "St" means stable classification attribute and "Fl" means flexible one.

In a classical representation, these two rules have a form:

$r1 = [a1 \wedge b1 \wedge c1 \wedge e1 \rightarrow d1]$, $r2 = [a1 \wedge b2 \wedge g2 \wedge h2 \rightarrow d2]$.

It is easy to check that $[[a1 \wedge g2 \wedge (b, b1 \rightarrow b2)] \Rightarrow (d, d1 \rightarrow d2)]$ is the (r1,r2)-action rule.

Assume now that object x supports rule r1 which means that x is classified as d1. In order to re-classify x to class d2, we need to change its value b from b1 to b2 but also we have to require that g(x)=g2 and that the value h for object x has to be changed to h2. This is the meaning of the extended (r1,r2)-action rule r given below:
$[[a1 \wedge g2 \wedge (b, b1 \rightarrow b2) \wedge (h, \rightarrow h2)] \Rightarrow (d, d1 \rightarrow d2)]$.

Let us observe that this extended action rule can be replaced by a class of new action rules. First, we need to define a new relation $\approx_h$ on the set $U^{<r>}$ as:

$$x \approx_h y \text{ iff } h(x)=h(y), \text{ for any } x, y \in U^{<r>}.$$

Now, let us assume that $V_h = \{h1,h2,h3\}$ and $U^{<r,hi>} = U^{<r>}/\approx_h = \{y \in U^{<r>}: h(y)=hi\}$, for any i=1,2,3. The extended action rule r can be replaced by two action rules:
$[[a1 \wedge g2 \wedge (b, b1 \rightarrow b2) \wedge (h, h1 \rightarrow h2)] \Rightarrow (d, d1 \rightarrow d2)]$ with supporting set $U^{<r,h1>}$ and
$[[a1 \wedge g2 \wedge (b, b1 \rightarrow b2) \wedge (h, h3 \rightarrow h2)] \Rightarrow (d, d1 \rightarrow d2)]$ with supporting set $U^{<r,h3>}$.

This examples shows that extended action rules can be seen as generalizations of action rules. Also, it gives us a hint of how to look for the most compact representations of action rules.

Main Thrust of the Chapter

1. Issues, Controversies, Problems

In this section we present a modification of the action-tree algorithm (Tsay & Ras, 2005) for discovering action rules. Namely, we partition the set of classification rules R discovered from a decision system $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$, where A_{St} is the set of stable attributes, A_{Fl} is the set of flexible attributes and, $V_d = \{d_1, d_2,..., d_k\}$ is the set of decision values, into subsets of rules having the same values of stable attributes in their classification parts and defining the same value of the decision attribute. Classification rules can be extracted from S using, for instance, discovery system LERS (Grzymala-Busse, 1997).

Action-tree algorithm for discovering extended action rules from a decision system S is as follows:

Build Action-Tree:

Step 1.

1. Partition the set of classification rules R in a way that two rules are in the same class if their stable attributes are the same.
2. Find the cardinality of the domain V_{vi} for each stable attribute v_i in S.
3. Take v_i which card(V_{vi}) is the smallest as the splitting attribute and divide R into subsets each of which contains rules having the same value of the stable attribute v_i.
4. For each subset, obtained in step 2, determine if it contains rules of different decision values and different values of flexible attributes. If it does, go to step 2. If it does not, there is no need to split the subset further and we place a mark.

Step 2.

Partition each resulting subset into new subsets each of which contains only rules having the same decision value.

Step 3.

Each leaf of the resulting tree represents a set of rules which do not contradict on stable attributes and also it uniquely defines decision value d_i. The path from the root to that leaf gives the description of objects supported by these rules.

Generate Extended Action Rules:

Form extended action rules by comparing all unmarked leaf nodes of the same parent.

The algorithm starts at the root node of the tree, called Action Tree, representing all classification rules extracted from S. A stable attribute is selected to partition these rules. For each value of that attribute an outgoing edge from the root node is created, and the corresponding subset of rules that have the attribute value assigned to that edge is moved to the newly created child node. This process is repeated recursively for each child node. When we are done with stable attributes, the last split is based on a decision attribute for each current leaf of action tree. If at any time all classification rules representing a node have the same decision value, then we stop constructing that part of the tree. We still have to explain which stable attributes are chosen to split classification rules representing a node of action tree. The algorithm selects any stable attribute which has the smallest number of possible values among all the remaining stable attributes. This step is justified by the need to apply a heuristic strategy (Ras, 1999) which will minimize the number of edges in the resulting tree and the same make the time-complexity of the algorithm lower.

We have two types of nodes: a leaf node and a non-leaf node. At a non-leaf node, the set of rules is partitioned along the branches and each child node gets its corresponding subset of rules. Every path to the decision attribute node, one level above the leaf node, represents a subset of the extracted classification rules when the stable attributes have the same value. Each leaf node represents a set of rules, which do not contradict on stable attributes and also define decision value d_i. The path from the root to that leaf gives the description which objects supporting these rules have to satisfy .

Instead of splitting the set of rules R by stable attributes and next by the decision attribute, we can also start the partitioning algorithm from a decision attribute. For instance, if a decision attribute has 3 values, we get 3 initial sub-trees. In the next step of the algorithm, we start splitting these sub-trees by stable attributes following the same strategy as the one presented for action trees. This new algorithm is called action-forest algorithm.

Now, let us take Table 2 as an example of a decision system S. Attributes a, c are stable and b, d flexible. Assume now that our plan is to re-classify some objects from the class (d,L) into the class (d,H).

	a	b	c	d
x_1	2	1	2	L
x_2	2	1	2	L
x_3	1	1	0	H
x_4	1	1	0	H
x_5	2	3	2	H
x_6	2	3	2	H
x_7	2	1	1	L
x_8	2	1	1	L
x_9	2	2	1	L
x_{10}	2	3	0	L

x_{11}	1	1	2	H
x_{12}	1	1	1	H

Table 2

Table 3 shows the set of classification rules extracted from Table 2 by LERS algorithm (Grzymala-Busse, 1997). The first column presents sets of objects supporting these rules.

Objects	a	b	c	d
$\{x_3,x_4,x_{11},x_{12}\}$	1			H
$\{x_1,x_2,x_7,x_8\}$	2		1	L
$\{x_7,x_8,x_9\}$	2		0	L
$\{x_3,x_4\}$		1	0	H
$\{x_5,x_6\}$		3	2	H

Table 3

First, we represent the set R of certain rules extracted from S as a table (see Table 3). The first column of this table shows objects in S supporting the rules from R (each row represents a rule). For instance, the second row represents the rule $[[(a,2) \wedge (c, 1)] \rightarrow (d, L)]$. The construction of an action tree starts with the set R as a table (see Table 3) representing the root of the tree T_1 in Fig. 1. The root node selection is based on a stable attribute with the smallest number of values among all stable attributes. The same strategy is used for a child node selection. After labeling the nodes of the tree by all stable attributes, the tree is split based on the value of the decision attribute. Referring back to the example in Table 3, we use stable attribute a to split that table into two sub-tables defined by values {1, 2} of attribute a. The domain V_a of attribute a is {1, 2}. Since $card[V_a] < card[V_c]$, then we partition the table into two: one table with rules containing the term (a,1) and another with rules containing term (a,2). Corresponding edges are labeled by values of attribute a. All rules in the sub-table T_2 have the same decision value. So, action rules can not be constructed from sub-table T_2 which means it is not divided any further. Because rules in the sub-table T_3 contain different decision values and a stable attribute c, T_3 is partitioned into three sub-tables, one with rules containing the term (c, 0), one with rules containing (c, 1), and one with rules containing (c, 2). Now, rules in each of the sub-tables do not contain any stable attributes. Sub-table T_6 is not split any further for the same reason as sub-table T_2. All objects in sub-table T_4 have the same value of flexible attribute b. So, there is no way to construct a workable strategy from this sub-table which means it is not partitioned any further. Sub-table T_5 is divided into two new sub-tables. Each leaf represents a set of rules, which do not contradict on stable attributes and also define decision value d_i.

The path from the root of the tree to that leaf gives the description of objects supported by these rules. Following the path described by the term $(a, 2)\wedge(c, 2)\wedge(d , L)$, we get table T_7. Following the path described by the term $(a, 2)\wedge(c, 2)\wedge(d, H)$, we get table T_8. Because T_7 and T_8 are sibling nodes, we can directly compare pairs of rules belonging to these two tables and construct an action rule $[[(a,2) \wedge (b, 1 \rightarrow 3)] \Rightarrow (d, L \rightarrow H)]$.

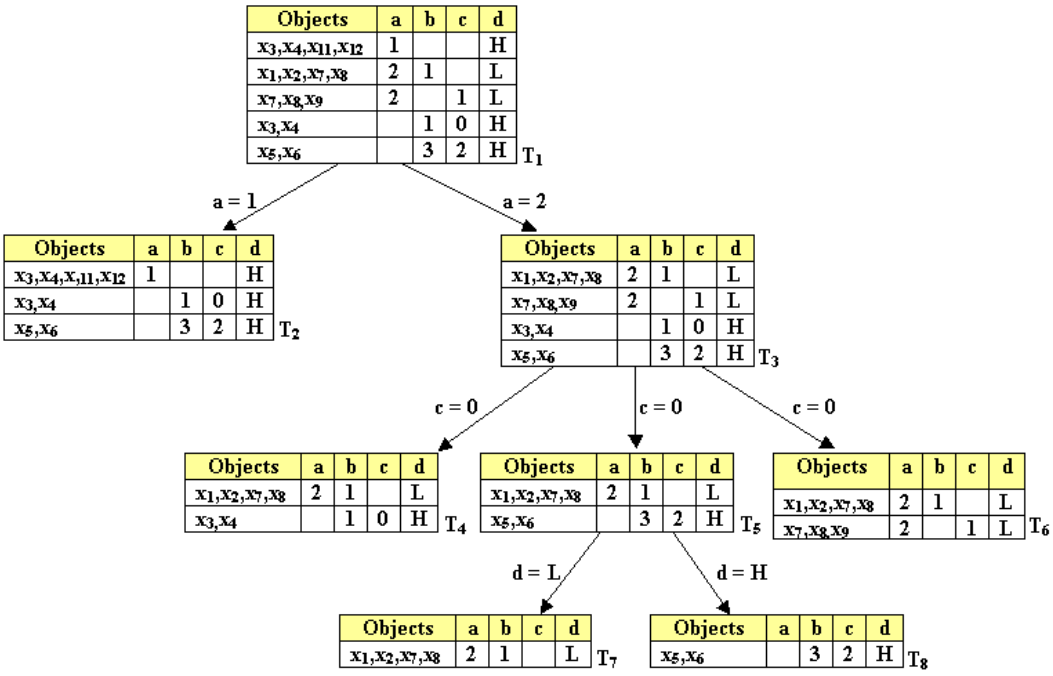

Figure 1

The action-tree algorithm proposed in this section requires the extraction of all classification rules from the decision system before any action rule is constructed. Additionally, the strategy of action rules extraction, presented above, has $0(k^2)$ complexity in the worth case, where k is the number of classification rules. The question is, if any action rule can be constructed from a single classification rule which guarantees the same time complexity of action rules construction as the time complexity of classification rules discovery?

2. Solutions and Recommendations.

Let us assume again that $S = (U, A_{St} \cup A_{Fl} \cup \{d\})$ is a decision system, where $d \notin A_{St} \cup A_{Fl}$ is a distinguished attribute called the decision. The elements of A_{St} are called stable conditions, whereas the elements of $A_{Fl} \cup \{d\}$ are called flexible. Assume that $d_1 \in V_d$ and $x \in U$. We say that x is a d_1-object if $d(x)=d_1$. We also assume that $\{a_1, a_2,..., a_p\} \subseteq A_{Fl}$, $\{b_1, b_2,..., b_q\} \subseteq A_{St}$, $a_{[i,j]}$ denotes a value of attribute a_i, $b_{[i,j]}$ denotes a value of attribute b_i, for any i, j and that

$$r = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge [b_{[1,1]} \wedge b_{[2,1]} \wedge ..... \wedge b_{[q,1]}] \rightarrow d_1]$$

is a classification rule extracted from S supporting some d_1-objects in S. By sup(r) and conf(r), we mean the support and the confidence of r, respectively. Class d_1 is a preferable class and our goal is to reclassify d_2-objects into d_1 class, where $d_2 \in V_d$.

By an extended action rule $r_{[d2 \rightarrow d1]}$ associated with r and the reclassification task $(d, d_2 \rightarrow d_1)$ we mean the following expression:

$$r_{[d2 \rightarrow d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge [(b_1, \rightarrow b_{[1,1]}) \wedge (b_2, \rightarrow b_{[2,1]}) \wedge ..... \wedge (b_q, \rightarrow b_{[q,1]})] \Rightarrow (d, d_2 \rightarrow d_1)].$$

In a similar way, by an extended action rule $r[\rightarrow d_1]$ associated with r and the reclassification task $(d, \rightarrow d_1)$ we mean the following expression:

$r_{[\rightarrow d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge [(b_1, \rightarrow b_{[1,1]}) \wedge (b_2, \rightarrow b_{[2,1]}) \wedge ..... \wedge (b_q, \rightarrow b_{[q,1]})] \Rightarrow (d, \rightarrow d_1)]$.

The term $[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}]$ built from values of stable attributes, is called the header of the action rule $r[d_2 \rightarrow d_1]$ and its values can not be changed.

The support set of the action rule $r[d_2 \rightarrow d_1]$ is defined as:

$Sup(r_{[d2 \rightarrow d1]}) = \{x \in U: (a_1(x)=a_{[1,1]} \wedge a_2(x)=a_{[2,1]} \wedge ... \wedge a_p(x)=a_{[p,1]}) \wedge (d(x)=d_2) \}$.

Clearly, if $conf(r) \neq 1$, then some objects in S satisfying the description

$[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]} \wedge b_{[1,1]} \wedge b_{[2,1]} \wedge ..... \wedge b_{[q,1]}]$

are classified as d_2. According to the rule $r_{[d2 \rightarrow d1]}$ they should be classified as d_1 which means that the confidence of $r_{[d2 \rightarrow d1]}$ will get decreased.

If $Sup(r_{[d2 \rightarrow d1]}) = \emptyset$, then $r_{[d2 \rightarrow d1]}$ can not be used for reclassification of objects. Similarly, $r[\rightarrow d_1]$ can not be used for reclassification, if $Sup(r_{[d2 \rightarrow d1]}) = \emptyset$ for each d_2 where $d_2 \neq d_1$. From the point of view of actionability, such rules are not interesting (Silberschatz & Tuzhilin, 1995, 1996).

In the following paragraph we show how to calculate the confidence of action rules and extended action rules. Let $r_{[d2 \rightarrow d1]}$, $r'_{[d2 \rightarrow d3]}$ are two action rules extracted from S. We say that these rules are p-equivalent ($\approx$), if the condition given below holds for every $b_i \in A_{Fl} \cup A_{St}$:

if r/b_i, r'/b_i are both defined, then $r/b_i = r'/b_i$.

Now, we explain how to calculate the confidence of $r_{[d2 \rightarrow d1]}$.
Let us take d_2-object $x \in Sup(r_{[d2 \rightarrow d1]})$. We say that x positively supports $r_{[d2 \rightarrow d1]}$ if there is no classification rule r' extracted from S and describing $d_3 \in V_d$, $d_3 \neq d_1$, which is p-equivalent to r, such that $x \in Sup(r'_{[d2 \rightarrow d3]})$. The corresponding subset of $Sup(r_{[d2 \rightarrow d1]})$ is denoted by $Sup^+(r_{[d2 \rightarrow d1]})$. Otherwise, we say that x negatively supports $r_{[d2 \rightarrow d1]}$. The corresponding subset of $Sup(r_{[d2 \rightarrow d1]})$ is denoted by $Sup^-(r_{[d2 \rightarrow d1]})$.

By the confidence of $r_{[d2 \rightarrow d1]}$ in S we mean:

$Conf(r_{[d2 \rightarrow d1]}) = [card[Sup^+(r_{[d2 \rightarrow d1]})]/card[Sup(r_{[d2 \rightarrow d1]})]] \cdot conf(r)$.

Now, let us go back to the definition of an extended action rule $r_{[d2 \rightarrow d1]}$ associated with r:

$r_{[d2 \rightarrow d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge ... \wedge a_{[p,1]}] \wedge [(b_1, \rightarrow b_{[1,1]}) \wedge (b_2, \rightarrow b_{[2,1]}) \wedge ..... \wedge (b_q, \rightarrow b_{[q,1]})] \Rightarrow (d, d_2 \rightarrow d_1)]$.

In the previous section, we introduced the relation $\approx_{bi}$ defined on the set $U^{<r>}$ as:

$$x \approx_{b_i} y \text{ iff } b_i(x)=b_i(y), \text{ for any } x, y \in U^{<r>}.$$

Now, assume that $B = \{b_1,b_2,\ldots,b_q\}$ and $\approx_B = \cap\{\approx_{b_i} : b_i \in B\}$. We say that B is b_i-reducible with respect to an extended action rule r in S, if $U^{<r>}/\approx_B = U^{<r>}/\approx_{B-\{b_i\}}$, for any $b_i \in B$.

We say that $C \subseteq B$ is a reduct with respect to an extended action rule r, if $U^{<r>}/\approx_B = U^{<r>}/\approx_C$ and for any $b_i \notin C$, the set C is not b_i-reducible with respect to an extended action rule r in S.

Theorem.
Each reduct with respect to an extended action rule r in S defines uniquely a new extended action rule in S.
Proof (sketch). Assume that

$$r_{[d2 \to d1]} = [[a_{[1,1]} \wedge a_{[2,1]} \wedge \ldots \wedge a_{[p,1]}] \wedge [(b_1, \to b_{[1,1]}) \wedge (b_2, \to b_{[2,1]}) \wedge \ldots\ldots \wedge (b_q, \to b_{[q,1]})] \Rightarrow (d, d_2 \to d_1)]$$

is an extended action rule in S and for instance $[B-\{b_1,b_2\}]$ is a reduct with respect to $r_{[d2 \to d1]}$. It basically means that all objects in $U^{<r>}$ have the same value in a case of attributes b_1, b_2. Let us assume that $b_{[1,i]}$, $b_{[2,j]}$] are these two values. Because $[B-\{b_1,b_2\}]$ is a reduct, then $\{x: [a_1(x)=a_{[1,1]} \wedge a_2(x)=a_{[2,1]} \wedge \ldots \wedge a_p(x)=a_{[p,1]}] \wedge d(x)=d_2 \wedge [b_1(x)=b_{[1,i]} \wedge b_2(x)=b_{[2,j]}]\}$ is not an empty set. The rule

$$[[a_{[1,1]} \wedge a_{[2,1]} \wedge \ldots \wedge a_{[p,1]}] \wedge [(b_1, b_{[1,i]} \to b_{[1,1]}) \wedge (b_2, b_{[2,j]}] \to b_{[2,1]}) \wedge \ldots\ldots \wedge (b_q, \to b_{[q,1]})] \Rightarrow (d, d_2 \to d_1)]$$

is a new extended action rule.

Extended action rules which construction is based on reducts are called optimal.

Future Trends

Assume that d is a decision attribute in S, $d_1, d_2 \in V_d$, and the user would like to re-classify objects in S from the group d_1 to the group d_2. Assuming that the cost of reclassification with respect to each attribute is given, he may look for an appropriate action rule, possibly of the lowest cost value, to get a hint which attribute values need to be changed (Tzacheva & Ras, 2004). To be more precise, let us assume that $R_S[(d, d_1 \to d_2)]$ denotes the set of all action rules in S having the term $(d, d_1 \to d_2)$ on their decision site. Additionally, we assume that the cost representing the left hand site of the rule in $R_S[(d, d_1 \to d_2)]$ is always lower than the cost associated with its right hand site.

Now, among all action rules in $R_S[(d, d_1 \to d_2)]$ we may identify a rule which has the lowest cost value. But the rule we get may still have the cost value which is much too high to be of any help. Let us notice that the cost of the action rule

$$r = [(b_1, v_1 \to w_1) \wedge (b_2, v_2 \to w_2) \wedge \ldots \wedge (b_p, v_p \to w_p)] \Rightarrow (d, d_1 \to d_2)$$

might be high only because of the high cost value of one of its sub-terms in the conditional part of the rule.

Let us assume that $(b_j, v_j \to w_j)$ is that term. In such a case, we may look for an action rule in $R_S[(b_j, v_j \to w_j)]$ which has the smallest cost value.

Assume that
$r_1 = [(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})] \Rightarrow (b_j, v_j \rightarrow w_j)$

is such a rule. Now, we can compose r with r_1 getting a new action rule given below:

$r = [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge ... \wedge$
$[(b_{j1}, v_{j1} \rightarrow w_{j1}) \wedge (b_{j2}, v_{j2} \rightarrow w_{j2}) \wedge ... \wedge (b_{jq}, v_{jq} \rightarrow w_{jq})]$
$\wedge \ldots \wedge (b_p, v_p \rightarrow w_p)] \Rightarrow (d, d_1 \rightarrow d_2)$

Clearly, the cost of this new rule is lower than the cost of r. However, if its support in S gets too low, then such a rule has no value to us. Otherwise, we may recursively follow this strategy trying to lower the cost needed to re-classify objects from the group d_1 into the group d_2. Each successful step will produce a new action rule which cost is lower than the cost of the current rule. Obviously, this heuristic strategy always ends. Interestingness of rules is closely linked with their cost. It means that new algorithms showing how too look for rules of the lowest cost are needed. An example of such an algorithm can be found in (Tzacheva & Ras, 2005).

Conclusion

Attributes are divided into two groups: stable and flexible. By stable attributes we mean attributes which values can not be changed (for instance, age or maiden name). On the other hand attributes (like percentage rate or loan approval to buy a house) which values can be changed are called flexible. Classification rules are extracted from a decision table, using standard KD methods, with preference given to flexible attributes - so mainly they are listed in a classification part of rules. Most of these rules can be seen as actionable rules and the same used to construct action-rules. Two methods for discovering extended action rules are presented. The first one, based on action trees, requires to compare pairs of classification rules and depending on the result of this comparison, an action rule is either build or not. The second strategy shows how to construct extended action rules from a single classification rule. In its first step, the most general extended action rule is build and next it is partitioned into a number of atomic expressions representing a new class of extended action rules which jointly represent the initial extended action rule. The first strategy can be seen as bottom up method whereas the second strategy is a classical example of a top-down method.

References

Adomavicius, G., Tuzhilin, A. (1997). *Discovery of actionable patterns in databases: the action hierarchy approach*, in Proceedings of KDD97 Conference, Newport Beach, CA, AAAI Press

Geffner, H., Wainer, J. (1998). *Modeling action, knowledge and control*, ECAI 98, 13th European Conference on AI, (Ed. H. Prade), John Wiley & Sons, 532-536

Greco, S., Matarazzo, B., Pappalardo, N., Slowinski, R. (2005). *Measuring expected effects of interventions based on decision rules*, Journal of Experimental and Theoretical Artificial Intelligence, Taylor Francis, Vol. 17, No. 1-2

Grzymala-Busse, J. (1997). *A new version of the rule induction system LERS*, Fundamenta Informaticae, Vol. 31, No. 1, 27-39

Liu, B., Hsu, W., Chen, S. (1997). *Using general impressions to analyze discovered classification rules*, Proceedings of KDD97 Conference, Newport Beach, CA, AAAI Press

Pawlak, Z. (1991). *Information systems - theoretical foundations*, Information Systems Journal, Vol. 6, 205-218

Ras, Z. (1999). *Discovering rules in information trees*, in Principles of Data Mining and Knowledge Discovery, (Eds. J. Zytkow, J. Rauch), Proceedings of PKDD'99, Prague, Czech Republic, LNAI, No. 1704, Springer, 518-523

Ras, Z.W., Dardzinska, A. (2006). *Action rules discovery, a new simplified strategy*, in Foundations of Intelligent Systems, Proceedings of ISMIS'06, F. Esposito et al. (Eds.), Bari, Italy, LNAI, No. 4203, Springer, 445-453

Ras, Z., Wieczorkowska, A. (2000). *Action Rules: how to increase profit of a company*, in Principles of Data Mining and Knowledge Discovery, (Eds. D.A. Zighed, J. Komorowski, J. Zytkow), Proceedings of PKDD'00, Lyon, France, LNAI, No. 1910, Springer, 587-592

Ras, Z.W., Tzacheva, A., Tsay, L.-S. (2005). *Action rules*, Encyclopedia of Data Warehousing and Mining, (Ed. J. Wang), Idea Group Inc., 1-5

Silberschatz, A., Tuzhilin, A., (1995). *On subjective measures of interestingness in knowledge discovery*, Proceedings of KDD'95 Conference, AAAI Press

Silberschatz, A., Tuzhilin, A., (1996). *What makes patterns interesting in knowledge discovery systems*, IEEE Transactions on Knowledge and Data Engineering Vol. 5, No. 6

Tsay, L.-S., Ras, Z.W. (2005). *Action rules discovery system DEAR, method and experiments*, Journal of Experimental and Theoretical Artificial Intelligence, Taylor & Francis, Vol. 17, No. 1-2, 119-128

Tsay, L.-S., Ras, Z.W. (2006). *Action rules discovery system DEAR3*, in Foundations of Intelligent Systems, Proceedings of ISMIS'06, F. Esposito et al. (Eds.), Bari, Italy, LNAI, No. 4203, Springer, 483-492

Tzacheva, A., Ras, Z.W. (2005). *Action rules mining*, International Journal of Intelligent Systems, Wiley, Vol. 20, No. 7, 719-736