TECHNICAL REPORT

No. CCLS-08-01

Title: From Classification Rules to Action Recommendations

Authors: Ansaf Salleb-Aouissi, Ronan Trépos, Marie-Odile Cordier, Véronique Masson

From Classification Rules to Action Recommendations

Ansaf Salleb-Aouissi[1] ANSAF@CCLS.COLUMBIA.EDU
Ronan Trepos[2,3] RONAN.TREPOS@IRISA.FR
Marie-Odile Cordier[2] MARIE-ODILE.CORDIER@IRISA.FR
Véronique Masson[2] VERONIQUE.MASSON@IRISA.FR

[1] *Center for Computational Learning Systems (CCLS)*
Columbia University 475 Riverside Drive, New York - NY 10115 USA

[2] *Université de Rennes 1*
Campus Universitaire de Beaulieu 35042 Rennes Cedex - France

[3] *Institut National de la Recherche Agronomique (INRA)*
UMR SAS Agrocampus rue de Saint Brieuc Rennes Cedex - France

Editor:

Abstract

Rule induction has attracted a great deal of attention in Machine Learning and Data Mining. However, generating rules is not an end in itself because their applicability is not straightforward especially when the number of rules is large. Ideally, the user would ultimately like to use these rules to decide which actions to take. In the literature, this notion is usually referred to as *actionability*. The contribution of this paper[1] is two-fold: first we propose a survey of the main approaches developed to address actionability. This topic has received growing attention in the past years. We present a classification of the main research in this area as well as a comparative study between the different approaches. Second, we propose a new framework to address actionability. Our goal is to lighten the burden of analyzing a large set of classification rules when the user is confronted with an "unsatisfactory situation" and needs help to decide what appropriate actions to take in order to remedy the situation. The method consists in comparing the situation to a set of classification rules. This is achieved by using a suitable distance that allows one to suggest action recommendations requiring minimal changes to improve the situation. We propose the algorithm DAKAR for learning action recommendations and we present an application to environment protection. Our experiment shows the usefulness of our contribution for action recommendation but also raises some concerns about the impact of the redundancy of a set of rules in learning action recommendations of good quality.

Keywords: Actionability, decision support, rule-based classifier, generalized Minkowski metrics, maximally discriminant descriptions.

1. A preliminary version of this paper appeared in the proceeding of the European Conference on Machine Learning 2005. Most of this work was done while the first author was a visitor at INRIA, Rennes.

1. Introduction

Rule induction has attracted a lot of attention in Machine Learning and Data Mining. However, the exploitation of a set of induced rules is usually left to the user. Overwhelmed by the number of rules, the user is often frustrated because the applicability of these rules is not immediate. This makes the post-analysis of induced rules a great challenge, yet it is a necessary step to assist the user in his work of decision making. Rather than simply presenting the listing of rules to the user, the ideal would be to translate these rules into feasible and concrete actions. This may be of great interest in many application domains such as health-care, customer analysis and environmental protection.

We are interested in rule-based classifiers. the user needs to use these rules not only for prediction, but possibly many additional tasks. For example, this occurs when the user wants to decide which action to take in order to improve a given unsatisfactory situation (for instance, how to cure an ill patient given a set of rules describing ill and not ill patients). This brings us to the notion of actionability, described in Silberschatz and Tuzhilin (1996) as follows: "A pattern is interesting to the user if the user can do something about it; that is, the user can react to it to his or her advantage".

There is much recent work for learning actionable knowledge (Piatetsky-Shapiro and Matheus (1994); Adomavicius and Tuzhilin (1997); Ras and Wieczorkowska (2000); Liu et al. (2001); Ling et al. (2002); Yang and Cheng (2002); Ras and Tsay (2003); Yang et al. (2003); Elovici and Braha (2003); Lavrač et al. (2004); He et al. (2005); Jiang et al. (2005); Zhao et al. (2005b,a)). In Section 2, we propose to survey the main approaches developed to address actionability. Basically, this work can be classified into 3 lines of research: mapping extracted knowledge with predefined actions, action rules identification, and finally, elaboration of actions. From our point of view, actionability remains under investigation and deserves more attention. For example, none of these approaches has addressed the task of mining action recommendations when the user faces an unsatisfactory situation he wants to improve thanks to a set of classification rules. We propose a new framework, described in Section 3, to address this kind of actionability. We suggest the algorithm DAKAR (Discovery of Actionable Knowledge And Recommendations) which works as follows: starting from an unsatisfactory situation and relying on a set of classification rules, DAKAR discovers a set of action recommendations that propose to the domain-expert the "little" changes needed in order to improve that situation.

More precisely, we focus on propositional frameworks, where a situation is expressed by a set of attribute-value pairs. In our case, an *action* is a modification of the values of some features describing a given situation. We compute actions involving "little" changes in the initial situation. This is achieved in DAKAR by using the generalized Minkowski metric proposed in Ichino and Yaguchi (1994). This metric gives the distance between two descriptions, a *description* being a set of attribute-value pairs. In our approach, a weight is assigned to each feature in order to take into account how flexible this feature is. The *search space of actions* is defined as the set of *maximally discriminant descriptions* that differentiate a set of classification rules distinguishing a satisfactory from an unsatisfactory situation, according to a distance threshold δ. Note that unlike other methods, all classification rules are considered in our framework. This is because induced rules are uncertain (not implications) and considering all of them enlarges the spectrum of possible actions to consider in the search space. The search space is explored considering two properties we define, which are the *consistency* and the *validity* of actions. DAKAR adopts a beam-search strategy to find the best actions to suggest to the user, according to a *quality* criterion we define. We have conducted an experimental evaluation of DAKAR on an environmental application as described in Section 3.5. The actionable knowledge discovered by DAKAR concerns the possible recommendations for reducing the pollution by pesticides in a catchment area. Experiments have shown the feasibility and the usefulness of this task. They have also raised some concerns about the impact of the redundancy of rules on the quality of actions recommended by our system.

In Section 4, we present a comparative study of the state-of-the-art methods and show how our approach relates to the previous work. Limits of our contribution as well as future directions are discussed in Section 5. Finally, we conclude in Section 6.

2. Related Work

This section is a survey of the main approaches developed to address actionability. In Silberschatz and Tuzhilin (1996), the term *actionability* is evoked for the first time in the context of interestingness measures for patterns evaluation. The authors have classified such measures into objective (data-driven) and subjective (user-driven) measures. According to them, from a subjective point of view, a pattern is interesting if it is :

- Actionable : the end-user can act on it to his advantage.
- Unexpected : the end-user is surprised by such a finding.

As pointed out by the authors, actionability is difficult to capture; they propose rather to capture it through unexpectedness, arguing that unexpected patterns are those that lead the expert to make some actions.

While many works have addressed the unexpectedness issue (see for instance Duval et al. (2007) for a survey), the actionability remains to be further investigated even if we noticed recently a great interest in developing new methods for the discovery of actionable knowledge. We will focus in the following on the methods developed to address the actionability issue by using classifiers. We refer the reader to He et al. (2005) for a relevant state of the art about actionability in other machine learning tasks such as clustering. We will postpone the comparison of the different state-of-the-art methods, including our contribution, until later in this article in a comparative study presented in Section 4.

Approaches to actionability can be classified into three main lines of research: the first one maps extracted knowledge, such as classification rules and deviations, to actions predefined by the users (Section 2.1). The second line of research (Section 2.2) sift through a large number of existing rules to identify actionable ones. Finally, the last approach is based on elaborating actions (Section 2.3). In the following, we survey the methods proposed in these lines in details.

2.1 Predefined actions

• Piatetsky-Shapiro and Matheus (1994) developed KEFIR, a pioneer health-care system dealing with *deviations*. The system embodies a *recommendation generator* that suggests corrective actions, defined *a priori* by health-care experts, in response to some relevant deviations discovered in the data. More precisely, a deviation is a difference between an observed value taken from the current data and a reference or an expected value. Deviations are also ranked according to a measure of interestingness which tries to capture the interest of the user to such patterns and identify the most relevant findings. Given a database, a domain knowledge and some norms, the KEFIR engine first computes the deviations in the databases according to the norms. These deviations are then evaluated and ordered using a domain-specific interestingness measure estimating the benefit obtained when actions are undertaken to correct detected deviations. Deviations along with their explanations and recommendations are presented in a written report to the domain expert.

• A decision-theoretic framework evaluating classification systems from an economic point of view is proposed in Elovici and Braha (2003). The authors suggest a model that consider a situation where a decision-maker uses a system to classify an individual (for instance a customer) and then chooses the action to accomplish with regard to: this classification, the prior probabilities of the actual classes in the population under study, a set of actions that can be undertaken, and the payoffs of actions according to the actual classes. For example, in the bank credit domain, an applicant may be classified as low, medium or high risk applicant. The task addressed in this paper is to help the decision-maker to decide which action to undertake. Should the credit application be approved or rejected according to the predicted class? More formally, let $S = (s_1, s_2, \ldots, s_{ns})$ be the set of actual classes, $\pi = (\pi_1, \pi_2, \ldots, \pi_{ns})$ the probabilities *a priori* of actual classes in the population under study, $Y = (y_1, y_2, \ldots, y_{ny})$ the predicted classes, which is generally equal to S, and $A = (a_1, a_2, \ldots, a_{na})$ the set of actions that can be taken by the decision-maker. The performance of a classification system is usually evaluated by a *confusion matrix*, P. It is a $n_s \times n_y$ matrix of conditional probabilities. A cell p_{ij} defines the probability of deciding a predicted class y_j given an example with actual class s_i. A payoff matrix U of size $n_a \times n_s$ associates payoffs to pairs of actions and actual classes. A cell u_{ij} gives the payoff when the decision maker applies action a_i and the example actual class is s_j. The decision rule is described by a $(n_y \times n_a)$ matrix D. Each cell d_{ij} expresses the probability that the decision-maker applies action a_j given the predicted class y_i. The following formula gives the expected payoff for a decision rule D where the matrix Π is square and has the vector of prior probabilities π in its diagonal and zero elsewhere.

$$EU = trace(P\ D\ U\ \Pi)$$

Maximizing the expected payoff is achieved by choosing an optimal binary matrix D^*, called here the *optimal decision rule*.

In order to compare the effectiveness of classification systems, the authors suggest to compare the *effectiveness* of their confusion matrices: “the confusion matrix Q is said to be *more effective* than the confusion matrix R if the maximal expected payoff yielded by R is not larger than that yielded by Q for *all* payoff matrices U and *all* prior probability matrices Π”. The authors rely on a theorem for ordering confusion matrices due to Blackwell (1951). It states that the confusion matrix Q is more effective than the confusion matrix R if and only if there exists a stochastic matrix M such that $Q.M = R$. In such case, we say that the classification system represented by Q is more effective than the one represented by R regardless of payoff or class information. Effective classification systems would increase the expected payoff, or at least do not worsen it. The authors suggest also to combine two or more classification methods into one Cartesian composite system using the *Cartesian product* operator they have introduced. In a certain way, systems composition provides a ”second opinion” about the classification of examples (Ahituv and Ronen (1988)) and improves the maximum expected payoff achieved by a standard classifier, regardless of payoff or class distribution information.

The authors also investigated the relationship between the notions of *investment* and *payoff*. For this purpose, they extended the model above-described to take into consideration the investment cost of the KDD process. They concluded that the larger the investment in the KDD process, the larger the profit. This work considers classification systems from their effectiveness in decision-making point of view. It relies on a solid theoretical framework and remains suitable for applications where domain knowledge is available and the decision-maker knows exactly the actions that can be undertaken given a classification result.

• Adomavicius and Tuzhilin (1997) proposed an approach to discover actionable patterns based on a concept they introduced called *action hierarchy* defined as a tree of actions. Here all possible actions are predefined by the expert of the domain and are organized into a hierarchy. Actions are organized from the more general actions on the top of the hierarchy to the more specific actions at the bottom. Note that the authors distinguish clearly the *actions* which are the nodes of the hierarchy from the *actionable patterns*, which are the patterns assigned by the user to each node. An actionable pattern is either an individual pattern (for e.g. an instantiated association rule), data mining queries (see for e.g. Imielinski et al. (1996)) or a pattern template (see Klemettinen et al. (1994)). When executed on a database, a data mining query produces a set of patterns matching this query. Likewise, when applied to a set of rules, a pattern template extracts only the rules matching the template. The discovery of actionable patterns is achieved in the proposed method in three steps, illustrated here through an example from Adomavicius and Tuzhilin (1997):

1. Build an action tree for the application at hand. The root is the most general action such as *Actions for market manager*. The internal nodes represent less general action such as *product stocking* or *how to arrange products in the store*. Leaves represent the most specific actions such as *put a product on sale*, *restock product*.

2. Assign a set of actionable patterns to each node. For example, an association rules on product categories can be assigned to the action node *how to arrange products in the store* in order to find the categories of products that are purchased together.

3. Execute the data-mining queries by traversing the whole action tree or focusing on a part of it. This produces a set of patterns and would give insights to the user about the decision to take. For example, when the previous template is executed, we get all the associations between the store products. Such associations may give the manager some hints about choosing the products to put together on shelves.

Organizing actions into a hierarchy can help the user to discover efficiently useful action patterns. However, this approach “identifies” some predefined action patterns and do not discover truly new actions. Moreover, conceiving such a hierarchy and finding the data mining queries to be assigned to its nodes remain a hard task to the user especially when the hierarchy structure and the nodes contents change quickly over time.

• In a recent work, Jiang et al. (2005) address the problem of extracting patterns that respond to actions. Given a data set $D(F_1, \ldots, F_m, U)$, where F_i are features and U is a utility measure to be maximized. The domain of F_i and U are small numbers of linearly ordered scales; *i.e.* the first few integers. Assuming that the user knows some simple actions $A_1, \ldots, A_n$ and how they influence the features. The authors are interested in how actionable are the actions in terms of increasing the utility U through the influence of those actions on some features. They assume that if certain features are correlated with the utility, a change in those features implies a certain change in the utility. For this purpose, an influence matrix M specifies how actions affect features: an element

M_{ij} specifies the influence of an action A_i on feature F_j and is called a *destination range* $[a, b]$ defined by:

1. $a = b = -$, if A_i has no influence on F_j
2. $a < b$, if A_i increases the current value c in $[a, b]$ for F_j to some target value in $[c, b]$
3. $a > b$, if A_i decreases the current value c in $[a, b]$ for F_j to some target value in $[b, c]$

The influence of an actionset (a set of actions) on a feature value $F_j = f_j$ is the union of the destination ranges of applying all the actions in that set separately.

A *destination space* denoted by $DS_{1,k} = ([l_1, h_1], \ldots, [l_k, h_k])$ describes the feature ranges after applying the action set to any customer described by the features values $\hat{f} = (f_1, \ldots, f_j)$, $j < k$. The actionability of an actionset AS on a customer c in a population P is estimated by computing the new utility of c after the change. Here, customers that fall into the destination space $DS_{1,k}$ are called a *role model* of the population P and is denoted by $rm(P)$. The underlying idea is to use the utility of the role model to estimate the utility of c.

The actionability of AS on P is defined as follows:

$$Act(P, AS, \hat{f}) = \sum_{c \in P} (agg(rm(P)) - c.U)$$

where the actionability of an actionset AS on a customer c in the population P is measured by $agg(rm(P)) - c.U$, $agg(rm(P))$ being some aggregate operator (for ex. average, sum) on the utility of customers in $rm(P)$ and $c.U$ denotes the utility of c.

More generally, the authors define the problem of *Action Feature Utility* mining as finding a partition $\{D_1, \ldots, D_q\}$ of the set of customers that maximizes the aggregated actionability, where the actionability is a real value denoted [2] by $PAct_i$

$$\sum_i PAct_i$$

A solution consists of action sets AS_i and the description $\hat{f}_i$ of the population D_i, $1 \leq i \leq q$.

Given a future customer matching the description D_i, one can recommend the actionset AS_i. An approximate solution to this problem is also proposed and consists in iteratively partitioning the data so as to maximize the utility by adopting decision-tree like approach: initially the root contains the whole dataset. Nodes are partitioned according to a feature selected by some criteria called the *actionability gain* defined for a feature F_k (taking its values in $(f_{k1}, \ldots, f_{ku})$) at a node N_p is computed by:

$$ActGain(F_k, N_p) = \sum_{i=1}^{u} Act(P_i, AS_i, \hat{f}_i) - Act(P_p, AS_p, \hat{f}_p)$$

where $\hat{f}_p = (f_1, \ldots, f_{k-1})$ and $\hat{f}_i = (f_1, \ldots, f_{k-1}, f_{ki})$. To partition the population P_p at node N_p, the feature F_k that maximizes $ActGain$ is selected. The tree is developed until for any leaf node, there is no available feature or the actionability gain becomes negative. To avoid over-fitting, a post-pruning is also used to maximize the actionability of future customer.

Only single actions are considered in this framework in the sense that the influence of an action on a feature is assumed to be independent of the other features and actions. The authors argue that a

2. $PAct_i$ refers to future customers whereas $Act(P, AS, \hat{f})$ refers to customers in the given data

user tends to do one thing at a time and independence hypothesis has already shown good results as for naive Bayes classifier. Also, categorical and numerical features are not handled by this approach since feature domains are restricted to the first few integers.

2.2 Actionable rules identification

• In Liu et al. (2001), the authors propose a method for pruning all non actionable rules from a large set of association rules. The method proposed here is two-step. First, association rules are generated. Only rules having a class-value in their right-hand sides are considered. Among them, non significant rules are pruned. The significance is evaluated thanks to the chi-squared χ^2 test[3]. Second, the remaining rules are analyzed to identify all potentially actionable rules and delete the rules that are not potentially actionable. In their framework, a potentially actionable rule is defined as a rule $R : A \rightarrow C$ which either:

1. does not have any descendant rule. A rule r is descendant of another rule R if both have the same right-hand side and if the left-hand side of r is a superset of the one of R,
2. or has some potentially actionable descendant rules such that, after removing the data tuples covered[4] by the descendants of R, the rule R is still significant w.r.t. the default rule "$\longrightarrow C$". This significance is measured once again with the χ^2 test.

In other words, the underlying idea is to discard the more general rules using their specialized rules (having more conditions in their left-hand sides) with higher quality. For instance, consider these rules, where s denotes the support and c the confidence:

$$R_1 : asthenia = yes \longrightarrow flu = yes \quad (s = 18\%,\ c = 55\%)$$
$$R_2 : asthenia = yes \wedge temperature = high \longrightarrow flu = yes \quad (s = 10\%,\ c = 90\%)$$
$$R_3 : asthenia = yes \wedge headache = yes \longrightarrow flu = yes \quad (s = 5\%,\ c = 80\%)$$

The idea is to use the rules R_2 and R_3 having better confidences rather than the general one R_1 for performing some actions (as for instance prescribing medicines for fever/headache). However, before removing R_1, one have to verify if, after removing the tuples covered by R_2 and R_3, R_1 is still not a strong rule.

As pointed out by the authors, the proposed approach may produce a useful by-product: association rules with negative conditions. Notice that such rules, that can be quite useful, cannot be mined by a traditional association rule discovery process. The following rule is an example:

$$asthenia = yes \wedge \neg(temperature = high) \wedge \neg(headache = yes) \longrightarrow flu = no$$

An experimental evaluation has been conducted on 30 datasets. It has shown that, on average, 34% of significant association rules are not actionable and are pruned by this method. Furthermore, this experiment has shown the efficiency of this method.

This approach indeed facilitates the hard task of analyzing large sets of rules by eliminating non-actionable rules. But, it lies more in the elimination of redundant rules approaches; strictly speaking, it does not truly identify action rules.

3. The reader interested in the evaluation of association rules thanks to the χ^2 test, can refer to the relevant work proposed in Brin et al. (1997)
4. A rule covers a tuple (record) if the tuple satisfies the condition part of this rule.

The authors suggested in another contribution Zhao et al. (2005b,a) to identify actionable rules through visualization. Notice that the rules considered here are *associative classification rules*, *i.e.*, association rules having a class value in their right-hand sides and handling well umbalanced data sets in comparison to decision trees for instance. In this work such rules are mined with the CBA algorithm proposed in Liu et al. (1998). They proposed a framework called *Opportunity Map Visualization* which works tightly with the user as follows: Data and discovered rules are summarized into an interactive matrix where the columns lists all the attributes and the lines all the classes. This matrix allows the user to situate where the opportunities are. By opportunity, one mean important rules (attribute-value combinations) that affect the important classes. The user can rearrange the disposition of attributes and classes in the matrix. The important classes can be placed on the top of the matrix and the actionable attributes on the left side. Such rearrangement produces 4 sectors in the matrix that are more or less actionable and interesting to the user. Obviously, the upper-left sector is the most important one since it concerns the most actionable attributes and the most important classes according to the user. When an interesting attribute is detected, one can focus on it by using a *drill-down visualization* on a particular cell (this attribute and a class value). This allows the user to visualize the values of that attribute, how many rules involve that attribute, their confidences and supports, and the original data covered by the rules. This work has been implemented in an interactive system and applied to a real-life case study for Motorola Mobile Company. The dataset used have more than 500 attributes, 15 classes and 100,000 tuples. The evaluation of the system has been assessed subjectively by Motorola's engineers as an interesting system for detecting truly useful and actionable rules. Although the concept of organization map is a quite interesting way to focus quickly on what is important, one may wonder how the system is successful in discovering actionable knowledge and visualize correctly rules, attributes and classes as their number goes up.

• The authors in Gamberger and Lavrač (2002) define actionable knowledge as a symbolic knowledge, typically presented in the form of rules, that allows the decision-maker to recognize some important relations and to perform some actions. This is achieved by uncovering the properties of subgroups of the population at hand helping in directing some targeting campaign. For instance, thanks to the following rule expressing a relationship between the weight (assessed by the Body Mass Index, BMI), age and possibly coronary heart illness of patient,

$$BMI > 25 \wedge age > 63 \longrightarrow Coronary\ Heart\ Disease$$

the practitioner can target the overweighted patients that are over 63 years old, warn them about the risk of a coronary heart disease and perhaps advise them to start a diet.

The authors consider the two most common approaches in predictive induction, *classification rule learning* and *decision tree learning*. They discussed in Lavrač et al. (2002); Gamberger and Lavrač (2002) their shortcomings and debated why such knowledge is not actionable just as it is. The main reasons evoked are that:

- Classification rules formed from the paths of decision trees (leading from the root node to the class labels in the leaves) are descriptions that best discriminate between classes and thus are not actionable.
- Classification rules generated by a covering algorithm and forming characteristic descriptions are not actionable. Indeed, except the first few rules with a sufficient coverage may be of interest. Covering does not allow to discover subgroup properties of the entire population.

- Both of classification rules and decision trees could be used to classify a selected population but this is unpractical.

Gamberger and Lavrač (2002) suggested a heuristic beam search rule learning algorithm called *Algorithm SD* for *Subgroup Discovery*. Given a target class, this algorithm uncovers rules of the form:

$$Condition \longrightarrow Target\ Class$$

where $Condition$ is a conjunction of features describing target class subpopulations. Such rules are relatively general rules and may cover also some non target examples. The authors pointed out the importance of effective expert-guided subgroup discovery in the TP/FP [5] space in order to induce knowledge with various generalization granularity level. This is achieved by tuning the parameters in *Algorithm SD*. More precisely, the algorithm search for the rules with a maximal q value computed using a user *TP/FP-tradeoff* function, known as ROC analysis. In the TP/FP space, False Positive Rate $FPr = \frac{FP}{Neg}$ is plotted on the x-axis and the *sensitivity* of True Positive Rate $TPr = \frac{TP}{Pos}$ plotted on the y-axis. *FPr* need to be minimized while *TPr* maximized.

Three case studies related to medical and marketing domains and the lessons learned are reported in Lavrač et al. (2002, 2004). It is also interesting to notice how the authors distinguish actionable rules from operational ones: Operational rules are somewhat actionable "ready to use" rules, that is, if applied, will affect a target population immediately.

2.3 Elaboration of actions

• Choosing high-utility actions that increase the probability of success of reclassification of an individual/case while reducing the costs is formulated as a case-based reasoning problem in Yang and Cheng (2002). In fact, given a new *case* to reclassify, it seems quite natural to search among the training cases (examples), those that are close to this case and having the desirable class. In their framework, mining action recommendations is a two-step process:

1. Mining case bases: Given a dataset containing examples labeled as *positive* and *negative*, perform an analysis on the dataset to find out a number of small and representative cases of examples that can be *role models*. Three approach to address mining case-bases are proposed:

    (a) Instance-based approach: uses the entire positive population. As pointed out by the authors, although simple and optimal, this approach is computationally inefficient.

    (b) Cluster-centroid-based approach: constructs clusters of positive instances and then extracts their centroids representing the case-bases. Clustering is done with the k-medoids method. In this method, we need to specify the number k of clusters to be generated by the k-medoids method. Moreover, only the positive class distribution is taken into consideration. The authors suggest the SVM-based mining method to overcome these two drawbacks.

    (c) SVM-based approach: in order to take into consideration both positive and negative class clusters, this approach identifies the positive cases on the boundary between positive and negative cases. The underlying idea is as follows: assuming that we have two

5. TP denotes true positive cases, FP false positives. Also, Neg (resp. Pos) denotes the number of negative (resp. positive) cases.

clusters, one positive and one negative. If we consider an example from the negative cluster, there is certainly a nearer example in the border of the positive cluster than its centroid and thus the total cost of reclassification would be less. These positive cases along the boundary hyper-plane correspond to the support vectors found by an SVM classifier. Let us notice that this approach is only appropriate when the distribution of data shows a net separation between classes. Furthermore, this idea holds for negative examples of the dataset and is not suitable if one want to reclassify an *incoming* example which does not belong necessarily to any negative cluster.

2. Switching plane generation: Negative cases are switched to positive cases using actions suggested by the role models. For a negative example we want to switch to positive, its nearest neighbor is identified among the positive cases. The difference between a negative case, one want to reclassify, and its one-nearest neighbor in the case base is then used to generate a *switch plan*. Given a test set DB of negative examples to reclassify, the total cost for converting all these examples into positive examples is computed in Yang and Cheng (2002) as follows:

$$Cost = \sum_{i=1}^{|DB|} \sum_{j=1}^{l} cost(A_{ij}, v_1, v_2)$$

   where $cost(A_{ij}, v_1, v_2)$ denotes the cost of changing the value of A_{ij}, the j^{th} attribute for the i^{th} example, from $v1$ to v_2. Those costs are set manually. For each nominal attribute, a cost matrix is associated while for numeric attributes, a mathematical function is defined.

   The plan used to recommend actions relies on the least cost, called the *MinCost* but also on the *rank* formula given by:

$$Rank(x,t) = p(+|t) - \frac{Cost(x,t)}{maxCost}$$

   where $p(+|t)$ is the probability density of positive instances around a target case t, $Cost(x,t)$ is the cost of switching from a negative example x to t and $maxCost$ is the maximum among the costs of switching from x to every possible case in the case base. The algorithm *MaxRank* evoked in this paper takes into account the probability of success of a switching plan from x to each t in the case base in order to choose the case that maximizes the rank formula and thus guarantees to generate a plan that will achieve the switching.

Experimental results conducted on both artificial and real datasets have shown that it is recommended to use the SVM-CBMine algorithm for the datasets having a clear separation between the two classes and to privilege the Centroid-CBMine for the other datasets. Moreover, considering switching plans that take into consideration the measure of rank are better than those relying only on the cost.

• Ling et al. (2002); Yang et al. (2003) consider mining *optimal actions* Customer Relationship Management (CRM) relying on decision tree models. They aim at finding those actions that change customers from undesired status to a desired one. In this work, an action is a change in the value of an attribute. The term *hard attribute* designates an attribute that cannot be changed while the term *soft* denotes an attribute that can be changed with a reasonable cost[6]. A cost matrix is associated to each soft attribute. Each cell (v_i, v_j) gives how much it costs to the company to move the value of the attribute from the value v_i to v_j (e.g. changing a loan rate). Actions are chosen so as to maximize the expected net profit by taking into consideration their costs.

$$P_N = P_E \times P_{gain} - \sum Cost$$

where P_N denotes the net profit, P_E the total profit of the customer in the desired status, P_{gain} the probability gain and $Cost$ designates the cost of each action involved.

The method consists in searching in the decision tree if it is possible to replace an incoming customer into a better leaf with a highest net profit. It can be resumed in these steps:

1. Given a database of customers, build a decision tree modeling their profiles. Notice here that both hard and soft attributes are used in order to get an accurate model.

2. For each incoming customer falling in a particular leaf with a certain probability of being in the desired status or class (e.g. being loyal), search for optimal actions that move this customer to another leaf with a better status. For example, if the decision tree predicts that a customer will churn in the future with a high probability, try to reclassify him/her into another leaf by modifying his/her attribute values.

As pointed out by the authors, hard attributes may prevent customers from being moved to other leaf nodes. For example, it will be impossible to move a male customer from a low benefit leaf to a high benefit leaf concerning women (the classification leading to this leaf was done on Sex=female). In order to improve this drawback the authors suggest to build multiple decision trees with subsets of hard attributes. The optimal actions are taken from the best tree with the highest net profit. This method has been implemented in a system called PROACTIVE SOLUTION in Ling et al. (2002). It was called *proactive* because it suggests actions before the situation gets worse, such as loosing a customer. It has been successfully applied to financial and insurance companies where it increases effectively the total spending of customers.

This approach maps each leaf node in a decision tree with a separate customer group. The idea is to find appropriate actions for each customer group to increase its net profit. But in practice, because of human resources limitations, we may need to merge some leaves together into a number of segments, say k. The problem is now to find k customer groups and their corresponding action sets from a decision tree such that applying the discovered actions to the corresponding group will maximize the net profit. This computational problem is addressed in Yang et al. (2003), where it is called *Bounded Segmentation Problem*. Two algorithms were proposed in order to efficiently discover the best k groups: GREEDY-BSP and OPTIMAL-BSP. They were evaluated on a dataset from an insurance company.

• In Ras and Wieczorkowska (2000), the authors propose to discover action rules in order to increase the profit of a company. In their framework, they consider that features are divided into two categories: stable features (those that cannot be changed, e.g. sex, date of birth) and flexible features (e.g. weight, loan). An action rule involves only flexible features and shows what are the changes to

6. As we will see in the rest of this Section, other approaches use rather the terms *flexible* and *stable* to designate *soft* and *hard* attributes respectively.

expend in order to bring some customers from a profit ranking class to a better one. To discover action rules, the authors exploit a kind of information system called decision table $\mathcal{S} = (\mathcal{U}, \mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{F}} \cup \{d\})$ where $\mathcal{U}$ is a set of objects (in this case customers), and $\mathcal{A} = \mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{F}} \cup \{d\}$ a set of attributes describing the objects in $\mathcal{U}$ (customers features). The sets $\mathcal{A}_{\mathcal{S}}$ and $\mathcal{A}_{\mathcal{F}}$ denote stable and flexible attributes respectively and $d \notin \mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{F}}$ is a distinguished flexible attribute called the decision attribute (generally called class attribute). Given an information system $\mathcal{S}$, the algorithm proposed in Ras and Wieczorkowska (2000) and implemented as a system called DAR extracts from $\mathcal{S}$ a set of action rules. Basically, discovering action rules in DAR is a two-step process:

1. Extract from $\mathcal{S}$ the set of optimal rules having the decision attribute d in their right-hand sides. Such rules are induced from $\mathcal{S}$ so that they describe the values of the decision attribute value (or class value) d with a minimal subset of $\mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{F}}$.

2. For each pair of rules extracted in Step 1, say r_1 and r_2, check if they have d_1 and d_2 as decision values (or class values) in their right-hand sides respectively and involve the same values for all stable attributes in their left-hand sides. If these conditions hold, find the set of flexible attributes $\{b_1,\ b_2, \ldots,\ b_p\}$ on which r_1 and r_2 differ and derive the (r_1, r_2)-action rule of the form:
$$[(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \cdots \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow [(d, d_1 \rightarrow d_2)](x)$$
The notation $(a, v \rightarrow w)$ is used to express that the value of the attribute a has been changed from v to w and such rule expresses that if the changes of values of the attributes concerning a customer x match the term $[(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \cdots \wedge (b_p, v_p \rightarrow w_p)](x)$ then the ranking profit of the customer x will change from d_1 to a better rank d_2.

This approach has been generalized to distributed knowledge systems by Ras and Gupta (2002).

DAR considers only rules involving the same stable attributes with the same values to construct action rules. Furthermore, only shared flexible attributes are involved in the generated action rule which may considerably restrict possible action rules. Consequently, all the situations where at least an attribute is present in only one of these rules are simply not considered. Ras and Tsay (2003) addressed this shortcoming of DAR by extending it to the system DEAR (Discovery of Extended Action Rules). Let us first explain the notion of extended action rule defined in Ras and Tsay (2003) through an example. Consider two rules:
$$r_1 : A = a_1 \wedge B = b_1 \wedge C = c_1 \wedge E = e_1 \Longrightarrow D = d_1$$
$$r_2 : A = a_1 \wedge B = b_2 \wedge G = g_2 \wedge H = h_2 \Longrightarrow D = d_2$$

Let us assume that an object x supports r_1 and thus assigned to class d_1. In order to reclassify x to the class d_2, we need to change its B value from b_1 to b_2, verify that its stable G value is g_2 and change its H value from its current value to h_2. This lead to the extended action rule:
$$[(B, b_1 \rightarrow b_2) \wedge (G = g_2) \wedge (H, \rightarrow h_2)](x) \Rightarrow [(D, d_1 \rightarrow d_2)](x)$$
Such a rule could not have been extracted with DAR. The process in DEAR is quite similar to DAR, except that the definition of an extended action rule embodies the attributes that are not shared by the pair of rules. More precisely, given two rules r_1 and r_2, the extended (r_1, r_2)-action rule as defined in Ras and Tsay (2003) is a rule of the form:
$$[(s_1 = u_1) \wedge (s_2 = u_2) \wedge \ldots \wedge (s_q = u_q) \wedge (b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \cdots \wedge (b_p, v_p \rightarrow w_p) \wedge (f_1, \rightarrow t_1) \wedge (f_2, \rightarrow t_2) \wedge \ldots \wedge (f_r, \rightarrow t_r)](x) \Longrightarrow [(d, d_1 \rightarrow d_2)](x)$$
In this rule, $s_1,\ \ldots, s_q$ are stable attributes that are not present in the rule r_1 but present in r_2 and have as values $u_1,\ \ldots, u_q$ respectively. Likewise, the attributes $f_1,\ \ldots, f_r$ denote flexible attributes not present in r_1 but having the values $t_1,\ \ldots, t_q$ respectively in r_2.

The system DEAR has also known significant improvements in Tsay and Ras (2005) where it has been upgraded to the system DEAR2. The authors suggest a way to organize the set of rules that are used to extract extended action rules. First, the set of rules is partitioned into a set of equivalence classes, where each class concerns rules having the same decision value. Then, for each decision value, a tree partitioning each class into new equivalence subclasses is built. This is done so that two rules belong to the same equivalence subclass if the values of their stable attributes are not contradicting each other. Finally, rather than considering all pairs of rules as done in DAR and DEAR, the authors consider only pairs of rules belonging to some of these equivalence subclasses to construct the extended action rules. The experiments conducted on 3 datasets and reported in Tsay and Ras (2005) have shown that this strategy implemented in DEAR2 speed up significantly DEAR.

The Support and confidence of an action rule are also defined in Ras and Tsay (2003); Tsay and Ras (2005). We say that an object x supports the action rule r if there are two rules r_1 and r_2 extracted from $\mathcal{S}$ such that:

- $(\forall i \leq p)[[b_i \in L(r)] \longrightarrow [b_i(x) = b_i(r)]]$ and $d(x) = d_1$,
- object x supports rule r_1,
- if object y is the outcome of the rule r applied on x, then y supports rule r_2.

where $L(R)$ refers to attribute values in the left-hand side (condition part) of the rule r. The support of an action rule in $\mathcal{S}$, denoted by $Supp_{\mathcal{S}}(r)$ is the number of objects in $\mathcal{S}$ supporting this action rule. The confidence of an action rule, denoted by $Conf_{\mathcal{S}}(r) = Supp_{\mathcal{S}}(r)/Supp_{\mathcal{S}}(r_1)$.

Let us now take a close look to the kind of rules used as a base to extract action rules. In the works above-cited, covering rules due to Pawlak (1992) called d-reduct and association rules are used to extract action rules. In another contribution, Ras et al. (2005) used a rough set-based approach, to induce a set of classification rules from $\mathcal{S}$. Such rules have the decision attribute d in their right-hand sides and attributes belonging to $\mathcal{A}_{\mathcal{S}} \cup \mathcal{A}_{\mathcal{F}}$ in their left-hand sides. To induce such rules, the authors extended the system LERS Grzymala-Busse (1992) to the system E-LERS which learns from a decision table the longest classification rules satisfying two thresholds, the minimum support and the minimum confidence. Action rules are then generated as usual from these rules by using DEAR or DEAR2.

The cost of an action rule r is introduced in this paper and is given by:

$$cost(r) = \sum\{\rho_{\mathcal{S}}(v_j, w_j) : 1 \leq j \leq p\}$$

where $\rho_{\mathcal{S}}(v_j, w_j)$ is the cost to modify value v_i in v_j. r is said to be feasible if $cost(r) < \rho_{\mathcal{S}}(d_1, d_2)$. Let r be an action rule:
$r:\ [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \cdots \wedge (b_p, v_p \rightarrow w_p)](x) \Longrightarrow [(d, d_1 \rightarrow d_2)](x)$

The cost of r might be high because of the high cost of one of its subterm, let $(b_j, v_j \rightarrow w_j)$ be this subterm. The authors suggest to look for a rule r_1 having the smallest cost value among the rules having this subterm in their right-hand sides.
$r_1:\ [(b_{j_1}, v_{j_1} \rightarrow w_{j_1}) \wedge (b_{j_2}, v_{j_2} \rightarrow w_{j_2}) \wedge \cdots \wedge (b_{j_q}, v_{j_q} \rightarrow w_{j_q})](y) \Longrightarrow [(b_j, v_j \rightarrow w_j)](y)$

In order to get another feasible rule with a lower cost than r, we can compose r with r_1 which replace the term $(b_j, v_j \rightarrow w_j)$ in r by the left-hand side of r_1:
$[(b_1, v_1 \rightarrow w_1) \wedge \ldots \wedge (b_{j_1}, v_{j_1} \rightarrow w_{j_1}) \wedge (b_{j_2}, v_{j_2} \rightarrow w_{j_2}) \wedge \cdots \wedge (b_{j_p}, v_{j_p} \rightarrow w_{j_p})$
$\wedge \cdots \wedge (b_p, v_p \rightarrow w_p)](x) \Longrightarrow [(d, d_1 \rightarrow d_2)](x)$

The exploitation of the notion of action *cost* to discover interesting action rules was further investigated in Tzacheva and Ras (2005) where a heuristic search algorithm for constructing feasible action rules that have high confidences and the lowest costs is proposed. This is achieved in this work by building dynamically a directed search graph $\mathcal{G}_\mathcal{S}$ and applying already discovered action rules to its nodes. Assuming that a set of action rules has been discovered and that we need to reclassify a set of objects. The initial node n_0 contains information provided by the user: the objects to reclassify from d_1 to d_2 and the current cost of their reclassification $\rho_\mathcal{S}(d_1, d_2)$.

Any other node n in $\mathcal{G}_\mathcal{S}$ shows an alternative way to achieve the same reclassification with a lower cost than all the nodes preceding n. Any node n is obtained by applying an action rule to it. For example, the rule $r : [(b_1, v_1 \rightarrow w_1) \wedge (b_2, v_2 \rightarrow w_2) \wedge \cdots \wedge (b_p, v_p \rightarrow w_p)](x) \Rightarrow [(d, d_1 \rightarrow d_2)](x)$ applied to the node $n_0 = \{[d_1 \rightarrow d_2, \rho_\mathcal{S}(d_1, d_2)]\}$ leads to the node $n_1 = \{[v_1 \rightarrow w_1, \rho_\mathcal{S}(v_1, w_1)], [v_2 \rightarrow w_2, \rho_\mathcal{S}(v_2, w_2)], \ldots, [v_p \rightarrow w_p, \rho_\mathcal{S}(v_p, w_p)]\}$.
An A^*-like heuristic algorithm named BUILD-AND-SEARCH is proposed in Tzacheva and Ras (2005) and aims at finding the shortest path from the root to the goal node in $\mathcal{G}_\mathcal{S}$ that achieve the lowest cost and the strongest reclassification. It has been implemented in a system called LOWERCOSTRECLASSIFIER and applied to three databases in medical and financial domain.

A new subclass of attributes called semi-stable is also introduced Ras and Tzacheva (2003); Tzacheva and Ras (2005). It involves the attributes that are functions of time as for instance *age* and *height*. Some semi-stable attributes can be treated as flexible attributes. The more we have flexible attributes the more is our chance to get cheaper action rules. This issue is discussed in Tzacheva and Ras (2005) and the detection of semi-stable attributes in a distributed information system is proposed in Ras and Tzacheva (2003).

While our approach and the framework presented in this paragraph share many points, we adopt in our method a quite different strategy: given a situation presented to the user, we construct actions with respect to this situation, whereas in Ras et al. work, to construct action rules, only couples of classification rules are considered. Action rules are already computed when a situation is presented. They are applied to the situation in order to get the lowest cost actions for classification.

Using classification systems to discover action knowledge, as we propose in the following section, is not new. Our approach which lies in the action-elaboration approach, works on a whole set of classification rules rather than considering rules separately. Comparisons between the methods presented in this section and our contribution are given throughout this paper and in Section 4.

3. Our Framework

We first describe in Section 3.1 the actionability task we want to address, then we discuss in 3.2 the need of a suitable distance metric in our framework. The search space of actions we explore in our approach is detailed in Section 3.3, DAKAR algorithm is given in 3.4. We present finally in Section 3.5 an application of DAKAR to the environmental domain related to pollution by pesticides.

3.1 The Learning task

We propose an approach for mining actionable knowledge. The task we address aims at improving a given situation with regard to a rule-based classifier. It can be defined informally as follows:

"Given a situation for which it corresponds an unsatisfactory class, what are the minimal changes (actions) to do in this situation in order to improve that situation to a satisfactory class?"

For instance, given a situation describing a high level pollution in an area, what are the advices to be suggested to reduce that pollution. In this paper, we propose to discover actions through a set of classification rules. Before giving the algorithm DAKAR, let us first introduce some basic definitions. In the following, let $X_1, .., X_n$ be features taking their values in the domains $Dom_1, .., Dom_n$ respectively. We denote an unsatisfactory class by $\ominus$ and a satisfactory class by $\oplus$. We also write $\mathcal{R}^{\oplus}$ to denote the set of classification rules restricted to the rules concluding on the class $\oplus$. Similarly, we write $\mathcal{R}^{\ominus}$ to denote $\mathcal{R}$ restricted to the rules concluding on the class $\ominus$. This can be easily extended to multi-class problems.

Definition 1 (Instance) *An instance is an object described by a conjunction of instantiated features denoted by:* $\bigwedge_{i=1,..,n} (X_i = v_i)$.

Definition 2 (Description) *A description is a conjunction defined on a subset of features as follows:*
$D = \bigwedge_{i=1,..,m} (X_{k_i} \in d_{k_i})$
where $d_{k_i} \subseteq Dom_{k_i}$, $\{k_1, ..., k_m\} \subseteq \{1, .., n\}$ *and* $k_i \neq k_j \forall i, j$.

Definition 3 (Extended description) *A description can be extended to all the features as follows:*
$\widehat{D} = \bigwedge_{i=1,..,m} (X_{k_i} \in d_{k_i}) \quad \bigwedge_{j \notin \{k_1,..,k_m\}} (X_j \in Dom_j)$
where $d_{k_i} \subseteq Dom_{k_i}$, $\{k_1, ..., k_m\} \subseteq \{1, .., n\}$.

In the rest of the paper, a description is considered in its extended form.

Definition 4 (Classification rule) *A classification rule is an implication of the form:*

$$Descr \Longrightarrow Class$$

where $Descr$ is a description and $Class$ is the corresponding class label ($\ominus$ or $\oplus$).

Definition 5 (Coverage) *We say that a description* $D = \bigwedge_{i=1,..,n} (X_i \in d_i)$ *covers an instance* $I = \bigwedge_{i=1,..,n} (X_i = v_i)$ *iff* $\forall i,\ v_i \in d_i$. *The set of instances (among all possible instances) covered by D will be denoted by cov(D).*

Note that cov(D) $=$ cov($\widehat{D}$). For a classification rule $R : Descr \Longrightarrow Class$, we define $cov(R) = cov(Descr)$.

Definition 6 (Outcome situation) *An action applied to a situation leads to another description called an outcome situation.*
Given a situation $S = \bigwedge_{i=1,..,n}(X_i \in s_i)$ *and an action* $A = \bigwedge_{i=1,..,n}(X_i \in a_i)$, *the outcome situation* $outcome(S, A)$ *is computed as follows:*

$$outcome(S, A) = \bigwedge_{i=1,..,n} (X_i \in o_i) \quad \textit{where } o_i = \begin{cases} s_i & \textit{if } a_i = Dom_i \\ a_i & \textit{otherwise} \end{cases}$$

Definition 7 (Situation and Action) *In our framework, a situation or an action is a description.*

Definition 8 (Task) *Given a set of rules $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$, an unsatisfactory situation $\mathcal{S}$ of class $\ominus$, find an action A that allows to move S to an outcome situation which belongs to the class $\oplus$.*

When proposing actions to the user, we have to take into consideration their practical applicability. This can be achieved by considering the *flexibility* of the features involved in the action. This notion is further explained in the following.

3.2 On the need of a suitable distance metric

In Ras and Tsay (2003), the notion of feature flexibility is used to find action rules. Features are divided into two groups: *stable* (features that cannot be changed) and *flexible*. In their approach, action rules are designed with flexible features.

This division into two groups is interesting but is rather strict and not always sufficient. For example, a physician would prefer, if possible, prescribing medicines to advising a surgical intervention to an ill patient. However, in their work, such two flexible features are identically considered. In our approach, to take into account this differentiation, each feature is assigned with a given *flexibility weight*. Moreover, the notion of feature flexibility is not sufficient. Here is an example:

Example 1 *Making a diet and exercising are in many cases the advices given by physicians to their overweighted patients. But the feasibility of these advices depends on the goal to achieve. If the weight to loose is about 80 kilograms, this weight-control option is impractical. The doctor may suggest other treatments for obesity such as weight-loss medicines or even surgery. On the other hand, loosing 2 kilograms is quite achievable through a diet and some physical activity.*

The outcome situation (when an action is applied) must be relatively "close" to the initial situation. For these reasons, a metric distance capturing the difficulty of improving a situation thanks to an action seems indispensable to our problem.

The literature abounds with definitions of metric distances between two descriptions. Relying on an empirical comparative study Malerba et al. (2001), we have chosen the generalized Minkowski metric proposed by Ichino and Yaguchi (1994) which handles both qualitative and quantitative features. It also integrates weights and deals with dissimilarities between two feature values. We rewrite it according to our notations. Let us consider two extended descriptions:

$$D = \bigwedge_{1..n} (X_i \in d_i) \qquad d_i \subseteq Dom_i$$

$$D' = \bigwedge_{1..n} (X_i \in d'_i) \qquad d'_i \subseteq Dom_i$$

Let us consider two sub-domains d_i and d'_i of Dom_i (domain of feature A_i), they define the two following operators (illustrated in Figure 1):

- The <u>Cartesian join</u> $\boxplus$ of d_i and d'_i is given by:

  - If the i^{th} feature is quantitative, $d_i = [l_i, u_i]$ and $d'_i = [l'_i, u'_i]$, we have:

$$d_i \boxplus d'_i = [min(l_i, l'_i), max(u_i, u'_i)]$$

- If the i^{th} feature is qualitative, d_i and d'_i are 2 sets and we have:

$$d_i \boxplus d'_i = d_i \cup d'_i$$

- The Cartesian meet $\boxtimes$ of d_i and d'_i is given by:

$$d_i \boxtimes d'_i = d_i \cap d'_i$$

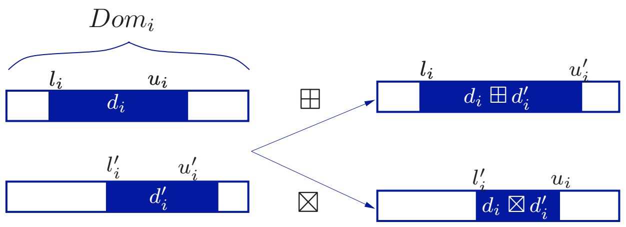

Figure 1: The operators $\boxplus$ and $\boxtimes$ for quantitative attributes.

Using these operators, the authors in Ichino and Yaguchi (1994) propose a dissimilarity measure between two feature sub-domains:

$$\phi(d_i, d'_i) = |d_i \boxplus d'_i| - |d_i \boxtimes d'_i| + \gamma(2|d_i \boxtimes d'_i| - |d_i| - |d'_i|)$$

where $i = 1..n$ and $\gamma \in [0, 0.5]$ controls the inner-side nearness and the outer-side nearness between d_i and d'_i. The cardinal of d_i (when A_i is a qualitative attribute) and the length of d_i (when d_i is a quantitative attribute) is noted $|d_i|$. When normalized, this measure is rewritten:

$$\psi(d_i, d'_i) = \frac{\phi(d_i, d'_i)}{|Dom_i|}, i \in \{1, .., n\}$$

so that $0 \leq \psi(d_i, d'_i) \leq 1$. Consider now two extended descriptions:

$$D = \bigwedge_{1..n} (A_i \in d_i) \qquad d_i \subseteq Dom_i$$

$$D' = \bigwedge_{1..n} (A_i \in d'_i) \qquad d'_i \subseteq Dom_i$$

The normalized and weighted dissimilarity measure called the generalized Minkowski distance of order p defined in Ichino and Yaguchi (1994) is given by:

$$d_p(D, D') = \left[\sum_{i=1}^{n} \{w_i \, \psi(d_i, d'_i)\}^p \right]^{1/p}$$

where the flexibility weights $w_i > 0, i \in \{1, .., n\}$ are chosen so that $\sum_{i=1}^{n} w_i = 1$ and $\psi(d_i, d'_i)$ is a normalized distance between the sub-domains d_i and d'_i of Dom_i (see Ichino and Yaguchi (1994) for more details). The generalized Minkowski distance satisfies $0 \leq d_p(D, D') \leq 1$ and it is proved that this distance satisfies all the axioms for a metric.

Finally, as pointed out in subsection 3.2, we favor actions that involve a little change in the initial situation. This is achieved by verifying that the distance between the outcome situation (obtained when an action is applied to the initial situation) and the initial situation itself does not exceed a threshold δ given by the user. We term such privileged actions δ-*cost* actions.

3.3 The search space of actions

In this subsection, we define the search space of actions. We are given an initial situation $S = \bigwedge_{i=1,..,n}(A_i \in s_i)$ classified in $\ominus$ and a set of rules $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$. An obvious approach is to define the search space of actions as the whole large space of descriptions. This space is first restricted thanks to the property of *validity* (Definition 9). The *consistency* property (Definition 11) of the actions constructed is checked and their *quality* values (Definition 12) are computed.

Definition 9 (The validity of an action relatively to a situation) *An action A is said to be valid relatively to a situation S if*

$$\text{cov}(S) \cap \text{cov}(A) = \{\}$$

Thus, an instance covered by the initial situation is not covered by the outcome situation when a valid action is applied. To consider only valid actions, we use the *maximally discriminant set* defined as follows:

Definition 10 *For a situation S and a description $D = \wedge_{i=1,..,n}(X_i \in d_i)$, the* maximally discriminant set *denoted by* discr *is defined by:*

$$discr(S, D) = \bigcup_{i=1}^{n} \bigcup_{\Delta \in diff(s_i,d_i)} \{X_i \in \Delta\}$$

The set diff(s_i, d_i) is computed according to the kind of the attribute A_i concerned. It is:

- *the set of elementary intervals of d_i with a maximal size, excluding s_i if A_i is a quantitative attribute.*
- *the set difference between d_i and s_i if A_i is a qualitative attribute.*

Example 2 *Let X_i be a qualitative attribute, a,b and c 3 values in Dom_i. If $d_i = \{a, b, c\}$ and $s_i = \{b\}$ then diff$(s_i, d_i) = \{\{a, c\}\}$.*
Let X_i be a quantitative attribute, x,y and z 3 values in Dom_i s.t. $x < y < z$. If $d_i = [x, z]$ and $s_i = [y]$ then diff$(s_i, d_i) = \{[x, y[,]y, z]\}$.

For a description D, an element of $discr(S, D)$ is a valid action w.r.t. the situation S. For a rule $R : Descr \implies Class$, we define $discr(S, R) = discr(S, Descr)$. Notice that an element in $discr(S, D)$ is a valid action relatively to S. The notion of maximally discriminant has already been used in Sebag (1996) in designing a learner inspired by the version spaces framework introduced in Mitchell (1982).

In order to consider descriptions that are likely to be "good" actions, we construct the maximally discriminant set between S and the description part of one rule in $\mathcal{R}^{\oplus}$. Intuitively, this set is composed of the attribute-value pairs that make the difference between the rules of class $\oplus$ and the situation of class $\ominus$.

Example 3 *Let \{ weight, medicines\} be a set of features taking their values in Dom$_1 = [40, 120]$ and Dom$_2$ = \{no, tablets, syrup\} respectively. The quantitative feature $weight$ represents the weight of a patient while the qualitative feature $medicines$ represents the treatment prescribed* *to the patient. Let $R_1^{\oplus}$, $R_2^{\oplus}$ and $R_3^{\ominus}$ be 3 rules:*

$R_1^{\oplus}$: $weight \in [50, 80] \rightarrow$ *not ill*
$R_2^{\oplus}$: $weight \in [90, 110] \wedge medicines \in \{syrup\} \rightarrow$ *not ill*
$R_3^{\ominus}$: $weight \in [65, 120] \wedge medicines \in \{no\} \rightarrow$ *ill*

We consider an ill patient in the following situation S: $weight \in [70] \wedge medicines \in \{no\}$.
We compute the maximally discriminant sets:

$discr(S, R_1^{\oplus}) = \{weight \in [50, 70[,\ weight \in]70, 80], medicines \in \{\textit{tablets,syrup}\}\}$
$discr(S, R_2^{\oplus}) = \{weight \in [90, 110], medicines \in \{\textit{syrup}\}\}$

In Ras and Wieczorkowska (2000), for each couple of rules in $\mathcal{R}^{\oplus}$ corresponds an action rule that guarantees the switch from $\ominus$ class to $\oplus$ class. An action rule relies only on one rule in $\mathcal{R}^{\oplus}$. In our point of view, since the rules are uncertain (not implications), an action should rely on the entire set of rules. Thus, by combining attribute-value pairs of these rules, new actions can be suggested.

The set of elementary actions $\mathcal{A}$ is given by

$$\mathcal{A} = \bigcup_{R \in \mathcal{R}^{\oplus}} discr(S, R)$$

and an action is a conjunction of such elements. Thus, our search space of actions is the set:

$$\{ \bigwedge_{\text{elem} \in E} \text{elem} | E \subseteq \mathcal{A}\}$$

Actions considered by our method embodies those that would be suggested by the system in Ras and Wieczorkowska (2000). Let us also notice that the search space as defined above embodies some actions that are not to be considered because they do not fulfill the *consistency* property defined as follows:

Definition 11 (The consistency of an action) *An action A is said to be consistent if* $\text{cov(A)} \neq \{\}$.

In practice, an action is not consistent if it involves an empty value for at least one attribute.

Example 4 *In Example 3, the set of elementary actions $\mathcal{A}$ is $\{weight \in [50, 70[, weight \in]70, 80], medicines \in \{\textit{tablets,syrup}\}, weight \in [90, 110], medicines \in \{\textit{syrup}\}\}$.*
An action is a conjunction composed by elements of a subset of $\mathcal{A}$. We have to discard the action $weight \in [50, 70[\wedge weight \in [90, 110]$ equivalent to $weight \in [\,]$ which is not consistent.

Let us emphasize that applying an action to a situation does not guarantee to get an outcome situation with a better class than the initial situation. That is why, we need a criterion for assessing the quality of actions. We classify the outcome situation using the set of rules. We rely on the confidence of the rules covering the outcome situation to evaluate the quality of an action.

Definition 12 (The quality of an action) *Let S be a situation, A an action and O the outcome situation $O = outcome(S, A)$, the quality of A is :*

$$\textit{quality}(A) = \sum_{R \in \mathcal{R}^{\oplus},\ cov(O) \subseteq cov(R)} \textit{conf}(R) \ - \sum_{R \in \mathcal{R}^{\ominus},\ cov(O) \subseteq cov(R)} \textit{conf}(R)$$

where $\textit{conf}(R)$ is the usual confidence[7] *of the rule R.*

7. The confidence of a rule $R : Descr \Longrightarrow C$ is the number of examples of class C covered by $Descr$ divided by the total number of examples covered by $Descr$.

Algorithm 1: Dakar algorithm

```
Input:
- a set of rules $\mathcal{R} = \mathcal{R}^{\oplus} \cup \mathcal{R}^{\ominus}$
- a situation $\mathcal{S}$
- a distance threshold $\delta$
- a maximal beam size $N$
Output: - a set of actions
1  $\mathcal{A} = \bigcup_{R \in \mathcal{R}^{\oplus}} discr(\mathcal{S}, R)$
2  $Beam = \{\}$
3  $NewBeam = \{true\}$
4  while $Beam \neq NewBeam$ do
5      $Beam \leftarrow NewBeam$
6      foreach $action \in Beam$ do
7          foreach $elem \in \mathcal{A}$ do
8              $newaction \leftarrow (action \wedge elem)$
9              if $newaction$ consistent and $\delta$-cost then
10                 $NewBeam \leftarrow NewBeam \cup \{newaction\}$
11                 while $card(Beam) > N$ do
12                     $worstaction \leftarrow Argmin_{A \in NewBeam} quality(A)$
13                     $NewBeam \leftarrow NewBeam - \{worstaction\}$
14 return sorted Beam
```

Notice that $quality(A) > 0$ means that O is covered by at least one rule in $\mathcal{R}^{\oplus}$.

Since we are relying on the whole set of classification rules to compute the set of actions and the quality of those actions, our method allows using an inconsistent set of rules as shown in the Example 5 below.

Example 5 *The set of rules given in Example 3 is inconsistent: both of the rules $R_1^{\oplus}$ and $R_3^{\ominus}$ cover the situation S. The action $A : weight \in]70, 80]$ belonging to the search space transform S into the situation $S' : weight \in]70, 80] \wedge medicines \in \{no\}$ which is covered by $R_1^{\oplus}$ and $R_3^{\ominus}$. We still can compute the quality of the action A by $quality(A) = conf(R_1^{\oplus}) - conf(R_3^{\ominus})$.*

3.4 DAKAR algorithm

The aim of DAKAR (Discovery of Actionable Knowledge and Recommendations) is to find the set of the *best* actions, *i.e.* those maximizing the quality criterion (Definition 12). DAKAR is given in Algorithm 1. The algorithm explores the search space of actions using a beam search strategy: it maintains a set (called a *beam*) of the best actions the algorithm has constructed up till now. Initially the beam is set to {true} (line 3), which means that no action is constructed. During the exploration of the search space, DAKAR specializes actions (line 8) of the beam and keeps only the best ones w.r.t. the criteria defined in sub sections 3.2 and 3.3 (consistency and distance in line 9 and quality in line 12). In a beam search, the size of the beam must not exceed a maximal size N, this is ensured by the lines 11-13 of the algorithm.

3.5 Experiments

We have implemented DAKAR in Sicstus Prolog and we have conducted an experimental evaluation of our algorithm on an environmental application related to stream-water pollution by pesticides. This application is developed in the context of the SACADEAU project Cordier (2005); Cordier et al. (2005, 2006). In our experiments, we used a dataset generated by a model which outputs a class of pollution given information about farming works, climate, soil, etc. The set of attributes and their descriptions is given in Table 1.

Name	Domain	Flexibility	Description
strat	{pre,post}	0.001	pesticide application strategy of the farmer
molec	{atrazine,new}	0.003	pesticide used by the farmer
hedge	{0%,90%}	0.006	percentage of river border with a hedge
basin	{concave,convex}	0.33	typology of the catchment area
orga_matter	{2%,5%}	0.33	soil composition in organic matter
climate	[1;5]	0.33	wetness of the climate (1: not wet)
class	{0,1,2,3,4}	-	severity of the pollution (0: no pollution)

Table 1: Some attributes of the SACADEAU application and their descriptions

It is a multi-class application where pollution classes are ordered by experts by taking into account legal thresholds (class 4 is the least satisfactory class). The model is an oracle since it provides the function $simulation : situation \rightarrow class$ and we define the *benefit* of an action A applied to a situation S, by $benefit(A) = simulation(S) - simulation(outcome(S, A))$. An action A is said *positive* when its benefit is positive.

Our system was tested on 150 unsatisfactory situations (of class 1, 2, 3 or 4). In the presented experiments, the size of the beam was 5; thus, DAKAR proposed 5 actions for each situation. We varied the distance threshold δ from 0 to 0.5. A set of rules, with minimal support 20, was generated by ICL Raedt and Laer (1995) (rules are not ordered).

A first evaluation is to show qualitatively the utility of using a distance for recommending actions. Let us give an example of DAKAR execution on a situation of class 4:

```
strategy=post, molecules=atrazine, hedge=0%,
basin=convex, orga_matter=5%, climate=4
```

The 2 best actions recommended by DAKAR, with benefit 2 and 1 respectively, are given below:

```
1 - hedge=90%, molec=new, strat=pre    2 - molec=new, strat=pre
  (quality=1.50 ; distance=0.005)       (quality=1.35 ; distance=0.002)
```

Both actions suggest to apply pesticides before plants grow up (pre-emergence strategy) and to use new molecules rather than atrazine. The actions have almost the same quality whereas the distance involved by the first action is more than two times the distance involved by the second. Concretely, experts could decide that in the short term, installing a hedge on 90% of the river border is not a necessary action for improving the situation.

A second qualitative evaluation concerns the interest of the quality criterion. Here is an example of situation of class 3:

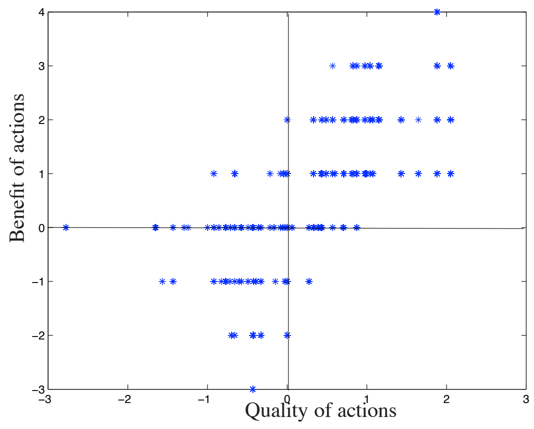

Figure 2: Actions plotted according to their quality and their effective benefit

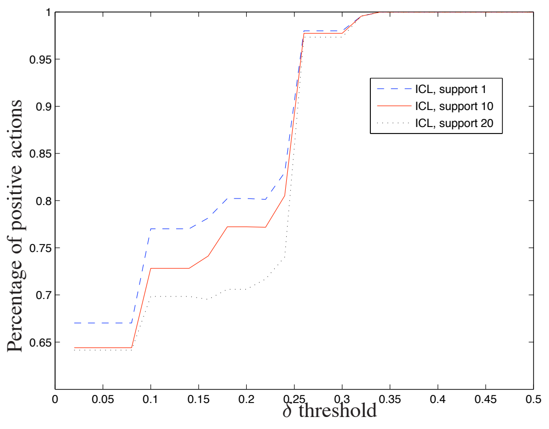

Figure 3: Percentage of positive recommended actions according to the size of the rule set and the distance parameter δ

```
strategy=pre, molecules=new, hedge=0%,
basin=convex, orga_matter=2%, climate=3
```

The 2 best actions recommended by our system (having a benefit of 1 and -1 respectively) are:

```
1 - hedge=90%                          2 - hedge=90%, strat=post
  (quality=0.64; distance=0.005)    (quality=-0.56; distance=0.005)
```

The first action is the only one, among the five recommended actions, to get a positive quality. Installing a hedge on 90% of the river border seems to be the necessary action for improving the situation.

From this point, only the action maximizing the quality criterion in the beam was considered. We plotted, in Figure 2, the recommended actions according to their quality and their effective benefit. Note that the quality of an action A is a good prediction of the efficiency of A if ($quality(A) < 0$ and $benefit(A) \leq 0$) or ($quality(A) > 0$ and $benefit(A) > 0$).

Our experiment raises another concern about the relationship between the redundancy in a set of classification rules and the efficiency of recommended actions. In Torgo (1993), the author studied the effect of the redundancy of a classification rule set on the task of predicting a class. In his experiments, a redundant classification rule set has better accuracy on classification (Figure 3).

Using ICL, three classification rule sets were learned with a support parameter equal to 1, 10 and 20, leading to set of 19, 35 and 63 rules respectively. We evaluated the three sets of rules, by comparing the efficiency of the actions they suggested when used in DAKAR. We plotted in Figure 3 the efficiency of recommended actions in function of the parameter δ for the three sets of rules. As expected, the efficiency of actions proposed by DAKAR is globally increasing in the parameter δ. Moreover, we notice that the more the classification rule set is redundant, the more the actions are effective. The impact of the redundancy in rules on Sebag's distance between two instances expressed by Horn clauses was also pointed out in Sebag (1997). Her distance is based on the coverage of instances by a theory and the more the theory is redundant, the more this distance is of interest.

We repeated the same experiment for three rule sets generated by three systems: C4.5 Quinlan (1993), ICL Raedt and Laer (1995) and Apriori [8] Agrawal et al. (1993). Supposing that Apriori is more redundant than ICL which is more redundant than C4.5, we compared the efficiency of the actions suggested by the three sets of rules. Remarkably, here again, the more the set of rules is redundant, the more the actions are effective.

4. Comparative Study

Table 2 summarizes and compares the main characteristics of DAKAR and the different approaches described in Section 2. In this table, the term approach refers either to an implemented system or to an algorithm. These approaches are compared according to:

- Knowledge extracted: The kind of actions discovered.
- Model/domain knowledge: The model on which the method relies to discover actionable knowledge and the domain knowledge used.
- Measures: The quality criteria used to discover and assess actionable knowledge.
- Method: A brief description of the learning strategy of the approach detailed in Section 2.

5. Discussion and Future Work

In our algorithm, the possible new actions are chosen if they are consistent and involve a little change in the initial situation (delta-cost actions). Then they are sorted under their quality and the n best actions are proposed if the beam cardinality is n. We could use a multicriterion choice for the new actions. We can imagine an algorithm using both quality and distance in the criterion as done in Ling et al. (2002) where a function of profit is maximized. The aim of profit-based optimization approach is to provide the user with the best-profit actions. The problem is to define an optimization function to maximize combining quality and distance. This task is complex and moreover it is also interesting to propose a list of feasible/possible actions with their computed quality and let the user operate the ultimate choice.

The metric distance, capturing the difficulty of improving a situation thanks to an action, is very important in our method. It relies on flexibility weights which take into account differentiation of features (more or less flexible). The most important difficulty for the user is to choose these weights. How to determine if changing pesticide application strategy is more or less flexible than changing pesticide molecule used? The expert is here essential and the choice may be subtle.

Another point is that a metric distance is a symmetric notion. For example, increasing or decreasing a quantitative attribute with the same value leads to the same distance with the initial value. In our application there is no problem with this but generally, we can imagine some non-symmetric attributes. In medical care applications, the attribute “age” can only grow. An action implying smaller age is no-sense. An adaptation for such attributes is then needed as in Tzacheva and Ras (2005).

8. We constrained the association rules generated by Apriori to contain only the class label in their right-hand side

	Approach	Knowledge extracted	Model/domain knowledge	Measures	Method
Predefined actions	KEFIR [Piatetsky-Shapiro and Matheus (1994)]	corrective actions	corrective actions	deviation, impact of deviation, interestingness	Compute deviations according to the norms, order them using interestingness/impact measure, present key deviations, explanations and corrective actions in a written report.
	[Adomavicius and Tuzhilin (1997)]	actionable patterns	actionable patterns	-	Build an action tree, assign an actionable pattern to each node. Traverse the tree in order to get an insight about decisions to undertake.
	[Elovici and Braha (2003)]	actions	any classification model, actions	payoff	Use a classification system to classify an individual and then chose the action to do so as to maximize the expected payoff.
	[Jiang et al. (2005)]	actionsets	influence matrix of actions on features	utility, actionability	Compute the new utility of an actionset on a population P using the utility on a role model of P. Partition data so as to maximize the actionability w.r.t. utility.
Actionable rules identification	SUBGROUP DISCOVERY [Lavrač et al. (2004)]	descriptive rules	-	coverage, weighted relative accuracy	Find rules conditions $\rightarrow$ target-class. Expert-guided subgroup discovery in TP/FP space to search the rules with a maximal q value computed using a user TP/FP-trade off function.
	FINDNA [Liu et al. (2001)]	association rules	association rules with a class-value on their right-hand sides	χ^2, coverage, support	Discard the more general rules using their specialized rules with higher quality. Before removing any general rule, check that it is not still a strong rule after removing the tuples covered by the specialized rules from the dataset.
	OPPORTUNITY MAP [Zhao et al. (2005b,a)]	-	associative classification rules (CBA)	-	Interactive visualization of attributes, classes, data and rules in a matrix. Drill-down visualization of cells.
Elaboration of actions	DAR AND DEAR [Ras and Wieczorkowska (2000)]	action rules	decision rules, stable and flexible features	coverage	Consider all pairs of decision rules with contradicting conclusions and same values for all stable attributes to generate action rules.
	DEAR2 [Tsay and Ras (2005)]	action rules	decision rules, stable and flexible features	coverage	Organize decision rules into equivalence subclasses. Consider only pairs of rules belonging to some equivalence subclasses to construct extended action rules.
	PROACTIVE SOLUTION [Ling et al. (2002)]	action rules	decision trees	net profit	Build a decision tree modeling customers profiles. If a customer falls in an undesired leaf, search for the optimal actions that move the customer to another leaf.
	[Yang and Cheng (2002)]	actions	case bases	cost, rank	Mining case bases to identify representative role models. Three approaches: instance-based, cluster-based and SVM-based. Switching plan generation where negative cases are switched to positive cases using actions suggested by role models.
	DAKAR [Trepos et al. (2005)]	actions	classification rules, weights of flexible features	coverage, quality, distance, benefit	Given an unsatisfactory situation and relying on a set of classification rules, discover a set of action recommendations that propose minimal changes to improve that situation.

Table 2: Characteristics of the state-of-the-art methods for learning actionable knowledge

In our application, flexibility weights were fixed in function of the amount of work to do in order to modify a value. But they could be fixed in function of the required time for modifying this value. In next future, we propose to build a user-friendly interface which allows the user to change easily flexibility weights.

A first issue we plan to address in future work is to extend our work to propose not only promising actions but also unwise actions with respect to a situation. Let us look closer at the actions computed in the example given before:

```
1 - hedge=90%                          2 - hedge=90%, strat=post
  (quality=0.64; distance=0.005)    (quality=-0.56; distance=0.005)
```

It is clear that the second best classified action has a bad quality and that the only one which should be recommended is the first one, *i.e.* make an hedge grow. However, the second one brings information: if this hedge planting action is combined with changing the pre-emergence strategy for a post-emergence strategy, the quality becomes low. The good recommendation is then: plant an hedge but without changing your weeding strategy. Our intention is thus to collect both good and bad quality rules and to combine them in order to suggest more sophisticated recommendations including what not to do ones.

A second issue we plan to address is a more intensive use of the simulator. The simulator is currently used to provide examples to the learning step in order to get classification rules. It is also used in the validation step to check whether the recommended actions are or not beneficial as explained in sub section 3.5. An extension of our work would be to use the simulator during the search step to evaluate the quality of a rule. In our current work, we evaluate the quality of a rule by using the classification rules on the outcome situation (see definition 12). Our plan is to use the simulator to assess the quality of an action during the search step. Another view could be to incorporate the recommendation step into an interactive process where the user requires help by describing a situation, gets recommendation rules and can test them by using the simulator.

6. Conclusion

We investigate the task of learning actionable knowledge and recommendations. We first propose a survey of the main approaches addressing actionability. We also suggest a classification of these approaches into 3 classes. There are approaches that map extracted knowledge (e.g. deviations) to predefined actions, others identify actionable rules among a large set of rules and finally, the approaches that rely on models to create new actions.

In the second part of this paper, we attempt to answer the following question: how to make classification rules actionable and go beyond their restricted use in prediction? We propose a new framework to address the actionability task which can be described as follows: given a situation, the algorithm we propose allows the user to further exploit a set of classification rules in order to decide what are the actions to accomplish in order to improve that situation. The algorithm looks for the best actions involving a little change in the initial situation. This is achieved thanks to an actionability approach relying on a *distance*.

Our framework has been applied to an environmental dataset related to pollution. We have learned some actionable knowledge concerning the possible recommendations one can adopt in order to reduce pollution. Such recommendations take into account the degree of *flexibility* of each feature. Experiments have shown the feasibility and usefulness of this task. They have also raised some concerns about the impact of the redundancy of rules on the quality of actions recommended by our system.

The approach we propose can be improved in the following directions. First, extend the framework to first order learners. In this case, we need other kinds of distance metrics handling literals, such as those proposed in Hutchinson (1997); Ramon et al. (1998). Second, study the impact of the chosen distance metric on the quality of the recommended actions.

Action recommendation is a promising issue in Machine Learning and Data Mining with many practical applications such as health-care, environmental protection and customer analysis. There is clearly much research to be done in the formalization of the task of learning useful and actionable knowledge from both methods and interestingness measures points of view.

References

G. Adomavicius and A. Tuzhilin. Discovery of actionable patterns in databases: The action hierarchy approach. In *KDD*, pages 111–114, 1997.

R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD*, pages 207–216. ACM Press, 1993.

N. Ahituv and B. Ronen. Orthogonal Information Structures: A Model to Evaluate the Information provided by a Second Opinion. *Decision Sciences*, 19(2):255–268, 1988.

D. Blackwell. Comparison of experiments. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 93–102, 1951.

S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 265–276. ACM Press, 1997. URL `citeseer.ist.psu.edu/brin97beyond.html`.

M.-O. Cordier. SACADEAU: A decision-aid system to improve stream-water quality. *ERCIM News. Special issue on Environmental Modelling*, (61):35–36, April 2005.

M.-O. Cordier, F. Garcia, C.Gascuel-Odoux, V. Masson, J. Salmon-Monviola, F. Tortrat, and R. Trepos. A machine learning approach for evaluating the impact of land use and management practices on streamwater pollution by pesticides. In *Proc. of MODSIM 2005*, Melbourne, Australia, 12-15 December 2005.

M.-O. Cordier, V. Masson, A. Salleb, R. Trepos, C. Gascuel, and F. Garcia. Sacadeau project : recommending actions from simulation results. *Poster at BESAI 2006, 5th Workshop on Binding Environmental Sciences and Artificial Intelligence in conjunction with the ECAI*, 2006.

B. Duval, A. Salleb, and C. Vrain. *On the Discovery of Exception Rules: A Survey*, pages 77–98. In Quality Measures in Data Mining book. Springer, Studies in Computational Intelligence Series, 2007.

Y. Elovici and D. Braha. A decision-theoretic approach to data mining. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 33(1):42–51, 2003.

D. Gamberger and N. Lavrač. Generating actionable knowledge by expert-guided subgroup discovery. In *PKDD'02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 163–174. Springer-Verlag, 2002. ISBN 3-540-44037-2.

J. W. Grzymala-Busse. LERS–A system for learning from examples based on rough sets. *Intelligent decision support. Handbook of applications and advances of the rough set theory*, pages 3–18, 1992.

Z. He, X. Xu, and S. Deng. Data Mining for Actionable Knowledge: A Survey. *ArXiv Computer Science e-prints*, January 2005.

A. Hutchinson. Metrics on terms and clauses. In *ECML*, pages 138–145, 1997.

M. Ichino and H. Yaguchi. Generalized minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):698–708, 1994.

T. Imielinski, A. Virmani, and A. Abdulghani. DataMine: Application Programming Interface and Query Language for Database Mining. In *KDD*, pages 256–262, 1996.

Y. Jiang, K. Wang, A. Tuzhilin, and A. W.-C. Fu. Mining patterns that respond to actions. In *ICDM*, pages 669–672, 2005.

M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In Nabil R. Adam, Bharat K. Bhargava, and Yelena Yesha, editors, *Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407. ACM Press, 1994. URL `citeseer.ist.psu.edu/klemettinen94finding.html`.

N . Lavrač, D. Gamberger, and P. Flach. Subgroup discovery for actionable knowledge generation: deficiencies of classification rule learning and the lesson learned. In N. Lavrac, T. Fawcett, and H. Motoda, editors, *ICML 2002 workshop on Data Mining Lessons Learned*, 2002.

N. Lavrač, B. Cestnik, D. Gamberger, and P. Flach. Decision support through subgroup discovery: Three case studies and the lessons learned. *Machine Learning*, 57(1-2):115–143, 2004. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/B:MACH.0000035474.48771.cd.

C. X. Ling, T. Chen, Q. Yang, and J. Cheng. Mining optimal actions for profitable CRM. In *ICDM*, pages 767–770, 2002.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD*, pages 80–86, 1998.

B. Liu, W. Hsu, and Y. Ma. Identifying non-actionable association rules. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–334, New York, NY, USA, 2001. ACM Press. ISBN 1-58113-391-X. doi: http://doi.acm.org/10.1145/502512.502560.

D. Malerba, F. Esposito, V. Gioviale, and V. Tamma. Comparing dissimilarity measures in symbolic data analysis. In *Joint Conferences on "New Techniques and Technologies for Statistcs" and "Exchange of Technology and Know-how"(ETK-NTTS'01)*, pages 473–481, 2001.

T. M. Mitchell. Generalization as search. *Artif. Intell.*, 18(2):203–226, 1982.

Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1992. ISBN 0792314727.

G. Piatetsky-Shapiro and C. Matheus. The interestingness of deviations. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 25–36, Menlo Park, CA, 1994. AAAI Press.

J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

L. De Raedt and W. Van Laer. Inductive constraint logic. In *ALT '95: Proceedings of the 6th International Conference on Algorithmic Learning Theory*, pages 80–94, London, UK, 1995. Springer-Verlag. ISBN 3-540-60454-5.

J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. In *CompulogNet Area Meeting on Computational Logic and Machine Learing*, pages 35–41. University of Manchester, UK, May 1998.

Z. W. Ras and S. Gupta. Global action rules in distributed knowledge systems. *Fundam. Inf.*, 51(1): 175–184, 2002. ISSN 0169-2968.

Z. W. Ras and L.-S. Tsay. Discovering extended action-rules (system dear). In *IIS*, pages 293–300, 2003.

Z. W. Ras and A. A. Tzacheva. Discovering semantic inconsistencies to improve action rules mining. In *IIS*, pages 301–310, 2003.

Z. W. Ras and A. Wieczorkowska. Action-rules: How to increase profit of a company. In *PKDD*, pages 587–592, 2000.

Z. W. Ras, A. A. Tzacheva, L.-S. Tsay, and O. Gurdal. Mining for Interesting Action Rules. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05)*, pages 187–193. IEEE, 2005.

M. Sebag. Delaying the choice of bias: A disjunctive version space approach. In *ICML*, pages 444–452, 1996.

M. Sebag. Distance induction in first order logic. In *ILP '97: Proceedings of the 7th International Workshop on Inductive Logic Programming*, pages 264–272. Springer-Verlag, 1997. ISBN 3-540-63514-9.

A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Trans. On Knowledge And Data Engineering*, 8:970–974, 1996. URL `citeseer.ist.psu.edu/silberschatz96what.html`.

L. Torgo. Controlled redundancy in incremental rule learning. In *ECML*, pages 185–195, 1993.

R. Trepos, A. Salleb, M.-O. Cordier, V. Masson, and C. Gascuel. A distance-based approach for action recommendation. In *ECML 2005, 16th European Conference on Machine Learning*, LNAI 3720, pages 425–436. Springer-Verlag Berlin Heidelberg, 2005.

L.-S. Tsay and Z. W. Ras. Action rules discovery: system DEAR2, method and experiments. *Journal of Experimental & Theoretical Artificial Intelligence*, 17(1-2):119–128, 2005.

A. A. Tzacheva and Z. W. Ras. Action Rules Mining. *Special Issue on knowledge discovery, Z.W. Ras (ed.) International Journal of Intelligent Systems*, 20(7):719–736, 2005.

Q. Yang and H. Cheng. Mining case bases for action recommendation. In *ICDM*, pages 522–529, 2002.

Q. Yang, J. Yin, C. X. Ling, and T. Chen. Postprocessing decision trees to extract actionable knowledge. In *ICDM*, pages 685–688, 2003.

K. Zhao, B. Liu, T. M. Tirpak, and W. Xiao. Opportunity Map: A Visualization Framework for Fast Identification of Actionable Knowledge. In *Proceedings of the ACM Fourteenth Conference on Information and Knowledge Management CIKM'05*, 2005a.

K. Zhao, B. Liu, T. M. Tirpak, and W. Xiao. A Visual Mining Frmework for Convenient Identification of Useful Knowledge. In *Proceedings Fifth IEEE International Conference on Data Mining ICDM'05*, 2005b.