An Evolutionary Algorithm Using Multivariate Discretization for Decision Rule Induction

Wojciech Kwedlo and Marek Krętowski

Institute of Computer Science, Technical University of Białystok, Poland
{wkwedlo, mkret}@ii.pb.bialystok.pl

Abstract. We describe EDRL-MD, an evolutionary algorithm-based system, for learning decision rules from databases. The main novelty of our approach lies in dealing with continuous - valued attributes. Most of decision rule learners use univariate discretization methods, which search for threshold values for one attribute at the same time. In contrast to them, EDRL-MD simultaneously searches for threshold values for all continuous-valued attributes, when inducing decision rules. We call this approach *multivariate discretization*. Since multivariate discretization is able to capture interdependencies between attributes it may improve the accuracy of obtained rules. The evolutionary algorithm uses problem specific operators and variable-length chromosomes, which allows it to search for complete rulesets rather than single rules. The preliminary results of the experiments on some real-life datasets are presented.

1 Introduction

Discovery of decision rules from databases is one of the most important problems in machine learning and data mining [6]. If a dataset contains some numerical (continuous-valued) attributes a decision rule learner must search for threshold values to create conditions (selectors) concerning these attributes. This process is called *discretization* and has attracted a lot of attention in the literature.

The simplest discretization algorithm called *equal interval binning* partitions the range of a continuous-valued attribute into several equal sized intervals. Since it does not make use of class labels it belongs to the group of *unsupervised* methods. Many experimental studies show [4], that *supervised* methods which use the class membership of examples, perform much better than their unsupervised counterparts.

The supervised discretization can be performed *globally* before the induction of rules by dividing the range of each continuous-valued attribute into intervals independent of the other attributes. An alternative approach, called *local* supervised discretization consists in searching for attribute thresholds during the generation of inductive hypothesis. However, even the systems, which use this method (e.g. C4.5 [10]) are often unable to search at the same time for threshold values for more than one attribute. Both approaches belong to the group of *univariate* discretization methods.

In the paper we propose a new system called EDRL-MD (EDRL-MD, for Evolutionary Decision Rule Learner with Multivariate Discretization) combining the two steps: the simultaneous search for threshold values for *all* continuous-valued attributes, which we call *multivariate discretization*, and the discovery of decision rules. As a search heuristic we use an *evolutionary algorithm* (EA) [9]. EAs are stochastic techniques, which have been inspired by the process of biological evolution. The success of EAs is attributed to the ability to avoid local optima, which is their main advantage over greedy search methods. Several systems, which employ EAs for learning decision rules (e.g. GABIL [3], GIL [7], EDRL [8]) were proposed. According to our knowledge all of them either work only with nominal attributes or discretize continuous-valued ones prior to induction of rules using univariate methods.

2 The Weakness of the Univariate Discretization

The univariate discretization methods, although computationally effective, are not able to capture interdependencies between attributes. For that reason they run the risk of missing information necessary for correct classification. A following example shows the shortcomings of univariate discretization.

Consider an artificial dataset shown (a similar idea was presented in [2]) on Fig. 1. Every example is described by two attributes A_1 and A_2 distributed uniformly on the interval $[0, 1]$. The examples are divided approximately equally into two classes denoted by $+$ and $\Box$. The optimal decision rules for this dataset are:

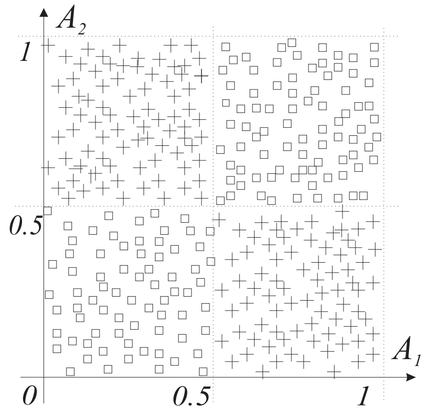

Fig. 1. An artificial dataset, for which the univariate approach is unlikely to find the proper thresholds.

$$(A_1 < 0.5) \wedge (A_2 < 0.5) \rightarrow \Box \qquad (A_1 < 0.5) \wedge (A_2 > 0.5) \rightarrow +$$
$$(A_1 > 0.5) \wedge (A_2 > 0.5) \rightarrow \Box \qquad (A_1 > 0.5) \wedge (A_2 < 0.5) \rightarrow +$$

Note that each of the conditions $A_i < th$ or $A_i > th$, where $th \in (0, 1)$ splits the examples into two subsets having roughly the same class distribution as the whole dataset. Hence a single condition related to one attribute A_1 or A_2 does not improve the separation of the classes. Let $H(A_i, th)$ be the *class information entropy* of the partition induced by the threshold th, a measure commonly used [4, 5] by supervised discretization algorithms. We can say that for each $A_i \in \{A_1, A_2\}$ and $th_1, th_2 \in (0, 1)$ $H(A_i, th_1) \cong H(A_i, th_2)$. This property holds for the other functions based on impurity or separation of the classes. Therefore any supervised univariate discretization algorithm will have difficulties with finding the proper thresholds for both attributes. This limitation does not apply to multivariate methods.

The above-mentioned example indicates, that in some cases a multivariate discretization is more appropriate and it leads to more accurate rules.

3 Description of the Method

We assume that a learning set $E = \{e_1, e_2, \ldots, e_M\}$ consists of M examples. Each example $e \in E$ is described by N attributes (features) $A_1(e), A_2(e), \ldots, A_N(e)$

and labeled by a class $c(e) \in C$. The domain of a nominal (discrete-valued) attribute A_j is a finite set $V(A_j)$, while the domain of a continuous-valued attribute A_i is an interval $V(A_i) = [a_i, b_i]$. For each class $c_k \in C$ by $E^+(c_k) = \{e \in E : c(e) = c_k\}$ we denote the set of *positive examples* and by $E^-(c_k) = E - E^+(c_k)$ the set of *negative examples*. A *decision rule* R takes the form $t_1 \wedge t_2 \wedge \ldots \wedge t_r \rightarrow c_k$, where $c_k \in C$ and the left-hand side (LHS) is a conjunction of $r (r \leq N)$ conditions $t_1, t_2, \ldots, t_r$; each of them concerns one attribute. The right-hand side (RHS) of the rule determines class membership of an example. A ruleset RS^{c_k} for a class c_k is defined as a disjunction of $K(c_k)$ decision rules $R_1^{c_k} \vee R_2^{c_k} \vee \cdots \vee R_{K(c_k)}^{c_k}$, provided that all the rules have c_k on the RHS.

In EDRL-MD the EA is called separately for each class $c_k \in C$ to find the ruleset RS^{c_k}. The search criterion, in terminology of EAs called the *fitness function* prefers rulesets consisting of few conditions, which cover many positive examples and very few negative ones.

3.1 Representation

The EA processes a population of candidate solutions to the search problem called *chromosomes*. In our case a single chromosome encodes a ruleset RS^{c_k}. Since the number of rules in the optimal ruleset for a given class is not known, we use variable-length chromosomes and provide the search operators, which change the number of rules. A chromosome representing the ruleset is a concatenation of *strings*. Each fixed-length string represents the LHS of one decision rule. Because the EA is called to find a ruleset for the given class c_k there is no need for encoding the RHS.

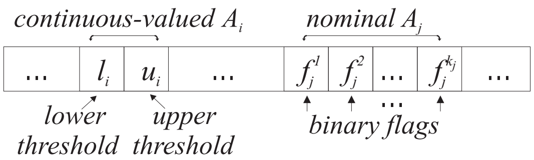

Fig. 2. The string encoding the LHS of a decision rule ($k_j = |V(A_j)|$). The chromosome representing the ruleset is concatenation of strings.

The string is composed (Fig. 2) of N *substrings*. Each substring encodes a condition related to one attribute. The LHS is the conjunction of these conditions. In case of a continuous-valued attribute A_i the substring encodes the lower l_i and the upper u_i threshold of the condition $l_i < A_i \leq u_i$. It is possible that $l_i = -\infty$ or $u_i = +\infty$.

Both l_i and u_i are selected from the finite set of all *boundary thresholds*. A boundary threshold for the attribute A_i is defined (Fig. 3) as a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of A_i, that one of the examples is positive and the other is negative. Fayyad and Irani proved [5], that evaluating only the boundary thresholds is sufficient for finding the maximum of class information entropy. This property also holds for the fitness function (1).

For a nominal attribute A_j the substring consists of binary flags; each of them corresponds to one value of the attribute. If e.g. the domain of attribute A_j is $\{low, moderate, high\}$ then the pattern 011 represents condition $A_j = (moderate \vee high)$, which stands for: "the value of A_j equals *moderate* or *high*".

Note, that it is possible, that a condition related to an attribute is not present on the LHS. For a continuous-valued attribute A_i it can be achieved by setting both $l_i = -\infty$ and $u_i = +\infty$. For a nominal A_j it is necessary to set all the flags $f_j^1, f_j^2, \ldots, f_j^{|V(A_j)|}$.

Each chromosome in the population is initialized using a randomly chosen positive example. The initial chromosome represents the ruleset consisting of a single rule, which covers the example.

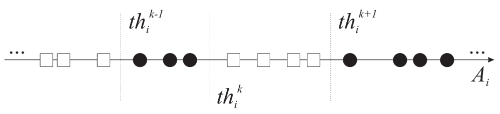

Fig. 3. An example illustrating the notion of boundary threshold. The boundary thresholds $th_i^1, \ldots, th_i^k, \ldots, th_i^{NT_i}$ for the continuous-valued attribute A_i are placed between groups of negative (•) and positive (□) examples.

3.2 The Fitness Function

Consider a ruleset RS^{c_k}, which covers *pos* positive examples and *neg* negative ones. The fitness function is given by:

$$f(RS^{c_k}) = \frac{pos - neg}{log_{10}(L + \alpha) + \beta}. \tag{1}$$

where $\alpha = 10$, $\beta = 10$, L is total the number of conditions in the ruleset RS^{c_k}. Note, that maximization of the numerator of (1) is equivalent to maximization of the probability of correct classification of an example. The denominator of (1) is a measure of complexity of the ruleset. An increase of the complexity results in a reduction of the fitness and thus prevents *overfitting*. To avoid overfitting we also limit the number of rules in a chromosome to max_R, where max_R is a user-supplied parameter.

The formula for function (1) including the values of the parameters α and β was chosen on the experimental basis. We found it performed well in comparison with other functions we tested.

3.3 Genetic Operators

Our system employs six search operators. Four of them: *changing condition*, *positive example insertion*, *negative example removal*, *rule drop* are applied to a single ruleset RS^{c_k} (represented by chromosome). The other two: *crossover* and *rule copy* require two arguments $RS_1^{c_k}$ and $RS_2^{c_k}$.

A similar approach was proposed by Janikow. However, his GIL [7] system is not able to handle continuous-valued attributes directly, since it represents a condition as a sequence of binary flags corresponding to the values of an attribute (we use the same representation for nominal attributes).

The changing condition is a mutation-like operator, which alters a single condition related to an attribute. For a nominal attribute A_j a flag randomly chosen from $f_j^1, f_j^2, \ldots, f_j^{|V(A_j)|}$ is flipped. For a continuous-valued A_i a threshold (l_i or u_i) is replaced by a random boundary threshold.

The positive example insertion operator modifies a single decision rule R^{c_k} in the ruleset RS^{c_k} to allow it to cover a new random positive example $e^+ \in E^+(c_k)$, currently uncovered by R^{c_k}. All conditions in the rule, which conflict with e^+ have to be altered. In case of a condition related to a nominal attribute A_j the flag, which corresponds to $A_j(e^+)$, is set. If a condition $l_i < A_i \leq u_i$ concerning continuous-valued attribute A_i is not satisfied because $u_i < A_i(e^+)$ the threshold u_i is replaced by $\hat{u}_i$, where $\hat{u}_i$ is the smallest boundary threshold such that $\hat{u}_i \geq A_i(e^+)$. The case when $A_i(e^+) \leq l_i$ is handled in a similar way.

The negative example removal operator alters a single rule R^{c_k} from the ruleset RS^{c_k}. It selects at random a negative example e^- from the set of all the negative examples covered by R^{c_k}. Then it alters a random condition in R in such way, that the modified rule does not cover e^-. If the chosen condition concerns a nominal attribute A_j the flag which corresponds to $A_j(e^-)$ is cleared. Otherwise the condition $l_i < A_i \leq u_i$ concerning continuous-valued A_i is narrowed down either to $\hat{l}_i < A_i \leq u_i$ or to $l_i < A_i \leq \hat{u}_i$, where $\hat{l}_i$ is the smallest boundary threshold such that $A_i(e^-) \leq \hat{l}_i$ and $\hat{u}_i$ is the largest one such that $\hat{u}_i < A_i(e^-)$.

Rule drop and rule copy operators [7] are the only ones capable of changing the number of rules in a ruleset. The single argument rule drop removes a random rule from a ruleset RS^{c_k}. The two argument rule copy adds to one of its arguments $RS_1^{c_k}$ a copy of a rule selected at random from $RS_2^{c_k}$, provided that the number of rules in $RS_1^{c_k}$ is lower than max_R.

The crossover operator selects at random two rules $R_1^{c_k}$ and $R_2^{c_k}$ from the respective arguments $RS_1^{c_k}$ and $RS_2^{c_k}$. It then applies an uniform crossover [9] to the strings representing $R_1^{c_k}$ and $R_2^{c_k}$.

4 Experiments

In this section some initial experimental results are presented. We have tested EDRL - MD on several datasets from UCI repository [1]. Table 1 shows the classification accuracy obtained by our method and C4.5 (Rel. 8) [10] algorithm. The accuracy was estimated by running ten times the complete ten-fold cross-validation. The mean of ten runs and the standard deviation are given. In all the experiments involving C4.5 decision rules were obtained from decision trees by C4.5rules program.

Dataset	C4.5	EDRL-MD
australian	84.8 ± 0.9	84.5 ± 0.5
bupa	66.5 ± 2.5	65.6 ± 1.5
breast-w	95.2 ± 0.4	95.2 ± 0.3
glass	67.5 ± 1.6	70.7 ± 2.9
hepatitis	80.6 ± 2.2	83.0 ± 2.4
iris	95.3 ± 0.7	95.4 ± 0.7
pima	74.2 ± 1.1	74.5 ± 0.6
wine	94.2 ± 1.4	93.6 ± 1.2

Table 1. The results of the experiments.

5 Conclusions

We have presented EDRL-MD, an EA-based system for decision rule learning, which uses a novel multivariate discretization method. The preliminary experimental results indicate, that both classification accuracy and complexity of discovered rules are comparable with the results obtained by C4.5.

Several directions of future research exist. One of them is the design of a better fitness function, which has a critical influence on the performance of the algorithm. The current version was chosen on the basis of very few experiments. Hence the classification results presented in the paper should be viewed as the lower limits of the attainable performance. We believe that the performance can be further improved.

It is a well-known fact, that many applications of KDD require the capability of efficient processing large databases. In such cases algorithms, which offer very good classification accuracy at the cost of high computational complexity cannot be applied. Fortunately, EAs are well suited for parallel architectures. We plan to develop a parallel implementation of EDRL-MD, which will be able to extract decision rules from large datasets.

Acknowledgments The authors are grateful to Prof. Leon Bobrowski for his support and useful comments. This work was supported by the grant W/II/1/97 from Technical University of Bialystok.

References

1. Blake, C., Keogh, E., Merz, C.J.: *UCI repository of machine learning databases*, available on-line: http://www.ics.uci.edu/~mlearn/MLRepository.html (1998).
2. Bobrowski, L.: Piecewise-linear classifiers, formal neurons and separability of the learning sets. *Proc. of 13th Int. Conf. on Pattern Recognition ICPR'96*. IEEE Computer Society Press (1996) 224-228.
3. De Jong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithm for concept learning. *Machine Learning* 13 (1993) 168-182.
4. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: *Machine Learning: Proc of* 12th *Int. Conference*. Morgan Kaufmann (1995) 194-202.
5. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of IJCAI'93*. Morgan Kaufmann (1993) 1022-1027.
6. Fayyad, U.M, Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): *Advances in Knowledge Discovery and Data Mining*. AAAI Press (1996).
7. Janikow, C.Z.: A knowledge intensive genetic algorithm for supervised learning. *Machine Learning* 13 (1993) 192-228.
8. Kwedlo, W., Krętowski, M.: Discovery of decision rules from databases: an evolutionary approach. In *Principles of Data Mining and Knowledge Discovery*. 2nd *European Symposium PKDD'98*. Springer LNCS 1510 (1998) 370-378.
9. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Springer (1996).
10. Quinlan, J.R.: Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research* 4 (1996) 77-90.