An evolutionary algorithm for cost-sensitive decision rule learning

Wojciech Kwedlo and Marek Krętowski

Institute of Computer Science, Technical University of Białystok
Wiejska 45a, 15-351 Białystok, Poland
e-mail: {wkwedlo, mkret}@ii.pb.bialystok.pl
http://aragorn.pb.bialystok.pl/~{wkwedlo, mkret}

Abstract. Most of classification learning methods aim at the reduction of the number of errors. However, in many real-life applications it is misclassification cost, which should be minimized. In the paper we propose a new method for cost-sensitive learning of decision rules from datasets. Our approach consists in modifying the existing system EDRL-MD (Evolutionary Decision Rule Learner with Multivariate Discretization). EDRL-MD learns decision rules using an evolutionary algorithm (EA). We propose a new fitness function, which allows the algorithm to minimize misclassification cost rather than the number of classification errors. The remaining components of EA i.e., the representation of solutions and the genetic search operators are not changed. The performance of our method is compared to that of C5.0 learning system. The results show, that the modified EDRL-MD is able to effectively process datasets with non-equal error costs.

1 Introduction

Classification is one of the most important tasks in machine learning and data mining. There exist many effective methods for building classifiers [16], but in most cases the goal is to minimize the number of prediction errors. However, in many practical applications the assumption that all errors are equally important is invalid. For example, in medical domain misclassifying an ill patient as a healthy one is usually much more harmful than treating a healthy patient as an ill one and sending him for additional examinations. In database marketing the cost of mailing to a non-respondent is very small, but the cost of not mailing to someone who would respond is the entire profit lost [6].

In the paper a new approach based on the existing system EDRL-MD [12] (EDRL-MD, for Evolutionary Decision Rule Learner with Multivariate Discretization) is proposed. The system learns decision rules using an *evolutionary algorithm* [15] (EA). EAs are stochastic techniques, which have been inspired by the process of biological evolution. Their advantage over greedy search methods is the ability to avoid local optima.

The main novelty of EDRL-MD lies in dealing with continuous-valued attributes. Most of decision rule systems employ univariate discretization methods, which search for threshold values for only one attribute at the same time. In contrast to them, EDRL-MD learns rules simultaneously searching for threshold values for all continuous-valued attributes. This approach is called [12] *multivariate discretization*.

The goal of the original EDRL-MD is to minimize the number of classification errors. In order to enable our system to minimize misclassification cost we modified the *fitness function*, which is optimized by the evolutionary search process.

Several EA-based systems, which learn decision rules in either propositional (e.g., GABIL [5], GIL [10], EDRL [13]) or first order form (e.g., REGAL [9]) were proposed. According to our knowledge, they are unable to process continuous-valued features directly and they cannot minimize misclassification costs.

The remainder of the paper is organized as follows. The next section briefly discusses the related research on cost-sensitive learning. Section 3 contains a short presentation of EDRL-MD. Section 4 describes the modifications of the fitness function, which enable the learning system to minimize misclassification cost. In Section 5 an experimental evaluation on several real-life datasets and a short discussion of the results are presented. The last section contains our conclusions and the directions of the future work.

2 Related work on cost-sensitive learning

The process of inductive learning may involve different costs [21] e.g., costs of tests (features), costs of cases, costs of errors. In the literature the latter kind of costs is the most commonly discussed one.

Several attempts to incorporate misclassification costs into decision tree or decision rule learning were made so far. The first approach was introduced by Breiman et al. [3] in CART decision tree learning system. Their method consists in modification of the class prior probabilities used in the splitting criterion. The cost-based measure is also used for tree pruning.

In a simpler approach (e.g., [2], [11]) error costs are taken into consideration during the pruning phase, but not during the induction phase. In such case the pruning procedure has a limited capability to change the structure of the classifier obtained by the error-based learning. Consequently, ignoring the misclassification cost at the first phase is the main drawback of this approach.

Pazzani et al. [17] introduced three cost-sensitive algorithms for decision list induction. Their method was applied to a real telephone network troubleshooting problem.

Ting [19] proposed a modified version of C4.5 using instance-weighting for induction of cost-sensitive decision trees. This approach requires the conversion of the cost matrix into the cost vector, which may result in poor performance in multi-class problems.

In [6] Domingos presented a method for making an arbitrary classifier cost-sensitive by wrapping a cost-minimizing procedure around it. However his approach may be computationally inefficient because it requires many runs of the basic learning algorithm.

As for applications of EAs to cost-sensitive learning, Turney [20] described a system called ICET, which learns decision trees taking into account both feature costs and misclassification costs. In his approach a genetic algorithm is used to evolve a population of biases for the induction algorithm (a modified C4.5).

3 Learning decision rules with EDRL-MD

In this section we briefly present the main topics (i.e., representation of solutions and genetic search operators) of the learning system EDRL-MD. More detailed description can be found in [12].

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) $A_1, A_2, \ldots, A_N$ and labeled by a class $c(e) \in C$. The domain of a nominal (discrete-valued) attribute A_i is a finite set $V(A_i)$, while the domain of a continuous-valued attribute A_j is an interval $V(A_j) = [l_j, u_j]$. 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 IF $t_1 \wedge t_2 \wedge \ldots \wedge t_r$ THEN 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 is a disjunctive set of decision rules with the same RHS. By $c_{RS} \in C$ we denote the class on the right-hand side of the ruleset RS.

In our approach the EA is called once for each class $c_k \in C$ to find the ruleset separating the set of positive examples $E^+(c_k)$ from the set of negative examples $E^-(c_k)$. 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. Detailed description of the fitness functions used for error reduction and misclassification cost reduction is presented in Section 4.

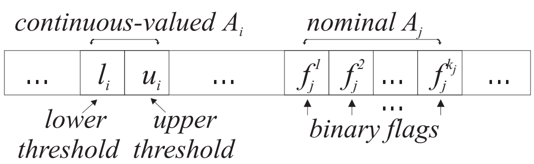

Fig. 1. The string encoding the LHS of a decision rule ($k_j = |V(A_j)|$). The chromosome representing the ruleset is the concatenation of strings. The number of strings in a chromosome can be adjusted by some search operators.

3.1 Representation

The EA processes a population of candidate solutions to a search problem called *chromosomes*. In our case a single chromosome encodes a ruleset RS. 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. The 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_{RS} there is no need for encoding the RHS.

The string is composed (Fig. 1) 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. 2) 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. Evaluating only the boundary thresholds is sufficient [7] for finding the maximum of two fitness functions (1) and (4) discussed in the paper.

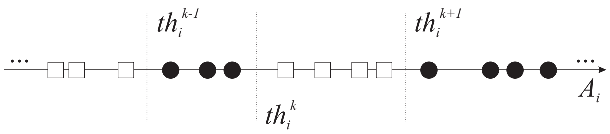

Fig. 2. An example illustrating the notion of boundary threshold. The boundary thresholds $th_i^1, \ldots, th_i^k, \ldots, th_i^{NT_i}$ for the continuous-valued attribute A_i are placed between groups of negative (•) and positive (□) examples.

For a nominal attribute A_j the substring consists of binary flags. Each of the flags corresponds to one value of the attribute.

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 $f_j^1, f_j^2, \ldots, f_j^{|V(A_j)|}$.

Figure 3 shows an example chromosome for a dataset with two numerical attributes: *Salary* and *Amount*, and one nominal attribute *Purpose*. It is assumed that the EA is searching for the optimal ruleset for the class *Accept*.

3.2 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 (represented by a chromosome). The other two: *crossover* and *rule copy* require two arguments RS_1 and RS_2.

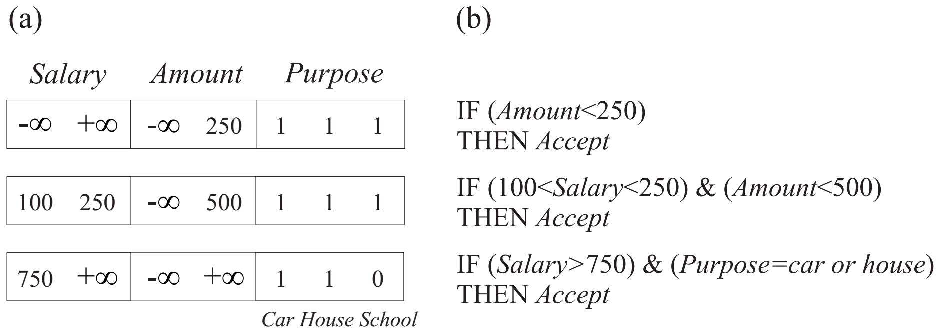

Fig. 3. Representation of rulesets: (a) an example chromosome consisting of three strings, (b) the corresponding ruleset.

A similar approach was proposed by Janikow. His GIL system [10] conducts the search for rulesets using 14 operators. However, GIL 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 A_i. If A_i is nominal, a flag randomly chosen from $f_i^1, f_i^2, \ldots, f_i^{|V(A_i)|}$ is flipped. For a continuous-valued attribute 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 in the ruleset RS to allow it to cover a new random positive example $e^+ \in E^+(c_{RS})$, currently uncovered by R. All conditions in the rule, which conflict with e^+ have to be altered. In case of a condition related to a nominal attribute A_i the flag, which corresponds to $A_i(e^+)$, is set. If A_i is a continuous-valued attribute and the condition $l_i < A_i \leq u_i$ 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 from the ruleset RS. It selects at random a negative example e^- from the set of all the negative examples covered by R. Then it alters a random condition in R in such a way, that the modified rule does not cover e^-. If the chosen condition concerns a nominal attribute A_i the flag which corresponds to $A_i(e^-)$ is cleared. If A_i is a continuous-valued attribute then the condition $l_i < A_i \leq u_i$ 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 boundary threshold such that $\hat{u}_i < A_i(e^-)$.

Rule drop and rule copy operators were used previously by Janikow. They 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. The two argument rule copy adds to one of its arguments RS_1, a copy of a rule selected at random from RS_2, provided that the number of rules in RS_1 is lower than max_R. max_R is an user-supplied parameter, which limits the maximal number of rules in the ruleset.

The crossover operator selects at random two rules R_1 and R_2 from the respective arguments RS_1 and RS_2. Then it applies an uniform crossover [15] to the strings representing R_1 and R_2.

4 Adaptation to cost-sensitive learning

To convert EDRL-MD for cost-sensitive learning we modified the fitness function used to guide the search process. We start this section with the presentation of the fitness function used when the goal of learning is the reduction of the number of errors [12]. Then we describe the changes which allow the EA to minimize the expected misclassification cost rather than the error rate. We also present the cost-sensitive methods for resolving conflicts between rules and for choosing a default class.

We define E_{RS} as the set of examples covered by ruleset RS. The class on the right-hand side of RS is denoted by c_{RS}. Then $E_{RS}^+ = E_{RS} \cap E^+(c_{RS})$ is the set of positive examples correctly classified by RS and $E_{RS}^- = E_{RS} \cap E^-(c_{RS})$ denotes the set of negative examples covered by the ruleset. The total number of positive and negative cases in the learning set are denoted by $POS = |E^+(c_{RS})|$ and $NEG = |E^-(c_{RS})| = M - POS$ respectively. The ruleset RS correctly classifies $pos = |E_{RS}^+|$ positive examples and $NEG - neg$ negative ones, where $neg = |E_{RS}^-|$.

4.1 The fitness for error reduction

In the case of the error-based classification (i.e., with equal misclassification costs) EDRL-MD employs the fitness function f_{error} given by:

$$f_{error}(RS) = \frac{Pr(RS)}{Compl(RS)}, \tag{1}$$

where $Pr(RS)$ is the probability of classifying correctly an example from the learning set by the ruleset RS and $Compl(RS)$ is the complexity of the ruleset. We are interested in maximizing the probability and minimizing the complexity (to obtain the compact ruleset and to avoid overfitting). $Pr(RS)$ and $Compl(RS)$ are given respectively by:

$$Pr(RS) = \frac{pos + NEG - neg}{POS + NEG}. \tag{2}$$

and

$$Compl(RS) = (L/N + 1)^{\alpha}, \tag{3}$$

where L is the total number of conditions in the ruleset, N is the number of attributes and α is an user supplied parameter (typically $\alpha \in [0.1 \ \ldots \ 0.001]$).

4.2 The fitness for misclassification cost reduction

Let $Cost(c_i, c_j)$ be the cost of misclassifying an object from the class c_j as belonging to class c_i. We assume the costs for correct decisions are equal zero i.e., $Cost(c_i, c_i) = 0$ for all c_i.

To adapt the fitness function (1) to the cost-sensitive classification we have to change only $Pr(RS)$, because the complexity term is independent of costs. We replace $Pr(RS)$ by the cost-sensitive $Pr_{cost}(RS)$. The new fitness f_{cost} is defined as follows:

$$f_{cost}(RS) = \frac{Pr_{cost}(RS)}{Compl(RS)}. \tag{4}$$

Classification errors made by the ruleset RS can be divided into two groups: the errors caused by covering a negative example and the errors caused by leaving a positive example not covered. The cost of the former group of errors is denoted by $NC(RS)$ (*Negative examples misclassification Cost*) defined as:

$$NC(RS) = \sum_{e \in E^-_{RS}} Cost(c_{RS}, c(e)). \tag{5}$$

The latter case, when positive examples are not covered by the ruleset is more complicated. Because the EA is called to find the ruleset for one class c_{RS} we cannot figure out to which class a positive example not covered by RS will be classified. This information is available after $|C|$ runs of the EA, when the complete classifier i.e., disjunction of $|C|$ rulesets is learned. Since we do not know the exact cost of a single misclassified (not covered) positive example, we use an approximate measure called $AvgPC(c_{RS})$ (*Average Positive example misclassification Cost*):

$$AvgPC(c_{RS}) = \sum_{c_i \neq c_{RS}} \frac{|E^+(c_i)|}{|E^-(c_{RS})|} * Cost(c_i, c_{RS}). \tag{6}$$

The cost of all misclassified (not covered) positive examples $PC(RS)$ (*Positive examples misclassification Cost*) is given by:

$$PC(RS) = (POS - pos) * AvgPC(c_{RS}). \tag{7}$$

Finally, we define the cost-sensitive replacement for $Pr(RS)$ as:

$$Pr_{cost}(RS) = \frac{MaxNC(c_{RS}) + MaxPC(c_{RS}) - NC(RS) - PC(RS)}{MaxNC(c_{RS}) + MaxPC(c_{RS})}, \tag{8}$$

where $MaxNC(c_{RS})$ and $MaxPC(c_{RS})$ are maximal possible values for $NC(RS)$ and $PC(RS)$ respectively. We can observe $MaxNC(c_{RS})$ when the ruleset covers all the negative examples:

$$MaxNC(c_{RS}) = \sum_{c_i \neq c_{RS}} |E^+(c_i)| * Cost(c_{RS}, c_i), \tag{9}$$

and $MaxPC(c_{RS})$ when none of the positive examples is covered by the ruleset:

$$MaxPC(c_{RS}) = POS * AvgPC(c_{RS}). \tag{10}$$

Because $MaxNC(c_{RS})$ and $MaxPC(c_{RS})$ do not depend on the left-hand side of the ruleset RS, through the maximization of $Pr_{cost}(RS)$ we minimize the total misclassification cost $(NC(RS) + PC(RS))$.

$Pr_{cost}(RS)$ has the following properties: $0 \leq Pr_{cost}(RS) \leq 1$, $Pr_{cost}(RS) = 1$ iff the ruleset covers all positive examples ($pos = POS$) and no negative ones ($neg = 0$), $Pr_{cost}(RS) = 0$ iff the ruleset covers all the negative examples ($neg = NEG$) and no positive ones ($pos = 0$). The above properties hold also for $Pr(RS)$. Furthermore one can show, that if the costs of classification errors are equal then $Pr_{cost}(RS) = Pr(RS)$. It means that Pr is a special case of Pr_{cost} when the goal is to minimize the number of errors rather than the cost.

4.3 Conflict resolution and default class

Contrary to decision trees, the prediction obtained by a set of decision rules can be ambiguous. Such a situation arises when either at least two rules predicting different classes cover an example or none of rules covers an example.

In the former case the classifier has to resolve a conflict between rules. The most common approach to conflict resolution is based on assessment of rule quality [4]. The rule with the highest quality among the conflicting rules is chosen as the 'winner' and the class on the RHS of this rule becomes the final decision of the classifier.

A similar approach was adopted in EDRL-MD. The conflicting rules are ranked according to *average conflict cost*(ACC), which is given by:

$$ACC(R) = \frac{1}{|E_R^{CL}|} * \sum_{e \in E_R^{CL}} Cost(c_R, c(e)), \tag{11}$$

where c_R denotes the class on the RHS of the rule R. It is assumed that each example from the set $E_R^{CL} \subset E$ causes a conflict involving the rule R. The rule, which offers the lowest ACC is chosen as the winner.

If an example is not covered by any rule it is classified into a *default class*. To select the default we consider the training examples not covered by any rule. The class, which minimizes the total misclassification cost of these examples is chosen as the default.

5 Experimental evaluation

In this section experimental results are presented. We have tested our method on ten datasets from UCI repository [1]. The description of the datasets is shown in Table 1. We compared EDRL-MD to C4.5 [18] and its newer, unpublished version C5.0[1]. Contrary to its predecessor C5.0 is capable of taking misclassification

[1] Both C4.5 and C5.0 generate rules by converting a decision tree.

costs into consideration. In all the experiments EDRL-MD was using the fitness function (4).

Dataset	Size	No. of attributes (Numeric/Nominal)	No. of classes
german credit	1000	7/13	2
heart disease	270	7/6	2
cmc	1473	2/7	3
housing	506	12/1	3
breast wisconsin	683	9/0	2
page blocks	5473	10/0	5
pima	532	7/0	2
bupa	345	6/0	2
vehicle	846	18/0	4
crx	690	6/9	2

Table 1. The datasets used in the experiments

5.1 Experimental methodology

Unfortunately, publicly available datasets with known misclassification costs are rare. In the UCI repository only two such datasets (*german credit* and *heart disease*) are provided. These datasets were previously the subject of the cost-sensitive classification experiments in the EU StatLog project [16]. In our experiments the expected misclassification cost and the error rate were estimated by running ten times complete ten-fold crossvalidation. The results are shown in the Table 2.

	Misclassification cost			Error rate		
Dataset	C4.5	C5.0	EDRL-MD	C4.5	C5.0	EDRL-MD
german credit	$.960 \pm .05$	$.656 \pm .02$	$\mathbf{.558 \pm .01}$	$.272 \pm .01$	$.333 \pm .01$	$.453 \pm .01$
heart disease	$.662 \pm .05$	$\mathbf{.476 \pm .04}$	$.499 \pm .04$	$.207 \pm .02$	$.243 \pm .02$	$.375 \pm .02$

Table 2. Misclassification cost and error rate for datasets with known cost matrices. Averages and standard deviations are given. The best results for each dataset are shown in bold.

For the remaining datasets, for which cost matrices are not available, we used a different experimental setup. The results presented in Table 3 are averaged over ten runs of ten-fold crossvalidation. In each run a cost matrix was generated randomly. The off-diagonal elements of the cost matrix were drawn from the uniform distribution over the range $[0, 10]$. The diagonal elements were always zero. For each single crossvalidation run the same random cost matrix was used for all three tested algorithms. For this experiment the standard deviations are not reported because they would incorporate the difference between cost matrices used in different crossvalidation runs.

Similar procedures for generating cost matrices were used in [19] and [14].

	Misclassification cost			Error rate		
Dataset	C4.5	C5.0	EDRL-MD	C4.5	C5.0	EDRL-MD
cmc	1.98	1.67	**1.39**	.460	.523	.566
housing	1.08	1.00	**0.85**	.246	.322	.334
breast wisconsin	.202	.235	**.187**	.040	.085	.063
page blocks	.140	.135	**.134**	.031	.058	.052
pima	1.22	1.00	**.982**	.240	.268	.287
bupa	1.65	1.44	**1.37**	.321	.411	.418
vehicle	1.20	1.13	**1.09**	.270	.392	.411
crx	.758	**.623**	.624	.152	.186	.207

Table 3. Average misclassification cost and error rate for datasets with cost matrices generated randomly. The best results for each dataset are shown in bold.

5.2 Discussion of the results

The results confirm that an algorithm (in our case C4.5), which optimizes the error rate cannot be applied to datasets with non-equal misclassification costs. For all datasets, except *breast wisconsin* C4.5 obtained the highest misclassification cost. As for two cost-sensitive algorithms, EDRL-MD outperformed C5.0 for eight datasets although for four datasets the difference was marginal (less than 10%). C5.0 achieved slightly better results only for two datasets.

C4.5 achieved lower error rates than its cost-sensitive competitors. One could expect such results since error rate and misclassification cost cannot be optimized at the same time.

It should be noted that EDRL-MD requires significantly more processing time than the other algorithms. Nevertheless the learning time required by our system is acceptable. For instance EDRL-MD needs 51 seconds of CPU time on a PC workstation (PIII 700 MHz) to learn decision rules for the largest of datasets (*page blocks*) whereas C5.0 needs about 1 s.

6 Conclusions and future work

In this paper we presented the method of incorporation of misclassification costs into the decision rule learning system EDRL-MD. Contrary to some approaches based on the cost-sensitive pruning our method takes the error costs into account during the inductive search. The results are promising, especially when compared with the results obtained by the cost-sensitive learning system C5.0.

Several directions of the future research exist. One of them is an extension of our cost model. For instance, it would be relatively easy to add costs of features (tests) to the fitness function. Moreover in some applications e.g., fraud detection [8] cost of an error depends on the amount of money involved in the given example. Our approach could be extended to cover such situations as well. In this case instead of using a single cost matrix for the whole dataset the algorithm should associate a cost vector with each single observation and employ a modified fitness function.

Since in many practical applications of classification one has to deal with increasingly large databases, the computational complexity of learning methods becomes important. As for evolutionary algorithms, it is the well known fact [9], that their efficiency can be significantly improved by implementation in a distributed environment (e.g., cluster of workstations). Currently we are developing such a solution for EDRL-MD.

Acknowledgments The authors are grateful to Prof. Leon Bobrowski for his support and useful comments. This work was supported by the grant W/II/1/00 from Technical University of Białystok.

References

1. Blake, C., Keogh, E., Merz, C.J.: *UCI repository of machine learning databases*,[http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Dept. of Computer Science (1998).
2. Bradford, J.P., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.E.: Pruning decision trees with misclassification costs. In *Proc. of the Tenth European Conf. on Machine Learning*. Springer Verlag (1998) 131-136.
3. Breiman, L., Friedman, R.A., Olshen, R., Stone, C.J.: *Classification and Regression Trees*. Wadsworth (1984).
4. Bruha, I., Quality of decision rules: Definitions and classification schemes for multiple rules. In: Nakhaeizadeh, G., Taylor, C.C., (eds.) *Machine Learning and Statistics. The Interface*. Wiley-Interscience (1997) 107-131.
5. De Jong, K., Spears, W.M., Gordon, D.F.: Using genetic algorithm for concept learning. *Machine Learning* 13 (1993) 168-182.
6. Domingos, P.: MetaCost: A general method for making classifiers cost-sensitive. In *Proc. of Int. Conf. on Knowledge Discovery and Data Mining, KDD'99*. ACM Press (1999) 155-164.
7. 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.
8. Fawcett, T., Provost, F.J., Adaptive fraud detection, *Data Mining and Knowledge Discovery* 1 (1997)
9. Giordana, A., Neri, F.: Search-intensive concept induction. *Evolutionary Computation* 3(4) (1995) 375-416.

10. Janikow, C.: A knowledge intensive genetic algorithm for supervised learning. *Machine Learning* 13 (1993) 192-228.
11. Knoll, U., Nakhaeizadeh, G., Tausend, B.: Cost-sensitive pruning of decision trees. In *Proc. of the* 8^{th} *European Conf. on Machine Learning.* Springer LNCS 784 (1994) 383-386.
12. Kwedlo, W., Krętowski, M.: An evolutionary algorithm using multivariate discretization for decision rule induction. In *Principles of Data Mining and Knowledge Discovery.* 3^{rd} *European Conference PKDD'99.* Springer LNCS 1704 (1999) 392-397.
13. Kwedlo, W., Krętowski, M.: Discovery of decision rules from databases: an evolutionary approach. In *Principles of Data Mining and Knowledge Discovery.* 2^{nd} *European Symposium PKDD'98.* Springer LNCS 1510 (1998) 370-378.
14. Margineantu, D.D., Dietterich, T.G.: Bootstrap methods for the cost-sensitive evaluation of classifiers, In *Proc.* 17^{th} *Int. Conf. on Machine Learning ICML'2000*, Morgan Kaufmann (2000) 583-590.
15. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs.* 3^{rd} edn. Springer (1996).
16. Michie, D., Spiegelhalter, D.J., Taylor, C.C.: *Machine Learning, Neural and Statistical Classification.* Ellis Horwood Ltd. (1994) [also available at http://www.amsta.leeds.ac.uk/∼charles/statlog/index.html].
17. Pazzani, M., Merz, C., Murphy, P., Ali, K., Hume, T., Brunk, C.: Reducing misclassification costs. In *Proc. of Int. Conf. on Machine Learning, ICML'94.* Morgan Kaufmann (1994) 217-225.
18. Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).
19. Ting, K.M.: Inducing cost-sensitive trees via instance weighting. In *Principles of Data Mining and Knowledge Discovery.* 2^{nd} *European Symposium PKDD'98.* Springer LNCS 1510 (1998) 139-147.
20. Turney, P.: Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research* 2 (1995) 369-409.
21. Turney, P.: Types of cost in inductive concept learning. In *Proc. of ICML'2000 Workshop on Cost-Sensitive Learning.* Stanford, CA (2000).