Evolutionary Induction of Decision Trees for Misclassification Cost Minimization

Marek Krętowski and Marek Grześ

Faculty of Computer Science, Białystok Technical University
Wiejska 45a, 15-351 Białystok, Poland
{mkret,marekg}@ii.pb.bialystok.pl

Abstract. In the paper, a new method of decision tree learning for cost-sensitive classification is presented. In contrast to the traditional greedy top-down inducer in the proposed approach optimal trees are searched in a global manner by using an evolutionary algorithm (EA). Specialized genetic operators are applied to modify both the tree structure and tests in non-terminal nodes. A suitably defined fitness function enables the algorithm to minimize the misclassification cost instead of the number of classification errors. The performance of the EA-based method is compared to three well-recognized algorithms on real-life problems with known and randomly generated cost-matrices. Obtained results show that the proposed approach is competitive both in terms of misclassification cost and compactness of the classifier at least for some datasets.

1 Introduction

Nowadays computer-aided decision support systems become more and more popular in solving complex decision-making problems in marketing, finance and medicine. Based on gathered datasets of examples they enable training various types of classifiers in form of neural networks, decision trees and rules. However, in most cases only the number of classification errors is taken into account during the induction process. In many practical applications this classical approach is not suitable because there are other factors, such as costs, which may influence final decisions. In [24] Turney discussed different types of costs associated with inductive learning (e.g., the cost of tests, the cost of objects and the misclassification cost). The term *cost-sensitive classification* encompasses all these types of costs.

There are two main groups of methods for making a classifier cost-sensitive. In the first group, individual error-based systems are converted into cost-sensitive ones. One of the first attempts to incorporate misclassification costs into decision tree learning was made in the *CART* system [4]. The method consists in the modification of the class prior probabilities used in the splitting criterion and in the application of the cost-based measure to a tree pruning. Another cost-sensitive methods for pruning decision trees are proposed in [10,3]. However, it should be emphasized that stand-alone pruning procedures have only a limited capability to change the tree structure created by an error-based inducer. In [22]

B. Beliczynski et al. (Eds.): ICANNGA 2007, Part I, LNCS 4431, pp. 1–10, 2007.

the *C4.5* system [21] was modified using instance-weighting, but the method requires converting of a cost matrix into a cost vector, which can result in poor performance in multi-class problems. Recently, Ling *et al.* [17,26] proposed an algorithm that minimizes the sum of the misclassification and test costs. This approach is based on a new splitting criterion (*total cost*) for nominal attributes and two-class problems.

The second group includes general methods for making an arbitrary classifier cost-sensitive. *MetaCost* [7] is based on wrapping a meta-learning stage around the error-based classifier. Another method proposed by Zadrozny *et al.* [25] uses cost-proportionate rejection sampling and ensemble aggregation. Iterative weighting and gradient boosting are investigated by Abe *et al.* [1] in multi-class problems.

The proposed approach consists in developing a specialized evolutionary algorithm for global induction of cost-sensitive decision tree classifiers. Several EA-based systems which learn decision trees in the top-down manner (e.g. *BTGA* [6], *OC1-ES* [5], *DDT-EA* [12]) have been proposed so far. Generally, they apply evolutionary approach to the search of splitting hyper-planes in non-terminal nodes of oblique decision trees.

In this paper, a global approach to decision tree induction is advocated. In contrast to a step-wise construction, the whole tree is being searched at the time. It means the simultaneous search for an optimal structure of the tree and for all tests in non-terminal nodes. The global approach was initially proposed by Koza in [11], where genetic programming was used for evolving LISP S-expressions that correspond to simple decision trees. A similar idea was investigated in the *GATree* system [20] which directly evolves classification trees with nominal tests. In our previous papers, we showed that EA-based global inducer can efficiently generate univariate [13], linear [14] and mixed [15] decision trees.

Concerning applications of evolutionary techniques to cost-sensitive learning of decision trees, according to our knowledge, only one attempt can be mentioned. In [23] Turney proposed the *ICET* system, which uses the standard genetic algorithm to evolve a population of biases for the modified *C4.5*. Both feature and misclassification costs are taken into account.

The rest of the paper is organized as follows. In the next section, the proposed evolutionary algorithm is described in details. Section 3 contains experimental validation of the approach on the benchmark classification problems with known and randomly generated cost-matrices. In the last section conclusions and possible directions of the future work are presented.

2 Evolutionary Algorithm for Global Induction

The structure of the proposed evolutionary algorithm follows the typical framework [18] and only application-specific issues (the fitness function, specialized genetic operators, ...) are described in more detail in this section.

2.1 Preliminaries

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 set of all examples from the class $c_k \in C$ is denoted by $C_k = \{e \in E : c(e) = c_k\}$ and the class assigned (predicted) by the tree T to the example e is denoted by $T(e)$.

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

2.2 Representation, Initialization and Termination Condition

In our system, decision trees are represented in their actual form as classical univariate trees where each test in a non-terminal node concerns only one attribute. Additionally, in every node information about learning vectors associated with the node is stored and it enables the algorithm to perform efficiently local structure and tests modifications during applications of genetic operators.

In case of a nominal attribute at least one value is associated with each branch. It means that an inner disjunction is built-in into the induction algorithm. For a continuous-valued feature typical inequality tests with boundary thresholds[1] [8] as potential splits are considered. All boundary thresholds for each continuous-valued attribute are calculated before starting the evolutionary induction [13]. It significantly limits the number of possible splits and focuses the search process.

In standard error-based decision trees class labels are associated with leaves by using the majority rule based on training objects which reached a leaf-node. In cost-sensitive case class labels for leaves are chosen to minimize the misclassification cost in each leaf.

Individuals in the initial population are generated as follows. The classical top-down algorithm is applied, but tests are chosen in a dipolar way [12]. Among feature vectors located in the considered node two objects from different classes (so called *mixed dipole*) are randomly chosen. An effective test, which separates two objects into sub-trees, is created randomly by taking into account only attributes with different feature values. Recursive divisions are repeated until the stopping condition (based on the minimal number of learning vectors in a node or homogeneity of a node) is met. Finally, the resulting tree is post-pruned according to the fitness function.

The algorithm terminates if the fitness of the best individual in the population does not improve during the fixed number of generations (default value is equal 200). Additionally, the maximum number of generations is specified which limits the computation time in case of a very slow convergence (default value: 1000).

[1] A boundary threshold for the given attribute is defined as a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of the attribute, in which the examples belong to two different classes.

2.3 Fitness Function

The properly defined fitness function is a crucial element for every evolutionary algorithm. In case of induction of decision structures it is obvious that there is no possibility to directly optimize accuracy of the classifier on unseen examples. Instead, the system performance on the training data is usually used to guide the search process and additional factors are introduced to prevent the overfitting and to increase the generalization power of the classifier (e.g. [13,16]). An analogous approach is applied in our system.

The misclassification cost $MC(T)$ of the tree T is estimated on the training data:

$$MC(T) = \frac{1}{M} \cdot \sum_{e \in E} Cost(T(e), c(e)). \tag{1}$$

The values of the misclassification cost do not fit into range $[0, 1]$ like classification errors, however, it is easy to calculate the maximal cost for a given dataset and a cost matrix:

$$MaxMC = \frac{1}{M} \cdot \sum_{c_k \in C} |C_k| \cdot \max_{i \neq k} Cost(c_i, c_k). \tag{2}$$

Hence, by dividing $MC(T)$ by $MaxMC$ one can obtain the *normalized misclassification cost*, which is equal 0 for the perfect prediction and 1 in the worst case. Finally, the fitness function, which is maximized, is defined as follows:

$$Fitness(T) = \left(1 - \frac{MC(T)}{MaxMC}\right) \cdot \frac{1}{1 + \alpha \cdot S(T)}, \tag{3}$$

where $S(T)$ is the size of the tree T expressed as a number of nodes and α is a user supplied parameter (default value is 0.001).

It should be expected that there is no one optimal value of α for all possible datasets and cost matrices. When the certain problem is analyzed, tuning this parameter may lead to the improvement of the results (in terms of the misclassification cost or classifier complexity).

2.4 Genetic Operators

There are two specialized genetic operators called *CrossTrees* and *MutateNode* which fulfill the role of crossover and mutation operators from the classic framework.

CrossTrees alike the standard crossover operator modifies two chromosomes (i.e. trees) by swapping their certain parts. There are three types of crossover-like modifications that can be performed on trees: two of them concern sub-trees and one only tests. Firstly, one random node is drawn from each tree. Then the type of modification is chosen. By default all types of these operations are equally probable, but the user can specify his own proportions. First one exchanges subtrees rooted in chosen nodes. This variant is analogous to the typical crossover operator introduced in genetic programming. Second operator replaces only tests

between chosen nodes. This type of modification is possible solely when tests have the same number of outcomes. Third operator is the most radical. It replaces in random order all branches of chosen nodes. It is evident that the same effect can be achieved by combining two or more exchanges of the first type.

MutateNode, like the standard mutation operator, takes one argument (i.e. a single node of the tree). This operator can be applied to each node with a given probability (default value 0.05). The result of this operator depends on what kind of a node is considered (i.e. a leaf or an internal node). When modifying an internal node the following possibilities are by default equally probable (this probabilities are again user specified):

- a test in the current node can be replaced by a completely new test chosen in a dipolar way,
- a less drastic possibility in reference to the previous one consists in changing the threshold of a test on a real attribute or modifying groups of nominal values (i.e. two branches of the current node can be merged into one branch or a group can be split creating an additional branch) according to the application of an inner disjunction to tests on nominal attributes,
- a test in the current node and a test from one of node's sons can be exchanged, it concerns nodes which have non-leaf descendants
- one of the branches of the current node can be multiplied and replace another branch which is to be dropped,
- each node, even the root, can be transformed into a leaf; this operator allows reducing in a straight way the tree size.

A leaf node can be modified on condition that it contains feature vectors belonging to different classes. Such a leaf can be replaced by:

- a non-terminal node with a randomly chosen test,
- a sub-tree generated according to the dipolar algorithm which is also applied during initialization.

After application of some of the described operators it may be necessary to alter locations of some of the learning vectors. It can lead to such a situation where there are nodes or even whole sub-trees without any training examples. For this reason empty parts of the tree have to be removed. It is done by using either simple algorithm which does it directly or by specialized operator called *MaximizeFitness*. This operator not only drops empty parts of the tree but also performs more sophisticated modifications. It visits all nodes of a tree in bottom-up order. It tries to replace each not-terminal node by a leaf while taking into account potential gain in the fitness. *MaximizeFitness* is by default applied to all trees from initial population and to sub-trees which appear after replacing leaves.

As a selection mechanism the ranking linear selection [18] is applied. Additionally, the chromosome with the highest value of the fitness function in the iteration is copied to the next population (the *elitist strategy*).

3 Experimental Results

In this section experimental validation is presented. The proposed approach (denoted in tables as *GDT-MC*) is compared with the commercial cost-sensitive classifier *C5.0* which is enhanced version of *C4.5* [21]. Our global inducer is also compared with *MetaCost* [7] and *CostSensitiveClassifier* (denoted as *CSJ48*) of the *Weka* system [9]. Both *MetaCost* and *CSJ48* are based on wrapping an error-based classifier (*J48* which is *Weka*'s implementation of *C4.5* was used in the experiments).

Performance of all presented systems is assessed on a subset of the well-known datasets publicly available from the *UCI Machine Learning Repository* [2]. More complex datasets with continuous-valued features and no missing values were chosen. All results presented in the tables correspond to averages of 10 runs and were obtained by using test sets (when available) or by the 10-fold stratified crossvalidation. The average number of leaves is given as a complexity measure of classifiers.

3.1 Known Cost-Matrices

Only for two datasets (namely *german* and *heart*) misclassification costs are provided. The cost matrix in both cases is the same and non-zero costs are as follows: $Cost(c_2, c_1) = 1$ and $Cost(c_1, c_2) = 5$.

Table 1. Misclassification costs and tree sizes obtained for datasets with known cost matrix

	MetaCost		*CSJ48*		*C5.0*		*GDT-MC*	
Dataset	Cost	Size	Cost	Size	Cost	Size	Cost	Size
german	1.26	31.61	0.69	67.31	0.71	81.37	**0.58**	11.69
heart	1.01	18.05	**0.52**	10.87	0.56	16.49	0.61	25.07
average	1.14	24.83	0.61	39.09	0.64	48.93	**0.6**	18.38

The results obtained with the default value of α are collected in Table 1. It should be noted that in case of the *german* dataset EA-based system performs better (both in terms of the misclassification cost and the classifier size) than all remaining algorithms. As for the second dataset *GDT-MC* is slightly worse than $C5.0$ and $CSJ48$ but still much better than $MetaCost$.

In order to verify the impact of the α parameter on the results, a series of experiments with varying α was prepared (see Fig. 1 and Fig. 2). As it could be expected, along with a decrease of α an increase of trees complexity can be observed. Concerning the misclassification cost, after initial small decrease global minimum is reached and the cost quickly rises for larger trees. It can be also observed that for both datasets the default setting of this parameter (denoted by vertical dotted line) is not really optimal. For the *german* dataset, where $GDT-MC$ obtained the best result (0.58) among all competitors,

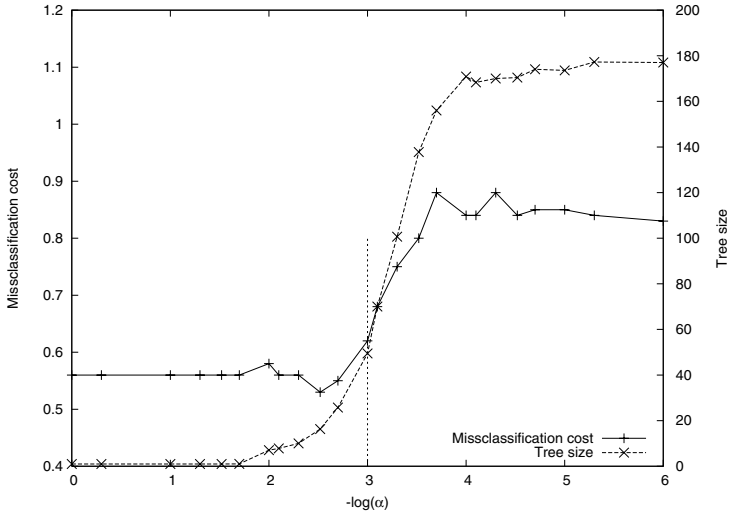

Fig. 1. The impact of α parameter on the misclassification cost and the tree complexity for *heart* dataset

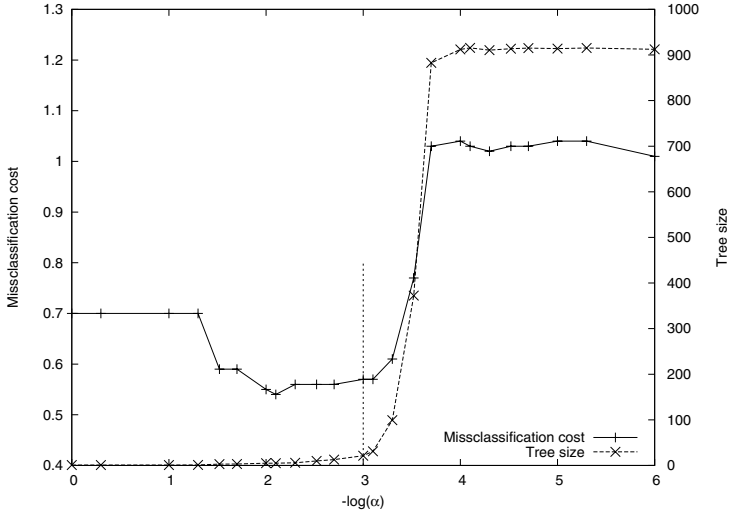

Fig. 2. The impact of α parameter on the misclassification cost and the tree complexity for *german* dataset

a further decrease of the misclassification cost to 0.54 is possible. Concerning the *heart* dataset it is also possible to achieve the misclassification cost lower than these obtained by $C5.0$ or $CSJ48$.

3.2 Simulated Cost-Matrices

For the remaining datasets, for which cost matrices are not provided, a different but typical experimental setup is applied (see e.g. [22,19,16]). In each run of the 10-fold crossvalidation a cost matrix was generated randomly. The off-diagonal elements of the cost matrix were drawn from the uniform distribution over the range $[1, 10]$. The diagonal elements were always zero. The standard deviations are not presented because if they had been calculated they would have expressed the effects of varying cost matrices [7]. For each single crossvalidation run the same random cost matrix and the same training data splits were used for all tested algorithms.

Table 2. The average misclassification cost and the tree size for datasets with randomly generated cost matrices

	MetaCost		*CSJ48*		*C5.0*		*GDT-MC*	
Dataset	Cost	Size	Cost	Size	Cost	Size	Cost	Size
$breast-w$	0.29	10.02	0.24	9.81	0.24	11.15	**0.22**	5.6
$australian$	0.63	7.0	**0.40**	8.0	0.60	12.50	0.64	12.0
$balance-scale$	1.33	34.0	1.40	19.0	1.27	29.80	**1.16**	22.4
$bupa$	2.44	22.57	1.61	14.94	**1.59**	17.02	1.70	43.76
$cars$	0.21	31.0	0.07	31.0	**0.05**	25.60	**0.05**	30.0
cmc	3.08	148.6	2.46	129.0	2.36	166.3	**2.04**	9.87
$glass$	2.40	14.0	2.40	14.0	**1.64**	20.70	1.69	33.1
$page-blocks$	0.19	41.09	**0.17**	36.57	**0.17**	34.57	0.24	4.33
$pima$	1.64	28.19	0.93	11.82	0.93	15.61	**0.91**	8.92
$wine$	1.08	5.0	1.08	5.0	0.88	5.10	**0.78**	6.40
$vehicle$	1.96	70.88	**1.53**	57.21	**1.53**	68.98	1.54	16.69
average	1.39	37.49	1.12	30.58	1.02	37.03	**1.0**	17.55

As it could be observed in Table 2, for 5 out of 11 datasets, misclassification costs of the classifiers generated by the proposed method are lower than costs of the competitors. It is an indicative achievement while taking into account the fact that it was compared with so many renowned and efficient counterparts. On the other hand it is worth mentioning that there is only one dataset - *page blocks* - on which *GDT-MC* is slightly worse than all remaining algorithms. But it was verified that tuning the parameter α leads to the real improvement: 0.11 ($\alpha = 0.00005$) which gives *GDT-MC* the best score.

Finally, it was confirmed one more time that global induction generally results in less complex decision structures than obtained by top-down inducers. Only for *bupa* dataset the resulting decision tree was overgrown.

It should be mentioned that the global induction requires more processing time compared to traditional top-down algorithms. Nevertheless, the learning time required by *GDT-MC* is acceptable. For instance the system needs 2 minutes and 20 second (on average) of CPU time on a PC workstation (PIV 3GHz, 1GB RAM) to generate classifier for the largest dataset (*page blocks* - 5473 examples, 10 features and 5 classes).

4 Conclusions

In the paper, a new method of univariate decision tree induction for misclassification cost minimization is presented. The proposed approach consists in developing specialized evolutionary algorithm which globally searches for optimal decision tree. Results of the experimental validation show that our system is able to generate competitive classifiers both in terms of the misclassification cost and the decision tree size. It should be also noted that by tuning the α parameter which corresponds to the importance of the complexity term in the fitness function, even better results for the given dataset can be obtained.

Furthermore, many possibilities for improvement still exist (e.g. better fitness function, new genetic operators, ...). One direction of current research is an extension of the cost model by incorporating costs of features (tests). It can be done mainly by modifying the fitness function.

The proposed approach is not the fastest one now, but hopefully it is well-known that evolutionary algorithms are well-suited for parallel architecture. We plan to speed up our system by re-implementing it in the distributed environment.

Acknowledgments. This work was supported by the grant W/WI/5/05 from Białystok Technical University.

References

1. Abe, N., Zadrozny, B., Langford, J.: An iterative method for multi-class cost-sensitive learning. In *KDD'04*. ACM Press, (2004) 3–11.
2. Blake, C., Keogh, E., Merz, C.: *UCI repository of machine learning databases*,[http://www.ics.uci.edu/˜mlearn/MLRepository.html], (1998).
3. Bradford, J.P., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.E.: Pruning decision trees with misclassification costs. In *Proc. of ECML'98*. Springer (1998) 131–136.
4. Breiman, L., Friedman, J., Olshen, R., Stone C.: *Classification and Regression Trees*. Wadsworth Int. Group (1984).
5. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **7**(1) (2003) 54–68.
6. Chai, B. *et al.*: Piecewise-linear classifiers using binary tree structure and genetic algorithm. *Pattern Recognition* **29**(11) (1996) 1905–1917.
7. Domingos, P.: MetaCost: A general method for making classifiers cost-sensitive. In *Proc. of KDD'99.*, ACM Press (1999) 155–164.
8. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of IJCAI'93.*, (1993) 1022–1027.

9. Frank, E. *et al.*: Weka 3 - Data Mining with Open Source Machine Learning Software in Java. [http://www.cs.waikato.ac.nz/~ml/weka]. University of Waikato (2000).
10. Knoll, U., Nakhaeizadeh, G., Tausend, B.: Cost-sensitive pruning of decision trees. *LNCS* 784 (1994) 383–386.
11. Koza, J.: Concept formation and decision tree induction using genetic programming paradigm. *LNCS* 496 (1991) 124–128.
12. Krętowski, M.: An evolutionary algorithm for oblique decision tree induction. *LNAI* 3070, (2004) 432–437.
13. Krętowski, M., Grześ, M.: Global learning of decision trees by an evolutionary algorithm. In: *Information Processing and Security Systems*, Springer, (2005) 401–410.
14. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection. *LNAI* 4029, (2006) 400–409.
15. Krętowski, M., Grześ, M.: Mixed decision trees: An evolutionary approach. *LNCS* 4081, (2006) 260–269.
16. Kwedlo, W., Krętowski, M.: An evolutionary algorithm for cost-sensitive decision rule learning. *LNAI* 2167, (2001) 288–299.
17. Ling, C., Yang, Q., Wang, J., Zhang, S.: Decision trees with minimal costs, In: *Proc. of ICML'04.*, ACM Press (2004), Article No. 69.
18. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3^{rd} edn. Springer (1996).
19. Margineantu, D.D., Dietterich, T.G.: Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Proc. of ICML'2000.*, Morgan Kaufmann (2000) 583–590.
20. Papagelis, A., Kalles, D.: Breeding decision trees using evolutionary techniques. In: *Proc. of ICML'01.*, Morgan Kaufmann (2001) 393–400.
21. Quinlan, J.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann (1993).
22. Ting, K.M.: An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering* **14**(3), (2002) 659–665.
23. Turney, P.: Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research* 2 (1995) 369–409.
24. Turney, P.: Types of cost in inductive concept learning. In *Proc. of ICML'2000 Workshop on Cost-Sensitive Learning.* Stanford, CA (2000).
25. Zadrozny, B., Langford, J., Abe. N.: Cost-sensitive learning by cost-proportionate example weighting concept learning. In *Proc. of ICDM'03.* IEEE Press (2003).
26. Zhang S., Qin, Z., Ling, C. Sheng, S.: Missing is usefull: Missing values in cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering* **17**(12), (2005) 1689–1693.