Global learning of decision trees by an evolutionary algorithm [1]

Marek Krętowski, Marek Grześ

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

Abstract: In the paper, an evolutionary algorithm for global induction of decision trees is presented. In contrast to greedy, top-down approaches it searches for the whole tree at the moment. Specialised genetic operators are proposed which allow modifying both tests used in the non-terminal nodes and structure of the tree. The proposed approach was validated on both artificial and real-life datasets. Experimental results show that the proposed algorithm is able to find competitive classifiers in terms of accuracy and especially complexity.

Keywords: Data mining, decision trees, evolutionary algorithms, global induction

1 Introduction

Amount of information, which is gathered in business and scientific database systems, is growing faster and faster. Efficient analysis of huge available data becomes one of the most crucial problems in computer science. Knowledge discovery in databases (KDD) is newly emerged discipline trying to cope with this problem [9]. One of the most well known data mining techniques used in KDD process is extraction of decision trees (DT). Many induction algorithms have been proposed so far, e.g. CART [5] or C4.5 [16] (see [15] for exhausting multidisciplinary review). The advantages of the DT-based approach include among other things natural representation and ease of interpretation.

Finding the best decision tree is very difficult optimisation problem (NP-complete) [10] and therefore most of the existing DT systems use heuristic approach based on the top-down induction. Starting from the root node, which contains all feature vectors from the learning set, an optimal split is searched. If effective test is found input vectors from the considered node are divided among newly created sub-

[1] This work was supported by the grant W/WI/1/02 from Białystok Technical University

K. Saeed, J. Pejaś: Information Processing and Security Systems, pp. 401-410, 2005.

nodes and for each one the procedure is recursively called. Such a greedy search technique is fast and generally leads to acceptable results in typical applications. In [14] effectiveness of the top-down induction was investigated on artificial datasets with known optimal tree and near optimal solution was found in most of the cases. However, it is evident that for certain classification problems (e.g. classical "chessboard" problem [3]) top-down approach fails and more sophisticated method should be applied.

In this paper, a global approach to decision tree induction is advocated. In contrast to typical stepwise construction, in the proposed method the whole tree is searched at the time. It means simultaneous search for an optimal structure of the tree and for all tests in non-terminal nodes. As it could be expected, global tree construction is far more complicated and computationally complex problem. As a first step toward global induction limited look-ahead algorithms are proposed (e.g. APDT [17] evaluates goodness of a split based also on the degree of linear separability of sub-nodes). Another approach consists in a two-stage induction, where a greedy algorithm is applied in the first stage and then the tree is refined to be as close to optimal as possible (e.g. GTO [1] is an example of linear programming based method for optimising trees with fixed structures). In [11] Koza proposed adopting genetic programming (GP) methods for evolving LISP S-expressions corresponding to decision trees. Another GP-based system for induction of classification trees with limited oblique splits is presented in [4].

The proposed approach consists in developing a specialised evolutionary algorithm for generation of decision tree classifiers. Evolutionary algorithms (EA) are flexible optimisation techniques, which were inspired by the process of biological evolution [13]. Their main advantage over greedy search methods is their ability to avoid local optima. Several EA-based systems, which learn decision trees in the top-down manner (e.g. BTGA [7], OC1-ES [6], DDT-EA [12]), have been proposed so far. Generally, they applied evolutionary approach to the test search, especially in the form of hyper-planes.

The rest of the paper is organised as follows. In the next section, the proposed evolutionary algorithm for global induction of decision tree is presented in details. Section 3 contains experimental validation of the approach on both artificial and real-life classification problems. In the last section conclusions and possible directions of the future work are presented.

2 Evolutionary Decision Tree Learning

The proposed evolutionary algorithm follows the general framework presented in [13]. In this section, only application-specific issues are described: representation, genetic operators and the fitness function.

2.1 Representation, initialisation and termination condition

There are two the most common strategies of applying evolutionary approach to solve optimisation problems. In the first approach, the candidate solutions are encoded in the fixed-size (usually binary) chromosomes, which allow using standard genetic operators: crossover and mutation. The second approach consists in applying more sophisticated representations (e.g. variable-length) and developing specialised genetic operators.

Decision trees are complicated tree structures, in which number of nodes, type of the tests and even number of test outcomes are not known in advance. It is why the second aforementioned approach is the most adequate, especially if the whole tree is searched in one run of EA. In our system, decision trees are not especially encoded in individuals and they are represented in their actual form.

Each test in non-terminal nodes concerns only one attribute. Depending on the type of the feature used (nominal or continuous-valued) two test forms are possible. In case of a nominal attribute each value is associated with one branch. For a continuous-valued feature only typical inequality test with two outcomes is allowed ($attribute_i \leq threshold_i$). It was shown [8] that for finding the maximum of certain class of target functions it is sufficient to consider only so-called *boundary thresholds* as potential splits. 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, that one of the examples is positive and the other is negative (Fig. 1). The above property also holds for our fitness function.

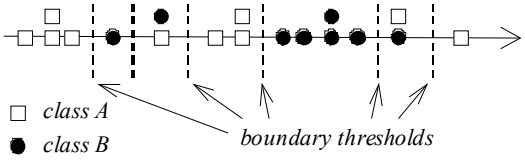

Fig. 1. The notion of the boundary threshold for the given attribute.

Hence before starting the actual evolutionary algorithm, all boundary thresholds for each continuous-valued attribute are calculated. It enables to significantly limit the number of possible spits and focus the search process. As a result the algorithm is faster and more robust.

Each individual in the initial population is 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 are randomly chosen. An effective test, which separates two objects into sub-trees, is created randomly taking into account only attributes with different feature

values. Recursive divisions are repeated until stopping conditions are met. Finally, the resulting tree is post-pruned based on 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 1000), which signalises, that the algorithm converged. Additionally the maximum number of generations is specified, which allows limiting the computation time in case of a very slow convergence (default value: 10000).

2.2 Genetic operators

Two specialised and complex genetic operators are proposed: *CrossTrees* and *MutateNode*. The first one is an equivalent of the standard crossover operator. It alters two chromosomes by exchanging certain parts of input trees. There are three possible types of the exchange: two types of sub-trees exchange and exchange of only tests. At the beginning, regardless of the type, one node in each tree is randomly chosen. Then the type of exchange between trees is decided. In the system implementing the presented EA, all variants of *CrossTree* operator are equally probable. In the simplest situation, sub-trees starting from the chosen nodes are substituted (Fig. 2a). This variant is analogous to the typical crossover operator utilised in genetic programming. The second possibility consists in exchanging tests, which are associated with the chosen nodes (Fig. 2b). This type of the operator is only possible, when tests have the same number of outcomes. The last variant of the *CrossTree* seems to be the most complicated. Branches (and their sub-trees), which start from the chosen nodes are exchanged in random order (Fig. 2c). It could be observed that the last possibility is somehow redundant, because the same effect can be achieved by combining two (or more) exchanges of the first type.

It should be noted that in all variants after application of the operator, locations of input feature vectors in altered parts of the tree should be determined once again. This can lead to such a situation, where there are nodes (or even sub-trees) without any feature vectors from the learning set. As a result, empty parts of trees have to be removed.

The second operator *MutateNode* is a mutation-like one and it is applied with the given probability (default value: 0.05) to every single node of the tree. This operator can cause a modification of the test or a change of the node structure. If a non-terminal node is concerned it can be pruned to a leaf or its test can be altered. When modifying the test the following four possibilities are equally probable:

- a new threshold can be randomly chosen without changing the attribute used in the test,
- a completely new test is applied with another randomly chosen attribute and threshold,

- the current sub-tree can be replaced by a sub-tree copied from the neighbouring node (this does not apply to the root node),
- the test can be exchanged with another test taken from randomly chosen son-node (this does not apply to nodes with only leaves as sons).

If a leaf node containing feature vectors from different classes is concerned two options are possible. The leaf can be replaced by:

- a non-terminal node with a new randomly chosen test,
- a sub-tree generated according to dipolar algorithm applied for initialisation.

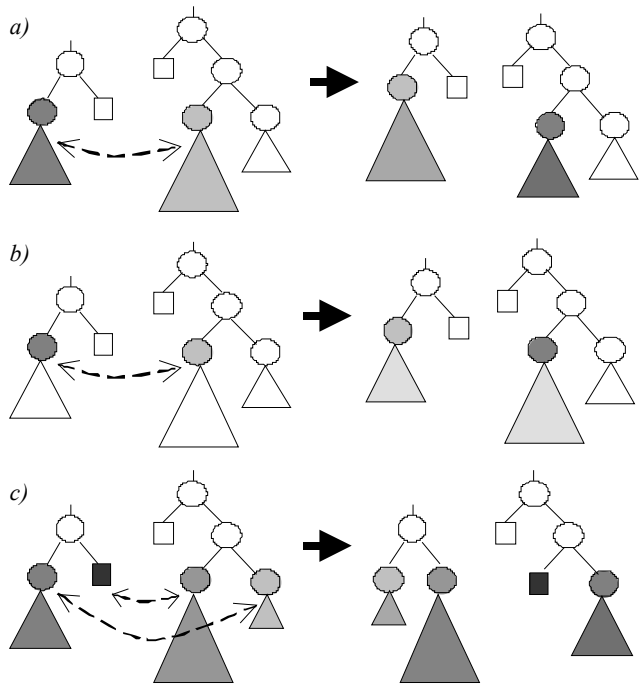

Fig. 2. CrossTree operator: *a)* exchange of sub-trees, *b)* exchange of tests in the randomly drawn nodes; the structure of sub-trees remains not changed, but input vectors can be redirected, *c)* exchange of branches in random order.

As a selection mechanism the linear ranking selection [13] is applied. Additionally, the chromosome with the highest value of the fitness function in the iteration is copied to the next population (*elitist* strategy).

2.3 Fitness function

The simplest form of the target function, which can be optimised in classification problems, concerns only the quality of reclassification. However, it is well known fact that this can lead to overspecialisation of the classifier. Introducing a complexity term allows mitigating the over-fitting problem. The fitness function, which is maximised, has the following form:

$$Fitness = Q_{Rclass} - \alpha \cdot S \ , \tag{1}$$

where Q_{Rclass} is the classification quality estimated on the learning set, S is the size of the tree (number of nodes) and α - is a relative importance of the complexity term and a user supplied parameter. It seems rather obvious that there is no one optimal value of α for all datasets and this parameter can be tuned for specific problems.

3 Experimental results

In this section some preliminary experimental results are presented. First, a few artificial datasets from so-called *chessboard* domain are analysed. It is well known that such problems are difficult to solve for traditional, top-down decision tree systems. In the second group of experiments, some real life datasets taken from UCI Repository [2] are used to generate DT classifiers. Classification accuracy is estimated by running 10 times either the complete ten-fold cross-validation or by using a test set. Size of the classifier is given as a number of all nodes in the tree. Our system (described as a GDT-EA in tables) is run with default parameters mentioned earlier in the system description. The population size is set to 50. For the purpose of the comparison, results obtained by one of the most popular decision tree system - C4.5 (release 8, default parameters) [16] are also presented.

3.1 Artificial datasets

Two examples of *chessboard* training datasets are depicted in Figure 3. Decision borders are defined analytically and both training and test sets were created by using random number generator (number of feature vectors in each compartment is equal 100).

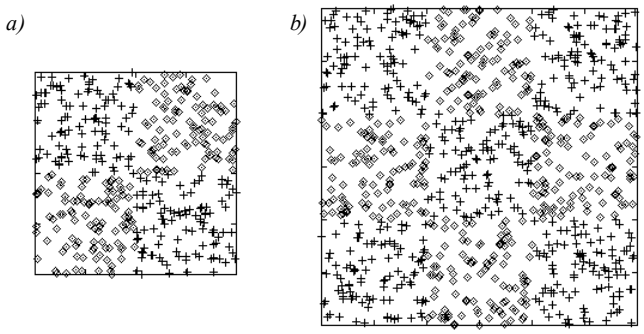

Fig. 3. Examples of *chessboard* datasets: a) 2x2, b) 3x3.

In the global approach parameter α from the fitness function can be used for finding appropriate balance between re-classification accuracy and generalisation power related to the tree complexity. In the first experiments, it is verified how GDT-EA is sensitive to the choice of α parameter (Figure 4). It can be observed that for relatively broad range of values (0.01-0.001) optimal trees were found. Further decrease of α parameter results in performance deterioration especially in terms of tree simplicity. All subsequent experiments are run with α equal 0.0025.

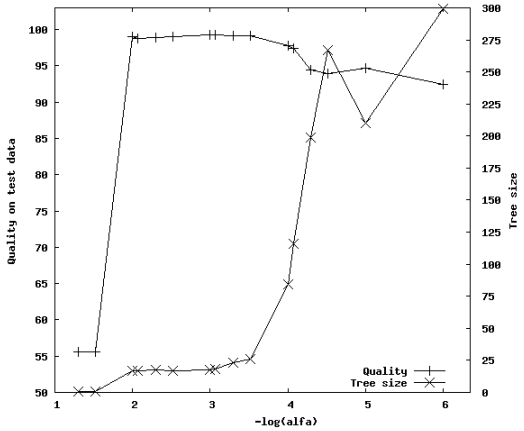

Fig. 4. Impact of α on classification quality and tree size (*chessboard* 3x3 dataset)

Results obtained by GEA-DT and by C4.5 are compared in Table 1. As it could be expected, global induction of decision tree allowed outperforming one of the most popular representatives of the top-down approach. It is especially visible for even problems, where C4.5 is able to only find trees degenerated to root nodes. It is worth to emphasise that decision trees proposed by GEA-DT were not only optimal or almost optimal in terms of the classification quality but also they were very compact.

Data set	GEA-DT		C4.5	
	Quality [%]	Size	Quality [%]	Size
2x2	99.9 ± 0.1	7.0 ± 0.0	50.0	1.0
3x3	99.1 ± 0.2	17.0 ± 0.0	98.6	23.0
4x4	97.9 ± 0.2	31.0 ± 0.0	50.0	1.0

Tab. 1. Results obtained for *chessboard* datasets.

EA-based methods are sometimes criticised as being too slow to be applied for really big datasets. In the last experiment, we try to verify how GDT-EA scales with the growth of learning set size. For this purpose a series of datasets with number of objects varying from 100 to 100000 corresponding to the chessboard 3x3 problem was prepared. In Figure 5 obtained results in terms of classification accuracy and computation time are depicted (on log scale). It should be noticed that even for the biggest dataset GDT-EA is able to find the optimal solution in reasonable time (approximately 1.5h).

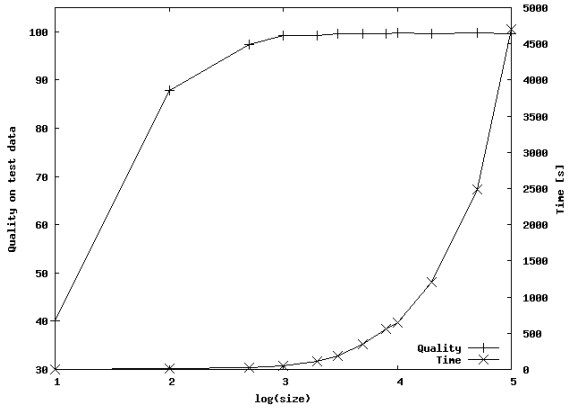

Fig. 5. Performance of GDT-EA with increasing dataset size (chessboard 3x3 domain).

3.2 Real datasets

Description of analysed real-life datasets is presented in Table 2 and results of experiments are collected in Table 3.

Name	Number of cases	Number of attributes	Number of classes
breast (bcw)	683	9	2
bupa	345	6	2
cmc	1473	9	3
iris	150	4	3

Name	Number of cases	Number of attributes	Number of classes
page-blocks	5473	10	5
pima	768	7	2
vehicle	846	18	4
wine	178	13	3

Tab. 2. Description of the real datasets.

It could be observed that results in terms of the classification quality obtained by the proposed approach and C4.5 are at least comparable. In a few domains GEA-DT performed better and in the others C4.5 was slightly more efficient. It could be also noticed that evolutionary approach was more efficient in term of the size of the classifier. Trees are significantly simpler and it is especially important in the context of understandability of discovered knowledge.

Dataset	GEA-DT		C4.5	
	Quality [%]	Size	Quality [%]	Size
breast (bcw)	95.4 ± 0.2	7.7 ± 0.1	94.9	26.0
bupa	65.6 ± 0.4	21.5 ± 0.2	64.7	44.6
cmc	54.9 ± 0.2	10.7 ± 0.2	52.2	223.8
iris	95.3 ± 0.2	8.2 ± 0.1	94.7	8.4
page-blocks	95.3 ± 0.1	8.2 ± 0.2	97.0	82.8
pima	73.8 ± 0.3	7.4 ± 0.2	74.6	40.6
vehicle	69.7 ± 0.3	29.6 ± 0.5	72.3	129.0
wine	88.1 ± 2.1	9.8 ± 0.4	85.0	9.0

Tab. 3. Results obtained for real datasets.

4 Conclusion

In the paper global approach to induction of decision trees is presented. Specialised evolutionary algorithm is proposed as an efficient search mechanism. Experimental validation shows that the proposed method is able to find accurate and very compact classifiers. Moreover, some improvement can still be obtained.

Several directions of future research exist. One of them could be more sophisticated fitness function, especially in part dealing with complexity of the classifier. We also plan to incorporate into the induction process variable misclassification costs and feature's cost, which could be especially useful in medical decision support. Another idea is an extension of the presented approach to induction of oblique trees, where not only axis-parallel tests are applied to split the data.

5 References

[1] Bennett K., “Global tree optimization: A non-greedy decision tree algorithm”, Computing Science and Statistics 26, pp.156-160, 1994.

[2] Blake C., Merz C., “UCI Repository of machine learning databases” [http://www.ics.uci.edu/~mlearn/MLRepository.html] Irvine, CA: University of California, 1998.

[3] Bobrowski L. “Piecewise-linear classifiers, formal neurons and separability of the learning sets”, Proc. of ICPR'96, IEEE CS Press, pp. 224-228, 1996.

[4] Bot M., Langdon W., “Application of genetic programming to induction of linear classification trees”, Proc. of EuroGP, LNCS 1802, pp.247-258, 2000.

[5] Breiman L., Friedman J., Olshen R., Stone C., “Classification and Regression Trees”, Wadsworth International Group, 1984.

[6] Cantu-Paz E., Kamath C., “Inducing oblique decision trees with evolutionary algorithms”, IEEE Trans. on Evol. Computation 7(1), pp. 54-68, 2003.

[7] Chai B., Huang T., Zhuang X., Zhao Y., Sklansky J., “Piecewise-linear classifiers using binary tree structure and genetic algorithm”, Pattern Recognition 29(11), pp. 1905-1917, 1996.

[8] Fayyad U., Irani K., “Multi-interval discretization of continuous-valued attributes for classification learning”, Proc. of IJCAI'93, Morgan Kaufmann, pp. 1022-1027, 1993.

[9] Fayyad U., Piatetsky-Shapiro G., Smyth P., Uthurusamy R., (eds.) Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996.

[10] Hayfil L., Rivest R., “Constructing optimal binary decision trees is NP.-complete”, Information Processing Letters 5(1), pp. 15-17, 1976.

[11] Koza J., “Concept formation and decision tree induction usisng genetic programming paradigm”, Proc. of PPSN 1, LNCS 496, pp. 124-128, 1991.

[12] Krętowski M., “An evolutionary algorithm for oblique decision tree induction”, Proc. of ICAISC’04, Springer, LNCS 3070, pp.432-437, 2004.

[13] Michalewicz Z., “Genetic Algorithms + Data Structures = Evolution Programs”, Springer, 1996.

[14] Murthy S., Salzberg S., “Decision tree induction: How effective is the greedy heuritics?”, Proc. of KDD-95, 1995.

[15] Murthy S., “Automatic construction of decision trees from data: A multi-disciplinary survey”, Data Mining and Know. Disc. 2, pp. 345-389, 1998.

[16] Quinlan J., “C4.5: Programs for Machine Learning”, Morgan Kauf., 1993.

[17] Shah S., Sastry P., “New algorithm for learning and pruning oblique decision trees”, IEEE Trans. on SMC - Part C 29(4), pp. 494-505, 1999.