A Memetic Algorithm for Global Induction of Decision Trees

Marek Krętowski

Faculty of Computer Science, Białystok Technical University
Wiejska 45a, 15-351 Białystok, Poland
mkret@wi.pb.edu.pl

Abstract. In the paper, a new memetic algorithm for decision tree learning is presented. The proposed approach consists in extending an existing evolutionary approach for global induction of classification trees. In contrast to the standard top-down methods, it searches for the optimal univariate tree by evolving a population of trees. Specialized genetic operators are selectively applied to modify both tree structures and tests in non-terminal nodes. Additionally, a local greedy search operator is embedded into the algorithm, which focusses and speeds up the evolutionary induction. The problem of over-fitting is mitigated by suitably defined fitness function. The proposed method is experimentally validated and preliminary results show that the proposed approach is able to effectively induce accurate and concise decision trees.

1 Introduction

Evolutionary Computations is the name commonly used for describing a group of optimization and search techniques inspired by the process of natural evolution. Their main advantages over greedy search methods is their ability to avoid local optima. On the other hand it is known that pure evolutionary methods are not the fastest methods and a lot of effort is put into speeding them up. One of the possible solutions is a combination of evolutionary approach with local search techniques, which is known as *Memetic Algorithms* [10]. However, designing a competent memetic algorithm for a given problem is not an easy task and a number of important issues have to be addressed (e.g. where and when local search should be applied during the evolutionary search).

In this paper, an evolutionary learning of decision trees based on the training dataset is investigated. There are two main approaches to induction of decision trees: top-down and global. In the first approach, the optimal test searches and data splitting are recursively repeated to consecutive subsets of the training data until the stoping condition is not met. Usually, the growing phase is followed by the post-pruning. Apart of the classical top-down system like *CART* [3] or *C4.5* [18], several EC-based systems which learn (mainly oblique) decision trees in the top-down manner (e.g. *BTGA* [5], *OC1-ES* [4], *DDT-EA* [11]) have been proposed so far.

V. Geffert et al. (Eds.): SOFSEM 2008, LNCS 4910, pp. 531–540, 2008.

In this paper, the second approach to decision tree induction is advocated. In contrast to the 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. This process is obviously much more computationally complex but it can reveal hidden regularities, which are almost undetectable by greedy methods

The global approach was initially proposed by Koza in [9], 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 [17] which directly evolves classification trees with nominal tests. Fu *et al.* proposed a genetic algorithm called *GAIT* [8], which evolves binary trees initially obtained by applying *C4.5* on small sub-samples of the original data. Two simple genetic operator are utilized and individual performance is judged by measuring classification accuracy on the validation set. It should be noted that only tests from initial trees can be used in the internal nodes. Another interesting global system is called *GALE* [15]. It is a fine-grained parallel evolutionary algorithm for evolving both orthogonal and oblique decision trees. *GALE* uses squared reclassification accuracy as a fitness and simple operators (one point cross-over from genetic programming and random perturbation of the test).

In the paper, for the first time a memetic algorithm is proposed for global induction of decision trees. It combines typical evolution of trees with the local search for optimal tests in non-terminal nodes. The local optimality criteria come from *CART* and *C4.5* systems. This kind of hybridization should profit from both global and greedy methods and should improve the efficiency of the search.

The rest of the paper is organized as follows. In the next section the proposed memetic algorithm for global induction of univariate decision trees is described. Experimental validation of the method on artificial and real-life data is presented in section 4. In the last section, the paper is concluded and possible future works are sketched.

2 Memetic Algorithm for Global Induction

As the presented system evolved from our previous classical evolutionary algorithm [12,13,14], the general structure of the memetic algorithm follows the standard evolutionary framework [16]. The local search component responsible of the optimal test search in internal nodes is introduced in the initialization and embedded into the mutation operator.

2.1 Representation, Initialization and Termination Condition

Representation. There are two ways of representing candidate solutions in the evolutionary search. In the first one, individuals are encoded in the fixed-size (usually binary) chromosomes and standard genetic operators can be used. The second possibility consists in applying more sophisticated representations

(e.g. variable-length) and developing specialized genetic operators. As a structure of the optimal decision tree for a given learning set is not known *a priori* it is obvious that the second approach is chosen for the global induction.

In the presented 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 (nominal or continuous valued). Additionally, in every node information about learning vectors associated with the node is stored. This enables the algorithm to perform more 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 are considered. Specialized genetic operators consider only boundary thresholds [7] 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, in which the examples belong to two different classes. All boundary thresholds for each continuous-valued attribute are calculated before starting the evolutionary induction [12]. It significantly limits the number of possible splits and focuses the search process. It should be however noted that locally applied the optimal test search can find a split, which is not based on a precalculated threshold. In all internal nodes except from the root only limited sub-sample of learning vectors can be used for the optimal test search.

Initialization. An initial population is usually randomly created with emphasis on diversity of candidate solutions, which is especially useful when large search space has to be penetrated. It is also known that proper initialization can focus and significantly speed up the search process.

In the presented system, initial individuals are created by applying the classical top-down algorithm to randomly chosen sub-samples of the original training data (10% of data, but not more then 500 examples). Additionally, for any initial tree one of five test search strategies in non-terminal nodes is applied. Three strategies come from the very well-known decision tree systems i.e. *CART* [3] and *C4.5* [18] and they are based on the corresponding optimality criteria: *GiniIndex*, *InfoGain* and *GainRatio*. The fourth strategy is dipolar [11], where a test splitting randomly selected mixed dipole (a pair of feature vectors from different classes) is found. The last strategy is a random combination of all the aforementioned strategies. The recursive partitioning is finished when all training objects in a node belong to the same class or the number of objects in a node is lower than the predefined value (default value: 5). Finally, the resulting trees are post-pruned according to the fitness function.

Termination condition. The evolution terminates when the fitness of the best individual in the population does not improve during the fixed number of generations (default value is equal 1000). This can be treated as a sign of algorithm convergence. 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

There are two specialized genetic operators corresponding to the classical mutation and cross-over. Application of both operators can result in changes of the tree structure and tests in non-terminal nodes. Additionally the local search component is built into the mutation-like operator.

Mutation operator. A mutation-like operator [14] is applied with a given probability to a tree (default value is 0.8) and it guarantees that at least one node of the selected individual is mutated. Firstly, the type of the node (leaf or internal node) is randomly chosen with equal probability and if a mutation of a node of this type is not possible, the other node type is chosen. A ranked list of nodes of the selected type is created and a mechanism analogous to ranking linear selection [16] is applied to decide which node will be affected.

While concerning internal nodes, the location (the level) of the node in the tree and the quality of the subtree starting in the considered node are taken into account. It is evident that a modification of the test in the root node affects the whole tree and has a great impact, whereas a mutation of an internal node in lower parts of the tree has only a local impact. In the proposed method, nodes on higher levels of the tree are mutated with lower probability and among nodes on the same level the number of misclassified objects by the subtree is used to sort them. Additionally, perfectly classifying nodes with only leaves as descendants and with a test composed of one feature are excluded from a ranking, because their mutation cannot improve the fitness.

As for leaves, the number of objects from other classes than the decision assigned to the leaf is used to put them in order, but homogenous leaves are not included. As a result, leaves which are worse in terms of classification accuracy are mutated with higher probability.

Modifications performed by a mutation operator depend on the node type (i.e. if the considered node is a leaf node or an internal node). For a non-terminal node a few possibilities exist:

- A completely new test can be found. With the user defined probability (default value: 0.05) a new test can be locally optimized or can be chosen to split a randomly drawn mixed dipole from the learning subset associated with the node. The local search for the optimal test can be based on the following criteria: *GiniIndex*, *InfoGain* and *GainRatio*. It should be noted that for nominal features only tests with the maximal number of outcomes (no inner disjunction) are analyzed due to the computational complexity constraints.
- The existing test can be altered by shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal features). These modifications can be purely random or can be guided by dipolar principles of splitting mixed dipoles and avoiding to split pure ones.
- A test can be replaced by another test or tests can be interchanged,
- One sub-tree can be replaced by another sub-tree from the same node.
- A node can be transformed (pruned) into a leaf.

Modifying a leaf makes sense only if it contains objects from different classes. The leaf is transformed into an internal node and a new test is chosen in the aforementioned way.

Cross-over operator. There are also several variants of cross-over operators. Three of them start with selecting of cross-over positions in two affected individuals. One node is randomly chosen in each of two trees. In the most straightforward variant, the subtrees starting in the selected nodes are exchanged. This corresponds to the classical cross-over from genetic programming. In the second variant, which can be applied only when non-internal nodes are randomly chosen and the numbers of outcomes are equal, only tests associated with the nodes are exchanged. The third variant is also applicable only when non-internal nodes are drawn and the numbers of descendants are equal. Branches which start from the selected nodes are exchanged in random order. There is also a variant of crossover inspired by the dipolar principles. In the internal node in the first tree a cut mixed dipole is randomly chosen and for the cross-over the node with the test splitting this dipole is selected in the second tree.

Additional operations. The application of any genetic operator can result in a necessity for relocation of the input vectors between parts of the tree rooted in the modified node. Additionally the local maximization of the fitness is performed by pruning lower parts of the sub-tree on the condition that it improves the value of the fitness.

It was observed by Bennett *et al.* [2] that in oblique trees enlarging the margin, it is profitable in terms of classification accuracy. In the presented system, a simple mechanism called *centering* based on this observation is introduced and it is applied to the best decision tree found. In case of an inequality test, the threshold can also be shifted to half-distance between corresponding feature values. It should be noted that such a post-processing does not change the fitness corresponding to the final tree. The centering cannot be applied to tests based on nominal features. For them, another kind of test improvement is used. If there is an internal node with nominal test, and there are descendant leaves which have the same decision, then such leaves are merged and inner disjunction is used in the splitting node.

2.3 Selection

As a selection mechanism the ranking linear selection [16] is applied. Additionally, the chromosome with the highest value of the fitness function in the iteration is copied to the next population *(elitist strategy)*.

2.4 Fitness Function

A fitness function drives the evolutionary search process and is the most important and sensitive component of the algorithm. The goal of any classification system is the correct prediction of class labels of new objects, however such a

target function cannot be defined directly. Instead, the accuracy on the training data is often used. However, it is well-known that their direct optimization leads to an over-fitting problem. In a typical top-down induction of decision trees, the over-specialization problem is mitigated by defining a stopping condition and by applying a post-pruning [6].

In the presented approach a complexity term is introduced into the fitness function preventing the over-specialization. The fitness function, which is maximized, has the following form:

$$Fitness(T) = Q_{Reclass}(T) - \alpha \cdot (S(T) - 1), \tag{1}$$

where $Q_{Reclass}(T)$ is the re-classification quality, $S(T)$ is the size of the tree T expressed as the number of nodes and α is a relative importance of the complexity term (default value is 0.001) and a user supplied parameter. Subtracting 1.0 eliminates the penalty when the tree is composed of only one leaf (in majority voting). It is worth to mention that the equation (1) is a form of regularization with $S(T) - 1$ playing the role of a stabilizer and α the role of a regularization parameter.

It is rather obvious that there is no optimal value of α for all possible datasets. When the concrete problem is analyzed, tuning this parameter may lead to the improvement of the results (in terms of classification accuracy or classifier complexity).

3 Experimental Results

The proposed memetic approach (denoted as *GDT-MA*) to learning decision trees is assessed on both artificial and real life datasets and is compared to the well-known $C4.5$ system. It is also compared to the pure evolutionary versions of the global inducer - *GDT-AP*. All prepared artificial datasets comprise training and testing parts. Examples of artificial datasets are presented in Fig. 1. In case of data from a UCI repository [1] for which testing data are not provided, a 10-fold stratified cross-validation was employed. Each experiment on evolutionary algorithms was performed 10 times and the average result of such an evaluation was presented. All systems were tested with a default set of parameters.

3.1 Artificial Datasets

Results of experiments with artificial datasets are gathered in the Table 1. For all domains *GDT-MA* and *GDT-AP* performed very well, both in terms of classification accuracy and tree complexity. Compared to the *C4.5* system both global inducers were able to find a proper decision trees when top-down system failed and returned a default class.

3.2 Real-Life Datasets

Results obtained for the real-life datasets are gathered in Table 2. It can be observed that in terms of the classification accuracy *GDT-MA* performs comparable to *C4.5* (for certain datasets it is slightly better for other is slightly

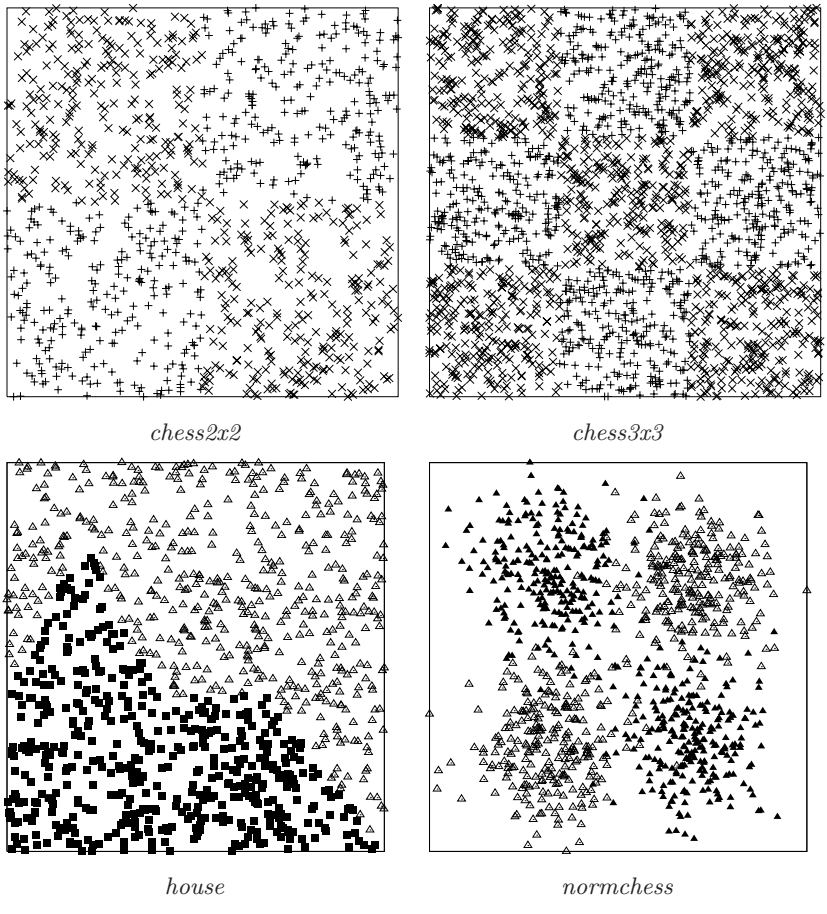

Fig. 1. Examples of artificial datasets

Table 1. Results on artificial data

	C4.5		GDT-MA		GDT-AP	
Dataset	size	quality	size	quality	size	quality
chess2x2	1	50	4	99.9	4	99.8
chess2x2x2	1	50	8	99.8	8	99.7
chess3x3	9	99.7	9	99.8	9	99.7
chess3x3x3	54	99.3	27.2	99.0	27.1	98.9
house	21	97.4	12.1	96.4	13.3	96.6
normchess	1	50	4.1	95.5	4.2	95.5
normwave	15	94	8.8	92.6	9.1	93.5

worse than its competitor). However, it is easily noticeable that in terms of the simplicity of the solution, the proposed memetic algorithm is significantly better

Table 2. Results on UCI datasets

Dataset	C4.5		GDT-MA		GDT-AP	
	size	quality	size	quality	size	quality
balance-scale	57	77.5	20.8	79.8	32.8	78.2
bcw	22.8	94.7	5.7	95.6	6.6	95.8
bupa	44.6	64.7	33.6	63.7	69.3	62.8
cars	31	97.7	3	97.9	4	98.7
cmc	136.8	52.2	19.2	55.7	13.1	53.8
german	77	73.3	18.4	74.2	16.5	73.4
glass	39	62.5	35.3	66.2	40.4	63.6
heart	22	77.1	29	76.5	44.9	74.2
page-blocks	82.8	97	7.4	96.5	7.5	96.4
pima	40.6	74.6	14.8	74.2	14.3	73.8
sat	435	85.5	18.9	83.8	19.2	83
vehicle	138.6	72.7	43.2	71.1	45.1	70.3
vote	5	97	10.9	96.2	13.5	95.6
waveform	107	73.5	30.7	71.9	36.2	72.3
wine	9	85	5.1	88.8	5.2	86.3

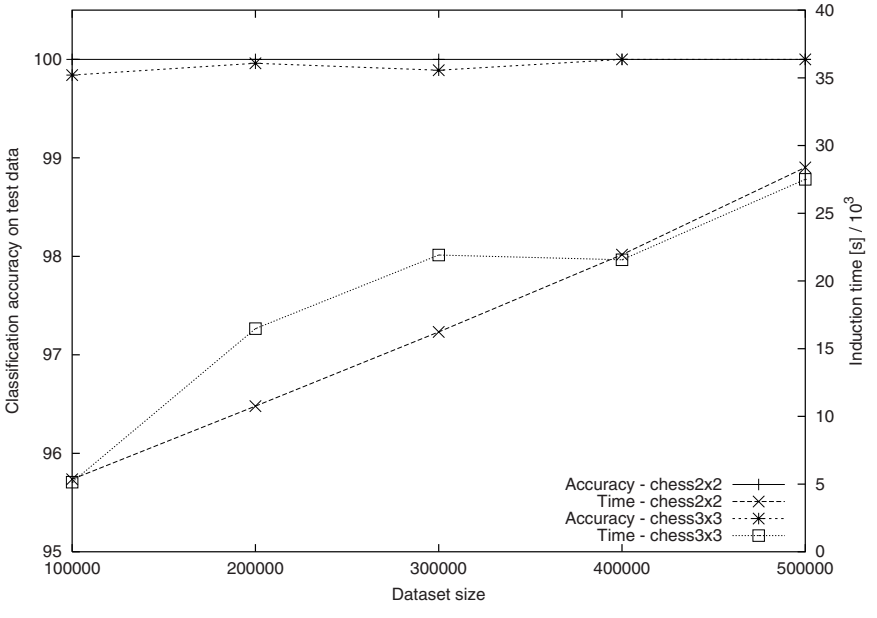

Fig. 2. Performance of the memetic algorithm on large datasets

that *C4.5*. It is also worth to mention that *GDT-MA* was more accurate than its pure evolutionary rival for 12 out of 15 analyzed real-life datasets.

3.3 Evaluation of Algorithm Performance on Large Datasets

In order to verify that the proposed method can be applied to large datasets, a performance test is conducted. The experiment was performed on two variants of the *chess* dataset: *ches2x2* and *chess3x3*, with increasing number of generated observations (starting from 100000 learning vectors up to 500000). In Fig. 2 obtained results in terms of the classification accuracy and the induction time are presented.

The promising outcome of this experiment is that it shows that the $GDT-MA$ system can deal with relatively large datasets (500000 observations) in acceptable time - 7 hours as measured on a typical machine (Xeon 3.2GHz, 2GB RAM). It should be noticed that for all datasets, optimal trees were found, both in terms of the classification accuracy and the tree size. It can be also observed that induction times scale almost linearly with the dataset size.

4 Conclusions

In the paper, for the first time a specialized memetic algorithm is developed for global induction of decision trees. The local search for optimal tests in non-terminal nodes based on the classical optimality criteria is embedded into the evolutionary search process. The necessary modification encompasses the initialization and the mutation operator. Even preliminary experimental validation shows that such a hybridization is profitable and improves the efficiency of the evolutionary induction.

The presented approach is still under development. First of all, the influence of the local search operator on the performance of the global inducer must be studied in more details. Furthermore, additional optimality criteria (e.g. *TwoingRule* from the *CART* system) are planned to be implemented.

Acknowledgments

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

References

1. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
2. Bennett, K., Cristianini, N., Shave-Taylor, J., Wu, D.: Enlarging the margins in perceptron decision trees. Machine Learning 41, 295–313 (2000)
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
4. Cantu-Paz, E., Kamath, C.: Inducing oblique decision trees with evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(1), 54–68 (2003)

5. Chai, B., Huang, T., Zhuang, X., Zhao, Y., Sklansky, J.: Piecewise-linear classifiers using binary tree structure and genetic algorithm. Pattern Recognition 29(11), 1905–1917 (1996)
6. Esposito, F., Malerba, D., Semeraro, G.: A comparative analysis of methods for pruning decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(5), 476–491 (1997)
7. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. of IJCAI 1993, pp. 1022–1027. Morgan Kaufmann, San Francisco (1993)
8. Fu, Z., Golden, B., Lele, S., Raghavan, S., Wasil, E.: A genetic algorithm-based approach for building accurate decision trees. INFORMS Journal on Computing 15(1), 3–22 (2003)
9. Koza, J.: Concept formation and decision tree induction using genetic programming paradigm. In: Schwefel, H.-P., Männer, R. (eds.) PPSN I. LNCS, vol. 496, pp. 124–128. Springer, Heidelberg (1991)
10. Krasnogor, N., Smith, J.E.: A tutorial for competent memetic algorithms: model, taxonomy and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
11. Krętowski, M.: An evolutionary algorithm for oblique decision tree induction. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 432–437. Springer, Heidelberg (2004)
12. Krętowski, M., Grześ, M.: Global learning of decision trees by an evolutionary algorithm. In: Information Processing and Security Systems, pp. 401–410. Springer, Heidelberg (2005)
13. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 400–409. Springer, Heidelberg (2006)
14. Krętowski, M., Grześ, M.: Evolutionary induction of mixed decision trees. International Journal of Data Warehousing and Mining 3(4), 68–82 (2007)
15. Llora, X., Garrell, J.: Evolution of decision trees. In: Proc. of CCAI 2001, pp. 115–122. ACIA Press (2001)
16. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
17. Papagelis, A., Kalles, D.: Breeding decision trees using evolutionary techniques. In: Proc. of ICML 2001, pp. 393–400. Morgan Kaufmann, San Francisco (2001)
18. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)