An Evolutionary Algorithm for Global Induction of Regression Trees

Marek Krętowski and Marcin Czajkowski

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45a, 15-351 Białystok, Poland
{m.kretowski,m.czajkowski}@pb.edu.pl

Abstract. In the paper a new evolutionary algorithm for induction of univariate regression trees is proposed. In contrast to typical top-down approaches it globally searches for the best tree structure and tests in internal nodes. The population of initial trees is created with diverse top-down methods on randomly chosen sub-samples of the training data. Specialized genetic operators allow the algorithm to efficiently evolve regression trees. The complexity term introduced in the fitness function helps to mitigate the over-fitting problem. The preliminary experimental validation is promising as the resulting trees can be significantly less complex with at least comparable performance to the classical top-down counterpart.

1 Introduction

Regression is beside classification the most common predictive task in data mining applications [4]. It relies on associating numeric outcomes to objects based on their feature values. Regression trees are now popular alternatives to classical statistical techniques like standard regression or logistic regression [6]. The popularity of the tree-based approaches can be explained by their ease of application, fast operation and what may be the most important, their effectiveness. Furthermore, the hierarchical tree structure, where appropriate tests from consecutive nodes are sequentially applied, closely resembles a human way of decision making which makes regression trees natural and easy to understand even for not experienced analyst.

There exists a lot of regression tree systems. One of the first solutions was presented in the seminal book describing the *CART* system [2]. *CART* uses the sum of squared residuals as an impurity function and builds a piecewise-constant model with each terminal node fitted by the training sample mean. A lot of effort was then placed in developing model trees, which extend standard regression trees by replacing in leaves single predicted values by more advanced models (e.g. linear). *M5* [13] proposed by Quinlan, *RT* [14] developed by Torgo or *SECRET* [3] are good examples of such model trees. Recently, another model tree system *SMOTI* [11] was introduced and it enables associating multiple linear

L. Rutkowski et al. (Eds.): ICAISC 2010, Part II, LNAI 6114, pp. 157–164, 2010.

model with a leaf by combining straight-line regressions reported along the path from the root.

It should be noticed that all aforementioned systems induce regression trees in a top-down manner. Starting from the root node they search for the locally optimal split (test) according to the given optimality measure and then the training data is redirected to newly created nodes. This procedure is recursively repeated until the stopping criteria is not met. Finally, the post-pruning is applied to improve the generalization power of the predictive model. Such a greedy technique is fast and generally efficient in many practical problem, but obviously does not guarantee the globally optimal solution. It can be expected that more global induction could be more adequate in certain situations.

In this paper we want to investigate a global approach to regression tree induction based on a specialized evolutionary algorithm. In our previous works we showed that evolutionary inducers are capable to efficiently induce various types of classification trees: univariate [8], oblique [9] and mixed [10]. In this paper we want to show that similar approach can be applied to obtain accurate and compact regression trees.

The rest of the paper is organized as follows. In the next section a new evolutionary algorithm for global induction of univariate regression trees is described. Experimental validation of the proposed approach on artificial and real-life data is presented in section 3. In the last section, the paper is concluded and possible future works are sketched.

2 Evolutionary Induction of Regression Trees

The general structure of the system follows a typical framework of evolutionary algorithms [12] with an unstructured population and a generational selection.

2.1 Representation, Initialization and Termination Condition

Representation. In the presented system, regression trees are represented in their actual form as classical univariate trees[1]. 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. Every leaf is associated with the predicted value estimated as a mean of dependent variable values from training objects in this leaf.

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 applied. As potential splits only precalculated candidate thresholds are considered. A candidate threshold for the given attribute is defined as a midpoint between such a successive pair

[1] Tree-based representation is typical for genetic programming and for the first idea of evolving decision trees was proposed by Koza [7].

of examples in the sequence sorted by the increasing value of the attribute, in which the examples are characterized by different predicted values. Such a solution significantly limits the number of possible splits and focuses the search process.

Initialization. In the proposed approach, initial individuals are created by applying the classical top-down algorithm to randomly chosen sub-samples of the original training data (arbitrary default: 10% of data, but not more than 500 examples). Additionally, for every initial tree one of three test search strategies in non-terminal nodes is applied. Two strategies come from the very well-known regression tree systems i.e. *CART* [2] and *M5* [13] and they are based on *Least Squares* or *Least Absolute Deviation*. The last strategy is called *dipolar*, where a pair of feature vectors (dipole) is selected and then a test is constructed which splits this dipole. Selection of the dipole is randomized but longer (with bigger difference between dependent variable values) dipoles are preferred and mechanism similar to the ranking linear selection [12] is applied. The recursive partitioning is finished when all training objects in a node are characterized by the same predicted value, the number of objects in a node is lower than the predefined value (default value: 5) or the maximum tree depth is reached (default value: 10).

Termination condition. The evolution terminates classically when the fitness of the best individual in the population does not improve during the fixed number of generations [12] (default value is equal 1000). Additionally maximum number of generations is specified, which allows limiting the computation time in case of a slow convergence (default value: 5000).

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. After applying any operator it is usually necessary to relocate learning vectors between parts of the tree rooted in the altered node. This can cause that certain parts of the tree does not contain any learning vectors and has to be pruned.

Mutation operator. A mutation-like operator 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 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 modification of the test in the root node affects whole tree and has

a great impact, whereas 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 absolute error calculated on the learning vectors located in the subtree is used to sort them. As for leaves, only absolute error is used to put them in order, but homogenous leaves are not included. As a result, leaves which are worse in terms of accuracy are mutated with higher probability.

Modifications performed by 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 by means of the dipolar method used for the initialization;
- The existing test can be altered by shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal features);
- 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 with different dependent variable values. The leaf is transformed into an internal node and a new test is chosen in the aforementioned way.

Cross-over operator. There are also three variants of recombination. All 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.

2.3 Selection

As a selection mechanism the ranking linear selection is applied. Additionally, the individual 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 very important and sensitive component of the algorithm. When concerning any prediction task it is well-known that the direct minimization of the prediction error measured on the learning set leads to an over-fitting problem. In a typical top-down induction

of decision trees, the over-specialization problem is partially mitigated by defining a stopping condition and by applying a post-pruning. In our approach, the search for an optimal structure is embedded into the evolutionary algorithm by incorporating a complexity term into the fitness. This term works as a penalty for increasing the tree size.

The fitness function is minimized and has the following form:

$$Fitness(T) = \frac{MAE(T)}{MAE_{Max}} + \alpha \cdot (S(T) - 1.0), \tag{1}$$

where $MAE(T)$ - *Mean Absolute Error* of the tree T measured on the learning set and $S(T)$ - tree size. Subtracting 1.0 eliminates the penalty when the tree is composed of only one leaf. MAE_{Max} is the maximal $MAE(T)$ for the learning set. It can be easily calculated, as it is observed when the tree is composed of only one leaf. For any (more complex) tree T $MAE(T) \leq MAE_{Max}$. α is the relative importance of the complexity term (default value is 0.001) and a user supplied parameter.

It should be noticed that there is no optimal value of α for all possible datasets and tuning it may lead to the improvement of results for the specific problem.

3 Experimental Validation

Two groups of experiments are performed to validate the global approach to induction of regression trees (denoted in tables as *GRT*). For the purpose of comparison, results obtained by the classical top-down inducer *REPTree*, which is publicly available in the *Weka* system [5], are also presented. *REPTree* builds a regression tree using variance and prunes it using reduced-error pruning (with backfitting). Both systems were run with default values of parameters. All results presented in the table correspond to averages of 10 runs and were obtained by using test sets (when available) or by 10-fold cross-validation. The average number of nodes is given as a complexity measure of regression trees.

Synthetical datasets. In the first group, two simple artificially generated datasets with analytically defined decision borders are analyzed. Both datasets contain two independent features and one dependent feature with only a few distinct values. Number of feature vectors in the learning sets is 1000.

In figure 1 three-dimensional visualization of the datasets with the corresponding regression trees generated by the global method are presented. It should be noticed that in both cases trees obtained by *GRT* have optimal structure (only 9 and 4 leaves correspondingly) and gain very small error (RMSE equal to 0.139 for *chess* and 0.102 for *armchair*) on the testing set.

For the top-down inducer both problem are too difficult. For the first dataset *REPTree* generates an overgrown tree with 21 leaves and as a result the testing error is significantly higher (0.288). For the *armchair* problem the solution found by the *REPTree* is also not completely optimal (6 leaves and error=0.149), as the first split ($x < 2.02$) is not the best.

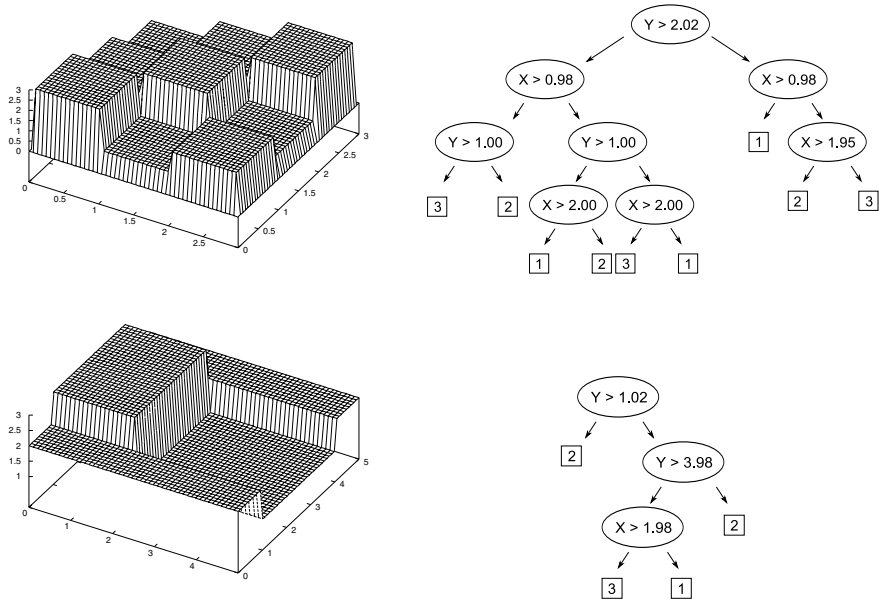

Fig. 1. Examples of artificial datasets (*chess* - top, *armchair* - bottom) and the corresponding regression trees generated by *GRT*

Real-life datasets. In the second series of experiments, several datasets taken from UCI Machine Learning Repository [1] or provided by L. Torgo on his website are analyzed to assess the performance of the proposed system in solving real-life problems. Table 1 presents characteristics of investigated datasets and obtained results.

It can be observed that the prediction accuracy of both analyzed system is comparable. It should be however noticed that globally induced trees can be significantly less complex. It is especially visible for the biggest trees generated by *REPTree*, where *GRT* is able to find smaller solutions (eg. 67.2 nodes as opposed to 819 nodes or 53 to 553).

It should be underline that the results collected in Table 1 were obtained with the default value of α parameter. In order to verify the impact of this parameter on the results, a series of experiments with varying α was prepared (see Fig. 2) on two exemplar datasets. As it could be expected, along with a decrease of α an increase of trees complexity can be observed. As for RMSE after initial quick decrease the error rate begins to rise slightly. It can be also observed that for both datasets the default setting of this parameter is not really optimal and smaller errors can easily obtained.

Table 1. Characteristics of the real-life datasets (number of objects/number of numeric features/number of nominal features) and obtained results. Root mean squared error (RMSE) is given as the error measure and number of nodes as the tree size.

		GRT		*REPTree*	
Dataset	Properties	RMSE	Tree size	RMSE	Tree size
Abalone	4177/7/1	2.30	49.3	2.36	201
Ailerons	13750/40/0	0.00022	53.3	0.00020	553
Auto-Mpg	392/4/3	3.59	145.6	3.65	94
Auto-Price	159/17/10	2505	91.2	2760	32
Delta Ailerons	7129/6/0	0.000182	29.8	0.000175	291
Delta Elevators	9517/6/0	0.00156	25.0	0.0015	319
Elevators	16559/40/0	0.0044	65.9	0.0040	503
Housing	506/14/0	4.29	88.6	4.84	41
Kinematics	8192/8/0	0.193	67.2	0.191	819
Machine CPU	209/7/0	60.4	75.8	92.3	15
Pole	15000/48/0	10.23	47.8	8.26	223
Pyrimidines	74/28/0	0.101	81.5	0.136	1
Stock	950/10/0	1.317	59.2	1.186	137
Triazines	186/61/0	0.149	159.0	0.152	7

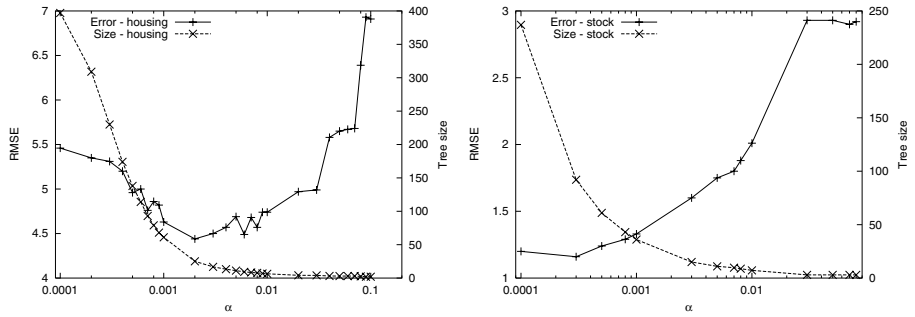

Fig. 2. Influence of α parameter on the performance of the *GRT* algorithm on two datasets

4 Conclusion

In the paper a new global approach to regression tree learning is presented. In contrast to classical top-down inducers, where locally optimal tests are sequentially chosen, both the tree structure and tests in internal nodes are searched in the same time by specialized evolutionary algorithm. This way the inducer is able to avoid local optima and to generate better predictive model. Even preliminary experimental results show that the globally evolved regression trees could be competitive compared to the top-down based counterparts, especially in term of tree size.

The presented approach is constantly improved and currently we are working on introducing oblique tests in the non-terminal nodes. On the other hand, we plan to extend the knowledge representation by evolving model trees.

Acknowledgments. This work was supported by the grant W/WI/5/10 from Bialystok University of Technology.

References

1. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
3. Dobra, A., Gehrke, J.: SECRET: A scalable linear regression tree algorithm. In: Proc. KDD 2002 (2002)
4. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (1996)
5. Frank, E., et al.: Weka 3 - Data Mining with Open Source Machine Learning Software in Java. University of Waikato (2000), http://www.cs.waikato.ac.nz/~ml/weka
6. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of statistical Learning. Data Mining, Inference, and Prediction, 2nd edn. Springer, Heidelberg (2009)
7. Koza, J.: Concept formation and decision tree induction using genetic programming paradigm. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 124–128. Springer, Heidelberg (1991)
8. 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)
9. Krętowski, M., Grześ, M.: Evolutionary learning of linear trees with embedded feature selection. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 400–409. Springer, Heidelberg (2006)
10. Krętowski, M., Grześ, M.: Evolutionary induction of mixed decision trees. International Journal of Data Warehousing and Mining 3(4), 68–82 (2007)
11. Malerba, D., Esposito, F., Ceci, M., Appice, A.: Top-down induction of model trees with regression and splitting nodes. IEEE Trans. on PAMI 26(5), 612–625 (2004)
12. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
13. Quinlan, J.: Learning with continuous classes. In: Proc. AI 1992, pp. 343–348. World Scientific, Singapore (1992)
14. Torgo, L.: Inductive learning of tree-based regression models. Ph.D. Thesis, University of Porto (1999)