An Evolutionary Algorithm for Global Induction of Regression Trees with Multivariate Linear Models

Marcin Czajkowski and Marek Kretowski

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

Abstract. In the paper we present a new evolutionary algorithm for induction of regression trees. In contrast to the typical top-down approaches it globally searches for the best tree structure, tests at internal nodes and models at the leaves. The general structure of proposed solution follows a framework of evolutionary algorithms with an unstructured population and a generational selection. Specialized genetic operators efficiently evolve regression trees with multivariate linear models. Bayesian information criterion as a fitness function mitigate the over-fitting problem. The preliminary experimental validation is promising as the resulting trees are less complex with at least comparable performance to the classical top-down counterpart.

Keywords: model trees, evolutionary algorithms, multivariate linear regression, BIC.

1 Introduction

The most common predictive tasks in data mining applications are classification and regression [5]. One of the most widely used prediction techniques are decision trees [19]. Regression and model trees may be considered as a variant of decision trees, designed to approximate real-valued functions instead of being used for classification tasks. Main difference between a typical regression tree and a model tree is that, for the latter, terminal node is replaced by a regression plane instead of a constant value. Those tree-based approaches are now popular alternatives to classical statistical techniques like standard regression or logistic regression.

In this paper we want to investigate a global approach to model tree induction based on a specialized evolutionary algorithm. This solution extends our previous research on evolutionary classification and regression trees which showed that a global induction could be more adequate in certain situations. Our work covers the induction of univariate trees with multivariate linear models at the leaves.

1.1 Global Versus Local Induction

Linear regression is a global model in which the single predictive function holds over the entire data-space [9]. However many regression problems cannot be

M. Kryszkiewicz et al. (Eds.): ISMIS 2011, LNAI 6804, pp. 230–239, 2011.

solved by a single regression model especially when the data has many attributes which interact in a complicated ways. Recursively partitioning the data and fitting local models to the smaller regions, where the interactions are more simple, is a good alternative to complicated, nonlinear regression approaches. The recursive partitioning [16] may be realized by top-down induced regression trees. 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 are met and in each of the terminal node called leaf, a locally optimal model is built for each region. Finally, the post-pruning is applied to improve the generalization power of the predictive model. Such a technique is fast and generally efficient in many practical problem, but obviously does not guarantee the globally optimal solution. Due to the greedy nature, algorithms may not generate the smallest possible number of rules for a given problem [17] and a large number of rules results in decreased comprehensibility. Therefore, in certain situations more global approach could lead to improvement in prediction and size of the resulting models.

1.2 Related Work

The *CART* system [2] is one of most known top-down induced prediction tree. The *CART* algorithm finds a split that minimizes the Residual Sum of Squares (RSS) of the model when predicting. Next, it builds a piecewise constant model with each terminal node fitted by the training sample mean. The *CART* algorithm was later improved by replacing single predicted values in the leaves by more advanced models like in *SECRET* [4] or *RT* [21]. The most popular system which induce top-down model tree is *M5* [23]. Like *CART*, it builds tree-based models but, whereas regression trees have values at their leaves, the tree constructed by *M5* can have multivariate linear models analogous to piecewise linear functions.

One of the first attempts to optimize the overall RSS was presented in *RETRIS* [8] model tree. Algorithm simultaneously optimized the split and the models at the terminal nodes to minimize the global RSS. However *RETRIS* is not scalable and does not support larger datasets because of the huge complexity [17]. More recent solution called *SMOTI* [14] allows regression models to exist not only in leaves but also in the upper parts of the tree. Authors claim that this allows for individual predictors to have both global and local effects on the model tree.

Our previously performed research showed that evolutionary inducers are capable to efficiently induce various types of classification trees: univariate [10], oblique [11] and mixed [12]. In our last papers we applied a similar approach to obtain accurate and compact regression trees [13] and we did preliminary experiments with the model trees that have simple linear regression models at the leaves [3].

Proposed solution denoted as *GMT* improved our previous work: starting with more heterogenous population, additional genetic operators and a new fitness

function based on Bayesian information criterion (*BIC*) [20]. The models at the leaves were extended from simple linear to multivariate linear regression models.

2 An Evolutionary Induction of Model Trees

Structure of the proposed solution follows a typical framework of evolutionary algorithms [15] with an unstructured population and a generational selection.

2.1 Representation

Model trees are represented in their actual form as typical univariate trees, similarly as in our previous work [3]. Each test in a non-terminal node concerns only one attribute (nominal or continuous valued). In case of a continuous-valued feature typical inequality tests are applied. As for potential splits only the precalculated candidate thresholds are considered. A candidate 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 are characterized by different predicted values. Such a solution significantly limits the number of possible splits. For a nominal attribute at least one value is associated with each branch. It means that an inner disjunction is built into the induction algorithm.

At each leaf a multivariate linear model is constructed using standard regression technique [18] with cases and feature vectors associated with that node. A dependent variable y is now explained not by single variable like in [3] but a linear combination of multiple independent variables $x_1, x_2, \ldots, x_p$:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_p * x_p \tag{1}$$

where p is the number of independent variables, x_i are independent variables, β_i are fixed coefficients that minimizes the sum of squared residuals of the model.

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.

2.2 Initialization

The initial tree construction is similar to the typical approaches like *CART* and *M5*. At first, we construct a standard regression tree with local means of dependent variable values from training objects in every leaf. Initial individuals are created by applying the classical top-down algorithm. The recursive partitioning is finished when all training objects in node are characterized by the same predicted value (or it varies only slightly [23]) or the number of objects in a node is lower than the predefined value (default value: 5). Additionally, user can set the maximum tree depth (default value: 10). Next, a multivariate linear model is built at each terminal node.

An appropriate trade off between a degree of heterogeneity and a computation time is obtained by various data-driven manners for selecting attributes and choosing search strategies in non-terminal nodes:

- initial individuals are induced from randomly chosen subsamples of the original training data (10% of data, but not more than 500 examples);
- each individual searches splitting tests from randomly chosen subsamples of the attribute set (50% of attributes);
- one of three test search strategies in non-terminal nodes is applied [3]:
  - *Least Squares (LS)* reduction,
  - *Least Absolute Deviation (LAD)* reduction,
  - *dipolar*, where a dipol (a pair of feature vectors) 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 [15] is applied.

2.3 Genetic Operators

Like in our previous papers [3][13] we have applied two specialized genetic operators corresponding to the classical mutation and cross-over. Both operators affect the tree structure, tests in non-terminal nodes and models at leaves. After each evolutionary iteration 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.

Cross-over. Cross-over solution starts with selecting positions in two affected individuals. In each of two trees one node is chosen randomly. We have proposed three variants of recombination:

- subtrees starting in the selected nodes are exchanged,
- tests associated with the nodes are exchanged (only when non-terminal nodes are chosen and the number of outcomes are equal),
- branches which start from the selected nodes are exchanged in random order (only when non-terminal nodes are chosen and the number of outcomes are equal).

Mutation. Mutation solution starts with randomly choosing the type of node (equal probability to select leaf or internal node). Next, the ranked list of nodes of the selected type is created and a mechanism analogous to ranking linear selection [15] is applied to decide which node will be affected. Depending on the type of node, ranking take into account:

- absolute error - worse in terms of accuracy leaves and internal nodes are mutated with higher probability (homogenous leaves are not included),

– location (level) of the internal node in the tree - 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. Therefore, internal nodes in lower parts of the tree are mutated with higher probability.

We have proposed new mutation operators for internal node:

– tests between father and son exchanged,
– symmetric mutation between sub-trees,
– test in node changed by: new random one or new dipolar (described in section 2.2),
– shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal features),
– node can be transformed (pruned) into a leaf,

and for the leaves:

– transform leaf into an internal node with a new dipolar test,
– extend linear model by adding new randomly chosen attribute,
– simplify linear model by removing the randomly chosen attribute,
– change linear model attributes with random ones,
– delete from linear model the least important attribute.

After performed mutation in internal nodes the models in corresponding leaves are not recalculated because adequate linear models can be found while performing the mutations at the leaves. Modifying and recalculating leaf model makes sense only if it contains objects with different dependent variable values or different independent variables that build the linear model.

2.4 Selection and Termination Condition

Evolution terminates when the fitness of the best individual in the population does not improve during the fixed number of generations. In case of a slow convergence, maximum number of generations is also specified, which allows us to limit computation time.

Ranking linear selection [15] is applied as a selection mechanism. Additionally, in each iteration, single individual with the highest value of fitness function in current population in copied to the next one *(elitist strategy)*.

2.5 Fitness Function

A fitness function drives evolutionary search process and therefore is one of the most important and sensitive component of the algorithm. Direct minimization of the prediction error measured on the learning set usually leads to the overfitting problem. In a typical top-down induction of decision trees, this problem is partially mitigated by defining a stopping condition and by applying a post-pruning.

In our previous works [3] we used Akaike's information criterion (*AIC*) [1] as the fitness function. Performed experiments suggested that penalty for increasing model size should depend on the number of observations in the data. Therefore we have replaced *AIC* with the Bayesian information criterion (*BIC*) [20] that is given by:

$$Fit_{BIC}(T) = -2 * ln(L(T)) + ln(n) * k(T) \tag{2}$$

where $L(T)$ is the maximum of the likelihood function of the tree T, $k(T)$ is the number of model parameters and n is the number of observations. The log(likelihood) function $L(T)$ is typical for regression models [7] and can be expressed as:

$$ln(L(T)) = -0.5n * [ln(2\pi) + ln(SS_e(T)/n) + 1] \tag{3}$$

where $SS_e(T)$ is the sum of squared residuals of the tree T. The term $2 * k(T)$ can also be viewed as a penalty for over-parametrization and has to include not only the tree size but also the number of attributes that build models at the leaves. The number of independent parameters $k(T)$ in the complexity penalty term is equal $2 * (Q(T) + M(T))$ where $Q(T)$ is the number of nodes in model tree T and $M(T)$ is the sum of all attributes in the linear models at the leaves.

3 Experimental Validation

In this section, we study the predictive accuracy and size of the proposed approach (denoted as *GMT*) to other methods. Validation was performed on synthetical and real-life datasets. Since our algorithm induces model trees we have compared it against the popular *M5* [23] counterpart. The *M5* algorithm has the same tree structure: univariate splits and multivariate linear models at the leaves, as the *GMT*. The most important difference between both solution is the tree construction where the *M5* is a traditional greedy top-down inducer and the *GMT* approach searches for optimal trees in a global manner by using an evolutionary algorithm. We also included results obtained by the *REPTree* which is another classical top-down inducer. *REPTree* builds a regression tree using variance and prunes it using reduced-error pruning (with backfitting). Both comparative algorithms are run using the implementations in WEKA [6], software that is publicly available.

Each tested algorithm run with default values of parameters through all datasets. All presented results correspond to averages of 20 runs and were obtained by using test sets (when available) or by 10-fold cross-validation. Root mean squared error (RMSE) is given as the error measure of the algorithms. The number of nodes is given as a complexity measure (size) of regression and model trees.

3.1 Synthetical Datasets

In the first group of experiments, two simple artificially generated datasets with analytically defined decision borders are analyzed. Both datasets contain a

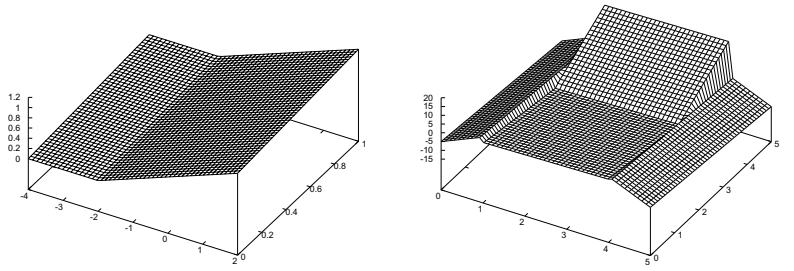

Fig. 1. Examples of artificial datasets (*split plane* - left, *armchair3* - right)

feature that is linearly dependent with one of two independent features. One thousand observations for each dataset were divided into a training set (33.3% of observations) and testing set (66.7%).

The artificial dataset *split plane* that is illustrated in the Fig. 1 can be perfectly predictable with regression lines on subsets of the data resulting from a single partition. The equation is:

$$y(x_1, x_2) = \begin{cases} 0.2 * x_2 & x_1 < -2 \\ 0.25 * x_1 + 0.2 * x_2 + 0.5 & x_1 \geq -2 \end{cases} \tag{4}$$

The test in the root node for both greedy top-down inducers is not optimal. *M5* approach minimizes the combined standard deviation of both partitions of each subset and sets first split at threshold $x_1 = -1.18$. *REPTree* is using the CART approach, partitions this dataset at $x_1 = -0.44$ minimizing the RSS and has size equal 88. *GMT* partitions the data at threshold $x_1 = -2.00$ because it is able to search globally for the best solution. This simple artificial problem illustrates general advantage of the global search solution to greedy algorithms. The induced *GMT* and *M5* trees are illustrated in Figure 2 and the Table 1 presents the generated multivariate linear models at the leaves.

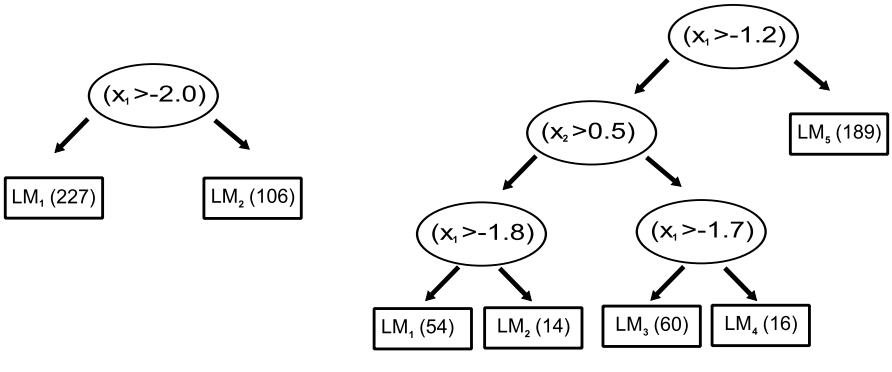

Fig. 2. Examples of model trees for *split plane* (GMT - left, M5 - right)

Table 1. Generated multivariate linear models for *GMT* and *M5*

GMT	LM_1: $y(x_1, x_2) = 0.25 * x_1 + 0.2 * x_2 + 0.5$
	LM_2: $y(x_1, x_2) = 0.2 * x_2 + 0.5$
M5	LM_1: $y(x_1, x_2) = 0.1865 * x_2 + 0.0052$
	LM_2: $y(x_1, x_2) = 0.25 * x_1 + 0.2 * x_2 + 0.5$
	LM_3: $y(x_1, x_2) = 0.1936 * x_2 + 0.0079$
	LM_4: $y(x_1, x_2) = 0.25 * x_1 + 0.2 * x_2 + 0.5$
	LM_5: $y(x_1, x_2) = 0.25 * x_1 + 0.2 * x_2 + 0.5$

Illustrated in the Fig. 1 dataset *Armchair3* is more complex than *split plane*. Many traditional approaches will fail to efficiently split the data as the greedy inducers search only for a locally optimal solutions. The equation is:

$$y(x_1, x_2) = \begin{cases} 10 * x_1 - 1.5 * x_2 - 5 & x_1 < 1 \\ -10 * x_1 - 1.5 * x_2 + 45 & x_1 \geq 4 \\ 0.5 * x_1 - 2.5 * x_2 + 1.5 & x_2 < 3;\ 1 \leq x_1 < 4 \\ 0.5 * x_1 + 10 * x_2 - 35 & x_2 \geq 3;\ 1 \leq x_1 < 4 \end{cases} \tag{5}$$

Similarly to previous experiment, *GMT* managed to find the best split at $x_1 = 1.00$ and induced optimal model tree. *M5* to build the tree needed 18 rules at the leaves and the first split threshold was set at $x_1 = 3.73$. *REPTree* using the CART approach has the first data partition at threshold $x_1 = 4.42$ and has a tree size equal 87.

3.2 Real-Life Datasets

Second group of experiments include several real-life datasets from UCI Machine Learning Repository [22]. Application of the *GMT* to the larger datasets showed that in contrast to RETRIS [8] our method scales well. In the proposed solution the smoothing function is not yet introduced therefore for more honest comparison we present the results of the unsmoothed *M5* and smoothed *M5 smot.* algorithm. The *REPTree* which is another classical top-down inducer build only regression trees and therefore has lower predictive accuracy. Table 2 presents characteristics of investigated datasets and obtained results.

It can be observed that on the real-life datasets the *GMT* managed to induce significantly smaller trees, similarly to the results on artificial data. Additionally, even without smoothing process that improves the prediction accuracy of tree-based models [23], *GMT* has at least comparable performance to smoothed *M5* and on two out of six datasets (*Elevators* and *Kinemaics*) is significantly better. The percentage deviation of $RMSE$ for *GMT* on all datasets was under 0.5%. As for the *REPTree* we may observe that the regression tree is no match for both model trees. Higher model comprehensibility of the *REPTree* thanks to simplified models at leaves is also doubtful because of the large tree size.

As with evolutionary data-mining systems, the proposed approach is more time consuming than the classical top-down inducers. However, experiments performed with typical desktop machine (Dual-Core CPU 1.66GHz with 2GB RAM)

Table 2. Characteristics of the real-life datasets (number of objects/number of numeric features/number of nominal features) and obtained results

		GMT		*M5*		*M5 smot.*		*REPTree*	
Dataset	Properties	RMSE	size	RMSE	size	RMSE	size	RMSE	size
Abalone	4177/7/1	2.150	**3.8**	2.134	12	**2.130**	12	2.358	201
Ailerons	13750/40/0	0.000165	**4.2**	**0.000164**	5.0	**0.000164**	5.0	0.000203	553
Delta Ailerons	7129/5/0	**0.000164**	**9.5**	0.000167	22	0.000165	22	0.000175	291
Delta Elevators	9517/6/0	**0.001424**	**3.1**	0.001427	8.0	0.001426	8.0	0.00150	319
Elevators	16599/18/0	**0.002448**	**14**	0.002702	45	0.002670	45	0.003984	503
Kinemaics	8192/8/0	**0.1457**	**24**	0.1654	106	0.1600	106	0.1906	819

showed that the calculation time even for the largest datasets are acceptable (from 5 minutes for the *Abalone* to around 2 hours for the *Ailerons*).

4 Conclusion

This paper presents a new global approach to the model tree learning. In contrast to classical top-down inducers, where locally optimal tests are sequentially chosen, in *GMT* the tree structure, tests in internal nodes and models at the leaves 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 models are competitive compared to the top-down based counterparts, especially in the term of tree size.

Proposed approach is constantly improved. Further research to determine more appropriate value of complexity penalty term in the *BIC* criterion is advised and other commonly used measures should be considered. Currently we are working on a smoothing process that will improve prediction accuracy. On the other hand, we plan to introduce oblique tests in the non-terminal nodes and more advance models at the leaves.

Acknowledgments. This work was supported by the grant S/WI/2/08 from Bialystok University of Technology.

References

1. Akaike, H.: A New Look at Statistical Model Identification. IEEE Transactions on Automatic Control 19, 716–723 (1974)
2. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group, Belmont (1984)
3. Czajkowski, M., Kretowski, M.: Globally Induced Model Trees: An Evolutionary Approach. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 324–333. Springer, Heidelberg (2010)

4. Dobra, A., Gehrke, J.: SECRET: A Scalable Linear Regression Tree Algorithm. In: Proc. of KDD 2002 (2002)
5. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (1996)
6. 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
7. Gagne, P., Dayton, C.M.: Best Regression Model Using Information Criteria. Journal of Modern Applied Statistical Methods 1, 479–488 (2002)
8. Karalic, A.: Linear Regression in Regression Tree Leaves. International School for Synthesis of Expert Knowledge, Bled, Slovenia (1992)
9. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning. Data Mining, Inference and Prediction, 2nd edn. Springer, Heidelberg (2009)
10. Kretowski, M., Grześ, M.: Global Learning of Decision Trees by an Evolutionary Algorithm. Information Processing and Security Systems, 401–410 (2005)
11. Kretowski, 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)
12. Kretowski, M., Grześ, M.: Evolutionary Induction of Mixed Decision Trees. International Journal of Data Warehousing and Mining 3(4), 68–82 (2007)
13. Kretowski, M., Czajkowski, M.: An Evolutionary Algorithm for Global Induction of Regression Trees. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010. LNCS, vol. 6114, pp. 157–164. Springer, Heidelberg (2010)
14. Malerba, D., Esposito, F., Ceci, M., Appice, A.: Top-down Induction of Model Trees with Regression and Splitting Nodes. IEEE Transactions on PAMI 26(5), 612–625 (2004)
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
16. Murthy, S.: Automatic construction of decision trees from data: A multi-disciplinary survey. Data Mining and Knowledge Discovery 2, 345–389 (1998)
17. Potts, D., Sammut, C.: Incremental Learning of Linear Model Trees. Machine Learning 62, 5–48 (2005)
18. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press, Cambridge (1988)
19. Rokach, L., Maimon, O.Z.: Data mining with decision trees: theory and application. Machine Perception Arfitical Intelligence 69 (2008)
20. Schwarz, G.: Estimating the Dimension of a Model. The Annals of Statistics 6, 461–464 (1978)
21. Torgo, L.: Inductive Learning of Tree-based Regression Models. Ph.D. Thesis, University of Porto (1999)
22. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases (1998), http://www.ics.uci.edu/~mlearn/MLRepository.html
23. Quinlan, J.: Learning with Continuous Classes. In: Proc. of AI 1992, pp. 343–348. World Scientific, Singapore (1992)