An Evolutionary Algorithm for Global Induction of Regression and Model Trees

Marcin Czajkowski *

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45a, 15-351 Białystok, Poland
E-mail: m.czajkowski@pb.edu.pl
*Corresponding author

Marek Kretowski

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45a, 15-351 Białystok, Poland
E-mail: m.kretowski@pb.edu.pl

Abstract: Most tree-based algorithms are typical top-down approaches that search only for locally optimal decisions at each node and does not guarantee the globally optimal solution. In this paper we would like to propose a new evolutionary algorithm for global induction of univariate regression trees and model trees that associate leaves with simple linear regression models. The general structure of our solution follows a typical framework of evolutionary algorithms with an unstructured population and a generational selection. We propose specialized genetic operators to mutate and cross-over individuals (trees), fitness function that base on the Bayesian information criterion and smoothing process that improves the prediction accuracy of the model tree. Performed experiments on 15 real-life datasets show that proposed solution can be significantly less complex with at least comparable performance to the classical top-down counterparts.

Keywords: evolutionary algorithms, regression trees, model trees, linear regression, SLR, BIC

Reference to this paper should be made as follows: Czajkowski, M. and Kretowski, M. (xxxx) 'An Evolutionary Algorithm for Global Induction of Regression and Model Trees', *Int. J. Data Mining, Modelling and Management*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Marcin Czajkowski received his Master's degree in Computer Science from the Bialystok University of Technology, Poland, in 2007. He joined the Faculty of Computer Science at the Bialystok University of Technology in 2008, where he is currently finishing his PhD thesis. His research activity mainly concerns machine learning and data mining, in particular, classification and regression trees.

Marek Kretowski received his Master's degree in Computer Science from the Bialystok University of Technology, Poland, in 1996. His Ph.D. thesis defended in 2002, was prepared in a framework of collaboration between Laboratory of Signal and Image Processing, University of Rennes 1, France and Faculty of Computer Science, Bialystok University of Technology, Poland. In 2009 he received D.Sc. (Habilitation) degree in Computer Science from Institute of Computer Science, Polish Academy of Science, Warsaw, Poland. Now he works as an Associate Professor in Faculty of Computer Science, Bialystok University of Technology. His current research focuses on data mining methods and biomedical applications of computer science.

1 Introduction

The most common predictive tasks in data mining (Fayyad et al., 1996) are classification and regression and the decision trees (Murthy, 1998; Rokach and Maimon, 2008) are one of the most widely used prediction techniques. Regression and model trees are now popular alternatives to classical statistical techniques like standard regression or logistic regression (Hastie et al., 2009). They are easy to understand and interpret which makes them particularly useful when the aim of modeling is to understand the underlying processes of the environment. Decision trees are also applicable when the data does not satisfy rigorous assumptions required by more traditional methods (Hastie et al., 2009). We focus on univariate trees since they are a 'white-box' technique and it makes them particularly interesting for scientific modeling. It is easy to find explanation for predictions of univariate regression and model trees.

1.1 Regression and model trees

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 regression tree and model tree is that, for the latter, constant value in the terminal node is replaced by a regression plane. One of the first and most known regression tree solution was presented in the seminal book by Breiman et al. (1984) describing the *CART* system. *CART* finds a split that minimizes the sum of squared residuals of the model when predicting and builds a piecewise constant model with each terminal node fitted by the training sample mean. The accuracy of prediction was later improved by replacing single values in the leaves by more advanced models. *M5* proposed by Quinlan (1992), induces a model tree that contains at leaves multivariate linear models analogous to piecewise linear functions. *HTL* presented by Torgo (1997) goes further and evaluate linear and nonlinear models in terminal nodes. Model trees can also be applied to the classification problems (Kotsiantis, 2010).

All aforementioned decision trees are built by a process that is known as recursive partitioning. Top-down induction starts from the root node where locally optimal split (test) is searched according to the given optimality measure. Then,

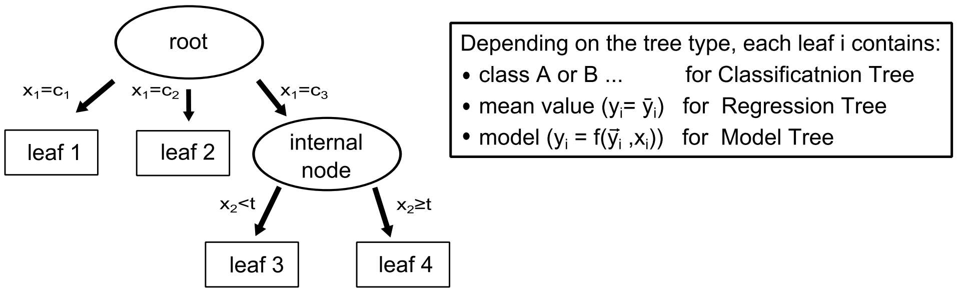
root
$x_1=c_1$
$x_1=c_2$
$x_1=c_3$
leaf 1
leaf 2
internal node
$x_2<t$
$x_2\geq t$
leaf 3
leaf 4
Depending on the tree type, each leaf i contains:
• class A or B ... for Classificatnion Tree
• mean value ($y_i=\bar{y}_i$) for Regression Tree
• model ($y_i = f(\bar{y}_i, x_i)$) for Model Tree

Figure 1 An example of univariate decision tree with tests on nominal and continuous-valued features. Depending on the tree type, leaves could contain class (classification tree), continuous value (regression tree) or some kind of model (model tree).

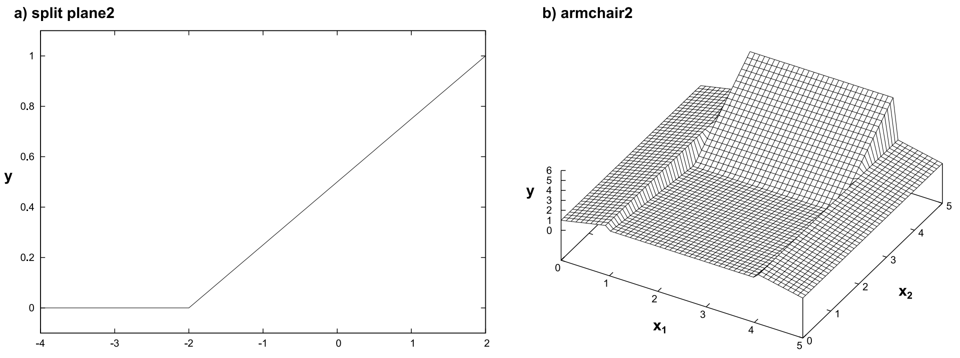

Figure 2 Examples of artificial datasets: a) *split plane2*, b) *armchair2*.

the training data is redirected to newly created nodes and this process is repeated for each node until some stopping-rule is violated. Finally, the post-pruning is applied to improve the generalization power of the predictive model.

1.2 Motivation

Inducing the decision tree by a greedy strategy is fast and generally efficient in many practical problems, but usually produces locally optimal solutions. It can be expected that a more global induction could improve the tree structure and the model prediction. Figure 2 illustrates two simple artificially generated datasets with analytically defined decision borders.

The left dataset *split plane2*, discussed also in (Vogel et al., 2007), can be perfectly predictable with regression lines on subsets of the data resulting from a single partition. The equation is:

$$y = \begin{cases} 0 & -4 \leq x_1 < -2 \\ 0.25x_1 + 0.5 & -2 \leq x_1 \leq 2 \end{cases} \tag{1}$$

Most of popular greedy top-down inducers that minimizes the residual sum of squares (RSS) like *CART* or standard deviation like *M5* will not find the best

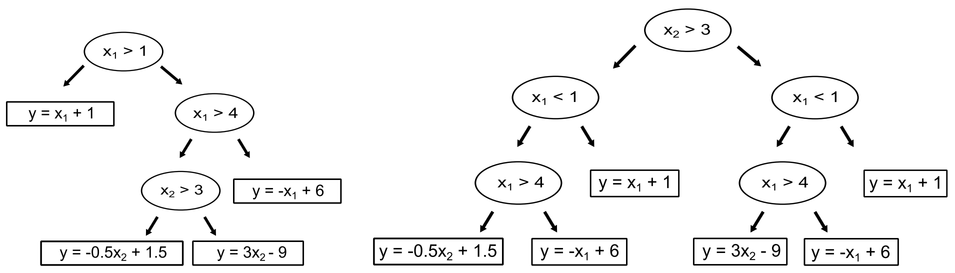

Figure 3 Examples of model trees for *armchair2* (global approach - left, greedy approach - right).

partitions (*CART* finds threshold at $x_1 = -0.44$, *M5* at $x_1 = -1.18$). Not optimal partition in the root node usually increases of the tree size and may result the higher prediction error.

Illustrated in the Figure 2 b) function is defined as:

$$y = \begin{cases} x_1 + 1 & 0 \leq x_1 \leq 1 \\ -x_1 + 6 & 4 < x_1 \leq 5 \\ -0.5x_2 + 1.5 & 1 < x_1 \leq 4,\ 0 \leq x_2 \leq 3 \\ 3x_2 - 9 & 1 < x_1 \leq 4,\ 3 < x_2 \leq 5 \end{cases} \tag{2}$$

It is a little more complex then *split plane2* and many traditional approaches will fail to efficiently split the data as the greedy inducers search only for a locally optimal solutions. The Figure 3 presents the optimal model trees that can be generated by globally induced and greedy top-down algorithms. These two simple artificial problems illustrate general advantage of the global search solutions to greedy algorithms.

1.3 Related work

Multiple authors have proposed methods to limit negative effects of inducing the decision tree with the greedy strategy. In the *SECRET* authors Dobra and Gehrke (2002) show that classification approach finds more globally optimal partitions than the *CART* system. Different solution was proposed in *SMOTI* by Malerba et al. (2004) where regression models exist not only in the leaves but also in the upper parts of the tree. Authors suggest that this allows for individual predictors to have both global and local effects on the model tree. A more recent innovation in order to find optimal splits in nodes was presented in *LLRT* by Vogel et al. (2007). *LLRT* allows for a near-exhaustive evaluation of all possible splits in a node, based on the quality of fit of linear regression models in the resulting branches.

In the literature there are also some attempts of applying evolutionary approach for induction of regression trees. In *TARGET*, Fan and Gray (2005) propose to evolve a *CART*-like regression tree with simple genetic operators. Bayesian information criterion (*BIC*) (Schwarz, 1978) is used as a fitness function which penalizes the tree for over-parametrization. Experiments performed on two real datasets suggest that *TARGET* outperforms in the terms of Mean Squared Error the *CART* solution. Much more advanced trees are presented in *GPMCC* where Potgieter and Engelbrecht (2008) make use of a genetic algorithm to evolve multivariate non-linear models at the leaves. Authors managed to decrease the size of the trees comparing to commercial version of *M5* (Cubist) and neutral network called *NeuroLinear* (Setiono et al., 2002), however their system was outperformed in terms of predictive accuracy.

Currently, there are no sufficient solutions with a good trade-off between predictive performance and a model comprehensibility. Model trees with complex rules at the leaves or ensemble methods generate accurate predictions but are difficult to interpret by a human experts. On the other side, regression trees have lower predictive performance but higher model comprehensibility. Finally, performed experiments suggest that regression and model trees usually built overgrown trees and therefore are more difficult to analyse and interpret.

In this paper we would like to present an evolutionary algorithm for global induction of regression and model trees. It fills the gap between simple regression trees and advanced but less comprehensible model trees. Previously performed research showed that global inducers are capable to efficiently evolve various types of classification trees: univariate (Kretowski and Grześ, 2005), oblique (Kretowski and Grześ, 2006) and mixed (Kretowski and Grześ, 2007). In our last paper we applied a similar approach to obtain accurate and compact regression trees (Kretowski and Czajkowski, 2010) called *GRT* and we did preliminary experiments with the model trees (Czajkowski and Kretowski, 2010) called *GMT*. Our work covers the induction of univariate regression trees and model trees with simple linear models at the leaves. Proposed solution denoted as *GMT* 2.0 improved our previous solutions in almost every step of evolutionary algorithm. Starting with more heterogeneity population, additional genetic operators and new fitness function, that extends the BIC. We also introduced the smoothing process that could improve the prediction accuracy of the model tree.

2 Evolutionary induction of model trees

The *GMT* 2.0 general structure follows a typical framework of evolutionary algorithms (Michalewicz, 1996) with an unstructured population and a generational selection.

2.1 Representation

Model trees are represented in their actual form as classical univariate trees with a simple linear model at each leaf. Each test in a non-terminal node concerns only one attribute (nominal or continuous valued). In a case of a continuous-valued feature typical inequality tests are applied. As potential splits only precalculated candidate thresholds (Fayyad and Irani, 1992) 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 and focuses the search process. 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.

A simple linear model is calculated at each terminal node of the model tree using standard regression technique (Press et al., 1988). A dependent variable y is modeled as a linear function of single variable x:

$$Y = \alpha + \beta * x, \tag{3}$$

where x is one of the independent variables, α is the intercept and β is the slope of the regression line that minimizes the sum of squared residuals of the model.

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

Initial individuals are created by applying the classical top-down algorithm, similar to the *CART* and *M5* approaches. At first, we learn a standard regression tree that has mean of dependent variable values from training objects at each leaf. The recursive partitioning is finished when all training objects in a node are characterized by the same predicted value (or it varies only slightly, default: 1%) or the number of objects at node is lower than the predefined value (default value: 5). Additionally, user can set the maximum tree depth (default value: 10) to limit initial tree size. Next, a simple linear model is calculated at each terminal node of the model tree.

Traditionally, the initial population should be generated randomly to cover the entire range of possible solutions. Due to the large solution space the exhaustive search may be infeasible. Therefore, while creating initial population we search for a good trade off between a high degree of heterogeneity and relatively low computation time. We propose several strategies:

- initial individuals are created by applying the classical top-down algorithm to randomly chosen subsamples of the original training data (10% of data, but not more than 500 examples);
- for each individual only tests based on the random subset of attributes (default 50% of attributes) are applied;
- at each individual for all non-terminal nodes one of the four test search strategies is randomly chosen:
  - *Least Squares (LS)* function reduces node impurity measured by sum of squares proposed in *CART*,
  - *Least Absolute Deviation (LAD)* function reduces the sum of absolute deviations. It has greater resistance to the influence of outlying values to *LS*,
  - *Mean Absolute Error (MAE)* function which is more robust and also less sensitive to outliers to *LS*,
  - *dipolar*, where a dipol (a pair of feature vectors) is selected and then a test is constructed which splits this dipole. First instance that constitutes dipol is randomly selected from the node. Rest of the feature

vectors are sorted decreasingly according to the difference between dependent variable values to the firstly chosen instance. To find a second instance that constitutes dipol we applied mechanism similar to the ranking linear selection (Michalewicz, 1996).

- one of three search strategies of predicted variable used in linear model at the leaves is applied:
  - *optimal*: finds the locally optimal model that minimizes the sum of squared residuals. It is the most time-consuming search strategy as it must calculate simple linear regression model for each attribute.
  - *random*: finds the simple linear model from training objects in this leaf on the randomly chosen independent variable.
  - *none*: the fastest strategy. No attribute is used to build the simple linear model, therefore each terminal node contains the sample mean.

Additionally, user can set the size of the population (default value: 50).

2.3 Genetic operators

To maintain genetic diversity, we have proposed two specialized genetic operators corresponding to the classical mutation and cross-over. Each evolutionary iteration starts with randomly choosing the operator type where default probability to select mutation equal 0.8 and cross-over 0.2. Both operators have influence on the tree structure, tests in non-terminal nodes and models at the leaves. After each operation 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. Modifying a leaf makes sense only if it contains objects with different dependent variable values.

2.3.1 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 (Czajkowski and Kretowski, 2010):

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

2.3.2 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 is applied to decide which node will be affected. Depending on the type of node, ranking take into account:

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

We have proposed new variants of mutation for internal node:

- node can be transformed (pruned) into a leaf,
- tests between father and son exchanged,
- mutation between subtrees that replaces all subtrees with randomly chosen one,
- test in node reinitialized by new random or new dipolar one (described in Section 2.2),
  - shifting the splitting threshold at continuous-valued feature,
  - re-grouping nominal feature values by adding, merging branches or moving value between them,

and for the leaf:

- transform leaf into an internal node with a new dipolar test,
- replace simple linear model by a new one that is recalculated on a random predictor variable,
- remove predictor variable and leave mean value at the leaf.

2.4 Selection and termination condition

Ranking linear selection 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)*.

Evolution terminates when the fitness of the best individual in the population does not improve during the fixed number of generations (default value: 1000). In case of a slow convergence, maximum number of generations is also specified (default value: 5000), which allows to limit the computation time.

2.5 Fitness function

Specification of a suitable fitness function is one of the most important and sensitive element in the design of the evolutionary algorithm. It measures how good a single individual is in terms of meeting the problem objective and drives the evolutionary search process. 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 (Rokach and Maimon, 2008), this problem is partially mitigated by defining a stopping condition and by applying a post-pruning (Esposito et al., 1997).

Fitness is computed for all members of the population after each generation. In our previous work (Czajkowski and Kretowski, 2010) we used Akaike's information criterion (*AIC*) (Akaike, 1974) as a fitness function. This measure of the goodness of fit worked also as a penalty for increasing the tree size. *AIC* is given by:

$$Fit_{AIC}(T) = -2 * ln(L(T)) + 2 * k(T), \tag{4}$$

where $L(T)$ is the maximum of the likelihood function of the tree T and $k(T)$ is the number of model parameters in the tree. Log(likelihood) function $L(T)$ is typical for regression models (Gagne and Dayton, 2002) and can be expressed as

$$ln(L(T)) = -0.5n * [ln(2\pi) + ln(SS_e(T)/n) + 1], \tag{5}$$

where $SS_e(T)$ is the sum of squared residuals of the tree T and n is the number of observations. The term $k(T)$ can also be viewed as a penalty for over-parametrization. This complexity penalty term was set to $Q(T) + 1$ in where $Q(T)$ is equal to the number of terminal nodes in model tree T.

We tested also a measure introduced by Schwarz (1978) called Bayesian information criterion (*BIC*) that seems to be more appropriate for the regression and model trees. In this information criterion, which is similar to *AIC*, the penalty for increasing model size depends on the n - number of observations in the data:

$$Fit_{BIC}(T) = -2 * ln(L(T)) + ln(n) * k(T). \tag{6}$$

However, performed research reveal that both information criteria in their base form were not able to find an optimal structure of *GMT* 2.0. Measures worked sufficiently good when the probability of mutation for leaves to transform into internal nodes was very low or equal zero. Higher probability of transforming leaves into the internal nodes caused rapid increase of size and error of the searched structure. However, not including this mutation operator strongly limits variants of the evolution of the tree structure.

In this paper we propose a new fitness function which extends the *BIC*. The number of independent parameters in the complexity penalty term $k(T)$ for *GMT* 2.0 was set to $2(Q(T) + W(T))$ where $W(T)$ is the number of attributes in the linear models at the leaves (equal 1 for model node or 0 for regression node). High value of penalty term, compared to our previous solution or the *TARGET* system allow *GMT* 2.0 to induce significantly smaller trees.

Performed research in determining appropriate value of the penalty term $k(T)$ suggests that the modification of the number of model parameters in the tree is only a partial solution. Higher value of $k(T)$ impact data with high and low value of likelihood function in a different way and therefore it is not universal. Complexity penalty term has the highest effect when the sum of squared residuals $SS_e(T)$ of the tree is high because of the logarithm function. Small value of fraction $SS_e(T)/n$ results in high value of likelihood function which makes $Fit_{BIC}(T)$ less sensitive to the penalty term $k(T)$. To obtain fitness function that is not sensitive to the various values of the likelihood function, we multiplied the $Fit_{BIC}(T)$ by, as we call it, the *Tree Size Factor*. This is additional complexity penalty tries to balance the penalty for the small and large datasets. The *Tree Size Factor* is denoted as $\varrho(T)$ and can be expressed as:

$$\varrho(T) = \frac{n + Q(T)}{n - Q(T)}. \tag{7}$$

Therefore the complete fitness function equation for *GMT* 2.0 is given by:

$$Fit_{GMT\ 2.0}(T) = \begin{cases} Fit_{BIC}(T) * \varrho(T)\ when\ Fit_{BIC}(T) \geq 0 \\ Fit_{BIC}(T) * \frac{1}{\varrho(T)}\ when\ Fit_{BIC}(T) < 0 \end{cases}. \tag{8}$$

The best single individuals are the ones with the lowest $Fit_{GMT\ 2.0}(T)$ value. The *Tree Size Factor* $\varrho(T)$ increases the value of the fitness function and depends on the number of observation and leaves.

2.6 Smooting

In *M5* algorithm, Quinlan (1992) proposed the smoothing process to improve the prediction accuracy of the tree-based models. The smoothing process modify the predicted by a model at the appropriate leaf, value of each case to reflect the predicted values at nodes along the path from that leaf to the root. It requires to generate additional linear models for every internal node of the tree. In the *GMT* 2.0 we developed the form of smoothing that is similar to the one in M5 algorithm. At first, predicted value for a test instance is computed by the leaf model. Then, this value is smoothed and updated along the path back to the root by linear models for each nodes. Let $P(T_i)$ denote the predicted value at T_i subtree of tree T, then:

$$P(T) = \frac{n_i * P(T_i) + k * M(T)}{n_i + k}, \tag{9}$$

where n_i is the number of training cases at T_i, $M(T)$ is the predicted value recalculated from the linear model at T and k is a smoothing constant (default 10).

Figure 4 illustrates the smoothing process for the test instances at the leaf with linear model denoted as $LM4$. If there were no smoothing process, the predicted value $P(T)$ for a tested instance would be equal the value calculated from the model $LM4$. However, with the smoothing process turned on, the models that are on the path from the leaf to the root ($LM5$ and $LM6$) have influence on the final predicted value $P(T)$.

According to Quinlan (1992) smoothing has most effect when some models were constructed for few training cases or when the models along the path predict instances very differently. However trees that adapt smoothing differs from the classical univariate model trees. Each test instance is predicted not only by single model at proper leaf but also by different linear models generated for every internal node up to the root node. Therefore smoothing affects the simplicity of the solution making it more difficult to understand and interpret.

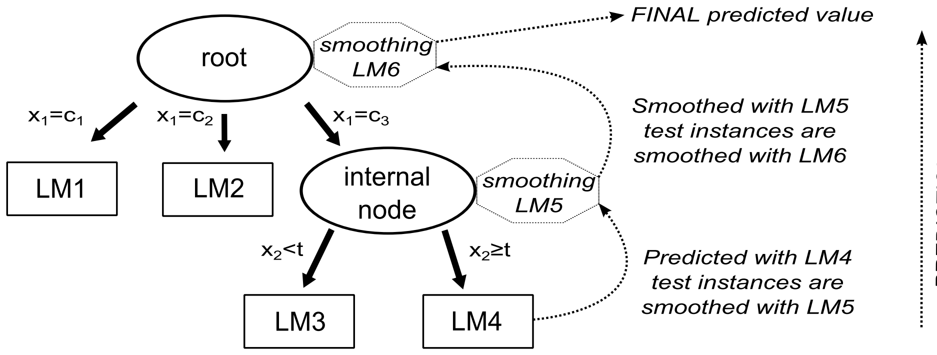

Figure 4 The smoothing process for the test instances at the leaf with linear model denoted as $LM4$.

3 Experimental validation

Two sets of experiments were performed - one for the regression trees and second for the model trees. *GMT* 2.0 was validated on several real-life datasets. Obtained results were compared with our previous solutions (Kretowski and Czajkowski, 2010; Czajkowski and Kretowski, 2010) and popular regression and model trees that are available in the *Weka* system (Hall et al., 2009).

3.1 Setup

To assess the performance of the proposed system in solving real-life problems, several datasets from UCI Machine Learning Repository (Blake et al., 1998) and provided by Torgo (2010) were analysed. Table 1 presents the details of each dataset. All results presented in this paper 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 prediction error measure of the tested systems. The number of nodes is given as a complexity measure (size) of regression and model trees. Each tested algorithm runs with default values of parameters through all datasets.

3.2 Regression trees

Regression trees are used for analysis that require simple predictions based on a few logical if-then conditions. However, most solutions induce overgrown regression trees which are difficult to analyze. Domain experts need solutions that are smaller and therefore easier to understand. Our main goal in this set of experiments was to decrease the tree size of our previous solution called *GRT* without significant increase of prediction error. It is expected that changes in the complexity penalty term $k(T)$ influence not only the tree size but also the prediction error of *GMT* 2.0. For a comparison purpose, we have tested four regression tree systems:

- *GMT* 2.0_{reg} - proposed solution, set to work as a regression tree;
- *GRT* - one of the predecessors of *GMT* 2.0. Globally induced regression tree proposed in (Kretowski and Czajkowski, 2010);

Table 1 Characteristics of the real-life datasets

Dataset			*Number of features*	
Name	Symbol	Number of instances	Numeric	Nominal
Abalone	AB	4177	7	1
Ailerons	AI	13750	40	0
Auto-Mpg	AM	392	4	3
Auto-Price	AP	159	14	1
Delta Ailerons	DA	7129	5	0
Delta Elevators	DE	9517	6	0
Elevators	EL	16599	18	0
Housing	HO	506	13	0
Kinemaics	KI	8192	8	0
Machine CPU	MC	209	6	0
Pole	PO	15000	48	0
Pyrimidines	PY	74	27	0
Stock	ST	950	9	0
Triazines	TR	186	60	0
Wisconsin Cancer	WC	194	32	0

- *REPTree* - popular top-down inducer. *REPTree* builds a regression tree using variance and prunes it using reduced-error pruning (with backfitting);

- $M5_{reg}$ - state of art model tree proposed by Quinlan (1992), set to work as a regression tree.

Table 2 presents results for the regression trees. *GMT* 2.0 alike *GRT* managed to induce significantly smaller trees compared to the tested counterparts. This can be especially noticed on large datasets. Almost all *GMT* 2.0 trees are smaller and therefore easier to analyze and interpret. The only exception appears in the *Pyrimidines (PY)* dataset where globally induced trees are little more complex to the tested counterparts however have significantly higher prediction accuracy. This suggests that greedy algorithms like $M5_reg$ and *REPTree* underfitted to the training data and did not capture the underlying structure.

The average prediction error of *GMT* 2.0 is similar to the *GRT* and *REPTree* however it is slightly worse than $M5_{reg}$. Comparing to our previous solution, *GMT* 2.0 managed to significantly decrease tree size in all datasets for over 70% (average). In the same time, in 7 out of 15 datasets the prediction error *RMSE* for *GMT* 2.0 decreased, compared to *GRT* or stayed on the same level. Lack of improvement on some datasets may be explained by the significantly smaller trees induced by the *GMT* 2.0. There is usually a trade-off between the predictive performance and the model comprehensibility. Additional experiments showed that the *GMT* 2.0 with lower value of the parameter $k(T)$ managed to induce larger but much more accurate regression trees. Modification of this penalty term allows to fine tune the decision tree algorithm.

Table 2 Obtained results for the regression trees

	GMT 2.0		*GRT*		*REPTree*		$M5_{reg}$	
Dataset	RMSE	size	RMSE	size	RMSE	size	RMSE	size
AB	2.33	**3.5**	2.31	51	2.35	201	**2.28**	36
AI	0.000213	**19**	0.000217	27	0.000203	553	**0.000199**	166
AM	3.96	**2.1**	3.57	45	3.6	94	**3.49**	19
AP	**2542**	**2.0**	2618	13	2760	32	2543	8.0
DA	0.000179	**7.4**	0.000179	82	**0.000175**	291	0.00176	74
DE	0.00150	**7.9**	**0.00148**	78	0.00150	319	**0.00148**	59
EL	0.00435	**22**	0.00443	32	**0.00398**	503	0.00413	189
HO	4.51	**6.5**	**4.17**	32	4.84	41	4.72	26
KI	0.194	**25**	0.194	34	0.191	819	**0.182**	264
MC	74.2	**2.7**	**63.9**	15	92.34	15	64.8	10
PO	9.91	**19**	10.32	25	**8.25**	223	9.32	139
PY	0.104	4.1	**0.101**	10	0.135	**1.0**	0.135	**1.0**
ST	1.41	**13**	1.33	39	1.19	137	**1.14**	88
TR	0.142	**4.9**	**0.139**	14	0.152	7.0	0.140	5.0
WC	**33.3**	**1.9**	39.2	16	35.9	9.0	35.0	3.0

3.3 Model trees

Model trees which are an extension of the regression trees, usually have higher performance in the terms of the accuracy prediction. However, model trees like *HTL* (Torgo, 1997) or *SMOTI* (Malerba et al., 2004) build complex models at the leaves that reduces simplicity of the predictions. Therefore in this set of experiments we focus on comparing the model trees with simple linear regression models at the leaves:

- *GMT* 2.0 - proposed solution with no smoothing.
- *GMT* - one of the predecessors of *GMT* 2.0. Globally induced model tree with simple linear regression models at the leaves proposed in (Czajkowski and Kretowski, 2010);
- $M5_{slr}$ - state of art model tree system proposed by Quinlan (1992), set to work with simple (instead of multivariate) linear regression model at the leaves.

We may observe from Table 3 that *GMT* 2.0 alike *GMT* managed to induce significantly smaller trees to the tested counterparts. Research showed that the sizes of induced *GMT* and *GMT* 2.0 trees are similar. However, in this set of experiment we focus on improving of the *GMT* prediction power. We managed to reduce $RMSE$ error, comparing to our previous solution, in 14 out of 15 datasets for *GMT* 2.0. Comparing to the $M5_{slr}$, proposed solution managed to not only significantly decrease tree size but also reduce the prediction error in most of the datasets.

Table 3 Comparison results for the model trees with simple linear regression models at the leaves

	GMT 2.0		*GMT*		$M5_{slr}$	
Dataset	RMSE	size	RMSE	size	RMSE	size
AB	**2.24**	**6.7**	2.30	7.7	**2.24**	35
AI	0.000200	24	0.000207	**18**	**0.000194**	109
AM	**3.23**	**4.7**	3.43	9.9	3.35	11
AP	2328	**2.9**	2507	3.7	**2183**	6
DA	0.000173	13	0.000178	**11**	**0.000170**	46
DE	**0.00147**	**8.6**	0.00150	9	0,00148	40
EL	0.00397	40	0.00444	**13**	**0.00386**	174
HO	**4.21**	**6.6**	4.32	9.1	4.36	21
KI	0.183	34	0.196	**20**	**0,178**	196
MC	**63.4**	6.1	67.5	**3.8**	89.5	7.0
PO	**9.37**	67	12.41	**12**	10.28	108
PY	**0.103**	4.4	0.109	4.5	0.118	**3.0**
ST	1.22	18	1.63	**7.1**	**1.08**	64
TR	0.142	4.9	**0.141**	4.7	**0.141**	**4.0**
WC	**32.7**	**1.0**	34.3	3.1	33.8	**1.0**

Table 4 Comparison results for the smoothed model trees with simple linear regression models at the leaves

	GMT 2.0 *smot*		$M5_{slr}$ *smot*	
Dataset	RMSE	size	RMSE	size
AB	2.23	**6.7**	**2.21**	35
AI	0.000200	**24**	**0.000186**	109
AM	**3.18**	**4.7**	3.22	11
AP	**2243**	**2.9**	2282	6
DA	0.000172	**14**	**0.000169**	46
DE	**0.00146**	**8.8**	0.00147	40
EL	0.00393	**39**	**0.00366**	174
HO	4.07	**6.9**	4.07	21
KI	0.182	**34**	**0.172**	196
MC	**62.6**	**5.8**	87.6	7.0
PO	**9.37**	**61**	9.61	108
PY	**0.093**	4.3	0.115	**3.0**
ST	1.22	**17**	1.22	64
TR	**0.132**	5.0	0.137	**4.0**
WC	**32.7**	**1.0**	33.5	**1.0**

3.4 Smoothed model trees

The smoothing process often improves the prediction accuracy of the tree-based models. Table 4 illustrates the impact of the smoothing function on *GMT* 2.0 and $M5_{slr}$ solutions. We may observe that both algorithms managed to slightly improve the prediction accuracy on most of the datasets. Low impact of smoothing function and weaker improvement of *RMSE* on proposed solution comparing to the $M5_{slr}$ *smot* may result from smaller and more optimal *GMT* 2.0 tree structure that cannot be so efficiently adjusted.

Table 4 results shows that smoothing process may have also a negative impact on the final prediction. In *Stock* and *Auto-Price* dataset, the *RMSE* calculated for the $M5_{slr}$ *smot* has increased. None of this happen to *GMT* 2.0 *smot*.

3.5 Calculation time

As with most evolutionary algorithms, calculation time of the proposed approach is more time consuming than the classical top-down inducers. Performed experiments with the Dual-Core CPU 1.66GHz with 2GB RAM on the dataset *Elevators* (16 559 instances, 18 attributes) showed that time:

- for regression trees: $M5_{reg}$ equal 7 seconds, *GMT* 2.0 equal 1.5 minutes;
- for model trees: $M5_{slr}$ equal 11 seconds, *GMT* 2.0 equal 33 minutes;
- for smoothed model trees: no relevant time differences.

Difference between $M5_{slr}$ and *GMT* 2.0 is caused by the evolutionary evaluation of linear models at the leaves. However, proposed solution is scalable and can manage large datasets.

4 Conclusion

Regression trees and model trees with simple linear models at the leaves are important "white box" solutions. In this paper we propose a new global approach to the model tree learning and compare it with classical top-down inducers. The structure of the *GMT* 2.0 tree, tests in non-terminal nodes and models at the leaves are searched in the same time by specialized evolutionary algorithm.

Experimental results show that the globally evolved regression models are highly competitive compared to the top-down based counterparts, especially in the term of tree size. *GMT* 2.0 managed to significantly improve our previous solution: *GRT* regression trees in the term of size and *GMT* in the predictive accuracy.

Proposed solution may be applied to the problems that are primarily concerned with the regression of an outcome onto a single predictor. As an example the original genetic epidemiology problem required only consideration of simple linear regression models like (Shannon et al., 2002) to locate genes associated with a quantitative trait of interests. *GMT* 2.0 is constantly improved. We plan to introduce oblique tests in the non-terminal nodes and more advance models at the leaves. We also plan to parallelize the evolutionary algorithm in order to speed-up its execution time.

Acknowledgements

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

References

Akaike, H. (1974) 'A New Look at Statistical Model Identification', *IEEE Transactions on Automatic Control*, Vol. 19, pp.716–723.

Breiman, L., Friedman, J., Olshen, R. and Stone C. (1984) *Classification and Regression Trees.* Wadsworth Int. Group.

Czajkowski, M. and Kretowski, M. (2010) 'Globally Induced Model Trees: An Evolutionary Approach', *In: Proc. of PPSN XI, Lecture Notes in Computer Science*, Vol. 6238, pp.324–333.

Dobra, A. and Gehrke, J. (2002) 'SECRET: A Scalable Linear Regression Tree Algorithm', *In: Proc. of KDD'02.*

Esposito, F., Malerba, D. and Semeraro, G. (1997) 'A comparative analysis of methods for pruning decision trees', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 5, pp.476–491.

Fan, G. and Gray, J.B. (2005) 'Regression tree analysis using target', *Journal of Computational and Graphical Statistics*, Vol. 14, No. 1, pp.206–218.

Fayyad, U.M. and Irani, K. B. (1992) 'On the Handling of Continuous-Valued Attributes in Decision Tree Generation', *Machine Learning*, Vol. 8, pp.87-102.

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

Gagne, P. and Dayton, C.M. (2002) 'Best Regression Model Using Information Criteria', *Journal of Modern Applied Statistical Methods*, Vol. 1, pp.479–488.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, H.I. (2009) 'The WEKA Data Mining Software', SIGKDD Explorations, Vol. 11, No. 1.

Hastie, T., Tibshirani, R. and Friedman, J.H. (2009) *The Elements of Statistical Learning. Data Mining, Inference and Prediction*, 2nd ed., Springer.

Kotsiantis, S.B. (2010) 'Rotation-based model trees for classification', *Int. J. Data Analysis Techniques and Strategies*, Vol. 2, No. 1, pp.22-37.

Kretowski, M. and Grześ, M. (2005) 'Global Learning of Decision Trees by an Evolutionary Algorithm', *Information Processing and Security Systems*, Springer, pp.401–410.

Kretowski, M. and Grześ, M. (2006) 'Evolutionary Learning of Linear Trees with Embedded Feature Selection', *In: Proc. of ICAISC'06, Lecture Notes in Artificial Intelligence*, Vol. 4029, pp.400–409.

Kretowski, M. and Grześ, M. (2007) 'Evolutionary Induction of Mixed Decision Trees', *International Journal of Data Warehousing and Mining*, Vol. 3, No. 4, pp.68–82.

Kretowski, M. and Czajkowski, M. (2010) 'An Evolutionary Algorithm for Global Induction of Regression Trees', *In: Proc. of ICAISC'10, Lecture Notes in Artificial Intelligence*, Vol. 6114, pp.157–164.

Malerba, D., Esposito, F., Ceci, M. and Appice, A. (2004) 'Top-down Induction of Model Trees with Regression and Splitting Nodes', *IEEE Transactions on PAMI*, Vol. 26, No. 5, pp.612–625.

Michalewicz, Z. (1996) *Genetic Algorithms + Data Structures = Evolution Programs*, 2rd ed., Springer.

Murthy, S. (1998) 'Automatic construction of decision trees from data: A multi-disciplinary survey', *Data Mining and Knowledge Discovery*, Vol. 2, pp.345–389.

Potgieter, G. and Engelbrecht, A. (2008) 'Evolving model trees for mining data sets with continuous-valued classes', *Expert Systems with Applications*, Vol. 35, pp.1513–1532.

Press, W.H., Flannery, B.P., Teukolsky, S.A. and Vetterling, W.T. (1988) *Numerical Recipes in C*, Cambridge University Press.

Rokach, L. and Maimon, O.Z. (2008) *Data mining with decision trees: theory and application*, World Scientific Publishing Co Pte Ltd.

Schwarz, G. (1978) 'Estimating the Dimension of a Model', *The Annals of Statistics*, Vol. 6, pp.461-464.

Setiono, R., Leow, W.K. and Zurada, J.M. (2002) 'Extraction of rules from artificial neural networks for nonlinear regression', *IEEE Transactions on Neural Networks*, Vol. 13, No. 3, pp.564–577.

Shannon, W.D., Province, M.A. and Rao, D.C. (2002) 'Tree-Based Models for Fiting Stratified Linear Regression Models', *Journal of Classification*, Vol. 19, pp.113–130.

Torgo, L. (1997) 'Functional Models for Regression Tree Leaves', *In: Proc. of ICML, Morgan Kaufmann*, pp.385–393.

Torgo, L. (2010) *Regression DataSets Repository*, [http://www.liaad.up.pt/ ltorgo].

Blake, C., Keogh, E. and Merz, C. (1998) *UCI Repository of Machine Learning Databases*, [http://archive.ics.uci.edu/ml/].

Quinlan, J. (1992) 'Learning with Continuous Classes', *In: Proc. of AI'92, World Scientific*, pp.343–348.

Vogel D., Asparouhov O. and Scheffer T. (2007) 'Scalable look-ahead linear regression trees', *In: Proc. of 13th ACM SIGKDD, ACM Press New York*, pp.757–764.