Partial Linear Trees

Luís Torgo LTORGO@NCC.UP.PT
LIACC-FEP, R. Campo Alegre, 823, 4150 Porto, Portugal

Abstract

Partial linear regression is a successful method of dealing with regression tasks. This technique integrates a linear polynomial with a kernel model with the goal of being able to handle non-linear problems. Kernel models are a kind of instance-based learners and as such do not obtain any intelligible model of the data, which is not the case of linear polynomials. By integrating these two approaches partial linear regression looses some of the intelligibility of linear polynomials, particularly in highly non-linear domains. In such cases the importance of the kernel component grows and the linear polynomial hardly describes the true regression surface approximated by the partial linear model. In this paper we describe partial linear trees that aim at increasing the description accuracy of partial linear models by decreasing the importance of the kernel corrections in non-linear domains. This is accomplished by identifying several regions of low variance where different partial linear models are fitted. Within these regions linear polynomials are more accurate and thus the kernel "adjustments" are smaller. Partial linear trees are able to maintain the high predictive accuracy of partial linear regression while providing a more accurate description of the regression surface and being computationally more efficiency.

1. Introduction

This paper describes a technique that deals with a learning task usually known as regression. This problem can be loosely defined as the search for a model of the relation between a continuous variable (the target) and a set of other variables (the inputs) based on a sample of this relationship.

Many different modeling techniques exist that tackle this problem. Least squares linear regression (e.g., Drapper & Smith, 1981) is among the most well-known. One of the reasons for its success is the simplicity of the resulting linear model (a first order polynomial on the input variables). However, linear models impose a very strong assumption on the form of the regression surface being modeled. This assumption is often inadequate particularly in non-linear problems leading to poor predictive accuracy. An elegant solution to this difficulty is to incorporate a non-parametric component to these models in order to overcome "local" inadequacies of the linearity assumption. The resulting models are often called semiparametric (e.g., Hardle, 1990) and have the advantage of maintaining some of the intelligibility of linear regression, while improving its accuracy. An example of such techniques are partial linear models (Spiegelman, 1976; Hardle, 1990) that integrate a standard least squares linear regression model with a kernel smoother. These models are able to handle non-linear problems by adding a "local correction" to the predictions of the linear polynomial. These "corrections" are calculated for each individual query case. They are an estimate of the inadequacy of the linear polynomial for the query case based on its performance on nearby training cases. However, these local adjustments decrease the comprehensibility of partial linear models. In effect, if the domain under consideration is clearly non-linear, the kernel component of partial linear models prevails and thus the polynomial component can only be regarded as a rough description of the surface being approximated.

In this paper we address this difficulty of partial linear models. One of the goals of the technique presented herein is to maintain the accuracy advantages of partial linear models, while trying to improve their comprehensibility on highly non-linear domains by decreasing the need for local corrections. This is accomplished by a kind of focusing effect. Instead of building a single partial linear model, we partition the input space[1] into regions and obtain one partial linear model for each of these regions. We manage to decrease the need for kernel adjustments by finding partitions with low variance in the target variable. Linear polynomials can easily fit this type of regions and thus the kernel corrections are smaller. We have used a regression tree as the method for obtaining the partitions. Tree-based methods (e.g., Breiman et al., 1984) use a fast divide and conquer algorithm that splits the given sample in a set of partitions. Moreover, these models provide an intelligible

[1] The multidimensional space defined by the input variables.

description of the obtained partitions. Contrary to "standard" regression trees that fit a constant in each partition we propose to use a partial linear model, leading to what we have called *partial linear trees*.

This paper starts with a description of the basic techniques leading to partial linear trees. We then present a series of experiments designed to evaluate different aspects of partial linear trees. Finally, we describe related work and the conclusions of this paper.

2. Partial Linear Regression

Partial linear regression (Spiegelman, 1976; Hardle, 1990) is a semiparametric technique that integrates a linear polynomial with a kernel smoother. From a Machine Learning perspective this can be seen as integrating a model-based approach (eager learner) with an instance-based technique (lazy learner). Given a query case a prediction using these models is obtained by summing the value predicted by a linear polynomial with the value resulting from smoothing (averaging) over the errors of the polynomial in the neighboring training points. The more inadequate the polynomial is to the given training sample the larger the importance of the smoothing component. In the extreme case where the polynomial perfectly fits the training data, a partial linear model reduces to a standard least squares linear polynomial.

Given a data set, $\{< \mathbf{x}_i, y_i >\}^n_{i=1}$, where $\mathbf{x}_i$ is a vector of input variable values, a linear polynomial of the form $Y = \beta_0 + \beta_1 X_1 + ... + \beta_a X_a$, can be obtained using a least squares error criterion. This consists of finding the vector of parameters $\boldsymbol{\beta}$ that minimizes the sum of the squared error, *i.e.* $(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})$, where $\mathbf{X}'$ denotes the transpose of the design matrix[2] $\mathbf{X}$. The minimization of this expression with respect to $\boldsymbol{\beta}$ leads to a set of equations usually known as the *normal equations* (e.g., Draper & Smith, 1981). Solving these equations with respect to $\boldsymbol{\beta}$ we obtain the values of the terms in the linear polynomial.

A kernel smoother (Watson,1964; Nadaraya,1964) can be seen as a form of lazy learner (Aha,1997) that delays learning till prediction time. Given a query point $\mathbf{x}_q$, a prediction is obtained using the expression

$$ks(\mathbf{x}_q) = \frac{1}{SKs}\sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times y_i \qquad (1)$$

where $d(.)$ is a distance function between instances; $K(.)$ is a kernel weighing function; h is a bandwidth neighborhood size) value; $<\mathbf{x}_i, y_i>$ is a training instance; and SKs is the sum of the weights of all training cases, i.e. $SKs = \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right)$.

This formula is a weighed average of the target values of the training cases that are nearer to the query point. The notion of neighborhood implies the definition of a metric over the multi-dimensional space defined by the input variables and of a distance function between any two cases in this space. The bandwidth size, h, defines the size of the neighborhood that "enters" the averaging process. The kernel function, $K(.)$, provides a smoothing effect, giving more weight to nearer training cases. Many different variants of all these "parameters" of kernel smoothing are described in the literature (e.g., Atkeson et al.,1997). In this paper we have used an Euclidean distance function and a gaussian kernel. As bandwidth we have used the distance to the 10^{th} nearest training case.

Partial linear models integrate a linear polynomial with a kernel smoother applied to the residuals (errors) of the polynomial. The kernel smoother provides an estimate of the error of the linear polynomial for the particular query case under consideration. This estimated error is then added to the linear polynomial prediction giving the final predicted value. Formally, this can be described by

$$pl(\mathbf{x}_q) = \boldsymbol{\beta}\mathbf{x}_q - \frac{1}{SKs}\sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times e_i \qquad (2)$$

where e_i is the error of the linear model in case $\langle \mathbf{x}_i, y_i \rangle$, given by $e_i = \boldsymbol{\beta}\mathbf{x}_i - y_i$.

To obtain a prediction for a query case using a partial linear model we first calculate the predicted value of the linear polynomial, $\boldsymbol{\beta}\mathbf{x}_q$. We then obtain the error of the linear polynomial in the training cases that are nearer to the query point, $\mathbf{x}_q$. Using these errors we get a kernel prediction of the error for the query case. These two values give the prediction of the partial linear model according to Equation 2.

Compared to linear polynomials, partial linear models have significant advantages when the domain is non-linear. On the other hand, when compared to kernel smoothers they provide some degree of intelligibility due to the use of a linear polynomial. However, as they also incorporate a kernel component (that is not comprehensible), they are less intelligible than linear polynomials. In effect, the polynomial component can only be regarded as a rough description of the true surface approximated by these models because of the kernel adjustments. The accuracy of this description is proportional to the linearity of the domain under study.

[2] Whose lines are the n input vectors $\mathbf{x}_i$.

3. Regression Trees

Regression trees (e.g., Breiman et al., 1984) are non-parametric models that are sometimes called *piecewise constant regression models* as they partition the input space in a set of regions and fit a constant value within each one.

Regression trees are built using a fast recursive partitioning algorithm. This algorithm starts with all available training data and at each step decides whether to stop partitioning the data or to split it further. Splitting is done according to a logical test on one of the input variables. This test divides the data in two partitions: one with the cases satisfying the test and the other with the remaining. The same algorithm is then recursively applied to the two resulting partitions creating a binary tree with logical tests on each node. Stopping the partitioning process has to do with the problem of overfitting the data and leads to the widely explored issue of pruning trees. When the partitioning is stopped a leaf node is created and a constant value is fitted to the data in the leaf. Each path from the top node of the tree to any leaf can be seen as a conjunction of logical test on the input variables describing a partition of the given training data. This partition is formed by the cases that logically entail the conjunction of tests.

The choice of the best split for a test node depends on the criterion used to evaluate the accuracy of the tree. The usual criterion is the minimization of the mean squared error (MSE), although alternatives exist like the mean absolute deviation (e.g., Torgo, 1999). Using the MSE criterion the error of a tree T can be measured by,

$$MSE(T) = \sum_{l \in \tilde{T}} P(l) \times MSE(l) = \sum_{l \in \tilde{T}} \frac{n_l}{n} \frac{1}{n_l} \sum_{D_l} (y_i - k_l)^2 = \frac{1}{n} \sum_{l \in \tilde{T}} \sum_{D_l} (y_i - k_l)^2 \quad (3)$$

where $P(l)$ is the probability of a case falling into leaf l; n is the total number of training cases; D_l is the set of training cases in l with cardinality n_l; k_l is the constant at the leaf l; and $\tilde{T}$ is the set of leaves of T.

The constants k_l that minimize this expression are the average target variable values of the cases in each leaf.

A split test s divides the current partition in two sub-sets depending on the outcome of the test. One can evaluate a split s by a weighed average of the error on each sub-partition, *i.e.*

$$MSE(s,t) = \frac{n_{t_L}}{n_t} \times MSE(t_L) + \frac{n_{t_R}}{n_t} \times MSE(t_R) \quad (4)$$

where t_L is the left child node of t defining a partition D_{t_L} that contains the set of cases that satisfy the test s, and n_{t_L} the cardinal of this set; and t_R is the right child node of t defining a partition D_{t_R} that contains the set of cases that do not satisfy the test s, and n_{t_R} the cardinal of this set.

Given this expression the best split for a node t is the split that maximizes the difference $MSE(t)$–$MSE(s,t)$, *i.e.* maximizes the decrease in error of the tree. Torgo (1999) has shown that if we use an efficient formula for calculating the MSE, $\sum_{i=1}^{n} y_i^2 \Big/ n + \left(\sum_{i=1}^{n} y_i \Big/ n \right)^2$, the maximization of the above mentioned difference is equivalent to maximizing

$$\frac{S_L^2}{n_{t_L}} + \frac{S_R^2}{n_{t_R}} \quad (5)$$

where $S_L = \sum_{D_{t_L}} y_i$ and $S_R = \sum_{D_{t_R}} y_i$.

This formulation allows a faster evaluation of all candidate splits for a given node t.

4. Partial Linear Trees

Partial linear trees combine partial linear models with regression trees. One of the goals of this integration is to maintain the accuracy of partial linear models while improving its comprehensibility by reducing the importance of the kernel adjustments. Partial linear trees can also be seen as a form of improving the accuracy of regression trees by using more complex functions in the leaves (e.g., Torgo, 1997). From the perspective of instance-based learning, partial linear trees can be regarded as a form of improving its computational complexity by focusing the search for the most similar cases on sub-sets of the overall training sample.

Regression trees grown with the MSE criterion search for partitions with low variance. We claim that partial linear models obtained within such partitions need smaller kernel adjustments because most of the variance in the data can be captured by the linear polynomial component. Thus the predictions of the partial linear model are easily "explained" by this component, which is more intelligible to the user. Figure 1 illustrates this argument with a small example. The training data consists of 10 cases described by a single input variable (X). Figure 1 (a) shows a linear polynomial obtained with this data and the surface approximated by the corresponding partial linear model.

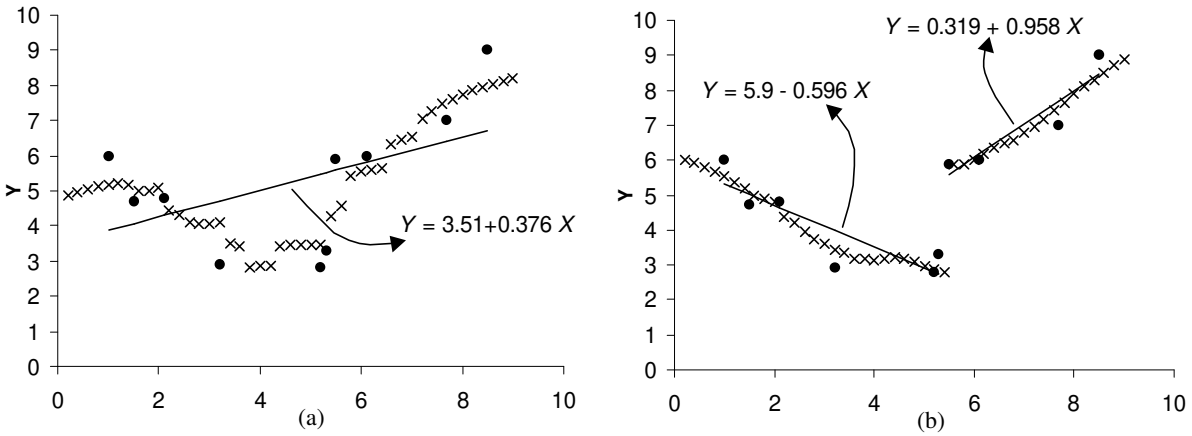

Figure 1. (a) A partial linear model; (b) a partial linear tree. The training data is shown with dots; the surfaces approximated by the partial linear models are represented with "x's"; and the lines show the linear polynomial components of the partial linear models.

As it can be seen there is a large amount of fitting error of the linear polynomial due to non-linearity of the data. This leads to large "corrections" made by the kernel component. Because of this the linear polynomial can hardly be seen as an intelligible description of the surface approximated by the partial linear model. If we apply a regression tree to this data we obtain a tree with two leaves. If we fit a partial linear model to each partition we get the result shown on Figure 1 (b). The two linear polynomials have an easier fitting task as there is lower variance within each partition. Their fitting error is smaller and thus the resulting partial linear models have less need for kernel corrections when making predictions. Each of the linear polynomials are "good" approximations of the surface represented by the partial linear models. The intelligible description of the partial linear tree is

IF $X < 5.4$ THEN $Y = 5.9 - 0.596\ X$
ELSE $Y = 0.319 + 0.958\ X$.

This is a comprehensible description stating that for values of X below 5.4 we can expect the target variable to slowly decrease with the growth of X. On the contrary, for X values above 5.4, Y increases faster with the growth of X. This intelligible description captures almost perfectly the surface approximated by the partial linear tree. On the contrary, the formulation

$Y = 3.51 + 0.376\ X$

can hardly be seen as an accurate description of the partial linear model of Figure 1 (a).

Partial linear trees are grown as standard regression trees (*c.f.* Section 3). After the growth phase we use a very efficient pruning algorithm (*LSS+ChiEst*) described in Torgo (1999). This algorithm starts by generating a sequence of nested pruned trees where each element is obtained from the previous one by pruning the node with less training cases. Each tree in this sequence is then evaluated by estimating its prediction error. We use a heuristic correction to the estimate obtained with the training data based on the sampling distribution properties of the MSE. This leads to the following formulation of the error in each leaf of a tree :

$$ChiEst = MSE \times \frac{n_t - 1}{2} \times \left(\frac{1}{\chi^2_{\alpha/2, n_t - 1}} + \frac{1}{\chi^2_{(1-\alpha/2) n_t - 1}} \right) \quad (6)$$

where *MSE* is the training set estimate of the error in t; n_t is the number of training cases in node t; and $\chi^2_{\alpha,n}$ is the tabulated value of the χ^2 distribution for a confidence level α and n degrees of freedom.

After selecting the tree with smallest estimated error we obtain a linear polynomial for each leaf of the tree concluding the learning stage. In prediction tasks each query case is dropped down the tree until a leaf is reached. At this stage a partial linear model prediction is obtained using the linear polynomial and the training cases of the leaf according to Equation 2.

The main drawback of partial linear trees is that they require all training data to be stored[3] to allow the use of partial linear models in the leaves.

5. Experimental Analysis of Partial Linear Trees

In the previous section we have described partial linear trees. In this section we present a series of experiments designed to evaluate partial linear trees in terms of their predictive accuracy, computational complexity and their impact on the amplitude of kernel corrections.

[3] Note that the same occurs with partial linear models.

5.1 The Predictive Accuracy of Partial Linear Trees

In our experimental comparisons of predictive accuracy we have used the DELVE performance assessment tool (Rasmussen, 1996). This methodology assesses the performance of learning systems for different training samples sizes and provides the level of statistical significance of the difference in accuracy between any pair of methods. Compared to the more usual 10-fold Cross Validation set-up, this methodology has the advantage of also taking into account differences due to variations on the training data through the use of non-overlapping training sets. This is particularly important with tree-based models that are known to be quite sensitive to these variations (Breiman, 1996). Regarding the used data sets their basic characteristics are summarized on Table 1.

Table 1. The characteristics of the used data sets.

Data Set	*Basic Characteristics*
Abalone	4177 cases; 1 nominal variable; 7 continuous variables
D2	40768 cases; 2 cont. vars.
Pole	9065 cases; 48 cont. vars.
Elevators	16559 cases; 18 cont. vars.
Ailerons	13750 cases; 40 cont. vars.
Mv	40768 cases; 3 nom. Vars.; 7 cont. vars.
Kinematics	8192 cases; 8 cont. vars.
CompAct	8192 cases; 22 cont. vars.
CompAct(s)	8192 cases; 8 cont. vars.
Census(16H)	22784 cases; 16 cont. vars.
Census(8L)	22784 cases; 8 cont. vars.
Fried	40768 cases; 10 cont. vars.

Figure 2 shows the results of comparing the predictive accuracy of partial linear trees with partial linear models. The graph shows the significance level of the accuracy differences between the two methods. Bars on the top side (positive values) represent wins of partial linear trees (RT+PL), while negative values represent wins of partial linear models (PL). For each data set we show the results for five different training set sizes. Bars above the 95% level can be considered statistically significant.

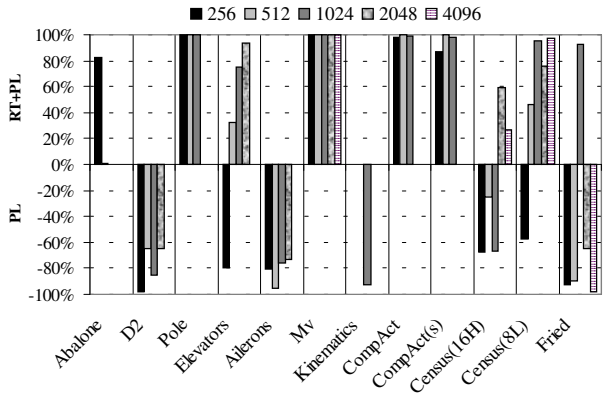

Figure 2. Significance level of the accuracy differences between partial linear trees and partial linear models.

The results of this experiment are contradictory because they depend on the data set. Still, we observed few significant losses of partial linear trees. These contradictions can be explained with the type of approximation of tree-based approaches. Regression trees divide the input space into disjoint partitions leading to a highly discontinuous regression surface. The same discontinuities occur with partial linear trees (*c.f.* example in Figure 1). The consequences of this approach in terms of predictive accuracy depend on the problem. Domains with strong discontinuities are more accurately fitted by a tree-based approach than with smoother approaches like partial linear models. This was confirmed with the artificial domain *Mv*, where such discontinuities were artificially created. In this domain we observed a clear advantage of partial linear trees. On the contrary, the artificial domain *Fried* (Friedman, 1991), was created with the goal of showing the disadvantages of the discontinuities of regression trees. As expected, partial linear trees also fail in this domain.

Partial linear trees can also be seen as a method of improving the accuracy of regression trees by including smoother models in the leaves. Figure 3 shows the results of an experiment comparing partial linear trees with regression trees. The advantages of partial linear trees in terms of predictive accuracy are overwhelming. Still, we should take into account that they come at the cost of some comprehensibility and higher computation times.

One of the motivations behind partial linear trees is to improve the intelligibility of partial linear models by decreasing the size of kernel corrections. We have also considered the hypothesis of completely dropping the kernel corrections of partial linear models. This would correspond to using only linear polynomials in the leaves

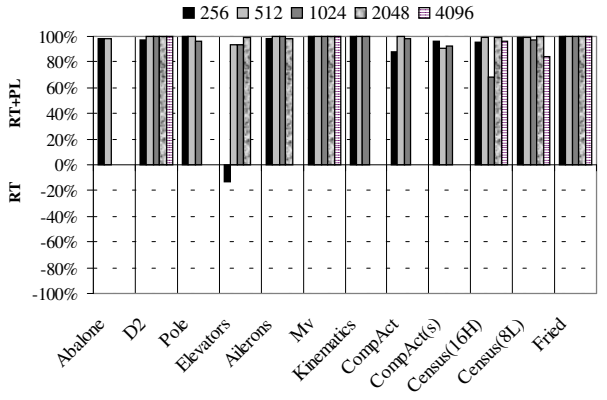

Figure 3. Significance level of the accuracy differences between partial linear trees and regression trees.

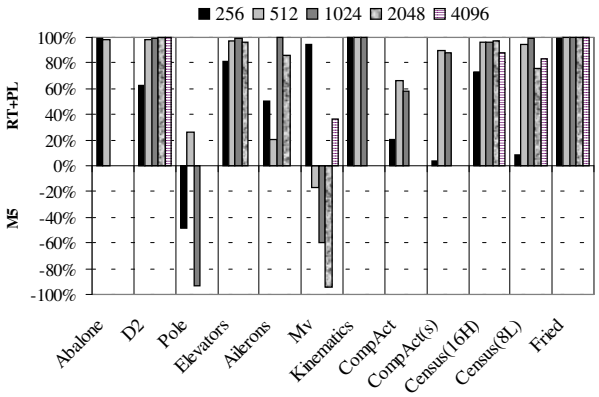

Figure 4. Significance level of the accuracy differences between partial linear trees and regression trees with linear polynomials in the leaves (M5).

of a regression tree, as in M5 (Quinlan, 1992, 1993). Figure 4 shows the results of comparing these two alternatives. These results indicate that kernel corrections remain important in terms of predictive accuracy, although decreased by the use in the leaves of a tree.

Finally, we have compared partial linear trees with three state-of-the-art regression systems. We have used a regression rule learner, CUBIST (http://www.rulequest.com); a statistical approach, MARS (Friedman,1991); and a standard regression tree, CART (Breiman et al.,1984). The results of this comparison are shown in Figure 5. This experiment shows that partial linear trees are quite competitive in terms of predictive accuracy in this large set of domains.

5.2 Computational Efficiency of Partial Linear Trees

The use of partial linear models involves a query-based lookup on the training data to search for the most similar instances. This is a computational heavy task although several techniques exist that try to decrease this cost. In partial linear trees the models are obtained using sub-sets of the training data. This largely decreases the complexity of the task of finding the most similar training cases. In this section we experimentally evaluate these gains. We have used the same data sets described in Table 1. For each data set we have collected the computation times taken to complete a full train+test cycle. These numbers were obtained for different sample sizes. For each size 70% of the data was used to train and the remaining 30% to test the models. The results we present are averages of 50 repetitions of this experimental setup.

Figure 6 shows the computation time ratio between partial linear trees and partial linear models. Values bellow one

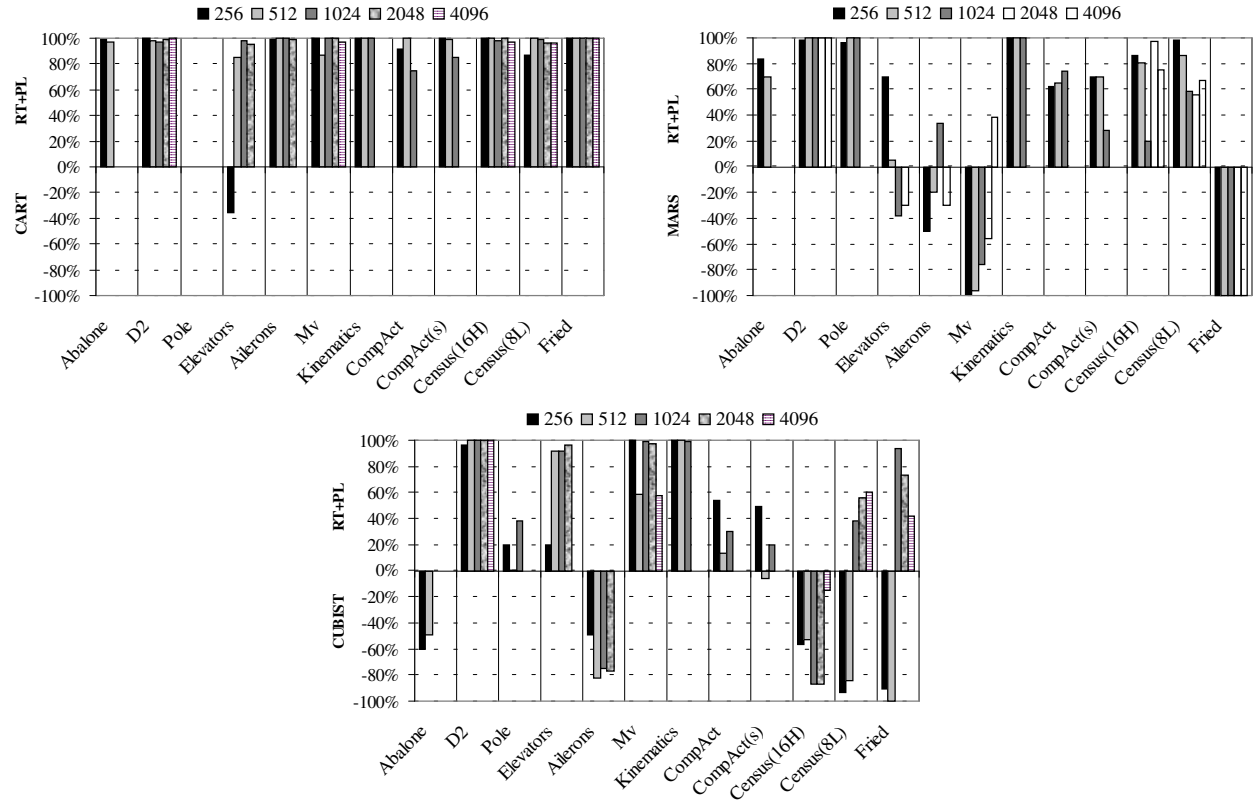

Figure 5. Significance level of the accuracy differences between partial linear trees and other regression systems.

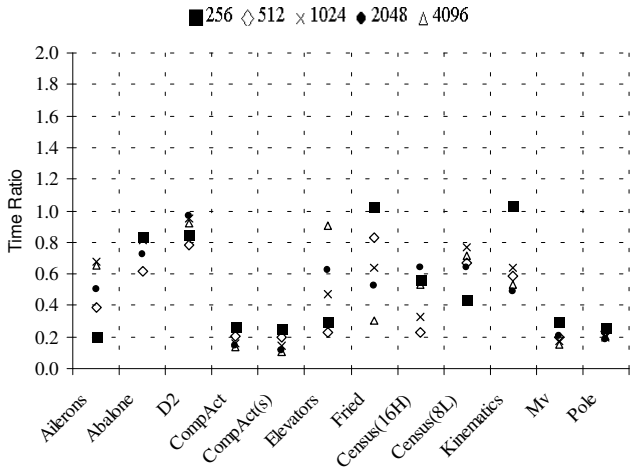

Figure 6. The computation time ratio between partial linear trees (RT+PL) and partial linear models (PL).

indicate that partial linear models took more time to complete the full train+test cycle. This experiment confirms the advantage of partial linear trees in terms of computation time. Moreover, this advantage increases with the sample size. We can expect that the gains should increase even more with larger test sets because that is the phase were the kernel component is used.

5.3 The Decrease of Kernel Corrections

We have claimed that partial linear trees where more intelligible than partial linear regression because they use smaller kernel corrections that decrease the interpretability of the model (*c.f.* Figure 1). In this Section we present an experiment designed to assert the validity of this claim. Using the same experimental design as in Section 5.2 , we have compared the average size of the kernel corrections for each test case of a partial linear model and a partial linear tree. Figure 7 shows the results

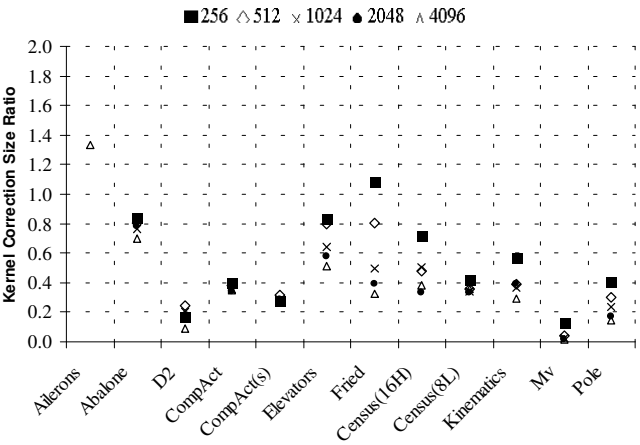

Figure 7. The Kernel Size Ratio between Partial Linear Trees and Partial Linear models.

of this experiment. The graph presents the size ratio between the corrections carried out by a partial linear tree and a partial linear model. Values bellow one indicate that the average size of the corrections of a partial linear tree are smaller. With the exception of the *Ailerons* data set this experiment strongly supports our claim. In effect, we have observed that partial linear trees use significantly smaller kernel corrections. We have investigated the reasons for the results in the *Ailerons* data set. We have concluded that this negative result was being caused by the fact that in this domain a standard linear polynomial achieved quite good results. As such a partial linear model needs very small kernel corrections, which explains the results. This can be seen as a limitation of partial linear trees.

6. Related and Future Work

Partial linear trees are implemented in system RT (http://www.ncc.up.pt/~ltorgo/RT), which learns tree-based models using different types of models in the leaves. Other authors have tried to take advantage of the different bias of lazy and eager learners through an integration of the two methodologies. Weiss and Indurkhya (1995) describe system SWAP1R that learns regression rules and uses nearest neighbors in the context of the partitions defined by each rule. Compared to our approach, this system uses a more complex partitioning method and fits less intelligible lazy models within each region.

M5 (Quinlan, 1993) learns regression trees with linear polynomials in the leaves. In prediction tasks this system combines the prediction of these trees with a nearest neighbor approach. This work differs from ours in that it performs model combination instead of model integration like we propose. This decreases the comprehensibility of the resulting model.

Zhou and Brodley (1999) described an approach (ALPL) to reduce the memory and computation requirements of lazy learning approaches. This approach consists of finding local (and possibly overlapping) regions of the input space. For each region these authors check if a linear polynomial is enough to fit the data or if a lazy model would be preferable. In the former case the data is replaced by the linear polynomial and this model is used for predictions within this region. This has a positive impact on the storage requirements and on the computation time necessary to obtain predictions for query cases within that region. Contrary to ours, this approach is not concerned with comprehensibility. However, it also improves the computation requirements of lazy models by applying them in the context of local regions of the input space.

A possible extension to the work describe in this paper is to consider the hypothesis of changing the criterion used to grow the regression trees. This criterion assumes that constants are used in the leaves of the trees (Equation 3).

Alternatively, we could assume that linear polynomials would be used in the leaves and calculate the MSE accordingly as it is done in system RETIS (Karalic, 1992). This would decrease even more the need for kernel "corrections", because the fit of the linear polynomials in each leaf would be more accurate. Still, this would also increase significantly the computation time taken to evaluate each candidate split during tree growth.

7. Conclusions

We have described partial linear trees that result from integrating partial linear models in the leaves of a regression tree. This integration was motivated by the lack of intelligibility of partial linear regression in highly non-linear domains. We have shown that using partial linear models in the context of low variance partitions decreases the size of the kernel adjustments, thus improving the intelligibility of the models.

Our experiments indicate that partial linear trees maintain the good predictive accuracy of partial linear regression, although with a different bias due to the discontinuities of tree-based approaches. Partial linear trees clearly outperform the accuracy of regression trees and also of trees with linear polynomials in the leaves. Moreover, when compared to state-of-the-art regression tools, partial linear trees proved to be quite competitive in the domains we have tested.

Finally, we have observed that partial linear trees are significantly more efficient in terms of computation time than partial linear regression.

Acknowledgements

I would like to acknowledge the insightful comments of the anonymous reviewers of this paper. This work is partially supported by project Sol-Eu-Net IST-1999-11495, by PRAXIS XXI pluriannual support to LIACC, and by ESPRIT LTR Project METAL 26.357.

References

Aha, D. (1997). *Lazy Learning*, edited by D. Aha.

Atkeson, C.G., Moore, A.W., Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, 11, 11-73. Special issue on lazy learning, Aha, D. (Ed.).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123-140.

Breiman, L. , Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and regression trees*. Belmont, California: Wadsworth Int. Group.

Drapper, N., Smith, H. (1981). *Applied regression analysis* (2nd edition). John Wiley & Sons.

Friedman, J. (1991). Multivariate adaptative regression splines. *Annals of Statistics*, *19*, 1-141.

Hardle, W. (1990). *Applied nonparametric regression*. Cambridge University Press.

Karalic, A. (1992). Employing linear regression in regression tree leaves. *Proceedings of European Conference on Artificial Intelligence*. Wiley & Sons.

Nadaraya, E.A. (1964). On estimating regression. *Theory of Probability and its Applications*, *9*, 141-142.

Quinlan, J. (1992). Learning with continuous classes. *Proceedings of the Fifth Australian Joint Conference on Artificial Intelligence*. Singapore: World Scientific.

Quinlan, J. (1993). Combining instance-based and model-based Learning. *Proceedings of the Tenth International Conference on Machine Learning*. Morgan Kaufmann.

Rasmussen, C. (1996). *Evaluation of gaussian processes and other methods for non-linear regression*. Ph.D. Thesis, Department of Computer Science, University of Toronto, Toronto, Canada.

Spiegelman, C. (1976). *Two techniques for estimating treatment effects in the presence of hidden variables: adaptive regression and a solution to Reiersol's problem*. Ph.D. Thesis. Dept. of Mathematics, Northwestern University, Evanston, Illinois.

Torgo, L. (1997). Functional models for regression tree leaves. In *Proceedings of the Fourteenth International Conference on Machine Learning*, Fisher,D. (ed.). Morgan Kaufmann Publishers.

Torgo, L. (1999). *Inductive learning of tree-based regression models*. Ph.D. thesis, Dept. of Computer Science, Faculty of Sciences, University of Porto, Porto, Portugal.

Watson, G.S. (1964). Smooth regression analysis. *Sankhya: The Indian Journal of Statistics*, Series A, *26,* 359-372.

Weiss, S., Indurkhya, N. (1995). Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, *3*, 383-403.

Zhou, Y., Brodley, C. (1999). A hybrid lazy-eager approach to reducing the computation and memory requirements of local parametric learning algorithms. *Proceedings of the Sixteenth International Conference on Machine Learning*. Bratko and Dseroski (eds.). Morgan Kaufmann Publishers.