1 Computationally Efficient Linear Regression Trees

Luis Torgo[1]

LIACC/FEP, University of Porto
R.Campo Alegre, 823
4150 PORTO, PORTUGAL
email: ltorgo@liacc.up.pt
URL: http://www.liacc.up.pt/~ltorgo

Abstract. This paper describes a method for obtaining regression trees using linear regression models in the leaves in a computationally efficient way that allows the use of this method on large data sets. This work is focused on deriving a set of formulae with the goal of allowing an efficient evaluation of all candidate tests that are considered during tree growth.

1.1 Introduction

Existing systems that build regression trees with linear models in the leaves, follow one of two paths: either they compromise the construction of the tree in order to achieve the desired computational efficiency; or they grow "correct" trees but with algorithms whose computational effort strongly impairs their applicability. In this work we describe an alternative approach that allows the construction of "correct" linear regression trees with a computational efficiency that ensures a wide range of application scenarios.

1.2 Existing Approaches to Linear Regression Trees

1.2.1 The Approach of M5

M5 (Quinlan (1992)) obtains trees using a standard least squares error criterion (*e.g.* Breiman et al. (1984), Torgo(1999)). This means that at each stage of the tree growth procedure M5 chooses the split that decreases more the variability with respect to the average in the respective partitions entailed by the split. This also means that the splits are chosen assuming that the leaves will have constant models consisting of averages. One can argue that these splits are not optimal as M5 will in effect use linear models in the leaves and not averages. The consequences of this can be easily illustrated by a simple artificial example. Let us suppose that we have a dataset generated by the function, $f(X) = \text{IF } X \leq 18.5 \text{ THEN } Y = -40+4.2X \text{ ELSE } Y = 70-1.5X$. We would expect that a system using linear models in the leaves, if given a reasonable amount of training data, would easily guess it, obtaining a tree with a single split test ($X \leq 18.5$) and with the two linear models on each of the leaves generated by this test. However, due to the criterion used by M5, this system can very easily fail to identify the best model even in this simple (although artificial) example. For instance, in a random sample with 24 observations we have generated, M5 obtained the following tree:

$$\begin{array}{l} \text{IF } X \leq 15.9 \text{ THEN } Y = -6.29 + 3.89X \\ \qquad\qquad \text{ELSE IF } X \leq 17.5 \text{ THEN } Y = 27.6 + 1.94X \\ \qquad\qquad\qquad\qquad \text{ELSE } Y = 28 + 1.94X \end{array}$$

This tree with two inner nodes and three linear models in the leaves, not only completely misses the true regression function, but it also leads to larger prediction error, and it is significantly more complex than the model that would be obtained if a "correct" criterion was used to choose the best split for each node. The reason for this failure is the use of a test selection criterion that does not take into account the models that will be used in the leaves. What is then the motivation for using such criterion as M5 does? The single motivation is computational complexity. In effect, the use of averages together with the least squares criterion allows the use of fast incremental algorithms that find the best split for each candidate variable in time linear to the number of training cases (*e.g.* Torgo (1999)).

1.2.2 The Approach of RETIS

RETIS (Karalic (1992)) is another tree-based regression system that uses linear regression models in the leaves. However, contrary to M5, RETIS chooses the tests for the tree nodes assuming that these models will be used in the leaves. This means that RETIS would find the "correct" model in the small artificial example described in the previous section.

Let us address the computational costs of the method used by RETIS. Each split node in a tree obtained by RETIS has two branches:- one with the cases satisfying the test; and other with the remaining cases. Evaluating each of these splits involves obtaining an estimate of the error on each branch. To obtain this estimate one has to obtain their respective linear models. This involves a series of standard matrix computations (*e.g.* Myers (1990)). This has to be done for all trial splits, of all variables, and for all nodes of the tree. The result is a system that hardly scales up to large problems, particularly if too many continuous variables exist[1]. This problem was not addressed by Karalic when presenting RETIS (Karalic (1992)). We claim that although the method used by RETIS can be considered more "correct" than the one used in M5, it will not scale up to large problems as well as M5 method. It is the goal of this paper to provide efficient methods of building trees similar to the ones obtained by RETIS, but without their computational drawbacks.

[1] Which are known to be the major computational bottleneck for growing tree-based models (Catlett (1991)).

1.3 A Correct and Computationally Efficient Approach

Recursive least squares algorithms (Goodwin and Sin (1984)) have been developed in model identification and adaptive control literature. These algorithms allow to identify the linear model parameters when the data is available sequentially instead of as a single batch. Bontempi (2000) has used these results in the context of local linear regression (*e.g.* Cleveland and Loader (1995)). This author has developed a series of recursive formulas that allow obtaining local linear models for increasingly larger bandwidths (sets of neighbouring training cases). Namely, they make possible to obtain the linear model parameters for a sample with $k+1$ cases using the model parameters obtained with k cases, without having to recalculate everything from scratch. Moreover, Bontempi has developed a similar recursive formulation for obtaining a reliable estimate of the leave-one-out cross validation error of the linear model using the PRESS statistic (Myers (1990)). The problem we face to obtain an efficient evaluation of all candidate splits of a continuous variable is similar. Let us now describe what we need to attain this goal. Our description assumes binary splits although the results generalize to the multiple branch case. We will focus on continuous variables that present the major computational challenges (Catlett (1991)).

The selection of the best split in a continuous variable for a node t, starts with the ordering of the observed values of the variable. After obtaining this ordering (say $\{v_1, v_2, ..., v_n\}$) we need to evaluate each possible test that leads to a different splitting of the cases through the left and right branches. Let us imagine that we are at a stage where we have finished evaluating the split $X_a \leq v_k$. This means that we have just built two linear models: one with the cases that satisfy this test (let this model parameters be $\widehat{\beta}_L(k)$); and other with the ones that do not satisfy this test ($\widehat{\beta}_R(k)$). If we want to evaluate a new split $X_a \leq v_{k+1}$, where $v_{k+1} > v_k$, we will have some cases changing from the right to the left branch. This means that we need to obtain two new models: $\widehat{\beta}_L(k+1)$ and $\widehat{\beta}_L(k+1)$. The model $\widehat{\beta}_L(k+1)$ is obtained with the cases used to obtain the model $\widehat{\beta}_L(k)$ plus some other cases, namely the cases defined by the set $J = \{x_i \in D_t : X_{i,a} > v_k \wedge X_{i,a} \leq v_{k+1}\}$. On the other hand we will need to obtain the model $\widehat{\beta}_R(k+1)$, which is obtained with the cases used to calculate $\widehat{\beta}_R(k)$ less the set J of observations. The goal in the remaining of this section is to obtain a set of formulas, which will allow us to obtain $\widehat{\beta}_L(k+1)$ and $\widehat{\beta}_R(k+1)$ based on $\widehat{\beta}_L(k)$ and $\widehat{\beta}_R(k)$.

Let us analyze the case of obtaining a new model resulting from removing a set J of observations from the set of cases used to obtain a previous model. Let this model be named $\widehat{\beta}_{-J}$. Within the regression tree context this would correspond to obtaining the model for the right branch of a new split where less cases fall in this branch (*i.e.* $\widehat{\beta}_R(k+1)$).

Using the standard *normal equations* we know that the parameters of our goal model could be obtained by,

$$\beta_{-J} = \left(X_{-J}^T X_{-J}\right)^{-1} X_{-J}^T y_{-J} \tag{1.1}$$

where X_{-J} is the matrix X with J lines (*i.e.* cases) removed; and y_{-J} the respective vector of target variable values that is obtained from y in the same way.

As mentioned before our goal is to avoid doing these calculations. Namely, inverting the matrix $X_{-J}^T X_{-J}$ that would have a cost of the order $O\left(r^3\right)$, where $r = n - j$; n is the number of lines of X; and j is the cardinality of the set J (*i.e.* the number of cases being removed from X). In effect, we want to take advantage of the model parameters obtained before the J cases were removed (i.e. $\widehat{\beta}$), thus avoinding extra computations.

Making $S = \left(X^T X\right)$, it is easy to show that the following holds,

$$S_{-J} = \left(X_{-J}^T X_{-J}\right) = \left(X^T X - X_J^T X_J\right) = S - X_J^T X_J \tag{1.2}$$

where X_J is the matrix consisting of the j lines of the matrix X (*i.e.* a $[j \times p]$ matrix, where p is the number of predictor variables of the domain).

Using the same reasoning we can derive,

$$X_{-J}^T y_{-J} = X^T y - X_J^T y_J$$

As $S\beta = \left(X^T X\right)\beta = X^T y$ (known as the *normal equations*) we have,

$$X_{-J}^T y_{-J} = S\beta - X_J^T y_J \tag{1.3}$$

From Equations 1.1, 1.2 and 1.3 we derive,

$$\beta_{-J} = S_{-J}^{-1}\left(S\beta - X_J^T y_J\right)$$

From Equation 1.2 we know that $S_{-J} = S - X_J^T X_J$, and thus $S = S_{-J} + X_J^T X_J$, which means that

$$\begin{aligned} \beta_{-J} &= S_{-J}^{-1}\left[\left(S_{-J} + X_J^T X_J\right)\beta - X_J^T y_J\right] = \\ &= S_{-J}^{-1}\left(S_{-J}\beta + X_J^T X_J \beta - X_J^T y_J\right) = \\ &= \beta + S_{-J}^{-1} X_J^T X_J \beta - S_{-J}^{-1} X_J^T y_J = \\ &= \beta + S_{-J}^{-1} X_J^T \left(X_J \beta - y_J\right) \end{aligned}$$

All this can be recast into the more convenient formulation,

$$\begin{cases} S_{-J} = S - X_J^T X_J \\ \gamma_{-J} = S_{-J}^{-1} X_J^T \\ e_{-J} = X_J \beta - y_J \\ \beta_{-J} = \beta + \gamma_{-J} e_{-J} \end{cases} \tag{1.4}$$

Although this formulation already allows us to obtain β_{-J} from β it still involves the inversion of matrix S_{-J}. However, we should remark that S_{-J}

is a $p \times p$ matrix, and as $p \ll r$ we are already far from the complexity of solving Equation 1.1.

Following the same reasoning equivalent formulas can be developped for the case of obtaining a model when we add a set of J observations $(\widehat{\beta}_{+J})$. This would correspond to the model for the left branch of a new split when new cases fall in this branch ($\widehat{\beta}_L(k+1)$ in the description given before).

The results obtained so far can still be improved by using the Sherman-Morrison formula for optimizing the inversion of S_{-J} and S_{+J}. Let us analyse the special case where the cardinality of J is 1.

The Sherman-Morrison formula states that if A is a $p \times p$ matrix, and u and v are p vectors, the following holds,

$$\left(A - uv^T\right)^{-1} = A^{-1} + \frac{A^{-1}uv^TA^{-1}}{1 - v^TA^{-1}u} \tag{1.5}$$

Making $A = S = X^TX$ and $u = v = x^T$, where x is the single case to remove from X, we have

$$\begin{cases} S_-^{-1} = S^{-1} + \frac{S^{-1}x^TxS^{-1}}{1-xS^{-1}x^T} \\ \gamma_- = S_-^{-1}x^T \\ e_- = x\beta - y \\ \beta_- = \beta + \gamma_- e_- \end{cases} \tag{1.6}$$

Using the equivalent Sherman-Morrison formula for $\left(A + uv^T\right)^{-1}$ we could obtain the formulas for adding a single case.

In this particular case where the cardinality of J is 1, the calculus of S_-^{-1} (S_+^{-1}) is strongly simplified by the use of the Sherman-Morrison formula, resulting in simple multiplications of matrices. For larger cardinality of the set J, we can either iterate the method outlined above, or we may use the Woodburry formula that can be seen as the matrix version of the Sherman-Morrison formula.

The following is an algorithm that obtains the new model parameters, β_-, after removing a single observation from the sample of data used to obtain β, based on the results of Equation 1.6. With very small changes one could easily obtain a similar algorithm for obtaining β_+.

Algorithm 1 *An $O\left(3p^2\right)$ algorithm for calculating β_-.*

```
den = 0                    % den will hold 1 − x_j S^{-1} x_j^T
FOR l = 1 TO p
    r1[l] = 0              % r1 will hold x_j S^{-1}
    r2[l] = 0              % r2 will hold S^{-1} x_j^T
    FOR c = 1 TO p
        r1[l] = r1[l] + x[c] × S[c,l]     % x[i] is the ith element of vector x_j
        r2[l] = r2[l] + S[l,c] × x[c]     % and S[a,b] is the element of matrix
    den = den + r1[l] × x[l]              % S_j^{-1} at line a, column b
```

$den = 1 - den$
$FOR\ l = 1\ \ TO\ p$
  $FOR\ c = 1\ \ TO\ p$
    $r3[l,c] = r1[l] \times r2[c]$ $\%\ r3\ holds\ S^{-1}x^TxS^{-1}$
    $r4[l,c] = S[l,c] + \frac{r3[l,c]}{den}$ $\%\ r4\ holds\ S_-^{-1}$
$p = 0$
$FOR\ l = 1\ \ TO\ p$
  $p = p + x[l] \times \beta[l]$ $\%\ p\ holds\ x\beta$
  $e = p - y$ $\%\ y\ is\ the\ target\ variable\ value\ of\ the\ observation$
  $g[l] = 0$ $\%\ g\ holds\ S_-^{-1}x^T\left(x\beta - y\right)$
  $FOR\ c = 1\ \ TO\ p$
    $g[l] = g[l] + r4[l,c] \times x[c] \times e$
  $\beta_-[l] = \beta[l] + g[l]$

1.4 Conclusions

In this paper we have addressed the issue of growing regression trees with linear models in the leaves in a computationally efficient way. We have derived a set of formulae that allow a significant reduction in the cost of evaluating all candidate test splits for each tree node. We have presented an algorithm based on these formulae, that has an average complexity of the order $O\left(3p^2\right)$. This algorithm ensures that this type of trees can be obtained in an efficient manner thus allowing the use of this method in larger data sets.

References

BONTEMPI,G. (2000): *Local Learning Techniques for Modeling, Prediction and Control.* PhD thesis, Universit Libre de Bruxelles, Belgium.

BREIMAN,L., FRIEDMAN,J., OLSHEN,R. and STONE,C. (1984): *Classification and Regression Trees.* Statistics/Probability Series. Wadsworth & Brooks/Cole Advanced Books & Software.

CATLETT,J. (1991): *Megainduction: machine learning on very large databases.* PhD thesis, Basser Department of Computer Science, University of Sidney.

CLEVELAND,W.S. and LOADER,C.R. (1995): Smoothing by local regression: Principles and methods (with discussion). *Computational Statistics.*

GOODWIN,G. and SIN,K. (1984): *Adaptive Filtering Prediction and Control.* Prentice-Hall.

KARALIC,A. (1992): Employing linear regression in regression tree leaves. In *Proceedings of ECAI-92.* Wiley & Sons.

MYERS,R. (1990): *Classical and modern Regression with Applications, 2nd edition.* Duxbury Press.

QUINLAN,J.R. (1992): Learning with continuous classes. In Adams & Sterling, editor, *Proceedings of AI'92*, pages 343–348. World Scientific.

TORGO,L. (1999): *Inductive Learning of Tree-based Regression Models.* PhD thesis, Faculty of Sciences, University of Porto.