Chapter 5

Local Regression Trees

In this chapter we explore the hypothesis of improving the accuracy of regression trees by using smoother models at the tree leaves. Our proposal consists of using local regression models to improve this smoothness. We call the resulting hybrid models, local regression trees. Local regression is a non-parametric statistical methodology that provides smooth modelling by not assuming any particular global form of the unknown regression function. On the contrary these models fit a functional form within the neighbourhood of the query points. These models are known to provide highly accurate predictions over a wide range of problems due to the absence of a "pre-defined" functional form. However, local regression techniques are also known by their computational cost, low comprehensibility and storage requirements. By integrating these techniques with regression trees, not only we improve the accuracy of the trees, but also increase the computational efficiency and comprehensibility of local models. In this chapter we describe the use of several alternative models for the leaves of regression trees. We study their behaviour in several domains an present their advantages and disadvantages. We show that local regression trees improve significantly the accuracy of "standard" trees at the cost of some additional computational requirements and some loss of comprehensibility. Moreover, local regression trees are also more accurate than trees that use linear models in the leaves. We have also observed that local regression trees are able to improve the accuracy of local modelling techniques in some domains. Compared to these latter models our local regression trees improve the level of computational efficiency allowing the application of local modelling techniques to large data sets. Finally, we have compared local regression trees with three state-of-the-art commercial regression systems, and observed that our models are quite competitive over a wide range of domains.

5.1 Introduction

In this chapter we propose to improve the accuracy of regression trees by using local models in their leaves, which leads to smoother approximations. The motivation for the integration of these two different regression methodologies lies in existing work on model selection and combination. In effect, extensive experimental (*e.g.* Michie *et. al.* 1994) and theoretical (*e.g.* Schaffer, 1994) results have demonstrated that it is difficult to identify a single best inductive method when considering a diverse set of problems. This has been termed the *selective superiority problem* by Broadley (1995) and results from the fact that different algorithms use different preference bias in the search for a model of the data. As proved by Schaffer (no free lunch theorem) it is always possible to find a data set for which any preference bias is inadquate. The same kind of difficulty arises when searching for the best parameter setting for a particular data set within a single learning method. Different parameter settings lead to different models and we need to select one of these. Moreover, there is no clear best setting over all possible domains. There have been two main distinct pathways for dealing with the selective superiority problem: model selection and model combination. Before presenting local regression trees that integrate ideas from local modelling and regression trees, we briefly describe the existing work on these two main strategies for addressing the selective superiority problem.

Model Selection through Resampling

One way of trying to address the selective superiority problem is to use resampling to perform a data-oriented model selection. Using resampling we can obtain estimates of the performance on unseen data of the different models and use these estimates to carry out a selection of the more adequate model (Schaffer, 1993b). However, this can be tedious, if the number of different algorithms is large. Moreover, resampling methods are always limited by the amount of "information" present in the given training data, which may lead to wrong decisions (Schaffer, 1993a). We have confirmed this by observing that pruning guided by resampling estimates may lead to worst trees than not pruning at all (*c.f.* Figure 4.23), meaning that the estimates were misleading.

Model Selection through Meta-learning

Another alternative for addressing the problem of selecting the most adequate algorithm for a given problem consists of trying to obtain meta-knowledge that somehow characterises the applicability of different systems (Aha, 1992; Brazdil *et. al.* 1994; Gama & Brazdil, 1995). Gama and Brazdil (1995) obtained this meta-knowledge with the help of an inductive algorithm. The data used in this meta-learning task contained information of previous experiments carried out with the alternative systems together with a set of attributes characterising each individual data set[53].

Combining Predictions of different Models

One way of dealing with the fact that different algorithms use different preference bias that can be useful in any domain, is to combine the predictions of the resulting models. Earlier work related to the combination of different predictions exploited the notion of redundancy. These notions are related because when we have redundant knowledge we need to solve conflicts and combine evidence (Torgo, 1993b). Cestnik and Bratko (1988) induced a set of redundant rules each characterised by a credibility score, which were then

---

[53] A recently approved European Community research project called METAL (http://www.ncc.up.pt/liacc/ML/METAL/) is expected to give a major boost to this research line.

used to classify new objects. This idea of redundancy was taken further in the work by Gams (1989), Buntine (1990), Kononenko (1991) and Torgo (1993a), among others. Following this initial thread, a series of works with stronger theoretical foundations has emerged. Among these we can mention Stacked Generalization (Wolpert, 1992), Boosting (Freund,1995) and Bagging (Breiman, 1996). All these approaches proved to be successful in terms of leading to a significant accuracy increase. However, all of them have drawbacks in terms of model comprehensibility. In effect, we no longer obtain a single model that can be used to "explain" the predictions.

Combining Models instead of Predictions

Brazdil & Torgo (1990) combined individual models producing a single integrated theory, instead of combining their predictions. Their INTEG system used randomly obtained sub-samples of the training set to generate several alternative models using different learning algorithms. In a subsequent phase the individual models were translated into a common rule-based representation language. Each rule was evaluated using an independent set of data and this evaluation was used to build a single integrated model. The authors reported significant accuracy gains using this method on some benchmark data sets, although this approach requires that certain assumptions have been met. The first is the necessity of a common model representation language to which all individual models must be translatable. The second is the requirement of a separate evaluation sample. While the latter can be considered more or less irrelevant in the light of the current trend in data set sizes, the former could be a problem for some kind of systems (*e.g.* a neural network). Still, this approach has the key advantage of producing a single comprehensible and accurate model of the given training set. Domingos (1997) presented a somewhat related approach that also generates a single final model. His CMM system starts by generating multiple models using variations of the training set. In a second step CMN creates a new training set consisting of all original cases plus a new set of cases appended at the end. This new data consists of randomly generated cases with the goal variable value given by the combined classification of the former individual models. This new training set is then used in a kind of second-level learning step that produces the final model.

Another alternative form of combining models consists of performing a tighter integration of the methodologies. Instead of integrating the models generated by different algorithms, we somehow integrate the methodology behind the algorithms aiming at being able to take advantage of their different preference biases. For instance, the RISE system (Domingos, 1996) integrates instance-based learning with rule-based induction. The integration is achieved by using a single algorithm that does not distinguish between rules and cases and uses the same representation language for both models. Another example of tight integration occurs when multiple models are used to extend the representation language of the learning systems. Gama (1998) describes a methodology that performs a kind of constructive induction that extends the number of attributes describing each case using other models to generate probability class distributions. In the more general framework of Cascade Generalisation (Gama, 1998), several learning systems are applied one after the other using increasingly more complex descriptions of the cases.

In summary, all these systems using tighter integration schemas try to take advantage of the different biases of several basic learning systems. This is the same motivation guiding the work described in this chapter, where we integrate local modelling with regression trees. Section 5.3 describes how we perform such integration. Before presenting such description we analyse in detail local modelling techniques.

5.2 Local Modelling

In Section 2.4.1.2 we have briefly described a set of regression methods usually known as local models that belong to a class of so called 'lazy learning' algorithms. These techniques have as main distinctive feature the fact that they do not obtain any comprehensible model of the given training data that could be stored for future use. In effect, the training phase of such methods consists basically of storing the training instances. Given a query case, local modelling methods search for the training cases in the local neighbourhood of the query case and use them to obtain the prediction. The fact that no rigid global functional form is assumed makes these techniques highly flexible as they can theoretically fit any regression curve.

5.2.1 Kernel Models

Kernel regression (Watson, 1964; Nadaraya, 1964) is a well-known local modelling method. Predictions for query cases are obtained by averaging over the *Y* values of the most similar training cases. The central issue of these models is thus the notion of similarity, which is determined using a particular metric over the input space. Given such metric we can calculate the distance between any two cases. Different distance functions exist like the Euclidean distance, L_p norms, etc. (see for instance Atkeson *et al.*, 1997 for an overview). In our work here we use an Euclidean function defined as,

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{v=1}^{a} w_v \times \delta(\mathbf{X}_{i,v}, \mathbf{X}_{j,v})^2} \tag{5.1}$$

where,

$\mathbf{x}_i$ is an instance vector; $\mathbf{X}_{i,v}$ is the value of variable v on instance i;
w_i is the weight of variable i;
and $\delta(v_1, v_2)$ is the distance between two variable values.

In order to avoid overweighing of discrete variables with respect to continuous variables we use a ramping function (Hong, 1994) for the latter. This leads to the following definition of distance between two variable values:

$$\delta(v_1, v_2) = \begin{cases} 0 & \text{if nominal variable and } v_1 = v_2 \\ 1 & \text{if nominal variable and } v_1 \neq v_2 \\ 0 & \text{if numeric variable and } |v_1 - v_2| \leq T_{eq} \\ 1 & \text{if numeric variable and } |v_1 - v_2| \geq T_{diff} \\ \left(|v_1 - v_2| - T_{eq}\right) \big/ \left(T_{diff} - T_{eq}\right) & \text{if numeric variable and } T_{eq} < |v_1 - v_2| < T_{diff} \end{cases} \tag{5.2}$$

where,

T_{eq} is a threshold for considering two numeric values equal;
and T_{diif} is a similar threshold for difference.

The distance between two numeric values can be better illustrated by Figure 5.1:

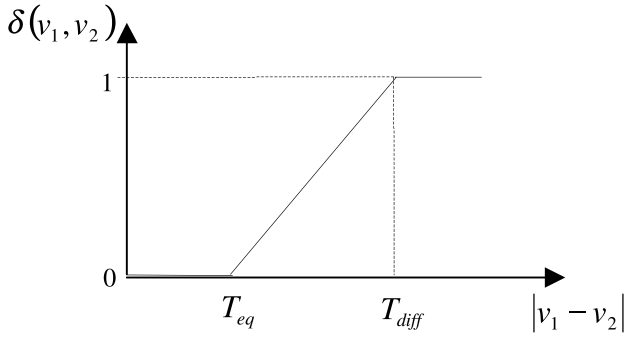

Figure 5.1. The ramping function (Hong, 1994).

Another important issue in the design of a distance function is feature weighing. Distance-based learners like kernel models can be quite sensitive to the presence of irrelevant features. These irrelevant variables can distort the distance between two cases. Feature weighing can help to reduce the influence of irrelevant features by better "tuning" the distance function. Several different approaches exist in the literature (see Wettscherek *et. al.*, 1997, for a review in the context of *k*-Nearest Neighbours). Recently, Robnik-Sikonja & Kononenko (1996) presented a method for estimating variable relevance in the context of regression, based on a previous work on *Relief* (Kira & Rendell, 1992; Kononenko, 1994). They described an algorithm called *RReliefF* and successfully applied it in the context of regression trees. We use this method for estimating the weights of the variables that are used in distance calculations.

The distance function defined above can be used to calculate the distance between a query point and any case belonging to the training data. Kernel models obtain the prediction for a query case using a weighed average of the *Y* values of the training cases within a certain neighbourhood. The size of this neighbourhood is another key issue of kernel regression and local modelling in general. This size is determined by what is usually known as the *bandwidth parameter, h.* Many alternatives exist in the vast literature of kernel modelling (*e.g.* Cleveland & Loader, 1995) that describe how this parameter should be set. The simplest strategy consists of using a maximum distance to the test case $\mathbf{x}_q$ as the bandwidth value (Fan & Marron, 1993). Nearest neighbour bandwidths (Stone,1977; Cleveland,1979), choose the distance to the k^{th} nearest training case as the bandwidth size. This was the method we have adopted. Other hypothesis include the use of optimisation processes that find the best value of the bandwidth either for each test point or globally (*e.g.* Atkeson *et al.*,1997).

All the training points within the specified bandwidth are used to calculate the prediction for a given query case. However, they enter with different weights. Training points that are nearer to the test case are given more importance. This weighing schema is accomplished through the use of what is usually known as the *kernel* (or *weighing*) *function*. We use a gaussian kernel function with the form,

$$K(d) = e^{-d^2} \tag{5.3}$$

where, d is the distance between the query point and the training cases under consideration.

Atkeson *et. al.* (1997) claim that the choice of the kernel function is not a critical design issue as long as the function is reasonably smooth. Still, other alternatives include tricube functions (Cleveland,1979), quadratic functions, etc. Notice that when the bandwidth parameter is set to the distance of the k^{th} nearest training case and we use a uniform kernel[54], kernel regression is in effect a k Nearest Neighbour model (Fix & Hodges,1951; Cover & Hart,1967).

Having defined the major design issues of kernel regression we can obtain the prediction for query point $\mathbf{x}_q$ using the following expression,

$$r(\mathbf{x}_q) = \frac{1}{SKs} \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times y_i \tag{5.4}$$

where,

$d(.)$ is the distance function between two instances (Equation 5.1);
$K(.)$ is a kernel function;

[54] A uniform kernel function gives the same weight to all cases within the bandwidth.

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

5.2.2 Local Polynomial Regression

Local polynomial regression is a generalisation of kernel regression. In effect, kernel regression can be regarded as fitting a polynomial of degree zero (a constant) in the neighbourhood of a given query point. Kernel regression is known to suffer from the lack of symmetry near the input space boundaries (Hastie & Loader, 1993). These difficulties lead to the development of local polynomial regression where a polynomial of degree p is fitted in the neighbourhood of the query point (Stone, 1977; Cleveland, 1979; Katkovnik, 1979). This includes kernel regression (p=0), local linear polynomials (p=1), and other settings. Within our work on local regression trees we use polynomials of degree 1, that is local linear polynomials.

Fitting a global linear polynomial using a least squares error criterion consists of finding the vector of parameters $\boldsymbol{\beta}$ that minimises the sum of the squared error, *i.e.* $(\mathbf{Y} - \mathbf{X}\beta)'(\mathbf{Y} - \mathbf{X}\beta)$, where $\mathbf{X}'$ denotes the transpose of matrix $\mathbf{X}$. After some matrix algebra the minimisation of this expression with respect to $\boldsymbol{\beta}$ leads to the following set of equations, usually referred to as the *normal equations* (*e.g.* Draper & Smith, 1981),

$$(\mathbf{X}'\mathbf{X})\beta = \mathbf{X}'\mathbf{Y} \tag{5.5}$$

The parameter values can be obtained solving the equation,

$$\beta = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} \tag{5.6}$$

where $\mathbf{Z}^{-1}$ denotes the inverse of matrix $\mathbf{Z}$.

As the inverse matrix does not always exist this process suffers from numerical instability. A better alternative (Press *et al.*, 1992) is to use a set of techniques known as *Singular*

Value Decomposition (*SVD*), that can be used to find solutions of systems of equations with the form $\mathbf{X\beta} = \mathbf{Y}$.

Regarding local linear polynomials the main difference with respect to global linear regression, is that each training point has an associated weight that is a function of its distance to the query point. In order to fit a local linear polynomial given a query point $\mathbf{x}_q$, we need to find the model parameters that minimise,

$$\sum_{i=1}^{n} \left(Y_i - \beta\, \mathbf{x}_i\right)^2 K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \tag{5.7}$$

where,

$K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right)$ is the weight associated to each training case;
$d(.)$ is a distance function;
$K(.)$ a kernel function;
and h the bandwidth.

According to Draper & Smith (1981, p.109) in the case of weighed least squares the normal equations are,

$$(\mathbf{X}'\mathbf{W}\mathbf{X})\boldsymbol{\beta} = \mathbf{X}'\mathbf{W}\mathbf{Y} \tag{5.8}$$

where, $\mathbf{W}$ is a diagonal matrix of weights, *i.e.*, $\mathbf{W} = \mathrm{diag}(w_1, w_2, \ldots, w_n)$.

Let us define the following auxiliary value for each training case,

$$v_i = \sqrt{K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right)} \tag{5.9}$$

Using matrix notation the following relation holds,

$$\mathbf{V}'\mathbf{V} = \mathbf{W} \tag{5.10}$$

where,

$\mathbf{V} = \mathrm{diag}(v_1, v_2, \ldots, v_n)$;
and $\mathbf{W} = \mathrm{diag}(w_1, w_2, \ldots, w_n)$.

Using this relation and Equation 5.8 we have,

$$(\mathbf{X}'\mathbf{V}'\mathbf{V}\,\mathbf{X})\boldsymbol{\beta} = \mathbf{X}'\mathbf{V}'\mathbf{V}\,\mathbf{Y} \tag{5.11}$$

or by a change of variables,

$$(\mathbf{Z}'\mathbf{Z})\beta = \mathbf{Z}'\mathbf{Q} \tag{5.12}$$

where,

$\mathbf{Z} = \mathbf{VX}$;
and $\mathbf{Q} = \mathbf{VY}$.

This latter equation is in the same format as Equation 5.5, which means that it can be solved using the same numerical procedures (*i.e.* inversion of **Z** or through the SVD method). This means that to obtain the solution for a local linear polynomial given a query point, we just need to calculate the auxiliary weights, v_i (Equation 5.9), and multiply these weights by the matrices **X** and **Y**, obtaining matrices **Z** and **Q** that can be feed in the SVD routine[55].

We also use a backward elimination technique (*e.g.* Drapper & Smith, 1981) to simplify the models obtained through the SVD procedure. This is an iterative technique for progressively eliminating terms of a linear polynomial on the basis of pre-defined thresholds of a *t*-Student test.

5.2.3 Semi-parametric Models

The main idea behind semi-parametric models (*e.g.* Hardle, 1990) is to incorporate in a single model both local (non-parametric) and parametric components. For instance, *partial linear models* (Spiegelman, 1976; Hardle, 1990) integrate a linear component with a kernel model. A partial linear model can be described by,

$$Y = \boldsymbol{\beta}\mathbf{X} + m(\mathbf{x}) \tag{5.13}$$

where,

$\boldsymbol{\beta}\mathbf{X}$ is a least squares linear model with parameters $\boldsymbol{\beta}$;
and $m(.)$ is a kernel regression model.

---

[55] Notice that if we include all training cases in the matrix **X**, and use a uniform weight for all cases, we have in effect a global least squares linear regression model.

One way of using these models is to first generate the parametric component of the model (*i.e.* generate the global linear polynomial) and then apply kernel regression on the residuals (errors) of this model. Here we follow this approach. We start by calculating the linear model using the standard SVD technique. Given a query point we identify the training points within the specified bandwidth. For each of these training cases we calculate the residuals of the linear model,

$$r_i = \boldsymbol{\beta}\mathbf{x}_i - y_i \ , \quad \text{for each } \langle \mathbf{x}_i, y_i \rangle \in \text{bandwidth of } \mathbf{x}_q \tag{5.14}$$

A kernel prediction for the residual of the query point is then obtained with these "training" residuals. Finally, the predicted residual is added to the linear model prediction for the query point, giving the partial linear model prediction. Formally, this corresponds to,

$$r(\mathbf{x}_q) = \boldsymbol{\beta}\mathbf{x}_q - \frac{1}{SKs}\sum_i^n K\left(\frac{d(\mathbf{x}_i, \mathbf{x}_q)}{h}\right) \times (\boldsymbol{\beta}\mathbf{x}_q - y_i) \tag{5.15}$$

where, SKs is the sum of the kernel weights.

These models can be seen as performing a local "correction" on the global linear model predictions represented by $\boldsymbol{\beta}\mathbf{x}_q$.

5.3 Integrating Local Modelling with Regression Trees

The methods described in Chapter 3 for inducing a regression tree assume constant values in the tree leaves. In the case of LS regression trees this constant is the average Y value of the training cases, while in the case of LAD trees it is the median. We have seen that these constants minimise the respective error criteria (Theorem 3.1 and Theorem 3.2). Regression functions usually have a certain degree of smoothness. Moreover, they tend to be continuous. The approximation provided by regression trees is a kind of histogram-type surface. As an example suppose we want to model the function $Y = \sin(X)$. Let us assume that the only information we give a regression tree learner is a set of 100 cases randomly generated using the function $Y = \sin(X) + N(0,1)$, where $N(0,1)$ is a gaussian noise function with average 0 and variance 1. The approximation provided by a LS regression tree is shown in Figure 5.2:

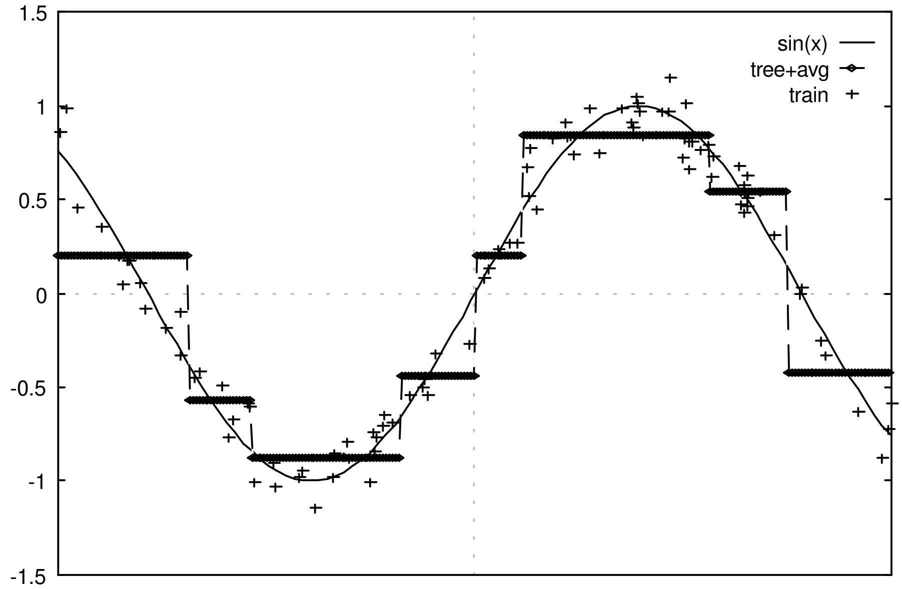

Figure 5.2. The approximation provided by a LS regression tree to the function sin(X).

The approximation of the true function provided by the tree is highly discontinuous and non-smooth curve. These two factors are usually considered the main disadvantages of tree-based regression (Friedman, 1991). In this chapter we describe approaches that integrate trees with certain smoother models so as to minimise these effects. This achieved by using different models in the tree leaves. As a motivating example, Figure 5.3 shows the approximation provided by a regression tree with kernel models in the leaves instead of averages, using exactly the same training data of the example presented above:

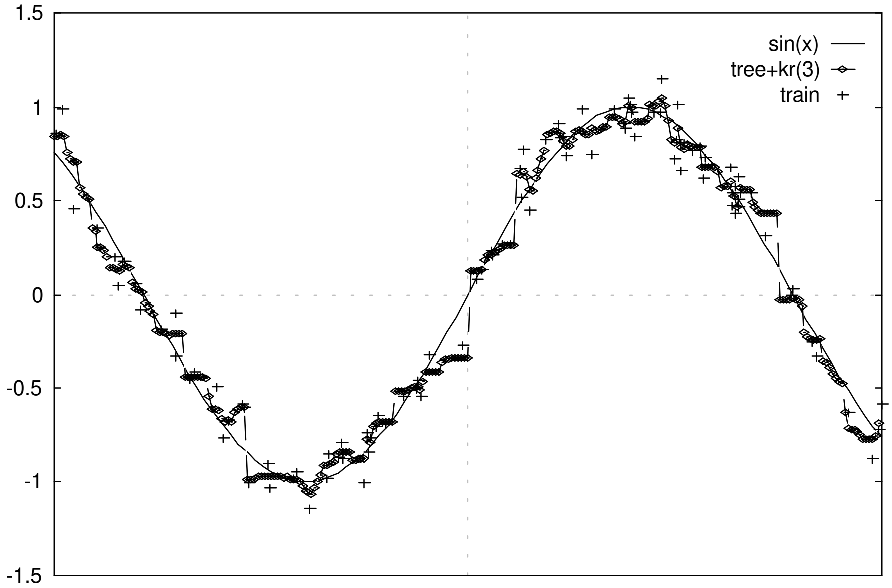

Figure 5.3. The approximation of an LS regression tree with kernel models in the leaves.

Local models by themselves provide highly smooth function approximations. However, they have several drawbacks. Firstly, these techniques do not generate any comprehensible model of the data. Secondly, they demand storing all given training data. Thirdly, they have difficulties in domains where there are strong discontinuities because they assume that nearby data points have similar target variable values. Finally, the computational complexity of local modelling methods is high when the number of training cases is large (*e.g.* Deng & Moore, 1995). This latter problem is particularly serious when using local polynomials. Several techniques exist that try to overcome these limitations. Sampling techniques, indexing schemes like *k*d-trees (Bentley, 1975; Deng & Moore, 1995) or *AD*trees (Moore & Lee, 1998), and instance prototypes[56] (Aha, 1990) are some of the methods used with large samples to decrease the computation time.

We propose to integrate local modelling with regression trees with the objective of overcoming some of the limitations mentioned earlier. On one hand, by using local models in the tree leaves we aim at achieving smoother function approximation, limiting thus the

[56] Which also have the advantage of decreasing the storage requirements.

effects of using constant value approximations. On the other hand, the use of local models in the tree leaves provides a focusing effect that allows calculating the predictions of local models more efficiently. This is achieved by avoiding going through all training points to find the appropriate nearest neighbours, limiting this search to the cases in the leaf reached by the query point. Moreover, the tree structure is a more comprehensible model of the regression surface, which is not the case when using the local models alone. Finally, the use of a tree-structured model allows us to better deal with domains with strong discontinuities. Thus the main goals of our integration proposal are the following:

- Improve standard regression trees smoothness, leading to superior predictive accuracy.
- Improve local modelling in the following aspects:
    - Computational efficiency.
    - Generation of models that are more comprehensible to human users.
    - Capability to deal with domains with strong discontinuities.

5.3.1 Method of Integration

The main decisions concerning the integration of local models with regression trees are how, and when to perform it. Three main alternatives exist:

- Assume the use of local models in the leaves during all tree induction (*i.e.* growth and pruning phases).
- Grow a standard regression tree and use local models only during the pruning stage.
- Grow and prune a standard regression tree. Use local models only in prediction tasks.

The first of these alternatives is more consistent from a theoretical point of view as the choice of the best split depends on the models at the leaves. This is the approach followed in RETIS (Karalic, 1992), which integrates global least squares linear polynomials in the tree leaves. However, obtaining such models is a computationally demanding task. For each trial split the left and right child models need to be obtained and their error calculated. We have seen that even with a simple model like the average, without an efficient incremental algorithm the evaluation of all candidate splits is too heavy. This means that if more complex models are to be used, like linear polynomials, this task is practically unfeasible for large domains[57]. Even with such incremental algorithms we have seen that there is a heavy penalty to pay with a simple model like the median. The experiments described by Karalic (1992) used data sets with few hundred cases. The author does not provide any results concerning the computation penalty of using linear models instead of averages. Still, we claim that when using more complex models like kernel regression this approach is not feasible if we want to achieve a reasonable computation time.

The second alternative is to introduce the more complex models only during the pruning stage. The tree is grown (*i.e.* the splits are chosen) assuming averages in the leaves. Only during the pruning stage we consider that the leaves will contain more complex models, which entails obtaining them for each node of the grown tree. Notice that this is much more efficient than the alternative mentioned above, which involved obtaining the models for each trial split considered during the tree growth. This latter method is the approach followed in M5 (Quinlan, 1992). This system uses global least squares linear polynomials in the tree leaves but these models are only added during the pruning stage. We have access to a version of M5[58] and we have confirmed that this is a computationally feasible solution even for large problems.

In our integration task we have followed the third alternative. In this approach the learning process is separated from prediction tasks. We generate the regression trees using any of the two "standard" methodologies described in Chapter 3. If we want to use the learned tree to make predictions for a set of unseen test cases, we can choose which model should be used in the tree leaves. These can include complex models like kernels or local linear polynomials. Using this approach we only have to fit as many models as there are leaves in the final pruned tree. The main advantage of this approach is its computational efficiency. Moreover, it also allows trying several alternative models without having to re-learn the tree. Using this approach the initial tree can be regarded as a kind of rough approximation of the regression surface which is comprehensible to the human user. On

[57] Particularly with large number of continuous variables.

[58] Version 5.1

top of this rough surface we may fit smoother models for each data partition generated in the leaves of the regression tree so as to increase the predictive accuracy[59].

In our study of integrating smoother models with regression trees we consider the use of the following alternative models in the leaves:

- Kernel models.
- Local linear polynomials.
- Semi-parametric models (partial linear models).

We will refer to the resulting models as local regression trees.

All the alternative models mentioned above are added during the prediction phase. In this aspect our system behaves like a 'lazy learner'. Given a query case we drop it down the tree until a leaf has been reached. Then we use the selected model type to obtain the prediction for the case. As we are using local models this involves obtaining the nearest training cases of the query case. This task is carried out using only the training cases in the leaf reached. This has large computational advantages when compared to approaches that need to consider all training cases when searching for the neighbours, as local modelling techniques do. In the case of partial linear models the computation of the parametric component needs to be done only once for each leaf (not for each query case).

5.3.2 An illustrative example

In this section we present a small example that illustrates the approximations provided by different models in the leaves of a regression tree. The example is simple so as to allow the graphical presentation of the obtained regression surfaces. The goal in this example is to approximate the function $f(X,Y) = \sin(X \times Y) + \cos(X+Y)$. We have randomly generated a training set with 1000 cases using this function plus some gaussian noise. Figure 5.4 shows the function and the training data:

---

[59] This strategy somehow resembles the two-tiered concept representation described in Michalski (1990) for classification problems.

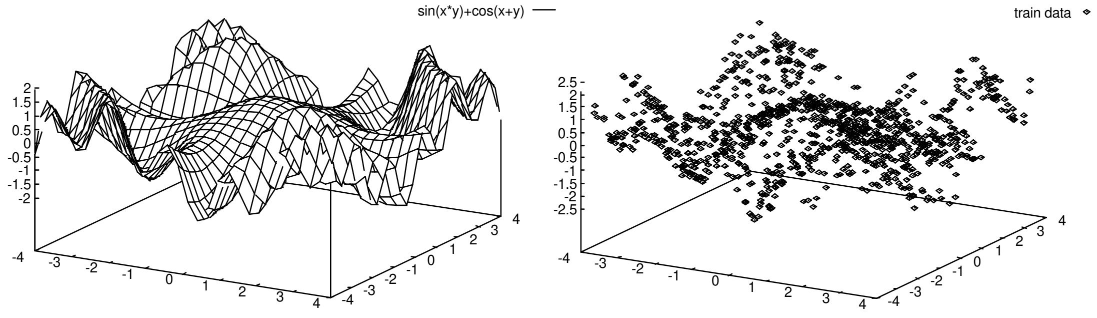

Figure 5.4. The curve and the training data.

We can use the training data to grow a regression tree and then plot the approximations provided when using the different models in the leaves. In Figure 5.5 we show the results when using averages, global linear polynomials (top part), kernel models and local linear polynomials (bottom part) in the leaves of the tree.

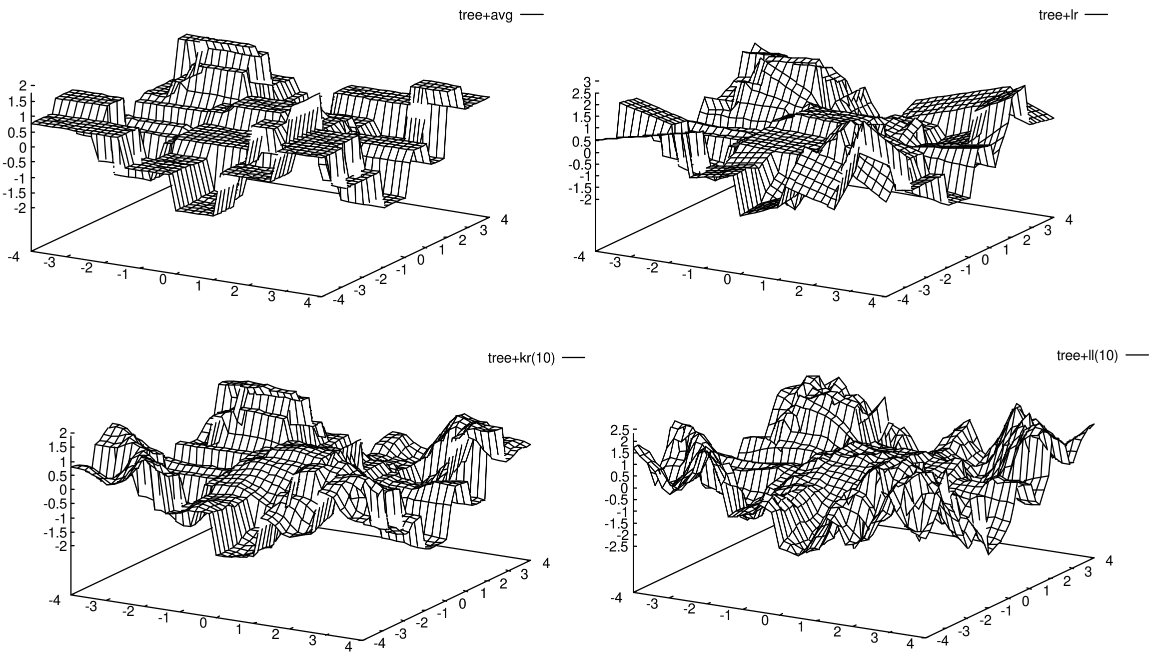

Figure 5.5. Approximations provided by using averages, linear polynomials, kernels and local linear polynomials.

Figure 5.5 shows that when using local models (kernels and local linear polynomials) the approximation is clearly much smoother.

5.3.3 Relations to Other Work

Integrating partition-based methods with other models is not an entirely new idea. Several authors have followed a similar integration schema for classification tasks. Assistant Professional (a commercial version of Assistance 86, by Cestnik *et al.*, 1987) allows the use of a naïve Bayesian classifier in the leaves of classification trees. Smyth *et. al.* (1995) integrate classification trees with kernel density estimators (*e.g.* Silverman, 1986) to obtain non-parametric estimates of class probabilities. The authors report significant improvements over decision trees and kernel density alone for certain class of problems. However, their approach deals with discrete goal variables (*i.e.* classification) and not regression as in our case. Other approaches within the classification scenario are EACH (Salzberg, 1991) and RISE (Domingos, 1996) that generalise instances into exemplars.

Deng & Moore (1995) describe a similar approach but for regression tasks. Their *multires* system integrates *k*d-trees with kernel regression producing what they call kernel regression trees. *K*d-trees (Bentley, 1975) are a method of structuring a set of records each containing a set of measured variables. They are binary trees built in a similar fashion as regression trees. However, while regression trees are built with the goal of grouping data with similar Y values, *k*d-trees try to optimise the storage of these data points in order to achieve faster access time. The consequence is that the splitting criterion of the two approaches is completely different. Moreover, while we use a single leaf of the regression tree to obtain the prediction for a test case, *multires* obtains the prediction by a combination of contributions of several nodes (not necessarily leaves). In *multires* the main issue is obtaining an efficient way of structuring all the training cases to make kernel regression computationally more efficient.

Quinlan (1992) and Karalic (1992) have used global least squares linear models in regression tree leaves. These authors follow a different integration method from ours as we have mentioned before. The M5 system (Quinlan, 1993) is also able to use regression trees and k-NN models. However, this system performs prediction combination instead of integrating the two methodologies like in our proposal. This means that two models are independently obtained and their predictions combined.

The work of Weiss & Indurkhya (1995) integrates a rule-based partitioning method with k-nearest neighbours. However, these authors deal with regression by mapping it into a classification problem. The original Y values are mapped into a set of i intervals. One problem of this approach is that the number of intervals needs to be determined, which can be computationally demanding if one wants to ensure "optimal" accuracy of the resulting models (Torgo & Gama, 1997). Moreover, the search space explored by rule learners is larger than the one of trees. This means that rule learning systems may find solutions that tree learners cannot, but at the cost of computational complexity (Indurkhya & Weiss, 1995). These two latter observations indicate that our hybrid tree learner should be able to cope with larger problems than Weiss & Indurkhya's system. Another important difference when compared to our work involves the type of local models. Weiss & Indurkhya's system uses k-NN while our system includes other more sophisticated local modelling techniques.

5.4 An Experimental Evaluation of Local Regression Trees

In this section we describe a series of experiments that have been designed to check the validity of our hypotheses concerning the advantages of local regression trees. Namely, our aim was to verify whether the variants of our system were able to overcome some of the limitations of both standard regression trees and local models. With respect to standard regression trees, we conjecture that the use of local trees brings a significant increase in accuracy due to the use of smoother models in the tree leaves. Regarding local modelling we hypothesise that local trees have three advantages: providing a comprehensible model of the regression surface; significant advantages in terms of computation time; predictive accuracy advantages in domains with strong discontinuities in the regression surface.

5.4.1 Local Regression Trees vs. Standard Regression Trees

In this section we describe a series of experiments whose objective was to check the validity of the following hypothesis:

The use of smoother models in the tree leaves improves the predictive accuracy of tree-based regression models.

We have also evaluated the costs in terms of computation time of local regression trees when compared to standard trees.

Regarding local regression trees we have considered the use of three different models in the tree leaves: kernel models, local linear polynomials and partial linear models. All these three local modelling variants were described in Section 5.2. Local modelling techniques have many parameters that can be tuned. In the experiments reported below we have not used feature weighing. The bandwidth value was set to the distance of the 10th nearest neighbour[60]. The method of growing and pruning a tree (using *LSS*+*ChiEst*(95%)) was exactly the same for all compared methods (*i.e.* standard and local regression trees). The results of comparing the accuracy of standard regression trees with local trees are shown in Figure 5.6.

---

[60] For local linear polynomials we have used a larger bandwidth (30% of all nearest neighbors) because these local models can lead to wild extrapolations if they are built on the basis of small amounts of training cases.

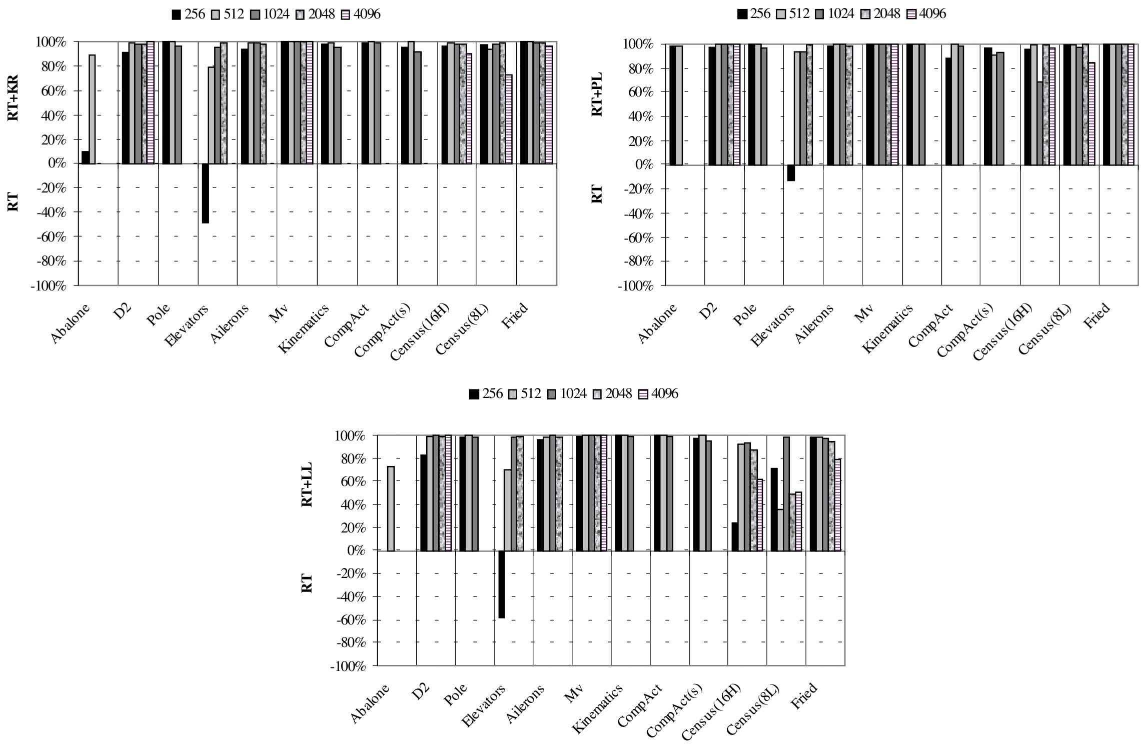

Figure 5.6 – Comparison of standard regression trees (RT) with local regression trees (RT+KR, RT+PL and RT+LL).

As these graphs show there is an overwhelming accuracy advantage of local trees, particularly with kernel (RT+KR) and partial linear models (RT+PL). However, even local trees with local linear polynomials (RT+LL) achieve a significant accuracy advantage in most domains. This experimental comparison shows that local regression trees are able to significantly outperform standard regression trees in terms of predictive accuracy. Standard regression trees (RT) provide a non-smooth approximation. The only difference between RT and the three variants of local trees is that the latter use local models in the leaves. Thus we can claim that these experiments confirm with high statistical confidence our hypothesis concerning the advantages of using smoother models in the leaves.

The accuracy gains reported above have a cost in terms of computation time. Figure 5.7 shows the computation time ratio between a standard *LSS+ChiEst*(*95%*) regression tree and the three local tree variants we have proposed. As the local trees were grown and pruned using the same procedure, the observed differences are only due to the use of local models in the leaves.

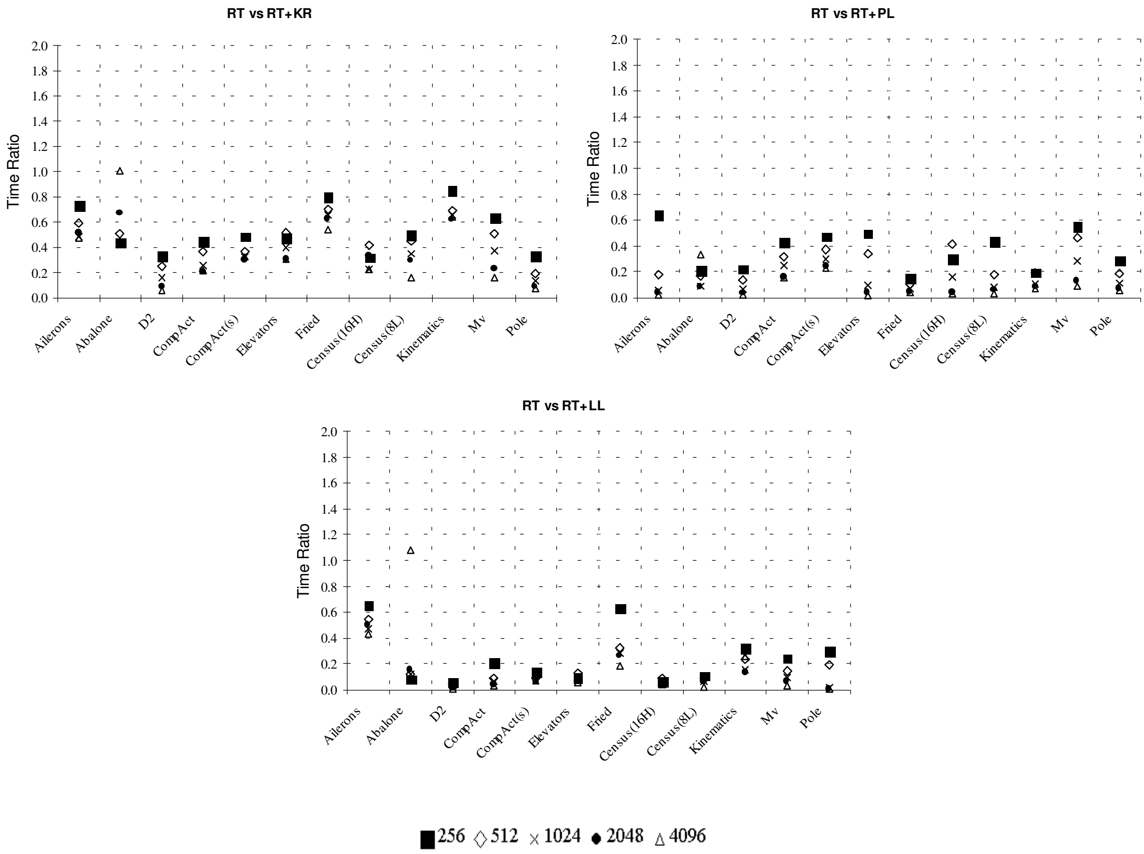

Figure 5.7 - Computation time ratio of standard and local regression trees.

These ratios show that we can expect a considerable increase of computation time due to the use of local models in the leaves. The increase is more marked for large data sets. Still, as we will see in the next section, local regression trees are much faster than the local modelling techniques alone.

5.4.2 Local Regression Trees vs. Local Models

In this section we compare local regression trees with the local models. We compare the accuracy and computation time of our three variants of local regression trees, with the local modelling techniques used alone (*i.e.* applied using all training data instead of with the data in the leaves of a tree). The local modelling techniques are the same used in the experiments reported in the previous section: kernel models, partial linear models and local linear polynomials. The models are applied with the same parameter settings as those used earlier.

The first experiment carried out was designed with the aim of checking the empirical validity of the following hypothesis:

The use of local models in the context of the leaves of a regression tree does not entail an accuracy loss and may even provide useful bias in domains with strong discontinuities.

Figure 5.8 shows the significance of the estimated accuracy difference between the local tree variants and the respective local models alone.

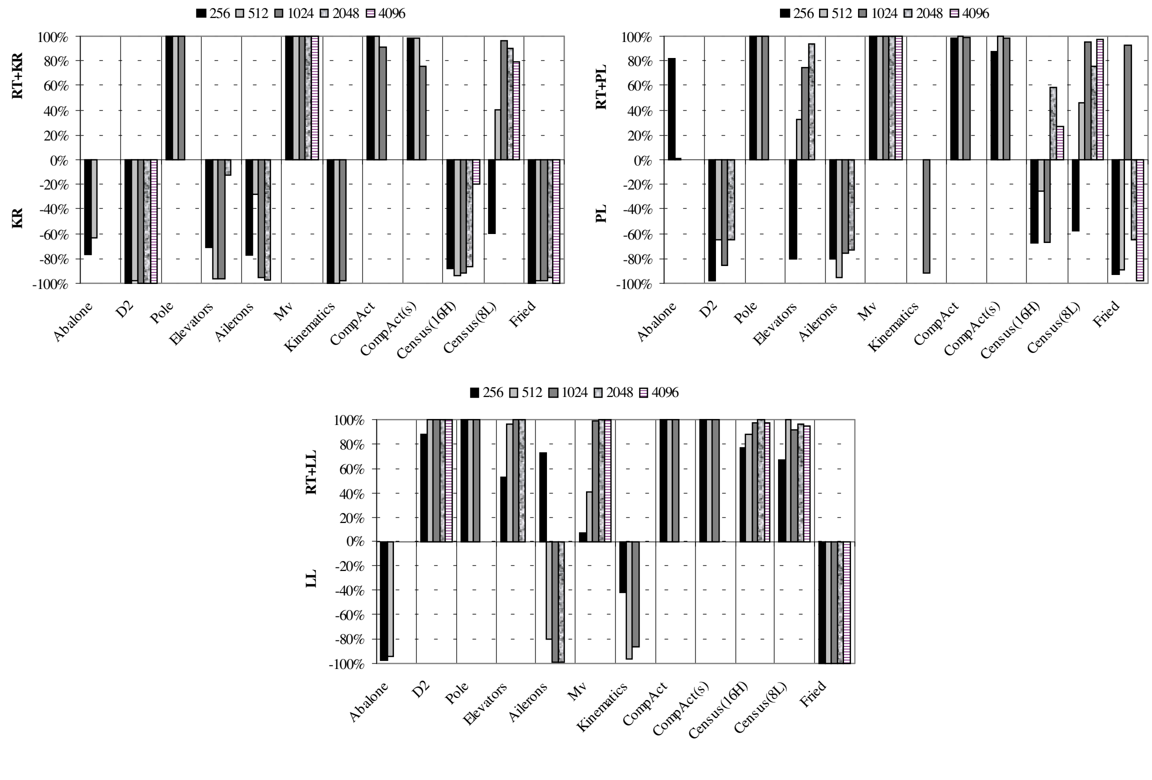

Figure 5.8 – Comparison of Local Trees and Local Models in terms of Significance of MSE difference.

The results of this comparison confirm our hypothesis. In effect, we observe that the two types of models entail quite different biases and both prove to be useful depending on the domain. For instance, we know that the regression surface of the *Mv* artificial domain has strong discontinuities (*c.f.* its definition on Appendix A.2). Our experiments confirm that local regression trees achieve better results in this domain due to the type of approximation provided. The reason for this performance is the following. A strong discontinuity occurs when nearby points of the unknown regression surface have quite different values of the goal variable. A local model is based on the notion of neighbourhood that is measured by a metric in the space of input variables. As such, these models are not able to use the information concerning the fact that two nearby points may have a very different value of the goal variable (as when facing a strong discontinuity). Thus local models will perform the usual smoothing of the goal variable values of the nearby points, which will contribute to a higher prediction error in the vicinity of the discontinuity. Regression trees, on the contrary, are based on the principle of separating points with different goal variable values because this will decrease the variance of the resulting partitions. Being so, when facing a discontinuity, a tree-based model will search for a split in the input variables that ensures a larger variance reduction, which most probably is achieved by separating the points on each "side" of the discontinuity. By separating these points in different partitions, the local models used within each one do not suffer the "bad influence" of the points on the "other side" of the discontinuity.

The other conjecture we have made regarding the comparison of local regression trees with local models was that the former are able to overcome the computation time limitations of local modelling. Figure 5.9 shows the results of comparing the computation time of local trees and the respective local models.

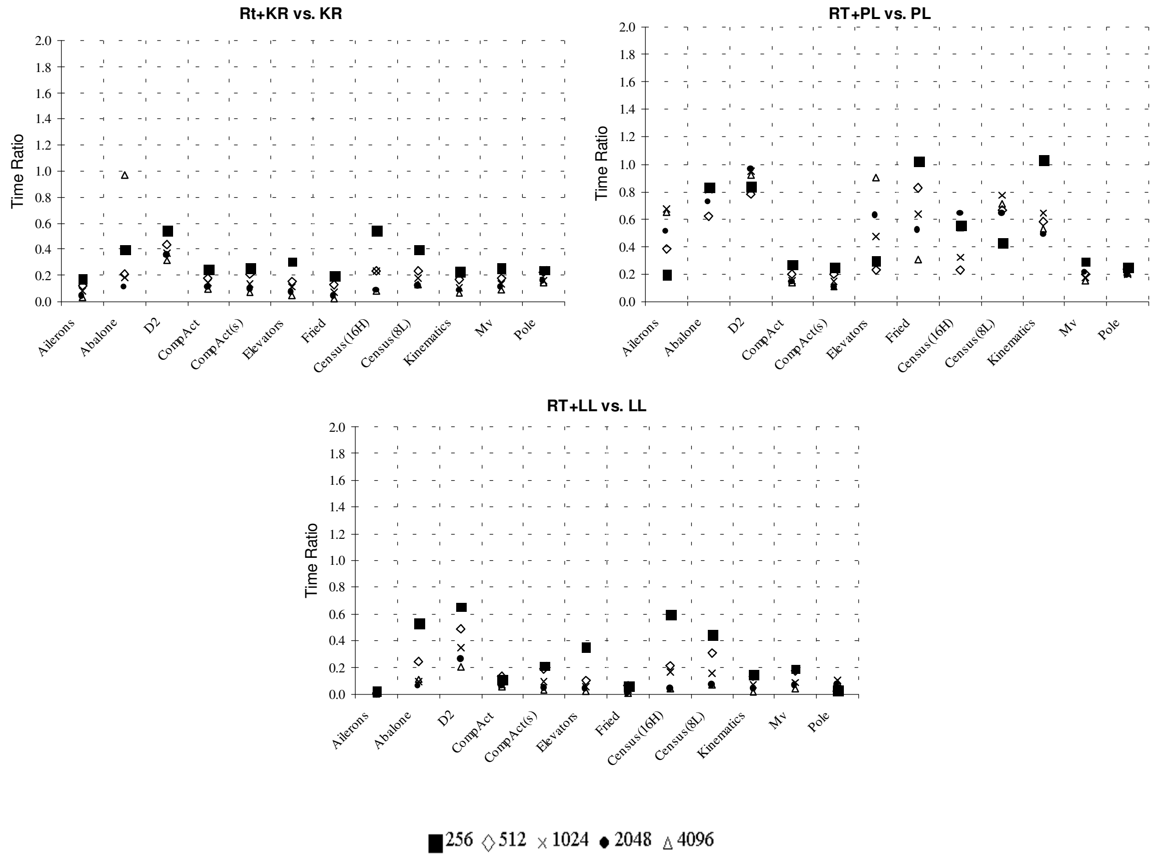

Figure 5.9 - Computation time ratio of Local Regression Trees and Local Models.

As it can be seen local regression trees successfully overcome the computational time limitations of local modelling. In effect, their advantage in terms of computation time is overwhelming in the domains we have examined.

Finally, we remark that local regression trees have another advantage over local models. In effect, local regression trees provide a symbolic representation of the regression surface, while local models do not obtain any compact model of the data. This is an important advantage despite the fact that this representation can only be regarded as a "rough" description of the true surface as it does not include a compact description of the local models used in the leaves.

5.4.3 Local Regression Trees vs. Linear Regression Trees

Other authors have tried to improve the accuracy of standard regression trees by using more complex models in their leaves. Namely, Quinlan (1992) and Karalic (1992) have used least squares linear polynomials in the leaves of regression trees. In this section we compare our local regression trees with system M5[61] (Quinlan, 1992) that obtains regression trees with linear polynomials in the leaves.

Figure 5.10 shows the accuracy comparison between our three alternative local regression trees and M5.

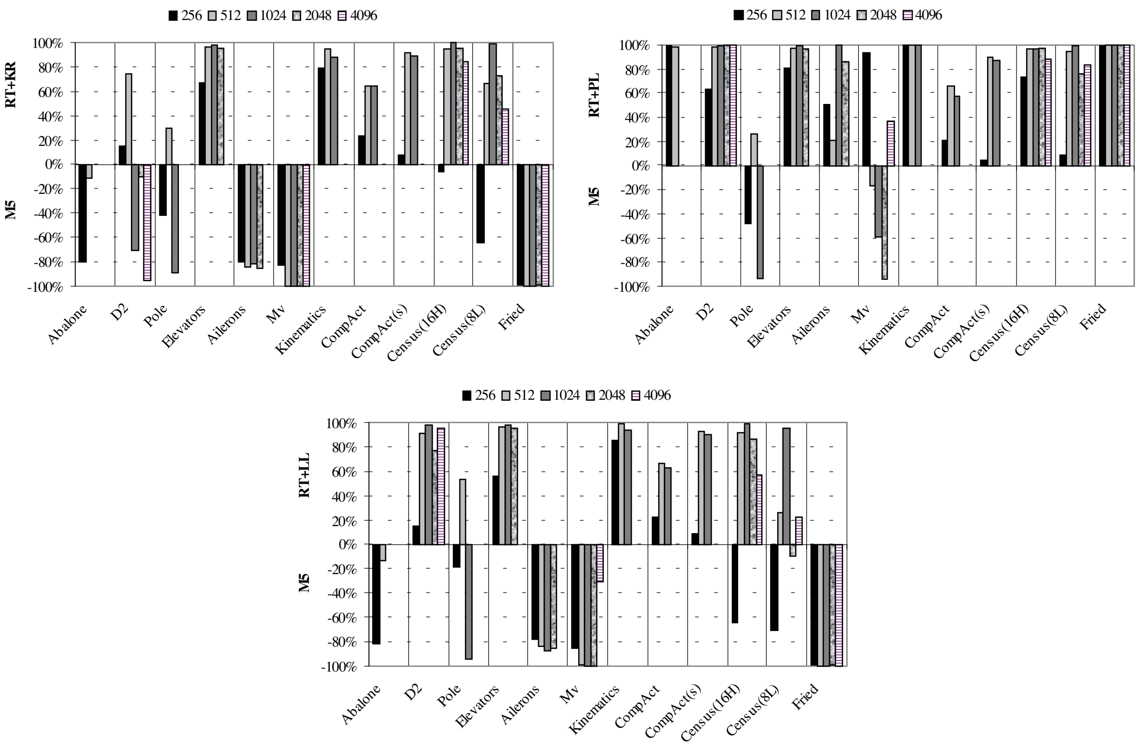

Figure 5.10 – Comparison between Local Regression Trees and M5 in terms of Significance Level of the MSE difference.

[61] M5 is also able to combine the predictions of the generated trees with *k*NN models. This feature was not used in this comparison, as the goal was to compare the use of local models in the leaves with other alternatives (in this case linear polynomials).

Figure 5.10 shows that partial linear trees[62] (RT+PL) clearly outperform M5 in most domains. Partial linear trees are similar to the models generated by M5. Both use linear polynomials in the leaves of a tree. The only difference is that partial linear models add a correction factor to the linear polynomial predictions based on a local estimation of its error (*c.f.* Section 5.2.3). These results show the importance of such smoothed correction. Both kernel regression trees[63] (RT+KR) and local linear trees[64] (RT+LL) show advantage with respect to M5 in *Elevators*, *Kinematics*, *CompAct*, *CompAct*(*s*), and *Census*(*8L* and *16H*) domains. However, they achieve worst results in the *Mv*, *Ailerons* and *Fried* domains.

5.4.4 Local Regression Trees versus Existing Regression Methods

In this section we describe a series of experiments that could determine how the best of our local regression tree variants compares with several state-of-the-art regression methods. We have selected for this comparison the following regression methods: CART (Breiman *et al.*, 1984) as a classical representative of regression trees; MARS (Friedman, 1991) as a sophisticated statistical regression tool; and CUBIST (http://www.rulequest.com) as a representative of a recent rule-based regression system. All these systems were briefly described in Chapter 2. As for local regression trees, we have selected local trees with partial linear models in the leaves (RT+PL), as the most successful variant of this type of models.

Figure 5.11 shows the results of an accuracy comparison of partial linear regression trees with CART, MARS and CUBIST.

---

[62] Regression trees with partial linear models in the leaves.

[63] Regression trees with kernel models in the leaves.

[64] Regression trees with local linear polynomials in the leaves.

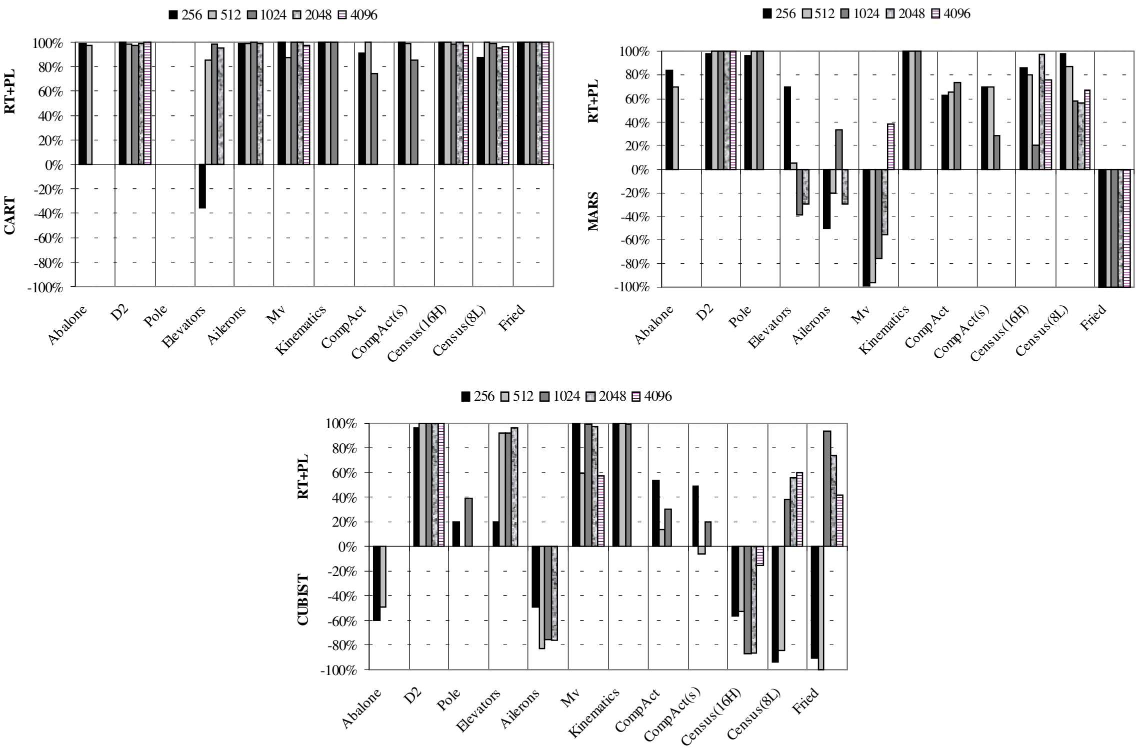

Figure 5.11 – Comparison between Partial Linear Trees and other regression techniques with respect to the Significance Level of the MSE difference.

Our system that generates partial linear trees clearly outperforms CART. In most domains the difference is statistically significant. With respect to MARS, our local trees proved to be superior in some data sets, particularly in *Pole*, *D2*, *Kinematics*, and the two *Census* domains. However, our local regression trees had difficulties in some domains like *Mv* and *Fried*[65].

Compared to CUBIST, partial linear models in the leaves of a tree proved particularly advantageous in the *D2*, *Elevators*, *Mv* and *Kinematics* domains. Less relevant advantages were achieved in the *Pole*, *CompAct* and *CompAct*(*s*) domains. CUBIST achieved some advantages in the *Abalone*, *Ailerons* and *Census*(*16H*) domains. Finally, there are two curious results in the *Census*(*8L*) and *Fried* domains. In both these two domains CUBIST

[65] We should remark that this artificial domain was obtained from the paper describing MARS (Friedman, 1991), so it is only natural that MARS behaves well on this data set.

shows advantage with small training sets, which is lost as more data becomes available. Still, the lack of statistical significance of most of these results does not allow any supported interpretation.

5.5 Conclusions

In this chapter we have presented a new type of tree-based regression models, referred to as local regression trees. We have described three variants of local trees that differ in the models used in their leaves: kernel models, partial linear models and local linear polynomials. Local regression trees are based on the assumption that using smoother models in the tree leaves can bring about gains in accuracy. We have confirmed this hypothesis by carrying out an extensive experimental comparison with "standard" regression trees. Among the three proposed variants, we have observed that the use of partial linear models in the leaves (RT+PL) led to the best overall accuracy. A comparison with the alternative of using linear models in the leaves (as in M5, for instance) revealed that local regression trees achieve better results in several domains. As partial linear models integrate both a parametric and a non-parametric component, this is a plausible explanation for the fact that partial linear trees proved to be the most competitive variant when compared to other variants of local regression trees.

Other motivations for considering the use of local models in the leaves of regression trees are the well-known drawbacks of local modelling. In effect, not only these modelling techniques are computationally demanding but they also do not provide a comprehensible model of the data. By integrating these models with regression trees we were able to overcome these two difficulties, as our experiments have shown. Moreover, we have also observed that the use of smoother models within the context of a partition-based representation can be advantageous in terms of accuracy, in domains where there are strong discontinuities on the regression surface.

Finally, accuracy comparisons with state-of-the-art regression techniques demonstrate that our partial linear trees achieve quite competitive overall results.

5.5.1 Open Research Issues

One of the drawbacks of local modelling techniques (or instance-based learning in general) is their requirement regarding the storage of all training data. This is also a drawback of our system based on local regression trees. It remains an open question whether we can adapt some of the existing techniques to improve this aspect in our system, without compromising too much its accuracy advantages.

Local modelling has many tuneable parameters. In our experiments we did not try out too many variants. It would be desirable to have some form of automatic tuning of these parameters to facilitate the task of the user.

In our study we have considered three kinds of local regression trees. All of them have shown some advantages in some domains. It is an open question whether it is useful and practical to have different local models in different leaves of the same regression tree. This could prove advantageous in domains where the regression surface is rather heterogeneous.