Kernel Regression Trees

Luís Torgo

email : ltorgo@ncc.up.pt
WWW: http://www.up.pt/~ltorgo

LIACC - University of Porto
R. Campo Alegre, 823 - 4150 Porto - Portugal
Phone : (+351) 2 6001672 Fax : (+351) 2 6003654

Abstract
This paper presents a novel method for learning in domains with continuous target variables. The method integrates regression trees with kernel regression models. The integration is done by adding kernel regressors at the tree leaves producing what we call kernel regression trees. The approach is motivated by the goal of trying to take advantage of the different biases of the two regression methodologies.
The presented method is implemented. Kernel regression trees are comprehensible and accurate regression models of the data. Experimental comparisons on both artificial and real world domains revealed the superiority of kernel regression trees when compared to the two individual approaches. The use of kernel regression at the trees leaves gives a significant performance gain. Moreover, a good performance level is achieved with much smaller trees than if kernel regression was not used. Compared to kernel regression our method improves its comprehensibility and execution time.

Keywords : Propositional learning, regression trees, instance-based methods.

Category : Full Paper

1. Introduction

In this paper we present a new method that deals with domains that have continuous goal variable (i.e. regression problems). Our method is implemented on system HTL which is a hybrid algorithm that integrates recursive partitioning with kernel regression. The learned regression model is a decision tree with kernel models on the leaves that we will refer as a kernel regression tree.

Experimental comparisons of different learning methods on various real world problems have shown the difficulty of selecting a method that performs better on all domains (Michie et al., 1994). This is sometimes called the selective superiority problem (Broadley , 1995). Several authors have tried to automatically identify the applicability of the different methods (Aha, 1992; Brazdil et al., 1994). Other line of research has tried to take advantage of the different biases of the methods and obtain a combined prediction (Breiman, 1996; Freund & Schapire, 1995; Wolpert, 1992). The main problem with these approaches is that the resulting predictions are no longer easily interpretable by the user. In effect, several different models are contributing for the final prediction[1] . Comprehensibility has always been considered a key advantage of Machine Learning (ML) approaches to the learning problem. This fact lead us to consider a different approach to the selective superiority problem. We have developed an algorithm that integrates two different approaches to regression but still provides a comprehensible model as a result of learning.

Our main goal was to obtain some performance boost without compromising comprehensibility. We have chosen to combine regression trees with kernel regression models. These methods use quite different biases for solving the problem of fitting an unknown regression function. Regression trees provide quite simple and interpretable regression models with reasonable accuracy. However, these methods are known for their instability (Breiman, 1996). Kernel regression models on the other hand, provide quite opaque models of the data, but are able to approximate highly non-linear functions and have good stability. By integrating both methods we aim at obtaining a stable and accurate system that provides comprehensible models of the data. Our experiments confirmed our expectations by showing the superiority of kernel regression trees when compared to the individual methods on both artificial and real domains. Moreover, the experiments revealed that kernel regression trees achieve significantly better accuracy than regression trees with smaller trees. This indicates that our hybrid method provides a better tradeoff between accuracy and simplicity which is considered highly relevant with real world applications (Bohanec & Bratko, 1994).

In the next section we briefly describe the regression problem and present the two approaches we propose to integrate. We then present the kernel regression tree approach on section 3. The experiments carried out are presented on section 4.

---

[1] A noticeable exception is an early work by Brazdil and Torgo (1990) where an integrated theory was built from different individually learned theories.

Section 5 relates our work to others. We them summarize with the future improvements and the conclusions of this work.

2. Regression Problems

The problem of regression consists on obtaining a functional model that relates the value of a target continuous variable y with the values of variables $x_1, x_2, ..., x_n$ (the predictors). This model is obtained using samples of the unknown regression function. These samples describe different mappings between the predictor and the target variables.

Traditional statistical approaches to this problem assume a particular parametric function and use all given samples to obtain the values of the function parameters that are optimal according to some fitting criterion[2]. These global parametric approaches have been widely used and give good predictive results when the assumed model correctly fits the data. Modern statistical approaches to regression include k-nearest neighbors (Fix & Hodges, 1951), kernel regression (Watson, 1964; Nadaraya, 1964), local regression (Stone, 1977; Cleveland, 1979), radial basis functions, neural networks, projection pursuit regression (Friedman & Stuetzle, 1981), adaptive splines (Friedman, 1991), and others.

ML researchers have always been mainly concerned with classification problems. The few existent systems dealing with regression usually build axis orthogonal partitions of the input space and fit a parametric model within each of these partitions. These models are usually very simple (like the average, the median or linear models). The non-linearities of the domain are captured by the axis-orthogonal partitions rather than by the models fitted within those partitions.

2.1 Regression Trees

Regression trees are a special type of decision trees that deal with continuous goal variable. These methods perform induction by means of an efficient recursive partitioning algorithm. The choice of the test at each node of the tree is usually guided by a least squares error criterion[3]. Binary trees consider two-way splits at each tree node. The best split at each node t is the split s that maximizes

$$\Delta Error(s,t) = Error(t) - P_L \times Error(t_L) - P_R \times Error(t_R) \qquad \textbf{(1)}$$

where P_L and P_R are the proportions of instances that fall respectively to the left and right branch of the node t
$Error(t)$ is the error at node t
and $Error(t_L)$ and $Error(t_R)$ are the errors of the left and right branches.

---

[2] One well-known example is Linear Least Squares Regression. A linear relation between the target variable and the predictors is assumed and the coefficients of this relation are calculated by minimizing the average squared error.

[3] Breiman et al. (1984) describe an absolute error criterion for growing regression trees.

Other important issues on this approach to regression are : the decision of when to stop tree growth; the related issue of how to post-prune the tree; and the model assignment rule used on the leaves. Some of the existent approaches to regression trees differ on this later issue. CART (Breiman et al., 1984) assigns a constant to the leaves (the average *y* value). RETIS (Karalic, 1992) and M5 (Quinlan, 1992) are able to use linear models of the predictor variables. We will see how our system HTL adds a further degree of smoothness by using kernel regressors at the tree leaves.

Regression trees obtain good predictive accuracy on many domains (Quinlan, 1993). However, the simple models used on their leaves have some limitations regarding the kind of functions they are able to approximate. Nevertheless, they provide very interpretable models of the data and have low computational demands (both running time and storage requirements).

2.2 Kernel Regression

Kernel regression (Nadaraya, 1964; Watson, 1964) is a non-parametric statistical methodology that belongs to a research field usually called local modeling[4] (Fan, 1995). These methods do not perform any kind of generalization of the given samples[5] and "delay learning" till prediction time.

Given a query point **q** a prediction is obtained using the training samples that are "most similar" to **q**. The notion of similarity is a key issue on these methods. Similarity is measured by means of a distance metric defined in the hyper-space of *V* predictor variables. Given the "neighborhood" of the query point **q**, kernel methods obtain the prediction by a weighted average of the *y* values of these samples. The weight of each neighbor is calculated by a function of its distance to **q** (usually called the kernel function). These kernel functions give more weight to neighbors that are nearer to **q**. The notion of neighborhood (or bandwidth) is defined in terms of distance from **q**. Formalizing, the prediction for query point **q** is obtained by

$$y'(\mathbf{q}) = \frac{1}{SKs} \sum_{i}^{N} K\left(\frac{D(\mathbf{x_i}, \mathbf{q})}{h}\right) \times y(\mathbf{x_i}) \qquad \textbf{(2)}$$

where $D(.)$ is the distance function between two instances
$K(.)$ is a kernel function
h is a bandwidth value
$\mathbf{x}_i$ are training samples
and $SKs = \sum_{i}^{N} K\left(\frac{D(\mathbf{x_i}, \mathbf{q})}{h}\right)$

---

4 Related methods exist within the ML community that are usually called lazzy learners or instance-based algorithms (Aha, *in press*).

5 This is not completly true for some kind of instance-based learners (Aha, 1990; Salzsberg, 1991).

One important design decision when applying these methods is the choice of the bandwidth. Many alternative methods exist (Loader, 1995; Cleveland & Loader, 1995). One possibility is to use a nearest neighbor bandwidth. This technique sets the bandwidth as the distance to the K nearest neighbor of **q**. This means that only the K-nearest neighbors will influence the prediction of **q**. Using this method Kernel regression turns out to be quite similar to using K-nearest neighbors in regression (Kibler et al., 1989). This is one of the bandwidth selection adopted on section 3.2.

Other design issues of kernel regression include the choice of the kernel function and the definition of a distance function. Again many alternatives exist in the literature. In the next section will discuss the choices made.

Kernel regression (and generally local modeling) can be very sensitive to the presence of irrelevant features. The computational complexity of these methods is high when a large number of samples is available (Deng & Moore, 1995). Several techniques exist that try to overcome these limitations. Feature weighing can help to the reduce the influence of irrelevant features by better "tuning" the distance function (Wettshereck et al. 1996). Sampling techniques, indexing schemes and instance prototypes are some of the methods used with large samples (Aha, 1990). Still, local regression methods have a strong limitation when it comes to getting a better insight of the structure of the domain being studied (Breiman et al., 1984). The resulting regression models have low interpretability.

3. Kernel Regression Trees

Our proposed method integrates regression trees with kernel regression. The approach is implemented and the system is referred to as HTL. It learns kernel regression trees from given training samples. The goal of this hybrid approach is to try to keep the efficiency and interpretability of regression trees while improving their accuracy by increasing the non-linearity of the functions used in the leaves. This goal is achieved by applying kernel regression in the tree leaves instead of simple averages (or other linear functions). From the point of view of kernel regression our proposal corresponds to the addition of an initial step that focuses the method on some sub-space of the original input space. In this initial stage we divide the input space into a set of mutually exclusive hyper-rectangles orthogonal to the input axis. Given a query point we start by situating it within one of these hyper-rectangles. Prediction is made using only the training samples that lie within this subregion.

3.1 The learning phase

The learning stage of our method consists of developing a regression tree. The algorithm we use is similar to the one described by Breiman et al. (1984) employed in system CART. We grow a binary tree with the goal of minimizing the mean squared error (MSE) obtained by the average y value. This corresponds to dividing the input space into regions with similar y values.

Splits on continuous variables are done by choosing the cutpoint value that maximizes the gain of MSE. Splits on discrete variables consist in finding the subset of values that maximizes the referred gain. To avoid the computational cost of finding all possible subsets we use the method proposed by Breiman et al. (1984) (Theorem 4.5, p.101) that reduces the complexity of this task from *O*(*NV*) to *O(*log *NV)*, where *NV* is the number of values of the discrete variable.

With respect to the stopping criterion for tree growth we use two thresholds. The first is a minimum number of examples in a node. The second is a minimum value on a statistic called *Coefficient of Variation* (Chatfield, 1983) that captures the relative spread of a set of values.

Our system HTL also provides post-pruning facilities by means of a pruning algorithm that is similar to the Nibblet & Bratko's (1986) method.

3.2 Making predictions

It is on prediction tasks that the hybrid nature of kernel regression trees becomes evident. Given a query point **q**, we run it through the tree until a leaf node is reached. At this stage instead of using an average *y* value, we perform a kernel based prediction. We obtain the prediction by equation 2 using the training samples of the leaf.

With respect to the distance function we use a diagonally weighted Euclidean distance (Atkeson et al., 1996) defined as

$$D(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_v FW_v \times \delta(\mathbf{x_{1,v}}, \mathbf{x_{2,v}})^2} \qquad \textbf{(3)}$$

where v is a variable (feature)
$\mathbf{x}_i$ is an instance
$\mathbf{x}_{i,j}$ is the value of variable j on instance i
FW_i is the weight of variable i
and

$$\delta(v_1, v_2) = \begin{cases} 0 & \text{if } v_1 = v_2 \text{ , for discrete variables} \\ 1 & \text{if } v_1 \neq v_2 \text{ , for discrete variables} \\ 0 & \text{if } |v_1 - v_2| < T_{eq} \text{ for continuous variables} \\ 1 & \text{if } |v_1 - v_2| \geq T_{diff} \text{ for continuous variables} \\ (|v_1 - v_2| - T_{eq}) \,/\, (T_{diff} - T_{eq}) & \text{if } T_{eq} < |v_1 - v_2| \leq T_{diff} \text{ for continuous variables} \end{cases}$$

In order to avoid overweighing of discrete variables with respect to continuous variables we use a ramping function (Hong, 1994) for continuous variables. This ramping function uses two thresholds. T_{eq} represents a limit for a difference of values to be considered irrelevant. On the other hand, T_{diff} states a limit for maximal difference relevance. Notice that if we set T_{eq} to zero and T_{diff} to the range of the variable we get the more common setup of the normalized difference of values. HTL

is able to use this latter approach or to estimate the values of the thresholds for each variable from the training data.

With respect to feature weights different approaches exist (see Wettscherek et al. (1996) for an excellent review in the context of K-nearest neighbors). Recently, Robnik-Sikonja & Kononenko (1996) presented a new method based on a previous work on *Relief* (Kira & Rendell, 1992; Kononenko, 1994) for estimating variable relevance in the context of regression. They describe an algorithm called *RReliefF* and successfully applied it in the context of regression trees. HTL uses this same method for estimating the weights of the variables that are used in distance calculations.

HTL uses as kernel function a gaussian kernel[6] given by

$$K(d) = e^{-d^2} \qquad \textbf{(4)}$$

Atkeson et al. (1996) refer that the choice of the kernel function is not a critical design issue as long as the function is reasonably smooth. These authors provide an extensive list of alternative kernel functions and discuss some of their merits.

Finally, HTL provides the following alternative ways of specifying the bandwidth used in predictions

- Set it as the distance of the K-nearest neighbor of the query point.
- Use all instances (infinite bandwidth).
- Specify a percentage of the instances as the bandwidth.
- Find automatically the optimal bandwidth setting by a process of Leave One Out Cross Validation (LOOCV).

Each of these four alternatives can be applied globally or locally. In the global setting all of the given training instances are used for estimation of the bandwidth. The local approach estimates the bandwidth for each of the tree leaves by using only the training instances of that leaf.

3.3 Some comments on the approach

The kernel regression formalization given in equation 2 provides a powerful theoretical framework that encapsulates several approaches to regression. Just by choosing different setups on some of its design issues, HTL is able to emulate other regression paradigms without any supplementary implementation cost. We have seen that a kernel regressor predicts the *y* value of a query point using a set of instances. If this set consists of all training data or just a sub-sample contained in a tree leaf is irrelevant from the point of view of the kernel model. This means that HTL is able to behave like a "normal" kernel regression system just by not growing a tree at all. On the other hand if we do learn a tree but set the bandwidth to "using all instances locally" and the kernel function to a uniform kernel ($K(\mathrm{d}) = 1, \ \forall\, \mathrm{d}$), we get a standard regression tree (like CART for instance). Many other setups are possible

---

6 Although the user may choose other functions through a parameter.

just by changing a small number of parameters of the framework (like a K-nearest neighbor algorithm for regression). This turns HTL into an highly flexible and powerful regression analysis tool which is an important issue when one wants to explore different perspectives of the same data.

The worst-case time complexity of tree growth using only discrete variables is $O(V^2 \times N)$ (Utgoff, 1989), where V is the number of variables and N is the number of training instances. However, this can reach $O(V^2 \times N^2)$ or even worse when continuous variables are involved due to sorting operations (Cattlet, 1991). Obtaining prediction using kernel regression is $O(N \times M \times V)$, where M is the number of testing instances. Learning a kernel regression tree is as complex as learning a regression tree. Prediction is a two stage process. First the case is dropped down the tree which is a $O(V)$ operation. Secondly, kernel regression is done using the training instances of the leaf. This operation is $O(N_L \times V)$ where N_L is the number of training instances in the leaf. This number depends on the depth of the tree. This means that the simpler the tree the higher the price in terms of time complexity for prediction tasks. Resuming prediction using kernel regression trees is $O(N_L \times M \times V)$. On average N_L will be much smaller than N which makes prediction much faster in HTL than in kernel regression. If we take both training and testing times together the analysis depends on the complexity of the learning stage[7] compared to the testing task. In the case of highly complex domains and few testing instances it is clear that HTL will suffer from the burden of having to learn a tree. However, even in such complex domains the advantages of kernel regression trees prediction schema will arise as soon as we get more testing instances.

Time complexity is just one of the facets of computational efficiency. Another important issue is storage requirements. HTL (like kernel regression methods) saves all training instances distributed by the tree leaves. As we have referred in section 2.2, several techniques exist to overcome this problem (Aha, 1990). These techniques could easily be applied within HTL which would reduce its storage requirements. An obvious thing to do is to take advantage of the instances generalization provided by the tree. Similar approaches were followed in EACH (Salzsberg, 1991) and RISE (Domingos, 1996) systems. We intend to explore this issue in the near future.

4. Experiments

HTL efficiently divides the input space into a set of mutually exclusive regions and applies kernel regression within these regions. The design of HTL took into account three main vectors : comprehensibility, predictive accuracy and computational efficiency. Both regression trees and kernel regression have different biases in these dimensions. In order to understand the benefits of kernel regression trees we have carried out several experiments in both artificial and real domains. The goal of the experiments was two fold. First we wanted to compare a kernel regression

---

[7] Which strongly depends on the average number of values per variable.

tree with a standard regression tree using average y values in the leaves. Secondly, we pretended to observe the effect of applying kernel regression on sub-sets of the training sample. For these purposes we took advantage of HTL's flexibility to generate three different types of regression models : a standard regression tree (RT); a kernel regression tree (KRT); and a kernel regression model (KR) using all instances of the training set. To ensure fair comparisons the regression trees on both RT and KRT are exactly the same. Similarly, the kernel regressors used by KRT and KR use exactly the same setups (we use a 3 nearest neighbor bandwidth).

The artificial dataset we have used was introduced by Breiman et al (1984). It consists of 10 predictor variables X_1, X_2, ..., X_{10} generated by

$P(X_1 = -1) = P(X_1 = 1) = 1/2$

$P(X_m = -1) = P(X_m = 0) = P(X_m = 1) = 1/3$, for $m = 2, ..., 10$

and the goal variable value being given by

$Y = 3 + 3X_2 + 2X_3 + X_4 + Z$ if $X_1 = 1$

or by $Y = -3 + 3X_5 + 2X_6 + X_7 + Z$ if $X_1 = -1$

where Z is a normally distributed variable with mean 0 and variance 2.

We have generated a training set consisting of 500 samples and a testing set of 5000 samples. For the tree-based approaches we have tried several "levels of comprehensibility" by growing trees of different sizes. The results of the comparisons along the three referred dimensions are shown in the floowing figure:

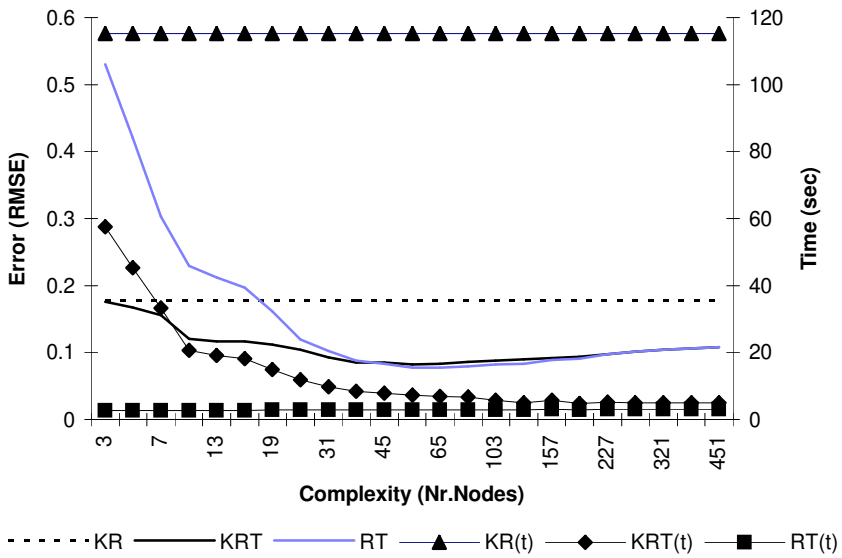

Figure 1 - Comparisons on an artificial dataset.

From the point of view of execution time (the (t) lines) we can easily observe the advantages of performing kernel regression on sub sets instead of all training data. This becomes more evident as the degree of partitioning increases (although this obviously decreases interpretability). The times shown on this figure correspond to a

training+testing cycle[8]. As the tree grows in size the time complexity of KRT reaches the minimum obtained by RT because the tree leaves have very few instances.

With respect to the error we observe that the hybrid nature of KRT makes the system obtain quite good results with very small trees. The typical U-shaped Error/Complexity curve is very flat for HTL which enables the system to obtain higher simplicity with almost no error cost. This tradeoff is considered very important in knowledge engineering tasks as it was referred by Bohanec & Bratko (1994). In these experiments the accuracy advantages of KRT over regression trees are not obvious. In our opinion this dataset is clearly biased toward tree-based approaches which makes the gains of using kernel regression in the leaves less relevant (which is confirmed by the the poor results of KR).

We have also carried out the same kind of experiments on four real world domains obtained from the UCI Machine Learning repository[9] :

Data Set	N. Instances	N. Variables
housing	506	13 (13C)
servo	167	4 (4D)
auto-mpg	398	7 (4C+3D)
machine-cpu	209	6 (6D)

Table 1 - The datasets (C-continuous variables; D-discrete variables).

As there are no test sets available for these datasets we have used 10-fold Cross Validation (CV) to estimate error and running times. All methods were trained and tested on the same data to ensure statistical significance of the results. Figure 2 shows the 10-fold CV average values for these experiments. The axis of the graphs have the same meaning as on the previous figure.

With respect to the analysis of prediction error we observe that kernel regression trees obtain always the best score of the three methods. Moreover, it obtains significant gains in terms of comprehensibility. With very simple trees KRT is able to achieve very good accuracy. These trees provide a very general description of the main characteristics of the problems. With respect to KR the accuracy gains are only statistically significant (at a 95% level) on the *Servo* dataset. However, one must recall that KRT produces a comprehensible model of the data while KR does not.

From the time complexity perspective the scenario is similar to the observed in the artificial problem with the exception of the Housing domain. The justification lies in the complexity of this learning task. In effect, KR is being evaluated by testing it on 1/10 of 506 instances using the remaining 9/10 as training samples. As the time complexity of KR is $O(N \times M \times V)$, the time taken by KR on the Housing domain is proportional to $450 \times 50 \times 13$. On the contrary, in the artificial problem we had something like $500 \times 5000 \times 10$, which is of a completely different dimension. The learning complexity of this domain compared to the simplicity of the testing task is

[8] Obviously, the KR method as no training time but that is one of its advantages.

[9] WWW: http://www.ics.uci.edu/~mlearn/MLRepository.html

penalizing tree-based methods. However, the increase of testing cases should inevitably show the advantages of partitioning the training set as it is done with kernel regression trees. On all other domains the prevailing task is testing due to the learning simplicity of the domains. This leads to a similar pattern to the one observed in the artificial dataset.

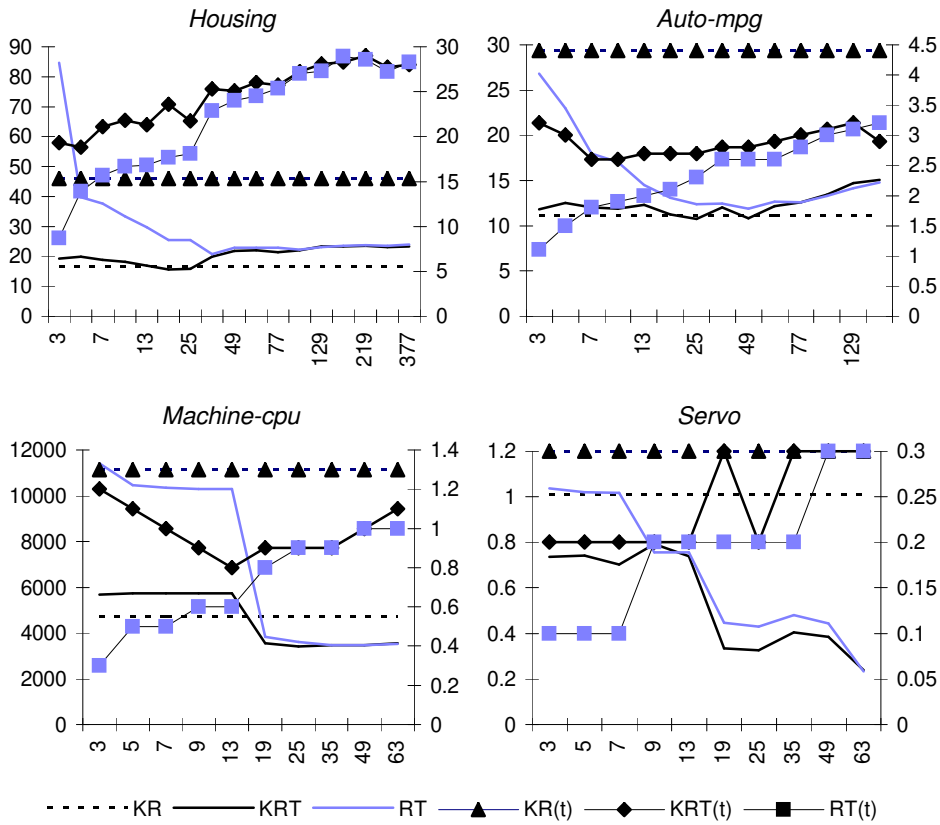

Figure 2 - 10-fold Cross Validation comparisons on real world domains.

Summarizing, kernel regression trees significantly outperform regression trees in terms of accuracy and size of the trees at the cost of some computational complexity. With respect to kernel regression the accuracy gains are not so significant but there are clear advantages in terms of comprehensibility and computation time.

5. Relations to other work

Smyth et al. (1995) describe a system that follows a similar integration schema but in the classification setup. They integrate class probability trees with kernel density estimators to achieve non-parametric estimates of class probabilities. The authors report significant improvements over decision trees and kernel density alone for certain class of problems. Their approach deals with continuous predictor

variables and discrete target variable as opposed to ours. Other differences include the feature weighing method. The authors use a feature selection approach. In the prediction of a leaf only the attributes present on the path from the root node are used (with constant weight). Breiman et al. (1984) referred that the fact that a variable does not appear in a tree does not mean that it is not important. A relevant variable may be constantly shadowed by other variables and thus not chosen for the tree nodes. Weiss & Indurkhya (1995) referred the same problem within rule-based systems. Finally, Smyth et al. (1995) provide a more sophisticated method of estimating the bandwidth by allowing different bandwidths for each variable.

Deng & Moore (1995) also describe a similar approach to regression. Their *multires* system integrates *k*d-trees with kernel regression producing what they also 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 decision trees. However, while regression trees are built with the goal of grouping data points with similar y values, *k*d-trees try to optimize the storage of these data points in order to achieve faster access time. The consequence is that the spliting criterion of the two approaches is completely different. Moreover, while HTL obtains the prediction using one leaf of the kernel regression tree, *multires* obtains the prediction by a combination of contribuitions of several nodes (not necessarily leaves). The integration in HTL is done with interpretability as one of the main goals, while in *multires* the main issue is obtaining an efficient way of structuring all the training set to make kernel regression computationally more efficient.

The work of Weiss & Indurkhya (1995) integrates a rule-based partitioning method with k-nearest neighbors. However, these authors deal with regression by mapping it into a classification problem. The original y values are transformed into a set of I intervals. One problem of this approach is that the number I needs to estimated which can be computationally demanding (Torgo & Gama, 1996). The local modeling used in this system is not elaborated as ours due to the absence of kernel functions, feature weights and flexible bandwidths.

M5 system (Quinlan, 1993) also uses regression trees and k-nearest neighbors. However, this system performs prediction combination instead of integrating the two methodologies like in HTL. This means that two models are independently obtained and their predictions combined. This has strong implications in terms of comprehensibility as mentioned before.

Several approaches to the integration of partition-based methods with instance-based learners exist for classification. Both EACH (Salzberg, 1991) and RISE (Domingos, 1996) try to generalize instances producing exemplars that can be seen as axis-parallel partitions of the input space. These systems proceed in a bottom-up way as opposed to the more efficient top-down recursive partitioning algorithm used in trees. Moreover, these systems address the different problem of classification.

Our work can also be seen as another method of obtaining a two-tiered concept representation (Michalski, 1990). The regression tree produced by HTL acts as a general *base concept representation* (BCR), while the tree together with the kernel regressors provide an *inferential concept interpretation* (ICI), according to

Michalski's terminology. Recently, Kubat (1996) presented a second tier for decision trees by adopting numeric closeness measures to tree branches to determine the query point class. This work was limited to continuous variables and dealt with classification problems. In this context our work could be seen as providing a second tier for regression trees.

6. Future Improvements

In the near future we intend to compare HTL with other inductive regression methods. The main difficulty of this task lies on the availability of some systems to ensure statistical significance of the comparisons.

By exploring the similarities between regression and classification trees, and between kernel density estimation and kernel regression, we intend to turn HTL into a general inductive system able to deal with both regression and classification.

A key problem we intend to address are the storage requirements of HTL. We will explore techniques for generating prototypes (or exemplars). This would significantly decrease the computational demands of HTL, but would also most probably carry some accuracy loss in some domains (Wettscherek, 1994). MDL coding schemes could bring some guidance for this tradeoff between compactness of the representation and accuracy.

Kernel regression is just one of the methods for performing local modeling. This method amounts to locally fitting a zero order polynomial (Cleveland & Loader, 1995). Recent work on local modeling considers high order polynomials as a means to overcome some problems with kernel approximations (Hastie & Loader, 1993). We intend to explore the idea of using local linear regression in the tree leaves instead of kernel regression. This would be similar to changing from average values (CART) to linear models (RETIS and M5), but within local approximation setups.

7. Conclusions

We have presented a novel propositional learning method that deals with regression. Our method integrates regression trees with kernel-based models. This integration is done by adding kernel regressors at the tree leaves generating what we have called a kernel regression tree. The work was motivated by trying to overcome the limitations of the simple models used in regression trees leaves, and at the same time, as a means to improve kernel regression comprehensibility and efficiency.

We have experimentally evaluated our method on artificial and real world domains. These experiments confirmed our expectations regarding the advantages of our hybrid method. Kernel regression trees significantly outperformed the accuracy of regression trees. Our method provides a better tradeoff between accuracy and tree size by achieving excellent prediction error with very simple trees. This allows the user to have a very simple and comprehensible general description of the domain without sacrificing accuracy. Compared to kernel regression, our method achieves a not so significant (but consistent) gain in accuracy, but provides a comprehensible model of the data. Applying kernel methods on sub sets of the training set has also proved to be beneficial in terms of execution time.

References

Aha, D. (1990) : A study of instance-based learning algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations. PhD Thesis. Technical Report 90-42. University of California at Irvine, Department of Information and Computer Science.

Aha, D. (1992) : Generalizing from case studies : A case study. In Proceedings of the 9th International Conference on Machine Learning. Sleeman,D. & Edwards,P. (eds.). Morgan Kaufmann.

Atkeson,C.G., Moore,A.W., Schaal,S. (in press) : Locally Weighted Learning. Special issue on lazy learning. Aha, D. (Ed.). *Artificial Intelligence Review*.

Bentley,J.L. (1975) : Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 509-517.

Bohanec, M., Bratko.I. (1994) : Trading Accuracy for Simplicity in Decision Trees. In *Machine Learning*, **15**-3, 223-250. Kluwer Academic Publishers.

Brazdil,P. , Gama,J., Henery,B. (1994) : Characterizing the applicability of classification algorithms using meta-level learning. In Proceedings of the European Conference on Machine Learning (ECML-94). Bergadano,F. & De Raedt,L. (eds.). Lecture Notes in Artificial Intelligence, 784. Springer-Verlag.

Brazdil,P. Torgo,L. (1990) : Knowledge Acquisition via Knowledge Integration. In *Current Trends in Knowledge Acquisition*. Wielinga,B. et al. (eds.). IOS Press.

Breiman,L. (1996) : Bagging Predictors. In *Machine Learning*, **24**, (p.123-140). Kluwer Academic Publishers.

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

Broadley, C. E. (1995) : Recursive automatic bias selection for classifier construction. *Machine Learning*, **20**, 63-94. Kluwer Academic Publishers.

Cattlet,J. (1991) : Megainduction : a test flight. In Proceedings of the 8th International Conference on Machine Learning. Birnbaum,L. & Collins,G. (eds.). Morgan Kaufmann.

Charfield, C. (1983) : *Statistics for technology* (third edition). Chapman and Hall, Ltd.

Cleveland, W.S. (1979) : Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, **74**, 829-836.

Cleveland,W.S., Loader,C.R. (1995) : Smoothing by Local Regression: Principles and Methods (with discussion). In *Computational Statistics*.

Deng,K., Moore,A.W. (1995) : Multiresolution Instance-based Learning. In Proceedings of IJCAI'95.

Domingos,P. (1996) : Unifying Instance-based and Rule-based Induction. In *Machine Learning*, **24**-2, 141-168. Kluwer Academic Publishers.

Fan,J. (1995) : Local Modelling. EES update: written for the Encyclopidea of Statistics Science.

Fix, E., Hodges, J.L. (1951) : Discriminatory analysis, nonparametric discrimination consistency properties. Technical Report 4, Randolph Filed, TX: US Air Force, School of Aviation Medicine.

Freund,Y., Schapire,R.E. (1995) : A decision-theoretic generalization of on-line learning and an application to boosting. Technical Report . AT & T Bell Laboratories.

Friedman, J. (1991) : Multivariate Adaptative Regression Splines. In *Annals of Statistics*, 19:1.

Friedman,J.H., Stuetzle,W. (1981) : Projection pursuit regression. *Journal of the American Statistical Association*, **76** (376), 817-823.

Hastie,T., Loader,C. (1993) : Local Regression: Automatic Kernel Carpentry. In *Statistical Science,* **8**, (p.120-143).

Hong,S.J. (1994) : Use of contextual information for feature ranking and discretization. Technical Report RC19664, IBM. To appear in IEEE Trans. on Knowledge and Data Engineering.

Karalic,A. (1992) : Employing Linear Regression in Regression Tree Leaves. In Proceedings of ECAI-92. Wiley & Sons.

Kibler,D., Aha,D., Albert,M. (1989) : Instance-based prediction of real-valued attributes. In *Computational Intelligence*, **5**.

Kira,K., Rendell,L.A. (1992) : The feature selection problem : traditional methods and new algorithm. In Proceedings of AAAI'92.

Kononenko,I. (1994) : Estimating attributes : analysis and extensions of Relief. In Proceedings of the European Conference on Machine Learning (ECML-94). Bergadano,F. & De Raedt,L. (eds.). Lecture Notes in Artificial Intelligence, 784. Springer-Verlag.

Kubat,M. (1996) : Second tier for decision trees. In Proceedings of the 13th International Conference on Machine Learning. Saitta,L. (ed.). Morgan Kaufmann.

Loader,C. (1995) : Old Faithful Erupts: Bandwidth Selection Reviewed. Technical Report 95.9. AT&T Bell Laboratories, Statistics Department.

Michalski,R. (1990) : Learning Flexible Concepts : Fundamental ideas and a method based on two-tiered representation. In Machine Learning : an artificial intelligence approach, vol. 3. Kodratoff,Y. & Michalski,R. (eds.). Morgan Kaufmann.

Michie,D., Spiegelhalter;D.J. & Taylor,C.C. (1994): *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Series in Artificial Intelligence, 1994.

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

Niblett,T., Bratko,I. (1986) : Learning decision rules in noisy domain. *Expert Systems*, **86**. Cambridge University Press.

Quinlan, J. R. (1993) : *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers.

Quinlan, J.R. (1992): Learning with Continuos Classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*. Singapore: World Scientific, 1992.

Quinlan, J.R. (1992): Learning with Continuos Classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*. Singapore: World Scientific, 1992.

Quinlan,J.R. (1993) : Combining Instance-based and Model-based Learning. Proceedings of the 10th ICML. Morgan Kaufmann.

Robnik-Sikonja,M., Kononenko,I. (1996) : Context-sensitive attribute estimation in regression. Proceedings of the ICML-96 Workshop on Learning in Context-Sensitive Domains.

Salzberg,S. (1991) : A nearest hyperrectangle learning method. *Machine Learning*, **6**-3, 251-276. Kluwer Academic Publishers.

Smyth,P., Gray,A., Fayyad,U.M. (1995) : Retrofitting Decision Tree Classifiers using Kernel Density Estimation. Proceedings of the 12th International Conference Machine Learning. Prieditis,A., Russel,S. (Eds.). Morgan Kaufmann.

Stone, C.J. (1977) : Consistent nonparametric regression. *The Annals of Statistics*, **5**, 595-645.

Torgo, L. (1995) : Data Fitting with Rule-based Regression. In *Proceedings of the 2nd international workshop on Artificial Intelligence Techniques (AIT'95)*, Zizka,J. and Brazdil,P. (eds.). Brno, Czech Republic.

Torgo,L. Gama,J. (1996) : Regression by Classification. To appear in Proceedings of SBIA'96. Springer-Verlag.

Utgoff,P. (1989) : Incremental induction of decision trees. *Machine Learning*, **4**-2, 161-186. Kluwer Academic Publishers.

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

Weiss, S. and Indurkhya, N. (1993) : Rule-base Regression. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1072-1078.

Weiss, S. and Indurkhya, N. (1995) : Rule-based Machine Learning Methods for Functional Prediction. In Journal Of Artificial Intelligence Research (JAIR), volume 3, pp.383-403.

Wettschereck,D. (1994) : A study of distance-based machine learning algorithms. PhD thesis. Oregon State University.

Wettschereck,D., Aha,D.W., Mohri,T. (in press) : A review and empirical evaluation of feature weighting methods for a class of lazzy learning algorithms. Special issue on lazy learning. Aha, D. (Ed.). *Artificial Intelligence Review*.

Wolpert,D.H. (1992) : Stacked Generalization. In *Neural Networks*, **5**, (p.241-259).