LIACC
Machine Learning group

Internal Report no. 99-2

Efficient and Comprehensible Local Regression

By
Luís Torgo

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

Nov,1999

Efficient and Comprehensible Local Regression

Luís Torgo

LIACC-FEP, University of Porto
R. Campo Alegre, 823 – 4150 Porto – Portugal
ltorgo@ncc.up.pt
Web Page : http://www.ncc.up.pt/~ltorgo

Abstract. This paper describes an approach to multivariate regression that aims at improving the computational efficiency and comprehensibility of local regression techniques. Local regression modeling is a non-parametric approach to multivariate regression. It is known for its ability to approximate quite diverse regression surfaces with high accuracy. However, these methods are also known for being computationally demanding and for not providing any comprehensible model of the data. These two characteristics can be regarded as major drawbacks in the context of a typical data mining scenario. The method we describe tackles these problems by integrating local regression within a partition-based induction method. This method uses an efficient recursive algorithm to divide the input space in a set of mutually exclusive regions. Moreover, it obtains a comprehensible description of these regions. By applying local regression within each of the regions we are able to overcome the computation efficiency limitations of local models. This integration schema also allows using the description of the partitions as a comprehensible overview of the approximation provided by our models.

Keywords. Multivariate regression, local regression, lazy learning, regression trees.

1 Introduction

This paper describes a hybrid approach to multivariate regression problems. Multivariate regression is a well known data analysis problem that can be loosely defined as the study of the relationship between a target continuous variable and a set of other input variables based on a sample of cases. In many important regression domains we cannot assume any fixed functional form for the model describing this relationship. This type of problems demand for what is usually known as non-parametric approaches. Several techniques exist that can be regarded as such. Regression trees (*e.g.* Breiman *et al.*,1984), for instance, do not assume any particular form of the surface being approximated. These models consist of a tree-based hierarchy of tests on the input variables. Each path from the root node to a leaf of the tree represents a sub-area of the regression surface. The tests in these paths can be interpreted as propositional logic descriptions of these areas. Although the form of the logical tests imposes some restrictions on the type of local areas that can be represented, the number of local areas and their "size" is not pre-defined. Because of this regression trees can be seen as non-parametric methods and are able to represent a wide range of regression surfaces through a flexible composition of smaller surfaces. The main features of tree-based regression are its *wide range of applicability*, *low computation time* and its *comprehensibility* due to the use of a logic description of the regression surface.

Another example of a non-parametric approach to this data analysis task is local regression modeling (*e.g.* Cleveland and Loader, 1995). The basic idea behind local regression consists of delaying the task of obtaining a model till prediction time. Instead of fitting a single model to all given data these methods fit one model for each query case using only the *most similar* training cases. As a result of this modeling methodology these techniques do not produce any visible and comprehensible model of the given training data. Moreover, for each query point its "neighbors" have to be found, which is a time-consuming task for any reasonably large problem. Still, these models are able to easily adapt to any form of regression surface, which leads to high advantages in terms of their ability to approximate a wide range of functions. In this paper we address the drawbacks of local modeling by integrating these models with regression trees. The resulting models provide a comprehensible model of the given data and improve significantly the computation cost of local modeling.

In the next section we describe very briefly local regression modeling. In Section 3 we then introduce our new regression method. Section 4 presents an experimental evaluation of our proposal and discusses its advantages over standard local modeling. Finally, we present the conclusions of this work.

2 Local Regression Modeling

According to Cleveland and Loader (1995) local regression modeling traces back to the 19th century. These authors provide a historical survey of the work done since then. The modern work on local modeling starts in the 1950's with *kernel methods* introduced within the probability density estimation setting (Rosenblatt,1956; Parzen,1962) and within the regression setting (Nadaraya,1964; Watson,1964). *Local polynomial regression* is a generalization of this early work on kernel regression. In effect, kernel regression amounts to fitting a polynomial of degree zero (a constant) in a neighborhood. Summarizing, we can state the general goal of local regression as trying to fit a polynomial of degree p around a query point (or test case) $\mathbf{q}$ using the training data in its neighborhood. This includes the various available settings like kernel regression (p=0), local linear regression (p=1), *etc*[1]. Local regression is strongly related to the work on lazy learning (*e.g.* Aha 1997), within the machine learning community. This modeling technique has been applied with success to several domains (*e.g.* Atkeson *et al.*,1997; Moore *et al.*,1997).

In this paper we focus on one particular type of local modeling, namely kernel regression. Still, the method that will be described is applicable to other local modeling techniques without any change. Within kernel regression predictions for query cases are obtained by averaging over the target variable 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).

[1] A further generalisation of this set-up consists of using polynomial mixing (Cleveland & Loader,1995), where p can take non-integer values.

The size of the neighborhood used for obtaining the prediction for a query case is another key issue of kernel regression and local modeling 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 modeling (*e.g.* Cleveland & Loader, 1995). The simplest strategy consists of using a maximum distance to the query case **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. 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*. 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, among different alternatives we can mention 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[2], 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 **q** using the following expression,

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

where,

$d(.)$ is the distance function between two instances;
$K(.)$ is a kernel function;
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.* $SKs = \sum_{i=1}^{n} K\left(\frac{d(\mathbf{x}_i, \mathbf{q})}{h}\right)$.

2.1 Local Regression from a KDD Perspective

The classical definition of the knowledge discovery in databases (KDD) (Fayyad *et al.*, 1996) refers this process as striving to identify valid, novel, potentially useful, and ultimately *understandable* patterns in data. From the perspective of understandability the local regression framework described above is very poor. In effect, with the exception of low dimensional (*i.e.* few input variables) problems it is hard to visualize/understand the kind of regression surface begin approximated by these models. Moreover, there is no single understandable model of the data that one can show to the typical user of a KDD tool.
Another characteristic of a typical data mining problem is its high dimensionality, that is the large number of cases and/or variables. Local modeling has a very high computational complexity if applied as described above. In effect, the prediction for each query case demands a look-up over all training cases to search for the most similar instances. This process has a complexity of the order of $O(n \times v)$ for each test case, where n is the number of training cases, and v is the number of variables. This is clearly too much even for moderate size problems.
In spite of these drawbacks, local regression modeling is a very successful data analysis technique in terms of predictive accuracy. In the next section we present a new type of regression models that aim at improving local modeling in terms of the above mentioned drawbacks, while trying to retain its accuracy advantages.

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

3 Local Regression Trees

Regression trees are non-parametric models that have as main advantages a high computational efficiency and a good compromise between comprehensibility and predictive accuracy. A regression tree consists of a hierarchy of nodes. Each node of the tree contains a *logical test* on one of the predictor (input) variables, with the exception of the bottom nodes of the hierarchy. These latter are usually known as the leaves of the tree and contain the predictions of the model. Each test has the form *Variable Operator Value* (*e.g.* Temperature < 45.7), and has two possible outcomes, true or false. Any path from the top node to a leaf can be seen as a conjunction of logical tests on the predictor variables. These conjunctions are logical representations of "sub-areas" of the overall regression surface being approximated. For each of these local areas different values of the goal variable can be predicted. The set of sub-areas obtained by a regression tree are mutually exclusive, *i.e.* there is no intersection between them. As such each training case falls in one of these areas. Thus, we can look at a regression tree as a partitioning of the given training sample. For each of these partitions (sub-areas) a constant value of the target variable is predicted. This originates a highly non-smooth and discontinuous regression surface, which is one of the main criticisms of regression trees (Friedman, 1991). Because of this constant value assumption, the criterion used to find the best split test for each node is to choose the one that leads to a higher decrease in the variance of the two resulting sub-samples. This criterion will reduce the predictive error resulting from assuming a constant value (usually the average target variable value) in each sub-area. There is a very efficient recursive partitioning (RP) algorithm (*e.g.* Torgo, 1999) that produces a regression tree from a sample of data.

The trees grown with the RP algorithm are usually post-pruned to ensure a better compromise between comprehensibility and predictive accuracy. The goal of post-pruning a tree is to reduce the number of nodes of the tree. This will decrease the number of partitions and thus will originate a more comprehensible model because each partition is described by a logical assertion. In most domains there is a clear trade-off between the number of partitions (leaves) and the predictive accuracy of the tree. Pruning more nodes leads to more comprehensible models but it also increases the variability in the leaves of the tree, which may have serious consequences in terms of predictive accuracy.

A possible solution to the increase of variability on the leaves originated by pruning is to use more complex models in the leaves of the trees instead of the "naive" average (Torgo, 1997). This is the strategy followed in this paper. Our proposal consists of obtaining a "standard" regression tree with the main goal of being comprehensible to the user. However, when using this tree for prediction tasks we use a kernel model in each leaf instead of the average target variable value. Given a query case we drop it down the tree until a leaf is reached as in standard regression trees. At this stage, we obtain the prediction using a kernel model with the training cases in the leaf. The resulting modeling technique, that we call a *local regression tree*, is able to achieve excellent predictive accuracy, while still producing a comprehensible model of the data that can be regarded as a rough overview of the true regression surface approximated by the local regression tree.

From the perspective of local modeling, local regression trees provide a tree-based organization of the training cases that ensures high computational efficiency. In effect, for each query case instead of looking for the most similar training cases on all training sample, we only use the partition of the training data corresponding to the leaf that is reached by the query case. This ensures a large efficiency gain allowing the application of local regression trees on very large data sets. Moreover, the resulting tree structure can be taken as a comprehensible representation of the model obtained from the training data. Still, as we have mentioned, this representation can only be regarded as a rough overview of the approximated surface, because it does illustrate the smoothing effect provided by the kernel models in the leaves. In spite of that, the regression tree provides the user with an idea of the regression surface, which is not the case in standard local modeling.

In order to clarify the ideas behind local regression trees we describe a simple example of these models. Suppose we have a sample of data consisting of 100 training cases generated by the function $Y = \sin(X) + \sigma(0,1)$, where $\sigma(0,1)$ is a gaussian noise component. Figure 1 shows on the left size the approximation provided by a standard regression tree using this training sample. As it can be seen the tree divides the input space in 8 partitions and for each of these areas uses the average target value of the respective training cases as the predicted value of the function being approximated. On the right size of the same figure we show the approximation if we have used a kernel model for each leaf instead of the average (*i.e.* the approximation of a local regression tree). This figure also illustrates the fact that the comprehensible model given by the standard regression tree can only be regarded as a rough view of the true surface approximated by the local regression tree.

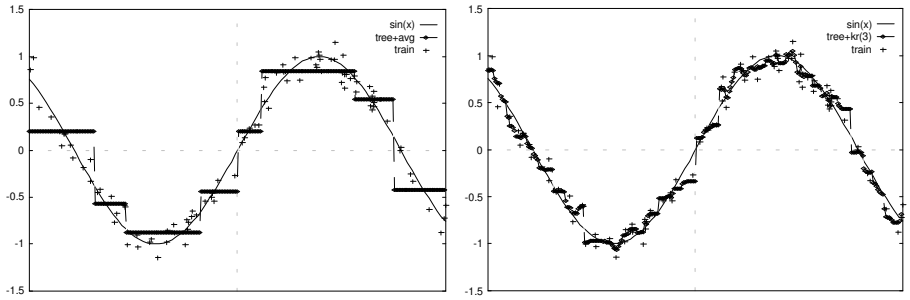

Fig. 1. "Standard" versus Local Regression Trees.

3.1 Relations to Other Work

One of the most successful efforts to improve the computational complexity of the task of finding the nearest neighbors of a query case are *k*d-trees (Friedman *et al.*, 1977). These trees were used by Deng & Moore (1995) in the context of kernel regression in their *multires* system. The form of a *k*d-tree strongly resembles a regression tree. However, the goals driving the construction of both models are completely different and as a result the trees are also different. Still, both provide a partitioning of a given sample of cases. The goal of growing a *k*d-tree is to insure a fast access to the nearest neighbors of any new query case. A *k*d-tree consists of a hierarchy of nodes, each containing a binary test on some input variable. Each test divides the sample of cases in a node in two sub-samples depending on the outcome of the test. The tests have the form of $X < v$, where X is a input variable and v a cut-point value that splits the cases in the current node in two sub-sets. The criterion to select the test for each node consists of selecting the variable where there is a widest spread of values (measured by the range). As cut-point value the usual choice is the median value. The growth of a *k*d-tree stops when the number of cases in a node drops below a pre-defined value (called the bucket size). The computational complexity of this growing procedure is comparable to that of growing a regression tree. The main difference in efficiency lies on the use of both trees. As we have seen, for local regression trees we only have to descend the tree until a leaf is reached and then use the training cases in this leaf to compute the kernel prediction. With respect to *k*d-trees the process is slightly more complex. After descending to a leaf we are not guaranteed to find there the nearest neighbors of the query case. As such it may be necessary to do some backtracking by climbing and descending through the *k*d-tree until we finally find all nearest neighbors to compute the kernel prediction. This means that the time taken to obtain a prediction with a *k*d-tree is larger than with a local regression tree. Still, we should remark that the two regression models are not equivalent. The use of *k*d-trees ensures exactly the same prediction as it would be obtained by a naive application of the kernel model, the difference being only the computational complexity taken by the two alternatives. On the contrary, local regression trees are a new local modeling technique and do not produce the same predictions as a kernel model. This means that although we can claim that local regression trees are more efficient than *k*d-trees, we should check the costs (if any) in terms of predictive accuracy of this improvement.

4 Experimental Evaluation

This section describes a series of experiments designed with the goal of comparing local regression trees with kernel regression modeling. The goal of these experiments is to compare the predictive accuracy of kernel models and local regression trees, and also to assert the computational efficiency gains of local regression trees. Computation times are measured by CPU seconds on a dual Pentium II 450 MHz machine running Linux.

4.1 Stock Market Indices : two case studies

The first experiments we present concern the problem of trying to predict the closing value of two well-known stock market indices (the Nikkei 225 and the Dow Jones industry average) based on a series of historical values of these and other major indices. The data was obtained from the Yahoo Finance site (http://quote.yahoo.com) and concerns the period since January of 1990 till September of 1999 amounting to 2399 observations. Each of the training cases was described by a series of 50 input

variables that included several types of information, like the previous values of the index being predicted and values of other indices.
Using these two problems we have compared the accuracy and running times of local regression trees and a kernel model. For local regression trees we have used exactly the same local modeling settings as for kernel regression, the single difference being that one is applied in the leaves of the trees while the other uses the information of all training set. The running times that were compared respect to the full train and test cycles[3]. The experimental methodology used was Cross Validation, namely a 10-fold cross validation (CV) experiment. The results that are shown are averages of 10 repetitions of 10-fold CV runs. The error of the models was measured by the mean squared error (MSE) between the predicted and truth values. Differences that can be considered statistically significant are marked by + signs (one sign means 95% confidence and two 99% confidence). The results of this experiment are shown in the following table:

Table 1. The results for the stock market indices experiments.

	Close Nikkei 225			***Close Dow Jones***		
	Local RT	*Kernel*		*Local RT*	*Kernel*	
MSE	140091.6±23508	**125951.1±20680**	++	**86.8±25.4**	214.5±40.4	++
CPU sec.	**4.4±0.5**	6.5±0.5	++	**2.47±0.3**	6.66±0.5	++

The results in terms of predictive accuracy are contradictory. In effect, both two methods achieve statistically significant (> 99% confidence) wins on different domains. However, local regression trees are able to significantly outperform kernel models in terms of computation efficiency, in spite of the small size of both the training and testing samples.

4.2 A Telecommunications Problem

Weiss and Indurkhya (1995) have used a telecommunications problem[4] to evaluate their *SWAP1r* system. Due to the commercial nature of the problem the authors do not provide any details of the data analysis task. The training set contains 5000 cases described by 48 continuous input variables, and the test set has a further 10000 cases.
Using a local regression tree we can complete the full train+test cycle in 160.3 CPU seconds, while a kernel regression model takes 1131.6 seconds to make predictions for the 10000 cases. Moreover, in this problem the local regression tree scores a mean squared error (MSE) of 58.5 with a tree consisting of 65 leaves, while the kernel model achieves a considerably larger error of 85.3 with no comprehensible model of the data.
We have run another experiment with this problem where we have joined the train and test sets in a single data file with 15000 cases and carried out a 10×10-fold CV experiment in order to get statistical significance results of the differences. The results of this experiment are shown in Table 2.

Table 2. Results with the telecommunications problem.

	Local RT	*Kernel*	
Error	**42.40±4.6**	57.19±6.6	++
CPU sec.	**63.57±3.1**	452.88±1.1	++

This other experiment again confirms the significance of the computational efficiency advantages of local regression trees over standard kernel modeling. Moreover, in this application these advantages are accompanied by significant gains in predictive accuracy.

4.3 Further Comparisons

In Torgo (1999) we have carried out similar experiments with a larger set of problems. For space reasons we only present here the summary of the results. These experiments were carried out using 12 different regression problems, and for each domain they were repeated for different training sample sizes (256, 512, 1024, 2048 and 4096). The following figure shows the computation time ratio between

[3] Obviously the training phase of a kernel model consists only of storing the cases.
[4] This data is available at http://www.cs.su.oz.au/~nitin/.

a kernel model and a local regression tree (values above 1 indicate advantages of local regression trees).

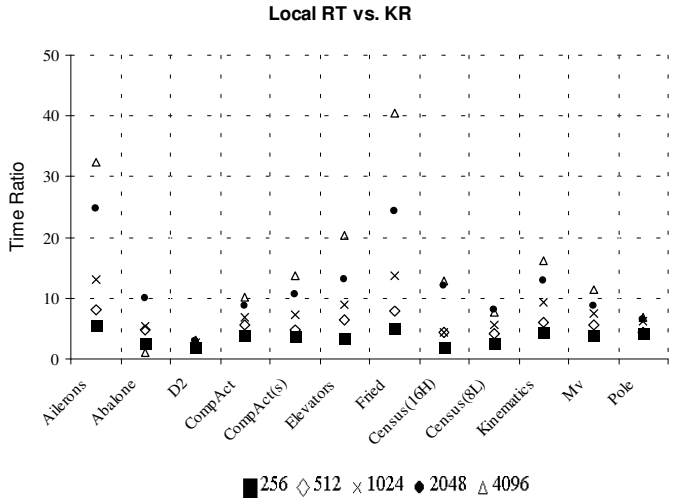

Fig. 2. Computation time ratios of kernel modeling versus local regression trees.

This larger set of experiments confirm an overwhelming computational efficiency advantage of local regression trees in spite of the small training set sizes used in these experiments.

4.4 A Simulation Study

We have carried out a simulation study using an artificial data set with 11 variables. We have generated increasingly larger training sets starting with 10000 cases until 150000 cases. For each trial we have obtained a local regression tree and a kernel model that were both tested on a separate set of 1000 cases. The computation time taken to complete the train and test cycle was recorded for both alternatives. The results of this experiment are shown in the following figure:

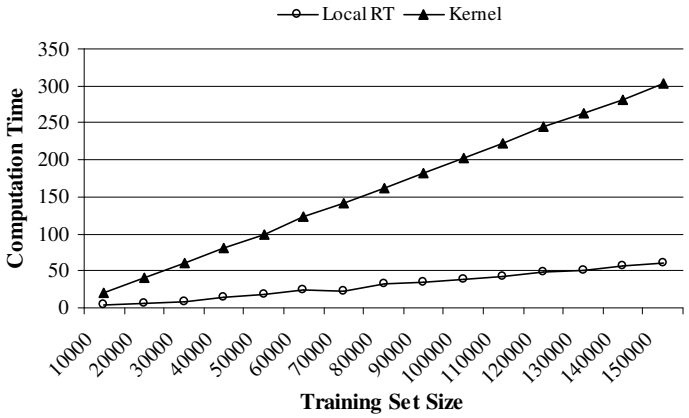

Fig. 3. A comparison of computation times using an artificial data set.

This experiment clear shows the scalability problems of standard kernel regression. On the contrary local regression trees have a much smoother growth of the computation time with the problem size. This experiment gives positive indications regarding the applicability of our proposed local regression trees to large problems that are often encountered in knowledge discovery in data bases.

4.5 Discussion

The experiments we have carried out clearly show that local regression trees are a very efficient and scalable local modeling method to handle multivariate regression problems. Moreover, these models provide some comprehensible insight of the regression surface, which can be crucial in some

applications. Finally, local regression trees are quite competitive with kernel modeling in terms of predictive accuracy. Moreover, as shown elsewhere (Torgo, 1999), local regression trees are quite competitive when compared to state of the art regression systems.

5 Conclusions

Local regression is a well-known data analysis method with excellent modeling abilities in a large range of problems. However, these techniques suffer from a high computational complexity and do not produce any visible and comprehensible model of the data. These can be considered major drawbacks in a typical data mining scenario.

In this paper we have described local regression trees that can be regarded as new type of regression models that integrate a partition-based technique with local modeling. Local regression trees provide the smoothing effects of local modeling within the efficiency and comprehensibility of partition-based methods. Through the use of kernel models in the leaves of a standard regression tree we are able to provide a focusing effect on the use of kernel models with large advantages in the computation necessary to obtain the predictions. At the same time, the partitioning obtained with the tree can be regarded as a comprehensible overview of the regression surface being used to obtain the predictions. We have carried out a large set of experiments that confirmed that local regression trees have an overwhelming advantage in terms of computation time with respect to standard local modeling techniques. Moreover, we have observed significant advantages in terms of predictive accuracy in several data sets.

The integration technique that was described can be easily extended to other local modeling methods as it is shown in Torgo (1999).

References

Aha, D. (1997) : *Lazy Learning*, edited by D. Aha. Kluwer Academic Publishers.

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

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

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

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

Cover,T., Hart,P. (1967) : Nearest Neighbor pattern classification. *IEEE Transactions on Information Theory*, **13**, 21-7.

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

Fan,J., Marron,J. (1993) : Comment on Hastie & Loader (1993). *Statistical Science,* **8**, 120-143.

Fayyad,U.,Shapiro,G.,Smyth,P.(1996):From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining*, Fayyad *et al.*(eds). AAAI Press.

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

Friedman, J., Bentley,J., Finkel,R. (1977) : An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, **3**-3,pp.209-226.

Friedman, J.(1991):Multivariate Adaptative Regression Splines. *Annals of Statistics*,**19**:1,1-141.

Moore,A. Schneider,J., Deng,K. (1997) : Efficient Locally Weighted Polynomial Regression Predictions. In *Proceedings of the 14th International Conference on Machine Learning (ICML-97)*, Fisher,D. (ed.). Morgan Kaufmann Publishers.

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

Parzen,E. (1962) : On estimation of a probability density function and mode. *Annals Mathematical Statistics*, **33**, 1065-1076.

Rosenblatt,M. (1956) : Remarks on some nonparametric estimates of a density function. *Annals Mathematical Statistics*, **27**, 832-837.

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

Torgo, L. (1999) : *Inductive Learning of Tree-based Regression Models*. Ph.D. Thesis. Dept. of Computer Science, Faculty of Sciences. University of Porto.

Torgo,L.(1997):Functional models for Regression Tree Leaves.In *Proc. International Confer. on Machine Learning (ICML-97)*, Fisher,D. (ed.). Morgan Kaufmann Publishers.

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

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