Using Turning Point Detection to Obtain Better Regression Trees

Paul K. Amalaman, Christoph F. Eick and Nouhad Rizk

pkamalam@uh.edu, ceick@uh.edu, nrizk@uh.edu

Department of Computer Science,
University of Houston, Houston, Texas 77204-3010, USA

Abstract. The issue of detecting optimal split points for linear regression trees is examined. A novel approach called Turning Point Regression Tree Induction (TPRTI) is proposed which uses turning points to identify the best split points. When this approach is used, first, a general trend is derived from the original dataset by dividing the dataset into subsets using a sliding window approach and a centroid for each subset is computed. Second, using those centroids, a set of turning points is identified, indicating points in the input space in which the regression function, associated with neighboring subsets, changes direction. Third, the turning points are then used as input to a novel linear regression tree induction algorithm as potential split points. TPRTI is compared in a set of experiments using artificial and real world data sets with state-of-the-art regression tree approaches, such as M5. The experimental results indicate that TPRTI has a high predictive accuracy and induces less complex trees than competing approaches, while still being scalable to cope with larger datasets.

Keywords: Prediction, Linear Regression Tree, Model Tree, Turning Point Detection

1 INTRODUCTION

Regression trees are widely used machine learning techniques for numerical prediction. Among the regression trees, we may distinguish those that associate a constant to each leaf node, for instance CART [2], and those that fit a less trivial model to each leaf node. Among the latter, we further distinguish a class of regression trees that fit a linear model to each leaf node, such as linear regression trees. Linear regression tree induction has been intensely researched. One of the first approach, M5[12], induces the tree using a CART-like splitting decision which is a binary split based on mean values and uses constant regression functions in the nodes; the attribute that best reduces the variance in the nodes is chosen for the split, and its mean value is selected as the split value. After the tree is fully developed M5 fits a linear model to each leaf-node during pruning phase. This splitting method is also used by many regression approaches which associate non-constant models with the leaf-nodes. However, in conjunction with non-constant regression models, using mean values of attributes as

split points and using variance reduction as an objective function does not necessarily obtain the best model [7].

To address this issue Karalic proposed RETIS [7] which fits a linear model in each node and uses minimization of the residual sum of squared errors (RSS) instead of the variance as splitting function. However, although the approach yields significantly better accuracy and smaller regression trees than M5 [12], it has been labeled "intractable" because to find the best pair {split attribute/split value} all potential split values for each input attribute need be evaluated, making this approach too expensive even for medium-sized datasets. Therefore, many more scalable algorithms for inducing linear regression trees have been proposed [1, 6, 7, 11, 12, 15, 5, 17] which rely on heuristics, sampling, approximations, and different node evaluation functions to reduce the computational cost of the RETIS algorithm.

Detecting turning points which indicate locations where the general trend changes direction can be useful in many applications. In this paper, we propose a new approach for Linear Regression Tree construction called Turning Point Regression Tree Induction (TPRTI) that infuses turning points into a regression tree induction algorithm to achieve improved scalability while maintaining accuracy and low model complexity. TPRTI induces decision trees using the following procedure. First, a general trend is derived from the dataset by dividing the dataset into a sequence of subsets of equal size using a sliding window, and by associating a centroid with each subset. Second, using those centroids, a set of turning points is identified, indicating points in the input space in which the piecewise linear function associated with neighboring subsets changes direction. Finally, the identified turning points are used in two novel top down linear regression tree induction algorithms as potential split points. The two algorithms which are called TPRTI-A and TPRTI-B are discussed in section 2.

Figure 1 illustrates our proposed approach to detecting turning points. The input dataset is shown in panel (A). In panel (B) the dataset is sorted by the input attribute and divided into subsets of equal size using a sliding window approach. In panel (C) the general trend in the dataset is derived by connecting the centroids of neighboring subsets, obtaining a piecewise linear function. Finally in panel (D), points m1 and m3 are selected as turning points as they exhibit sharp turns in the piecewise linear function. The selected turning points are later fed to a linear regression tree algorithm as potential split points. Algorithm 1 gives the pseudo code of turning point detection algorithm.

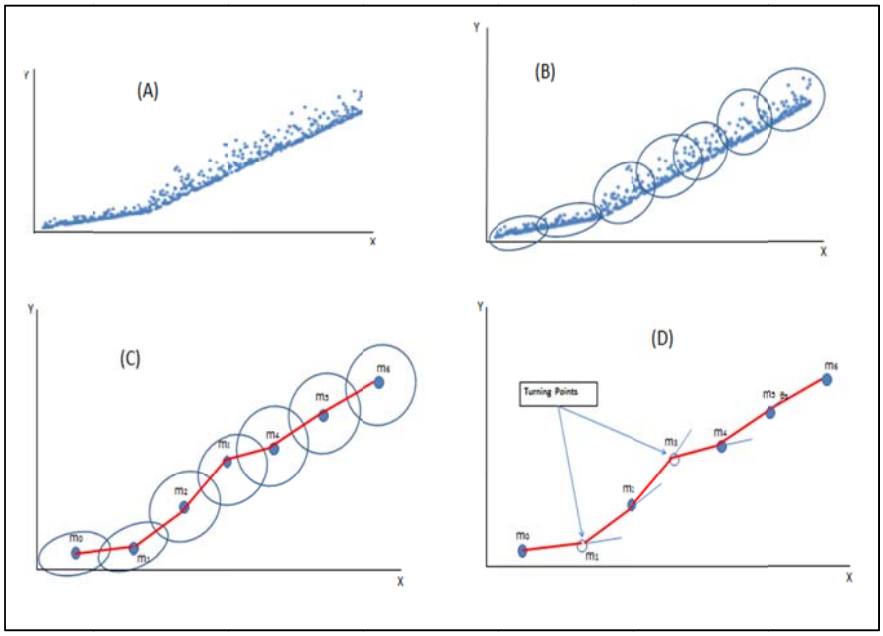

Fig. 1. Panel (A) represents hypothetical dataset in a plane (x,y). (B) The dataset is subdivided into overlapping subsets of equal size. (C) Centroids are joined by straight lines to form a general trend in the dataset. In panel (D) m_1 and m_3 are detected as turning points

Algorithm 1: Determining turning points

```
Inputs
  plane (Xk,Y) where Xk's are input attributes (k=1,2,..p)
  Xk: real valued discrete or continuous variable
  y: target variable
Outputs
  Turning points set

Project dataset onto each plane (Xk,Y)
For each plane (Xk,Y)
  Sort the data per attribute Xk
  if Xk discrete attribute then
    Compute centroids for each distinct value of Xk
    Label centroids as turning points
  else
    Use a sliding window of fixed size subsets for the input
    attribute to split the data into neighboring subsets from
    small values of Xk to the largest value of Xk
```

```
16     Determine general trends by computing the centroids of
       each subset and connect them to obtain piecewise linear
       functions
17     Identify turning points by analyzing the angle θ between
       neighboring subsets of the piecewise linear function
18  Output the set of turning points
```

The main contributions of the paper include:

1. A novel approach for turning point detection which relies on window sub-setting is introduced.
2. Two novel linear regression tree induction algorithms called TPRTI-A and TPRTI-B which incorporate turning points into the node evaluation are introduced.
3. State of the art linear regression tree algorithms are compared with each other and with TPRTI-A and TPRTI-B for a challenging benchmark involving 12 datasets.
4. The experimental results indicate that TPRTI is a scalable algorithm that is capable of obtaining a high predictive accuracy using smaller decision trees than other approaches.

The rest of the paper is organized as follows. Section 2 contains the description of our proposed methods for linear regression trees. In section 3 we show results of experimental study and we conclude in Section 4.

Table 1. Notation used in the remaining of the paper

K	User defined overlapping parameter characterizing the number of examples pertaining to two consecutive subsets.
S	Size of each subset
θ	Angle at a centroid
β	User-defined threshold angle such that if $\cos\theta < \cos\beta$ then the centroid with angle θ is a turning point
Stp_{XY}	Set of turning points in the XY-plane
Stp	Union of all Stp_{XY} (for all planes)
Stp_left	Turning Points set for left sub-node such that Stp=Stp_left U Stp_right
Stp_right	Turning Point set for right sub-node such that Stp=Stp_left U Stp_right
RSS	Residual Sum of Squared errors

2 THE TPRTI APPROACH

Linear regression algorithms can be divided into three groups: The first group fits a constant regression model to each intermediate node. M5 [12] is an example of this approach. The second group fits a more complex regression model to each node; usually at least a model is needed per input attribute. RETIS [7] is one such example since it fits multiple regression models per input attribute; one regression model for each distinct value of each input attribute. The third group uses linear regression

models in the leaves, but at each node transforms the regression problem into a classification problem in order to use more efficient node evaluation function. SECRET [5], GUIDE [9], and SUPPORT [3], are examples of such approaches. Each group can be distinguished by how well it performs with respect to model accuracy, model complexity, and runtime complexity, which is how scalable the approach is when data sizes increase. To evaluate a node, the first group is computationally more efficient since the evaluation function in each node is the simplest; however it often yields complex models and low accuracy. The second group has a much better accuracy, and model complexity but it comes at the expense of a higher cost node evaluation. Between both ends of the spectrum lies the third group. The major limitation of the first group of algorithms is illustrated next.

2.1 Illustrating the limitations of the variance based-approaches

Many widely used first group algorithms use variance as node evaluation function and we will refer to them as "variance-based approaches". M5 [12] is one such an example. As pointed out in [7] variance based algorithms have a fundamental imperfection to induce linear regression trees that are optimal because the variance is not a good node evaluation criterion. To illustrate this point let us consider a dataset called dataset#2 whose instances were generated with respect to the function defined below without using any noise, assuming a uniform distribution of input values x1 in [0,250]:

$$x1 \in [0,250] \text{ and } x2=0, \text{ and } y \in R$$

$$\begin{cases} y = x1; & if\ 0 \le x1 < 50 \\ y = 100 - x1; & if\ 50 \le x1 < 250 \end{cases}$$

It is clear that the best split value is located at x1=50. The variance based approaches, like M5, will split at x1=125 (the mean value of x1) leading to the necessity to introduce further possible split to the data in the left node. RETIS however, which tests each distinct value of the input variable x1, selects the optimal split value 50.

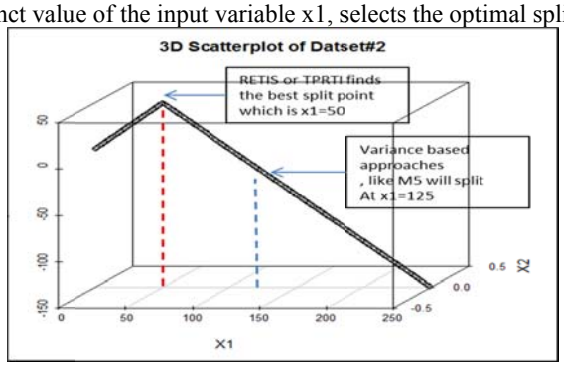

Fig. 2. Illustrating the imperfection of M5

As pointed out earlier, many second group approaches like RETIS yield more accurate, and shorter linear regression trees but do not scale well to large dataset. Likewise the third group approaches resolve the scalability issue of RETIS but at a cost to accuracy and/or model complexity.

In this paper we use the term "look-ahead RSS" to mean the following: First, the dataset associated with current node is split into two sub-sets; one pertaining to each sub-node. Second, each sub-node is fitted with a linear regression model and the Residual Sum of Squared errors (RSS) are computed for both nodes. Next, a weighted sum of both RSS is computed. This is done for all potential split pairs {attribute variable, attribute value} available. One such split is retained and the rest discarded. Likewise, we use the term "split point" to designate the pair {split attribute, split attribute value}. We also use it to refer to the actual point in the plane, or point in the input space. When used as a point in a plane it refers to the pair (x_q, y) where x_q is one of the input variables, $q \in \{1,..p\}$. When used to mean a point in the input space, it refers to a tuple; $(\mathbf{x},y)$ where $\mathbf{x}$ is an input vector $(x_1, ..x_p)$ and y the associate response. Key notations used in the remaining of the paper are provided in table1. Algorithm1 presents the turning point detection approach. We provide next, in section 2.2 detailed description of how turning points are computed.

2.2 Centroids and turning points computation

First, a general trend is derived from the dataset by sorting the examples based on the input attributes, by dividing the dataset into subsets of equal size using a sliding window, and by associating a centroid with each subset. Second, using those centroids, a set of turning points is identified, indicating points in the input space in which the piecewise linear function—which was obtained by connecting centroids of neighboring subsets—significantly changes direction.

The input attributes to the algorithm are real valued attributes that are either discrete or continuous. Lines 8, 9, and 10 in algorithm 1 project the dataset onto the p x_ky-planes (k=1,..,p). For the remaining of the lines, line 11 to 17 we consider one plane at a time. Line 10 ensures that the dataset associated with the plane is sorted with respect to the input attribute. Lines 11, 12, and 13 treat the case of discrete attributes. First, the algorithm queries the distinct values of the discrete attribute. It then computes the centroids for each attribute. Next, each centroid is labeled turning point.

Lines 14 to 17 treat the case where the input attribute is continuous. There are three user-defined parameters K, S, β that need to be set. Let assume that a centroid has angle θ. β is a user-defined angle such that if $\cos\theta < \cos\beta$ then the centroid is a turning point. K is the overlapping parameter characterizing the number of examples pertaining to two consecutive subsets. S is the size of each subset. In line 15 subsets of equal size S are created using the sorted dataset as follows: S_0 is composed of the first S examples. At any given step i, (i>0), the examples in subset S_i are determined by dropping the first K examples in S_{i-1} and by adding the next K new examples. When

K=S, the subsets are disjoint. When 0<K< S the subsets overlap. In line 17 turning points are computed for each plane by analyzing the angle at each centroid.

2.3 Node Evaluation

We introduce TPRTI-A, which is a mixture of first group and second group approach in that its node evaluation avoids exhaustive search by evaluating only a supplied list of turning points. We also introduce TPRTI-B which is a mixture of all 3 groups in that it uses a two-step node evaluation. It avoids exhaustive search by evaluating only a supplied set of turning points. It first fits a model to current node, and uses a simple evaluation function which is the distance of each turning point to the fitted model, to select the turning point which distance is the largest. TPRTI-A and TPRTI-B differ by their node evaluation function. They both use as input, a set of predetermined turning points.

2.3.1 Node evaluation for TPRTI-A

The first approach, TPRTI-A, evaluates all turning points by a look-ahead strategy and selects the one that yields the minimum RSS.

Algorithm 2: Node evaluation for TPRTI-A

```
1 Inputs
2   Stp_XY: Set of turning points in the XY-plane
3   Stp: Union of all Stp_XY (for all planes)
4   TP_XY(x,y): Turning point in the XY-plane with
    coordinate (x,y)
5  If stopping criteria is reached then
6    Return
7  For each Stp_XY in Stp
8   For each TP_XY(x,y) in Stp_XY
9       Split data in S_Left and S_right
10      Compute look-ahead weighted RSS
11 Select the turning point (x.,y.) that minimizes
   Weighted RSS for the split
12 Split Stp into Left_Stp, and Right_Stp based on x.
```

2.3.2 Node evaluation for TPRTI-B

The second approach, TPRTI-B, is a multi-step evaluation approach. With this approach, first the current node is fitted with a model and the distances of the turning points (actual tuples) to the fitted model are computed. The turning point for which the distance to the model is the largest is selected as split point. Next, each coordinate (value of input attributes) of the split point needs be evaluated by a look-ahead RSS minimization method to determine the best pair {split variable, split value}. That is, in contrast to TPRTI-A, only a single split value per input attribute and not a set of split values is considered; reducing runtime complexity. Figure 3 illustrates the general idea. In figure 3, the dotted line represents a linear model F fitted to current node. In

this hypothetical node, the turning point set has three turning points TP1, TP2 and TP3. We assume d1 < d2 and d3 < d2. Hence TP2(x_{21},x_{22},y_2) is chosen as split point. Its coordinates x_1= x_{21},and x_2= x_{22} need next, to be evaluated to select the pair {split variable, split value} that best minimizes RSS. Algorithm 3 summarizes the concept.

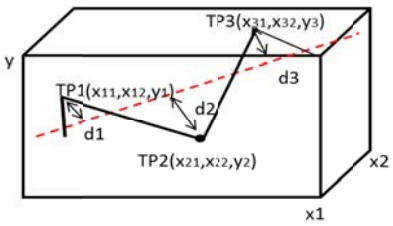

Fig. 3. To illustrate the node evaluation of TPRTI-B; the dotted line is a model fitted to current node. TP2 is selected as split point because it is the turning point further away from the fitted model F.

Algorithm 3: Node evaluation for TPRTI-B

```
1  If stopping criteria is reached then
2    Return
3  Fit current node with linear model F
4  For each turning point tp in Stp
5    Compute distance to F
6  Select tp_max the point with the largest distance to F
7  For each input coordinate of tp_max
8   Split data in S_Left and S_right
9   Compute look-ahead weighted RSS
10 Select the turning point that minimizes weighted RSS as split
    point
11  split Stp in Left_Stp, and Right_Stp
```

In line 3 a model F is fitted to the node. Lines 4, 5, and 6 select the point with the largest distance to F. Line 7 to 11 are similar to what was presented for TPRTI-A in 2.3.1 except that here the points are represented in the actual space; not in a plane. Hence the turning points set Stp, although computed as presented previously in algorithm 1, is formed of points with their coordinates in the data space. This is so because line 3 needs to compute the distance to F in the entire space.

2.4 Runtime Complexity

We compute the runtime cost to evaluate a node. Let N be the total number of training examples in the node, m the number of subsets, p the number of input attributes, and t total number of turning points. Assuming N>>p, evaluating each split candidate costs $O(p^2N)$; If TPRTI-A method is used, all the turning points are evaluated and the cost

to evaluate a node is $O(p^2Nt)$. If TPRTI-B is used the distance of each turning point in the data space to the fitted curve cost $O(p)$ which leads to $O(pt)$ for the t turning points. $O(p^2N)$ is needed to evaluate each of the p input attribute, and $O(p^2N)$ is as well needed to fit a model to current node; obtaining: $O(pt+p^2N+Np^3) = O(p^2N(p+1))$. With M5, the split point is the mean point of each p variable. Hence, t=p obtaining $O(p^3N)$; In the worst case, RETIS will test each value of each variable leading to t=pN ; thus $O(p^3N^2)$. TPRTI-A worse case happens when each centroid is a turning point; which leads to t=pm hence $O(p^3Nm)$. Table 2 summarizes the runtime complexity of each approach.

Table 2. Node Runtime complexity of TPRTI in comparison to M5 and RETIS

	RETIS	TPRTI-A	TPRTI-B	M5
Runtime complexity	$O(p^3N^2)$	$O(p^3Nm)$	$O(p^2N(p+1))$	$O(p^3N)$

2.5 Stopping criteria:

RETIS, M5 and TPRTI share the following stopping criteria

- The algorithm stops if the number of examples in the node is less than a minimum number set by user.
- The algorithm stops if the subsequent split sub-nodes do not improve the error function more than a minimum value set by user.
- However, TPRTI has an additional stopping condition: The algorithm may terminate if there is no more turning point in the node dataset to evaluate.

3 EMPIRICAL EVALUATION

In this section results of extensive experimental study of TPRTI-A and TPRTI-B are provided. We compare both algorithms with each other and against the well-known M5 [12], SECRET [5], GUIDE [9], and RETIS [7] algorithms in term of accuracy, scalability and model complexity. The experimental result published in [5], for GUIDE and SECRET, two state-of-the-art scalable linear regression tree induction algorithms are used in this study for comparison. The GUIDE and SECRET algorithms were built to be both fast and accurate. Model complexity results for previously used datasets were not available for comparison. We performed all the experiments reported in this paper on a 64-bit PC i7-2630 CPU at 2Ghz running Windows 7. Datasets and extended version of this paper are available at [19].

3.1 Datasets

Five artificial and 7 real-world datasets were used in the experiments; Table 3 and 4 give a summary for the 12 datasets. The last column in both tables contains in parenthesis the number of attributes.

Table 3. Artificial datasets.

Dataset	Description	Number of examples
dataset #1	x1, x2 are the input variables and y the output variable; x1 ∈ R ,x2 ∈ R, y∈ R $\begin{cases} Y = -2 * x1 & if\ 0 \le x1 < 100\ and\ x2 = 0 \\ Y = -700 + 5 * x1\ if & 100 \le x1 < 200\ and\ x2 = 0 \\ Y = 300 - 3 * x2 & if\ 0 \le x2 < 100\ and\ x1 = 200 \end{cases}$	300 (3)
dataset #2	x1 ∈ [0,250] and x2=0, and y ∈ R $\begin{cases} y = x1; & if\ 0 \le x1 < 50 \\ y = 100 - x1; & if\ 50 \le x1 < 250 \end{cases}$	2500 (3)
CART dataset	This dataset was found in [2]with 10 predictor attributes:X1 ϵ {−1, 1} with equal probability that X1 =1 or X1 = -1; Xi ϵ {−1, 0, 1}, with i ϵ {2 . . . 10} and the predicted attribute y determined by $\begin{cases} Y = 3 + 3X2 + 2X3 + X4 & if\ x1 = 1 \\ Y = -3 + 3X5 + 2X6 + X7 & if\ x = -1 \end{cases}$ A random noise ε between [-2 and 2] was added to Y	750 (11)
3DSin dataset	This dataset has two continuous predictor attributes x1, x2 uniformly distributed in interval [−3, 3] determined by Y = 3 sin(X1) sin(X2).	500(3)
Fried dataset	This dataset has 10 continuous predictor attributes with independent values uniformly distributed in the interval [0, 1] Y = 10 sin(πX1X2) + 20(X3−0.5)^2 + 10X4+5X5;A random noise ε between [-1;1] was added	700(11)

Table 4. Real world dataset.

Dataset	**Description**	**Number of examples (number of attributes)**
Abalone	This dataset was obtained from UCI [16] machine learning repository.	4177 (8)
Auto-mpg	This dataset obtained from UCI [16] repository. Tuples with missing data were removed.	392 (8)
Boston Housing	This dataset obtained from UCI[16] repository	506 (14)
Kin8nm	This dataset was obtained from the DELVE [4]repository.	8192 (9)
Normalized Auto-mpg	This is the auto-mpg dataset from UCI [16] repository that has been normalized with z-score values	392 (8)
STOCK	This dataset is from SatLib[14]. The dataset contains 950 examples. However, the first tuple was removed because it did not appear correctly formatted	949 (10)
Tecator Dataset	This dataset originated from the StatLib [14] repository.	240 (11)

3.2 Experimental Methodology

All the experiments were done with a 10-fold cross validation and repeated 10 times with different seeds for each run. The average values are reported in table 7 along with corresponding standard deviation. Five artificial datasets were used three of which were previously used and results for GUIDE and SECRET are available in [5]. We also used 7 real world datasets of which 6 were previously used and results for GUIDE and SECRET available in [5]. The datasets which have not been previously used are dataset#1, dataset#2 and the normalized auto mpg dataset. We implemented TPRTI making use of R software [13] libraries. We run M5 using Weka [18]. R and

Weka are publicly available software. Our implementation of RETIS relies on running TPRTI-A with all input attributes set as discrete attributes.

3.2.1 Input parameters and stopping rules used for the experiments.

Table 5. Input and stopping parameters for TPRTI

	TPRTI-A				**TPRTI-B**			
	Input parameters		**Stopping rules*		*Input parameters*		*Stopping rules*	
	Subset Size	cos β	Min. Node Size (in %)	Min. RSS imp. (in%)	Subset size	cos β	Min. Node Size (in %)	Min. RSS imp. (in %)
Dataset #1	3	0.8	10	10	3	0.8	10	10
Dataset #2	9	0.8	10	10	9	0.8	10	10
CART	5	0.8	10	10	5	0.8	10	10
3DSin	4	0.8	10	10	5	0.8	10	10
Fried	3	0.85	10	10	3	0.85	10	10
Abalone	55	0.8	10	10	21	0.8	10	10
Auto mpg	4	0.85	10	10	4	0.85	10	10
Boston Housing	14	0.8	12	12	14	85	12	12
Normalized auto mpg (z-score)	4	0.85	10	10	4	0.85	10	10
Stock Data	4	0.8	10	10	13	0.85	10	10
Tecator	8	0.8	10	10	21	0.7	10	10
Kin8nm	250	0.95	3	1	250	0.95	3	1

**Stopping rules:* The stopping parameters set for TPRTI-A are same parameters used for RETIS.

For each dataset, different input parameters and stopping rules were used for TPRTI-A, and TPRTI-B. Table 5 summarizes the parameters used for each dataset where cosβ is the cosine threshold used to determine the turning points, "Min. Node Size" is the minimum required size of a node expressed as 100*(current node size)/(initial dataset), and "Min. RSS imp." is the minimum improvement of the sum of the weighted RSS of both sub-nodes required to split a node. It is expressed as 100*(Parent RSS - weighted sum of children nodes RSS)/Parent RSS.

The input parameter "Subset Size" is used to subdivide the input data into subsets of equal size in order to compute the centroids. RETIS was run without post pruning.

3.3 Results

Accuracy was measured by the MSE (Mean Squared Error). Model Complexity was measured by the number of leaf-nodes. However, a bad model may have small number of leaf-nodes. Thus complexity was slightly redefined as number of times an approach obtained the combination (best accuracy, fewest leaf-nodes). Both the number of leaf-nodes and MSE are provided as $\mu \pm c$ where μ is the average MSE (or number of leaf-node) and c the standard deviation over several runs. Let $\mu_1 \pm c_1$ and $\mu_2 \pm c_2$ be two results such that $\mu_1 < \mu_2$. We consider a tie between the two results if $\mu_1 + c_1 > \mu_2$. Both Accuracy and number of leaf-nodes are reported in table 7 with the number of leaf-nodes in parenthesis. The main findings of our study are:

3.3.1 On Accuracy

Table 6. Comparison between TPRTI-A, TPRTI-B and state-of-the-art approaches with respect to accuracy

	M5	TPRTI-A	TPRTI-B	RETIS	GUIDE	SECRET
TPRTI-A	(6/5/1)	-	(4/6/2)	(4/6/0)	(5/1/2)	(4/2/2)
TPRTI-B	(4/6/2)	(2/6/4)	-	(3/7/0)	(5/1/2)	(1/4/3)

TPRTI-A, and TPRTI-B are compared with the approaches in the columns. The number in each cell denotes (number of wins/number of ties/number of losses). For example, (6/5/1) in the first column of the first row means that TPRTI-A is more accurate than M5 on 6 datasets, ties M5 on 5 datasets and loses on 1 dataset. Overall, TPRTI yields comparable result as or slightly better result than RETIS. It has better accuracy than GUIDE and SECRET.

Table 7. Accuracy results

	M5	**RETIS**	**GUIDE**	**SECRET**	**TPRTI-A**	**TPRTI-B**
Dataset #1	446.8996 ±36.45 (11±0.00)	**0.000 ±0.0000 (3±0.00)**	N.A	N.A	0.089 ±0.0000 **(3±0.00)**	0.089 ±0.0000 **(3±0.00)**
Dataset #2	4.75 ±0.239 (11±0.00)	**0.000 ±0.0000 (2±0.00)**	N.A	N.A	**0.000 ±0.0000 (2±0.00)**	**0.000 ±0.0000 (2±0.00)**
CART	0.0932 ±0.0009 **(2±0.00)**	0.085 ±0.0125 (4.1±0.32)	N.A	N.A	**0.07614 ±0.0100** (4.1±0.32)	0.098±0.33 (6.1±0.32)
3DSin	**0.0015 ±0.0002** (20±0.00)	0.01 ±0.0074 (4±0.00)	0.0448 ±0.0018	0.0384 ±0.0026	0.0074 ±0.01 (4±0.00)	0.0063 ±0.01 **(3±0.00)**
Fried	4.888 ±0.1536 **(3±0.00)**	4.773 ±0.3893 **(3±0.00)**	**1.21 ±0.0000**	1.26 ±0.010	3.1114 ±0.80 (4±0.00)	1.4968 ±0.60 (6.7±0.48)
Abalone	4.6910 ±0.586 **(2±0.00)**	*N.A	4.63 ±0.04	4.67 ±0.04	4.3806 ±2.71 (4±0.00)	**4.1527±2.59** (5.1±0.45)
Auto mpg	8.5073 ±0.3105 (5±0.00)	8.8470 ±7.2183 **(3.1±0.32)**	34.92 ±21.92	15.88 ±0.68	**7.6021 ±6.33** (5±0.00)	8.4493 ±6.39 (4.6±0.52)
Boston Housing	28.8397 ±30.8896 (10±0.00)	24.569±20.090 **(4.2 ±0.92)**	40.63 ±6.63	24.01 ±0.69	**16.0922±10.29** (5.5±0.53)	19.6237 ±9.24 (4.8±0.92)
Normalized Auto mpg (z-score)	0.1396 ±0.0051 (5±0.00)	0.1186±0.0895 (4.0 ±0.00)	N.A	N.A	**0.1169 ±0.07 (3.8±0.63)**	0.1342 ±0.09 (4.7±0.82)

Stock Data	1.0389 ±0.1008 (19±0.00)	11.977±7.884 (3.9 ±0.32)	1.49 ±0.09	1.35 ±0.05	**0.2067 ±0.10 (3±0.47)**	4.8867 ±3.09 (4.9±0.88)
Tecator	9.4513 ±2.9519 (6±0.00)	6.6310±6.3036 (5.4 ±0.51)	13.46 ±0.72	12.08 ±0.53	**2.8315 ±1.412 (3.1 ±0.31)**	7.1266 ±8.20 (6.4±0.70)
Kin8nm	0.0303 ±0.0009 (24±0.00)	*N.A.	0.0235 ±0.0002	**0.0222 ±0.0002**	0.0303 ±0.001 (5.33±0.57)	0.0227±0.0020 (25.5±0.17)

*N.A is used to express the fact that the program runs more than 3 hours without outputting a result or runs out of memory whereas N.A is used to express the fact that the result is not available.

3.3.2 On Model Complexity

In this study we consider a linear regression model to have a good model complexity when it is both accurate and has a small number of leaf-nodes. Table 8, which is compiled from table 7, presents the number of cases where an approach obtained both best accuracy and fewest nodes at the same time.

Table 8. Number of time an approach obtained the combination (best accuracy, fewest leaf nodes) for a dataset

M5	TPRTI-A	TPRTI-B	RETIS	GUIDE	SECRET
0	5	3	5	N.A	N.A

RETIS and TPRTI-A won the combination (best accuracy, fewest leaf-nodes) five times while M5 never won, and TPRTI-B won 3 times. This suggests that TPRTI hold comparable model complexity as RETIS.

3.3.3 On Scalability With Respect to Dataset Size

We use a direct comparison of runtime in seconds for TPRTI-A, TPRTI-B, and RETIS because they were implemented and run in the same machine. We use an indirect comparison to compare the different approaches. The indirect comparison consists of setting a baseline dataset size, and measuring the percent increase in runtime in relation to percent increase in baseline dataset size. Figure 4 summarizes our findings. TPRTI-B outperforms M5 consistently on all dataset sizes and number of input attribute. This suggests that TPRTI-B is a more scalable approach than M5. This is because models generated by TPRTI tend to have fewer nodes. On small to medium size dataset there is no significant difference between TPRTI and SECRET. Overall SECRET outperforms TPRTI consistently on all dataset size. Figure 5 summarizes our result for the direct comparison. In figure 5, Panel (A) shows that RETIS has the worst performance even on dataset with small number of input attribute. Panel (B) provides evidence that as the number of input attribute increases, performance decreases. Panel(C) and (D) demonstrate that TPRTI-B consistently outperform TPRTI-A.

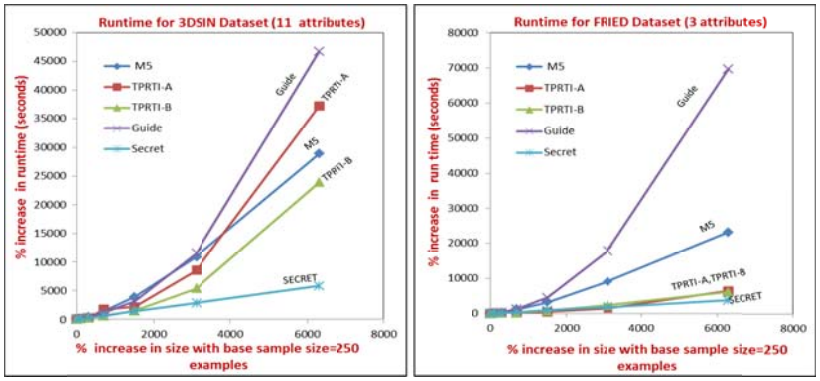

Fig. 4. Indirect Runtime comparison: Percent increase in baseline dataset size and the resulting percent increase in runtime with baseline size set to 250 examples

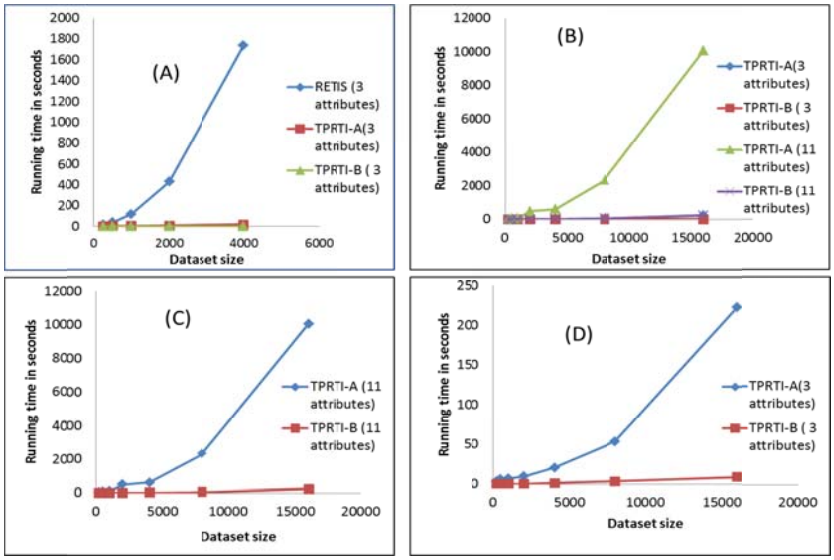

Fig. 5. Direct runtime comparison among TPRTI-A, TPRTI-B and RETIS

4 CONCLUSION

The paper proposes a new approach for Linear Regression Tree construction called Turning Point Regression Tree Induction (TPRTI) that infuses turning points into a regression tree induction algorithm to achieve improved scalability while maintaining accuracy and low model complexity. Two novel linear regression tree induction algorithms called TPRTI-A and TPRTI-B which incorporate turning points into the node

evaluation were introduced and experimental results indicate that TPRTI is a scalable algorithm that is capable of obtaining a high predictive accuracy using smaller decision trees than other approaches.

FUTURE WORK

We are investigating how turning point detection can also be used to induce better classification trees.

REFERENCES

1. W.P. Alexander and S.D. Grimshaw. Treed regression. Journal of Computational and Gra aphical Statistics, 5:156-175, 1996.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees.Wadsworth, Belmont, 1984.
3. P. Chaudhuri, M.-C. Huang, W.-Y. Loh, and R. Yao. Piecewise-polynomial regression trees. Statistica Sinica, 4:143–167, 1994.
4. DELVE repository of data http://www.cs.toronto.edu/~delve/ as of 12/04/2012
5. Alin Dobra, Johannes Gehrke SECRET:a scalable linear regression tree algorithm SIGKDD-2002.
6. J. Friedman. Multivariate adaptive regression splines (with discussion). Annals of Statistics, 19:1-142, 1991.
7. A. Karalic. Employing linear regression in regression tree leaves. In European Conference on Artificial Intelligence, pages 440-441, 1992.
8. Li, K.C., Lue, H.H., Chen, C.H. Interactive Tree-Structured Regression via Principal Hessian Directions. Journal of the American Statistical Association, vol. 95, pp. 547-560, 2000.
9. W.-Y. Loh. Regression trees with unbiased variable selection and interaction detection. Statistica Sinica, 12:361-386, 2002
10. Loh, W.-Y., and Shih, Y.-S. (1997). Split Selection Methods for Classification Trees. Statistica Sinica, Vol.7, 1997, pp. 815-840
11. D. Malerba, F. Esposito, M. Ceci, and A. Appice. Top-down induction of model trees with regression and splitting nodes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(5):612-625, 2004.
12. J. R. Quinlan. Learning with Continuous Classes. In 5th Australian Joint Conference on Artificial Intelligence, pages 343–348, 1992.
13. http://www.r-project.org/ The R Project for Statistical Computing official website as of 8/8/2012
14. StatLib repository (Dataset Archive) at http://lib.stat.cmu.edu/datasets/ as of 12/04/2013
15. L. Torgo. Functional models for regression tree leaves. In Proc. 14th International Conference on Machine Learning, pages 385–393. Morgan Kaufmann, 1997.
16. UCI repository at http://archive.ics.uci.edu/ml/datasets.html as of 12/04/2012.
17. David S. Vogel, Ognian Asparouhov, Tobias Scheffer. Scalable Look-Ahead Linear Regression Trees KDD'07, August 12–15, 2007, San Jose, California, USA.
18. http://www.cs.waikato.ac.nz/ml/weka/ Weka software official website as of 8/8/2012
19. http://www2.cs.uh.edu/~UH-DMML/index.html